Assignment: Blog Admin Dashboard with Next.js, Tailwind CSS, Node.js, and SQL

Objective

Build a responsive blog management system where an admin can write, edit, and publish blog posts. Each post can include images, videos, meta tags, and SEO fields. The published blogs should be viewable on a frontend page, styled with Tailwind CSS.

Requirements

Tech Stack

- **Frontend**: Next.js, Tailwind CSS
- **Backend**: Node.js, Express.js
- **Database**: SQL (MySQL, PostgreSQL, or SQLite)

Features

1. Admin Dashboard

The admin dashboard should have the following functionality:

- **Create New Blog Post**:
    - Add a title, content (rich text editor is optional).
    - Upload images and add video links.
    - Add SEO fields: Meta title, meta description, and tags.
    - Set the blog post as published or draft.
- **Edit Blog Post**:
    - Admin can edit all the fields in an existing post.
    - Update images and video links.
    - Update SEO fields.
- **Delete Blog Post**:
    - Ability to delete any blog post from the admin dashboard.
- **Post List**:
    - Display a list of all posts with titles, publish status, and creation date.
    - Filter posts by status (published/draft).

2. Blog Display Page

Create a blog page in Next.js to display individual blog posts:

- Fetch and display the blog content, images, and videos.

- Display SEO metadata (meta title and description) in the `<head>` section for search engine optimization.
- Show related posts or tags (optional).

3. Responsiveness

- The dashboard and blog page should be fully responsive and optimized for mobile, tablet, and desktop screens.

4. Database Schema

- Create a table for storing blog data with fields like `id`, `title`, `content`, `image_url`, `video_url`, `meta_title`, `meta_description`, `tags`, and `status` (published/draft).
- Store metadata and SEO fields in a way that can be easily retrieved for each post.

Development Guidelines

Backend (Node.js & SQL)

1. Set up an Express server to handle blog post CRUD operations.
2. Connect the server to an SQL database for storing blog data.
3. Create API routes for:
    - Creating, updating, and deleting blog posts.
    - Fetching individual blog post data for the display page.
4. Use a SQL database of your choice to create and manage the `blogs` table.

Frontend (Next.js & Tailwind CSS)

1. **Admin Dashboard**:
    - Build a Next.js page for the admin dashboard.
    - Use Tailwind CSS to style the page and ensure it's responsive.
2. **Blog Display Page**:
    - Create a dynamic Next.js page to render individual blog posts.
    - Apply SEO best practices by adding dynamic meta tags using Next.js `<Head>` component.
    - Display images, videos, and blog content.

Optional Enhancements

- Add rich text editing capabilities in the admin dashboard.
- Implement user authentication for admin access to the dashboard.

- Add image upload functionality using a third-party service (e.g., Cloudinary).
- Implement pagination for the list of blog posts.

Submission Requirements

Submit your project with the following:

- A README.md file with setup instructions.
- Code files for both frontend and backend.
- SQL script to initialize the database tables.
- Optional: Screenshots or a video walkthrough of the application.

Deploy this to Vercel or other platform and share link