https://doi.org/10.31891/2307-5732-2025-349-29
УДК 004.42

КРИЛИК ЛЮДМИЛА
Вінницький національний технічний університет
https://orcid.org/0000-0001-6642-754X
e-mail: lyudmila.krylik@gmail.com
АЛГАШ АНАТОЛІЙ
Вінницький національний технічний університет
e-mail: algash04@gmail.com

РОЗРОБКА ПРОГРАМНОГО МОДУЛЯ ПЕРЕВІРКИ КОРЕКТНОСТІ ВИКОНАННЯ СТУДЕНТСЬКИХ РОБІТ

У роботі представлено розробку програмного модуля перевірки коректності виконання студентських робіт. Метою розробки є розширення функціональних можливостей програмного модуля за рахунок застосування нових програмних інструментів. Програмний модуль є сучасним WEB-застосунком і складається з клієнтської та серверної частин. Архітектура розробки основана на принципі CQS, який дозволяє ефективно організувати бізнес-логіку. Серверна частина створена на базі C# та ASP.NET, це забезпечує стабільність, продуктивність та можливість масштабування. Клієнтська частина розроблена за допомогою TypeScript та React, що гарантує гнучкість, інтерактивність та зручний інтерфейс. Для роботи з базою даних використовувалася SQL, це забезпечує ефективне управління збереженими даними. Як середовище виконання розробки обрано WEB-браузери, що дозволяє використовувати програмний модуль без необхідності встановлення додаткового програмного забезпечення. Для кращого розуміння внутрішньої логіки програмного модуля розроблено UML-діаграми класів для серверної та клієнтської частини модуля. Для перевірки коректності виконання студентських робіт програмний модуль використовує нейромережу OpenAI Davinci, яка забезпечує високу точність і глибину аналізу. Програмний модуль успішно пройшов тестування та має розширений функціонал на 4 можливості: локалізація усіх мов світу дозволяє перевіряти роботи незалежно від мови їх написання; підтримка будь-якого формату файлів робіт студентів забезпечує універсальність використання; глибокий аналіз коректності виконання роботи студента сприяє більш точному оцінюванню результатів; гнучке налаштування параметрів програмного модуля дає можливість адаптувати процес перевірки відповідно до вимог викладачів. Розробка є ефективним інструментом для автоматизації перевірки студентських робіт, що значно спрощує процес оцінювання, забезпечує об'єктивність та зменшує витрати часу викладачів.

Ключові слова: студентські роботи, оцінювання, автоматизована перевірка, допомога викладачам.

KRYLIK LYUDMILA
Vinnytsia National Technical University
ALHASH ANATOLII
Vinnytsia National Technical University

DEVELOPMENT OF A SOFTWARE MODULE FOR CHECKING THE CORRECTNESS OF STUDENT WORKS

The paper presents the development of a software module for checking the correctness of student work. The goal of the development is to expand the functionality of the software module by using new software tools. The software module is a modern WEB application and consists of client and server parts. The development architecture is based on the CQS principle, which allows for effective organization of business logic. The server part is created on the basis of C# and ASP.NET, which provides stability, performance and scalability. The client part is developed using TypeScript and React, which guarantees flexibility, interactivity and a convenient interface. SQL was used to work with the database, which provides effective management of stored data. WEB browsers were chosen as the development execution environment, which allows using the software module without the need to install additional software. For a better understanding of the internal logic of the software module, UML class diagrams for the server and client parts of the module were developed. To check the correctness of student work, the software module uses the OpenAI Davinci neural network, which provides high accuracy and depth of analysis. The software module has successfully passed testing and has expanded functionality with 4 features: localization of all world languages allows you to check papers regardless of the language of their writing; support for any file format of students' papers ensures universal use; in-depth analysis of the correctness of the student's work contributes to a more accurate assessment of the results; flexible configuration of the software module parameters makes it possible to adapt the verification process in accordance with the requirements of teachers. The development is an effective tool for automating the verification of student papers, which significantly simplifies the evaluation process, ensures objectivity and reduces the time spent by teachers.

Keywords: student work, assessment, automated checking, assistance to teachers.

Постановка проблеми у загальному вигляді та її зв'язок із важливими науковими чи практичними завданнями

Головна мета вищих навчальних закладів – це забезпечити якісну освіту, вдосконалюючи студентський контингент, викладацький склад, методи навчання, а також поглиблюючи інтеграцію освіти, науки та інновацій відповідно до діючих стандартів освіти. Крім того, якість перевірки виконаних студентських робіт є критично важливим фактором для забезпечення якісної освіти.

Зі збільшенням кількості студентів ускладняється процес перевірки студентських робіт для викладачів. Часто перевірка забирає занадто багато часу та зусиль, тому використання програмного забезпечення для перевірки коректності виконання студентських робіт може значно спростити процес перевірки і вирішити ці проблеми [1 – 3].

У сучасному світі цифрових технологій, де освіта стає доступнішою, а конкуренція між освітніми платформами та університетами зростає, розробка програмного модуля для перевірки коректності виконання студентських робіт є доцільною та має практичне значення. Програмний модуль є ключовим елементом у забезпеченні якості навчального процесу, об'єктивності оцінок та оптимізації роботи викладачів.

Аналіз досліджень та публікацій

На сьогодні існує велика кількість додатків для перевірки коректності виконання студентських робіт, проте переважна кількість з них не покриває всі потреби користувачів. Ці додатки часто мають обмежений функціонал, зокрема не підтримують аналіз робіт різними мовами, не працюють з усіма форматами файлів або надають лише поверхневий аналіз. Більшість із них орієнтовані на вузькоспеціалізовані завдання та не дозволяють гнучко налаштовувати параметри перевірки, що обмежує їх застосування в освітніх закладах із різними вимогами. Крім того, існуючі системи можуть бути недостатньо швидкими або точними в оцінюванні, що зменшує їх ефективність у масштабних процесах перевірки [4− 6]. Тому виникає потреба в універсальному інструменті, який би враховував усі ці аспекти та відповідав сучасним вимогам користувачів.

Формулювання цілей статті

Метою роботи є розширення функціональних можливостей програмного модуля перевірки коректності виконання студентських робіт за рахунок застосування нових програмних інструментів. Це сприятиме підвищенню якості навчального процесу, забезпечить доступ викладачів та студентів до ефективного інструменту автоматизованої перевірки завдань, підвищить об’єктивність та якість оцінювання виконаних студентських робіт.

Для досягнення цієї мети потрібно вирішити такі задачі:

1) провести аналіз технічного рівня існуючих систем-аналогів перевірки коректності виконання студентських робіт;

2) розробити архітектуру і структуру програмного модуля перевірки коректності виконання студентських робіт;

3) виконати програмну реалізацію програмного модуля перевірки коректності виконання студентських робіт, яка забезпечить розширення його функціональних можливостей;

4) провести тестування розробки;

5) з проведених досліджень зробити висновки.

Виклад основного матеріалу

Розроблюваний програмний модуль перевірки коректності виконання студентських робіт – це сучасний WEB-застосунком, який складається з клієнтської та серверної частин. Клієнтська частина відповідає за взаємодію користувача з системою через зручний інтерфейс, а серверна частина забезпечує виконання основної логіки роботи модуля, а саме, перевірку виконаних студентських робіт, обробку запитів та генерацію результатів.

Серверна частина створена на базі C# та ASP.NET, це забезпечує стабільність, продуктивність та можливість масштабування [7, 8]. Для роботи з базою даних використовувалася SQL, це забезпечує ефективне управління збереженими даними [8]. Клієнтська частина розроблена за допомогою TypeScript та React, що гарантує гнучкість, інтерактивність і зручний інтерфейс [9, 10]. Для спрощення процесу розробки та підвищення зручності підтримки кожна з цих частин поділена на модулі, які відповідають за конкретні функції. Модулі серверної частини називаються хендлерами. Вся архітектура розробки базується на принципі CQS (Command Query Separation), який дозволяє ефективно організувати бізнес-логіку [11, 12]. CQS-архітектура чітко розділяє логіку виконання на команди та запити і сприяє покращенню продуктивності та модульності програмного забезпечення. Крім того, розділення завдань зменшує залежність між компонентами, це спрощує як управління змінами так і внесення оновлень. Зазначимо, що завдяки такому підходу кожен модуль легко тестується, тому що виконує лише одну конкретну функцію. Використання CQS забезпечує кращу масштабованість системи, оскільки розробники можуть додавати нові функції без ризику вплинути на існуючу логіку.

Хендлери є основою серверної частини програмного модуля і забезпечують виконання окремих завдань системи. Кожен хендлер виконує конкретну функцію, що робить код структурованим і зрозумілим для розробників. На рис. 1 подана загальна структурна схема функціонування програмного модуля перевірки коректності виконання студентських робіт.

Головна сторінка є стартовою точкою взаємодії користувача з програмним модулем. Вона містить вікно зі списком останніх перевірених робіт, де користувач може переглянути вже проаналізовані завдання. Це вікно формує API-запит до модуля перегляду останніх перевірених робіт, який обробляє запит на сервері, отримує дані з бази даних та відправляє відповідь до клієнтської частини. Усі отримані дані, включаючи назву роботи, оцінку, дату перевірки та зауваження виводяться у зручному форматі. Для перевірки коректності виконання студентських робіт програмний модуль використовує нейромережу OpenAI Davinci, яка забезпечує високу точність і глибину аналізу. На рис. 2 подано загальну структурну схему компонентів програмного модуля перевірки коректності виконання студентських робіт.

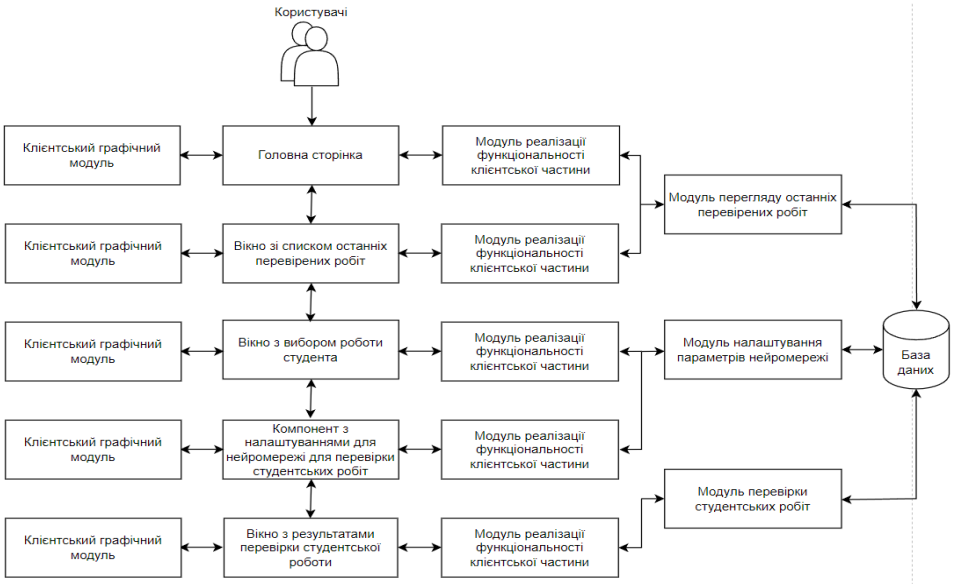

Рис. 1. Загальна структурна схема функціонування програмного модуля перевірки коректності виконання студентських робіт

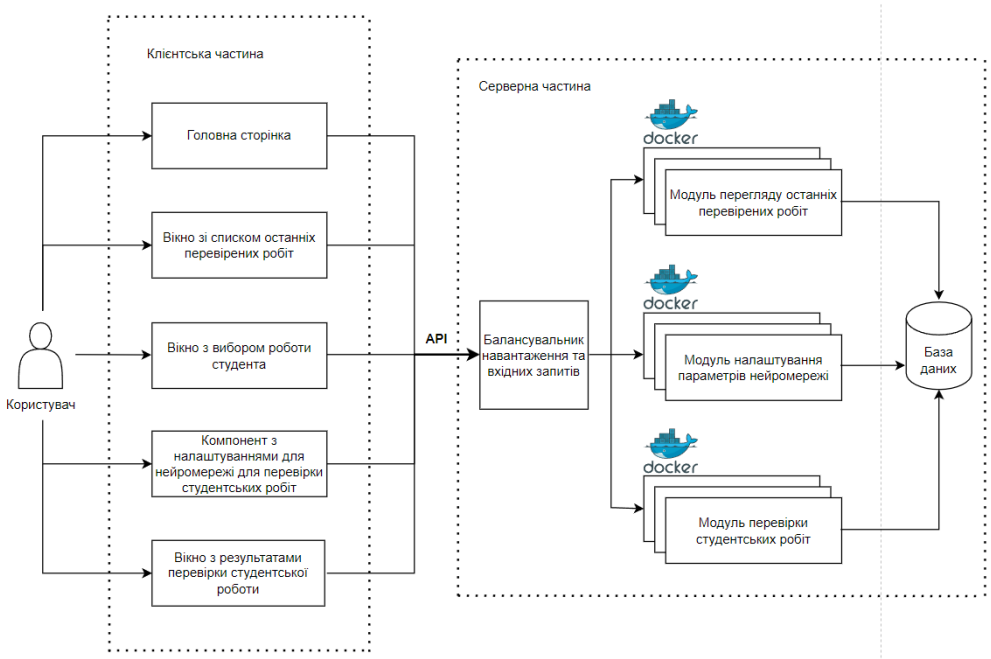

Рис. 2. Загальна структурна схема компонентів програмного модуля перевірки коректності виконання студентських робіт

Усі API-запити від клієнтської частини до серверних мікросервісів проходять через балансувальник навантаження. Основною задачею балансувальника є рівномірне розподілення навантаження між мікросервісами, що забезпечує оптимальну продуктивність системи. Це дозволяє уникнути перевантаження окремих серверів, забезпечити стабільність роботи всієї системи та мінімізувати час відповіді на запити користувачів. Балансувальник також відіграє ключову роль у підвищенні надійності системи та гарантує доступність сервісу навіть за умов пікових навантажень.

Для кращого розуміння внутрішньої логіки програмного модуля розроблено UML-діаграми класів для серверної та клієнтської частини модуля [13]. На рис. 3 зображена UML-діаграма класів серверної частини, а на рис. 4 − UML-діаграма класів клієнтської частини розроблюваного програмного модуля перевірки коректності виконання студентських робіт.

Розглянемо основні програмні рішення розробки, які забезпечують розширення функціональних можливостей програмного модуля перевірки коректності виконання студентських робіт. UML-діаграма серверної частини (рис. 3) містить клас MarkLabHandler, який відповідає за обробку запитів до нейромережі та обробку її результатів. Програмна реалізація якого має такий зміст (рис. 5).

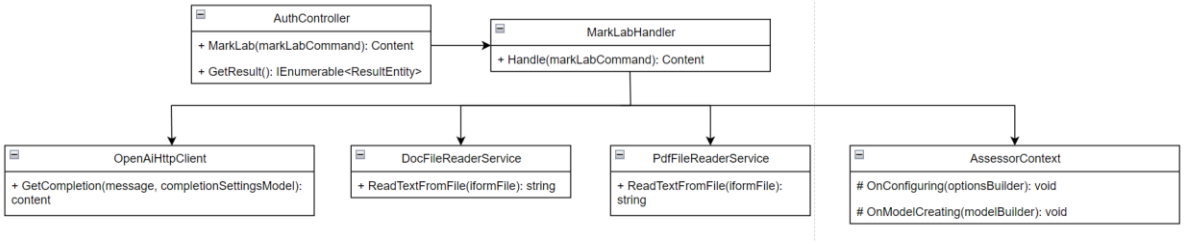

Рис. 3. UML-діаграма класів серверної частини програмного модуля перевірки коректності виконання студентських робіт

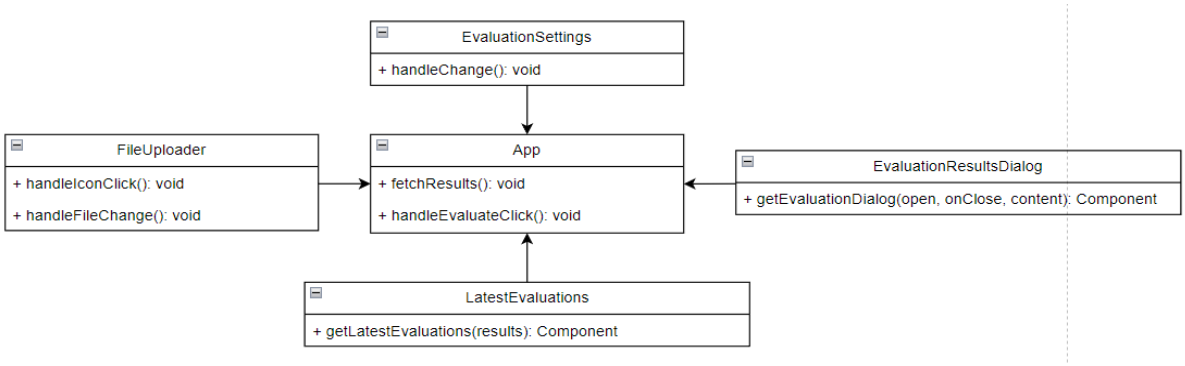

Рис. 4. UML-діаграма класів клієнтської частини програмного модуля перевірки коректності виконання студентських робіт

```
using System.Net;
using Assessor.Server.Application.Common.Interfaces;
using Assessor.Server.Application.Extensions;
using Assessor.Server.Domain.Contexts;
using Assessor.Server.Domain.Entities;
using Assessor.Server.Domain.Enums;
using Assessor.Server.Domain.Extensions;
using Assessor.Server.Domain.Models;
using MediatR;
using Microsoft.Extensions.DependencyInjection;

namespace Assessor.Server.Application.Lab.Commands.MarkLab;

public class MarkLabHandler : IRequestHandler<MarkLabCommand, Result<Content, Error>>
{
    private readonly IAiHttpClient _aiHttpClient;
    private readonly IServiceProvider _serviceProvider;
    private readonly AssessorContext _context;

    public MarkLabHandler(IAiHttpClient aiHttpClient, IServiceProvider serviceProvider,
AssessorContext context)
    {
        _aiHttpClient = aiHttpClient;
        _serviceProvider = serviceProvider;
        _context = context;
    }

    public async Task<Result<Content, Error>> Handle(MarkLabCommand request,
CancellationToken cancellationToken)
    {
        var fileReaderType = request.LabFile.GetFileReaderType();

        if (fileReaderType is FileReaderTypes.Unknown)
        {
            return new Error(HttpStatusCode.BadRequest, "Invalid file type");
        }

        var scope = _serviceProvider.CreateScope();
        var fileReaderService =
scope.ServiceProvider.GetRequiredKeyedService<IFileReaderService>(fileReaderType);
        var completionSettingsModel = request.MapToCompletionSettings();

        var result = await _aiHttpClient.GetCompletion(readTextResult.Value,
completionSettingsModel);

        if (result.IsSuccess)
        {
            var resultEntity = new ResultEntity
            {
                Mark = result.Value.Mark,
                Remark = result.Value.Remark,
                Conclusion = result.Value.Conclusion,
                Lab = new LabEntity
                {
                    Name = request.LabFile.FileName,
                    Extension = fileReaderType.ToString(),
                    Data = await request.LabFile.ConvertToBase64WithPrefixAsync(),
                    CompletionSettings = new CompletionSettingsEntity
                    {
                        Temperature = request.Temperature,
                        MaxTokens = request.MaxTokens,
                        TopP = request.TopP,
                        FrequencyPenalty = request.FrequencyPenalty,
                        PresencePenalty = request.PresencePenalty,
                    }
                }
            };

            _context.Results.Add(resultEntity);

            var rowsAffected = await _context.SaveChangesAsync(cancellationToken);

            if (rowsAffected is 0)
            {
                return new Error(HttpStatusCode.BadRequest, "No rows affected");
            }
        }

        return result;
    }
}
```

Рис. 5. Фрагмент програмної реалізації запиту та обробки відповіді від нейромережі «MarkLabHandler»

Клас MarkLabHandler.Handle(MarkLabCommand request, CancellationToken cancellationToken) обробляє команду MarkLabCommand та повертає результат у вигляді Result<Content, Error>. Спочатку визначається тип зчитувача файлу (FileReaderTypes) на основі переданого файлу. Якщо тип невідомий, повертається помилка HttpStatusCode.BadRequest. Потім створюється скоуп для отримання сервісу IFileReaderService відповідного типу та виконується читання тексту з файлу. Якщо читання не вдалось, повертається відповідна помилка. Створюється модель CompletionSettingsModel і відправляється запит до _aiHttpClient.GetCompletion, щоб отримати оцінку за виконану лабораторну роботу від ШІ. Якщо відповідь успішна, створюється сутність ResultEntity, яка містить результати оцінювання та збережені параметри лабораторної роботи. Ця сутність додається в контекст бази даних _context.Results, після чого зміни зберігаються. Якщо жоден рядок не було змінено (rowsAffected is 0), повертається помилка HttpStatusCode.BadRequest. В іншому випадку повертається отриманий від ШІ результат.

На рис. 6. подано загальний вигляд інтерфейсного вікна «MarkLabHandler», де візуалізується процес налаштування параметрів нейромережі.

Розроблений програмний модуль перевірки коректності виконання студентських робіт успішно пройшов тестування. Результати тестування розробленого програмного модуля та аналогів: Grammarly, Codio, SonarQube подано в табл. 1.

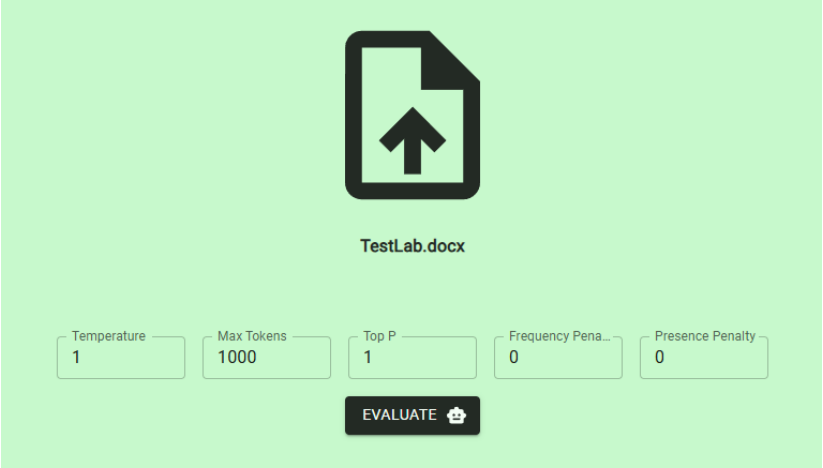

Рис. 6. Загальний вигляд інтерфейсного вікна «MarkLabHandler»

Таблиця 1

Результати тестування розробленого програмного модуля перевірки коректності виконання студентських робіт та аналогів

Характеристика	Grammarly	Codio	SonarQube	Розроблений програмний модуль
Основна функція	Перевірка тексту англійською мовою	Автоматизоване тестування програмування	Статичний аналіз коду	Перевірка коректності студентських робіт
Локалізація	Тільки англійська	Обмежена підтримка	Обмежена підтримка	Локалізація усіх мов світу
Формати файлів	Текстові файли	Підтримка обмеженої кількості форматів	Код програмування	Підтримка будь-якого формату файлів
Глибокий аналіз	Стилістичні та граматичні рекомендації	Аналіз коду, але обмежений автоматизацією	Статичний аналіз якості коду	Глибокий аналіз коректності виконання роботи
Гнучкість налаштувань	Мінімальні налаштування	Гнучкість у межах налаш-тувань оціню-вання коду	Складні налаштування для аналізу	Гнучке налаштування параметрів модуля
Підтримка мов програмування	Не підтримується	Обмежена кількість мов	Понад 25 мов	Підтримка всіх мов, необхідних для студентських робіт
Продуктивність	Висока для текстових перевірок	Середня, залежить від складності завдань	Висока, але залежить від розміру проекту	Висока для всіх типів завдань
Автоматизація	Виправлення стилю та граматики	Тестування програмного коду	Автоматичний аналіз коду	Повністю автомати-зована перевірка будь-яких завдань
Характеристика	Grammarly	Codio	SonarQube	Розроблений програмний модуль
Ресурсомісткість	Низька	Середня	Висока	Оптимізована для роботи на стан-дартному обладнанні
Додаткові можливості	Рекомендації щодо стилю	Інтерактивне навчання	Метрики якості, оцінка технічного боргу	Інноваційні функції відсутні в аналогах

На основі аналізу даних табл. 1 видно, що розроблений програмний модуль в порівнянні з аналогами має розширений функціонал на 4 можливості: локалізація усіх мов світу; підтримка будь-якого формату файлів робіт студентів; глибокий аналіз коректності виконання роботи студента; гнучке налаштування параметрів програмного модуля.

Особливістю розробленого програмного модуля перевірки коректності виконання студентських робіт є його висока швидкість роботи, точність оцінювання та можливість надання розгорнутого висновку щодо виконання роботи. На рис. 5 та рис. 6 представлено аналіз швидкості перевірки та якості оцінювання студентських робіт порівняно з системами-аналогами, такими як Grammarly, Codio та SonarQube.

На рис. 5 представлено графік залежності швидкості перевірки студентських робіт від їх кількості.

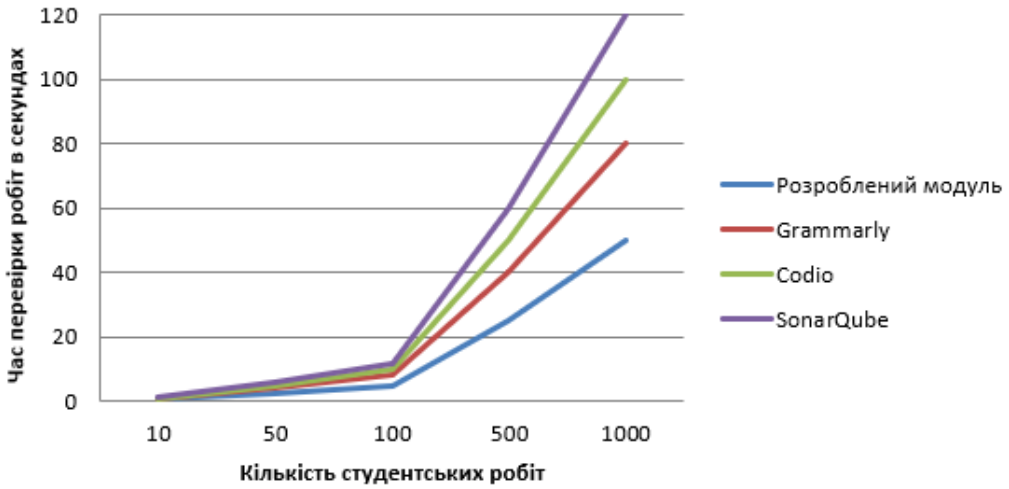

Рис. 5. Графік швидкості перевірки коректності виконання студентських робіт

З рис. 5 видно, що розроблений програмний модуль демонструє найвищу продуктивність, особливо за великої кількості робіт. Наприклад, при 1000 роботах час перевірки розробки становить лише 50 секунд, тоді як для Grammarly – 80 секунд, Codio – 100 секунд, а SonarQube – 120 секунд. Це свідчить про оптимізацію алгоритмів модуля, що дозволяє значно скоротити час обробки даних порівняно з системами-аналогами.

На рис. 6 подано порівняння якості оцінювання студентських робіт за трьома параметрами: глибина аналізу, точність оцінки, детальний висновок. Оцінювання проводилось за 10-ти бальною шкалою. Розроблений модуль отримав найвищі оцінки за всіма параметрами. Глибина аналізу розробки становить 9.5 балів і перевищує Grammarly (7 балів), Codio (8 балів), SonarQube (8.5 балів). Точність оцінки розробленого програмного продукту досягає 9.8 балів, що значно краще за Grammarly (8 балів) та SonarQube (8 балів), Codio (7.5 балів). Детальний висновок розробки оцінений у 9.7 балів і перевищує показники Grammarly (7.5 балів), Codio (8.2 балів) і SonarQube(8.5 балів).

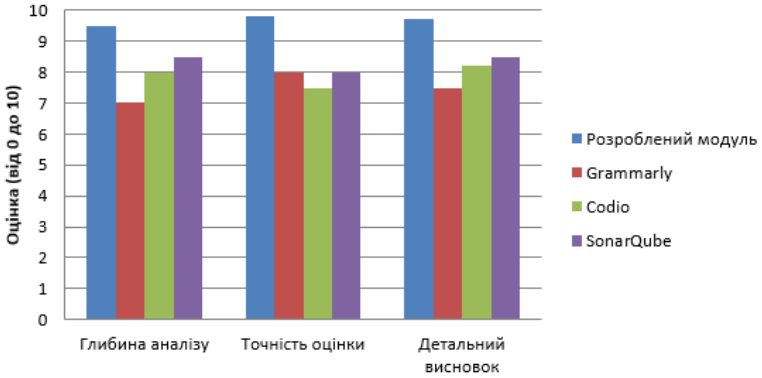

Рис. 6. Графік якості оцінки студентських робіт

Зазначимо, результати аналізу та тестування розробленого програмного модуля підтвердили досягнення поставленої мети. Як видно з результатів табл. 1, рис. 5 та рис. 6, програмний модуль демонструє покращений функціонал, високу швидкість перевірки та якість оцінювання робіт порівняно з аналогами. Це забезпечує підвищення ефективності та задоволеності користувачів при використанні такого програмного продукту.

Висновки з даного дослідження і перспективи подальших розвідок у даному напрямі

1. Відповідно до проведеного дослідження встановлено, що ефективність та якість освітнього процесу суттєво залежить від коректної перевірки виконаних студентських робіт. Використання сучасних платформ для перевірки коректності виконаних студентських робіт є необхідністю в сучасних вищих навчальних закладах, оскільки їх застосування сприятиме підвищенню якості освіти та економії часу викладачів. Однак, існуючі технологічні рішення мають недостатній функціонал, це обмежує їх застосування в освітніх закладах із різними вимогами. Це слугувало передумовою в розробці програмного модуля для перевірки коректності виконання студентських робіт.

2. Для реалізації програмного модуля перевірки коректності виконання студентських робіт були використані сучасні технології та мови програмування. Серверна частина програмного модуля розроблена на основі C# та ASP.NET, це забезпечує стабільність, продуктивність та можливість масштабування. Використовуючи TypeScript та React розроблена клієнтська частина, це гарантує гнучкість, інтерактивність і зручний інтерфейс. Для роботи з базою даних використовувалася SQL. Це забезпечує ефективне управління збереженими даними. Середовищем виконання розробки слугує WEB-браузер. Застосування WEB-браузера дозволяє використовувати програмний модуль без необхідності встановлення додаткового програмного забезпечення. Програмний модуль не потребує інсталяції на комп'ютері користувача, а працює повністю у WEB-середовищі. Вимоги до обладнання мінімальні: процесор із тактовою частотою не менше 1.4 GHz, оперативна пам'ять обсягом не менше 1024 Mb та стабільне інтернет-з'єднання.

3. Під час тестування програмний модуль перевірки коректності виконання студентських робіт продемонстрував високу ефективність і має переваги над системами-аналогами, а саме, має розширений функціонал на 4 можливості: локалізація усіх мов світу дозволяє перевіряти роботи незалежно від мови їх написання; підтримка будь-якого формату файлів робіт студентів забезпечує універсальність використання; глибокий аналіз коректності виконання роботи студента сприяє більш точному оцінюванню результатів; гнучке налаштування параметрів програмного модуля дає можливість адаптувати процес перевірки відповідно до вимог викладачів. Розробка є ефективним інструментом для автоматизації перевірки студентських робіт, що значно спрощує процес оцінювання, забезпечує об'єктивність та зменшує витрати часу викладачів.

Література

1. Державна служба статистики України. URL: https://ukrstat.gov.ua/gend_rivnist/metadata_gr/04/04.htm (дата звернення 02.03.2025).

2. Об'єктивне оцінювання − фундаментальна задача освіти. URL: https://osvita.ua/school/method/6590/ (дата звернення 02.03.2025).

3. Переваги штучного інтелекту в освіті: Як штучний інтелект трансформує школи. URL: https://undetectable.ai/blog/uk/переваги-ай-в-освіті/ (дата звернення: 02.03.2025).

4. Grammarly . URL: https://www.grammarly.com/ (дата звернення: 02.03.2025).

5. Codio. URL: https://www.codio.com/ (дата звернення: 02.03.2025).

6. SonarCube. URL: https://www.sonarsource.com/ (дата звернення: 02.03.2025).

7. Прайс М. C# 10 і .NET 6. Сучасна кросплатформенна розробка. 6-те видання. Київ: Видавництво «Діалектика», 2023. 820 с.

8. Нейлор Лі. ASP.NET MVC з Entity Framework та CSS. Нью-Йорк: Apress, 2016. 608 с.

9. Вандеркам Д. Effective TypeScript: 62 Specific Ways to Improve Your TypeScript. Київ: O'Райлі Медіа, 2019. 261 с.

10. Фленаган Д. JavaScript: The Definitive Guide: Master the World's Most-Used Programming Language. Київ: O'Райлі Медіа, 2020. 706 с.

11. Річардсон К. Мікросервіси. Патерни розробки та рефакторингу / Пер. з англ. Київ: Видавництво «Діалектика», 2019. 432 с.

12. Гришанович Т. О., Глинчук Л. Я. Основи об'єктно-орієнтованого програмування. Луцьк: Волинський національний університет імені Лесі Українки, 2022. 120 с.

13. Савчук Т. О., Ваховська Л. М. Основи інформатики та обчислювальної техніки. Навчальні посібники Вінницького національного технічного університету. Мережні електронні видання. URL: https://web.posibnyky.vntu.edu.ua/fitki/1savchuk_osnovy_inform_obchisl_tehn/Topic5.html (дата звернення: 02.03.2025).

References

1. Derzhavna sluzhba statystyky Ukrainy. URL: https://ukrstat.gov.ua/gend_rivnist/metadata_gr/04/04.htm (data zvernennia 02.03.2025).

2. Obiektyvne otsiniuvannia − fundamentalna zadacha osvity. URL: https://osvita.ua/school/method/6590/ (data zvernennia 02.03.2025).

3.Perevahy shtuchnoho intelektu v osviti: Yak shtuchnyi intelekt transformuie shkoly. URL: https://undetectable.ai/blog/uk/perevahy-ai-v-osviti/ (data zvernennia: 02.03.2025).

4. Grammarly . URL: https://www.grammarly.com/ (data zvernennia: 02.03.2025).

5. Codio. URL: https://www.codio.com/ (data zvernennia 02.03.2025).
6. SonarCube. URL: https://www.sonarsource.com/ (data zvernennia: 02.03.2025).
7. Prais M. C# 10 i .NET 6. Suchasna krosplatformenna rozrobka. 6-te vydannia. Kyiv: Vydavnytstvo «Dialektyka», 2023. 820 s.
8. Neilor Li. ASP.NET MVC z Entity Framework ta CSS. Niu-York: Apress, 2016. 608 s.
9. Vanderkam D. Effective TypeScript: 62 Specific Ways to Improve Your TypeScript. Kyiv: ORaili Media, 2019. 261 s.
10. Flenahan D. JavaScript: The Definitive Guide: Master the Worlds Most-Used Programming Language. Kyiv: ORaili Media, 2020. 706 s.
11. Richardson K. Mikroservisy. Paterny rozrobky ta refaktorynhu / Per. z anhl. Kyiv: Vydavnytstvo «Dialektyka», 2019. 432 s.
12. Hryshanovych T. O., Hlynchuk L. Ya. Osnovy obiektno-oriientovanoho prohramuvannia. Lutsk: Volynskyi natsionalnyi universytet imeni Lesi Ukrainky, 2022. 120 s.
13. Savchuk T. O., Vakhovska L. M. Osnovy informatyky ta obchysliuvalnoi tekhniky. Navchalni posibnyky Vinnytskoho natsionalnoho tekhnichnoho universytetu. Merezhni elektronni vydannia. URL: https://web.posibnyky.vntu.edu.ua/fitki/1savchuk_osnovy_inform_obchisl_tehn/Topic5.html (data zvernennia: 02.03.2025).