Blank Page — Project Documentation

1. Introduction

Blank Page is a modern blogging platform designed to empower users to create, share, and discover content. It allows anyone to read published posts, subscribe to newsletters, or reach out to site administrators. Registered users can write posts, bookmark favorites, and interact through likes and dislikes, while administrators ensure the quality and security of all published content.

2. Background of the Project

Problem Statement: With the explosion of online content, independent writers often lack an accessible, well-structured platform to publish articles and build an audience. Many blogging systems are either too restrictive or lack moderation, resulting in poor content quality or spam.

Motivation: Blank Page aims to strike a balance between freedom of expression and content quality through user-driven publishing with admin moderation. It encourages community engagement and enables aspiring writers to manage their own content easily.

3. Objectives

- Enable public reading and sharing of high-quality articles.
- Allow users to register, write, edit, and manage their posts.
- Provide features for liking, disliking, and bookmarking posts.
- Ensure that only admin-approved posts are published publicly.
- Allow users to update their profiles and manage subscriptions.
- Provide an efficient admin interface for moderation, user management, and content control.

4. Scope

In Scope:

- Public browsing and reading of published blogs.

- User registration, login, profile management.
- Creating, editing, saving drafts, and submitting posts for review.
- Interaction features: likes, dislikes, bookmarks.
- Contact form and newsletter subscription.
- Admin approval/rejection of posts, user banning, content moderation.
- Replying to user contact submissions.

Out of Scope:

- Built-in advertising or monetization tools.
- Third-party integration with external CMS platforms.
- AI-generated content moderation.

5. Literature Review / Related Work

Similar Platforms:

- **Medium:** Allows anyone to write and publish articles with an emphasis on curation.
- **WordPress.com:** A popular blogging tool for personal or business blogs.
- **Substack:** Combines blogging and newsletter delivery.

Relevant Models:

- **Role-Based Access Control (RBAC):** Users, authors, and admins have different permissions.
- **Content Moderation Workflow:** Draft → Review → Publish.
- **MVC Architecture:** Applied in Django backend.

6. Methodology

Technologies & Tools:

- **Frontend:** React.js, Tailwind CSS, Vite
- **Backend:** Django REST Framework
- **Database:** SQLite (dev) → PostgreSQL/MySQL (production-ready)
- **APIs:** RESTful APIs for auth, articles, categories, tags, comments, subscribers.
- **Version Control:** Git, GitHub
- **Design Tools:** Figma or Miro for user flows, Lucidchart for UML

Development Phases:

1. **Planning:** Define features, workflows, and user stories.
2. **Design:** Wireframes for article pages, profile pages, admin dashboard.
3. **Development:** Build React frontend components, Django REST backend.
4. **Integration:** Secure auth, CRUD APIs, front-back connection.
5. **Testing:** Functional testing for all CRUD operations and user flows.
6. **Deployment:** Deploy on cloud server (e.g., Heroku, Vercel, or DigitalOcean).

Design Models:

- **Use Case Diagram:** Shows interactions for Readers, Authors, and Admins.
- **Class Diagram:** Django models for User, Article, Comment, Contact, Subscriber.
- **Flowchart:** Blog publishing approval flow.

7. Implementation / Development

How It Was Built:

- **Frontend:** Reusable React components for blog cards, article pages, profile settings, admin panels.
- **Backend:** Django REST Framework for robust APIs with token-based authentication.
- **CRUD:** Articles, Comments, Categories, Tags, Contacts, Users, Subscriptions.
- **Security:** Django middleware for permissions and safe data handling.

Key File Highlights:

- `src/components/articles/ArticleActions.tsx` — Controls like/bookmark logic.
- `src/components/admin/AdminLayout.tsx` — Admin dashboard wrapper.
- `admin_api/views.py` — Endpoints for approving posts, banning users.
- `articles/models.py` — Core `Article` model with draft vs. published states.
- `contact/views.py` — Contact form handling and admin replies.

System Architecture:

```
Client (React)
    ⬇️⬆️
API Gateway (Django REST Framework)
    ⬇️⬆️
```

```
Database (SQLite)
```

Database Schema (Sample):

- **User:** id, username, email, password, avatar, bio, role
- **Article:** id, author_id, title, content, status, created_at
- **Comment:** id, article_id, author_id, content, created_at
- **Contact:** id, name, email, message, is_replied
- **Subscriber:** id, email, date_subscribed

8. Results / Analysis

- Users can register and fully manage posts, likes, bookmarks.
- Admin can approve/reject posts, manage contacts, moderate users.
- Public visitors can browse, read, and subscribe without an account.
- Validated user flows: writing drafts, publishing, contact, and subscription.

9. Challenges Faced

- **Moderation Workflow:** Building an efficient post approval system with minimal delay.
- **Role-Based Access:** Securing backend routes to prevent unauthorized CRUD.
- **User Engagement:** Designing intuitive interactions for likes/bookmarks.
- **Resolution:** Clear API permission layers, reusable React components, user testing for UX issues.

10. Conclusion

Blank Page successfully provides a full-fledged blogging platform with community-driven content creation and strong admin oversight. The combined power of React and Django delivers a robust, scalable solution for modern publishing needs.

11. Future Scope

- Add analytics dashboard for authors.
- Enable third-party sign-ins (OAuth).
- Implement SEO optimizations and rich media embedding.