International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 13, Issue 3, 2025

Smart Teaching Assistant

Ishrath Nousheen, Ruthvika Saluvadi, Saanvi Macha

[1]Assistant Professor, Department of Information Technology, Bhoj Reddy Engineering College for Women.

[2.3]B,tech students, Department of Information Technology, Bhoj Reddy Engineering College for Women.

ruthvika369@gmail.com

ABSTRACT

The Smart Teaching Assistant (STA) is an AI-powered educational tool designed to automate the creation and assessment of academic questions. It uses Natural Language Processing (NLP) and machine learning techniques, especially Recurrent Neural Networks (RNNs), to generate relevant, diverse questions from provided text and evaluate student answers effectively. The system enables teachers to upload learning material, from which questions such as multiple-choice, short answer, and descriptive types are generated. It also evaluates answers based on semantic similarity with predefined responses, providing accurate feedback and scores. STA addresses major challenges in traditional education systems, such as time constraints, inconsistencies in evaluation, and lack of personalization. With a modular design supporting continuous model improvement via user feedback, the system ensures scalability, adaptability.

1. INTRODUCTION

In today's digital era, the integration of artificial intelligence into educational systems is becoming increasingly essential. One of the most promising applications is in automating the assessment process, which is often time-consuming and resource-intensive when handled manually. Teachers spend hours preparing question papers and evaluating student answers, which can result in delays, inconsistencies, and educator fatigue. Furthermore, students do not always receive timely feedback, which affects learning continuity and performance tracking. The Smart Teaching Assistant (STA) is an AI-based system designed to automate two core functions of the academic evaluation cycle: question generation and answer evaluation. It uses deep learning models, specifically Recurrent Neural Networks (RNNs), to generate contextually relevant questions from educational content provided by teachers. These questions are not limited to a specific type—they include multiple choice, short answer, and long descriptive formats, allowing for comprehensive assessments. Additionally, the system includes a sophisticated evaluation module that assesses student responses based on semantic similarity with ideal answers. This approach ensures fairness and consistency in grading, especially for subjective questions. STA not only reduces the workload of educators but also enhances learning by offering personalized, immediate feedback to students. The system is designed to be scalable, adaptable, and easy to integrate with existing Learning Management Systems (LMS).

Existing System:

The existing system uses rule-based approaches and automated grading for educational assessments. Rule-based methods rely on predefined templates to generate questions, which limits flexibility and adaptability across domains. Automated grading systems apply natural language processing techniques such as keyword matching, syntactic analysis, and semantic similarity to evaluate answers. However, these systems face challenges with accuracy and reliability, especially with complex or ambiguous input. They are often biased due to limited training data and lack adaptability across languages and educational levels. Consequently, these systems require significant manual customization, limiting their scalability and effectiveness in dynamic, real-world educational environments.

Proposed System:

The proposed Smart Teaching Assistant system utilizes advanced artificial intelligence, specifically Recurrent Neural Networks (RNNs), to enhance question generation and answer evaluation. It automates the creation of diverse question types—MCQs, short answers, and descriptive—by understanding the context of input text. The RNN model is fine-tuned using techniques like teacher forcing and beam search for better accuracy and variation. For evaluation, another RNN model assesses student responses by comparing them with ideal answers using semantic similarity and rubric-based scoring, providing detailed feedback. The system supports continuous learning through model updates and user feedback, ensuring adaptability, scalability, and personalized learning experiences in various educational and training environments.

2-RELATED WORK

The development of automated educational tools has evolved significantly, with early systems relying on rule-based methods for question generation and answer evaluation. These systems used predefined templates and rigid algorithms to create questions from textual input. While simple to implement, they lacked the flexibility to handle diverse content, adapt to different subject domains, or generate varied question types. This limitation led to reduced effectiveness in real-world educational settings [5].Subsequent research introduced automated grading systems using Natural Language Processing (NLP) techniques. These systems leveraged keyword matching, syntactic parsing, and semantic similarity metrics to evaluate the correctness of student answers. Although more sophisticated than rule-based systems, they still faced issues with accuracy, particularly when evaluating descriptive or complex responses [4]. Moreover, these systems often failed to provide personalized feedback, which is crucial for effective learning [3].Recent advances in artificial intelligence have enabled the use of machine learning models, particularly Recurrent Neural Networks (RNNs), to process sequential data such as natural language. Studies have shown that RNNs can generate contextually relevant and coherent questions from educational content, significantly improving the quality and diversity of assessments [1][2]. Additionally, RNNs used for answer evaluation can provide nuanced scoring and meaningful feedback by understanding the semantic structure of student responses. Research also emphasizes the importance of fairness and bias mitigation in educational AI. Ensuring unbiased evaluation and equitable learning opportunities remains a priority. Continuous learning and adaptation based on user feedback are emerging trends to keep the system relevant and accurate [1]. These advancements form the foundation for building robust smart teaching assistants.

3. REQUIREMENT ANALYSIS

Functional Requirements:
The Smart Teaching Assistant system is designed to deliver intelligent educational support through two core functionalities: automated question generation and answer evaluation. Key functional requirements include:

User Management

- Role-based access (Student, Teacher, Admin) including registration, login, and profile handling.

Content Input & Preprocessing

- Accepts textual data (paragraphs, articles, etc.) and prepares it using tokenization, cleaning, and segmentation for downstream processing.

Question Generation

- Uses an RNN model to generate multiple types of questions (objective,subjective) based on the provided content.

Answer Submission

- Enables students to input and submit responses through the platform interface.

Answer Evaluation

- Evaluates answers using semantic similarity metrics between student answers and model-generated ideal responses; provides rubric-based scores.

Data Management

- Handles storage and retrieval of uploaded content, generated questions, submitted answers, and student performance.

Model Training & Updates

- Incorporates continuous learning by training the RNN model with new data.

Non-Functional Requirements:

Performance

- Real-time response for question generation and answer evaluation; optimized model inference for smooth user experience.

Scalability

- Easily extendable to support large student bases and additional educational content..

Usability

- Intuitive UI built with HTML/CSS; clear navigation across modules for all user roles.

- **Software Requirements:**
- Operating System : Windows
- Frontend : HTML, CSS, JavaScript
- Machine Learning Libraries : TensorFlow, NLTK, Scikit-learn
- Backend : Python
- Web Development Framework : Flask
- Storage : MySQL
-
- **Hardware Requirements:**
- Processor : Intel i5
- RAM : 8GB
- Storage : 250GB SSD
- System-type : 64-bit / 32-bit OS

4. DESIGN

System Architecture:
The **Smart Teaching Assistant** system is designed with a modular, client-server-based architecture focused on scalability, efficiency, and ease of integration. This architecture divides the system into three main layers: the frontend interface, the backend processing engine, and the data storage layer. Each component operates independently yet communicates seamlessly to provide a responsive and intelligent educational experience for students, teachers, and administrators. The **frontend**, developed using HTML, CSS, and Node.js, serves as

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 13, Issue 3, 2025

the primary user interaction point. It provides a clean and intuitive web interface that adapts to the needs of different user roles. Students can view dynamically generated questions, submit answers, and receive feedback, while teachers and administrators can upload learning content, create quizzes, and monitor student progress. All frontend-to-backend communication is handled through secure HTTP requests, ensuring smooth data transmission and role-based access control. The **backend**, built using Python and the Flask framework, functions as the system's processing core. It handles content preprocessing, question generation, answer evaluation, and feedback delivery. Central to these tasks are two AI models based on **Recurrent Neural Networks (RNNs)**—one dedicated to generating questions from uploaded materials and another for evaluating student answers. These models are trained on domain-specific datasets and continually refined using real user feedback, enabling accurate and context-aware performance. The system's **MySQL database** manages structured data such as user information, uploaded content, questions, responses, scores, and feedback records. It supports efficient data retrieval and secure storage, with strict role-based access control to ensure data privacy and integrity. The overall architecture allows for real-time processing, high responsiveness under load, and the integration of additional intelligent features like quizzes, speech processing, and performance analytics—making the Smart Teaching Assistant a robust and future-ready educational tool.

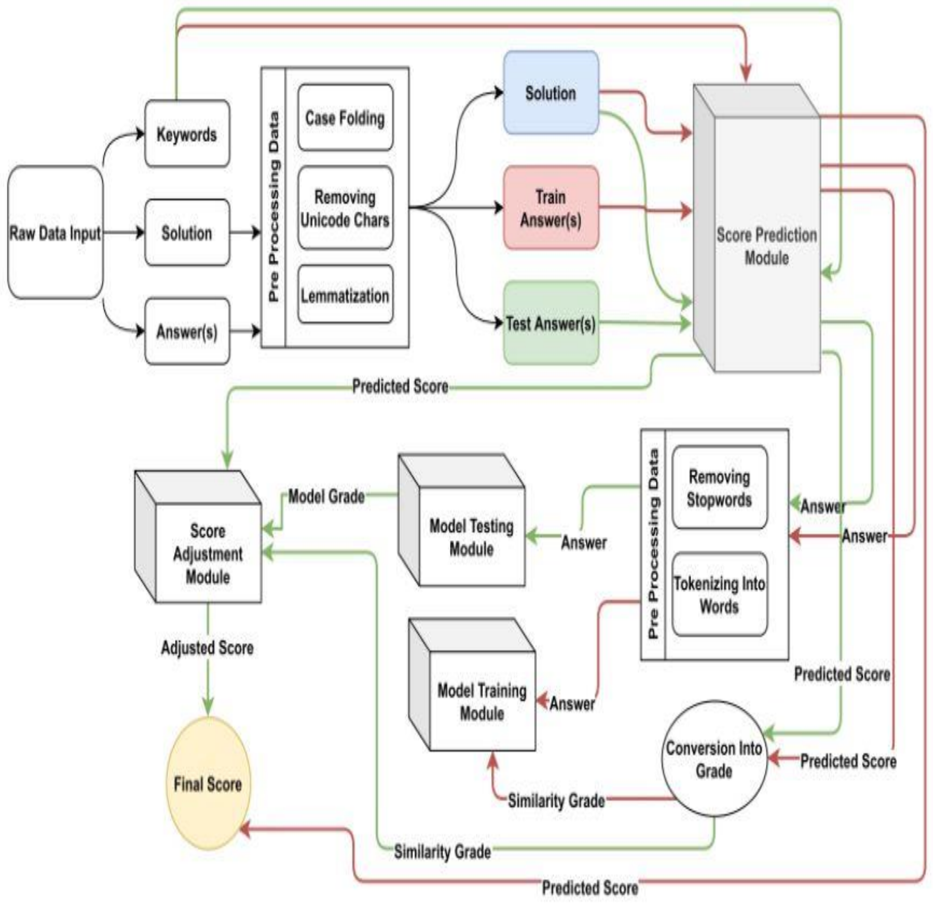


Fig. 4.1.1.1 System Architecture

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 13, Issue 3, 2025

Technical Architecture:
The technical architecture of the Smart Teaching Assistant is built upon a robust and scalable full-stack web framework, seamlessly integrating a responsive frontend with intelligent, AI-powered backend services. This architecture is designed to deliver a dynamic, real-time educational experience for users, ensuring efficient content management, automated assessmentsall within a secure and modular framework. The frontend of the application is developed using HTML, CSS, and Node.js, creating a responsive and interactive user interface. This interface allows students, teachers, and administrators to perform key tasks such as uploading learning content (like notes or lecture materials), viewing auto-generated questions, and submitting answers. The design is role-sensitive, meaning each user type gets access to features tailored specifically to their needs. The frontend communicates with the backend through secure RESTful APIs, enabling smooth and efficient data transmission. The backend is developed in Python using the lightweight Flask framework, which serves as the central processing hub for the system. It manages core functionalities such as content preprocessing, automatic question generation, and answer evaluation. At the core of these functions are Recurrent Neural Network (RNN)-based machine learning models. The question generation model intelligently transforms educational content into relevant and context-aware questions, while the answer evaluation model interprets student responses to determine their correctness, completeness, and relevance. To power these AI features, the system leverages advanced machine learning libraries including TensorFlow for neural network construction and training, NLTK (Natural Language Toolkit) for natural language preprocessing and tokenization, and Scikit-learn for additional ML utilities and evaluation metrics. These libraries allow the models to perform tasks such as semantic understanding, syntactic analysis, and sequence prediction—crucial for educational NLP tasks. For structured data storage, the system uses a MySQL relational database. It efficiently manages and stores user information, uploaded content, generated questions, student responses, evaluation scores, and feedback records. The database schema is optimized for relational integrity, quick retrieval, and scalability. It supports role-based access control, ensuring that sensitive data is accessible only to authorized users. This integrated architecture ensures modularity, allowing individual components (like AI models, UI modules, or the database) to be updated or scaled independently. It supports seamless data flow between layers and enables real-time AI processing, ensuring a responsive experience for users. Combined with encryption, session management, and access control mechanisms, the system maintains a secure environment while offering intelligent support for teaching and learning activities

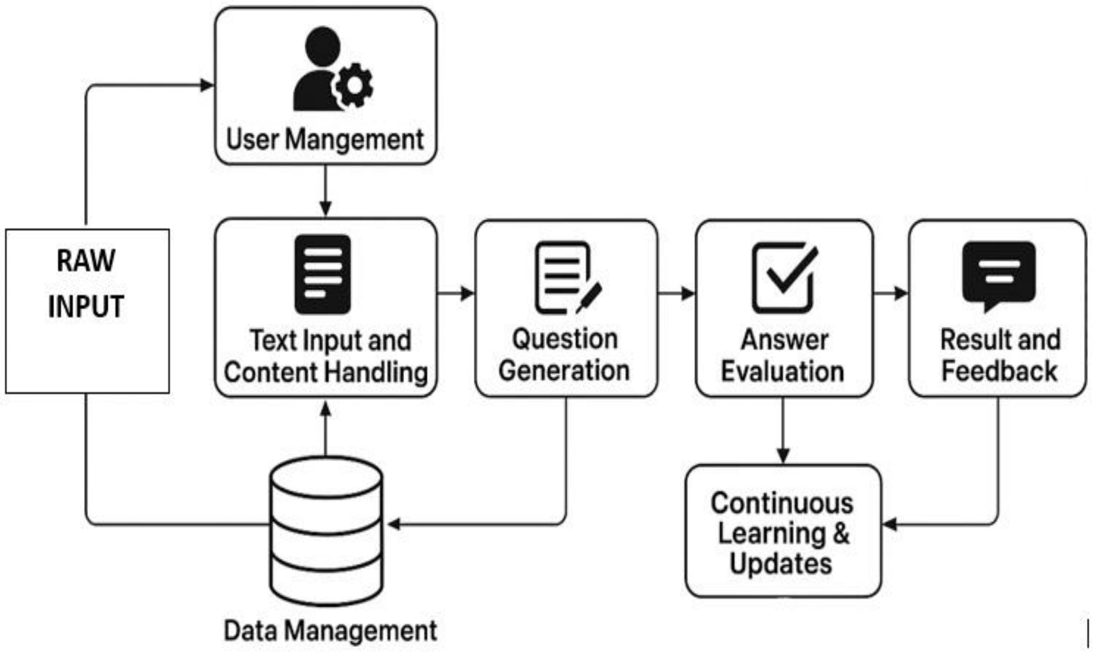


Fig. 4.1.2.1 Technical Architecture

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 13, Issue 3, 2025

5. IMPLEMENTATION

The Smart Teaching Assistant project is implemented as a web-based application that combines front-end user interaction with back-end AI-driven logic. The development involves multiple components including user management, content handling, question generation, and answer evaluation. The system is built using Python (Flask) for backend logic, HTML/CSS with Node.js for the frontend, and MySQL for database management. AI models are developed and integrated using machine learning libraries such as TensorFlow, NLTK, and Scikit-learn.

Libraries Used:

5.1.1 Flask – Lightweight web framework used to build RESTful backend APIs and manage server-side rendering.

5.1.2 TensorFlow – Used for training and deploying the Recurrent Neural Networks (RNNs) that generate questions and evaluate answers.

5.1.3 NLTK (Natural Language Toolkit) – Used for natural language processing tasks like tokenization, stemming, lemmatization, and sentence segmentation.

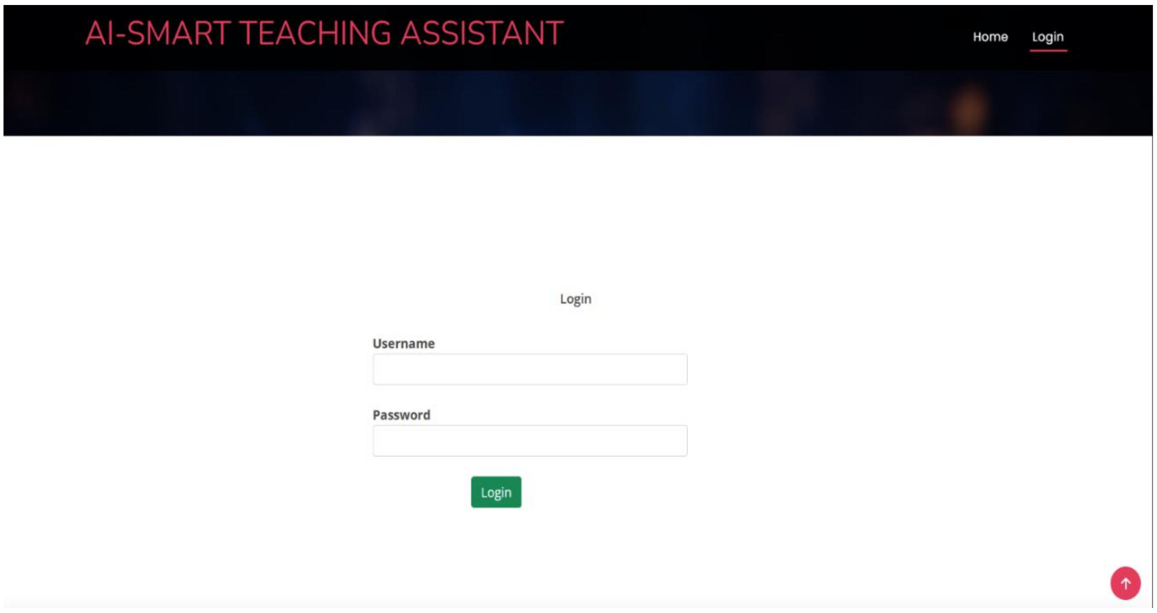
5.1.4 Scikit-learn – Utilized for calculating cosine similarity and implementing TF-IDF vectorization for answer evaluation.

5.1.5 LanguageTool – Checks grammatical accuracy and identifies errors in student-submitted answers.

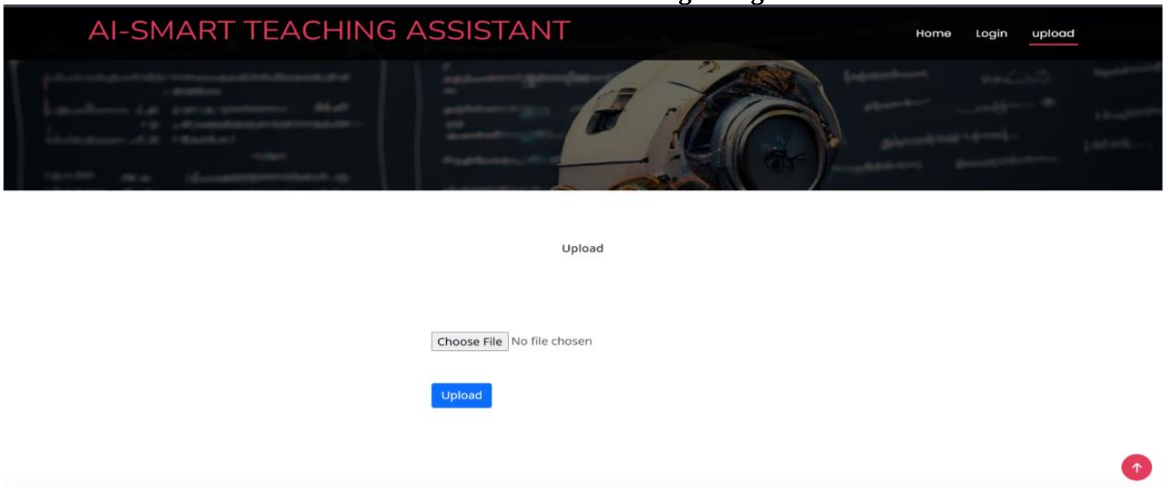
5.1.6 RAKE (Rapid Automatic Keyword Extraction) – Extracts keywords from text for question generation and evaluation.

6. SCREENSHOTS

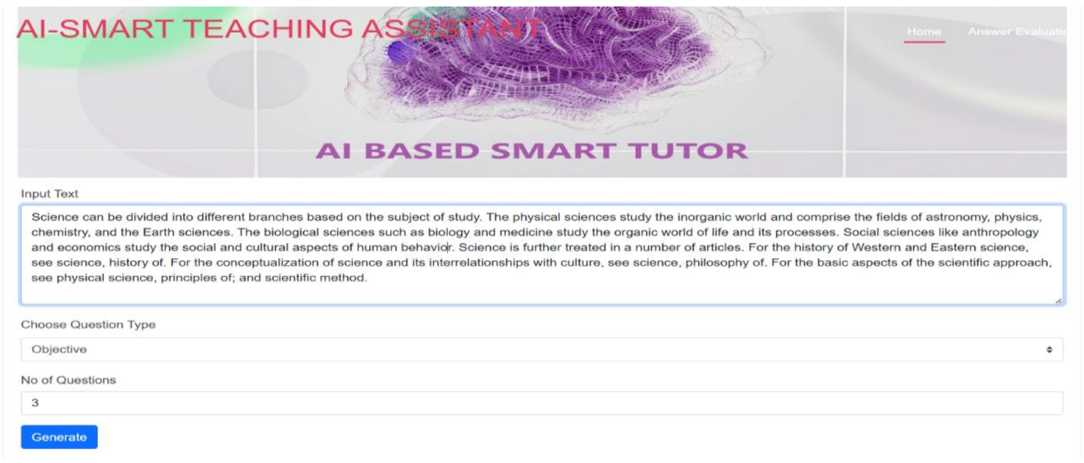
Screenshot 6.1 Home Page



Screenshot 6.2 Admin Login Page



Screenshot 6.3 Upload Page



Input Text

Science can be divided into different branches based on the subject of study. The physical sciences study the inorganic world and comprise the fields of astronomy, physics, chemistry, and the Earth sciences. The biological sciences such as biology and medicine study the organic world of life and its processes. Social sciences like anthropology and economics study the social and cultural aspects of human behavior. Science is further treated in a number of articles. For the history of Western and Eastern science, see science, history of. For the conceptualization of science and its interrelationships with culture, see science, philosophy of. For the basic aspects of the scientific approach, see physical science, principles of; and scientific method.

Choose Question Type: Objective

No of Questions: 3

Screenshot 6.5 Input data

ISSN 2347–3657

Volume 13, Issue 3, 2025

7. CONCLUSION

The Smart Teaching Assistant project demonstrates a powerful integration of artificial intelligence with education technology to automate and enhance learning experiences. By leveraging Recurrent Neural Networks (RNNs), the system effectively generates contextually accurate and diverse questions from any input text. It also evaluates student responses through semantic similarity and keyword analysis, providing immediate and meaningful feedback. This not only reduces the workload for educators but also enables personalized learning for students through adaptive assessment techniques. The modular architecture, combining a web-based interface with robust backend models, ensures the system is scalable, efficient, and user-friendly. It supports multiple user roles, secure data handling, and continuous model improvement through user feedback. The implementation of AI-driven evaluation and question generation bridges the gap between manual teaching methods and modern intelligent tutoring systems. Overall, the Smart Teaching Assistant provides a practical, innovative solution for digital education, promoting efficiency, fairness, and deeper understanding in academic assessments and self-paced learning environments.

REFERENCES

[1] E. Aflalo, “Students generating questions as a way of learning,” Act. Learn. Higher Educ., vol. 22, no. 1, pp. 63–75, 2021, doi: 10.1177/1469787418769120.

[2] L.Maplethorpe,H.Kim,M.R.Hunte,M.Vincett,andE. E.Jang,“Student generated questions in literacy education and assessment,” J. Lit. Res., vol. 54, no. 1, pp. 74–97, 2022, doi: 10.1177/1086296X221076436.

[3] F. Y. Yu, “Multiple peer-assessment modes to augment online student question-generation processes,” Comput. Educ., vol. 56, no. 2, pp. 484–494, 2011, doi: 10.1016/j.compedu.2010.08.025.

[4] F. Y. Yu and Y. H. Liu, “Creating a psychologically safe online space for a student-generated questions learning activity via different identity revelation modes,” Brit. J. Educ. Technol., vol. 40, no. 6, pp. 1109–1123, 2009.

[5] F. Y. Yu, “Scaffolding student-generated questions: Design and develop ment of a customizable online learning system,” Comput. Hum. Behav., vol. 25, no. 5, pp. 1129–1138, 2009.