International Journal of
AI ELECTRICAL CIVIL AND MECHANICAL ENGINEERING
Peer Reviewed, Referred & Indexed Journal
ISSN: 3070-0434 www.zesterapublications.com Original Research Paper

# Blogify: A Full-Stack Social Media Blogging Web Application

**Millan Kumar Patra**
2201298109
Student, Dept. of CSE
GIFT Autonomous, Bhubaneswar

**Biswaprakash Nayak**
2201298317
Student, Dept. of CSE
GIFT Autonomous, Bhubaneswar

**Prof. Smruti Ranjan Swain**
Guide, Dept. of CSE
GIFT Autonomous, Bhubaneswar
BPUT, Rourkela, Odisha

***Abstract*** — The Blogify Social Media Web Application is a modern full-stack platform that combines blogging and social networking into a single unified system. Existing platforms either focus on long-form content publishing or short-form social interaction, creating a gap for users who need both. Blogify addresses this gap by providing blog creation, image uploads, likes, comments, a follower system, and personalized content feeds — all within one application. The system is developed using the MERN Stack (MongoDB, Express.js, React.js, Node.js) with JWT-based authentication for secure access control. All 5 core modules passed functional testing, and system performance metrics including API response times and UI responsiveness met defined targets, confirming readiness for production deployment.

**Keywords —** Blogify, Social Media, Blogging Platform, MERN Stack, JWT Authentication, React.js, Node.js, MongoDB, RESTful API, Full-Stack Web Application.

## I. INTRODUCTION

In today's digital era, the internet has transformed how people communicate, share knowledge, and express their ideas. Social media platforms and blogging websites have become essential tools for individuals worldwide. However, most traditional blogging platforms mainly focus on content publishing and lack strong social networking features, while many social media platforms prioritize short-form content over detailed knowledge sharing.

This creates a significant gap for users who want both professional blogging capabilities and meaningful social engagement in a single platform. The Blogify Social Media Web Application is designed to bridge this gap by combining the power of blogging with the interactivity of social media.

The platform allows users to register securely, write and publish blog posts, upload images, interact through likes and comments, follow other users, and receive personalized content feeds — all in one centralized environment built on the MERN Stack.

### A. Problem Statement

Existing platforms for blogging and social networking suffer from several limitations:

- Most systems provide either blogging or social networking, but not both in an integrated manner.
- Traditional blogging platforms lack real-time engagement features such as likes, follows, and personalized feeds.
- Social media platforms focus on short-form content and are not optimized for long-form article publishing.
- Users are forced to use multiple platforms for content creation and social engagement, increasing complexity.
- Poor content discoverability and lack of personalized recommendations reduce user satisfaction.

### B. Objectives

- To develop a centralized full-stack blogging platform integrated with social networking features using the MERN Stack.
- To implement secure user authentication and authorization using JWT (JSON Web Token).
- To enable social interaction through likes, comments, and a follow/unfollow system.
- To generate personalized content feeds based on user interests and social connections.
- To build a responsive and modern user interface accessible on desktop and mobile devices.
- To provide an admin dashboard for monitoring users, posts, and platform activities.

## II. LITERATURE REVIEW

Social media and blogging systems have evolved significantly with the advancement of web technologies. Research indicates that web-based

## IV. MODULE DESCRIPTION

### A. User Management Module

Manages user accounts and profiles. Users can register, edit their profile, upload a profile picture, and track their posts, followers, and following counts. Profile data is stored securely in the Users collection.

### B. Authentication Module

Implements secure login and logout using JWT stored in HttpOnly cookies. Passwords are hashed with bcrypt before storage. Protected routes verify the JWT token via Express middleware before granting access.

### C. Post Creation Module

The core content module. Users can create rich blog posts with titles, body content, and image uploads via Cloudinary. Posts can be edited and deleted by their authors. All posts are stored in the Posts collection.

### D. Social Interaction Module

Handles likes, comments, and the follow/unfollow system. Users can like or unlike any post, add comments to initiate discussions, and follow or unfollow other users to curate their personalized feed.

### E. Feed & Admin Module

The Feed Module generates a personalized content feed by aggregating posts from followed users and trending content. The Admin Module provides role-based access for administrators to monitor users, manage posts, remove inappropriate content, and maintain platform security.

## V. IMPLEMENTATION

### A. Frontend Development

The frontend follows a component-based architecture. Directory structure separates pages (top-level views), components (reusable UI elements), store (Redux Toolkit slices), and services (Axios API calls). Redux state is divided into slices: authSlice, postSlice, feedSlice, and socialSlice. Each slice uses createAsyncThunk for async API calls with loading and error state handling.

### B. Backend Development

The Express.js server configures CORS middleware for frontend-backend communication, cookie-parser for HttpOnly JWT cookies, and JSON body parser. The authentication controller hashes passwords using bcryptjs with a salt factor of 12 and signs JWT tokens with a defined expiry period. The post controller handles Cloudinary image uploads by converting Multer memory buffer to base64 before uploading via the Cloudinary Node.js SDK.

### C. API Design

The system exposes RESTful API endpoints returning JSON with the consistent structure { success: boolean, data/message: value }. Key endpoint groups: /api/auth/* for authentication, /api/posts/* for blog operations, /api/social/* for likes, comments, and follows, and

Received: 20-04-2026 | Accepted: 28-05-2026 | Published: 04-06-2026

International Journal of
AI ELECTRICAL CIVIL AND MECHANICAL ENGINEERING
Peer Reviewed, Referred & Indexed Journal
ISSN: 3070-0434 www.zesterapublications.com Original Research Paper

platforms built using modern full-stack technologies such as the MERN Stack provide efficient, scalable, and maintainable solutions for real-time social applications [1].

Studies on React.js-based Single Page Application (SPA) architectures demonstrate that component-based rendering provides significantly better user experiences by eliminating full page reloads and enabling dynamic content updates [2]. This makes React.js a preferred choice for social media frontends.

MongoDB's document-oriented NoSQL model is well-suited for social networking data structures where relationships between users, posts, comments, and likes are dynamic and schema-flexible [3]. JWT-based stateless authentication is widely adopted in modern REST API systems for its scalability and security advantages over traditional session-based approaches [4].

Existing platforms such as Medium focus primarily on blogging while Twitter and Instagram prioritize social interaction. Research highlights the need for an integrated platform that combines both paradigms to improve user retention, engagement, and content discoverability [5].

## III. SYSTEM DESIGN

### A. System Architecture

Blogify follows a Three-Tier Architecture that cleanly separates concerns across the Presentation, Application, and Data layers. This modular design ensures each layer can be independently developed, tested, and scaled.

- Presentation Layer (Frontend): Built with React.js for dynamic component rendering and Redux for global state management. Communicates with the backend via RESTful API calls using Axios.
- Application Layer (Backend): Built with Node.js and Express.js. Handles all business logic, JWT authentication middleware, blog management, social interaction processing, and database communication.
- Data Layer (Database): MongoDB stores all user profiles, blog posts, comments, likes, and follower relationships in flexible JSON-like BSON documents via Mongoose ODM.

### B. Technology Stack

| Layer | Technologies |
|---|---|
| Frontend | React.js, HTML5, CSS3, JavaScript, Tailwind CSS / Bootstrap, Axios |
| Backend | Node.js, Express.js, JWT, bcrypt, REST APIs, Middleware functions |
| Database | MongoDB, Mongoose ODM |
| Media | Cloudinary (image/media storage) |
| Dev Tools | VS Code, Postman, GitHub, MongoDB Compass |

### C. Database Design

The system uses six primary MongoDB collections:

- Users: Stores user credentials, profile picture, bio, followers/following counts, and creation date.
- Posts: Stores post ID, author reference, title, blog content, image URL, likes count, comments count, and creation date.
- Comments: Stores comment ID, post reference, user reference, comment text, and timestamp.
- Likes: Stores like ID, user reference, post reference, and liked date.
- Followers: Stores follower-following user relationships and follow date.
- Admin: Stores admin credentials, role, and activity logs for platform moderation.

/api/admin/* for administrative operations. All sensitive routes are protected with JWT middleware.

### D. System Workflow

- User registers and logs in via JWT-authenticated endpoints.
- User creates a blog post with optional image upload via Cloudinary.
- System stores post data securely in MongoDB.
- Other users interact via likes, comments, and follow actions.
- Personalized feed aggregates posts from followed users and trending content.
- Admin monitors and moderates platform activities via the Admin Dashboard.

## VI. TESTING AND RESULTS

### A. Testing Overview

The system underwent unit testing of individual module functions, integration testing of frontend-backend-database communication, system testing through complete end-to-end user workflows, and browser compatibility testing across Chrome, Firefox, Edge, and Safari. Responsive design was verified from 320px to 1920px screen widths.

### B. Functional Test Results

A total of 15 test cases were designed and executed covering all modules. All 15 test cases passed successfully with no failures detected.

| TC ID | Module | Test Description | Result |
|---|---|---|---|
| TC01 | Auth | Register with valid credentials | Pass |
| TC02 | Auth | Register with existing email — error | Pass |
| TC03 | Auth | Login with correct credentials | Pass |
| TC04 | Auth | Login with wrong password — error | Pass |
| TC05 | Auth | Logout — session cleared | Pass |
| TC06 | Posts | Create blog post with image upload | Pass |
| TC07 | Posts | Edit existing blog post | Pass |
| TC08 | Posts | Delete blog post | Pass |
| TC09 | Social | Like a post — count increments | Pass |
| TC10 | Social | Unlike a post — count decrements | Pass |
| TC11 | Social | Add comment to a post | Pass |
| TC12 | Social | Follow a user — appears in feed | Pass |
| TC13 | Social | Unfollow a user — removed from feed | Pass |
| TC14 | Feed | Personalized feed loads correctly | Pass |
| TC15 | Admin | Admin removes inappropriate post | Pass |

*TABLE I. Functional Test Case Results*

### C. Performance Analysis

System performance was evaluated across key metrics for both frontend and backend. The frontend achieved a Lighthouse score of 90/100 on desktop with fast paint times, confirming an optimized user experience.

| Metric | Measured Value | Target | Status |
|---|---|---|---|
| API Avg Response Time | < 160 ms | < 200 ms | Pass |
| Frontend FCP (Desktop) | 0.9 sec | < 1.8 sec | Pass |
| Frontend LCP (Desktop) | 1.7 sec | < 2.5 sec | Pass |
| Lighthouse Score | 90/100 | > 85/100 | Pass |
| Image Upload (Cloud) | 1.4 sec | < 3 sec | Pass |
| Feed Load Time | < 500 ms | < 1 sec | Pass |
| Concurrent Users | 20+ | > 10 | Pass |

*TABLE II. Performance Analysis*

Received: 20-04-2026 | Accepted: 28-05-2026 | Published: 04-06-2026

International Journal of
AI ELECTRICAL CIVIL AND MECHANICAL ENGINEERING
Peer Reviewed, Referred & Indexed Journal
ISSN: 3070-0434 www.zesterapublications.com Original Research Paper

### D. Comparison with Existing Systems

| Aspect | Medium / WordPress | Twitter / Instagram | Blogify |
|---|---|---|---|
| Content Type | Long-form blogs only | Short-form only | Both long & short-form |
| Social Interaction | Limited | High | High |
| Personalized Feed | Basic | Algorithm-based | Follow-based + trending |
| Community Building | Weak | Moderate | Strong |
| Integrated Platform | No | No | Yes — unified |
| Authentication | Basic | Basic | JWT secured |

*TABLE III. Comparison with Existing Systems*

## VII. CONCLUSION

This paper successfully demonstrated the complete design, development, and testing of Blogify — a full-stack Social Media Blogging Web Application using the MERN Stack. The system effectively addresses the limitations of traditional blogging and social networking platforms by integrating both paradigms into a single, centralized, and scalable platform.

The implementation of JWT-based authentication, Cloudinary media storage, personalized feed generation, and a follow/like/comment system provides users with a rich and engaging experience. React.js delivered a dynamic and responsive frontend, while Node.js and Express.js ensured efficient backend operations. MongoDB provided flexible and scalable data storage for all social and content entities.

All 15 functional test cases passed and all performance metrics met or exceeded defined targets, confirming the system is well-optimized and ready for production deployment.

**Future Work**

- AI-based personalized content recommendation engine using collaborative filtering.
- Real-time notifications using WebSockets (Socket.IO) for likes, comments, and follows.
- Direct messaging and chat system between users.
- React Native mobile application for Android and iOS platforms.
- Creator monetization features including subscriptions and premium content.
- Advanced analytics dashboard for content performance and audience insights.
- Multi-language support for global accessibility.
- Progressive Web App (PWA) support for offline reading.

## REFERENCES

[1] I. Sommerville, Software Engineering, 10th ed. Pearson Education, 2016.
[2] A. Banks and E. Porcello, Learning React. O'Reilly Media, 2020.
[3] B. Dayley, Node.js, MongoDB and Angular Web Development. Addison-Wesley, 2018.
[4] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," Ph.D. dissertation, Univ. of California, Irvine, 2000.
[5] K. C. Laudon and C. G. Traver, E-commerce: Business, Technology, Society, 15th ed. Pearson Education, 2019.
[6] React Documentation, Meta Open Source, 2024. [Online]. Available: https://reactjs.org/docs/
[7] Node.js Documentation, OpenJS Foundation, 2024. [Online]. Available: https://nodejs.org/en/docs/
[8] MongoDB Documentation, MongoDB Inc., 2024. [Online]. Available: https://www.mongodb.com/docs/
[9] OWASP Foundation, OWASP Top Ten Web Application Security Risks, 2021.
[10] Cloudinary Documentation, Cloudinary Ltd., 2024. [Online]. Available: https://cloudinary.com/documentation/
[11] D. Flanagan, JavaScript: The Definitive Guide, 7th ed. O'Reilly Media, 2020.
[12] R. C. Martin, Clean Code: A Handbook of Agile Software Craftsmanship. Pearson Education, 2008.

Received: 20-04-2026 | Accepted: 28-05-2026 | Published: 04-06-2026