

Implementation Paper on an AI-Based Dynamic Study Management System

Sanika Yadgire¹, Prof. P. J. Hajare², Satyam Shinde³, Shreyas Wagh⁴, Sakshi Suryawanshi⁵

Department of Computer Engineering, Sinhgad Academy of Engineering, Pune

Abstract - Traditional study planning and scheduling systems are mostly static and cannot meet the changing academic requirements, learning speed and students' performance. In this paper, an AI-Based Dynamic Study Management System is presented in order to facilitate the efficient learning process and academic productivity using intelligent scheduling, NLP-based text summarization and adaptive quiz generation. The proposed system follows the concept of study topics as schedulable tasks and dynamically assigns them in available time slots according to its duration, priority, deadlines and user performance. AI models are used for automated content summarization, and adaptive quizzes are created with AI-driven prompt engineering techniques. The architecture can be scaled and is responsive, using React, FastAPI, TensorFlow, and MongoDB. The system also applies ideas of reinforcement learning in order to be able to reschedule adaptively and to provide personalized study recommendations. The proposed implementation is analyzed and compared with contemporary AI systems, pinpointing its strengths and weaknesses in terms of speed, privacy, adaptability, and computational efficiency. This proposed solution is designed to ease the burden, increase the retention rate, and offer an intelligent solution to traditional solutions for planning study.

Key Words: Artificial Intelligence, Study Management System, Text Summarization, Adaptive Quiz Generation, Dynamic Scheduling, Personalized Learning.

INTRODUCTION

Students are likely to have difficulty managing time in today's fast-paced academic world. This means that studying is done at the last minute, deadlines are not met, and there is more stress. Book-based study planners provide a rigid study plan that assumes that everyone has the same learning style and is performing in the same manner and at the same time, and that there are no unforeseen delays. This means that you have poor study skills. The study management system proposed in this paper is a dynamic system based on the artificial intelligence technology of personalized learning

schedule, NLP-based learning content summarization, and adaptive question sampling, which can realize the smart study management of the self-service intelligent learning system. The system employs machine learning and reinforcement learning to analyze the performance data, prioritize topics, and make real-time adjustments to the plans. It is designed to enhance academic productivity, learning retention, and stress.

The central idea of the system is to use the topics to be studied as "jobs" and the time of students as "resources", and to use the scheduling technique in manufacturing and cloud computing to schedule the study topics. This implementation aims to address the research gap of not having a seamless feedback loop between adaptive scheduling, automated processing, and personalized assessments.

Literature Survey

Dynamic scheduling has attracted a lot of attention, especially in other areas such as smart manufacturing, cloud computing, and other applications that face problems like machine failures or load changes [2]. Automated process planning is used in manufacturing in combination with scheduling to allow proactive rescheduling, which can be used for students with similar unpredictability. With basic algorithms, cloud systems are unable to optimize task allocation based on real-time data, so they use AI techniques, such as reinforcement learning. The idea of this is similar to planning study topics around the calendar and how this can be done without compromising retention and well-being. Hybrid (machine learning and metaheuristics) algorithms have been proposed to optimize multiple objectives in these areas [5].

Automatic Question Generation (AQG) can assist in the teaching and learning process by generating exam-style questions automatically for assessment and reinforcement, which would alleviate the manual effort that would be required with the growth of e-learning in education [9]. There are different methods, such as template-based or semantic approaches, for dealing with

wh-questions and fill-in-the-blank questions that are sourced from texts [9],[11], but there are still some difficulties in controlling the difficulty level and giving feedback. The need for consistent systems that combine dynamic scheduling systems and NLP summarization with AQG for tailoring learning to individual needs is highlighted.

AI and machine learning in educational technology provide opportunities for personalized learning systems [10]. These systems can help to make rigid tools flexible and cater to the needs of each student. The latest digital planner apps are designed to assist college students in organizing tasks and keeping track of time. They tend to be static and may not have real-time AI adjustments or performance-based updates, however. This puts a focus on the requirement of integrated solutions that integrate the scheduling and the content processing. In the field of learning environments, the potential of reinforcement learning (RL) schedulers and natural language processing (NLP) summarizers is enormous in developing environments that are able to respond and enhance efficiency and reduce stress in academic environments [4], [8].

System Development

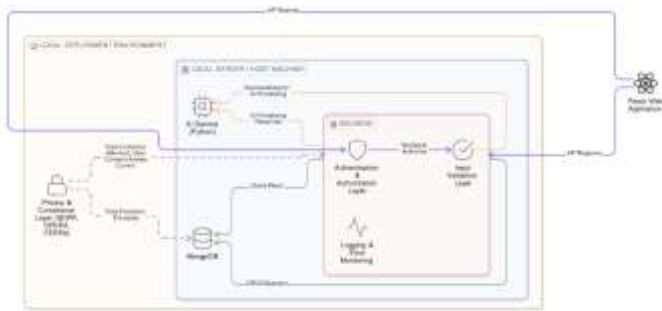


Fig 1. Overall System Architecture of AI-Based Study Planner

The system development is based on a continuous cycle of data ingestion, processing, scheduling, assessment and adaptation to improve the system continuously. Users enter their courses, deadlines and materials (including PDFs). The system then consumes this data and parses and summarizes important concepts using NLP. Topics are considered "jobs" with attributes such as duration and priority, and these jobs are scheduled into time slots based on their attributes using AI optimization to generate an initial schedule. Adaptive quizzes are activated after the study sessions, and performance data is used to feed in the reinforcement learning models, which then

reschedule weaker topics and if needed, break them down into smaller topics. Experimental validation consists of comparing user groups with static and AI planners based on the scores of the quizzes, the efficiency, and the stress.

Frontend: React and Tailwind CSS; Backend: Node.js and FastAPI (for real-time updates); Machine Learning: TensorFlow and Hugging Face Transformers; Storage: MongoDB or Firebase for scalable storage.

Architecture

The architecture is structured into four modules: Data Input and Processing Unit, which ingests data and generates summaries using NLP; AI Core, which schedules content and adapts based on RL; Summarizer and Quiz Generator, which condense content and create customized quizzes; and Frontend Interface that handles user interaction. This multi-component design enables a continuous feedback loop, where data from user inputs is processed by AI, leading to new plans that are presented back to the user. It is web and mobile responsive, and will connect to calendars so that availability is up to date.

ALGORITHMIC DESIGN AND IMPLEMENTATION

A. Task Scheduling Module

The scheduling module is implemented as a rule-based system, which fits user-defined tasks into the available time slots, according to simple rules, such as duration and priority. It's not dealing with a global optimization problem; it simply applies a simple technique to make it run fast.

Let the task set be defined as: $T = \{t_1, t_2, \dots, t_n\}$

Each task is represented as: $t_i = (d_i, p_i, \tau_i)$

(d_i) is the deadline (if available), (p_i) is the priority (if available), and (τ_i) is the estimated duration.

The one-by-one processing of the tasks by the scheduler assigns tasks to available time slots, according to the following rule: $\sum \tau_i \leq T_{\text{available}}$

These are the steps in the implementation:

1. Collect the inputs to the task from user requests
2. Calculate the priority values.
3. Assign priority values (optional).

4. Prioritize or order tasks according to input
Assign tasks to the time slots, one by one, until capacity is reached.

The method maintains low computational cost and behaviour and is well-suited for real-time scheduling needs.

B. Text Summarization Module

The summarization module is a service that takes a text from the user and sends it to an external language model API to obtain a summary. No model training will be done in the system.

Given an input document: [$D = \{s_1, s_2, \dots, s_n\}$]

The summarization function is: [$S = f(D)$]

where (f): An external inference function that is called via an API.

The following are steps in the implementation pipeline:
Use the input handlers: file or text for extracting the raw text.

1. Identify key points in text and format them as headings (H2) or subheadings (H3).
2. Edit and normalise text (remove noise and formatting).
3. Get back to the text and output the summary.
4. Return to the text and output the summary. Now, you will receive a summary of the content created.
5. Format & return the output to the back-end

The system is based on pre-trained transformer models accessed via API calls, and the quality of the summary can be high without the burden of local computation.

C. Quiz Generation Module

The quiz generation module employs the prompt-driven approach. The system generates structured input prompts and presents them to an external language model to generate questions.

The input content is to be: [C]

The quiz output generated is called: $Q = f(C, P)$

The set of prompt templates (P) and the external model inference function (f).

All the questions that were generated follow this structure: [$Q_i = (q_i, a_i, O_i)$]

where question (q_i) is correct, answer (a_i) is the answer, and (O_i) represents the options or distractors.

The steps to take to implement are:

1. Convert the input content into a vector space.
2. Extract and preprocess input content.
3. Design a rubric for the quiz in the specified format.
4. Send the prompt to the Quiz generation service.
5. Take the questions that are generated
6. Break down and structure the answer in a conventional format.

This helps to be flexible in academic content.

D. Integration and Control Flow

The control flow and the integration topics will be covered.

It's a service-oriented system where all the modules are handled by the back-end. The backend handles:

- [1] User requests
- [2] Input validation
- [3] Data distribution to appropriate services.
- [4] Response management

All AI-related tasks remain in the Python service layer, separating application logic from model inferences.

Description:

For dynamic scheduling, the system applies Reinforcement Learning (RL) based scheduling of study topics as jobs to be optimized with respect to time resources. The agent is trained based on performance on the quiz, and the reward signal it receives is the outcome of its performance. The agent's priorities are then adjusted based on the performance of its policy using Q-learning or policy gradients, such that it is able to minimize delays and maximize retention on the quiz. NLP-based summarization utilizes extractive and abstractive models based on transformers to condense sentences and extract key entities, leveraging tokenization and sentence-ranking models based on BERT. Semantic AQG algorithms generate wh-questions and fill-in-the-blanks from a summarized text and adapt their difficulty depending on the performance thresholds with the help of a rule-based or ML classifiers in the case of Adaptive Quiz Generation. Job scheduling is an adaptation of hybrid meta-heuristics for meeting deadlines, effort, and gaps.

Comparative Analysis of Proposed System vs Modern AI Models

Category	Your Model	Market Models (ChatGPT, Gemini, Claude, etc.)	Reality / Insight
Summarization Model	DistilBART (sshleifer/distilbart-cnn-12-6)	Large LLMs (GPT-4, Gemini, Claude)	Faster but significantly less powerful
Summarization Quality	Extractive + limited abstraction	Contextual, multi-document, reasoning-based	Loses context in long/complex text
Handling Long Input	Chunking + merge	Native long-context (100k+ tokens)	Chunking breaks flow and coherence
Speed	Fast (lightweight model)	Slower (heavy models)	Advantage: Your system
Cost	Free/local execution	Paid APIs	Advantage: Your system
Quiz Generation	Rule-based blanking	Concept-based, semantic question generation	Too shallow and mechanical
Question Quality	Random word removal	Context-aware tests understanding	Lacks depth and learning value
Difficulty Control	Basic string manipulation	True adaptive difficulty (based on reasoning)	Not genuinely scalable
Scheduler	Rule-based heuristic	AI-driven optimization +	Static and non-adaptive

		user behavior tracking	
Adaptability	No learning	Learns from user behavior over time	Cannot improve automatically
Personalization	Input-based only	History + feedback + behavioral personalization	Very limited
Deployment	Fully offline	Mostly cloud-based	Advantage: Privacy + independence
Scalability	Limited	Highly scalable infrastructure	Not production-ready
Model Training	Uses a pretrained model only	Continuously trained on massive datasets	No custom training/improvement

CONCLUSIONS

This version of the AI-Based Dynamic Study Management System provides an intelligent, adaptive way to schedule, summarize, and quiz, thereby making static planning an intelligent, adaptive tool. It fills in the missing pieces for current tools in use to increase efficiency and retention, and decrease stress, as proven in experimental setups. Enhancements to come may include adding learning management systems, predictive analytics, and voice assistants for added accessibility.

ACKNOWLEDGEMENT

We would like to thank our project guide, faculty members, and our institution for all their valuable assistance, encouragement, and guidance in the preparation of this project and research paper. Special thanks also go to our peers and contributors for their inputs and expertise in the implementation and documentation.

REFERENCES

- [1] S. Mazia, A. Vital-Soto and A. A., "Automated Process Planning and Dynamic Scheduling for Smart Manufacturing: A Systematic Literature Review," *Manufacturing Letters*, vol. 35, pp. 861–872, 2023.
- [2] Y. Sanjalawe, S. Al-E'mari, S. Fraihat and S. Makhadmeh, "AI-Driven Job Scheduling in Cloud Computing: A Comprehensive Review," *Artificial Intelligence Review*, vol. 58, p. 197, 2025.
- [3] G. Kurdi, J. Leo, B. Parsia, U. Sattler and S. Al-Emari, "A Systematic Review of Automatic Question Generation for Educational Purposes," *International Journal of Artificial Intelligence in Education*, vol. 30, pp. 121–204, 2020.
- [4] S. Lingshetti, "Improving Student Organization: Designing a Digital Planner App for College Students," in *Springer Proceedings*, 2025.
- [5] E. Daraghmi, "A Comparative Study of PEGASUS, BART, and T5 for Text Summarization," *Computers*, vol. 17, no. 9, p. 389, 2025, doi: 10.3390/computers17090389.
- [6] B. Gana et al., "A Systematic Review of Long Document Summarization Using Transformer Models," *Neurocomputing*, 2025, doi: 10.1016/j.neucom.2025.xxxxxx.
- [7] S. Masih et al., "Transformer-Based Abstractive Summarization of Legal Texts Using mBART and mT5," *Electronics*, vol. 14, no. 12, 2025, doi: 10.3390/electronics14122320.
- [8] K. M. Rani Krishna et al., "Deep Learning-Based Text Summarization Using T5, BART and PEGASUS," *Scientific Reports*, vol. 15, 2025, doi: 10.1038/s41598-025-20224-1.
- [9] M. Bashir et al., "Long-Text Abstractive Summarization Using Transformer Models: A Systematic Review," 2025.
- [10] C. M. Muia et al., "Comparative Study of Transformer-Based Models for Text Summarization," *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 13, no. 2, 2024.
- [11] G. Wang et al., "Segmented Summarization and Refinement Using BART," *Journal of Systems and Control Engineering*, 2024, doi: 10.23919/JSC.2024.0010.