**Frontend Technologies & Features**

1. **Modern React with Best Practices**

* Uses functional components with hooks (useState, useEffect, useSelector)
* Implements React Router v6 for routing with protected routes
* Redux Toolkit for state management with proper slicing patterns
* Persistent state management using redux-persist

1. **UI/UX Features**

* Responsive design using Tailwind CSS
* Dark/Light theme support with theme persistence
* Modern UI components from Flowbite React
* Rich text editing using React Quill
* Loading states with spinners
* Circular progress indicators for uploads

1. **Authentication & Security**

* JWT-based authentication
* Google OAuth integration using Firebase
* Protected routes for admin and authenticated users
* Secure cookie handling

1. **Modern Frontend Development**

* Vite.js as build tool
* Environment variables management
* Proper file structuring and component organization
* Code splitting and lazy loading
* Custom hooks and reusable components

**Backend Technologies & Features**

1. **Node.js/Express Architecture**

* RESTful API design
* Middleware pattern for authentication and error handling
* Route separation and controller logic organization
* Environment configuration using dotenv

1. **Database & Models**

* MongoDB with Mongoose ODM
* Well-structured schemas for User, Post, and Comment models
* Proper data relationships and referencing
* Index optimization

1. **Security Features**

* Password hashing using bcryptjs
* JWT token generation and verification
* CORS configuration
* Cookie-based authentication
* Input validation and sanitization

1. **API Features**

* CRUD operations for posts, comments, and users
* Search functionality with regex support
* Pagination support
* File upload handling with Firebase Storage
* Error handling middleware

**Best Practices Implemented**

1. **Code Organization**

backend/

  ├── config/      # Database and app configuration

  ├── controllers/ # Business logic

  ├── models/      # Database schemas

  ├── routes/      # API routes

  └── utils/       # Helper functions

frontend/

  ├── components/  # Reusable UI components

  ├── pages/       # Route components

  ├── redux/       # State management

  └── utils/       # Helper functions

1. **Security Best Practices**

* Secure password storage
* Protected API endpoints
* Input validation
* HTTP-only cookies
* Environment variable usage

1. **Performance Optimization**

* Proper indexing in MongoDB
* Image optimization and lazy loading
* Pagination for large datasets
* Caching strategies with Redux

1. **State Management**

* Centralized Redux store
* Proper action creators and reducers
* Persistent state handling
* Loading and error states

1. **Modern Development Features**

* ESModule syntax
* Async/await for promises
* Error boundaries
* Proper error handling
* TypeScript-ready structure