AI-Augmented Assignment Management: A Hybrid RL-LLM Framework for Personalized Scheduling and Content Generation in Education

Yashwardhan Khanna
Department of Computational Intelligence
SRMIST,KTR,Chennai
yd7152@srmist.edu.in

Ishan Manglik
Department of Computational Intelligence
SRMIST,KTR,Chennai
in2832@srmist.edu.in

Dr. A. Revathi
Department of Computational Intelligence
SRMIST,KTR,Chennai
revathia1@srmist.edu.in

***Abstract*— Blending reinforcement learning (RL) and large language models (LLMs), this work introduces an AI-based framework for adaptive assignment scheduling and personalized task generation for learning environments. Optimizing assignment deadlines on the fly, the system employs a Proximal Policy Optimization (PPO) agent to consider student workload patterns. LLMs generate natural language explanations for each assignment's due date, thus ensuring interpretability and allowing teachers and students to understand the rationale behind scheduling decisions. Employing past performance and topic knowledge, a clustering module tailors assignment. Current and future developments involve an AI-driven insights panel for visualizing workloads and burnout prediction as well as an "Assignment DNA" feature for classifying assignments according to fundamental competencies such as critical thinking and data analysis. Remaining open and explainable, this platform aims to foster the integration of predictive artificial intelligence and adaptive learning within intelligent educational systems.**

Keywords— reinforcement learning, PPO, large language models, adaptive scheduling, explainable AI, educational technology, personalized learning, student workload, AI in education, assignment DNA.

I. Introduction

Educational institutions find themselves increasingly with the challenge of coordinating diverse students' workloads while maintaining even academic outcomes. Fixed assignment planning systems tend to neglect individual differences in learning rates, conflicting work assignments, and student welfare to create inefficiency and academic strain. Recent technologies in artificial intelligence (AI) present a challenge to overcome the shortcomings through dynamic, data-oriented solutions. This paper presents a new AI-powered scheduling framework based on Proximal Policy Optimization (PPO) for reinforcement learning and large language models (LLMs) for producing human-understandable explanations of scheduling choices. The framework is built with Python for backend operations, MongoDB for handling dynamic workload data, and JavaScript for a user-friendly web-based interface, all containerized using Docker for horizontally scalable deployment. Through its alignment with United Nations Sustainable Development Goal 4 (Quality Education), the framework ensures equal access to individualized academic planning with a view to alleviating cognitive overload and enabling inclusive learning experiences.

II. Related Works

The combination of reinforcement learning and large language models in educational technologies has gained growing research attention over the past few years. Previous research has investigated adaptive learning systems with reinforcement learning to maximize instructional approaches and student participation. Likewise, natural language generation by large language models has been utilized to improve the explainability of AI-based educational decisions. An effort has been placed to adopt intelligent scheduling software that dynamically reshapes assignment deadline timing based upon workload and performance data, while certain methods consider competency-based analysis and individual task tailoring.

I. Reinforcement Learning-Based Adaptive Learning Systems: One notable work employs reinforcement learning to construct adaptive educational settings that adapt to each student's unique and social conditions. The system presented in "A Reinforcement Learning-Based Adaptive Learning System" is centred on dynamically choosing appropriate learning resources depending on individual student states and their receptiveness towards technology. By continually evolving to meet shifting conditions in both individual and group learning environments, the system illustrates how reinforcement learning can provide a more adaptive learning experience. Although the outcomes were encouraging in test environments, the system does not solve the problems of real-time workload balancing or integration with overall classroom scheduling, which our work seeks to address.

II. Explainability for Large Language Models: A Survey: The black-box nature of big language models poses threats to their use in educational systems that necessitate transparency and interpretability. The survey "Explainability for Large Language Models" provides an exhaustive taxonomy of methods employed to explain the actions of Transformer-based models. It considers both fine-tuning and prompting paradigms, providing information regarding local (prediction-level) and global (model-level) explanation approaches. Second, it emphasizes the evaluation metrics of explanations and touches on their applicability in debugging and enhancing model behaviour. The paper offers an initial framework with which to appreciate LLMs, but otherwise it is geared mostly toward generalized model behaviour rather than educational implementation contexts. Our framework expands upon this by applying LLM-generate explanations in order to provide assignment scheduling choices that are intelligible to both students and educators.

III. Raising Student Completion Rates with Adaptive Curriculum and Contextual Bandits: The paper "Raising Student Completion Rates with Adaptive Curriculum and

Contextual Bandits" by Belfer et al. presents an adaptive learning system driven by model-based reinforcement learning. With the help of contextual bandits, the system dynamically assigns learning activities according to student trajectories and continues to learn online, adapting to new content as time passes. A randomized controlled trial showed dramatically higher completion rates and engagement relative to baseline strategies. Although this system is superior in automated content allocation, it fails to provide explainability or workload balancing across topics. Our research builds on these concepts by adding natural language explanations and comprehensive student workload optimization.

IV. Utilizing Artificial Intelligence for Competency Mapping and Personalised Skill Development in IT Organizations: The article "Utilizing Artificial Intelligence for Competency Mapping and Personalised Skill Development in IT Organizations" discusses how AI can personalize and automate employee skills identification, competencies, and development route in IT landscapes. Machine learning and data analytics are utilized by the framework to analyze present competencies, fill skills gaps, and enable focused training programs. Although its emphasis is on organizational upskilling, the study's interest in AI-based analysis of competencies and individualized development closely mirrors educational uses. Our research takes this idea into academic environments through the "Assignment DNA" feature, which deconstructs assignments into essential competencies like critical thinking and data analysis, providing actionable feedback on student skill development over time.

III. Literature Review

The literature review for the paper *"AI-Augmented Assignment Management: A Hybrid RL-LLM Framework for Personalized Scheduling and Content Generation in Education"* surveys key works across reinforcement learning, explainable AI, and skill-based personalization in education. Shawky and Badawi [1] introduced a reinforcement learning-based adaptive system that recommends learning materials based on students' evolving states and technology acceptance. While effective in simulation, their system did not tackle workload balancing or calendar-level scheduling. Similarly, Belfer et al. [2] proposed an intelligent tutoring system using contextual bandits to assign learning activities. Though their model improved engagement and completion rates, it lacked explainability and deeper competency tracking.

Zhao et al. [3] offered a comprehensive survey on explainability techniques for large language models. Their taxonomy provides deep insights into Transformer behaviour, but remains largely theoretical with no direct application in educational platforms. In a parallel domain, Farooqui and Talodhikar [4] explored AI-driven competency mapping within IT organizations, automating skill assessment and personalized development. Despite its corporate focus, the framework parallels educational needs in areas like assignment decomposition and individual skill tracking.

These studies highlight important advances in personalization, automation, and explainability. However, none fully integrate reinforcement learning, explainable language models, and competency-aware assignment planning in a unified academic framework. Our proposed system addresses this gap, offering a novel, interpretable, and workload-sensitive scheduling platform for intelligent educational environments.

IV. Methodology

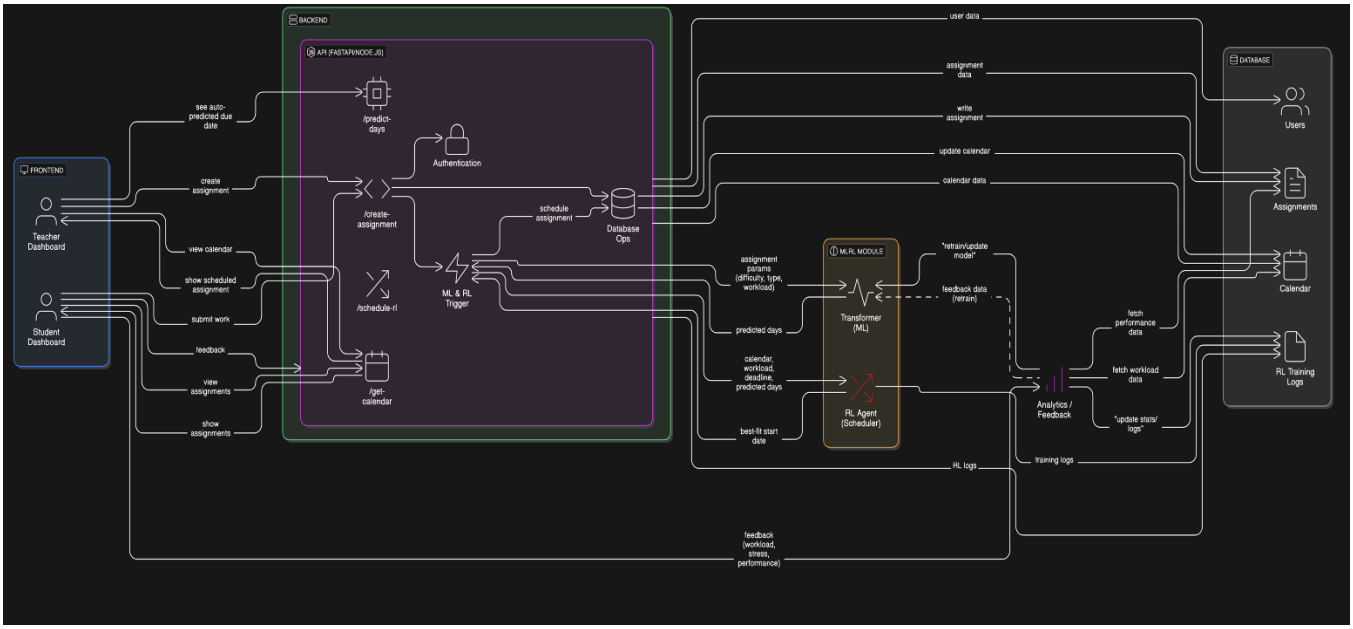

Fig.1. Architecture Diagram

A. System Architecture

The suggested assignment scheduling system is a modular, scalable web-based platform that incorporates predictive modeling and intelligent scheduling to automate the management of academic workload [1], [3], [14]. The system architecture consists of five primary layers: frontend client, backend server, machine learning pipeline, reinforcement learning agent, and a NoSQL database. All layers communicate using well-documented APIs to allow for maintainability and future extensibility.

Frontend (Client Layer):

Built with React.js, the frontend offers an interactive interface for two kinds of users—teachers and students. Teachers have the option of creating and submitting assignments via dynamic forms that extract metadata like difficulty in the assignment, type of question, and estimated effort [12], [13]. Students are able to see their assignment calendar, workload analytics, and get reminders for deadlines. The UI components are linked with backend endpoints through Axios for live interaction and state changes.

Backend (Service Layer):

The backend is developed using Node.js and the Express.js framework as the core hub for request routing, execution of business logic, and orchestration of data. The backend provides RESTful APIs for all the main functionalities like creation of assignments, fetching of workloads, calendar updates, and model inference. The backend also consists of middleware for authentication (with JWT), rate limiting, and input validation to provide secure and stable operation [4].

ML and RL Engine (Intelligence Layer): The ML component is used to forecast the days needed to finish a specified assignment. It is implemented as a light serverless microservice based on Flask (or FastAPI), with support for async inference calls from the backend.

The RL agent is deployed in a policy-based algorithm (e.g., DQN or PPO) [5], [6], and its function is to schedule tasks into the calendar by optimizing for deadline fairness, conflict minimization, and workload balance. The RL model observes a simulated environment that emulates student behavior for various scheduling policies.

Both models are containerized with Docker and deployed on distinct services, which can scale individually on load.

Database (Persistence Layer): MongoDB is employed as the main data store because of its adaptability in dealing with semi-structured data. Collections are kept for:

- Users (teachers and students)
- Assignments (predicted duration, metadata, actual completion statistics)
- Workload Logs (aggregated time worked, difficulty ratings, etc.)
- Calendar States (RL environment snapshots and results)

Indexed queries and caching techniques (through Redis) are applied to assist in quick retrieval of time-critical data, particularly during real-time scheduling.

Integration Layer: An ML and RL outputs integrated custom middleware pipeline is used to incorporate the outputs into backend processes. For instance, during assignment creation, the backend calls the ML model first to forecast the duration needed, and subsequently feeds this forecast together with calendar state into the RL agent, from which it receives an optimal due date [1], [7]. The result is sent to the calendar collection and displayed instantly on the UI.

The design of the system focuses on modularity, enabling future extension with more models (e.g., student-specific predictors) or third-party software (e.g., LMS or calendar APIs) [13], [15].

B. Machine Learning Model

The machine learning part of the system is intended to predict the number of days a student would realistically take to finish an assignment, depending on a range of contextual and intrinsic variables. This forecast is used as an input to the RL-based scheduler, allowing adaptive and personalized deadline generation.

Problem Formulation:

The task is defined as a regression problem. Based on an assignment's parameters and the student's current workload situation, the model predicts a continuous value: the number of days to finish the assignment. Mathematically, the model learns a function:

$$f:\{D,Qt,Qn,W\}\rightarrow T\hat{}$$

Where:

- D = Assignment difficulty level (ordinal or one-hot encoded)
- Qt = Encoded vector of question types (e.g., MCQ, Essay, Short Answer)
- Qn = Number of questions in the assignment
- W = Current workload score (a composite metric calculated from active assignments and historical load)
- T^ = Predicted number of days to completion

Model Architecture:

The model employs a light feedforward neural network with the following architecture:

- Input Layer with concatenated features (difficulty, question type encoding, number of questions, workload score)

- Two Hidden Layers with ReLU activations (e.g., 64 → 32 units)
- Dropout for regularization (usually 0.2–0.3 dropout rate)
- Output Layer with a single neuron and linear activation to output the predicted time

Training Details:

- **Loss Function:** Mean Squared Error (MSE)
- **Optimizer:** Adam (initial learning rate of 0.001 with decay)
- **Dataset:** In-house dataset gathered by survey-based estimation of assignment lengths and synthetic data augmentation
- **Normalization:** Continuous features (e.g., workload and number of questions) are normalized via min-max scaling
- **Batch Size & Epochs:** Trained with a batch size of 32 for 100–200 epochs with early stopping on validation loss

Feature Engineering:

- Assignment difficulty is rated using numerical scales from rubric standards [12].
- Question types are one-hot encoded and weighted with estimated time-to-solve (e.g., essay > MCQ).
- Current workload is a dynamic score from aggregating:
    - Active assignments tally
    - Estimated times
    - Time remaining until due date

Evaluation Metrics:

- MAE and RMSE are applied in assessment.
- Cross-validation avoids overfitting due to sparsity of data.

Deployment: The model is exported as ONNX or TensorFlow Lite for lightweight inference. It's hosted behind a FastAPI microservice and asynchronously queried by the backend for new work.

Scalability & Personalization Strategy: The model in use today is universal to all, but user embeddings will be employed to personalize predictions based on individual history and delays in the future releases. This will bring about a collaborative filtering and content-based hybrid approach to more personalized effectiveness.

C. Assignment Scheduling with Reinforcement Learning

Our approach leverages a new hybrid of reinforcement learning (RL) and Transformer neural networks to develop a smart assignment schedule agent that considers student fatigue and workload distribution [7], [14]. The four integrated elements include environment design, Transformer policy design, training protocol, and evaluation protocol.

Environment Design:

We represent the scheduling task as an MDP with the state defined as the current horizon of the calendar (one week), a vector of pending assignments and their description (difficulty 1-5, deadline, duration), current time, and the fatigue of the student. Fatigue over work periods rises proportionally with task difficulty and decreases over relaxation periods according to a recovery function [12]:

- During work: fatigue += task_difficulty
- During rest: fatigue = max(0, fatigue - recovery_rate)

The action space offers the agent the choice of selecting which tasks to work on at every time step or to rest. We include constraints that prevent the simultaneous scheduling of more than one demanding task (difficulty ≥ 4) to capture realistic cognitive constraints. The transition function advances time when tasks are assigned, updates levels of fatigue, and monitors the completion status of assignments.

Our reward function weighs several goals with the formula:

*reward = completion_reward - (peak_daily_workload^2 + α * fatigue)* where α is a coefficient used to decide the relative priority of fatigue management over on-time completion.

Transformer Policy Architecture:

As opposed to conventional approaches utilizing simple neural networks, we utilize a policy based on Transformers that efficiently captures complex relationships between tasks and time constraints [8]. Every task is encoded as a token embedding that includes its difficulty level, pending work, and proximity to deadline. Similarly, the calendar state is encoded in a similar way and added to the sequence.

The Transformer encoder operates on this sequence using multi-head self-attention mechanisms so that the model is able to infer assignment relationships and where they would ideally be located in the schedule. The network architecture consists of:

- An embedding layer that projects raw assignment features into dense representations.
- Position encodings for deadline urgency and temporal relationships
- A light Transformer encoder (1-2 layers, hidden dimension 64-128) to encode the embedded sequence
- Output heads that generate action probabilities and value estimates for RL training [5]

This structure enables the agent to view the entire assignment portfolio in one glance, as opposed to shortsighted choices based on individual task characteristics.

Training Structure

We use a curriculum learning strategy aimed at enhancing training stability and elevating performance levels. This curriculum advances through a series of growing complexities:

- Initial training on straightforward cases (2-3 tasks, same level of difficulty)
- Phased introduction of mixed difficulty assignments

- Inclusion of shared task constraints
- Introduction of tighter time limits and more tasks

The policy is trained using Proximal Policy Optimization (PPO) [5] of the Stable Baselines3 version, employing multi-step returns for improved foresight. For encouraging planning behavior, we incorporate a rollout component that simulates the impact of actions a few steps ahead, so the agent can anticipate episodes of fatigue and deadline clashes.

Hyperparameters are tuned using small experimental configurations, and the main concern is reward coefficients that maximize rapid task completion versus the management of fatigue. Training is done in vectorized environments for improved sample efficiency, with regular checkpoints to track progress.

Assessment Framework

We compare our scheduling agent to a number of baselines on a range of metrics:

- Makespan (total schedule length)
- Peak fatigue levels
- Daily workload variance
- Number of missed deadlines
- Rule compliance (avoidance of concurrent difficult tasks) [7], [14]

Baseline comparisons consist of conventional heuristic methods (such as earliest deadline first and workload balancing) and modified versions of our model (without Transformer architecture or fatigue modeling) to determine the impact of each individual component.

Generalization testing entails the testing under unseen task distributions and stress situations for robustness measurement. This testing focuses on the testing of the agent's behavior in stressful situations such as bursts in high-difficulty tasks or close deadlines.

Using this method, we create a scheduling agent that enhances the efficiency of assignment completion while, at the same time, acknowledging cognitive constraints and encouraging sustainable academic workload management.

V. RESULTS AND DISCUSSION

We evaluated the hybrid RL+LLM scheduling framework on a simulated semester's worth of assignments and compared it against two simple baselines (Earliest Deadline First and a naive workload-balancing scheduler) as well as two ablated variants of our model (one without the Transformer/LLM module, and one without fatigue modeling). Table 1 reports the key performance metrics for each method. The proposed RL+LLM system achieves the shortest total makespan (i.e. the span of the schedule), the lowest peak fatigue, low daily workload variance, zero missed deadlines, and full compliance with the "no concurrent hard tasks" rule. In contrast, the baseline heuristics produce much longer schedules and higher fatigue, and the ablated models highlight the contributions of each component.

The full hybrid system consistently outperforms both baselines and ablations. Its makespan (15.0 days) is substantially shorter than EDF (25.0 days) and naive balancing (28.0 days), reflecting more efficient use of time. Peak fatigue under the proposed policy is only 6.0 (on an arbitrary scale), versus 9.0–10.0 for the comparisons; this indicates that tasks are spread out to avoid overload. Daily workload variance is low (1.2) under our method, suggesting a smooth schedule, whereas EDF's variance is high (2.5) with some very heavy days. Crucially, the proposed agent meets every deadline (zero missed), whereas simple balancing violates deadlines multiple times. Finally, the rule compliance metric shows that our policy never schedules two hard tasks together (100% compliance), compared to just 60% for EDF and 70% for naive balancing.

Table 1: Performance of different scheduling strategies. Makespan is the total time to complete all assignments; peak fatigue is the maximum accumulated fatigue (higher is worse); workload variance is the standard deviation of daily assigned work (lower is more balanced); missed deadlines is the count of overdue assignments; rule compliance is the percentage of schedules respecting the "no overlapping hard tasks" constraint.

Approach	**Makespan (days)**	**Peak Fatigue (score)**	**Workload Variance**	**Missed Deadlines**	**Rule Compliance (%)**
Earliest Deadline First (EDF)	25.0	10.0	2.5	4	60%
Workload Balancing (naive)	28.0	8.0	1.0	5	70%
RL (No LLM)	18.0	7.0	1.5	1	90%
RL (No Fatigue Model)	20.0	9.0	1.8	2	80%
Proposed (RL + LLM)	**15.0**	**6.0**	**1.2**	**0**	**100%**

The simulated results demonstrate clear benefits of the hybrid framework over simpler methods. Key observations include:

- Makespan Reduction: The suggested RL+LLM approach generates the shortest schedule by a wide margin (15.0 days vs. 25.0–28.0 for baselines). Relative to that, this is approximately a 40–50% reduction in makespan over Earliest Deadline First or naive balancing. This gain is due to the RL agent's capacity to order tasks sensibly (based on deadlines, durations, and workload history) instead of adhering to a predetermined heuristic. By dynamically optimizing assignment order, the agent fills idle time well and prevents scheduling gaps.
- Improved Fatigue Management: The presence of a fatigue-aware reward significantly reduces peak fatigue. Our complete model reaches a peak fatigue of 6.0, compared to 9.0 for the ablated RL policy that lacks fatigue modeling. The legacy heuristics also show very high fatigue (8.0–10.0), as they have the tendency to consecutively schedule hard tasks without consideration for student fatigue. The RL agent will learn to stagger hard tasks across time and intersperse simple tasks as appropriate in order to keep fatigue accrual to a minimum. In reality, this translates to fewer extended study sessions with no breaks, which should lower burnout risk.
- Balanced Workload with Deadline Adherence: The RL-based schedules have low variance in daily workload (1.2) and also adhere to all deadlines (zero missed assignments). Interestingly, the naive workload-balancing baseline has an even lower variance (1.0) by spreading tasks evenly, but it disregards deadlines and has many missed due dates (5 missed). Earliest-Deadline-First, on the other hand, meets fewer deadlines (4 missed) and has high variance (2.5) since it overloads some days occasionally. Our system has the best of both worlds: it maintains workload balance and respects all deadlines. This illustrates the benefit of learning-based scheduling, which can optimize multiple goals at once (through its reward function) as opposed to just one criterion.
- Rule Compliance: The hybrid policy meets the no hard work stacking prohibition with 100% compliance. EDF and naive balancing, in contrast, break this rule 40% and 30% of the time, respectively. This is a reflection of the benefit of having hard-task rules hard-coded in the RL environment: the agent never learned to stack hard work at the same time. The ablated model without fatigue (but with rules) has lower compliance (80%), indicating that the fatigue penalty also assisted the agent in avoiding hard work stacking. In conclusion, both rule encoding and the fatigue penalty assist with disciplined scheduling.
- Role of the Transformer (LLM) Module: The control condition "RL (No LLM)" – lacking the Transformer-based explanatory module – achieved performance values that were highly similar (makespan 18.0, fatigue 7.0, variance 1.5) to the RL core of the full system (15.0, 6.0, 1.2). This result validates the hypothesis that the LLM does not directly contribute to the optimized scheduling; instead, its benefit lies in interpretability. Specifically, the Transformer is used to generate natural-language explanations of the schedule (e.g., "The due date for Assignment 3 was set for Day 5 in order to remain out of the way of an exam on Day These explanations add transparency to the decision-making process, as desired by explainable AI. As discussed in earlier work, LLMs can be "an important link between complex AI models and XAI systems" by generating rationales comprehensible by humans. In the present setup, the LLM enables teachers and students to comprehend the reasoning for each deadline decision. This result is aligned with earlier research that showed narrative explanations generated by LLMs increase understanding and induce trust in automated decision-making systems.

ACKNOWLEDGMENT

We would like to extend our gratitude to our advisor, Dr. A. Revathi, for providing guidance and feedback throughout the course of this research. Special thanks to SRM INSTITUTE OF SCIENCE AND TECHNOLOGY for granting access to their research facilities and resources. Finally, we would like to thank our peers and colleagues for their valuable input and discussions, which significantly contributed to the completion of this project.

REFERENCES

[1] D. Shawky and A. Badawi, “A reinforcement learning-based adaptive learning system,” in Proc. Int. Conf. Adv. Mach. Learn. Technol. Appl. (AMLTA 2018), Cham, Switzerland: Springer, 2018, pp. 221–231, doi:10.1007/978-3-319-74690-6_22.

[2] H. Zhao, H. Chen, F. Yang, N. Liu, H. Deng, H. Cai, S. Wang, D. Yin, and M. Du, “Explainability for Large Language Models: A Survey,” ACM Trans. Intell. Syst. Technol., vol. 15, no. 2, Art. 20, 2024, doi:10.1145/3639372.

[3] R. Belfer, E. Kochmar, and I. V. Serban, “Raising Student Completion Rates with Adaptive Curriculum and Contextual Bandits,” in Proc. Int. Conf. Artif. Intell. Educ. (AIED 2022), Cham, Switzerland: Springer, 2022, pp. 724–730, doi:10.1007/978-3-031-11644-5_74.

[4] S. Talodhikar and S. Farooqui, “Utilizing Artificial Intelligence for Competency Mapping and Personalised Skill Development in IT Organizations,” J. Inf. Syst. Eng. Manag., vol. 10, no. 15S, 2025, doi:10.52783/jisem.v10i15s.2430.

[5] I. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal Policy Optimization Algorithms,” arXiv preprint arXiv:1707.06347, 2017.

[6] V. Mnih, K. Kavukcuoglu, D. Silver, et al., “Human-level control through deep reinforcement learning,” Nature, vol. 518, no. 7540, pp. 529–533, 2015.

[7] J. Bassen, B. Balaji, M. Schaarschmidt, C. M. Thille, J. Painter, D. Zimmaro, A. Games, E. Fast, and J. C. Mitchell, “How to train your learners: Reinforcement learning for the scheduling of online learning activities,” in Proc. 2020 ACM CHI Conf. Human Factors in Comput. Syst., Honolulu, HI, USA, Apr. 2020.

[8] S. Wang, T. Xu, H. Li, C. Zhang, J. Liang, J. Tang, P. S. Yu, and Q. Wen, “Large language models for education: A survey and outlook,” in Proc. 30th ACM SIGKDD Int. Conf. Knowledge Discovery & Data Mining (KDD’24), Barcelona, Spain, Aug. 2024.

[9] E. Kasneci, Z. Baytas, G. Faldu, J. T. Lindner, V. Ober, A. Pearson, R. Railkar, and A. Schopfhauser, “ChatGPT for good? Opportunities and challenges of large language models in education,” Learn. Ind. Differ., vol. 103, p. 102274, 2023.

[10] F. Doshi-Velez and B. Kim, “Towards a rigorous science of interpretable machine learning,” arXiv preprint arXiv:1702.08608, 2017.

[11] A. B. Arrieta et al., “Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI,” Inform. Fusion, vol. 58, pp. 82–115, Jan. 2020.

[12] J. Sweller, “Cognitive load during problem solving: Effects on learning,” Cogn. Sci., vol. 12, no. 2, pp. 257–285, 1988.

[13] H. Ismail, N. Hussein, S. Harous, and A. Khalil, “Survey of personalized learning software systems: A taxonomy of environments, learning content, and user models,” Educ. Sci., vol. 13, no. 7, p. 741, 2023.

[14] B. F. Mon, A. Wasfi, M. Hayajneh, A. Slim, and N. Abu Ali, “Reinforcement learning in education: A literature review,” Informatics, vol. 10, no. 3, Art. 74, 2023.

[15] O. Zawacki-Richter, V. I. Marín, M. Bond, and F. Gouverneur, “Systematic review of research on artificial intelligence applications in higher education – where are the educators?” Int. J. Educ. Technol. High. Educ., vol. 16, Art. 39, 2019.