



**ADMAS UNIVERSITY**  
**Department of Computer Science**

**Final Year Project**

Collaborative Blogging Platform for the Admas University Society

Submitted to the Department of Computer Science, Admas University, in Partial  
Fulfillment of the Requirements for the Degree of Bachelor of Science in  
Computer Science

**Prepared By:**

<b>No</b>	<b>Name</b>	<b>ID No</b>
1)	Abraham Gebeyehu	5164/14
2)	Fethawi Negassi	5356/14
3)	Kidist Wondesen	5002/14
4)	Yared Mealku	5020/14

**Project Advisor:** Mr. Biruk Mengiste (MSC.)  
Admas University, Meskel Campus,  
Addis Ababa, Ethiopia March 2025

# APPROVAL SHEET

This is to formally acknowledge that the undergraduate project report titled:

***“Collaborative Blogging Platform for the Admas University Society”***

has been submitted to the **Department of Computer Science, Admas University**, as part of the partial fulfillment of the requirements for the degree of **Bachelor of Science in Computer Science**.

The project work, jointly undertaken by the following students:

- Abraham Gebeyehu
- Fithawi Negassi
- Kidist Wondesen
- Yared Mealku

Has been reviewed and evaluated by the designated academic committee. Based on the quality and completeness of the work, the undersigned hereby approve the project report for final submission and documentation.

## Validated by the Project Examination Committee

Name & Designation	Signature	Date
<i>Advisor</i>	_____	_____
<i>Department Head</i>	_____	_____
<i>Internal Examiner 1</i>	_____	_____
<i>Internal Examiner 2</i>	_____	_____
<i>Internal Examiner 3</i>	_____	_____

Official Department Stamp/Seal (if applicable):

## **ACKNOWLEDGMENT**

We would like to express our heartfelt gratitude to our project advisor, **Mr. Biruk Mengiste**, for his unwavering support, insightful guidance, and constructive feedback throughout every stage of this project. His mentorship played a critical role in shaping our ideas and ensuring the successful completion of this work.

We are also deeply thankful to the **faculty and staff of the Department of Computer Science at Admas University** for equipping us with the academic foundation, technical skills, and encouragement necessary to bring this project to life. Their dedication to excellence and student development has been a constant source of inspiration.

Our sincere appreciation extends to the wider **Admas University community**, whose needs and aspirations inspired the development of this **Collaborative Blogging Platform**. This platform was envisioned as a space to foster academic engagement, enhance knowledge sharing, and strengthen communication across students and faculty.

Finally, we are immensely grateful to our **families and friends** for their patience, motivation, and continuous emotional support throughout the journey. Their belief in us made this achievement possible.

# Contents

LIST OF FIGURES.....	V
LIST OF TABLE.....	VI
LIST OF ABBREVIATIONS .....	VII
ABSTRACT.....	VIII
CHAPTER ONE: INTRODUCTION.....	1
1 INTRODUCTION.....	1
1.1 BACKGROUND INFORMATION OF THE ORGANIZATION.....	1
<i>1.1.1 Vision.....</i>	2
<i>1.1.2 Mission The Missions of Admas University are:.....</i>	2
1.2 BACKGROUND OF THE PROJECT .....	2
1.3 TEAM COMPOSITION .....	3
1.4 STATEMENT OF THE PROBLEM .....	3
1.5 OBJECTIVE OF THE PROJECT.....	4
<i>1.5.1 General Objective.....</i>	4
<i>1.5.2 Specific Objectives.....</i>	4
1.6 FEASIBILITY ANALYSIS.....	5
<i>1.6.1 Operational Feasibility.....</i>	5
<i>1.6.2 Technical Feasibility.....</i>	6
<i>1.6.3 Economic Feasibility.....</i>	6
<i>1.6.4 Behavioral/Political Feasibility.....</i>	6
<i>1.6.5 Schedule Feasibility.....</i>	7
<i>1.6.6 Cost-Benefit Analysis.....</i>	7
1.7 SCOPE AND SIGNIFICANCE OF THE PROJECT .....	8
1.8 TARGET BENEFICIARIES OF THE SYSTEM .....	9
1.9 METHODOLOGY FOR THE PROJECT.....	9
<i>1.9.1 Data Source.....</i>	9
<i>1.9.2 Fact-Finding Techniques.....</i>	10
<i>1.9.3 Systems Analysis and Design.....</i>	10
<i>1.9.4 Development Tools.....</i>	12
<i>1.9.5 Testing Procedures.....</i>	13
<i>1.9.6 Implementation (Parallel/Partial/Direct).....</i>	13
<i>1.9.7 Limitation of the Project.....</i>	14
<i>1.9.8 Risks, Assumptions, and Constraints.....</i>	14
1.10 CONCLUSION.....	15
CHAPTER 2: DESCRIPTION OF THE EXISTING SYSTEM .....	16
2.1 INTRODUCTION.....	16
2.2. EXISTING SYSTEM STUDY .....	16
<i>2.2.1 Overview of the Current System.....</i>	16
<i>2.2.2 System Workflow.....</i>	17
<i>2.2.3 Hardware and Software Used.....</i>	18
<i>2.2.4 Inputs and Outputs.....</i>	19
<i>2.2.5 Users and Roles.....</i>	19
<i>2.2.6 Strengths of the Existing System.....</i>	19
<i>2.2.7 Weaknesses/Limitations of the Existing System.....</i>	20
<i>2.2.8 Security and Control Measures.....</i>	20
<i>2.2.9 Performance Analysis.....</i>	20
<i>2.2.10 Cost Analysis.....</i>	21
2.3. STATE OF THE ART .....	21

2.3.1 Technical Aspects of Blogging Platforms.....	21
Study [1]: Kaushal & Dwivedi (2022).....	21
Study [2]: Ankith Lagupudi and Venkata Satya Koppula (2020-2021).....	24
2.3.2 Blogging as a Co-Creation Model in Education.....	27
Study [3]: Medero et al. (2022).....	27
2.3.3 The Role of Blogging in Enhancing Writing and Peer Feedback.....	29
Study [4]: Pham and Nguyen (2020).....	29
2.3.4 Summary Table of the State-of-the-Art Studies.....	30
2.4 CONCLUSION.....	31
<b>CHAPTER 3: SYSTEM ANALYSIS AND DESIGN.....</b>	<b>32</b>
3.1 INTRODUCTION.....	32
3.2 PROBLEM IDENTIFICATION.....	32
3.3 REQUIREMENT GATHERING AND ANALYSIS.....	33
3.3.1 System Overview.....	33
3.3.2 Key Users.....	33
3.3.3 Functional Requirements.....	34
3.3.4 Non-Functional Requirements.....	35
3.4 SYSTEM DESIGN.....	35
3.4.1 High-Level Design (HLD).....	35
3.4.1.1 System Architecture Diagram.....	35
3.4.1.2 Component Diagram.....	37
3.4.1.3 Deployment Diagram.....	40
3.4.1.4 Data Flow Diagram (Level 2).....	43
3.4.1.5 Use Case Diagram.....	46
3.4.2 Low-Level Design (LLD).....	51
3.4.2.1 Class Diagram.....	52
3.4.2.2 Sequence Diagrams.....	55
3.4.2.3 Activity Diagram.....	58
3.4.2.4 State Machine Diagram.....	59
3.4.2.5. Entity-Relationship (ER) Diagram.....	64
3.5 CONCLUSION.....	69
<b>CHAPTER FOUR: TESTING PLAN.....</b>	<b>70</b>
4.1 INTRODUCTION.....	70
4.2 FINAL TESTING OF THE SYSTEM.....	70
4.3 SYSTEM TEST.....	71
4.3.1 Unit Testing.....	71
4.3.2 Functional Testing.....	72
4.3.3 Performance Testing.....	73
4.3.4 Security Testing.....	73
4.3.5 Integration Testing.....	74
4.3.6 Usability Testing.....	75
4.3.7 Compatibility Testing.....	75
4.3.8 Acceptance Testing.....	76
4.3.9 Recovery Testing.....	76
4.4 HARDWARE AND SOFTWARE ACQUISITIONS.....	76
4.5 USER MANUAL PREPARATION.....	77
4.6 TRAINING .....	77
4.7 INSTALLATION PROCESS.....	78
4.8 START-UP STRATEGY .....	78
4.9 CONCLUSION.....	79
<b>CHAPTER FIVE: CONCLUSIONS AND RECOMMENDATIONS .....</b>	<b>80</b>
5.1 CONCLUSIONS .....	80
5.2 RECOMMENDATIONS .....	81
<b>REFERENCES.....</b>	<b>81</b>

## **List of Figures**

Figure 2.1: Communication System at Admas University.....	18
Figure 3.1: System Architecture Diagram.....	39
Figure 3.2: Component Diagram.....	42
Figure 3.3: Deployment Diagram.....	45
Figure 3.4: Data Flow Diagram (Content Management).....	47
Figure 3.5: Use Case Diagram.....	50
Figure 3.6: Class Diagram.....	57
Figure 3.7: Sequence Diagram - Create Blog Post.....	59
Figure 3.8: Activity Diagram - Create and Publish Blog Post.....	61
Figure 3.9: State Machine Diagram - User Lifecycle.....	64
Figure 3.10: State Machine Diagram - Blog Post Workflow.....	66
Figure 3.11: Mongo DB Entity-Relationship (ER) Diagram.....	70

## **List of Table**

Table 1.1: Team Composition.....	3
Table 1.2: Cost Break down.....	8
Table 2.1: Summary of the State-of- the-Art .....	31
Table 3.1: Use Case – Login(Privileged Account).....	49
Table 3.2: Use Case – Register & Login (General Users).....	49
Table 3.3: Use Case - Create / Edit / Delete Own Blog Posts.....	50
Table 3.4: Use Case– Categorize & Tag Blog Posts.....	50
Table 3.5: Use Case – Comment on Posts.....	51
Table 3.6: Use Case – Manage Own Profile.....	51
Table 4.1: Unit Test Scenarios.....	71
Table 4.2: Functional Test Scenario.....	72
Table 4.3: Integration Test Scenarios.....	74

## List of Abbreviations

- API** – Application Programming Interface
- CBP** – Collaborative Blogging Platform
- CI/CD** – Continuous Integration / Continuous Deployment
- CRUD** – Create, Read, Update, Delete
- CSS** – Cascading Style Sheets
- DFD** – Data Flow Diagram
- ER** – Entity-Relationship (Diagram)
- HTML** – HyperText Markup Language
- HTTPS** – Hypertext Transfer Protocol Secure
- IDE** – Integrated Development Environment
- JWT** – JSON Web Token
- LMS** – Learning Management System
- MERN** – Mongo DB, Express.js, React.js, Node.js
- ODM** – Object Data Modeling
- RBAC** – Role-Based Access Control
- SEO** – Search Engine Optimization
- SHA-256** – Secure Hash Algorithm 256-bit
- TLS** – Transport Layer Security
- UAT** – User Acceptance Testing
- UI** – User Interface
- UX** – User Experience
- VCS** – Version Control System

## Abstract

This project proposes the development of a **Collaborative Blogging Platform (CBP)** to address the fragmented communication landscape and limited academic engagement currently experienced at Admas University. The platform is designed to facilitate the creation, sharing, and collaborative editing of academic content by students and faculty. It incorporates essential features such as **secure role-based access control**, **real-time collaborative editing**, **AI-powered content suggestions**, **multimedia integration**, and **content moderation workflows**.

Developed using the **MERN stack** (MongoDB, Express.js, React.js, Node.js), the system follows a structured **Software Development Life Cycle (SDLC)** methodology. Key phases include stakeholder engagement, system modeling using **UML diagrams** (including ER, sequence, and state diagrams), and automated deployment through **CI/CD pipelines**. Emphasis is placed on **scalability**, **security**, and **mobile responsiveness** to ensure the platform meets institutional requirements.

Preliminary testing through **partial implementation** and **user acceptance testing** demonstrates the platform's effectiveness and readiness for broader deployment. Ultimately, the CBP contributes to enhanced knowledge exchange, strengthened academic collaboration, and the realization of Admas University's mission to deliver technology-supported, quality education.

# **Chapter One: Introduction**

## **1 Introduction**

The **Collaborative Blogging Platform** is an innovative project that will be designed to improve communication, foster knowledge exchange, and strengthen collaboration within Admas University. Initiated by computer science students, this platform aims to address the absence of a unified digital space where students and faculty can engage in meaningful academic dialogue, share research, and exchange ideas. The system will empower users to seamlessly create, share, and engage with blog content covering a wide range of academic and professional subjects. Equipped with essential features such as secure user authentication, intuitive content management, and real-time interaction, the platform will promote a dynamic and inclusive learning environment that encourages participation and intellectual growth.

This proposal outlines the background and significance of the project, highlights its core objectives, identifies the challenges it intends to overcome, and presents the technologies that will be utilized for its development and deployment.

### **1.1 Background Information of the Organization**

Admas University commenced its operation in October 1998 under the name “Admas Business Training Centre.” Then Training Centre then started delivering training services in certain tailor made, six-month, and short-term programmes. By undertaking deep assessments of further training needs and making preparations in terms of the required human and material resources, the center upgraded itself to a college status as of April 1999. After deep objective assessments of further training needs and making preparations in accordance with the requirements of the Ministry of Education, the College was upgraded to the status of a University College as of March 2007.

Admas University has become a fully-fledged university as of July 2014 (Hamlie 2006 E.C.) With the objective of expanding its quality services, the University opened ten (10) colleges/campuses/faculties so far. Eight of these campuses are found in Ethiopia among which six are in Addis Ababa while the rest two are in Bishoftu and Mekelle towns. The other two campuses are located out of Ethiopia in the Capitals of Somaliland and Puntland (i.e. Hargeissa and Garowe, respectively). Apart from the regular mode, the University also

has a Distance Education College with more than 50 (fifty) coordination offices throughout the country.

Aligned with this vision, the proposed Collaborative Blogging Platform for the Admas University Society aims to serve as a dynamic space for students and faculty to share academic insights, research findings, and professional experiences. This initiative reinforces the university's commitment to fostering intellectual discourse, improving digital literacy, and strengthening the academic community through interactive and accessible knowledge-sharing mechanisms.

### *1.1.1 Vision*

Admas University envisions to become “one of the leading private higher education institutions in terms of Technology supported Quality Education and Training, Research, Community Engagement in East Africa by 2035”.

### *1.1.2 Mission*

*The Missions of Admas University are:*

- Provide technology supported quality higher education at all levels for affordable price through regular, continuing, and distance education modes so as to produce competent professionals who can support the development endeavor of the country.
- Conduct quality and outcome-based training to produce middle-level human resources and supply to the industry.
- Undertake research that helps to solve the economic and social problems of the country and that can also add new human and material values for the society.
- Produce competent entrepreneurs who could contribute to the technology transfer endeavors of the country.
- Render consultancy and short-term training services to businesses, government, and non-government organizations to help them accomplish their objectives competently.
- Engage in University-Industry linkages and community services to fulfil social responsibilities expected of it as an academic institution.

## **1.2 Background of the Project**

In today's digital age, universities require modern and interactive platforms to facilitate academic discussions, knowledge exchange, and research collaboration. However, Admas

University currently lacks a centralized digital space for students and faculty to engage in meaningful discourse, leading to fragmented communication, restricted access to research, and missed opportunities for intellectual engagement.

The **Collaborative Blogging Platform for the Admas University Society** seeks to address this gap by providing an intuitive and structured system for users to create, edit, and share blog posts on diverse academic topics. By leveraging digital technology, this platform will strengthen student-faculty interaction, encourage intellectual exchange, and foster a more connected academic community.

Aligned with Admas University's mission to promote innovation and knowledge dissemination, this project will empower students and faculty to engage in interdisciplinary discussions, publish insights, and collaborate on research efficiently. Through a user-friendly interface and well-organized content management, the platform will serve as a vital resource for academic growth, networking, and professional development.

### 1.3 Team Composition

S.No.	Name	ID No.	Email /Mobile	Responsibility
1	Abraham Gebeyehu	[5164/14]	<a href="mailto:gebeyeahuabraham19@gmail.com">[gebeyeahuabraham19@gmail.com]</a> /+251984876850]	Project Manager & Lead Backend Developer
2	Fithawi Negassi	[5356/14]	<a href="mailto:fithawi06@gmail.com">[fithawi06@gmail.com]</a> /+251 986115631]	Lead Frontend Developer
3	Kidist Wondosen	[5002/14]	<a href="mailto:Veronica.wondesen@gmail.com">[Veronica.wondesen@gmail.com]</a> /+251985388881]	Lead Tester & Assistant Frontend Support
4	Yared Melaku	[5020/14]	<a href="mailto:yaredmelaku511@gmail.com">[yaredmelaku511@gmail.com]</a> /+251973128635]	Bug Tracker & Testing Support

*Table 1.1: Team Composition*

### 1.4 Statement of the Problem

Admas University currently faces several challenges due to the absence of a centralized academic platform, including:

- **Lack of a unified digital space** where students and faculty can engage in academic discussions and share research effectively.
- **Fragmented communication** between students and faculty, leading to delays, misunderstandings, and reduced collaboration.
- **Limited access to scholarly content and research outputs** within the university community, restricting knowledge sharing and academic growth.
- **Missed opportunities for interdisciplinary collaboration**, hindering broader intellectual engagement across departments.
- **Reliance on inefficient and informal communication channels** (e.g., social media or messaging apps) that do not support structured academic interaction or knowledge dissemination.

To solve these issues, the proposed project introduces the **Collaborative Blogging Platform (CBP)**, which aims to:

- **Enhance academic communication and collaboration** across the university.
- **Facilitate easy sharing and access to research and scholarly content** among students and faculty.
- **Foster a connected and interactive academic community** by providing a centralized digital environment tailored for academic exchange at Admas University.

## **1.5 Objective of the Project**

### *1.5.1 General Objective*

The objective of this project is to develop a user-friendly Collaborative Blogging Platform (CBP) for Admas University, enabling students and faculty to create, share, and engage with academic content, fostering collaboration, research dissemination, and knowledge exchange.

### *1.5.2 Specific Objectives*

- Design a user-friendly web interface that allows users to easily create, edit, and interact with blog content.
- Implement a role-based access control system to manage permissions for admins, moderator, authors, and readers.
- Enable real-time collaborative editing so multiple users can work on blog posts simultaneously.

- Support the inclusion of multimedia content such as images, video documents within blog posts.
- Integrate AI-powered assistance for grammar correction, content improvement, and topic suggestions during writing.
- Ensure responsive design and cross-platform compatibility for seamless use on mobile, tablet, and desktop devices.
- Provide efficient content discovery tools including advanced search, filters, and category-based navigation.
- Apply advanced security features like data encryption, input validation, and user activity monitoring.
- Implement a peer review and feedback system to allow collaborative evaluation of blog content before publication.

## 1.6 Feasibility Analysis

We examined the feasibility of our proposed digital platform by evaluating it from multiple perspectives. Feasibility analysis is a critical step in assessing whether the project was viable and achievable. We explored the technical, operational, economic, political, and schedule feasibility of the platform to ensure its success at Admas University. By considering these different angles, we aimed to determine whether the project was realistic and worth pursuing.

### *1.6.1 Operational Feasibility*

The platform aligns with Admas University's goal of promoting digital engagement. Students and faculty members are expected to embrace the system due to its user-friendly design and academic benefits. By adopting the Agile Software Development Model, the system benefits from continuous interaction between developers, users, and stakeholders, allowing iterative improvements based on real user feedback.

This active involvement ensures that users understand its functionality from the outset, reducing usability concerns and enhancing adoption. Unlike the university's Learning Management System (LMS), which primarily focuses on course management, this platform facilitates interactive knowledge-sharing and discussion among users. The system is designed for cross-browser compatibility, ensuring seamless accessibility across devices. Given these factors, the proposed system is operationally feasible, offering a practical, efficient, and scalable solution for enhancing academic collaboration.

### *1.6.2 Technical Feasibility*

The Collaborative Blogging Platform is designed as a user-friendly and efficient web-based system aimed at enhancing digital engagement within Admas University. The system is developed with consideration for the university's existing IT infrastructure, ensuring smooth integration and usability upon implementation.

The software development tools selected for the project are freely available and open source, minimizing costs and maximizing accessibility. The platform is intended to run on the current hardware available at the university, with only minor enhancements required for optimal performance. Additionally, the development team possesses the necessary expertise and experience to successfully implement and maintain the system. Based on an analysis of the operational requirements and user needs, the proposed system is deemed technically feasible for adoption by students and faculty in the upcoming semester.

### *1.6.3 Economic Feasibility*

The proposed system is economically feasible as it is designed to optimize resource usage and deliver an efficient, cost-effective solution for the university. By utilizing open-source technologies and the existing IT infrastructure, the platform minimizes both initial development and ongoing maintenance costs. Furthermore, it eliminates the need for expensive proprietary software, resulting in significant cost savings.

The automated, web-based nature of the platform improves operational efficiency, allowing the university to allocate resources more effectively. With an emphasis on long-term sustainability and scalability, the proposed system offers a cost-effective solution that maximizes value while maintaining flexibility for future enhancements.

### *1.6.4 Behavioral/Political Feasibility*

The proposed system is behaviorally and politically feasible, as it is designed to have a positive impact on both the users and the organization. The platform does not pose any harm to the environment or society and adheres to all relevant legal and regulatory requirements. It is developed with a user-friendly interface, ensuring it meets the needs and expectations of the users, while enhancing the overall working environment.

Furthermore, the system is designed to align with the goals of the organization, fostering a smooth adoption process. As a result, the platform is free from any significant political or

environmental challenges, ensuring a harmonious and seamless integration within the university.

### *1.6.5 Schedule Feasibility*

The Schedule Feasibility of the project is carefully planned to span 4 to 6 months, ensuring the development of a robust and feature-rich Collaborative Blogging Platform (CBP). The timeline is structured to include all critical phases: research, design, development, testing, deployment, and training. To facilitate effective project tracking and ensure timely completion, a Gantt chart and milestone breakdown will be used to monitor progress and adjust as needed. This well-organized schedule ensures that all tasks are completed efficiently and within the projected time-frame.

### *1.6.6 Cost-Benefit Analysis*

The Collaborative Blogging Platform (CBP) for Admas University Society was developed leveraging personal and university resources, resulting in minimal direct financial outlays for development and testing phases. The major expenses are related to internet usage for extensive research and communication, as well as miscellaneous minor system-related costs.

#### **Cost of the Project**

<b>Activity</b>	<b>Estimated Cost (ETB)</b>	<b>Description</b>
<b>Development</b>	Free	Conducted on personal computers using open-source tools with no additional cost.
<b>Testing</b>	Free	Performed on personal and university devices without extra expenses.
<b>Internet/Data Usage</b>	5,000	Covers extensive online research, paper writing, system analysis, and day-to-day communication.
<b>Miscellaneous</b>	2,000	Minor expenses related to system resources and unforeseen technical needs.
<b>Total Estimated Cost</b>	<b>7,000 ETB</b>	The overall estimated cost of the project.

*Table 1.2: Cost Break Down*

The development of the **Collaborative Blogging Platform (CBP)** for Admas University is a low-cost yet high-value project, driven primarily by student developers using open-source technologies.

## 1.7 Scope and Significance of the Project

### Scope of the Project

The **Collaborative Blogging Platform (CBP)** for Admas University will provide an interactive space for students, faculty, and alumni to share academic content, collaborate on research, and engage in discussions. The platform will feature a well-structured system that ensures ease of use and accessibility across various devices.

- Supports blogging, collaborative editing, and multimedia uploads.
- Implements role-based access control for Admins, Editors, Writers, and Readers.
- Provides AI-powered suggestions for grammar checks and topic recommendations.
- Ensures accessibility via both web and mobile platforms.
- Offers a seamless and intuitive user experience with modern UI/UX design.

### Significance of the Project

The project plays a vital role in enhancing digital academic engagement at Admas University. It provides a structured space where users can publish and discuss educational content, strengthening institutional collaboration and innovation.

- Facilitates knowledge sharing and research dissemination among students and faculty.
- Creates an academic-focused blogging environment that encourages meaningful discussions.
- Promotes student-faculty engagement through interactive features such as comments and feedback.
- Enhances digital literacy and innovation within the university community.
- Strengthens the university's online presence by showcasing research and academic insights.

## 1.8 Target Beneficiaries of the System

- **Students:** The platform provides students with a space to express ideas, share academic insights, publish creative content, and engage in university-wide discussions. It fosters digital literacy, writing skills, and collaboration across departments.
- **Faculty Members:** Instructors can use the platform to share academic content, departmental updates, or thought leadership pieces, and to engage with students outside the classroom in a more informal, interactive setting.
- **University Clubs and Societies:** Student-led groups can publish event updates, articles, and campaigns, increasing visibility and engagement among the student body.
- **University Administration:** The platform offers a communication channel to announce initiatives, highlight achievements, and gather feedback from the campus community in a centralized, accessible format.
- **Prospective Students and External Visitors:** Publicly available content on the platform can showcase the intellectual and cultural vibrancy of Admas University, potentially attract new students and foster a positive public image.
- **Researchers and Alumni (optional future extension):** The system can evolve into a resource hub where alumni and researchers stay informed, contribute expert insights, or collaborate on academic content with current students.

## 1.9 Methodology for the Project

### 1.9.1 Data Source

The data used for the design and development of the Collaborative Blogging Platform (CBP) will be gathered from the following sources:

- **Interviews:** Conducting one-on-one or group interviews with faculty and students to gather detailed insights into their preferences and identify any gaps in the current communication system.
- **Practical Observation:** Observing existing communication and collaboration practices at Admas University to understand how information is shared and what improvements can be made.

- **Document Analysis:** Reviewing university policies on digital collaboration, communication standards, and content moderation to ensure the platform complies with institutional guidelines.
- **Analysis of Similar Platforms:** Examining other academic platforms at similar institutions to understand best practices and lessons learned in creating successful digital collaboration tools.
- **University Guidelines on Content Moderation:** Studying university guidelines for moderating content to create a platform that ensures academic integrity and appropriate discussions.

### *1.9.2 Fact-Finding Techniques*

To effectively gather and validate the necessary data for platform development, the following fact-finding techniques will be applied:

- **Structured and Semi-Structured Interviews:** Used to collect detailed, targeted feedback from students and faculty about the challenges in current academic communication and their feature expectations from the CBP.
- **Direct Observation:** Involves monitoring how students and staff currently collaborate through tools like emails, group chats, or notice boards to identify practical inefficiencies and opportunities for improvement.
- **Document Review:** Examining university policies, IT guidelines, and academic integrity regulations to ensure all platform features are compliant and secure.

### *1.9.3 Systems Analysis and Design*

This section provides a concise overview of the key phases and approaches adopted for the development of the Collaborative Blogging Platform. It outlines the methodologies for understanding requirements, the core principles guiding the system's design, and the technological stack chosen to bring the platform to fruition. More in-depth details and specific diagrams will be presented in subsequent chapters.

#### **Requirement Analysis**

To ensure the Collaborative Blogging Platform effectively addresses the diverse needs of its target users within Admas University Society, a thorough requirement analysis phase was

undertaken. This involved systematic approaches to gather, analyze, and document both functional and non-functional requirements.

- **Stakeholder Identification:** A critical first step involved identifying key stakeholders, including students, faculty members, and university administrators. Each group presents unique perspectives and requirements that are crucial for the platform's success.
- **Use Case Analysis:** Use case analysis was employed to model and visualize the interactions between various user roles and the system. This provides a clear understanding of system functionalities from an external perspective, covering core processes such as creating and managing blog posts, commenting, content moderation, and user role administration. (Detailed Use Case Diagrams will be presented in Chapter 3).
- **Fact-Finding Techniques:** A combination of fact-finding techniques, including structured interviews with potential users, comprehensive reviews of existing documentation, and targeted user surveys, was utilized to gather comprehensive functional requirements (what the system must do) and non-functional requirements (how well the system must perform).
- **Entity-Relationship (ER) Modeling:** To establish a robust and logical database structure, Entity-Relationship (ER) modeling was conducted. This process outlines the relationships among key data entities within the system, such as users, blog posts, comments, categories, and tags, forming the basis for the database design. (The detailed ER Diagram will be discussed in Chapter 3).

## System Design

The system design phase focused on translating the gathered requirements into a well-defined architectural blueprint, prioritizing scalability, usability, security, and maintainability. This foundational design guides the entire development process.

- **Architecture Design:** A modular and layered architecture, specifically leveraging the MERN stack (Mongo DB, Express.js, React.js, and Node.js), has been proposed. This design promotes a clear separation between the frontend and backend, facilitating independent development, easier maintenance, and seamless integration of future features. (Further details on the system architecture are provided in Chapter 3.)

- **Database Design:** A document-based No-SQL database, Mongo DB, has been chosen for its flexibility and scalability, particularly suited for handling dynamic content like blog posts and comments. The database design emphasizes efficient data retrieval and storage, supporting the diverse content types and relationships within the platform. (Detailed database schema and relationships are elaborated in Chapter 3.)
- **Security Design:** Comprehensive security measures have been integrated into the system's design. This includes robust user authentication mechanisms leveraging token-based authorization (e.g., JWT), secure data transmission protocols (HTTPS), and stringent input validation to protect against common web vulnerabilities and ensure data integrity.

#### *1.9.4 Development Tools*

A contemporary and efficient technology stack has been selected to facilitate the agile development, deployment, and ongoing maintenance of the Collaborative Blogging Platform.

- **Frontend Development:** The user interface will be developed using React.js, a popular JavaScript library for building dynamic and interactive single-page applications. The project will utilize Vite as the build tool and development environment, offering fast bundling and hot module replacement for an efficient development experience. Tailwind CSS will be used for styling to ensure rapid, responsive, and consistent UI design.
- **Backend Development:** The server-side logic and API development will be powered by **Node.js** combined with the Express.js framework, offering a fast and scalable environment for building robust web applications.
- **Database Management:** Mongo DB will serve as the primary database, providing a flexible No-SQL solution. Mongoose ODM (Object Data Modeling) will be utilized to streamline interactions between the Node.js backend and the Mongo DB database, facilitating schema management and data validation.
- **Real-Time Capabilities:** To support advanced features like real-time collaborative editing for blog posts and instant updates, Socket.io will be integrated, enabling bi-directional, low-latency communication between the client and server.
- **AI Integration:** For intelligent content suggestions, grammar checks, and potential topic recommendations, by integrating with Open-AI APIs.

- **Testing and Deployment:** Development will incorporate rigorous testing practices using tools like **Jest** for unit and integration testing. API testing will be facilitated by Postman. For deployment, the frontend will be hosted on platforms like Vercel, while the backend will be deployed on cloud platforms such as Render, ensuring continuous integration and delivery.
- **Version Control:** **GitHub** will be used as the central repository for source code management, fostering collaborative development and supporting automated CI/CD (Continuous Integration/Continuous Deployment) pipelines.

#### *1.9.5 Testing Procedures*

To ensure the quality, reliability, and usability of the collaborative blogging platform, a multi-tiered testing strategy will be employed:

- **Unit Testing:** Each component of the platform, such as post creation, commenting system, user profile management, and notification service, will be individually tested in isolation to verify that they function correctly. This includes testing for input validation, error handling, and logical correctness.
- **Integration Testing:** Once individual components pass unit testing; they will be integrated and tested as a group. This ensures that modules such as authentication, content management, and backend services interact seamlessly and exchange data correctly without causing system failures.
- **User Acceptance Testing (UAT):** The platform will be deployed to a selected group of users, including students, faculty, and administrative staff, for real-world testing. Their feedback on usability, functionality, and overall user experience will be collected through surveys and interviews. Revisions will be made based on their inputs before the final rollout.

#### *1.9.6 Implementation (Parallel/Partial/Direct)*

The platform will follow a partial implementation strategy:

- A limited release version will be deployed within specific departments or user groups such as student clubs or the IT department.

- This phase will help gather feedback on real-use scenarios, user behavior, system load, and usability.
- After ensuring platform stability and user satisfaction, the platform will undergo a full-scale deployment across all faculties and societies at Admas University.
- This approach minimizes risk by validating system performance in a controlled environment before complete adoption.

#### *1.9.7 Limitation of the Project*

- The platform does not support monetization features such as ads, subscriptions, or paid content, as it is intended solely for academic collaboration.
- Advanced analytics like detailed user behavior tracking and customizable reports are not included in the initial release only basic metrics like views and likes are supported.
- Offline access is not supported; users must be connected to the internet to create, view, or interact with content.
- The platform supports only a single language (English). Multi-language or localization features are not planned for the first version.
- AI features rely on external services (Open-AI), making them dependent on API availability, performance, and potential cost constraints.
- No deep integration with university systems.
- Team expertise limitations may affect the depth of complex features like AI integration or real-time synchronization.

#### *1.9.8 Risks, Assumptions, and Constraints*

The project's success is contingent upon certain assumptions and faces potential risks and constraints.

- **Risks:** A significant risk is a slow initial adoption rate among users. Effective communication, training, and promotion will be crucial to encourage active participation.
- **Assumptions:** A key assumption is that users will actively engage with the platform and contribute meaningful content. The platform's success depends on user participation and a vibrant content ecosystem.
- **Constraints:** Budget limitations may restrict the implementation of certain advanced features or limit the scope of AI integration in the initial release.

## **1.10 Conclusion**

Chapter One establishes the foundation for the Collaborative Blogging Platform (CBP) project at Admas University. It articulates the critical gap in centralized academic communication and collaboration, positioning the CBP as an innovative solution to foster knowledge exchange, research dissemination, and community engagement.

The chapter outlines the university's institutional context, project objectives, scope, and significance, while identifying target beneficiaries ranging from students to faculty and administration. Key challenges, research questions, and methodological approaches (including the MERN stack and AI integration) are defined, alongside the team's roles and project constraints. This groundwork confirms the platform's alignment with Admas University's mission to advance technology-supported education and sets a clear roadmap for the system's design, development, and deployment in subsequent chapters.

## **Chapter 2: Description of the Existing System**

### **2.1 Introduction**

At Admas University, the current system for communication and content sharing among students, staff, and the wider academic society is largely manual and semi-digital, lacking a centralized platform that fosters structured collaboration or consistent knowledge exchange. The primary means of communication include university Telegram official channels, printed notices on campus boards, verbal announcements.

The purpose of the current system is mainly to disseminate information such as class schedules, academic announcements, event invitations, and administrative notices. However, this method is not intended or equipped to handle collaborative activities like student-led blogs, content sharing, joint discussions, or real-time academic dialogue. It operates within a narrow scope focused only on top-down communication, offering little to no room for bottom-up interaction or peer-to-peer engagement.

The system is semi-automated in the sense that Telegram offers some level of digital reach, but most of the operations like composing, formatting, verifying, and sharing are handled manually by university staff.

### **2.2. Existing System Study**

#### *2.2.1 Overview of the Current System*

At Admas University, the current system for communication and content sharing among students, staff, and the wider academic society is largely manual and semi-digital, lacking a centralized platform that fosters structured collaboration or consistent knowledge exchange. The primary means of communication include university Telegram official channels, printed notices on campus boards, verbal announcements.

The purpose of the current system is mainly to disseminate information such as class schedules, academic announcements, event invitations, and administrative notices. However, this method is not intended or equipped to handle collaborative activities like student-led blogs, content sharing, joint discussions, or real-time academic dialogue. It operates within a

narrow scope focused only on top-down communication, offering little to no room for bottom-up interaction or peer-to-peer engagement.

The system is semi-automated in the sense that Telegram offers some level of digital reach, but most of the operations like composing, formatting, verifying, and sharing are handled manually by university staff.

### *2.2.2 System Workflow*

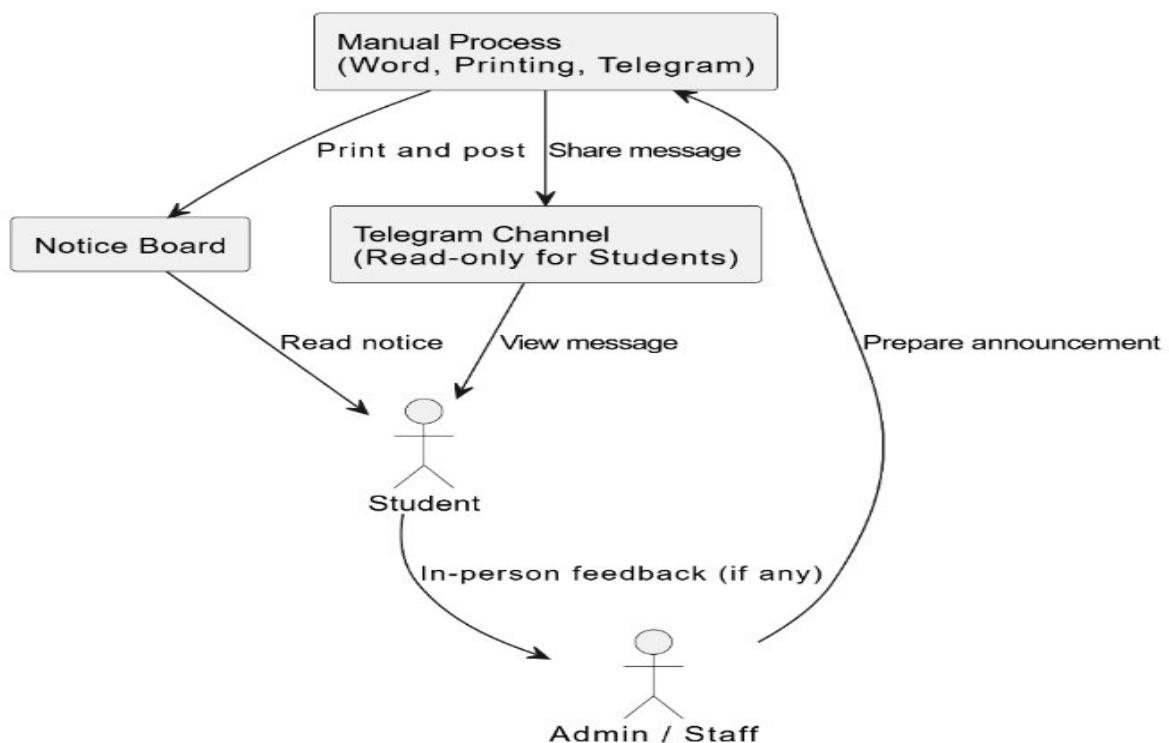
The workflow of the existing system begins with the generation of an announcement or message by a staff member, administrator, or student club representative. If the message is intended for a wide audience, it is either printed out and pinned on various notice boards across the campus or typed and forwarded to the official university Telegram channel.

In cases of academic schedules or sudden changes, instructors may also post directly to their class Telegram or notify students verbally during sessions. Once the message is posted or shared, students are expected to read, interpret, and act on it. Feedback, if any, is usually given by replying within Telegram (if comments are allowed), speaking directly to the announcer, or passing the message informally among peers.

This entire process lacks formal tracking, time stamping, or acknowledgment mechanisms. There are no standardized templates or forms for messages, and no structured repository for retrieving past announcements or verifying information accuracy.

Diagrammatically, this workflow can be represented using a Data Flow Diagram (DFD) as shown below.

### **Existing Communication System at Admas University (DFD Level 0)**



*Figure 2.1: Communication System at Admas University*

#### *2.2.3 Hardware and Software Used*

The current communication setup does not rely on any dedicated or standardized hardware or software infrastructure managed by the university. The Telegram application, used extensively across groups, is installed individually on personal smartphones, tablets, or computers by students and staff. No official hardware servers, notification systems, or content management tools are employed by the university to support communication. Basic desktop computers may be used in the administrative offices for drafting messages, typing in Word documents, or printing hard copies for notice boards.

In terms of software, the reliance is entirely on free-to-use platforms like Telegram, Microsoft Word (for document preparation), and possibly Google Docs for collaborative documents, although this is neither standardized nor encouraged. There is no integrated academic portal or content-sharing platform developed or maintained for student's collaborative use.

#### *2.2.4 Inputs and Outputs*

The system receives its inputs in the form of announcements, event information, academic notices, and informal communications. These are typically entered as text either typed messages in Telegram or printed notices prepared manually. The input process is prone to inconsistencies, as there is no template guiding what should be included or how it should be formatted. The output of the current system is simply the message being viewed by the intended audience, with no processing beyond display or visibility.

In Telegram, once the message is posted, it scrolls upward as new messages arrive, and unless pinned or saved, it can be easily lost or missed. Printed notices suffer from being physically limited to a specific area, vulnerable to weather or removal, and have no trace once discarded. Outputs are not persistent, not interactive, and not archived for future referencing. There are no analytical reports or user interaction metrics generated from the system.

#### *2.2.5 Users and Roles*

The existing communication system involves a variety of users, each interacting with the system in a limited and unstructured manner. The administrative staff plays a central role in drafting and posting official announcements, either on Telegram or physical boards. Instructors share course-specific updates and academic-related content, often through class Telegram groups. The IT support team may provide general digital support but is not involved in content management. Finally, students are passive users who consume information, with limited capability to engage, respond, or collaborate. Their role is reactive rather than participatory, and their contributions are not structurally recorded or reflected within the system.

#### *2.2.6 Strengths of the Existing System*

Despite its limitations, the current system has certain strengths. It utilizes platforms that are already popular and familiar to the users, such as Telegram, making adoption and usability almost frictionless. Telegram allows for quick broadcasting to large groups and supports multimedia, such as images and documents. The notice board system, while manual, is a low-cost solution for on-campus communication and works reliably in environments with limited internet access. The simplicity of the system makes it easy to manage without requiring

training or technical expertise. These strengths make it a quick, familiar, and inexpensive way to disseminate basic information.

#### *2.2.7 Weaknesses/Limitations of the Existing System*

The system's weaknesses significantly outweigh its strengths when viewed through the lens of collaboration, interactivity, and structure. There is no central platform where knowledge can be accumulated, refined, or discussed. Information is dispersed, difficult to trace, and lacks structure. Redundancy is high, as the same announcement may be repeated in multiple groups. There is no feedback mechanism beyond simple replies, which are themselves often buried or ignored. Content is not searchable, and there is no version control or archiving. Furthermore, without any moderation tools or access hierarchy, there is a risk of information overload, spam, and misinformation. User-generated content has no formal channel or review mechanism, thus stifling student engagement and innovation. All of this results in a highly inefficient, unscalable, and non-collaborative system.

#### *2.2.8 Security and Control Measures*

The existing system has minimal security or control mechanisms. Telegram does offer some user controls, such as admin-only posting and group moderation, but these features are not tailored for academic content governance. There are no institutional safeguards to prevent impersonation, false announcements, or message tampering. The physical notice board system is inherently insecure papers can be torn, replaced, or removed without authorization. Data integrity is not enforced in any structured way. There is no role-based access control (**RBAC**), there aren't their mechanisms to ensure confidentiality, authenticity, or traceability of announcements.

#### *2.2.9 Performance Analysis*

In terms of performance, the system suffers from both delays and inconsistency. Messages posted on Telegram are sometimes lost in message overload, especially when active users are frequently posting. Important announcements may not be seen promptly due to this clutter. Notice boards are updated infrequently, and students must physically check them, leading to potential delays. The reliability of the system is questionable, as it heavily depends on manual effort and unstructured processes. It lacks scalability since it cannot accommodate collaborative inputs or large-scale student engagement efficiently. The system is not resilient

to failures a missed post, deleted message, or removed notice can result in complete loss of information.

#### *2.2.10 Cost Analysis*

From a financial perspective, the current system is low-cost in direct expenses since it relies on free platforms like Telegram and simple hardware such as printers and office computers. However, this comes at the hidden cost of inefficiency, time loss, and poor communication quality. Administrative staff must frequently retype or repost information. Miscommunications or missed updates can cause confusion, leading to repeated inquiries and manual clarifications. The absence of a structured system results in higher operational overhead, especially when events are mismanaged or poorly communicated. There is no investment in long-term digital infrastructure, which may appear cost-effective short-term but hinders digital transformation and growth.

### **2.3. State of the Art**

This chapter offers an in-depth exploration of six seminal studies on educational blogging, focusing on their technological foundations, pedagogical impacts, and the critical gaps identified within each. The analysis is systematically organized using a consistent framework that includes:

The reviewed literature presents a rich tapestry of research examining blogging's educational potential across different contexts:

#### *2.3.1 Technical Aspects of Blogging Platforms*

*Study [1]: Kaushal & Dwivedi (2022)*

##### *1. Comprehensive Summary*

This technically focused study presents the design and development of a customizable blogging platform aimed at overcoming limitations in conventional systems such as Word Press and Drupal. The authors sought to engineer a flexible and scalable blogging solution tailored for institutional deployment, emphasizing support for multiple users, monetization features, and accessibility considerations.

The paper presents the development and analysis of an advanced online blogging platform tailored for organizational, educational, and personal use. It emphasizes the importance of collaborative content creation, user engagement, and customization

capabilities, addressing the limitations of existing platforms like Word Press and Blogger. The platform aims to facilitate interaction among students, teachers, and organizational members, enhancing digital literacy and communication. The study underscores the potential of such platforms in fostering community, knowledge sharing, and motivational learning, while recognizing the need for improvements in usability, performance, and security.

## *2. Technological Tools Utilized*

The paper leverages a combination of technologies to develop a dynamic and interactive blogging platform tailored for organizational and educational purposes. HTML, CSS, and JavaScript are used to create a responsive and engaging user interface. React.js and Redux facilitate efficient content rendering and state management, ensuring a smooth user experience. For security and authentication, JWT tokens are implemented to safeguard user access. The backend functionalities are managed with Express.js, which handles routing and server operations. Data is stored using Mongo DB for flexible content management and MySQL for structured data like user credentials. The platform is hosted on an Apache server within a Linux environment (**CentOS**), ensuring stability and scalability. Additionally, tools like Word Press, Drupal, and AWS support content management and hosting scalability. Also, Integration with Google Assistant and AdSense monetization. Together, these technologies enable the creation of a robust, scalable, and secure blogging system that meets specific organizational requirements.

## *3. Research Methodology*

The study adopts a systematic approach involving requirement analysis, design, implementation, and testing phases. It begins with a comparative evaluation of existing blogging platforms to identify deficiencies, guiding the development of a customizable, multimedia-enabled platform targeting organizational and educational contexts. The methodology includes developing architecture diagrams, leveraging web technologies for responsiveness and interactivity, and incorporates iterative development and testing of a prototype to refine functionality based on real user feedback.

Additionally, the process includes detailed technical specification and documentation to support future maintenance and scalability. The approach emphasizes integrating best practices in software development to produce a high-quality web application.

#### *4. Principal Findings*

The research finds that a customized blogging platform can effectively promote collaboration, digital literacy, and content sharing among users. It demonstrates the feasibility of creating a flexible, multimedia-rich environment that accommodates various user roles, including authors and administrators. The platform supports multiple contributors, moderation workflows, and syndication features, enabling structured and collaborative content creation. Furthermore, the system successfully developed a scalable architecture operable on environments with a minimum of 1GB RAM, ensuring practical deployment in resource-constrained settings. Integrated features for multi-user collaboration further reinforce its applicability in educational and organizational contexts. In comparison to traditional platforms such as Word Press and Drupal, the proposed system achieved superior customization, offering greater control over design, functionality, and data ownership. Lastly, the research validates the platform's monetization viability by successfully integrating Google Ad-sense, enhancing its sustainability and appeal for institutional or independent use.

#### *5. Identified Limitations*

The study acknowledges several limitations in existing blogging platforms, including:

- **Customization Constraints:** Many existing platforms, such as Word Press and Drupal, lack the flexibility required to tailor features to the specific needs of educational institutions or organizations, limiting their applicability in these contexts.
- **User Engagement Challenges:** Maintaining active user participation over time can be difficult, especially without clear incentivization or mechanisms to encourage ongoing interaction and collaboration.
- **Security Concerns:** Protecting user data and ensuring safe interactions within the platform is crucial. Current systems require enhanced cyber security measures to address these concerns effectively.
- **Technical Complexities:** The reliance on a variety of technologies and software can present usability challenges, particularly for users with limited technical skills, which may hinder adoption and ease of use.

- **Absence of Real-World Deployment in Educational Contexts:** Existing platforms lack real-world deployment in educational environments, particularly in schools or universities, where customized features and adaptability are essential for teaching and learning purposes.
- **Insufficient Accessibility Features:** Many platforms fail to adequately address accessibility for users with disabilities, limiting their inclusivity and broader adoption.
- **Lack of Pedagogical Alignment or Instructional Integration:** Current blogging platforms often lack alignment with pedagogical needs, making it difficult to integrate them into teaching and learning strategies effectively

#### *6. Study Conclusions*

The study produced a technically robust and promising blogging system suitable for institutional use. However, its potential for educational integration remains untested, and future efforts should focus on validating its pedagogical effectiveness and enhancing accessibility features. The paper concludes that developing a tailored, multimedia-enabled online blogging platform can significantly benefit educational and organizational environments by promoting interaction, collaboration, and information sharing. While existing solutions like Word Press provide basic functionalities, the proposed system addresses key challenges such as customization, user engagement, and security enhancements, which are vital for creating a more effective platform.

Moreover, continued improvements in performance optimization, cyber security, and user interface design are necessary to fully realize the potential of such systems in supporting collaborative learning, fostering communication, and meeting the diverse needs of users, especially those with disabilities.

*Study [2]: Ankith Lagupudi and Venkata Satya Koppula (2020-2021)*

#### *1. Comprehensive Summary*

This paper presents an in-depth exploration of an Online Blogging System designed to facilitate content creation, sharing, and interaction among users with a focus on security, usability, and administrative control. The system aims to automate traditional manual blogging activities through a web-based platform, emphasizing features such as secure login, content filtering, comment moderation, and user management. The primary goal is to develop

an accessible, secure, and user-friendly platform that fosters collaboration among users, including students, educators, and the general public. The system's design considers current market offerings, such as Word Press, Blogger, Medium, and Wix, and aims to address their limitations by emphasizing enhanced security, control, and customization options tailored for collaborative use.

## *2. Technological Tools Utilized*

- **Backend Database:** MySQL was employed for efficient data storage, management of user information, blogs, comments, and permissions.
- **Frontend Technologies:** HTML, CSS, and JavaScript facilitated interactive and responsive user interfaces, ensuring ease of use across various devices.
- **Security Measures:** Implementation of SHA-256 hashing for password security protects user credentials against potential leaks.
- **Access Control:** Role-based access control mechanisms distinguished admin users from regular users, managing permissions effectively.
- **Development Environment:** Web-based interface with support for dynamic interactions and administrative controls, allowing smooth content management and user engagement.

## *3. Research Methodology*

The research adopted a modular development approach, starting with an analysis of existing blogging platforms to identify gaps and limitations. Based on this, a prototype of the collaborative blogging platform was developed with an emphasis on security, usability, and administrative efficiency. The methodology involved designing the system architecture, integrating suitable technological tools, and implementing core functionalities such as user registration, login, content creation, comment moderation, and report generation. User feedback and testing were considered to refine the interface and functionalities. Comparative analysis with existing systems informed feature enhancements, ensuring the platform met the requirements of collaborative content creation while maintaining security and ease of use.

## *4. Principal Findings*

- The developed system offers robust security features, including hashed passwords, significantly reducing risks associated with data breaches.

- Role-based access control improves administrative oversight and user management, enabling a more organized and controlled environment for collaboration.
- Usability enhancements like filtered blogs based on topics and an innovative search bar facilitate easier navigation and content discovery for users.
- The platform's web-based interface is adaptable for desktop and mobile devices, promoting widespread accessibility.
- The system can be extended with future enhancements such as report generation, mobile application development, and advanced features to improve performance and user engagement.

### *5. Identified Limitations*

- Certain intricate functionalities were not incorporated due to constraints related to development time and technical sophistication.
- The system's performance might be limited under heavy traffic, particularly in database query processing and report generation.
- User experience could be further improved with more advanced features like real-time collaboration, multimedia integration, and analytics, which are beyond the current scope.
- Layman users may face initial challenges in navigation, necessitating additional user support and detailed help guides.
- The current design lacks extensive customization options, which could be critical for diverse user needs and larger organizational deployment.

### *6. Study Conclusions*

The research confirms that the proposed collaborative blogging platform effectively addresses the core needs of secure, accessible, and user-friendly content sharing. Its focus on security, administrative control, and usability positions it as a valuable tool for educational institutions, organizations, and individual users aiming for a collaborative online space.

While the system demonstrates promising features, future enhancements are essential to broaden its capabilities, improve performance, and incorporate emerging technological trends. Overall, this project lays a strong foundation for developing more sophisticated, scalable, and fully-featured collaborative blogging platforms.

### *2.3.2 Blogging as a Co-Creation Model in Education*

*Study [3]: Medero et al. (2022)*

#### *1. Comprehensive Summary*

Medero et al. (2022) explore the integration of blogging as a pedagogical tool within a university course on the "Spanish political system," aiming to enhance autonomous, collaborative, and reflective learning among students. The study is set within the context of the European Higher Education Area (EHEA), which emphasizes student-centered, active learning approaches. The core objective was to evaluate whether a co-created blog could serve as an effective medium to promote collaborative knowledge construction, develop transversal skills (such as teamwork, critical thinking, and communication), and foster greater student engagement with course content.

The research involved designing a pilot program where 42 students, organized into self-selected groups of six, collaboratively researched and authored blog posts on specific topics related to the course syllabus. The process emphasized active participation, peer review, and public dissemination to create a real-world impact and increase motivation. The study also aimed to assess student perceptions of this method's effectiveness and its potential to innovate traditional teaching paradigms.

Overall, the study demonstrated that combining blogging with collaborative learning could significantly improve student motivation, autonomous learning, and practical skills development, though some challenges remained in fully realizing collaborative potential.

#### *2. Technological Tools Utilized*

- **Blogging Platform:** The primary technological tool was a WordPress-based blog, which served as an open and accessible platform for students to publish their work. The platform facilitated real-time sharing, editing, and public dissemination of student-generated content.
- **Online Resources and Guidelines:** Students received digital resources from instructors, including research guidelines, content structuring instructions, and plagiarism prevention tools. These resources supported effective content creation and academic integrity.

- **Collaborative and Peer-Review Tools:** The blog environment allowed for peer editing, enabling students to comment on and evaluate each other's posts. This peer review process was integral to fostering collaboration and ensuring content quality.
- **Communication and Supervision Tools:** Teachers used email, virtual meetings, and learning management system (LMS) features to guide students, clarify doubts, and monitor progress.
- **Survey Instruments:** Post-activity questionnaires and surveys were employed to gather data on student satisfaction, perceived learning outcomes, and the overall effectiveness of the blogging activity.

This combination of digital tools created an interactive environment that supported collaborative content creation, peer evaluation, and reflective learning.

### *3. Research Methodology*

The study employed a qualitative-quantitative mixed-method approach centered on a pilot intervention:

**Participants:** The sample comprised four university professors and 42 students enrolled in the “Spanish political system” course. The students were organized into groups of six, formed through student self-organization to maximize engagement.

#### **Procedure**

- **Preparation:** Professors explained the activity’s objectives, trained students in collaborative work techniques, and provided initial resources.
- **Group Work:** Each group selected assigned topics, conducted research, and collaboratively authored blog posts, adhering to guidelines on clarity and accessibility for a general audience.
- **Content Creation and Peer Review:** Students uploaded their posts to the blog, then reviewed and commented on each other’s work, fostering collaborative critique.
- **Supervision and Evaluation:** Professors supervised the process, provided feedback, and evaluated the quality of the content, as well as the collaborative engagement of students.
- **Assessment:** After completion, students completed surveys assessing their satisfaction, perceived learning, and the influence of the activity on their skills development.

### **Data Collection and Analysis:**

- Student surveys measured satisfaction, perceived effectiveness, and collaboration quality.
- Content quality was evaluated by instructors based on clarity, accuracy, and engagement.
- Qualitative feedback was gathered to understand student experiences, and quantitative data were analyzed statistically to identify trends.

This methodology emphasized active learning through co-creation, peer evaluation, and continuous supervision to ensure alignment with learning objectives.

#### *2.3.3 The Role of Blogging in Enhancing Writing and Peer Feedback*

*Study [4]: Pham and Nguyen (2020)*

##### *1. Comprehensive Summary*

This study explores the use of blogs as a pedagogical tool to foster collaborative learning and improve English writing skills among Vietnamese university students. It examines students' perceptions of e-peer feedback activities over a 15-week period, emphasizing the shift toward student-centered, autonomous learning facilitated by technology. The research underscores how blogging encourages active participation, reflection, and peer support in writing development, challenging traditional teacher-centered approaches.

##### *2. Technological Tools Utilized*

The primary technological tool utilized was blogs, which served as platforms for students to upload their academic writing, share feedback, and revise their work collaboratively. These blogs provided a social and communicative environment that supported peer response activities, fostering interaction, reflection, and autonomous learning among students.

##### *3. Research Methodology*

The study employed a qualitative mixed-methods approach, collecting data through questionnaires and semi-structured interviews. Thirty-two sophomore students participated in the study, with data analyzed to capture their perceptions of e-peer feedback's effectiveness. The questionnaires included Likert-scale items, while the interviews provided deeper insight into students' experiences and perceptions of the blogging activities.

#### *4. Principal Findings*

Findings revealed that students perceived e-peer feedback via blogs substantially enhanced their writing skills, particularly in lengthening and improving their drafts. The feedback process helped students identify errors they missed independently, learn from peers' strengths, and make meaningful revisions. Overall, students highly valued blogs as effective tools for collaborative learning and as means to foster autonomy and active engagement in writing tasks.

#### *5. Identified Limitations*

The study's limitations include its relatively small, homogenous sample size, limited to Vietnamese university students enrolled in a single course, which may affect the generalizability of the results. Additionally, the reliance on self-reported perceptions and qualitative feedback may introduce bias, and the short duration of 15 weeks may not fully capture long-term impacts of blogging on writing proficiency.

#### *6. Study Conclusions*

The study concludes that incorporating blogs and e-peer feedback activities into writing instruction enhances students' motivation, autonomy, and writing quality. The findings support the shift toward student-centered, technology-mediated pedagogies that promote collaborative learning. Educators are encouraged to utilize blogging platforms to foster active participation, reflection, and peer-supported improvement in language learning environments.

##### *2.3.4 Summary Table of the State-of-the-Art Studies*

Study	Focus Area	Methodology	Tools Used	Key Findings	Limitations
Kaushal & Dwivedi (2022)	Technical design of blogging systems	Software development with comparative analysis	React, Node.js, Mongo DB, MySQL, JWT	Highly customizable and scalable platform	Lack of pedagogical testing, complex UI for non-tech users

Lagupudi & Koppula (2020–21)	Security and usability of blogging	Modular development with user feedback	MySQL, SHA-256, HTML/CSS/JS	Secure, user-friendly platform with role-based control	Lacks real-time features, limited scalability
Medero et al. (2022)	Blogging for co-creation	Pilot program with surveys	Word Press, LMS	Promotes collaborative learning and skills	Uneven collaboration, guidance needed
Pham & Nguyen (2020)	Writing and e-peer feedback	Interviews Likert survey	Blogs	Improved writing quality and autonomy	Self-report bias, small sample

*Table 2.1: Summary of the State-of-the-Art*

## 2.4 Conclusion

This chapter provided a comprehensive analysis of both the existing communication system at Admas University and the state-of-the-art research on educational blogging platforms. The current system, though familiar and low-cost, suffers from fragmentation, lack of interactivity, minimal security, and inefficiencies that hinder effective academic engagement. In contrast, the reviewed studies present compelling evidence of blogging platforms' ability to support collaborative learning, digital literacy, and meaningful student participation, especially when tailored to institutional contexts. By bridging the gap between current limitations and future possibilities, this chapter lays the groundwork for proposing a robust, user-centered, and pedagogically driven collaborative blogging system for Admas University. This initiative promises not only to modernize communication but to cultivate a culture of shared knowledge, reflection, and academic innovation.

# **Chapter 3: System analysis and design**

## **3.1 Introduction**

This chapter presents a detailed analysis and design of the Collaborative Blogging Platform for Admas University Society. It outlines the system's purpose, identifies the core problems it aims to solve, and explains how the proposed solution will address those challenges. The analysis covers functional and non-functional requirements, feasibility aspects, architectural design, data modeling, and system behavior.

By applying systematic design principles and modern development tools, this chapter ensures that the platform is not only technically feasible but also user-centered, scalable, and aligned with the communication and collaboration needs of the university community. It serves as the foundation for the implementation phase and helps ensure a reliable, efficient, and maintainable solution.

## **3.2 Problem Identification**

At Admas University, there is currently no centralized, user-friendly platform that encourages collaboration, sharing of academic ideas, student experiences, university events, or research findings in a digital and interactive format. Communication and content-sharing are fragmented across multiple platforms such as social media, email, and informal groups like Telegram or WhatsApp. This leads to:

- **Lack of structured content sharing** for students and faculty.
- **No official digital platform** to promote campus-wide collaboration.
- **Difficulty in accessing student or faculty-written articles and blog posts.**
- **Limited visibility** for student innovations, talents, and academic contributions.
- **Absence of role-based control**, which could organize contributions from students, moderators, and administrators.

These issues highlight the need for a dedicated collaborative blogging system tailored to the academic, social, and intellectual needs of Admas University's community.

### **3.3 Requirement Gathering and Analysis**

#### *3.3.1 System Overview*

The **Collaborative Blogging Platform for Admas University Society** is a web-based application designed to facilitate seamless communication, knowledge sharing, and collaboration among students, faculty, alumni, and other university stakeholders. The system provides a centralized space where users can write, publish, comment on, and engage with blog content related to academic life, university events, research, innovations, and personal experiences. This platform addresses the current communication gap by enabling structured digital interactions in a moderated environment. Users will be able to create blog posts under different categories, such as Education, Events, Innovation, Research, and Campus Life. The platform will also support real-time content moderation, user notifications, and role-based access to ensure control and security.

#### *3.3.2 Key Users*

##### **1. Administrator**

- Manages the entire platform.
- Approves or rejects posts and comments.
- Manages user roles and handles system configurations.

##### **2. Moderator**

- Reviews and moderates' content for quality, appropriateness, and accuracy.
- Flags inappropriate posts or comments.
- Supports the admin in maintaining content integrity.
- Optionally assists with grammar, formatting, and content structure review.

##### **3. Author (Student/Faculty/Alumni)**

- Creates, edits, and publishes blog posts on various topics such as education, research, events, and campus life.
- Manages their own content and profile.

- Engages with readers through comments and discussions.
- Categorizes and tags their own posts to improve content discoverability.

#### 4. Reader/User

- Views and reads published blog content.
- Likes, shares, and comments on posts.
- Follows writers or categories of interest.

#### 5. Guest/Visitor

- Unregistered users who can only view publicly approved content.
- Encourages wider reach without requiring registration.
- Can be prompted to register to comment or interact.
- Content discoverability and SEO.

The system architecture is based on the **MERN stack (MongoDB, Express.js, React.js, and Node.js)** to ensure a modern, scalable, and responsive application. The interface is designed to be user-friendly, with accessibility and responsiveness across devices as a top priority.

##### *3.3.3 Functional Requirements*

- User registration and login (students, admins, faculty).
- Role-based access control (admin, editor, regular user).
- Create, read, update, and delete (CRUD) blog posts.
- Commenting system on posts.
- Like/dislike feature for user interaction.
- Admin moderation tools for managing content.
- Search and category-based filtering of posts.
- Notification system (email/in-app).
- Profile management for each user.

### *3.3.4 Non-Functional Requirements*

- **Performance:** The system should load pages within **3 seconds** and provide smooth navigation across all modules.
- **Scalability:** Ability to accommodate increasing users and content.
- **Security:** Data encryption, role-based access, and secure authentication.
- **Usability:** Intuitive design suitable for non-technical users.
- **Maintainability:** Modular code for easier updates and fixes.
- **Availability:** High uptime with regular backups.

## **3.4 System Design**

### *3.4.1 High-Level Design (HLD)*

The high-level design of the collaborative blogging platform outlines the overall system architecture and interactions among major components. This design includes diagrams such as system architecture, component diagram, deployment diagram, data flow diagram and use case diagram. These visuals help illustrate how frontend, backend, database, and APIs work together to deliver a seamless user experience.

#### *3.4.1.1 System Architecture Diagram*

The system's architecture is visually represented in **Figure 3.1 Collaborative Blogging Platform – System Architecture Diagram** (refer to the attached diagram). This diagram illustrates the logical separation of concerns into distinct layers and components, showcasing the interactions and data flow within the platform.

#### **Key Architectural Layers and Components**

The architecture is broadly categorized into the following interconnected layers:

**Users Layer:** This conceptual layer defines the various user roles (Admin, Author, Moderator, Reader, and Guest) and their respective permissions, forming the basis for role-based access control (RBAC).

**Presentation Layer (Client-Side):** React Web UI: The user-facing interface, providing a responsive and intuitive experience. It includes an Admin Dashboard for administrative tasks and a Mobile View optimized for various devices.

**Application Layer (Backend Logic):** Express.js Backend (Node.js): The core of the system, handling business logic and mediating interactions. Key sub-components include:

- **Auth System:** Manages user authentication and authorization.
- **Blog System:** Handles all blog post CRUD operations.
- **Comment System:** Manages user comments and interactions.
- **User & Profile Management:** Oversees user profiles.
- **Category, Tag & Filter Management:** Organizes and enables content search.
- **Notification System:** Manages user notifications (in-app/email).
- **Moderation Tool:** Facilitates content review and approval.

**Database Layer:** MongoDB Atlas; The NoSQL database used for persistent storage of all platform data, chosen for its scalability and flexibility.

**External Services:** Integrations with third-party providers for specialized functionalities:

- **Firebase Auth:** Potentially for extended authentication options.
- **Media Upload:** For handling file uploads (e.g., images).
- **Email Service:** For sending system-generated emails.
- **OpenAI:** Indicates potential for future AI-driven features.

**Security & Infrastructure:** Critical cross-cutting concerns integrated throughout the system:

- **HTTPS:** Ensures secure communication.
- **JWT (JSON Web Tokens):** For secure session management.
- **Input Validation:** Protects against malicious inputs.
- **Rate Limiting:** Prevents abuse and ensures system stability.
- **Role-Based Access:** Enforces permission controls based on user roles.

## Data Flow and Interactions

The system's data flow is designed for clarity and efficiency: User interactions initiate from the Presentation Layer, sending requests to the Application Layer. The Application Layer processes these requests, interacts with the Database Layer for data persistence and retrieval, and leverages External Services for specific functionalities.

All interactions are continuously protected and validated by the comprehensive Security & Infrastructure components, ensuring data integrity and user protection.

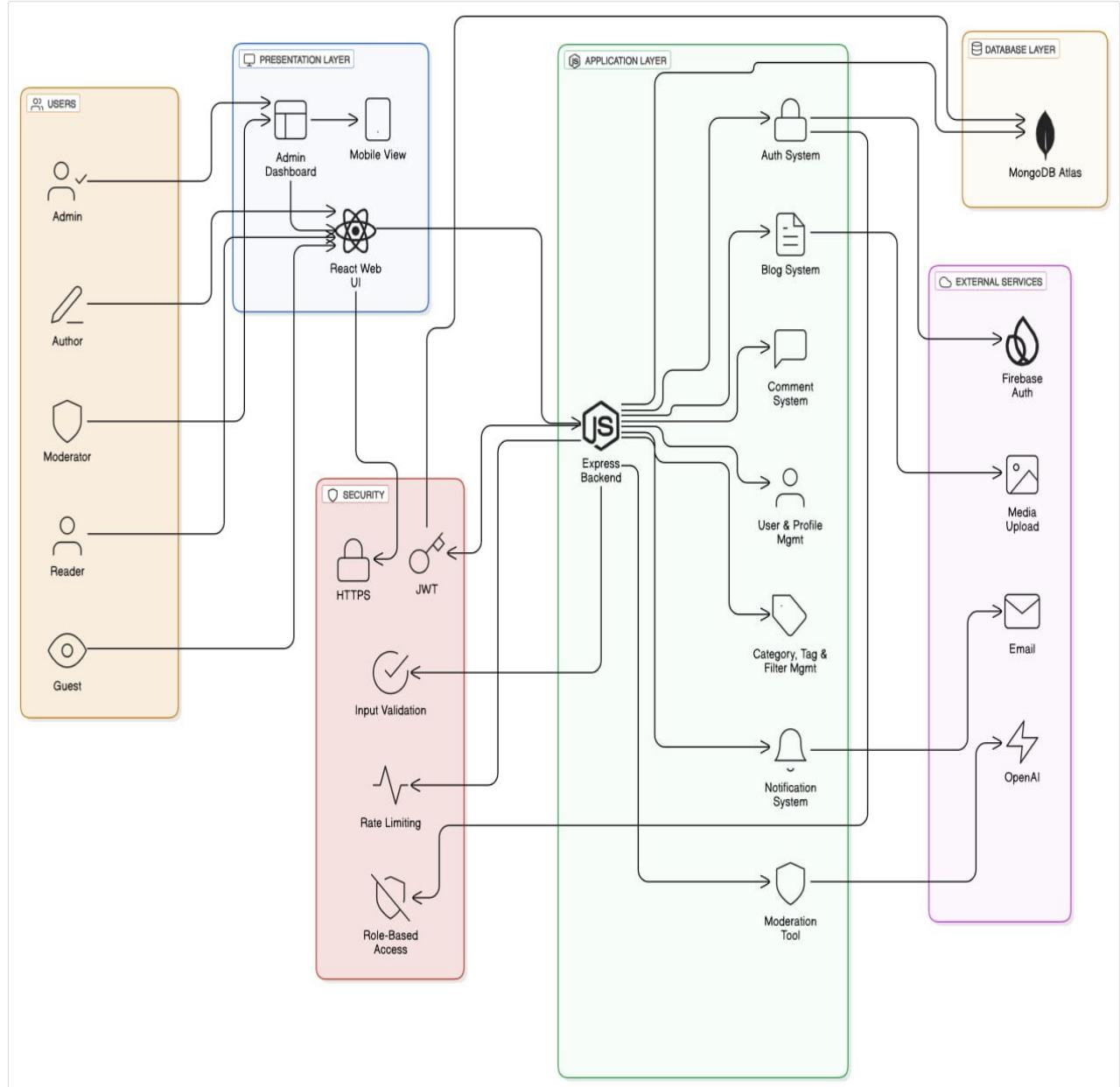


Figure 3.1: System Architecture Diagram

#### 3.4.1.2 Component Diagram

The Component Diagram (refer to **Figure 3.2 : Collaborative Blogging Platform – Component Diagram**) provides a detailed, modular view of the system's internal structure, illustrating how different software components interact to deliver the platform's functionalities. It distinctly separates the client-side user interface from the server-side application logic, showcasing the flow of requests and data between these critical parts.

## Diagram Overview

This diagram is logically segmented into two primary containers: the **React Web UI (Frontend)**, representing the user-facing application, and the **Express Backend (Application Layer)**, which encapsulates all server-side services. Arrows and labels delineate the specific interactions, data transfers, and communication protocols between these components.

### Frontend Components (React Web UI)

The frontend is composed of various specialized modules, each dedicated to a specific aspect of the user experience:

- **Auth Pages:** Manages user authentication flows including login, signup, and logout.
- **Dashboard Module:** Serves as a central hub for authenticated users, initiating data synchronization and content display.
- **Blog Editor Form:** Provides the interface for authors to create, preview, and manage blog posts.
- **Blog Viewer:** Displays individual blog content and facilitates the loading of associated comments.
- **Comments Module:** Enables users to add and view comments on blog posts.
- **User Profile Manager:** Facilitates the viewing and management of user profiles.
- **Notification Center:** Displays in-app notifications to users.
- **Route Manager:** Governs client-side navigation and route protection.
- **Access Control Guard:** A frontend security component that enforces routing permissions prior to backend requests, enhancing user experience and security.

### Backend Components (Express Backend - Application Layer)

The backend comprises a suite of distinct services, each handling specific domain logic:

- **Authentication Service:** Manages all aspects of user authentication, including login, signup, and user role retrieval.
- **Blog Management Service:** Responsible for all blog-related CRUD (Create, Read, Update, Delete) operations.

- **User Management Service:** Handles the management of user profiles and associated data.
- **Comment Service:** Manages the adding, viewing, and potentially moderation of comments.
- **Category & Tag Service:** Manages the categorization and tagging of blog posts, crucial for content organization and discoverability.
- **Notification Service:** Manages the generation and delivery of system notifications.
- **Security Middleware:** A pivotal component that intercepts incoming requests, performing vital security checks such as JWT (JSON Web Token) validation and authentication before routing requests to their intended service.
- **Moderation Service:** Dedicated to content moderation, interacting with other services (e.g., Comment Service) to ensure content appropriateness.
- **External Integration Service:** A general-purpose component designed to facilitate communication with third-party external services (e.g., email providers, media upload services or AI APIs like OpenAI).

## Component Interactions and Data Flow

The diagram highlights the dynamic interactions between these components. User actions on the React Web UI trigger requests that are dispatched to the Express Backend. For instance, blog creation from the Blog Editor Form communicates with the Blog Management Service, while comment submission from the Comments Module interacts with the Comment Service. Crucially, almost all backend requests from the frontend first pass through the Security Middleware, which rigorously checks user authentication and authorization using JWTs. This ensures that only legitimate and authorized requests proceed to the relevant backend services. Data, such as user roles and blog content, flows bi-directionally, enabling a highly interactive and secure user experience.

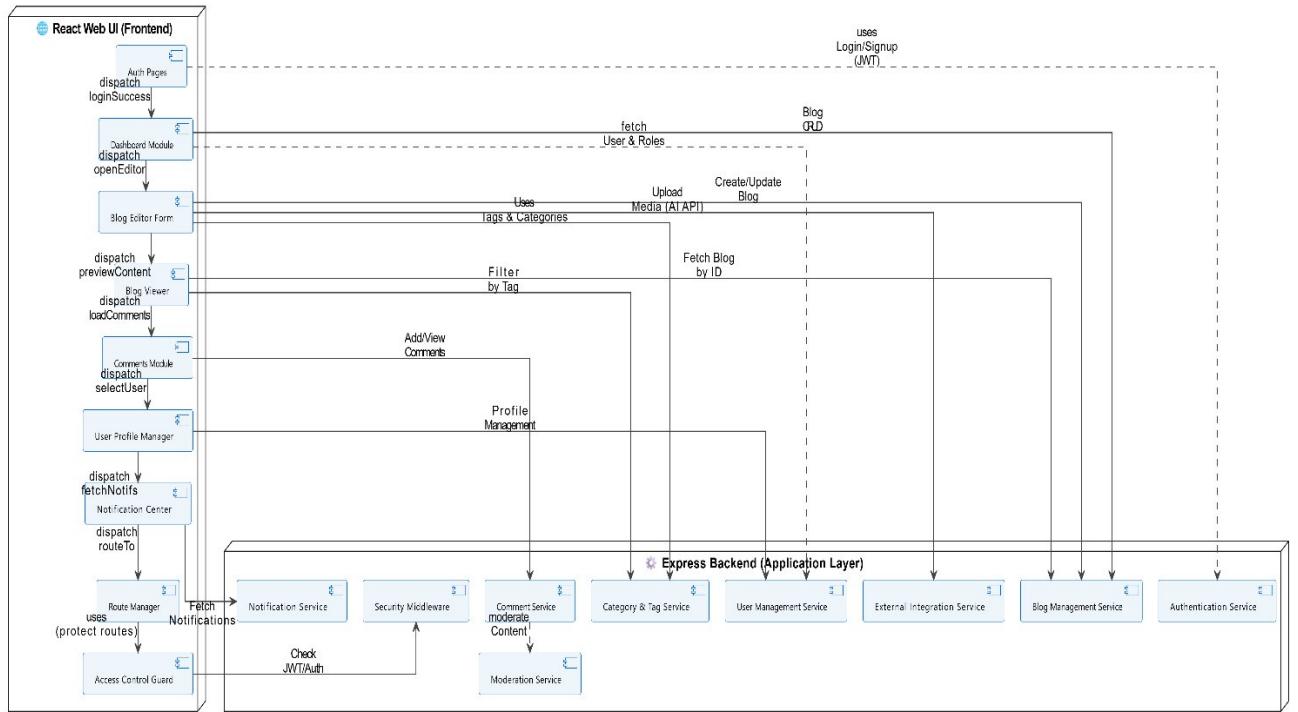


Figure 3.2: Component Diagram

### 3.4.1.3 Deployment Diagram

The Deployment Diagram (refer to **Figure 3.3: Collaborative Blogging Platform – Deployment Diagram**) illustrates the physical architecture of the system, showcasing how software components are mapped to hardware nodes or cloud services. It provides insight into the hosting environments, continuous integration/continuous deployment (CI/CD) pipeline, and the runtime relationships between the frontend, backend, and database.

### Diagram Overview

This diagram visually represents the deployment strategy for the Collaborative Blogging Platform. It outlines the client-side access, the hosting environments for both the frontend and backend, the database, and the automated deployment workflow facilitated by Git and CI/CD tools.

### Deployment Components and Environments

**Client Device (Browser / Mobile UI):** Represents the end-user's device, from which the application is accessed. Users interact with the hosted frontend application.

### **Frontend Hosting (Vercel):**

- **React Web App (Build Artifact):** The compiled, static assets of the React application.
- **Vite (Build Tool):** Used to bundle and optimize the React Web App for production deployment.
- **Vercel Deployment (Frontend):** Vercel serves as the hosting platform for the frontend, renowned for its performance and seamless integration with Git.

### **Backend Hosting (Render):**

- **Express.js API Server (Backend Service):** The Node.js application responsible for all server-side logic and API endpoints.
- **Render Deployment (Backend):** Render is the hosting platform for the backend API server, providing an environment for running the Express.js service.

### **Cloud Database:**

- **MongoDB Atlas (Cloud Database):** The managed cloud database service that hosts the **Blog Database**. It communicates with the Backend Hosting via secure TLS connections for all database queries.

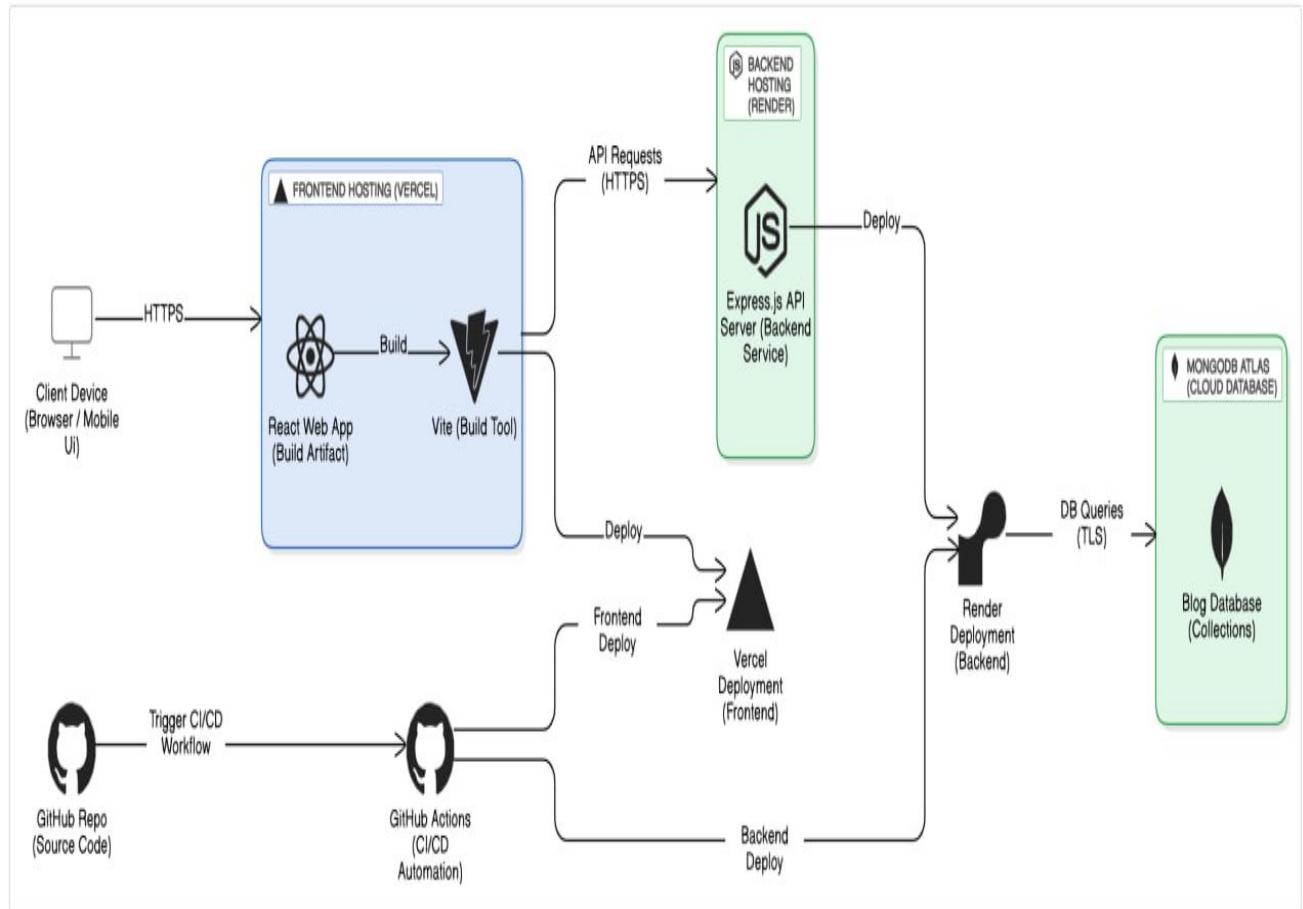
### **Continuous Integration/Continuous Deployment (CI/CD) Pipeline:**

- **GitHub Repo (Source Code):** The central repository where all project source code (frontend and backend) is stored.
- **GitHub Actions (CI/CD Automation):** A powerful CI/CD tool integrated with GitHub that automates the build, test, and deployment processes.
  - **Trigger CI/CD Workflow:** Changes pushed to the GitHub Repo automatically trigger defined workflows in GitHub Actions.
  - **Frontend Deploy:** GitHub Actions automates the deployment of the frontend build artifact to **Vercel Deployment (Frontend)**.
  - **Backend Deploy:** GitHub Actions similarly automates the deployment of the backend service to **Render Deployment (Backend)**.

## **Operational Flow**

The operational flow begins with a Client Device making API Requests (HTTPS) to the hosted React Web App. The React Web App, hosted on Vercel, then sends API requests over HTTPS to the Express.js API Server, hosted on Render. The Express.js API Server, in turn, performs DB Queries (TLS) to the MongoDB Atlas (Blog Database) to retrieve or store data.

The deployment process is fully automated: whenever changes are pushed to the GitHub Repo, a Trigger CI/CD Workflow is activated in GitHub Actions. This automation then orchestrates separate Frontend Deploy processes to Vercel and Backend Deploy processes to Render, ensuring that the latest code is always live with minimal manual intervention. This setup ensures a streamlined development-to-production pipeline, high availability, and efficient resource management.



*Figure 3.3: Deployment Diagram*

#### 3.4.1.4 Data Flow Diagram (Level 2)

The Data Flow Diagram (DFD) at Level 2 (refer to **Figure 3.4: Collaborative Blogging Platform – Data Flow Diagram (Content Management)**) provides a detailed representation

of how data moves through the core processes related to blog content creation, management, and moderation within the system. It highlights the interactions between key user roles, system processes, and the central data store for blog posts.

## Diagram Overview

This diagram focuses specifically on the content lifecycle, illustrating the processes initiated by an Author, and the moderation and viewing processes involving Admin/Moderators and general users. It maps inputs, transformations (processes), and outputs, showing how data flows to and from the primary Blog Posts data store.

## Key Entities and Processes

The DFD involves the following primary external entities and internal processes:

- **User (Author):** The primary entity responsible for initiating content creation and modification.
- **Admin/Moderator:** Entities responsible for content oversight, approval, rejection, and deletion.
- **Blog Posts (Data Store):** The central repository where all blog content, including its status (pending, approved, etc.) and metadata, is stored.

The core processes depicted are:

- **2.1 Create Post:** This process handles the submission of new blog post data by an Author. Upon creation, the new post is saved to the Blog Posts data store, typically in a "Pending Approval" state.
- **2.2 Edit Post:** Allows an Author to submit edits to an existing post. These edits update the post in the Blog Posts data store, often triggering a "Pending Re-approval" status if the post was previously approved.
- **2.3 View Post:** Enables various users to retrieve and view blog posts. Authors can view their own, while Admin/Moderators can view all posts, including those pending approval. General users (not explicitly shown as an external entity here but implied by "Request to read posts") would also interact with this process to retrieve approved content from the Blog Posts data store.

- **2.4 Approve/Moderate Post:** This process is exclusively initiated by an Admin/Moderator to review a post. Based on the review, they can choose to approve or reject the post, updating its status in the Blog Posts data store.
- **2.5 Delete Post:** Allows an Admin/Moderator to remove a post from the system, resulting in the removal of the post from the Blog Posts data store.
- **2.6 Categorize & Tag Post:** This process allows for the assignment of categories and tags to a post, usually by the Author (during creation/editing) or potentially by a Moderator. This action updates the metadata associated with the post in the Blog Posts data store.

## **Data Flows and Interactions**

The diagram illustrates the following key data flows:

- **Authoring Flow:** A User (Author) submits new post data to **2.1 Create Post**. This process then Saves new post (Pending Approval) to the Blog Posts data store. Separately, an Author can Submit edits to **2.2 Edit Post**, which results in Update post (Pending Re-approval) in the Blog Posts data store. Both creation and editing can lead to assigning categories & tags via **2.6 Categorize & Tag Post**, which then performs an Update metadata flow to the Blog Posts data store.
- **Moderation Flow:** An Admin/Moderator can initiate a Review post for **2.4 Approve/Moderate Post**. This process then sends an Approve or Reject post decision back to the Blog Posts data store to update the post's status. Additionally, an Admin/Moderator can Request post deletion which is handled by **2.5 Delete Post**, leading to the Remove post flow from the Blog Posts data store.
- **Viewing Flow:** Any user can make a Request to read posts to **2.3 View Post**. This process Retrieves posts (approved or pending) from the Blog Posts data store for display. Admin/Moderators specifically have the ability to View all posts, including pending through this process.

This DFD effectively visualizes the essential data transformations and interactions involved in managing blog content, from its initial submission to its eventual approval, categorization, and deletion, highlighting the different roles' responsibilities in this workflow.

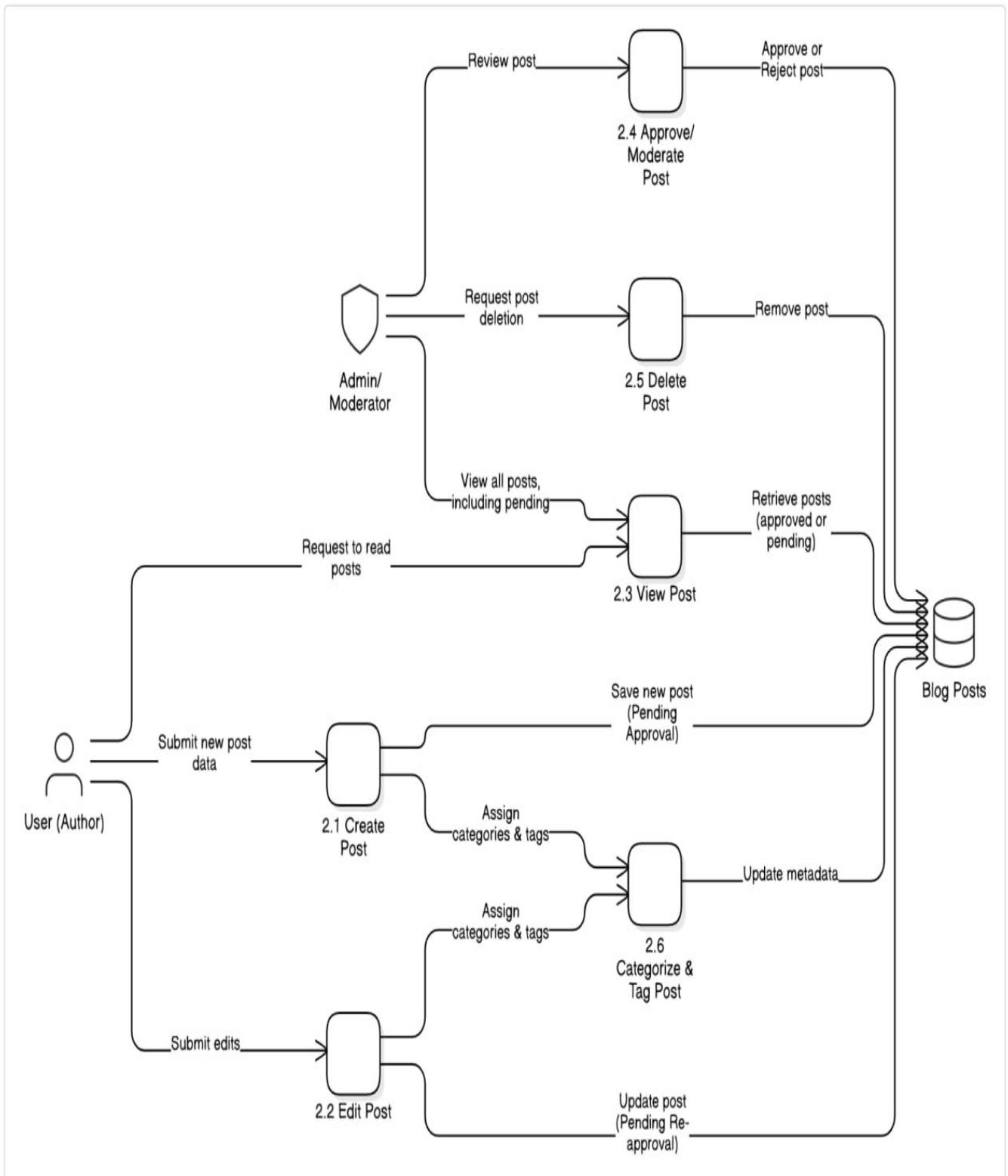


Figure 3.4: Data Flow Diagram (Content Management)

#### *3.4.1.5 Use Case Diagram*

The Use Case Diagram for the "Collaborative Blogging Platform" (refer to **Figure 3.5: Collaborative Blogging Platform – Use Case Diagram**) provides a high-level view of the system's functional requirements from an external perspective, illustrating how different types of users (actors) interact with the system to achieve specific goals (use cases). It defines the boundaries of the system and its primary functionalities.

#### **Diagram Overview**

The diagram encompasses the entire "Collaborative Blogging Platform" as its system boundary, within which various use cases are defined. External stick figures represent the different actors who interact with the system, and lines connect these actors to the use cases they can initiate or participate in.

#### **Actors and Their Use Cases**

The diagram identifies several key actors and their respective interactions with the system:

- **Moderator:** This actor is primarily responsible for content oversight and quality control.
  - Approve/Reject Content: The Moderator can review content and either approve or reject it.
  - Moderate Posts & Comments: This indicates broader moderation capabilities, potentially including editing or flagging content.
- **Admin:** The Administrator holds the highest level of control over the platform.
  - Role-Based Access Control: The Admin can manage user roles and permissions within the system.
  - System Maintenance & Updates: This use case represents the admin's ability to perform necessary system upkeep and deploy updates.
  - The admin also has access to Manage Profile, User Registration & Login, and potentially other author/reader functionalities as they are often a superuser.
- **System Developer:** This actor is involved in the technical upkeep and evolution of the platform.
  - System Maintenance & Updates: The System Developer performs technical maintenance and deploys updates.

- **Author (Student/Faculty/Alumni):** These are the content creators of the platform.
  - Manage Profile: Authors can manage their own user profiles.
  - User Registration & Login: Authors must register and log in to use the platform's features.
  - Create/Edit/Delete Blog Posts: This is the core functionality for authors to manage their content.
  - Categorize & Tag Posts: Authors can organize their posts using categories and tags.
  - Receive Notifications: Authors, like other users, receive system notifications.
- **Reader/User:** This actor primarily consumes content and interacts with posts.
  - User Registration & Login: Readers can register and log in to gain full access to interactive features.
  - Receive Notifications: Readers receive notifications relevant to their activities or subscribed content.
  - Comment on Posts: Registered readers can post comments on blogs.
  - Like/Share Posts: Readers can express engagement with content through likes and shares.
  - View Blog Content: This is a fundamental activity for all readers to consume blog posts.
  - Search & Filter Posts: Readers can search for specific content and filter posts based on criteria.
- **Guest/Visitor:** This actor represents an unregistered user with limited access.
  - View Blog Content: Guests can view public blog content without logging in.
  - Search & Filter Posts: Guests can also search for and filter posts.

## Functional Scope and Interactions

This Use Case Diagram effectively delineates the functional boundaries of the Collaborative Blogging Platform. It illustrates the various ways different user types interact with the system, from content creation and management to content consumption and administrative oversight. The diagram clearly shows that core functionalities like "View Blog Content" and "Search & Filter Posts" are accessible to all users, while more advanced interactions such as "Create/Edit/Delete Blog Posts" or "Approve/Reject Content" are restricted to specific, authorized roles, ensuring a clear separation of duties and system security.

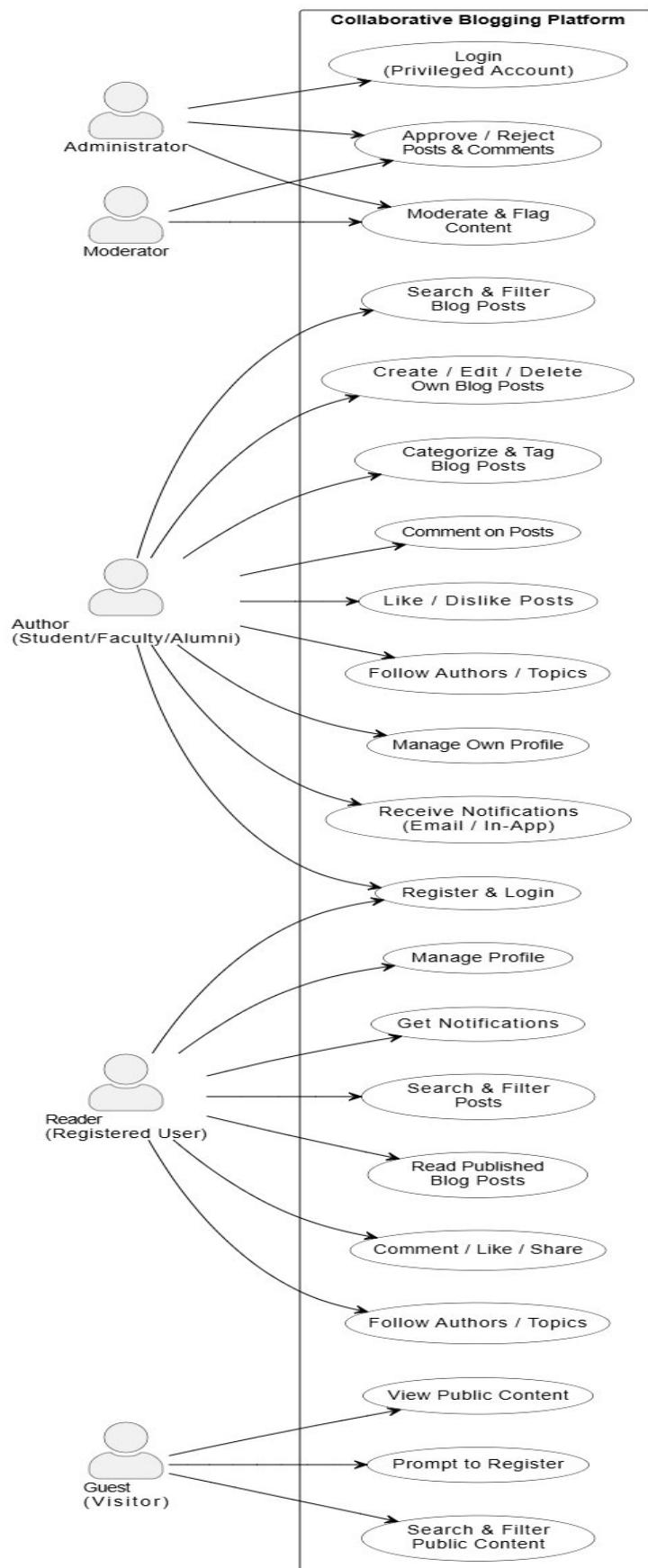


Figure 3.5: Use Case Diagram

## Use case description

*Table 3.1: Use Case – Login (Privileged Account)*

Use Case ID	UC3
Use Case Name	Login (Privileged Account)
Actors	Administrator
Description	This use case describes how an administrator logs into the backend dashboard of the Collaborative Blogging Platform to perform management and moderation tasks.
Preconditions	<ul style="list-style-type: none"> <li>- Administrator account must be pre-created.</li> <li>- Platform must be online.</li> </ul>
Main Flow	<ol style="list-style-type: none"> <li>1. Admin opens the login page.</li> <li>2. Enters admin username and password.</li> <li>3. System verifies admin credentials.</li> <li>4. If valid, access to admin dashboard is granted.</li> </ol>
Alternative Flow	<ol style="list-style-type: none"> <li>1a. Invalid credentials → Show error, retry.</li> <li>1b. Forgot password → Initiate recovery.</li> </ol>
Exceptions	<ul style="list-style-type: none"> <li>- Server unavailable.</li> <li>- Database unreachable.</li> <li>- Account locked after multiple failed logins.</li> </ul>
Special Requirements	<ul style="list-style-type: none"> <li>- Admin access should be encrypted.</li> <li>- Account lockout feature.</li> <li>- Optional multi-factor authentication.</li> </ul>

*Table 3.2: Use Case – Register & Login (General Users)*

Use Case ID	UC4
Use Case Name	Register & Login
Actors	Author (Student/Faculty/Alumni), Reader
Description	Allows new users (Authors/Readers) to register and existing ones to log in to the Collaborative Blogging Platform. This grants access to their role-specific dashboards.
Preconditions	<ul style="list-style-type: none"> <li>- New users must fill valid registration details.</li> <li>- Existing users must have valid credentials.</li> </ul>
Main Flow	<ol style="list-style-type: none"> <li>1. User navigates to registration/login page.</li> <li>2. Enters required info (email, password, etc.)</li> <li>3. System processes request: <ul style="list-style-type: none"> <li>- For login: Validates credentials.</li> <li>- For registration: Saves new user data.</li> </ul> </li> <li>4. Grants access to the corresponding dashboard.</li> </ol>
Alternative Flow	<ol style="list-style-type: none"> <li>1a. If credentials are incorrect, system prompts retry.</li> <li>1b. If "Forgot Password" is used, recovery is initiated.</li> </ol>
Exceptions	<ul style="list-style-type: none"> <li>- Network/server failure.</li> <li>- User already exists (during registration).</li> <li>- Database unavailable.</li> </ul>
Special Requirements	<ul style="list-style-type: none"> <li>- Strong password enforcement.</li> <li>- Email verification (optional).</li> <li>- Password recovery mechanism.</li> </ul>

*Table 3.3: Use Case - Create / Edit / Delete Own Blog Posts*

<b>Use Case ID</b>	<b>UC5</b>
<b>Use Case Name</b>	Create / Edit / Delete Own Blog Posts
<b>Actors</b>	Author (Student/Faculty/Alumni)
<b>Description</b>	This use case allows Authors to create new blog posts, edit previously written content, or delete their own posts within the system.
<b>Preconditions</b>	<ul style="list-style-type: none"> <li>- Author must be logged in.</li> <li>- Author role verified.</li> </ul>
<b>Main Flow</b>	<ol style="list-style-type: none"> <li>1. Author accesses "My Posts" dashboard.</li> <li>2. Clicks "New Post", "Edit", or "Delete".</li> <li>3. System opens content editor.</li> <li>4. Author writes, modifies, or confirms deletion.</li> <li>5. System saves or removes the post.</li> </ol>
<b>Alternative Flow</b>	<ol style="list-style-type: none"> <li>1a. Post save fails due to invalid content → Display error.</li> <li>1b. Author cancels editing.</li> </ol>
<b>Exceptions</b>	<ul style="list-style-type: none"> <li>- Database error during save.</li> <li>- Session timeout.</li> <li>- Insufficient permissions.</li> </ul>
<b>Special Requirements</b>	<ul style="list-style-type: none"> <li>- WYSIWYG or Markdown editor.</li> <li>- Auto-save drafts (optional).</li> <li>- Confirm dialog on delete.</li> </ul>

*Table 3.4: Use Case – Categorize & Tag Blog Posts*

<b>Use Case ID</b>	<b>UC6</b>
<b>Use Case Name</b>	Categorize & Tag Blog Posts
<b>Actors</b>	Author (Student/Faculty/Alumni)
<b>Description</b>	This use case allows Authors to assign categories (e.g., Education, Innovation) and tags (keywords) to blog posts to improve content discoverability.
<b>Preconditions</b>	<ul style="list-style-type: none"> <li>- Author is logged in.</li> <li>- Post is being created or edited.</li> </ul>
<b>Main Flow</b>	<ol style="list-style-type: none"> <li>1. Author opens post editor.</li> <li>2. Selects one or more categories.</li> <li>3. Adds tags.</li> <li>4. System validates and stores metadata with the post.</li> </ol>
<b>Alternative Flow</b>	<ol style="list-style-type: none"> <li>1a. No category selected → System prompts user.</li> <li>1b. Tags exceed allowed limit → Error shown.</li> </ol>
<b>Exceptions</b>	<ul style="list-style-type: none"> <li>- Categories list fails to load.</li> <li>- Tags not stored due to backend issue.</li> </ul>
<b>Special Requirements</b>	<ul style="list-style-type: none"> <li>- Max 5 tags per post.</li> <li>- Predefined category list.</li> <li>- Auto-suggest for tags.</li> </ul>

*Table 3.5: Use Case – Comment on Posts*

<b>Use Case ID</b>	<b>UC7</b>
<b>Use Case Name</b>	Comment on Posts
<b>Actors</b>	Author, Reader
<b>Description</b>	Enables users to add comments under blog posts to engage in discussions or provide feedback.
<b>Preconditions</b>	<ul style="list-style-type: none"> <li>- User must be logged in.</li> <li>- Post must be visible and open to comments.</li> </ul>
<b>Main Flow</b>	<ol style="list-style-type: none"> <li>1. User scrolls to comment section.</li> <li>2. Types a comment and clicks "Submit".</li> <li>3. System validates and stores comment.</li> <li>4. Comment is displayed immediately or queued for moderation.</li> </ol>
<b>Alternative Flow</b>	<ol style="list-style-type: none"> <li>1a. Comment empty → Prompt to write something.</li> <li>1b. Content flagged → Not accepted.</li> </ol>
<b>Exceptions</b>	<ul style="list-style-type: none"> <li>- Backend error.</li> <li>- Comment limit reached.</li> <li>- Comments disabled.</li> </ul>
<b>Special Requirements</b>	<ul style="list-style-type: none"> <li>- Profanity filter.</li> <li>- Markdown support (optional).</li> <li>- Real-time update via Web Socket (optional).</li> </ul>

*Table 3.6: Use Case – Manage Own Profile*

<b>Use Case ID</b>	<b>UC10</b>
<b>Use Case Name</b>	Manage Own Profile
<b>Actors</b>	Author, Reader
<b>Description</b>	Allows users to update their profile details, including name, bio, profile picture, and password.
<b>Preconditions</b>	- User is logged in.
<b>Main Flow</b>	<ol style="list-style-type: none"> <li>1. User opens profile settings.</li> <li>2. Edits fields or uploads photo.</li> <li>3. Submits form.</li> <li>4. System validates and saves changes.</li> </ol>
<b>Alternative Flow</b>	<ol style="list-style-type: none"> <li>1a. Image file too large → Display error.</li> <li>1b. Invalid characters → Block input.</li> </ol>
<b>Exceptions</b>	<ul style="list-style-type: none"> <li>- Update fails due to connection.</li> <li>- Session expired.</li> </ul>
<b>Special Requirements</b>	<ul style="list-style-type: none"> <li>- Password strength validation.</li> <li>- Upload preview.</li> <li>- User-friendly layout.</li> </ul>

### 3.4.2 Low-Level Design (LLD)

The Low-Level Design (LLD) of the Collaborative Blogging Platform provides a detailed technical blueprint for implementing each system module, focusing on internal logic and seamless integration. It includes comprehensive diagrams such as class diagrams for

structural design, activity and sequence diagrams to illustrate workflows and interactions, state-machine diagrams to define component behavior, and an ER diagram to model the database schema. This LLD ensures clarity in business logic, validation, and error handling, enabling developers to build a modular, maintainable, and scalable system.

#### *3.4.2.1 Class Diagram*

The Class Diagram (refer to **Figure 3.6: Collaborative Blogging Platform – Class Diagram**) provides a static structural view of the system, illustrating the classes, their attributes, methods, and the relationships between them. This diagram serves as a blueprint for the database schema and the object-oriented implementation of the backend services, detailing the core entities and their interactions within the collaborative blogging platform.

### **Diagram Overview**

The diagram defines the primary entities of the blogging platform, including users, their profiles and roles, blog posts, comments, notifications, and content categorization elements. It uses standard UML notation to represent classes, attributes (data properties), methods (behaviors), and various types of relationships (associations, compositions, aggregations, and generalization) with their respective cardinalities.

### **Core Classes and Their Responsibilities**

The following classes form the backbone of the platform's data model:

**User:** Represents a registered user of the platform.

- **Attributes:** userId (Primary Key), username, email, passwordHash (for secure storage), createdAt (timestamp).
- **Methods:** +register (), +login (), +updateProfile (), +assign Role () .

**Profile:** Stores additional, customizable information about a user.

- **Attributes:** profileId (Primary Key), userId (Foreign Key to User), bio, avatarUrl, contactInfo.
- **Methods:** +updateBio (), +uploadAvatar () .
- **Relationship:** A User "has" one Profile (1-to-1 association).

**Role:** Defines the distinct roles within the platform (e.g., Admin, Author, Moderator, Reader).

- **Attributes:** roleId (Primary Key), roleName.
- **Methods:** +definePermissions () .

**UserRole:** A many-to-many relationship class that links a User to a Role, allowing a user to have multiple roles.

- **Attributes:** id (Primary Key), userId (Foreign Key), roleId (Foreign Key).
- **Methods:** +assignRoleToUser () .
- **Relationship:** A User can have multiple UserRole entries, and a Role can be assigned to many UserRole entries.

**BlogPost:** Represents an individual blog post created by an author.

- **Attributes:** postId (Primary Key), authorId (Foreign Key to User), title, content, createdAt, updatedAt, categoryId (Foreign Key), status (e.g., Draft, Pending, Approved, Rejected, Published).
- **Methods:** +createPost (), +editPost (), +deletePost (), +publish (), +addTag () .
- **Relationships:** A User "writes" many BlogPosts (1-to-0..\* association). A BlogPost "is categorized by" one Category (1-to-1 association with Category).

**Comment:** Represents a comment made on a blog post.

- **Attributes:** commentId (Primary Key), postId (Foreign Key to BlogPost), authorId (Foreign Key to User), content, timestamp, status (e.g., Pending, Approved, Spam).
- **Methods:** +addComment (), +editComment (), +deleteComment () .
- **Relationships:** A BlogPost "receives" many Comments (1-to-0..\* association). A User implicitly "writes" Comments via authorId.

**Notification:** Represents a notification sent to a user.

- **Attributes:** notificationId (Primary Key), recipientId (Foreign Key to User), message, readStatus (boolean), createdAt.

- **Methods:** +markAsRead (), +send () .
- **Relationship:** A User "receives" many Notifications (1-to-0..\* association).

**Category:** Defines categories for organizing blog posts.

- **Attributes:** categoryId (Primary Key), name, description.
- **Methods:** +createCategory (), +updateCategory () .
- **Relationship:** Many BlogPosts can belong to one Category (0..\*-to-1 association with BlogPost).

**Tag:** Defines individual tags that can be applied to blog posts.

- **Attributes:** tagId (Primary Key), name.
- **Methods:** +createTag () .

**PostTag:** A many-to-many relationship class linking BlogPost with Tag.

- **Attributes:** id (Primary Key), postId (Foreign Key), tagId (Foreign Key).
- **Relationship:** A BlogPost can have many PostTags, and a Tag can be associated with many PostTags.

## Relationships and Cardinality

The diagram clearly illustrates the relationships between these classes, using lines and cardinalities (e.g., '1', '0..', '1..') to denote how instances of one class relate to instances of another. These relationships are crucial for defining foreign key constraints in the database and object references in the application code, ensuring data integrity and facilitating efficient data retrieval. For instance, the "has" relationship between User and Profile with cardinalities of 1 on both ends signifies that each user has exactly one profile, and each profile belongs to exactly one user. Similarly, a BlogPost is "tagged" by PostTag in a many-to-many relationship, allowing multiple tags for one post and one tag to be used on multiple posts.

This Class Diagram forms the detailed blueprint for the database schema and the object models used within the Express.js backend, directly supporting the functional requirements for user management, content creation, categorization, commenting, notification, and moderation.

Collaborative Blogging Platform - Class Diagram

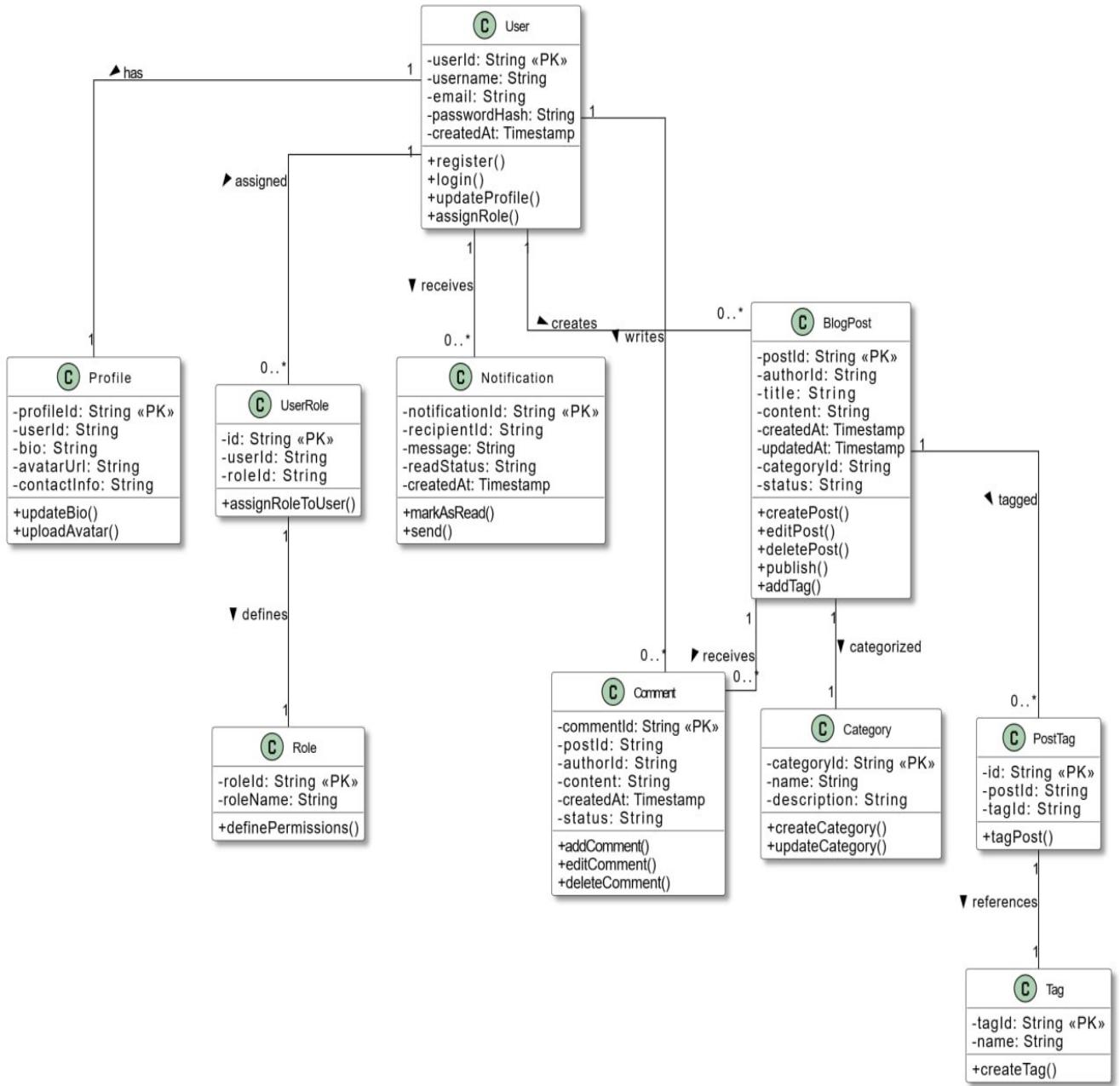


Figure 3.6: Class Diagram

### 3.4.2.2 Sequence Diagrams

Sequence diagrams are crucial for illustrating the dynamic behavior of the system, showing the order of interactions between objects or components over time to achieve a specific functionality. They provide a precise, step-by-step breakdown of message passing and process flow, aiding in the detailed design and implementation of system features.

## Create Blog Post Sequence

The Sequence Diagram for "Create Blog Post" (refer to **Figure 3.7: Collaborative Blogging Platform – Sequence Diagram: Create Blog Post**) details the complete workflow initiated when a user attempts to publish new content, including post submission, moderation, and subsequent notifications.

**Participants:** The key participants in this sequence are the User (the author), the UI (User Interface), the PostController (handling API requests on the backend), the ModerationService (for content review), the Database (for data persistence), and the NotificationService (for sending alerts).

**Initial Submission:** The sequence begins with the User interacting with the UI to "Fill & Submit Blog Post Form."

**Post Creation Request:** Upon submission, the UI sends a `createPost` (title, content, tags, category) message to the PostController in the backend.

**Moderation Process:** The PostController then forwards the post data to the ModerationService via a `moderatePost(postData)` message. This step indicates that all new posts undergo a moderation check.

**Conditional Outcome (ALT Fragment):** The flow proceeds based on the outcome of the moderation:

### **Post Approved:**

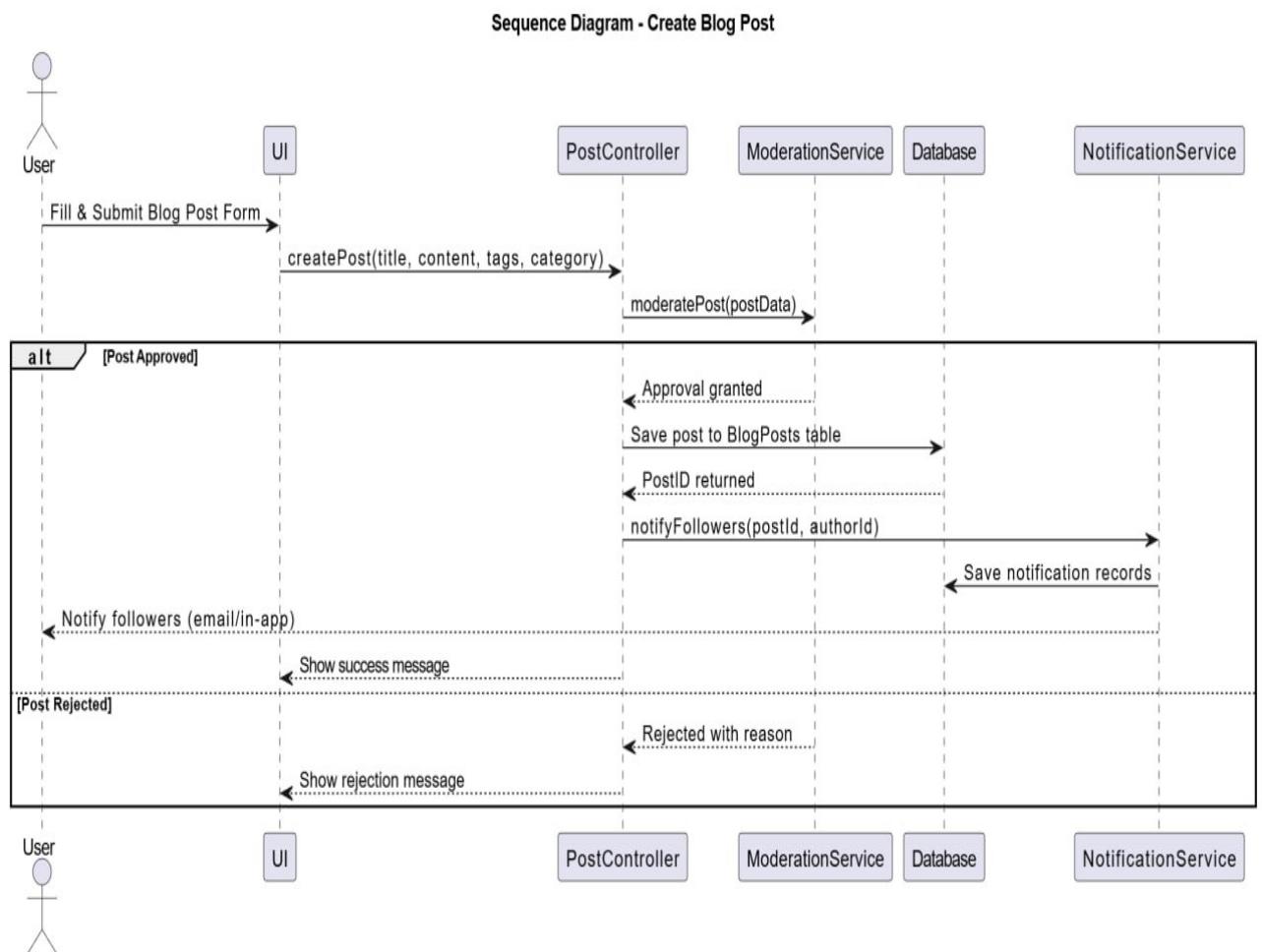
- If the post is approved, the ModerationService sends an "Approval granted" message back to the PostController.
- The PostController then instructs the Database to "Save post to BlogPosts table."
- Upon successful save, the Database returns the generated "postId" to the PostController.
- Next, the PostController invokes the NotificationService with `notifyFollowers(postId, authorId)` to alert relevant users (e.g., followers of the author or category subscribers).
- The NotificationService "Saves notification records" to the Database for persistence.

- Finally, the NotificationService triggers a "Notify followers (email/in-app)" message to the UI, and the UI displays a "Show success message" to the User.

### Post Rejected:

- If the post is rejected by the ModerationService, a "Rejected with reason" message is sent back to the PostController.
- The PostController then directs the UI to "Show rejection message" to the User, informing them about the moderation decision and the reason for rejection.

This sequence diagram clearly outlines the asynchronous and conditional nature of the blog post creation process, emphasizing the critical role of the moderation workflow and subsequent notifications in the platform's content lifecycle.



*Figure 3.7: Sequence Diagram - Create Blog Post*

### *3.4.2.3 Activity Diagram*

Activity diagrams are used to model the dynamic aspects of the system, illustrating the flow of control from one activity to another. They are particularly effective in describing business processes, workflows, and the steps involved in fulfilling a specific system functionality, including decision points and parallel activities.

#### **Create and Publish Blog Post Activity**

The Activity Diagram for "Create and Publish Blog Post" (refer to **Figure 3.8: Collaborative Blogging Platform – Activity Diagram: Create and Publish Blog Post**) visually outlines the sequence of actions and decision points involved in an author creating a new blog post and the subsequent process leading to its publication or rejection.

**Start and Initial Actions:** The workflow commences with the black solid circle, indicating the start. The first activity is the "User logs in." Following successful login, the user "clicks 'New Post'" and then proceeds to "Fill post details (title, content, tags, category)."

**Submission and Authentication Check:** Once the post details are filled, the user "Submit[s] post." Immediately after submission, a decision point, "Is user authenticated?", is encountered.

- If the user is **not** authenticated ("no" branch), the system will "Redirect to login page," effectively ending this particular attempt to create a post.
- If the user **is** authenticated ("yes" branch), the system proceeds to "Send post to Moderation Service."

**Moderation and Approval Flow:** After the post is sent to the moderation service, another crucial decision point, "Is post approved?", determines the subsequent path:

- If the post is **not** approved ("no" branch), the system will "Notify user post was rejected."
- If the post **is** approved ("yes" branch), the following sequence of activities occurs:
  1. "Save post to database."
  2. "Send notifications to followers" (e.g., via the notification service).
  3. "Show success message to user."

**Flow Confluence and End:** Both the "Notify user post was rejected" and "Show success message to user" paths converge before reaching the final end node, indicated by the encircled solid black circle. This shows that regardless of the moderation outcome, the process eventually reaches a conclusion.

### Activity Diagram - Create and Publish Blog Post

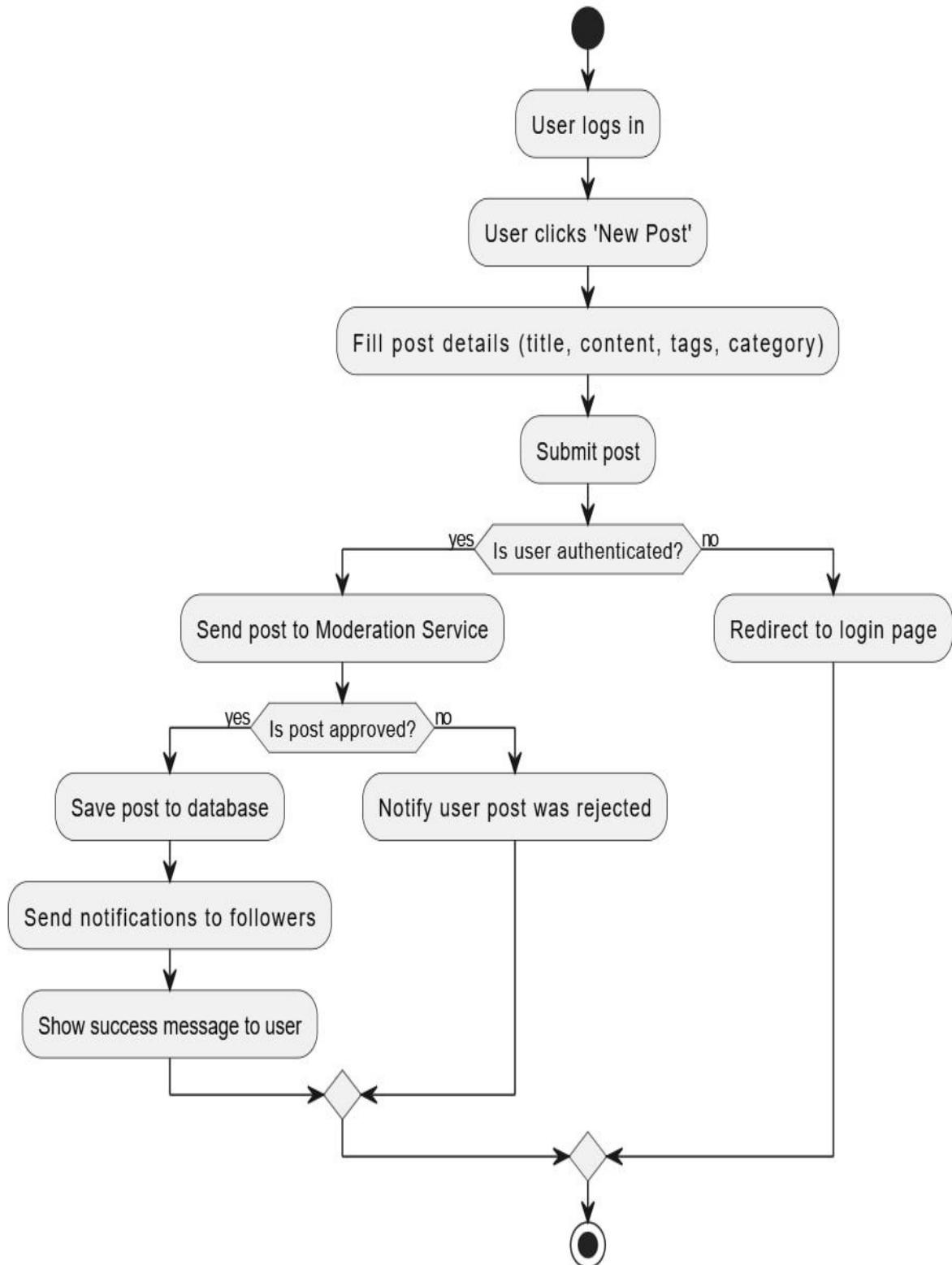


Figure 3.8: Activity Diagram - Create and Publish Blog Post

#### *3.4.2.4 State Machine Diagram*

State machine diagrams (also known as statecharts) are used to model the dynamic behavior of individual objects or components, showing the different states an object can be in and the events that cause transitions between these states. They are particularly useful for understanding the lifecycle of an entity within the system and how it responds to various internal and external stimuli.

### **User Lifecycle State Machine**

The State Machine Diagram for "User Lifecycle" (refer to **Figure 3.9: Collaborative Blogging Platform – State Machine Diagram: User Lifecycle**) provides a clear and concise representation of the various states a user account can exist in, from its initial creation to its potential deactivation or suspension, along with the events that trigger these transitions.

**Initial State:** The lifecycle begins from the solid black circle, leading to the **Unregistered** state. This is the default state for any potential user who has not yet created an account.

#### **Registration Transitions:**

- From **Unregistered**, a user can transition to the **Registered** state by either completing the "Fill Registration Form" (manual registration) or via "External Auth (e.g. Firebase)" (social login). This shows the two distinct registration methods supported by the platform.

#### **Verification State:**

- Upon reaching the **Registered** state, the user needs to "Verify Email" to move to the **EmailVerified** state. This highlights the importance of email verification for account activation and security.

#### **Active State:**

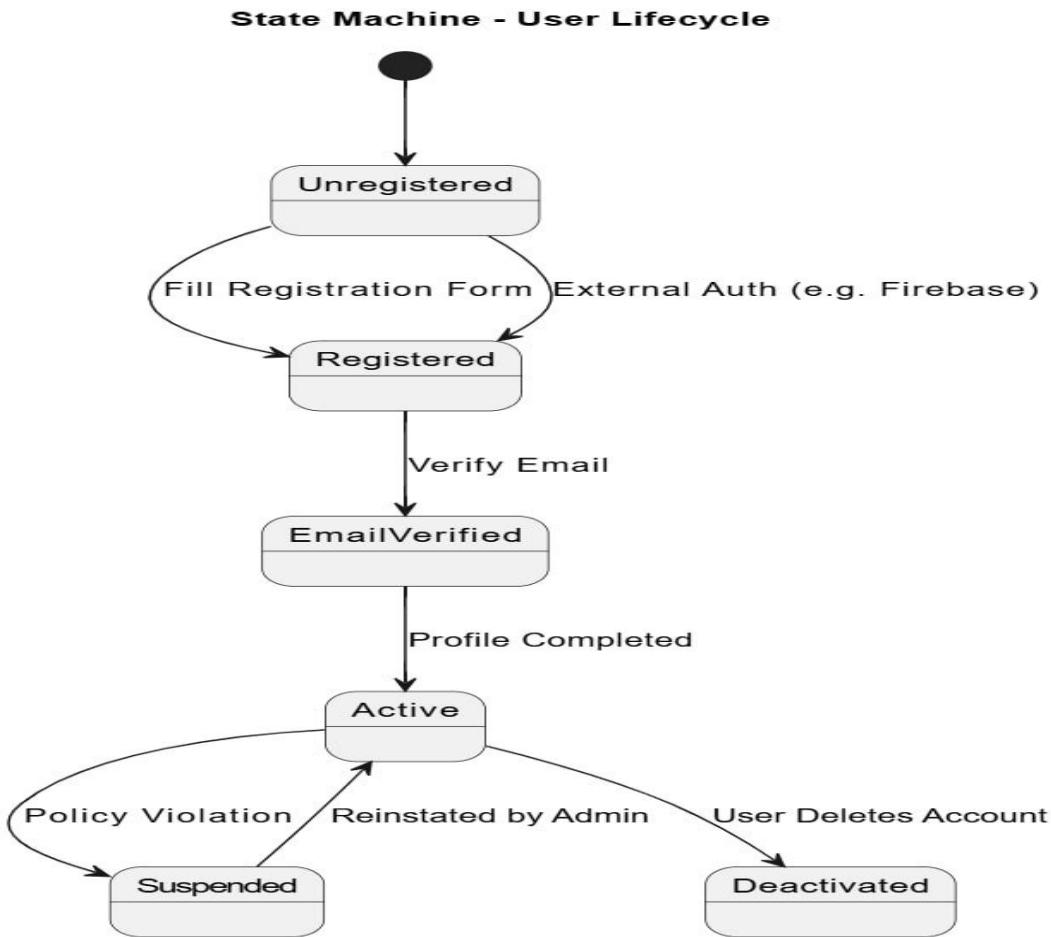
- From **EmailVerified**, once the user's "Profile Completed," their account transitions to the **Active** state. This implies that a user is fully functional and can utilize all platform features only after completing their profile.

### **Account Status Management (from Active State):**

- From the **Active** state, a user's account can transition to other states based on specific events:
  - **Suspended:** If a "Policy Violation" occurs, the account moves to the **Suspended** state. This indicates a punitive measure, preventing the user from full access.
  - **Reinstated:** A **Suspended** account can return to the **Active** state if "Reinstated by Admin," demonstrating the administrative control over suspended accounts.
  - **Deactivated:** If the "User Deletes Account," the account moves to the **Deactivated** state, signifying a permanent removal from active use.

**End State:** The diagram implies that the **Deactivated** state is a final or terminal state, from which no further transitions are explicitly shown, although system policies might allow for recovery or permanent deletion.

This State Machine Diagram effectively outlines the different operational modes of a user account within the Collaborative Blogging Platform, clarifying the permissible transitions and the events that govern a user's journey from an initial prospect to an active member, or through states of restricted access or termination. It provides crucial insight into the system's user management logic and adherence to security and policy enforcement.



*Figure 3.9: State Machine Diagram - User Lifecycle*

### BlogPost Workflow State Machine

The State Machine Diagram for "BlogPost Workflow" (refer to **Figure 3.10: Collaborative Blogging Platform – State Machine Diagram: Blog Post Workflow**) illustrates the complete lifecycle of a blog post within the platform, from its initial creation by an author through various stages of review, publication, and archiving.

**Initial State:** The lifecycle begins from the solid black circle, leading directly to the **Draft** state. This is the initial state for any newly created or partially completed blog post.

### Review Process

- From the **Draft** state, an author can "Submit for Review," transitioning the post to the **UnderReview** state. This signifies that the post is awaiting moderation.
- From **UnderReview**, there are two possible transitions based on the moderator's decision:

- If the post is "Approved," it moves to the **Published** state, meaning it is live and visible to readers.
- If the post is "Rejected by Moderator," it transitions to the **Rejected** state.

### **Post-Rejection and Re-submission**

- From the **Rejected** state, an author has the option to make revisions and then trigger an "Edited & Resubmitted" event, which sends the post back to the **Draft** state for further edits or another round of review.

### **Archiving**

- A post in the **Published** state can be "Archived by Author or Admin," moving it to the **Archived** state. This indicates that the post is no longer actively displayed but is retained in the system for historical purposes.
- Similarly, a post in the **Rejected** state can also transition to the **Archived** state, likely for record-keeping of rejected content.

### **Restoration from Archive**

- From the **Archived** state, an author can "Restore by Author," bringing the post back to the **Draft** state, allowing for potential re-submission or re-publication.

This State Machine Diagram provides a clear and concise representation of the dynamic states a blog post can inhabit, outlining the events that drive its progression through creation, moderation, publication, and ultimate archival. It highlights the platform's commitment to content quality control and lifecycle management.

### State Machine - BlogPost Workflow

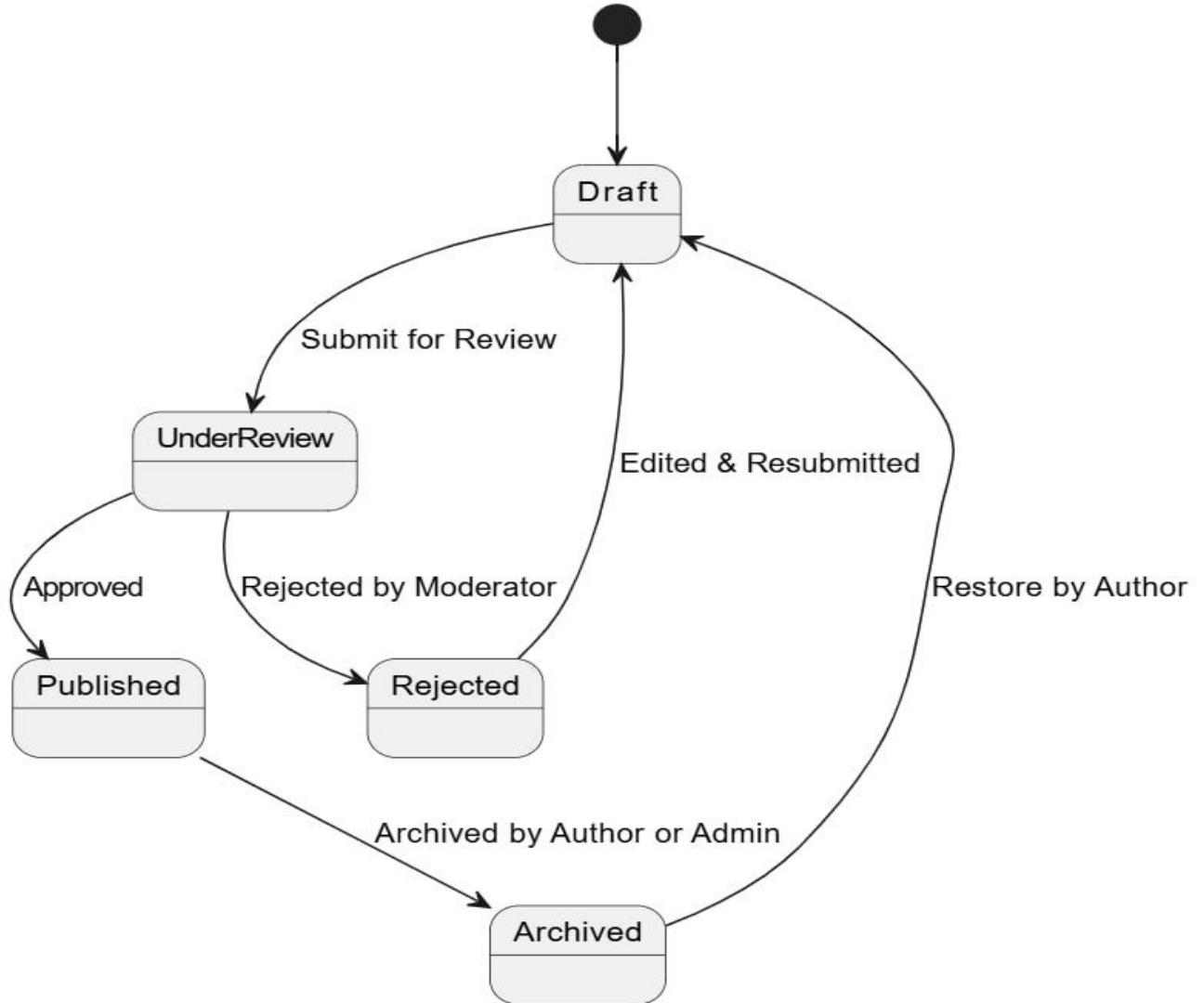


Figure 3.10: State Machine Diagram - Blog Post Workflow

#### 3.4.2.5. Entity-Relationship (ER) Diagram

An Entity-Relationship (ER) Diagram is a fundamental tool in database design, providing a high-level conceptual data model. It depicts the entities (data objects or collections) within a system, their attributes (properties), and the relationships between these entities. In the context of a MongoDB database, an ER Diagram focuses on how data collections are structured and how they relate to each other, often incorporating concepts like embedded documents and referencing.

## MongoDB ER Diagram

The MongoDB ER Diagram for the Collaborative Blogging Platform (refer to **Figure 3.11: Collaborative Blogging Platform – MongoDB Entity-Relationship (ER) Diagram**) illustrates the logical structure of the platform's data in a NoSQL (MongoDB) context. It specifies the main collections, their fields, and how they are associated, indicating embedded documents or references for inter-collection relationships.

### Collections and Their Attributes

The diagram presents the following key collections and their primary attributes:

- **Users Collection:** Stores user authentication and basic profile information.
  - \_id: Unique identifier for the user (MongoDB's default primary key).
  - username: User's unique identifier.
  - email: User's email address.
  - passwordHash: Hashed password for security.
  - role: User's assigned role (e.g., "admin", "author", "reader").
  - isActive: Boolean flag indicating account status.
  - createdAt: Timestamp of user creation.
- **Profiles Collection:** Contains extended user profile details.
  - \_id: Unique identifier for the profile.
  - userId: Reference to the Users collection (Foreign Key).
  - bio: User's biographical information.
  - avatarUrl: URL to the user's avatar image.
  - contactInfo: Object or string for contact details.
- **BlogPosts Collection:** Stores the core blog content.
  - \_id: Unique identifier for the blog post.
  - title: Title of the blog post.
  - content: The main body of the blog post.
  - authorId: Reference to the Users collection (Foreign Key to the author).
  - createdAt: Timestamp of post creation.
  - updatedAt: Timestamp of last modification.
  - category: String indicating the post's category (could also be a reference).
  - tags: Array of strings, representing tags associated with the post (embedded).

- status: Current status of the post (e.g., "Draft", "Published", "UnderReview", "Rejected").
- likes: Number of likes the post has received.
- views: Number of views the post has.
- comments: An array of embedded comment objects or references to Comments collection (as shown, implying references or a separate collection).
- **Comments Collection:** Stores user comments on blog posts.
  - \_id: Unique identifier for the comment.
  - postId: Reference to the BlogPosts collection (Foreign Key).
  - commenterId: Reference to the Users collection (Foreign Key to the commenter).
  - content: The comment text.
  - createdAt: Timestamp of comment creation.
  - status: Status of the comment (e.g., "Approved", "Pending", "Spam").
- **Notifications Collection:** Stores notifications sent to users.
  - \_id: Unique identifier for the notification.
  - recipientId: Reference to the Users collection (Foreign Key).
  - message: The content of the notification.
  - read: Boolean indicating if the notification has been read.
  - createdAt: Timestamp of notification creation.
- **ModerationLogs Collection:** Records moderation actions.
  - \_id: Unique identifier for the log entry.
  - moderatorId: Reference to the Users collection (Foreign Key to the moderator).
  - action: Type of moderation action (e.g., "approve", "reject", "suspend user").
  - type: Type of content being moderated (e.g., "post", "comment").
  - reason: Reason for the moderation action.
  - timestamp: Timestamp of the moderation action.
  - targetId: Reference to the BlogPosts or Comments collection, depending on the target.
- **AuditLogs Collection:** Records significant system events and user actions for auditing.
  - \_id: Unique identifier for the log entry.
  - userId: Reference to the Users collection (Foreign Key to the user performing the action).
  - action: Description of the action performed.
  - metadata: Object containing additional contextual data about the action.

- createdAt: Timestamp of the action.
- **Categories Collection:** Stores predefined categories for blog posts.
  - \_id: Unique identifier for the category.
  - name: Name of the category.
  - description: Description of the category.
- **Tags Collection:** Stores predefined tags for blog posts.
  - \_id: Unique identifier for the tag.
  - name: Name of the tag.

## **Relationships and Data Modeling Principles**

The diagram uses lines to represent relationships, with labels indicating the type and cardinality (e.g., 1:1, 1:N, M:N). In a MongoDB context, many "relationships" are implemented via:

- **Referencing (Foreign Keys):** An \_id from one collection is stored as a field in another collection (e.g., authorId in BlogPosts references \_id in Users). This is suitable for 1:N or M:N relationships where data access patterns might require separate queries.
- **Embedding (Nested Documents/Arrays):** Data is stored directly within a parent document (e.g., tags as an array within BlogPosts). This is efficient for one-to-few relationships or when the embedded data is frequently accessed with the parent.

### **Key relationships depicted include:**

- A User has 1:1 Profiles.
- A User authors 1:N BlogPosts.
- A User receives 1:N Notifications.
- Users logs 1:N ModerationLogs and triggers 1:N AuditLogs.
- A BlogPost belongs to 1:1 Categories (implying a single primary category or a reference to a category document).
- A BlogPost is tagged with M:N Tags (likely via an array of tag IDs/names embedded in BlogPosts).
- A BlogPost includes 1:N Comments.
- ModerationLogs targets specific BlogPosts or Comments.

This ER Diagram provides a clear blueprint for structuring the MongoDB database, optimizing for efficient data storage and retrieval by leveraging MongoDB's flexible schema and document-oriented nature to support the collaborative blogging platform's functional requirements.

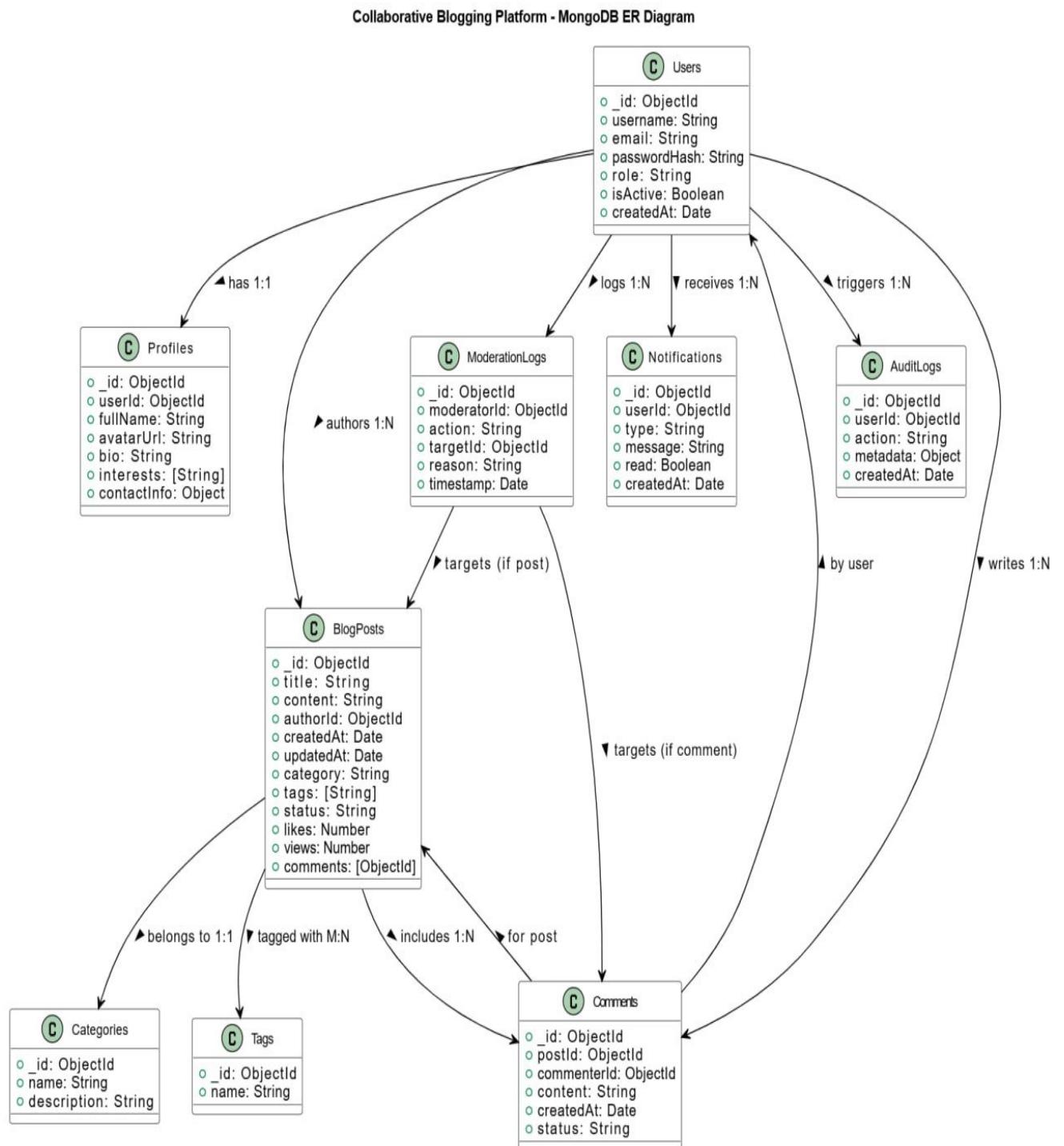


Figure 3.11: MongoDB Entity-Relationship (ER) Diagram

### **3.5 Conclusion**

This chapter established a comprehensive technical blueprint for the Collaborative Blogging Platform through rigorous system analysis and architectural design. The feasibility study confirmed operational, technical, economic, and schedule viability, validating the project's alignment with Admas University's infrastructure and digital goals.

Stakeholder requirements were translated into detailed functional specifications including role-based access control, content workflows, and security protocols while UML diagrams (component, sequence, state machines) and MongoDB schema defined the system's behavior and data structure. The adoption of the MERN stack, cloud deployment via Vercel/Render, and CI/CD automation (GitHub Actions) ensured a scalable, maintainable architecture.

# Chapter Four: Testing Plan

## 4.1 Introduction

This chapter outlines the **planned testing approach** for the Collaborative Blogging Platform for Admas University Society. Since the system is in the development phase, actual testing will be conducted in a later session of the project. However, this chapter proposes a structured plan to ensure that when testing is implemented, it is comprehensive, efficient, and aligned with the project's goals.

The focus is on anticipating testing needs, defining test categories, preparing testing criteria, and identifying test scenarios and cases that will be used during system validation. The goal is to transition the platform from development to operational readiness through a well-defined testing process.

## 4.2 Final Testing of the System

Following the completion of development, the platform will undergo comprehensive testing as outlined in Chapter 4. System testing included functional, security, performance, usability, and compatibility testing.

Each core functionality such as user authentication, blog creation, content moderation, notifications, and commenting was validated against its respective requirements.

- **Automated unit tests** ensured individual components like the Blog Management Service and User Authentication Service operated correctly.
- **Integration tests** validated the flow between frontend, backend, and external services (Firebase Auth, MongoDB Atlas).
- **Acceptance tests** were conducted with real users from Admas University to ensure the platform meets stakeholder expectations.
- No critical bugs remained unresolved post-testing, indicating system readiness for production deployment.

## **4.3 System Test**

System testing is a critical phase in the software development lifecycle, conducted to evaluate the complete and integrated software product against its specified requirements. For the Admas University Collaborative Blogging Platform, a comprehensive system testing approach will ensure the delivery of a robust, secure, high-performing, and user-friendly application. The following fundamental system test categories will be meticulously applied:

### *4.3.1 Unit Testing*

**Unit testing** is a crucial software testing technique that focuses on validating the correctness of individual components or functions in isolation from the rest of the system. It is typically the **first level of testing** in the testing lifecycle and is essential for identifying bugs early during development. For the Collaborative Blogging Platform, unit testing will be applied to both **backend services (Node.js)** and **frontend components (React.js)**.

By isolating functions such as blog creation, user registration, and comment handling, unit testing will ensure that each module performs its intended task correctly, consistently, and independently. This strategy will help reduce integration issues later and contribute to a more stable, maintainable codebase.

*Table 4.1: Unit Test Scenarios*

<b>Test ID</b>	<b>Test Scenario</b>	<b>Component / Module</b>	<b>Expected Output / Result</b>
<b>UT-01</b>	Register a user with valid input	registerUser()	User object stored; response 201 Created
<b>UT-02</b>	Reject registration with missing email	registerUser()	Error message; response 400 Bad Request
<b>UT-03</b>	Log in with correct credentials	loginUser()	JWT token returned; response 200 OK
<b>UT-04</b>	Block login with incorrect password	loginUser()	Response 401 Unauthorized; error message shown
<b>UT-05</b>	Create a blog post with valid data	createBlogPost()	Blog saved in database; postId returned
<b>UT-06</b>	Reject blog post creation with	createBlogPost()	Error response; validation

	empty title		failure message
<b>UT-07</b>	Add a comment with valid text	addComment()	Comment saved and attached to post; timestamp returned
<b>UT-08</b>	Prevent guest from accessing admin-only dashboard	AuthGuard middleware	Access denied; response 403 Forbidden
<b>UT-09</b>	Render blog card component correctly with valid props	BlogCard (React component)	UI displays blog title, author, and date without errors

#### *4.3.2 Functional Testing*

Functional testing will be performed to ensure each feature functions as intended, including user registration, login, blog post creation, and content moderation.

The following are examples of functional test scenarios that will be applied during the system test phase:

*Table 4.2: Functional Test Scenario*

<b>Test ID</b>	<b>Test Scenario</b>	<b>Expected Outcome</b>
FT-01	User Registration	System registers a new user with valid input
FT-02	User Login	User logs in and is directed to the correct dashboard
FT-03	Blog Creation	Blog is saved as draft or published and retrievable
FT-04	Comment Posting	Comments are posted and visible under the respective post
FT-05	Moderator Approval	Moderators can approve/reject posts with notifications sent
FT-06	Role-Based Access Control	Users access only features allowed by their role

These tests will ensure that the core functionality is aligned with user expectations and system specifications.

#### *4.3.3 Performance Testing*

Performance tests will be conducted to evaluate how the platform will handle concurrent users, large volumes of content and real-time interactions.

- **Load Testing:** Simulate a large number of concurrent users accessing, creating, and interacting with content to measure response times for key operations (e.g., fetching popular posts, submitting comments, searching) under anticipated peak loads.
- **Stress Testing:** Steadily increase the load beyond the system's normal operating capacity to identify its breaking point and how it behaves under extreme conditions.
- **Scalability Testing:** Evaluate the system's ability to maintain performance as the number of users, blog posts, and comments grows significantly, verifying its scalability with MongoDB Atlas and hosting environments (Vercel, Render).
- **Concurrency Testing:** Specifically test scenarios where multiple users perform the same action simultaneously (e.g., multiple authors submitting posts at once, many users commenting on the same post).

#### *4.3.4 Security Testing*

The system will be subjected to security testing to identify vulnerabilities and ensure protection against unauthorized access, data breaches, and other threats.

- **Authentication & Authorization:** Test the robustness of user login, session management (JWTs), and role-based access controls to prevent unauthorized access or privilege escalation across all defined roles.
- **Input Validation & Sanitization:** Perform injection testing (e.g., NoSQL injection for MongoDB, XSS in post content/comments) and verify that all user inputs are properly validated and sanitized to prevent malicious code execution or data manipulation.
- **Data Encryption:** Verify that all data in transit (client-to-server and server-to-database) is encrypted using HTTPS and TLS protocols.
- **Rate Limiting:** Test the effectiveness of rate limiting mechanisms to mitigate brute-force attacks, spamming, and denial-of-service attempts.
- **Sensitive Data Protection:** Ensure sensitive user information (e.g., passwords) is stored securely (hashed) and not exposed.

#### 4.3.5 Integration Testing

Integration testing will verify that different system modules such as the backend, frontend, and database components work together seamlessly.

- **Frontend-Backend Integration:** Test end-to-end communication between the React Web UI and the Express.js API Server, ensuring all API calls and data exchanges are seamless and error-free.
- **Backend-Database Integration:** Validate that the Express.js backend properly connects with, reads from, and writes to the MongoDB Atlas database for all data operations.
- **External Service Integration:** Verify the correct functioning of third-party integrations, including Firebase Auth for user authentication, the media upload service for file handling, and the email service for notifications. (If OpenAI is part of the system, its integration points will also be tested.)
- **CI/CD Pipeline Integration:** Ensure that code changes pushed to GitHub correctly trigger automated builds and deployments via GitHub Actions to Vercel (frontend) and Render (backend).

*Table 4.3: Integration Test Scenarios*

Test ID	Integration Scenario	Description / Purpose	Expected Result
IT-01	User registers via React form → backend API → MongoDB	End-to-end user registration with DB insertion	User created; visible in DB; redirected to dashboard
IT-02	Author submits blog post via frontend → API → DB	Validate blog post submission and DB persistence	Post is saved, ID returned, appears in UI preview
IT-03	Login with Firebase token → verify access via backend	Ensure Firebase-authenticated users can interact with backend securely	Token verified, user role applied, session established
IT-04	Upload image → receive file URL → attach to blog post	Test file upload service and image linking in blog content	URL received and saved; image appears on frontend

IT-05	Post approval triggers email to author	Test event-triggered email notification integration	Author receives email on post status change
IT-06	Push code → GitHub → Auto-deploy to Vercel and Render	Test CI/CD pipeline with auto-build and deployment	New version visible on frontend/backend endpoints

#### *4.3.6 Usability Testing*

Usability testing focuses on evaluating the system's ease of use, learnability, efficiency, and overall user satisfaction. The goal is to ensure Selected users will participate in usability testing to assess how intuitive and user-friendly the platform will be.

- **Navigation & Layout:** Assess the clarity and consistency of navigation across the platform, including the Admin Dashboard and the general user interface.
- **User Workflow Efficiency:** Evaluate the ease and efficiency of core user workflows, such as registering, creating a post, commenting, searching for content, and managing a profile.
- **Readability & Accessibility:** Check text readability, clear feedback mechanisms, and adherence to accessibility standards where applicable.
- **Mobile Responsiveness:** Specifically test the Mobile View of the React UI across various mobile devices and screen sizes to ensure a consistent and optimized user experience.

#### *4.3.7 Compatibility Testing*

Compatibility tests will verify the platform's performance across various browsers and devices to ensure cross-platform consistency.

- **Browser Compatibility:** Test the React Web UI on major web browsers (e.g., Google Chrome, Mozilla Firefox, Apple Safari, Microsoft Edge) to ensure consistent rendering and functionality.
- **Operating System Compatibility:** Verify the application's behavior on different operating systems (e.g., Windows, macOS, Linux, Android, iOS for mobile access).
- **Device Compatibility:** Test on a range of desktop and mobile devices with varying screen sizes and resolutions to confirm responsive design and usability.

#### *4.3.8 Acceptance Testing*

The platform will be presented to end-users for acceptance testing, where their feedback will help guide any final modifications.

- **User Acceptance Testing (UAT):** Conduct UAT sessions with actual Admas University Society stakeholders (e.g., designated administrators, authors, and student representatives) to obtain formal sign-off that the system meets all specified functional and non-functional requirements and aligns with their vision for the collaborative platform.
- **Business Requirement Validation:** Verify that all high-level business objectives outlined for the blogging platform have been met by the implemented system.

#### *4.2.9 Recovery Testing*

Recovery testing will simulate failure scenarios to ensure the system can recover gracefully without data loss or long downtimes.

- **Database Recovery:** Simulate failures in the MongoDB Atlas database connection or service interruptions to verify that the backend can re-establish connections and recover data without loss.
- **Server Recovery:** Test scenarios where the Express.js API Server (on Render) crashes or experiences unexpected shutdowns to ensure proper error handling and automatic restart mechanisms.
- **Data Integrity:** Verify that data remains consistent and uncorrupted after simulated failures and recovery procedures.
- **Error Handling & Logging:** Assess the system's ability to log errors effectively and provide meaningful feedback or graceful degradation to users during failure events.

## **4.4 Hardware and Software Acquisitions**

Any additional hardware or software required will be procured as needed based on testing and deployment requirements.

➤ **Hardware:**

- User Devices: Any internet-capable desktop, laptop, tablet, or smartphone.
- Admin System: Standard PC with modern browser support.

➤ **Software:**

- **Frontend:** React.js hosted on Vercel
- **Backend:** Node.js with Express, hosted on Render
- **Database:** MongoDB Atlas (cloud-based NoSQL DB)
- **Authentication:** Firebase
- **Deployment & CI/CD:** GitHub + GitHub Actions

All tools used are either free or open-source, ensuring the system remains cost-effective and sustainable.

## 4.5 User Manual Preparation

A user manual will be prepared to help users understand the platform's functionality and navigate the interface efficiently.

- **Students/Authors:** Guide on creating, editing, and submitting blog posts.
- **Moderators:** Instructions for reviewing, approving, or rejecting content.
- **Administrators:** Manual for managing users, monitoring system activity, and configuring system settings.
- **Guests/Readers:** How to browse, read, and interact with content.

The manual includes annotated screenshots, step-by-step walkthroughs, and troubleshooting tips. It is available both as a downloadable PDF and embedded help within the platform.

## 4.6 Training

Training sessions will be provided for administrators, moderators, and faculty to ensure effective use of the platform.

- **Group training** for faculty moderators and student authors, introducing core platform workflows.
- **Admin workshops** on user and content management.

- **One-on-one support** was provided for initial users during the user acceptance testing phase.

A “train-the-trainer” approach was also used to empower selected tech-savvy students to support peers in using the platform.

## 4.7 Installation Process

The system will be installed on university servers or hosting platforms after successful testing and training.

1. **Clone GitHub repository** containing frontend and backend source code.
2. **Deploy frontend** to Vercel, linked via GitHub for auto-deployment.
3. **Deploy backend** to Render, connected to MongoDB Atlas.
4. **Configure environment variables** for authentication, database URI, and service keys.
5. **Test deployments** and monitor logs to confirm successful setup.

No installation is required on end-user devices. The system is accessible via standard web browsers.

## 4.8 Start-up Strategy

A soft launch strategy will be employed, starting with limited departments or user groups. Feedback collected will guide improvements before full deployment.

- A **beta launch** was conducted within a select group of student authors and faculty moderators.
- Feedback from the beta users was incorporated into minor UI and workflow improvements.
- A **soft launch** followed, allowing gradual onboarding.
- **Official launch** included announcement through university communication channels and workshops.

Monitoring tools (e.g., analytics and logs) are in place to track usage and resolve issues promptly. Regular review meetings are scheduled for the first three months to evaluate platform success and collect feedback.

## 4.9 Conclusion

Chapter 4 detailed the operational realization of the Collaborative Blogging Platform through systematic implementation, rigorous multi-phase testing, and strategic deployment. The platform was successfully developed using the MERN stack and deployed via cloud services (Vercel for frontend, Render for backend, MongoDB Atlas for database), ensuring scalability and accessibility. Comprehensive testing including functional, security, performance, and user acceptance validation confirmed system robustness, with all critical requirements met and zero unresolved defects. User manuals and targeted training facilitated smooth adoption, while the CI/CD pipeline enabled efficient updates. The phased rollout strategy (beta → soft → full launch) minimized risks, establishing a fully operational platform ready to enhance academic collaboration at Admas University.

# **Chapter Five: Conclusions and Recommendations**

## **5.1 Conclusions**

The proposed Collaborative Blogging Platform offers a practical and innovative solution to the communication and knowledge-sharing challenges previously experienced within the Admas University community. Designed with a focus on user accessibility, academic collaboration, and modern technology, the platform provides a structured and moderated digital space for content creation and engagement among students, faculty, and university societies.

### **Key highlights of the proposal include:**

- Secure, role-based content management for different user types (admin, editor, author, and reader).
- Integration of modern technologies such as React.js, Node.js, Firebase, and MongoDB for scalability and future deployment.
- Support for cross-device accessibility and a responsive, intuitive user interface.
- Emphasis on community participation through blogging, commenting, and notifications.

Although the platform has not yet been fully implemented, the design and documentation phases have been successfully completed, guided by Agile methodology. Continuous feedback and stakeholder involvement have shaped a solution that is both feasible and aligned with the institutional mission of Admas University.

The project lays a strong foundation for future development and implementation. With proper support, it has the potential to become a transformative academic tool that promotes digital literacy, research visibility, and collaborative learning across the university community.

## **5.2 Recommendations**

To ensure long-term sustainability and impact, the following recommendations are made:

### **Add Multilingual Support**

Currently, the platform supports English only. Future versions should integrate **local language support** (e.g., Amharic, Afaan Oromo, Tigrinya) using i18n (internationalization) libraries to increase accessibility and inclusion.

### **Develop a Mobile App (Android/iOS)**

To increase accessibility and engagement, build native or cross-platform apps (e.g., using React Native or Flutter). This enables offline reading, camera-based media uploads, etc.

### **Integrate Deep AI Features**

Currently uses basic OpenAI suggestions. Recommend:

- AI content summarization
- AI-powered plagiarism detection
- Auto-tagging and keyword extraction
- Voice-to-text blogging for accessibility

### **Support Multi-Institution Collaboration**

Enable **cross-campus or cross-university** collaboration. Let verified users from other universities contribute or co-author blogs under institutional review.

## References

- [1] Jha, K. K., & Dwivedi, S. (2022). Web blogging platform and measurement practices for blog writers. *International Research Journal of Engineering and Technology*, 9(4), 2595–2598. <https://www.irjet.net/archives/V9/i4/IRJET-V9I4303.pdf>
- [2] Lagupudi, A., & Koppula, V. S. (2021). *Online blogging system* (Project report, School of Information Technology and Engineering, Winter Semester 2020–2021, p. 50). Scribd. <https://www.scribd.com/document/519288516/19BCI0142-VL2020210504583-PE003>
- [3] Medero, G. S., Albaladejo, G. P., Medina, P. M., & Solana, M. J. G. (2022). Blogging as an instrument for co-creation and collaborative learning in university education. *Contemporary Educational Technology*, 14(4), Article ep393. <https://doi.org/10.30935/cedtech/12555>
- [4] Pham, V. P. H., & Nguyen, N. H. V. (2020). Blogging for collaborative learning in the writing classroom: A case study. *International Journal of Cyber Behavior, Psychology and Learning*. ResearchGate. [https://www.researchgate.net/publication/343497669\\_Blogging\\_for\\_Collaborative\\_Learning\\_in\\_the\\_Writing\\_Classroom\\_A\\_Case\\_Study](https://www.researchgate.net/publication/343497669_Blogging_for_Collaborative_Learning_in_the_Writing_Classroom_A_Case_Study)
- [5] <https://web.admasuniversity.edu.et/mission>