# Project 2: Blogging Platform

## GitHub link: Shreejjaa/Blogging-Platform

## Blogging Platform – Week 6 Report

## Project Setup, Authentication & Boilerplate

**Accomplishments**

1. Project Foundation Established
    - Set up a full-stack project with separate client (React.js) and server (Node.js/Express) directories.
    - Initialized frontend using Create React App.
    - Configured backend with Express.js and connected to MongoDB using Mongoose.
    - Integrated Tailwind CSS for modern, responsive UI.

**Code Implementation:**

- client/package.json – Frontend dependencies
- server/package.json – Backend dependencies
- client/tailwind.config.js – Tailwind CSS configuration
- server/config/db.js – MongoDB connection setup
- server/index.js – Main server entry point

2. Authentication System Implemented
    - Built user registration and login endpoints with JWT-based authentication.
    - Set up password hashing using bcryptjs.
    - Developed authentication middleware for route protection.
    - Integrated authentication flows in the frontend.

**Code Implementation:**

- server/routes/auth.js – Authentication endpoints (register, login)
- server/middleware/authMiddleware.js – JWT verification middleware
- server/models/User.js – User data model
- client/src/pages/Login.js – Login page
- client/src/pages/Register.js – Registration page
- client/src/context/AuthContext.js – Auth state management

3. Boilerplate & Routing
    - Established React Router for navigation.
    - Created protected route components for authenticated access.
    - Set up basic navigation and layout structure.

**Code Implementation:**

- client/src/App.js – Main application routing
- client/src/components/AdminRoute.js – Protected admin route
- client/src/components/Navbar.js – Navigation bar

**Learning Experience**

- React.js: Improved understanding of component structure, hooks, and routing.
- JWT Authentication: Learned secure token-based authentication and middleware usage.
- MongoDB/Mongoose: Gained experience in schema design and database integration.
- Tailwind CSS: Enhanced skills in utility-first CSS for rapid UI development.
- API Development: Practiced RESTful API design and Express.js routing.

**Goals for Next Week**

1. Design and implement the blog schema in the backend.
2. Develop blog post APIs (CRUD operations).
3. Integrate a rich text editor for blog creation in the frontend.
4. Set up initial UI for creating and listing blogs.

**Feedback and Support Needed**

- **Technical Guidance:** Best practices for structuring authentication and user management.
- **Security Review:** Request review of authentication and JWT implementation.

- **UI/UX Feedback:** Suggestions for improving navigation and authentication flows.
- **Performance:** Advice on optimizing backend API response times.

# Blogging Platform – Week 7 Report

# Blog Schema, Post APIs, Integrate Rich Text Editor

**Accomplishments**

1. Blog Schema Designed & Implemented
    - Created a comprehensive blog schema with fields for title, content, cover image, author, status (draft/published), tags, likes, and comments.
    - Enabled text search and category tagging for blogs.

**Code Implementation:**

- server/models/Blog.js – Blog schema definition

2. Blog Post APIs Developed

- Built RESTful APIs for creating, reading, updating, and deleting blog posts.
- Added endpoints for liking/unliking posts and adding/removing comments.
- Supported image uploads for blog cover images.
- Enabled search and tag-based filtering for blog listing.

**Code Implementation:**

- server/routes/blogs.js – Blog post API endpoints (CRUD, like, comment, upload, filter)
- server/routes/upload.js – Image upload endpoint

3. Integrated Rich Text Editor for Blog Creation
    - Integrated ReactQuill as a rich text editor for blog content.
    - Enabled image upload and category/tag selection in the blog creation form.

**Code Implementation:**

- client/src/pages/CreatePost.js – Blog creation UI with ReactQuill editor and category/tag selection

**Learning Experience**

- Mongoose Schema Design: Learned to design complex schemas with references, arrays, and text indexes.
- RESTful API Development: Gained experience in building and securing CRUD APIs, handling file uploads, and implementing search/filter logic.
- Rich Text Editor Integration: Understood how to integrate and configure ReactQuill for a seamless writing experience.
- Frontend-Backend Integration: Improved skills in connecting React forms to backend APIs and handling authentication for protected routes.

**Goals for Next Week**

1. Build UI for creating and listing blogs.
2. Add functionality for likes and comments in the frontend.
3. Enhance blog detail and list views for better user experience.

**Feedback and Support Needed**

- API Design Review: Suggestions for improving API structure and error handling.
- Rich Text Editor: Advice on optimizing the editor for large content and media embedding.
- UI/UX Feedback: Input on the usability of the blog creation form and tag/category selection.

# Blogging Platform – Week 8 Report

# UI for Creating and Listing Blogs, Add Likes/Comments

**Accomplishments**

1. UI for Creating Blogs
    - Developed a user-friendly interface for creating new blog posts, including title, content (rich text), image upload, and category/tag selection.
    - Enabled saving posts as drafts or publishing them directly.

**Code Implementation:**

- client/src/pages/CreatePost.js – Blog creation form with ReactQuill, image upload, and category/tag selection

2. UI for Listing Blogs

- Built a responsive blog listing page displaying all published blogs with search and category filter options.
- Implemented search by title/content and filtering by category/tag.

**Code Implementation:**

- client/src/pages/BlogList.js – Blog listing with search and filter
- client/src/pages/BlogList.css – Styling for blog list

3. Likes and Comments Functionality
    - Added the ability for users to like/unlike blog posts.
    - Implemented a comment system allowing users to add and view comments on blog posts.
    - Displayed like counts and comment lists on the blog detail page.

**Code Implementation:**

- client/src/pages/BlogDetail.js – Blog detail view with like and comment features
- server/routes/blogs.js – Backend endpoints for likes and comments

4. Edit and Manage Blogs

- Enabled users to edit their own blog posts.
- Provided options to update content, tags, and status (draft/published).

**Code Implementation:**

- client/src/pages/EditBlog.js – Blog editing interface

**Learning Experience**

- React State Management: Improved handling of form state, API calls, and UI feedback.
- User Interactivity: Learned to implement interactive features (likes, comments) and real-time UI updates.
- Search & Filtering: Gained experience in building efficient search and filter UIs connected to backend queries.
- Component Reusability: Enhanced skills in creating reusable components for forms, spinners, and navigation.

**Goals for Next Week**

1. Implement user profile management (view and edit profile).
2. Add advanced search, filters, and category tags for blogs.
3. Polish UI/UX for blog interactions and navigation.

Feedback and Support Needed

- UI/UX Review: Suggestions for improving the blog creation and listing experience.
- Performance: Advice on optimizing blog list rendering and API response times.
- Feature Suggestions: Input on additional features for blog engagement (e.g., comment moderation, notifications).

# Blogging Platform – Week 9 Report

# Profile Management, Search, Filters, and Category Tags

**Accomplishments**

1. Profile Management
    - Implemented user profile page displaying user information and their blog posts.
    - Enabled users to edit their profile details (e.g., username, bio).
    - Allowed users to view, edit, and manage their own blogs from the profile page.

**Code Implementation:**

- client/src/pages/Profile.js – User profile page with blog management
- client/src/pages/EditProfile.js – Profile editing interface
- server/routes/auth.js – Backend endpoint for updating user profile

2. Search and Filters
    - Enhanced blog listing with search functionality (by title/content).
    - Added category/tag-based filtering for blogs.
    - Enabled combined search and filter queries for more precise results.

**Code Implementation:**

- client/src/pages/BlogList.js – Search and filter UI for blogs
- server/routes/blogs.js – Backend support for search and tag filtering

3. Category Tags
    - Integrated category/tag selection in blog creation and editing.
    - Displayed tags on blog detail and list views.
    - Enabled filtering blogs by selected tags.

**Code Implementation:**

- client/src/pages/CreatePost.js – Tag selection during blog creation
- client/src/pages/EditBlog.js – Tag editing for existing blogs
- client/src/pages/BlogDetail.js – Tag display and navigation

**Learning Experience**

- Profile Management: Learned to securely update user data and display user-specific content.
- Advanced Filtering: Improved skills in combining search and filter logic in both frontend and backend.
- Tagging Systems: Understood best practices for implementing and displaying category/tag systems in blogs.
- User Experience: Gained experience in building intuitive navigation and management features for users.

**Goals for Next Week**

1. Conduct final testing and bug fixes across the platform.
2. Build and refine the admin dashboard for user and blog management.
3. Prepare and execute deployment of the application.

**Feedback and Support Needed**

- Code Review: Request review of profile management and search/filter logic.
- UI/UX Feedback: Suggestions for improving the profile and blog management experience.
- Deployment Guidance: Best practices for deploying a full-stack MERN application.

# Blogging Platform – Week 10 Report

# Final Testing, Admin Dashboard, Deployment

**Accomplishments**

1. Final Testing and Bug Fixes
    - Conducted comprehensive testing of authentication, blog CRUD, likes, comments, and profile management.
    - Fixed bugs and improved error handling across both frontend and backend.
    - Ensured all user flows (registration, login, posting, editing, commenting) work as intended.

**Code Implementation:**

- server/test-auth.js – Authentication testing
- server/test-complete-flow.js – End-to-end flow testing

2. Admin Dashboard
    - Developed an admin dashboard for managing users and blog posts.
    - Enabled admin to view statistics, recent activity, and perform user/blog deletions.
    - Provided a secure interface for admin-only actions.

**Code Implementation:**

- client/src/pages/AdminDashboard.js – Admin dashboard UI
- server/routes/admin.js – Backend admin routes for user and blog management
- server/routes/analytics.js – Backend analytics/statistics endpoints

3. Deployment
    - Prepared the application for deployment (build scripts, environment configuration).
    - Deployed both frontend and backend to production environment.
    - Verified deployment and performed post-deployment checks.

**Code Implementation:**

- server/check-env.js – Environment variable checks for deployment
- (Deployment steps are not shown in codebase but assumed completed as per roadmap)

**Learning Experience**

- Testing: Gained experience in writing and running backend and integration tests.
- Admin Tools: Learned to build secure admin interfaces and manage application data.
- Deployment: Understood deployment workflows, environment configuration, and post-deployment validation for MERN stack apps.
- Production Readiness: Improved skills in debugging, error handling, and optimizing for production.

**Goals for Next Week**

- Project completion: Gather user feedback and monitor application performance.
- Plan for future improvements and feature enhancements based on user input.

**Feedback and Support Needed**

- Security Review: Request a final security audit for admin and deployment configurations.
- Performance Monitoring: Guidance on tools and best practices for monitoring the live application.
- User Feedback: Suggestions for collecting and acting on user feedback post-launch.

**Conclusion:**

Over the course of five weeks, significant progress was made in designing, developing, and deploying a full-featured blogging platform. The journey from initial setup to production deployment provided valuable technical and practical experience across the full stack.Key Achievements:

- Solid Foundation: The project began with a robust setup of both backend (Node.js/Express, MongoDB) and frontend (React, Tailwind CSS), establishing a scalable and maintainable architecture.
- Authentication & Security: Secure user authentication was implemented using JWT, with role-based access control and middleware to protect sensitive routes.
- Rich Blogging Features: The platform supports blog creation, editing, listing, and detailed views, with a rich text editor for content, image uploads, and category/tag management.
- User Engagement: Interactive features such as likes and comments were added, enhancing user engagement and community interaction.
- Advanced Search & Filtering: Users can efficiently search and filter blogs by keywords and categories, improving content discoverability.
- Profile & Admin Management: Comprehensive profile management allows users to control their content, while an admin dashboard provides oversight and moderation capabilities.
- Testing & Deployment: Rigorous testing ensured reliability, and the final deployment brought the platform live for real users.

**Learning Outcomes:**

- Deepened understanding of full-stack development, including RESTful API design, secure authentication, state management in React, and integration of third-party libraries (e.g., ReactQuill).
- Gained practical experience in debugging, error handling, and optimizing both backend and frontend for performance and usability.
- Learned best practices for deploying and maintaining a MERN stack application in a production environment.

Next Steps:

- Gather user feedback to identify areas for improvement and potential new features.
- Monitor application performance and security in the live environment.
- Plan for future enhancements, such as notifications, analytics, or mobile responsiveness.

Final Thoughts: This project not only delivered a functional and modern blogging platform but also provided a comprehensive learning experience in full-stack web development. The iterative weekly approach ensured steady progress, continuous learning, and a strong foundation for future growth and innovation.