Project Report

Does our project differ from our original proposal?
Our project remains faithful to the original proposal with no alterations. We've addressed all the requirements outlined in our initial submission.

What has our application achieved? How is it useful?
All backend routes have been completed successfully, along with the deployment of the frontend, MySQL server, and backend on Google Cloud Platforms.

Our project is a full-fledged Classroom Management System (CMS) which we call Illini CMS. The Illini CMS is a highly beneficial project for educational institutions, teachers, students, and parents alike. Centralizing classroom management tasks and communication, it streamlines administrative processes, which can save time and resources for everyone involved.

Teachers can benefit from features designed to optimize their instructional workflows, making it easier for them to manage classrooms, track attendance, and monitor student progress effectively. For students, the CMS facilitates transparent communication with teachers and provides easy access to resources like live Zoom links and assignment submissions. This can enhance the learning experience and promote student engagement.

Most importantly, parents are given a valuable window into their child's academic journey. By providing timely notifications for grade and attendance updates and access to grades and performance data, the CMS empowers parents to stay informed and actively participate in their child's education. They can also track the progress of their child through dynamic visualizations for grade and attendance data. This collaboration between home and school can greatly support student success and overall well-being.

Overall, the Illini CMS addresses the diverse needs of stakeholders within educational environments, fostering a supportive and enriching learning environment through its comprehensive features and functionalities.

Changes in Schema or data source
We've trained our machine learning model using the dataset specified in the proposal, employing user names to populate the user table.

Changes to ER Diagram or table implementations
We adhered to the guidelines outlined in our initial design, and there have been no alterations in our final design.

Functionalities changes:
We have added the following functionality:

- Admins can view the STAR students and below-average students in a ClassroomGroup.
- We have added the appropriate authorization. Only parents can access the parent dashboard. Only admins can create, delete, and modify classroom groups. This was necessary to protect the integrity of the system.

How database programs complement our application
Following are how advanced database programs complemented our application:

1. Data Management: Advanced database programs can handle large volumes of data efficiently, allowing the CMS to store and manage vast amounts of information related to classrooms, students, teachers, assignments, grades, and more.
2. Data Integration: Advanced database programs support seamless integration with other systems and applications, allowing the Illini CMS to exchange data with external systems. Adding integration functionalities is a part of our future work.
3. Performance Optimization: Advanced database programs are optimized for performance, enabling the Illini CMS to deliver fast response times and handle concurrent user interactions effectively. This ensures that users can access and interact with the system quickly, even during peak usage periods.
4. Advanced Querying and Reporting: Advanced database programs support complex querying and reporting capabilities. For example, we were able to make use of a trigger to delete all the grades linked to a student if a certain student was deleted from the system.
5. Scalability and Flexibility: Advanced database programs allow Illini CMS to scale horizontally and vertically. So our system can accommodate growing user bases and evolving requirements over time without sacrificing performance or reliability.

Technical challenges encountered by each member

- **Panshul** - I faced a challenge in connecting the NodeJS server with the MySQL server and addressing how to execute the query. I used this YouTube tutorial for understanding https://www.youtube.com/watch?v=Hej48pi_lOc
- **Pranav -** I faced issues when deploying the machine learning model to a GCP server using Flask. Initially, we used a Gunicorn HTTP server, but that led to several issues loading the model in the Flask app. As a result, I referred to an article online which suggested using pm2 rather than Gunicorn which is better for the Node backend. Article. Apart from this difficulty, I faced issues while fetching backend API endpoints in the frontend views and parsing query parameters effectively.
- **Ketaki -** I faced difficulties while developing the Parent dashboard with student data. This was crucial for our project but posed several challenges, particularly in aggregating and displaying data from various sources in a user-friendly manner. To fix this I used advanced front-end technologies and React libraries like Chart.js that support dynamic data visualization. I learned about the use of these React libraries

through the following YouTube tutorial: https://www.youtube.com/watch?v=RF57yDglDfE&t=717s

- **Jimmy -** I faced a challenge in formatting the front end using CSS in a way that did not cause merge conflicts. In order to fix this two different CSS files were created and conflicting class names for divs were changed.

Future work

Following are the ideas we have for future work to enhance the system's capabilities and utility:

1. Integration with Existing Educational Platforms: Extend the CMS to integrate with existing educational tools and platforms such as Google Classroom, Blackboard, or Moodle. This could include syncing grades, assignments, and class schedules to provide a seamless user experience.
2. Advanced Analytics and Reporting: Implement more sophisticated analytics features that can provide insights into student performance trends, teacher efficiency, and resource utilization. This could include predictive analytics to forecast student outcomes based on historical data.
3. Customizable Dashboards for Users: Create customizable dashboards for different user roles (students, teachers, parents, administrators) that can be tailored to show the most relevant information for each user.
4. AI-driven Tutoring and Assistance: Integrate AI capabilities to provide personalized tutoring and assistance to students based on their learning patterns and performance. AI can also be used to automate routine tasks such as grading and scheduling.
5. Support for Special Education Needs: Adapt the CMS to better support students with special education needs by incorporating features tailored to individual learning plans, accessibility options, and integration with therapeutic tools.
6. Real-time Classroom Interaction Features: Integrate real-time interaction tools such as live polls, quizzes, and collaborative document editing to enhance engagement in a hybrid or remote learning environment.

Division of labor

Our division of labor remained more or less similar to what was initially decided. Details of it are as follows:

- Jimmy was tasked with creating the frontend components, using HTML and CSS to craft an engaging user interface. He also developed APIs that support the addition and updating of student attendance, integrating these functionalities into the user-friendly environment he helped establish.
- Ketaki played a dual role, handling both frontend and backend responsibilities. She worked on API integrations and visualizations to enhance user interaction on the frontend, while also managing database connections and server integrations on the backend, ensuring seamless operation and data flow within the system.
- Pranav focused on applying machine learning techniques to classify teachers' remarks, enhancing the educational feedback system. He also developed several backend APIs

for managing classroom dynamics, including adding or removing students and updating archival classroom content, such as Zoom links for previous classes.

- Panshul was instrumental in setting up backend infrastructure, including routing with Express and developing a robust authentication system for different users. He worked on assignment management functionalities and established a notification system to alert users via email, contributing to an effective communication stream within the CMS. Together with Pranav, he was responsible for deploying the MySQL server on Google Cloud Platforms, ensuring scalability and security.

Teamwork Management

Throughout the project lifecycle, our team demonstrated exceptional collaboration and cohesion, contributing to the successful delivery of the Illini CMS. We established clear communication channels from the outset, ensuring that all team members were kept informed of project updates, timelines, and dependencies. Regular meetings and status updates through WhatsApp helped maintain alignment and transparency across different workstreams. Also, each team member had well-defined roles and responsibilities based on their expertise and skills. This clarity allowed us to work efficiently and autonomously within our respective domains while also facilitating collaboration. Additionally, we adopted a Git branching strategy, with each team member working on individual branches for their assigned tasks. This approach enabled parallel development, minimized conflicts, and facilitated code review and collaboration through pull requests. Overall, we had good teamwork, ensuring we stayed on track towards our goals.