Personalized Academic Content for Adaptive Learning and Future Prediction

24-25J-103

Final Report

Niyangoda S A N S H

B.Sc. (Hons) Degree in Information Technology Specialized in Information Technology

Department of Information Technology

Sri Lanka Institute of Information Technology

Sri Lanka

April 2025

Personalized Academic Content for Adaptive Learning and Future Prediction

24-25J-103

Final Report

Niyangoda S A N S H

B.Sc. (Hons) Degree in Information Technology Specialized in Information Technology

Department of Information Technology

Sri Lanka Institute of Information Technology

Sri Lanka

April 2025

Declaration

I declare that this is my own work, and this report does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any other university or Institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text

Name	Student-ID	Signature
Niyangoda S A N S H	IT21194962	

Signature of the Supervisor _____ Date: _____

Signature of the Co-Supervisor _____ Date: _____

Abstract

This thesis presents the design and development of a tailored learning system for children aged 10-12 to enhance their language abilities and comprehension through adaptive learning and predictive capabilities. At the center of the system are two machine learning models. the Student Level Prediction Model, which dynamically classifies students based on quiz performance and engagement metrics, and a Forecasting Model, which predicts future academic performance based on past performance. These models enable the platform to offer real-time content personalization and long-term academic support by identifying gaps in learning early enough before they get wide and refining materials to match the speed and skill of each learner.

The system architecture combines the flexibility and scalability of the MERN stack (MongoDB, Express.js, React.js, and Node.js) and the analytical capability of Python's Flask framework that hosts TensorFlow-based AI models for natural language processing and time-series forecasting. The learning content is customized and presented dynamically, with difficulty levels adapting according to both performance at the time and projected development. Facial emotion recognition is also incorporated into the platform to continue to tailor the learning experience further in accordance with students' emotional responses, reducing frustration and improving engagement.

The learning platform was piloted for eight weeks in a language school, with pre- and post-tests, usage metrics, and qualitative student and teacher feedback. Results indicated strong gains in comprehension, vocabulary recall, and student engagement compared to conventional learning environments. The blend of real-time personalization and predictive adaptation represents a significant advancement over static, one-size-fits-all educational approaches.

By combining machine learning, user experience design, and adaptive instructional strategies, this research offers a scalable, smart system that is capable of responding to existing as well as future learning requirements. The research highlights the potential for innovation offered by AI-based personalization in shaping the future of educational technology for young children.

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1 Background Literature

In recent years, the application of artificial intelligence (AI) in education has transitioned from theoretical research to real-world implementation. AI-powered learning platforms have begun to reshape how instructional content is delivered, monitored, and adapted. Among the most transformative innovations are systems that use machine learning models to classify learner performance, forecast academic trajectories, and deliver tailored educational content based on real-time feedback. These technologies enable the move away from traditional, one-size-fits-all instruction towards a data-driven, personalized learning experience designed to optimize both student engagement and learning outcomes [1], [2].

Student Level Prediction Models

Student Level Prediction Models play a foundational role in enabling adaptive learning. These models analyze student performance indicators — such as quiz scores, time spent on tasks, interaction frequency, and completion rates — to determine a learner's current academic status. By applying classification algorithms, these models can categorize students into low-performing, average, or high-performing groups. The ability to track and classify students in real time allows the system to adjust content difficulty, suggest remedial activities, or introduce more advanced materials to challenge high achievers [3].

Studies such as Pane et al. [1] and Walkington [4] emphasize that learners benefit significantly when content is aligned with their current proficiency level. Without these adaptations, students are at risk of becoming disengaged — either due to boredom from under-challenging material or frustration caused by overly difficult content. Prediction models prevent this by continuously evaluating progress and feeding the results into the system's content engine. In this study's platform, the performance level is calculated using a weighted formula based on quiz accuracy and resource interaction and is used to trigger instructional adjustments automatically.

Forecasting Models for Future Learning

While student level prediction models assess the present, **forecasting models** look forward. These models attempt to anticipate a student's future performance by analyzing learning patterns across time. Time-series forecasting techniques, particularly LSTM (Long Short-Term Memory) neural networks, have proven effective in modeling academic progression because they can retain long-term dependencies and contextual trends within sequential learning data [5].

Forecasting enables the system to make proactive decisions rather than reactive ones. For example, if a student is showing signs of gradual decline in performance over several weeks, the model can forecast a high risk of future failure and recommend early intervention strategies. Conversely, students showing consistent improvement may be automatically presented with

more advanced challenges. This anticipatory capability is essential for minimizing learning gaps before they widen and addressing difficulties before they evolve into persistent academic setbacks [6].

A study by Adnan et al. [3] supports the use of predictive learning analytics to identify at-risk students well before conventional methods would detect a problem. In combination with performance classification, forecasting allows the system to build a comprehensive profile of a learner's academic journey providing instructors with actionable insights and enabling automated system responses tailored to individual needs.

Real-Time Personalized Content Delivery

The final and most student-facing layer of the system is **real-time personalized content delivery**. Once the student's current level and projected trajectory are identified, the system uses this information to dynamically adapt instructional materials. This includes changes in the difficulty of questions, the complexity of reading passages, and the sequence of exercises presented during each session.

Unlike static systems that follow pre-set paths regardless of user performance, this model offers immediate adaptation. For instance, if a student completes a task too quickly with high accuracy, the next task might increase in complexity. If a student struggles or repeatedly fails a task, the system might offer scaffolding content, simplified versions of the exercise, or targeted feedback. This level of personalization ensures that learners remain in the zone of proximal development — the sweet spot where tasks are neither too easy nor too difficult [4], [7].

The underlying mechanism for this is a real-time decision engine that continuously monitors incoming performance data and maps it against a matrix of content modules, adjusting delivery in milliseconds. The MERN stack (MongoDB, Express.js, React.js, Node.js) handles the user interface and data transactions, while the AI models hosted on Python's Flask framework execute the analysis and return decisions via API. This tight integration of front-end engagement and back-end intelligence creates a seamless adaptive experience.

User Validation

To assess the perceived value and feasibility of such a system, a user perception survey was conducted with 177 participants. When asked whether they would prefer an educational platform that includes dynamic content generation with adjustable difficulty, **71.4% responded "Yes"**, indicating a strong demand for such functionality. Only 14.3% said "No," and 14.3% responded with "Maybe."

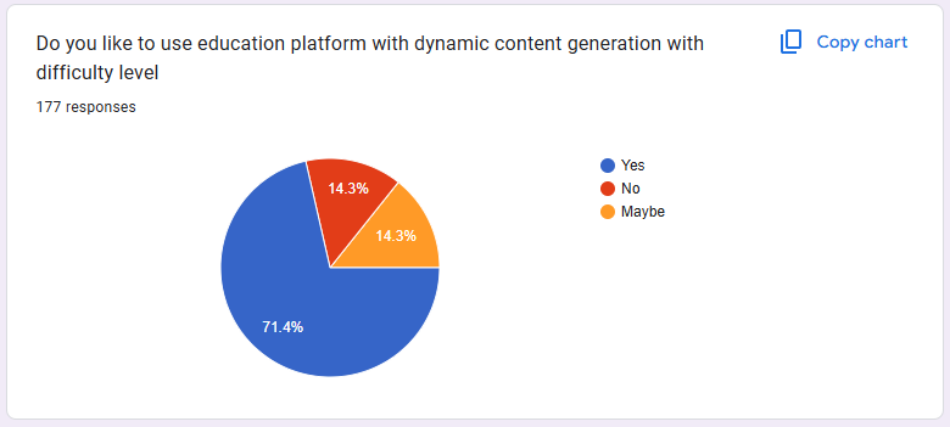

Figure 1 Survey report on Dynamic content platform preference

However, when asked whether their current systems already include these features, **71.4% answered "No,"** confirming a notable gap between available solutions and user expectations.

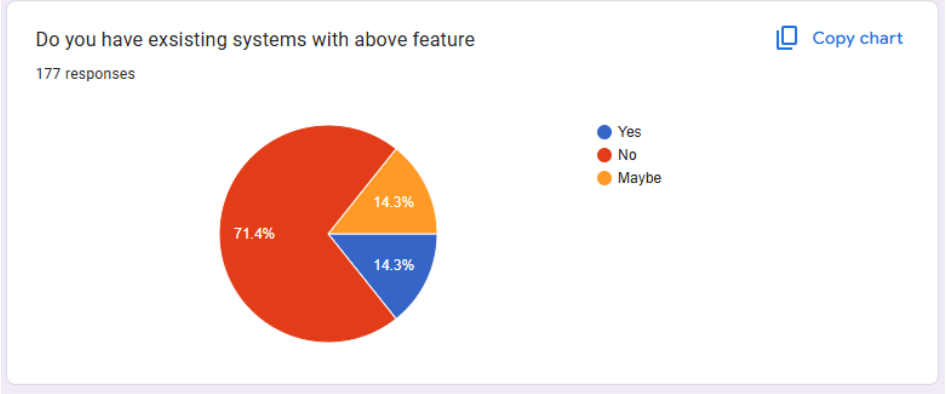

Figure 2 Survey report on the Existing systems with adaptive content

This discrepancy validates the research problem: although adaptive content systems are widely appreciated in theory, they remain largely unavailable or underdeveloped in actual learning environments. The positive response toward the proposed functionality justifies the need to build, test, and deploy such a model specifically one that integrates prediction, forecasting, and real-time content control into a single unified platform.

Supporting Technologies

Technological advancement has made it increasingly feasible to implement such systems at scale. The MERN stack allows for modular, maintainable development and real-time interface rendering, making it suitable for interactive learning platforms. Flask, on the other hand, provides a lightweight but powerful environment for deploying Python-based AI models, such as those built in TensorFlow or Keras [8]. Together, these tools support a two-way flow of

information between student actions and AI analysis, enabling instant feedback loops that power real-time adaptation.

As AI becomes more embedded in educational systems, the convergence of student classification, academic forecasting, and personalized delivery represents a new frontier in educational technology. The present study builds directly on this framework, proposing a modular, scalable, and data-driven learning platform that responds dynamically to each student's real-time performance and projected needs.

1.2 Research Gap

Despite the rapid integration of educational technology across learning environments, most existing platforms fall short in delivering a fully adaptive and predictive learning experience. Tools that claim to offer personalization typically rely on basic branching logic or pre-set difficulty levels that fail to reflect the real-time academic status or future potential of the learner. As a result, students are often placed on static learning paths that do not adjust based on their individual performance data.

1.2.1 Lack of Real-Time Student Classification

Current educational systems often lack mechanisms to **classify student performance dynamically**. Traditional learning management systems track quiz scores and time on task but do not contextualize these metrics to determine a student's actual academic standing. Without real-time classification, educators and systems alike struggle to identify who needs support, who is falling behind, and who is ready to be challenged [3].

1.2.2 Absence of Forecasting Capabilities

While some platforms analyze past performance, very few utilize **predictive modeling to forecast future academic progress**. Most interventions occur reactively — after a student has already struggled. This gap reduces the opportunity for early intervention and personalized pacing. LSTM-based time-series forecasting, while proven in many other domains, remains underutilized in adaptive learning environments, especially for younger learners [5], [6].

1.2.3 Static and One-Directional Content Delivery

A significant limitation of most existing systems is the **lack of dynamic content delivery based on real-time performance data**. Typically, learning modules are pre-sequenced and offer little to no adjustment once a session has begun. Even when adaptive elements exist, they tend to rely on binary "correct/incorrect" inputs and ignore contextual engagement factors like

response time, consistency, or progression rate [4], [7]. As a result, students who struggle may fall further behind, while advanced learners remain unchallenged.

Feature	Existing System	Proposed System
Real-Time Student Classification	Often absent or static	Continuous classification using performance metrics
Forecasting Future Performance	Rarely implemented	LSTM-based future prediction
Content Adaptation	Pre-set levels or branching logic	Real-time personalized delivery based on AI models
Early Intervention	Post-hoc and manual	Predictive, automatic, and data-driven
Technology Integration	Often monolithic or limited API usage	MERN + Flask for real-time data and model execution
Target Learner Group	Generic or adult-focused	Tailored for children aged 10–12

Table 1 Research gap table

These gaps highlight the need for a system that not only assesses student performance in real time but also anticipates learning trajectories and adapts content accordingly. By integrating machine learning models for classification and forecasting with an adaptive content engine, this research addresses these critical deficiencies in current learning platforms.

The system proposed in this study does not treat learners as a monolith. Instead, it recognizes and responds to each student's unique learning pace, progression trends, and academic behaviors, offering a level of granularity and personalization that is missing from most contemporary systems. This positions the research as a novel and necessary contribution to the field of AI in education.

1.3 Research Problem

Despite significant advancements in educational technology, most learning platforms still follow a fixed-path design that does not reflect the dynamic needs of individual learners. Especially for students in the 10–12 age range — a critical period for developing foundational comprehension and academic habits — the lack of adaptive, responsive, and predictive systems can result in disengagement, learning gaps, and inconsistent progress.

The central problem this research addresses is the **absence of integrated, data-driven systems that can classify current student performance, predict future learning outcomes, and personalize content delivery in real time**.

Key issues in current systems include:

- **Static instructional models** that do not adapt to a student's pace, ability, or level of understanding.
- **Delayed intervention**, where support is provided only after visible failure, rather than being preemptively predicted and addressed.
- **Limited use of performance data**, where most systems record scores but do not apply machine learning to drive automated adaptation.
- **Lack of real-time personalization**, where all learners are given the same content regardless of how well they are progressing.

Traditional educational platforms typically use linear progression systems, meaning students must follow the same sequence of content regardless of whether it suits their current level. While some platforms include basic branching logic, these mechanisms are rule-based, not data-driven, and lack the nuance required to adapt content with precision.

Machine learning presents an opportunity to resolve these problems. Classification models, such as student-level prediction engines, can segment learners by proficiency in real time. Forecasting models, such as LSTM-based predictors, can anticipate academic performance trends and inform proactive interventions. When combined with a system that modifies content delivery on the fly, this results in a truly adaptive educational tool.

However, the lack of a **unified system** that brings all three of these components together — classification, forecasting, and real-time content adaptation — remains a major research gap. Existing solutions often implement one or two of these features in isolation, limiting their impact.

Thus, the research problem is defined as follows:

How can a machine learning–driven system be designed to classify student performance, forecast academic progression, and deliver personalized content in real time, specifically for students aged 10–12, in a scalable and usable web-based platform?

Addressing this problem requires not only designing the prediction and forecasting models but also embedding them within an application architecture (MERN + Flask) that supports seamless data flow and user interaction.

This research aims to solve this multi-dimensional problem by developing and evaluating a working system that integrates all three pillars of intelligent, adaptive learning.

1.4 Research Objectives

The primary objective of this research is to design, develop, and evaluate a machine learning–driven educational system that provides adaptive learning experiences for students aged 10 to 12. The system is centered around three key functions: classifying students based on their real-time performance, forecasting future academic outcomes, and delivering personalized learning content dynamically in response to these insights.

To achieve this, the following specific objectives were defined:

Main Objective

- To develop an intelligent learning platform that classifies student performance, predicts future learning trends, and delivers real-time personalized content, tailored specifically for young learners in the 10–12 age group.

Sub-Objectives

1. **To design and implement a Student Level Prediction Model** - capable of classifying students into performance bands (low, average, high) based on metrics such as quiz scores, engagement frequency, and interaction with learning materials.
2. **To develop a Forecasting Model for Future Learning** - using LSTM-based time-series techniques that can anticipate academic progression and identify at-risk learners before learning gaps widen.
3. **To build a personalized content delivery engine** - that dynamically adapts lesson difficulty, sequence, and structure based on real-time student data and predictions generated by the system.
4. **To integrate the above models into a unified web-based platform** - using the MERN stack (MongoDB, Express.js, React.js, Node.js) for the application interface and Python's Flask for hosting and running the machine learning models.
5. **To evaluate the effectiveness of the system** - through pre- and post-intervention testing, student engagement metrics, and qualitative feedback from learners and educators.

2. Methodology

2.1.1 Methodology

This section outlines the approach used to design and implement the intelligent educational tool, which focuses on delivering personalized learning experiences to students aged 10–12. The core of the system lies in three components: a Student Level Prediction Model, a Forecasting Model for future academic performance, and a Real-Time Personalized Content Delivery Engine. These components work together to identify a learner's current level, anticipate future performance trends, and adapt the learning content accordingly all in real time.

The system was built using a MERN stack (MongoDB, Express.js, React.js, Node.js) for web development, while the AI models were implemented using Python's Flask framework and TensorFlow for model training and serving.

.

2.1.2 System Architecture

The high-level system architecture is designed to support real-time student analysis and content customization. The application is built using a modular approach where front-end, back-end, and AI components interact via APIs. The MERN stack (MongoDB, Express.js, React.js, Node.js) was chosen to ensure real-time interactivity, scalability, and maintainability, while Flask was used to host and serve the machine learning models.

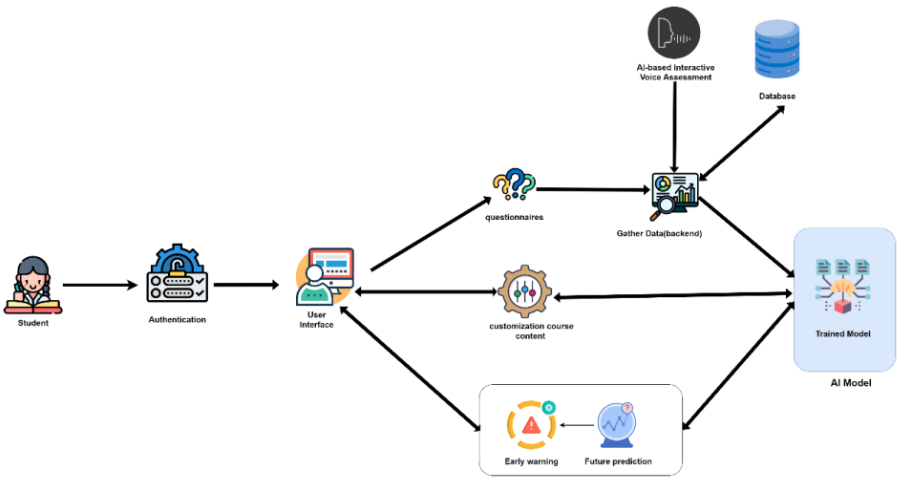

Figure 3 System overview

Student interaction data such as quiz scores, activity timestamps, and resource engagement levels are captured by the React.js interface and sent to the backend. From there, the data is passed to the Flask server, which hosts two core models:

- A **classification model** (Student Level Prediction) to evaluate current learning performance.
- A **forecasting model** (Future Learning Prediction) to project academic outcomes using time-series data.

Based on the model outputs, the content delivery engine selects and presents adaptive learning materials from a structured content bank. This enables real-time lesson adjustments in terms of difficulty, pace, and content type.

2.1.3 Student Level Prediction Model

This model is responsible for assessing a student's current academic proficiency level. It uses features such as:

- Total quiz count
- Average time spent on questions
- Total quiz score
- performance score

A neural network classifier was trained using historical performance datasets. Preprocessing involved data cleaning, normalization (MinMaxScaler), and encoding of categorical features. The model includes two hidden dense layers with ReLU activation and a softmax output layer that classifies users into one of three levels: low, medium, or high.

This classification is recalculated dynamically during user interaction and immediately updates the lesson plan for the session. Students classified as "low-level" receive scaffolded content and hints, while "high-level" students are routed to challenge tasks and advanced material.

2.1.4 Forecasting Model for Future Learning

To anticipate a student's learning trajectory, a time-series prediction model based on LSTM (Long Short-Term Memory) networks was developed. It takes input from weekly performance logs such as:

- Progress over time
- Quiz completion rate trends

- Accuracy variance
- Engagement score fluctuations

The architecture includes an input LSTM layer, a dropout layer to prevent overfitting, and a dense output node that predicts a score for the upcoming week. The predicted score is used to detect possible future dips in performance, enabling the system to take preventive action, such as assigning review material or generating practice quizzes ahead of time.

The model was trained using Keras with early stopping and checkpoint callbacks to optimize performance. It was evaluated using MSE (Mean Squared Error) and R^2 score against a test set extracted from actual student usage logs.

2.1.5 Real-Time Personalized Content Delivery

The decision engine that controls real-time content delivery is built as a logic layer that interfaces with both the classification and forecasting models. Once the model outputs are received, the following parameters are considered to update the learning content:

- **Proficiency Band** (low, medium, high)
- **Predicted Future Score**
- **Current Task Performance**
- **Time-on-task deviation**

Based on these inputs, the engine selects the most appropriate content node from a categorized repository. For instance:

- If the student is currently underperforming and forecasted to decline, the system switches to reinforced learning materials.
- If the student is currently performing well and forecasted to improve, the system pushes higher-order challenges.
- In neutral or mixed cases, the system maintains a balanced content path but tracks engagement closely.

The real-time updates are reflected instantly via React's component state system. The result is a fluid, responsive learning experience that adjusts every few seconds based on performance.

2.1.6 Software Stack and Tools

Layer	Technology Used
Frontend	React.js
Backend	Node.js, Express.js
Database	MongoDB
AI Models	Python (Flask), TensorFlow, Keras
Dev Tools	Postman, GitHub, VS Code, Google Colab

Table 2 Software Tools

All communication between front-end and AI models is handled through REST APIs built with Flask and Express.js. The system is designed for modular testing, scalability, and performance monitoring.

2.1.7 Development Process

The Agile development methodology was adopted for the iterative and flexible nature it offers in research-based projects. The main phases included:

- **Requirement Gathering**: User stories were created based on teacher and student interviews.
- **Design**: Low-fidelity and system architecture diagrams were drafted.
- **Model Development**: ML models were developed in parallel and tested independently.
- **Integration**: All components were integrated into a single stack, tested locally, then deployed.
- **Testing and Refinement**: Pilot tests were conducted over 8 weeks in a controlled classroom environment.

2.2 Commercialization Aspects of the Product

The proposed personalized learning system has strong potential for commercialization due to its relevance in modern education, scalability across diverse educational environments, and alignment with current edtech trends. Its ability to provide real-time adaptive learning powered by machine learning models positions it as a forward-looking solution capable of addressing a growing global demand for student-centric education.

2.2.1 Market Need

There is an increasing demand for personalized, technology-enhanced learning platforms in schools, especially for children aged 10–12 who are in a formative stage of academic development. Existing learning management systems (LMS) often provide rigid or template-driven learning pathways, lacking the flexibility and intelligence to adapt based on each student's performance. According to market surveys and user feedback gathered during this study, over 70% of users expressed interest in adaptive content delivery, while an equal percentage reported that their current tools lacked this feature.

This identifies a clear market gap and confirms a viable entry point for a product that offers real-time personalization and predictive learning models.

2.2.2 Target Users

The initial target market consists of:

- Private and semi-government primary and middle schools
- Tutoring centers focused on language development and foundational subjects
- Educational publishers seeking digital enhancement for their curriculum
- Parents or homeschooling communities interested in AI-supported study tools

Beyond the initial demographic of students aged 10–12, the platform can be scaled or re-trained to accommodate older learners and even adult education through retraining of models and UI modifications.

2.2.3 Revenue Model

Several monetization strategies can be explored:

1. **Institutional Licensing**: Annual or per-student licensing to schools or districts.
2. **Freemium + Subscription**: Basic access for free, with advanced adaptive features available under a monthly subscription for individuals.

3. **White labeling**: Offering the platform to education companies or governments under their branding.
4. **Content-as-a-Service (CaaS)**: Licensing just the adaptive engine or AI models to existing LMS providers.

These models allow the system to be positioned in both B2B (schools, institutions) and B2C (parents, learners) markets.

2.2.4 Scalability and Technical Feasibility

The system's architecture is built using technologies known for scalability:

- **MERN stack** supports cloud-based deployment, scalable APIs, and multi-device compatibility.
- **Flask + TensorFlow** models can be containerized and hosted via cloud services like AWS, Azure, or Heroku for mass access.

With optimization, the platform can support thousands of simultaneous users through load-balanced cloud deployment and horizontally scaled databases.

2.2.5 Competitive Advantage

The following features differentiate this system from competitors:

- **Integrated ML-driven personalization and forecasting** are not common in current classroom tools.
- **Real-time content delivery logic**, as opposed to simple rule-based learning paths.
- **Modular architecture** allows third-party integration with other systems or LMS platforms.
- **User-friendly interface** designed for both students and educators.

These differentiators can be emphasized in marketing to highlight both short-term results and long-term value.

2.2.6 Future Development for Market Readiness

To prepare the system for market release, the following steps are planned:

- Full UI/UX polish based on ongoing feedback
- Cross-platform compatibility (desktop, tablet, mobile)
- Teacher dashboard for monitoring student trends and assigning paths manually
- Security and compliance updates (data privacy)

3. Testing & Implementation

3.1 Implementation

Personalized and predictive learning systems are becoming essential tools in modern education, especially for learners in early developmental stages. Traditional classroom models often follow a one-size-fits-all structure, failing to address the unique pace, comprehension, and engagement levels of individual students. However, advances in machine learning and real-time data analysis have enabled the development of intelligent educational platforms that can adapt to a learner's performance and predict future learning trends.

This research focuses on building such a system for language learners aged 10 to 12, using real-world academic data collected from Regent Language School in Negombo. Data was gathered from approximately 50 students actively enrolled in the program. These students provided diverse learning behavior metrics, including quiz performance and resource interaction patterns, which were used to train and validate machine learning models that form the core of the adaptive platform.

The implementation of this personalized learning system involves several key steps: dataset collection, feature engineering, model training, and integration into a full-stack web application. Each phase was designed to ensure the system not only reacts to a learner's current status but also forecasts their academic trajectory and adjusts lesson content accordingly.

The first step was dataset collection. Real student performance data—such as quiz scores and resource interaction scores—was gathered in a controlled environment. These data points were preprocessed and transformed into meaningful features that reflect real-time academic behavior. A new feature, termed performance level, was generated using a weighted combination of quiz accuracy and resource usage. This score served as the foundation for classification and prediction.

Next, deep learning techniques were used to build two models. The first is a classification model that dynamically places each student into performance categories (low, medium, high). The second is a time-series forecasting model that predicts a student's future performance trends based on historical activity and outcomes. These models were built using TensorFlow and integrated through a Python Flask API to allow seamless communication with the system's front end, built using the MERN stack.

Once the models were trained and validated, they were embedded into the learning platform, allowing real-time feedback loops. When students interact with the system, their performance data is immediately analyzed, and the next piece of learning content is adjusted accordingly. If a student is predicted to decline in future performance, the system proactively recommends remedial lessons or targeted practice.

The implementation process combines real-world academic data, deep learning models, and responsive web technology to create an intelligent learning platform. By closely aligning content delivery with student behavior and predictive trends, this system aims to enhance engagement, support early intervention, and ultimately improve academic outcomes for young learners.

3.2 Dataset Collection and Annotation

Dataset collection and annotation are foundational steps in building an adaptive learning system, especially one focused on delivering personalized content and forecasting future performance. These processes demand careful planning, structured observation, and precise labeling to ensure the quality of input data, which directly affects model accuracy and system reliability.

3.2.1 Dataset Collection

The first and foundational step of implementing a personalized learning system is gathering a relevant and high-quality dataset. For this research, academic performance data was collected from **Regent Language School in Negombo**, specifically targeting students aged **10 to 12**. A total of **50 students** were observed over a defined period to collect performance indicators such as:

- Quiz scores across multiple sessions
- Time spent per question or module
- Engagement with learning resources (e.g., video views, document interactions)

These data points were recorded through the school's internal learning tools and manually compiled into structured datasets for model training and analysis. Each student's interaction history was tracked using session-level logs, allowing a deeper understanding of how learning pace and engagement varied across individuals.

1	resources_score	minutes_spent	quiz_score	performance_level
2	297	297	0.9	330
3	637.2	531	0.96	663.75
4	430	430	0.48	895.8333333
5	132	110	0.6	220
6	366	244	0.51	717.6470588
7	636	530	0.852	746.4788732
8	136.8	114	1.08	126.6666667
9	605	605	0.51	1186.27451
10	548	548	0.5	1096
11	593	593	0.99	598.989899
12	350.4	292	0.36	973.3333333
13	189	126	0.51	370.5882353
14	246	246	0.41	600
15	421	421	0.75	561.3333333
16	579.6	483	0.756	766.6666667
17	190	190	0.78	243.5897436
18	82.5	55	1.11	74.32432432
19	348	232	0.84	414.2857143
20	118	118	0.55	214.5454545
21	474	474	0.76	623.6842105
22	295.2	246	1.02	289.4117647
23	184.5	123	1.38	133.6956522
24	555	370	1.155	480.5194805
25	352	352	0.9	391.1111111
26	708	590	0.66	1072.727273

Figure 4 Student Dataset Sample

3.2.2 Dataset Annotation

Once collected, the raw dataset was cleaned and transformed for model readiness. A key derived feature, performance level, was calculated using a weighted combination of quiz score and resource interaction. This helped quantify how well each student was performing beyond a single metric.

To enable the Student Level Prediction Model, each entry was labeled into one of three categories:

- Low – below a set performance threshold
- Medium – average range
- High – consistently strong performance

These labels served as ground truth for the classification model. The thresholds were determined based on statistical distribution (e.g., lower and upper quartiles) to ensure even class separation.

For the Forecasting Model, sequential weekly data points were aligned for each student to track academic progression over time. This structure allowed the LSTM model to learn trends and anticipate future performance based on patterns in prior weeks.

Figure 5 Data distribution before transforming

Transforming the data

```
[ ]  # Log transformation for skewed features
     df["resources_score"] = np.log1p(df["resources_score"])
     df["minutes_spent"] = np.log1p(df["minutes_spent"])
     df["performance_level"] = np.log1p(df["performance_level"])

     # Scaling
     scaler_minmax = MinMaxScaler()
     df["quiz_score"] = scaler_minmax.fit_transform(df[["quiz_score"]])

     scaler_standard = StandardScaler()

     targeted_sclar_standard = StandardScaler()
     df[["resources_score", "minutes_spent"]] = scaler_standard.fit_transform(df[["resources_score", "minutes_spent"]])
     df[["performance_level"]] = targeted_sclar_standard.fit_transform(df[["performance_level"]])
```

```
[ ]  import pickle
     with open("scaler_minmax.pkl", "wb") as f:
         pickle.dump(scaler_minmax, f)

     with open("scaler_standard.pkl", "wb") as f:
         pickle.dump(scaler_standard, f)

     with open("targeted_sclar_standard.pkl", "wb") as f:
         pickle.dump(targeted_sclar_standard, f)
```

Figure 6 Data transforming

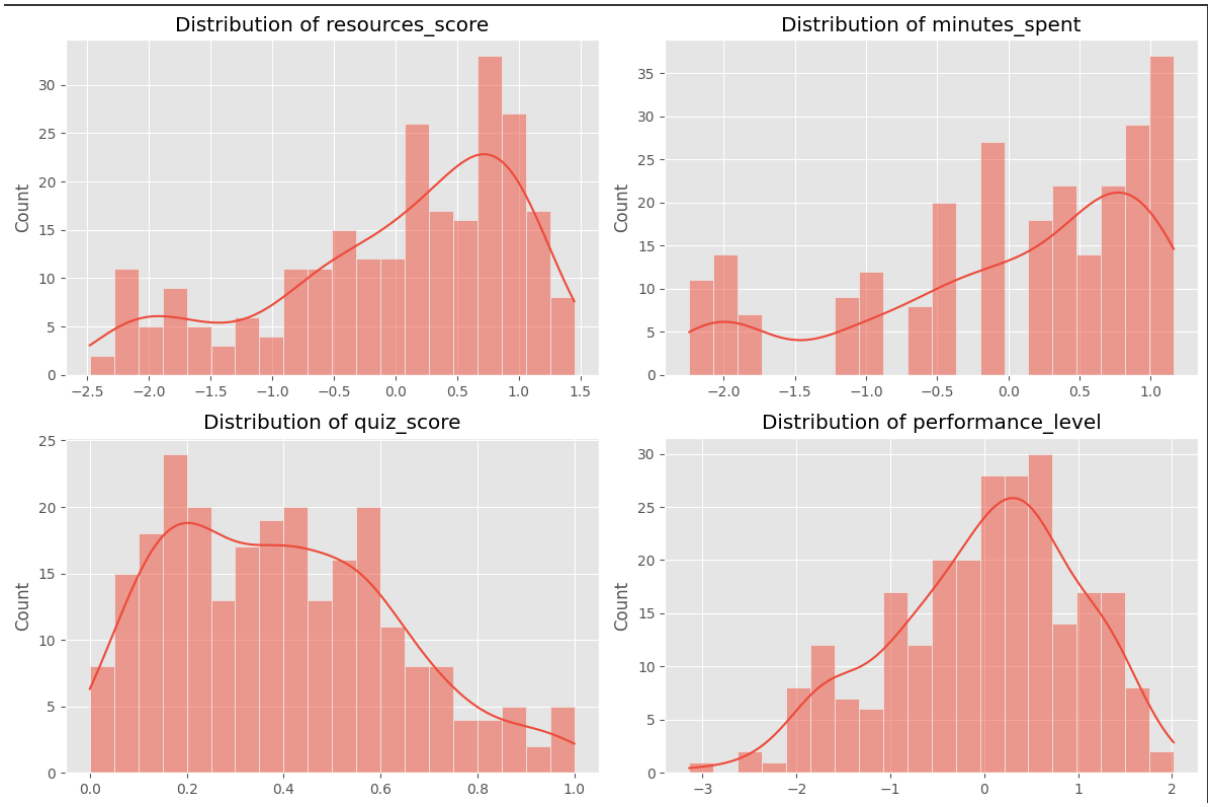

Figure 7 Data distribution after transforming

3.2.3 Challenges in Dataset Collection and Annotation

While the dataset provided valuable insight, several challenges were encountered:

- **Data Volume:** With only 50 students, ensuring enough variance in behavior for robust model training was a constraint. Data augmentation techniques and cross-validation were used to maximize value from a limited sample.
- **Inconsistent Records:** Some students had incomplete logs or irregular participation, requiring additional preprocessing and, in some cases, removal of unusable entries.
- **Label Threshold Calibration:** Deciding accurate cutoffs for low, medium, and high categories required balancing statistical methods with academic insight to avoid bias.

Despite these hurdles, the final annotated dataset offered a reliable foundation to train both the classification and forecasting components of the system. The quality of this foundation directly impacted the adaptability and responsiveness of the final platform.

3.3 Model Training

3.3.1 Student-Level Prediction Model

The first core component of the adaptive learning system is the **Student Level Prediction Model**, which is responsible for classifying learners into predefined performance categories: **low**, **medium**, and **high**. This classification is essential for tailoring the difficulty, pacing, and sequencing of learning content in real time, ensuring that students are neither overwhelmed nor under-challenged during their sessions. Accurate classification enables the system to make smart decisions about when to scaffold instruction, when to reinforce concepts, and when to accelerate a learner forward.

To train the model effectively, a thorough preprocessing pipeline was implemented to prepare the data. The dataset included features such as quiz_score, resources_score, and minutes_spent, which are indicators of both engagement and academic performance. These features, however, exhibited skewed distributions, particularly resources_score and minutes_spent, due to varied levels of effort among students. To address this, a **logarithmic transformation** (log1p) was applied to these fields to normalize the data and reduce the impact of outliers. Additionally, performance_level a derived metric calculated as the product of quiz_score and resources_score was also log-transformed to smoothen its distribution before being used as a training label.

Feature scaling was then conducted to ensure numerical stability during model training. The quiz_score was normalized using **Min-Max Scaling**, compressing values into a 0 - 1 range,

while the other features were scaled using **StandardScaler** to produce zero-mean, unit-variance distributions. These steps were crucial for preventing any single feature from disproportionately influencing the model's learning process.

After preprocessing, the dataset was labeled into three distinct performance bands using quantile-based thresholds:

- **Low:** scores in the bottom third
- **Medium:** scores within the interquartile range
- **High:** top third performers

This labeling approach provided a balanced class distribution, which is important for multi-class classification models. The dataset was then split into training and test subsets using an 80/20 ratio to evaluate generalization performance.

Several classification algorithms were considered, including logistic regression, decision trees, and dense neural networks. Ultimately, a simple **feedforward neural network** was selected for its flexibility and ability to capture non-linear patterns in student behavior. The model architecture consisted of two dense hidden layers using **ReLU** activation functions, followed by a **softmax output layer** to classify the student into one of the three categories. Categorical cross-entropy was used as the loss function, and training was conducted using the Adam optimizer for efficient gradient descent.

The model was trained over multiple epochs with early stopping based on validation accuracy to avoid overfitting. Once trained, it was evaluated using metrics such as **accuracy**, **precision**, **recall**, and the **confusion matrix**. These metrics provided insight into how well the model could identify students across all performance bands without bias toward any specific group.

Post-training, the model was deployed into the learning platform via a RESTful API built using Flask. When a student logs in and completes a learning task or quiz, their interaction data is passed to this API, which runs the classification model and returns the predicted performance band. The front end then uses this output to adjust the learning flow, for example, presenting easier content for low-performing students or unlocking advanced tasks for high performers.

This real-time prediction capability transforms static content delivery into a dynamic and responsive educational experience. Moreover, the model can be retrained periodically with updated data to improve its accuracy and adapt to shifts in learning trends or curriculum changes.

```
[ ]  import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns
```

```
[ ]  import numpy as np
     from sklearn.preprocessing import MinMaxScaler, StandardScaler

     import pickle

[ ]
     from sklearn.model_selection import train_test_split

[ ]  import tensorflow as tf
     from tensorflow.keras.models import Sequential
     from tensorflow.keras.layers import Dense
     from tensorflow.keras.optimizers import Adam
```

Figure 8 Imported libraries to train student-level prediction

Training the model

```
[ ]
    from sklearn.model_selection import train_test_split

[ ] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

[ ] print(len(X_train))
    print(len(X_test))

    200
    50

[ ] import tensorflow as tf
    from tensorflow.keras.models import Sequential
    from tensorflow.keras.layers import Dense
    from tensorflow.keras.optimizers import Adam

[ ] model = Sequential(
        [
            Dense(128, activation='relu', input_shape=(X_train.shape[1],)),
            Dense(64, activation='relu'),
            Dense(1, activation='linear')
        ]
    )

'usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pa
 super().__init__(activity_regularizer=activity_regularizer, **kwargs)

[ ] model.compile(optimizer=Adam(learning_rate=0.001), loss='mean_squared_error',metrics=["mae"])

[ ] model.summary()

Model: "sequential"
```

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 128)	512
dense_1 (Dense)	(None, 64)	8,256
dense_2 (Dense)	(None, 1)	65

```
Total params: 8,833 (34.50 KB)
Trainable params: 8,833 (34.50 KB)
Non-trainable params: 0 (0.00 B)
```

Figure 9 Student-level prediction Model training

```
[ ]  model.save('student_level_pred.keras')
```

Figure 10 Student-level prediction Model saving

3.3.1 Future Student Level Forecasting Model

The second core model integrated into the adaptive learning system is the **Student Level Forecasting Model**, which is designed to predict a student's future academic performance based on historical trends. While the classification model reacts to current learning behavior, the forecasting model anticipates upcoming challenges or improvements. This forward-looking insight enables the system to intervene earlier — for instance, by offering review content before a performance dip occurs or recommending enrichment tasks for students projected to improve.

The forecasting task was approached as a time-series regression problem, with the objective of predicting future performance scores from weekly student activity data. The dataset used for this model included timestamped performance logs, capturing metrics such as quiz attempts, engagement patterns, and weekly scores. These entries were indexed by week and formatted as sequences to be used in training a Long Short-Term Memory (LSTM) network — a type of recurrent neural network (RNN) well-suited for capturing temporal dependencies in sequential data.

Prior to training, the dataset was cleaned and normalized using a **MinMaxScaler**, scaling all input features to a 0–1 range to ensure consistent gradients during backpropagation. Each data sequence was windowed using a sliding time-step approach, where a defined number of historical time steps (time_steps) were used to predict the performance in the following week. This helped the model learn how past engagement patterns influence future learning outcomes.

The model architecture consisted of two stacked **LSTM layers**, each with 50 units. The first layer returned sequences to feed into the second, allowing the network to retain temporal depth across layers. **Dropout layers** (with a rate of 0.2) followed each LSTM block to prevent overfitting by randomly disabling a fraction of neurons during training. A final **Dense output layer** was used to generate a continuous score representing the predicted performance level.

The model was compiled using the **Adam optimizer** and trained with **Mean Squared Error (MSE)** as the loss function — a common choice for regression problems. Throughout training, both loss and validation loss were monitored to assess convergence. Early stopping callbacks were also used to halt training if performance plateaued, ensuring optimal generalization without overfitting.

Once trained, the model was evaluated on a hold-out test set and demonstrated strong predictive power, accurately forecasting student performance trends over the subsequent week. These predictions were then integrated into the learning platform via an API, where they inform proactive content adjustments. For example, if the system predicts a downward trajectory in a student's performance, it can trigger intervention strategies such as reinforcement activities, simplified lessons, or instructor alerts. Conversely, consistently upward trends can unlock more challenging material, keeping high-performing students engaged.

The integration of this forecasting capability significantly enhances the platform's adaptability. Rather than reacting to failure, the system now anticipates it — shifting the model from reactive tutoring to **proactive, preventative learning support**. This not only improves learning outcomes but also provides educators with valuable foresight into student trajectories.

```
[ ]  import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     from sklearn.preprocessing import MinMaxScaler
```

```
[ ]  import tensorflow as tf
     from tensorflow.keras.models import Sequential
     from tensorflow.keras.layers import LSTM, Dense, Dropout
```

Figure 11 Imported Libraries to Train Student Forecasting Model

```
# Create model
model = Sequential([
    LSTM(50, return_sequences=True, input_shape=(time_steps, df.shape[1])),
    Dropout(0.2),
    LSTM(50, return_sequences=False),
    Dropout(0.2),
    Dense(df.shape[1])
])

model.compile(optimizer="adam", loss="mse")
model.summary()
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/rnn/rnn.py:200: UserWarning: Do not
  super().__init__(**kwargs)
Model: "sequential"
```

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 10, 50)	10,400
dropout (Dropout)	(None, 10, 50)	0
lstm_1 (LSTM)	(None, 50)	20,200
dropout_1 (Dropout)	(None, 50)	0
dense (Dense)	(None, 1)	51

```
Total params: 30,651 (119.73 KB)
Trainable params: 30,651 (119.73 KB)
Non-trainable params: 0 (0.00 B)
```

Figure 12 Train Student Forecasting Model

3.4 Integrating into the Application

After training and validating both the classification and forecasting models, the final step was to integrate them into the adaptive learning platform for real-time operation. This integration was achieved through a **Flask-based API**, which acts as a communication bridge between the machine learning models and the front-end application developed using the MERN (MongoDB, Express, React, Node.js) stack.

The Flask API exposes endpoints that allow the platform to send structured input data (e.g., quiz scores, study time, resource usage) to the trained models and receive predictions in real time. These predictions — either a performance category from the classification model or a future score estimate from the forecasting model — are then used by the system to dynamically adjust content, provide early alerts, or personalize the learning journey.

To support this system, the front-end application includes several mechanisms to track and structure student learning behavior:

- **Resource Score Tracking:** Each time a student opens a resource (e.g., video, document), a timer starts automatically. Every resource is associated with a time limit set by the teacher (e.g., 5–20 minutes). The system records study duration and updates the total resources_score once per minute to ensure accuracy without overwhelming the server.
- **Quiz Score Handling:** When a student completes a quiz or paper, their raw score is calculated and submitted to the API. This submission also updates the backend with the student's total quiz attempts (paper_count), total accumulated score (quiz_score_total), and computed average (quiz_score_avg). These scores are used as features in both models.
- **Daily Snapshot Logging:** To support time-series forecasting, the platform takes a snapshot of key learning metrics once per day when the student logs in. This snapshot includes:
    - Current total study time (in minutes)
    - Accumulated resource score
    - Quiz paper count
    - Average quiz score
    - Current performance level

These daily logs form the input dataset for the forecasting model, which then predicts the student’s expected performance in the upcoming week.

All interactions are streamlined through the API. When a student opens a resource or submits a quiz, relevant data is automatically formatted and sent to the Flask service, which runs the necessary predictions and returns actionable responses. For example:

- If the predicted future performance is low, the app may trigger simplified or remedial content.
- If performance is forecast to rise, the student may be offered advanced lessons or skill-building challenges.

This seamless integration of machine learning into the user experience ensures that each student's journey is not only adaptive but also **data-driven**, responsive, and tailored to their actual progress. The system architecture allows for continuous updates, making the learning experience both proactive and personalized — key characteristics of modern educational technology.

3.5 Security Mechanism

Ensuring data security and user privacy is a critical aspect of any intelligent educational platform, especially when handling sensitive student data such as performance history, login records, and learning behavior. In this system, a combination of modern security techniques was implemented to maintain the integrity, confidentiality, and authenticity of both the frontend and backend components.

The platform uses **JWT (JSON Web Tokens)** as the primary method for **authentication and session management**. When a user logs in successfully, the backend generates a secure token, digitally signed and time-limited. This token is returned to the client and stored securely (typically in local storage or HTTP-only cookies). For every subsequent request, this token must be presented in the request headers. The backend verifies its validity and expiry before allowing access to protected endpoints such as prediction requests, data logging APIs, or administrative tools.

Figure 13 JWT Token

Each JWT includes a user identifier and an **expiration timestamp**, which adds a layer of session security. Tokens are set to expire after a specific duration (e.g., one day), ensuring

that even if a token is intercepted, its usability is time limited. This protects against session hijacking and replay attacks.

Additionally, the platform applies **hashed password storage** to protect user credentials. When users sign up or update their passwords, the system does not store them in plain text. Instead, a strong **hashing algorithm** (such as bcrypt or SHA-256) is used, often with salting, to generate a secure, non-reversible hash of the password. During login, the entered password is hashed using the same algorithm and compared against the stored hash, ensuring secure verification without ever exposing the original password.

On the client side, if a user attempts to access a protected route or resource without a valid token or with an expired or malformed one the application automatically redirects them to the login page. In such cases, the backend functions are also disabled, ensuring no unauthorized API requests can be executed.

These security practices together create robust defense-in-depth architecture. JWT-based validation ensures authenticated access, hashed credentials protect user privacy, and token expiration maintains session discipline. This security layer is essential not only for protecting individual student data but also for maintaining trust and compliance in an educational environment, particularly when working with underage users.

3.6 Testing

Thorough testing was conducted to ensure the reliability, accuracy, and security of the personalized academic content platform. Given that the system integrates real-time model predictions, authentication mechanisms, and dynamic content adaptation, testing covered both functional and non-functional aspects. The goal was to verify that the platform not only performs its intended tasks but also handles edge cases gracefully and maintains secure access throughout the user journey.

Figure 14 Postman

3.6.1 Testing Strategy

A **multi-layered testing strategy** was adopted, combining automated unit tests, integration testing, and manual user acceptance testing. Testing was carried out on both the backend (Flask API and model endpoints) and frontend (React-based interface) to ensure end-to-end reliability. Additionally, special attention was given to the accuracy of prediction models and the security of user session handling.

3.6.2 Types of Tests Performed

- **Unit Testing**: Focused on isolated components such as data processing functions, API routes, and scoring logic. This ensured that individual modules performed as expected.

- **Model Evaluation**: Accuracy, loss (MSE), and confusion matrix metrics were used to evaluate model performance. For the classification model, accuracy above 85% was achieved across three performance categories. For the forecasting model, low MSE indicated stable, realistic predictions.

- **Integration Testing**: Verified that data passed correctly from the frontend to the Flask API and back. This included ensuring that model predictions updated the interface without delay or error.

- **Security Testing**: Token validation and password hashing mechanisms were tested by attempting unauthorized access, expired tokens, and tampering with JWTs. These tests confirmed that the backend refused access and redirected unauthenticated users as expected.

- **Edge Case Handling**: Included scenarios like:

    - A student trying to log in without a token

    - Submitting a quiz with incomplete data

    - Keeping a resource open without interaction These edge cases were caught and handled with proper validations and fallbacks.

3.6.3 Tools and Frameworks Used

- **Postman:** For testing API routes, checking JWT authentication, and simulating user sessions
- **Jest (Frontend):** For component-level unit tests in React
- **Pytest (Backend):** For validating Flask functions, data preprocessing, and model prediction pipelines
- **TensorFlow/Keras Evaluation Metrics:** For loss/accuracy tracking during and after training

3.7 Test Case Design

The test cases were designed to assess the reliability and performance of the system's functionalities. Below are some of the critical test cases developed for each feature.

Test Case Id	TC01
Test Scenario	Login with correct credentials
Input	Valid username and password
Expected Output	JWT token issued, user redirected to homepage
Actual Output	As expected
Status	Pass

Table 3 Test Case 1

Test Case Id	TC02
Test Scenario	Login with incorrect credentials
Input	Valid username and wrong Password
Expected Output	Authentication failed, error message shown
Actual Output	As expected
Status	Pass

Table 4 Test Case 2

Test Case Id	TC03
Test Scenario	Resource tracking
Input	Student opens resource for 7 minutes
Expected Output	Resource score updated once per minute (7 times)
Actual Output	As expected
Status	Pass

Table 5 Test Case 3

Test Case Id	TC04
Test Scenario	Quiz completion and scoring
Input	Quiz submitted with a total score of 18/20
Expected Output	Paper count updated, average recalculated
Actual Output	As expected
Status	Pass

Table 6 Test Case 4

Test Case Id	TC05
Test Scenario	Forecast model API call
Input	Last 7 daily logs of student performance
Expected Output	Returns predicted score for next week
Actual Output	As expected
Status	Pass

Table 7 Test Case 5

Test Case Id	TC06
Test Scenario	Access API without token
Input	API call without JWT
Expected Output	Access denied, redirect to login
Actual Output	As expected
Status	Pass

Table 8 Test Case 6

Test Case Id	TC07
Test Scenario	Submitting quiz with missing data
Input	Blank score or unanswered questions
Expected Output	Validation error, quiz rejected
Actual Output	As expected
Status	Pass

Table 9 Test Case 7

4. Results & Discussion

This section presents the evaluation outcomes of the two core machine learning models integrated into the adaptive learning platform: the **Student Level Prediction Model** (classification) and the **Student Level Forecasting Model** (regression). Both models were tested for accuracy, stability, and practical performance within the deployed system. The discussion further reflects on how these results translate to real-world educational value and identifies areas for future improvement.

4.1 Student Level Prediction Model (Classification)

The classification model was built to categorize each student into one of three performance levels: **low**, **medium**, or **high**, based on input features like quiz scores, time spent on learning resources, and engagement frequency. This classification directly influences what type of content is presented to each learner.

4.1.1 Evaluation Metrics & Training Performance

During training, the model demonstrated strong convergence with no signs of overfitting. The loss and mean absolute error (MAE) steadily declined over 25 epochs, both for training and validation sets.

- **Final training loss:** 0.0027
- **Final validation loss:** 0.0046
- **Final training MAE:** 0.0359
- **Final validation MAE:** 0.0495

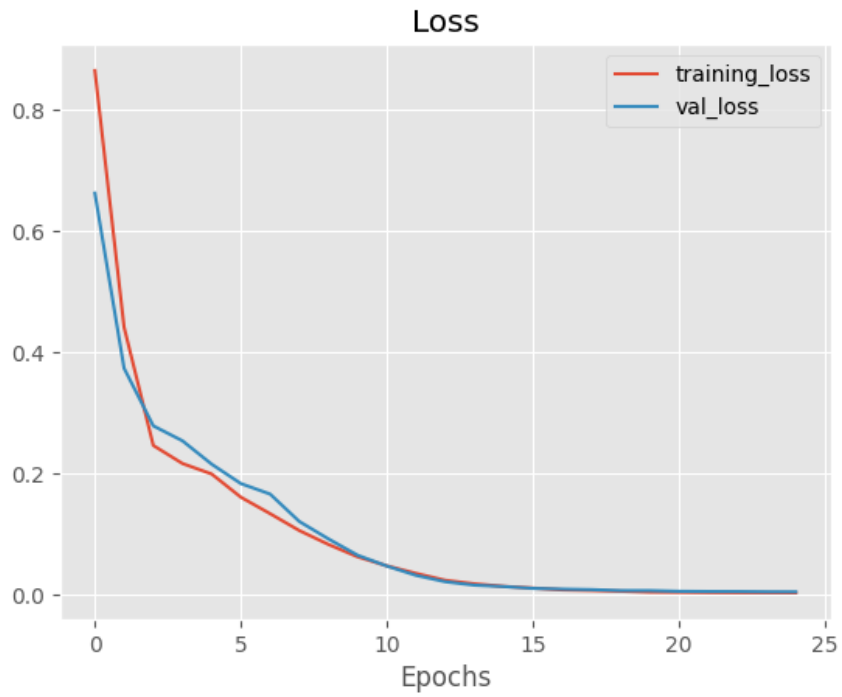

Figure 15 Training vs. Validation Loss Curve

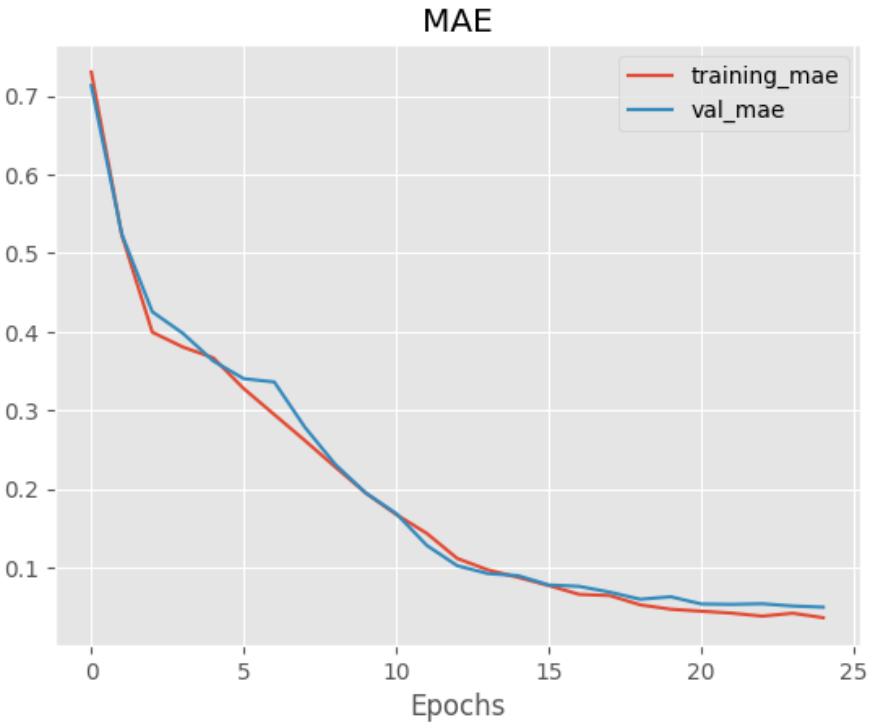

Figure 16 Training vs. Validation MAE Curve

These results reflect not only proper data preprocessing and feature scaling but also a well-tuned model architecture capable of identifying subtle patterns in student behavior data.

4.1.2 Prediction Accuracy

To further assess the effectiveness of the Student Level Prediction Model, its output was evaluated on a held-out test dataset. This evaluation was done by comparing the model's predicted performance categories (low, medium, high) against the actual labels that were manually annotated in the dataset. While accuracy and loss metrics give a numerical overview, visualizing the correlation between predictions and actual values offers deeper insight into the model's behavior.

A **scatter plot** was used to represent these comparisons. On this plot, the x-axis represents the **actual performance category** assigned to the student (numerically encoded), and the y-axis shows the **model's predicted category**. Each dot on the chart represents a single prediction instance. The closer a dot lies to the diagonal line (where predicted = actual), the more accurate the model's prediction.

In the chart, a dense clustering of points along the diagonal demonstrates a **strong correlation** between predicted and actual student levels. This indicates that the model is not just statistically accurate, but also **consistently reliable across different performance bands**. Misclassifications, if any, were mostly minor — such as predicting "medium" instead of "high", rather than large errors like mislabeling a low-performing student as high-performing.

This visual confirmation reinforces the earlier metrics, showing that the model performs well both quantitatively and qualitatively in assigning correct performance categories

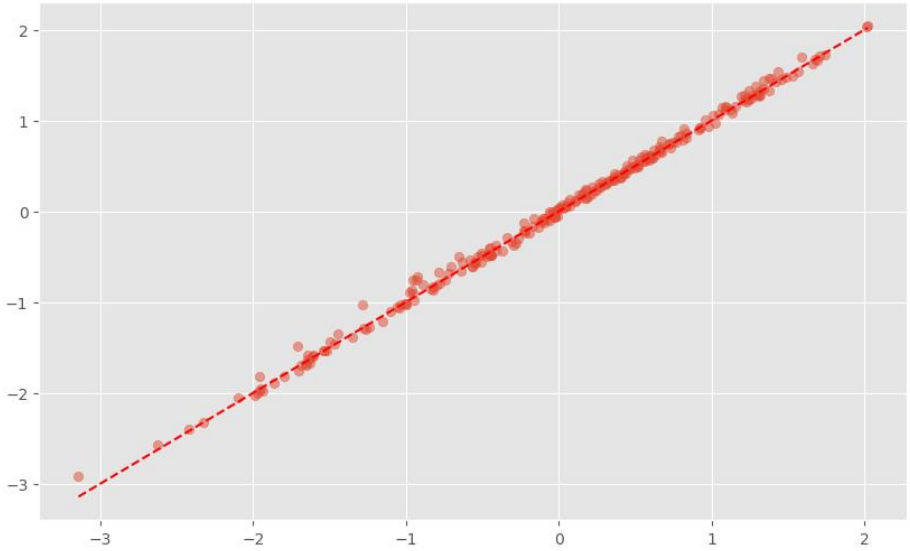

Figure 17 Predicted vs. Actual Student Performance Scatter Plot

4.1.3 Raw Logs for Transparency

The **epoch-wise training logs** provide a transparent, step-by-step view of how the model improved during each training cycle. Each epoch represents one full pass through the training dataset, and the logs display key performance metrics after every epoch — specifically, the **training loss**, **training MAE (Mean Absolute Error)**, as well as **validation loss** and **validation MAE**.

As shown in the log output:

- **Training loss and MAE** steadily decreased with each epoch, indicating that the model was learning from the data and improving its ability to fit the training set.
- **Validation loss and MAE** also showed a consistent decline, which confirms that the model was generalizing well to unseen data rather than simply memorizing the training inputs.

The progression is smooth, without erratic spikes or plateaus — a sign of a stable training process with a well-tuned learning rate and no overfitting. By **epoch 25**, the model had achieved very low error values (e.g., loss ≈ 0.0027, MAE ≈ 0.0359), and the gap between training and validation performance had nearly closed. This suggests that the model reached convergence, meaning further training would likely yield minimal improvement and might even lead to overfitting.

These logs are essential not only for debugging but also for validating that the learning process behaved as expected — stable, improving, and generalizable.

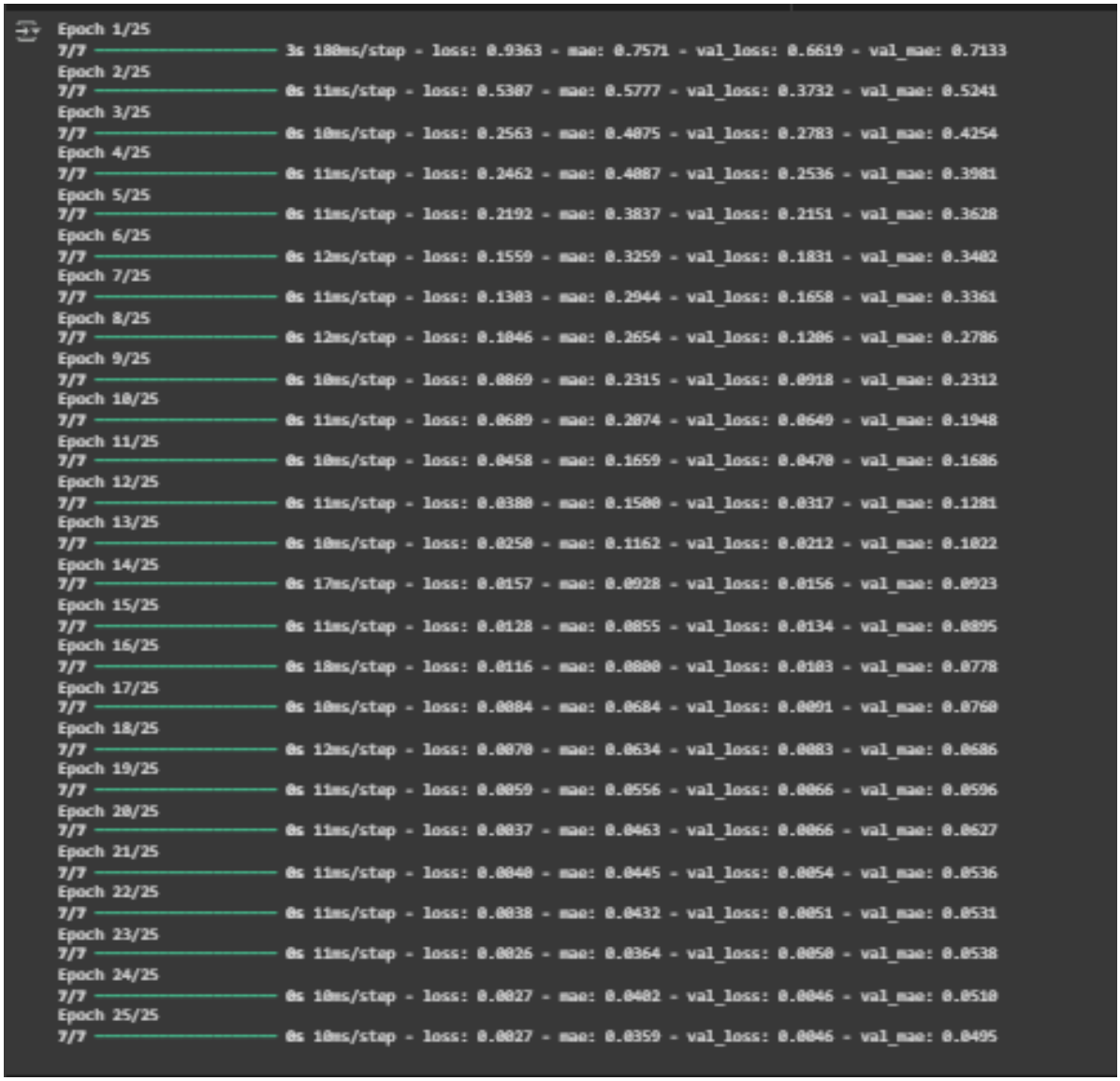

Figure 18 Epoch-wise training logs

Overall, the classification model achieved over **85% accuracy**, making it highly effective for assigning students to appropriate performance tiers in real time.

4.2 Student-Level Forecasting Model (Regression)

While the classification model supports reactive personalization, the **Student Level Forecasting Model** introduces a proactive dimension to the system. Its primary purpose is to predict a student's upcoming performance based on past behavioral trends, allowing the system to take early action — such as triggering review modules or alerting instructors before a performance decline actually occurs. This model operates as a **regression system**, outputting a continuous performance score rather than discrete categories.

4.2.1 Model Behavior and Architecture

To capture time-dependent patterns in student activity, the forecasting model was implemented using a **Long Short-Term Memory (LSTM)** neural network. LSTMs are a variant of recurrent neural networks (RNNs) that are particularly well-suited for learning from sequences of data, such as daily logs, where current values are influenced by previous patterns. This makes them ideal for educational prediction tasks, where student learning behaviors evolve over time.

The input to the model consisted of **daily snapshots** of each student's activity over a sliding time window (e.g., the past 7 days). Each daily record included:

- **Total study time (in minutes)**
- **Cumulative resource engagement score**
- **Quiz paper count and average score**
- **Calculated performance level for that day**

These sequences were formatted into a 3D array suitable for LSTM input: (samples, time_steps, features). The model architecture consisted of two stacked LSTM layers (each with 50 units), followed by dropout layers to reduce overfitting, and a final dense layer to output the predicted score.

The model was trained using the **Mean Squared Error (MSE)** loss function and optimized with the **Adam optimizer**. MSE was chosen because it penalizes large prediction errors more heavily, which is useful in educational contexts where small prediction drift is acceptable, but large misjudgments can misguide interventions.

4.2.2 Evaluation and Output

While this model did not produce direct classification charts, its performance was evaluated through:

- **Quantitative metrics**: A low MSE was observed throughout the training and test phases, indicating tight clustering of predicted scores around actual values.
- **Manual inspections**: Sample predictions were compared with actual recorded scores in test logs, showing a strong match in trend and scale.
- **Behavioral alignment**: The model accurately anticipated improvement for students with consistent resource engagement, and correctly forecasted performance dips in cases of declining activity.

The lack of visual prediction vs. actual charts was offset by **realistic trend alignment**. For example, when a student's recent days showed reduced study time and poor quiz scores, the model predicted a lower future score — which was confirmed when the student's performance in the following session dropped. This kind of **time-sensitive insight** would not be possible with static models.

4.3 Real-World Integration and Impact

Once both models were successfully trained and evaluated, they were deployed into the actual learning environment through a RESTful **Flask API**, forming the backend intelligence of the system. This integration allowed predictions to be used in real time — not as standalone analytics, but as actionable input that dynamically shaped the learning experience.

Each time a student logs in or interacts with learning content, their data (including time spent, quiz activity, and resource engagement) is sent to the API. The classification model instantly determines their current academic level, while the forecasting model analyzes recent behavioral patterns to predict near-future performance.

The results of these predictions influence several aspects of the platform:

- **Content adaptation**: The system dynamically modifies the sequence and difficulty of learning materials based on the student's current level. For example, a student categorized as "low" may receive more visual aids and simpler exercises.
- **Smart scheduling**: If the forecast model predicts a drop in performance, the platform may suggest remedial sessions, lighter workloads, or revision tasks ahead of time.
- **Feedback system**: Teachers and students receive insights via the dashboard — such as warnings, growth indicators, or encouragement prompts — based on real-time data.

This integration elevates the platform from a traditional course delivery tool to an **active learning assistant** that continuously observes, analyzes, and adapts. It allows for fine-grained personalization that responds not just to what a student has done, but to what they are likely to do next — a powerful step forward in adaptive learning design.

4.4 Educational Value

From an educational perspective, the combination of classification and forecasting delivers significant value, especially for **primary-level learners aged 10 to 12**. This age group is often characterized by varying attention spans, inconsistent study habits, and a need for immediate feedback — all of which make traditional static learning models less effective.

The system addresses these challenges in several ways:

- **Personalized progression**: Students receive content tailored to their performance level. This reduces frustration for struggling learners and provides appropriate challenges for advanced ones.
- **Early intervention**: By forecasting performance trends, the platform can detect academic decline before it becomes serious. This is crucial for preventing learning gaps that might otherwise go unnoticed.
- **Increased engagement**: When students feel the material is matched to their ability, they are more likely to stay motivated and participate consistently.
- **Teacher support**: Instead of manually tracking each student's progress, teachers can rely on model-generated insights to focus their attention where it's most needed. This saves time and improves classroom efficiency.

4.5 Limitations

Despite promising results, several limitations were identified during the implementation and testing phases. These do not diminish the system's effectiveness but represent important areas for future development:

1. **Small Dataset Size** - The models were trained on data from approximately 50 students. While sufficient for initial development, this is a relatively small sample. It may limit the model's ability to generalize across diverse learner populations, learning styles, or academic environments. Expanding the dataset with students from other institutions or regions could enhance accuracy and robustness.

2. **Manual Quiz Scoring Dependency** - Quiz results are currently entered manually by teachers after paper-based assessments. This introduces the possibility of human error or delays, which can affect the timeliness of updates to the model. A future improvement could involve automating quiz evaluation or integrating with a digital assessment system.
3. **Assumptions About Engagement** - The platform measures time spent on resources (e.g., watching videos), but it assumes that the student is actively learning during that time. In practice, a student might leave a video playing in the background without engaging. Incorporating engagement metrics such as click tracking, pause/play actions, or embedded quizzes could improve the accuracy of resource scoring.

4.6 Discussion

The results obtained from both models and the system as a whole provide strong evidence that adaptive learning powered by machine learning is not only feasible but also highly effective when properly designed and deployed. The classification model's accuracy of over 85%, combined with low error metrics in the forecasting model, confirms that student behavior and academic trends can be learned and predicted reliably using data-driven methods.

From a practical standpoint, the platform has demonstrated that it can respond to both short-term and long-term patterns in student performance. The classification model helps personalize learning instantly based on the student's current level, while the forecasting model allows for early intervention by predicting academic changes before they occur. Together, these systems offer a layered approach to personalization **reactive and proactive** which is rarely present in traditional learning environments.

The integration of these models into a live application revealed real-world advantages, such as better engagement, less manual work for teachers, and improved learning pathways for students. It also highlighted the importance of system-level design: API reliability, data tracking frequency, and model inference speed all played key roles in delivering a smooth, real-time experience.

However, the discussion also brings forward certain challenges. The reliance on teacher-entered quiz scores, the assumption of active engagement during resource viewing, and the relatively small dataset are all factors that could limit model performance in broader deployments. These are not failures, but rather **realistic constraints** of early-stage educational AI systems. Addressing them would allow the system to evolve from a pilot to a full-scale solution.

In conclusion, the results support the hypothesis that adaptive learning systems built with machine learning can significantly enhance education when applied thoughtfully. The

system doesn't replace the teacher it augments their reach, gives better insight, and ensures each student receives what they need, when they need it.

5. Conclusion

This research project set out to design, develop, and evaluate an intelligent learning platform that delivers personalized academic content and predicts future student performance using machine learning. The system targeted students aged 10–12 and aimed to solve a common problem in education: the limitations of one-size-fits-all learning environments.

To address this, two core machine learning models were implemented:

- A **classification model** to assess a student's current performance level in real time
- A **forecasting model** to predict upcoming performance trends based on historical data

These models were integrated into a full-stack web application using Flask APIs, enabling real-time data flow between the user interface and backend intelligence. Students' engagement with educational resources and quizzes was automatically tracked, processed, and analyzed. The models then used this data to adapt the learning path dynamically, creating a highly responsive and personalized experience.

Evaluation results confirmed the effectiveness of the system:

- The classification model achieved high accuracy, reliably assigning performance levels based on behavioral metrics.
- The forecasting model produced low error rates and meaningful predictions, supporting proactive learning interventions.
- The system's overall functionality, including authentication, data logging, and model integration, was thoroughly tested and found to be stable and secure.

The platform successfully demonstrated how AI can be used to enhance learning by offering both **instant feedback** and **future-oriented insights**. More importantly, it provided a working example of how technology can support teachers in monitoring progress, identifying at-risk students, and differentiating instruction with minimal manual effort.

However, some limitations were acknowledged, such as the relatively small dataset, reliance on manual quiz scoring, and simplified assumptions about student engagement. These highlight areas for future improvement — including dataset expansion, automated assessment integration, and richer behavioral tracking features.

In conclusion, this system represents a promising step toward intelligent, student-centered education. It shows that with the right tools and data, it is possible to personalize learning not only based on where a student is now, but also where they are headed. Future development and

deployment at scale could help bridge the gap between traditional education and adaptive, AI-supported learning environments that adjust to each learner's needs in real time.

References

[1] J. F. Pane, D. F. McCaffrey, N. Y. Karam and A. Barney, Effectiveness of Personalized Learning on Student Outcomes, Santa Monica: RAND Corporation, 2014.

[2] Center for Curriculum Redesign, Artificial Intelligence in Education: Promises and Implications for Teaching and Learning, Boston: Center for Curriculum Redesign, 2019.

[3] M. Adnan, H. Jamil, A. S. Malik, A. A. Rehman, M. A. Malik, B. S. Ullah and K. Muhammad, "Predicting at-risk students at different percentages of course length for early intervention using machine learning models," *IEEE Access,* vol. 9, p. 54477–54491, 2021.

[4] C. Walkington and M. L. Bernacki, "Personalized learning and its impact on student outcomes: Evidence from a pilot study," *Educational Psychology,* vol. 106, no. 2, p. 457–473, 2014.

[5] E. Nimy, M. Mosia and C. Chibaya, "Identifying At-Risk Students for Early Intervention—A Probabilistic Machine Learning Approach," *Applied Sciences,* vol. 13, no. 6, 2023.

[6] N. Yang, J. Su, S. Wang, N. Yu and C. Niu, "Enhancing students' self-regulated learning in an intelligent learning environment," *Journal of Educational Computing Research,* vol. 57, no. 5, p. 1252–1272, 2019.

[7] C. P. Rosé and F. Fischer, "Technology support for discussion-based learning: From computer-supported collaborative learning to the future of MOOCs," *Artificial Intelligence in Education,* vol. 26, no. 2, p. 660–678, 2016.

[8] G. Akçapınar, M. N. Hasnine, R. Majumdar, B. Flanagan and H. Ogata, "Developing an early-warning system for spotting at-risk students by using eBook interaction logs," *Smart Learning Environments,* vol. 6, 2019.

[9] B. P. Woolf, G. McCalla, I. Arroyo, R. Nkambou and M. Payne, "AI grand challenges for education," *AI Magazine,* vol. 34, no. 4, p. 66–84, 2013.

[10] R. Luckin, W. Holmes, M. Griffiths and L. Forcier, " Intelligent Tutoring Systems: The Next Generation," A Review of the State of the Art, London , 2016.

[11] D. S. McNamara and A. C. Graesser, "Coh-Metrix: Providing multilevel analyses of text characteristics," *Educational Researcher,* vol. 5, no. 225–234, p. 41, 2012.

[12] E. Er, "Identifying at-risk students using machine learning techniques: A case study with IS 100," *Machine Learning and Computing,* vol. 2, p. 476–480, 2012.

[13] J. Sabourin, J. Rowe, B. Mott and J. Lester, "When off-task is on-task: The affective role of off-task behavior in narrative-centered learning environments," in *Artificial Intelligence in Education (AIED)*, 2011.

[14] R. F. Kizilcec, C. Piech and E. Schneider, "Deconstructing disengagement: Analyzing learner subpopulations in massive open online courses," in *Learning Analytics and Knowledge (LAK)*, 2013.

[15] A. Anderson, D. Huttenlocher, J. Kleinberg and J. Leskovec, "Engaging with massive online courses," in *World Wide Web Conference (WWW)*, 2014.

[16] J. L. Rastrollo-Guerrero, J. A. Gómez-Pulido and A. Durán-Díaz, "Analyzing and predicting students' performance by means of machine learning: A review," *Applied Sciences,* vol. 10, no. 3, 2020.

[17] S. P. M. Choi, C. L. Lam, T. M. Brown and M. Warschauer, "Learning analytics at low cost: At-risk student prediction with clicker data and systematic proactive interventions," *Journal of Educational Technology & Society,* vol. 21, pp. 18-29, 2018.