

# A Quantum-Inspired Adaptive AI Tutor for Personalized Learning: A Quiz-Driven Knowledge Framework

T Manju<sup>1\*</sup>, P NaveenKumar <sup>1</sup>, S Vinothan <sup>1</sup>, and A Prashant <sup>1</sup>

<sup>1</sup>Department of Information Technology, Sri Sai Ram Engineering College, Chennai, Tamil Nadu, India

**Abstract.** Adaptive learning systems aims to prepare materials based on user needs, but many of them are still depend on strict rules and fixed content. To overcome the limitations of traditional online tutoring system, our proposed work introduces a Quantum-inspired tutor system. The purpose of this proposed system is to update each learner's pace through quiz driven updates and its dynamically generated content. Learner understanding is modeled as a probability distribution over knowledge levels-conceptually similar to a quantum state. After each quiz, this distribution is updated, and several teaching strategies such as hints, examples, and explanations, are evaluated in parallel. One of them is selected using probabilistic sampling, echoing quantum-style collapse. This helps the system to balance exploration with targeted feedback. Multilingual contents, learner profiles and modular deployment are included in our work. Our work offers a scalable and interpretable foundation for intelligent tutoring system in software and programming education for learners.

## 1 Introduction

In technical areas like a programming educational environment needs learning system based on their needs. Individuals are different in skill levels and learning speeds. The traditional tutor system i.e., one-size-fits-all model is not fit for most of the beginners. This problem creates a high influence on the students who needs to study the programming courses. Various reports stated that personalized feedback and guidance can meaningfully improve learning outcomes. However, scaling such support across large groups remains a challenge. As a result, digital learning platforms are now being developed to respond to each learner's progress and preferences. In practical terms, this involves dynamic programming assignments, content, and feedback in response to continuous evaluation and student behavior.

Artificial intelligence (AI) offers new tools for personalized learning. Today's adaptive platforms can modify content to suit different learning style and paces. On the other hand, AI-based analytics helps to anticipate where students may need support.

---

\*Corresponding author: [manju.it@sairam.edu.in](mailto:manju.it@sairam.edu.in)

A key area of development is Intelligent Tutoring Systems (ITS), which are designed to mimic one-on-one instruction by continuously monitoring the student progress and adjusting lesson flow according to it. Reviews of ITS research indicates that these systems often provide positive learning outcomes over the traditional approaches. Furthermore, large language models (LLMs) like Google's Gemini and OpenAI's ChatGPT, which can generate human-like text and explanations on demand, have been made possible by the development of generative AI. LLMs have been used to create tutoring systems in programming education that provide real-time feedback, tips, and sample code. One example is ITS-CAL by Lai and Lin, which offers layered feedback and advice for students learning to code. Such systems suggest that LLMs can supply rich, on-the-fly instructional content without requiring hand-coded lessons.

However, current AI tutoring approaches often lack deeper instructional planning. Most LLM-based tutors handle short-term tasks – answering questions or giving hints – but do not plan multi-step strategies or long-term personalization. Similarly, traditional ITS can adapt at certain decision points, but typically rely on fixed rules or simple branching logic and cannot easily incorporate dynamic, open-ended content. In this work, we propose a framework that marries these strengths: we use LLMs to generate educational content and interactive quizzes, while using an adaptive learner model to choose among teaching strategies. Our model is “quantum-inspired”: we treat the learner’s state as a probabilistic superposition of mastery levels and select actions by sampling that distribution.

While based on classical computers, this system is guided by quantum concepts (e.g., parallel evaluation and probabilistic collapse) and adapts similarly. The end outcome is that each student receives a quiz-driven tutoring system that adapts the content presented to them, as well as their tutor's strategy for them.

This paper is structured in following ways: Section 2 includes Literature survey; Section 3 introduces our system’s architecture and the core logic behind its adaptation model. In Section 4, describes our proposed system’s workflow, Section 5 outlines the advantages of this approach, Section 6 discusses about the current limitations and possible future improvements and finally Section 7 concludes the paper.

## **2 Literature survey**

Huang et al., developed SP-TeachLLM[2], a modular tutoring assistant that integrates the cognitive learning strategies into adaptive programming lessons. This study found that combining the LLM reasoning with instructional planning leads to better problem-solving outcomes than static prompting alone. Lai and Lin employ an LLM model named ITS-CAL delivers the layered hints for the coding tasks. The use of hint generation improves the student success rates, emphasizing the benefit of adaptive feedback over repetition. Another notable tutoring system by Liu et al., LPITutor combines retrieval-augmented generation (RAG) with learner-specific prompts to craft individualized responses. LLM-based tutors are best at generating immediate content and feedback. However, it may still lack in structured decision frameworks for long-term adaptation.

Tang (2019) stated that using  $\ell^2$ -norm sampling in recommendation methods can effectively helps to simulate the probabilistic behavior observed in quantum states. He demonstrated the classical probability distributions which could replicate quantum-like behavior in variety of tasks. Arrazola et al [3]., extended this idea to develop quantum-inspired algorithms for solving large linear systems and recommendation problems, showing practical speedups under certain assumptions. Although these approaches are well-established in computational domains, their role in educational systems remains very limited.

Some studies have proposed using quantum-style models to manage uncertainty in student modeling. However, to the best of our knowledge there is no existing intelligent tutoring system applies quantum-inspired decision-making in a direct way. The framework presented in this paper addresses that gap representing students using probabilistic student states and sampling-based on strategy selection, providing a quantum-inspired analog to classical adaptation methods.

## 3 Proposed system

### 3.1 System architecture

Our work is implemented as a modular web application with the four primary components, shown in Fig 1. Firstly, at the front end, a single-page application is built with React.js which manages the user interface for learners. It delivers the lessons for us, presents quizzes to solve and displays feedback at end. This interface communicates with a backend server and it is built using Java Spring Boot through RESTful APIs. The backend service actually handles the main learning workflow of our project. It maintains the learner's user profiles, processes the quiz submissions details, updates the learner state and also it coordinates the requests to the AI engine. The persistent data including user credentials details, question-answer logs, proficiency metrics and the generated content are stored by PostgreSQL database.

The Python microservice handles the core AI operations of the systems. This component integrates with Google's Gemini Pro API (or a compatible LLM) to generate instructional content and analyze the student inputs. When a new quiz item or explanation is needed, the backend issues a request to the Python service, which prepares a prompt, invokes the LLM and processes the result. In addition to this content generation, the microservice also handles learner model updates and selects the next instructional for action. By keeping this logic as a separate service, LLM layer allows to scale or evolve independently.

After a learner completes a quiz, the backend records responses, updates the learner model, and it also optionally calls the AI service to generate the follow-up materials such as hints, explanations or new problems. The resulting content is stored and return back to the learner through the user interface.

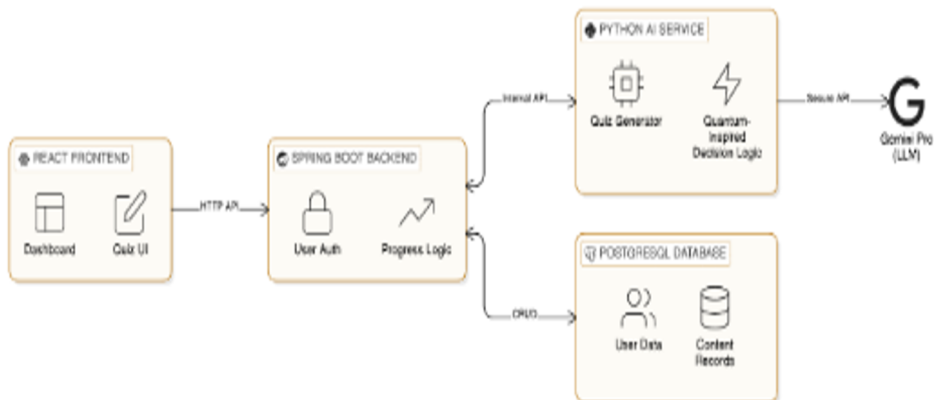


Fig. 1. Overall system architecture of the proposed System

### 3.2 Quantum-inspired adaptive model

A novel aspect of our system is the quantum-inspired adaptive learner model. This paper represents a student's understanding of each concept as a probability vector (analogous to a quantum state) over discrete knowledge levels. For simplicity, consider a vector

$$p = [p_0, p_1, \dots, p_N]$$

where  $p_k$  is the probability that the learner has achieved level  $k$  on a given topic.

By a diagnostic pre-test, the learner's state distribution was identified. It can be uniform or informed. After each quiz, this distribution ( $p$ ) is adjusted using Bayesian updates or heuristic rules, depending on how the learner's response affects our estimate of their understanding. This process parallels how quantum systems evolve through transformations on the state vectors.

At the same time, the system considers a many instructional strategies such as types of hints, examples and explanation styles. Each strategy is assigned an expected outcome based on how well it aligns with the current learner state. From quantum-inspired methods, the system evaluates several candidates parallelly and they select one by sampling. Instead of commitment to single best option, they are using weights derived from their estimated effectiveness they perform. Probabilistic selection is used by them to maintain flexibility. This idea of our project mirrors the idea of quantum collapse, a system in superposition narrows to one outcome based on probabilities.

Following each interaction, the learner's updated response is used to revise ( $p$ ) that once again deliver a question or hint. A correct answer gradually increases the confidence in higher proficiency states. But an incorrect one shifts the belief into downwards. This step functions updating the state to reflect new evidence of the learner progress. The loop then continues and the strategies are re-evaluated completely. Over time, this loop like cycle made the tutor to adjust content and the instructional path itself, using soft, data-driven adaptation.

## 4 System workflow

When a student begins a lesson, they first log in and select a topic. In that topic they will receive an assignment to know the understanding level about the topic. Based on the score, the backend initializes the learner model for that student. The student is then presented with an **LLM-generated learning module** (e.g. a short explanation, example, or video snippet) tailored to their initial state. Following this, the system generates a **quiz question** using the Python AI service. This question is drawn from a repository or generated via the LLM, chosen to probe the student's understanding at an appropriate level.

When the student submits an answer to the quiz, the React frontend sends the response to the Spring Boot backend. The backend uses pre-defined rules (or a Bayesian update) to adjust the student's probability distribution over knowledge levels. For instance, a correct answer may increase the probability that the student is in a higher mastery state, whereas an incorrect answer decreases it. Concurrently, the backend computes the weighted probability of several next steps. These weights are based on the updated learner state and pedagogical heuristics. The system then **samples** a next strategy according to these weights. If the learner appears weak on a concept, the system might with high probability pick a remedial explanation.

If the chosen action requires new content, the backend issues an API call to the Python service. In that service, prompts are constructed that include the student's performance and the educational context. The Gemini Pro LLM then generates the requested content. when the Python service processes the LLM response, it formats the content based on that response by LLM and sends it back to the backend. A hint, explanation, and the quiz item are stored by The PostgreSQL database for logging or reuse. The contents are forwarded

from backend to the frontend. The React interface displays the forwarded content to the learner. Here the single iteration of the loop is completed. When the learner interacts with the newly presented material then the system starts initiating the next cycle based on their response.

The system captures timestamps, user responses and generated content throughout this process. These are logged to support both real-time adaptation and the future analysis because the learner model maintains the state across all the previous interactions where each new quiz is shaped by the full history of the learner's activity and the instructors also have oversight. The created modular design allows for flexible deployment services, that can run independently on cloud infrastructure and multiple LLM instances can support the larger user bases. This design integrates well with the real-time assessment with adaptive content delivery which offers flexibility while maintaining the classical implementation.

## 5 Discussion

The proposed quiz in the tutoring system offers the several notable advantages like the ability to generate human like instructional content without any manual creation. In real world existing work, such as ITS-CAL and SP-TeachLLM, shows that LLMs are effective tools for structuring the adaptive programming instruction. Beyond the content generation, the quantum inspired the learner model which adds flexibility to the system's adaptation strategy and also it maintains a probabilistic view of the learner proficiency. This also allows the tutor to adjust dynamically by choosing between easier or more challenging questions to better assessing the student's current level of progress. Such behavior resembles how quantum inspired algorithms explore the solution spaces through probabilistic sampling.

This approach supports both the exploration and exploitation unlike traditional ITS. It may select non-optimal strategies to uncover deeper misunderstandings or to test the boundaries of learner readiness. Over time, this can lead to stronger personalization and more robust learning. It keeps the adaptation process transparent and easier to interpret using a well-defined probabilistic model. One can inspect the student's probability distribution and reasoning weights to explain why a particular hint or question was chosen, which is harder in black-box LLM approaches. Our hybrid design draws the strengths of recent LLM driven while adding an explicit learner model which is inspired by quantum theory to guide decisions.

It is important to note that despite the "quantum" terminology, the implementation runs on classical computing. The quantum-inspired aspects serve as an analogy or metaphor to motivate the design. As Tang observed,  $\ell^2$ -norm sampling in classical space can replicate the effect of quantum superposition in certain algorithms. Similarly, we use classical probability distributions and sampling instead of actual quantum hardware. This makes the system feasible today. It can be prototyped on standard servers using existing AI APIs. In this way, the work can be seen as a first step toward quantum-aware education technology – incorporating ideas from quantum computing research into education without requiring quantum computers.

## 6 Limitations and future work

Our framework is the design that has not yet been fully implemented or empirically tested. In quantum-inspired model, the promising, relies on manually chosen update and sampling rules. In practice, these rules would need to be calibrated carefully. Right now, we assume simple update (e.g. increase/decrease probabilities by fixed amounts), but future work should explore more on data-driven methods.

Another concern is the system that reliance on LLMs which may produce inaccurate, irrelevant, or biased outputs. For real-world use, before introducing the content to students, it will be essential to introduce validation steps either on automated filters or optional instructor review. Bias mitigation generates hints and feedback also warrants attention, if the system is deployed at scale.

The next step is building the working prototype. It is using the proposed architecture stack React, Spring Boot, Python services, and a cloud-based LLM like Gemini Pro or GPT-4 which is applied to a focused domain like introductory/beginner programming. Data which are collected from the student interactions could be used to completely refine the learner model and the data also evaluate their performance in a classroom setting. Comparative trials for learner would help to determine whether this adaptive whereas the quiz-driven approach produces measurable gains over conventional ITS or static e-learning systems.

In the future, the system could be improved by exploring the hybrid extensions such as simulating quantum algorithms to help choose more effective teaching strategy. Additionally, recent findings from dequantization research could further improve the underlying decision logic used by the system. Another possible enhancement is the use of incorporating multi-modal inputs such as keystroke patterns or eye-tracking which could provide deeper knowledge into learner engagement. This information would allow better understand learners and offer more personalized instruction.

## 7 Conclusion

This paper presented a quantum-inspired AI-powered personalized learning tutor for programming education. Motivated by the limitations of one-size-fits-all teaching and the growing complexity of learner-centric systems, the proposed framework adopts ideas from Quantum Artificial Intelligence— such as parallel evaluation of possibilities and probabilistic state transitions—to design a richer adaptive decision model for tutoring.

The tutor represents learner understanding as a probability distribution over multiple pedagogical states and evaluates several instructional strategies in parallel before selecting one based on quiz performance, time taken, and interaction history. Large Language Models are used to generate the actual instructional content once the strategy is chosen, enabling scalable and flexible personalization on classical hardware.

Although the system does not use real quantum computation, its design is inspired by quantum decision paradigms and fits naturally within the broader vision of Quantum AI and Knowledge Systems. This work offers a feasible direction for educational institutions to experiment with quantum-themed innovations by connecting QAI concepts with practical LLM-based tutoring.

## References

- [1] R. Baltezarević, Introducing Quantum Artificial Intelligence (QAI) in Education, (2025).
- [2] S. Huang, Y. Sun, X. Yu, SP-TeachLLM: An LLM-Driven Framework for Personalized and Adaptive Programming Education, *Inf.* **16**, 1045 (2025).  
<https://doi.org/10.3390/info16121045>
- [3] J.M. Arrazola, et al., Quantum-Inspired Algorithms in Practice, *Quantum* **4**, 307 (2020).  
<https://doi.org/10.22331/q-2020-08-13-307>
- [4] E. Tang, A Quantum-Inspired Classical Algorithm for Recommendation Systems, *Proc. ACM Symp. Theory Comput.* (2018).
- [5] A. Rahman, et al., Integrating Quantum AI, Gamification, and Adaptive Learning in Higher Education, *Int. J. Emerg. Technol. Learn.*, (2025).

- [6] G. Pilato, et al., A Survey on Quantum Computing for Recommendation Systems, *Inf.* **13**, 567 (2022).
- [7] V. Dinesh, S. Paramesh, AI and Quantum Technology in Educational Research, *Sch. Res. J. Interdiscip. Stud.* **13**, 86, 1074–1079 (2025).
- [8] C.-H. Lai, et al., ITS-CAL: An Intelligent Tutoring System for Coding and Algorithm Learning Powered by an LLM, *Appl. Sci.* **15**, 1922 (2025).  
<https://doi.org/10.3390/app15041922>
- [9] J.M. Reddig, et al., Generating In-Context Personalized Feedback for Algebra with GPT-4 in an Intelligent Tutoring System, *Int. J. Artif. Intell. Educ.* (2025).
- [10] Z. Liu, P. Agrawal, S. Singhal, V. Madaan, M. Kumar, P.K. Verma, LPITutor: An LLM-based Personalized Intelligent Tutoring System using RAG and Prompt Engineering, *PeerJ Comput. Sci.* **11**, e2991 (2025).  
<https://doi.org/10.7717/peerj-cs.2991>