MilestoneMe: *”Bridging the Gap Between Faculty and Students in Coding Journey”*

Project Mentor: Dr. Kapil Aggarwal
Associate Professor and Mentor, Department of Computer Science and Engineering,
Parul Institute of Engineering and Technology, Vadodara, Gujarat, India 391760
Uday Gandhi
Team Lead, MERN Stack Developer
Vraj Rathva
Research and Development, Full Stack Developer
Nevil Modi
Front-end Web Designer and Documentation
Yash Chauhan
Backend Developer and Logic Building

***Abstract*—In recent years, technological advancements have accelerated, with computers and intelligent systems being widely used to solve real-world problems. Effective interaction with machines requires proficiency in programming languages such as C, C++, Java, and Python. Consequently, students are increasingly engaging in coding practice on various online platforms to strengthen their problem-solving and programming skills. While this self-driven approach enhances technical expertise, a critical gap remains between students and faculty: the lack of an efficient monitoring and feedback mechanism.**

Currently, universities focus on teaching fundamental concepts such as Data Structures and Algorithms (DSA), but faculty members often lack visibility into students’ coding activities outside the classroom. Questions such as “How are my students performing on coding platforms?” and “What progress are they making in their daily practice sessions?” remain largely unanswered. This research addresses the missing bridge by proposing a system that enables faculty to monitor, analyze, and evaluate students’ coding performance across platforms. By providing insights into practice patterns, problem-solving consistency, and overall progress, the system aims to foster accountability, enhance learning outcomes, and strengthen the student–faculty academic relationship.

***Index Terms*—Programming Education, Coding Platforms, Student Performance Monitoring, Data Structures and Algorithms (DSA), Learning Analytics, Faculty–Student Engagement, Problem-Solving Skills**

I. Introduction

In the era of rapid technological advancement, programming has emerged as a fundamental skill for students pursuing careers in computer science and engineering. Online coding platforms such as Codeforces, LeetCode, CodeChef, and HackerRank have become integral to skill development by providing access to diverse problem sets, contests, and peer-to-peer learning opportunities. These platforms not only strengthen students’ technical abilities but also cultivate essential problem-solving and logical reasoning skills that are critical in both academic and professional contexts.

Despite the widespread use of such platforms, a significant challenge persists in academic environments: faculty members often lack direct visibility into students’ engagement, performance, and progress outside the classroom. While universities emphasize teaching foundational concepts like Data Structures and Algorithms (DSA), there is no streamlined mechanism to monitor how students apply these concepts in real-world problem-solving scenarios. As a result, valuable insights into practice frequency, problem-solving patterns, and overall skill growth remain inaccessible to educators.

This limitation underscores the need for a system that bridges the gap between independent student practice and faculty evaluation. A robust monitoring tool can enable instructors to observe student activity on competitive programming platforms, identify inactive learners, and provide timely interventions to encourage continuous improvement. Such a system not only supports accountability and personalized guidance but also enhances the overall learning experience by fostering stronger engagement between faculty and students.

The present research introduces *MilestoneMe*, a faculty-focused monitoring platform designed to track and analyze students’ coding journeys. By integrating real-time activity tracking, performance analytics, and automated feedback mechanisms, the system aims to create a structured environment that aligns independent learning with academic oversight, thereby addressing a critical gap in programming education.

II. Literature Review

Several platforms currently exist to support programming education by enabling faculty to assign tasks and monitor student progress. Tools such as *CodeHS*, *CodeGrade*, *JDoodle*, and *Codio* provide automated grading, real-time feedback, and progress dashboards, thereby assisting instructors in managing coding assignments efficiently. Platforms like *Code.org* focus on K–12 education, while research-driven systems such as *Dodona* and *Prutor* offer deeper insights into students' coding behavior through process tracking and intelligent feedback. More recently, tools like *TrackThinkDashboard* have emphasized visualizing learner strategies using detailed activity logs.

While these systems contribute significantly to programming education, they often lack a unified mechanism that bridges the continuous coding practices on popular competitive programming platforms with faculty-level monitoring and evaluation. This gap highlights the need for a specialized system that not only tracks students' engagement and progress but also strengthens the faculty–student academic relationship.

III. Proposed Solution

MilestoneMe is proposed as a comprehensive platform designed to bridge the gap between students' independent coding practices and faculty oversight. The system enables faculty members to create a centralized class dashboard, where students can be manually added and monitored collectively. By integrating with competitive programming platforms such as *Codeforces*, the system fetches real-time data on students' coding activities, including problem-solving attempts, submission frequency, and overall engagement trends.

This functionality allows faculty to observe performance patterns, identify inactive students, and evaluate progress over time with greater accuracy. Additionally, the platform incorporates both automated and manual notification systems, enabling faculty to send reminders or motivational messages to students who display reduced activity. By promoting accountability and encouraging consistency in practice, *MilestoneMe* enhances faculty–student interaction and fosters improved learning outcomes in programming education.

IV. Software Requirements Specification (SRS)

The Software Requirements Specification (SRS) for **MilestoneMe** outlines the functional and non-functional requirements necessary for developing a platform that enables faculty to monitor and evaluate students' programming activities on competitive coding platforms.

Functionally, the system must provide a **Faculty Dashboard** where instructors can manually add students, view their profiles, and monitor class-level progress. The platform should integrate with external coding platforms such as **Codeforces** through APIs, fetching data related to problem attempts, solved problems, submission frequency, coding streaks, and accuracy rates. The system should incorporate an **inactivity detection mechanism** that identifies students who are not engaging in regular practice and allows faculty to send notifications manually or trigger automated reminders. Additionally, the platform should include features for **data visualization**, providing graphical insights into students' progress, trends, and comparative performance.

From a **non-functional perspective**, the system must ensure data security and privacy, particularly in handling students' coding platform credentials and activity data. The solution should be scalable, supporting multiple classes and hundreds of students simultaneously without performance degradation. It should also be platform-independent, accessible via web browsers across different devices, and offer a user-friendly interface that requires minimal training for faculty adoption. The system is expected to deliver real-time or near real-time synchronization with external platforms to ensure that performance data remains current.

Overall, the SRS establishes a framework for **MilestoneMe** as a reliable, secure, and efficient monitoring solution that bridges the gap between student practice and faculty evaluation in programming education.

V. Design

The design of **MilestoneMe** prioritizes a user-friendly and interactive interface, ensuring that faculty members can navigate the platform seamlessly. The system features a lightweight UI with four primary tabs that provide clear insights and actionable tools.

A. Dashboard

The **Dashboard** provides an overview of the class, including the total number of students, active students, and inactive students (defined as seven days of inactivity). Inactive students are automatically highlighted and notified through the mailing system. Two key visualizations are included:

- **Rating Distribution Graph** – Displays the rating ranges at which students are solving problems.
- **Submissions per Day** – Depicts the daily volume of problem submissions.

B. Students Tab

The **Students Tab** presents a tabular view of all students, including their coding platform ID, name, email, phone number, and rating details. Data is synchronized daily to ensure up-to-date insights. Faculty can search for specific students, filter records, or export the data into CSV/Excel format. A detailed view of each student is accessible via the **View** button.

C. Student Profile

The **Student Profile** section provides an in-depth analysis of an individual student's coding journey. Features include:

- Contest history and solved problems
- Problem-solving frequency (bar graph)
- Total problems solved
- Rating of the most difficult problem attempted
- Average problem rating
- Average daily submissions (card format)
- Submission heatmap visualization

D. Settings

The **Settings** section enables faculty to configure synchronization intervals and manage system preferences according to their requirements.

E. Special Feature: Automated Notifications

A unique feature of the platform is its **automated notification system**, which sends motivational emails to inactive students. Faculty members retain control over this feature, with the ability to enable, disable, or send manual notifications as needed.

VI. Methodology

The development of **MilestoneMe** is based on the MERN stack architecture, which combines MongoDB, Express.js, React, and Node.js to provide a scalable, efficient, and modern web application framework. The choice of MERN ensures end-to-end JavaScript development, enabling smooth integration between the frontend, backend, and database layers while maintaining flexibility for real-time data visualization and system scalability.

A. MongoDB (Database Layer)

MongoDB is employed as the core database to store faculty records, student information, coding activities, contest histories, and notification logs. Its schema-less, document-oriented design allows the system to handle diverse and dynamic datasets such as problem submissions, ratings, and activity streaks. Daily synchronization ensures that student progress data remains up to date.

B. Express.js (Backend Framework)

Express.js functions as the middleware that manages the interaction between the frontend and the database. It exposes RESTful APIs to handle faculty requests, authenticate users, fetch data from Codeforces APIs, and manage notification triggers. The inactivity detection logic is implemented in this layer, which continuously analyzes student engagement and communicates results to the faculty dashboard.

C. React (Frontend Framework)

React is used to design a responsive and interactive user interface for faculty members. The platform consists of modular components such as the Dashboard, Students Tab, Student Profile, and Settings, each designed for ease of navigation. React's component-based architecture and state management enable dynamic rendering of data visualizations such as rating distribution graphs, submission frequency charts, and submission heatmaps. It also powers features like search, filtering, and exporting student data in CSV/Excel format.

D. Node.js (Server Environment)

Node.js provides the runtime environment for executing backend services. Its non-blocking, event-driven architecture supports asynchronous tasks such as fetching coding activity from Codeforces APIs, updating MongoDB collections, and handling bulk student requests. Node.js also schedules background jobs like daily synchronization of student data and automated notifications, ensuring system reliability and scalability.

E. System Workflow

Faculty members begin by manually adding students to the platform, including their coding platform IDs. The backend, built on Express.js and Node.js, periodically fetches activity data from Codeforces APIs and updates MongoDB. React then retrieves this data through API calls and presents it in the form of dashboards, graphs, and detailed student profiles. Inactivity detection algorithms identify students with prolonged inactivity (seven days or more), after which the system triggers automated or manual notifications via the mailing service.

F. Advantages of MERN Stack

The MERN architecture provides several advantages for MilestoneMe:

- **Unified Development Language** – JavaScript is used across all layers, reducing complexity and development overhead.
- **Scalability and Performance** – MongoDB and Node.js ensure the system can scale to handle multiple classes and large student datasets.
- **Interactive Visualization** – React enables real-time, modular, and highly responsive UI components for effective data presentation.
- **Flexibility and Extensibility** – The architecture supports integration with additional competitive programming platforms in the future.

By leveraging the MERN stack, **MilestoneMe** achieves a balance between real-time data processing, efficient monitoring, and user-friendly interaction, making it a practical and scalable solution for bridging the gap between students' coding practices and faculty supervision.

VII. Implementation

The implementation of the project was carried out using the MERN stack architecture, which combines MongoDB, Express.js, React.js, and Node.js to provide a robust, scalable, and modular solution. Each component of the stack was integrated systematically to achieve seamless communication between the frontend, backend, and database.

A. Frontend Implementation

On the frontend, **React.js** was employed to design an interactive and responsive user interface. The use of *Recharts* allowed the visualization of coding progress and performance data in the form of dynamic, graphical charts. These visualizations made the platform more intuitive and provided faculty with clear insights into learning trends and milestones.

B. Backend Implementation

The backend was developed with **Node.js** and **Express.js**, which handled routing, API endpoints, and middleware operations. The backend also integrated external APIs using *node-fetch*, enabling real-time retrieval and synchronization of student performance data from coding platforms. Additionally, *node-cron* was implemented to automate recurring tasks such as periodic data updates and notifications, ensuring that users consistently receive up-to-date information without manual intervention.

C. Database Layer

The database layer, powered by **MongoDB**, stored student profiles, coding activity records, and faculty dashboards in a structured and query-efficient manner. MongoDB's flexibility in handling semi-structured data proved effective in managing diverse inputs from multiple coding platforms.

D. Extended Functionality

For extended functionality, additional libraries were integrated:

- **Nodemailer** – Facilitated the automatic sending of notifications, reminders, and alerts to both faculty and students, thereby enhancing communication and engagement.
- **json2csv** – Enabled exporting of performance records into CSV format, making it easier for faculty to analyze or archive data externally.

E. Summary

Overall, the implementation demonstrates the synergy of MERN stack technologies combined with additional libraries to create a feature-rich, efficient, and user-friendly system. The modularity of the stack ensures scalability for future enhancements such as AI-driven recommendations, integration with multiple coding platforms, and advanced analytics features.

VIII. Conclusion

The project successfully demonstrates how the MERN stack, when combined with supporting libraries such as *node-fetch*, *Recharts*, *node-cron*, *nodemailer*, and *json2csv*, can be leveraged to build an efficient, user-friendly, and scalable platform for monitoring and enhancing student performance. By integrating data visualization, automated notifications, real-time data fetching, and export functionalities, the system not only streamlines faculty oversight but also empowers students to track their learning journey more effectively. The modularity of the design ensures adaptability for future upgrades, including advanced analytics and AI-driven recommendations, making this solution a strong foundation for progressive academic monitoring and personalized learning support.

IX. Future Scope

A. Integration of Multiple Platforms

Extend support beyond Codeforces to platforms like **LeetCode, CodeChef, HackerRank, GeeksforGeeks, and AtCoder** for holistic progress tracking.

B. Student-Side Dashboard

Provide a dedicated interface for students to monitor their own coding journey, analyze performance, and identify weak areas.

C. Task Panel for Faculty

Enable faculty to assign coding tasks or problem sets to students, helping irregular learners re-engage and maintain consistency.

D. Personalized Mails & Recommendations

Automate customized motivational mails and problem suggestions based on a student's past activity and performance trends.

E. Improved User Experience

Enhance the system's design to be more user-friendly, interactive, and engaging, ensuring higher student participation.

F. Gamification Features

Introduce leaderboards, badges, and streaks to make the learning journey competitive and motivating.

G. Machine Learning Integration

Use ML models to predict student performance, suggest targeted problems, and forecast coding progress.

H. Peer-to-Peer and Mentor Support

Build features for student collaboration, mentor-mentee guidance, and healthy peer comparisons.

I. Advanced Analytics

Provide faculty with detailed insights like trend analysis, performance forecasting, and inactivity predictions.

References

[1] ProgressIQ, "Student Academic Tracking System," ProgressIQ, [Online]. Available: Know more.

[2] A. Bogomolov, A. Kouznetsov, A. Chistikov, "CodeBuddy: A Programming Assignment Management System," Open Research Software, 2020. [Online]. Available: Know more.

[3] Asma Zaidi, "Student Academic and Performance Tracking System," GitHub Project, [Online]. Available: Know more.

[4] Anwesha Chakraborty, "Student Performance Tracker Design," GitHub Project, [Online]. Available: Know more.

[5] CodeCombat, "Teacher Dashboard," [Online]. Available: Know more.

[6] CodeGrade, "The Engaging Code Learning Platform," [Online]. Available: Know more.

[7] Codio, "Hands-On Learning Experience Platform," [Online]. Available: Know more.

[8] Codesters, "Coding in Your Classroom," [Online]. Available: Know more.

[9] Kitaboo, "Student Performance Tracking for Individualized Instruction," [Online]. Available: Know more.

[10] BMSCE IEEE, "A Tool to Track Student Performance on Various Online Platforms," [Online]. Available: Know more.

[11] M. O'Connor, "Student's Programming Activity Tracking System to Help Instructors of the Programming Exercise," ResearchGate, [Online]. Available: Know more.

[12] K. Tanaka, "Watcher: Cloud-Based Coding Activity Tracker for Fair and Convenient Online Programming Hands-On Education," MDPI Sensors, 2022. [Online]. Available: Know more.

[13] S. Sharma, "Student Performance Tracking System," International Journal of Research and Production, 2021. [Online]. Available: Know more.

[14] P. Gupta, "An Artificial Intelligence Approach to Monitor Student Performance and Devise Preventive Measures," ResearchGate, 2021. [Online]. Available: Know more.

[15] R. Verma, "Learning Analytics for Tracking Student Progress in LMS," ResearchGate, 2022. [Online]. Available: Know more.