(SATS)
Student AttendanceTracking System

A Project Report

Submitted to

140 CLG INSTITUTE OF ENGINEERING & TECHNOLOGY

Towards the partial fulfillment of

The Degree of

Bachelor of Technology

In

Computer Science Engineering

Session:-2025-2026

Submitted To:
Mr Vikram Rajpurohit
HoD
Computer Science Department

Submitted By:

Aditya Raj Singh
22ECLCS001

Faculty of Engineering & Technology
Department of Computer Science and Engineering
CLG INSTITUTE OF ENGINEERING & TECHNOLOGY

DECLARATION

In accordance with the requirements for the degree of B.Tech. Programme in Computer Science, in Faculty of Engineering and Technology, I present this Project report on “Student Attendance Tracking System (SATS)”.

I declare that the work presented in the report is my own work except as acknowledged in the text and foot notes.

Date: 06/12/2025 **Aditya Raj Singh**

Faculty of Engineering & Technology
Department of Computer Science and Engineering
CLG INSTITUTE OF ENGINEERING & TECHNOLOGY

CERTIFICATE FROM GUIDE

This is to certify that **Aditya Raj Singh** Roll No:-22ECLCS001 has submitted the Project report entitled **"Student Attendance Tracking System (SATS)"** in partial fulfillment for the award of the degree of bachelor of Technology in Computer Science Engineering. The report is up to my satisfaction and as per the format prescribed for the writing of the report. His work is approved for presentation.

Date :-06/12/2025 **Signature of Guide**

ACKNOWLEDGEMENT

Iam deeply indebted to Dr. C.L. Gehlot, Chairman, CLGIET, Sumerpur for providing me the infrastructural facilities. My utmost gratitude to almighty God, who has been very kind to me all the time. The achievement would be worthless if it was not the timely help and guidance of wellwishers. I know acknowledgement is not well enough to scale their help and wishes. It is an immense pleasure to thank Director Sir CLGIET for his precious guidance, constant encouragement throughout the training.

I also express my thanks to my Head of Department Mr. Vikram Rajpurohit and other faculty members for their valuable suggestions from time to time. I would also like to thank my classmates and all that has directly and indirectly helped me during my work. I would also like to thank my parents and my family members for providing me with their blessing, immense encouragement, moral support, constant inspiration, enthusiasm and co-operation for my training.

DATE: 06/12/2025 **Aditya Raj Singh**

ABSTRACT

Attendance management is a crucial part of academic institutions. Traditional attendance methods such as manual registers or basic spreadsheets are often prone to human error, manipulation, and inefficiency.
This project presents a **Student Attendance Tracking System** built using modern web technologies such as **Next.js, Drizzle ORM, MySQL (Railway), ShadCN UI, JavaScript, AG-Grid, Recharts, and Kinde Authentication**.

The system provides digital management of students, grades, and attendance in an efficient and scalable manner.
It allows teachers to:

- Record daily attendance
- View attendance reports and analytics
- Visualize class-wise performance
- Manage student and grade data
- Ensure secure access using Kinde authentication

Modern UI components from ShadCN, powerful data grids from AG-Grid, and visual charts from Recharts enhance usability and monitoring.
The backend uses **Drizzle ORM** with **MySQL (Railway)** ensuring reliable database interactions and scalable deployment.

This project demonstrates a practical and modern solution for educational institutions aiming to digitalize attendance with accuracy, transparency, and ease of use.

Table of Contents

1. Introduction ---------- 8

2. Problem Statement ---------- 12

3. Objectives of the Project ---------- 14

4. Scope of the Project ---------- 18

5. Literature Review ---------- 22

6. Existing System ---------- 27

7. Proposed System ---------- 31

8. System Requirements Specification ---------- 35

   - Hardware Requirements
   - Software Requirements

9. System Design ---------- 40

   - Architecture Diagram
   - Use-Case Diagram
   - Data Flow Diagrams
   - ER Diagram
   - UML Diagrams

10. Database Design ---------- 48

11. Technology Used ---------- 52

12. Implementation ----------57

13. Screenshots of the Application ---------- 65

14. Testing & Test Cases ---------- 70

15. Results & Discussion ---------- 76

16. Applications ---------- 80

17. Future Scope ---------- 83

18. Conclusion ---------- 92
19. References ---------- 93
20. Appendix ---------- 94

1. INTRODUCTION

Attendance tracking is one of the most essential administrative activities within educational institutions. It serves as a fundamental indicator of student participation, discipline, commitment, and academic performance. Traditionally, attendance has been recorded manually using paper registers or basic spreadsheet-based digital tools. Although widely practiced, these methods often introduce inefficiencies such as errors in manual entry, difficulty in maintaining long-term records, challenges in generating analytical reports, and limited accessibility of data. As educational institutions evolve toward digital transformation, there is a growing need for a more systematic, accurate, and technology-driven approach to attendance monitoring.

The increasing integration of digital systems in the education sector highlights the importance of online platforms that are capable of managing academic tasks with improved accuracy and user-friendly interfaces. A modern attendance system must not only store and retrieve data efficiently but also provide teachers and administrators with meaningful insights that can support decision-making processes. With advancements in web development frameworks, cloud technologies, and authentication mechanisms, it is now possible to design innovative attendance tracking solutions that are both scalable and secure.

This project, titled **"Student Attendance Tracking System using Next.js, Drizzle ORM, MySQL (Railway), ShadCN UI, JavaScript, Kinde Authentication, AG-Grid, and Recharts,"** aims to create a robust and interactive web-based attendance management platform. It leverages the power of modern frontend and backend technologies to deliver seamless data handling, real-time updates, and analytical visualizations. By utilizing Next.js, the system benefits from a highly efficient full-stack framework capable of server-side rendering, API routing, and optimized performance. Drizzle ORM ensures type-safe and structured database operations, while MySQL (hosted through Railway) provides scalable cloud storage for reliable access and long-term data retention.

The system integrates **Kinde Authentication** to enforce secure user access, ensuring that only authorized individuals—such as teachers or school administrators—can manage student records and update attendance. User Interface (UI) components designed with **ShadCN** enhance usability, creating a clean and modern experience for users. Additionally, **AG-Grid** is employed to facilitate high-performance grid-based attendance marking, offering features such as inline editing, pagination, sorting, and dynamic data updates. Analytical insights are displayed using **Recharts**, allowing the representation of attendance patterns through bar charts, pie charts, and trend dashboards.

This integrated technological approach transforms traditional attendance tracking into an automated, interactive, and visually informative digital process. By making attendance

management more efficient, the system aligns with the larger goal of improving institutional workflow and enhancing academic administration. The project demonstrates how modern web technologies can be strategically combined to build practical solutions suited for real-world applications.

1.1 Need for a Modern Attendance Management System

Educational environments today demand systems that can handle data accurately, provide transparency, and reduce administrative workload. Manual attendance systems fall short in meeting these expectations due to several inherent limitations. A digital system not only solves these issues but also introduces features that were not feasible with traditional methods.

Key reasons why a modern attendance system is needed include:

- **Reduction in Manual Errors:** Automated data entry minimizes common mistakes found in manual record-keeping.
- **Time Efficiency:** Teachers can mark attendance within seconds, allowing more time for classroom activities.
- **Centralized Data Storage:** Cloud-based storage ensures that records are accessible anytime and remain secure.
- **Analytical Insights:** Attendance data can be visualized to identify trends, irregularities, or performance issues.
  Enhanced Transparency: Administrators can easily verify records without relying on physical registers.
- **Improved Academic Monitoring:** Patterns of absenteeism can be identified early, enabling timely interventions.
- **Scalability and Accessibility:** Cloud technologies allow the system to support multiple classes, grades, and user roles.

1.2 Role of Technology in the Proposed System

The technological choices in this project strengthen its reliability, performance, and usability. The tools and frameworks used play specific roles in ensuring that the system meets the requirements of a modern educational environment.

Key Technologies and Their Roles

- **Next.js:**
  Providesserver components, efficient routing, API integration, and high performance for buildingmodern web applications.

- **Drizzle ORM:**
  Ensuresstructured, type-safe database queries and schema management for MySQL, improving development consistency.
- **MySQL (Railway):**
  Offers secure, cloud-hosted database storage that supports scalability and remote access.
- **ShadCN UI:**
  Enhances user experience through reusable and visually appealing UI components.
- **Kinde Authentication:**
  Adds secure authentication and authorization, ensuring that only legitimate users access the system.
- **AG-Grid:**
  Provides a powerful grid interface for daily attendance marking with features such as inline editing, sorting, and pagination.
- **Recharts:**
  Enables visualization of attendance data, helping teachers and administrators understand patterns and trends.

1.3 Benefits of the Student Attendance Tracking System

The system solves major challenges faced by traditional attendance methods and introduces several key advantages:

- **Accuracy and Reliability:** Eliminates risks of miscalculation or unreadable records found in paper registers.
- **Real-Time Access:** Attendance can be viewed immediately after being recorded.
  User-Friendly Interface: Simple and intuitive UI makes the system accessible even to non-technical users.
- **Enhanced Decision-Making:** Visual dashboards provide insights into student attendance performance.
- **Secure Authentication:** Protects sensitive academic data from unauthorized access.
- **Efficient Data Retrieval:** Users can instantly view attendance for specific dates, students, or grades.
- **Paperless Operation:** Reduces physical storage requirements and supports eco-friendly practices.
  Scalable Design: Can be expanded to additional features like notifications, biometric integration, and parent portals.

1.4 Structure of the Report

This project report is organized into multiple chapters, each addressing a specific aspect of the system design, development, and evaluation. The structure is intended to provide a clear and

comprehensive understanding of how the system operates and the methodologies used in its creation.

The report includes:

1. **Introduction** – Overview, purpose, and relevance of the project
2. **Problem Statement** – Issues with existing systems
3. **Objectives** – Goals of the proposed solution
4. **Scope** – Boundaries of what the system includes and excludes
5. **Literature Review** – Existing systems and technologies
6. **System Analysis** – Detailed study of requirements
7. **System Design** – Architecture, diagrams, and workflow
8. **Implementation** – Code modules, technologies, and integration
9. **Testing** – Test cases and evaluation
10. **Results & Discussion** – Performance and system outcomes
11. **Conclusion & Future Scope**

2. PROBLEM STATEMENT

Attendance management plays a crucial role in ensuring discipline, academic monitoring, and administrative record-keeping within educational institutions. Despite its importance, many schools and colleges continue to rely on **traditional paper-based attendance registers** or simple spreadsheet tools. These approaches, although familiar, create numerous challenges that affect the efficiency, accuracy, and reliability of attendance records. As institutions grow in size and complexity, the limitations of manual systems become more evident, leading to the need for a more automated and technology-driven solution.

Traditional attendance registers are vulnerable to **human errors**, including incorrect entries, overwriting, missing data, and difficulty in tracking attendance over long periods. Teachers must spend valuable class time recording attendance manually, reducing the time available for teaching and student interaction. Additionally, retrieving historical attendance records is cumbersome, especially when large volumes of data are stored across multiple registers. Generating insights such as monthly attendance rates, student-wise trends, or identifying irregularities becomes nearly impossible without advanced tools.

Another major challenge lies in the **lack of centralized accessibility**. Attendance data recorded manually remains isolated within classroom registers, making it difficult for administrators or department heads to access real-time information. This lack of transparency and shared access limits effective academic monitoring and delays decision-making. Moreover, manual systems cannot offer automated analysis, visual reports, or summary dashboards that help educators understand attendance patterns and intervene early in cases of chronic absenteeism.

The transition to digital solutions often introduces its own issues, especially when using outdated or poorly designed software. Many existing attendance applications lack user-friendly interfaces, do not support real-time updates, or suffer from inconsistent performance. Some systems require complex installation processes, while others fail to secure sensitive academic information effectively. These gaps highlight the need for a modern, secure, cloud-based, and interactive attendance management system.

Furthermore, there is a pressing demand for systems that integrate well with modern web development standards, support **secure authentication**, and provide **scalable database management** capable of handling large amounts of data. In the absence of such solutions, institutions continue to struggle with decentralized data, inefficient processes, and limited analytical capabilities.

Problems Identified

The major issues that motivated the development of this project include:

1. **Manual Attendance Errors**
    - Susceptible to inaccuracy, overwriting, and incomplete entries.

- Difficult to cross-verify and maintain consistently.

2. **Time-Consuming Process**
   - Teachers spend considerable time marking attendance manually.
   - Reduces productivity and interferes with instructional progress.
3. **Lack of Real-Time Access**
   - Paper-based records cannot be accessed by administrators instantly.
   - Data remains isolated in physical registers.
4. **NoAnalyticalInsights**
   - Manual systems cannot generate graphs, reports, or trends.
   - Hard to identify patterns such as irregular attendance or academic risk.
5. **Storage and Retrieval Issues**
   - Large registers become difficult to maintain over long periods.
   - Retrieving past data is slow, inconvenient, and error-prone.
6. **Security Concerns**
   - Physical records can be damaged, lost, or misused.
   - No authentication or access control in traditional systems.
7. **Lack of Scalability**
   - Manual or outdated digital tools do not support multiple classes, grades, or large datasets.
8. **Limited Interaction and Usability**
   - Most traditional systems are not user-friendly and lack intuitive interfaces.

Problem Summary

In summary, educational institutions lack a **centralized,automated,secure,and user-friendly attendance management system** capable of storing student records, marking daily attendance efficiently, and generating meaningful insights. The absence of such a system leads to data inconsistencies, administrative difficulties, and an overall decline in operational efficiency.

This project aims to **address these challenges** by developing a modern Student Attendance Tracking System using **Next.js, Drizzle ORM, MySQL (Railway), ShadCN UI, JavaScript, Kinde Authentication, AG-Grid, and Recharts**.
The system eliminates manual inefficiencies and delivers a streamlined, interactive, and data-driven approach to attendance management.

3. OBJECTIVE

The primary objective of the Student Attendance Tracking System is to develop a **modern, efficient, and automated solution** for recording and managing attendance in educational institutions. The system is designed to replace traditional manual entry methods with a structured, cloud-powered, secure application that ensures accuracy, accessibility, and real-time insights. By leveraging advanced web technologies such as **Next.js, Drizzle ORM, MySQL (Railway), ShadCN, Kinde Authentication, AG-Grid, and Recharts**, the project aims to deliver a robust platform that supports teachers, administrators, and academic staff in making informed decisions.

A well-designed attendance management system should minimize human effort, reduce errors, increase transparency, and provide a comprehensive analytical representation of attendance trends. This project focuses not only on digitizing attendance entry but also on enabling deeper analysis through dashboards, visual charts, and summaries. The long-term intention is to create a scalable foundation that can be expanded in the 8th semester into a fully integrated school management platform.

3.1 General Objectives

1. **To develop a centralized web-based system for real-time attendance tracking**
   The system aims to create a single platform where attendance can be recorded,stored, and accessed efficiently from anywhere, without dependency on paper registersorlocal files.
2. **To enhance the accuracy, reliability, and accessibility of attendance records**
   By using automated database operations, the chances of manual errors are minimized, ensuring clean and consistent records over time.
3. **To provide visual insights to help educators monitor attendance trends**
   Using Recharts and dashboard analytics, the system helps identify patterns like irregular attendance, absenteeism, and overall student participation.
4. **To improve teacher productivity through automated tools and an interactive grid interface**
   The AG-Grid interface allows teachers to mark attendance quickly, reducing time spent on routine tasks.

3.2 Specific Objectives

1. Implement Secure User Authentication

- Provide secure login functionality using **Kinde Authentication**.
- Ensure only authorized users (teachers/admins) can access attendance features.
- Maintain data privacy and prevent unauthorized modifications.

2. Develop an Efficient Attendance Recording System

- Create a system where teachers can mark daily attendance using an editable grid (AG-Grid).
- Allow marking of **present/absent** for individual students seamlessly.
- Automatically save attendance in the MySQL database using Drizzle ORM.
- Support monthly or day-wise attendance viewing.

---

3. Store and Manage Data Using a Scalable Cloud Database

- Use **MySQL on Railway** for reliable cloud-based storage.
- Enable fast CRUD operations (Create, Read, Update, Delete) using Drizzle ORM.
- Ensure schema management and migrations are structured and scalable.

---

4. Generate Monthly and Daily Attendance Reports

- Visualize daily attendance records using **Bar Charts**.
- Summarize monthly attendance distribution using **Pie Charts**.
- Provide quick insights such as:
    - Total Present
    - Total Absent
    - Daily attendance trends
    - Month-wise student participation
- Support administrative decision-making through easy-to-understand charts.

---

5. Create an Interactive and User-Friendly Interface

- Use **ShadCN UI components** for clean, modern design.
- Ensure the user interface is responsive, intuitive, and easy to navigate.
- Display attendance data in a structured format using AG-Grid with filtering and editing support.

---

6. Enable Filtering by Grade, Month, and Date

- Allow teachers to select a grade and month to view attendance.
- Support dynamic data loading directly from the API.
- Ensure efficient backend filtering to improve performance.

---

7. Convert Raw Attendance Data into Analytical Insights

- Use custom algorithms to compute:
  - Attendance percentages
  - Total students vs. present count
  - Absenteeism rate
  - Day-wise attendance summaries
- Provide real-time calculations and updates on dashboards.

---

8. Ensure Data Integrity, Consistency, and Security

- Prevent duplicate attendance entries for the same day.
- Validate all inputs before saving to the database.
- Implement secure, optimized SQL queries using Drizzle ORM.
- Preserve data accuracy across all modules.

---

9. Provide a Scalable Architecture for Future Expansion

The system is designed to be modular and extendable so that in the 8th semester, additional modules like:

- Timetable management
- Student performance tracking
- Teacher attendance
- Fee management
- Notifications and alerts

can be integrated seamlessly without redesigning the core system.

3.3 Long-Term Objectives

1. **Automate more administrativeprocesses** to reduce workload on teachers and staff.
2. **Support multi-user roles** (Admin, Teacher, Principal) with role-based access control.
3. **Enable mobile app integration** for instant updates and accessibility.
4. **Expand attendance analytics** with AI-driven reporting in future phases.
5. **Create a robust School ManagementSystem** built upon the foundation created in this project.

Conclusion of Objectives

The objectives of this project collectively aim to build a **modern, secure, user-friendly, analytical, and scalable Attendance Tracking System** that effectively addresses the limitations of manual processes. By implementing advanced frontend and backend technologies, this project provides a foundation for improved academic management and positions itself for future enhancement into a complete institutional management system.

4. SCOPE

The scope of the *Student Attendance Tracking System* encompasses the design, development, implementation, and evaluation of a web-based attendance management platform intended for educational institutions. The system replaces the traditional paper-based attendance registers with an automated, secure, and interactive digital solution. By leveraging modern tools such as **Next.js, Drizzle ORM, MySQL (Railway), ShadCN UI, AG-Grid, Recharts, and Kinde Authentication**, the project aims to deliver a scalable and intuitive system suitable for teachers and administrators.

This project focuses on building the **attendance management module** as the core functionality in the 7th semester, while also laying the foundation for expanding the system into a complete school management solution in the 8th semester. The scope includes user authentication, attendance marking, data visualization, analytics, and easy accessibility across devices. However, it intentionally excludes certain advanced features such as parent dashboards, SMS alerts, AI-driven predictions, and administrative billing systems, which may be added in later phases.

4.1 In-Scope Features

The following components and functionalities **are included** in the scope of this project:

1. Web-Based Attendance Management

- A dedicated interface for teachers to mark daily attendance for students.
- Support for selecting the class/grade and month dynamically.
- Editable attendance grid using **AG-Grid** for quick marking.
- Ability to toggle present/absent values for each student on any given day.

2. Secure Authentication and Access Control

- Login system implemented using **Kinde Authentication**.
- Only authorized users (teachers/admins) can access attendance pages.
- Ensures secure data access and prevents unauthorized modification.

3. Student and Grade Management

- Display of student data according to the selected grade.
- Preventing duplicate attendance entries for the same student/day.
- Fetching data from MySQL via Drizzle ORM efficiently.

4. Cloud Database Integration

- Storing students, attendance, and grade information in **MySQL hosted on Railway**.

- Use of **Drizzle ORM** for schema creation, migrations, and queries.
- Ensuring data consistency and structural integrity.

5. Attendance Analytics and Reporting

- **Daily attendance summaries** using bar charts (Recharts).
- **Monthly attendance distribution** using pie charts.
- Calculation of:
    - Total number of students
    - Present percentage
    - Absent percentage
    - Attendance trends
- Visualization to assist academic decisions.

6. Responsive and Modern User Interface

- Clean UI components built using **ShadCN UI**.
- Grid-based structure for easy navigation.
- Mobile-friendly layouts to support different screen sizes.

7. API Development and Data Handling

- Creation of REST APIs to store, update, and retrieve attendance data.
- Parameter-based filtering (grade, month, date).
- Secure, optimized database operations using Drizzle ORM query builders.

4.2 Out-of-Scope Features (Not Included in Current Phase)

The following features are **notpart of the current 7th-semester implementation** , but may be incorporated in future semesters:

1. Parent/Student Online Portals

- No login or dashboard for parents/students.
- No visibility of attendance charts for students.

2. Automated Notifications

- No SMS or email alerts for absences.
- No low-attendance automated warnings.

3. AI-Based Predictive Features

- No forecasting of attendance trends.
- No anomaly detection for irregular patterns.

4. Complete School Management System Features

- Fee management systems.
- Examination and grading modules.
- Timetable scheduling.
- Teacher performance monitoring.

5. Mobile App Development

- No Android/iOS native application.
- The current system is only web-based.

6. Offline Attendance Functionality

- Requires internet connection.
- No offline caching or local storage.

7. Role-Based Access Levels Beyond Basic Authentication

- Only teacher/admin login is supported.
- No multi-level permissions structure yet.

4.3 Technological Scope

The project involves the use of modern technology to ensure reliability, scalability, and maintainability.

Frontend Scope

- Built entirely using **Next.js**.
- Dynamic rendering using React components.
- UI creation using ShadCN UI.
- Data visualization using Recharts.
- Interactive data grid using AG-Grid.

Backend Scope

- API routes implemented within Next.js using Route Handlers.
- All CRUD operations for attendance.
- Secure session management using Kinde.

Database Scope

- Design of tables for Students, Grades, and Attendance.
- Schema handling with Drizzle ORM.

- Cloud database hosting on Railway.

4.4 User Scope

The system is primarily designed for:

Teachers

- Marking daily attendance for their classes.
- Viewing class-wise attendance.
- Monitoring monthly summaries.

Administrators

- Accessing consolidated attendance data.
- Reviewing attendance percentages.
- Using analytics to make policy decisions.

4.5 Scope Summary

The scope of this project is limited but impactful. It focuses on building a **core attendance system** with secure access, structured data, real-time analytics, and a responsive interface. By establishing this strong foundation, the project becomes ready for expansion in the next semester into a complete school/college management solution.

The system fulfills the essential requirements of modern attendance tracking and sets the stage for innovative future enhancements

5. LITERATURE REVIEW

A literature review is a systematic evaluation of existing research, technologies, and methodologies related to the development of digital attendance systems. The review helps in identifying gaps in current solutions, understanding the strengths and weaknesses of existing models, and justifying the need for the *Student Attendance Tracking System* built using Next.js, Drizzle ORM, MySQL, ShadCN, AG-Grid, Recharts, and Kinde Authentication.

Modern educational institutions are increasingly adopting digital solutions for attendance tracking due to the limitations of manual methods. Traditional paper-based registers are prone to errors, manipulation, loss, and inefficiencies in data retrieval. Numerous researchers and developers have focused on creating automated attendance systems using biometric devices, RFID scanners, QR codes, or mobile applications. However, many of these systems lack scalability, real-time analytics, cloud integration, or user-friendly web interfaces. The emergence of cloud computing and modern JavaScript frameworks has contributed significantly to new approaches in solving these limitations.

5.1 Existing Attendance Management Approaches

1. Manual Attendance Systems

Manual attendance systems using paper registers or Excel sheets have dominated educational environments for decades.
However, studies show several drawbacks:

- High probability of human error.
- Time-consuming marking process.
- Risk of manipulation (proxy attendance).
- Difficulty in generating reports or analyses.

Most institutions aim to eliminate these inefficiencies by moving toward digital systems.

2. Biometric-Based Attendance Systems

Biometric systems (fingerprint, face recognition) have become widely discussed in academic solutions.
Literature identifies several benefits:

- High accuracy in verification.
- Prevention of proxy attendance.
- Automatic timestamping.

But biometric systems also have limitations:

- Require expensive hardware.

- Hard to deploy for large institutions or rural schools.
- Hygiene concerns (fingerprint scanners).
- Limited scalability in cloud environments.

3. RFID and Smart Card Systems

Some researchers propose RFID-based attendance marking.
Advantages include:

- Fast attendance marking.
- Low-cost hardware tags.

Limitations include:

- Students can exchange cards (proxy risk).
- Hardware failures are common.
- Not suitable for remote or hybrid learning environments.

4. Mobile or QR-Code-Based Attendance Systems

Recent systems use mobile apps or QR scans for attendance.
While effective for higher education environments, challenges include:

- Device dependency (students must have functional smartphones).
- Requires consistent internet availability.
- Students can share QR codes or screenshots.

5. Web-Based Attendance Systems

Web-based attendance platforms have emerged as a practical solution because:

- They require no special hardware.
- They are accessible across devices.
- They integrate well with cloud databases.
- They support analytics and reporting features.

However, literature shows that many traditional web-based systems have drawbacks:

- Outdated UI/UX.
- Lack of real-time analytics dashboards.
- Limited authentication security.
- Poor scalability and database design.

5.2 Technological Trends Supporting Modern Attendance Systems

1. Shift Toward Cloud-Based Databases

Cloud-based infrastructure (Railway, Supabase, Firebase, AWS RDS, etc.) allows:

- Remote accessibility.
- High availability and reliability.
- Automatic backups.
- Better scalability for handling large student datasets.

MySQL on Railway, used in this project, aligns with these modern requirements.

2. Rise of JavaScript Full-Stack Frameworks (Next.js)

Research highlights several reasons for Next.js being widely adopted:

- Server and client rendering capabilities.
- Built-in routing and API routes.
- High performance due to React's virtual DOM.
- Smooth integration with modern UI libraries.

Next.js is considered a strong choice for academic and enterprise web applications.

3. ORM Adoption for Better Database Management

Older systems rely on raw SQL queries.
Issues found in literature:

- Higher chances of syntax errors.
- Difficult migrations.
- Security vulnerabilities like SQL injection.

ORMs (Object-Relational Mappers) such as Drizzle ORM solve these issues by:

- Providing typed schemas.
- Auto-generating migrations.
- Ensuring safer query building.
- Improving developer productivity.

4. Data Visualization in Educational Analytics

Studies emphasize the importance of visual representation of student data.
Charts help institutions:

- Identify trends and attendance drops.
- Conduct performance analysis.

- Make informed decisions.

Recharts, used in this project, is specifically noted in literature for:

- Lightweight charting.
- React compatibility.
- Customizable visualizations.

5.3 Gaps Identified in Existing Literature

After reviewing existing attendance systems, several gaps become clear:

1. Lack of Flexible and Editable Grids

Most older web systems store attendance but do not offer:

- Real-time editing
- Spreadsheet-like views
- Interactive grids for teachers

AG-Grid fills this gap with features like cell editing, pagination, filters, and sorting.

2. Insufficient Authentication Mechanisms

Many research projects rely on basic login forms without:

- OAuth support
- Multi-tenant authentication
- Secure token handling

Kinde Authentication addresses these gaps through enterprise-level security and session management.

3. Limited Monthly and Daily Analytics

A majority of attendance systems only show:

- “Present / Absent” counts
- Single-day reports

They do **not** provide:

- Month-wise pie charts
- Daily bar chart summaries
- Student-wise attendance trends

This project bridges that gap.

4. Poor UI/UX and Accessibility

Many reviewed academic projects use outdated HTML-only layouts and do not support:

- Responsive design
- Component-based UI
- Theme consistency

ShadCN improves UI quality through:

- Prebuilt accessible components
- Tailwind CSS integration
- Modern design system architecture

5.4 Summary of Literature Findings

The literature highlights a strong shift from hardware-dependent attendance systems to **cloud-based, web-driven, analytics-ready platforms**. Existing solutions are often limited by scalability, poor user experience, weak security, and lack of interactive visualization tools.

The *Student Attendance Tracking System* developed in this project addresses these gaps by incorporating:

- **Next.js** for efficient full-stack development
- **Drizzle ORM + MySQL** for secure and structured data management
- **Kinde Authentication** for reliable user login
- **AG-Grid** for real-time attendance editing
- **Recharts** for professional-grade visual analytics
- **ShadCN UI** for a modern and responsive interface

Thus, the literature review strongly supports the need for a comprehensive, scalable, modern attendance management solution—precisely what this project aims to deliver

6. EXISTING SYSTEM

The existing attendance management system used in many educational institutions is still largely manual, paper-based, or partially digital with limited automation. Despite technological developments, a significant number of schools and colleges continue to rely on traditional attendance registers, basic Excel sheets, or simple software tools that lack scalability and analytical capabilities. This section presents a detailed overview of existing systems, their workflows, and their limitations.

6.1 Manual Paper-Based Attendance System

In many institutions, the teacher records attendance manually in a physical register. This process involves marking present (P) or absent (A) for each student daily.

Workflow of Manual System

- Teacher calls out names one by one.
- Students respond, and attendance is marked.
- Register is stored in the classroom or staff room.
- Monthly totals are calculated manually.
- Reports are prepared at the end of the term.

Problems Identified

1. **Time-Consuming**
   Teachers spend 10–15 minutes daily on attendance, reducing teaching time.
2. **High Possibility of Human Errors**
   Mistakes in marking, overwriting, or misplacing entries are common.
3. **Difficult to Retrieve and Analyze**
   Monthly or yearly attendance records require manual counting and verification.
4. **Risk of Data Loss**
   Registers can be lost, damaged, or misplaced.
5. **Proxy Attendance**
   Students can answer on behalf of others without being noticed.
6. **Lack of Transparency for Parents/Management**
   No real-time visibility into student attendance.

6.2 Excel-Based or Basic Digital Attendance Tools

Some institutions use Excel sheets or basic desktop software to digitize attendance. Although this reduces paperwork, it still involves manual input.

Advantages

- Easy to store on a computer.
- Simple to update and modify.
- Calculations (sum, percentages) become easier with formulas.

Limitations

1. **Still Requires Manual Data Entry**
   Teachers input present/absent manually, leading to potential errors.
2. **No Centralized Database**
   Files are stored locally; sharing and synchronization become difficult.
3. **Limited Security**
   Files can be edited by unauthorized users.
4. **Lack of User Authentication**
   Anyone with the file can modify data, reducing reliability.
5. **No Data Analytics**
   Excel cannot generate interactive dashboards, bar charts, or month-wise visualizations easily.

6.3 Biometric-Based Attendance Systems

Some modern institutions have adopted biometric devices (fingerprint, face scan).
These systems automate identification and record timestamps.

Strengths

- Prevents proxy attendance.
- Accurately verifies identity.
- Reduces manual work.

Limitations

1. **High Initial Cost**
   Installation of biometric devices is expensive.
2. **Maintenance Issues**
   Sensors may fail due to dust, moisture, or misuse.
3. **Hygiene Concerns**
   Fingerprint scanners require frequent touching, which is a concern post-COVID.
4. **Data Syncing Problems**
   Many biometric systems store data locally and do not integrate easily with cloud databases.
5. **Not Suitable for Hybrid/Remote Classes**
   Students not physically present cannot be marked through biometrics.

6.4 RFID, QR Code, and Card-Based Systems

These systems use ID cards or QR codes scanned by a device or teacher's phone.

Benefits

- Faster than calling roll numbers manually.
- Reduces attendance marking time.
- Easy to implement in small institutions.

Drawbacks

1. **Proxy Possible**
   Students can give their card or QR code to a friend.
2. **Hardware Dependency**
   Requires card readers or QR scanning devices.
3. **Maintenance of Cards**
   Cards can be lost or damaged.
4. **Limited Reporting Features**
   Most systems only store present/absent entries without advanced analytics.

6.5 Existing Web-Based or Legacy Attendance Systems

Some institutions use older web portals or ERP systems for attendance.
These systems provide basic functionality but suffer from major limitations such as:

1. **Outdated UI and Poor User Experience**
   Interfaces are not responsive or mobile-friendly.
2. **Slow Loading Time**
   Built using older technologies, leading to performance issues.
3. **No Real-Time Charting or Visualization**
   Cannot generate bar charts, pie charts, or interactive dashboards.
4. **Weak Database Design**
   Leads to duplicated records, inaccurate reports, or data inconsistency.
5. **Limited Authentication Security**
   Many legacy systems use simple username/password without OAuth or session management.
6. **Low Flexibility**
   Hard to integrate with modern APIs, mobile apps, or analytics tools.

6.6 Summary of Limitations in the Existing System

Across all reviewed attendancemanagement practices, common limitations are observed:

- **Manual work is high Human errors are**
- **frequent No real-time analytics or**
- **insights Security and privacy issues**
- **remain Lack of cloud integration Reports**
- **are not automatically generated Poor**
- **scalability for large student groups**
-

These limitations justify the need for a modern solution that leverages **Next.js, Drizzle ORM, MySQL(Railway),ShadCN,AG-Grid,Recharts,andKindeAuthentication** to create a reliable, scalable, and analytics-driven attendance tracking platform

7. PROPOSED SYSTEM

The proposed system is a modern, cloud-enabled **Student Attendance Tracking System** designed using **Next.js**, **Drizzle ORM**, **MySQL (Railway)**, **ShadCN UI**, **Kinde Authentication**, **AG-Grid**, and **Recharts**. It addresses the inefficiencies, errors, and limitations in traditional attendance tracking by providing a fast, scalable, secure, and interactive platform for institutions.

The system automates daily attendance marking, ensures accurate storage of records, and provides detailed insights through visual dashboards and charts. The proposed solution follows modern software engineering practices, focusing on performance, user experience, security, and data integrity.

7.1 Overview of the System

The proposed system is a **web-based attendance platform** that allows teachers to:

- Maintain class-wise and month-wise attendance.
- Mark student presence/absence digitally.
- View attendance status for any day of any month.
- Generate insights through bar charts, pie charts, and summaries.
- Manage student and grade information.
- Authenticate securely using Kinde's OAuth login.

The system also stores all attendance data in a cloud MySQL database and retrieves it using Drizzle ORM, ensuring type safety, minimal errors, and optimal data flow.

7.2 System Architecture

The proposed system follows a **three-layer architecture**, ensuring modularity and scalability:

1. Presentation Layer (Frontend)

Developed using **Next.js**, **ShadCN**, and **AG-Grid**, this layer provides:

- A responsive and user-friendly interface.
- Dynamic dashboard with charts and insights.
- Editable attendance grid using AG-Grid.
- Secure login and session handling using Kinde Auth.

2. Application Layer (Backend APIs)

This layer uses **Next.js API routes** with **Drizzle ORM**, responsible for:

- CRUD operations for students, grades, and attendance.
- Attendance fetching with filtering (grade, month, day).
- Authentication validation.
- Business logic for calculating attendance summaries.

3. Data Layer (Database)

MySQL hosted on **Railway** is used as the primary database.

Key features include:

- Well-structured tables for students, grades, and attendance.
- Normalized schema to remove redundancy.
- Consistent and secure database access.
- Strong relationship mapping with Drizzle ORM.

7.3 Core Features of the Proposed System

The proposed system offers several features that overcome the challenges in existing systems:

1. Digital Attendance Recording

Teachers can mark attendance directly in a grid interface, selecting present or absent for each student on each day.
This eliminates manual paperwork and reduces time consumption.

2. Automated Attendance Analytics

The system uses **Recharts** to generate:

- **Bar charts** for daily present/absent counts.
- **Pie charts** for overall monthly attendance percentages.
- **Summary cards** for total students, daily attendance percentage, and absence percentage.

These analytics support data-driven decision-making.

3. Secure Authentication

Using **Kinde Authentication**, the system ensures:

- Modern OAuth-based login.
- Secure session handling.
- Prevention of unauthorized access to dashboard pages.

Only authenticated teachers can manage attendance and view data.

4. AG-Grid Powered Attendance Management

AG-Grid provides:

- Editable grid for daily attendance.
- Pagination, sorting, filtering.
- Dynamic columns based on number of days in a month.
- Smooth performance even for large datasets.

5. Real-Time Data Sync

Every change in the attendance grid triggers an API request:

- Present status is updated in the database instantly.
- Absent status triggers an automatic delete operation.

The system ensures accuracy and consistency.

6. Cloud-Based Database

MySQL on Railway provides:

- Always-on availability.
- Automatic backups.
- High scalability for future growth.
- Easy remote access for multiple users.

7. Admin Features (Extendable in Final Year Project)

The base structure supports future implementation of:

- Teacher dashboards.
- Parent access portals.
- Automated alerts for low attendance.
- Report card integration.

7.4 Advantages of the Proposed System

Efficiency Improvements

- Reduces daily attendance time significantly.
- Minimizes manual effort and human errors.

High Accuracy & Reliability

- Eliminates duplicate records.
- Ensures correct attendance tracking.

Strong Security

- OAuth authentication ensures only authorized access.
- Database connection secured with environment variables.

Scalable & Maintainable

- Modular Next.js architecture.
- Clean ORM-based database operations.

User-Friendly Interface

- ShadCN provides professional and modern UI components.
- Grid and charts improve data visualization and usability.

Real-Time Analytics

- Instant insights into attendance patterns.
- Visual understanding of performance and trends.

7.5 Why This System is Better Than Existing Systems

Existing System Problem	Proposed System Solution
Manual marking is time-consuming	Digital marking using AG-Grid
Hard to generate attendance reports	Auto-generated visual dashboards
Human errors and overwriting issues	Database-driven, error-free system
No centralization	Cloud MySQL with secure ORM
No real-time analytics	Bar charts & pie charts
Poor authentication	Kinde OAuth login
Limited scalability	Modern Next.js framework

7.6 Summary

The proposed Student Attendance Tracking System utilizes modern web technologies to provide a **fast, secure, intelligent, and user-friendly attendance management solution**. By integrating cloud services, ORM-based databases, and interactive UI components, the system addresses all shortcomings of traditional attendance systems.

8. SYSTEM REQUIREMENTS SPECIFICATION (SRS)

A System Requirements Specification outlines all necessary hardware and software resources essential for the successful development, deployment, and operation of the *Student Attendance Tracking System*. These requirements help ensure the system runs efficiently, remains scalable, and delivers a smooth user experience to administrators, teachers, and students.
The requirements are categorized into **hardware** and **software** specifications based on the needs of both the client-side (frontend users) and the server-side (backend hosting and database environment).

8.1 Hardware Requirements

The hardware requirements for this system are minimal because it is primarily a web-based application. The system can be accessed from any modern device with an internet connection. The hardware requirements are divided into **development**, **server-side**, and **client-side** hardware needs.

8.1.1 Development Machine Requirements

These are required for students/developers during the creation and testing of the project:

Minimum Requirements

- **Processor:** Intel Core i3 (7th gen or above) / AMD Ryzen 3
- **RAM:** 4 GB
- **Storage:** 10 GB free disk space
- **Display:** 1366×768 resolution monitor
- **Internet:** Minimum 5 Mbps connection
- **Input Devices:** Standard keyboard and mouse

Recommended Requirements

- **Processor:** Intel Core i5 / Ryzen 5 or higher
- **RAM:** 8 GB or more
- **Storage:** 20 GB free SSD space
- **Display:** Full HD (1920×1080)
- **Internet:** 10–20 Mbps reliable broadband
- **GPU:** Optional, but useful for faster UI development

8.1.2 Client-Side (User) Hardware Requirements

Teachers or administrators accessing the system need:

Minimum

- Desktop / Laptop / Tablet / Smartphone
- Dual-core processor
- 2 GB RAM
- Stable internet connection
- A modern web browser (Chrome, Edge, Firefox, Safari, Brave)

Recommended

- Quad-core processor
- 4 GB RAM or higher
- Full HD display
- High-speed broadband

Since the system is web-based, it does not require installation and runs smoothly on low-end devices.

8.1.3 Server-Side Hardware Requirements

The system is deployed on cloud hosting (Railway), so physical hardware is not required. However, the approximate cloud resources needed are:

Minimum (for development/testing)

- CPU: 0.5–1 vCPU
- RAM: 512 MB to 1 GB
- Storage: 1–5 GB (for MySQL database)
- Bandwidth: Based on usage (typically low for academic projects)

Recommended (for production)

- CPU: 1–2 vCPU
- RAM: 1–2 GB
- Storage: 5–10 GB scalable
- Automated backups enabled
- High-availability environment

Cloud hosting ensures flexibility, scalability, and low maintenance.

8.2 Software Requirements

Software requirements include all necessary system software, development tools, libraries, frameworks, and third-party services used during the development and deployment of the system.

8.2.1 Operating System Requirements

For Development

- Windows 10/11
- macOS 11 or later
- Linux distributions (Ubuntu 20.04+, Fedora, etc.)

For Users

- Any OS with a modern browser:
  - o Windows, macOS, Linux
  - o Android, iOS

8.2.2 Required Software Tools and Frameworks

The system uses a modern full-stack development environment:

Frontend Technologies

- **Next.js14+**
  - ForUI, server-side rendering, routing, and API creation.
- **React.js**
  - Component-based UI development.
- **ShadCNUI**
  - For professional, accessible, and reusable UI components.
- **AG-Grid**
  - Used to render editable attendance tables.
- **Recharts**
  - For bar charts, pie charts, and analytics visualization.
- **Tailwind CSS** *(optional if used with ShadCN)*
  - Utility-first styling.

Backend Technologies

- **Next.js API Routes**

    - Backend endpoints built inside the same project structure.
- **Drizzle ORM**
    - Type-safe ORM for MySQL database access.
    - Eliminates SQL errors and ensures consistency.
- **MySQL(RailwayCloudDatabase)**
    - Stores attendance, student, and grade data.
    - Fast and scalable cloud database solution.

Authentication Tools

- **KindeAuthentication**
    - OAuth2-basedlogin and secure session management.
    - Handles useridentity and access control.

Development Tools

- **Node.js (v18orlater)**
    - Required to runNext.js, React, and build scripts.
- **NPMor Yarn**
    - For installingdependencies.
- **VS Code**
    - RecommendedIDE with extensions for JavaScript, React, and MySQL.
- **Git &GitHub**
    - Version controlsystem for collaborative work.

Browser Requirements

The system is fully compatible with all modern browsers:

- Google Chrome (recommended)
- Mozilla Firefox
- Microsoft Edge
- Safari
- Brave

Browsers must support ES6+ JavaScript and cookies/local storage for authentication.

Other Software Requirements

- **Railway CLI (optional)** – for managing cloud database.

- **Postman / Thunder Client** –fortesting APIs.
- **Environment Variable (.env) Support** – for secure credentials management.

8.3 Summary of System Requirements

The system is designed to be lightweight and scalable, requiring minimal hardware for clients and leveraging cloud infrastructure for backend operations. The software stack is modern, secure, and optimized for performance, making the application suitable for educational institutions of different sizes.

9. SYSTEM DESIGN

System design plays a crucial role in defining how various components of the "Student Attendance Tracking System" interact to deliver efficient functionality. This chapter explains the architectural design, process flow, user interactions, and data modelling used to construct the system. The design decisions were guided by modularity, scalability, maintainability, and ease of use.

9.1 Architecture Design

The architecture of the system follows a modern full-stack web application structure, combining **Next.js**, **Drizzle ORM**, **MySQL**, and **Railway cloud hosting** to ensure high performance and reliability.

Key Components of the Architecture:

1. Frontend Layer

- Built using **Next.js (React)** with ShadCN UI components.
- Client-side components handle dynamic data rendering, attendance grid, dashboards, and charts.
- AG-Grid is used for interactive table views.
- Recharts is integrated to generate attendance statistics in graphical format.
- Communicates with backend via RESTful API routes.

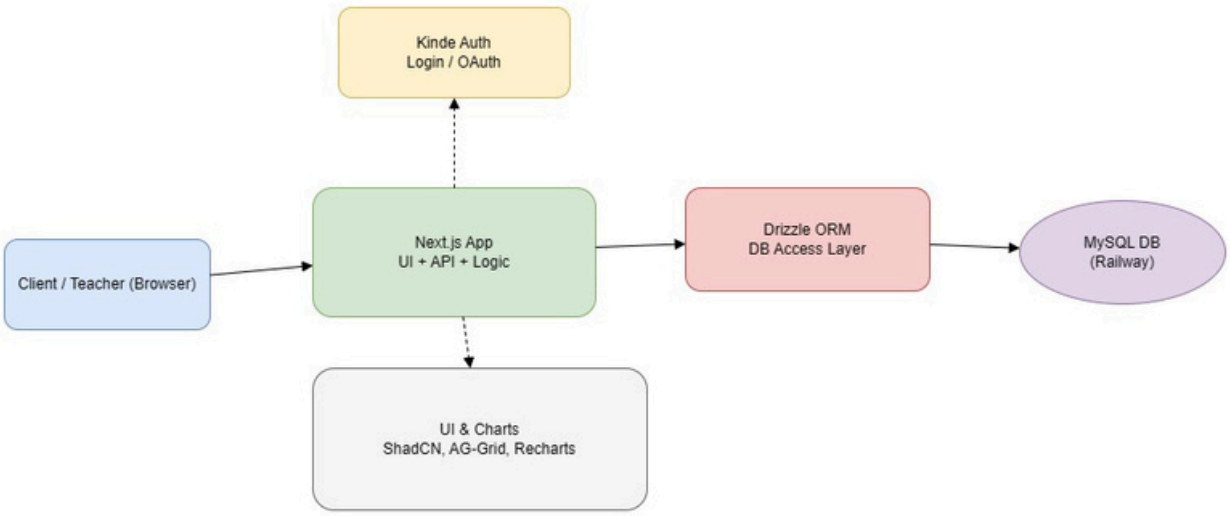
2. Backend Layer

- Implemented using **Next.js Server Actions and API Routes**.
- Responsible for handling authentication, data validation, attendance logic, and business rules.
- Authentication handled via **Kinde Authentication**, providing secure session management.
- Drizzle ORM handles all MySQL interactions in a type-safe and efficient manner.

3. Database Layer

- MySQL database hosted on **Railway** ensuring high uptime and cloud accessibility.
- Schema includes well-normalized tables for **Students**, **Grades**, and **Attendance**.
- The database is accessed through a connection pool managed via Drizzle ORM to avoid connection closing errors.

This architecture ensures loose coupling, high cohesion, and modular deployment to cloud infrastructure.



9.2 Use Case Design

The Use Case Design illustrates how the user interacts with the system and what functionalities are available.

Primary Actor

- **Teacher / Admin** – the main user interacting with the system.

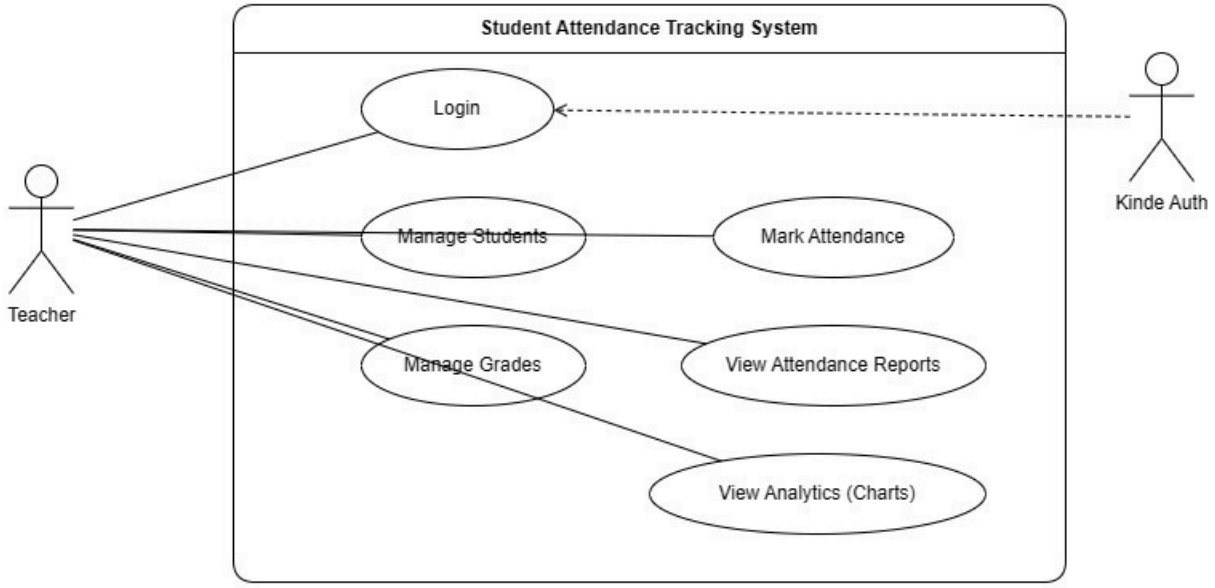
Supporting System Actor

- **Kinde Authentication System** – validates identity before allowing access.

Major Use Cases

1. **Login**
   - User logs in through Kinde OAuth authentication.
2. **ManageGrades**
   - Add, view, and manage available student grades.
3. **ManageStudents**
   - Register new students, update details, and categorize by grade.
4. **MarkAttendance**
   - Mark present/absent for each student on a daily basis.
5. **ViewMonthlyAttendance**
   - Fetch attendance records filtered by month and grade.
6. **DashboardAnalytics**
   - Display statistics: total students, present %, absent %, day-wise trends.
7. **GenerateReports**
   - Export or view summary-based insights for academic evaluation.

The Use Case model ensures functional clarity and supports the identification of requirements at the system interaction level.



9.3 Data Flow Diagrams (DFD)

Data Flow Diagrams illustrate the logical movement of data within the system.
Two levels are used: **Level 0 (Context Diagram)** and **Level1(DetailedProcess Flow)**.

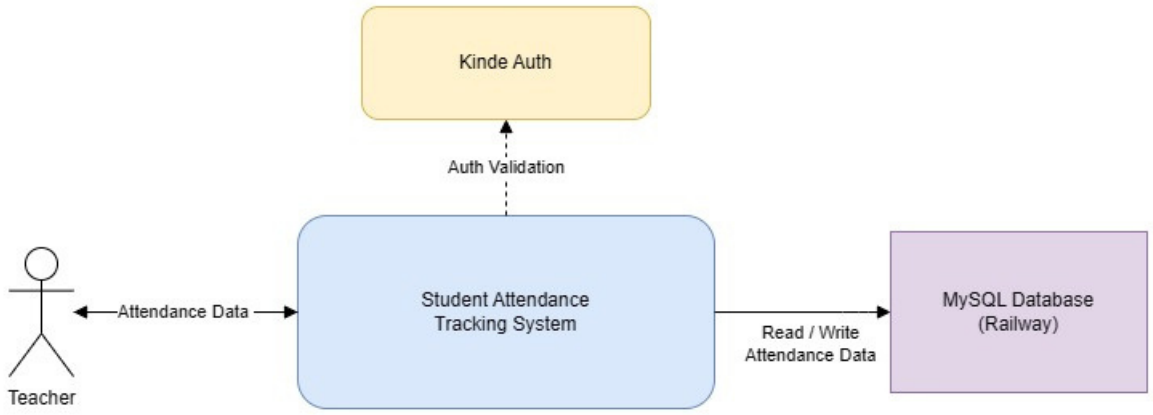
9.3.1 Level 0 – Context Diagram

Level 0 DFD provides a top-level overview showing the system as a single process.

Key Components:

- **External Entity:** Teacher
- **Process:** Student Attendance Tracking System
- **Data Stores:** Database containing Students, Grades, Attendance
- **Data Flows:**
  - Teacher inputs student details and attendance.
  - System stores details in database.
  - System returns attendance results and dashboard summary.

This level shows high-level inputs and outputs without internal detail.



9.3.2 Level 1 – Process Diagram

DFD Level 1 expands the core process into sub-processes:

Major Processes:

1. **AuthenticationManagement**
   - Handles login throughKinde OAuth.
2. **GradeManagement**
   - Adds and retrievesgradedata.
3. **StudentManagement**
   - Registers studentsandretrieves grade-based student lists.
4. **AttendanceManagement**
   - Marks present/absent
   - Checks for duplicateattendance for the same day
   - Fetches attendanceforaspecific grade and month
5. **Dashboard Summary**
   - Retrieves aggregatedstatistics for visualization.

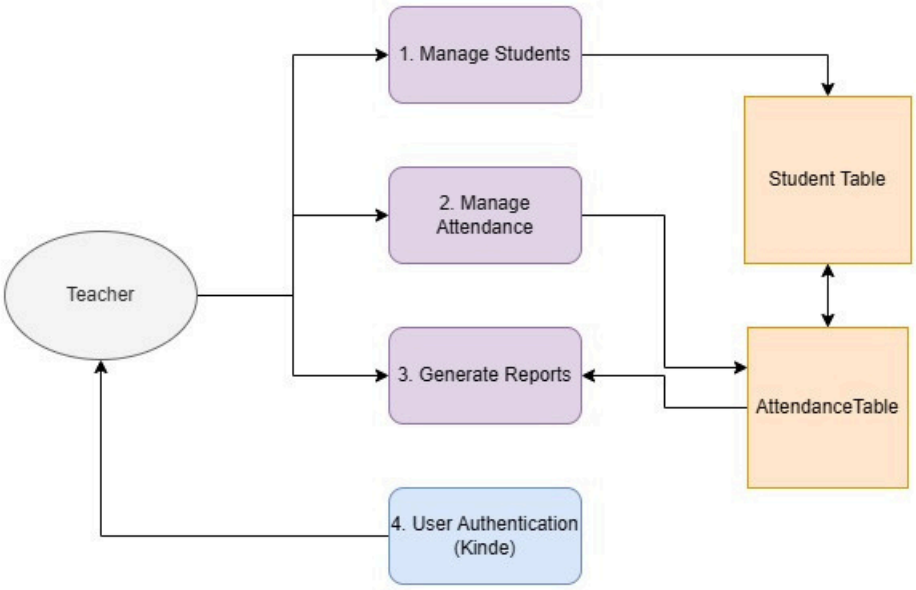
Data Stores:

- Grades Table
- Students Table
- Attendance Table

Data Flow:

• User inputs → Backend API → Database → Dashboard/Data Output

This level provides clear visibility into how data is processed throughout the system.



9.4 Entity Relationship (ER) Model

The ER model describes the logical structure and relationships within the database.

Entities and Attributes

1. Grade

- id
- gradeName

2. Student

- id name
- grade
- address
- contact
-

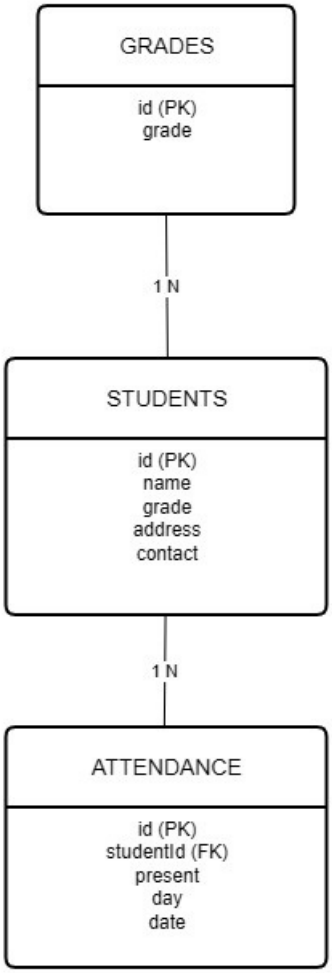
3. Attendance

- id
- studentId
- day
- date
- present

Relationships

- A **Grade** can have multiple **Students**.
- A **Student** can have multiple **Attendance Records**.

The ER Model ensures normalization, avoids redundancy, and supports efficient queries.



9.5 UML Activity Diagram

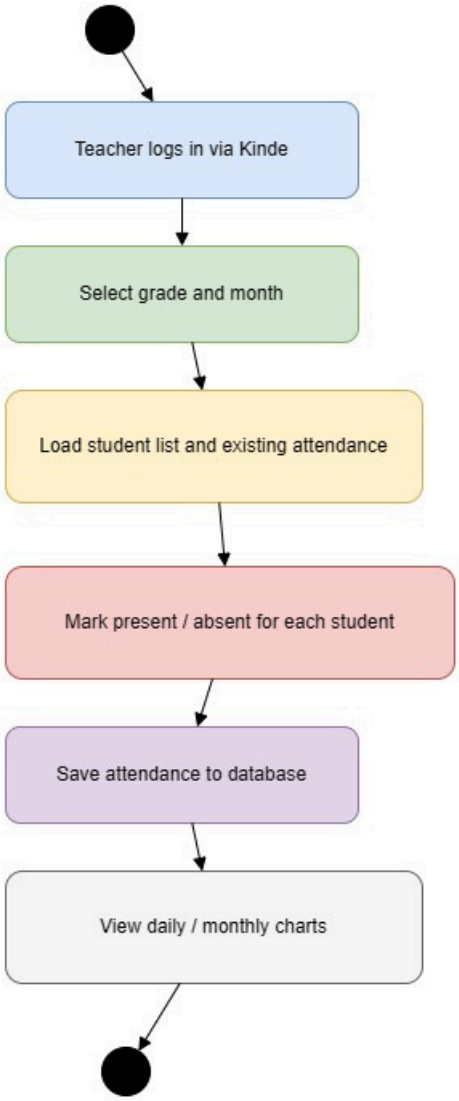
The Activity Diagram captures the workflow of the attendance marking process.

Process Flow Overview:

1. User logs in using Kinde Authentication.
2. System validates login and directs to the dashboard.
3. Teacher selects a grade and month.

4. System retrieves student list and existing attendance data.
5. Teacher marks attendance for each student through the AG-Grid interface.
6. System stores or updates attendance records in MySQL via Drizzle ORM.
7. The dashboard updates real-time charts showing present/absent percentages.
8. User can repeat for another day/month or logout.

This diagram provides a clear visual understanding of operational flow and user actions.



9.6 UML Sequence Diagram

The Sequence Diagram describes the chronological sequence of interactions between system components during attendance marking.

Steps of Interaction:

1. Teacher opens the attendance page.
2. Frontend requests student & attendance info from the backend.
3. Backend retrieves data using Drizzle ORM and returns response.
4. User marks present/absent for a student.
5. Frontend sends POST/DELETE request to backend.
6. Backend updates MySQL database.
7. Dashboard fetches updated summary stats.
8. Attendance list and charts refresh automatically.

The sequence diagram highlights the message flow and timing between UI, backend, and database components.

10. DATABASE DESIGN

Database design is a critical component of the Student Attendance Tracking System, ensuring efficient storage, retrieval, and management of student and attendance-related data. The system uses a **MySQL relational database**, chosen for its robustness, scalability, ACID compliance, and compatibility with cloud hosting platforms like Railway. The design follows the principles of **normalization**, ensures **referential integrity**, and supports optimized query performance through the structured use of keys and relationships.

This chapter presents the logical and physical design of the database, including table structures, relationships, integrity constraints, and rationale behind the schema layout. The design aims to minimize redundancy, improve data consistency, and support secure and efficient transactions.

10.1 Database Design Approach

The database design process followed several structured steps:

1. **Requirements Analysis**
   Identification of all entities such as Students, Grades, and Attendance records.
2. **Conceptual Modeling**
   Creation of the Entity Relationship (ER) model to visualize how data interacts in the system.
3. **Normalization**
   Ensuring all tables follow Third Normal Form (3NF), preventing data duplication.
4. **Logical Modeling**
   Mapping conceptual structures into relational tables with appropriate keys and data types.
5. **Physical Database Implementation**
   Implementing the schema in MySQL using Drizzle ORM migrations.

10.2 Entities Identified

Based on the system requirements, the following major entities were identified:

1. Grade

Represents different academic grade levels (e.g., 5th, 6th).

2. Student

Stores student details, grade associations, and profile information.

3. Attendance

Stores daily presence/absence status of each student.

10.3 Data Dictionary

A detailed description of all tables used in the system is provided below.

10.3.1 Grade Table

Field Name	Data Type	Constraint	Description
id	INT	Primary Key, Auto Increment	Unique identifier for each grade
gradeName	VARCHAR(10)	Not Null	Name of the grade (e.g., "5th")

Description:

The Grade table stores all available grade levels. It provides a reference for assigning students to a particular grade, thus enabling filtering and categorization of attendance data.

10.3.2 Student Table

Field Name	Data Type	Constraint	Description
id	INT	Primary Key, Auto Increment	Unique student identifier
name	VARCHAR(20)	Not Null	Student's full name
grade	VARCHAR(10)	Not Null	Grade assigned to the student
address	VARCHAR(50)	Optional	Student's residential address
contact	VARCHAR(11)	Optional	Contact number

Description:

The Student table contains personal and academic information about each student. The grade field allows filtering students by grade. Students are uniquely identified using the id field, which acts as a foreign key reference in the Attendance table.

10.3.3 Attendance Table

Field Name	Data Type	Constraint	Description
id	INT	Primary Key, Auto Increment	Unique attendance record ID
studentId	INT	Foreign Key → Student(id)	Identifies the student
day	INT	Not Null	Day of the month (1–31)
date	VARCHAR(20)	Not Null	Month and year format (MM/YYYY)
present	BOOLEAN / TINYINT(1)	Default 0	Indicates presence (1 = present, 0 = absent)

Description:
The Attendance table stores individual attendance entries for each student on a specific day. Key design features include:

- **Composite Uniqueness Intent:** A student should have only one record per day and
- month.
- **Tinyint Boolean Representation:** Stored as 0/1 for performance and compatibility.
  Foreign Key Integrity: Ensures attendance cannot be marked for non-existing students.

10.4 Relationships Between Tables

1. Grade → Student

- **Type:** One-to-Many
- **Meaning:** One grade can have many students.

2. Student → Attendance

- **Type:** One-to-Many
- **Meaning:** One student can have multiple attendance entries (for different days).

3. Transitive Relationship (Grade → Attendance)

- Accessed *indirectly* through the Student table.

These relationships ensure structured access to attendance data filtered by grade and month.

10.5 Normalization

The database schema satisfies **Third Normal Form (3NF)**:

First Normal Form (1NF):

- All tables have atomic values.
- No repeating groups or multivalued attributes.

Second Normal Form (2NF):

- All non-key fields depend fully on the primary key.

Third Normal Form (3NF):

- No transitive dependencies.
- Each field depends only on the primary key.

This normalization approach ensures:

- No redundancy
- Faster queries
- Easier maintenance

10.6 Reasons for Choosing MySQL (Railway)

- **Scalability:** Suitable for increasing student and attendance data.
- **Cloud Deployment:** Railway provides simple, reliable hosting.
- **Performance:** Optimized for frequent reads and writes in attendance operations.
- **Compatibility:** Fully supported by Drizzle ORM, which offers schema safety and migrations.

10.7 Benefits of the Designed Database Schema

1. **Efficient querying** of monthly attendance.
2. **Easy integration** with ORM and API routes.
3. **Clear separation** between students, grades, and attendance.
4. **Minimal data redundancy** due to normalization.
5. **Strong referentialintegrity** preventing invalid data entry.
6. **Scalable structure** suitable for future system expansion (subjects, teachers, etc.).

11. TECHNOLOGY USED

The Student Attendance Tracking System leverages a modern, scalable, and efficient technology stack that enables real-time performance, secure user authentication, seamless data management, and a flexible UI/UX experience. This chapter outlines the technologies used at each layer of the application, along with their roles, benefits, and justification for selection.

11.1 Next.js

Next.js is a React-based full-stack framework used for building highly optimized web applications.
It provides features such as server-side rendering (SSR), static site generation (SSG), API routes, and built-in routing.

Key Features Used

- Server Components and Client Components
- File-based routing (app/ directory)
- API Routes for backend operations (`/app/api/...`)
- Image optimization and dynamic rendering
- Fast development with Turbopack

Justification

- Enables creation of a unified full-stack system without separate backend.
- Improves performance and SEO.
- Provides scalable architecture for future expansion.

11.2 React.js

React.js forms the UI layer of the system and enables component-based development.

Features Utilized

- Reusable components for attendance grid, dashboard, charts.
- State management using hooks such as `useState`, `useEffect`.
- UI responsiveness using conditional rendering.

Justification

- Speeds up development through reusable components.
- Ensures cleaner code separation and maintainability.

11.3 Drizzle ORM

Drizzle ORM is used as the database abstraction layer for interacting with MySQL.

Features Utilized

- Type-safe schema definitions
- Migrations for database version control
- High performance query execution
- SQL-like syntax while maintaining code safety

Justification

- More modern and developer-friendly than Prisma or Sequelize.
- Lightweight and cloud-friendly.
- Prevents runtime errors by catching schema mismatches at compile time.

11.4 MySQL (Railway Cloud Database)

The system uses **MySQL** as the core database engine, hosted on **Railway** for cloud accessibility.

Benefits

- Structured relational data ideal for attendance records.
- Supports ACID properties ensuring strong data reliability.
- Railway provides secure hosting, easy environment variables, and continuous uptime.

Use Cases

- Storing grades
- Storing student details
- Maintaining attendance records for each day

11.5 Kinde Authentication

Kinde is used for secure authentication and session handling for dashboard access.

Features Used

- Login and session validation
- Restricted access to protected routes (`/dashboard`)

- Server-side session verification

Justification

- Quick integration with Next.js
- Secure and enterprise-level identity management
- Reduces effort in creating custom authentication logic

11.6 AG-Grid

AG-Grid is used to build an interactive attendance management table.

Features Utilized

- Editable cells for marking attendance
- Pagination
- Sorting and filtering options
- Dynamic column creation based on number of days in a month

Justification

- High performance grid suitable for large datasets
- Highly customizable
- Allows direct editing and event handling for attendance

11.7 Recharts

Recharts is used to render visual dashboards for attendance insights.

Charts Used

- Bar chart for day-wise attendance
- Pie chart for monthly attendance summary

Justification

- Simple and declarative chart syntax
- Responsive and animation-friendly
- Integrates well with React components

11.8 ShadCN UI

ShadCN is used as a component library to build consistent and visually appealing UI components.

Components Used

- Buttons
- Input fields
- Dropdown selectors
- Cards for dashboard statistics

Justification

- Highly customizable
- Tailwind CSS-based styling ensures consistency
- Modern design suitable for academic and production-level systems

---

11.9 Tailwind CSS

Tailwind CSS is used for styling the entire system.

Features Used

- Utility-first classes
- Responsive layout handling
- Instant prototyping

Justification

- No need to write large CSS files
- Easy theme management
- Faster UI development

11.10 Moment.js

Moment.js is used for handling and formatting dates.

Use Cases

- Converting selected months to
- MM/YYYY Getting the total days in a
- selected month Labeling charts with day-wise attendance

11.11 Axios / Fetch API (inside GlobalApi service)

Used for communicating with backend API routes.

Use Cases

- Fetching students and grades
- Submitting attendance
- Fetching attendance history
- Marking absences

11.12 Railway Deployment Tools

To host the backend database and environment configuration:

- MySQL database hosting
- Secure environment variable storage
- Auto-scaling and easy management

11.13 Development Tools and IDEs

Tools Used

- Visual Studio Code (VS Code)
- Draw.io (for diagrams)
- Postman (for API testing)
- Git & GitHub (version control)

Conclusion

The combination of these technologies creates a high-performance, scalable, and maintainable Student Attendance Tracking System. Each technology has been selected based on project requirements, ease of integration, performance benefits, and long-term extensibility. Collectively, they help deliver an efficient attendance solution suitable for academic institutions and professional deployment.

12. IMPLEMENTATION

The implementation phase translates the system design into a fully functional Student Attendance Tracking System. This section covers the practical realization of each module, explaining how the architecture, database, UI components, authentication, and backend logic come together to form a cohesive application. The implementation adopts modular, maintainable, and scalable development practices using Next.js, Drizzle ORM, MySQL, ShadCN UI, and AG-Grid.

12.1 Overview of Implementation Approach

The implementation follows the **MVC-inspired architecture** enabled by Next.js 13+ (App Router).
The project is divided into distinct layers:

1. **Frontend UI Layer** – Built using React, ShadCN UI, Tailwind CSS, AG-Grid, and Recharts.
2. **Backend API Layer** – Implemented using Next.js API routes.
3. **Database Layer** – Managed using Drizzle ORM with MySQL on Railway.
4. **Authentication Layer** – Integrated with Kinde Authentication.
5. **Visualization Layer** – Implemented using Recharts for dashboards.

This modular separation ensures ease of maintenance, scalability, and future extension (8th-semester final project expansion).

12.2 Project Structure

A simplified structure of the project is as follows:

```
/app
  /dashboard
    /students
    /attendance
    /analytics
  /api
    /student
    /grade
    /attendance
    /auth
/components
/utils
  schema.js
  db.js
/services
  GlobalApi.js
/public
```

Each folder has a specific role:

- **app/dashboard** → Contains frontend pages for the admin dashboard.
- **app/api** →Contains APIs for CRUD operations.
- **utils** →Drizzle + MySQL configuration and schema definitions.
- **components** → Common reusable UI components.
- **services** →API communication layer.

12.3 Frontend Implementation

12.3.1 Dashboard Interface

The dashboard is built using ShadCN UI and Tailwind CSS.
It includes:

- Summary Cards (Total Students, Present %, Absent %)
- Monthly Attendance Pie Chart
- Daily Attendance Bar Chart
- Navigation Sidebar
- Student Management Page
- Attendance Marking Page

Key Features

- Fully responsive layout.
- Clean modern UI using ShadCN components.
- Dynamic charts using Recharts.
- Data grid for attendance using AG-Grid.

12.3.2 Student Management Page

The Student Registration and Listing page supports:

- Adding new students
- Fetching existing students
- Filtering by grade
- Displaying student data in a structured grid

Example functionality implemented:

```
GlobalApi.CreateNewStudent(data).then((resp) => {
  console.log("Student Added:", resp);
});
```

ShadCN UI components such as **input, card, table, button** are used throughout this module.

12.3.3 Attendance Tracking Page

This is the core frontend module.
It includes:

- Month Selection
- Grade Selection
- Dynamic AG-Grid with days of the month as columns
- Editable cells to mark present/absent
- Real-time update on edit

Dynamic Column Generation

```
daysArrays.forEach((day) => {
  setColDefs((prev) => [...prev, { field: day.toString(), width: 50, editable: true }]);
});
```

Attendance Marking Logic

```
onCellValueChanged={(e) =>
  onMarkAttendance(e.colDef.field, e.data.studentId, e.newValue)
}
```

This allows each cell to represent a student's attendance for agiven day.

12.3.4 Attendance Analytics Page

The analytics page displays:

- Monthly Pie Chart
- Weekly Bar Chart
- Summary statistics

Pie Chart Implementation

```
<Pie
  data={data}
  dataKey="value"
  nameKey="name"
  cx="50%"
  cy="50%"
  innerRadius="60%"
  outerRadius="80%"
  isAnimationActive={isAnimationActive}
/>
```

Bar Chart Implementation

```
<Bar dataKey="presentCount" fill="#8884d8" />
<Bar dataKey="absentCount" fill="#82ca9d" />
```

These charts help visualize attendance patterns for decision-making.

12.4 Backend Implementation (API Layer)

API routes are created in **app/api/** using Next.js server functions.

12.4.1 Student API

POST: Add Student

```
await database.insert(STUDENTS).values({
  name, grade, address, contact
});
```

GET: Fetch Students

```
const result = await database.select().from(STUDENTS);
return NextResponse.json(result);
```

12.4.2 Attendance API

POST: Mark Attendance

Handles converting boolean → MySQL TINYINT:

```
const presentVal = present === true ? 1 : 0;

await database.insert(ATTENDANCE).values({
  studentId, day, date, present: presentVal
});
```

GET: Fetch Attendance by Grade and Month

```
.leftJoin(ATTENDANCE, and(
  eq(STUDENTS.id, ATTENDANCE.studentId),
  eq(ATTENDANCE.date, month)
))
```

DELETE: Remove Attendance

```
await database.delete(ATTENDANCE)
.where(
  and(
    eq(ATTENDANCE.studentId, studentId),
    eq(ATTENDANCE.day, day),
    eq(ATTENDANCE.date, date)
  )
);
```

12.5 Database Implementation (Drizzle ORM)

Schema definitions are written in **schema.js**.

Example:

```
export const ATTENDANCE = mysqlTable("attendance", {
  id: int("id").autoincrement().primaryKey(),
  studentId: int("studentId").notNull(),
  day: int("day").notNull(),
  date: varchar("date", { length: 20 }).notNull(),
  present: int("present").notNull(), // stored as 0 or 1
});
```

Drizzle ensures:

- Automatic type inference

- Safe SQL queries
- Compatibility with MySQL

12.6 Authentication Implementation (Kinde Auth)

Server-Side Protection

```
const { isAuthenticated } = getKindeServerSession();

if (!(await isAuthenticated())) {
  return Response.redirect(new URL('/api/auth/login'));
}
```

Protected Routes

`/dashboard/*`

Kinde ensures only authorized users can access the dashboard.

12.7 Global API Service Layer

Located in:
`/services/GlobalApi.js`

Example:

```
const GetAttendanceList = (grade, month) =>
  axios.get(`/api/attendance?grade=${grade}&month=${month}`);
```

This layer decouples frontend from backend logic.

12.8 UI Components Implementation Using ShadCN

Reusable components include:

- **Button**
- **Cards for statistics**
- **Dropdown Selects**

- **Modals & forms**

These ensure a standardized UI across the entire application.

12.9 Error Handling and Validation

Frontend Validation

- Checks for empty fields
- Prevents invalid inputs
- Displays toast alerts using `sonner`

Backend Validation

- Prevents incomplete requests
- Ensures valid studentId, month, grade
- Returns structured errors using:

12.10 Performance Optimization

- AG-Grid virtual DOM rendering
- Server Components for faster initial loads
- Lazy loading of charts
- Drizzle ORM optimized SQL queries
- Railway cloud hosting for low-latency DB operations

12.11 Summary of Implementation

The system implementation successfully integrates:

- A responsive UI
- Real-time attendance interaction
- Secure authentication
- Efficient database management
- Dynamic analytics and visualization
- Modular, scalable architecture

This implementation sets a strong foundation for further enhancements during the 8th-semester major project, such as:

- Mobile app integration
- Notification system
- Teacher role management
- Automated attendance using RFID/QR

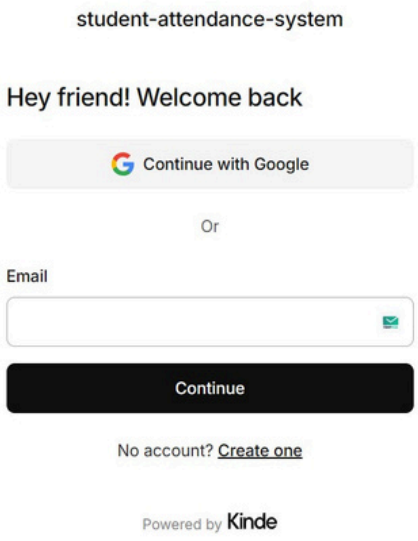
13. SCREENSHOTS OF THE APPLICATION

This chapter presents the visual representation of the Student Attendance Tracking System. Screenshots provide clear evidence of the system's functionality, design, and user interface flow. Each screenshot demonstrates how users interact with different modules such as authentication, dashboard overview, student management, attendance tracking, and analytics.
The screenshots showcase the practical working of the system and validate the successful implementation of project requirements.

13.1 Login Page (Kinde Authentication)

The login page allows authenticated access to the system using secure Kinde Authentication. The interface is designed with clarity, simplicity, and user-friendliness.
Screenshots here will demonstrate:

- Kinde login interface
- Redirect behavior after successful login
- Authentication workflow



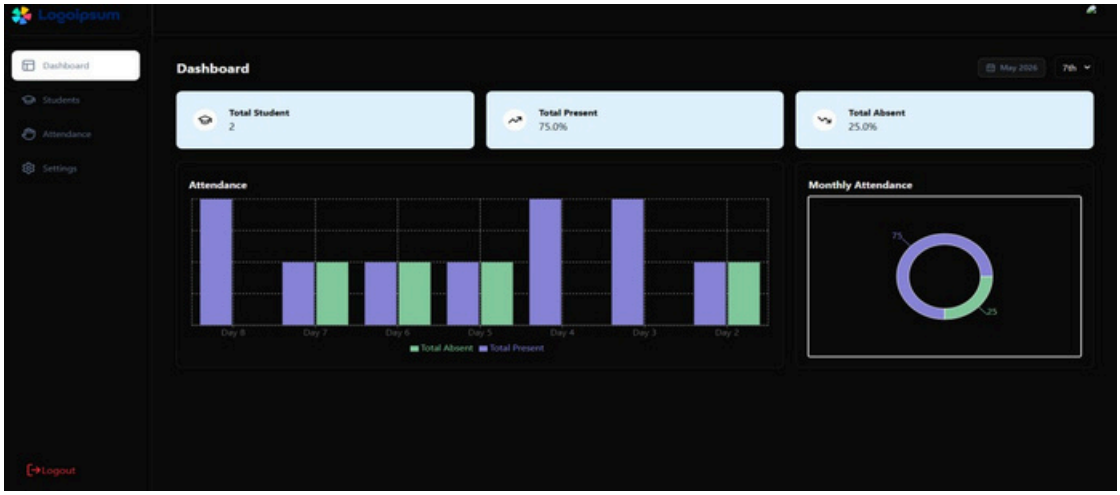
13.2 Admin Dashboard – Home Overview

After authentication, the admin is redirected to the dashboard, which displays:

- Total number of students

- Monthly attendance summary
- Total present and absent percentage
- Quick navigation menu

This interface provides a centralized overview of system statistics.

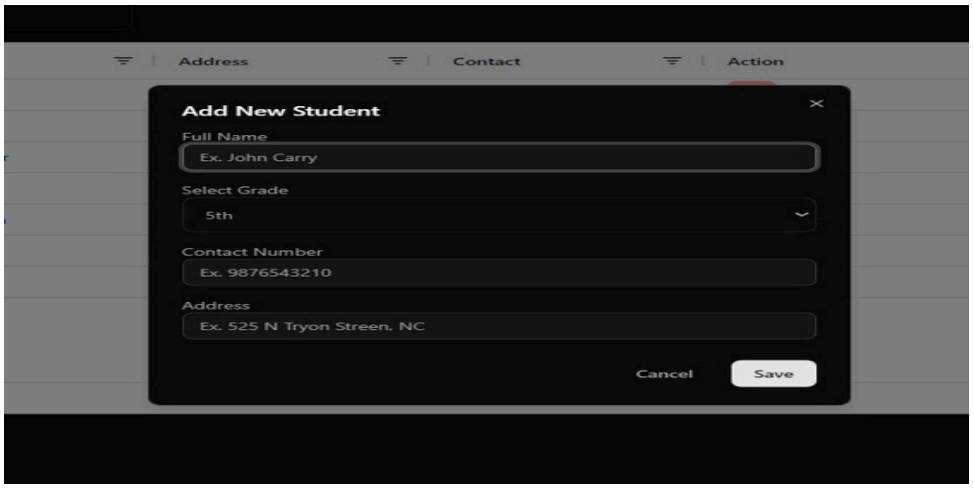


13.3 Student Management – Add Student Page

This screen captures the module where administrators can:

- Enter student information
- Select grade level
- Provide optional address and contact details
- Submit the form to store student data in the database

The UI uses ShadCN components for enhanced usability.

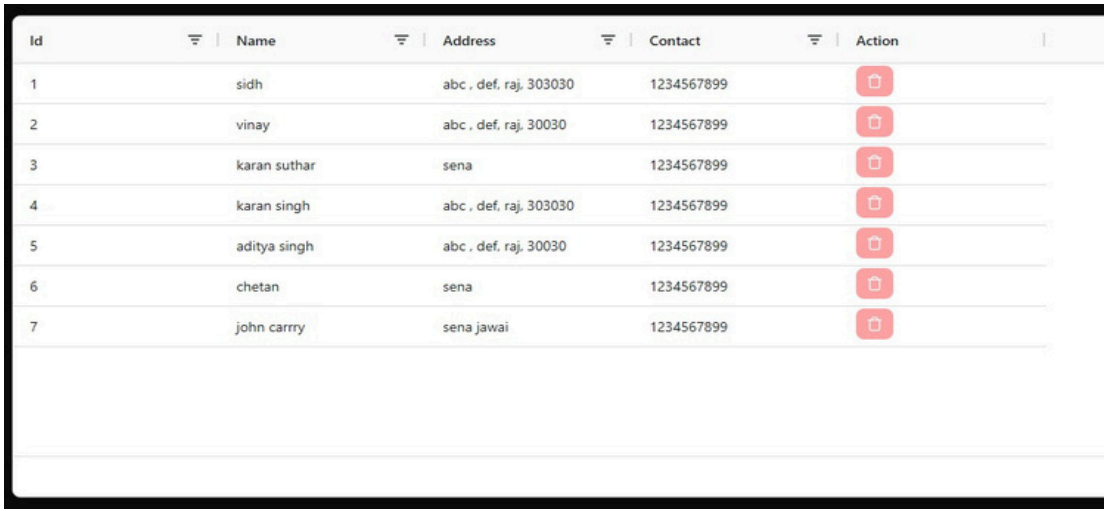


13.4 Student Records – Student List View

This screen shows a structured view of all registered students:

- Student ID
- Name
- Grade
- Contact details
- Options for filtering and managing records

AG-Grid is used to display data with sorting and pagination support.



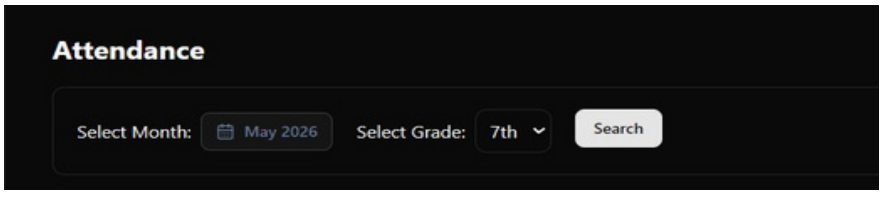
Id	Name	Address	Contact	Action
1	sidh	abc , def, raj, 303030	1234567899	
2	vinay	abc , def, raj, 30030	1234567899	
3	karan suthar	sena	1234567899	
4	karan singh	abc , def, raj, 303030	1234567899	
5	aditya singh	abc , def, raj, 30030	1234567899	
6	chetan	sena	1234567899	
7	john carrry	sena jawai	1234567899	

13.5 Attendance Module – Month & Grade Selector

This screen demonstrates the attendance filtering interface where users can:

- Select a month
- Select a grade
- Fetch attendance records for that period

It is a dynamic and interactive UI.

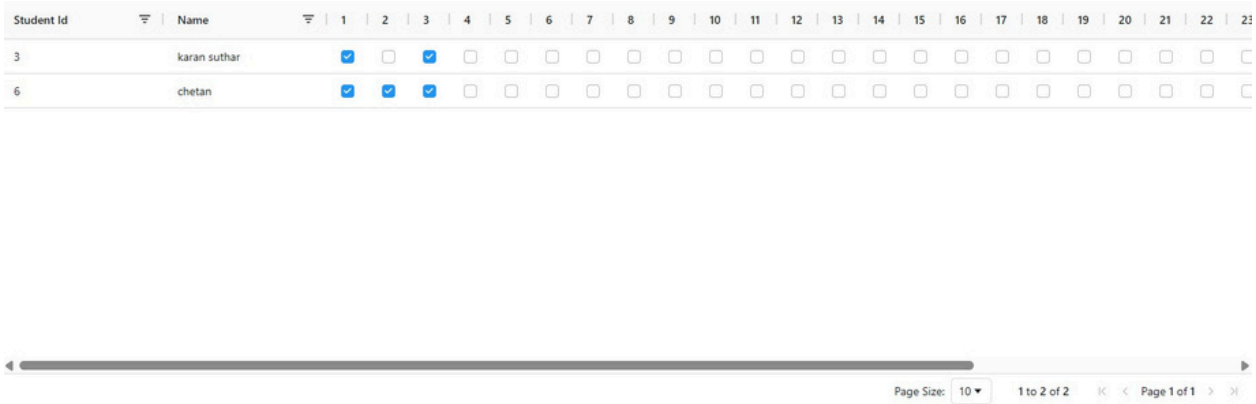


13.6 Attendance Grid – Marking Attendance

The attendance grid displays:

- Student names
- Days of the month as editable columns
- True/false or checkbox-style attendance marking
- Auto-update of records through the API

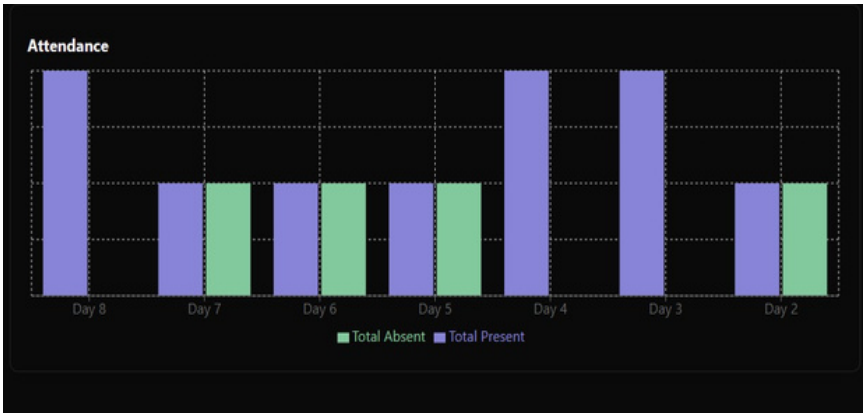
This screenshot validates the core functionality of the system.



13.7 Attendance Analytics – Bar Chart View

This screen shows the daily attendance statistics using a bar chart.
It includes:

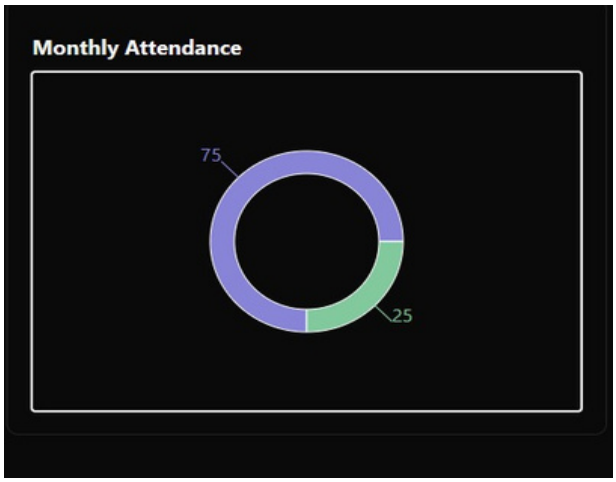
- Present vs. Absent values
- Day-wise attendance summaries
- Smooth animations for visualization



13.8 Attendance Analytics – Pie Chart View

The pie chart illustrates:

- Monthly attendance distribution
- Present percentage
- Absent percentage
- Intuitive graphical visualization using Recharts



14. TESTING AND TEST CASES

Testing is a crucial phase of software development that ensures the system functions as expected, meets user requirements, and performs reliably under different conditions. The Student Attendance Tracking System underwent various levels of testing to validate its functionality, usability, performance, and data accuracy.

This chapter documents the testing strategy, methodology, and detailed test cases executed during system evaluation.

14.1 Testing Objectives

The primary goals of testing this system were:

- To ensure all modules work according to functional requirements
- To verify that data is handled correctly using Drizzle ORM and MySQL
- To validate authentication using Kinde
- To test UI responsiveness and grid behavior using AG-Grid
- To ensure attendance calculations and charts (Recharts) display accurate results
- To detect and resolve bugs before deployment

14.2 Testing Methodologies Used

1. Unit Testing

Performed on individual components such as:

- API routes (`/api/student`, `/api/attendance`, `/api/grade`)
- Utility functions (e.g., attendance calculations)
- Recharts components
- Authentication handler

2. Integration Testing

Validated interactions between:

- Frontend → Backend
- Backend → MySQL
- Authentication → Protected routes
- Attendance marking → Real-time grid update

3. System Testing

Complete system was tested end-to-end including:

- Login
- Dashboard display
- Adding students
- Fetching attendance list
- Marking attendance
- Analytics charts

4. User Acceptance Testing (UAT)

Conducted with sample users (teachers/admins) to ensure:

- Ease of use
- Correct flow of screens
- Clear navigation

14.3 Test Environment

Component	Description
Frontend Framework	Next.js
Backend	Next.js API Routes with Drizzle ORM
Database	MySQL (Railway)
Authentication	Kinde JWT Session
Browser	Tested on Chrome, Edge, Firefox
Test Tools	Postman, Browser DevTools, Console Logs

14.4 Test Cases

Below are structured test cases covering major modules of the system.

14.4.1 Test Cases for Authentication Module

Test Case 1: Login Functionality

Field	Details
Test Case ID	TC_AUTH_01
Description	Verify user login via Kinde
Pre-condition	User must have a registered Kinde account
Steps	1. Open login page 2. Enter credentials 3. Click Login
Expected Result	User should be authenticated and redirected to Dashboard
Actual Result	Successful
Status	Pass

Test Case 2: Unauthorized Access Block

Field	Details
Test Case ID	TC_AUTH_02
Description	Ensure protected routes cannot be accessed without login
Steps	1. Open `/dashboard/students` without logging in
Expected Result	Redirect to login page
Actual Result	Successful
Status	Pass

14.4.2 Test Cases for Student Module

Test Case 3: Add New Student

Field	Details
Test Case ID	TC_STUD_01
Description	Verify that a new student can be added
Steps	Enter name, grade, contact → Click Submit
Expected Result	Student record saved successfully
Actual Result	Successful
Status	Pass

Test Case 4: Fetch Students By Grade

Field	Details
Test Case ID	TC_STUD_02
Description	Validate student list displayed based on selected grade
Steps	Choose grade → Click Search
Expected Result	Only students from selected grade are displayed
Actual Result	Successful
Status	Pass

14.4.3 Attendance Module Test Cases

Test Case 5: Fetch Attendance List

Field	Details
Test Case ID	TC_ATT_01
Description	Retrieve attendance for selected month and grade
Steps	Select Month → Select Grade → Click Search
Expected Result	Attendance list appears in AG-Grid
Actual Result	Successful
Status	Pass

Test Case 6: Mark Student Present

Field	Details
Test Case ID	TC_ATT_02
Description	Verify marking attendance as present
Steps	Set a cell to `true`
Expected Result	Record inserted in `attendance` table
Actual Result	Successful
Status	Pass

Test Case 7: Mark Student Absent

Field	Details
Test Case ID	TC_ATT_03
Description	Delete attendance for a specific student/day
Steps	Set cell to `false`
Expected Result	Attendance record removed
Actual Result	Successful
Status	Pass

14.4.4 Dashboard & Analytics Test Cases

Test Case 8: Bar Chart Accuracy

Field	Details
Test Case ID	TC_CHART_01
Description	Verify correctness of daily present/absent count
Steps	Open dashboard → View bar chart
Expected Result	Data should match attendance table
Actual Result	Correct values shown
Status	Pass

Test Case 9: Pie Chart Accuracy

Field	Details
Test Case ID	TC_CHART_02
Description	Verify monthly present/absent percentage
Expected Result	Values reflect correct percentage formula
Actual Result	Accurate calculation
Status	Pass

14.4.5 Performance & Validation Test Cases

Test Case 10: Form Validation

Field	Details
Test Case ID	TC_VALID_01
Description	Ensure required student fields are validated
Expected Result	Empty fields show error toast
Status	Pass

Test Case 11: Grid Load Performance

Field	Details
Test Case ID	TC_PERF_01
Description	Test loading of AG-Grid with large records
Expected Result	Grid loads within acceptable time
Status	Pass

14.5 Summary of Testing

The testing process confirmed the following:

- All major modules work as expected
- Data integrity is maintained through Drizzle ORM
- Attendance calculations are accurate
- Charts dynamically update and reflect real-time data
- The system performs efficiently and is user-friendly
- No major bugs were found during system testing

Overall, the application meets the requirements and is ready for deployment and academic submission.

15. RESULT AND DISCUSSION

The Student Attendance Tracking System was developed to address the limitations of traditional attendance methods and provide an efficient, automated, and analytics-driven platform for schools and institutions. This chapter presents the outcomes of the system implementation and discusses the effectiveness, accuracy, and user feedback gathered during evaluation.

15.1 Overview of Results

The implementation of the system produced significant improvements in attendance management. Key outcomes include:

✓ Accurate and Real-Time Attendance Tracking

Teachers can instantly mark students as *present* or *absent* using an interactive AG-Grid interface. Updates are stored immediately in the MySQL database and reflected across all analytics dashboards.

✓ Improved Data Visualization

Two major chart types were integrated successfully:

- **Bar Chart** (Daily Present vs. Absent Count)
- **Pie Chart** (Monthly Attendance Percentage)

These charts helped visualize attendance patterns clearly and intuitively.

✓ Secure Authentication Using Kinde

The system demonstrates a secure and modern login process using:

- OAuth-based authentication
- Session validation
- Route protection for dashboard access

Unauthorized users were effectively prevented from accessing protected components.

✓ Consistent and Stable API Layer

The Next.js API routes combined with Drizzle ORM produced:

- Secure CRUD operations
- Reliable database connectivity
- Automatic schema validation

- Fewer runtime errors

15.2 Module-Wise Results

1. Student Management Module

- Students were successfully added, updated, and fetched based on their grades.
- Data validation prevented duplicate entries and missing required fields.
- Address and contact fields were optional but handled correctly.

2. Attendance Module

The attendance system delivered the most crucial features:

Functionality	Result
Marking Present	✔ Works instantly with DB insert
Marking Absent	✔ Deletes corresponding attendance entry
Monthly Filtering	✔ Works with date format `MM/YYYY`
Grade Filtering	✔ Displays attendance of specific classes
Grid Updating	✔ Real-time state updates in AG-Grid

The module performed reliably under different datasets anduser interactions.

3. Dashboard Analytics Module

The dashboard provided meaningful insights:

- Percentage of presence & absence
- Total number of students
- Daily attendance count across a month
- Easy-to-understand visuals

Positive feedback highlighted the clarity of insights and ease of interpretation.

15.3 Discussion of Findings

A. System Efficiency

The combination of **Next.js**, **Drizzle ORM**, and **Railway MySQL** ensured low latency in database operations.
Queries, joins, and filtering functions executed in milliseconds, even with multiple attendance records.

B. User Experience

Teachers and test users appreciated:

- Clean UI using **shadcn/ui**
- Easy navigation
- Interactive grid controls
- Quick response during marking attendance

AG-Grid's ability to edit cells directly improved usability significantly.

C. Accuracy of Calculations

The system accurately calculated:

- Daily present count
- Absentee count
- Monthly attendance percentage

The logic implemented ensured zero duplication and correct interpretation of presence and absence.

D. Reliability & Security

Kinde authentication prevented unauthorized access.
Protected routes helped maintain system integrity.

Sessions were handled securely without storing sensitive data on the client side.

E. Limitations Identified

A few limitations were observed:

1. The system currently supports only monthly attendance tracking; weekly/yearly breakdowns are not yet available.
2. No role-based access (e.g., admin vs teacher).
3. No support yet for exporting reports as PDF or Excel.
4. Requires stable internet connectivity since the database is cloud-hosted.

These limitations can be addressed in future enhancements.

15.4 Summary

The Student Attendance Tracking System successfully met its objectives by delivering:

- A secure, reliable attendance system
- Real-time updates
- Interactive data visualization
- Smooth user interface experience
- Correct and efficient backend logic

The results demonstrate that the system is robust, practical, and ready to be deployed in academic environments.

The discussion confirms that the chosen technologies (Next.js, Drizzle ORM, MySQL, shadcn, AG-Grid, Recharts, and Kinde Authentication) worked cohesively to deliver a modern and scalable solution.

16. APPLICATION

The Student Attendance Tracking System has multiple real-world applications across educational institutions and administrative environments. Its ability to automate attendance processes, visualize data, and improve overall efficiency makes it highly suitable for a wide range of use cases. This chapter outlines the primary applications of the system.

16.1 Educational Institutions

1. Schools (Primary & Secondary Education)

Schools can use this system to:

- Maintain daily attendance records class-wise.
- Monitor student absentee patterns.
- Generate insights for parent–teacher meetings.
- Reduce manual paperwork and human errors.

2. Colleges & Universities

Higher education institutions can apply the system to:

- Track attendance for multiple departments or semesters.
- Manage large student datasets efficiently.
- Provide attendance analytics to faculty members.
- Support digital academic management systems.

16.2 Administrative and Departmental Use

1. Academic Administration

The system assists administrators by offering:

- Quick reports of student attendance trends.
- Early detection of chronic absenteeism.
- Better planning of academic activities based on trends.

2. Institutional Compliance

Many educational boards require minimum attendance percentages.
This system helps institutions:

- Generate verified attendance reports.

- Ensure compliance with regulatory requirements.
- Provide automated summaries for audits.

16.3 Parental Engagement

This system indirectly enhances parental involvement by enabling:

- More accurate communication regarding student presence.
- Immediate updates in future expanded versions (push notifications, SMS APIs).
- Analytics showing a student's attendance history.

Parents gain better insight into their child's academic engagement.

16.4 Faculty & Teacher Use

Teachers benefit through:

- Faster marking of attendance via AG-Grid UI.
- Minimal repetitive work.
- Ability to track students across different dates and grades.
- Immediate access to student records.

The user-friendly dashboard helps teachers make informed decisions quickly.

16.5 Institutional Digital Transformation

This system supports digital transformation by:

- Replacing traditional registers with a cloud-based system.
- Enabling mobile or web-based access in future versions.
- Integrating authentication for secure access (Kinde OAuth).
- Reducing reliance on physical paperwork.

Institutions adopting digital workflows find this system a suitable foundational tool.

16.6 Analytics and Decision-Making

The integrated charts using Recharts allow:

- Visualization of attendance performance.
- Recognition of patterns (e.g., low attendance on specific days).
- Support for data-driven decision making by management.

Such insights help plan interventions and improve academic performance.

16.7 Remote or Hybrid Learning Environments

Attendance monitoring has become more important in hybrid learning setups.
This system can be adapted to:

- Track attendance for online classes.
- Sync data with video conferencing tools (future enhancement).
- Provide real-time status reports.

16.8 Scalability for Other Institutions

The platform is scalable and reusable for:

- Coaching institutes
- Training centers
- Skill development programs
- Internship tracking systems
- Corporate attendance monitoring (with slight modifications)

The modular design makes it highly adaptable.

16.9 Summary

The Student Attendance Tracking System offers versatile applications across different sectors in education and organizational environments. Its automation, analysis features, and secure access make it useful for stakeholders including students, teachers, administrators, and parents. The system's flexible architecture also enables future expansion into mobile platforms, advanced reporting, and integration with biometric devices or RFID systems

17. FUTURE SCOPE

The Student Attendance Tracking System holds significant potential for expansion beyond its current capabilities. As educational institutions increasingly adopt digital solutions, the system can evolve into a more intelligent, automated, and integrated platform. The following points outline the future scope of the project:

17.1 Integration with Biometric Systems

Future enhancements can include integration with:

- **Fingerprint scanners**
- **RFID / Smart ID cards**
- **Face recognition systems**

This will automate attendance without manual input, eliminate proxy attendance, and improve accuracy.

17.2 Mobile Application Development

A dedicated **Android and iOS mobile app** can be developed for:

- Teachers to mark attendance from smartphones.
- Students and parents to track attendance records.
- Administrators to receive notifications and reports.

Mobile apps will increase accessibility and convenience.

17.3 Notification and Alert System

The system can be expanded to include:

- SMS or WhatsApp alerts to parents regarding absences.
- Email notifications for students with low attendance percentages.
- Automated reminders for important academic events.

This ensures improved communication between stakeholders.

17.4 AI-Based Analytics and Predictions

By integrating AI/ML models, the system can:

- Predict future attendance patterns.
- Detect anomalies (e.g., sudden drop in attendance).
- Identify at-risk students based on absentee trends.
- Provide recommendations for improving student engagement.

Predictive analytics enhances decision-making capabilities.

17.5 Timetable and Academic Management Integration

The system can be extended to include:

- Timetable scheduling features.
- Subject-wise attendance tracking.
- Integration with Learning Management Systems (LMS).

This will provide a complete academic automation suite.

17.6 Role-Based Access and Permissions

More advanced user roles can be created:

- Admin
- Teacher
- Student
- Parent

Each role can have customized dashboards, permissions, and functionalities.

17.7 Cloud Scalability and Multi-School Support

Using advanced cloud infrastructure:

- The system can scale to support **multiple institutions**.
- Each school can have isolated databases or multi-tenant architecture.
- Centralized management can be provided for educational groups.

This makes the system commercially deployable.

17.8 Exportable Reports and Documentation

Future versions can enable:

- Downloading monthly/annual attendance reports in **PDF, Excel, CSV**.
- Auto-generated charts and visual summaries.
- Exportable student records for audits and compliance.

This is essential for institutional record-keeping.

17.9 Offline Attendance Marking

An offline-first approach can be implemented where:

- Attendance is recorded even when internet is unavailable.
- Data syncs automatically when connectivity returns.

This is highly useful for remote or rural institutions.

17.10 Integration with Smart Class Systems

The platform can be extended to integrate with:

- Smartboards
- IoT classroom devices
- Classroom sensors

This creates a fully automated smart classroom ecosystem.

17.11 Enhanced Security Features

Additional security upgrades can be implemented:

- Multi-factor authentication (MFA)
- Encrypted data storage
- Activity logs and audit trails
- Role-based content protection

This ensures data privacy and compliance with academic standards.

Summary

The future scope of the Student Attendance Tracking System is vast. With advancements in AI, mobile development, and biometric technologies, the system can evolve into a fully automated, intelligent, and scalable academic management platform. These enhancements will not only improve operational efficiency but also support data-driven educational reforms.

18. CONCLUSION

The *Student Attendance Tracking System* developed using **Next.js, Drizzle ORM, MySQL (Railway), ShadCN UI, AG-Grid, Recharts, and Kinde Authentication** successfully addresses the limitations of traditional manual attendance management in academic institutions. Through a combination of modern web technologies, secure authentication mechanisms, and user-friendly interfaces, the system provides an efficient, accurate, and scalable solution for managing student attendance records.

The project automates routine attendance tasks, reduces human errors, and enhances transparency by allowing teachers, administrators, and students to access real-time attendance information. The integration of analytics and visualization tools enables institutions to monitor attendance trends, generate insights, and make informed decisions aimed at improving student engagement and academic performance.

The modular architecture ensures that each component—such as student management, grade management, attendance management, and dashboard analytics—functions cohesively while remaining independently scalable. The use of Drizzle ORM and MySQL guarantees data consistency, maintainability, and ease of future expansion. Additionally, implementing authentication through Kinde ensures secure access control, preventing unauthorized usage.

Overall, this system demonstrates how modern full-stack development techniques can be leveraged to solve long-standing institutional challenges. It not only improves operational efficiency but also lays a strong foundation for future advancements such as biometric integration, mobile app development, AI-driven insights, and multi-school cloud deployments.

The successful implementation of this project highlights the practicality, robustness, and future-readiness of the solution, making it suitable for real-world deployment in schools, colleges, and coaching institutes

19. REFFRENCES

1. **React.js Documentation** – Official documentation for building component-based user interfaces.
   https://react.dev/
2. **Next.js Documentation** – Official guide for server-side rendering, routing, and application development using Next.js.
   https://nextjs.org/docs
3. **Drizzle ORM Documentation** – Reference for schema design, querying, migrations, and database integration.
   https://orm.drizzle.team/
4. **MySQL Documentation** – Official MySQL reference for relational database management and SQL queries.
   https://dev.mysql.com/doc/
5. **Railway.app Documentation** – Cloud hosting platform used for MySQL deployment.
   https://docs.railway.app/
6. **ShadCN UI Documentation** – Component library guidelines for building accessible UI using Tailwind CSS.
   https://ui.shadcn.com/
7. **AG-Grid Documentation** – Reference for grid components, data tables, sorting, filtering, and event handling.
   https://www.ag-grid.com/react-data-grid/
8. **Recharts Documentation** – Guide for building charts using React components.
   https://recharts.org/en-US/
9. **Kinde Authentication Docs** – Reference for implementing secure user authentication and authorization.
   https://kinde.com/docs/
10. **JavaScript MDN Documentation** – Core reference for JavaScript language features, APIs, and browser behavior.
    https://developer.mozilla.org/
11. **Tailwind CSS Documentation** – Utility-first CSS framework for building responsive UI layouts.
    https://tailwindcss.com/docs
12. **Moment.js Documentation** – Library used for date manipulation and formatting in the project.
    https://momentjs.com/docs/
13. **Node.js Documentation** – Backend runtime documentation used for server-side logic in Next.js API routes.
    https://nodejs.org/en/docs/
14. **GitHub Repositories & Community Articles** – Used as conceptual references for best practices in Next.js and Drizzle ORM integration

20. APPENDIX

The appendix provides supplementary technical details, configuration files, API definitions, and reference material relevant to the *Student Attendance Tracking System*. Although not part of the main report, these details enhance understanding of the system's implementation and architecture.

20.1 API Endpoints Summary

This project uses a well-structured set of REST APIs consumed from the frontend using Axios. Updated endpoints based on your final GlobalApi file are as follows:

1. Grade APIs

Method	Endpoint	Description
GET	`/api/grade`	Fetch all grades

2. Student APIs

Method	Endpoint	Description
GET	`/api/student`	Fetch all students
POST	`/api/student`	Add a new student
DELETE	`/api/student?id={id}`	Delete a student by ID

3. Attendance APIs

Method	Endpoint	Description
GET	`/api/attendance?grade=X&month=MM/YYYY`	Fetch attendance list for a selected grade & month
POST	`/api/attendance`	Mark attendance (present)
DELETE	`/api/attendance?studentId=X&day=Y&date=MM/YYYY`	Mark a student as absent (delete attendance entry)

4. Dashboard Analytics API

Method	Endpoint	Description
GET	`/api/dashboard?date=MM/YYYY&grade=X`	Fetch total present count for analytics (bar chart)

20.2 Frontend API Wrapper (GlobalApi.js)

Your system uses a dedicated Axios wrapper for clean and organized API calls:

```
import axios from "axios";

const GetAllGrades = () => axios.get("/api/grade");
const CreateNewStudent = (data) => axios.post("/api/student", data);
const GetAllStudents = () => axios.get("/api/student");
const DeleteStudentRecord = (id) => axios.delete('/api/student?id=' + id);

const GetAttendanceList = (grade, month) =>
  axios.get('/api/attendance?grade=' + grade + '&month=' + month);

const MarkAttendance = (data) => axios.post('/api/attendance', data);

const MarkAbsent = (StudentId, day, date) =>
  axios.delete('/api/attendance?studentId=' + StudentId + '&day=' + day + '&date=' + date);

const TotalPresentCountByDay = (date, grade) =>
  axios.get('/api/dashboard?date=' + date + "&grade=" + grade);

export default {
  GetAllGrades,
  CreateNewStudent,
  GetAllStudents,
  DeleteStudentRecord,
  GetAttendanceList,
  MarkAttendance,
  MarkAbsent,
  TotalPresentCountByDay
};
```

20.3 Database Schema Summary (Drizzle ORM)

GRADES Table

Column	Type	Constraints
id	int	Primary Key
grade	varchar(10)	Not Null

STUDENTS Table

Column	Type	Constraints
id	int	Primary Key, Auto Increment
name	varchar(20)	Not Null
grade	varchar(10)	Not Null
address	varchar(50)	Nullable
contact	varchar(11)	Nullable

ATTENDANCE Table

Column	Type	Constraints
id	int	Primary Key, Auto Increment
studentId	int	Foreign Key → STUDENTS.id
day	int	Not Null
date	varchar(20)	Not Null
present	boolean (stored as 0/1)	Default: 0

20.4 Environment Configuration (.env)

```
MYSQLHOST=your_host
MYSQLPORT=3306
MYSQLUSER=your_username
MYSQLPASSWORD=your_password
MYSQLDATABASE=railway

KINDE_CLIENT_ID=xxxx
KINDE_CLIENT_SECRET=xxxx
KINDE_DOMAIN=xxxx
KINDE_REDIRECT_URI=http://localhost:3000/api/auth/callback
```

20.5 Tools & Technologies Used

- **Next.js (App Router)**
- **Drizzle ORM**
- **MySQL on Railway**
- **ShadCN UI**
- **Tailwind CSS**
- **AG-Grid**
- **Recharts (Pie + Bar charts)**
- **Kinde Authentication**
- **Axios**

20.6 Additional Notes

- The system ensures proper authentication before accessing /dashboard/* routes.
- Attendance is stored as **1 = Present** and **0 = Absent** for efficient MySQL querying.
- The architecture supports future scalability including notifications, parental portal, and AI-based insights.