**Technology Stack:**

* Frontend: React.js
* Backend: Express.js, Node.js
* Database: MongoDB

**Features:**

* Like Functionality (Backend Handled)
* Pagination for Blog Posts
* Content Management System

**Optional Features:**

* Newsletter Subscription (Frontend only)

**Steps:**

1. **Project Setup:**
   * Create two directories: client (for React frontend) and server (for Node.js backend)
   * Initialize both directories with npm init -y
   * Install dependencies:
     + Client (client directory): npm install react react-dom axios
     + Server (server directory): npm install express mongoose cors body-parser (more dependencies might be needed later)
2. **Backend (server directory):**
   * Create a file named index.js
   * Configure Express server
   * Connect to MongoDB using Mongoose
   * Define a Mongoose schema for blog posts (title, content, likes, etc.)
   * Create CRUD (Create, Read, Update, Delete) operations for blog posts using Express routes and Mongoose methods.
   * Implement like functionality on a specific blog post using a PUT/PATCH route that updates the likes property in the database.
   * Implement pagination using query parameters like page and limit to retrieve a specific number of blog posts on each page.
3. **Frontend (client directory):**
   * Create React components for:
     + Blog list
     + Individual blog post
     + Like button
   * Use React Router for navigation between blog list and individual posts.
   * Fetch blog post data from the backend API using axios calls.
   * Implement like functionality using a button that sends a PUT/PATCH request to the like endpoint in the backend. Update the UI to reflect the like count.
   * Display blog posts with pagination using libraries like react-paginate or custom logic based on received data from the backend.
4. **Content Management:**
   * Consider using a headless CMS like Contentful or Strapi to manage blog content outside the codebase. This allows for easy editing and updates without code changes.
   * Alternatively, implement a basic content management system in the backend using a separate route to add, edit, and delete blog posts directly through the application.
5. **Images:**
   * Use placeholder images for now. You can find free dummy images on websites like [PlaceIMG](https://placeimg.com/) or [Unsplash](https://unsplash.com/).
6. **Optional: Newsletter Subscription (client directory):**
   * Create a form component for email signup.
   * Implement basic validation for email format. (Functionality to store emails is not required)
7. **Deployment (Optional):**
   * After completing development, you can deploy the application to a platform like Netlify or Render. These platforms offer easy deployment options for React applications.