



Industrial Internship Report

Bharti Soft Tech Pvt. Ltd.

Submitted By

KETAN BADGUJAR

12102110501009

In partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

Computer Science & Engineering (IOT)

G H Patel College of Engineering & Technology

The Charutar Vidya Mandal (CVM) University,

Vallabh Vidyanagar – 388120

May, 2025



G H Patel College of Engineering & Technology

Computer Science & Engineering (IOT)

CERTIFICATE

This is to certify that Ketan Badgujar (12102110501009) has submitted the Industrial Internship report based on internship undergone at Bharti Soft Tech Pvt. Ltd. for a period of 16 weeks from 06/01/2025 to 30/04/2025 in partial fulfillment for the degree of Bachelor of Technology in CSE(IOT), G H Patel college of engineering & technology at The Charutar Vidya Mandal (CVM) University, Vallabh Vidyanagar during the academic year 2024 – 25.

Dr. Miral Patel

Internal Guide

Dr. Nikhil Gondaliya

Head of the Department



Bharti Soft Tech Pvt. Ltd.

Date: 30/04/2025

TO WHOM IT MAY CONCERN

This is to certify that KETAN BADGUJAR, a student of BACHELOR OF COMPUTER SCIENCE & ENGINEERING [IoT] of G H PATEL COLLEGE OF ENGINEERING & TECHNOLOGY, affiliated with CVM UNIVERSITY, VALLABH VIDYANAGAR, has successfully completed his internship in the field of MACHINE LEARNING from 06/01/2024 to 30/04/2025 under the guidance of Mrs. Nimisha Vyas, Project Manager at Bharti Soft Tech Pvt Ltd. His internship activities include successful completion of the assigned project at the given period of time along with abiding by companies' rules and regulations. During the period of his internship program with us, he had been exposed to different processes and was found diligent, hardworking, and inquisitive. We Wish him every success in his life and career.

For Bharti Soft Tech Pvt. Ltd



DECLARATION

I, Ketan Badgujar (12102110501009), hereby declare that the Industrial Internship report submitted in partial fulfillment for the degree of Bachelor of Technology in Computer Science & Engineering (IoT), G H Patel college of Engineering & Technology, The Charutar Vidya Mandal (CVM) University, Vallabh Vidyanagar, is a Bonafide record of work carried out by me at Bharti Soft Tech Pvt. Ltd. under the supervision of Mrs. Nimisha Vyas and that no part of this report has been directly copied from any students' reports or taken from any other source, without providing due reference.

Name of the student

Sign of student



ACKNOWLEDGEMENT

I would like to express my sincere gratitude to Bharti Soft Tech Pvt. Ltd. for providing me with the opportunity to undertake my industrial internship at their esteemed organization. This experience has been invaluable for my professional growth and has significantly enhanced my understanding of real-world software development practices.

My special thanks to Mrs. Nimisha Vyas [Project Manager], for her guidance and support throughout my internship period.

I would also like to express my appreciation to Dr. Miral Patel, my internal guide from G H Patel College of Engineering & Technology, for the constant guidance, suggestions, and encouragement which helped me to complete this project successfully.

I am grateful to Dr. Nikhil Gondaliya, Head of the Department of Information Technology, for providing the necessary infrastructure and resources required for the completion of my internship project.

Finally, I would like to thank my family and friends for their unwavering support and encouragement throughout my academic journey.

KETAN BADGUJAR
12102110501009



ABSTRACT

*This internship report outlines the development of a full-stack **Blog application** built using modern web technologies, primarily **Next.js** for the frontend and backend logic, **MongoDB** as the database, **Postman** for API testing, and **Git** for version control. The internship was carried out at Bharti Soft Tech Pvt. Ltd. where the focus was to create a fast, scalable, and responsive platform for users to publish, manage, and interact with blog content efficiently.*

The blog application was designed to support multiple key features such as blog creation and editing, comment management, category-based filtering, and profile handling. Using Next.js allowed server-side rendering (SSR) and static site generation (SSG), which significantly improved page load times and SEO performance. MongoDB provided a flexible and schema-less database solution to manage blog posts, user data, and interactions efficiently.

RESTful APIs were created and thoroughly tested using Postman to ensure smooth communication between the frontend and backend. Emphasis was placed on building reusable React components, maintaining clean code, and following responsive design principles for better user experience across all devices.

The report provides a detailed walkthrough of the development lifecycle, from the planning and design phases through to implementation, testing, and deployment. Additionally, the internship experience strengthened understanding of software development practices, collaboration through Git, and the importance of thorough API documentation using Postman.



List of Figures

Figure 1 : System Architecture	23
Figure 2 : State Transition for Landing Page	24
Figure 3 : Sequence Diagram	27
Figure 4 : Website	34
Figure 5 : Admin Dashboard	34
Figure 6 : Blogs Created	37

Table of Content

DECLARATION	i
ACKNOWLEDGEMENT	ii
ABSTRACT	iii
List of Figures	iv
CHAPTER 1 OVERVIEW OF THE COMPANY	1
1.1 COMPANY HISTORY.....	2
1.2 PRODUCTS AND SERVICES.....	2
1.3 ORGANIZATION STRUCTURE.....	3
1.4 OPERATIONAL CAPACITY.....	3
1.5 DEPARTMENT OVERVIEW.....	3
1.5.1 Software Development.....	3
1.5.2 IT Infrastructure Management.....	4
1.5.3 Hardware Maintenance.....	4
1.5.4 Technical Support.....	4
CHAPTER 2 INTRODUCTION TO PROJECT	5
2.1 PROJECT SUMMARY.....	6
2.2 PURPOSE.....	7
2.3 OBJECTIVES.....	8
2.3.1 Primary Objectives.....	8
2.3.2 Technical Objectives.....	10
2.4 SCOPE.....	11
2.4.1 In Scope.....	11
2.4.2 Out of Scope:.....	12
2.5 TECHNOLOGY AND LITERATURE REVIEW.....	12
2.5.1 Technology Stack.....	12
2.5.2 Literature Review.....	13

CHAPTER 3 SYSTEM ANALYSIS	15
3.1 STUDY OF CURRENT SYSTEM.....	16
3.2 PROBLEM AND WEAKNESSES OF CURRENT SYSTEM.....	16
3.3 REQUIREMENTS OF NEW SYSTEM.....	16
3.4 SYSTEM FEASIBILITY.....	17
3.4.1 Contribution to Organization's Objectives.....	17
3.4.2 Feasibility within Current Technology, Cost & Schedule.....	17
3.4.3 Integration with Existing Systems.....	17
3.5 PROPOSED SYSTEM WORKFLOWS.....	17
3.6 FEATURES OF NEW SYSTEM / PROPOSED SYSTEM.....	18
3.7 MAIN MODULES, PROCESSES, AND TECHNIQUES OF THE PROPOSED CRM SYSTEM.....	18
3.8 SOFTWARE AND METHODOLOGY.....	20
CHAPTER 4 SYSTEM DESIGN	21
4.1 SYSTEM DESIGN & METHODOLOGY.....	22
4.2 INPUT/OUTPUT AND INTERFACE DESIGN.....	24
4.2.1 State Transition Diagram.....	24
4.3.2 Samples of Forms, Reports and Interface	24
4.3.3 Access Control/Mechanism/Security	25
4.3 SEQUENCE DIAGRAM.....	27
CHAPTER 5 IMPLEMENTATION	28
5.1 IMPLEMENTATION PLATFORM/ENVIRONMENT.....	29
5.2 MODULES SPECIFICATION(S).....	29
5.3 CHALLENGES FACED DURING IMPLEMENTATION.....	31
5.4 CODE QUALITY AND BEST PRACTICES FOLLOWED.....	32
5.5 RESULT.....	34
CHAPTER 6 TESTING	35
6.1 TESTING STRATEGY.....	36
6.2 TEST RESULTS AND ANALYSIS.....	37



CHAPTER 7 CONCLUSION AND DISCUSSION	38
7.1 OVERALL ANALYSIS OF INTERNSHIP/PROJECT VIABILITIES.....	39
7.2 INTERNSHIP PROGRESS REVIEW MEETING WITH INDUSTRY GUIDE.....	40
7.3 PROBLEM ENCOUNTERED AND POSSIBLE SOLUTIONS.....	40
7.4 SUMMARY OF INTERNSHIP.....	41
7.5 LIMITATION AND FUTURE ENHANCEMENT.....	42

CHAPTER 1

OVERVIEW OF THE COMPANY

1.1 COMPANY HISTORY

Founded in 2006, Bharti Soft Tech Pvt. Ltd. is an ISO 9001:2008 certified IT service provider with its headquarters in Paris, France, and operational centers in Germany and India. The company was established to deliver comprehensive product development and consulting services to a global clientele. Over the years, Bharti Soft Tech has evolved into a multifaceted IT solutions provider, executing turnkey projects across various industry verticals worldwide. The organization prides itself on a team of highly qualified, dynamic professionals with international experience, committed to delivering timely, reliable, user-friendly, and cost-effective solutions using the latest technologies.

1.2 PRODUCTS AND SERVICES

Bharti Soft Tech offers a wide array of IT solutions tailored to meet diverse business needs:

- **Software Development:** Custom software solutions, including responsive web and mobile applications, e-commerce platforms, and enterprise resource planning (ERP) systems.
- **IT Infrastructure Services:** Comprehensive services encompassing hardware maintenance, network design and management, service desk operations, messaging systems, and robust data backup solutions.
- **Software Licensing:** Implementation and support for Microsoft and Linux-based systems, along with efficient software license management.
- **Cloud Computing:** Development and deployment of applications using cloud platforms like Amazon Web Services (AWS), Google Cloud, and Microsoft Azure.
- **AI-Based Solutions:** Customized AI products extending traditional machine learning capabilities for active collaboration and decision-making.
- **Digital Marketing:** AI-driven digital marketing solutions aimed at enhancing client engagement and optimizing marketing strategies.
- **Testing and Quality Assurance:** Comprehensive testing services, including manual testing, automation, and security testing, ensuring the delivery of high-quality software products.

1.3 ORGANIZATION STRUCTURE

Bharti Soft Tech operates with a streamlined hierarchical structure to ensure effective communication and project execution:

- **Board of Directors:** Comprising experienced professionals providing strategic leadership and vision.
- **Executive Management:** Overseeing daily operations and strategic initiatives across various departments.
- **Departmental Teams:** Including department managers, associate managers, senior engineers, engineers, and trainees collaborating across functions to deliver integrated solutions, fostering accountability and career growth.

1.4 OPERATIONAL CAPACITY

With a workforce exceeding 90 professionals skilled in technologies like .NET, PHP, Oracle, and open-source platforms, Bharti Soft Tech boasts robust operational capabilities:

- **Infrastructure:** State-of-the-art development facilities equipped with modern hardware and software tools.
- **Project Management:** Adoption of Agile methodologies and rigorous quality assurance processes to ensure timely, high-quality project delivery.
- **Support Services:** Provision of 24/7 client support infrastructure for continuous assistance.
- **Talent Acquisition:** Strategic locations in Vadodara, Surat, Anand, and Mehsana, Gujarat, providing access to a steady talent pool, enhancing innovation and growth.

1.5 DEPARTMENT OVERVIEW

1.5.1 Software Development

The Software Development Department is the core of Bharti Soft Tech's innovation, designing tailored software solutions:

- **Team Composition:** Development leads, full-stack, front-end, back-end, QA, and DevOps engineers.

- **Technologies:** Expertise in MERN stack (MongoDB, Express.js, React.js, Node.js), .NET, PHP, React Native, and databases like MongoDB, MySQL, and Oracle, with Git for version control.
- **Development Process:** Follows Agile methodology with two-week sprints, continuous integration/deployment (CI/CD), and tools like Visual Studio, Jira, Jenkins, and Figma.
- **Workflow:** Encompasses requirement analysis, system design, coding, testing, deployment, and ongoing maintenance, ensuring scalable and user-friendly applications.

1.5.2 IT Infrastructure Management

This department ensures seamless IT infrastructure for internal and client operations:

- **Team Composition:** Network engineers, system administrators, cloud and security specialists, and help desk personnel.
- **Services:** Network design and monitoring, server management, cloud operations (AWS, Azure, Google Cloud), data backups, and cybersecurity measures.
- **Tools Utilized:** Cisco routers, VMware, Nagios, Zabbix, and cloud platforms for reliable performance.

1.5.3 Hardware Maintenance

The Hardware Maintenance Department ensures optimal hardware performance:

- **Team Composition:** Desktop, server, network, and peripheral technicians, supported by inventory managers.
- **Services:** Preventive and corrective maintenance, hardware upgrades, deployment, asset tracking, and end-of-life equipment management.
- **Facilities:** Equipped with hardware labs, testing areas, anti-static workstations, and diagnostic tools for precise repairs.

CHAPTER 2

INTRODUCTION TO PROJECT

2.1 PROJECT SUMMARY

The Blogger – a blog app developed during this internship is a dynamic web-based application built using Next.js, MongoDB, and Postman, with Git for version control. The platform is designed to provide an intuitive and streamlined experience for both blog creators and readers by combining user-centric design with efficient backend operations.

The application comprises three core sections accessible through a centralized entry point:

- **Home Page:** This serves as the public-facing interface of the blog platform. It displays all previously created blogs, allowing users to browse posts with ease. Blogs are categorized based on topics, and users can filter them accordingly for a more focused reading experience. The home page also features a subscription section where users can enter their email addresses to subscribe to the blog and receive updates.
- **Get Started Dashboard:** Upon clicking the “Get Started” button, users are navigated to a three-part dashboard that provides essential blogging functionalities:
 1. **Add a Blog:** Enables users to create and submit new blog posts through a simple and effective form-based interface.
 2. **Blog Lists:** Displays all existing blogs with their timestamps, and includes the ability to delete posts as needed, offering easy content management.
 3. **Subscription Management:** Allows users to enter and manage their email addresses for blog updates.

Key features of the Blogger system include:

- Clean and responsive user interface using Next.js
- Category-wise blog filtering for focused content discovery
- Email subscription system for user engagement
- Blog creation and deletion functionalities
- Timestamp display for blog posts
- Modular and scalable architecture
- Secure data handling and API testing using Postman
- Version control and collaborative development using Git

The project was developed using modern development practices, including component-based architecture, responsive design, and secure data operations. Git ensured efficient version tracking and collaborative updates throughout the development lifecycle. Postman was instrumental in testing API endpoints for accurate backend communication with MongoDB.

This blogging platform offers a cohesive, user-friendly solution for blog publishing and discovery, representing a modern alternative to traditional blogging systems through simplified workflows and an engaging interface.

2.2 PURPOSE

The purpose of developing the *Blogger – a blog app* is to create a streamlined and accessible platform for managing, publishing, and exploring blog content in a structured, category-driven environment. In today's digital-first content landscape, individuals and small content creators need a centralized, user-friendly solution to share their thoughts while engaging audiences through a clean interface and effective subscription model.

The primary purposes of the Blogger app include:

1. Simplified Content Publishing

The Blogger app is designed to make blog creation intuitive and efficient. It allows users to add, view, and delete blog posts easily without requiring technical expertise. By offering a structured UI, the platform reduces friction in content generation, allowing bloggers to focus more on writing than on managing the technical aspects of publishing.

2. Category-Based Content Organization

To improve content discoverability and user experience, the application includes category filters. This allows readers to easily explore blogs based on specific topics of interest, ensuring relevant content reaches the right audience. It enhances navigation while supporting content curation around specific themes.

3. Subscription and Audience Engagement

A major purpose of the project is to enable bloggers to build and retain an audience. The

the email subscription feature invites users to subscribe to the blog, helping content creators reach readers directly. This functionality supports ongoing engagement, reader retention, and potential content monetization strategies in future versions.

4. Backend Functionality and Blog Management

Through the blog list and management section, users can access all created blogs with timestamps and delete entries as needed. This administrative capability ensures that users can maintain their blog space efficiently, manage outdated content, and track blog publishing history.

5. Responsive and Scalable Web Application

Built with **Next.js** and **MongoDB**, the app offers a high-performance, scalable architecture that can grow with increased content and user activity. It is designed to deliver a responsive experience across all devices, making content accessible on desktops, tablets, and smartphones alike.

6. Technical Skill Development

An academic and developmental goal of the project is to strengthen full-stack web development skills. It offers practical exposure to modern tools such as Git for version control, Postman for backend API testing, and MongoDB for NoSQL database management. This hands-on implementation of a real-world web application supports academic learning objectives and career readiness.

2.3 OBJECTIVES

The development of the *Blogger – a blog app* project is guided by a well-defined set of objectives that focus on delivering a seamless user experience, robust content management capabilities, and efficient subscription handling. These objectives address technical, functional, and user-centered outcomes essential for the successful implementation of a modern blogging platform

2.3.1 Primary Objectives

1. Develop a Centralized Blogging Platform

- Build a unified platform that allows content creators to add, manage, and delete blogs efficiently
- Implement structured storage of blogs using MongoDB with proper categorization and timestamping
- Ensure content is properly retrieved and displayed in real-time on the homepage

2. Enable Category-Based Blog Exploration

- Create an intuitive filtering mechanism that allows users to explore blogs based on predefined categories
- Ensure seamless navigation between general blog views and topic-specific blog lists
- Improve user engagement by presenting relevant content based on user interests

3. Simplify Blog Management

- Provide a clean interface for users to view all their created blogs in a single list
- Enable blog deletion to support flexible and dynamic content control
- Ensure each blog entry is accompanied by accurate creation timestamps

4. Facilitate Email-Based Subscriptions

- Design a user-friendly subscription form that captures and stores user email addresses
- Ensure validation and secure storage of subscription data in the backend
- Lay the foundation for future email campaigns or blog notifications

5. Deliver an Accessible and Responsive User Interface

- Create a modern UI using Next.js, ensuring responsiveness across devices and screen sizes
- Apply accessibility standards to ensure usability for a wide audience
- Maintain consistency in design across all components including homepage, blog manager, and subscription form

6. Improve User Engagement and Content Discovery

- Provide clear call-to-action buttons (e.g., “Get Started”) to encourage deeper interaction with the platform
- Guide users through content creation, reading, and subscription in a seamless flow
- Highlight featured or recent blogs to draw user attention

2.3.2 Technical Objectives

1. Build a Scalable Full-Stack Architecture

- Use the MERN (MongoDB, Express, React/Next.js, Node.js) stack to build a modular and maintainable application
- Ensure support for potential scale-up in content volume and user base
- Maintain a clear folder and code structure to ease debugging and feature expansion

2. Ensure High Performance and API Efficiency

- Use Postman to test and optimize RESTful API endpoints for CRUD operations
- Minimize latency in content rendering and database interactions
- Cache repetitive queries when necessary to reduce server load

3. Implement Version Control and Deployment Readiness

- Use Git for managing source code, version history, and collaborative development
- Ensure the project is deployment-ready with environmental configuration and build optimization
- Facilitate integration with deployment services such as Vercel or Netlify in future upgrades

4. Ensure Cross-Browser and Device Compatibility

- Test UI components and interactions on popular web browsers (Chrome, Firefox, Edge, Safari)
- Validate mobile responsiveness using developer tools and real devices
- Implement adaptive layouts and media queries to ensure optimal viewing on tablets and smartphones

2.4 SCOPE

The scope of the *Blogger – a blog app* project defines the functional and technical boundaries of the system, ensuring that development efforts remain focused on delivering a clean, user-friendly blogging platform. This section outlines what the system will include (**In Scope**) and what it will explicitly exclude (**Out of Scope**) for the current implementation phase.

2.4.1 In Scope

1. Home Page & User Interaction

- Hero section displaying a welcome message and a clear call-to-action (CTA) like "Get Started"
- Dynamic display of all existing blogs on the homepage, fetched from MongoDB
- Category filter buttons for browsing blogs by topics
- Responsive design for consistent experience across devices
- Smooth navigation between sections of the app

2. Blog Management

- Blog creation form with input fields for title, content, and category
- Functionality to add a new blog and automatically log the timestamp of creation
- Blog listing section showing all created blogs with options to view or delete
- Delete functionality with confirmation prompt to remove unwanted blogs

3. Email Subscription

- Email subscription form with frontend validation
- Backend functionality to store email addresses securely in MongoDB
- Simple acknowledgment on successful subscription

4. User Interface and Experience

- Modern and minimalistic UI designed using Next.js
- Full responsiveness for desktops, tablets, and smartphones
- Clear layout division: Home, Add Blog, Blog List, Subscription
- Smooth routing and visual feedback for user actions

5. Development and Testing Tools

1. Integration with Postman for testing and debugging API endpoints
2. Source code management using Git with proper version control practices
3. Environment setup for local development and potential deployment

2.4.2 Out of Scope:

- User authentication and login functionality (no login/signup system)
- Rich text formatting or image uploading in blog content
- Email automation for newsletters or blog updates to subscribers
- Admin/user role-based access control
- SEO optimization features for public blogs
- Analytics dashboard for blog views or user engagement
- Comments section or user interaction on blogs
- Integration with third-party blogging platforms (e.g., Medium, WordPress)
- Deployment to a live server or domain hosting
- Mobile app version of the blogging system

2.5 TECHNOLOGY AND LITERATURE REVIEW

2.5.1 Technology Stack

The *Blogger – a blog app* project is built using the **MERN stack**, with a key modification: it leverages **Next.js** instead of Express.js for enhanced performance, built-in routing, and SEO-friendly server-side rendering. The technology stack supports full-stack development, scalable architecture, and a smooth developer experience.

1. MongoDB

- A NoSQL, document-oriented database used to store blog data and subscriber information
- Offers flexible schema design, suitable for unstructured and evolving content
- Scalable horizontally to accommodate increasing data volume

2. Next.js

- A React-based full-stack framework used for both frontend and backend logic

- Provides server-side rendering (SSR) and static site generation (SSG) for performance and SEO benefits
- Includes built-in API routes for handling server-side operations like blog CRUD and subscription handling
- Offers routing, image optimization, and fast refresh out of the box

3. React.js

- JavaScript library for building user interfaces with reusable components
- Powers the dynamic frontend with a responsive and interactive design
- Virtual DOM enables efficient rendering and state management

4. Additional Technologies

- **Git**: Version control system for collaborative development
- **Axios**: Promise-based HTTP client used for making API calls
- **Mongoose**: ODM library for structured and schema-based interaction with MongoDB
- **Nodemailer**: Enables sending confirmation and notification emails to subscribers
- **Bcrypt** (*planned for future enhancement*): For secure password hashing in user login features, if implemented
- **Postman**: API testing and debugging during development
- **CSS Modules / TailwindCSS** (*optional*): For styling modular and responsive UI components

3.5.2 Literature Review

1. Blog Applications and Content Management Trends

Blogging systems have evolved from simple text-based platforms to feature-rich, content-driven applications. According to *HubSpot's State of Marketing Report (2023)*, blogs remain one of the top three content strategies for increasing website traffic and lead generation. Modern blog apps prioritize readability, content filtering, and personalization.

2. Effectiveness of MERN Stack in Web Development

The MERN stack continues to gain traction for its unified JavaScript ecosystem. According to the *Stack Overflow Developer Survey 2023*, React remains the most

loved frontend library, while MongoDB ranks among the top preferred databases. The MERN stack supports agile development practices and offers high performance for scalable web apps.

3. User Experience in Web Applications

Simple and responsive UI design has become essential in retaining user engagement. Research from the *Nielsen Norman Group* highlights that users are more likely to interact with websites that feature minimalistic design, easy navigation, and mobile compatibility. This blog app incorporates these principles with a clear interface and responsiveness.

4. Email Subscription in User Engagement

Email subscription remains an effective method for content distribution and user retention. According to *Campaign Monitor (2023)*, personalized email updates have a 26% higher open rate and 14% greater click-through rate compared to generic newsletters. The subscription feature in this blog app is designed with this in mind to support future email marketing extensions.

CHAPTER 3

SYSTEM ANALYSIS

3.1 STUDY OF CURRENT SYSTEM

The existing scenario for blog and content sharing platforms is either highly generalized (like Medium or Blogger) or overly complex for small-scale usage. For developers or content creators looking for a minimal yet feature-rich blogging platform with category filtering, blog management, and subscriber interaction, current platforms offer limited customization and data ownership. Additionally, small-scale bloggers lack analytics and content control over third-party platforms.

3.2 PROBLEM AND WEAKNESSES OF CURRENT SYSTEM

The analysis revealed several critical weaknesses in the existing system:

- **Lack of Customization:** Most existing platforms do not offer the ability to customize blog layouts or user interactions unless premium services are purchased.
- **Data Ownership Issues:** Bloggers don't have complete control over the data, as it's stored and managed by third-party platforms.
- **No Direct Subscriber Engagement:** Many blogging tools lack efficient email subscription or management features to directly notify users of updates.
- **Complex Interfaces:** Platforms can be too complicated for beginners or users who need a straightforward experience for posting and managing content.
- **Limited CRUD Control:** Fine-grained control over blog editing, deletion, and time tracking is often hidden behind complex dashboards or unavailable in free versions.

3.3 REQUIREMENTS OF NEW SYSTEM

- A **simple, intuitive blog management interface** with clear navigation.
- **User roles** to separate content creators from general viewers.
- **Blog CRUD operations** with timestamps and delete control.
- **Category-based filtering** for blog discovery.
- **Email subscription** feature to notify users of updates.
- A **responsive and mobile-friendly UI** using Next.js and React.

- Backend APIs and database setup using **MongoDB** and **Next.js API routes** for full-stack control.

3.4 SYSTEM FEASIBILITY

a) Contribution to Organization's Objectives

Yes. The system aligns with organizational goals by:

- Promoting knowledge sharing and blogging culture among developers and users.
- Enhancing visibility and SEO for creators through well-structured, clean blog layouts.
- Building a scalable platform for future growth, like analytics or monetization.

b) Feasibility within Current Technology, Cost & Schedule

Yes. The system uses modern, **open-source technologies (Next.js, MongoDB, React.js)**, minimizing licensing costs. Hosting can be managed through **Vercel, Netlify, or Render**, and the system is fully implementable within a student or early-stage project timeline (4–8 weeks).

c) Integration with Existing Systems

Yes. The system supports integration via:

- API-based architecture allowing easy integration with mailing services (e.g., Mailchimp, SendGrid)
- Google services (optional) for future extension into calendar/event-based blog publishing
- Analytics tools like Google Analytics or Plausible for tracking blog performance

3.5 PROPOSED SYSTEM WORKFLOWS

- **User visits the homepage** with a blog list and filtering options.
- **User subscribes via email**, triggering a confirmation using Nodemailer.
- **Admin logs in**, adds blogs with content, category, and timestamp.
- Blogs are **stored in MongoDB** and fetched dynamically using Next.js API routes.

- **Admin can edit/delete** any blog post with immediate effect.
- **Email notifications** may be sent to subscribers on new blog posts (optional enhancement).
- Admin or user can manage **subscription data** via dashboard.

3.6 FEATURES OF NEW SYSTEM / PROPOSED SYSTEM

- **Homepage Display:** List of blogs with categories and timestamps
- **Email Subscription:** Users can subscribe for updates
- **Get Started Section:**
 - Add Blog (title, content, timestamp)
 - Blog List with edit/delete functionality
 - Subscription list view
- **Admin/User Roles:** Role-based view and access
- **Responsive UI:** Mobile-friendly and accessible interface
- **Modern Tech Stack:** Built with Next.js, MongoDB, and React
- **Scalable Backend:** Easy to extend with APIs, email services, and analytics
- **Deployment Ready:** Can be deployed on Vercel or similar platforms
- **Security Features:** Basic input validation and form protection
- **Clean UI/UX:** Smooth, distraction-free reading and posting experience

3.7 MAIN MODULES, PROCESSES, AND TECHNIQUES OF THE PROPOSED CRM SYSTEM

Landing Page Module

- Hero Section (Value Proposition)
- Features and Blog Category Overview
- Contact Form & Email Subscription

Authentication Module

- Sign Up / Sign In
- Role-Based Access (Admin/User)

- "Remember Me" & Password Recovery

Admin Dashboard Module

- Add, Edit, Delete Blogs (CRUD)
- View Subscribers
- Track Timestamps and Category Filtering

Blog Management Module

- Display Blogs in List Format
- Category-wise Filtering
- Blog Timestamp and Sorting Logic

Subscription Module

- Email Input and Validation
- Save Subscriber Info to Database
- Notification Trigger (optional via Nodemailer)

User Interface Module

- Responsive Design (Desktop/Mobile)
- Light/Dark Mode (optional)
- Smooth Navigation and Clean Layout

API Communication Layer

- Next.js API Routes for Backend Logic
- CRUD Operations for Blogs and Subscribers

Database Module

- MongoDB Collections for Blogs and Subscribers
- Mongoose Schema Definitions and Validation

3.8 SOFTWARE AND METHODOLOGY

Hardware Selection

- **Development Machine:**
 - Minimum 8 GB RAM, i5 or equivalent processor, SSD recommended
 - Justification: Smooth development experience and fast build times
- **Deployment Environment:**
 - Vercel (Frontend & Backend)
 - Justification: Supports Next.js out of the box, offers CI/CD, serverless functions

Software Selection

1. Frontend:

- React.js + Next.js
- Justification: SEO-friendly, fast rendering via SSR, component-based structure

2. Backend:

- Next.js API Routes
- Justification: Integrated serverless backend, reduces project complexity

3. Database:

- MongoDB with Mongoose
- Justification: NoSQL structure ideal for blogs and subscriber data, schema flexibility

4. Version Control:

- Git & GitHub
- Justification: Industry-standard for tracking changes, collaboration, and deployment

5. HTTP Client:

- Axios
- Justification: Promise-based, easy API interaction

6. Styling:

- Tailwind CSS (optional)
- Justification: Utility-first, responsive design with minimal CSS code

CHAPTER 4

SYSTEM DESIGN

4.1 SYSTEM DESIGN & METHODOLOGY

The *Blogger – A Blog App* is designed with a modular, scalable architecture using the **Next.js** framework combined with **MongoDB**, ensuring a clean separation between client-side and server-side logic. The overall methodology emphasizes component reusability, maintainability, and user-focused design.

- **Separation of Concerns:** The project adopts a clear distinction between the frontend (Next.js pages and components) and backend API routes (Next.js API handlers). The database operations are isolated in separate utility files for clarity and reusability.
- **Component Reusability:** Reusable UI components such as BlogCards, CategoryFilters, Navbar, Footer, and Form Inputs are developed to maintain consistency across the application and reduce redundancy.
- **Scalability:** The app is structured to allow independent scaling of modules such as blog creation, category filtering, and subscription handling without affecting other parts of the system.
- **Maintainability:** A modular folder structure is followed, with logical grouping into components, pages, API routes, and services. This makes it easier to update, debug, or expand features in the future.
- **Security:** The app includes email subscription verification and secure API route handling using token-based validation and input sanitization to protect against injection attacks or unauthorized access.

System Architecture Diagram:

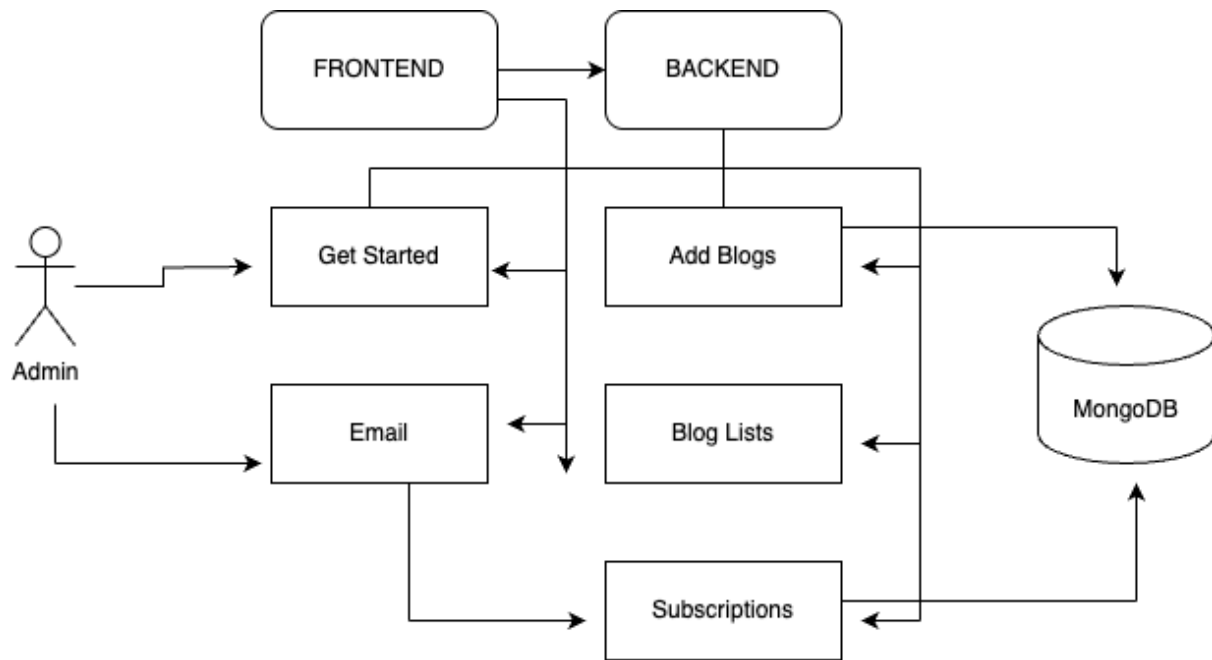


Figure 1 : System Architecture

Frontend Architecture:

- Component-based structure using React (via Next.js App Router)
- Client-server data fetching using axios for API integration
- File-based routing powered by Next.js dynamic routes (/blogs/[id])
- Responsive design with Tailwind CSS
- Reusable UI components in the /components directory (if applicable)
- Toast notifications with react-toastify for user feedback

Backend Architecture:

- API routes handled via Next.js API handlers (/api)
- MongoDB integration using mongoose for schema and data access
- Simple authentication logic (to be verified—if not included, recommend adding middleware)
- Modular service design for blog CRUD operations (e.g., controllers, models)
- Optional enhancements like middleware for logging, error handling, etc.

4.2 INPUT/OUTPUT AND INTERFACE DESIGN

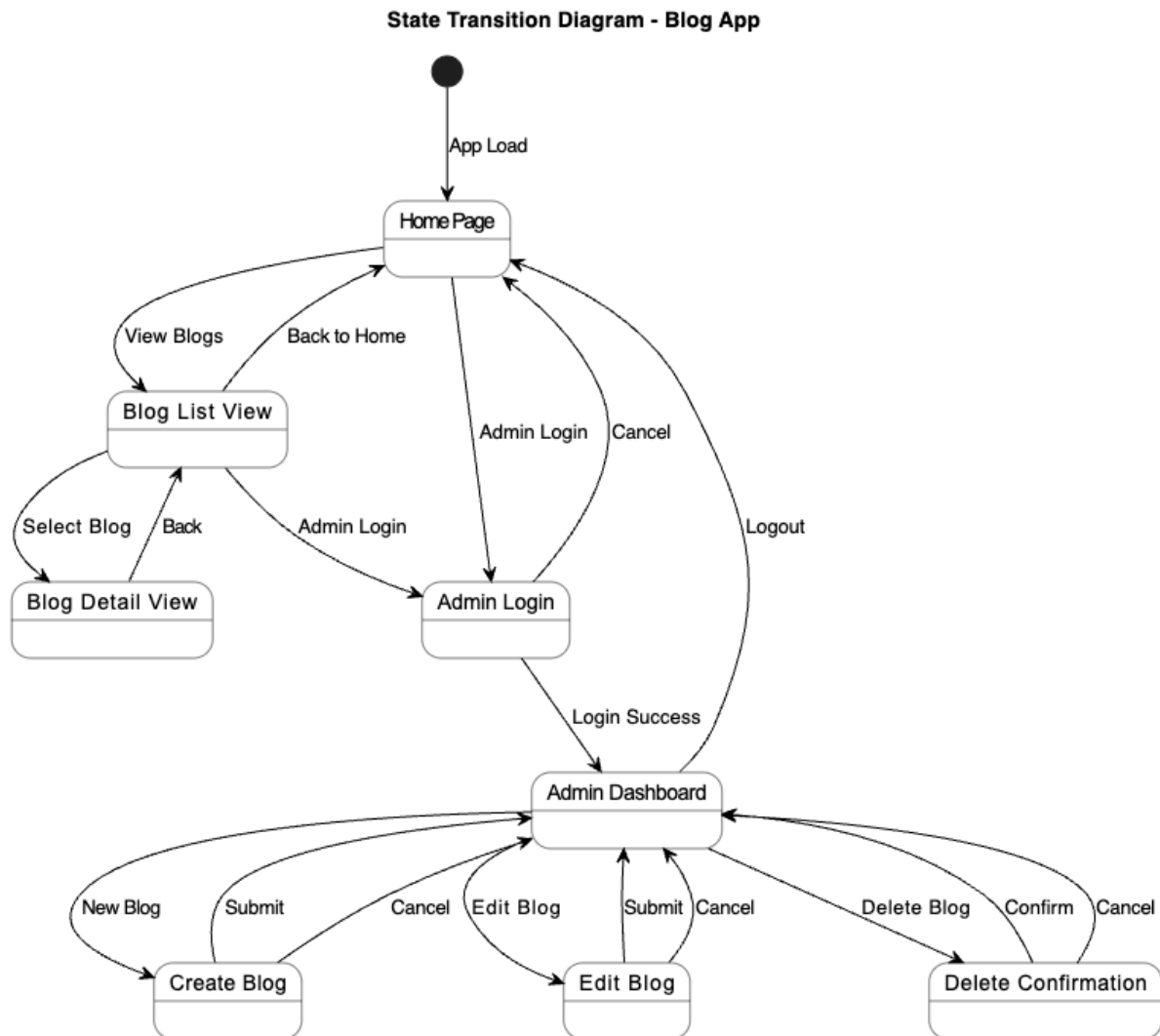


Figure 2 : State Diagram for Home Page

4.2.2 Samples of Forms, Reports, and Interfaces

The blog system includes several interfaces optimized for ease of use, content management, and administrative control:

Public Website Interface:

- Home page featuring recent blog posts and highlights
- Blog listing page with search and category filtering

- Individual blog detail pages with formatted content
- Contact or feedback form (optional enhancement)

Responsive design for mobile and desktop views

Admin Dashboard Interface:

- Dashboard overview with post stats and system health (optional)
- Blog management interface for creating, editing, and deleting posts
- Rich text editor for writing blog content
- Post list view with pagination, search, and date sorting
- Authentication-protected access to prevent unauthorized modifications

User Interface (if applicable):

- Login and signup forms
- User profile management (if implemented)
- Feedback form or comment section (optional addition)

4.2.3 Access Control / Mechanism / Security

The blog system enforces secure access and role-based control to maintain data integrity and privacy:

Authentication Mechanisms:

- Email/password-based login
- Session-based authentication with cookie or JWT (depending on implementation)
- Password reset flow (to be implemented if not yet included)
- Optional enhancements: Two-factor authentication (2FA)

Authorization Levels:

- Admin role:
 - Full access to post creation, editing, and deletion
 - Access to the admin dashboard and all post data
- Public user:
 - Read-only access to published blogs
 - No access to admin routes or unpublished content

Data Protection:

- Hashed and salted password storage using industry best practices (via **bcrypt** if implemented)
- Session timeout handling for inactive users
- Database backups and optional version control for content
- Environment-based configuration for secure API keys and DB URIs

4.3 SEQUENCE DIAGRAM

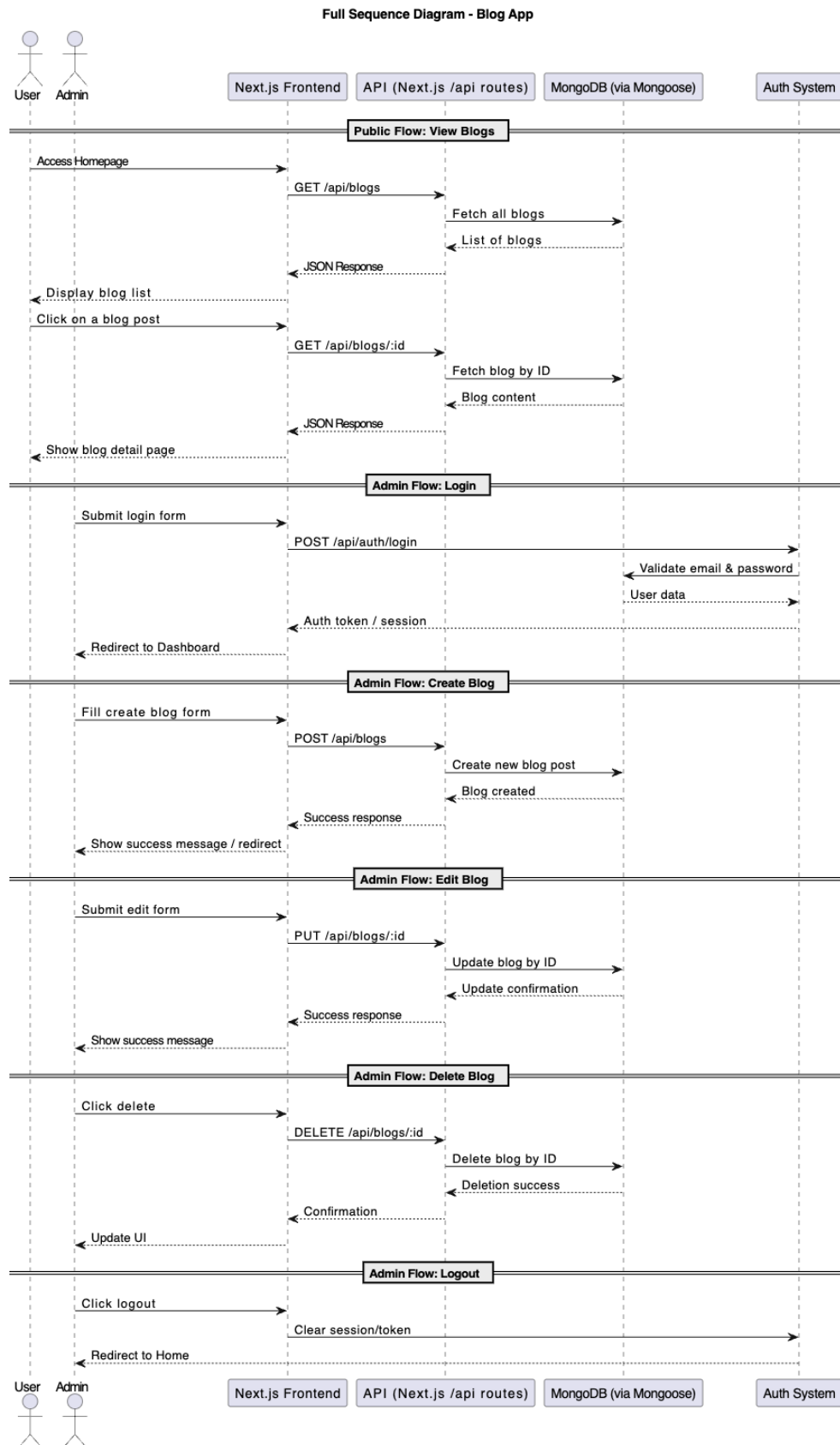


Figure 3 : Sequence Diagram

CHAPTER 5

IMPLEMENTATION

5.1 IMPLEMENTATION PLATFORM/ENVIRONMENT

The *Blogger – A Blog App* system was implemented using the following technology stack and development tools:

Frontend:

- Next.js for server-side rendered React components and routing
- Tailwind CSS for clean and responsive UI styling

Backend:

- Next.js API routes to handle server-side logic within the same application framework

Database:

- MongoDB for flexible and scalable document-oriented storage of blogs and subscriptions

Version Control:

- Git for source code management and collaborative development

API Testing:

- Postman for testing the blog creation, deletion, and subscription endpoints

Development Tools:

- **Visual Studio Code** – Primary code editor for development
- **Postman** – Used for testing and validating RESTful API requests
- **MongoDB Compass** – GUI tool for managing and visualizing MongoDB collections
- **GitHub** – Hosted the code repository for version control and collaboration

5.2 MODULES SPECIFICATION(S)

1. Blog Management Module:

- Add, view, and delete blog posts
- Categorization of blogs based on topics
- Display blogs by category on the homepage
- Blog listing with timestamp display for chronological order
- Option to delete a blog by authorized users

2. Category Filtering Module:

- Predefined blog categories (e.g., Technology, Lifestyle, Education, etc.)
- Dynamic filtering of blogs based on selected category
- Enhances user experience through focused content discovery

3. Subscription Module:

- Email subscription form available on homepage and dedicated section
- Stores subscriber emails securely in the database
- Email confirmation or notification capability (future scope)

4. Navigation and Routing Module:

- Get Started button for easy user navigation
- Routes to:
 - Add Blog section
 - Blog List section
 - Subscription section
- Utilizes Next.js routing for seamless page transitions

5. API & Backend Services Module:

- Uses Next.js API routes for backend operations
- Handles CRUD operations for blogs and subscriptions
- Ensures secure interaction between frontend and database

6. Database Module (MongoDB):

- Stores blog content with metadata (title, body, date, category)
- Stores subscriber information (email)
- Supports fast querying and document-based structure

7. Version Control Module:

- Git used for tracking changes and collaborative development
- Hosted on GitHub for repository management

8. Testing Module:

- API testing performed using Postman

- Validates API responses for blog creation, listing, and deletion
- Ensures robustness and correctness of backend logic

5.3 CHALLENGES FACED DURING IMPLEMENTATION

1. Dynamic Category-Based Filtering

- **Challenge:** Filtering blogs based on categories dynamically from the database without full page reloads posed UI consistency and performance issues.
- **Solution:** Implemented dynamic routing and server-side rendering with Next.js to fetch blogs by category. Utilized React state management to ensure smooth transitions and minimize re-renders.

2. Blog Deletion Consistency

- **Challenge:** Deleting a blog post required updating both the frontend state and backend data simultaneously, which led to occasional UI mismatch.
- **Solution:** Used optimistic UI updates and ensured backend confirmation through API response before final state update. Also implemented error handling to revert UI in case of failure.

3. Form Handling and Validation

- **Challenge:** Creating robust and user-friendly forms for blog creation and email subscription while maintaining data validation.
- **Solution:** Used controlled components in React and integrated validation using simple logic checks to prevent empty or malformed inputs before submission.

4. API Route Management with Next.js

- **Challenge:** Structuring Next.js API routes for CRUD operations and maintaining clean code was initially difficult.
- **Solution:** Organized API endpoints modularly (e.g., api/blogs, api/subscribe) and separated logic for GET, POST, and DELETE requests, ensuring maintainability and scalability.

5. MongoDB Query Optimization

- **Challenge:** Fetching and filtering a large number of blogs led to increased loading times.
- **Solution:** Implemented query filters and used MongoDB indexes on category and date fields to speed up data retrieval.

6. Email Subscription Data Integrity

- **Challenge:** Avoiding duplicate subscriptions and ensuring valid email formats was critical to maintain clean data.
- **Solution:** Applied server-side checks for duplicates and implemented regex-based email validation before inserting into the database.

7. Postman API Testing

- **Challenge:** Ensuring all endpoints handled edge cases, like missing fields or invalid requests.
- **Solution:** Thoroughly tested all CRUD APIs using Postman collections with different test cases to confirm expected behavior and error handling.

5.4 CODE QUALITY AND BEST PRACTICES FOLLOWED

Maintaining clean, modular, and well-structured code was a key focus during the development of the *Blogger – A Blog App*. The goal was to ensure the application is scalable, readable, and easy to maintain in the long term.

1. Folder Structure and Modularity

- Followed a clear and logical structure separating concerns such as pages/, components/, utils/, and api/.
- Common logic and utility functions (e.g., database connections, API response formatters) were abstracted for reusability.
- Blog operations and subscription logic were modularized into separate API routes for better maintainability.

2. Code Standards

- Applied DRY (Don't Repeat Yourself) and KISS (Keep It Simple, Stupid) principles throughout the codebase.
- Used consistent naming conventions: PascalCase for components, camelCase for functions and variables.
- Ensured separation of concerns by keeping business logic out of UI components and within API handlers or helper functions.

3. Version Control

- Managed project using Git and GitHub with a feature-branch workflow.
- Commits were descriptive and consistent to track progress and rollback if needed.
- A .gitignore file was configured to exclude unnecessary or sensitive files like node_modules/, .env, and .next/.

4. Error Handling and Feedback

- Implemented basic error handling for all API routes to return standardized JSON responses.
- Frontend included client-side checks and toast notifications to inform users of success or failure (e.g., for blog creation or email subscription).
- Used try-catch blocks and status codes effectively across API functions.

5. Security and Data Validation

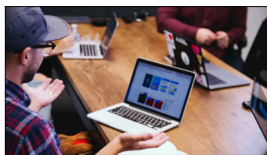
- Basic input validation and sanitization were applied on both frontend and backend to prevent malformed data and basic injection attacks.
- Checked for duplicate email entries in subscription to ensure data integrity.
- Applied HTTP method validation to secure API endpoints.

5.5 Result

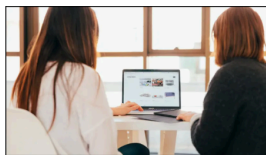

[Get Started →](#)

LATEST BLOGS

A dynamic space where ideas ignite, knowledge flows, and transformative insights across technology, lifestyle, personal growth, innovation, and entrepreneurship empower you to shape a smarter, more meaningful future—one blog at a time

[All](#)
[Technology](#)
[Startup](#)
[Lifestyle](#)

Lifestyle

A detailed step by step guide to manage your lifestyle


Startup

How to create an effective startup roadmap or ideas


Technology

Learning new technology to boost your career in software


Technology

Tips for getting the most out of apps and software

Figure 4 : Website

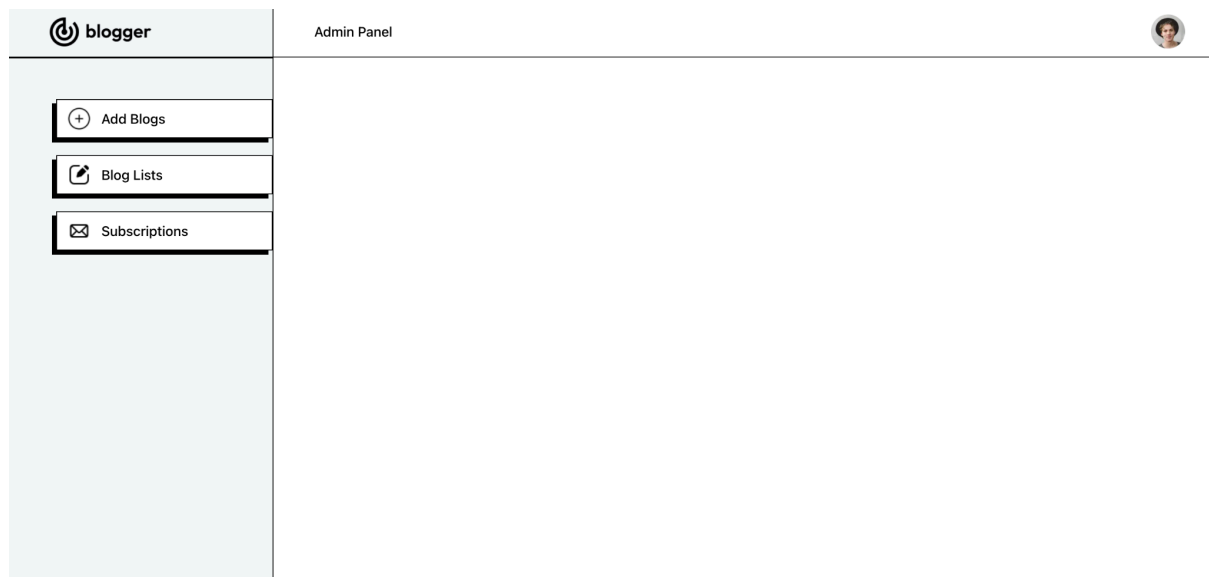


Figure 5 : Admin Dashboard

CHAPTER 6

TESTING

6.1 TESTING STRATEGY

The testing strategy for the *Blogger – A Blog App* focused on ensuring reliability, correctness, and a smooth user experience through a structured approach:

1. Unit Testing

- Tested individual React components such as blog cards, category filters, and form inputs in isolation.
- Utility functions and API handlers were verified for expected outputs with various inputs.

2. Integration Testing

- API endpoints for blog creation, deletion, and email subscriptions were tested using Postman with valid and invalid data.
- Verified seamless interaction between frontend components and backend APIs.
- Database operations were validated to ensure correct data storage and retrieval from MongoDB.

3. System Testing

- Tested complete blog submission and subscription flows from the user interface to the backend and database.
- Verified navigation from homepage to 'Get Started' page and its three modules (Add Blog, Blog List, Subscription).
- Checked compatibility across major browsers (Chrome, Firefox, Edge).
- Mobile responsiveness tested using browser dev tools and multiple device resolutions.

4. Performance Testing

- Simulated multiple concurrent blog creation and deletion operations to check response time and app stability.
- MongoDB query performance reviewed and optimized using indexes where applicable.


5. Security Testing

- Validated API routes with appropriate HTTP method restrictions (GET, POST, DELETE).
- Checked input sanitization for blog content and email fields to prevent injection attacks.
- Verified safe handling of duplicate subscriptions.


6. User Acceptance Testing


- Conducted testing with users unfamiliar with the system to validate usability and intuitive navigation.
- Gathered feedback on design, functionality, and responsiveness.
- Refined UI and flow based on real-world feedback to ensure a user-friendly blogging experience.

6.2 TEST RESULTS AND ANALYSIS




+ Add Blogs

 Blog Lists

 Subscriptions

Admin Panel



All Blogs









AUTHOR NAME	BLOG TITLE	DATE	ACTION
 Alex Bennett	A detailed step by step guide to manage your lifestyle	Wed Apr 23 2025	x
 Alex Bennett	How to create an effective startup roadmap or ideas	Wed Apr 23 2025	x
 Alex Bennett	Learning new technology to boost your career in software	Wed Apr 23 2025	x
 Alex Bennett	Tips for getting the most out of apps and software	Wed Apr 23 2025	x
 Alex Bennett	Enhancing your skills and capturing memorable moments	Wed Apr 23 2025	x
 Alex Bennett	Maximizing returns by minimizing resources in your startup	Wed Apr 23 2025	x
 Alex Bennett	Technology for Career advancement in development	Wed Apr 23 2025	x
 Alex Bennett	A comprehensive roadmap for effective lifestyle management	Wed Apr 23 2025	x

Figure 6: Blogs Created

CHAPTER 7

CONCLUSION AND DISCUSSION

7.1 OVERALL ANALYSIS OF INTERNSHIP/PROJECT VIABILITIES

The *Blogger – A Blog App* project developed during this internship has demonstrated solid technical execution and practical usability for modern content-sharing needs:

1. Technical Viability:

- Utilization of **Next.js** and **MongoDB** provided a fast, scalable, and modern development stack suitable for dynamic blog applications.
- Component-based architecture ensured maintainable and modular code, allowing seamless addition of new features.
- Functional integration between frontend forms, backend APIs, and MongoDB storage offered a reliable content management flow.
- Implementation of email subscription and category-based blog filtering enhanced the user experience and application depth.

2. Commercial Viability:

- The application addresses the core requirement of content creation, publishing, and distribution in a minimal and user-friendly format.
- The blog category filtering and subscription module provide engagement and retention value, making it market-viable for small-scale creators or blogging communities.
- Admin-level blog deletion functionality adds control over content management and moderation.
- Simple and intuitive UI ensures adoption by non-technical users, making it ideal for bloggers, writers, or educational institutes.

3. Learning Outcomes:

- Acquired practical experience in full-stack development using **Next.js** and **MongoDB**.
- Gained hands-on skills in API development, form handling, and integration testing using **Postman**.
- Strengthened understanding of asynchronous operations, database interactions, and UI responsiveness.

- Learned essential version control practices through **Git and GitHub**, including branch management and collaboration.
- Gained insight into user-centric design and basic subscription mechanisms that enhance platform usability.

7.2 INTERNSHIP PROGRESS REVIEW MEETING WITH INDUSTRY GUIDE

Regular progress review meetings were conducted with Mr. Satyam Raval, Deputy General Manager at Tech Elecon:

1. Initial Planning Meeting - February 10, 2025 (On-site)

- Project requirements and scope finalization
- Technology stack selection
- Timeline and milestone planning

2. Final Review - April 28, 2025 (On-site)

- Complete system demonstration
- Performance review
- Documentation verification

7.3 PROBLEM ENCOUNTERED AND POSSIBLE SOLUTIONS

During the development of the *Blogger – A Blog App* project, several technical and functional challenges were encountered. Each issue was addressed with practical and effective solutions to ensure a smooth and robust user experience.

1. Challenge: Dynamic Blog Filtering by Category

- **Problem:** Initially, filtering blogs based on selected categories did not update the view dynamically, causing inconsistencies in blog listings.
- **Solution:** Implemented state management and conditional rendering using Next.js hooks to ensure real-time filtering and smooth UI updates.

2. Challenge: Email Subscription Handling

- **Problem:** Handling duplicate email subscriptions and validating email formats required additional logic.

- **Solution:** Added backend validation and a check to prevent duplicate entries in the MongoDB database, along with client-side alerts for feedback.

3. Challenge: Blog Deletion and State Sync

- **Problem:** After deleting a blog post, the UI did not reflect the updated blog list until the page was manually refreshed.
- **Solution:** Used asynchronous state updates and re-fetching logic after successful deletion to instantly reflect changes in the blog list.

4. Challenge: Responsive Design Across Devices

- **Problem:** Some layout components overlapped or misaligned on smaller screen sizes.
- **Solution:** Applied custom CSS with media queries and flexbox/grid layouts to ensure responsiveness across all devices.

7.4 SUMMARY OF INTERNSHIP

The internship provided an excellent opportunity to design and develop *Blogger – A Blog App* using modern web development technologies like Next.js and MongoDB. The project covered all stages of the software development life cycle, from conceptualization to development, testing, and deployment.

Key accomplishments include:

- Development of a clean and responsive homepage displaying blogs by category
- Implementation of a dynamic blog creation system with delete functionality
- Integration of an email subscription module to grow user engagement
- Design of a user-friendly navigation flow through a “Get Started” interface
- Backend development for data storage, routing, and API management
- Robust testing using Postman and browser-based validations for reliability

The blog app effectively serves as a basic content management system, providing an organized platform to publish, manage, and subscribe to blogs. It demonstrates practical implementation of full-stack skills, efficient state management, and modern UI design.

7.5 LIMITATION AND FUTURE ENHANCEMENT

Current Limitations:

- No authentication system; any user can add or delete blogs without restriction
- Limited backend validation and error handling for form submissions
- Absence of a rich text editor for blog formatting
- No support for comments or user interactions on blog posts
- Basic UI design without role-based dashboard segmentation
- Email subscription lacks confirmation or double opt-in mechanism

Future Enhancements:

1. Authentication & Role Management:

- Implement secure user authentication using JWT
- Add role-based access (e.g., admin, editor, reader)

2. Rich Text Editor & Blog Enhancement:

- Integrate Markdown or WYSIWYG editor for better content formatting
- Add support for media (images/videos) within blog posts

3. Commenting and Engagement:

- Enable comment sections for reader interaction
- Implement like/share functionality and blog view counters

4. Advanced Subscription System:

- Add confirmation emails for subscriptions
- Integrate with platforms like Mailchimp for campaign management

5. Mobile Optimization:

- Build mobile-responsive UI and consider future mobile app development
- Add offline read functionality using service workers

REFERENCES

1. React.js. (n.d.). *React Documentation*. Retrieved from <https://reactjs.org/docs/getting-started.html>
2. MongoDB Inc. (n.d.). *MongoDB Documentation*. Retrieved from <https://www.mongodb.com/docs/>
3. Express.js. (n.d.). *Express.js Documentation*. Retrieved from <https://expressjs.com/>
4. GitHub. (n.d.). *Git Hub Documentation*. Retrieved from <https://docs.github.com/>
5. Postman. (n.d.). *Postman API Platform Documentation*. Retrieved from <https://learning.postman.com/docs/>
6. Material UI. (n.d.). *Material UI Documentation*. Retrieved from <https://mui.com/>
7. Next.js. (n.d.). *Next.js Documentation*. Retrieved from <https://nextjs.org/docs>
8. Nodemailer. (n.d.). *Nodemailer Documentation*. Retrieved from <https://nodemailer.com/about/>
9. React Icons. (n.d.). *React Icons Documentation*. Retrieved from <https://react-icons.github.io/react-icons/>
10. MongoDB Compass. (n.d.). *GUI for MongoDB*. Retrieved from <https://www.mongodb.com/products/compass>
11. O'Reilly Media. Richards, M. (n.d.). *Software Architecture Patterns*.