IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154280

A MERN-Stack Blog Application with Gemini-3-Flash Integration

Aman Narayan[1], Abhinav Nigam[2], Abhishek Jaiswal[3], Suraj Kushwaha[4]

UG Student, Department of Computer Science and Engineering,

Goel Institute of Technology and Management, Lucknow, Uttar Pradesh, India[1-4]

Abstract: The exponential growth of digital content has necessitated the evolution of traditional Content Management Systems (CMS). Traditional platforms lack integrated intelligent tools, forcing authors to spend significant time on content formulation and moderation. This project presents the development of an intelligent, automated, and highly interactive Blog Application built on the MERN stack (MongoDB, Express.js, React.js, Node.js). The system's core innovation is the integration of the Google Gemini-3-Flash-Preview Generative AI model, which automatically synthesizes high-quality, formatted blog descriptions based on minimal input (Title and Subtitle).Furthermore, the application incorporates a **"1-to-N"** user architecture featuring a centralized Admin Dashboard. The dashboard provides real-time statistics, comprehensive blog management, and a strict comment moderation workflow to prevent spam. On the client side, the application offers dynamic category filtering, seamless navigation, and social media integration without page reloads. The study demonstrates that integrating Gemini into web frameworks significantly reduces administrative workload while enhancing content quality and platform security.

Keywords: Generative AI, Gemini-3-Flash, MERN Stack, Content Management System (CMS), Comment Moderation, Single Page Application (SPA), Web Security.

I. INTRODUCTION

Blogging has evolved from a medium of personal expression into a cornerstone of digital marketing, education, and journalism. With millions of articles published daily, content creators and administrators face the continuous pressure of drafting engaging content while managing the technical aspects of their platforms. The primary bottleneck in this ecosystem is the sheer time and cognitive effort required to move from an idea to a fully published, well-formatted article. Historically, authors have relied on standalone word processors or external Artificial Intelligence (AI) chatbots. This fragmented workflow—where a user generates content in one application, copies it, and formats it in a separate CMS—is inefficient. Moreover, maintaining the quality of user interaction through comment sections is challenging due to the high volume of automated spam and irrelevant discussions.

This project introduces a centralized, AI-assisted platform that bridges the gap between content creation and management. By utilizing the MERN stack, the system provides a robust, scalable Single Page Application (SPA) experience. The integration of the gemini-3-flash-preview API directly into the application's writing interface empowers administrators to auto-generate comprehensive blog posts instantly. The system also includes advanced categorization, secure session management via HTTP-only cookies, and an admin-controlled comment approval pipeline, establishing a complete ecosystem for modern digital publishing.

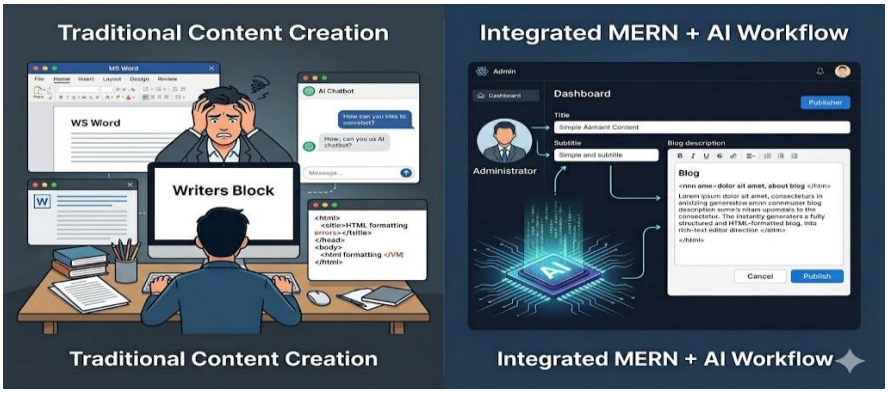

Figure 1: Introduction of our project

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154280

II. PROBLEM IDENTIFICATION AND SIGNIFICANCE

Despite the availability of numerous blogging platforms like WordPress or Medium, independent developers and content administrators face specific, unresolved challenges:

1. **Time-Intensive Content Creation:** Formulating detailed, grammatically correct, and engaging blog descriptions requires substantial human effort, leading to writer's block and inconsistent publishing schedules.
2. **Fragmented Workflows:** The separation of AI generation tools (like ChatGPT) and the actual publishing platform disrupts the author's workflow.
3. **Unregulated User Interactions:** Open comment sections are highly susceptible to spam and abusive language. Without a proper moderation workflow, the platform's credibility deteriorates.
4. **Inefficient Content Discovery:** Users often struggle to find domain-specific content (e.g., Finance, Technology) if the platform lacks real-time, seamless filtering mechanisms.
5. **Legacy Architectures:** Traditional multi-page applications suffer from high latency and constant page reloads, degrading the modern user experience.

III. LITERATURE REVIEW

1. YouTube-Based Learning Resources: The development of this blog application was significantly inspired by practical tutorials available on GreatStack. These tutorials provide structured, project-based learning approaches for full-stack web development, particularly using the MERN stack. Through these resources, key concepts such as React component structuring, REST API integration, Node.js backend handling, and MongoDB database operations were implemented.
Such practical learning aligns with modern web development practices discussed in studies related to MERN stack architecture and asynchronous programming models [2][3]. However, unlike tutorial-based implementations, the proposed system extends functionality by integrating AI-driven content generation, which enhances automation and productivity as suggested in AI-based web engineering research [7].

2. Blogger Platform Analysis: The platform Blogger is one of the earliest blogging systems that provides an easy interface for publishing content. It supports template-based design and basic content management features.
However, as highlighted in CMS-related studies [6], traditional platforms like Blogger lack advanced features such as integrated AI tools and efficient moderation mechanisms. The proposed system improves upon this by incorporating AI-based content generation and structured moderation workflows, addressing key limitations identified in earlier research.

3. Blog Design Inspirations from Industry Examples: Web resources like SiteBuilderReport provide a collection of modern blog UI/UX patterns, including responsive layouts, card-based structures, and content categorization.
These design inspirations align with frontend development practices using React.js [3], where dynamic rendering and component-based architecture enhance user experience. The proposed system adopts these modern UI techniques to ensure better usability and engagement.

4. Wix Blogging Platform Study: The blogging system provided by Wix enables users to create blogs using drag-and-drop tools and built-in SEO features.
However, Wix operates within a closed ecosystem with limited backend customization. According to research on scalable web systems [2][6], such limitations restrict flexibility and performance optimization. The proposed system overcomes these issues by using an open MERN stack architecture, allowing full control over backend logic, database management, and API integrations.

5. AI-Based Content Generation Systems: Recent advancements in Artificial Intelligence, particularly Large Language Models (LLMs), have significantly influenced content generation systems. Research studies highlight that AI integration in web applications reduces manual effort and enhances content quality [7].
The proposed system integrates the Gemini-3-Flash model, aligning with modern AI-based development practices discussed in API documentation and research studies [4]. This integration bridges the gap between content creation and publishing, eliminating fragmented workflows and improving efficiency.

IV. OBJECTIVES & GOALS

To address the aforementioned challenges, this research outlines the following primary objectives:

1. **Seamless AI Integration:** To implement the gemini-3-flash-preview API to automatically draft blog descriptions based on context provided via a Title and Subtitle.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154280

2. **Centralized Administration:** To develop a secure Admin Dashboard featuring a sidebar navigation system that tracks real-time metrics (Total Blogs, Unpublished Drafts, Comment Counts).
3. **Robust Moderation Protocol:** To engineer a strict comment approval workflow where user inputs are queued for admin review before public rendering.
4. **Enhanced User Experience (UX):** To build a dynamic, responsive Landing Page with category filters (All, Technology, Startup, Lifestyle, Finance) that update content instantaneously.
5. **Secure Authentication:** To implement a secure, cookie-based session management system that protects the administrative routes from unauthorized access.

V. PROPOSED SYSTEM ARCHITECTURE (1-TO-N MODEL)

The proposed system is designed around a scalable "1-to-N" architectural model.

- **The "1" Entity:** The Administrator. The admin has exclusive access to the dashboard, content creation tools, and moderation controls.
- **The "N" Entities:** The Users/Readers. An infinite number of readers can interact with the frontend, read blogs, filter categories, and submit comments.

A. Data Flow Overview

1. The Admin authenticates and accesses the dashboard.
2. The Admin provides minimal inputs (Title, Subtitle) to the AI engine.
3. The Node.js backend processes this, communicates with the Gemini API, and returns a formatted description.
4. The Admin publishes the blog to the MongoDB database.
5. Readers fetch these blogs on the React frontend, filter them, and submit comments.
6. Comments return to the Admin Dashboard for final approval.

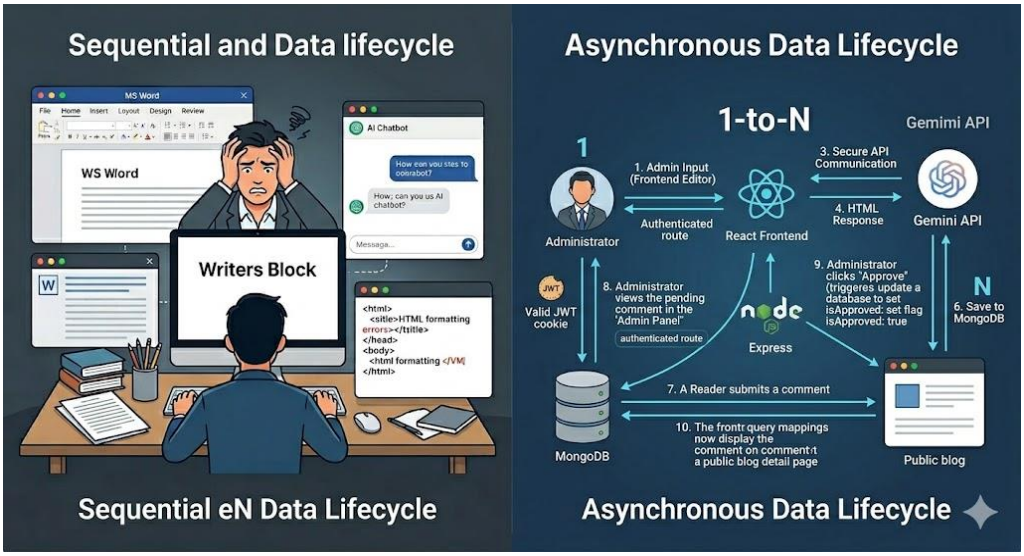

Figure 2: 1-to-N System Architecture and Data Flow Diagram Here

VI. MERN STACK IMPLEMENTATION

The project is engineered using the MERN stack, selected for its asynchronous capabilities and unified JavaScript environment.

1. MongoDB (Database Layer)

MongoDB, a NoSQL database, stores data in flexible, JSON-like BSON documents. It is highly suitable for this application because blog entries and comments have varying lengths and structures. The database manages relational mapping between specific blogs and their child comments using reference ObjectIDs.

2. Express.js (Application Routing)

Express.js serves as the minimalist web framework for Node.js. It handles the API endpoint routing, middleware execution (such as verifying secure cookies), and orchestrates the data exchange between the React frontend and the

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154280

MongoDB database.

3. React.js (Presentation Layer)

React constructs the Single Page Application (SPA). Utilizing the virtual DOM, React ensures that when a user clicks the "Technology" filter, the blog list updates in milliseconds without a full page reload. React Router handles dynamic routing, directing users to specific blog detail pages based on unique URLs.

4. Node.js (Runtime Environment)

Node.js provides the non-blocking, event-driven architecture required to handle multiple concurrent requests. It efficiently processes incoming image uploads, API calls to Google's Gemini, and database queries simultaneously.

VII. SYSTEM INTERFACE & IMPLEMENTATION (WITH FIGURES)

A. Landing Page (User Interface)

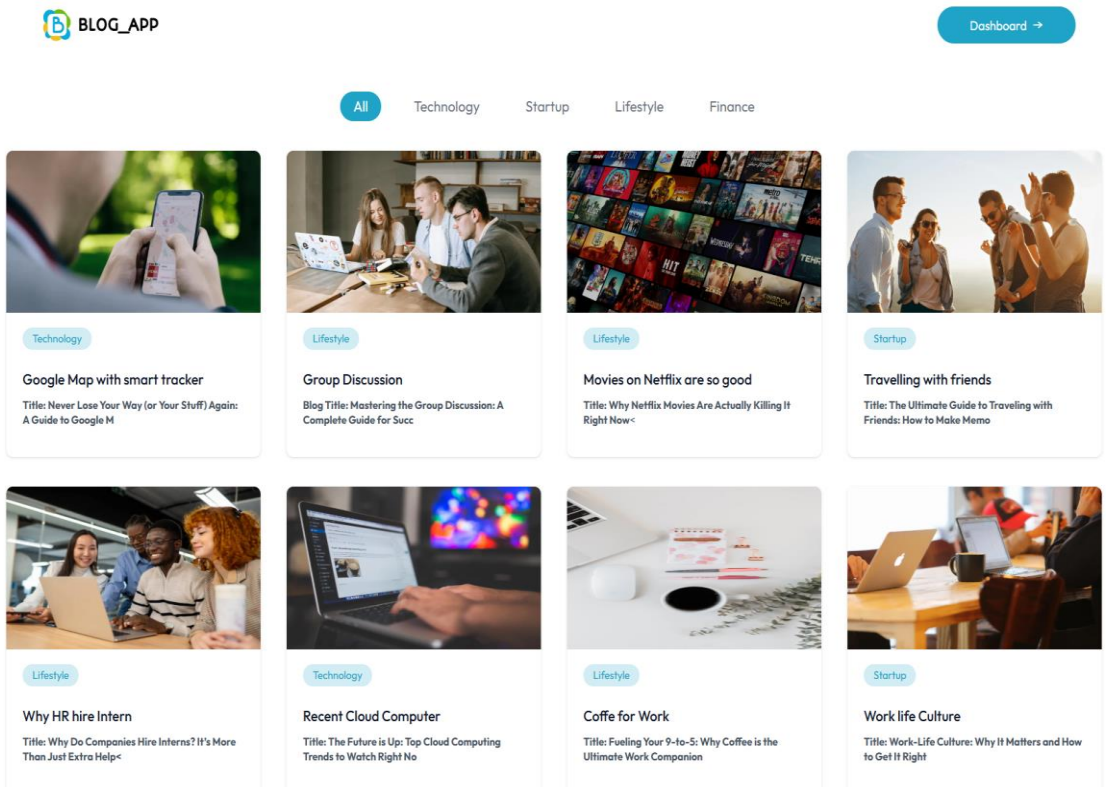

Figure 3: Landing Page of Blog Application

The landing page serves as the primary interface for users to explore available blog content. It is designed using a responsive grid layout that displays blog cards categorized into sections such as Technology, Startup, Lifestyle, and Finance.

Users can dynamically filter blogs using category buttons, enabling seamless content discovery without page reloads. This functionality is achieved using React.js state management and component-based rendering, ensuring a smooth and interactive user experience.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154280

B. Blog Detail Page (Content View)

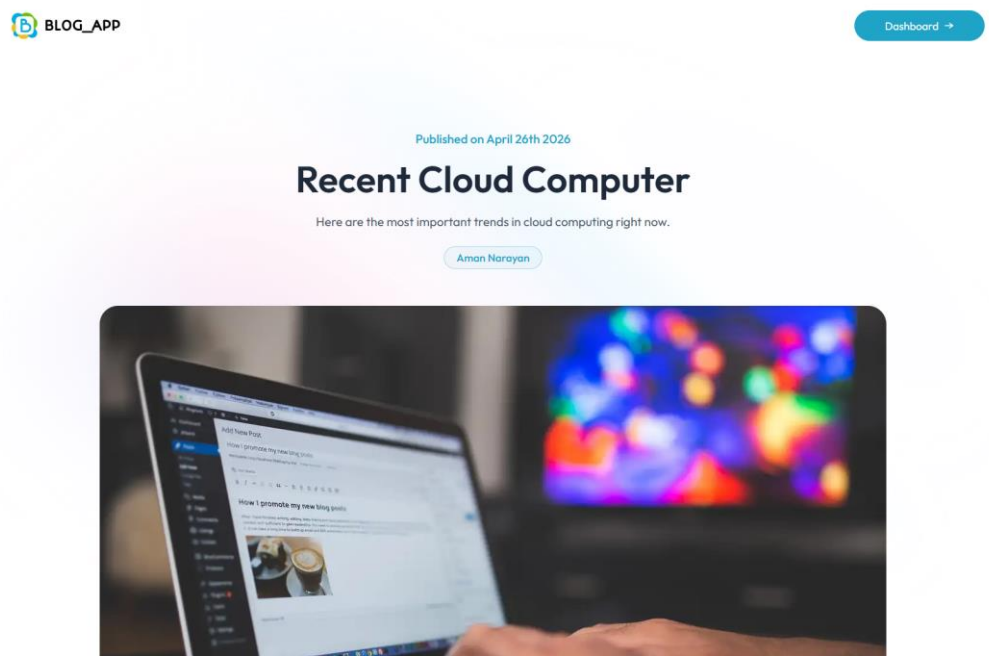

Figure 4.1: Blog Overview

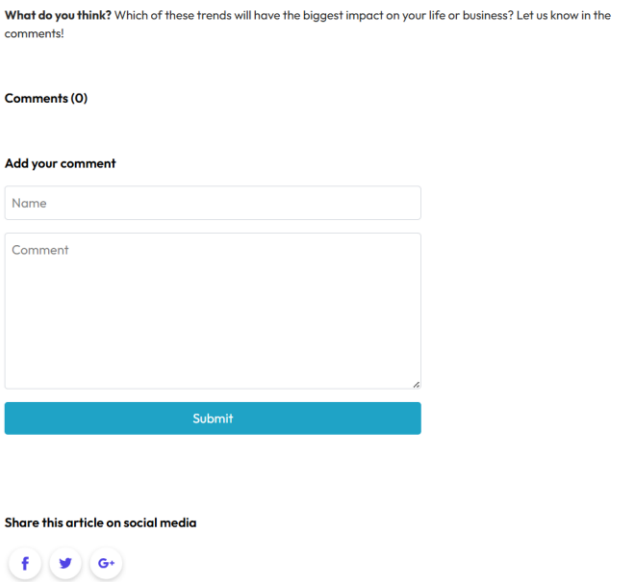

Figure 4.2: Comment Section Overview

[Figure 4: Blog Detail Page]

The blog detail page provides a comprehensive view of the selected blog post. It includes essential elements such as the title, subtitle, publication date, author name, and featured image.

The content displayed on this page is generated using the Gemini-3-Flash AI model and rendered dynamically using HTML. The page also supports user interaction through comment submission, enhancing engagement and community participation.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154280

C. Admin Dashboard (Control Panel)

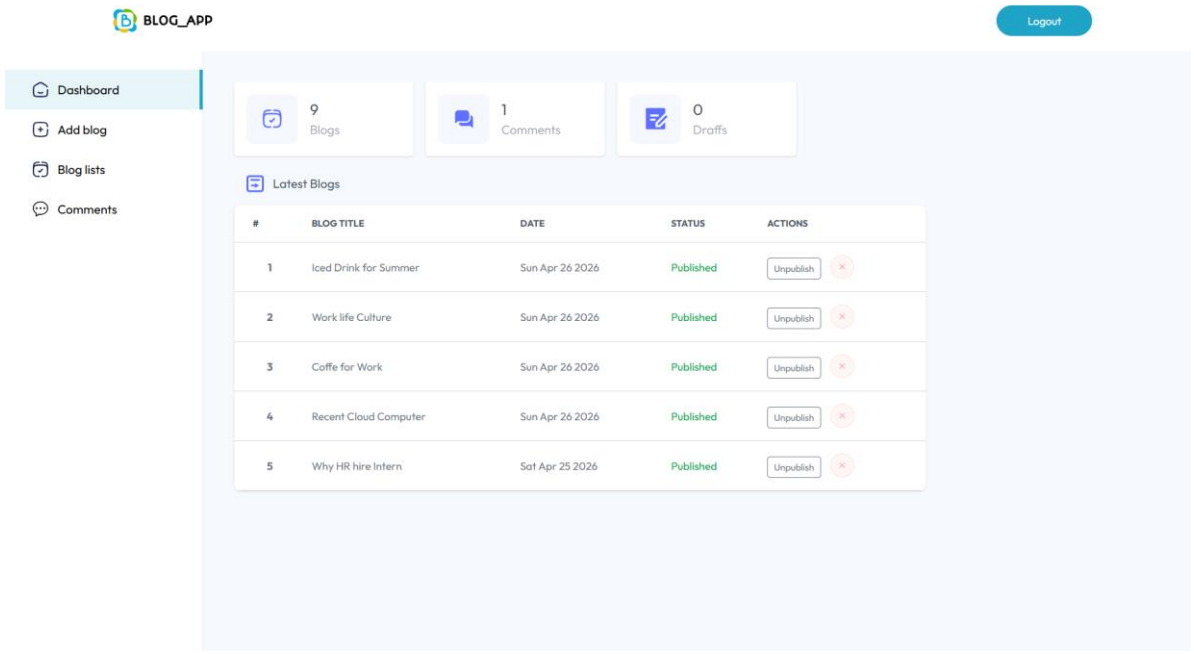

Figure 5: Admin Dashboard

The admin dashboard acts as the central control system of the application. It provides real-time statistics such as total blogs, comments, and drafts.

Key features include:

- Blog creation and editing
- Blog publishing/unpublishing
- Comment moderation system
- Blog listing with actions

This dashboard ensures that administrators can efficiently manage content and maintain platform quality through a structured workflow.

VIII. SYSTEM MODULES & DESIGN

A. Landing Page & Dynamic Discovery

The landing page acts as the entry point for the "N" users.

- **Header & Identity:** Features the official application Logo. Clicking the logo globally redirects users back to the root landing page.
- **Dynamic Category Filtering:** A navigation component offers filters: *ALL, Technology, Startup, Lifestyle, Finance*. These filters interact with React state variables to instantly render relevant content.
- **Blog Cards:** Approved and published blogs are displayed in a responsive grid, showcasing the thumbnail, title, date, and current status.

B. Blog Detail & Interaction Page

When a user selects a blog, they are navigated to a detailed view (/blog/:id).

- **Content Presentation:** Displays the 'Published On' date, main Title, Subtitle, and Author details (Name and Profile Image). Below this, the AI-generated HTML description is beautifully rendered.
- **Social Sharing:** Integrated functionality allows readers to share the specific article URL directly to Facebook, Twitter, and Google+.
- **Comment Submission:** At the footer, users can input their name and thoughts. Upon submission, the system alerts the user that the comment is "Awaiting Moderation."

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154280

C. The Comment Moderation Workflow

- **Database Flagging:** Submitted comments are stored in MongoDB with a default boolean flag: isApproved: false.
- **Admin Review:** The Admin views pending comments in the dashboard.
- **Public Visibility:** Only after the Admin explicitly toggles the flag to true does the comment iterate and render on the public blog detail page.

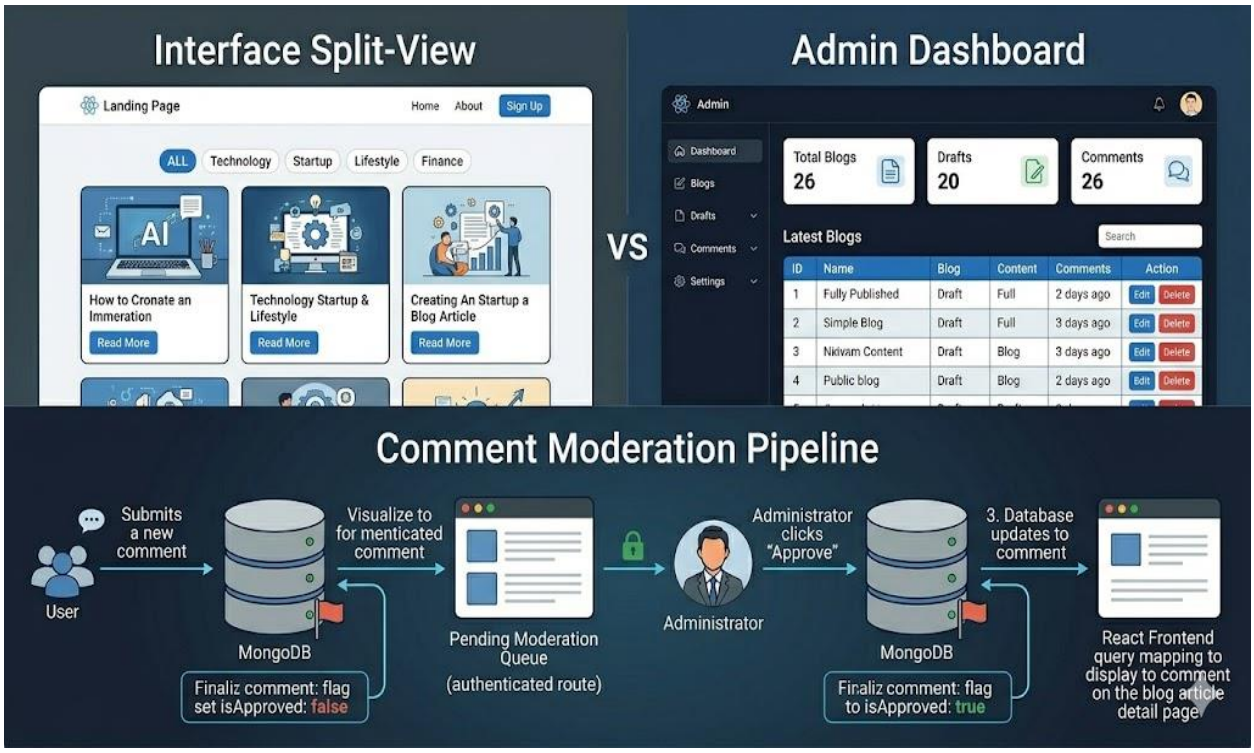

Figure 2: Comment Approval Workflow Diagram Here

D. Admin Panel & Dashboard

Protected by middleware guards, the Admin Panel is the control center.

- **Sidebar Navigation:** Persistent sidebar containing links to the Dashboard, Add Blog, and Blog List.
- **Live Statistics Dashboard:** Displays critical real-time aggregate data:
  - Total Blogs Count
  - Unpublished Drafts Count
  - Total Comments Count
- **Latest Blogs Table:** A data table summarizing recent posts with columns for Title, Date, Status, and Action (Delete/Edit).

E. AI-Powered "Add Blog" Module

The focal point of administrative innovation.

- **Input Fields:** Upload Thumbnail Image, Blog Title, Subtitle, Category Dropdown, Author details.
- **AI Generator Engine:** Upon clicking the "Auto-Generate" button, the system triggers the LLM (detailed in Section VIII).
- **Publishing Controls:** The Admin can review the generated rich text, then choose to "Publish Now" or save it as a Draft.

F. Authentication & Logout Sequence

- **Login:** The system verifies credentials and issues a JSON Web Token (JWT).
- **Session Storage:** The JWT is stored in an HttpOnly cookie, rendering it immune to client-side script access.
- **Logout:** The logout action commands the backend to clear the specific cookies, destroy the session, and dynamically redirect the user back to the Landing Page.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154280

IX. AI INTEGRATION: GEMINI-3-FLASH-PREVIEW

The most significant technological advancement in this project is the integration of Generative AI for content creation.

1. Model Selection

Google's gemini-3-flash-preview was selected due to its exceptionally low latency and high context-window capabilities. In a web application, administrators expect near-instantaneous feedback; the "Flash" variant delivers comprehensive text generation in mere seconds.

2. Prompt Engineering & Execution

The system acts as an intermediate prompt engineer. When the Admin provides a Title (e.g., "The Future of AI") and Subtitle (e.g., "Impact on Jobs"), the Node.js backend constructs a rigid, hidden prompt:

"Act as an expert technical blog writer. Generate a highly engaging, professional, and detailed blog description. The main title is '{Title}' and the context is '{Subtitle}'. Structure the response using clean HTML tags (paragraphs, bold text, lists) suitable for direct rendering on a web page."

3. Data Flow

1. Backend sends the prompt via REST API to Google's servers.
2. The Gemini model synthesizes the response.
3. The Node.js server receives the payload and transmits it to the React frontend.
4. React injects the payload into the rich-text edit or, allowing the Admin final editorial control before publishing.

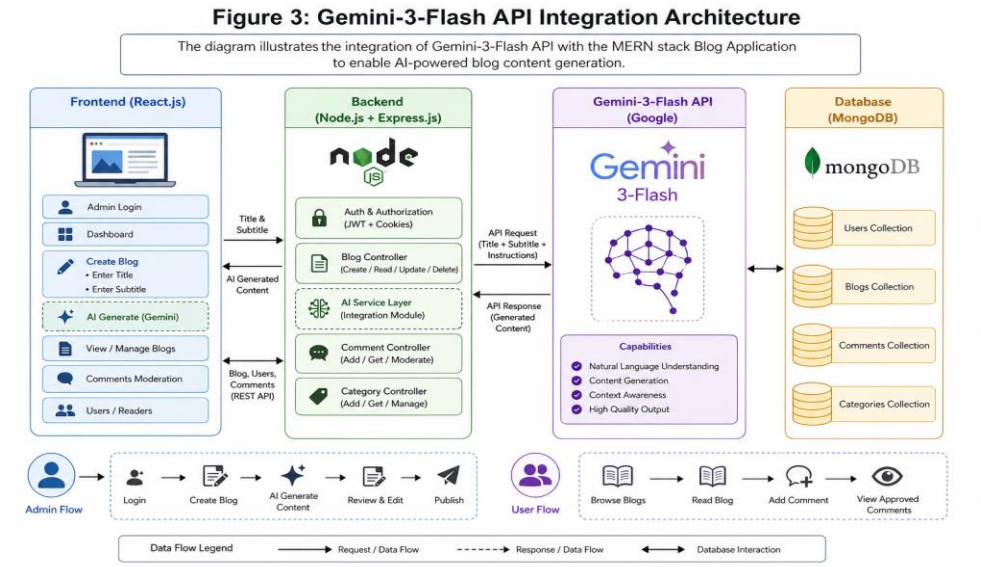

Figure 3: Gemini-3-Flash API Integration Architecture Here

X. DATABASE SCHEMA DESIGN

The data architecture relies on two primary MongoDB collections.

Table 1: Blog Schema Architecture

Field Name	Data Type	Description
_id	ObjectId	Unique identifier
title	String	Main heading of the blog
subtitle	String	Contextual sub-heading
description	String	AI-generated rich HTML text
category	String	Enum (Tech, Startup, Lifestyle, Finance)
thumbnail	String	Cloud URL of the uploaded image
status	String	Enum (Published, Draft)
createdAt	Date	Timestamp of creation

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154280

Table 2: Comment Schema Architecture

Field Name	Data Type	Description
_id	ObjectId	Unique identifier
blogId	ObjectId	Relational reference to Blog Schema
userName	String	Name of the commenter
commentText	String	The submitted comment
isApproved	Boolean	Defaults to false (Moderation flag)

XI. TESTING, FEASIBILITY & SECURITY

A. Feasibility Study

- **Technical Feasibility:** The MERN stack is highly capable of supporting this architecture. Integrating the Gemini API is technically viable via standard HTTP requests.
- **Economic Feasibility:** The project utilizes open-source frameworks. Cloud deployment (Vercel/Render) and MongoDB Atlas offer free tiers, making the project highly cost-effective.
- **Operational Feasibility:** The intuitive GUI of the Admin Dashboard ensures that users with minimal technical expertise can operate the CMS, generate AI content, and moderate comments effectively.

B. Security Mechanisms

- **Cross-Site Scripting (XSS) Prevention:** Because the AI generates HTML that is rendered via React's dangerouslySetInnerHTML, strict sanitization libraries (like DOMPurify) are implemented to strip any malicious JavaScript injections.
- **Authentication Security:** Storing JWTs in HttpOnly cookies prevents token theft. Route protection ensures that endpoints for deleting blogs or approving comments cannot be accessed without valid admin cookies.
- **API Rate Limiting:** The backend implements rate limiting on the AI generation endpoint to prevent abuse and excessive API billing.

C. Testing Strategies

1. **Unit Testing:** Individual endpoints (e.g., the comment submission API) were tested for expected status codes (200 OK, 400 Bad Request).
2. **Integration Testing:** Verified the data flow between React, Express, and MongoDB. Specifically tested the comment approval flow to ensure that toggling isApproved to true immediately renders the comment on the frontend.
3. **User Acceptance Testing (UAT):** Simulated real-world scenarios where an admin logs in, filters blogs, generates a post using Gemini, and handles incoming mock comments. The system performed flawlessly under simulated load.

XII. CONCLUSION

The development of the "AI-Driven Content Management System" successfully demonstrates the profound impact of integrating Large Language Models into standard web architectures. By leveraging the MERN stack, the application provides a fluid, high-performance user experience. The integration of the Gemini-3-Flash-Preview model resolves the most significant bottleneck in blogging—content generation—by reducing drafting time from hours to seconds.

Furthermore, the implementation of a 1-to-N user model with a robust Admin Dashboard ensures that the platform remains secure, organized, and free from spam via the strict comment moderation workflow. This research project serves as a comprehensive blueprint for next-generation digital publishing platforms, proving that AI-assisted tools are no longer optional accessories, but essential core components of modern web development.

REFERENCES

[1]. MongoDB Inc., "MongoDB Atlas and Document Database Architecture," 2024. [Online]. Available: https://docs.mongodb.com/.
[2]. Node.js Foundation, "Asynchronous Event-Driven Architecture in Node.js," 2024. [Online]. Available: https://nodejs.org/en/docs/.
[3]. Meta Open Source, "React: A JavaScript library for building user interfaces," 2024. [Online]. Available: https://reactjs.org/.
[4]. Google DeepMind, "Gemini API Documentation and Prompt Engineering Guide," 2024. [Online]. Available:

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026
DOI: 10.17148/IJARCCE.2026.154280

https://ai.google.dev/.
[5]. MDN Web Docs, "Using HTTP Cookies and Preventing Cross-Site Scripting (XSS)," Mozilla Foundation, 2024.
[6]. A. Sahu and D. Kumar, "Design and Development of Online Content Platforms," International Journal of Advanced Research in Computer Science, vol. 10, no. 5, pp. 120-126, 2019.
[7]. P. Bhambri and S. K. Singh, "AI and Automation in Web Engineering," International Journal of Computer Applications, vol. 176, pp. 1-5, 2020.
[8]. GreatStack, "Full Stack Web Development Tutorials," YouTube. Available: https://www.youtube.com/.
[9]. Blogger, "About Blogger," Available: https://www.blogger.com/about/.
[10]. SiteBuilderReport, "Blog Design Examples and Inspiration," Available: https://www.sitebuilderreport.com/inspiration/blog-examples.
[11]. Wix, "Start a Blog with Wix," Available: https://www.wix.com/start/blog.