A
Mini Project Report
on

"ShikshaSetu- Digital Education Platform"

Submitted in partial fulfillment of the requirements of the Third Year of Engineering in Computer Science & Engineering (AI&ML)

By

Sumeet Monde	A-24
Shravan Parthe	A-28
Tanmay Rasal	B-31
Soham Satpute	B-54

Under the Guidance of

Prof. Hemant Kamble

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING (AI&ML)

VISHWANIKETAN'S INSTITUTE OF MANAGEMENT ENTREPRENEURSHIP & ENGINEERING TECHNOLOGY

UNIVERSITY OF MUMBAI
(2025-2026)

VISHWANIKETAN'S INSTITUTE OF MANAGEMENT

ENTREPRENEURSHIP & ENGINEERING TECHNOLOGY

(Affiliated to University of Mumbai, Approved by A.I.C.T.E., New Delhi)

Department Of Computer Science & Engineering (AI&ML)

Certificate

This is to certify the Mini Project entitled "**SikshaSetu-Digital Education Platform**" is a bonafide work of "**Shravan Parthe, Sumeet Monde, Tanmay Rasal, Soham Satpute**" submitted in partial fulfillment of the requirements of the Bachelor of Engineering in Computer Science & Engineering (AI&ML).

Prof . Hemant Kamble

Guide

Dr. Ankush B. Pawar

Head of Department, CSE(AI&ML)

Dr. B. R. Patil

Principal, ViMEET

VISHWANIKETAN'S INSTITUTE OF MANAGEMENT ENTREPRENEURSHIP & ENGINEERING TECHNOLOGY

(Affiliated to University of Mumbai, Approved by A.I.C.T.E., New Delhi.

Department Of Computer Science & Engineering (AI&ML)

Project Report Approval

This Mini-project report entitled **"SikshaSetu-Digital Education Platform"** by "**Sumeet Monde (A-24) , Shravan Parthe (A-28) , Tanmay Rasal (B-31) , Soham Satpute (B-54)"**is approved for the Third Year/Sem V of Computer Science & Engineering (AI&ML).

Examiners

1...

2...

Date.

Place.

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, We have adequately cited and referenced the original sources. We also declare that We have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my/our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Sumeet Monde (A-24)	--
Shravan Parthe (A-28)	--
Tanmay Rasal (B-31)	--
Soham Satpute (B-54)	--

Date:

Acknowledgement

We, "**Shravan Parthe, Sumeet Monde, Tanmay Rasal, and Soham Satpute**", express our profound gratitude to **Prof. Hemant Kamble** for his expert guidance, constant encouragement, and invaluable support throughout the development of this project — from its inception to successful completion. His insights and motivation have been instrumental in helping us achieve our objectives effectively.

We would also like to extend our heartfelt appreciation to **Dr. B. R. Patil**, Principal, **Vishwaniketan's Institute of Management Entrepreneurship and Engineering Technology**, and **Dr. Ankush B. Pawar**, Head of the Department of **Computer Science and Engineering (Artificial Intelligence and Machine Learning)**, for their continuous support, valuable suggestions, and for providing us with the necessary facilities to complete this project successfully.

Lastly, we sincerely thank all the faculty members and staff of the CSE(AIML) who directly or indirectly guided and supported us throughout this project journey. Their encouragement and cooperation have played a vital role in the successful completion of our work.

ABSTRACT

SikshaSetu – Smart Education Management and Virtual Learning Platform aims to revolutionize institutional management by integrating academic, administrative, and virtual learning functionalities into a unified digital ecosystem. The system provides seamless management of students, faculty, branches, attendance, results, notes, and events while incorporating advanced modules such as AI-assisted query support and real-time virtual classrooms powered by ZegoCloud.

The platform enhances transparency, accessibility, and collaboration across educational stakeholders through role-based authentication, ensuring secure access for administrators, faculty, and students. By leveraging modern technologies such as Spring Boot, React, MySQL, JWT, and ZegoCloud SDK, the system achieves efficient data handling, smooth user interaction, and reliable video communication.

In addition to its management capabilities, SikshaSetu integrates AI-driven assistance using the Perplexity API, providing instant academic support and intelligent content delivery. This combination of automation and interactivity optimizes academic workflows, reduces manual overhead, and fosters an engaging digital learning experience. The project contributes to the evolution of smart education by bridging traditional classroom management with next-generation virtual learning and intelligent analytics tools, ensuring scalability, usability, and real-time performance in institutional environments.

List of Tables

List of Figures

Chapter 1

INTRODUCTION

1.1 Introduction and Motivation

1.1.1 Introduction

Education in rural areas faces significant challenges including limited infrastructure, scarce resources, and inadequate access to quality teaching. This project develops a rural education platform designed to bridge these gaps using technology tailored for resource-constrained settings. The platform offers a virtual classroom powered by ZegoCloud, attendance and academic tracking, and a resource management system delivering curriculum-aligned content based on each student's academic board, standard, and year. Additionally, AI features built with Spring AI support teachers in content creation and provide personalized content recommendations, all while strictly adhering to the official curriculum to ensure accuracy and relevance. This solution aims to empower rural students and educators by providing accessible, effective, and personalized learning experiences, fostering educational equity across diverse rural communities.

1.1.2 Motivation

Access to quality education remains a critical challenge in rural India due to inadequate infrastructure, scarce teaching resources, and limited connectivity. Many students face obstacles that hinder their ability to learn effectively and reach their full potential. This project is motivated by the pressing need to bridge the digital and educational divide faced by rural communities. By leveraging technology tailored to the unique constraints and needs of these areas, the platform aims to provide an inclusive, scalable, and personalized learning environment. The integration of virtual classrooms, curriculum-specific resources, and AI-assisted content creation will empower students and teachers alike, improving educational outcomes and opening pathways to lifelong learning. Ultimately, this initiative seeks to foster equitable educational opportunities and contribute to the socio-economic upliftment of rural populations through accessible and quality education.

1.2 Existing System

Existing rural education platforms have emerged to address the deep-rooted challenges faced by students in underserved areas, including limited infrastructure, shortage of qualified teachers, and socio-economic barriers. Platforms like Skoodos Bridge, eVidyaloka, and government initiatives such as DIKSHA strive to democratize access to quality education by leveraging digital tools. These systems enable students in remote locations to access expert teaching, interactive learning resources, and personalized guidance that were previously unavailable.

The motivation behind these solutions lies in bridging the educational disparity between urban and rural regions, ensuring that geography does not determine the quality of learning. They also overcome teacher shortages by connecting students with qualified educators through virtual classrooms. Affordable and localized content offerings further ensure inclusivity across linguistic and resource constraints. By empowering rural learners with digital literacy and flexible learning modalities, these platforms contribute to better academic outcomes and socio-economic upliftment, paving the way for a more equitable education landscape in India.

1.3 Problem Statement

Rural education in India faces critical challenges that limit students' access to quality education and hinder their academic growth. These challenges include inadequate infrastructure such as lack of classrooms, sanitation, electricity, and learning materials. Many rural schools suffer from shortages of qualified teachers, especially in key subjects like mathematics and science, resulting in compromised teaching quality. Socio-economic factors, including poverty and gender disparities, further restrict enrollment and consistent attendance. Additionally, limited digital access and language barriers exacerbate educational inequities, preventing rural students from benefiting fully from modern learning methods and resources.

These obstacles collectively contribute to low learning outcomes and restricted opportunities for rural children. Despite government initiatives and the rising usage of technology, significant gaps remain in providing personalized, curriculum-aligned education tailored to the needs of rural students. This calls for innovative, scalable solutions that leverage digital tools while addressing infrastructure limitations and contextual challenges to democratize education and foster inclusive growth in rural India.

1.4 Objectives

1. To improve access to quality education in rural areas by leveraging technology tailored to local infrastructure limitations.
2. To enable interactive virtual classrooms using ZegoCloud, facilitating real-time learning despite geographic barriers.
3. To deliver curriculum-aligned resources and notes customized to students' academic board, standard, and year for personalized learning.
4. To implement efficient attendance and academic performance tracking that supports data-driven improvements.
5. To integrate AI-assisted content creation and recommendation using Spring AI, ensuring accurate, curriculum-focused support for educators and learners.

1.5 Scope

1. The platform will serve rural schools and students across diverse Indian states, focusing on Classes 1 to 10 and multiple academic boards (CBSE, ICSE, State boards).
2. It offers virtual classrooms via ZegoCloud, supporting real-time live teaching optimized for low-bandwidth rural connectivity.
3. Attendance tracking, academic performance management, and resource/notes distribution personalized by each student's curriculum context (board, standard, academic year) will be included.
4. AI-enabled tools using Spring AI will assist teachers with syllabus-aligned content generation and offer tailored content recommendations to students, ensuring accuracy and relevancy.
5. Educational materials such as lesson notes, quizzes, and resources will be managed locally to accommodate infrastructure constraints, avoiding reliance on cloud storage.
6. Real-time and historical analytics dashboards will provide actionable insights for educators and administrators on student engagement, learning progress, and infrastructure status.
7. The system will be scalable and maintainable, implemented with React frontend and Spring Boot backend architecture, adaptable to evolving rural education needs and technology improvements.

.

1.6 Proposed System

The proposed system is a comprehensive rural education platform designed to bridge educational gaps in remote areas by harnessing technology while adapting to infrastructure limitations.

1. **Virtual Classroom:** It integrates live interactive classes powered by ZegoCloud, allowing students and teachers to engage in real-time sessions optimized for low-bandwidth rural settings without recording functionalities, supplemented by note-sharing.
2. **Attendance and Academic Tracking:** It incorporates systematic attendance marking and comprehensive academic performance tracking, enabling teachers and administrators to monitor student progress effectively.
3. **AI-Powered Content Assistance:** Using Spring AI with tightly controlled prompt engineering, the system offers syllabus-compliant content generation tools to support educators in creating quizzes, summaries, and recommendation engines, mitigating misinformation risks.
4. **Resource Management with Local Storage:** Due to connectivity constraints, educational materials and session notes are managed and stored locally rather than relying on cloud storage, ensuring accessibility and reliability in rural settings.
5. **Analytics and Reporting:** The system features dashboards and analytical tools that provide insights into student engagement, learning outcomes, and infrastructure health to assist data-driven decision-making.
6. **Layered and Modular Architecture:** Built on React frontend and Spring Boot backend structured into model, repository, service, and controller layers, the system ensures scalability, maintainability, and security tailored to rural educational needs.

Chapter 2

LITRATURE REVIEW

2.1 Secondary Research

Secondary research for this project involves analyzing existing literature, reports, and implementations related to rural education, education technology, and AI integration in educational platforms, particularly in the Indian context. This research provides a foundational understanding of the challenges, current solutions, and best practices to inform the design and implementation of the proposed rural education platform.

Key areas explored include:

1. **Challenges in Rural Education:** Extensive studies reveal persistent barriers such as poor infrastructure, lack of qualified teachers, socio-economic constraints, and limited access to quality learning materials that hinder rural students' academic success. Reports also highlight the digital divide impacting rural areas, emphasizing the need for solutions tailored to low connectivity and resource availability.

2. **Existing Rural Education Initiatives:** Successful government programs like DIKSHA and NGO-led projects such as eVidyaloka demonstrate the potential of technology-enabled education to reach underserved populations. These platforms leverage digital classrooms, multilingual content, and volunteer-based teaching models to expand educational access.
3. **Virtual Classroom Technologies:** Research into virtual classroom solutions shows a growing trend in adopting platforms powered by WebRTC and cloud services like ZegoCloud to enable real-time, interactive remote teaching. These systems address infrastructure challenges by optimizing bandwidth and providing rich engagement tools.
4. **AI in Education:** Literature on AI-driven educational technologies highlights the effectiveness of AI-assisted content creation, personalized learning recommendations, and automated assessment in improving instructional quality and learner engagement. Studies stress the importance of strict curriculum alignment and prompt engineering to avoid misinformation when deploying AI in sensitive learning contexts.
5. **Technology Adoption and Impact:** Reviews of edtech adoption in rural India confirm improvements in digital literacy and learning outcomes when solutions are context-aware and include stakeholder training. Impact assessments underscore the value of integrated platforms that combine classroom interaction, resource management, and data analytics for systemic educational improvements.

This secondary research informs the project’s objectives, scope, and design choices, ensuring the platform addresses real-world needs with validated strategies and cutting-edge technology adapted for rural educational environments.

2.2 Primary Research

Primary research for this project involves collecting original data and insights directly from the target stakeholders and environments to better understand the specific educational challenges, needs, and opportunities within rural areas. This research helps tailor the platform's design and features to the realities faced by rural students, teachers, and administrators.

Key components of the primary research include:

1. **Field Surveys and Interviews:**
    - Conduct surveys with rural students, teachers, and school administrators to gather data on current educational infrastructure, availability of teaching resources, and digital literacy levels.
    - Interviews with educators to identify pain points in classroom management, content delivery, and student engagement.
    - Assess technological readiness, including availability and reliability of internet connectivity, access to devices, and familiarity with digital tools.
2. **Focus Group Discussions:**
    - Facilitate focus groups involving community members, parents, and local education officers to understand cultural, social, and economic factors impacting education.
    - Explore perceptions of technology in education and barriers to adoption.
3. **Observation Studies:**
    - Direct observation of rural classrooms to evaluate learning environments, teaching methods, student participation, and resource usage.
    - Monitor connectivity conditions and infrastructure challenges affecting digital education implementation.
4. **Data Analysis:**
    - Analyze attendance records, academic performance, and resource utilization statistics obtained from pilot deployments.
    - Use this data to refine content recommendation algorithms and AI prompt designs.
    - This primary research approach ensures that the platform is grounded in the lived experiences and specific contexts of rural educational stakeholders, leading to a practical, adoptable, and impactful solution.

2.3 Brief

The proposed system is a comprehensive rural education platform designed to improve access, quality, and personalization of education for students in remote areas. It integrates:

1. **Virtual Classroom:** Powered by ZegoCloud, enabling real-time interactive learning tailored for low-bandwidth rural environments, focusing on live sessions and note sharing without recordings.

2. **Academic Context-Aware Content Delivery:** Provides students with curriculum-aligned resources and notes filtered by academic board (CBSE, ICSE, State), standard, and year, ensuring personalized and relevant learning material.

3. **Attendance and Academic Tracking:** Implements systematic tracking of student attendance and academic performance to help educators monitor progress and intervene as needed.

4. **AI-Assisted Content Creation:** Utilizes Spring AI integrated with Ollama to support teachers with syllabus-compliant content generation such as quizzes, summaries, and recommendations, minimizing misinformation.

5. **Resource Management:** Manages educational materials and session notes stored locally, addressing rural infrastructure constraints and guaranteeing content accessibility.

6. **Analytics and Reporting:** Offers actionable insights through dashboards for educators and administrators on student engagement, performance, and infrastructure health.

7. **Robust Architecture:** Built using React frontend with a layered Spring Boot backend (model, repository, service, controller), ensuring scalability, maintainability, and security suited for rural education needs.

This system aims to empower rural students and educators by bridging educational gaps through innovative, context-sensitive technology adoption.

2.4 Comparative Analysis of Existing System

Aspect	**Existing Systems**	**Proposed Platform**
Target Audience	Rural schools, national government initiatives	Rural Indian schools, Classes 1–10, multiple boards
Virtual Classroom	WebRTC-based, cloud-dependent	ZegoCloud-powered, optimized for low bandwidth, without recordings
Curriculum Alignment	Varied; mostly CBSE, State boards	Strict alignment with board, standard, academic year
AI Integration	Limited or pilot projects	Spring AI with controlled prompts for syllabus-aligned content creation
Resource Management	Cloud or volunteer-managed	Local storage, syllabus-filtered resources and notes
Attendance & Academic Tracking	Basic to moderate	Integrated, detailed with analytics and reporting
Analytics & Reporting	Basic usage statistics	Advanced dashboards for educators/admins

Table 1 – Comparative Analysis

Comparative Analysis Summary:

Existing rural education systems like eVidyaloka and DIKSHA have made significant strides in providing digital learning to underserved areas but often lack integrated AI support or detailed academic tracking aligned tightly to diverse curricula. Platforms such as BigBlueButton offer robust virtual classrooms but lack curriculum-specific content management and localized resource control needed for rural contexts.

The proposed system aims to combine the best aspects of these existing solutions scalable virtual classrooms, strict curriculum-aligned resource delivery, detailed attendance and performance tracking, and AI-assisted content generation while addressing critical gaps like local resource management and network optimization for rural India. This integrated, modular approach is designed specifically to empower rural learners and educators with relevant, accessible, and personalized education solutions.

2.5 Research Gap Analysis

The SikshaSetu Education Platform while modern and feature-rich demonstrates substantial gaps affecting its effectiveness, scalability, and accessibility. Security is compromised by lack of multi-factor authentication, missing token revocation and audit trails, and unaddressed vulnerabilities due to reliance on basic JWT and Spring Boot defaults. These gaps raise

compliance and breach risks, especially in environments requiring strong student data protection.

Its infrastructure, designed as a single-tenant monolith, depends entirely on stable internet connectivity. Without offline capabilities, progressive web app support, caching, or scalable

horizontal deployment, SikshaSetu cannot reliably support rural or high-volume institutional usage. Existing solutions in similar contexts establish offline synchronization and asset optimization as essential, which SikshaSetu currently lacks.

Learning analytics and AI integration are underdeveloped. The platform’s chatbot struggles with accuracy and context, and there is no system for tracking student engagement or predicting performance outcomes. Modern educational systems increasingly rely on comprehensive analytics and predictive models to intervene and personalize instruction, but such features are absent here.

The virtual classroom setup relies on ZegoCloud but omits features critical for modern digital education: session recording, breakout groups, screen sharing, and collaboration tools. Accessibility is further limited by a lack of closed captions, gesture recognition, and optimization for lower bandwidth leading to exclusion of students in under-connected regions.

Accessibility overall remains unaddressed; there is no evidence of standards compliance (WCAG 2.2 AA), support for assistive technologies, nor multi-language inclusivity. Data privacy protocols, regulation compliance, and mobile offline access are also unaccounted for, impacting the platform's reach and trustworthiness.

Assessment tools are basic there is no automation for grading, no question bank, no plagiarism detection, and analytics do not inform instructional strategies or adaptive feedback. Integration with the wider educational ecosystem is minimal: there is no connection to external systems, no standards compliance for data interchange, and the API lacks robust documentation and extensibility.

Performance, quality assurance, and operational reliability are limited by the absence of monitoring, error tracking, backup, disaster recovery, and automated test suites. These deficiencies result in elevated risks for downtime and data loss, especially during periods of high activity.

Chapter 3

REQUIREMENT ANALYSIS

3.1 Product Analysis and Market Research for Business Potential

Market Size & Growth Opportunity

- The education ERP market in India is projected to grow from **USD 0.8 million in 2024 to USD 3.4 million by 2030**, at a **CAGR of ~27.7%**.
- Globally, the education apps market was valued at **USD 6.01 billion in 2024** and is expected to reach **USD 33.51 billion by 2033**, growing at a **CAGR of ~21.0%**.
- Over **6,000+ ed-tech and school management startups** exist in India, collectively raising over **USD 4 billion** in investments.

Implication:
The education ERP and institutional management segment is in a high-growth phase. Increasing digital transformation in educational institutions, demand for hybrid learning, and the adoption of AI-enabled systems create a strong business opportunity.

Target Customers & Segments

- **Primary customers:** Schools, colleges, and universities seeking unified administrative and academic management systems.
- **Secondary customers:** Coaching centers and hybrid learning institutions that require both offline and online management tools.

Product–Market Fit:
The product aligns well with institutional needs—offering admin, faculty, and student management modules, integrated virtual classrooms, and an AI chatbot. It caters to a critical pain point: fragmentation of digital tools in educational operations. However, localization, affordability, and mobile optimization will be key for penetration in rural and semi-urban institutions.

Competitive Landscape & Differentiation

- The market includes strong competitors such as Teno, Teachmint, School TMS, and Fedena, many focusing on ERP or online classroom functionalities independently.
- **Differentiators:**
    - Unified system integrating admin, teaching, and learning processes.
    - Modern tech stack (React + Spring Boot).
    - Built-in virtual classroom and AI chatbot support.
- **Areas for improvement:**
    - Enhanced mobile-first design.
    - Localization support (regional languages).
    - Simplified onboarding and configuration for non-technical users.

SWOT Analysis

Strengths:

- Comprehensive solution covering key academic and administrative functions.
- Integration of AI chatbot and live-class functionality.
- Scalable and modern architecture.

Weaknesses:

- Potential setup complexity for institutions.
- Lack of offline/low-bandwidth support.
- Limited brand presence and marketing reach at the current stage.

Opportunities:

- Rising demand for digital education and hybrid learning systems.
- Expansion potential in Tier-2 and Tier-3 educational markets.
- Partnerships with educational boards and government programs.

Threats:

- Competition from established ed-tech solutions.
- Price sensitivity among small institutions.
- Data security and privacy compliance challenges.

Business Potential & Go-to-Market Considerations

- **Revenue Models:** Subscription per institution or per-user, tiered plans based on features, or freemium models.

- **Market Entry Strategy:**
    - Initial focus on small and mid-sized colleges.
    - Build case studies to demonstrate ROI and institutional efficiency gains.
    - Gradual expansion into K-12 and coaching markets.
- **Growth Enablers:**
    - Cloud-based scalability and modular architecture.
    - Partnerships with content providers and ed-tech ecosystems.
    - Mobile app versions and localized interfaces for broader reach.
- **Risk Mitigation:**
    - Optimize for low-bandwidth use cases.
    - Maintain compliance with data privacy standards.
    - Continuous UX improvements and user training initiatives.

3.2 Ideation

1. Problem Identification

In many educational institutions, academic and administrative processes are fragmented across multiple systems — attendance, results, communication, and online classes are often handled separately. This leads to:

- Lack of a unified management system for administrators, faculty, and students.
- Inefficient communication between stakeholders.
- Challenges in adapting to hybrid or remote learning environments.
- Limited use of data analytics or AI to enhance learning and engagement.

Core Problem:
Educational institutions lack an integrated, modern, and affordable digital ecosystem to manage academics, administration, and online interaction efficiently.

2. Ideation Objective

The goal of this ideation process was to conceptualize and design a **unified educational management platform** that bridges the gap between offline and online learning environments. The platform should:

- Centralize academic, administrative, and communication operations.
- Support virtual classrooms and AI-based assistance.
- Be scalable, affordable, and user-friendly for diverse institutions.

3. Idea Generation Process

The ideation process followed a structured approach consisting of brainstorming, market study, user feedback, and feasibility analysis:

a. Brainstorming Sessions:
Initial brainstorming focused on identifying recurring challenges faced by students, teachers, and administrators — such as attendance tracking, result management, and online engagement.

b. Market and Competitor Research:
Analysis of existing ed-tech solutions (like Teachmint, Teno, and Fedena) revealed that most lacked a unified structure combining ERP features with AI and virtual classrooms.

c. User-Centric Design Thinking:
Inputs from teachers and students highlighted the need for simplicity, intuitive interfaces, and quick access to key functions (e.g., viewing attendance, uploading notes, scheduling lectures).

d. Feasibility Analysis:
Technological feasibility was confirmed through a full-stack architecture using **React.js (frontend)** and **Spring Boot (backend)** with database support (MySQL). Cloud deployment and scalability were considered in the design phase.

4. Proposed Solution Concept

The outcome of the ideation phase was **SikshaSetu Edu App** an all-in-one academic management system designed to enhance operational efficiency and learning experiences.

Key Conceptual Features:

- **Unified User Roles:** Admin, Faculty, and Student portals with specific dashboards and permissions.
- **Attendance and Results Management:** Automated record maintenance and easy report generation.
- **Virtual Classroom:** Real-time video sessions using **ZEGOCLOUD API** for hybrid education.
- **AI Chatbot Integration:** Personalized academic support via **Perplexity API**.
- **Event and Notification System:** Centralized updates for academic and extracurricular activities.
- **Digital Resource Sharing:** Faculty can upload and manage notes, assignments, and materials.

Unique Value Proposition:
Unlike traditional ERPs, SikshaSetu combines administrative automation with live teaching and AI-driven academic assistance — creating a truly holistic digital education environment.

5. Innovation Aspect

The innovative elements of SikshaSetu include:

- Integration of **AI-powered chatbot** for instant academic support.
- Real-time **virtual classrooms** within the same ecosystem.
- Modular and scalable architecture for institutions of varying sizes.
- Focus on affordability and ease of adoption for schools and colleges, especially in developing regions.

6. Expected Outcomes

- Streamlined communication and coordination across departments.
- Reduced manual workload for teachers and administrators.
- Enhanced student engagement through interactive and AI-assisted learning tools.
- Improved institutional productivity and record management.
- A scalable solution adaptable to various educational levels and regions.

3.3 Functional Requirements of System

1. User Management

The system shall support registration, authentication, and profile management for Admins, Faculty, and Students. Admins can create, update, and delete user accounts, while others can update limited personal details.

2. Role-Based Access Control

Access shall be based on user roles. Admins have full privileges, faculty can manage attendance, results, and notes, while students have view-only access to their records and materials.

3. Attendance Management

Faculty can mark and update attendance. The system will auto-calculate attendance percentages, and students can view their records in real time.

4. Result Management

Faculty can enter and publish results after admin approval. Students can securely view their marks and performance reports.

5. Notes and Material Sharing

Faculty can upload and manage study materials, assignments, and notes. Students can view and download them as per their courses.

6. Event and Notification System

Admins and faculty can post announcements and events. All users receive real-time notifications through their dashboards.

7. Virtual Classroom

The system supports live video lectures using the ZEGOCLOUD API. Faculty can schedule and host classes, and students can join directly from the platform.

8. AI Chatbot Support

An integrated AI chatbot (via Perplexity API) provides academic assistance and answers general queries for students and faculty.

9. Branch and Department Management

Admins can manage multiple departments, allocate faculty and students, and organize academic resources accordingly.

10. Security and Data Management

The system uses JWT-based authentication and encrypted passwords. All data is stored securely in a MySQL database with proper validation and access control.

11. Dashboard and Analytics

Each user has a personalized dashboard displaying attendance, results, events, and notifications. Admins can view institutional analytics for monitoring performance.

12. Scalability and Administration

The system supports multiple users simultaneously with optimized APIs and performance monitoring. Admins can manage configurations and maintain the system efficiently.

3.4 Non-Functional Requirements of System

1. Performance Requirements

The system shall ensure fast response times and handle multiple users simultaneously without performance degradation.
APIs and database queries shall be optimized for efficiency and low latency.

2. Reliability

The system shall maintain consistent performance and availability under normal and peak loads.
Automatic backups and recovery mechanisms shall ensure data integrity and continuity.

3. Security

All user data shall be encrypted and protected using JWT authentication.
Role-based access control shall prevent unauthorized access, and passwords shall be securely hashed before storage.

4. Usability

The interface shall be intuitive and user-friendly, ensuring easy navigation for Admins, Faculty, and Students.
Clear instructions and minimal steps shall be provided for all key operations.

5. Scalability

The system shall be capable of scaling to support growing numbers of users and institutions without major redesign.
Modular architecture shall allow easy addition of new features or integrations.

6. Availability

The system shall ensure high uptime with minimal maintenance interruptions.
Cloud deployment or containerization shall support round-the-clock access.

7. Maintainability

The codebase shall follow clean coding practices and proper documentation for easy updates and debugging.
Version control using GitHub shall help track changes and manage collaborative development.

8. Portability

The system shall be deployable across multiple platforms and environments with minimal configuration changes.
It shall support both web browsers and, in the future, mobile applications.

9. Compatibility

The system shall be compatible with modern browsers such as Chrome, Edge, and Firefox.
It shall integrate seamlessly with APIs like ZEGOCLOUD and Perplexity for extended functionalities.

10. Data Integrity

All transactions and updates shall ensure consistency and prevent data loss or duplication.
Proper validation shall be enforced before storing or modifying any data.

3.5 Software Requirements

1. Operating System

- Windows 10 or higher (for development and testing)
- Linux/Ubuntu (for server deployment)

2. Development Tools and IDEs

- **Visual Studio Code** – for frontend development using React.js
- **IntelliJ IDEA / Eclipse** – for backend development using Spring Boot
- **Postman** – for API testing and validation
- **Git & GitHub** – for version control and collaboration

3. Programming Languages and Frameworks

- **Frontend:** React.js (JavaScript, JSX), Tailwind CSS for UI styling
- **Backend:** Java 17 with Spring Boot 3 framework
- **Database:** MySQL for structured data storage
- **APIs:** ZEGOCLOUD API (video conferencing), Perplexity API (AI chatbot integration)

4. Server and Deployment Environment

- **Application Server:** Apache Tomcat (embedded with Spring Boot)
- **Deployment:** Cloud-based hosting (AWS / Render / Vercel) or local deployment
- **Containerization (Optional):** Docker for scalable environment setup

5. Libraries and Dependencies

- Spring Security and JWT for authentication and authorization
- Axios for API communication in React

- Hibernate / JPA for ORM database operations
- Node.js and npm for managing frontend dependencies

6. Database Requirements

- MySQL server version 8.0 or above
- Properly normalized relational schema for user, attendance, and result management
- Backup and restore support for data safety

7. Browser Requirements

- Compatible with modern browsers such as Google Chrome, Microsoft Edge, Mozilla Firefox, and Safari (latest versions).
- Fully responsive UI optimized for both desktop and mobile devices.

8. Testing and Documentation Tools

- Postman for API testing and debugging.
- Swagger for auto-generating API documentation.
- JUnit for backend and Jest for frontend testing to ensure reliability and performance.

9. Software Standards and Architecture

- Modular client-server architecture using RESTful APIs.
- Follows clean coding, documentation, and logging practices.
- Ensures scalability, maintainability, and security for long-term system stability.

CHAPTER 4

DESIGN AND PLANNING

4.1 System Architecture

1. Architectural Model

- The system follows a **three-tier client-server architecture**, consisting of the **Presentation Layer (Frontend)**, **Application Layer (Backend)**, and **Data Layer (Database)**.
- Each layer is modular, ensuring independent development, testing, and maintenance.

2. Presentation Layer (Frontend)

- Developed using **React.js**, providing an interactive and responsive user interface.
- Communicates with the backend through RESTful APIs using Axios.
- Ensures role-based access for students, teachers, and administrators with dynamic content rendering.

3. Application Layer (Backend)

- Built using **Spring Boot (Java)**, handling the core business logic and processing client requests.
- Implements authentication and authorization using **Spring Security and JWT tokens**.
- Exposes REST APIs for all data operations such as attendance, events, results, and chatbot interactions.
- Integrates with **ZEGOCLOUD API** for live video sessions and **Perplexity API** for AI assistance.

4. Data Layer (Database)

- Uses **MySQL** to store structured data such as user profiles, attendance records, and academic information.
- Utilizes **Hibernate/JPA** for ORM mapping to manage data efficiently.
- Implements database normalization and indexing to enhance performance and reduce redundancy.

5. Communication Flow

- The frontend sends user requests via HTTP to the backend REST APIs.
- The backend processes the requests, interacts with the database, and returns structured JSON responses to the frontend.
- Secure communication is maintained through HTTPS and token-based authentication.

6. Deployment Architecture

- The system can be deployed on cloud platforms like **AWS**, **Render**, or **Vercel**.
- Backend services run on an embedded **Apache Tomcat server**, while the frontend is hosted as a static web application.
- CI/CD pipelines may be integrated for automated deployment and updates.

7. Scalability and Modularity

- Each module (attendance, events, chatbot, live classes) functions independently, enabling easy scaling and updates.
- The architecture supports horizontal scaling to handle more users and requests as demand grows.

8. Security Measures

- Implements **JWT-based authentication** and **role-based access control** to secure user data.
- Data transmission is encrypted via HTTPS.
- Regular database backups and logging mechanisms ensure system reliability and auditability.

9. System Interaction Overview

- The architecture ensures seamless interaction between all layers — from user interface to backend and database.
- External APIs enhance the system's capability, enabling real-time communication and AI-driven support for users.

10. Performance and Maintainability

• The system is designed for optimal performance with efficient data handling and minimal latency between components.

• Its modular structure ensures easy maintenance, scalability, and the flexibility to integrate future enhancements or additional modules without major architectural changes.

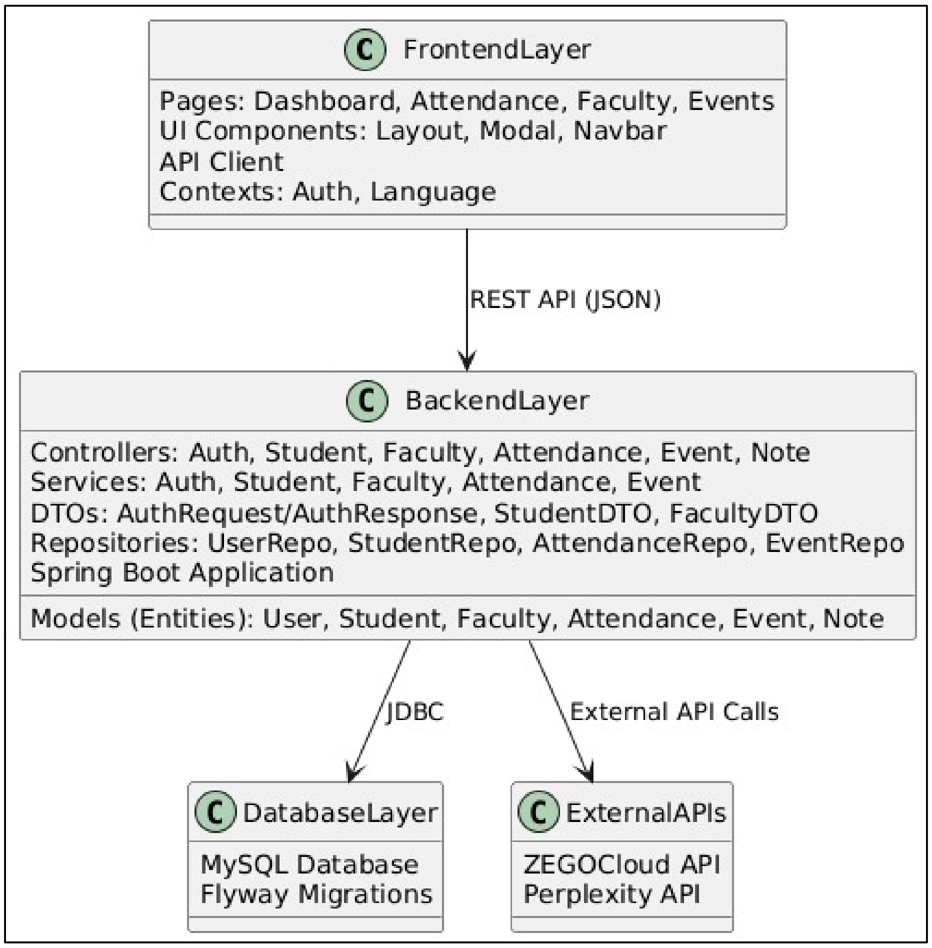

C FrontendLayer
Pages: Dashboard, Attendance, Faculty, Events
UI Components: Layout, Modal, Navbar
API Client
Contexts: Auth, Language
REST API (JSON)
C BackendLayer
Controllers: Auth, Student, Faculty, Attendance, Event, Note
Services: Auth, Student, Faculty, Attendance, Event
DTOs: AuthRequest/AuthResponse, StudentDTO, FacultyDTO
Repositories: UserRepo, StudentRepo, AttendanceRepo, EventRepo
Spring Boot Application
Models (Entities): User, Student, Faculty, Attendance, Event, Note
JDBC
External API Calls
C DatabaseLayer
MySQL Database
Flyway Migrations
C ExternalAPIs
ZEGOCloud API
Perplexity API

Figure 1 – System Architecture

4.2 Flow Chart

The system flowchart illustrates the step-by-step workflow of the **SikshaSetu Edu App**, showing how users interact with different components. It begins with user authentication, followed by access to respective dashboards based on roles (student, teacher, or admin). Each operation such as attendance, events, or communication triggers a backend process that interacts with the database to fetch or store data. The processed results are then returned to the frontend and displayed dynamically. This logical flow ensures smooth communication between the user interface, backend, and database, maintaining efficiency and reliability throughout the system.

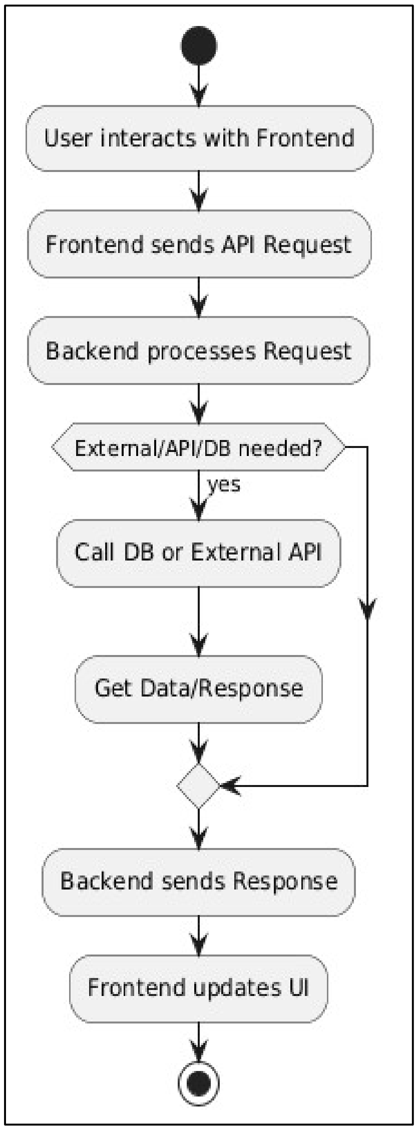


Figure 2 –Flow Chart

4.3 Data Flow Diagram (DFD)

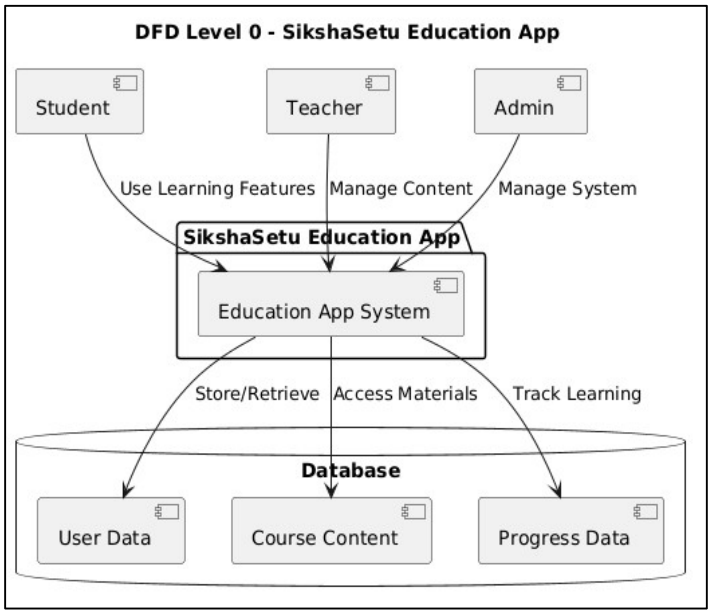


Figure 3 – DFD Level 0

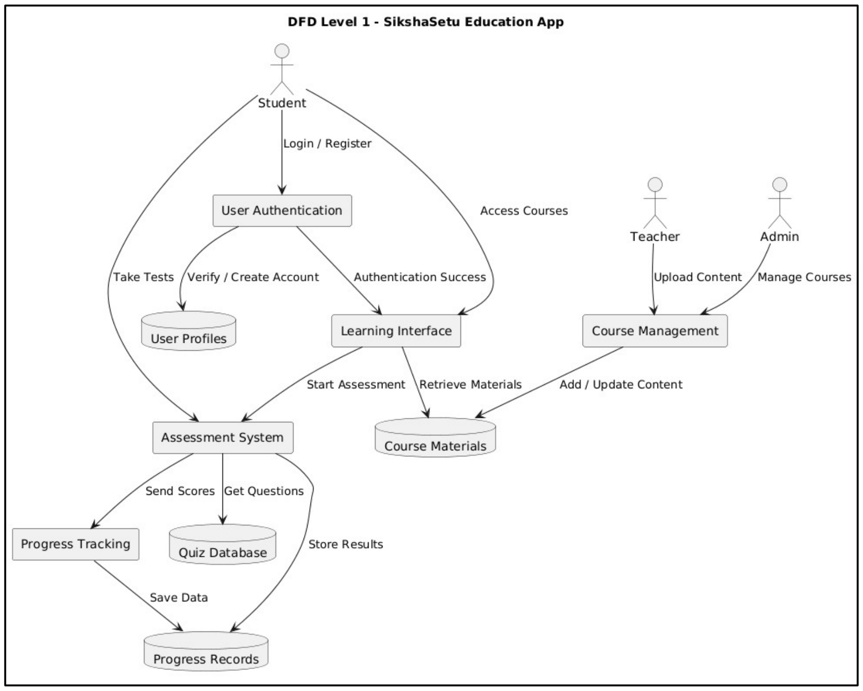


Figure 4 – DFD Level 1

4.4 Activity Diagram

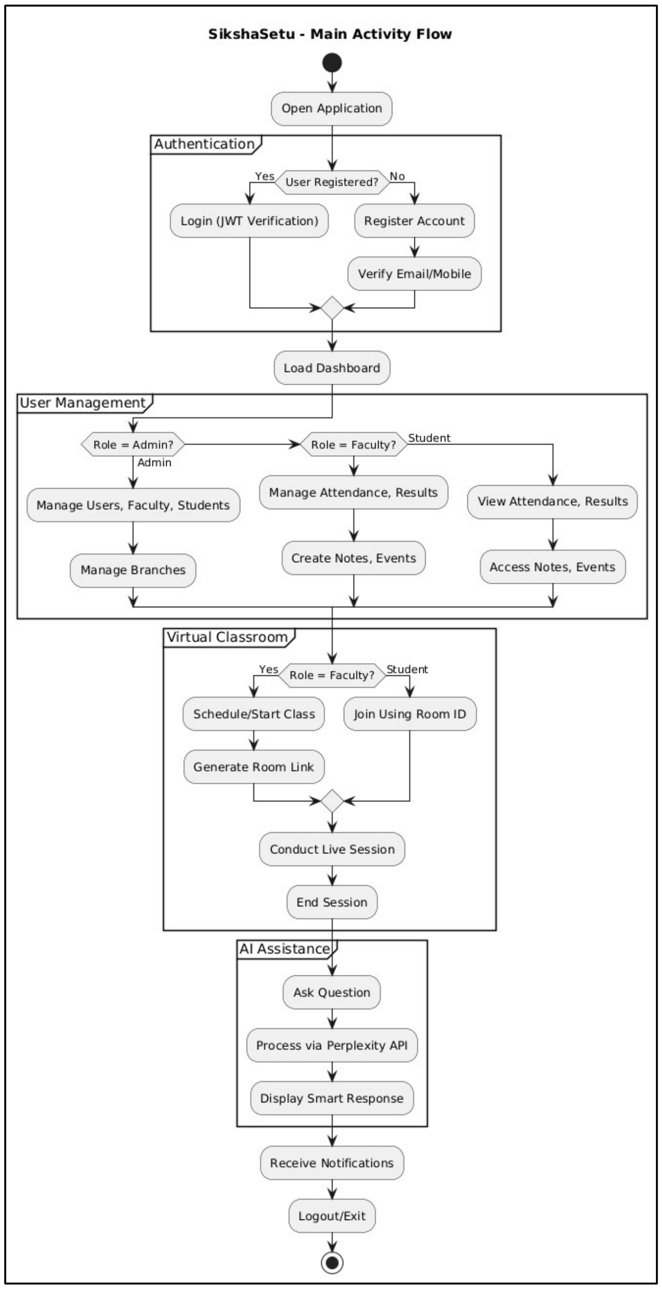


Figure 5 –Activity Diagram

1. **Start:** User opens the SikshaSetu application.
2. **Authentication:**
    - If already registered, user logs in via JWT authentication.
    - If new, user registers an account.

3. **Dashboard Loading:** The system loads the personalized dashboard based on the user's role.
4. **Role-Based Access:**
   - **Admin:** Manages users, students, faculty, and branches.
   - **Faculty:** Manages attendance, results, creates notes or events.
   - **Student:** Views attendance, results, and accesses notes or events.
5. **Virtual Classroom:**
   - Faculty schedules or starts live sessions and generates room links.
   - Students join using the provided room ID.
6. **AI Assistance:**
   - User submits a query.
   - The system processes it through the **Perplexity API** and displays the AI-generated response.
7. **Notifications:** System sends updates and alerts related to sessions, events, and academics.
8. **Logout:** User safely logs out, ending the session.

4.5 Sequence Diagram

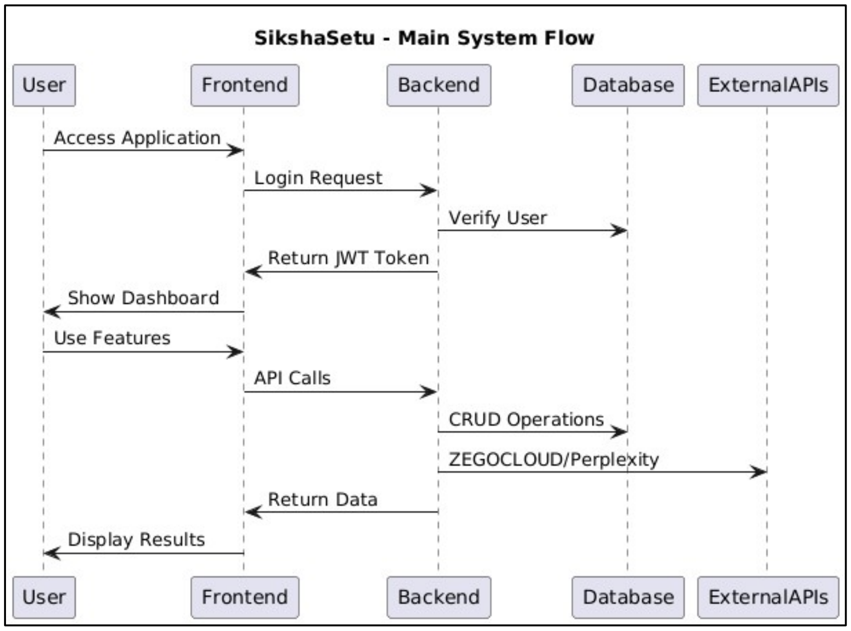


Figure 6 –Sequence Diagram

Authentication Phase

1. **User Access**: User opens the web application
2. **Login Request**: Frontend sends credentials to Backend
3. **User Verification**: Backend validates user against Database
4. **Token Generation**: Backend returns JWT token for authentication
5. **Dashboard Access**: User gains access to main dashboard

Feature Usage Phase

1. **User Interaction**: User interacts with various features
2. **API Communication**: Frontend makes API calls to Backend
3. **Data Operations**: Backend performs CRUD operations on Database
4. **External Integrations**: Backend communicates with external services:
   - ZEGOCLOUD for virtual classrooms
   - Perplexity API for AI chatbot
5. **Data Return**: Backend sends processed data to Frontend
6. **Result Display**: Frontend displays results to User

4.6 Use Case Diagram

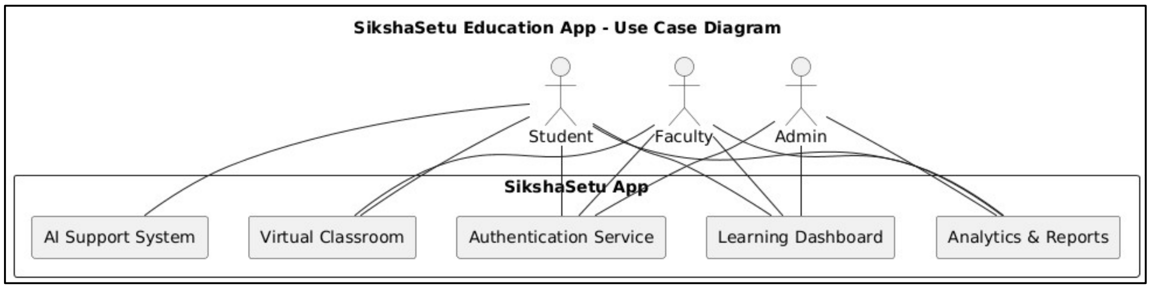


Figure 7 – Use case Diagram

Key Points of the Use Case Diagram

- The **SikshaSetu App** connects Admin, Faculty, and Students in a unified platform.
- **Admin** manages users, courses, and institutional data.
- **Faculty** handles attendance, uploads notes, and conducts virtual classes.
- **Students** access courses, attend classes, and track performance.
- The system includes **AI assistance**, **real-time communication**, and **analytics** for better learning outcomes.
- Ensures **secure login, smooth interaction, and efficient data management** across all roles.

4.7 System Design

SikshaSetu follows a modern three-tier architecture that ensures clear separation of concerns and scalability. The system is built using industry-standard technologies that provide robustness and maintainability.

Frontend Layer:

- React 18 with functional components and hooks
- Tailwind CSS for responsive design
- React Router for client-side navigation
- Context API for state management

Backend Layer:

- Spring Boot 3 REST API framework
- JWT-based authentication system
- Role-based access control (Admin, Faculty, Student)
- Modular service architecture for business logic

Data Layer:

- MySQL relational database management
- JPA/Hibernate for object-relational mapping
- Flyway for database version control and migrations

External Integrations & Services

The system leverages third-party services to enhance functionality while maintaining focus on core educational features. ZEGOCLOUD integration provides comprehensive video conferencing capabilities for virtual classrooms, including session management, real-time communication, and participant handling. The Perplexity AI integration enables intelligent chatbot functionality that supports markdown formatting and provides contextual educational assistance to users.

Core Features & Modules

User Management System:

- Multi-role authentication and authorization
- Profile management for students, faculty, and administrators
- Branch and department organization

Academic Management:

- Attendance tracking and reporting
- Result management and grade processing
- Study materials and notes distribution
- Event scheduling and notifications

Virtual Learning Environment:

- Classroom scheduling and session management
- Real-time video conferencing capabilities

Security Framework

The platform implements a comprehensive security model that protects user data and system integrity. JWT tokens handle session management with configurable expiration times, while role-based access control ensures users only access authorized features. All API endpoints are secured against common vulnerabilities, and password encryption using industry-standard algorithms protects user credentials.

Deployment & Scalability

The system is designed for flexible deployment options including cloud-based and on-premises installations. Environment-specific configurations support development, testing, and production setups. The modular architecture allows for horizontal scaling of components based on user load, while database optimization ensures consistent performance during peak usage periods.

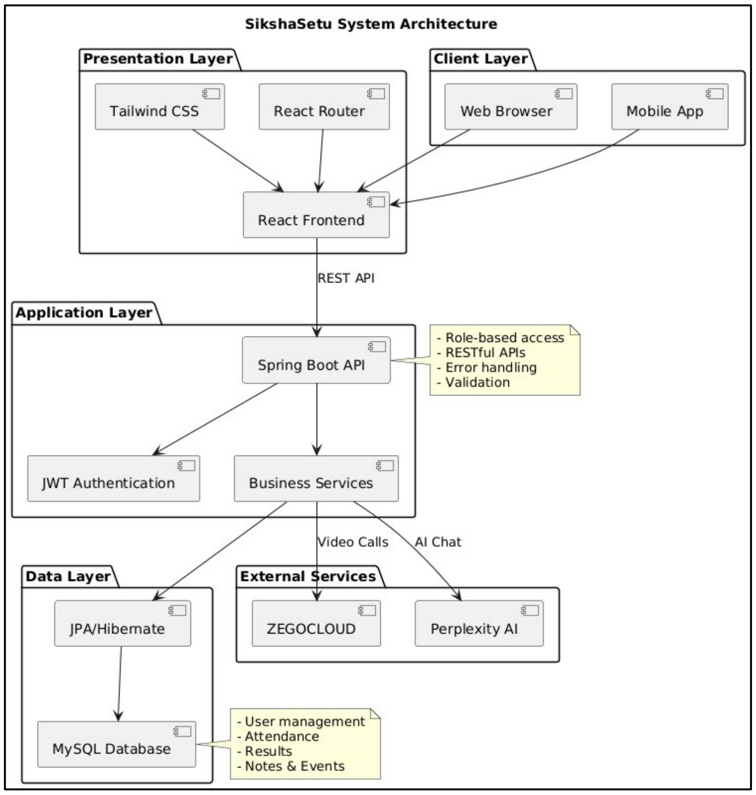


Figure 8 – System Design

4.8 Gantt Chart

Tasks	July	August	September	October
Assessment	🟦			
Remediation	🟦			
Verification		🟦		
Advanced Security Configurations			🟦	
Monitoring & Continuous Improvement				🟦

CHAPTER 5

IMPLEMENTATION

5.1 Code

```
package com.ssid.collegeportal.config;

import com.ssid.collegeportal.service.AuthService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

import org.springframework.security.core.context.SecurityContextHolder;

import org.springframework.security.core.userdetails.UserDetails;

import org.springframework.security.web.authentication.WebAuthenticationDetailsSource;

import org.springframework.stereotype.Component;

import org.springframework.web.filter.OncePerRequestFilter;

import jakarta.servlet.FilterChain;

import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

import java.io.IOException;

@Component

public class JwtAuthenticationFilter extends OncePerRequestFilter {

  @Autowired

  private JwtProvider jwtProvider;

  @Autowired

  private AuthService authService;

  @Override

  protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response,
   FilterChain filterChain)

      throws ServletException, IOException {

    String authHeader = request.getHeader("Authorization");

    String token = null;

    String username = null;

    if (authHeader != null && authHeader.startsWith("Bearer ")) {

      token = authHeader.substring(7);

      username = jwtProvider.getUsernameFromToken(token);

    }

    if (username != null && SecurityContextHolder.getContext().getAuthentication() ==
   null) {

      UserDetails userDetails = authService.loadUserByUsername(username);
if (jwtProvider.validateToken(token)) {
            UsernamePasswordAuthenticationToken authToken = new
UsernamePasswordAuthenticationToken(
                    userDetails, null, userDetails.getAuthorities());
            authToken.setDetails(new
WebAuthenticationDetailsSource().buildDetails(request));
            SecurityContextHolder.getContext().setAuthentication(authToken);
        }
    }
    filterChain.doFilter(request, response);
  }
}
package com.ssid.collegeportal.model;
import jakarta.persistence.*;
import java.time.LocalDateTime;
import java.util.HashSet;
import java.util.Set;
@Entity
@Table(name = "users")
public class User {
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private Long id;
```

```
@Column(nullable = false)

private String name;

@Column(unique = true, nullable = false)

private String email;

@Column(nullable = false)

private String password;

@ManyToMany(fetch = FetchType.EAGER)

@JoinTable(name = "user_roles", joinColumns = @JoinColumn(name = "user_id"),
 inverseJoinColumns = @JoinColumn(name = "role_id"))

private Set<Role> roles = new HashSet<>();

private boolean active = true;

private LocalDateTime createdAt = LocalDateTime.now();

private String resetToken;

private LocalDateTime resetTokenExpiry;

public Long getId() {

   return id;

}

public String getName() {

   return name;

}

public void setName(String name) {

   this.name = name;

}
```

```
public String getEmail() {
    return email;
}
public void setEmail(String email) {
    this.email = email;
}
public String getPassword() {
    return password;
}
public void setPassword(String password) {
    this.password = password;
}
public Set<Role> getRoles() {
    return roles;
}
public void setRoles(Set<Role> roles) {
    this.roles = roles;
}
public boolean isActive() {
    return active;
}
```

```
    public void setActive(boolean active) {

        this.active = active;

    }

    public LocalDateTime getCreatedAt() {

        return createdAt;

    }

    public void setCreatedAt(LocalDateTime createdAt) {

        this.createdAt = createdAt;

    }

    public String getResetToken() {

        return resetToken;

    }

    public void setResetToken(String resetToken) {

        this.resetToken = resetToken;

    }

    public LocalDateTime getResetTokenExpiry() {

        return resetTokenExpiry;

            }

    public void setResetTokenExpiry(LocalDateTime resetTokenExpiry) {

        this.resetTokenExpiry = resetTokenExpiry;

    }

}
```

```
package com.ssid.collegeportal.controller;

import com.ssid.collegeportal.dto.StudentRequestDTO;

import com.ssid.collegeportal.dto.StudentResponseDTO;

import com.ssid.collegeportal.model.Student;

import com.ssid.collegeportal.service.StudentService;

import com.ssid.collegeportal.repository.UserRepository;

import com.ssid.collegeportal.repository.BranchRepository;

import jakarta.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.*;

import java.util.List;

import java.util.stream.Collectors;


@RestController

@RequestMapping("/api/students")

public class StudentController {

@Autowired

private StudentService studentService;

@Autowired

private UserRepository userRepository;
```

```
@Autowired

private BranchRepository branchRepository;

@Autowired

private com.ssid.collegeportal.repository.FacultyRepository facultyRepository;

@GetMapping

@org.springframework.security.access.prepost.PreAuthorize("hasAnyRole('ADMIN',
'FACULTY')")

public List<StudentResponseDTO> getAllStudents(

@RequestParam(required = false) Long branchId,

@RequestParam(required = false) Integer year,

@RequestParam(required = false) Integer semester,

org.springframework.security.core.Authentication authentication) {

com.ssid.collegeportal.model.User currentUser =
userRepository.findByEmail(authentication.getName()).orElse(null);

boolean isAdmin = currentUser != null && currentUser.getRoles().stream().anyMatch(r ->
r.getName().equals("ADMIN"));

boolean isFaculty = currentUser != null && currentUser.getRoles().stream().anyMatch(r -
> r.getName().equals("FACULTY"));

List<Student> students = studentService.getAllStudents();

if (isFaculty) {

// Only allow faculty to see students from their own branch

com.ssid.collegeportal.model.Faculty faculty = facultyRepository.findAll().stream()

    .filter(f -> f.getUser() != null && f.getUser().getId().equals(currentUser.getId()))

    .findFirst().orElse(null);
    if (faculty != null && faculty.getBranch() != null) {
            Long facultyBranchId = faculty.getBranch().getId();
            students = students.stream().filter(s -> s.getBranch() != null &&
s.getBranch().getId().equals(facultyBranchId)).collect(Collectors.toList());
        } else {
            students = List.of();
        }
    } else if (!isAdmin) {
        // Not admin or faculty: students cannot list others
        students = List.of();
    }
    if (branchId != null) {
        students = students.stream().filter(s -> s.getBranch() != null &&
s.getBranch().getId().equals(branchId)).collect(Collectors.toList());
    }
    if (year != null) {
        students = students.stream().filter(s -> s.getYear() ==
year).collect(Collectors.toList());
    }
    if (semester != null) {
        students = students.stream().filter(s -> s.getSemester() ==
semester).collect(Collectors.toList());
    }
    return students.stream().map(this::toResponseDTO).collect(Collectors.toList());

    }

    @GetMapping("/{id}")

    @org.springframework.security.access.prepost.PreAuthorize("hasAnyRole('ADMIN',
'FACULTY') or (hasRole('STUDENT') and #id == principal.id)")

    public ResponseEntity<StudentResponseDTO> getStudentById(@PathVariable Long id,
org.springframework.security.core.Authentication authentication) {

        com.ssid.collegeportal.model.User currentUser =
userRepository.findByEmail(authentication.getName()).orElse(null);

        boolean isAdmin = currentUser != null && currentUser.getRoles().stream().anyMatch(r
-> r.getName().equals("ADMIN"));

        boolean isFaculty = currentUser != null && currentUser.getRoles().stream().anyMatch(r
-> r.getName().equals("FACULTY"));

        boolean isStudent = currentUser != null && currentUser.getRoles().stream().anyMatch(r
-> r.getName().equals("STUDENT"));

        return studentService.getStudentById(id)

                .filter(student -> {

                    if (isAdmin) return true;

                    if (isFaculty) {

                        com.ssid.collegeportal.model.Faculty faculty =
facultyRepository.findAll().stream()

                            .filter(f -> f.getUser() != null &&
f.getUser().getId().equals(currentUser.getId()))

                            .findFirst().orElse(null);
return faculty != null && faculty.getBranch() != null && student.getBranch() != null &&
faculty.getBranch().getId().equals(student.getBranch().getId());
                    }
                    // Student: can only view self
                    return isStudent && student.getUser() != null &&
student.getUser().getId().equals(currentUser.getId());
                })
                .map(this::toResponseDTO)
                .map(ResponseEntity::ok)
                .orElse(ResponseEntity.notFound().build());
    }
    @PostMapping
    @org.springframework.security.access.prepost.PreAuthorize("hasAnyRole('ADMIN',
'FACULTY')")
    public StudentResponseDTO createStudent(@Valid @RequestBody StudentRequestDTO
dto) {
        Student student = new Student();
        student.setUser(userRepository.findById(dto.getUserId()).orElse(null));
        student.setBranch(branchRepository.findById(dto.getBranchId()).orElse(null));
        student.setYear(dto.getYear());
        student.setSemester(dto.getSemester());
        Student saved = studentService.createStudent(student);
        return toResponseDTO(saved);
    }


    @PutMapping("/{id}")
    @org.springframework.security.access.prepost.PreAuthorize("hasAnyRole('ADMIN',
'FACULTY') or (hasRole('STUDENT') and #id == principal.id)")
    public ResponseEntity<StudentResponseDTO> updateStudent(@PathVariable Long id,
            @Valid @RequestBody StudentRequestDTO dto) {
        try {
            Student student = new Student();
            student.setUser(userRepository.findById(dto.getUserId()).orElse(null));
            student.setBranch(branchRepository.findById(dto.getBranchId()).orElse(null));
            student.setYear(dto.getYear());
            student.setSemester(dto.getSemester());
            Student updated = studentService.updateStudent(id, student);
            return ResponseEntity.ok(toResponseDTO(updated));
        } catch (RuntimeException e) {
            return ResponseEntity.notFound().build();
        }
    }
    @DeleteMapping("/{id}")
    @org.springframework.security.access.prepost.PreAuthorize("hasAnyRole('ADMIN',
'FACULTY')")
    public ResponseEntity<Void> deleteStudent(@PathVariable Long id) {
        studentService.deleteStudent(id);
        return ResponseEntity.noContent().build();
    private StudentResponseDTO toResponseDTO(Student student) {
        StudentResponseDTO dto = new StudentResponseDTO();
        dto.setId(student.getId());
        if (student.getUser() != null) {
            dto.setUserId(student.getUser().getId());
            dto.setUserName(student.getUser().getName());
            dto.setUserEmail(student.getUser().getEmail());
        }
        if (student.getBranch() != null) {
            dto.setBranchId(student.getBranch().getId());
            dto.setBranchName(student.getBranch().getName());
        }
        dto.setYear(student.getYear());
        dto.setSemester(student.getSemester());
        return dto;
    }
}
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/logo.png" type="image/png" />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo.png" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#2563eb" />
    <meta
      name="description"
      content="SikshaSetu - Comprehensive student management system"
    />
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link
href="https://fonts.googleapis.com/css2?family=Inter:wght@300;400;500;600;700&display=
swap" rel="stylesheet">
    <title>SikshaSetu</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
  </body>
</html>
```

```
import React from 'react';

import { Routes, Route, Navigate } from 'react-router-dom';

import { useAuth } from './contexts/AuthContext';

import Layout from './components/layout/Layout';

import ProtectedRoute from './components/common/ProtectedRoute';

import LoadingSpinner from './components/common/LoadingSpinner';

import DashboardRedirect from './components/common/DashboardRedirect';

import { ROLES } from './constants/roles';
```

```
// Auth pages

import Login from './pages/auth/Login';

import Register from './pages/auth/Register';

// Main pages

import Dashboard from './pages/Dashboard';

import Users from './pages/Users';

import Students from './pages/Students';

import Faculty from './pages/Faculty';

import Branches from './pages/Branches';

import Attendance from './pages/Attendance';

import Results from './pages/Results';

import Events from './pages/Events';

import Notes from './pages/Notes';

import Notifications from './pages/Notifications';

import Chatbot from './pages/Chatbot';

import VirtualClassroom from './pages/VirtualClassroom';

import JoinVirtualClassroom from './pages/JoinVirtualClassroom';

import VideoCall from './pages/VideoCall';

// Error pages

import Unauthorized from './pages/Unauthorized';

import NotFound from './pages/NotFound';
```

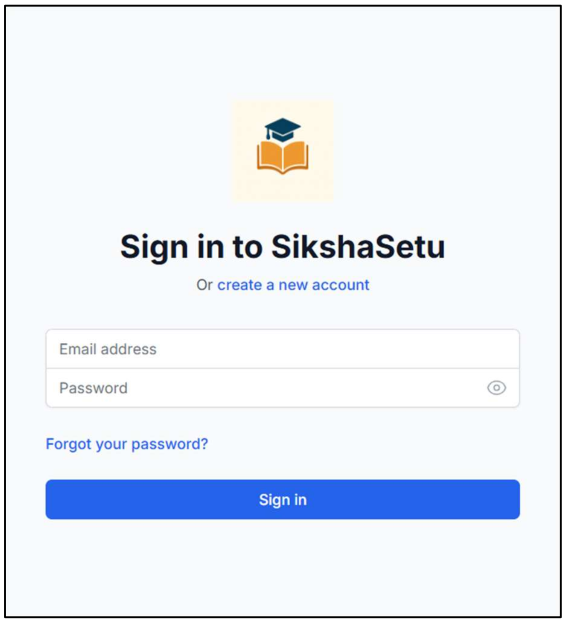
```
function App() {
  const { loading, isAuthenticated } = useAuth();
  if (loading) {
    return <LoadingSpinner text="Loading application..." />;
  }
  return (
    <Routes>
      {/* Public routes */}
      <Route
        path="/login"
        element={!isAuthenticated() ? <Login /> : <DashboardRedirect />}
      />
      <Route
        path="/register"
        element={!isAuthenticated() ? <Register /> : <DashboardRedirect />}
      />
      {/* Protected routes */}
      <Route path="/" element={
        <ProtectedRoute>
          <Layout />
        </ProtectedRoute>
      }>
        <Route index element={<DashboardRedirect />} />
```

```
<Route path="dashboard" element={<Dashboard />}
<Route path="users" element={
  <ProtectedRoute roles={[ROLES.ADMIN]}>
    <Users />
  </ProtectedRoute>
}
<Route path="students" element={
  <ProtectedRoute roles={[ROLES.ADMIN, ROLES.FACULTY]}>
    <Students />
  </ProtectedRoute>
} />
<Route path="faculty" element={
  <ProtectedRoute roles={[ROLES.ADMIN, ROLES.FACULTY]}>
    <Faculty />
  </ProtectedRoute>
} />
<Route path="branches" element={
  <ProtectedRoute roles={[ROLES.ADMIN]}>
    <Branches />
  </ProtectedRoute>
} />
<Route path="attendance" element={<Attendance />} />
<Route path="results" element={<Results />} />
```

```
 <Route path="events" element={<Events />} />
<Route path="notes" element={<Notes />} />
<Route path="notifications" element={<Notifications />} />
<Route path="chatbot" element={<Chatbot />} />
<Route path="virtual-classroom" element={
  <ProtectedRoute roles={[ROLES.FACULTY]}>
    <VirtualClassroom />
  </ProtectedRoute>
} />
<Route path="join" element={
  <ProtectedRoute roles={[ROLES.STUDENT, ROLES.FACULTY]}>
    <JoinVirtualClassroom />
  </ProtectedRoute>
} />
<Route path="join/:roomId" element={
  <ProtectedRoute roles={[ROLES.STUDENT, ROLES.FACULTY]}>
    <JoinVirtualClassroom />
  </ProtectedRoute>
} />
<Route path="video-call/:roomId" element={
  <ProtectedRoute roles={[ROLES.STUDENT, ROLES.FACULTY]}>
    <VideoCall />
```

```
              </ProtectedRoute>
          } />
          <Route path="unauthorized" element={<Unauthorized />} />
          <Route path="*" element={<NotFound />} />
        </Route>
        {/* Catch all */}
        <Route path="*" element={<NotFound />} />
      </Routes>
  );
}
export default App;
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <BrowserRouter
      future={{
        v7_startTransition: true,
        v7_relativeSplatPath: true
      }}
    >
      <AuthProvider>
        <App />
        <ToastContainer
    position="top-right"
        autoClose={5000}
        hideProgressBar={false}
        newestOnTop={false}
        closeOnClick
        rtl={false}
        pauseOnFocusLoss
        draggable
        pauseOnHover
        theme="light"
      />
    </AuthProvider>
  </BrowserRouter>
 </React.StrictMode>
);
```

5.2 Result


Sign in to SikshaSetu

Or create a new account

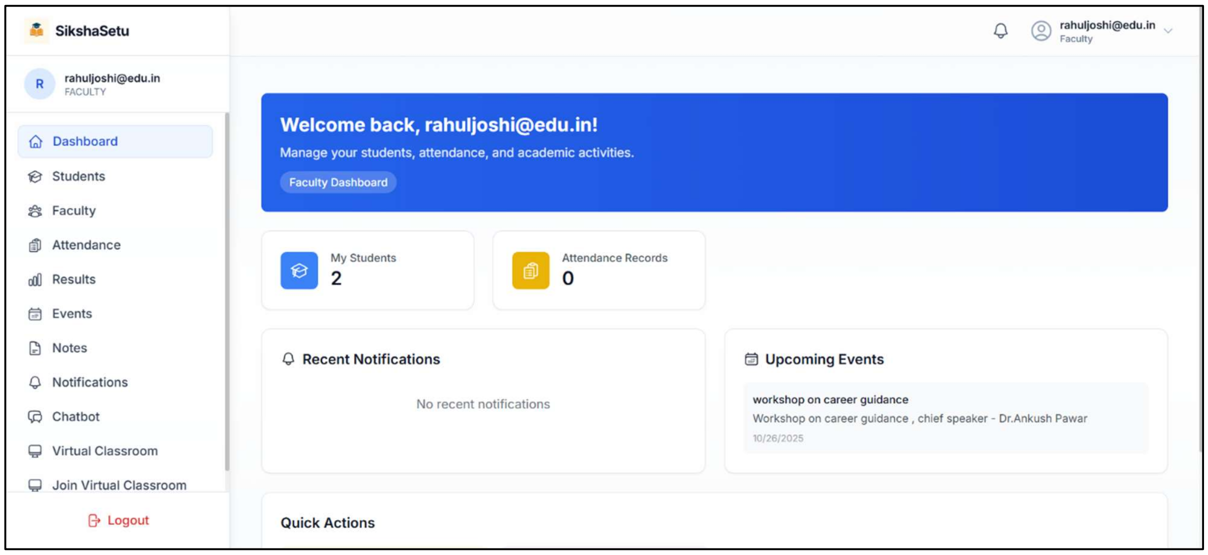
Email address

Password

Forgot your password?

Sign in

Figure 9 –Result 1


SikshaSetu

rahuljoshi@edu.in
FACULTY

Dashboard
Students
Faculty
Attendance
Results
Events
Notes
Notifications
Chatbot
Virtual Classroom
Join Virtual Classroom
Logout

rahuljoshi@edu.in
Faculty

Welcome back, rahuljoshi@edu.in!

Manage your students, attendance, and academic activities.

Faculty Dashboard

My Students 2

Attendance Records 0

Recent Notifications

No recent notifications

Upcoming Events

workshop on career guidance
Workshop on career guidance , chief speaker - Dr.Ankush Pawar
10/26/2025

Quick Actions

Figure 10 –Result 2

CHAPTER 6

CONCLUSION & FUTURE SCOPE

The SikshaSetu Education Portal successfully delivers a comprehensive digital platform that transforms educational management through its robust technical implementation. By integrating Spring Boot for backend services, React for frontend interfaces, and MySQL for database management, the system provides a secure and scalable solution that effectively serves administrators, faculty, and students. The platform's successful incorporation of virtual classrooms using ZEGOCLOUD and AI assistance through Perplexity API demonstrates its capacity to handle advanced educational technologies.

The platform holds substantial potential for expansion through several strategic developments. Mobile applications with offline capabilities would significantly enhance accessibility for users. The integration of AI-powered personalization could create adaptive learning paths tailored to individual student needs. Blockchain technology could be incorporated for secure credential verification and academic record management.

Looking ahead, Virtual Reality classrooms could provide immersive learning experiences, while multi-institution support would enable broader educational collaboration. Enhanced analytics would offer deeper insights into student performance and institutional effectiveness. These progressive enhancements would transform SikshaSetu into a more intelligent and comprehensive educational ecosystem, continuously adapting to meet evolving digital learning requirements while maintaining its commitment to transforming education through innovative technology solutions.

REFERENCES

1. Central Management System for Educational Institutions Using Spring Boot

- Authors: Muruganantham, R., Reddy, C.G.S., Prakash, G.B., & Nayak, J.J.
- Year: 2023
- Publication: ZKG International Journal, Vol. VIII, Issue I
- Relevance: Spring Boot educational management system architecture and implementation

2. Design and Implementation of a Student Management System Based on Springboot Framework Technology

- Author: Tang, Y.
- Year: 2023
- Publication: Proceedings of the 3rd EAI International Conference on AICIT
- DOI: 10.4108/eai.17-11-2023.2342765
- Relevance: Spring Boot framework for student information management with CRUD operations

3. Design and Development of an Online Education System Based on SpringBoot

- Author: Li, L.
- Year: 2024
- Publication: 2024 3rd International Conference on AICIT
- DOI: 10.1109/AICIT62434.2024.10730277
- Relevance: Online education system architecture with Spring Boot and modern teaching methods

4. Research on Intelligent Learning Platform System Based on Spring Boot

- Author: Gan, L.
- Year: 2022
- Publication: Atlantis Press (Cited by 2)
- Relevance: Intelligent learning platform with user interaction, course video, and data analysis modules

5. OAuth 2.0 Resource Server JWT

- Author: Spring Security Documentation
- Year: 2024
- Publication: Spring.io Official Documentation

- URL: https://docs.spring.io/spring-security/reference/servlet/oauth2/resource-server/jwt.html
- Relevance: JWT authentication implementation with Spring Security framework

6. WebRTC Video Call: A Full Guide

- Author: ZEGOCLOUD
- Year: 2024
- Publication: ZEGOCLOUD Official Documentation
- URL: https://www.zegocloud.com/blog/build-video-call-with-webrtc
- Relevance: WebRTC implementation for virtual classroom and video conferencing integration

7. Towards Designing Educational System Using Role-Based Access Control

- Authors: Kabier, M.K., Yassin, A.A., & Abduljabbar, Z.A.
- Year: 2023
- Publication: International Journal of Intelligent Engineering and Systems, Vol.16, No.2 (Cited by 4)
- DOI: 10.22266/ijies2023.0430.05
- Relevance: RBAC implementation for educational systems with multi-factor authentication

8. Attendance Management System

- Author: Singh, M.
- Year: 2015
- Publication: IEEE Xplore (Cited by 47)
- DOI: 10.1109/ICIINFS.2015.7124938
- Relevance: Automated attendance tracking systems for educational institutions

9. Database Migrations with Flyway

- Author: Flyway Documentation
- Year: 2024
- Publication: Baeldung.com
- URL: https://www.baeldung.com/database-migrations-with-flyway
- Relevance: Database schema versioning and migration management for Spring Boot applications

10. Authentication with React Router v6: A Complete Guide

- Author: React Router Team
- Year: 2024
- Publication: LogRocket Blog
- URL: https://blog.logrocket.com/authentication-react-router-v6/
- Relevance: Protected routes and authentication implementation in React applications