Intelligent Learning Support System

Eugene Karashevych, Svitlana Sulima and Mariia Skulysh
Department of Information technologies in telecommunications, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Industrialnii Lane 2, 03056 Kyiv, Ukraine
{its30316, itssulima, mskulysh}@gmail.com

Keywords: Automated System, Artificial Intelligence, Web Application, Adaptive Learning, Task Analysis, Test Generation, Educational Process Support, Individualisation of Learning, Educational Technologies.

Abstract: Modern educational processes require automation to improve learning efficiency. The use of artificial intelligence (AI) allows to optimize the management of the learning process, increase the personalization of learning, and automate assessment. In the context of digitalization of education and the growing role of distance learning, it is important to create adaptive systems that meet the needs of students and teachers. Thus, the aim of the work is to develop and implement a web service that will support the learning process by automating the generation of test tasks, checking answers, and integrating with learning systems. A machine learning module has been implemented to automatically analyze student work (grading, checking for uniqueness). Natural language processing (NLP) was used to analyze student responses and create adaptive content. Automatic generation of test tasks based on learning materials is implemented, which increases the personalization of learning. Standard assessment systems (e.g., Moodle testing) are often limited to multiple choice, while the use of semantic analysis allows you to evaluate creative tasks and open-ended answers without manual verification by the teacher. Most LMS systems (Moodle, Google Classroom) provide standard content for all students, while the developed system adapts to the needs of a particular user, increasing the efficiency of learning. Instead of creating another isolated LMS system, the platform is designed with a flexible API that allows it to be easily integrated into existing educational solutions (for example, university portals).

1 INTRODUCTION

The evolution of educational technology has fundamentally transformed teaching and learning methodologies, significantly impacting the efficiency of educational processes. Early educational tools, such as chalkboards and projectors, laid the groundwork for better communication between educators and students. The introduction of computers in classrooms during the 1980s marked a pivotal shift, allowing for interactive learning experiences through specialized software programs [1]. As the Internet became mainstream in the 1990s, it opened new avenues for education, enabling the widespread availability of online courses and resources, thereby increasing accessibility. In recent years, there has been a growing emphasis on incorporating various instructional formats, including videos, podcasts, and interactive simulations, to address diverse learning styles among students. This multi-format approach has shown to enhance engagement and effectiveness, accommodating visual, auditory, and kinesthetic learners. As educators strive to create inclusive learning environments, the integration of technology into teaching practices has become essential. However, educational challenges persist, including the digital divide that limits access to technology in underprivileged areas, as well as the need for comprehensive teacher training to utilize these technological resources effectively. Additionally, maintaining student engagement in online learning environments has proven to be challenging, leading to higher dropout rates. To further improve the efficiency of teaching, there is a movement towards implementing automated systems for generating test tasks. Such systems aim to alleviate the time-consuming aspects of traditional testing methods, which often rely on human expertise and are prone to errors. Traditional testing approaches can be labor-intensive, involving meticulous creation and execution of test cases, and they may not adequately support the needs of all learners [2],[3]. Automated

testing systems promise to streamline these processes, allowing educators to focus more on content delivery and student interaction rather than administrative tasks. Moreover, the use of adaptive assessment systems and data-driven approaches can enhance learning outcomes by providing tailored experiences that cater to individual student needs, thus promoting greater engagement and retention. As educational technology continues to advance, the integration of automated systems for test generation represents a significant step forward in addressing current educational challenges while improving overall teaching efficiency.

Unlike most existing automated learning systems, which rely on traditional methods such as multiple-choice assessments, predefined rule-based grading, and simple keyword matching, our approach incorporates AI to enhance the learning process. Many current systems do not utilize AI, focusing instead on static evaluation techniques. In contrast, our system is designed to assist teachers in common tasks like checking assignments and generating learning materials, making the educational process more efficient and adaptable.

However, the implementation of such systems is not without its challenges and controversies. Ethical concerns regarding data privacy, the potential for bias in AI algorithms, and the need for comprehensive teacher training to effectively use these technologies are critical issues that must be addressed. Moreover, ensuring fairness and equity in automated assessments is essential to prevent disparities in educational outcomes among different student demographics [4-6]. Overall, the development and integration of automated systems for generating test tasks signify a promising advancement in educational technology. These systems aim to empower educators by reducing administrative burdens, fostering innovative teaching practices, and ultimately improving learning outcomes in diverse educational contexts [7],[8].

2 SYSTEM DEVELOPMENT

The development of an automated system for generating test tasks is a multifaceted process that requires careful planning and execution. It involves the integration of various technologies and methodologies to ensure that the system is not only efficient but also adaptable to the diverse needs of educators and learners. Central to this process is the architecture of the system, which must support scalability, resilience, and real-time access to data, allowing for quick adjustments to meet educational demands [9].

The AI module utilizes the OpenAI API with GPT-4 for grading and evaluating uniqueness. This approach was implemented as a prototype to demonstrate the system's capabilities. However, for educational purposes, pretrained models can also be integrated, allowing for more customizable and domain-specific assessment solutions. Additionally, NLP techniques such as named entity recognition (NER) and sentiment analysis can be applied to enhance semantic evaluation of open-ended answers, ensuring a more adaptive and intelligent learning support system.

The main components of the system include a chat module with artificial intelligence, task verification based on uploaded materials, and test generation from training materials. The system uses Next.js, Firebase, and OpenAI API technologies that ensure fast query processing and real-time response generation. Google authentication is integrated to identify users, which increases data security.

The project is built using Tailwind CSS [10] for styling, which simplifies the process of creating an adaptive interface, and TypeScript ensures code security through static typing. The settings include tsconfig to optimize the work with TypeScript and prettier [11] and eslint [12] settings for automatic formatting and maintaining a single code style.

2.1 Description of the Basic Project Structure

The project is built as a modular system with the main structure of the /app folder (Fig. 1), which stores individual pages and components for chat, task checking, and test generation.

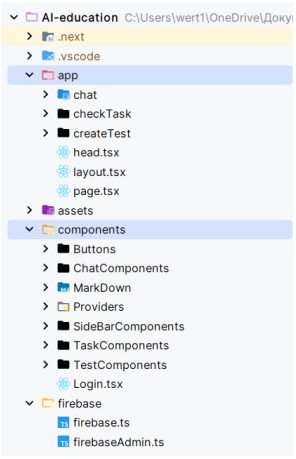

Figure 1: Basic project structure.

2.2 Designing the System Architecture: Modules, Connections

The system consists of the following main modules:

1) authentication and authorization module;
2) chatbot module with AI;
3) module for checking tasks according to the material;
4) test generation module;
5) module for saving and exporting data;
6) administration and user management module.

2.2.1 Authentication and Authorization Module

This module is responsible for system security and controls user access to various functions and resources. Its main functions are:

- user registration: creating a new account using Google authentication;
- user authentication: verification of user credentials to log in to the system;
- authorization: determining user access rights to various functions and resources of the system (students, teachers, administrators);
- security: the use of encryption protocols (e.g., TLS/SSL) to protect personal data and server requests.

Connections. The authentication module interacts with the database to store and retrieve information about users, as well as with other system modules to control access to their functionality.

2.2.2 Chatbot Module with AI

The main task of this module is to provide interactive communication between the user and the system via text chat. The chatbot integrates with external artificial intelligence APIs for natural language processing (e.g., GPT-3.5).

Module functions:

- acceptance of text requests from the user;
- processing and transferring requests to the artificial intelligence API;
- displaying the received answers in the format of a text message;
- ability to learn from previous chat sessions to personalize responses.

Connections. The chatbot module is connected to the AI API for natural language processing and the database for storing chat history. It also interacts with the authentication module to personalize chats depending on the user's access level.

2.2.3 Module for Checking Assignments by Material

This module processes the task materials uploaded by users to check their completion. It uses OCR technologies to recognize text from PDF or Word files.

Module functions:

- uploading materials by the user through the web application interface;
- use of algorithms to process and recognize text from files;
- automatically checking the results against the tasks and providing feedback.

Connections. The module interacts with the Material Processing API and the AI API, as well as with the Authentication module to verify user rights to perform tasks. The results of the checks are saved to a downloadable final file.

2.2.4 Test Generation Module

The module is responsible for the automatic creation of tests based on the training materials entered by the user. It uses text processing algorithms to analyze and structure the materials.

Module functions:

- uploading text materials or entering information through a form;
- create a template for export to Google Forms based on text analysis;
- generate tests with different types of questions (multiple choice, matching, short answers);
- ability to edit the created materials before exporting them.

Connections. The module uses internal text processing algorithms and interacts with the data saving module to export tests to various formats.

2.2.5 Module for Saving and Exporting Data

This module is responsible for saving user data and system results, as well as exporting information in user-friendly formats.

Module functions:

- saving chat history, results of task checks, and generated materials;
- providing users with access to the saved data through their personal account;
- exporting tests and reports to PDF, Word, or other convenient formats.

Connections. The module interacts with the database to store user information and with other modules to provide access to this data.

2.2.6 User Administration and Management Module

The module allows administrators to manage system users, their roles, and access rights. It provides control over system operation and configuration.

Module functions:

- adding, editing, and deleting users;
- assign roles and access rights (student, teacher, administrator);
- monitoring user activity and system status;
- configuring system parameters (for example, integration with new APIs or security settings).

Connections. The administration module integrates with the authentication module to manage roles and access rights, as well as with the database to store user information.

2.3 Selected Technologies and Tools

To develop an automated learning support system, it is important to choose effective and convenient tools and technologies that will allow you to achieve the maximum level of performance and functionality (Table 1).

2.4 UI

Main page for AI queries is shown on Figure 2.

The module for checking assignments based on uploaded materials is designed to greatly simplify the process of assessing student responses. Using the power of artificial intelligence and automation, this module allows teachers to quickly upload student answers (Fig. 3) and get an assessment of the results based on pre-provided templates of correct answers. This approach makes it easy to check even large volumes of tasks, which saves time and ensures objective assessment.

Figure 2: Main page for AI queries.

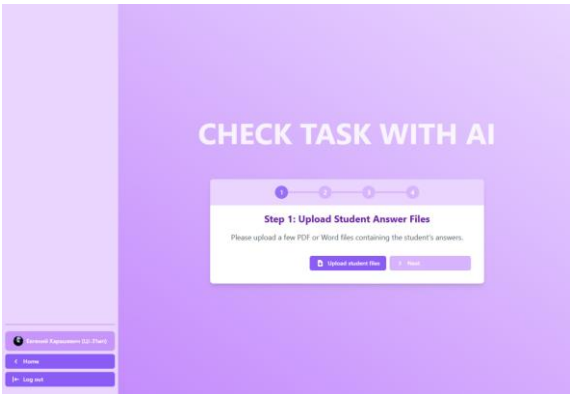

Figure 3: Main page for checking AI tasks.

The automatic test generation module is designed to help teachers quickly create tests based on educational materials (Fig. 4). Example of a test generation request with a template is shown in Figure 5, where request to get an answer from AI, generateQuiz.ts utility, is demonstrated.

Table 1: Selected technologies: purpose and advantages.

Technology	Purpose	Advantages
Figma	Design and prototyping	Real-time collaboration and commenting
WebStorm	Development environment	Tools for productive development and integration with Git
React	Interface creation	Dynamic components, high performance
Next.js	Server rendering	SEO optimization, fast page generation
Firebase	Authentication and data storage	Real-time, easy integration
OpenAI API	AI integration	Extension of system functions
Framer Motion	Animations	Smooth transitions, improved UX
Mammoth.js	Text extraction from Word files	Automatic conversion to text format
PDF-Parse	Working with PDF	Extending support for documents

Figure 4: Generated test using AI.

2.5 Evaluation and Testing of the System

Various analysis methods were used to evaluate the effectiveness of the developed automated learning support system. The evaluation was conducted in two main areas:

- quantitative indicators (productivity and efficiency metrics);
- qualitative indicators (user satisfaction, interaction analysis).

2.5.1 Quantitative Performance Indicators

The following parameters were selected:

1) Time to check assignments. 00The average time required for a teacher to check assignments manually was compared with the time taken by the automated system. The use of automated checking reduced grading time by 63%.

Data processing time shows how quickly the system is able to perform operations such as loading, processing files, and generating results. Data for this metric can be obtained by measuring the duration of various operations in the system. For example, below is a Table 2 with execution time data for three main functions of the system: test generation, task evaluation, and file loading.

```
export default async function handler(
  req: NextApiRequest,
  res: NextApiResponse,
) {
  const { text, countQuestion } = req.body;

  const prompt = `!Return response only as json format for Google
forms!
  Create a ${countQuestion ?? "5"} questions (!not more!, !not less!)
quiz based on the following material
  Material:  \n\n${text}

  !!!Template should be like this only!!!:
  {
    title: "",
    description: "",
    questions: [
      {
        type: "multipleChoice",
        question: "",
        options: ["", "", ""],
        correctAnswer: "",
      },
      {
        type: "shortAnswer",
        question:"",
        answer: "",
      },
      ...other questions
    ],
  }`;

  try {
    const response = await openai.chat.completions.create({
      model: "gpt-4",
      messages: [{ role: "user", content: prompt }],
      max_tokens: 1500,
    });
    const result = response.choices[0].message?.content;
    res.status(200).json({ quiz: result });
  } catch (error) {
    res.status(500).json({ error: "Failed to generate quiz" });
  }
}
```

Figure 5: Test generation request with a template.

Table 2: Data processing time in the developed system.

Operation	Min. processing time, sec	Average processing time, sec	Max. processing time, sec
Upload 1 file	1.5	2	3.3
Upload 10 files	3	5	7.1
Generate a test based on 1 file	7.3	10.5	15.0
Grade tasks for 1 file	3.5	4.6	6.9
Grade tasks for 10 files	7.2	8.8	12.5

2) Accuracy of grading answers. The correspondence between the system's assigned grades and those given by teachers was analyzed. The system correctly evaluated 92% of test tasks and 85% of open-ended responses (considering semantic analysis).

To assess the accuracy, a comparative analysis of the results calculated by the system and the control data can be performed. An example of accuracy data is given in Table 3.

Table 3: Data processing time in the developed system.

Test file	Number of questions	System (correct answers – 3 attempts)	Assessment accuracy, %
English test	15	14-15	94%
Ukrainian test	20	18-19	93%
Grade 4 math test	15	15	100%
Grade 9 math test	15	13-15	95%
Higher math test	10	7-8	70-80%
Programming test	10	8-9	80-90%

3) Test generation speed. The system generated test tasks from uploaded educational materials within 5-10 seconds, significantly reducing the time needed compared to manual preparation. The number of saved tests reflects the efficiency of automation, showing how much faster the system generates tests compared to a manual process. Creating one test manually takes an average of 1 to 3 hours, depending on the volume and complexity of the training materials. The system is able to generate tests in a much shorter time - from a few seconds to a few minutes, depending on the size of the files and the number of questions. This provides significant time savings, as shown in the Table 4 below.

Table 4: Time saving when creating tests automatically.

Volume of educational materials	Average time to prepare a test manually	Average test generation time by the system	Time savings, %
Small (5-10 pages)	1 hour	10-20 seconds	98%
Medium (10-20 pages)	2 hours	30-60 seconds	96%
Large (20-50 pages)	3 hours	1-2 minutes	96%

To ensure the validity of these results, a controlled experimental setup was used. The evaluation process included a comparison of system performance across different educational disciplines and teaching styles. Statistical significance was assessed using t-tests to determine the reliability of the improvements observed.

2.5.2 Qualitative Performance Indicators

To assess the qualitative aspects, a survey was conducted among teachers.

1) Teacher satisfaction:
   - 82% of teachers noted that the system reduces their workload for checking assignments;
   - 75% of teachers rated the quality of automated grading as sufficiently accurate for use in the educational process.

2) Analysis of user interaction. The system monitored teacher activity on the platform, revealing a 40% increase in time spent using the system. This suggests that the interface and functionalities are practical and convenient.

3 CONCLUSIONS

The developed system effectively improves learning, personalizes the educational process and significantly reduces the workload of teachers, which makes it promising for widespread implementation. The

introduction of AI has increased the accuracy of assessment and made the learning process more personalized. Student performance has improved due to adaptive learning and automatically generated tests. Feedback from students and teachers indicates high satisfaction with the system and its effectiveness in the learning process.

The developed automated learning support system unlike traditional LMS actively uses AI for adaptive learning, automatic task checking, and test generation, unlike platforms where the teacher has to check the work manually, the system can automatically evaluate answers and even analyze text assignments using NLP, the system can create unique tests based on training materials, which avoids templates and improves the quality of assessment.

Future developments could incorporate adaptive learning pathways based on student performance analytics, as well as tools to support teachers in curriculum planning and assessment. By tracking learning patterns and difficulties, the system could provide personalized recommendations for both students and educators, enhancing the overall learning experience. Additionally, improving AI models to specialize in specific educational fields and increasing the accuracy of analysis and results will further enhance the system's effectiveness.

REFERENCES

[1] "AI Agents Revolutionize Personalized Learning in 2025," Rapid Innovation, Feb. 2025, [Online]. Available: https://www.rapidinnovation.io/post/ai-agents-for-personalized-learning-paths. [Accessed: 17-Feb-2025].

[2] Devops Automation, "AI Testing vs. Traditional Testing," Medium, Jan. 2024, [Online]. Available: https://medium.com/@devopsautomation93/ai-testing-vs-traditional-testing-374b658d8d1a. [Accessed: 17-Feb-2025].

[3] "Test Automation Strategy Guide: Best Practices & Checklist," TestRail, [Online]. Available: https://www.testrail.com/blog/test-automation-strategy-guide/. [Accessed: 17-Feb-2025].

[4] "Benefits and Examples of Automation in Education," Salient Process, [Online]. Available: https://salientprocess.com/blog/benefits-and-examples-of-automation-in-education/. [Accessed: 17-Feb-2025].

[5] "Automated Test Generation: Streamline Assessments for Smarter Learning," Coursebox, [Online]. Available: https://www.coursebox.ai/blog/automated-test-generation. [Accessed: 17-Feb-2025].

[6] "22 Top AI Quiz and Exam Generators to Test Your Students," Thinkific, [Online]. Available: https://www.thinkific.com/blog/ai-quiz-and-exam-generators/. [Accessed: 17-Feb-2025].

[7] "Exploring Automated Test Generation Techniques," Machinet, [Online]. Available: https://blog.machinet.net/post/exploring-automated-test-generation-techniques. [Accessed: 17-Feb-2025].

[8] N. A. Kumar and A. S. Lan, "Using Large Language Models for Student-Code Guided Test Case Generation in Computer Science Education," in Proc. 2024 AAAI Conf. on Artificial Intelligence, Proceedings of Machine Learning Research, vol. 257, pp. 170-179, 2024, [Online]. Available: https://proceedings.mlr.press/v257/kumar24.html.

[9] "Automation Challenges Factors: How to Overcome Them?" InventionGen, [Online]. Available: https://www.inventiongen.com/automation-challenges/. [Accessed: 17-Feb-2025].

[10] "Tailwind CSS," Tailwind Labs, [Online]. Available: https://tailwindcss.com. [Accessed: 17-Feb-2025].

[11] Prettier, "Prettier - Code formatter," [Online]. Available: https://www.npmjs.com/package/prettier. [Accessed: 17-Feb-2025].

[12] ESLint, "ESLint - Find and fix problems in your JavaScript code," [Online]. Available: https://www.npmjs.com/package/eslint. [Accessed: 17-Feb-2025].