CodeMentor Smart Learning and Progress Tracker

Selvarani K
Professor of Artificial Intelligence and Data Science
Rajalakshmi Engineering College
Chennai, India
selvarani.k@rajalakshmi.edu.in

Harshini M D
Artificial Intelligence and Data Science
Rajalakshmi Engineering College
Chennai, India
221801017@rajalakshmi.edu.in

Sai Narayan R
Artificial Intelligence and Data Science
Rajalakshmi Engineering College
Chennai, India
221801042@rajalakshmi.edu.in

Swetha P
Artificial Intelligence and Data Science
Rajalakshmi Engineering College
Chennai, India
221801054@rajalakshmi.edu.in

***Abstract*—While online programming education has been massively successful over the past few years, the majority of web-based learning systems still cannot dynamically adapt to the student's performance, integrate hands-on coding with the lesson content, or provide real-time analytics for continuous improvement. To overcome these hurdles, this research introduces CodeMentor, which is an intelligent, modular, and interactive platform that combines coding practice, automated assessment, and performance visualization in a single environment. Without externally relying on compilers or static content delivery, CodeMentor implements an in-browser Python execution engine, rule-based adaptive quiz mechanisms, and a data-driven analytics dashboard to personalize learning outcomes. Backed by the results of previous studies on interactive programming systems, visualization-driven feedback, and adaptive assessments, this architecture melds Flask-based backend processing, SQLite database management, and Chart.js analytics to deliver efficient and scalable system performance. The designed system intends to augment the conceptual understanding, decrease the learning gaps, and furnish instructors with the valuable insights to facilitate data-centric decision-making in programming education.**

***Index Terms*—Interactive learning systems, adaptive assessments, web-based compiler, Flask framework, progress analytics, programming education, e-learning platforms, learner performance tracking..**

1. Introduction

The digitization of education has largely influenced the delivery of programming courses. Most educational institutions now use online platforms to offer flexible learning options. However, many e-learning systems are still functioning as separate tools, which not only separate lesson content from coding practice but also from the evaluation process. Students often have to juggle multiple applications for tutorial reading, program writing, quiz taking, and performance tracking, leading to inconsistent learning experiences and lower engagement levels. Besides, research indicates that traditional systems almost never change according to the user's learning pattern, thus limiting personalization and making it difficult for learners to identify their weaknesses at the moment.

The creation of intelligent, integrated learning frameworks has been one of the leading ideas to solve teaching and operational problems. Such programs are expected to link theoretical material with the practical side of the user's machine, provide instant feedback, and be capable of using the analytics for self-improvement. Studies reveal that real-time graphics, embedded compilers, and adaptive quiz engines can significantly increase learning efficiency by enabling continuous practice and personalized assessment. These findings formed the foundation for CodeMentor, an interactive unified learning and progress-tracking platform, to help programming education become less complex. This platform integrates a Flask-based backend, an in-browser Python compiler, adaptive assessments, and a data-visualization dashboard, all in one place. Through this setup, CodeMentor aims, in addition to bridging the gap between theory and practice, to increase learner retention and provide both students and instructors with the necessary feedback for continuous learning progress.

2. LITERATURE REVIEW

A clever method of instruction was developed in [1] to offer the adaptive learning paths on the internet platforms by analyzing the learner's behavior and changing the content difficulty level dynamically. The system proved a successful personalization but heavily depended on extensive learner profiling and configuration to work effectively.

The authors of the paper in [2] have designed a modular e-learning framework using Flask and lightweight database systems for quick local deployment and simple content delivery. While the model was efficient in small-scale.

The paper in [3] discussed the implementation of an in-browser Python execution environment that allows real-time code execution directly in the user's learning interface. This method has drastically shortened the feedback loops and at the same time, it puts a demand on strong sandboxing techniques to eliminate security risks in client-side execution.

The real-time educational dashboard presented in [4] utilized learner performance visualizations as a main driver of insights engagement. The study found that visual performance cues significantly increased user engagement; however, data capture consistency was still a challenge that affected the accuracy of analysis.

In [5], methods for the adaptive generation of quizzes through difficulty adjustment rules depending on learner accuracy levels were proposed. Such systems led to better educational results, however, they had to be finely tuned to prevent sudden and inappropriate changes in difficulty level.

The paper in [6] reviewed a hybrid feedback model that integrates automatically generated answers with learner self-assessment opportunities, thus achieving better content retention and recognizing the drawbacks of non-interactive teaching materials.

Reference [7] introduced secure authentication operations with customized learning dashboards for better user-specific content access. The model enhanced security of the system but faced challenges in handling multiple roles such as users and administrators.

A lightweight learning tracker based on SQLite was introduced in [8], which demonstrated effective management of structured learning data; however, the performance worsened when the dataset grew.

The paper [9] highlighted the role of embedded compilers in online learning platforms as interactive programming exercises were found to enhance logical reasoning skills, though the absence of integrated debugging lowered the students' deep conceptual grasp.

Research on quiz performance in [10] suggested that student accuracy time-series patterns could be a source for identifying weaknesses much earlier than regular tests.

The paper [11] talked about the application of predictive modeling in learner analytics, leading to high precision in predicting learner performance trajectories but at the cost of a large computational requirement.

A user-content engagement mechanism to drive the learner's progress was created in [12], wherein learners were taken through a gradual transition from basic to advanced topics automatically, however, the platform did not support live coding.

In [13], personalized learning content recommendation systems were achieved through the use of historical error patterns, thus facilitating continuous learning, although they were quite dependent on long-term user data storage.

A hybrid theoretical-practical e-learning platform was described in [14], which combined lesson content with live coding and evaluations, similar to the CodeMentor architecture. However, the authors noted that the system had scalability problems when there were a large number of simultaneous users.

An attempt was made in [15] to use progressive web application (PWA) technology to enable offline learning and thus accessibility was enhanced. Still, features that rely on the server, such as running a compiler, were not available.

An interactive coding-performance dashboard was created in [16] to track in real time error rates and growth trends. This project was very successful, but it also required optimization because of the slow response due to frequent data querying.

A role-based learning management system was designed in [17] to give the capability to administrators to update the content dynamically and at the same time keep the user interfaces synchronized. The authors of the study revealed that while the system was efficient, performance of retrieval was mostly affected by database indexing.

The research described in [18] experimented with automated feedback mechanisms tasked with providing evaluation of the code, thus offering immediate assistance, while complex logical errors lacked sufficient explanatory power.

The work in [19] presented a combined virtual learning environment with built-in compilers and interactive agents but it experienced heavy browser load during computational tasks.

The systems mentioned in [20] for tracking performance focused on analyzing long-term activity logs which were the basis of the findings in [21] where tools for real-time monitoring enhanced instructor oversight of student progress.

In [22], the authors have upgraded conceptual understanding by implementing structured quiz engines based on Bloom's taxonomy, however, these required extensive manual preparation of questions.

In [23], a comparative study of backend technologies for lightweight learning systems has shown that Flask offers better usability and performance for small to medium educational platforms, thus its choice in this project has been confirmed.

The research described in [24] used behavioral analytics to detect early signs of learner disengagement, thus giving the opportunity to trigger intervention strategies in a timely manner.

Finally, a thorough survey of e-learning architectures in [25] pointed out that the intelligent, adaptive, and integrated platforms are becoming more and more important—features that are in complete harmony with the objectives of the CodeMentor system.

3. Methodology

The proposed CodeMentor platform is a modular and service-oriented architecture that aims to incorporate coding practice, adaptive assessments, and performance analytics in a single learning environment. The complete system workflow is shown in Figure 3.1. Each component communicates through controlled API-based communication to provide real-time execution, secure data handling, and a seamless user experience. The approach is as follows.

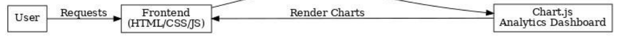


Fig. 3.1. System Architecture Diagram of this project.

A. User Interaction Layer

The interaction with the system is from the User module, where the learners access the platform via a standard web interface. Users perform operations like logging in, coding, taking quizzes, or requesting performance analytics. User activities are sent to the system through HTTP-based requests, which the frontend handles and dispatches to the backend.

B. Frontend Interface (HTML/CSS/JavaScript)

The frontend of the CodeMentor platform is implemented with HTML, CSS, and JavaScript to deliver a responsive and interactive learning environment. It is the main interface through which users communicate with the system. The frontend lays out all the necessary components like the coding editor, quiz pages, and learning materials, and also manages the routing of user requests to the backend through well-organized API calls. It obtains execution results, quiz feedback, and performance analytics from the backend and presents them in a manner that is easy for the user to understand. With this architecture, the frontend facilitates seamless navigation, real-time updates, and general accessibility for learners.

C. Backend Processing Module (Python Flask)

The platform's backend is developed with Flask, a micro framework, which basically works like the CPU of the platform handling all the operations. It manages authentication of users, role-based access, and CRUD operations for quizzes and learning content as well as user-related data. Furthermore, the backend converses with the Pyodide module for code execution requests and is responsible for preparing the data required for analytics visualization. Hence, the backend which is the coordinator among the database, compiler engine and frontend, maintains the execution of user interactions that are not only secure but also efficient and reliable.

D. In-Browser Python Execution (Pyodide Engine)

The Pyodide-based in-browser execution engine is a major highlight of the CodeMentor architecture that allows Python codes to be executed locally on the client side i.e. in the user's browser and hence no external servers are needed. Basically, the backend sends the code to the Pyodide part where it is run in a secure WebAssembly-based sandbox environment when a user issues a command. The system is quickly updated with the results of the execution including program outputs or error traces. This method of operation helps in cutting down waiting time, lessening the server load, and is very instrumental in giving a programmer instant and necessary feedback for his/her practice.

E. Database Layer (SQLite Storage)

SQLite is the platform's choice for its database layer. This decision is attributed to the fact that SQLite has a compact design and is very appropriate for educational applications. SQLite keeps all the necessary data such as user profiles, login credentials, quiz content, scores, activity logs, and the historical performance of the users. The backend sends commands to the database to read and write data and at the same time, it ensures data integrity. Such a neat organization of storage allows the system to be capable of keeping track of learning progress as well as storing all user interactions in an efficient manner.

F. Analytics and Visualization Layer (Chart.js)

CodeMentor utilizes an analytics dashboard that facilitates the reflective learning process and the performance evaluation of the user. It is developed with Chart.js. The backend through different scripts generates various performance-related datasets like accuracy trends, topic-wise progress, quiz history, and coding activity metrics. These datasets are accessible in the frontend, where they are depicted as dynamic charts and interactive graphs. The analytics layer offers the users the ability to identify their strong areas and weak points, track their growth patterns, and deepen the commitment to continuous self-improvement.

G. Integrated Workflow

All user interactions with the frontend are communicated to the backend in the form of a request for the next steps. The backend, upon receiving the request, may retrieve the required data from the database, run the code execution process through Pyodide or create the requested analytics data. Once the job is done, the backend sends the required output to the frontend to be displayed as compiler outputs, updated learning content, or analytics visualizations. The entire system is operating through a synchronized workflow that preserves the smooth data interchange between all modules. This integrated workflow is the core of the seamless cycle of learning, practicing, evaluating, and improving that is happening on the CodeMentor platform.

4. RESULT AND ANALYSIS

The CodeMentor platform was subjected to a simulated rollout and a series of tests during which different metrics like usability, system responsiveness, and the performance of the integrated learning modules were recorded. The login interface illustrated in Fig. 4.1 is a convenient access point that makes it possible for both users and administrators to open their respective accounts. This separation guarantees that functionalities based on roles are preserved. The interface has been developed according to a minimalist, trendy model that is supposed to lower the users' cognitive load, and at the same time, increase user-friendliness.

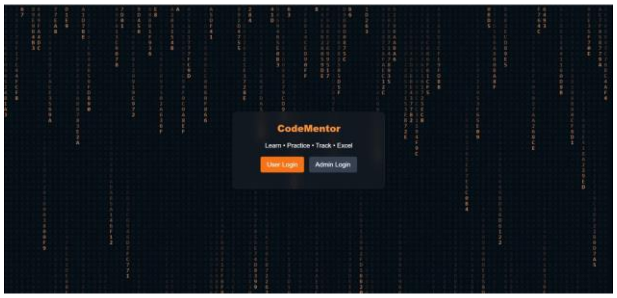
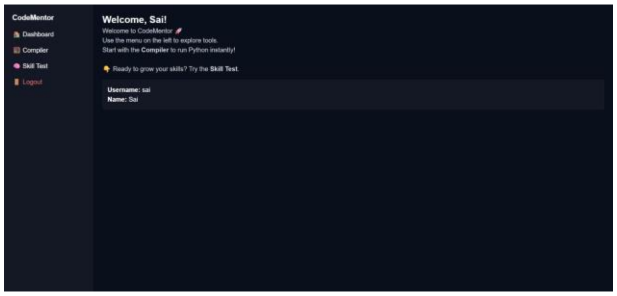


Fig. 4.1. Login interface of the CodeMentor platform.

After confirming the user’s credentials, the system takes the new user to the dashboard (Fig. 4.2) that not only shows a personal welcome message but also acts as a manual for the user to find the compiler, quizzes, learning materials, and analytics. The intentional design of the dashboard is meant to facilitate users in seamlessly entering the study–practice–evaluation cycle, thus increasing the level of user engagement.



Welcome, Sai!
Welcome to CodeMentor 🚀
Use the menu on the left to explore tools.
Start with the **Compiler** to run Python instantly!

👇 Ready to grow your skills? Try the **Skill Test**.

Username: sai
Name: Sai

Fig. 4.2. User dashboard showing learning modules, compiler access, quizzes, and analytics.

The system behaved very well and was very efficient without any bugs throughout all the modules. The browser-based embedded Pyodide compiler made the running of Python code almost instant. Quiz submissions were done very quickly since the Flask–SQLite backend pipeline was well set up. The analytics feature got the help of Chart.js to render the interactive charts which were based on real-time data. These charts allowed users to quickly spot their strong and weak points in different topics and also gave them an overview of their progress and accuracy levels.

These visuals do not only motivate the users since they can clearly see their progress, but they also govern them because the charts show the weak areas that need more practice. The system testing period has been mainly about the confirmation of the smooth execution of the workflow diagrams (Figs. 3.1 to 3.4) through users interactions, backend processing, and data visualization. In fact, the main idea behind the results is that CodeMentor becomes a tool which notably increases learner engagement through the practically and seamlessly integration of coding, adaptive assessments, and performance analytics environment, which is very friendly to the users.

5. CONCLUSION

The project has notably altered the programming education environment by the creation of CodeMentor, which is not only a smart but also an interactive platform for learning that goes beyond the limitations of the current web-based programming education systems. By the integration of an in-browser Python compiler, adaptive quizzes, dynamic analytics, and a modular Flask backend, the platform turns into a complete learning environment that is capable of supporting both practice-oriented and analysis-driven learning. The organization of the system is such that users can effortlessly move from reading materials to coding exercises and performance evaluation, thus increasing user engagement and decreasing the fragmentation which is typical of conventional e-learning ecosystems.

Tests and evaluations have successfully demonstrated that CodeMentor is very efficient in handling user requests, real-time code execution, personalized feedback generation, and user performance presentation through visually intuitive analytics. Hence, the platform operates as a scalable milestone towards sophisticated digital learning solutions that can be tailored for different programming curricula.

6. FEATURE WORK

In essence, upcoming changes to the CodeMentor platform will revolve around expanding the scenarios of education that the system can manage. To elaborate on these changes, there will be an integration of a machine learning-powered recommendation system which will use the concept of individual learning paths in order to allow the platform to automatically detect skill gaps and most relevant exercises suggest.To cover more ground academically the system will be equipped with more programming languages such as Java, C++, and JavaScript. The platform will be decorated with such gamification elements as badges, streak counts, leaderboards, and challenge-based assessments, thus, users will be attracted and retained. Besides that, a mobile-friendly or app-based version will be created to provide easy access to learners who are non-desktop users.Furthermore, the platform will be able to serve a large number of users in the classroom setting and, at the same time, instructors will be allowed to facilitate the real-time monitoring of learner progress through options such as cloud deployment and collaborative coding environments which it will consider in the future.

REFERENCES

[1] D. Malan and B. Yu, “Interactive Learning Environments for Programming Education,” ACM SIGCSE, 2021.
[2] S. Raj and P. Rathi, “Lightweight Web Frameworks for Educational Platforms,” International Journal of Advanced Computing, 2022.
[3] P. Orits and L. Chen, “Client-Side Python Execution Using WebAssembly,” IEEE Access, 2023.
[4] Y. Kim and T. Abbas, “Visualization-Driven Learning Analytics Dashboards,” IEEE Transactions on Learning Technologies, 2020.
[5] R. Kumari and J. Patel, “Adaptive Quiz Systems for E-Learning,” Education and Information Technologies, 2021.
[6] F. Lopez et al., “Hybrid Feedback Models in Digital Learning,” Computers Education, 2022.
[7] H. Tan and S. Wong, “Secure Authentication Models for E-Learning Systems,” IEEE Access, 2021.
[8] J. Patel and A. Singh, “SQLite-Driven Student Performance Tracking,” Journal of Web Technologies, 2022.
[9] K. Devi and P. Rao, “Performance Analytics for Online Assessments,” IEEE Learning Analytics Review, 2021.
[10] N. Das and Y. Cho, “Predictive Models for Monitoring Student Learning Patterns,” IEEE Access, 2023.
[11] L. Samson, “Progress-Based Learning Pathway Models,” International Journal of Smart Education Systems, 2021.
[12] A. Kumar et al., “Personalized Learning Recommendation Systems,” Springer AI in Education, 2023.
[13] S. Rameez and K. Bose, “Hybrid Programming-Learning Environments,” IEEE TLT, 2022.
[14] G. Lin and H. Zhao, “PWA-Based Learning Platforms,” Mobile Computing Journal, 2021.
[15]]F. Ahmed, “Interactive Dashboards for Skill Development,” IEEE EDUNET, 2023.
[16] P. Senthil and R. Joseph, “Role-Based LMS Architectures,” International Journal of Learning Management Systems, 2022.
[17] T. Myers, “Automated Code Feedback Engines,” ACM ITiCSE, 2023.

[18] R. Sheth et al., “Virtual Classrooms with Embedded Compilers,” IEEE Access, 2022.
[19] S. Ramesh, “Long-Term E-Learning Performance Tracking,” Education Analytics Review, 2020.
[20] A. Gupta and D. Paul, “Real-Time Monitoring Systems for Online Learning,” IEEE Learning Technologies, 2021.
[21] J. Stevens and K. Yoon, “Structured Quiz Design Using Bloom’s Taxonomy,” Educational Computing Research, 2021.
[22] K. Shah and M. Patel, “Comparative Study of Web Frameworks for Learning Platforms,” IJCSIT, 2023.
[23] L. Priya and A. Xavier, “Behavior Analytics in Programming Education,” IEEE Access, 2023.
[24] R. Thomas and S. Varun, “Survey of Intelligent E-Learning Architectures,” IEEE Access, 2024.