



iJRASET

International Journal For Research in
Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** IX **Month of publication:** September 2026

DOI: <https://doi.org/10.22214/ijraset.2026.85002>

www.ijraset.com

Call: ☎ 08813907089

E-mail ID: ijraset@gmail.com



CampusIQ: A Secure Multi-Portal Web Platform for Intelligent School Management

MR Kunal Kanchankar¹, Siddhant Shende², Pratiksha Khobragade³, Kajal Shahu⁴, Dipesh Kanathe⁵, Aditi Bokde⁶,
Shrutik Katkar⁷

¹⁻⁷Department of Computer Science and Engineering, G H Raisoni University, Amravati, Maharashtra, India

Abstract: *CampusIQ is a web-based school management system developed as a single place for handling the academic and day-to-day administrative work of a school. The system was built using React and Vite on the frontend, Express.js for the REST backend, PostgreSQL for data storage, Prisma as the ORM, and JWT for authentication. A user's school context is taken from the authenticated request, which helps prevent records from one school being exposed to another. The completed system has four connected portals for Admin, Teacher, Student, and Parent users. These portals cover academic-year setup, master data, student and staff records, attendance, examinations, fees, timetable, assignments, and notifications. The project also includes three AI-assisted features. Parents can use the Academic Performance Predictor, teachers can use the AI Teacher Co-Pilot to prepare classroom material, and administrators can use the AI-assisted Complaint Management module to organize and review complaints. This paper explains the way these parts were put together, the security approach used in the project, and the features that were completed and deployed.*

Keywords— School ERP, Multi-Tenancy, Role-Based Access Control, React, Node.js, Express.js, PostgreSQL, Prisma, JWT, Learning Analytics, Educational AI, Random Forest.

I. INTRODUCTION

Schools handle a surprisingly large amount of information during a normal working day. Student profiles, staff details, classes, sections, subjects, attendance, examinations, fees, timetables, assignments, notices, and communication all have to be maintained. When some of these activities are kept in spreadsheets, some in paper registers, and others in separate applications, the same information can end up being entered several times. It also becomes difficult for a teacher, administrator, or parent to know whether the information they are seeing is the latest one.

We developed CampusIQ to bring these activities into one connected web application. Rather than building a separate application for every school activity, the project gives each type of user a suitable portal and connects all of those portals to the same backend and database. An administrator can maintain the school's records, a teacher can handle classroom activities, and students and parents can view the information meant for them. Because the data is shared through the backend, a change made in one workflow can also be used by another authorized workflow without creating another copy of the record.

The backend currently follows a modular-monolith approach. The code is divided into logical areas such as students, staff, attendance, examinations, timetable, and master data, but these areas run inside one Express.js application. This was a practical choice for our project because it makes local development, deployment, and debugging easier. It also gives the application clear module boundaries, so a heavily used area could be separated later if the number of users or requests becomes much larger.

CampusIQ is therefore more than a basic application for adding, editing, and viewing records. The information entered during normal school work is also used for dashboards, notifications, reports, and the intelligent features included in the project. We have treated the AI features as support tools rather than as replacements for teachers or administrators. The person using the system can look at the available information and decide what action should be taken.

The project brings together three areas that are usually handled as separate topics in educational technology. School information systems mainly deal with organizing student, staff, and administrative records. Learning analytics uses academic data to understand progress and identify students who may need attention, and models such as Random Forest are commonly discussed for structured data. Multi-tenant software has a different concern: records belonging to different organizations must remain separated even when they use the same application. In CampusIQ, these three ideas meet in the same system. The ERP modules produce the operational data, the analytics features use selected academic records, and the backend applies the school-specific access rules whenever those records are requested.

A. Research Contribution

- A working architecture that connects the main academic and administrative activities through a shared REST API and relational database.



- Tenant-based database checks together with role authorization, helping ensure that users receive only the records belonging to their school and permitted role.
- Four role-specific portals—Admin, Teacher, Student, and Parent—using the same backend and database foundation.
- Three implemented AI-assisted services covering academic performance analysis, teacher content preparation, and complaint analysis.
- A publicly deployed version of the project that allows the completed workflows to be demonstrated and checked outside the local development environment.

II. RESEARCH GAP AND OBJECTIVES

There are already well-established approaches for school information systems, learning analytics, access control, and multi-tenant software. The difficulty for a practical project is making these parts work together without turning them into unrelated features. For example, a prediction model can produce a result, but that result is more useful when a parent or teacher can see it inside the same application used for attendance and academic records. CampusIQ approaches this integration by using the school's own ERP data as the base for the intelligent features and placing those features inside the normal portal workflows.

A. Research Gap

- A number of school management systems are good at storing records and displaying them, but storing a record is not the same as helping a user understand it. Users may still have to compare attendance, marks, assignments, and other information themselves. [2], [8]
- Many academic prediction studies focus mainly on the model, its input data, and the prediction result. In a working school application, another question is important as well: how will a teacher or parent actually see and use that result during everyday work? [3], [4], [5]
- A learning-analytics dashboard can show a useful trend, but the information becomes easier to act on when it is connected to the existing school workflow. A teacher, for example, should be able to move from viewing attendance or marks to deciding what needs attention without switching between unrelated systems. [2], [8]
- Multi-tenant applications need to keep organizations separate, but school software has another level of access control because every role has different responsibilities. A teacher may need a class roster and attendance records, while a student should normally see only their own information. A parent's access is tied to their child rather than to the complete student database. [6], [7]
- AI tools can produce useful educational material and analysis, but a real application also has to answer practical questions about authorization, human review, and access to real data. A feature shown in a demonstration is not enough if it cannot safely work with the records already maintained by the school. [4], [5], [6]

B. Project Objectives

- Build a single platform for the main academic, administrative, and routine school activities.
- Provide secure login and role-based access so that each user can work only with the functions and information intended for that role.
- Apply tenant information in backend queries so that records belonging to one school are not returned when another school's user makes a request.
- Connect students, staff, academic structures, attendance, fees, examinations, timetable, assignments, and notifications through shared APIs.
- Provide working Admin, Teacher, Student, and Parent portals that receive their information from the backend and are not dependent on fixed sample screens.
- Deploy AI-assisted functions for academic performance, teacher support, and complaint management as integrated parts of the same application.

III. SYSTEM ARCHITECTURE AND DESIGN

CampusIQ follows a layered structure so that each part of the application has a clear job. The React/Vite frontend is responsible for the screens and portal navigation, while Axios handles communication with the backend and attaches the JWT to requests. Express.js provides the REST endpoints. When a protected request reaches the server, the authentication middleware checks the token, the authorization logic checks the user's role, and the service layer applies the rules for that operation. Prisma then handles the database operation with PostgreSQL.

The database uses a relational and tenant-aware design. It contains records for users, tenants, students, staff, classes, sections, subjects, attendance, examinations, fees, timetable entries, notifications, and their related data. The relationships between these records are important because the same student, class, subject, or teacher may be used by several modules. Compound uniqueness rules are also used where duplicate attendance or timetable conflicts should be prevented.

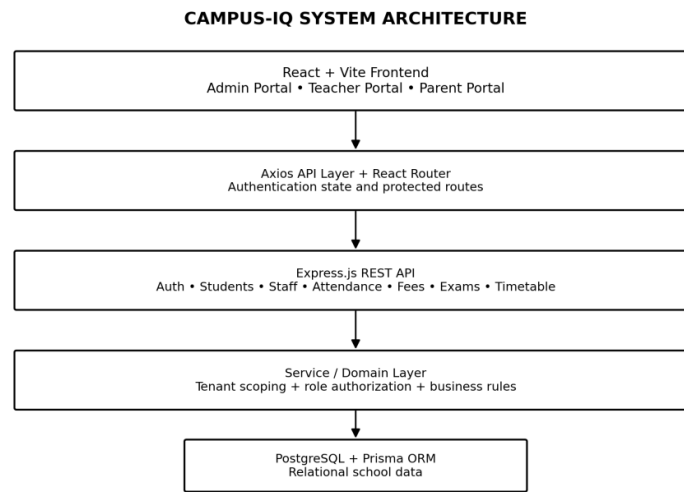


Fig. 1. CampusIQ layered architecture showing the frontend, API, service/domain, and relational data layers.

A. Modular Monolith Design

The backend is divided into logical modules, but all of them currently run inside one Express application. For our project this made the system easier to develop, test, deploy, and debug without losing the separation between areas such as authentication, students, staff, attendance, fees, examinations, timetable, and master data. If the system grows substantially, a module that receives much more traffic can be extracted into a separate service while keeping the existing API responsibilities clear.

B. Request and Data Flow

- The user carries out an authorized action in the portal, for example marking attendance for a student.
- Axios sends the request to the appropriate versioned REST endpoint and attaches the user's JWT.
- The authentication middleware checks the token and establishes the identity and tenant information for the request.
- The backend then checks whether the user's role has permission to perform that operation.
- The controller and service layers validate the request and apply the relevant business rules before the database is accessed.
- Prisma performs the tenant-scoped query against PostgreSQL and passes the result back to the service layer.
- Finally, the response reaches the portal and the relevant screen is updated for the current user.

IV. TECHNOLOGY STACK AND IMPLEMENTATION

Table I gives an overview of the main technologies used in CampusIQ. The choices were made around the actual requirements of the project: several connected portals, one shared API, relational academic records, secure authentication, and the AI-assisted modules.

TABLE I
TECHNOLOGY STACK OF CAMPUSIQ

Layer	Technology	Purpose
Frontend	React 19, Vite, React Router	Component-based interface, navigation, and role-specific portals.
UI / Visualization	Tailwind CSS, Lucide, Recharts, Toastify	Responsive UI, icons, dashboards, charts, and user feedback.
API Client	Axios	Handles API requests and attaches the JWT.
Backend	Node.js, Express.js	REST APIs, routing, controllers, and application services.

Layer	Technology	Purpose
Authentication	JWT, bcrypt/bcryptjs	User authentication and secure password storage.
Validation & Security	Zod, Helmet, Express Rate Limit	Input checks, security headers, and request-rate controls.
Database	PostgreSQL	Relational storage for school and academic records.
ORM	Prisma	Database queries, relationships, and constraints.
Testing / Deployment	Jest, Supertest, Docker/Compose, Render	Testing tools and live cloud deployment.
AI Extension	Random Forest / comparable ML, LLM-based services	Academic performance support, teacher assistance, and complaint analysis.

The complete platform has been deployed and is publicly accessible at <https://campus-iq-kabt.onrender.com/> [9]. The hosted version provides the same Admin, Teacher, Student, and Parent workflows described in this paper, so the implementation can be checked as a deployed application rather than only as a local development project.

V. MULTI-PORTAL DESIGN

The people using a school system do not all need the same information or controls. CampusIQ therefore provides a separate portal for each major role, while keeping authentication, authorization, tenant checks, and the underlying database common to all of them. This keeps each interface focused without requiring four separate backend systems.

TABLE II
MULTI-PORTAL SCOPE OF CAMPUSIQ

Portal	Primary Users	Key Responsibilities	Status
Admin	Administrators	Academic structure, students, staff, attendance, fees, exams, timetable, complaint management, dashboards, and settings.	Implemented
Teacher	Teachers	Classes, students, attendance, assignments, timetable, exams, notifications, and AI Teacher Co-Pilot.	Implemented
Student	Students	Dashboard, attendance, assignments, timetable, exams and results, performance, and complaints.	Implemented
Parent	Parents/guardians	Dashboard, children, calendar, attendance, assignments, timetable, exams, AI Performance Predictor, and complaints.	Implemented

A. Admin Portal

The Admin Portal is the main management area of CampusIQ. Its dashboard shows figures such as the total number of students and teachers, complaints, and monthly fee collection. From the same area, an administrator can reach academic-year settings, master data, departments, classes, sections, students, staff, attendance, examinations, fees, timetable, custom fields, complaints, and system settings. The dashboard is intended to give the administrator a quick picture of the institution before opening a particular module.

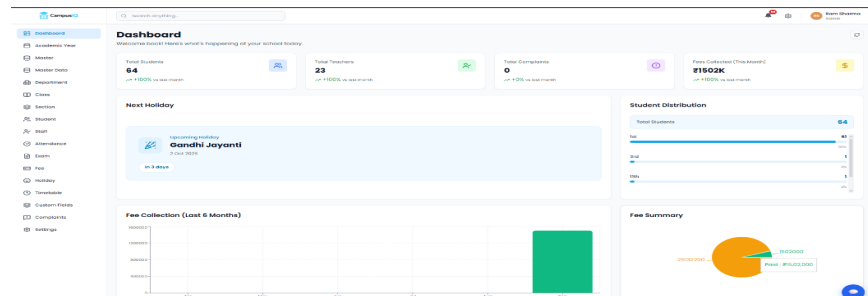


Fig. 2. CampusIQ Admin Dashboard on the live deployment, showing institutional KPIs and navigation to core ERP modules.

B. Teacher Portal

The Teacher Portal is designed around the work a teacher performs during the day. A teacher can open assigned classes, view student lists, mark attendance, manage assignments, check the timetable and examinations, read notifications, and use the AI Teacher Co-Pilot. The dashboard brings together items such as class count, student count, today's attendance, pending assignments, recent attendance activity, and the day's schedule. These screens are connected to the same backend records used by the other authorized portals.

C. Student and Parent Portals

The Student Portal gives a student access to the records connected with their own account. It includes the dashboard, attendance, assignments, timetable, examinations and results, performance information, and a Complaints section. The backend also checks the student identity linked to the logged-in account. This is important because changing an ID in a request should not be enough for one student to read another student's records.

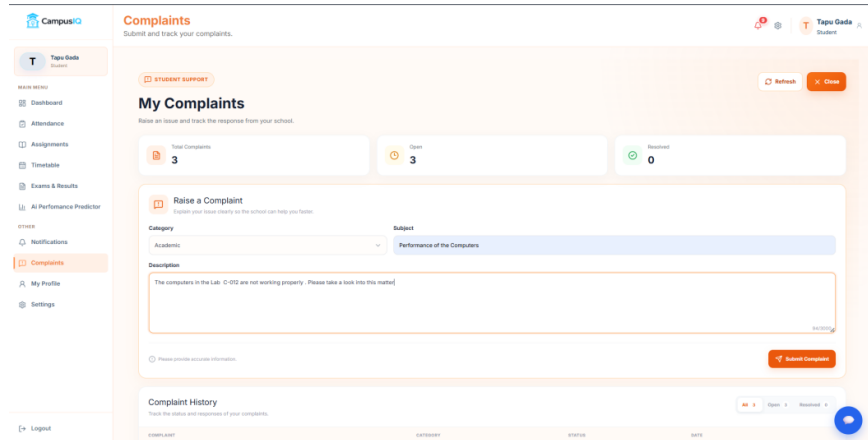


Fig. 3. CampusIQ Student Portal — Complaints module, showing complaint submission and tracking.

The Parent Portal is mainly meant for following a child's school activity and academic progress. Parents can view the dashboard, calendar, attendance, assignments, timetable, examinations, AI Performance Predictor, and complaint reporting for an authorized child. The child relationship is checked by the backend, so the parent view is not a general entry point into the student database.

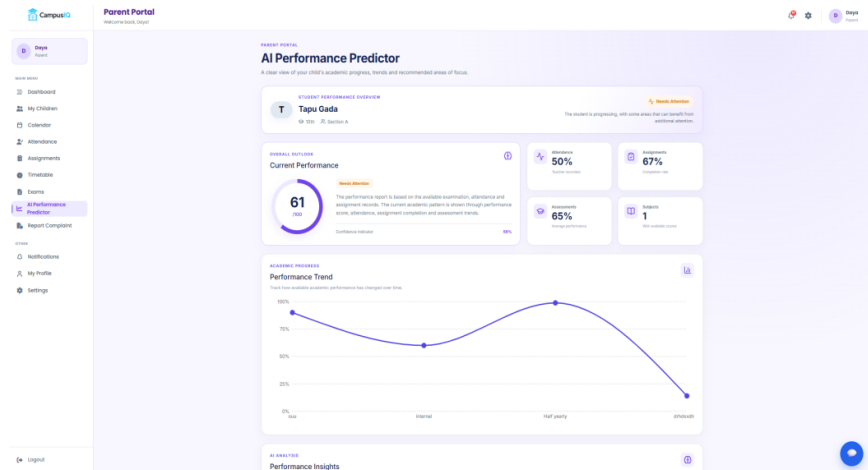


Fig. 4. CampusIQ Parent Portal — AI Performance Predictor, showing an overall score, attendance, assignments, assessments, and a performance trend.

VI. CORE FUNCTIONAL MODULES

The four portals share a set of common functional modules that supply the data shown in different parts of the application. These modules use the same database relationships and access rules. Because of that, information entered in one workflow can be reused by another workflow when the user is authorized to see it.



- 1) Academic Structure: Academic years, departments, classes, sections, subjects, and holidays provide the basic structure for the rest of the system. Attendance, timetable, and examination records depend on these relationships, so incorrect academic setup can create problems in several modules at the same time.
- 2) People Management: The student and staff modules maintain the identity information used by both academic and administrative features. Each record is associated with the appropriate tenant, which helps the backend keep people and their related records within the school to which they belong.
- 3) Attendance: Attendance is recorded through date-based workflows used by teachers and administrative users. A compound uniqueness rule involving the student, date, and tenant prevents the same entry from being stored twice. This is also useful for keeping attendance percentages and trends consistent across the dashboards.
- 4) Examinations and Fees: The examination module stores assessments and marks, while the fee module manages financial records. Examination entries use the same class and subject relationships used by the other academic modules. This makes it possible to use the stored marks for reports and performance views without creating a separate academic reference system.
- 5) Timetable: Timetable entries connect classes, sections, subjects, teachers, periods, and days. Database constraints help catch conflicts such as assigning a teacher or section to two entries in the same period. The database check works alongside the frontend checks, giving the system another way to prevent inconsistent schedules.
- 6) Assignments and Notifications: Teachers can create and manage assignments, while notifications are organized around areas such as attendance, assignments, timetable changes, examinations, and announcements. These features make it possible for information created in one portal to reach the users for whom it is relevant.
- 7) Complaint Management: Students and parents can raise a complaint by choosing a category, entering a subject, and describing the problem. The available categories include academic, fees, transport, infrastructure, technical, teacher, hostel, safety, administration, and other. Administrators can then review and manage the submitted complaints from the complaint dashboard.
- 8) Dashboards and Analytics: The dashboards present cards, charts, and trends selected for each role. Even though the dashboard shows summarized information, its requests still go through the same tenant and role checks as other API requests. The numbers shown therefore come only from records the current user is allowed to access.

VII. SECURITY, AUTHENTICATION, AND MULTI-TENANCY

Security is important in CampusIQ because the system contains personal and academic information. The project does not depend on a single security check. It uses password hashing, JWT authentication, role checks, protected routes, tenant-scoped queries, security headers, input validation, and rate limiting. These controls cover both the login process and the later requests used to read or change data.

A. Authentication

During login, the backend verifies the supplied credentials and creates a JWT for the authenticated user. Axios sends that token with later API requests. Frontend route protection helps stop users from casually opening pages that are not meant for them, but it is not the final security boundary. The backend checks the token and permissions again before protected data is returned.

B. Role-Based Authorization

After the JWT has been verified, the application identifies the user's role, such as administrator, teacher, student, or parent. The role is checked before a protected operation is allowed. In this way, having a valid login does not automatically give a user access to every function in the application.

C. Tenant Isolation

The tenantId used for database access comes from the authenticated user context instead of being trusted from a value sent by the browser. The backend applies that tenant information to the relevant queries. This is the main mechanism used in CampusIQ to keep the records of different schools separate.

D. Data Integrity

Relationships in the Prisma schema and compound unique constraints help keep the stored data consistent. For example, attendance rules prevent duplicate entries for the same student and date, while timetable constraints reduce conflicting assignments. Validation at the API boundary adds another check before records are written to the database.

VIII. IMPLEMENTED AI-ASSISTED FEATURES

The AI features were added on top of the ERP rather than built as a completely separate demonstration. The normal school records remain the source of information, while the AI services use authorized data to prepare a prediction, generate teaching material, or organize complaint information. The result is meant to help the responsible user, who can review the underlying information before acting on it.

A. AI Academic Performance Predictor

AI Academic Performance Predictor is available through the Parent Portal. It uses the academic information available for a student The, including attendance, assignment completion, and assessment results, to produce an overall score out of 100. The screen also shows the individual measures, the number of subjects with available scores, a performance trend, a confidence indicator, and a short explanation. Random Forest and similar ensemble methods are discussed in the related literature as suitable approaches for structured academic data [1], [3], [4]. In CampusIQ, the result is presented as support for the parent rather than as a final judgment about the student. A larger validated production dataset is still needed before a formal accuracy figure can be reported.

B. AI Teacher Co-Pilot

The AI Teacher Co-Pilot is a chat-based tool intended to save teachers some preparation time. A teacher can ask for material for a particular class, subject, or topic, and the assistant can generate question papers, unit tests, MCQs, revision quizzes, lesson plans, simplified explanations, formula sheets, mind maps, classroom activities, and assignment ideas. The teacher can edit the generated material before using it in class. The conversation history also allows an earlier request to be opened again instead of preparing the same prompt from the beginning.

C. AI-Assisted Complaint Management

Students and parents can submit complaints by selecting a category, entering a subject, and describing the issue. Administrators see these records in the Complaint Management dashboard, where the system shows total, pending, in-progress, resolved, and critical-open counts along with resolution-time and resolution-rate information. Complaints can also be viewed by category, while the AI-assisted analysis provides sentiment and priority information. The original complaint remains available to the administrator, so the automated analysis does not replace the record submitted by the user.

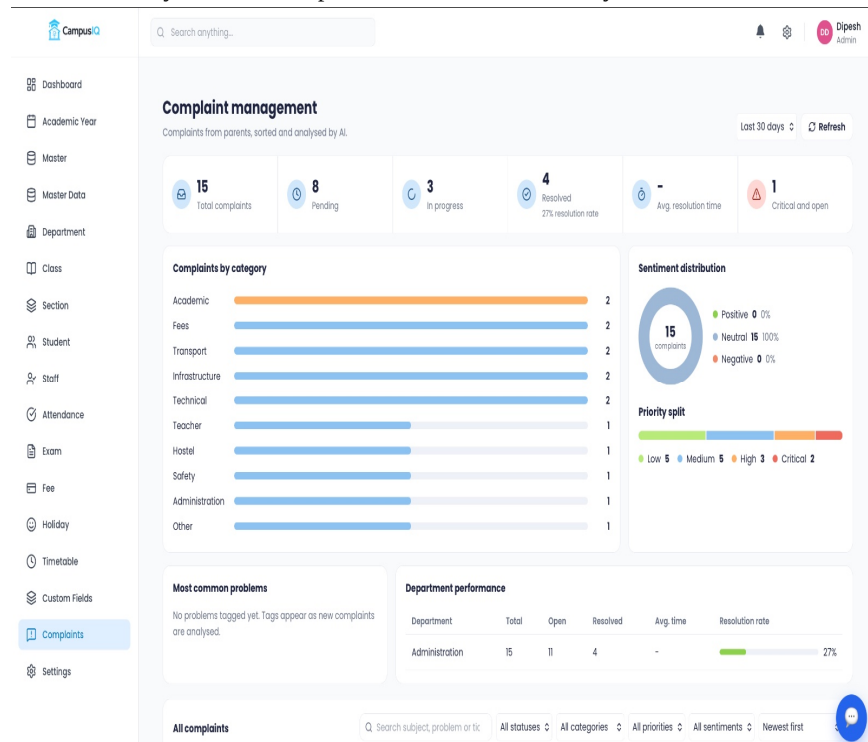


Fig. 5. CampusIQ Admin Portal — AI-assisted Complaint Management dashboard, showing category, sentiment, and priority analysis

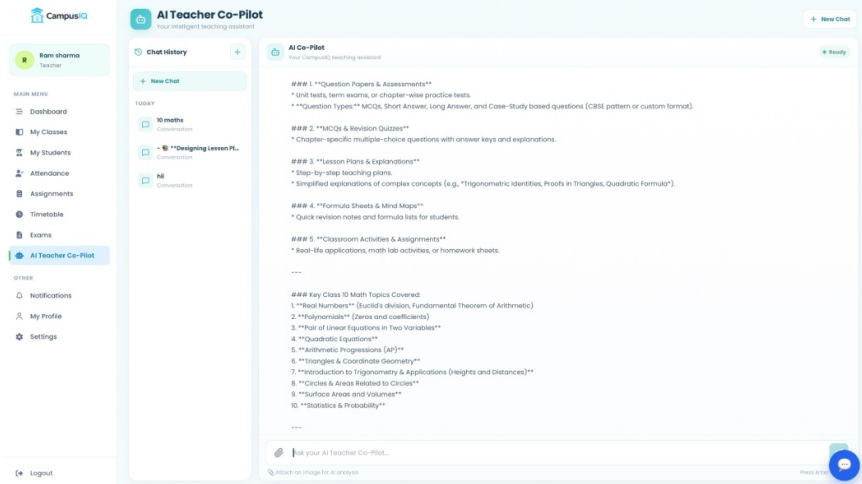


Fig. 6. CampusIQ Teacher Portal — AI Teacher Co-Pilot, generating categorized classroom resources from a chat interface

D. Conversational Access and Human-in-the-Loop Review

A conversational assistant is available in the Admin, Teacher, Student, and Parent portals. At the present stage it provides a base for natural-language interaction and quick access to common information. A later version could connect the conversation directly to authorized ERP queries, but it would have to follow the same tenant and role restrictions already used by the APIs. For the AI features that are currently implemented, a person remains involved in the final step: teachers or administrators can review generated material, predictions, or complaint analysis before deciding what to do.

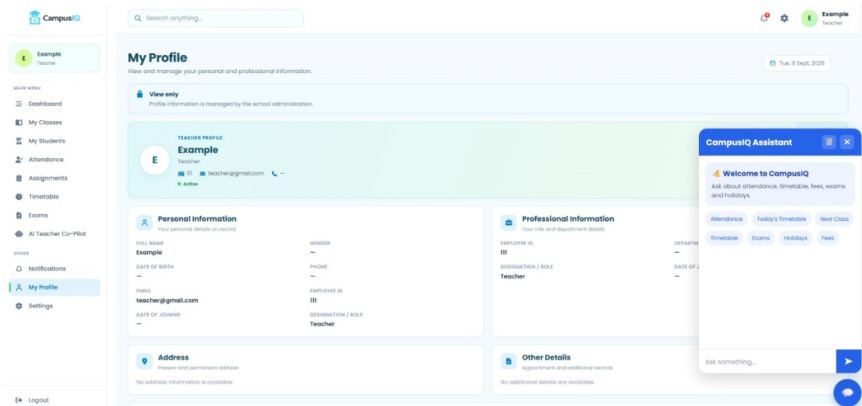


Fig. 7. CampusIQ Assistant — conversational chatbot widget integrated into the portal, offering quick access to attendance, timetable, exams, holidays, and fees.

IX. RESULTS, EVALUATION, AND FUTURE SCOPE

CampusIQ has been completed and deployed. The four portals and the three AI-assisted features described in this paper are implemented and can be checked on the live deployment at <https://campus-iq-kabt.onrender.com/> [9]. The result described here is therefore based on a working hosted system rather than only on screenshots or interface mock-ups. Table III summarizes the main capabilities observed in the deployed project.

TABLE III
OBSERVED PROJECT CAPABILITY

Area	Observed Capability	Evidence
Administration	Dashboard, academic structure, student/staff and operational modules, plus AI complaint analytics.	Admin Dashboard and Complaint Management, Figs. 2, 5.
Teacher workflows	Classes, students, attendance, assignments, timetable, exams, notifications, and AI Co-Pilot.	Teacher Portal and AI Teacher Co-Pilot, Fig. 6.
Student & Parent	Attendance, assignments, timetable, exams,	Student Complaints and Parent AI



Area	Observed Capability	Evidence
workflows	complaints, and AI Performance Predictor.	Predictor, Figs. 3, 4.
Security	JWT authentication, role checks, and tenant-scoped backend queries.	Backend architecture and implementation.
Data layer	Relational entities and database constraints.	PostgreSQL + Prisma schema design.
Deployment	Publicly accessible hosted version of the full platform.	Live deployment [9].

A. Evaluation Plan

- Functional testing of the main CRUD operations and workflows, including attendance, timetable, assignments, notifications, and role-specific features.
- Security testing to check that unauthorized roles, invalid JWTs, and attempts to access another tenant's records are rejected.
- Usability testing based on task completion time, mistakes, and user feedback for representative activities performed by administrators, teachers, and parents.
- Performance testing of API response times under a representative number of simultaneous requests on the deployed application.
- Evaluation of the AI prediction model using measures such as accuracy, precision, recall, F1-score, confusion matrix, and ROC-AUC once a sufficiently large production dataset is available.

The paper does not report a specific AI accuracy value because the project does not yet have a sufficiently large production dataset with formally validated labels for a reliable model evaluation. Giving an accuracy number without that validation could make the results look more certain than they are. For this reason, the present work documents the implemented feature and leaves detailed benchmarking for the next evaluation stage.

B. Discussion

The implementation showed that the main architectural choices made at the start of the project could be reused across all four portals. A shared REST API allowed the portals to work with the same records, while tenant-scoped queries and role checks provided one consistent security approach. The three AI-assisted features are connected to real ERP or complaint data on the deployed system, so they are part of the working application rather than only proposed additions. The main piece still to be evaluated formally is the Academic Performance Predictor, for which a larger labeled dataset is required.

C. Future Scope

A common issue with school-management software is that record keeping and learning analytics are often treated as separate products. CampusIQ was developed with the intention of keeping these parts together. Its AI features use the same attendance, assignment, assessment, and complaint records that support the daily operation of the school. This means the information shown to a parent can change when new academic records are entered, while the complaint dashboard can reflect a newly submitted issue without a separate data-entry process.

The current platform also gives the project a practical starting point for further development. Because tenant information is already built into the backend, the same approach can be extended to a larger multi-school setup while keeping each school's records separated. With more real usage, the project can build an ethically governed dataset for proper evaluation and retraining of the Academic Performance Predictor. The Teacher Co-Pilot can be expanded with regional-language support and more student-specific suggestions. The complaint module could also use patterns from previously resolved cases to support administrators. In the longer term, high-use modules could be moved into independent services and mobile applications could be developed for teachers and parents.

- Build a larger and ethically governed dataset from actual system usage so the Academic Performance Predictor can be benchmarked and, where appropriate, retrained with explainable outputs.
- Extend the conversational assistant into a natural-language ERP chatbot while continuing to enforce role-based authorization when it accesses student, parent, teacher, or administrator data.
- Add regional-language support and more adaptive, student-specific content suggestions to the AI Teacher Co-Pilot.
- Support several schools within one district-level deployment while continuing to enforce tenant isolation.



- Develop a mobile application for teachers and parents and study whether high-use modules should eventually be separated into independent services.

X. CONCLUSION

CampusIQ has been completed and deployed as a web platform for handling academic and administrative school activities through connected role-based portals. React, Express.js, PostgreSQL, and Prisma provide the main full-stack foundation, while JWT authentication, role checks, validation, and tenant-scoped queries protect the application's important access points. The Admin, Teacher, Student, and Parent portals are live, and the Academic Performance Predictor, Teacher Co-Pilot, and AI-assisted Complaint Management features are connected to the deployed system. <https://campus-iq-kabt.onrender.com/>

From a research perspective, the project brings school information management, secure multi-tenant software, learning analytics, and AI-assisted tools into one application. The next useful step is to evaluate the features that have already been built using larger and better-structured production data. It is more important to understand how well the current features work in practice than to keep adding AI features simply to increase the size of the system. Keeping teachers and administrators involved in reviewing the outputs will remain important as CampusIQ is tested and extended.

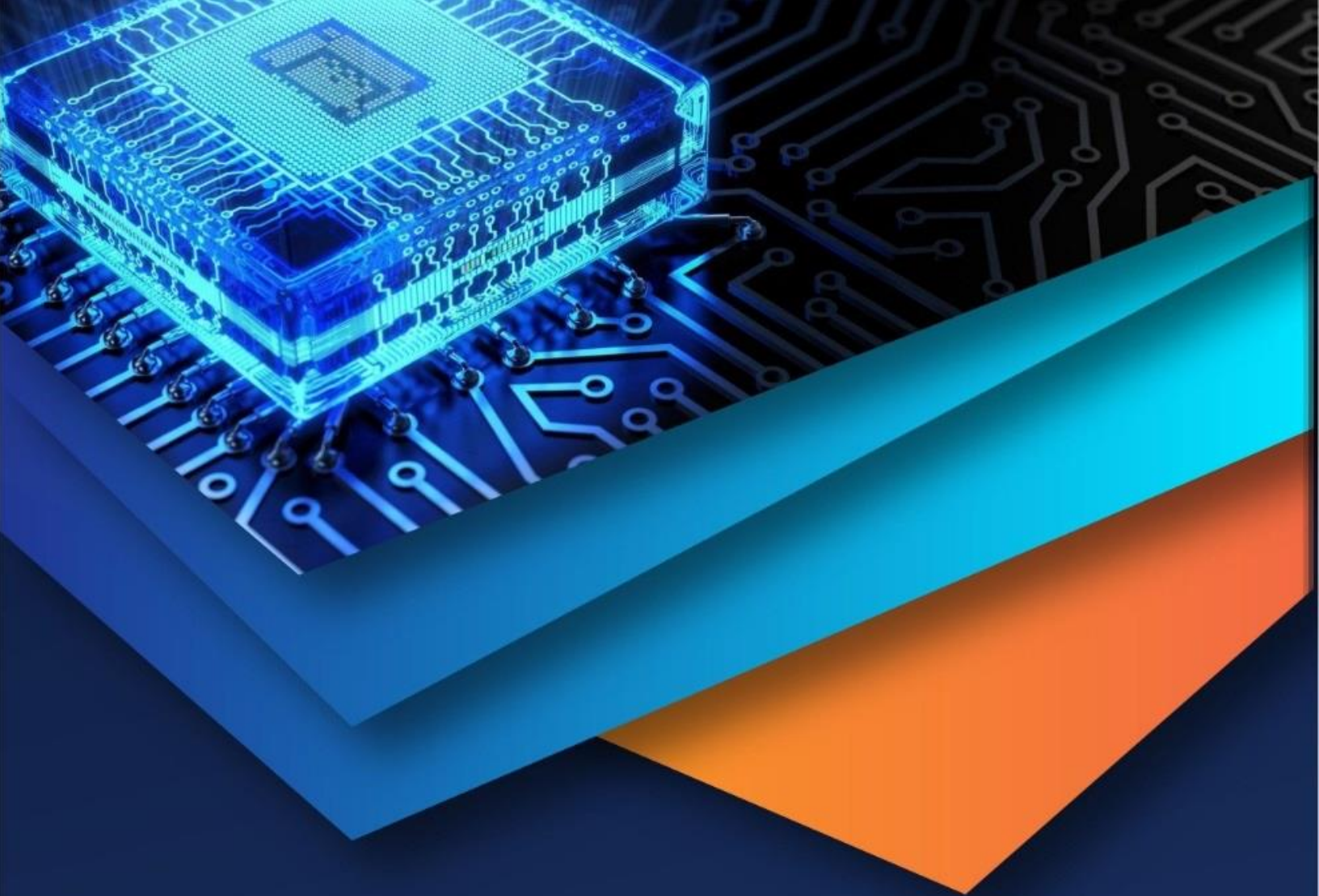
XI. ACKNOWLEDGMENT

The authors would like to thank Mr. Kunal Kanchankar for his guidance, feedback, and technical support during the development and deployment of CampusIQ in the academic session 2026–2027. We also thank the Department of Computer Science and Engineering, G H Raisoni University, Amravati, for the academic support and infrastructure provided during the project.

Pratiksha Khobragade worked on the AI Complaint Management module, AI Teacher Co-Pilot, dynamic dashboards, and the landing and login pages of all the four panels and also handled the deployment work. Siddhant Shende worked on the AI Academic Performance Predictor, Fee and Holiday management, and the Notification and Assignment subsystems, including their REST APIs and database schema. Aditi Bokde worked on Settings, My Profile, Students, Attendance, and Custom Fields. ShrutiKatkar contributed to the Admin Panel user interface. Kajal Shahu worked on Master and Master Data, Timetable, Examinations, and the AI Chatbot. Dipesh Kanathe contributed extensively to the backend development across the Admin, Student, Teacher, and Parent modules, including REST API development, database schema design, server-side logic, authentication and authorization, API integration, and overall system architecture.

REFERENCES

- [1] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, pp. 5–32, 2001, doi: 10.1023/A:1010933404324.
- [2] D. Ifenthaler and J. Y.-K. Yau, "Utilising learning analytics to support study success in higher education: A systematic review," *Educational Technology Research and Development*, vol. 68, pp. 1961–1990, 2020, doi: 10.1007/s11423-020-09788-z.
- [3] N. Tomasevic, N. Gvozdenovic, and S. Vranes, "An overview and comparison of supervised data mining techniques for student exam performance prediction," *Computers & Education*, vol. 143, 103676, 2020, doi: 10.1016/j.compedu.2019.103676.
- [4] Z. Ersozlu, S. Taheri, and I. Koch, "A review of machine learning methods used for educational data," *Education and Information Technologies*, vol. 29, pp. 22125–22145, 2024, doi: 10.1007/s10639-024-12704-0.
- [5] S. Boujmiraz, H. Darhmaoui, and A. D. El Maliani, "Predicting student performance: A comprehensive review of machine learning, deep learning, and explainable AI approaches," *Computers and Education: Artificial Intelligence*, vol. 10, 100548, 2026, doi: 10.1016/j.caeai.2026.100548.
- [6] "Role and Object Domain-Based Access Control Model for Graduate Education Information System," *Procedia Computer Science*, vol. 176, pp. 1241–1250, 2020, doi: 10.1016/j.procs.2020.09.133.
- [7] "ITADP: An inter-tenant attack detection and prevention framework for multi-tenant SaaS," *Journal of Information Security and Applications*, vol. 49, 102395, 2019, doi: 10.1016/j.jisa.2019.102395.
- [8] F. J. García-García, M. I. Gómez-Núñez, and C. Molla-Esparza, "Applications of learning analytics in the study of academic performance in higher education: A meta-review with an educational perspective," *Higher Education*, 2026, doi: 10.1007/s10734-026-01709-y.
- [9] CampusIQ Project Team, "CampusIQ — Live Deployment," Department of Computer Science and Engineering, G H Raisoni University, Amravati, accessed Sept. 2026. [Online]. Available: <https://campus-iq-kabt.onrender.com/>
- [10] CampusIQ Project Team, "CampusIQ: A Multi-Agent AI Platform for Intelligent School Management," Project Progress Seminar, Department of Computer Science and Engineering, G H Raisoni University, Amravati, Sept. 2026.



10.22214/IJRASET



45.98



IMPACT FACTOR:
7.129



IMPACT FACTOR:
7.429



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089  (24*7 Support on Whatsapp)