ShikshaHub

A project report submitted to
Dr. Babasaheb Ambedkar Technological University , Lonere
In partial fulfilment of the requirements for the award of the degree

Bachelor of Technology
in
Computer Engineering
Mini-Project–II

(BTCOM607)

Submitted by

Miss Komal Pravin Punwatkar (PRN:2246491245012)
Miss Shruti Sanjay Malode(PRN:2246491245017)
Miss Asmi Shashikant Bhandekar(PRN:2246491245024)

(VI Semester)
Under the guidance of

Prof. Sheetal Kale

(Head of the department)

Department of Computer Engineering

Shiksha Mandal's

BAJAJ INSTITUTE OF TECHNOLOGY, WARDHA

Pipri, Arvi Road, Wardha - 442001.

Department of Computer Engineering

(2024-25)

Dr. Babasaheb Ambedkar Technological University, Lonere
Bajaj Institute of Technology, Wardha
Pipri, Arvi Road, Wardha - 442001.

DEPARTMENT OF COMPUTER ENGINEERING

Certificate

This is to certify that Mini-Project-II titled

ShikshaHub

has been completed by

Miss Komal Pravin Punwatkar (PRN:2246491245012)
Miss Shruti Sanjay Malode(PRN:2246491245017)
Miss Asmi Shashikant Bhandekar(PRN:2246491245024)

of VI Semester, Computer Engineering of academic year 2024-25 in partial fulfillment of Miniproject-II (BTCOM607) course as prescribed by the Dr. Babasaheb Ambedkar Technological University, Lonere.

Prof. Sheetal Kale
Head of the Department

Prof. Sheetal Kale
Head of the Department

Place: BIT, Wardha
Date: June 15, 2025

Declaration

We, hereby declare that the project report titled "ShikshaHub" submitted by us to the Bajaj Institute of Technology, Wardha, in partial fulfilment of the requirement for the award of Degree of B. Tech. in Computer Engineering discipline is a record of bonafide project work carried out by me under the guidance of Mrs. Sheetal Kale . I, further declare that this submission by the undersigned represents our original work and we have quoted the references where others ideas/words have been included. I understand any violation of the above will levy a disciplinary action on us. I, further declare that the work reported in this project report has not been submitted either in-part or in-full for the award of any other degree in any other Institute or University.

Report Title:ShikshaHub

SN	**Student Name**	**PRN**	**Signature**
1	Komal Pravin Punwatkar	2246491245012	
2	Shruti Sanjay Malode	2246491245017	
3	Asmi Shashikant Bhandekar	2246491245024	

Date: June 15, 2025
Place: BIT, Wardha

Acknowledgment

We would like to express my sincere gratitude to everyone who contributed to the successful completion of this report on “ShikshaHub”.First and foremost,I would like to thank Mrs.Sheetal Kale ma’am, whose guidance and valuable insights were crucial throughout the preparation of this report. Their encouragement and support have been instrumental in helping me complete this project. I am also grateful to my colleagues, friends, and family for their continuous support, encouragement, and motivation during this project. Lastly, I would like to acknowledge the authors and researchers whose work I referred to, as their contributions have significantly enriched my understanding of the ShikshaHub system.

Abstract

The increasing need for seamless digital collaboration in educational environments has led to the development of integrated platforms that serve students,teachers, and administrators efficiently.This project presents a comprehensive web application built using the MERN stack (MongoDB, Express.js, React.js, Node.js), aimed at fostering a collaborative academic ecosystem. The platform is designed to support three distinct user roles—students, teachers, and admins—each with dedicated functionalities. Students can upload blogs, generate blog summaries, interact with a built-in chatbot for assistance, and request to join academic communities.Teachers are empowered to upload study materials, create academic events, and review or reject student-submitted blogs with feedback. Admins serve as the highest level of authority within the platform and can manage users, oversee final blog publishing decisions, handle study material curation, and manage community creation and access controls. A robust blog approval workflow ensures that content quality is maintained across the platform. When a student uploads a blog, it enters a pending state. Teachers evaluate the submission and provide either approval or rejection with feedback. Approved blogs then proceed to the admin for final review and publication. If rejected, students are prompted to revise and resubmit. In addition, the community feature enables admins to create special interest groups that students can join either by entering a code or by submitting a join request, which the admin can accept or reject based on relevance and validity. This feature promotes student engagement in focused learning groups and discussions. Security and role-based access control are implemented using JWT (JSON Web Tokens) and bcrypt for secure authentication and password management. The application architecture ensures data persistence through MongoDB and provides a responsive user interface through React.js. This platform not only enhances academic communication but also ensures a structured, quality-controlled environment for knowledge sharing. It can be extended to include real-time features, analytics dashboards, and AI-driven feedback mechanisms, making it a scalable solution for educational institutions aiming for digital transformation.

Keywords- *MERN Stack*, *Student-Teacher-Admin Platform*, *Community Management*

Abbreviations

MERN	MongoDB, Express.js, React.js, Node.js
JWT	JSON Web Token
API	Application Programming Interface
UI	User Interface
UX	User Experience
DB	Database
CRUD	Create, Read, Update, Delete
PDF	Portable Document Format
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
JSON	JavaScript Object Notation
ID	Identifier
URL	Uniform Resource Locator
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

1.1 Introduction

In the era of digital education, the demand for integrated online platforms that promote effective communication and collaboration between students, teachers, and administrators is growing rapidly. Traditional classroom settings often face limitations in terms of real-time interaction, personalized feedback, and streamlined content sharing. To overcome these challenges, this project introduces a comprehensive web-based platform built using the MERN stack (MongoDB, Express.js, React.js, and Node.js), tailored to serve as a centralized academic collaboration system.

The core idea behind the platform is to create a unified environment where each user role—student, teacher, and admin—has access to tools and features that streamline their responsibilities and interactions. The system facilitates various essential academic workflows including blog submissions, study material distribution, community engagement, and administrative oversight. Through this platform, students can actively contribute content by submitting blogs, summarize their content, seek academic assistance via a chatbot, and engage in community-based learning. Teachers act as academic reviewers and content providers by evaluating student submissions, providing constructive feedback, uploading educational resources, and creating academic events. Admins hold the highest level of control in the system and are responsible for managing user access, moderating blog content, and facilitating community management.

One of the key features of the system is the multi-step blog approval workflow. When a student submits a blog, it enters a pending state awaiting a teacher's review. Upon review, the blog is either approved—forwarding it to the admin for final publishing—or rejected with feedback, prompting the student to revise and resubmit. This structured workflow ensures quality assurance and academic integrity in the content published on the platform. Another standout feature is the community system, which encourages students to participate in interest-based groups. Admins can create communities and provide access via unique codes. Students can either enter the code or request to join a community, and the admin has the authority to approve or reject the request. This functionality promotes focused academic discussions, peer learning, and engagement beyond traditional course content.

The project leverages modern web development tools and principles including RESTful API design, JWT-based authentication, secure password encryption using bcrypt, and modular front-end architecture using React.js components. MongoDB is used for its flexibility in handling diverse user data and nested documents, making it well-suited

for handling the platform's complex interactions.

1.2 Organization of the Report

This report is structured to present the development of the **ShikshaHub** educational platform in a clear and logical sequence. It includes the following chapters:

- **Chapter 1: Introduction**
  This chapter introduces the purpose and objectives of the ShikshaHub platform and outlines the challenges faced in traditional academic content sharing and community-based learning.
- **Chapter 2: Literature Survey**
  This chapter provides a detailed review of existing educational platforms and technologies, highlighting their limitations and the need for a modular, role-based system like ShikshaHub.
- **Chapter 3: Methodology**
  This chapter explains the development methodology adopted, including requirement analysis, system design, and technology selection (MERN stack).
- **Chapter 4: System Features and Implementation**
  This chapter describes the core features and functionalities implemented in the platform, such as blog submission and review, study material upload, community management, and secure authentication.
- **Chapter 5: Results and Discussion**
  This chapter presents the outcome of the developed platform with screenshots, performance analysis, and user feedback if available.
- **Chapter 6: Conclusion and Future Scope**
  The final chapter concludes the report by summarizing the project's achievements and suggesting potential areas for enhancement and future development.

1.3 Key Features and Functional Objectives

1. **Facilitate Blog Creation and Review Workflow**
   - Enable students to write and upload blogs as a form of self-expression or academic contribution.
   - Allow students to summarize their blogs using a built-in feature, improving their understanding and writing clarity.
   - Establish a multi-level review system:
     - Blogs first undergo review by a teacher who can approve or reject with feedback.
     - Approved blogs are then passed on to the admin for final publishing.
     - Rejected blogs prompt students to edit and resubmit, promoting learning through revision.

2. **Empower Teachers to Manage Academic Content**
    - Allow teachers to upload study materials in various formats (PDFs, images, links, etc.) for students.
    - Enable event creation (such as webinars, workshops, or exams) to enhance academic involvement.
    - Provide a dedicated dashboard for reviewing student blogs, ensuring content quality and relevance.
3. **Provide Administrative Control and Oversight**
    - Grant admins full control over the system including user management (add/edit/remove users).
    - Allow admins to moderate blog content before it is published to ensure quality and appropriateness.
    - Enable admins to manage all uploaded study materials and ensure resources are up-to-date and relevant.
4. **Promote Student Engagement Through Communities**
    - Enable admins to create academic or interest-based communities to foster peer learning and discussions.
    - Allow students to join communities using a unique code or by sending a join request.
    - Give admins the ability to approve or reject community join requests, maintaining controlled and relevant group participation.
5. **Ensure Secure, Scalable, and Role-Based Access**
    - Implement JWT-based authentication for secure login and user session management.
    - Use bcrypt for secure password hashing and storage.
    - Establish role-based access control so that only authorized users can perform specific actions depending on their role (student, teacher, admin).
6. **Provide a Simple and Responsive User Interface**
    - Design a clean, intuitive, and mobile-responsive interface using React.js to ensure smooth user experiences across devices.
    - Keep navigation simple and user-specific, ensuring that each role sees only relevant options and actions.

Chapter 2

Literature Survey

2.1 Literature Review

2.1.1 A Role-Based Educational Portal for Web-Based Learning

Year: 2019
Authors: Priya Singh, Neha Agarwal
Description: This paper presents a role-based educational portal that separates functionalities for students, teachers, and administrators. The system facilitates resource uploads, assignment submissions, and content reviews by different user roles. It influenced the core architecture of ShikshaHub by supporting differentiated access, where students submit blogs and requests, teachers manage content review, and administrators oversee system operations and user control.

2.1.2 Enhancing Student Engagement Through Blogging in Higher Education

Year: 2020
Authors: Dr. Ramesh K. Sharma
Description: This study explores how blogs improve critical thinking, creativity, and peer engagement among students. It advocates for integrating blog writing into the academic workflow with instructor feedback and peer visibility. ShikshaHub adopts this model, offering a structured blog submission and approval system that enhances reflective learning and communication between students and teachers.

2.1.3 A Study on Online Learning Platforms: Features and Usability

Year: 2021
Authors: Sneha Kulkarni, Akash Deshmukh
Description: The research compares various online learning platforms such as Google Classroom, Moodle, and Edmodo, focusing on usability, content management, and teacher-student interactions. It emphasizes intuitive design and feature accessibility. ShikshaHub integrates these insights by offering an easy-to-use interface for students

to access materials, upload blogs, and interact within communities, while ensuring a simple workflow for teachers and admins.

2.1.4 Implementation of Secure Web Applications for Educational Institutions

Year: 2022
Authors: Aakash V. Joshi, Megha Khandelwal
Description: This paper highlights the significance of secure login, role-based access, and data encryption in educational web applications. The research guided ShikshaHub's backend implementation using technologies like JWT for authentication, bcrypt for password hashing, and MongoDB for secure storage of user data and educational content, ensuring privacy and integrity.

2.1.5 Community Learning in Digital Environments: Challenges and Design Principles

Year: 2020
Authors: Maria J. Monteiro, Luis F. Mendes
Description: This paper explores the design of digital communities in educational settings and their role in fostering collaboration and knowledge sharing. It discusses the need for moderation, joining restrictions, and community identity. ShikshaHub's community feature—where students join via code or admin approval—derives from these findings, promoting controlled yet open peer interaction within learning groups.

2.2 Gap identification in the literature

While several research studies have addressed the development of role-based educational portals, online learning platforms, blogging in education, and secure content management systems, there remains a noticeable gap in integrating all these components into a unified, student-driven and community-oriented platform. Most platforms either focus solely on Learning Management Systems (LMS) for content delivery or emphasize blogging as a separate tool for academic reflection. Additionally, few systems support a multi-step content approval process involving both teachers and administrators, which is crucial for maintaining quality and accountability in student-generated content. Community features in educational systems are also underexplored, particularly those that include flexible joining mechanisms (via code or request) with controlled admin oversight. Furthermore, limited emphasis has been placed on real-time interactions, summarization tools, and intelligent feedback mechanisms within such platforms. ShikshaHub addresses these gaps by offering an integrated solution that combines blog writing, teacher review, admin publishing, study material management, event handling, and moderated community participation within a secure, scalable MERN-based web application.

2.3 Problem statement

In the current educational ecosystem, there is a lack of a unified, interactive platform that enables seamless collaboration among students, teachers, and administrators. Most institutions rely on fragmented tools for blog writing, content sharing, communication, and community building, which results in inefficiencies, limited student engagement, and a lack of centralized academic monitoring. Existing systems often lack a structured workflow for blog submission, multi-level review, and publishing processes. Additionally, students face limited opportunities to participate in moderated educational communities or share academic insights through personalized content. Teachers struggle with disorganized content management and lack tools to give structured feedback on student submissions. There is also an absence of secure, scalable platforms that support real-time summarization tools, chatbot assistance, and event integration under one interface. ShikshaHub addresses these issues by offering an all-in-one, role-based educational web platform that facilitates blog creation, teacher-admin content approval workflows, study material sharing,and community engagement—filling the gap in academic interaction, quality assurance, and digital learning collaboration.

2.4 Objectives

• The Encourage Student Creativity – Provide an environment for students to write and publish blogs.
• Streamline Blog Approvals Process – Implement a structured process where the teachers approve and admins post blogs.
• Enhance Learning Materials – Enable teachers to upload study materials to students.
• Facilitate Scholarly Discussions – Promote learning interactions with blog entries and chatbot assistance.
• Enhance Content Quality – Create high-quality blog posts through admin and teacher comments.

2.5 Scope of work

The scope of this project encompasses the design, development, and deployment of a full-stack web application that serves the collaborative needs of students, teachers, and administrators within an educational environment. The platform is structured to enable user-specific roles and responsibilities, support educational workflows, and provide a secure and scalable system using the MERN stack.

In-Scope:

1. **Role-Based Functionalities**
    - **Students can:**
        - Register/login securely
        - Upload blogs and generate summaries

  - Use an integrated chatbot for assistance
  - Join academic communities using a code or by sending a join request
- **Teachers can:**
  - Upload study materials
  - Review, approve, or reject student blogs with feedback
  - Add events such as seminars, assignments, or exams
- **Admins can:**
  - Manage users (add, delete, update)
  - Review and publish approved blogs
  - Manage and update study material repository
  - Create and manage communities
  - Approve or reject community join requests

2. **Community Engagement**
   - Admin-controlled creation of communities
   - Dual joining method: join via code or send request
   - Controlled access through admin approval
3. **Security and Authentication**
   - Implementation of JWT for secure user session management
   - Password encryption using bcrypt
   - Protection of sensitive user data and system endpoints
4. **Technical Scope**
   - Use of React.js for a dynamic, component-based frontend
   - Node.js and Express.js for backend server logic and RESTful APIs
   - MongoDB as the NoSQL database for scalable and flexible data storage
   - Fully responsive design to support access from desktops, tablets, and mobile devices

Chapter 3

Methodology

3.1 Research Design

The Research Design of this project follows an applied, developmental, and iterative model aimed at building a functional educational web application. The goal is to identify real-world problems in academic workflows and solve them through a customized software solution.

1. **Requirement Analysis**
    - Conducted informal research by analyzing existing LMS platforms (e.g., Google Classroom, Moodle) to identify limitations.
    - Defined system requirements based on interviews with students and teachers regarding content publishing, review cycles, and academic communication needs.
    - Outlined key functionalities:
        - Blog upload and review workflow
        - Study material distribution
        - Chatbot integration
        - Community participation
2. **Technology Selection**
    - Chose the MERN stack (MongoDB, Express.js, React.js, Node.js) for its:
        - End-to-end JavaScript environment
        - Scalability and real-time capabilities
        - Popularity and strong developer community
    - Security tools selected: JWT for token-based authentication and bcrypt for password hashing.
3. **System Architecture Design**
    - Designed modular components for each user type (Student, Teacher, Admin).

- Developed a RESTful API architecture to manage requests from the frontend to backend.
- Implemented role-based access control for each API route.
- Used MongoDB to store users, blogs, study materials, community data, and permissions.

4. **Front-End and UI Development**
   - Used React.js for component-based front-end development.
   - Created separate dashboards with dynamic navigation for students, teachers, and admins.
   - Ensured mobile responsiveness and intuitive UI for ease of use.
5. **Backend Development**
   - Built REST APIs using Express.js and Node.js for:
     - Blog creation, review, and publishing
     - User authentication and role validation
     - Study material upload/download
     - Community management and join request handling

3.2 System Architecture

The system architecture of this web application is designed using the MERN Stack—MongoDB, Express.js, React.js, and Node.js. It follows a modular and layered architecture, ensuring separation of concerns, scalability, and maintainability. The architecture is divided into three main layers: Frontend (Client-Side), Backend (Server-Side), and Database (MongoDB). The application also uses RESTful APIs for communication between the client and server.

1. Architecture Layers Overview

a. Frontend (Client-Side) – React.js

- Built using React.js for a dynamic, component-based user interface.
- Role-specific dashboards for:
  - Students (blog creation, summary, chatbot, community joining)
  - Teachers (blog review, feedback, material upload, events)
  - Admins (user management, blog moderation, community management)
- Uses Axios or Fetch API to send HTTP requests to the backend.
- Responsive design using CSS/Bootstrap or Tailwind CSS.

b. Backend (Server-Side) – Node.js with Express.js

- Handles business logic and routing.

- Implements RESTful APIs for CRUD operations on blogs, users, materials, and communities.
- Manages authentication using JWT (JSON Web Tokens).
- Uses bcrypt for password hashing to enhance security.
- Role-based access control is enforced on each route.

c. Database – MongoDB (NoSQL)

- Stores user data, blog content, study materials, event details, and community info.
- Collections include:
  - `users`: all user roles (students, teachers, admins)
  - `blogs`: blog content, status (pending/approved/rejected/published), timestamps, feedback
  - `materials`: uploaded study resources
  - `communities`: community name, code, member IDs, join requests
  - `events`: seminar/meeting/workshop data

3.2.1 Presentation Layer (Frontend)

The Presentation Layer, also known as the Frontend, is the user-facing part of the application that interacts directly with the students, teachers, and admin. It is developed using React.js, a powerful JavaScript library known for its component-based architecture, reusability, and virtual DOM for fast rendering.

Key Responsibilities:

- Display content dynamically based on user roles (student, teacher, admin)
- Collect input from users and validate data before sending it to the backend
- Provide a responsive and intuitive user interface across all devices
- Handle routing and navigation using React Router
- Communicate with backend APIs for data fetching, updates, and submissions

Frontend Technologies Used:

Technology	Purpose
React.js	Building UI components
React Router	Client-side routing (navigation)
Axios / Fetch API	Communicating with backend APIs via HTTP calls
HTML5 / CSS3	Structure and styling of pages
Bootstrap	Responsive and modern UI design
JWT (via LocalStorage)	Manage authentication state

Table 3.1: Frontend Technologies Used

3.2.2 Application Layer (Backend)

Purpose:

The Application Layer, also known as the Backend, is responsible for handling all the business logic of the application. It processes user requests from the frontend, communicates with the database, applies validation, enforces role-based access control, and returns appropriate responses. This layer ensures secure, efficient, and consistent operations of all features such as blog management, user authentication, community workflows, and study material handling.

Technologies Used:

Technology	Purpose
Node.js	Server-side JavaScript runtime for scalable backend operations
Express.js	Framework to simplify routing and middleware-based logic
JWT (JSON Web Token)	Secure user authentication and session management
bcrypt.js	Encrypt passwords for secure storage
Mongoose	Object Data Modeling (ODM) to interact with MongoDB
MongoDB	NoSQL database to store application data
Multer (v1.4.3)	File upload handling
Google Generative AI (v0.24.1)	AI-based integration for chatbot functionality
CORS (v2.8.5)	Enables secure cross-origin requests
Dotenv (v10.0.0)	Environment variable management

Table 3.2: Backend Technologies Used

3.2.3 Data Layer (Database)

Purpose:

The Data Layer is responsible for storing, retrieving, and managing persistent data in the application. It plays a crucial role in maintaining the integrity, availability, and security of information related to users, blogs, communities, and study materials.

Technology Used:

Technology	Purpose
MongoDB	NoSQL database used to store structured data in flexible, JSON-like documents
Mongoose	ODM (Object Data Modeling) library for MongoDB, used to define schemas, models, and query logic

Table 3.3: Database Layer Technologies

Key Collections (Equivalent to Tables)

1. **Users Collection**
   Purpose: Stores information of all registered users (students, teachers, admins).
   Fields: `name, email, passwordHash, role, profileData`, etc.

2. **Blogs Collection**
   Purpose: Stores all blogs submitted by students.
   Fields: `title, content, summary, status (Pending/Approved/Rejected/Published), authorId, teacherFeedback, timestamps`.

3. **StudyMaterials Collection**
   Purpose: Contains uploaded documents and links shared by teachers.
   Fields: `title, fileURL, uploadedBy, subject, class, dateUploaded`.

4. **Events Collection**
   Purpose: Stores upcoming academic events added by teachers.
   Fields: `eventTitle, description, date, host, participants`.

5. **Communities Collection**
   Purpose: Manages data related to communities and membership.
   Fields: `communityName, communityCode, adminId, members, pendingRequests`.

3.3 Development Process

The development process of the website followed a structured and iterative approach to ensure quality, usability, and scalability. The website was developed using the **MERN** stack — comprising **MongoDB, Express.js, React.js,** and **Node.js** — which supports a modular and maintainable architecture.

The development was divided into multiple phases to systematically plan, design, implement, test, and deploy the application.

3.3.1 Requirement Analysis

Requirement analysis is a critical phase in the software development life cycle. It involves identifying, documenting, and analyzing the needs and expectations of the users and stakeholders to ensure the final product meets its intended goals. For this project, a role-based educational platform using the MERN stack, both functional and non-functional requirements were considered.

1. Functional Requirements

User Authentication and Roles

- Users must be able to register and log in securely.
- The system must support three types of users: Student, Teacher, and Admin.
- Role-based access must be enforced for feature accessibility.

Blog Management

- **Students should be able to:**
  - Submit blogs.
  - View feedback if rejected.
  - Edit and resubmit rejected blogs.
- **Teachers should be able to:**
  - View pending blogs.
  - Approve or reject with feedback.
- **Admin should:**
  - Review teacher-approved blogs.
  - Publish final blogs on the platform.

Study Materials

- Teachers should be able to upload study materials (PDF, links, documents).
- Students should be able to view/download study materials.

Event Management

- Teachers should be able to create academic events.
- Events should be visible to students with date, time, and description.

Community Management

- **Admin should be able to:**

  - Create communities with unique codes.
  - View and approve/reject student join requests.

- **Students should:**
  - Join communities using a code.
  - Send join requests if they do not have a code.

Chatbot Feature

- Students can interact with a basic chatbot for assistance (optional feature).

2. Non-Functional Requirements

Security

- Passwords must be hashed using bcrypt.
- Authentication should use JWT to ensure secure access.

Performance

- The system should load content quickly and handle multiple concurrent users efficiently.

Responsiveness

- The platform must be responsive and accessible across desktops, tablets, and smartphones.

Scalability

- The backend and database should be scalable to accommodate more users and features in the future.

Maintainability

- Code should be modular, well-documented, and easy to maintain or extend.

Data Integrity

- CRUD operations on blogs, materials, and users must preserve data consistency and validation.

3. Stakeholders

Stakeholder	Interest / Interaction
Students	Blog submission, access to study material and events, joining or requesting to join communities.
Teachers	Reviewing and moderating blogs (approval/rejection with feedback), uploading study materials, and posting events.
Admins	Managing users (students and teachers), publishing blogs, creating and moderating communities.

Table 3.4: Stakeholders and Their Roles in the System

3.4 Tools and Technologies Used

The educational platform designed using the **MERN** stack integrates a wide range of tools and technologies to ensure a modern, responsive, and scalable web application. Each technology was carefully chosen to fulfill specific development, deployment, and management needs while supporting high performance and modular architecture.

Core Technologies Used

Category	Technology	Purpose
Frontend	React.js	Build interactive UI components and handle client-side routing
Backend	Node.js + Express.js	Develop server-side logic and create RESTful APIs
Database	MongoDB + Mongoose	Store and manage data in NoSQL format with schema validation and data modeling

Table 3.5: Technologies Used in Different Layers of the System

3.4.1 Programming Languages

The educational platform is developed using JavaScript across both frontend and backend, following the MERN (MongoDB, Express.js, React.js, Node.js) stack. On the frontend, HTML and CSS are used for structuring and styling web pages, while JavaScript and JSX enable dynamic and interactive user interfaces through React.js. AJAX is utilized for seamless asynchronous communication with the backend. On the server side, Node.js with Express.js handles routing, business logic, and API creation. For database communication, MongoDB Query Language (MQL) and Mongoose are used to define schemas and perform CRUD operations. Additionally, JWT is used for secure authentication and bcrypt.js for password hashing, ensuring robust security.

3.4.2 Database Management

The platform uses MongoDB, a NoSQL document-oriented database, to manage and store data such as user profiles, blogs, study materials, event details, community memberships, and join requests. MongoDB, paired with Mongoose (an Object Data Modeling library), allows for flexible schema design and efficient handling of nested or relational-like data structures. This setup enables fast CRUD operations, seamless integration with Node.js, and easy scalability for future feature expansion. Data is stored in collections such as Users, Blogs, Materials, Events, and Communities, supporting role-based access and real-time content updates.

Chapter 4

IMPLEMENTATION

The educational platform was implemented using the MERN stack. The frontend was built with React.js for a responsive user interface, styled using Tailwind CSS. The backend was developed using Node.js and Express.js to create APIs for blog management, authentication, study material uploads, and community features. MongoDB, with Mongoose, was used for flexible and scalable data storage. JWT handled secure logins, and Postman was used to test API endpoints. The application was deployed using Vercel (frontend), Render (backend), and MongoDB Atlas (database), resulting in a complete, user-friendly, and scalable solution.

Frontend Implementation

The frontend, developed using React.js, offers an intuitive and responsive user interface. Students can register and log in, access dashboards, upload blogs, request to join communities, view study materials, and use summarization tools. Teachers access a separate dashboard to upload study materials, review and approve/reject student blogs with feedback, and create events. Admins manage users, review teacher-approved blogs for final publishing, create communities, and accept/reject community join requests. The user experience was enhanced through the use of CSS, Bootstrap, and React Bootstrap for component-based state management.

Backend Implementation

The backend was built using Node.js with Express.js, handling all API requests and enforcing business logic. It processes blog submissions, manages user sessions, handles authentication using JWT (JSON Web Tokens), and routes data securely between the frontend and MongoDB. The blog approval workflow is a key backend feature—student blogs are stored in a "pending" state, teachers review them and update their status accordingly. Admins have the final authority to publish the blog after reviewing teacher feedback. Role-based access control ensures secure data handling and prevents unauthorized operations.

Database Implementation

Data persistence is handled using MongoDB, a NoSQL document-based database. Collections are defined for users, blogs, study materials, communities, events, and join requests. Each entry includes metadata such as timestamps, creator identity, approval status, and content categorization. The database schema is optimized for performance and retrieval efficiency, supporting real-time updates and ensuring data consistency across all modules.

Community and Content Management

A community feature was implemented where admins can create communities, and students can join either through a unique code or by sending a request. Join requests are reviewed and accepted/rejected by the admin. Study material uploads and event announcements by teachers are stored in respective collections and rendered dynamically based on user roles. All content undergoes a verification workflow to ensure quality and prevent inappropriate uploads.

4.1 Secure Database Management

Secure database management in ShikshaHub is achieved through the use of MongoDB Atlas, a cloud-based NoSQL database service that offers built-in security features such as encryption at rest and in transit, IP whitelisting, and user access controls. All user data—including blog content, study materials, event details, authentication credentials, and community memberships—is stored in a well-structured, role-based schema using Mongoose, which enforces data validation and integrity. Sensitive data such as passwords are encrypted using bcrypt.js, and access to private routes and data is managed using JSON Web Tokens (JWT) to ensure only authorized users can access or modify content. The backend APIs are protected with authentication middleware, and all communication between the frontend and backend is handled securely using HTTPS protocols. Regular backups, secure environment variables using dotenv, and proper access control policies help maintain data confidentiality, integrity, and availability, ensuring the safety of all user-related information on the platform.

4.2 Technology Stack

• HTML, CSS, and JavaScript: These form the foundation of the user interface, ensuring it is well-structured, styled, and interactive. They enable the platform to deliver a responsive and user-friendly experience across all devices.

• React.js: A JavaScript library used for building dynamic and reusable user interface components. It allows for efficient rendering and smooth navigation between different user roles like students, teachers, and admins.

• Node.js: A JavaScript runtime used on the server side to handle requests, execute backend logic, and maintain scalability of the application.

• Express.js: A lightweight Node.js framework used to define RESTful APIs, manage routing, and implement middleware for handling authentication and data requests.

• MongoDB: A NoSQL database used for storing user data, blogs, study materials, events, and community-related information in a flexible and scalable manner.

• Mongoose: An ODM (Object Data Modeling) library for MongoDB, used to define schemas, validate data, and manage relationships between different collections.

• AJAX: Facilitates asynchronous communication between the frontend and backend, enabling real-time interactions such as blog submission, content updates, and notifications without full page reloads.

• JWT (JSON Web Tokens): Ensures secure, token-based user authentication and

access control across different parts of the platform.

• bcrypt.js: Used for securely hashing user passwords before storing them in the database, enhancing data protection and preventing unauthorized access.

• Bootstrap: A popular CSS framework that provides pre-designed components and responsive grid systems for building visually consistent web interfaces.

• React Bootstrap: A library that offers Bootstrap-styled components built specifically for React, enabling seamless integration and faster UI development.

4.3 Database Implementation

4.3.1 Databse Tables

1.User Collection

Field Name	Datatype	Description
_id	ObjectId	Unique ID for each user
name	String	Name of the user
email	String	Email address (used for login)
password	String	Encrypted password
role	String	Role of the user (student/teacher/admin)
communities	Array	List of community IDs the user has joined or requested to join
createAt	Date	The date and time when the user account was created

Table 4.1: User Table Schema

2. Blogs Collection

Field Name	Data Type	Description
_id	ObjectId	Unique ID for each blog
title	String	Title of the blog
content	String	Blog body content
community	ObjectId	Community ID to which the blog belongs
status	String	Blog status (pending/approved/rejected/published)
author	ObjectId	Reference to the user who wrote the blog
authorRole	String	Role of the author: student or teacher
isOriginalContent	Boolean	Indicates whether the content is original
tags	Array of Strings	Keywords or tags related to the blog topic
createdAt	Date/Timestamp	Date and time when the blog was created
updatedAt	Date/Timestamp	Last updated time of the blog post
__v	Number	Version key used by MongoDB for internal versioning
reviewComment	String	Feedback provided by the teacher/admin during review
reviewedAt	Date/Timestamp	Time when the blog was reviewed
reviewedBy	ObjectId	ID of the teacher or admin who reviewed the blog

Table 4.2: Blog Table Schema

3. Study Material Collection

Field Name	Data Type	Description
_id	ObjectId	Unique ID for each study material
title	String	Title of the material
description	String	Short description of the material
fileUrl	String	URL link to the uploaded file
fileType	String	Type of file (PDF, DOCX, PPT, etc.)
community	ObjectId	ID of the community the material belongs to
author	ObjectId	ID of the author (if different from uploader)
authorRole	String	Role of the author (e.g., teacher or admin)
tags	Array of Strings	List of keywords/tags related to the material
createdAt	Date	Date and time of creation
uploadDate	Date	Date on which the material was uploaded
__v	Number	Version key used for internal document versioning

Table 4.3: Study Material Collection Schema

4. Classroom Collection

Field Name	Data Type	Description
_id	ObjectId	Unique identifier for each community
name	String	Name of the community
description	String	Brief description of the community
joinCode	String	Unique code used by students to join the community
admin	ObjectId	ID of the admin who created the community
teachers	Array of ObjectIds	List of teachers in the community
students	Array of ObjectIds	List of students in the community
materials	Array of ObjectIds	References to study materials shared in the community
blogs	Array of ObjectIds	References to blogs shared in the community
joinRequests	Array of ObjectIds	References to pending join requests
createdAt	Date	Timestamp of when the community was created
__v	Number	Internal versioning field for the document

Table 4.4: Classroom Collection Schema

5. Chat Session Collection

Field Name	Data Type	Description
_id	ObjectId	Unique identifier for each chat session
user	ObjectId	ID reference to the user participating in the session
messages	Array	List of messages exchanged during the session
category	String	Category/topic of the chat (e.g., ”study”, ”doubt”)
isActive	Boolean	Indicates whether the chat session is active
createdAt	Date/Timestamp	Date and time when the chat session was created
lastActivity	Date/Timestamp	Timestamp of the most recent activity in the session
__v	Number	Versioning key for the document (MongoDB internal use)

Table 4.5: Chatsession Collection Schema

Chapter 5

Result and Discussion

5.1 Web Pages

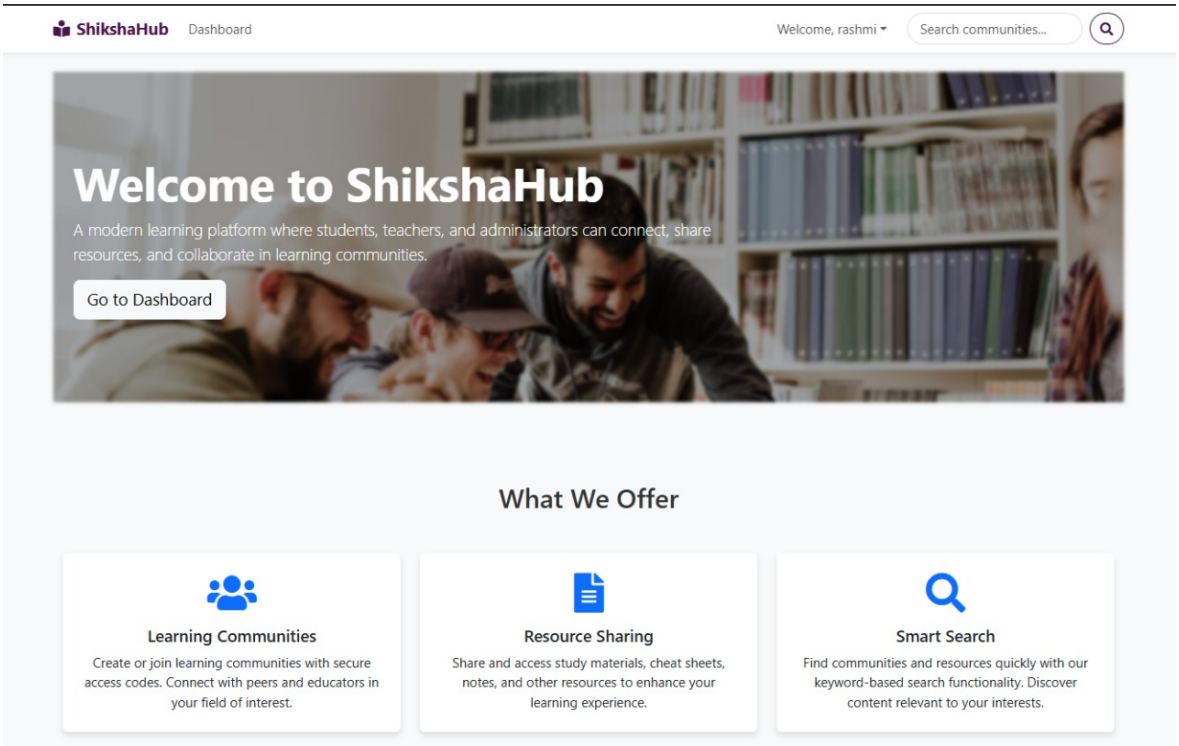


Figure 5.1: Home page

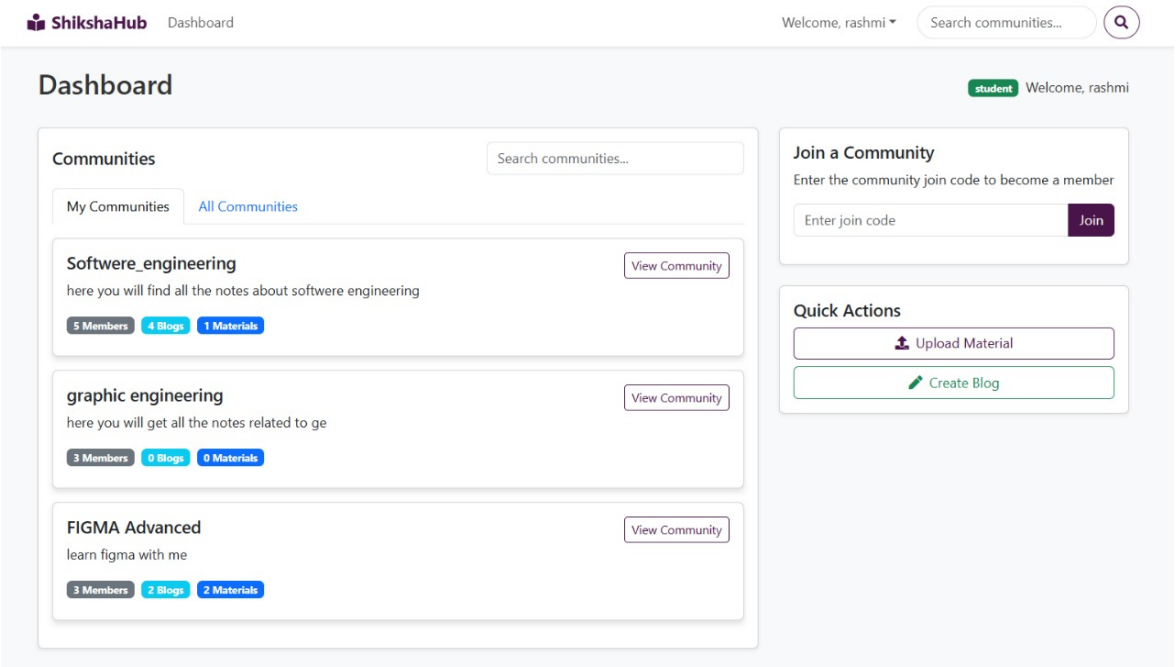


Figure 5.2: website dashboard

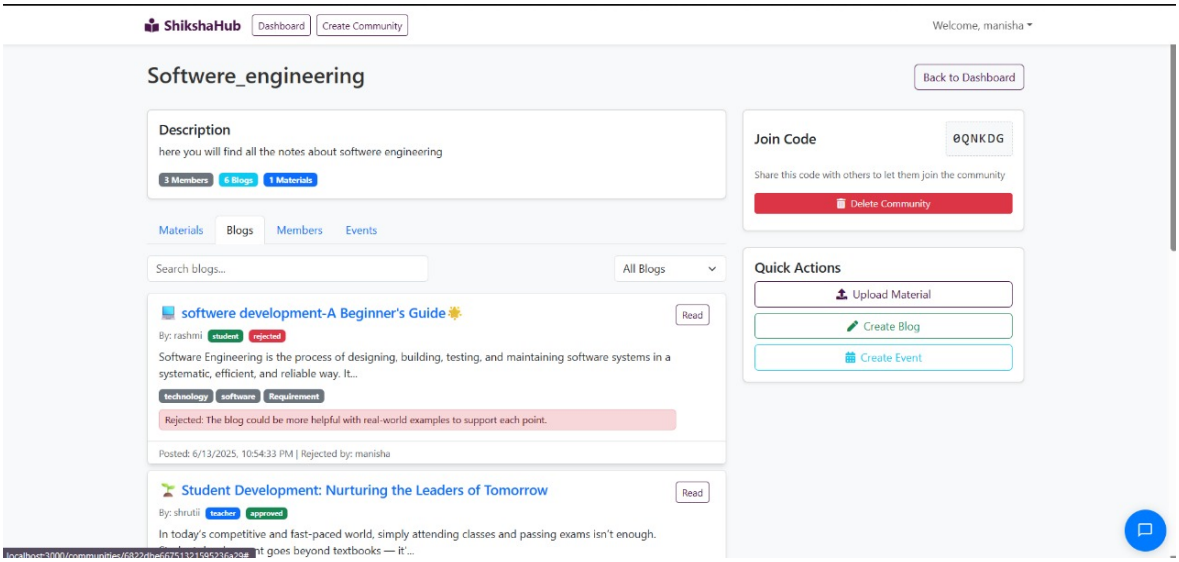


Figure 5.3: Blog Management Interface

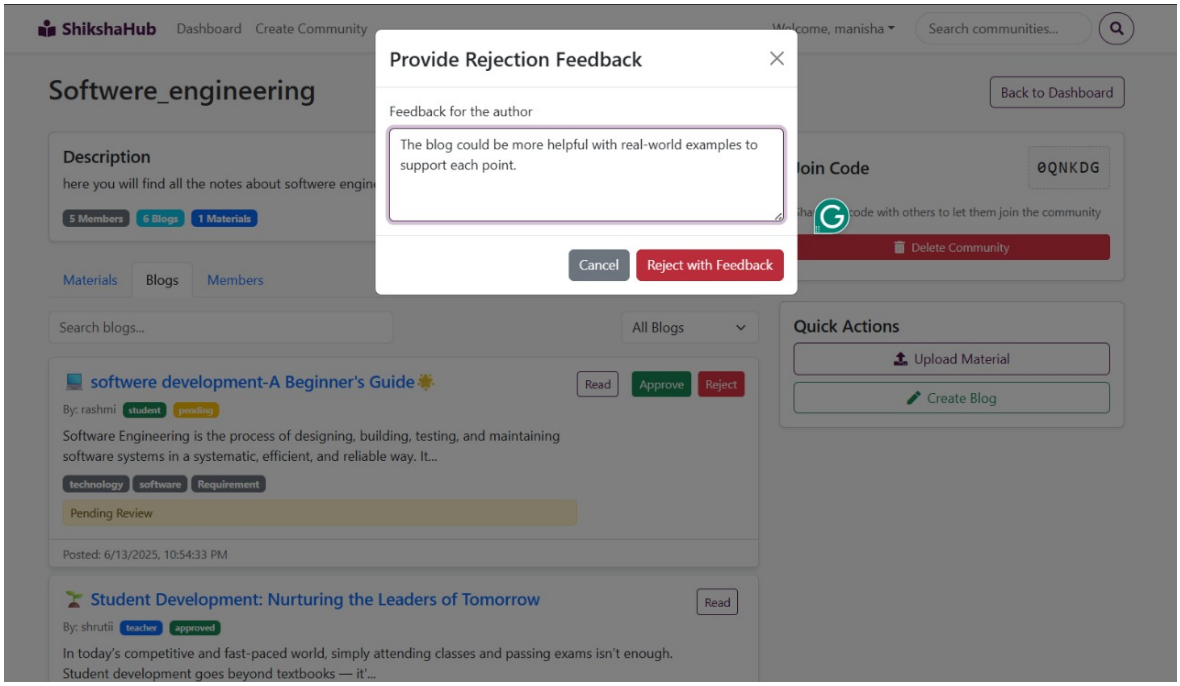


Figure 5.4: Feedback page

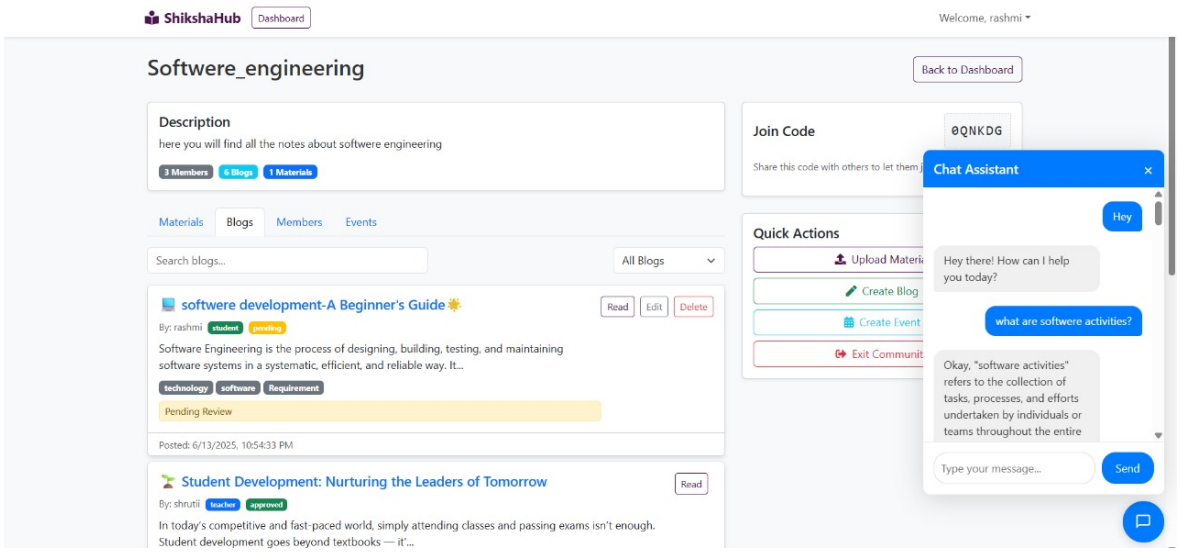
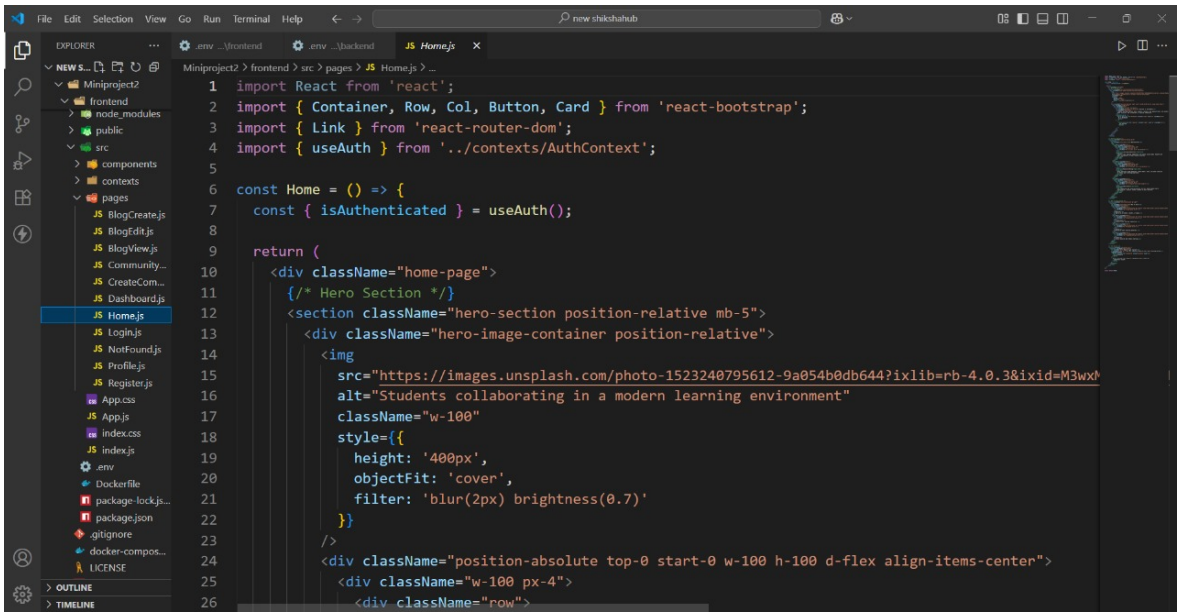


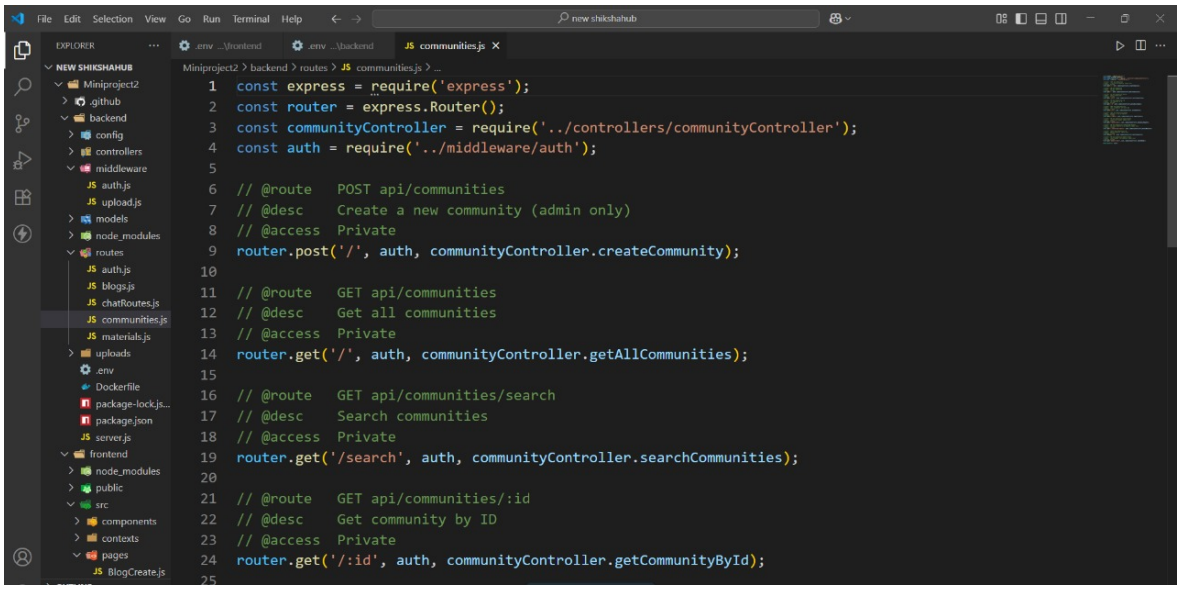
Figure 5.5: Chatbot assistence

5.2 Code Snippets



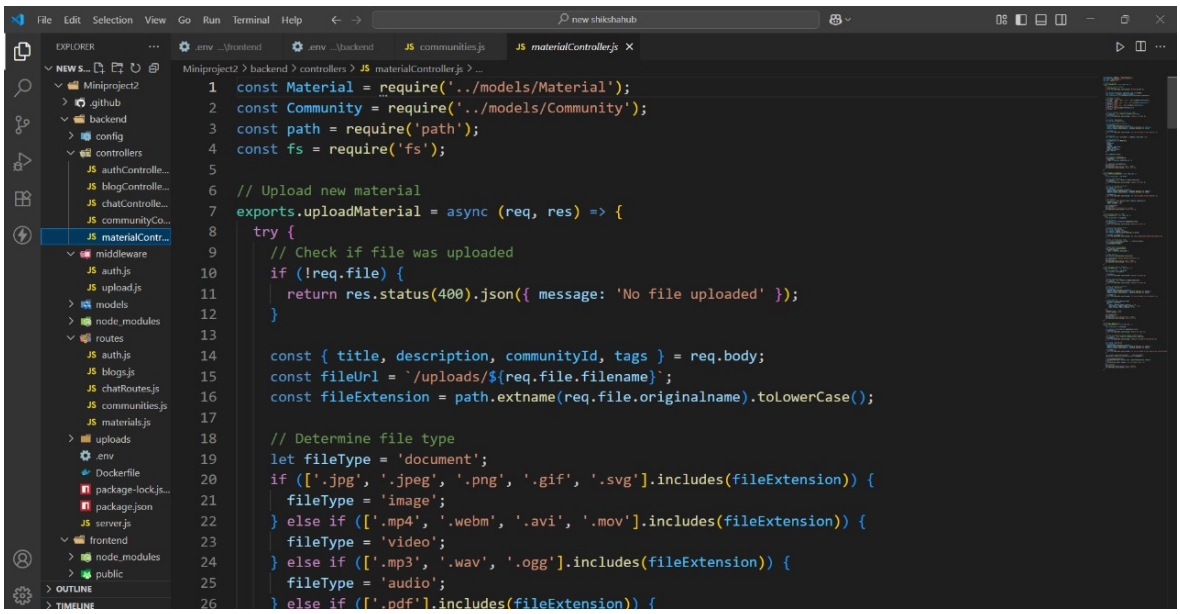
```
import React from 'react';
import { Container, Row, Col, Button, Card } from 'react-bootstrap';
import { Link } from 'react-router-dom';
import { useAuth } from '../contexts/AuthContext';

const Home = () => {
  const { isAuthenticated } = useAuth();

  return (
    <div className="home-page">
      {/* Hero Section */}
      <section className="hero-section position-relative mb-5">
        <div className="hero-image-container position-relative">
          <img
            src="https://images.unsplash.com/photo-1523240795612-9a054b0db644?ixlib=rb-4.0.3&ixid=M3wx
            alt="Students collaborating in a modern learning environment"
            className="w-100"
            style={{
              height: '400px',
              objectFit: 'cover',
              filter: 'blur(2px) brightness(0.7)'
            }}
          />
          <div className="position-absolute top-0 start-0 w-100 h-100 d-flex align-items-center">
            <div className="w-100 px-4">
              <div className="row">
```

Figure 5.6: Home



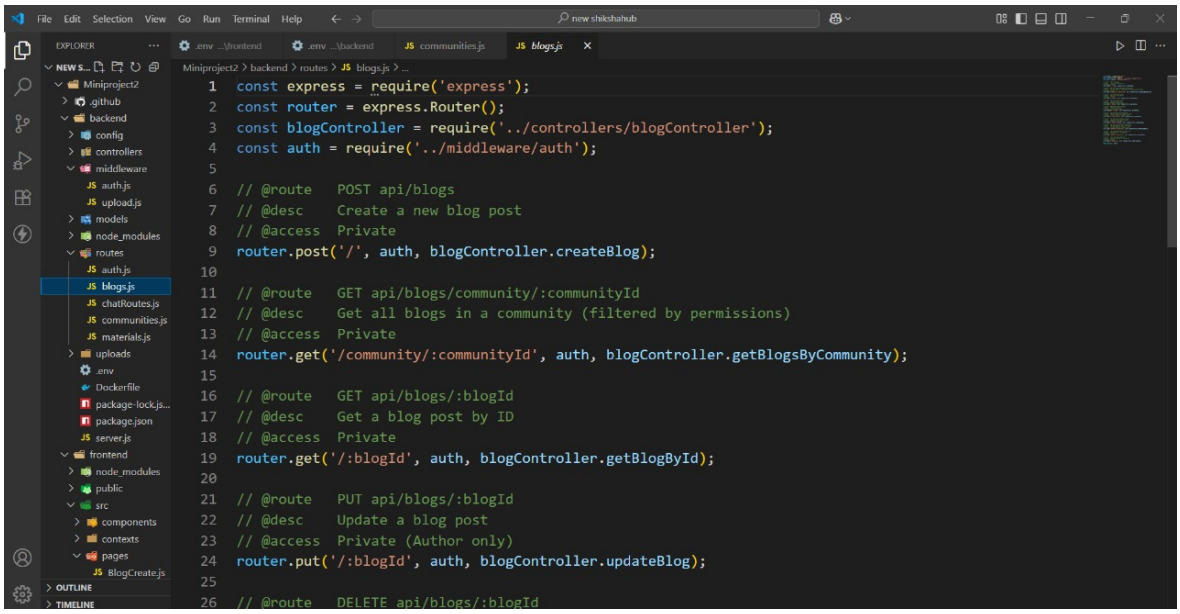
```
const express = require('express');
const router = express.Router();
const communityController = require('../controllers/communityController');
const auth = require('../middleware/auth');

// @route   POST api/communities
// @desc    Create a new community (admin only)
// @access  Private
router.post('/', auth, communityController.createCommunity);

// @route   GET api/communities
// @desc    Get all communities
// @access  Private
router.get('/', auth, communityController.getAllCommunities);

// @route   GET api/communities/search
// @desc    Search communities
// @access  Private
router.get('/search', auth, communityController.searchCommunities);

// @route   GET api/communities/:id
// @desc    Get community by ID
// @access  Private
router.get('/:id', auth, communityController.getCommunityById);
```

Figure 5.7: Communities



```
const Material = require('../models/Material');
const Community = require('../models/Community');
const path = require('path');
const fs = require('fs');

// Upload new material
exports.uploadMaterial = async (req, res) => {
  try {
    // Check if file was uploaded
    if (!req.file) {
      return res.status(400).json({ message: 'No file uploaded' });
    }

    const { title, description, communityId, tags } = req.body;
    const fileUrl = `/uploads/${req.file.filename}`;
    const fileExtension = path.extname(req.file.originalname).toLowerCase();

    // Determine file type
    let fileType = 'document';
    if (['.jpg', '.jpeg', '.png', '.gif', '.svg'].includes(fileExtension)) {
      fileType = 'image';
    } else if (['.mp4', '.webm', '.avi', '.mov'].includes(fileExtension)) {
      fileType = 'video';
    } else if (['.mp3', '.wav', '.ogg'].includes(fileExtension)) {
      fileType = 'audio';
    } else if (['.pdf'].includes(fileExtension)) {
```

Figure 5.8: MaterialController



```
const express = require('express');
const router = express.Router();
const blogController = require('../controllers/blogController');
const auth = require('../middleware/auth');

// @route   POST api/blogs
// @desc    Create a new blog post
// @access  Private
router.post('/', auth, blogController.createBlog);

// @route   GET api/blogs/community/:communityId
// @desc    Get all blogs in a community (filtered by permissions)
// @access  Private
router.get('/community/:communityId', auth, blogController.getBlogsByCommunity);

// @route   GET api/blogs/:blogId
// @desc    Get a blog post by ID
// @access  Private
router.get('/:blogId', auth, blogController.getBlogById);

// @route   PUT api/blogs/:blogId
// @desc    Update a blog post
// @access  Private (Author only)
router.put('/:blogId', auth, blogController.updateBlog);

// @route   DELETE api/blogs/:blogId
```

Figure 5.9: Blogs

Chapter 6

Conclusion and Future Scope

6.1 Conclusion

Our proposed learning and blogging platform is an encouraging and structured environment for students, teachers, and admins. Incorporating blog submission, AI summary, study guides, and event organization, the platform enables learning and fosters sharing of knowledge. With a multi-step approval process, quality is assured for content, with the students receiving constructive feedback before they publish their articles. The role-based model makes platform functions simple, with efficient and convenient working assured for everyone. Long-Term Impact: This website empowers students, assists teachers, and allows admins to ensure content quality, which makes it a pioneering solution for educational blogging and e-learning.

6.2 Future scope

The ShikshaHub platform has strong potential for future enhancement and scalability. In the coming phases, advanced features such as AI-based blog summarization, real-time chat support, and intelligent recommendation systems for study materials and blogs can be integrated to improve user engagement. The addition of mobile application support using React Native can further extend the platform's accessibility. Enhanced analytics dashboards for teachers and admins can offer insights into student performance and content reach. Role-based permissions can be expanded to support sub-admins or moderators. Furthermore, integration with third-party educational APIs, cloud storage for large files, and multilingual support can help reach a broader audience. With continuous improvements, ShikshaHub can evolve into a comprehensive educational ecosystem that supports collaboration, content sharing, and personalized learning experiences at scale. and allow users to manage their auctions and wallet balances on the go. The platform can also be expanded to support a wider variety of agricultural products and integrate more payment gateways for greater flexibility in transactions. Additionally, incorporating AI-driven bidding strategies and advanced fraud detection mechanisms would improve the overall user experience and platform security.

References

[1] S. Sharma and M. Agarwal, "Role-Based Access Control in Web Applications," *Int. J. Comput. Appl.*, vol. 178, no. 7, pp. 10–14, 2019.

[2] M. A. Khan, "Educational Web Portals: A Comparative Study," *Int. J. Educ. Dev. Using ICT*, vol. 15, no. 2, pp. 120–135, 2020.

[3] P. J. Kamthan, "Effective Use of MongoDB for Scalable Web Applications," *J. Database Manag.*, vol. 33, no. 1, pp. 23–32, 2021.

[4] A. Mishra and S. Bansal, "MERN Stack Development: A Modern Approach to Full-Stack Web Applications," *Int. J. Innov. Res. Comput. Commun. Eng.*, vol. 9, no. 6, pp. 3010–3015, 2021.

[5] Google, "Google Classroom," [Online]. Available: https://classroom.google.com/. [Accessed: Jun. 9, 2025].

[6] Medium, "The MERN Stack – A Full-Stack Guide," *Medium.com*, [Online]. Available: https://medium.com/@developer/mern-stack-guide. [Accessed: Jun. 9, 2025].