Вінницький національний технічний університет
(повне найменування вищого навчального закладу)
Факультет інтелектуальних інформаційних технологій та автоматизації
(повне найменування інституту, назва факультету (відділення))
Кафедра комп'ютерних наук
(повна назва кафедри (предметної, циклової комісії))

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Інформаційна технологія моніторингу процесу навчання»

Виконав: студент 2-го курсу групи 2КН-23м
спеціальності 122 «Комп'ютерні науки»
(шифр і назва напряму підготовки, спеціальності)
Вишневський А.В.
(прізвище та ініціали)

Керівник: д.т.н., проф. каф. КН
Іванчук Я.В.
(прізвище та ініціали)
«05» 12 2024 р.

Опонент: д.т.н., проф. зав. каф. КСУ
Ковтун В. В.
(прізвище та ініціали)
«05» 12 2024 р.

Допущено до захисту
Завідувач кафедри КН
д.т.н., проф. Яровий А.А.
(прізвище та ініціали)
«06» 12 2024 р.

Вінниця ВНТУ - 2024 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти ІІ-й (магістерський)
Галузь знань – 12 «Інформаційні технології»
Спеціальність – 122 «Комп'ютерні науки»
Освітньо-професійна програма – «Системи штучного інтелекту»

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
д.т.н., проф. Яровий А.А.
«11» 10 2024 р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Вишневському Артуру В'ячеславовичу
(прізвище, ім'я, по батькові)

1. Тема роботи: Інформаційна технологія моніторингу процесу навчання
керівник роботи: д.т.н., проф. каф. КН, Іванчук Я.В.
затверджені наказом ВНТУ від «17» 09 2024 року № 310
2. Строк подання студентом роботи «11» 11 2024 року
3. Вихідні дані до роботи: час надання відповіді – секунди, кількість підряд правильно наданих відповідей, попередня кількість підряд правильно наданих відповідей, кількість підряд неправильно наданих відповідей, попередня кількість підряд неправильно наданих відповідей, рейтинг – 0…50, відсоток правильності надання відповіді – 0…100, мінімум коефіцієнту класифікації відповіді – 50, рівень впевненості – 1…5, мова програмування C#, фреймворк ASP.NET, середовище розробки Visual Studio 2022.
4. Зміст текстової частини: вступ, аналіз предметної області моніторингу процесу навчання, розробка інформаційної технології моніторингу процесу навчання, програмна реалізація інформаційної технології моніторингу процесу навчання, економічна частина, висновки, перелік використаних джерел, додатки.
5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень): схема функціонування інформаційної системи моніторингу прогресу навчання, схема алгоритму рейтингування при проходженні тесту, схема алгоритму формування рекомендацій, UML-діаграма послідовностей алгоритму рейтингування для тестів, вигляд головного меню розробленої програми, меню надання рекомендацій розробленої програми.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-3	Іванчук. Я.В., д.т.н., проф. каф. КН		
4	Адлер О.О., к.т.н., доцент каф. ЕПВМ		

7. Дата видачі завдання «02» 09 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз сучасних засобів моніторингу процесу навчання. Постановка задач дослідження	02.09.2024	20.09.2024	
2	Розробка інформаційної технології моніторингу процесу навчання	21.09.2024	05.10.2024	
3	Програмна реалізація інформаційної технології моніторингу процесу навчання	06.10.2024	28.10.2024	
4	Підготовка економічної частини	29.10.2024	01.11.2024	
5	Апробація та/або впровадження результатів дослідження	02.11.2024	04.11.2024	
6	Оформлення матеріалів до захисту МКР	05.11.2024	11.11.2024	

Студент _____ (підпис) Вишневський А.В. (прізвище та ініціали)

Керівник роботи _____ (підпис) Іванчук Я.В. (прізвище та ініціали)

АНОТАЦІЯ

УДК 004.78:37.09-047.36

Вишневський А.В. Інформаційна технологія моніторингу процесу навчання. Магістерська кваліфікаційна робота зі спеціальності 122 «Комп'ютерні науки», освітня програма «Системи штучного інтелекту». Вінниця: ВНТУ, 2024. 138 с.

Укр. мовою. Бібліографія: 29 назв; рисунків: 25; таблиць: 10.

Дана магістерська кваліфікаційна робота присвячена розробці інформаційної технології моніторингу процесу навчання. Були розглянуті та проаналізовані існуючі методи моніторингу навчального прогресу, серед яких було обрано метод, що поєднує аналітичні інструменти з адаптивним навчанням для підвищення ефективності контролю знань. Проаналізовано сучасні підходи до створення систем моніторингу навчання та обґрунтовано вибір архітектури з інтеграційними модулями, які забезпечують адаптивність системи до індивідуальних потреб користувачів. Програмна система реалізована мовою програмування C# у середовищі розробки Visual Studio 2022 з використанням технології .NET. Ефективність розробленої програми перевищує існуючі аналоги на 30%

Графічна частина складається з 6 плакатів.

У економічному розділі проведено аналіз конкурентоспроможності розробки, де розраховано суму витрат на розробку та виготовлення нового технічного рішення, яка складає 900534 гривень, також встановлено, що її термін окупності складає 0,67 року. Це свідчить про комерційну привабливість розробки, та може зацікавити потенційних інвесторів для фінансування її впровадження на ринок.

Ключові слова: моніторинг, процес навчання, оцінювання, рейтингування, коефіцієнт, інформаційна технологія, процес, алгоритм, код.

ABSTRACT

Vyshnevskyi A.V. Information technology for monitoring the learning process. Master's paper in specialty 122 «Computer Science», educational program «Artificial intelligence systems». Vinnytsia: VNTU, 2024. 138 pp.

In Ukrainian language. Bibliography: 29 titles; figures: 25; tables: 10.

This master's paper is devoted to the development of information technology for monitoring the learning process. Existing methods of monitoring learning progress were reviewed and analyzed, among which a method was chosen that combines analytical tools with adaptive learning to improve the effectiveness of knowledge control. Modern approaches to the creation of learning monitoring systems are analyzed and the choice of architecture with integration modules that ensure the adaptability of the system to the individual needs of users is substantiated. The software system is implemented in the C# programming language in the Visual Studio 2022 development environment using .NET technology. The efficiency of the developed program exceeds existing analogues by 30%.

The graphic part consists of 6 posters.

The economic section analyzes the competitiveness of the development, calculating the cost of developing and manufacturing a new technical solution, which amounts to UAH 900534, and establishes that its payback period is 0,67 years. This indicates the commercial attractiveness of the development and may be of interest to potential investors to finance its introduction to the market.

Keywords: monitoring, learning process, evaluation, rating, coefficient, information technology, process, algorithm, code.

ЗМІСТ

ВСТУП

Актуальність теми дослідження. Актуальність інформаційних технологій для моніторингу навчального процесу [1] зумовлена потребою в сучасних освітніх системах, які забезпечують об'єктивне, прозоре та своєчасне оцінювання знань студентів. З розвитком технологій та збільшенням кількості онлайн-курсів і дистанційного навчання традиційні методи контролю знань втрачають свою ефективність, оскільки не можуть повною мірою враховувати індивідуальні потреби студентів і не здатні забезпечити зворотний зв'язок для навчання в режимі реального часу. Інформаційні технології моніторингу дозволяють автоматизувати цей процес, забезпечуючи безперервний збір та аналіз даних про успішність студентів, що сприяє більш гнучкому та ефективному навчальному процесу.

Актуальність технології також зростає в умовах попиту на індивідуалізацію та персоналізацію навчання. Завдяки системам моніторингу викладачі можуть краще зрозуміти сильні та слабкі сторони кожного студента, що дозволяє їм адаптувати програму до конкретних потреб. Це особливо важливо в умовах великої кількості учнів, коли особистий моніторинг прогресу кожного учня стає занадто складним і ресурсномістким. Інформаційні технології моніторингу можуть вирішити цю проблему, забезпечивши необхідний індивідуальний підхід.

Крім того, актуальність моніторингу навчального процесу зростає у зв'язку з необхідністю підвищення якості освіти. Використання інформаційних технологій моніторингу дає можливість запровадити більш об'єктивні критерії оцінювання, які не залежать від людського фактору. Система надає можливість аналізувати результати тестування за різними показниками, такими як час відповіді, динаміка змін, рейтинг, що підвищує точність і справедливість оцінок. Це також мотивує студентів до більш активного навчання та дає їм чітке розуміння своїх досягнень.

Сучасні інформаційні технології моніторингу також сприяють інтеграції навчальних процесів із зовнішніми інформаційними ресурсами, такими як системи управління навчанням (СУН) [2] та інші освітні платформи. Це забезпечує безперервність навчального процесу та спрощує доступ до різноманітних ресурсів і матеріалів, що, в свою чергу, підвищує якість і доступність освітніх послуг.

Загалом, інформаційні технології моніторингу навчального процесу є необхідною складовою сучасної освіти, яка допомагає забезпечити ефективний процес навчання, сприяє індивідуальному підходу до студентів, підвищує об'єктивність та прозорість оцінювання, а також відповідає сучасним викликам та потребам інтеграції новітніх технологій у навчальний процес.

Зв’язок роботи з науковими програмами, планами, темами. Магістерська кваліфікаційна робота виконана відповідно до напряму наукових досліджень кафедри комп’ютерних наук Вінницького національного технічного університету 23 К1 «Розробка прикладних інтелектуальних інформаційних технологій та систем» та плану наукової та навчально-методичної роботи кафедри.

Мета та завдання дослідження. Метою магістерської кваліфікаційної роботи є підвищення ефективності процесу навчання студентів за допомогою розробки рекомендаційної системи, яка сприяє покращенню якості освіти шляхом надання рекомендацій щодо матеріалу який потребує подальшого вивчення.

Для досягнення мети роботи потрібно виконати такі задачі:

− провести аналіз існуючих методів та засобів моніторингу процесу навчання;

− визначити вимоги до інформаційної технології моніторингу процесу навчання;

− розробити математичну модель інформаційної технології моніторингу процесу навчання;

− розробити структуру інформаційної системи підтримки прийняття рішень процесу навчання студентів;

− виконати програмну реалізацію інформаційної технології моніторингу процесу навчання;

− провести тестування програмного забезпечення та проаналізувати отримані результати;

− провести розрахунок економічної ефективності науково-технічної розробки за можливого її застосування.

Об’єктом дослідження є процес тестування знань з моніторингом процесу вивчення матеріалу.

Предметом дослідження є програмні засоби для реалізації та платформи для розробки системи моніторингу процесу навчання.

Методи дослідження. У роботі використані такі методи наукових досліджень: методи та моделі подання знань, моделювання та проектування систем, метод системного аналізу, метод порівняльного аналізу, методи математичного моделювання та статистичного аналізу, методи тестування програмних засобів, методи об’єктно-орієнтованого програмування.

Наукова новизна одержаних результатів: розроблено метод незалежного оцінювання успішності студента, яка відрізняється від існуючих використанням удосконаленої математичної моделі тестового рейтингування, що дозволяє знизити похибку рейтингування при наданні випадково правильної відповіді, а також підвищити адаптивність системи до рівня підготовки студентів.

Практичне значення одержаних результатів полягає у такому:

1. Розроблено алгоритм функціонування інформаційної технології моніторингу процесу навчання;

2. Розроблено програмний засіб для моніторингу процесу навчання студентів.

Розроблений засіб може бути інтегрований у існуючі навчальні системи задля підвищення ефективності і якості навчального процесу у закладах вищої освіти, школах чи інших навчальних установах.

Достовірність теоретичних положень магістерської кваліфікаційної роботи підтверджується коректністю постановки завдання, точністю використання математичних моделей, використанням перевірних методів дослідження, експериментальними дослідженнями, тестуванням програмної реалізації інформаційної технології моніторингу процесу навчання. Адекватність розроблених математичних моделей підтверджується результатами експериментальних досліджень.

Особистий внесок магістранта. Усі результати, наведені у магістерській кваліфікаційній роботі, отримані самостійно. У роботах, опублікованих у співавторстві, автору належать такі результати: [3] – модифіікація коефіцієнту класифікації відповіді; [4] – введення коефіцієнту класифікації відповіді до рейтингування тестів; [5] – розробка удосконаленого алгоритму рейтингування тестів.

Апробація результатів роботи. Результати роботи були апробовані на міжнародній науково-практичній інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи 2025» [3], «Science and technology: problems, prospects and innovations. Proceedings of the 9th International scientific and practical conference» [4], «European scientific congress. Proceedings of the 5th International scientific and practical conference» [5]. Крім того, отримано акт про впровадження програмного засобу для моніторингу процесу навчання в «Асоціація IT компаній Вінниці».

Публікації. За результатами досліджень опубліковано 3 тези доповідей на науково-технічних конференціях [3 – 5], отримано свідоцтво про реєстрацію авторського права на твір [6].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Обґрунтування доцільності розробки системи моніторингу процесу навчання

Створення системи моніторингу навчання [7] є надзвичайно доречним кроком у розвитку сучасної освіти, оскільки дозволяє більш точно та об'єктивно оцінити рівень засвоєння студентами навчального матеріалу. На відміну від традиційного підходу, коли оцінка часто базується на кінцевих результатах і може не відображати реальної картини прогресу, система моніторингу забезпечує безперервне відстеження результатів. Це допомагає краще зрозуміти, як змінюються знання та навички студентів протягом навчального процесу. Для обґрунтування доцільності створення такої системи проаналізуємо наступні ключові моменти:

- актуальність проблеми моніторингу навчального прогресу;
- покращення ефективності навчального процесу;
- забезпечення індивідуального підходу до навчання;
- аналітичні можливості для викладачів та студентів;
- підвищення об'єктивності та точності оцінювання.

Актуальність проблеми моніторингу успішності навчання стає все більш помітною в сучасному освітньому середовищі. З розвитком цифрових технологій [8] та зміною підходів до викладання [9] виникає потреба в систематичному відстеженні знань та навичок студентів. Це дозволяє вчасно виявляти прогалини в навчанні та надавати своєчасну підтримку, що є важливим для підвищення загальної якості освіти.

Відсутність ефективних інструментів моніторингу призводить до зниження точності оцінки результатів навчання. У традиційній системі освіти часто використовуються лише підсумкові оцінки, які не завжди відображають реальний рівень знань учня. Це ускладнює виявлення викладачами та студентами

слабких місць у навчальному процесі, що, в свою чергу, впливає на ефективність подальшого розвитку.

Моніторинг академічної успішності дозволяє здійснювати постійний контроль за досягненнями студентів, що сприяє своєчасному коригуванню навчальної програми та індивідуальному підходу до кожного студента. Це дає можливість розробляти більш адаптивні методи навчання, пристосовані до потреб окремих студентів. Це особливо важливо в умовах дистанційного навчання, де рівень контролю за якістю навчального процесу знижується без належної системи моніторингу.

Згідно проведених досліджень «Використання інформаційних технологій в процесі навчання математики» [10] на прикладі учнів 7 класу загальноосвітньої школи, можна побачити позитивну динаміку покращення успішності учнів, яка виражається у підвищенні середнього балу класу. Для цього було обрано 2 класи, один з яких використовував комп’ютерні технології під час вивчення теми. Всього було проведено 21 урок.

Зі стартового рівня до 6 уроку помітний спад у експериментальній групі (рис. 1.1, 1.2). Висновок – учні вчяться працювати з комп’ютерною технологією (КТ), відбувається процес звикання до техніки, виробляються вміння користуватися системою.

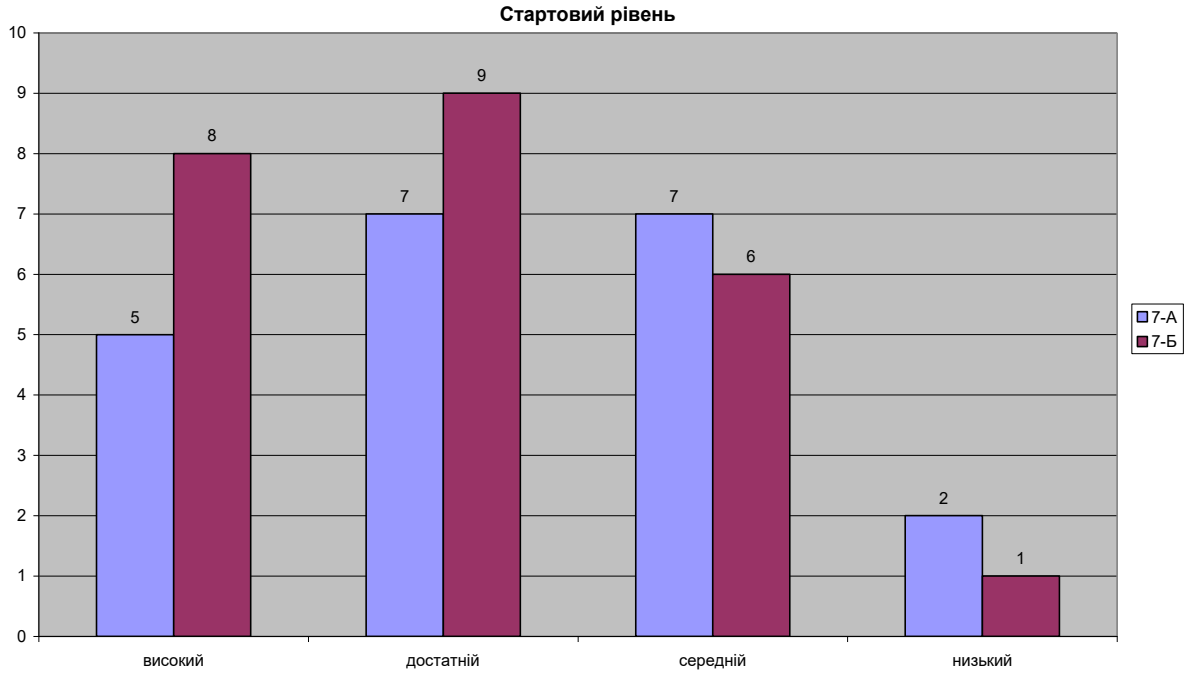

Рисунок 1.1 – Стовпчаста діаграма початкового рівня успішності учнів класу

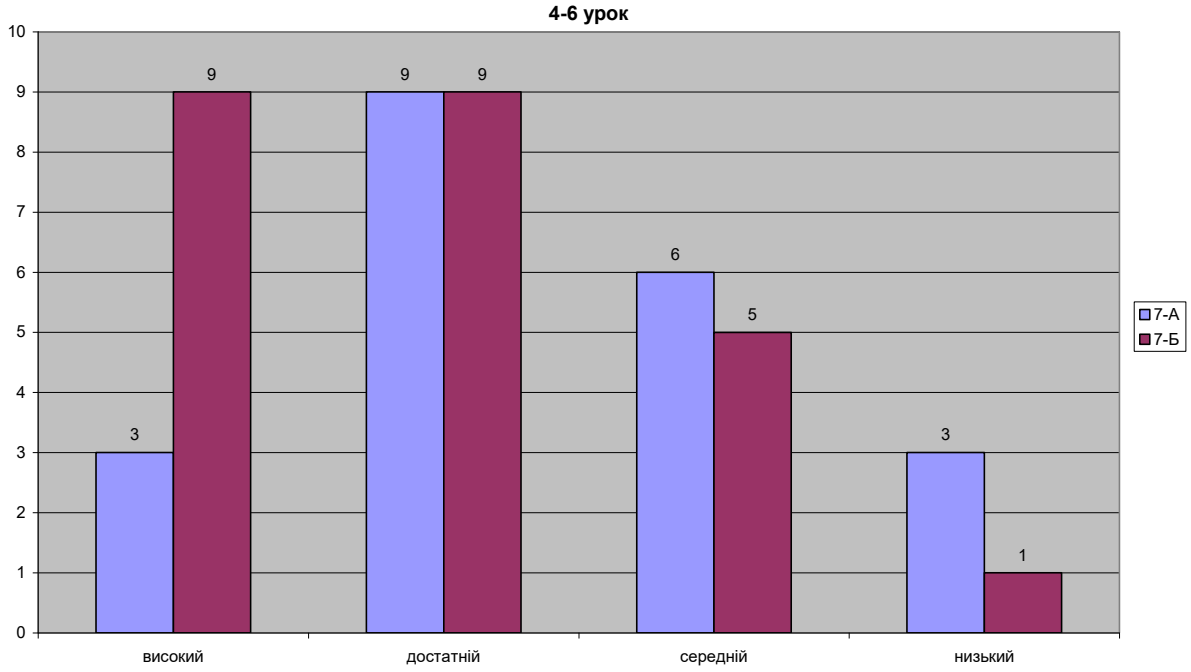

Рисунок 1.2 – Стовпчаста діаграма рівня успішності учнів класу до 6 уроку

Надалі очевидно, що експериментальна група при використанні широкої різнорівневої диференціації краще оволоділа навчальним матеріалом (рис. 1.3,

1.4). Підвищилась ефективність навчального процесу, особливо індивідуального навчання.

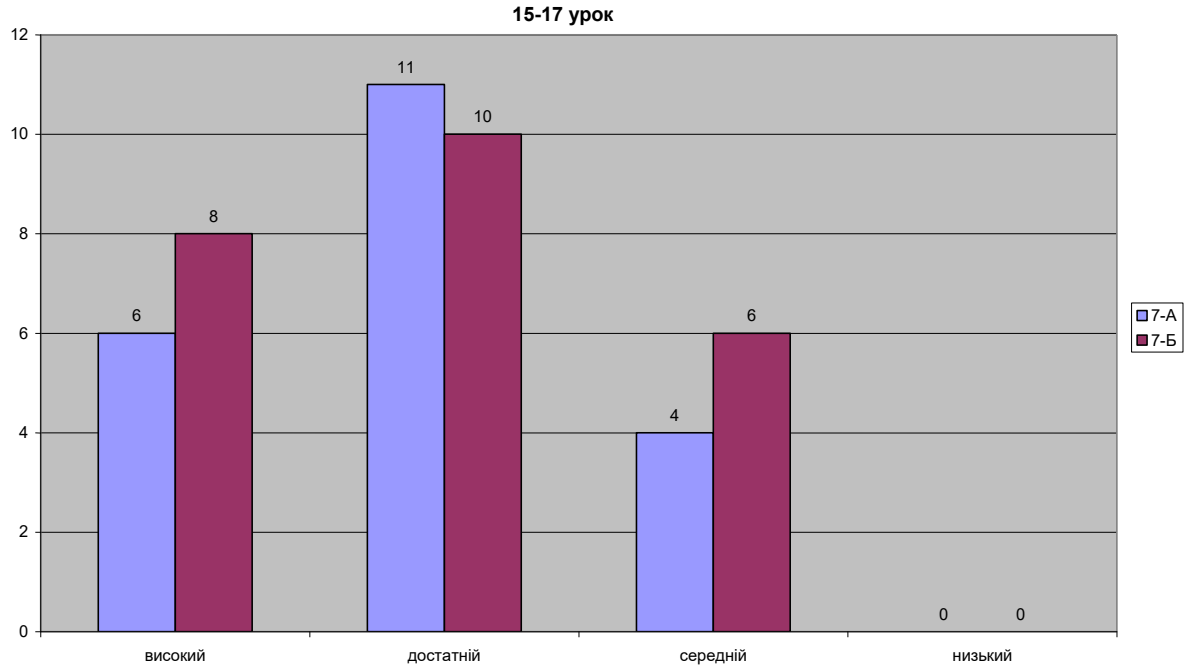

Рисунок 1.3 – Стовпчаста діаграма рівня успішності учнів класу до 17 уроку

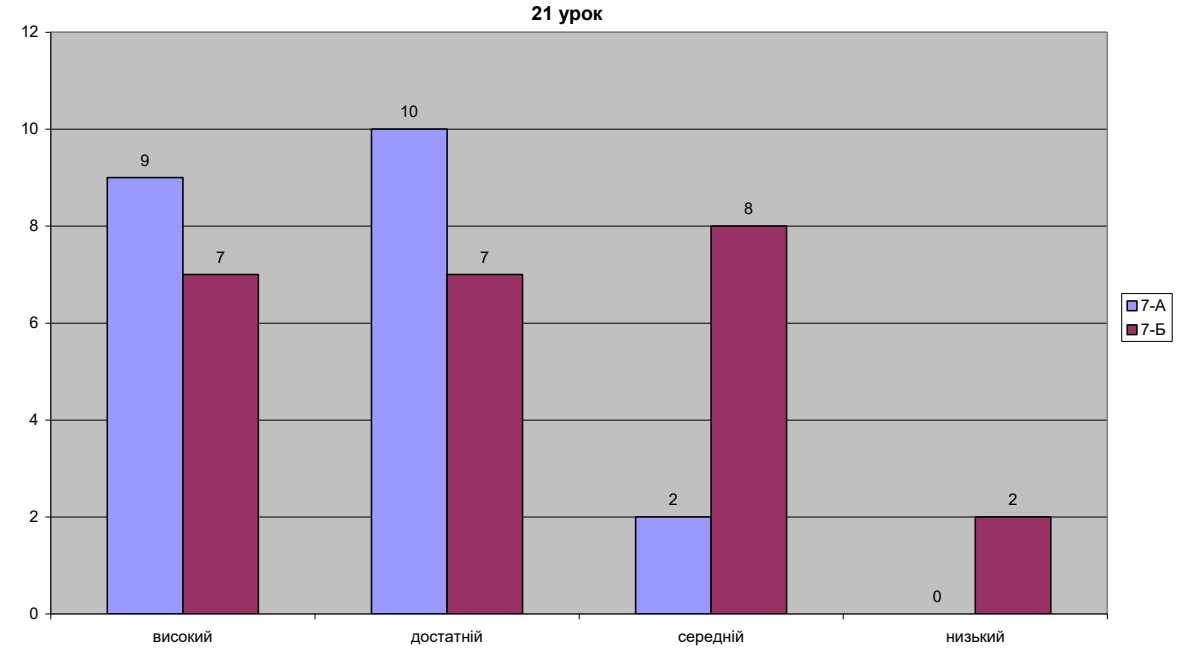

Рисунок 1.4 – Стовпчаста діаграма рівня успішності учнів класу до 21 уроку

Після аналізу динаміки (рис. 1.5, 1.6) успішності учнів на момент кожного контролю знань, стане очевидним, що використання КТ підвищує ефективність навчального процесу в цілому.

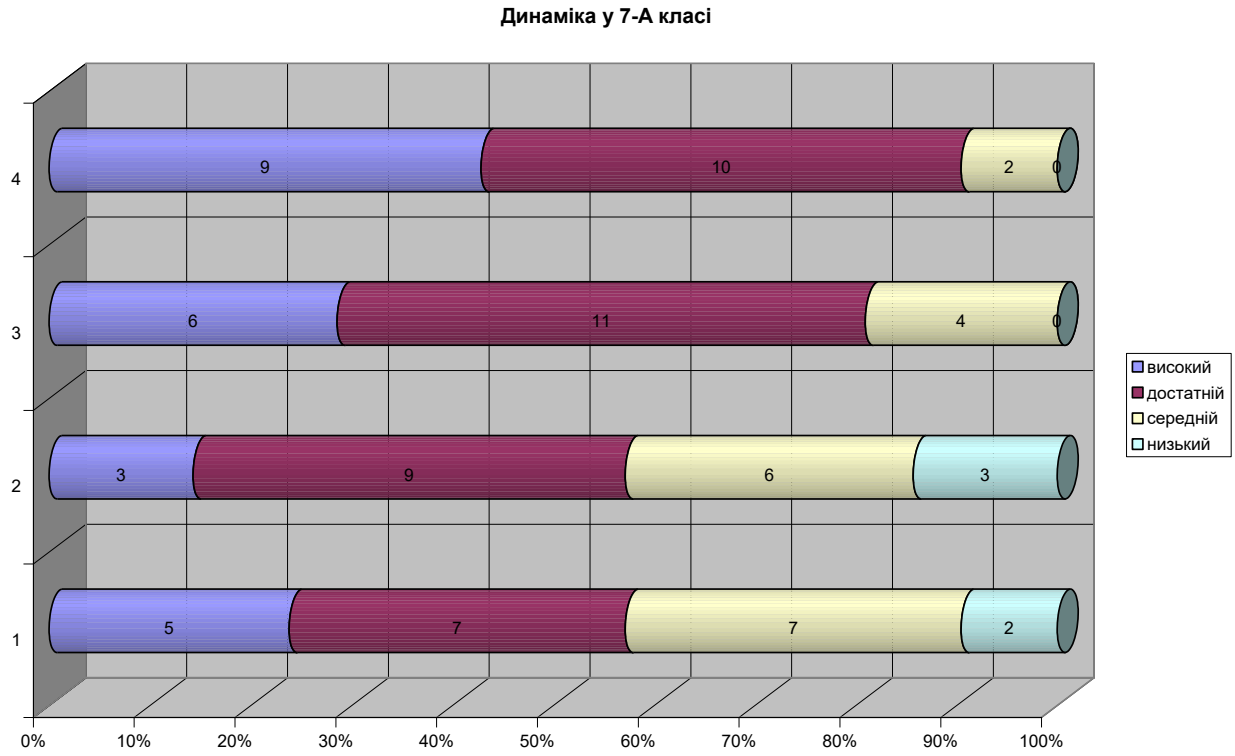

Рисунок 1.5 – Гістограма динаміки успішності класів для кожного етапу контролю у класі де використовувались КТ

Сучасні технології відкривають можливості для створення гнучких та інтерактивних систем моніторингу, які можуть автоматично аналізувати досягнення студентів та формувати звіти про їхній прогрес. Така система допоможе студентам розвинути навички самоконтролю, а викладачам – краще зрозуміти індивідуальні особливості студентів. Це є важливим фактором підвищення якості освіти та вдосконалення навчального процесу.

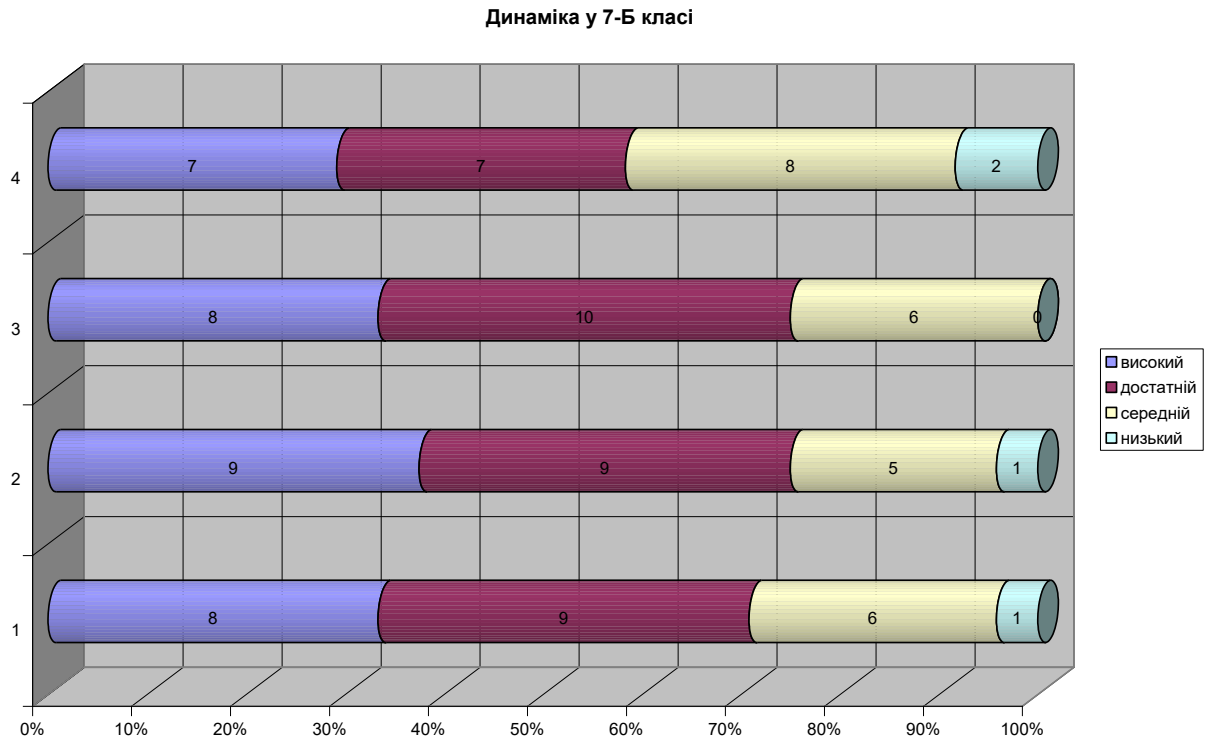

Рисунок 1.6 – Гістограма динаміки успішності класів для кожного етапу контролю у класі де не використовувались КТ

Загалом, актуальність моніторингу успішності важко переоцінити, оскільки він є ключовим елементом у забезпеченні ефективної та результативної освіти. Інструменти моніторингу сприяють побудові прозорого та об'єктивного освітнього процесу, допомагають адаптувати навчальні програми до конкретних потреб учнів.

Підвищення ефективності навчального процесу є ключовою метою сучасної освіти, особливо в умовах стрімкого розвитку інформаційних технологій. Ефективний навчальний процес спрямований на максимальне засвоєння студентами знань та розвиток необхідних навичок і компетенцій. Це можливо лише тоді, коли студенти отримують своєчасний зворотній зв'язок і можуть оцінити свій прогрес в режимі реального часу.

Одним з найважливіших елементів підвищення ефективності навчання є використання систем моніторингу, які дозволяють здійснювати безперервний контроль успішності студентів. За допомогою таких систем викладачі можуть

аналізувати, як студенти навчаються, і коригувати свої методи викладання на основі отриманих результатів. Це сприяє більш персоналізованому підходу до навчання, що дозволяє викладачам ефективніше працювати з кожним студентом.

Важливим аспектом є те, що системи моніторингу дозволяють не лише оцінити досягнення студентів, але й допомогти їм краще організувати свій навчальний процес. Студенти можуть самостійно відстежувати свій прогрес, виявляти слабкі місця та працювати над ними, що підвищує їхню відповідальність за власне навчання. Це, в свою чергу, сприяє розвитку навичок самоорганізації, що є важливим фактором підвищення загальної ефективності навчання.

Додатковим інструментом підвищення ефективності є використання аналітичних можливостей таких систем, які дозволяють отримувати детальні звіти про успішність як окремих студентів, так і цілих груп. Ці дані допомагають виявити загальні проблеми у викладанні певних тем чи модулів і вчасно ввести необхідні зміни до навчальної програми. Це покращує не лише індивідуальний рівень знань, а й загальну якість освіти в закладі.

Загалом, підвищення ефективності навчального процесу за допомогою систем моніторингу дозволяє створити динамічне, гнучке та індивідуально орієнтоване освітнє середовище. Це не лише сприяє глибшому засвоєнню знань студентами, але й робить навчальний процес більш прозорим, контрольованим та ефективним. Як наслідок, це позитивно впливає на загальний рівень освіти та підготовки фахівців у різних галузях.

Забезпечення індивідуального підходу до навчання є одним з ключових факторів успішного засвоєння знань студентами. Кожен студент має власний темп навчання, інтереси, сильні та слабкі сторони, що робить універсальні методи навчання менш ефективними. Системи моніторингу навчання допомагають врахувати ці індивідуальні особливості, дозволяючи адаптувати навчальний процес до потреб кожного студента.

Ці системи можуть відстежувати прогрес кожного учня в режимі реального часу і надавати йому зворотній зв'язок про його успіхи та недоліки. Викладачі

можуть використовувати ці дані для коригування навчальної програми, пропонуючи додаткові завдання або матеріали учням, які відстають, і навпаки, надаючи складніші завдання тим, хто вчиться швидше. Це дозволяє уникнути ситуацій, коли одні учні відчувають себе перевантаженими, а інші не отримують достатньо інтелектуальних викликів.

Індивідуальний підхід також допомагає підвищити мотивацію студентів. Коли учні бачать, що їхній прогрес активно відстежується і вони отримують підтримку відповідно до своїх потреб, це сприяє підвищенню їхнього інтересу до навчання. Крім того, студенти мають змогу самостійно аналізувати свої результати та вносити корективи у свій навчальний план, що спонукає їх брати на себе більшу відповідальність за власне навчання.

Системи моніторингу також сприяють більш персоналізованій взаємодії між студентами та викладачами. Викладачі можуть надавати індивідуальні рекомендації та зворотній зв’язок, що враховує особисті досягнення кожного студента. Це робить процес навчання більш гнучким та адаптованим до потреб студента, дозволяючи кожному розвиватися у власному темпі та здобувати знання у найбільш ефективний спосіб.

Таким чином, індивідуальний підхід до навчання, підкріплений сучасними системами моніторингу, є важливим фактором підвищення якості освіти. Він допомагає краще враховувати потреби кожного студента, сприяє розвитку самостійності та відповідальності, підвищує загальний рівень академічної успішності.

Аналітичні можливості систем моніторингу навчання є потужним інструментом як для викладачів, так і для студентів. Вчителі мають змогу глибоко аналізувати прогрес кожного учня, виявляти тенденції в навчанні та прогнозувати можливі проблеми в навчанні. Вони бачать, які теми є найскладнішими для групи в цілому або для окремих студентів, і відповідно коригують навчальну програму або методи викладання.

Для студентів аналітичні інструменти також є величезною перевагою. Вони можуть переглядати свої результати в часі, бачити, де вони роблять

прогрес, а де відстають. Це дозволяє їм краще зрозуміти власні сильні та слабкі сторони і спланувати час для повторення або вивчення певних матеріалів. Такий підхід допомагає учням стати більш самостійними в процесі навчання і брати на себе відповідальність за свої результати.

Крім того, аналітичні можливості дозволяють порівнювати результати окремих учнів або цілих груп між собою. Викладачі можуть виявляти групи ризику – студентів, які можуть не встигати за навчальним процесом – і надавати їм додаткову допомогу. З іншого боку, системи можуть автоматично визначати найуспішніших учнів і пропонувати їм індивідуальні виклики або додаткові завдання для розвитку їхнього потенціалу.

Аналітичні системи також забезпечують ретроспективний аналіз, що дозволяє викладачам оцінити ефективність методів навчання в часі. Викладач може бачити, наскільки успішними були певні педагогічні підходи, і планувати майбутні курси на основі цих даних. Це дозволяє постійно вдосконалювати навчальний процес та адаптувати його до реальних потреб студентів.

Загалом, аналітичні можливості систем моніторингу надають викладачам і студентам цінні дані для прийняття рішень і планування навчального процесу. Вони сприяють гнучкості та адаптивності, дозволяючи кожному учаснику навчального процесу максимально використовувати свій потенціал і ресурси.

Підвищення об'єктивності та точності оцінювання є однією з основних переваг сучасних систем моніторингу навчання. Традиційні методи оцінювання, такі як письмові іспити чи усні відповіді, часто містять елемент суб'єктивності з боку викладача, що може вплинути на точність оцінки. Використання автоматизованих систем моніторингу дозволяє мінімізувати цей людський фактор і забезпечити більш об'єктивне оцінювання знань студентів. Системи, засновані на алгоритмах, однаково оцінюють всі відповіді, що виключає можливість упередженості або випадкових помилок.

Крім того, сучасні інформаційні технології можуть аналізувати широкий спектр показників для оцінювання прогресу студента. Це не лише результати тестів, але й активність під час навчання, частота виконання завдань, швидкість

засвоєння матеріалу та інші показники. Такий підхід дозволяє отримати більш повну і точну картину знань студента, ніж оцінювання, яке базується лише на підсумкових іспитах. Збір і аналіз різних даних допомагає об'єктивніше визначити рівень компетентності студента.

Системи моніторингу також забезпечують прозорість процесу оцінювання. Студенти мають доступ до власних результатів у реальному часі та можуть зрозуміти, як була отримана та чи інша оцінка. Це дозволяє уникнути непорозумінь і підвищує рівень довіри студентів до системи оцінювання. Прозорість сприяє тому, що студенти чітко розуміють свої помилки і можуть оперативно працювати над їх виправленням, що додатково підвищує якість навчання.

Точність оцінювання також підвищується завдяки можливості проведення багаторазових тестувань. На відміну від одноразових іспитів, системи моніторингу дозволяють проводити регулярні перевірки знань з проміжними результатами. Це зменшує стрес для студентів і дає можливість точно відстежити, як змінюється їхній рівень знань з часом. Такі частіші, але менш стресові перевірки дозволяють виявити прогалини в знаннях на ранніх етапах і своєчасно надати допомогу.

Отже, інформаційні технології моніторингу навчання сприяють підвищенню точності та об'єктивності оцінювання завдяки автоматизації, аналізу комплексних даних, прозорості процесу і можливості регулярного контролю знань. Це, в свою чергу, покращує загальну якість навчального процесу та стимулює студентів до постійного розвитку.

1.2 Аналіз існуючих методів і засобів моніторингу процесу навчання

Існує багато інструментів і систем для моніторингу навчання, кожна з яких має свої особливості та функціональність. Серед найпоширеніших – Moodle [11], Google Classroom [12], Blackboard [13], Canvas [14] та Microsoft Teams [15]. Кожна з них пропонує унікальні можливості для моніторингу прогресу

студентів, але також має свої обмеження. Для порівняння ми використаємо чотири основні критерії: функціонал моніторингу, інтеграція з іншими сервісами, зручність використання та аналітичні можливості.

Функціональність моніторингу є ключовим критерієм, оскільки він визначає, наскільки добре система може відстежувати прогрес студентів. Moodle пропонує широкий спектр інструментів моніторингу, включаючи детальну аналітику оцінок, відстеження активності та персоналізовані звіти. Google Classroom (рис. 1.7, 1.8) має більш обмежену функціональність у цьому аспекті, зосереджуючись на загальному огляді успішності студентів. Blackboard (рис. 1.9) та Canvas (рис. 1.10) пропонують схожі функції з акцентом на детальному відстеженні оцінок та активності, але Canvas надає кращі можливості для персоналізованого моніторингу.

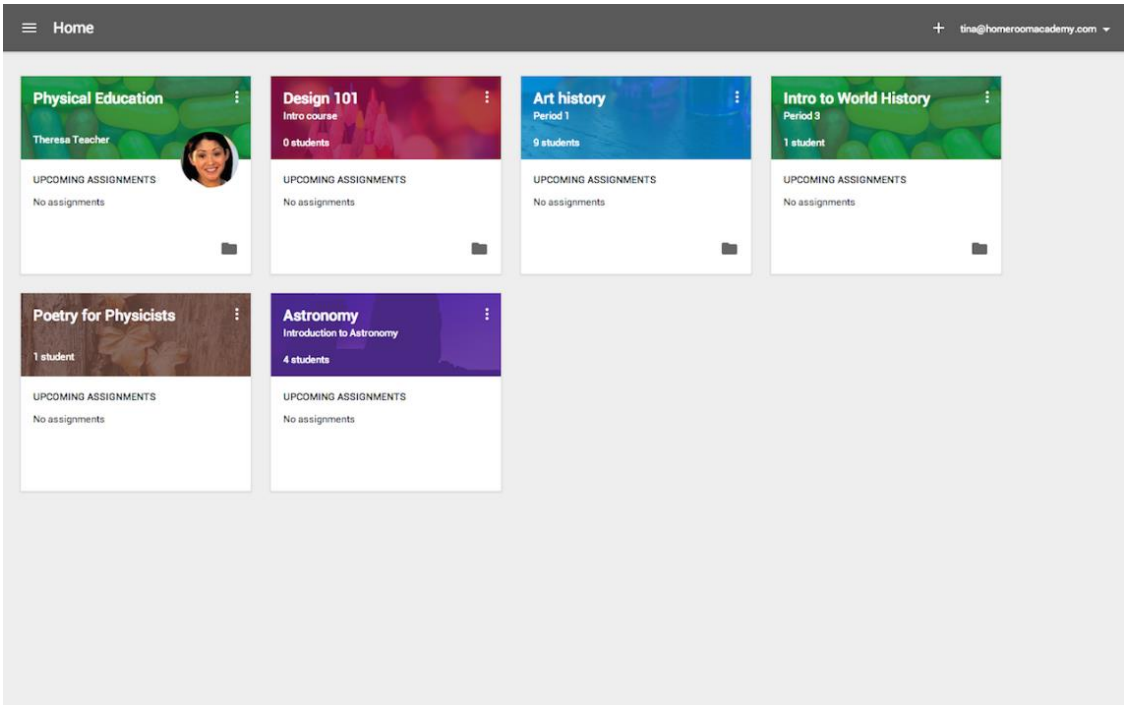

Рисунок 1.7 – Приклад інтерфейсу сторінки із класами Google Classroom

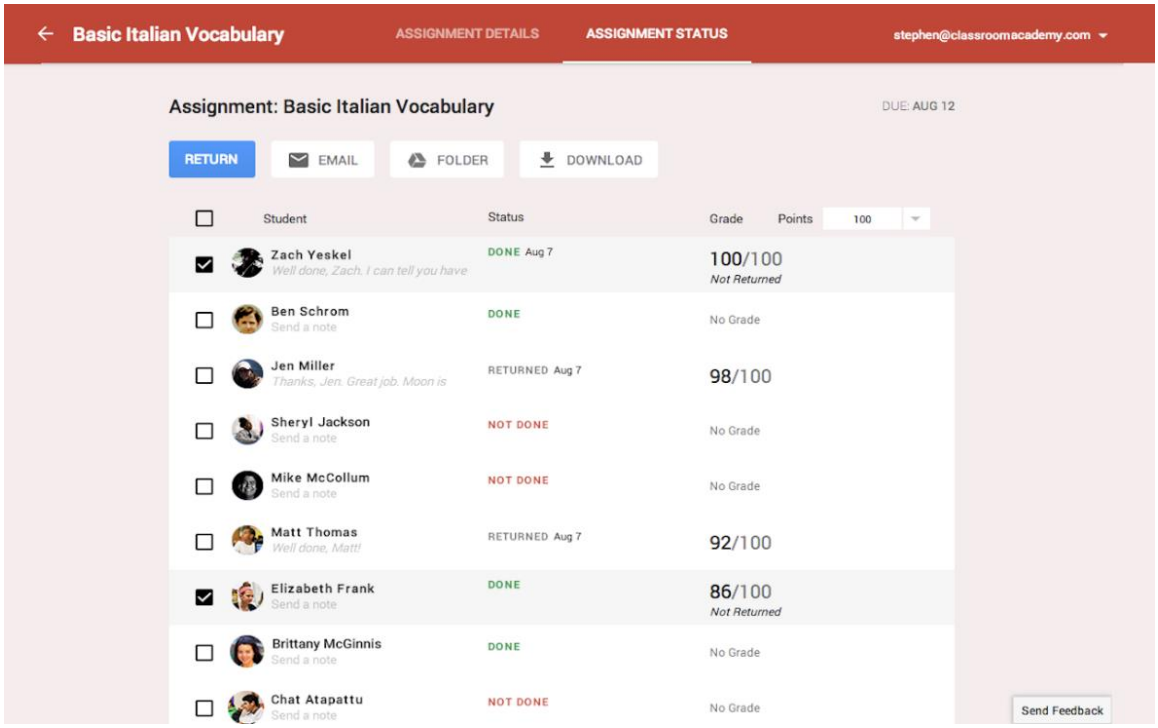

Рисунок 1.8 – Приклад інтерфейсу сторінки зі списком студентів класу Google Classroom

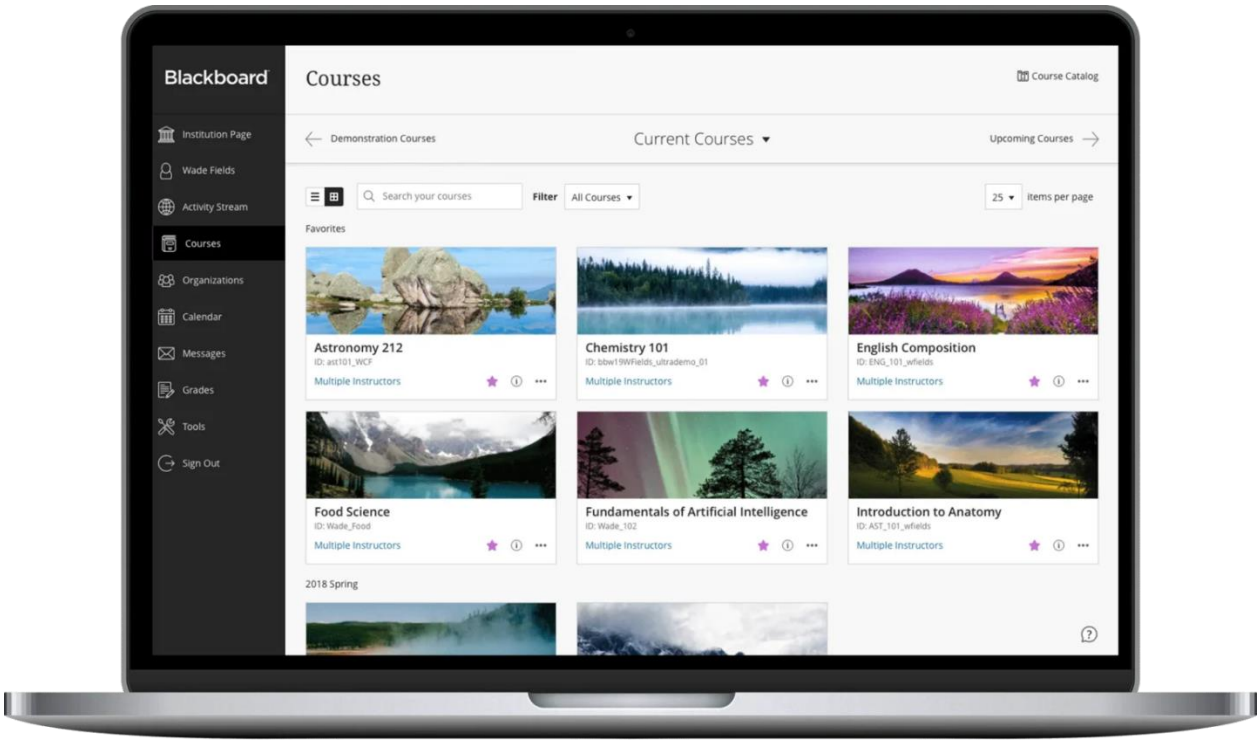

Рисунок 1.9 – Приклад інтерфейсу Blackboard

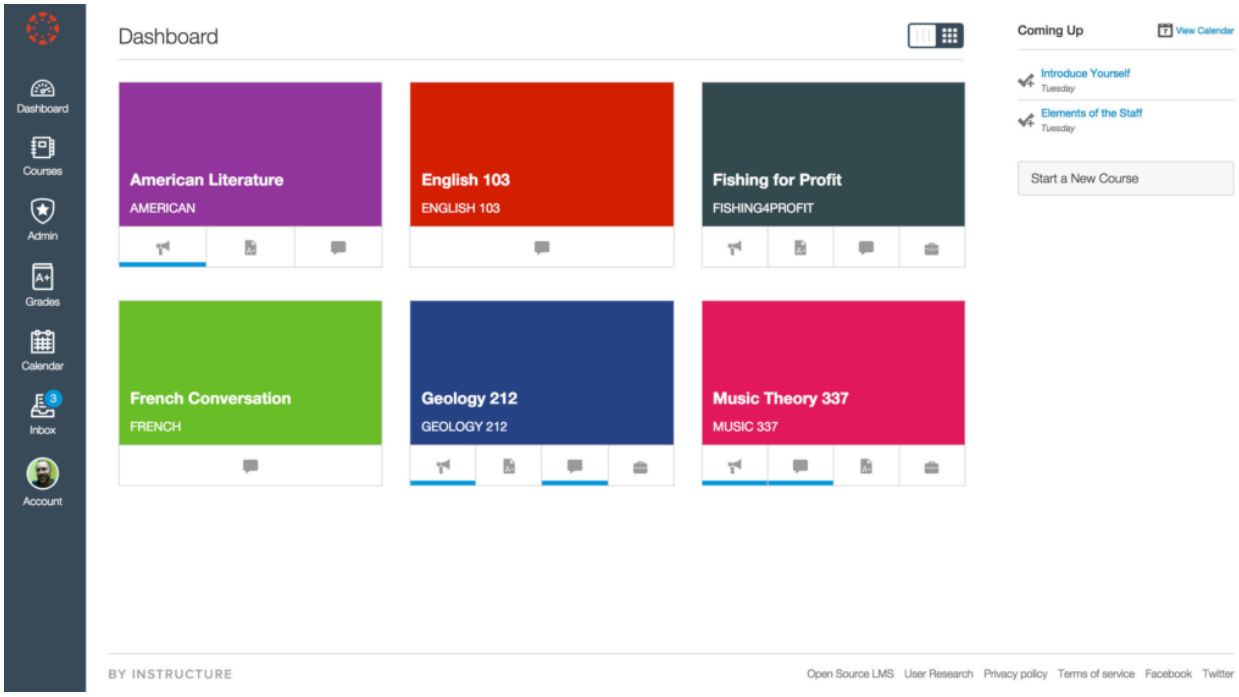

Рисунок 1.10 – Приклад інтерфейсу Canvas

Другий критерій – інтеграція з іншими сервісами, що дозволяє системам легко взаємодіяти з різними інструментами та платформами, такими як хмарні сервіси, додатки для спільної роботи та інші освітні ресурси. Moodle має тісну інтеграцію з численними плагінами та сторонніми сервісами, що робить його гнучким інструментом для освітян. Google Classroom тісно інтегрований з іншими сервісами Google, такими як Google Drive і Google Docs, що робить його простим у використанні для користувачів Google. Blackboard і Canvas також пропонують можливість інтеграції з численними освітніми ресурсами, але Canvas пропонує більшу гнучкість завдяки своїй відкритій архітектурі API для сторонніх додатків.

Простота використання є важливим фактором для користувачів будь-якої системи управління навчанням. Moodle (рис. 1.11, 1.12), незважаючи на свою багатофункціональність, може здатися складним для новачків через велику кількість налаштувань і параметрів. Google Classroom, з іншого боку, дуже простий у використанні, і багато користувачів цінують його за інтуїтивно зрозумілий інтерфейс, особливо якщо вони вже знайомі з іншими сервісами Google. Blackboard, хоча і є багатофункціональним, має дещо старомодний

інтерфейс, який може бути менш зручним для нових користувачів. Canvas забезпечує баланс між функціональністю та зручністю використання, з інтуїтивно зрозумілим інтерфейсом та широкими можливостями кастомізації.

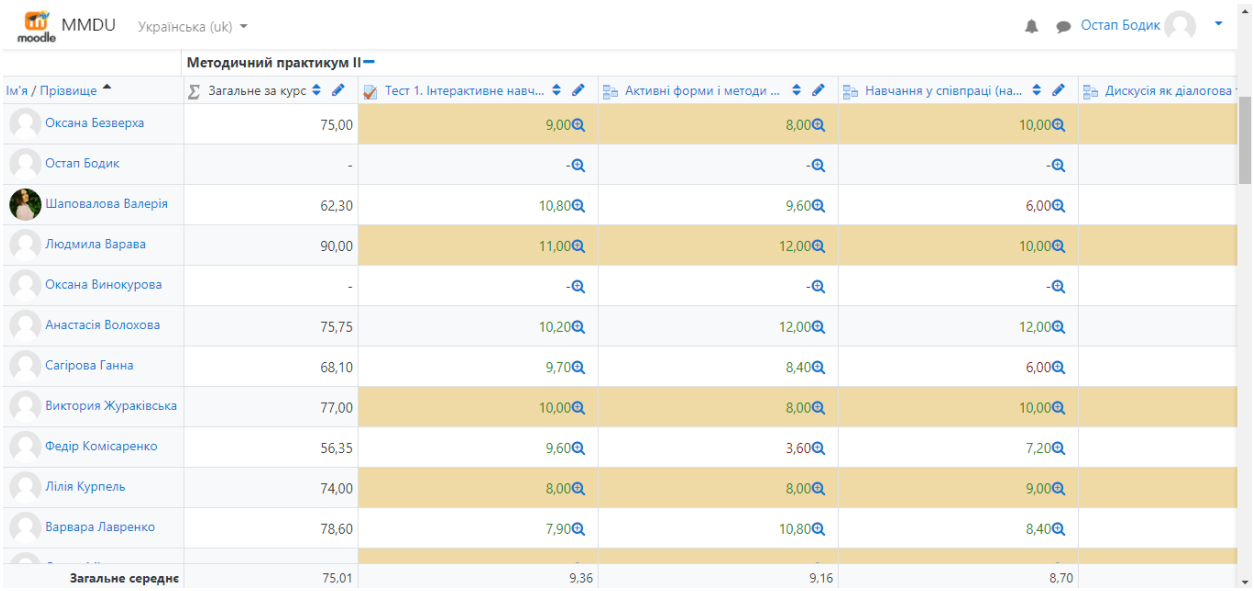

Рисунок 1.11 – Приклад інтерфейсу списку студентів Moodle

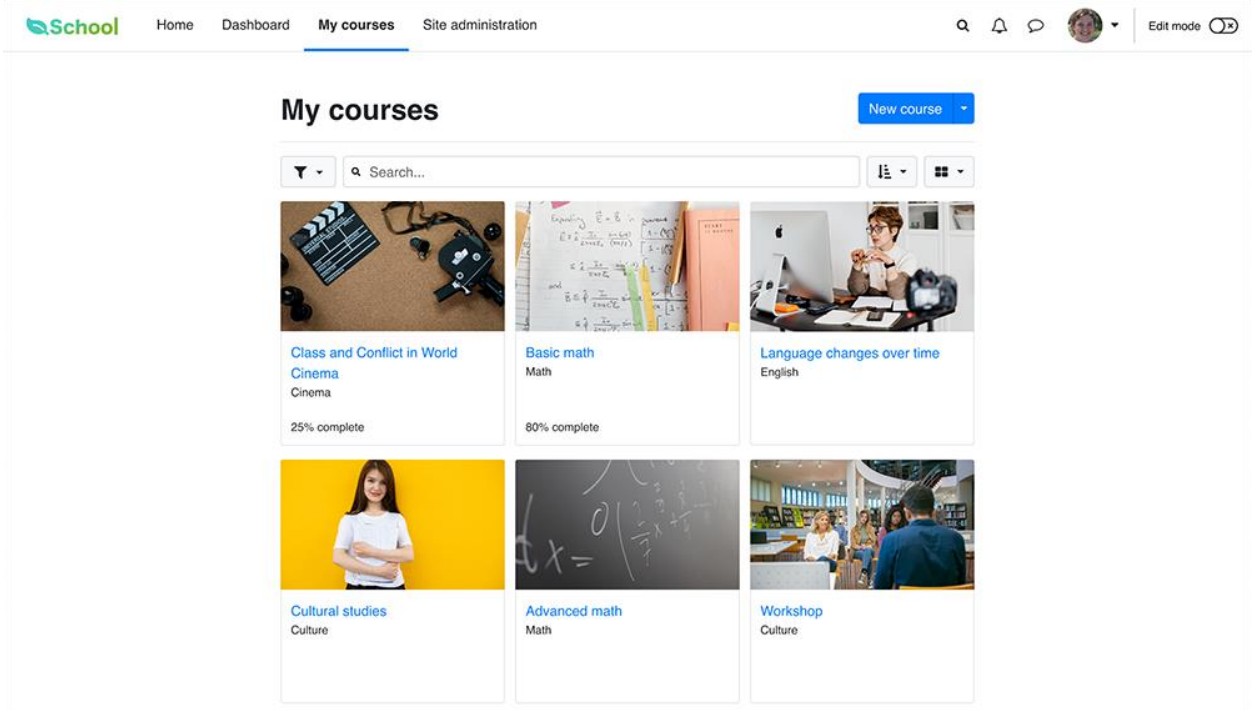

Рисунок 1.12 – Приклад інтерфейсу списку класів Moodle

Аналітичні можливості визначають здатність системи надавати користувачам глибоке розуміння процесу навчання за допомогою звітів, графіків та прогнозування ефективності. Moodle пропонує розширені аналітичні інструменти, що дозволяють викладачам створювати детальні звіти про прогрес кожного студента. Google Classroom, хоча і надає базові звіти про прогрес, має обмежені можливості для поглибленого аналізу. Blackboard пропонує потужні аналітичні інструменти з можливостями візуалізації даних, а Canvas відомий своїми передовими інструментами прогнозування та аналізу, що дозволяє викладачам швидко і точно оцінювати прогрес студентів і передбачати потенційні проблеми в навчанні.

Порівняно з цими системами, Moodle вирізняється своєю гнучкістю та широким спектром інструментів моніторингу та аналізу, але його складність може бути недоліком для деяких користувачів. Google Classroom, хоча і має обмежену функціональність для моніторингу та аналізу, компенсує це своєю простотою та інтеграцією з екосистемою Google. Blackboard – надійний інструмент з потужними аналітичними функціями, але його інтерфейс потребує оновлення для покращення зручності використання. Canvas виявляється найбільш збалансованою системою, що поєднує інтуїтивно зрозуміле використання з розширеними аналітичними можливостями.

Microsoft Teams (рис. 1.13) – популярна платформа для організації навчального процесу, особливо в контексті дистанційного навчання. Її функціонал охоплює різноманітні можливості, включаючи відеоконференції, організацію груп для курсів чи семінарів, інтеграцію з електронною поштою та обмін файлами. Однак, якщо розглядати Teams з точки зору моніторингу прогресу навчання, то платформа не пропонує повного набору інструментів для оцінювання студентів або відстеження їхнього прогресу в режимі реального часу. Основні можливості для моніторингу – це проведення тестів через інтеграцію з Microsoft Forms або перегляд активності студентів через відвідування лекцій та виконання завдань.

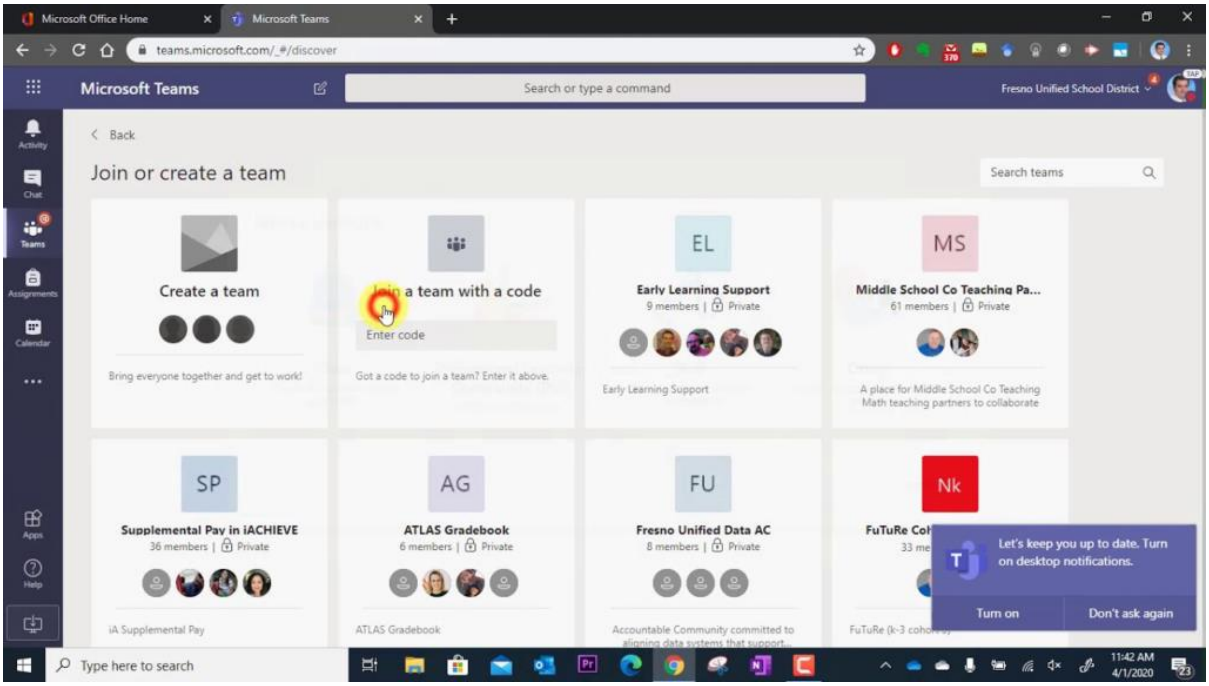

Рисунок 1.13 – Приклад інтерфейсу Microsoft Teams

Інтеграція Microsoft Teams з іншими сервісами є однією з ключових переваг. Платформа безперешкодно працює з іншими продуктами Microsoft, такими як OneDrive, OneNote, Outlook, SharePoint та Office 365, що значно спрощує організацію та управління навчальними ресурсами. Крім того, доступна інтеграція зі сторонніми сервісами та інструментами аналітики, що може розширити можливості моніторингу. Це дозволяє використовувати додаткові ресурси для більш глибокого аналізу успішності учнів, але базових можливостей моніторингу в Teams недостатньо для повноцінного аналізу без сторонніх рішень.

Простота використання Microsoft Teams – одна з головних причин її популярності. Інтерфейс досить інтуїтивно зрозумілий і не вимагає особливих зусиль для освоєння як студентами, так і викладачами. Навчальні матеріали та комунікація легко організовуються в окремі канали, а функції швидкого доступу до файлів, планування зустрічей та спільної роботи роблять процес організації навчання зручним. Однак на ранніх етапах використання може виникнути крива навчання через велику кількість функцій, які можуть здатися складними для новачків.

Аналітичні можливості Microsoft Teams з точки зору моніторингу навчання досить обмежені. Платформа надає базову аналітику, таку як відвідуваність студентів, виконання завдань та активність у чаті, але не має розширених інструментів для детального відстеження продуктивності або поглибленого аналізу результатів навчання. Розширення цих можливостей вимагає інтеграції із зовнішніми аналітичними платформами, такими як Power BI або іншими інструментами для роботи з великими даними, що може ускладнити моніторинг для викладачів, які не мають досвіду роботи з такими системами.

Таким чином, Microsoft Teams має певні переваги як платформа для організації навчального процесу та базового моніторингу, але її можливості в контексті поглибленого аналізу результатів навчання та комплексного моніторингу є обмеженими. Для того, щоб повністю відповідати вимогам сучасної системи моніторингу навчання, необхідна інтеграція з додатковими інструментами.

Таким чином, вибір системи моніторингу навчання залежить від конкретних потреб навчального закладу та його користувачів. Якщо потрібна багатофункціональна система з потужними інструментами аналітики, Moodle та Blackboard можуть бути хорошими варіантами. Якщо важлива простота використання та швидка інтеграція з іншими інструментами, кращим вибором буде Google Classroom. Canvas підійде тим, хто шукає систему, яка поєднує функціональність з простотою використання та аналітичними можливостями. Порівняння усіх описаних систем у даному розділі наведене в таблиці 1.1.

Таблиця 1.1 – Порівняльний аналіз різних систем моніторингу навчання за шкалою від 1 до 5

Система	Функціональність	Інтеграція з іншими сервісами	Зручність використання	Аналітичні можливості
Moodle	5	5	3	5
Google Classroom	3	5	5	2
Blackboard	4	4	3	4
Canvas	4	4	4	5
Microsoft Teams	4	5	4	3

1.3 Визначення вимог до інформаційних систем моніторингу процесу навчання

Визначаючи вимоги до системи моніторингу навчання на основі попереднього аналізу, можна сформулювати наступні критерії:

1. Розширені аналітичні можливості: Система повинна надавати потужні інструменти для збору й аналізу даних про успішність студентів. Це дозволить викладачам і адміністраторам виявляти проблемні зони, тенденції та потенційні шляхи для покращення якості навчання.

2. Інтеграція з іншими платформами: Важливо, щоб система могла легко взаємодіяти з іншими навчальними сервісами та платформами, як-от відеоконференції, сховища для матеріалів, бібліотеки тощо. Це забезпечить зручність і універсальність у користуванні, а також підвищить ефективність роботи викладачів і студентів.

3. Зручність використання: Користувачі, включаючи як викладачів, так і студентів, повинні легко взаємодіяти з системою. Інтуїтивно зрозумілий

інтерфейс і проста навігація знижують час на адаптацію та зменшують ймовірність технічних проблем.

4. Гнучкість налаштувань: Система повинна надавати можливість налаштовувати модулі відповідно до потреб окремих навчальних закладів або викладачів. Це дозволить адаптувати інструменти моніторингу для різних дисциплін, форм навчання і типів оцінювання.

1.4 Висновок до розділу 1

Створення інформаційної технології моніторингу навчання є актуальним завданням у сучасному освітньому середовищі, оскільки вона здатна забезпечити глибокий аналіз прогресу студентів і допомогти виявити їхні сильні та слабкі сторони. Ця технологія повинна базуватися на надійних аналітичних інструментах, які дозволяють збирати та обробляти дані про навчальний процес, а також надавати рекомендації для покращення якості навчання. Важливим аспектом є також можливість адаптації системи до специфічних потреб різних навчальних закладів та дисциплін.

Критерії, які будуть визначати ефективність системи моніторингу, включають аналітичні можливості, інтеграцію з іншими освітніми платформами, зручність використання та доступність. Наявність потужних аналітичних функцій дозволить викладачам отримувати детальні звіти про результати студентів, а інтеграція з існуючими освітніми технологіями забезпечить безперервний обмін даними між різними системами. Це, в свою чергу, створить більш узгоджене навчальне середовище.

Узагальнюючи, можна стверджувати, що розробка інформаційної технології моніторингу навчання є важливим кроком у вдосконаленні освітнього процесу. Визначені критерії для системи повинні бути спрямовані на створення інструменту, який не лише надасть можливість об'єктивно оцінювати знання студентів, але й підтримуватиме їх у процесі навчання.

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ МОНІТОРИНГУ ПРОГРЕСУ НАВЧАННЯ

2.1 Розробка моделі інформаційної системи моніторингу процесу навчання

Розробка моделі інформаційної технології моніторингу навчання вимагає системного підходу, який враховує всі аспекти освітнього процесу. Основною метою цієї моделі є забезпечення постійного контролю та аналізу навчальних досягнень студентів із використанням автоматизованих засобів обробки даних. Модель повинна охоплювати як кількісні, так і якісні показники успішності, що дозволить надавати викладачам і студентам повноцінний зворотний зв'язок. Це забезпечить можливість вчасного коригування навчального процесу для досягнення кращих результатів.

Інформаційна технологія моніторингу навчання буде мати клієнт-серверну архітектуру (рис. 2.1), що включатиме в себе:

- користувацький інтерфейс;
- система управління базами даних (СУБД);
- серверна частина.

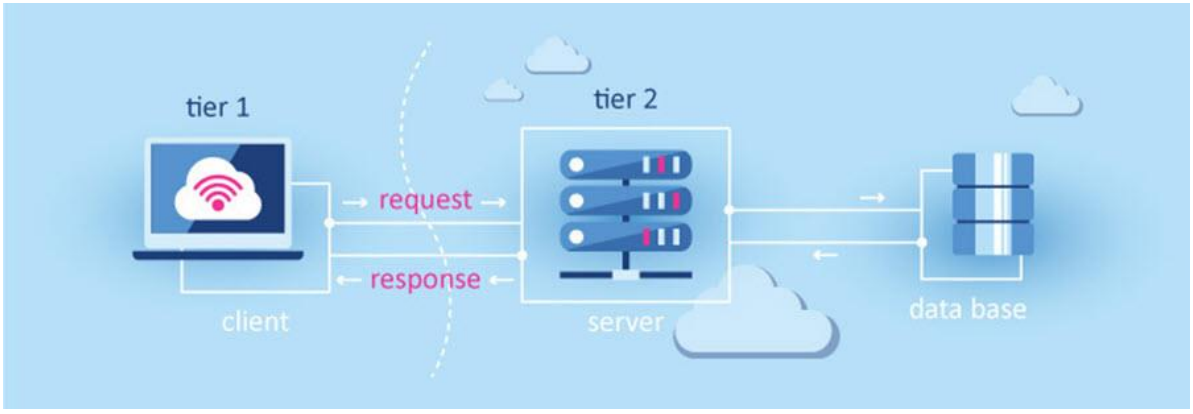

Рисунок 2.1 – Клієнт-серверна архітектура

Користувацький інтерфейс є критично важливим компонентом архітектури системи моніторингу навчання, оскільки він визначає, як користувачі взаємодіють із системою. Основною метою інтерфейсу користувача є забезпечення зручного та інтуїтивно зрозумілого середовища для студентів, викладачів та адміністрації. Це включає в себе не лише естетичний вигляд, а й логічну структуру навігації, що дозволяє користувачам швидко знаходити необхідну інформацію, наприклад, результати тестування, навчальні матеріали або аналітичні звіти. Важливими аспектами є розробка адаптивного дизайну, що забезпечує коректне відображення інтерфейсу на різних пристроях, включаючи мобільні телефони та планшети.

Дизайн користувацького інтерфейсу повинен враховувати специфіку цільової аудиторії. Студенти, викладачі та адміністрація мають різні потреби і очікування від системи. Наприклад, студенти можуть потребувати швидкого доступу до навчальних матеріалів і результатів тестів, тоді як викладачі зосереджуються на моніторингу успішності студентів та підготовці навчальних планів. Тому інтерфейс повинен бути розроблений з урахуванням цих аспектів, включаючи функціональність, яка дозволяє кожній групі користувачів швидко і ефективно виконувати свої завдання.

Важливою частиною інтерфейсу є система візуалізації даних, що дозволяє користувачам отримувати зрозумілу та доступну інформацію про їхній прогрес у навчанні. Це може включати графіки, діаграми та таблиці, які відображають результати тестування, прогрес у виконанні курсу та інші метрики. Візуалізація даних допомагає користувачам швидше усвідомлювати свої сильні та слабкі сторони, а також виявляти тенденції в навчанні. Розробка ефективних механізмів візуалізації є ключовим елементом, який вплине на здатність користувачів приймати обґрунтовані рішення щодо покращення своїх навчальних результатів.

Ще одним важливим аспектом інтерфейсу користувача є інтерактивність. Користувачі повинні мати можливість взаємодіяти з системою, залишаючи коментарі, ставлячи запитання або оцінюючи навчальні матеріали. Це створює середовище для зворотного зв'язку, що дозволяє викладачам та адміністрації

оперативно реагувати на потреби студентів. Інтерактивність також може включати елементи гейміфікації, які підвищують мотивацію студентів до навчання, наприклад, систему нагород або досягнень.

Користувацький інтерфейс повинен бути розроблений з акцентом на простоту використання та доступність. Це включає в себе не тільки естетичний аспект, а й можливість налаштування інтерфейсу відповідно до потреб користувача. Доступність є особливо важливою для забезпечення рівних можливостей для всіх користувачів, включаючи осіб з обмеженими можливостями. Тому важливо дотримуватися принципів універсального дизайну, що дозволить системі бути доступною для якомога більшої кількості людей. У результаті, добре спроектований користувацький інтерфейс стане основою для успішної інформаційної технології моніторингу навчання, що підвищить ефективність навчального процесу.

Система управління базами даних (СУБД, рис. 2.2) є ключовим компонентом архітектури інформаційної технології моніторингу навчання, оскільки вона відповідає за зберігання, обробку та управління даними, пов'язаними з навчальним процесом. Вибір правильного типу бази даних має вирішальне значення для ефективності системи. Для цієї мети доцільно обрати реляційну модель організації баз даних (рис. 2.3), таку як MySQL або PostgreSQL. Реляційні бази даних дозволяють структурувати дані у вигляді таблиць, що забезпечує простоту в обробці запитів та маніпуляцій з даними, а також забезпечує цілісність і зв'язність даних.

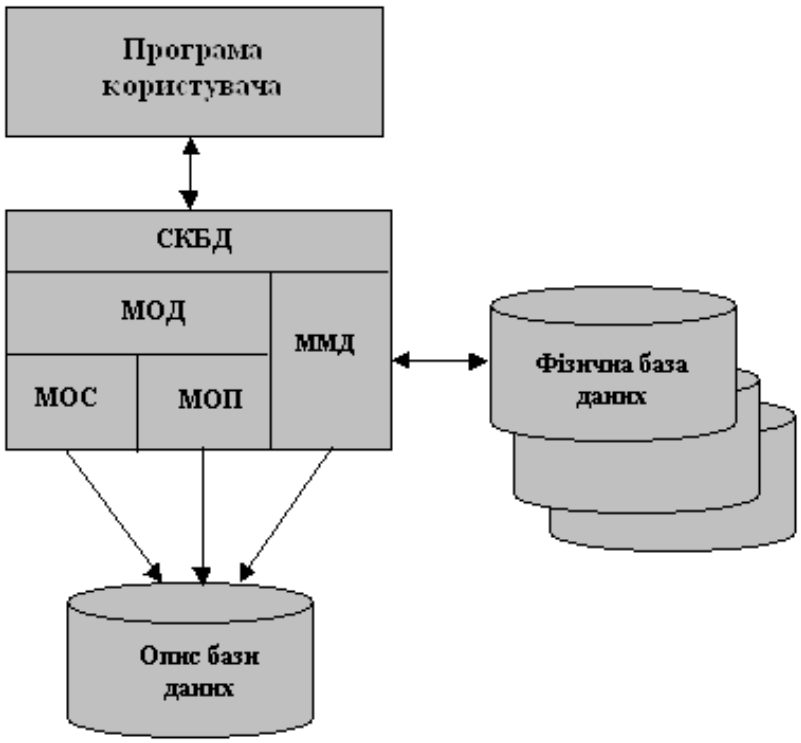

Рисунок 2.2 – Схема функціонування СУБД

Рисунок 2.3 – Приклад організації даних у реляційній БД

У контексті моніторингу навчання реляційна база даних може містити кілька таблиць, які описують різні аспекти навчального процесу. Наприклад, одна таблиця може зберігати дані про студентів, включаючи їх імена, прізвища, ідентифікаційні номери, а інша – результати тестування. Зв'язки між таблицями дозволяють легко отримувати комплексну інформацію, наприклад, визначати, які студенти досягли певних результатів у тестах, а також які курси вони

проходили. Це забезпечує ефективний аналіз і звітність, що є критично важливим для викладачів та адміністрації.

Окрім того, реляційні бази даних забезпечують можливості для впровадження механізмів контролю доступу і безпеки даних. Це важливо для захисту особистої інформації студентів та результатів їх навчання. З системами управління базами даних, такими як MySQL або PostgreSQL, можна реалізувати різні рівні доступу для викладачів, студентів і адміністративного персоналу, що дозволяє запобігти несанкціонованому доступу до чутливої інформації.

Не менш важливою є можливість масштабування реляційної бази даних. З ростом кількості студентів, курсів і тестів система повинна бути здатна обробляти збільшений обсяг даних без значних втрат у швидкості та продуктивності. Реляційні бази даних надають інструменти для оптимізації запитів та налаштування продуктивності, що дозволяє системі адаптуватися до зростаючих потреб навчального закладу.

Вибір реляційної моделі бази даних для системи моніторингу навчання забезпечує структурованість, надійність і масштабованість даних, що є критично важливими для успішного функціонування навчальної системи. Залежно від потреб навчального закладу, система управління базами даних може бути адаптована і налаштована для забезпечення максимальної продуктивності та зручності для користувачів. Таким чином, DBMS стає основою, на якій будується вся архітектура інформаційної технології моніторингу навчання.

Серверна частина системи моніторингу навчання відіграє критично важливу роль у забезпеченні її функціональності, надійності та безпеки. Вона відповідає за обробку запитів від користувачів, виконання бізнес-логіки, взаємодію з базою даних і управління даними, які використовуються в навчальному процесі. Серверна частина зазвичай реалізується на різних мовах програмування, таких як Python, Java, або Node.js, в залежності від вимог проекту та досвіду команди розробників. Правильний вибір технологій для серверної частини є важливим для забезпечення ефективності та масштабованості системи.

Основна функція серверної частини полягає в обробці запитів, які надходять від клієнтських додатків. Коли користувач виконує дії, такі як надсилання тесту, отримання результатів або перегляд звітів, ці запити передаються на сервер, де виконується відповідна логіка. Сервер здійснює валідацію даних, перевіряє права доступу, обробляє бізнес-правила та взаємодіє з базою даних для отримання або зберігання необхідної інформації. Це дозволяє зберегти цілісність даних і забезпечити коректність результатів.

Також важливим аспектом серверної частини є безпека. Вона повинна реалізовувати механізми аутентифікації та авторизації, щоб забезпечити захист чутливих даних, таких як особисті дані студентів і результати їх навчання. Для цього можна використовувати різні протоколи безпеки, такі як OAuth або JWT (JSON Web Tokens), які дозволяють безпечно ідентифікувати користувачів та контролювати доступ до різних частин системи.

Серверна частина також повинна забезпечувати масштабованість системи. Зростання кількості користувачів та обсягу даних може призвести до перевантаження сервера, тому важливо впровадити механізми для горизонтального або вертикального масштабування. Використання контейнеризації з Docker, мікросервісної архітектури або хмарних технологій, таких як AWS або Azure, може значно підвищити гнучкість і можливості системи, дозволяючи швидко адаптуватися до змін у навантаженні.

Серверна частина (рис. 2.4) є основою, на якій будується вся архітектура системи моніторингу навчання. Вона забезпечує ефективну обробку запитів, захист даних та масштабованість системи. Розробка надійної та ефективної серверної частини є важливим етапом у створенні інформаційної технології, яка сприятиме покращенню навчального процесу і досягненню більш високих результатів для студентів і викладачів.

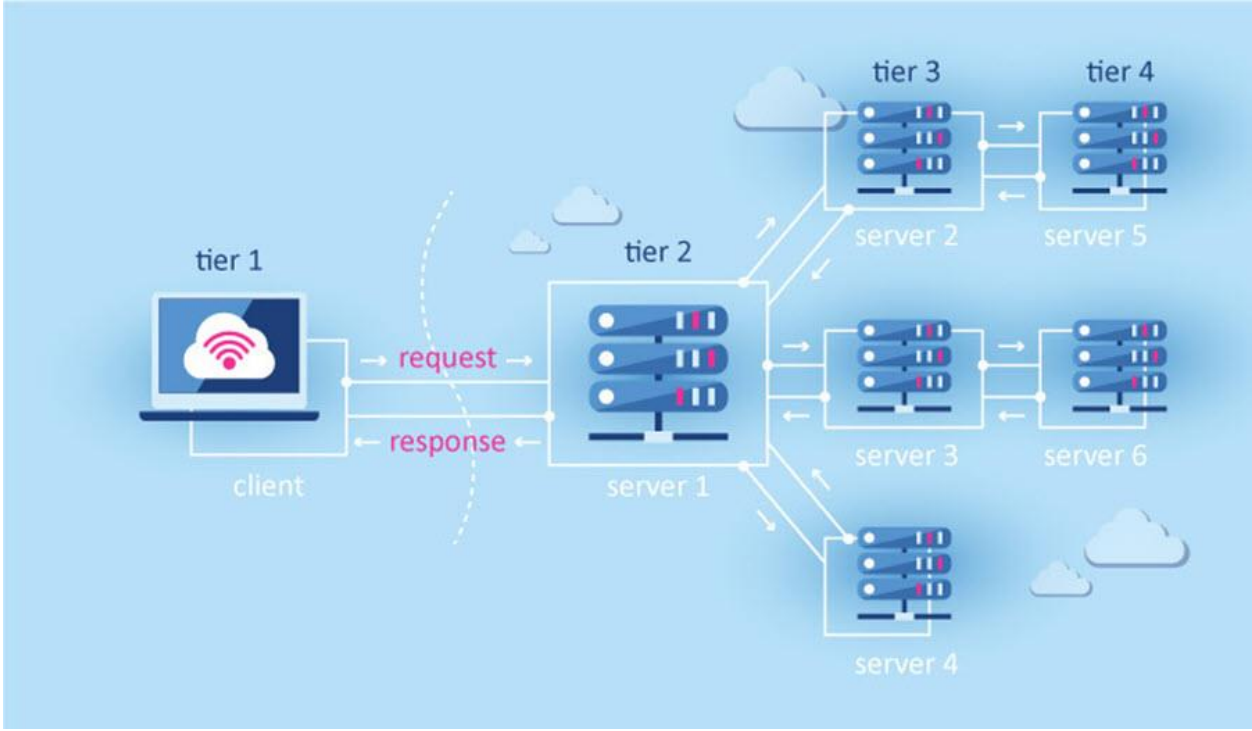

Рисунок 2.4 – Схема організації серверної частини додатку з масштабуванням обчислювальних можливостей

Кінцевим результатом роботи такої інформаційної системи буде певна рекомендація для учня, студента чи викладача. Рекомендаційна система (РС) – це програмне забезпечення або алгоритм, який допомагає користувачам знаходити інформацію або товари на основі їхніх інтересів (рис. 2.5), поведінки та вподобань [16]. Такі системи використовуються в різних сферах, таких як інтернет-магазини, стримінгові сервіси, соціальні мережі, і допомагають користувачам отримувати персоналізовані пропозиції, наприклад, фільми, продукти або музику. Мета рекомендаційних систем – полегшити процес пошуку та підвищити задоволеність користувачів платформою.

Типи РС:

1. колективна фільтрація;
2. контент фільтрація.

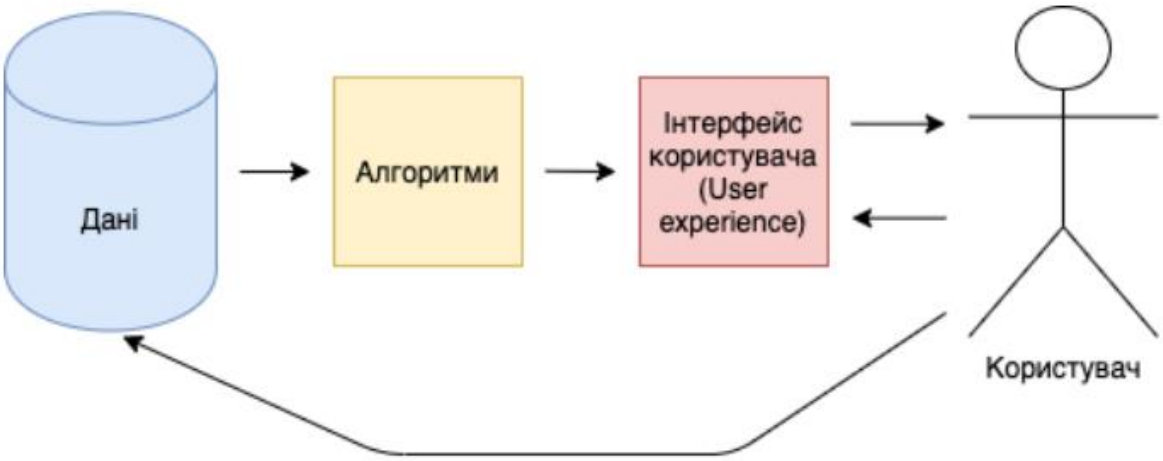

Рисунок 2.5 – Принцип роботи РС

Колективна фільтрація працює на основі аналізу поведінки користувачів. Система не враховує зміст продуктів або інформації, які вона рекомендує, а орієнтується на те, що вибрали інші користувачі зі схожими вподобаннями. Наприклад, якщо двоє людей мають схожі смаки в кіно, колаборативна фільтрація може порекомендувати фільм, який сподобався іншій людині зі схожими інтересами. Вона формує рекомендації на основі збігів між користувачами.

Контент-фільтрація, на відміну від спільної фільтрації, аналізує самі характеристики продуктів або інформації. Система порівнює вміст продукту з тим, що вже подобалося користувачеві, і пропонує схожі варіанти. Наприклад, якщо ви часто читаєте статті про технології, контент-фільтрація запропонує вам більше матеріалів з цієї категорії. Ключовим моментом тут є робота з контентом: ключовими словами, жанрами або іншими характеристиками продукту.

Колаборативна фільтрація хороша тим, що може показувати неочікувані рекомендації, тобто ті, які користувач не знайшов би самостійно, адже система базується на досвіді інших людей. У той же час вона може зіткнутися з проблемами, коли даних про нових користувачів або продукти мало, оскільки не вистачає інформації для порівняння. З іншого боку, контент-фільтрація добре працює, коли є обмежена кількість користувачів, оскільки вона аналізує сам

контент. Однак він може бути обмеженим з точки зору різноманітності рекомендацій, оскільки пропонує лише схожі продукти або контент.

Інформаційна технологія моніторингу навчання є прикладом гібридної рекомендаційної системи (рис. 2.6), яка може поєднувати елементи спільної роботи та контент-фільтрації для надання персоналізованих рекомендацій студентам. Така система може аналізувати як особисту навчальну діяльність студента (контент-фільтрація), так і поведінку інших студентів зі схожими інтересами або успішністю (колаборативна фільтрація). Це дозволяє системі надавати більш точні та релевантні поради щодо навчальних ресурсів або тем, на які варто звернути увагу.

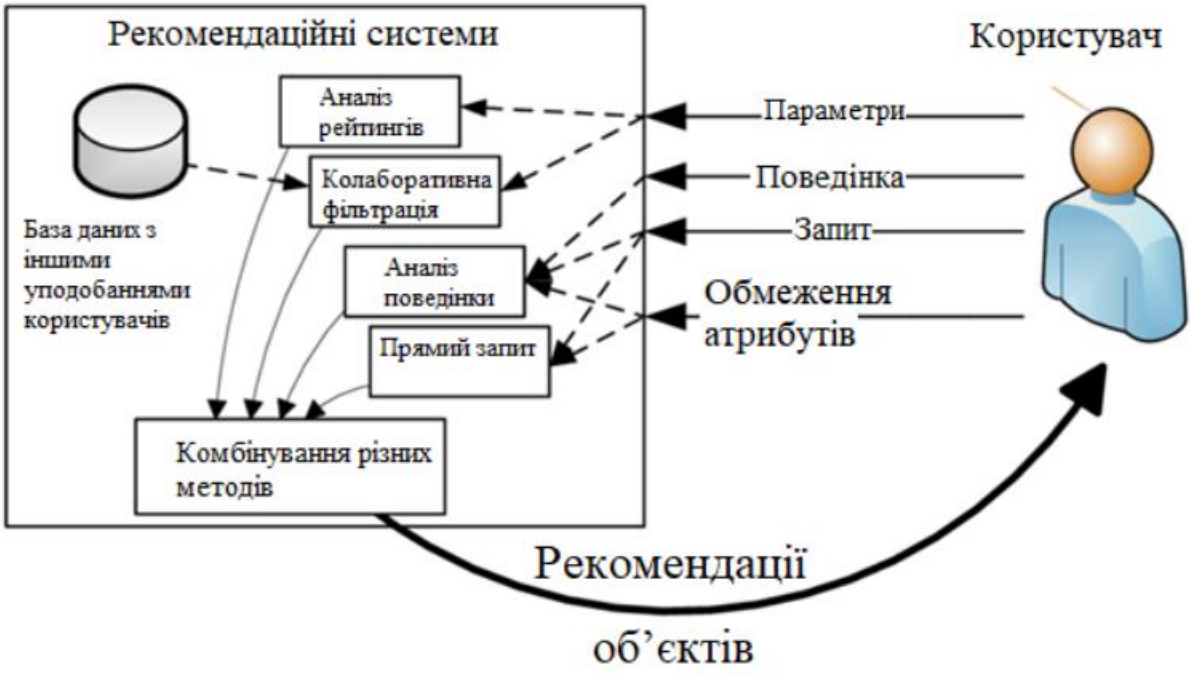

Рисунок 2.6 – Структура гібридної РС

Основна роль цієї системи – відстежувати прогрес і допомагати оптимізувати навчальний процес. Система може відстежувати прогрес студента в режимі реального часу, визначаючи сильні та слабкі сторони в навчанні. На основі цих даних вона може рекомендувати відповідні ресурси, такі як додаткові

лекції, практичні вправи або тести, щоб покращити результати в тих сферах, де студент відчуває труднощі.

В результаті користувач повинен отримати персоналізовані рекомендації, які допоможуть йому покращити свій навчальний процес. Це можуть бути поради щодо того, яким темам приділити більше уваги, які ресурси використовувати для поглиблення знань або які інструменти використовувати для більш ефективного засвоєння матеріалу. Отож схема інформаційної технології моніторингу прогресу навчання зображена на рисунку 2.7.

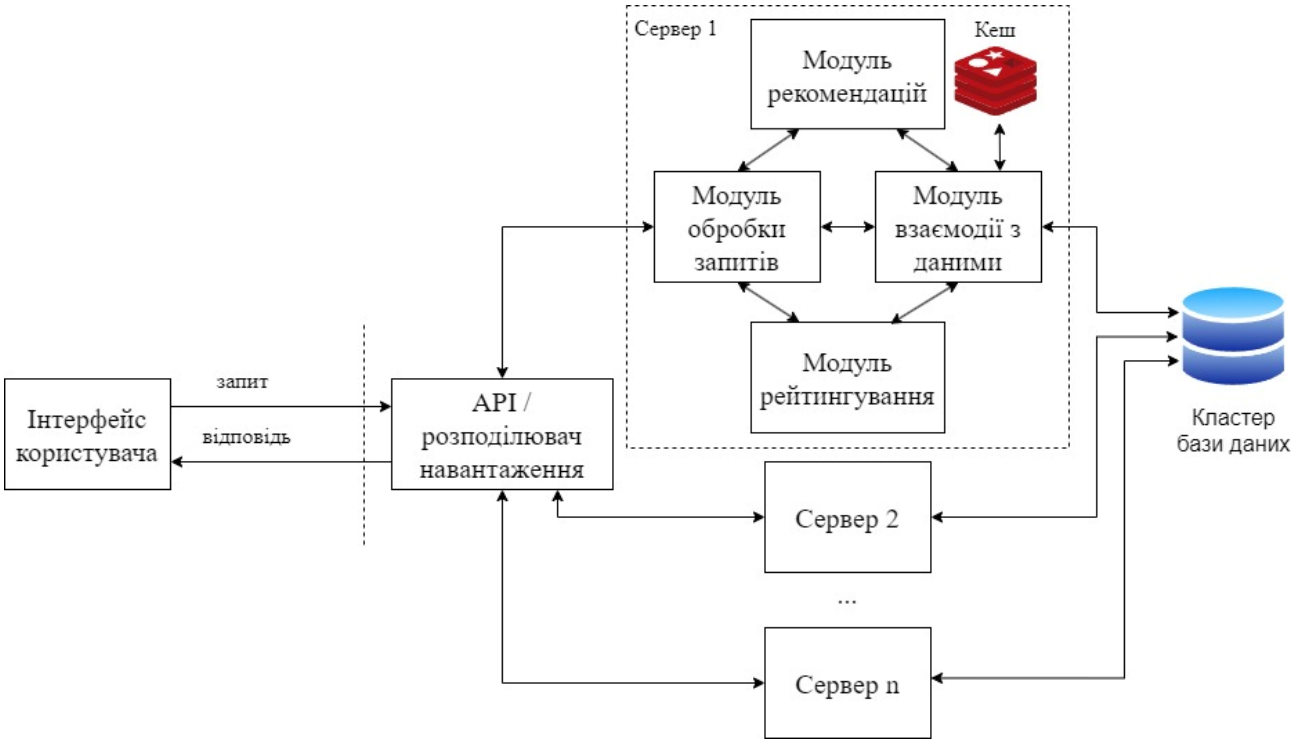

Рисунок 2.7 – Схема функціонування інформаційної системи моніторингу прогресу навчання

Для викладачів система також надає аналітичні інструменти, які допомагають оцінити загальний прогрес групи студентів, виявити найпоширеніші проблеми та відповідно скоригувати методи викладання. Викладачі можуть отримувати зведені дані про прогрес своїх студентів і бачити, де потрібна додаткова підтримка.

2.2 Розробка методу незалежного оцінювання успішності студента на основі тестового рейтингування

Будь яка система рейтингування потребує певного калібрування для первинного налаштування системи. З цією метою багато програм, що використовують рейтингову систему Glicko [17] чи її вдосконалену версію Glicko-2 [18], мають в собі сценарії тестової взаємодії з користувачем для визначення первинного рівня рейтингу. У контексті моніторингу процесу навчання при проходженні тестування це буде доречно зробити у рамках математичної моделі розрахунку коефіцієнту класифікації відповіді при першому проходженні тесту користувачем для визначення рівня вивчення матеріалу.

Для модифікації розрахунку коефіцієнта класифікації відповіді при рейтингуванні тестів [4] додатково було обрано наступні параметри: час надання відповіді – T, рівень впевненості студента у виборі відповіді – C. Час надання відповіді обов'язково повинен враховувати лише конкретний час надання відповіді виключаючи час на прочитання самого питання. Рівень упевненості тестованого визначає сам учень при першому проходженні тесту. Надалі така оцінка не є потрібною, оскільки починається сам процес навчання і засвоєння матеріалу з використанням стандартних методів рейтингування тестів. Внесення таких факторів дозволить здійснювати початкове калібрування системи, більш гнучко відстежувати прогрес студента та враховувати не тільки правильність відповіді, а й інші аспекти процесу тестування, що мають важливе значення для об'єктивного рейтингування.

Коефіцієнт класифікації відповіді наразі відображає значення за якого слід відповідь вважати правильною. Відсоток відповіді на запитання a лежить у діапазоні:

$$a \in [0;\ 100].$$

Визначення мінімуму коефіцієнту класифікації відповіді $k_a - k_{amin}$ відбувається згідно моделі. При чому:

$$k_{amin} \in (\min(a)\,;\max(a)).$$

Рекомендованим значенням для k_{amin} незмінно дорівнює 50. За такого значення усі відповіді, що мають більше ніж 50% правильності, будуть вважатися правильними. При відсотку нижче ніж 50, відповідь буде вважатись неправильною. При 50% відповідь не буде зарахована і рейтинг ніяк не зміниться. Проте ці параметри можна змінити та налаштувати під кожну систему і потреби індивідуально.

Проміжне значення k_a буде обраховуватись зігдно формули:

$$k_a = a - k_{amin}. \tag{2.1}$$

Тепер введемо у вищенаведену математичну модель два нові параметри для отримання модифікованого коефіцієнту класифікації відповіді k' [3].

Час надання відповіді t – час, витрачений студентом на відповідь у секундах. Чим менше часу знадобилося студенту на правильну відповідь, тим вищий показник його впевненості та знань. Звідки очевидно, що час буде мати обернено пропорційну залежність до результату:

$$k' \sim \frac{1}{t}. \tag{2.2}$$

Якщо час більший, це може свідчити про труднощі із завданням. Не менш важливо враховувати лише час на надання відповіді, який виключає час на прочитання питання і варіантів відповіді. Для визначення часу потрібного на прочитання t_r питання і варіанту відповіді використаємо формулу:

$$t_r = \frac{Q}{V} \times 60, \tag{2.3}$$

де Q – кількість слів у тексті, V – швидкість читання (слів на хвилину).

Параметр V буде визначено під час калібрування системи, шляхом фіксування часу на прочитання питання певнох кількості знаків [19], а множник 60 відповідає кількості секунд у хвилині, що приводить результат до секунд.

Відповідно для отримання чистого часу на роздуми використаємо наступну формулу:

$$t = t_{all} - t_r, \tag{2.4}$$

де t_{all} – загальний час проведення користувача над запитанням.

Якщо ж $t < 0$ це сигналізує, про те що користувач найімовірніше надав відповідь навмання. В такому випадку коефіцієнт класифікації відповіді приймаємо таким що дорівнює нулю. Дана відповідь не буде враховуватись при розрахунку рейтингу оскільки не несе жодного навчального сенсу.

Рівень впевненості C – показник впевненості студента у відповіді. Нехай множина рівня впевненості C для мінімального c_{min} та максимального c_{max} визначатиметься як:

$$C \in [c_{min}, c_{max}],$$

де $c_{min} < c_{max}$ і $c_{min} \geq 0$.

Важливими умовами є те, що час та рівень впевненості повинні бути більшими за нуль.

Отже, з урахуванням усіх вищеописаних особливостей нова формула розрахунку коефіцієнту класифікації відповіді буде обчислюватись наступним чином:

$$k' = k_a \times (\frac{1}{1+t}) \times \frac{C}{c_{max}}. \quad (2.5)$$

Множник $\frac{1}{1+t}$ враховує час надання відповіді. Чим більше часу витрачає студент, тим меншим буде значення цього множника (враховуючи, що час t більше нуля). Це дозволяє знижувати коефіцієнт для студентів, які потребували більше часу, тим самим враховуючи швидкість реакції як показник знань.

Множник $\frac{C}{\max(C)}$ враховує рівень впевненості студента, масштабуючи його значення від 0 до 1. Якщо студент дуже впевнений у відповіді, множник наближається до 1, а якщо невпевнений – до 0. Це допомагає коригувати оцінку залежно від суб'єктивного рівня впевненості та здійснювати початкове калібрування системи. Надалі після здійснення налаштування системи, даний параметр повинен ігноруватись системою.

Оскільки обрахування зміни рейтингу залежить безпосередньо від значення k_a, бо воно напряму вказує наскільки точно була надана відповідь. Маємо удосконалену математичну модель обрахування зміни рейтингу [5], яка включає новий комплексний коефіцієнт класифікації відповіді:

$$\begin{aligned}
\Delta r &= \frac{v \times k\prime \times rd}{2 \times (r+1)}, \text{при } k' < 0 \text{ і } \Delta r_0 > 0, \\
\Delta r &= \Delta r_0 + \frac{v \times k\prime \times rd}{r+1}, \text{при } k' < 0 \text{ і } \Delta r_0 \leq 0, \\
\Delta r &= \frac{v \times k\prime \times rd}{2 \times (r_{max} - r + 1)}, \text{при } k' \geq 0 \text{ і } \Delta r_0 < 0, \\
\Delta r &= \Delta r_0 + \frac{v \times k\prime \times rd}{r_{max} - r + 1}, \text{при } k' \geq 0 \text{ і } \Delta r_0 \geq 0,
\end{aligned} \quad (2.6)$$

де Δr – значення зміни рейтингу, Δr_0 – попереднє значення зміни рейтингу, v – значення волатильності, r – поточний рейтинг, rd – поточне відхилення рейтингу, r_{max} – максимально допустиме значення рейтингу.

2.3 Розробка алгоритму функціонування інформаційної технології моніторингу процесу навчання

Алгоритм роботи такої системи базується на взаємодії декількох модулів, кожен з яких виконує свою унікальну функцію:

1. аналітичний модуль;
2. модуль управління контентом;
3. модуль підтримки користувачів;
4. модуль інтеграції.

Кожен з цих модулів має власну структуру та завдання, що забезпечує ефективну та злагоджену роботу системи в цілому.

Аналітичний модуль є серцем системи моніторингу навчання, оскільки відповідає за збір, обробку та аналіз даних про успішність студентів. До нього постійно надходить інформація про результати тестування, активність у навчальному процесі, час, витрачений на виконання завдань, та інші важливі показники. На основі цих даних аналітичний модуль формує звіти для викладачів та студентів, дозволяючи їм бачити загальний прогрес та проблемні місця у навчанні.

Основний алгоритм роботи аналітичного модуля полягає в автоматичному аналізі великих обсягів даних та формуванні індивідуальних рекомендацій для кожного студента на основі рейтингування тестів (рис. 2.8). Система використовує методи машинного навчання для виявлення закономірностей у навчанні, допомагаючи викладачам адаптувати свої підходи до потреб студентів. Крім того, аналітичний модуль забезпечує прозорість оцінювання, що підвищує об'єктивність і точність оцінювання результатів навчання.

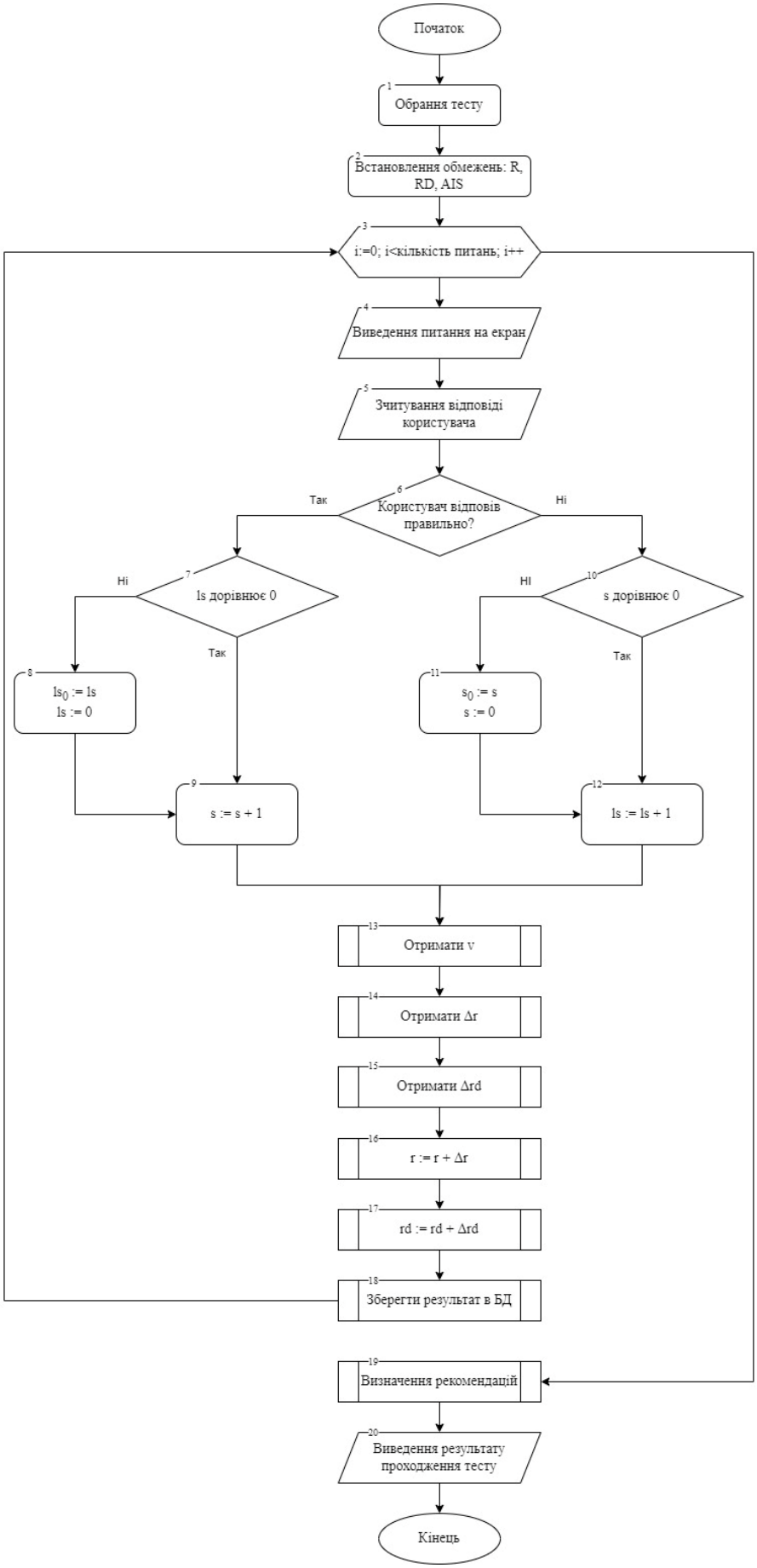

Рисунок 2.8 – Схема алгоритму рейтингування при проходженні тесту

Модуль управління контентом надає можливість працювати з навчальними матеріалами, що використовуються в навчальному процесі. Його основна функція – централізоване зберігання та організація доступу до навчальних ресурсів, таких як лекції, відео, презентації та тести. Викладачі можуть створювати та редагувати контент, а також керувати доступом до нього, дозволяючи студентам працювати з матеріалами як під час занять, так і індивідуально.

Алгоритм роботи модуля управління контентом включає підтримку різних форматів файлів, інтеграцію із зовнішніми платформами для обміну матеріалами та можливість відстежувати використання ресурсів студентами, а також підбиранням тестових запитань (рис. 2.9). Це дозволяє визначити, які матеріали є найбільш ефективними для вивчення певних тем, і відповідно адаптувати навчальний процес. Модуль також інтегрований з модулем аналітики для надання зворотного зв'язку щодо ефективності використання різних ресурсів.

Модуль підтримки користувачів забезпечує постійну взаємодію між студентами, викладачами та технічною підтримкою. Основною функцією цього модуля є забезпечення швидкого реагування на питання, проблеми та технічні труднощі, з якими користувачі можуть зіткнутися під час роботи з системою. Цей модуль також підтримує можливість консультацій між студентами та викладачами, надаючи інструменти для обговорення проблемних питань у навчанні.

Алгоритм роботи модуля підтримки користувачів передбачає автоматизацію обробки запитів через використання чат-ботів, систем відстеження запитів та інтеграцію з іншими частинами системи для ефективного вирішення проблем. Крім того, модуль має функцію сповіщень про важливі події, такі як дедлайни, що допомагає студентам бути в курсі свого навчального прогресу. Викладачі також можуть використовувати модуль для надсилання індивідуальних повідомлень студентам.

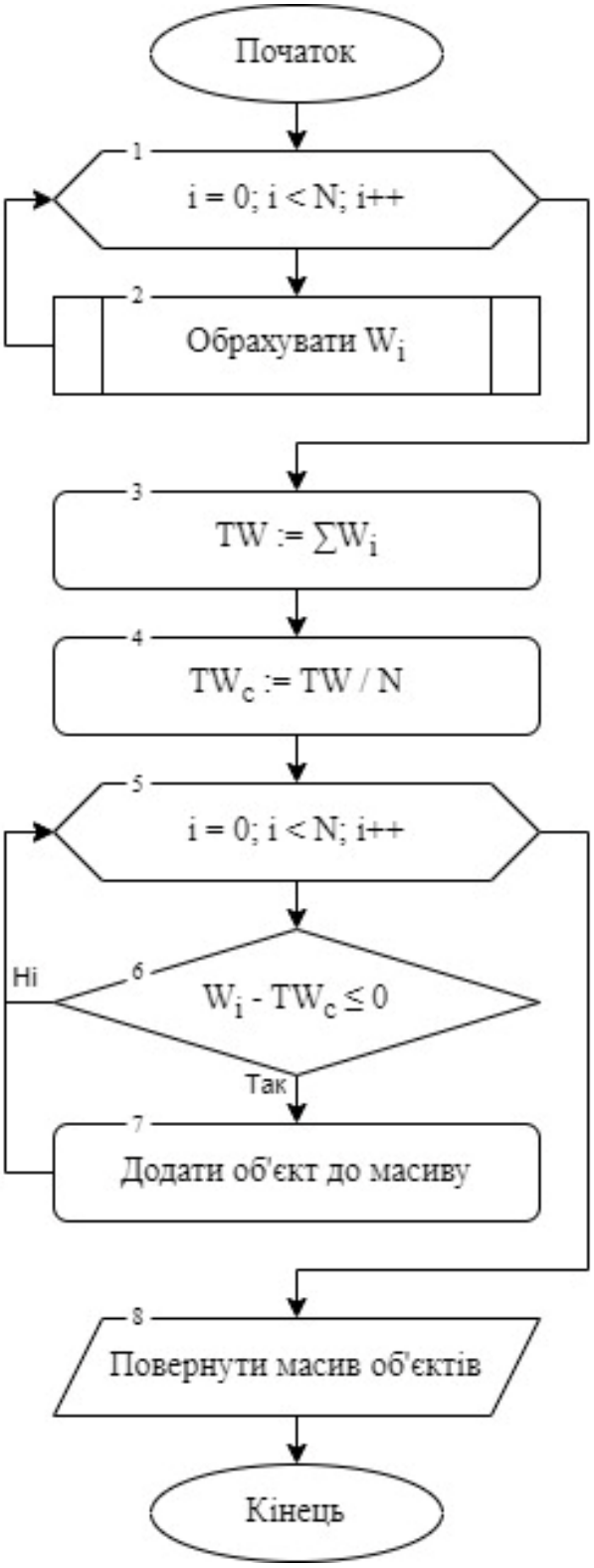

Рисунок 2.9 – Схема алгоритму формування рекомендацій

Модуль інтеграції надає можливість об'єднати інформаційну технологію моніторингу навчання з іншими зовнішніми системами та сервісами. Це дозволяє використовувати різноманітні освітні платформи, календарі, СУН та інші

інструменти, які можуть покращити якість навчання. Завдяки цьому модулю система стає більш універсальною та адаптивною до вимог конкретного навчального закладу чи викладача.

Алгоритм роботи модуля інтеграції базується на підтримці різних протоколів обміну даними, що дозволяє легко підключати зовнішні сервіси. Модуль інтеграції також забезпечує обмін даними між системами, що дозволяє використовувати інформацію з інших платформ для більш точного аналізу та кращої адаптації системи до потреб користувачів.

На рисунку 2.10 зображено діаграму послідовностей алгоритму рейтингування для тестів, який буде використовувати аналітичний модуль для надання навчальних рекомендацій. Дана діаграма відображає взаємодію між кількома модулями системи під час процесу перевірки відповіді користувача в навчальній системі з рейтингуванням.

Основні кроки системи:

1. Користувач активує систему. Користувач активує взаємодію із системою через модуль ініціалізації. Модуль ініціалізації викликає метод «Activate()», щоб налаштувати систему і встановити обмеження (R, RD, AIS).

2. Ініціалізація системи. Модуль ініціалізації викликає «Initialize()» для запуску модуля введення/виведення та модулю даних. Модуль даних ініціалізується і готовий надати питання.

3. Запитання та відповідь. Після ініціалізації користувач запитує питання через модуль введення/виведення («Get Question()»), і модуль даних надає наступне запитання. Після того, як користувач дає відповідь («Check Answer(Answer)»), ця відповідь перевіряється через модуль введення/виведення і передається модулю даних.

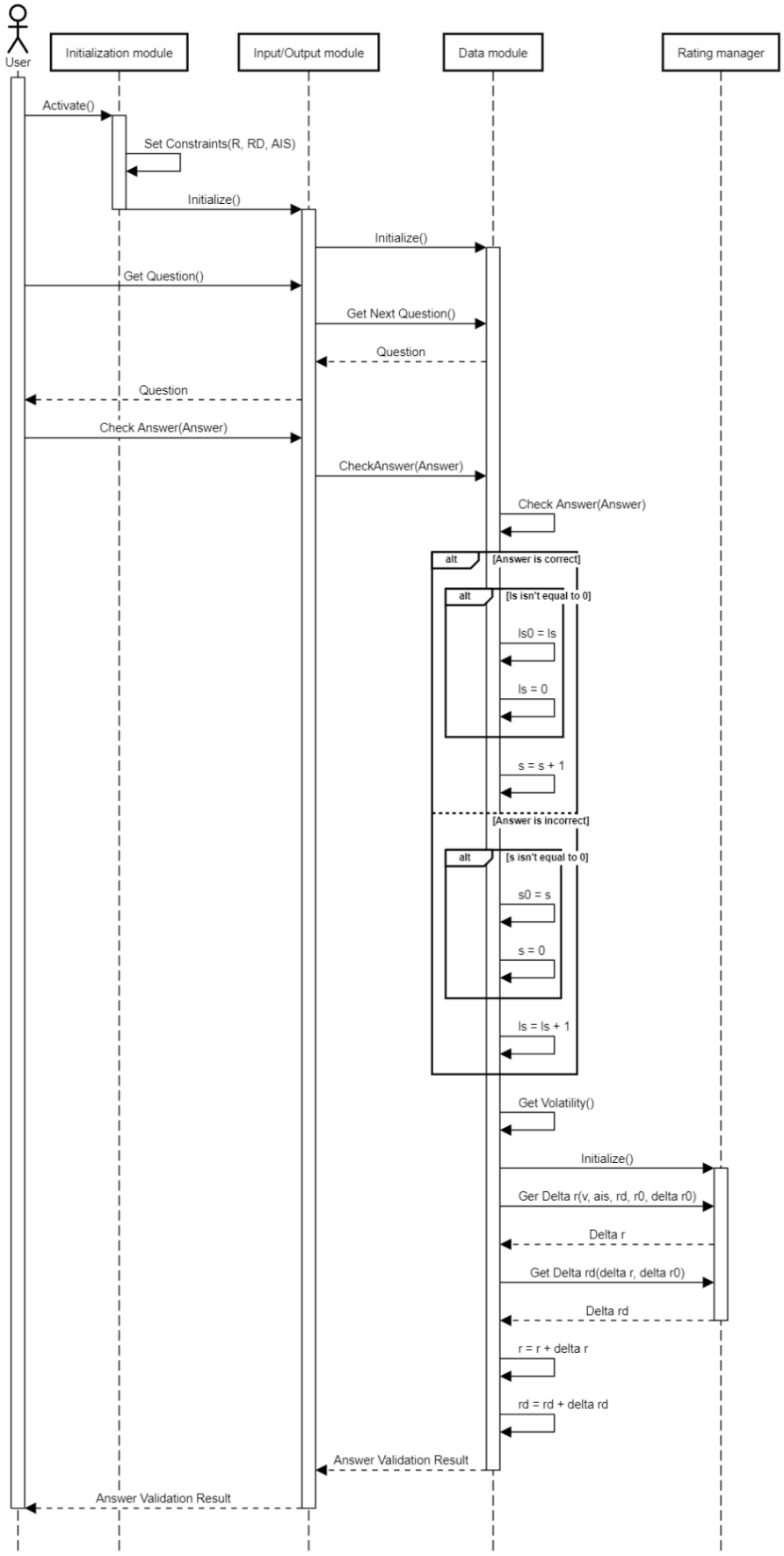

Рисунок 2.10 – UML-діаграма послідовностей алгоритму рейтингування для тестів

4. Перевірка відповіді. Модуль даних отримує відповідь і запускає процес перевірки через метод «CheckAnswer(Answer)». У залежності від правильності відповіді відбувається обробка: якщо відповідь правильна, змінюються відповідні змінні (наприклад, збільшення кількості правильних відповідей $s = s + 1$). Якщо відповідь неправильна, змінюється кількість помилок ($ls = ls + 1$).

5. Оновлення рейтингу. Незалежно від результату, система оцінює волатильність відповіді (перевірка стабільності рішень користувача) через метод «Get Volatility()». На основі цих даних розраховується зміна рейтингу користувача («GetDeltaR()» та «GetDeltaRD()»), після чого оновлюється поточний рейтинг ($r = r + \Delta r$) та інші показники.

6. Повернення результату. Після завершення перевірки системою обчислюється остаточний результат валідації відповіді, який повертається користувачу.

Загальний процес: користувач ініціює взаємодію з системою, отримує питання і надає відповідь; система перевіряє відповідь, аналізує результати, оновлює рейтинг та відправляє результат користувачу.

Ця діаграма ілюструє комплексний процес перевірки та оцінювання відповідей користувача із врахуванням рейтингів і волатильності для об'єктивного та справедливого аналізу результатів навчання.

2.4 Висновок до розділу 2

У процесі розробки інформаційної технології моніторингу навчання було здійснено декілька важливих етапів, які охоплюють як побудову моделі системи, так і розробку алгоритмів для незалежного оцінювання успішності студентів.

Була запропонована модель інформаційної технології моніторингу, яка враховує основні етапи освітнього процесу, включаючи збір, аналіз та візуалізацію навчальних даних. Особливу увагу приділено інтеграції системи з існуючими навчальними платформами, забезпеченню доступу до даних у реальному часі та впровадженню інструментів персоналізованих рекомендацій.

Розроблена модель дозволяє забезпечити гнучке й ефективне керування навчальним процесом, своєчасно виявляти проблемні зони у знаннях студентів та надавати їм необхідну підтримку.

Був розроблений метод незалежного оцінювання успішності студентів на основі рейтингування результатів тестування, що використовує у собі модифікований коефіцієнт класифікації відповіді. Даний метод включає у себе нові параметри, такі як чистий час читання запитання та впевненість. Дані параметри зменшать шанс врахування надання випадкової відповіді студентом та нададуть можливість первинного калібрування системи.

Алгоритм функціонування інформаційної системи використовує технології аналізу даних та враховує не тільки абсолютні результати тестування, а й динаміку навчальних досягнень студентів. Це дозволяє надати об'єктивну та точну оцінку успішності кожного студента, незалежно від зовнішніх факторів. Такий підхід сприяє формуванню індивідуальних траєкторій навчання та мотивує студентів до постійного вдосконалення своїх знань.

Отже, розробка інформаційної технології моніторингу прогресу навчання включає комплексну інтеграцію різноманітних модулів та алгоритмів, які спрямовані на забезпечення прозорості, об'єктивності та індивідуального підходу до навчального процесу. Запропонована технологія є важливим інструментом для підвищення ефективності освітнього процесу та покращення якості знань студентів.

3 ПРОГРАМНА РОЗРОБКА ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ МОНІТОРИНГУ ПРОЦЕСУ НАВЧАННЯ

3.1 Обґрунтування вибору середовища розробки та мови програмування

Вибір середовища розробки є ключовим фактором для програмістів, оскільки від нього залежить продуктивність, зручність роботи та якість створюваних програмних продуктів. Порівняємо три популярних середовища розробки: Visual Studio 2022 [20], Visual Studio Code [21] та IntelliJ IDEA [22], за такими параметрами: набір інструментів розробки, вимогливість ресурсів комп’ютера, можливість налагодження програмних продуктів, зручність користування та підтримка різних мов, фреймворків чи технологій.

Для вибору середовища розробки між Visual Studio 2022, Visual Studio Code та IntelliJ IDEA доцільно використовувати такі ключові критерії: підтримка мов програмування та інтегрованих інструментів, простота використання та інтеграція з платформами розробки, наявність функцій налагодження, можливості розширення та масштабування. Проаналізувавши ці критерії, вибір припав на Visual Studio 2022 як найбільш підходящий інструмент для розробки інформаційної технології моніторингу навчання.

Підтримка мов програмування та інтегрованих інструментів. Visual Studio 2022 має вбудовану підтримку великої кількості мов, серед яких C#, .NET, Python, JavaScript та багато інших. Це важливо для розробки інформаційних систем, де різні мови можуть використовуватися для реалізації інтерфейсної та бекенд-частин системи. Visual Studio Code також підтримує багато мов, але його основний функціонал більше орієнтований на просте редагування тексту та коду, ніж на повноцінне середовище розробки. IntelliJ IDEA, з іншого боку, орієнтована на мови JVM (Java Virtual Machine), такі як Java, Kotlin та інші, і хоча вона підтримує додаткові мови, їй не вистачає деяких інтегрованих функцій для .NET.

Простота використання та інтеграція з платформами. Visual Studio 2022 має зручний, добре структурований інтерфейс, який легко адаптується для великих проектів. Інструмент також добре інтегрується з іншими продуктами Microsoft, такими як Azure DevOps, що є перевагою для розробки в корпоративних середовищах або проектів з високими вимогами до масштабованості. Visual Studio Code, хоча і є більш легким інструментом, не має такої ж глибини інтеграції з платформами Microsoft. IntelliJ IDEA також пропонує інтуїтивно зрозумілий інтерфейс та інтеграцію з різними платформами, але найкраще підходить для розробки проектів на основі JVM.

Інструменти налагодження і тестування. Visual Studio 2022 надає потужні можливості налагодження, включаючи підтримку багатопотокових і паралельних процесів, що важливо для розробки складних інформаційних систем. Інтегровані засоби тестування дозволяють зручно створювати і запускати тестові сценарії безпосередньо в середовищі розробки. Visual Studio Code, хоч і підтримує деякі інструменти налагодження через розширення, не має такої глибини можливостей, що робить його менш зручним для великих проектів. IntelliJ IDEA також має розвинену систему налагодження, але вона більше орієнтована на мови JVM, і її можливості обмежені для .NET проектів.

Можливості розширення і масштабування. Visual Studio 2022 має розширену підтримку різних плагінів і розширень, що дозволяє легко адаптувати середовище під конкретні потреби проекту. Це робить його ідеальним вибором для великих команд або складних проектів, де потрібно інтегрувати спеціалізовані інструменти або додаткову функціональність. Visual Studio Code також має велику кількість розширень, але як більш легке середовище, воно обмежене в можливостях, необхідних для масштабування великих проектів. IntelliJ IDEA також має хорошу підтримку розширень, але знову ж таки, вона найбільш адаптована для мов JVM, а її налаштування під .NET або C# може вимагати додаткових інструментів і не завжди є зручним.

Враховуючи всі критерії, Visual Studio 2022 є найбільш підходящим середовищем розробки для створення інформаційної технології моніторингу

навчання. Воно забезпечує максимальну підтримку мов та інтегрованих інструментів, має потужні можливості налагодження та тестування, а також можливість масштабування та розширення проекту. Порівняльна характеристика середовищ розробки подана в таблиці 3.1.

Таблиця 3.1 – Порівняльна характеристика середовищ розробки за п’ятибальною шкалою

Критерій	Visual Studio 2022	Visual Studio Code	IntelliJ IDEA
Підтримка мов програмування та інструментів	5	4	3
Зручність використання та інтеграція	5	3	4
Налагодження та тестування	5	3	4
Розширення та масштабування	5	4	4

Вибір мови програмування також є важливим етапом в розробці програмного забезпечення. Для порівняння мов програмування C#, C++, Python та Java розглянемо такі параметри, як інтеграція з платформами та технологіями, продуктивність, зручність розробки та налагодження, підтримка бібліотек та інструментів.

Для вибору мови програмування для розробки інформаційної технології моніторингу навчального процесу розглядалися C#, C++, Python та Java. Вибір здійснювався за такими критеріями: інтеграція з необхідними платформами та технологіями, продуктивність, простота розробки, підтримка необхідних

бібліотек та інструментів для створення користувацького інтерфейсу та аналітичних модулів. На основі порівняння цих критеріїв найкращим вибором для розробки системи є мова програмування C#.

Інтеграція з платформами і технологіями. C# є базовою мовою для платформи .NET, що забезпечує легку інтеграцію з іншими інструментами Microsoft, такими як Azure, SQL Server та Visual Studio. Це значна перевага при розробці інформаційної системи моніторингу, яка, ймовірно, використовуватиме базу даних та інші хмарні сервіси. Python також пропонує широку інтеграцію, особливо з аналітичними платформами, але має обмеження при роботі з корпоративними платформами Microsoft. C++ менш пристосована для таких інтеграцій, а Java більше орієнтована на інтеграцію з платформою JVM і менш оптимальна для використання з середовищем .NET.

Продуктивність. C# забезпечує високу продуктивність і є достатньо швидкою для більшості завдань, пов’язаних з моніторингом навчального процесу, включаючи обробку даних і виконання аналітичних запитів. Хоча C++ є найпродуктивнішою з розглянутих мов, вона вимагає більше зусиль при розробці та налагодженні, особливо при створенні користувацьких інтерфейсів та роботі з великими базами даних. Python, з іншого боку, дещо поступається в продуктивності, особливо при роботі з великими обсягами даних. Java забезпечує середній рівень продуктивності, але її екосистема також не настільки оптимізована для роботи з технологіями Microsoft.

Простота розробки та налагодження. C# відома своїм простим та інтуїтивно зрозумілим синтаксисом, що дозволяє розробникам швидко створювати, налагоджувати та тестувати програми. Visual Studio, основне середовище розробки для C#, також додає зручності завдяки потужним інструментам налагодження. Python, завдяки своїй простоті, також легкий у вивченні та розробці, але менш зручний для створення продуктивних корпоративних додатків. C++ має складніший синтаксис і вимагає більше зусиль для налагодження. Java має зручний синтаксис, але не забезпечує такої ж легкості розробки для середовища .NET, як C#.

Підтримка необхідних бібліотек та інструментів. C# має велику кількість бібліотек для роботи з базами даних, аналітичних інструментів та засобів інтерфейсу користувача. Це дозволяє легко реалізувати функції моніторингу та аналізу навчального процесу. Python також має великий вибір бібліотек, особливо для обробки даних, але її екосистема менш інтегрована з корпоративними середовищами. C++ потребує більше часу на розробку, оскільки основні бібліотеки менш зручні для користувача, а Java також має значні можливості, але не забезпечує легкого доступу до інструментів розробки Microsoft.

Виходячи з проведеного аналізу, C# є найкращим вибором для розробки інформаційної технології моніторингу навчального процесу, забезпечуючи оптимальний баланс між продуктивністю, простотою розробки та інтеграцією з платформою .NET і технологіями Microsoft. Порівняльна характеристика мов програмування подана в таблиці 3.2.

Таблиця 3.2 – Порівняльна характеристика мов програмування за п’ятибальною шкалою

Критерій	C#	C++	Python	Java
Інтеграція з платформами та технологіями	5	3	4	4
Продуктивність	4	5	3	4
Зручність розробки та налагодження	5	3	4	4
Підтримка бібліотек та інструментів	5	3	4	4

3.2 Програмна реалізація складових інформаційної технології моніторингу процесу навчання

Модуль рейтингування імплементує в собі метод незалежного оцінювання студента, що включає у себе обчислення модифікрваного коефіцієнту класифікації відповіді та рейтингу для кожної наданої відповіді.

Згідно принципів SOLID та ООП даний блок буде представляти 2 класи.

Перший клас Consts буде зберігати у собі усі обмеження та базові незмінні величини для обрахування рейтингу, які будуть використовуватись в подальшому при обчисленнях зміни рейтингу.

Нижче представлено статичний клас Consts:

```
public static class Consts
{
    const double Rmin = 0;
    const double Rmax = 50;
    const double RDmin = Rmax;
    const double RDmax = 100;
    const double Amin = 0;
    const double Amax = 100;
    const double Kamin = 50;
    const double Cmin = 1;
    const double Cmax = 5;
    public static void DoCorrections(ref double r, ref double rd)
    {
        if (r < Rmin) { r = Rmin; }
        if (r > Rmax) { r = Rmax; }
        if (rd < RDmin) { rd = RDmin; }
        if (rd > RDmax) { rd = RDmax; }
    }
}.
```

Метод DoCorrections здійснює корекцію значень, якщо ті виходять за допустимий діапазон.

Інший клас RatingManager містить у собі усі необхідні методи для обчислення параметрів математичної моделі незалежного оцінювання студента:

```
public class RatingModel
{
    public double GetK(double a)
    {
        return a - Kamin;
    }
    public double GetKmodified(double k, double t, double c)
    {
        return k * (1 / (1 + t)) * (c / Cmax);
    }
    public bool IsAnswered(double a)
    {
        return a > Kamin;
    }
    public double GetV(bool isAnswered, double s, double s0, double ls, double ls0)
    {
        if (isAnswered)
        {
            if (s == s0 && s0 == 0) { return 1; }
            if (s - ls0 == 0) { return 0; }
            return Math.Abs(1 / (s - ls0));
        }
        if (ls == ls0 && ls0 == 0) { return 1; }
        if (ls - s0 == 0) { return 0; }
        return Math.Abs(1 / (ls - s0));
```

```
        }
        public double GetDeltaR(double v, double k, double r, double rd, double
deltar0)
        {
            if (k < 0 && deltar0 > 0)
            {
                return (v * k * rd) / (2 * (r + 1));
            }
            else if (k < 0 && deltar0 <= 0)
            {
                return deltar0 + ((v * k * rd) / (r + 1));
            }
            else if (k >= 0 && deltar0 < 0)
            {
                return (v * k * rd) / (2 * (Rmax - r + 1));
            }
                else if (k >= 0 && deltar0 >= 0)
            {
                return deltar0 + ((v * k * rd) / (Rmax - r + 1));
            }
            throw new NotSupportedException("Not acceptable parameters for
rating");
        }
        public double GetDeltaRD(double deltar, double deltar0)
        {
            if (deltar == 0 || deltar0 == 0)
            {
                return (deltar + deltar0) / 2;
            }
            else if (IsSameSign(deltar, deltar0))
```

```
            {
                return deltar + deltar0;
            }
            else if (!IsSameSign(deltar, deltar0))
            {
                return -1 * (Math.Abs(deltar) + Math.Abs(deltar0));
            }
            throw new NotSupportedException("Not acceptable parameters for rating
devation");
        }
        public bool IsSameSign(double x, double y)
        {
            if (x == 0 || y == 0)
            {
                throw new ArgumentException("Value cannot be zero during
comparation");
            }
            return (x > 0) == (y > 0);
        }
    }.
```

Даний клас містить в собі методи для обчислення зміни рейтингу, його відхилення і усіх необхідних для цього параметрів: коефіцієнт класифікації відповіді, модифікований коефіцієнт класифікації відповіді, волатильність. Також присутній метод для перевірки чи відповідь була надано правильно, який у результаті дає одне з двох значень «true» або «false» для додаткових перевірок, які здійснює та аналізує система.

Модуль рекомендацій імплементує в собі алгоритм формування рекомендацій. Даний модуль представлений класом RecomendationManager, методи якого виконують обчислення ваги окремого об’єкту, обчислення загальної ваги списку об’єктів, визначення середнього значення, підбір

наступного питання для користувача з тесту, формують список рекомендованих тем що потребують подальшого вивчення.

Метод GetWeight з класу RecomendationManager призначений для обчислення ваги окремого об’єкту:

```
public double GetWeight(RatingParams ratingParams)
{
   return ratingParams.RD / (ratingParams.R * ratingParams.V * ratingParams.N);
}.
```

Метод GetWeight з класу RecomendationManager призначений для обчислення ваги списку об’єктів:

```
public double GetWeight(List<RatingParams> ratingParams)
{
   var result = 0;
   foreach(var item in ratingParams)
   {
      result += item.RD / (item.R * item.V * item.N);
   }
   return result / ratingParams.Count;
}.
```

Імплементація алгоритму формування рекомендацій здійснена у методі GetRecommendation класу RecomendationManager:

```
public List<ITopic> GetRecommendations(List<ITopic> topics)
{
   List<ITopic> recommendations = new List<ITopic>();
   List<double> weights = new List<double>();
   double totalWeight = 0;
   double avgTotalWeight = 0;
   foreach (ITopic topic in topics)
   {
```

```
            var w = GetWeight(new RatingParams() { R = topic.R, RD = topic.RD, V = 1, N = topic.N });
            weights.Add(w);
            totalWeight += w;
        }
        avgTotalWeight = totalWeight / weights.Count;
        for (int i = 0; i < weights.Count; i++)
        {
            if (weights[i] – avgTotalWeight <= 0)
            {
                recommendations.Add(topics[i]);
            }
        }
        return recommendations;
    }.
```

Волатильність передаємо у метод GetWeight завжди зі значенням 1, оскільки при формуванні рекомендацій відхилення не грає ролі. Волатильність має значення при підборі наступного питання з тесту для користувача.

Відбір запитання здійснюється аналогічно підбору тематики. Цей процес включає в себе обчислення ваги кожного запитання, обчислення їх загальної ваги і визначення випадково числа в межах від 1 до загальної ваги. Після чого здійснюється відбір наступного запитання, яке отримає користувач для проходження.

Програмна реалізація підбору запитань здійснена в методі GetNext<TType> класу RecomendationManager:

```
public TType GetNext<TType> (List< TType > items)
{
    List<double> weights = new List<double>();
    double totalWeight = 0;
    foreach (TType item in items)
```

```
        {
            var w = GetWeight(new RatingParams() { R = item.R, RD = item.RD, V =
item.Volatility, 1);
            weights.Add(w);
            totalWeight += w;
        }
        Random rnd = new Random();
        while (true)
        {
            double sigma = rnd.Next(1, (int)totalWeight);
            for (int i = 0; i < weights.Count; i++)
            {
                if (weights[i] – sigma <= 0)
                {
                    return item[i];
                }
            }
        }
    }.
```

При визначенні ваги питання останнім параметром в метод GetWeight передається 1, оскільки у питання відсутній будь який список всередині, на відміну від тематики, де міститься список питань.

Для зберігання та ефективного отримання даних необхідно мати базу даних та кеш. База даних зберігає дані на довготривалій основі. Кеш призначений для тимчасовго зберігання частовикористовуваних даних з метою швидкого їх отримання для оптимізації та збільшення швидкодії програми.

Згідно принципів SOLID та ООП даний блок буде представляти 2 класи. Перший InMemoryCache що буде зберігати дані в кеші, надавати частозапитувані дані та здійснювати їх очистку після певного зазначеного терміну:

```
public class InMemoryCache
```

```
{
    public event Action<object, object, EvictionReason> EvictionCallback;
    public EntrySource GetOrAdd<TItem>(string key, Func<TItem> getItemDelegate, out TItem data, CacheItemPriority priority = CacheItemPriority.Normal)
    {
        _lock.EnterUpgradeableReadLock();
        data = default;
        try
        {
            if (!_cache.TryGetValue(key, out data))
            {
                data = getItemDelegate();
                if (data == null)
                {
                    return EntrySource.NotFound;
                }
                var cacheOptions = new MemoryCacheEntryOptions()
                    .SetSize(cacheEntrySize)
                    .SetSlidingExpirationcacheSlideExpirationHours.Value)
                    .SetAbsoluteExpirationccheAbsoluteExpirationHours.Value)
                    .RegisterPostEvictionCallback(EvictionDelegate);
                _lock.EnterWriteLock();
                _cache.Set(key, data, cacheOptions);
                _lock.ExitWriteLock();
                return EntrySource.Source;
            }
            return EntrySource.Cache;
        }
        catch (Exception ex)
```

```
            {
                return EntrySource.Failed;
            }
            finally
            {
                _lock.ExitUpgradeableReadLock();
            }
        }
        public void RemoveItem(string key)
        {
            _lock.EnterWriteLock();
            try
            {
                _cache.Remove(key);
            }
            catch (Exception ex)
            { }
            finally
            {
                _lock.ExitWriteLock();
            }
        }
        private void EvictionDelegate(object key, object? value, EvictionReason reason, object? state)
        {
            if (value == null || reason == EvictionReason.None)
            {
                return;
            }
```

```
            EvictionCallback?.Invoke(key, value, reason);
        }
    }.
```

Уся робота з базою даних MongoDB виконується за допомогою бібліотеки MongoDb.Driver. Для початку необхідно створити клас, який міститиме клієнта для підключення до бази даних. Згідно з документацією [23], рекомендовано реалізовувати даний клас згідно патерну Singleton.

Наступним кроком є отримання екземпляра об'єкта, який реалізує інтерфейс IMongoDatabase. Цей об'єкт надає доступ до бази даних і дозволяє працювати з її колекціями.

Останній етап — доступ до конкретної колекції. Він здійснюється через екземпляр об'єкта, який реалізує інтерфейс IMongoCollection<TType>. Цей об'єкт можна отримати за допомогою IMongoDatabase.

Реалізація взаємодії з базою даних здійснена у класі MongoConnector:

```
public class MongoConnector
{
    private readonly IMongoClient _client;
    private readonly IMongoDatabase _database;
    private MongoConnector()
    {
        _client = new MongoClient(Environment.GetEnvironmentVariable(Consts.MongoConnectionString));
        _database = _client.GetDatabase(Consts.DataBaseName);
    }
    private static MongoConnector _instance;
    private static readonly object _lock = new object();
    public static MongoConnector GetInstance()
    {
        if (_instance == null)
```

```
        {
            lock (_lock)
            {
                if (_instance == null)
                {
                    _instance = new MongoConnector();
                }
            }
        }
        return _instance;
    }
    public IMongoCollection<T> GetCollection<T>(string name)
    {
        return _database.GetCollection<T>(name);
    }
}.
```

Важливим елементом системи є обробка вхідних запитів від усіх користувачів, розмежування окремих користувачів та їх авторизація.

Модуль обробки запитів пердчтавлений у вигляді API, що реалізує інтерфейс взаємодії між користувачами та системою моніторингу навчання. Його основними завданнями є прийом запитів з боку клієнта, перенаправлення їх до відповідних сервісів, виконання обчислень або обробки та повернення результатів у відповідному форматі, наприклад, JSON [24].

API реалізовано на основі REST-архітектури [25], що забезпечує легкість інтеграції, масштабованість та зручність використання. Всі запити обробляються через центральний сервер з використанням відповідних маршрутизаторів.

API включає у свою структуру: роутинг, контролери, сервіси та форматування відповіді. Роутинг відповідає за прийом і розподіл запитів за функціональністю, а також за верифікацію та перевірку вхідних параметрів. Контролери здійснюють обробку конкретних записів та викликають відповідні

сервіси для виконання бізнес-логіки. Сервіси – логіка обробки запитів, взаємодія з базою даних та кешем або іншими серверами. Форматування відповіді потрібне для приведення інформації до стандартизованого та фіксованого вигляду аби користувач міг обробляти інформацію без помилок.

Запити REST API включають у себе такі види як: GET, POST, PUT, PATCH, DELETE [26]. Нижче наведено список деяких запитів до API.

Запити для управління користувачами:

- POST /users/register – реєстрація нового користувача;
- POST /users/login – аутентифікація користувача;
- GET /users/{user_id} – отримання інформації про користувача;
- PUT /users/{user_id} – оновлення інформації користувача;
- DELETE /users/{user_id} – видалення користувача.

Список запитів для роботи з тестами:

- POST /tests – створення нового тесту;
- GET /tests – отримання списку доступних тестів;
- GET /tests/{test_id} – отримання деталей тесту;
- PUT /tests/{test_id} – оновлення інформації про тест;
- DELETE /tests/{test_id} – видалення тесту;
- POST /tests/{test_id}/submit – надсилання відповідей на тест;
- GET /tests/{test_id}/results/{user_id} – отримання результатів тесту для конкретного користувача.

Запити для рейтингування та аналітики:

- GET /analytics/leaderboard – отримання загального рейтингу студентів;
- GET /analytics/user/{user_id} – аналітика навчального прогресу конкретного студента;
- GET /analytics/recommendations/{user_id} – рекомендації для студента на основі результатів.

Деякі адміністративні запити:

- POST /admin/config – оновлення конфігурації системи;

– DELETE /admin/reset – скидання всіх даних (тільки для тестування).

Взаємодія з кешом (тільки для адміністраторів):

– GET /cache/clear – очистка кешу;

– GET /cache/status – отримання стану кешу.

Для зберігання інформації про поточний стан кожного користувача та розділення усіх, хто одночасно користується системою необхідно створювати сесії для кожного з них і керувати ними. Для цього було створено клас SessionManager, який відповідає за створення сесії і зберігання її в актуальному стані відокремлено від усіх інших сесій, не допустивши колізії. Реалізація класу виглядає наступним чином:

```
public class SessionManager
{
    public bool EstablishSession(Message message, CancellationToken token, out ISession session)
    {
        return InternalEstablishSession(message.From, token, out session);
    }
    public bool EstablishSession(CallbackQuery callback, CancellationToken token, out ISession session)
    {
        return InternalEstablishSession(callback.From, token, out session);
    }
    private bool InternalEstablishSession(User user, CancellationToken token, out ISession session)
    {
        session = GetOrCreateSession(user.Id, out var source);
        if (session == null)
        {
            SendFailureMessageToClient(user.Id, user.LanguageCode, "FailureMessage", token).Wait();
```

```
            return false;
        }
        var differ = session.UserInfo.CompareAndUpdateUserData(user);
        if (differ)
        {
            _userDb.UpdateUser(session.UserInfo);
        }
        return true;
    }
    public ISession GetOrCreateSession(long userId, out EntrySource source, Func<ISession> getInstance = null)
    {
        var key = Key.GetCacheKey<ISession>(userId.ToString());
        var func = getInstance == null ? CreateSession : getInstance;
        source = _cache.GetOrAdd(key, func, out ISession session);
        return session;
    }
    public bool IsSessionExists(long userId)
    {
        var key = Key.GetCacheKey<ISession>(userId.ToString());
        var source = _cache.GetOrAdd(key, () => null, out ISession session);
        return source switch
        {
            EntrySource.NotFound => false,
            EntrySource.Failed => false,
            _ => true
        };
    }
    public ISession CreateSession(long id)
    {
```

```
            var user = _userDb.GetUser(id);
            return user;
        }
        public ISession CreateSession(EduUser user)
        {
            return new Session(user);
        }
        private void SessionEvictionFromCache(object key, object value,
EvictionReason reason)
        {
            if (!key.ToString().Contains(Consts.SessionCachePrefix))
            {
                return;
            }
            var res = _cache.GetOrAdd<bool>("ShutdownOptions", () => false, out var
isShutDown);
            if (value is ISession session && session.SessionData.MenuMsg !=
Consts.NoMessage && isShutDown)
            {
                DeleteMessage(session.UserInfo.Id, session.SessionData.MenuMsg,
_botToken).Wait();
                var msg = LocalizationManager.GetMessage("RenewSession",
session.UserInfo.LanguageCode);
                SendMessage(session.UserInfo.Id, msg, cancellationToken: _botToken,
disableNotification: true).Wait();
            }
        }
    }.
```

Метод EstablishSession вказаного класу ініціює створення нової сесії, якщо такої не було створено раніше. Метод GetOrCreateSession здійснює запит для

отримання інформації про користувача з указаного ресурсу. Метод IsSessionExists призначений для перевірки існування сесії певного користувача в певний момент часу. Приватний метод SessionEvictionFromCache призначений для очищення сесії після її видалення з кешу після того, як спливає зазначений час існування об’єкту.

Такий підхід дозволить ефективно оперувати даними і одночасно обслуговувати багато користувачів одночасно.

Отож розроблений програмний засіб відповідає усім поставленим задачам та реалізує у собі створені методи, алгоритми та відповідає розробленій схемі.

3.3 Тестування роботи розробленого програмного забезпечення та аналіз результатів

Для перевірки функціонування розробленого програмного забезпечення було створено та завантажено 3 різних тести. Один з тестів мав 50 питань, не містив реальних даних і мав всього 4 однакових варіанти відповіді. Довжина питання 500 знаків.

Для тестування в якості інтерфейсу користувача було обрано соціальну мережу Телеграм. З цією метою було створено бота, що взаємодіє з системою. Вибір Телеграму обґрунтований популярністю даної соціальної мережі та легкодоступністю ресурса.

Головне меню зображене на рисунку 3.1 і містить такі пункти: профіль, тести, класи, статистика, налаштування, допомога. Кожен пункт містить свої підпункти, що відповідають API запитам. Для тестування знадобиться Лише пункт меню «Тести».

Реєстрація користувача відбувається автоматично при отриманні стартової команди «/start». Тому для тестування користувачі одразу переходять до необхідного меню «Тести». Далі для пошуку підготованого тесту натискають на меню «Пошук» та надсилають повідомлення з назвою «Test-Set-01».

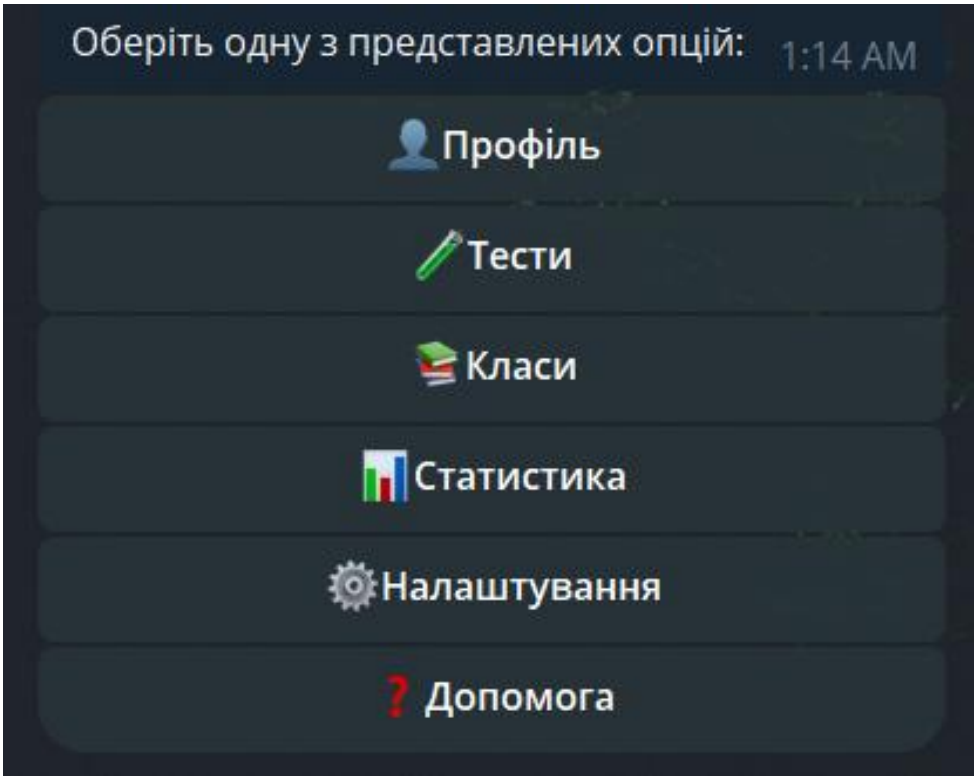

Рисунок 3.1 – Головне меню телеграм бота для взаємодії з системою

На рисунку 3.2 зображений процес проходження тесту користувачем.

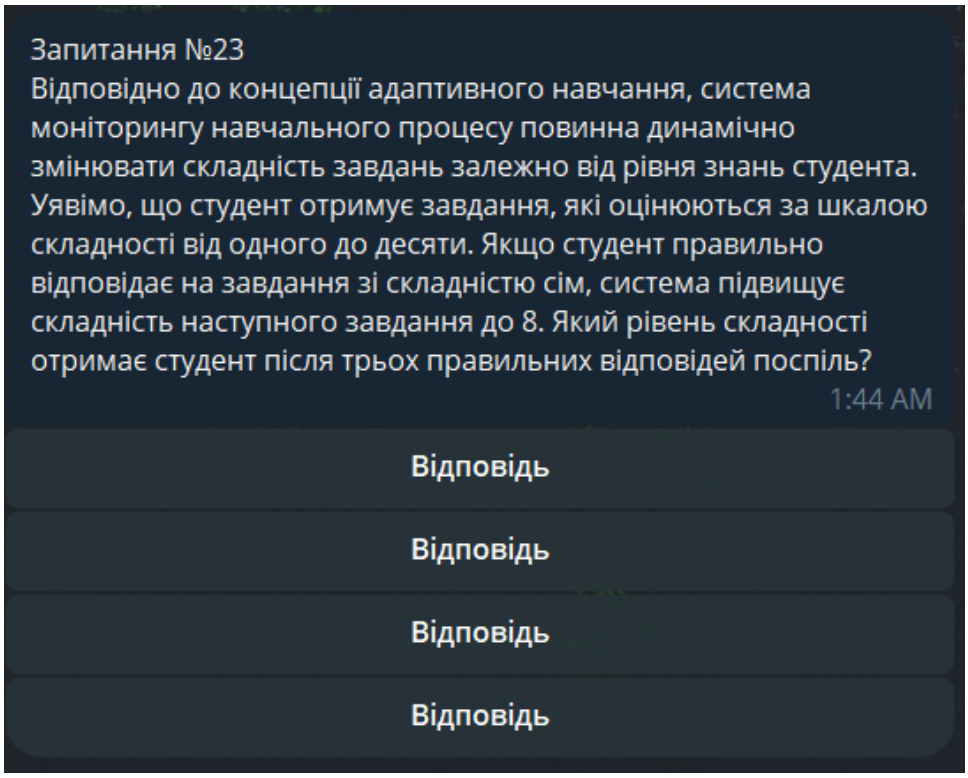

Рисунок 3.2 – Процес проходження тесту користувачем

Після зібрання результатів усіх проходжень тестів перше в чому впевнились це у стабільності роботи системи. Зібрані відгуки користувачів вказують на те, що система адаптується відповідно до рівня заннь опитуваних і надає проблемні питання частіше, порівняно з тими на які користувачі успішно і швидко відповідають.

Для розрахунку покращення ефективності перш за все розрахуємо статистику правильних відповідей без відсіювання випадкових відповідей. В такому випадку система не враховує час проходження і приймає абсолютно усі відповіді і враховує їх як правильні або ж ні. Враховувати потрібно лише правильно надані відповіді, оскільки така відповідь була надана випадково без реального знання, то це знижує рівень рекомендацій через похтбку в рейтингуванні. Ймовірність правильно наданої відповіді при чотирьох варіантах відповіді становить:

$$P_{\text{правильно}} = \frac{1}{4} = 0.25. \quad (3.1)$$

Якщо система не відсіює випадкові відповіді, то кожен користувач надає у середньому 25% правильних відповідей завдяки випадковості. Серед усіх 50 відповідей у середньому буде становити:

$$50 \times 0.25 = 12.5$$

Для 50 користувачів загальна кількість правильних відповідей без відсіювання буде становити:

$$50 \times 12{,}5 = 625$$

Згідно отриманих даних тестування система відсіює 30% випадково наданих відповідей. Це досягається завдяки тому, що користувачі не знаючи

відповіді, пропускають запитання вибором випадкової відповіді. Система фіксує, що час надання відповіді становить менше часу ніж необхідно було на прочитання запитання і фіксує таку відповідь як неправильну, навіть якщо варіант відповіді був правильний.

Ймовірність правильної відповіді з урахуванням відсіювання:

$$P_{\text{сис.правильно}} = P_{\text{правильно}} \times (1 - 0{,}3) = 0{,}25 \times 0{,}7 = 0{,}175. \quad (3.2)$$

Розрахунок кількості правильних відповідей в середньому для одного користувача:

$$50 \times 0{,}175 = 8{,}75.$$

Розрахунок кількості правильних відповідей для 50 користувачів:

$$50 \times 8{,}75 = 437{,}5.$$

Тепер маючи дані будь якої системи, яка не має відсіювання випадково наданих правильних відповідей та розробленої, обчислимо ефективність зменшення похибки. Ефективність визначається як відносне зменшення кількості випадково правильних відповідей:

$$\text{Ефективність} = \left(\frac{\text{Похибка аналогів} - \text{Похибка системи}}{\text{Похибка аналогів}}\right) \times 100\%, \quad (3.3)$$

де Похибка аналогів – кількість випадково правильно наданих відповідей в системі без відсіювання, Похибка системи – кількість випадково правильно наданих відповідей в розробленій системі.

Обчислимо ефективність:

$$\text{Ефективність} = \left(\frac{625-437,5}{625}\right) \times 100\% = \left(\frac{187,5}{625}\right) \times 100\% \approx 30\%.$$

Розроблена система знижує похибку рейтингування при випадково правильних відповідях на 30% завдяки механізму відсіювання відповідей. В таблиці 3.3 показано числове порівняння розробленої системи та систем аналогів, включаючи Moodle, Google Classroom, Microsoft Teams, Blackboard, Canvas.

Таблиця 3.3 – Порівняння ефективності сучасних засобів тестування

Характеристика	Розроблена інформаційна технологія	Аналоги (Moodle, Google Classroom, Microsoft Teams, Blackboard, Canvas)
Ймовірність випадкового надання правильної відповіді	25%	25%
Кількість випадково наданих правильних відповідей	437,5	625
Відсоток відсіювання випадково наданих правильних відповідей	30%	0%
Ефективність розробленої системи відносно аналогів	30%	

Під час тестування було встановлено, що система демонструє високу ефективність у зменшенні похибки рейтингування при випадково правильних

відповідях завдяки використанню вдосконаленої математичної моделі. Середнє зниження похибки склало 30% порівняно з аналогами.

3.4 Висновок до розділу 3

У цьому розділі було розглянуто основні етапи розробки складових інформаційної технології моніторингу навчального процесу, що забезпечують ефективний збір, обробку та аналіз даних про діяльність учнів. Було визначено ключові компоненти технології, такі як системи збору інформації, алгоритми її обробки, а також інтерфейси для надання результатів моніторингу користувачам, зокрема викладачам та адміністрації навчального закладу. Окрему увагу приділено вибору відповідних програмних засобів, що дозволяють автоматизувати процеси моніторингу та аналізу результатів, забезпечуючи точність і оперативність отриманих даних.

Розроблена інформаційна технологія виконала поставлені задачі та практично показала підвищення ефективності на 30% порівняно з існуючими аналогами, що були розглянуті у роділі 1.1.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Проведення комерційного та технологічного аудиту інформаційної технології моніторингу процесу навчання.

Метою проведення комерційного і технологічного аудиту є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності, тобто під час виконання магістерської кваліфікаційної роботи.

Для проведення комерційного та технологічного аудиту залучаємо 3-х незалежних експертів, якими є провідні викладачі випускової або спорідненої кафедри.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу здійснюємо із застосуванням п’ятибальної системи оцінювання за 12-ма критеріями, а результати зводимо до таблиці 4.1.

За результатами розрахунків, наведених в таблиці 1 робимо висновок про те, що науково-технічний рівень та комерційний потенціал інформаційної технології моніторингу процесу навчання – вище середнього.

Таблиця 4.1 – Результати оцінювання науково-технічного рівня і комерційного потенціалу засобу інформаційної технології моніторингу процесу навчання

Критерії	Експерти		
	Експерт 1	Експерт 2	Експерт 3
	Бали, виставлені експертами		
1	2	3	4
Технічна здійсненність концепції	3	3	3
Ринкові переваги (наявність аналогів)	2	2	3
Ринкові переваги (ціна продукту)	3	2	3

Продовження таблиці 4.1

1	2	3	4
Ринкові переваги (технічні властивості)	3	2	3
Ринкові переваги (експлуатаційні витрати)	2	3	3
Ринкові перспективи (розмір ринку)	3	3	3
Ринкові перспективи (конкуренція)	2	3	3
Практична здійсненність (наявність фахівців)	3	2	3
Практична здійсненність (наявність фінансів)	2	3	2
Практична здійсненність (необхідність нових матеріалів)	3	3	3
Практична здійсненність (термін реалізації)	3	3	3
Практична здійсненність (розробка документів)	3	3	3
Сума балів	32	32	35
Середньоарифметична сума балів, СБ	33		

4.2 Розрахунок витрат на здійснення науково-дослідної роботи розробки інформаційної технології моніторингу процесу навчання

Належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої

за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці, також будь-які види грошових і матеріальних доплат, які належать до елемента «Витрати на оплату праці».

Витрати на основну заробітну плату дослідників ($З_о$) розраховують відповідно до посадових окладів працівників, за формулою:

$$З_о = \sum_{i=1}^{к} \frac{М_{ni} \cdot t_i}{Т_р}, \quad (4.1)$$

де к – кількість посад дослідників, залучених до процесу дослідження; $М_{ni}$ – місячний посадовий оклад конкретного розробника (інженера, дослідника, науковця тощо), грн.; $Т_р$ – число робочих днів в місяці; приблизно $Т_р$ = (21…23) дні; t_i – число робочих днів роботи розробника (дослідника).

Зроблені розрахунки зводимо до таблиці 4.2.

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт розраховують за формулою (4.2).

Таблиця 4.2 – Витрати на заробітну плату дослідників

Посада	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник	35000	1591	22	35000
Розробник	35000	1591	22	35000
Консультанти	20000	909	18	1636
Всього:	71636			

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i, \quad (4.2)$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год; t_i – час роботи робітника на виконання певної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду С і можна визначити за формулою:

$$C_i = \frac{М_м \cdot К_i \cdot К_c}{Т_р \cdot t_{зм}}, \quad (4.3)$$

де $М_м$ – розмір прожиткового мінімуму працездатної особи або мінімальної місячної заробітної плати (залежно від діючого законодавства), у 2024 році $М_м$=8000 грн; $К_i$ – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду; $К_с$ – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об'єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати; $Т_р$ – середня кількість робочих днів в місяці, приблизно $Т_р$ = 21…23 дні; $t_{зм}$ – тривалість зміни, год.

Витрати на заробітну плату робітників подана у таблиці 4.3.

Додаткова заробітна плата $З_д$ всіх розробників та робітників, які брали участь у виконанні даного етапу роботи, розраховується як (10…12)% від суми основної заробітної плати всіх розробників та робітників, як показано у формулі:

$$З_д = 0{,}1 \cdot (З_о + З_р) = 0{,}12 \cdot (71636 + 19185) = 10899 \text{ грн.} \quad (4.4)$$

Таблиця 4.3 – Витрати на заробітну плату робітників

Найменування робіт	Трудомісткість, н-год.	Розряд роботи	Погодинна тарифна ставка	Тариф. коеф.	Величина, грн.
Аналіз предметної області	23	5	61,8	1,36	1422
Моделювання інформаційної технології	113	5	61,8	1,36	6983
Створення основної структури проєкту	18	6	65,9	1,45	1186
Створення та наповнення бази даних	79	6	65,9	1,45	5206
Тестування інформаційної технології	71	5	61,8	1,36	4388
Всього					19185

Відрахування на соціальні заходи. Нарахування на заробітну плату $Н_{зп}$ розробників та робітників, які брали участь у виконанні даного етапу роботи, розраховуються за формулою:

$$Н_{зп} = \beta \cdot (З_о + З_р + З_д) = \\ = 0{,}22 \cdot (71636 + 19185 + 10899) = 22378 \text{ грн}, \quad (4.5)$$

де $З_о$ – основна заробітна плата розробників, грн.; $З_р$ – основна заробітна плата робітників, грн.; $З_д$ – додаткова заробітна плата всіх розробників та робітників, грн.; β – ставка єдиного внеску на загальнообов'язкове державне соціальне страхування, % (приймаємо для 1-го класу професійності ризику 22%).

До балансової вартості програмного забезпечення входять витрати на його інсталяцію, тому ці витрати беруться додатково в розмірі 10…12% від вартості програмного забезпечення. Балансову вартість програмного забезпечення розраховують за формулою:

$$В_{прг} = \sum_{1}^{к} Ц_{іпрг} \cdot С_{прг.і} \cdot К_і, \quad (4.6)$$

де $Ц_{іпрг}$ – ціна придбання програмного забезпечення і-го виду, грн.; $С_{прг.і}$ – кількість одиниць програмного забезпечення відповідного виду, шт.; $К_і$ – коефіцієнт, що враховує інсталяцію, налагодження програмного забезпечення, $К_і$ = (1,1…1,12); к – кількість видів програмного забезпечення.

Витрати на придбання програмного забезпечення подані у таблиці 4.4.

Таблиця 4.4 – Витрати на придбання програмного забезпечення

Найменування програмного забезпечення	Ціна за одиницю, грн.	Витрачено	Вартість програмного забезпечення, грн.
VS 2022 Enterprise	8299,5	1	8299,5
Dedicated MongoDB Cluster	2390	1	2390
GMHost	180,08	1	180,08
Всього, з врахуванням коефіцієнта інсталяції та налагодження			10870

Амортизація обладнання, комп’ютерів та приміщень, які використовувались під час (чи для) виконання даного етапу роботи.

У спрощеному вигляді амортизаційні відрахування А в цілому бути розраховані за формулою:

$$А = \frac{Ц_б}{Т_в} \cdot \frac{t}{12}, \quad (4.7)$$

де $Ц_б$ – загальна балансова вартість всього обладнання, комп’ютерів, приміщень тощо, що використовувались для виконання даного етапу роботи, грн.; t – термін використання основного фонду, місяці; $Т_в$ – термін корисного використання основного фонду, роки.

Амортизаційні відрахування за видами основних фондів подані у таблиці 4.5.

Таблиця 4.5 – Амортизаційні відрахування за видами основних фондів

Найменування	Балансова вартість, грн.	Строк корисного використання, років	Термін використання, місяців	Сума амортизації, грн.
Комп’ютер	40000	2	1,5	2500
Монітор	10000	2	1,5	625
Стіл	8000	5	1,5	200
Стілець	4000	5	1,5	100
Мишка	1000	2	1,5	62,5
Клавіатура	1200	2	1,5	75
Настільна лампа	1250	3	1,5	52
Всього	3614,5			

Витрати на електроенергію для науково-виробничих цілей подані в таблиці 4.6. Витрати на силову електроенергію $В_е$, якщо ця стаття має суттєве значення для виконання даного етапу роботи, розраховуються за формулою (4.8).

Таблиця 4.6 – Витрати на електроенергію

Найменування обладнання	Потужність, кВт	Тривалість годин роботи
Комп'ютер	0,8	198
Освітлення	0,05	200

$$Be = \sum \frac{W_i \cdot t_i \cdot Ц_е \cdot К_{впі}}{ККД} = \frac{0,8 \cdot 198 \cdot 7,5 \cdot 0,85}{0,98} + \\ + \frac{0,05 \cdot 200 \cdot 7,5 \cdot 0,85}{0,98} = 1095 \text{ грн}, \quad (4.8)$$

де W_i – встановлена потужність обладнання, кВт; t_i – тривалість роботи обладнання на етапі дослідження, год.; Це – вартість 1 кВт електроенергії, грн.; $К_{впі}$ – коефіцієнт використання потужності; ККД – коефіцієнт корисної дії обладнання.

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$Ів = \left(З_о + З_р\right) \cdot \frac{Н_{ів}}{100\%} = (71636 + 19185) \cdot \frac{65}{100} = 59034 \text{ грн.}, \quad (4.9)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати».

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов'язані з управлінням організацією; витрати на винахідництво та

раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов'язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов'язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$\text{Внзв} = \left(З_о + З_р\right) \cdot \frac{Н_{нзв}}{100\%} = (71636 + 19185 + 59034) \cdot \frac{145}{100} = \\ = 217290 \text{ грн.}, \tag{4.10}$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати».

Витрати на проведення науково-дослідної роботи розраховуються як сума всіх попередніх статей витрат за формулою:

$$В_{заг} = З_о + З_р + З_{дод} + З_н + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + \\ + I_в + В_{нзв} = 71636 + 19185 + 10899 + 22378 + \\ + 108{,}7 + 3614{,}5 + 1095 + 59034 + 217290 = \\ = 405240 \text{ грн.} \tag{4.11}$$

Загальні витрати (ЗВ) на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються за формулою:

$$ЗВ = \frac{В_{заг}}{\eta} = \frac{405240}{0{,}9} = 450267 \text{ грн.}, \tag{4.12}$$

де η – коефіцієнт, що характеризує етап виконання науково-дослідної роботи. Оскільки, якщо науково-технічна розробка знаходиться на стадії впровадження, то η=0,9.

4.3 Розрахунок економічної ефективності науково-технічної розробки інформаційної технології моніторингу процесу навчання за її можливої комерціалізації потенційним інвестором

У ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

У даному випадку відбувається розробка засобу, тому основу майбутнього економічного ефекту буде формувати: ΔN – збільшення кількості споживачів, яким надається відповідна інформаційна послуга в аналізовані періоди часу; N – кількість споживачів, яким надавалась відповідна інформаційна послуга у році до впровадження результатів нової науково-технічної розробки; $Ц_б$ – вартість послуги у році до впровадження інформаційної системи; $\pm\Delta Ц_о$ – зміна вартості послуги (зростання чи зниження) від впровадження результатів науково-технічної розробки в аналізовані періоди часу.

Можливе збільшення чистого прибутку у потенційного інвестора ΔПі для кожного із років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховується за формулою:

$$\Delta П = (\pm\Delta Ц_о \cdot N + Ц_о \cdot \Delta N_i)_i \cdot \lambda \cdot \rho \cdot \left(1 - \frac{\vartheta}{100}\right), \qquad (4.13)$$

де ±ΔЦ – зміна основного якісного показника від впровадження результатів науково-технічної розробки в аналізованому році. Зазвичай, таким показником може бути зміна ціни реалізації одиниці нової розробки в аналізованому році (відносно року до впровадження цієї розробки); $\pm\Delta Ц_о$ може мати як додатне, так і від'ємне значення (від'ємне – при зниженні ціни відносно року до впровадження цієї розробки, додатне – при зростанні ціни); N – основний

кількісний показник, який визначає величину попиту на аналогічні чи подібні розробки у році до впровадження результатів нової науково-технічної розробки; $Ц_о$ – основний якісний показник, який визначає ціну реалізації нової науково-технічної розробки в аналізованому році; $Ц_б$ – основний якісний показник, який визначає ціну реалізації існуючої (базової) науково-технічної розробки у році до впровадження результатів; ΔN – зміна основного кількісного показника від впровадження результатів науково-технічної розробки в аналізованому році. Зазвичай таким показником може бути зростання попиту на науково-технічну розробку в аналізованому році (відносно року до впровадження цієї розробки); λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2024 році ставка податку на додану вартість становить 20%, а коефіцієнт λ =0,8333; ρ – коефіцієнт, який враховує рентабельність інноваційного продукту (послуги). Рекомендується брати ρ =0,2...0,5; ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2024 році ϑ =18%.

Очікуваний термін життєвого циклу розробки 1 рік, тому:

$$\Delta П = \big((300 - 250) \times 30000 - (21000 - 30000) \times 300\big) \times 0{,}8333 \times 0{,}3 \times$$
$$\times \left(1 - \frac{18}{100}\right) = 6244920 \text{ грн.}$$

Далі розраховують приведену вартість збільшення всіх чистих прибутків ПП, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки за формулою:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t} = \frac{6244920}{(1+0{,}1)^1} = 5677200 \text{ грн.}, \qquad (4.14)$$

де $\Delta П$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн.; Т –

період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки (приймаємо Т=1 рік); τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =0,05…0,15; t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

Далі розраховують величину початкових інвестицій *PV*, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки. Для цього можна використати формулу:

$$PV = k_{\text{інв}} \cdot \text{ЗВ} = 2 \cdot 450267 = 900534 \text{ грн.}, \quad (4.15)$$

де $k_{\text{інв}}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай kінв=2...5, але може бути і більшим; ЗВ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.

Тоді абсолютний економічний ефект $E_{\text{абс}}$ або чистий приведений дохід для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки розрахуємо за формулою:

$$E_{\text{абс}} = \text{ПП} - PV = 5677200 - 900534 = 4776666 \text{ грн.}, \quad (4.16)$$

де ПП – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, грн.; PV – теперішня вартість початкових інвестицій, грн.

Оскільки $E_{\text{абс}}>0$, то можемо припустити про потенційну зацікавленість інвесторів у розробці.

Для остаточного прийняття рішення з цього питання необхідно розрахувати внутрішню економічну дохідність Ев або показник внутрішньої норми дохідності вкладених інвестицій та порівняти її з так званою бар'єрною ставкою дисконтування, яка визначає ту мінімальну внутрішню економічну дохідність, нижче якої інвестиції в будь-яку науково-технічну розробку вкладати буде економічно недоцільно.

Внутрішня економічна дохідність інвестицій Ев, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки, розраховується за формулою:

$$E_в = \sqrt[T_ж]{1 + \frac{Еабс}{PV}} - 1 = \sqrt[1]{1 + \frac{4776666}{900534}} - 1 = 1{,}5, \quad (4.17)$$

де $T_ж$ – життєвий цикл розробки, роки.

Визначимо бар'єрну ставку дисконтування $\tau_{мін}$, тобто мінімальну внутрішню економічну дохідність інвестицій, нижче якої кошти у впровадження науково-технічної розробки та її комерціалізацію вкладатися не будуть.

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$ визначається за формулою:

$$\tau_{мін} = d + f = 0{,}9 + 0{,}05 = 0{,}95, \quad (4.18)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2024 році в Україні d =0,9...0,12; f – показник, що характеризує ризикованість вкладення інвестицій; зазвичай величина f =0,05...0,5, але може бути і значно вищою.

Оскільки $E_в$=1,5>$\tau_{мін}$=0,95, то потенційний інвестор може бути зацікавлений у фінансуванні впровадження науково-технічної розробки та виведенні її на ринок, тобто в її комерціалізації.

Далі розраховуємо період окупності інвестицій $T_о$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки за формулою:

$$T_о = \frac{1}{E_в} = \frac{1}{1,5} = 0,67 \text{ року.} \quad (4.19)$$

Оскільки $T_о$<1…3-х років, то це свідчить про комерційну привабливість науково-технічної розробки інформаційної технології моніторингу процесу навчання і може спонукати потенційного інвестора профінансувати впровадження цієї розробки та виведення її на ринок.

4.4 Висновок до розділу 4

Економічна частина даної роботи містить розрахунок витрат на розробку інформаційної технології моніторингу процесу навчання, сума якої складає 900534 гривень. Було спрогнозовано орієнтовану величину витрат по кожній з статей витрат. Також розраховано чистий прибуток, який може отримати виробник від реалізації нового технічного рішення, розраховано період окупності витрат для інвестора та економічний ефект при використанні даної розробки.

У результаті аналізу розрахунків можна зробити висновок, що розроблений програмний продукт за ціною дешевший за аналоги і є високо конкурентоспроможним. Період окупності складе близько 0,67 роки.

ВИСНОВКИ

Для досягнення мети магістерської кваліфікаційної роботи було виконано такі задачі: проведено аналіз існуючих методів та засобів моніторингу процесу навчання, визначено вимоги до інформаційної технології моніторингу процесу навчання, розроблено математичну модель інформаційної технології моніторингу процесу навчання, розроблено структуру інформаційної системи підтримки прийняття рішень процесу навчання студентів, виконано програмну реалізацію інформаційної технології моніторингу процесу навчання, проведено тестування програмного забезпечення та проаналізовано отримані результати, проведено розрахунок економічної ефективності науково-технічної розробки за можливого її застосування.

У результаті виконання усіх поставлених задач було підвищено ефективність процесу навчання студентів за допомогою розробленого методу незалежного оцінювання успішності студента, який відрізняється від існуючих використанням удосконаленої математичної моделі тестового рейтингування, що дозволяє знизити похибку рейтингування при наданні випадково правильної відповіді на 30%, а також підвищити адаптивність системи до рівня підготовки студентів розробки рекомендаційної системи, що сприяє покращенню якості освіти шляхом надання рекомендацій щодо матеріалу який потребує подальшого вивчення.

Згідно проведених економічних розрахунків сума витрат на розробку та виготовлення нового технічного рішення складає 900534 гривень. Було спрогнозовано орієнтовану величину витрат по кожній з статей витрат. Також розраховано чистий прибуток, який потенційно може отримати виробник від реалізації розробленого технічного рішення та знайдено термін окупності витрат виробника, який складе близько 0,67 роки.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Сєркова Л. Е. Інформаційна технологія моніторингу організації учбового процесу вищого навчального закладу: дис. д-ра філософії. Черкаси, 2006.

2. Столяренко І. С. Системи управління навчанням. Звітна наукова конференція Інституту інформаційних технологій і засобів навчання НАПН України: Матеріали наук. конф., м. Київ, 19 берез. 2015 р. Київ, 2015. С. 141.

3. Вишневський А. В., Іванчук Я. В. Математична модель класифікації відповіді при рейтингуванні тестів // Молодь в науці: дослідження, проблеми, перспективи. 2025. URL: https://conferences.vntu.edu.ua/index.php/mn/mn2025/paper/viewFile/22581/18649.

4. Савчук Т. О., Вишневський А. В., Ольшанська О. В. Використання коефіцієнту класифікації відповіді при рейтингуванні тестів // Science and technology: problems, prospects and innovations. Proceedings of the 9th International scientific and practical conference. CPN Publishing Group. Osaka, Japan. 2023. Pp. 213-218. URL: https://sci-conf.com.ua/ix-mizhnarodna-naukovo-praktichna-konferentsiya-science-and-technology-problems-prospects-and-innovations-8-10-06-2023-osaka-yaponiya-arhiv/.

5. Савчук Т. О., Вишневський А. В. Розробка удосконаленого алгоритму рейтингування тестів // European scientific congress. Proceedings of the 5th International scientific and practical conference. Barca Academy Publishing. Madrid, Spain. 2023. Pp. 136-142. URL: https://sci-conf.com.ua/v-mizhnarodna-naukovo-praktichna-konferentsiya-european-scientific-congress-12-14-06-2023-madrid-ispaniya-arhiv/.

6. Савчук Т. О., Вишневський А. В. Свідоцтво про реєстрацію авторського права на твір вх.№ С202304024. Комп'ютерна програма «Програмне забезпечення для моніторингу прогресу навчання» // Т. О. Савчук, А. В. Вишневський (Україна), Національний орган інтелектуальної власності

державна організація «Український національний офіс інтелектуальної власності та інновацій» (УКРНОВІ) – Дата реєстрації 31-05-2023 р.

7. Кравцов Г. М. Система моніторингу якості електронних інформаційних ресурсів вузу. Інформаційні технології в освіті. 2008. № 2. С. 42–46.

8. Бонч-Брує́вич Г. Ф. Технічні засоби навчання з використанням інформаційниї комп'ютерних технологій: Навч. посіб. – К.: КМПУ імені Б. Д. Грінченка, 2007. – 64 с.

9. Семеняко Ю., Брюховецька І., Бохонько Є. Цифровізація у вищій освіті: інституційні підходи до викладання та навчання. Педагогіка формування творчої особистості у вищій і загальноосвітній школах. 2023.

10. Грушко Н. Використання інформаційних технологій у процесі навчання математики. Slideshare. URL: https://www.slideshare.net/NataliaGrychko/ss-25462259.

11. Moodle. URL: https://moodle.org/?lang=uk.

12. Google Classroom. URL: https://sites.google.com/view/classroom-workspace/.

13. Blackboard. URL: https://www.anthology.com/products/teaching-and-learning/learning-effectiveness/blackboard.

14. Canvas LMS. URL: https://www.instructure.com/canvas.

15. Microsoft Teams LMS. URL: https://support.microsoft.com/en-au/topic/your-lms-and-teams-better-together-for-distance-learning-35e3c70f-11b7-447d-a4d4-3964b27911ae.

16. Голенко М. Ю., Вакалюк Т. А. Область застосування та види рекомендаційних систем // Державний університет «Житомирська політехніка». URL: https://conf.ztu.edu.ua/wp-content/uploads/2021/01/41-2.pdf.

17. Glickman M. E. Exampleofthe Glicko-2 system. URL: http://www.glicko.net/glicko/glicko2.pdf.

18. Glickman M. E. The Glicko system. URL: http://www.glicko.net/glicko/glicko.pdf.

19. Швидкочитання для дорослих: яка норма читання, підрахунок кількості слів за хвилину. Швидкочитання для дорослих: вправи, поради та рекомендації, література. Irinin Journal. 2021. URL: https://irinin.com/domivka/shvidkochitannya-dlya-doroslikh-yaka-norma-chitannya-pidrakhunok-kilkosti-sliv-za-khvilinu-shvidko chitannya-dlya-doroslikh-vpravi-poradi-ta-rekomendatsiji-literatura.html#shvydkochytannia-dlia-doroslykh-iakoiu-maie-buty-norma.

20. Visual Studio 2022. Microsoft. URL: https://visualstudio.microsoft.com.

21. Visual Studio Code. Microsoft. URL: https://code.visualstudio.com/.

22. IntellJ Idea. JetBrains. URL: https://www.jetbrains.com/idea/.

23. MongoDB C# Driver. MongoDB. URL: https://www.mongodb.com/docs/drivers/csharp/current/.

24. JSON. URL: https://www.json.org/json-en.html.

25. What is REST? Codeacademy team. URL: https://www.codecademy.com/article/what-is-rest.

26. Lokesh Gupta. HTTP Methods. URL: https://restfulapi.net/http-methods/.

27. Методичні вказівки до виконання магістерських кваліфікаційних робіт для студентів спеціальності 122 «Комп’ютерні науки» // уклад.: А. А. Яровий, О. К. Колесницький. – Вінниця: ВНТУ, 2023. – 58 с.

28. Сучасні інформаційні технології у сфері штучного інтелекту: електронний навчальний посібник комбінованого (локального та мережного) використання // Яровий А. А., Крилик Л. В., Козловський А. В. – Вінниця: ВНТУ, 2023. – 145 с.

29. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт // Уклад.: В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця: ВНТУ, 2021. – 42 с.

Додаток А (обов'язковий)

Протокол перевірки кваліфікаційної роботи на наявність текстових запозичень

Назва роботи: Інформаційна технологія моніторингу процесу навчання
Тип роботи: магістерська кваліфікаційна робота
(БДР, МКР)

Підрозділ кафедра комп'ютерних наук, ФІТА
(кафедра, факультет)

Показники звіту подібності Turnitin

Оригінальність 97% Схожість 3%

Аналіз звіту подібності (відмітити потрібне):

- ✓ Запозичення, виявлення у роботі, оформлені коректно і не містять ознак плагіату.
- □ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.
- □ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Ознайомлені з повним звітом подібності, який був згенерований системою Turnitin щодо роботи.

Автор роботи _____ Вишневський А.В.

Керівник роботи _____ Іванчук Я.В.

Опис прийнятого рішення

Магістерську кваліфікаційну роботу допущено до захисту

Особа, відповідальна за перевірку _____ Озеранський В.С.

Додаток Б (обов'язковий)

Лістинг програми

```
using Common;
using Common.Interfaces;
using TestingBot.Handlers;
using TestingBot.Providers;

namespace TestingBot
{
    internal class Program
    {
        static async Task Main(string[] args)
        {
            try
            {
                ILogManager logManager = LogManager.Instance;
                IConfigManager cfg = ConfigMgr.Instance;

                //Call manually first time due to LogManager initialized prior
ConfigMgr
                logManager.UpdateConfigEventSubscriber();
                cfg.OnConfigChanged += logManager.UpdateConfigEventSubscriber;

                IInMemoryCache cache = InMemoryCache.Instance;
                IBotHost botHost = new
BotHost(Environment.GetEnvironmentVariable(Consts.ApiTokenEnvVar), logManager);

                ISessionManager sessionManager = new SessionManager(botHost.BotClient,
botHost.BotToken, logManager, cache);
                ICommandHandler commandHandler = new CommandHandler(sessionManager,
logManager);
                IMenuHandler menuHandler = new MenuHandler(sessionManager,
commandHandler, logManager);
                IMessageHandler messageHandler = new MessageHandler(commandHandler,
sessionManager, logManager);
                ICallbackHandler callbackHandler = new CallbackHandler(commandHandler,
sessionManager, menuHandler, logManager);
                IBotHandlers botHandlers = new BotHandlers(messageHandler,
callbackHandler, cache, cfg, logManager);

                botHost.OnStart += botHandlers.OnBotStart;
                botHost.OnStop += botHandlers.OnBotStart;
                botHost.UpdateHandler += botHandlers.UpdateHandler;
                botHost.ExceptionHandler += botHandlers.ErrorHandler;

                botHost.StartBot();

                await botHandlers.KeepAlive(botHost.BotClient, botHost.BotToken);

                botHost.StopBot();
            }
            catch (Exception e)
            {
                LogManager.Instance.Log(nameof(Program), nameof(Main), e);
            }
        }
    }
}
```

```
namespace Common
{
    public static class Consts
    {
        public const string SysCorrelation = "sys";

        public const string ApiTokenEnvVar = "BotToken";
        public const string BotTag = "EdutestTrackBot";

        public const string KeepAliveMessage = "Keep alive message #{0}";
        public const string ShutdownOptions = "ShutdownOptions";
        public const string OnBotStartupMessage = "Bot started.";
        public const string OnBotShutdownMessage = "Bot stopped.";

        public const string ObjectCachePrefix = "Object#";
        public const string SessionCachePrefix = "Session#";
        public const string TestCachePrefix = "Test#";
        public const string ClassCachePrefix = "Class#";
        public const string TestCreationCachePrefix = "TestCreation#";
        public const string ClassCreationCachePrefix = "ClassCreation#";

        public const int NoMessage = -1;

        public const string CallbackPrefix = "Callback#";
        public const string MenuOptionsPrefix = "MenuOption";
        public const int MenuOptions = 10;
        public const int MaxMenuDepth = 3;
        public const int BackMenuOptionId = 9999;
        public const int CancelMenuOptionId = 9998;
        public const int MainMenuStartIndex = 0;
        public const int ProfileMenuStartIndex = 10;
        public const int HelpMenuStartIndex = 20;
        public const int ClassroomMenuStartIndex = 30;
        public const int SettingsMenuStartIndex = 40;
        public const int StatisticsMenuStartIndex = 50;
        public const int TestsMenuStartIndex = 60;
    }
}

using Common;
using Common.Interfaces;
using Newtonsoft.Json;
using System.Globalization;

namespace TestingBot
{
    public class ConfigMgr : IConfigManager
    {
        private const string _directory = "config";
        private const string ModuleName = nameof(ConfigMgr);

        private Dictionary<string, object> _config;

        private readonly ReaderWriterLockSlim _lock;
        private readonly FileSystemWatcher _watcher;
        private readonly ILogManager _logManager;

        private static readonly ConfigMgr _instance = new();

        static ConfigMgr() { }
        private ConfigMgr()
        {
            if (!Directory.Exists(_directory))
            {
```

```
                Directory.CreateDirectory(_directory);
            }

            _lock = new ReaderWriterLockSlim();
            _logManager = LogManager.Instance;

            ReadConfig();

            _watcher = new
FileSystemWatcher($"{Environment.CurrentDirectory}\\{_directory}");
            _watcher.Changed += RaiseEvent;
            _watcher.EnableRaisingEvents = true;
        }

        public event Action OnConfigChanged;

        public static ConfigMgr Instance
        {
            get
            {
                return _instance;
            }
        }

        public bool GetData<TItem>(string key, out TItem item)
        {
            item = default(TItem);

            _lock.EnterReadLock();
            var res = _config.TryGetValue(key, out object cfg);
            if (res)
            {
                item = (TItem)Convert.ChangeType(cfg, typeof(TItem),
CultureInfo.InvariantCulture);
            }
            _lock.ExitReadLock();

            return res;
        }

        private void ReadConfig()
        {
            var sFuncName = nameof(ReadConfig);

            _lock.EnterWriteLock();
            var textConfig = File.ReadAllText(GetFileName());
            var options = new JsonSerializerSettings()
            {
                NullValueHandling = NullValueHandling.Ignore
            };
            var cfg = JsonConvert.DeserializeObject<Dictionary<string,
object>>(textConfig, options);

            if (_config == null && cfg == null)
            {
                _logManager.Log(MsgLevel.WARNING, ModuleName, $"{sFuncName} -
Configuration file is empty.", string.Empty, string.Empty, string.Empty,
Consts.SysCorrelation);
                _lock.ExitWriteLock();
                throw new ArgumentException(nameof(_config));
            }
            if (cfg != null)
            {
                _config = cfg;
```

```
            }
            _lock.ExitWriteLock();
        }

        private string GetFileName()
        {
            return $"{_directory}\\config.json";
        }

        private void RaiseEvent(object sender, FileSystemEventArgs e)
        {
            var sFuncName = nameof(OnConfigChanged);

            _logManager.Log(MsgLevel.INFO, ModuleName, $"{sFuncName} - Configuration
changed.", string.Empty, string.Empty, string.Empty, Consts.SysCorrelation);
            ReadConfig();
            OnConfigChanged?.Invoke();
        }
    }
}

using Common;
using Common.Interfaces;
using Common.Models;
using Microsoft.Extensions.Caching.Memory;

namespace TestingBot
{
    public class InMemoryCache : IInMemoryCache
    {
        private const int CacheEntrySize = 1;
        private const string ModuleName = nameof(InMemoryCache);

        private MemoryCache _cache;

        private readonly ReaderWriterLockSlim _lock = new ReaderWriterLockSlim();
        private readonly CacheOptions _cachingOptions;
        private readonly ILogManager _logManager;
        private readonly IConfigManager _config;

        private static readonly InMemoryCache _instance = new();

        static InMemoryCache() { }
        private InMemoryCache()
        {
            _logManager = LogManager.Instance;
            _config = ConfigMgr.Instance;
            _cachingOptions = new();
            UpdateConfigEventSubscriber();
            _cache = new MemoryCache(new MemoryCacheOptions()
            {
                SizeLimit = _cachingOptions.CacheMaxSize.Value
            });
            _config.OnConfigChanged += UpdateConfigEventSubscriber;
        }

        public static InMemoryCache Instance
        {
            get
            {
                return _instance;
            }
        }
```

```
        public event Action<object, object, EvictionReason> EvictionCallback;

        public EntrySource GetOrAdd<TItem>(string key, Func<TItem> getItemDelegate,
out TItem data, CacheItemPriority priority = CacheItemPriority.Normal)
        {
            var sFuncName = nameof(GetOrAdd);

            _lock.EnterUpgradeableReadLock();
            data = default;
            try
            {
                if (!_cache.TryGetValue(key, out data))
                {
                    data = getItemDelegate();

                    if (data == null)
                    {
                        return EntrySource.NotFound;
                    }

                    var cacheOptions = new MemoryCacheEntryOptions()
                        .SetSize(CacheEntrySize)

.SetSlidingExpiration(_cachingOptions.CacheSlideExpirationHours.Value)

.SetAbsoluteExpiration(_cachingOptions.CacheAbsoluteExpirationHours.Value)
                        .RegisterPostEvictionCallback(EvictionDelegate);

                    _lock.EnterWriteLock();
                    _cache.Set(key, data, cacheOptions);
                    _lock.ExitWriteLock();
                    _logManager.Log(MsgLevel.TRACE, ModuleName, $"{sFuncName} -
Successfully cashed item [{data}]", string.Empty, string.Empty, key);
                    return EntrySource.Source;
                }
                _logManager.Log(MsgLevel.TRACE, ModuleName, $"{sFuncName} -
Successfully get item from cache [{data}]", string.Empty, string.Empty, key);
                return EntrySource.Cache;
            }
            catch (Exception ex)
            {
                _logManager.Log(ModuleName, sFuncName, ex, string.Empty, string.Empty,
key);
                return EntrySource.Failed;
            }
            finally
            {
                _lock.ExitUpgradeableReadLock();
            }
        }

        public List<TItem> GetAll<TItem>(string partialKey)
        {
            var sFuncName = nameof(GetAll);
            _logManager.Log(MsgLevel.DEBUG, ModuleName, $"{sFuncName} - Requested to
get all items in cache type of '{typeof(TItem).Name}'.", string.Empty, string.Empty,
partialKey);

            _lock.EnterReadLock();
            var data = new List<TItem>();
            try
            {
                var keys = _cache.Keys.Where(k =>
k.ToString().Contains(partialKey)).ToList();
```

```
                if (keys.Count != 0)
                {
                    foreach (var key in keys)
                    {
                        var dataItem = _cache.Get<TItem>(key);
                        if (dataItem != null)
                        {
                            data.Add(dataItem);
                        }
                    }
                }
                _logManager.Log(MsgLevel.INFO, ModuleName, $"{sFuncName} - Found
{data.Count} elements of '{typeof(TItem).Name}'.", string.Empty, string.Empty,
partialKey);
            }
            catch (Exception ex)
            {
                _logManager.Log(ModuleName, sFuncName, ex, string.Empty, string.Empty,
partialKey);
            }
            finally
            {
                _lock.ExitReadLock();
            }

            return data;
        }

        public void RemoveItem(string key)
        {
            var sFuncName = nameof(RemoveItem);

            _lock.EnterWriteLock();
            try
            {
                _cache.Remove(key);
            }
            catch (Exception ex)
            {
                _logManager.Log(ModuleName, sFuncName, ex, string.Empty, string.Empty,
key);
            }
            finally
            {
                _lock.ExitWriteLock();
            }
        }

        public void UpdateConfigEventSubscriber()
        {
            var sFuncName = nameof(UpdateConfigEventSubscriber);
            var reFalg = false;

            if (_config.GetData(nameof(_cachingOptions.CacheMaxSize), out int
cacheMaxSize) && _cachingOptions.CacheMaxSize.Value != cacheMaxSize)
            {
                _logManager.Log(MsgLevel.INFO, ModuleName, $"{sFuncName} - Cache
MaxSize was changed from {_cachingOptions.CacheMaxSize.Value} to {cacheMaxSize}.",
string.Empty, string.Empty, string.Empty, Consts.SysCorrelation);
                _cachingOptions.CacheMaxSize.Value = cacheMaxSize;
                reFalg = true;
            }
            if (_config.GetData(nameof(_cachingOptions.CacheSlideExpirationHours), out
int cacheSlideExpirationHours) &&
```

```
_cachingOptions.CacheSlideExpirationHours.Value.TotalMinutes !=
cacheSlideExpirationHours)
            {
                _logManager.Log(MsgLevel.INFO, ModuleName, $"{sFuncName} - Cache
SlideExpirationHours was changed from
{_cachingOptions.CacheSlideExpirationHours.Value.TotalHours} to
{cacheSlideExpirationHours}.", string.Empty, string.Empty, string.Empty,
Consts.SysCorrelation);
                _cachingOptions.CacheSlideExpirationHours.Value =
TimeSpan.FromMinutes(cacheSlideExpirationHours);
            }
            if (_config.GetData(nameof(_cachingOptions.CacheAbsoluteExpirationHours),
out int cacheAbsoluteExpirationHours) &&
_cachingOptions.CacheAbsoluteExpirationHours.Value.TotalHours !=
cacheAbsoluteExpirationHours)
            {
                _logManager.Log(MsgLevel.INFO, ModuleName, $"{sFuncName} - Cache
AbsoluteExpirationHours was changed from
{_cachingOptions.CacheAbsoluteExpirationHours.Value.TotalHours} to
{cacheAbsoluteExpirationHours}.", string.Empty, string.Empty, string.Empty,
Consts.SysCorrelation);
                _cachingOptions.CacheAbsoluteExpirationHours.Value =
TimeSpan.FromMinutes(cacheAbsoluteExpirationHours);
            }

            if (reFalg)
            {
                RecreateCache();
            }
        }

        private void RecreateCache()
        {
            var sFuncName = nameof(RecreateCache);
            var currValues = new Dictionary<string, object>();
            _logManager.Log(MsgLevel.INFO, ModuleName, $"{sFuncName} - Cache re-
creation requested due to config change.", string.Empty, string.Empty, string.Empty,
Consts.SysCorrelation);

            _lock.EnterUpgradeableReadLock();
            try
            {
                var keys = _cache.Keys.ToList();
                if (keys.Count != 0)
                {
                    foreach (var key in keys)
                    {
                        var dataItem = _cache.Get<object>(key);
                        if (dataItem != null)
                        {
                            currValues.Add(key.ToString(), dataItem);
                        }
                    }

                    var cacheOptions = new MemoryCacheEntryOptions()
                        .SetSize(CacheEntrySize)

.SetSlidingExpiration(_cachingOptions.CacheSlideExpirationHours.Value)

.SetAbsoluteExpiration(_cachingOptions.CacheAbsoluteExpirationHours.Value)
                        .RegisterPostEvictionCallback(EvictionDelegate);

                    _lock.EnterWriteLock();
                    _cache = new MemoryCache(new MemoryCacheOptions()
```

```
                    {
                        SizeLimit = _cachingOptions.CacheMaxSize.Value
                    });
                    foreach (var kv in currValues)
                    {
                        _cache.Set(kv.Key, kv.Value, cacheOptions);
                    }
                    _lock.ExitWriteLock();
                    _logManager.Log(MsgLevel.INFO, ModuleName, $"{sFuncName} - Cache
was re-created successfully with {currValues.Count} entities kept.", string.Empty,
string.Empty, string.Empty, Consts.SysCorrelation);
                }
                else
                {
                    _lock.EnterWriteLock();
                    _cache = new MemoryCache(new MemoryCacheOptions()
                    {
                        SizeLimit = _cachingOptions.CacheMaxSize.Value
                    });
                    _lock.ExitWriteLock();
                    _logManager.Log(MsgLevel.INFO, ModuleName, $"{sFuncName} - Cache
was re-created successfully with no entities kept.", string.Empty, string.Empty,
string.Empty, Consts.SysCorrelation);
                }
            }
            catch (Exception ex)
            {
                _logManager.Log(ModuleName, sFuncName, ex, string.Empty, string.Empty,
string.Empty, Consts.SysCorrelation);
            }
            finally
            {
                _lock.ExitUpgradeableReadLock();
            }
        }

        private void EvictionDelegate(object key, object? value, EvictionReason
reason, object? state)
        {
            if (value == null || reason == EvictionReason.None)
            {
                return;
            }

            EvictionCallback?.Invoke(key, value, reason);
        }
    }
}

using Common;
using Common.Helpers;
using Common.Interfaces;

namespace TestingBot
{
    public class LogManager : ILogManager
    {
        private const string ModuleName = nameof(LogManager);
        private const string _directory = "logs";
        private const MsgLevel _defaultLevel = MsgLevel.INFO;

        private readonly ThreadSafeProp<MsgLevel> _currentLevel;

        private readonly object _lock = new();
```

```
        private static readonly LogManager _instance = new();

        static LogManager() { }
        private LogManager()
        {
            if (!Directory.Exists(_directory))
            {
                Directory.CreateDirectory(_directory);
            }

            _currentLevel = new()
            {
                Value = _defaultLevel
            };
        }

        public static LogManager Instance
        {
            get
            {
                return _instance;
            }
        }

        public void Log(MsgLevel logLevel, string moduleName, string message, params
string[] correlationData)
        {
            if (logLevel < _currentLevel.Value)
            {
                return;
            }

            if (correlationData == null)
            {
                correlationData = [];
            }

            string correlationData1 = correlationData.Length > 0 ? correlationData[0]
: string.Empty; //UID
            string correlationData2 = correlationData.Length > 1 ? correlationData[1]
: string.Empty; //SID
            string correlationData3 = correlationData.Length > 2 ? correlationData[2]
: string.Empty; //Cache
            string correlationData4 = correlationData.Length > 3 ? correlationData[3]
: string.Empty; //System

            string logMessage = string.Empty;
            if (string.IsNullOrEmpty(moduleName))
            {
                logMessage = $"{DateTime.Now:G} {logLevel} [{correlationData1}]
[{correlationData2}] [{correlationData3}] [{correlationData4}] {message}\n";
            }
            else
            {
                logMessage = $"{DateTime.Now:G} {logLevel} {moduleName}
[{correlationData1}] [{correlationData2}] [{correlationData3}] [{correlationData4}]
{message}\n";
            }

            try
            {
                lock (_lock)
                {
```

```
                    File.AppendAllText($"{_directory}\\{GetFileName()}", logMessage);
                }
            }
            finally { }
        }

        public void Log(MsgLevel logLevel, string message)
        {
            Log(logLevel, string.Empty, message);
        }

        public void Log(string moduleName, string funcName, Exception exception,
params string[] correlationData)
        {
            var errorMessage = $"{funcName} - Exception: {exception.Message}\nStack
trace: {exception.StackTrace}\n" +
                $"Inner exception: {exception.InnerException?.Message}\nInner
exception stack trace: {exception.InnerException?.StackTrace}";
            Log(MsgLevel.ERROR, moduleName, errorMessage, correlationData);
        }

        private string GetFileName()
        {
            var date = DateTime.Now;
            return $"log-{date.Year}-{date.Month:00}-{date.Day:00}.log";
        }

        public void UpdateConfigEventSubscriber()
        {
            var sFuncName = nameof(UpdateConfigEventSubscriber);

            if (ConfigMgr.Instance.GetData("LogLevel", out string level) &&
Enum.TryParse<MsgLevel>(level, true, out var logLevel))
            {
                if (_currentLevel.Value != logLevel)
                {
                    Log(MsgLevel.INFO, ModuleName, $"{sFuncName} - Log level was
updated from {_currentLevel.Value} to {logLevel}", string.Empty, string.Empty,
string.Empty, Consts.SysCorrelation);
                    _currentLevel.Value = logLevel;
                }
            }
            else
            {
                _currentLevel.Value = _defaultLevel;
            }
        }
    }
}

using Common.Helpers;
using Common.Interfaces;
using Common.Models;
using Common;
using Telegram.Bot.Types;
using Telegram.Bot;
using Microsoft.Extensions.Caching.Memory;

namespace TestingBot
{
    public class SessionManager : ISessionManager
    {
        private const string ModuleName = nameof(SessionManager);
```

```
        private readonly ITelegramBotClient _bot;
        private readonly CancellationToken _botToken;
        private readonly ILogManager _logManager;
        private readonly IInMemoryCache _cache;

        public SessionManager(ITelegramBotClient bot, CancellationToken botToken,
ILogManager logManager, IInMemoryCache cache)
        {
            _bot = bot;
            _botToken = botToken;
            _logManager = logManager;
            _cache = cache;
            _cache.EvictionCallback += SessionEvictionFromCache;
        }

        public bool EstablishSession(ITelegramBotClient bot, Message message,
CancellationToken token, out ISession session)
        {
            return InternalEstablishSession(message.From, bot, token, out session);
        }

        public bool EstablishSession(ITelegramBotClient bot, CallbackQuery callback,
CancellationToken token, out ISession session)
        {
            return InternalEstablishSession(callback.From, bot, token, out session);
        }

        public bool EstablishSession(ITelegramBotClient bot, Chat chat, string
langCode, CancellationToken token, out ISession session)
        {
            var user = new User()
            {
                Id = chat.Id,
                LanguageCode = langCode,
                FirstName = chat.FirstName,
                Username = chat.Username
            };
            return InternalEstablishSession(user, bot, token, out session);
        }

        private bool InternalEstablishSession(User user, ITelegramBotClient bot,
CancellationToken token, out ISession session)
        {
            var sFuncName = nameof(EstablishSession);
            session = GetOrCreateSession(user.Id, out var source);

            if (session == null)
            {
                _logManager.Log(MsgLevel.WARNING, ModuleName, $"{sFuncName} - Failed
to get or add user session to cache.", user.Id.ToString());
                Tg.SendFailureMessageToClient(user.Id, user.LanguageCode,
"FailureMessage", bot, token).Wait();
                return false;
            }

            var differ = session.UserInfo.CompareAndUpdateUserData(user);
            if (differ)
            {
                //Update user info in DB
            }
            return true;
        }
```

```
        public ISession GetOrCreateSession(long userId, out EntrySource source,
Func<ISession> getInstance = null)
        {
            var key = Key.GetCacheKey<ISession>(userId.ToString());
            var func = getInstance == null ? CreateSession : getInstance;
            source = _cache.GetOrAdd(key, func, out ISession session);
            return session;
        }

        public bool IsSessionExists(long userId)
        {
            var key = Key.GetCacheKey<ISession>(userId.ToString());
            var source = _cache.GetOrAdd(key, () => null, out ISession session);
            return source switch
            {
                EntrySource.NotFound => false,
                EntrySource.Failed => false,
                _ => true
            };
        }

        public ISession CreateSession()
        {
            //Retrieve user from DB
            return null;
        }

        public ISession CreateSession(EduUser user)
        {
            return new Session(user);
        }

        private void SessionEvictionFromCache(object key, object value, EvictionReason
reason)
        {
            var sFuncName = nameof(SessionEvictionFromCache);

            if (!key.ToString().Contains(Consts.SessionCachePrefix))
            {
                _logManager.Log(MsgLevel.TRACE, ModuleName, $"{sFuncName} - Removed
item from cache is not session.", string.Empty, string.Empty, key.ToString());
                return;
            }

            var res = _cache.GetOrAdd<bool>("ShutdownOptions", () => false, out var
isShutDown);
            if (value is ISession session && session.SessionData.MenuMsg !=
Consts.NoMessage && isShutDown)
            {
                _logManager.Log(MsgLevel.TRACE, ModuleName, $"{sFuncName} - Cleaning
up session from cache by eviction.", string.Empty, string.Empty, key.ToString());
                _bot.DeleteMessage(session.UserInfo.Id, session.SessionData.MenuMsg,
_botToken).Wait();
                var msg = LocalizationManager.GetMessage("RenewSession",
session.UserInfo.LanguageCode);
                _bot.SendMessage(session.UserInfo.Id, msg, cancellationToken:
_botToken, disableNotification: true).Wait();
            }
        }
    }
}

using Common.Interfaces;
using Telegram.Bot.Types;
```

```
using Telegram.Bot;
using Common;

namespace TestingBot.Providers
{
    public class BotHost : IBotHost
    {
        private const string ModuleName = nameof(BotHost);

        private CancellationTokenSource _cancellationTokenSource;
        private readonly ITelegramBotClient _bot;
        private readonly TimeSpan _delay = TimeSpan.FromSeconds(5);

        protected readonly ILogManager _logManager;

        public event Func<ITelegramBotClient, Update, CancellationToken, Task>
UpdateHandler;
        public event Func<ITelegramBotClient, Exception, CancellationToken, Task>
ExceptionHandler;
        public event Func<ITelegramBotClient, CancellationToken, Task> OnStart;
        public event Func<ITelegramBotClient, CancellationToken, Task> OnStop;

        public ITelegramBotClient BotClient
        {
            get { return _bot; }
        }

        public CancellationToken BotToken
        {
            get
            {
                return _cancellationTokenSource?.Token ?? CancellationToken.None;
            }
        }

        public BotHost(string botToken, ILogManager logManager)
        {
            _bot = new TelegramBotClient(botToken);
            _logManager = logManager;
        }

        public void StartBot()
        {
            var sFuncName = nameof(StartBot);

            _cancellationTokenSource = new CancellationTokenSource();
            var token = _cancellationTokenSource.Token;
            _bot.StartReceiving(BotUpdateHandler, BotErrorHandler, cancellationToken:
token);
            _logManager.Log(MsgLevel.INFO, ModuleName, $"{sFuncName} - Started bot.",
string.Empty, string.Empty, string.Empty, Consts.SysCorrelation);
            Task.Delay(_delay, token).Wait();
            OnStart?.Invoke(BotClient, BotToken);
        }

        public void StopBot()
        {
            var sFuncName = nameof(StopBot);

            OnStop?.Invoke(BotClient, BotToken);
            var token = _cancellationTokenSource.Token;
            Task.Delay(_delay, token).Wait();
            _cancellationTokenSource?.Cancel();
            _cancellationTokenSource?.Dispose();
```

```
            _cancellationTokenSource = null;
            _logManager.Log(MsgLevel.INFO, ModuleName, $"{sFuncName} - Stopped bot.",
string.Empty, string.Empty, string.Empty, Consts.SysCorrelation);
        }

        public void RestartBot()
        {
            if (_cancellationTokenSource != null)
            {
                StopBot();
            }
            StartBot();
        }

        private async Task BotUpdateHandler(ITelegramBotClient client, Update update,
CancellationToken token)
        {
            var sFuncName = nameof(BotUpdateHandler);

            try
            {
                var task = new Task(() =>
                {
                    UpdateHandler?.Invoke(client, update, token);
                }, token);
                _ = Task.Factory.StartNew(task.RunSynchronously, token);
            }
            catch (Exception ex)
            {
                _logManager.Log(ModuleName, sFuncName, ex, string.Empty, string.Empty,
string.Empty, Consts.SysCorrelation);
            }
        }

        private async Task BotErrorHandler(ITelegramBotClient client, Exception
exception, CancellationToken token)
        {
            var sFuncName = nameof(BotErrorHandler);

            try
            {
                var task = new Task(() =>
                {
                    ExceptionHandler?.Invoke(client, exception, token);
                }, token);
                _ = Task.Factory.StartNew(task.RunSynchronously, token);
            }
            catch (Exception ex)
            {
                _logManager.Log(ModuleName, sFuncName, ex, string.Empty, string.Empty,
string.Empty, Consts.SysCorrelation);
            }
        }
    }
}

using Common;
using Common.Interfaces;
using Common.Models;
using Newtonsoft.Json;
using Telegram.Bot;
using Telegram.Bot.Types;
using Telegram.Bot.Types.Enums;
```

```
namespace TestingBot.Handlers
{
    public class BotHandlers : IBotHandlers
    {
        private const string ModuleName = nameof(BotHandlers);
        private const string _recentSessionsFile = "RecentSessions.json";

        private readonly KeepAliveOptions _keepAliveOptions;
        private readonly IMessageHandler _messageHandler;
        private readonly ICallbackHandler _callbackHandler;
        private readonly IInMemoryCache _cache;
        private readonly IConfigManager _config;
        private readonly ILogManager _logManager;

        public BotHandlers(IMessageHandler messageHandler, ICallbackHandler
callbackHandler, IInMemoryCache cache, IConfigManager config, ILogManager logManager)
        {
            _messageHandler = messageHandler;
            _callbackHandler = callbackHandler;
            _cache = cache;
            _config = config;
            _logManager = logManager;
            _keepAliveOptions = new();
            UpdateConfigEventSubscriber();
            _config.OnConfigChanged += UpdateConfigEventSubscriber;
        }

        public async Task UpdateHandler(ITelegramBotClient bot, Update update,
CancellationToken cancellationToken)
        {
            var sFuncName = nameof(UpdateHandler);

            if (update.Type == UpdateType.Message)
            {
                var message = update.Message;

                if (message.Chat.Id < 0)
                {
                    _logManager.Log(MsgLevel.TRACE, ModuleName, $"{sFuncName} - Group
chat attempt to use bot.", message.Chat.Id.ToString());
                    return;
                }
                if (message.From.IsBot)
                {
                    _logManager.Log(MsgLevel.TRACE, ModuleName, $"{sFuncName} -
Another bot attempt to use bot.", message.Chat.Id.ToString());
                    return;
                }

                try
                {
                    _logManager.Log(MsgLevel.TRACE, ModuleName, $"{sFuncName} -
Received new message.", message.From.Id.ToString());
                    await _messageHandler.HandleMessage(bot, message,
cancellationToken);
                }
                catch (Exception ex)
                {
                    _logManager.Log(ModuleName, sFuncName, ex,
message.From.Id.ToString());
                }
            }
            else if (update.Type == UpdateType.CallbackQuery)
            {
```

```
                var callback = update.CallbackQuery;

                if (!callback.Data.Contains(Consts.CallbackPrefix))
                {
                    _logManager.Log(MsgLevel.TRACE, ModuleName, $"{sFuncName} -
Invalid callback.", callback.From.Id.ToString());
                    return;
                }

                try
                {
                    _logManager.Log(MsgLevel.TRACE, ModuleName, $"{sFuncName} -
Received new callback.", callback.From.Id.ToString());
                    await _callbackHandler.HandleCallback(bot, callback,
cancellationToken);
                }
                catch (Exception ex)
                {
                    _logManager.Log(ModuleName, sFuncName, ex,
callback.From.Id.ToString());
                }
            }
        }

        public async Task ErrorHandler(ITelegramBotClient bot, Exception exception,
CancellationToken cancellationToken)
        {
            var sFuncName = nameof(ErrorHandler);

            _logManager.Log(ModuleName, sFuncName, exception, string.Empty,
string.Empty, string.Empty, Consts.SysCorrelation);
        }

        public async Task OnBotStart(ITelegramBotClient bot, CancellationToken token)
        {
            var sFuncName = nameof(OnBotStart);

            try
            {
                await
bot.SendMessage(_keepAliveOptions.KeepAliveCommunicationId.Value, "Bot started.",
cancellationToken: token);

                if (System.IO.File.Exists(_recentSessionsFile))
                {
                    _logManager.Log(MsgLevel.INFO, ModuleName, $"{sFuncName} - Recent
sessions found. Start notifying users.", string.Empty, string.Empty, string.Empty,
Consts.SysCorrelation);
                    var text = System.IO.File.ReadAllText(_recentSessionsFile);
                    var recentSessions =
JsonConvert.DeserializeObject<RecentSessions>(text);
                    if (recentSessions != null && recentSessions.RecentActiveUsers !=
null)
                    {
                        foreach (var session in recentSessions.RecentActiveUsers)
                        {
                            _logManager.Log(MsgLevel.TRACE, ModuleName, $"{sFuncName}
- Notifying user {session.UserInfo.Name}.", session.UserInfo.Id.ToString(),
string.Empty, string.Empty, Consts.SysCorrelation);
                            var localizedMessage =
LocalizationManager.GetMessage("StartupNotification", session.UserInfo.LanguageCode);
                            await bot.SendMessage(session.UserInfo.Id,
localizedMessage, cancellationToken: token);
                        }
```

```
                    }
                    System.IO.File.Delete(_recentSessionsFile);
                }
                else
                {
                    _logManager.Log(MsgLevel.INFO, ModuleName, $"{sFuncName} - No
recent sessions found.", string.Empty, string.Empty, string.Empty,
Consts.SysCorrelation);
                }
            }
            catch (Exception ex)
            {
                _logManager.Log(ModuleName, sFuncName, ex, string.Empty, string.Empty,
string.Empty, Consts.SysCorrelation);
            }
        }

        public async Task OnBotStop(ITelegramBotClient bot, CancellationToken token)
        {
            var sFuncName = nameof(OnBotStop);

            try
            {
                await
bot.SendMessage(_keepAliveOptions.KeepAliveCommunicationId.Value, "Bot stopped.",
cancellationToken: token);

                var recentSessions =
_cache.GetAll<ISession>(Consts.SessionCachePrefix);
                if (recentSessions != null && recentSessions.Count != 0)
                {
                    var saveRecentSessions = new RecentSessions();
                    foreach (var session in recentSessions)
                    {
                        saveRecentSessions.RecentActiveUsers.Add(session);
                        var localizedMessage =
LocalizationManager.GetMessage("ShutdownNotification", session.UserInfo.LanguageCode);
                        await bot.SendMessage(session.UserInfo.Id, localizedMessage,
cancellationToken: token);
                        if (session.SessionData.MenuMsg != Consts.NoMessage)
                        {
                            await bot.DeleteMessage(session.UserInfo.Id,
session.SessionData.MenuMsg, cancellationToken: token);
                        }
                    }
                    System.IO.File.WriteAllText(_recentSessionsFile,
JsonConvert.SerializeObject(saveRecentSessions, Formatting.Indented));
                }
            }
            catch (Exception ex)
            {
                _logManager.Log(ModuleName, sFuncName, ex, string.Empty, string.Empty,
string.Empty, Consts.SysCorrelation);
            }
        }

        public void UpdateConfigEventSubscriber()
        {
            var sFuncName = nameof(UpdateConfigEventSubscriber);

            if (_config.GetData(Consts.ShutdownOptions, out bool isShutdownRequested)
&& isShutdownRequested)
            {
```

```
                _logManager.Log(MsgLevel.INFO, ModuleName, $"{sFuncName} - Bot
shutdown requested.", string.Empty, string.Empty, string.Empty,
Consts.SysCorrelation);
                _keepAliveOptions.Cts?.Cancel();
                return;
            }
            if (_config.GetData(nameof(_keepAliveOptions.KeepAliveCommunicationId),
out long keepAliveCommunicationId) && _keepAliveOptions.KeepAliveCommunicationId.Value
!= keepAliveCommunicationId)
            {
                _logManager.Log(MsgLevel.INFO, ModuleName, $"{sFuncName} - Keep alive
CommunicationId was changed from {_keepAliveOptions.KeepAliveCommunicationId.Value} to
{keepAliveCommunicationId}.", string.Empty, string.Empty, string.Empty,
Consts.SysCorrelation);
                _keepAliveOptions.KeepAliveCommunicationId.Value =
keepAliveCommunicationId;
            }
            if (_config.GetData(nameof(_keepAliveOptions.KeepAliveMessagesStore), out
int keepAliveMessagesStore) && _keepAliveOptions.KeepAliveMessagesStore.Value !=
keepAliveMessagesStore)
            {
                _logManager.Log(MsgLevel.INFO, ModuleName, $"{sFuncName} - Keep alive
MessagesStore was changed from {_keepAliveOptions.KeepAliveMessagesStore.Value} to
{keepAliveMessagesStore}.", string.Empty, string.Empty, string.Empty,
Consts.SysCorrelation);
                _keepAliveOptions.KeepAliveMessagesStore.Value =
keepAliveMessagesStore;
            }
        }

        public async Task KeepAlive(ITelegramBotClient bot, CancellationToken
botToken)
        {
            ArgumentNullException.ThrowIfNull(bot);

            var sFuncName = nameof(KeepAlive);

            var token = _keepAliveOptions.Cts.Token;
            long incrementor = 0;
            long communicationId = 0;
            while (!token.IsCancellationRequested)
            {
                try
                {
                    await Task.Delay(TimeSpan.FromHours(1), token);
                    communicationId =
_keepAliveOptions.KeepAliveCommunicationId.Value;
                    if (!token.IsCancellationRequested)
                    {
                        var message = await bot.SendMessage(communicationId,
string.Format(Consts.KeepAliveMessage, incrementor), cancellationToken: botToken);
                        _keepAliveOptions.KeepAliveMessages.Add(message);
                        await DoMessagesCleanUp(bot, botToken);
                        incrementor++;
                        _logManager.Log(MsgLevel.INFO, ModuleName, $"{sFuncName} -
Keep alive message sent.", communicationId.ToString(), string.Empty, string.Empty,
Consts.SysCorrelation);
                    }
                    else
                    {
                        _logManager.Log(MsgLevel.WARNING, ModuleName, $"{sFuncName} -
Keep alive message hasn't been sent.", communicationId.ToString(), string.Empty,
string.Empty, Consts.SysCorrelation);
                    }
```

```
                }
                catch (OperationCanceledException)
                {
                    _logManager.Log(MsgLevel.WARNING, ModuleName, $"{sFuncName} - Keep
alive operation cancelled.", communicationId.ToString(), string.Empty, string.Empty,
Consts.SysCorrelation);
                }
                catch (Exception e)
                {
                    _logManager.Log(ModuleName, sFuncName, e,
communicationId.ToString(), string.Empty, string.Empty, Consts.SysCorrelation);
                }
            }
            _logManager.Log(MsgLevel.INFO, ModuleName, $"{sFuncName} - Shutdown bot.",
string.Empty, string.Empty, string.Empty, Consts.SysCorrelation);
        }

        private async Task DoMessagesCleanUp(ITelegramBotClient bot, CancellationToken
token)
        {
            var sFuncName = nameof(DoMessagesCleanUp);

            var storeSize = _keepAliveOptions.KeepAliveMessagesStore.Value;
            if (_keepAliveOptions.KeepAliveMessages.Count >= storeSize * 2)
            {
                for (int i = 0; i < storeSize; i++)
                {
                    await
bot.DeleteMessage(_keepAliveOptions.KeepAliveMessages[i].Chat.Id,
_keepAliveOptions.KeepAliveMessages[i].MessageId, cancellationToken: token);
                }
                _keepAliveOptions.KeepAliveMessages.RemoveRange(0, storeSize);
                _logManager.Log(MsgLevel.DEBUG, ModuleName, $"{sFuncName} -
{storeSize} keep alive messages cleaned.", string.Empty, string.Empty, string.Empty,
Consts.SysCorrelation);
            }
        }
    }
}

using Common;
using Common.Helpers;
using Common.Interfaces;
using Telegram.Bot;
using Telegram.Bot.Types;

namespace TestingBot.Handlers
{
    public class MessageHandler : IMessageHandler
    {
        private const string ModuleName = nameof(MessageHandler);

        private readonly ISessionManager _session;
        private readonly ICommandHandler _commandHandler;
        private readonly ILogManager _logManager;

        public MessageHandler(ICommandHandler commandHandler, ISessionManager
sessionManager, ILogManager logManager)
        {
            _session = sessionManager;
            _commandHandler = commandHandler;
            _logManager = logManager;
        }
```

```
        public async Task HandleMessage(ITelegramBotClient bot, Message message,
CancellationToken token)
        {
            var sFuncName = nameof(HandleMessage);

            if (Tg.IsIncomingMessageCommand(message))
            {
                var command = Tg.DetectCommand(message.Text);
                if (command != BotCommands.None)
                {
                    _logManager.Log(MsgLevel.INFO, ModuleName, $"{sFuncName} - Command
received '{command}'.", message.From.Id.ToString());
                    await HandleCommand(command, bot, message, token);
                }
            }
            else
            {
                _logManager.Log(MsgLevel.INFO, ModuleName, $"{sFuncName} - Text
message received '{message.Text}'.", message.From.Id.ToString());
                await HandleTextMessage(bot, message, token);
            }
        }

        private async Task HandleTextMessage(ITelegramBotClient bot, Message message,
CancellationToken token)
        {
            var sFuncName = nameof(HandleTextMessage);

            if (!_session.EstablishSession(bot, message, token, out var session) &&
session.NextUserInput == ExpectedInput.None)
            {
                return;
            }

            switch (session.NextUserInput)
            {
                case ExpectedInput.UserAlias:
                    session.UserInfo.Alias = message.Text;
                    await Tg.RespondWithOk(bot, token, session.UserInfo.Id,
session.UserInfo.LanguageCode);
                    _logManager.Log(MsgLevel.INFO, ModuleName, $"{sFuncName} - User
alias was updated succesfully.", session.UserInfo.Id.ToString(),
session.Id.ToString());
                    await _commandHandler.HandleMenu(bot, message, token);
                    break;
            };

            session.NextUserInput = ExpectedInput.None;
        }

        private async Task HandleCommand(BotCommands command, ITelegramBotClient bot,
Message message, CancellationToken token)
        {
            switch (command)
            {
                case BotCommands.Start:
                    await _commandHandler.HandleStart(bot, message, token);
                    break;
                case BotCommands.Menu:
                    await _commandHandler.HandleMenu(bot, message, token);
                    break;
                case BotCommands.About:
                    await _commandHandler.HandleAbout(bot, message, token);
                    break;
```

```
                case BotCommands.EndTask:
                    await _commandHandler.HandleEndTask(bot, message, token);
                    break;
            }
        }
    }
}

using Common;
using Common.Interfaces;
using Telegram.Bot;
using Telegram.Bot.Types;

namespace TestingBot.Handlers
{
    public class CallbackHandler : ICallbackHandler
    {
        private const string ModuleName = nameof(CallbackHandler);

        private readonly IMenuHandler _menuHandler;
        private readonly ICommandHandler _commandHandler;
        private readonly ISessionManager _session;
        private readonly ILogManager _logManager;

        public CallbackHandler(ICommandHandler commandHandler, ISessionManager
session, IMenuHandler menuHandler, ILogManager logManager)
        {
            _commandHandler = commandHandler;
            _logManager = logManager;
            _session = session;
            _menuHandler = menuHandler;
        }

        public async Task HandleCallback(ITelegramBotClient bot, CallbackQuery
callback, CancellationToken token)
        {
            var parsedData = ParseCallbackData(callback.Data);

            if (parsedData[0] == Consts.MenuOptionsPrefix)
            {
                await _menuHandler.HandleMenuCallback(bot, parsedData[1..], callback,
token);
            }
        }

        private static List<string> ParseCallbackData(string data)
        {
            var cleared = data.Replace(Consts.CallbackPrefix, string.Empty);
            return cleared.Split('#').ToList();
        }
    }
}

using Common;
using Common.Helpers;
using Common.Interfaces;
using Common.Models;
using Telegram.Bot;
using Telegram.Bot.Types;
using Telegram.Bot.Types.ReplyMarkups;

namespace TestingBot.Handlers
{
    public class CommandHandler : ICommandHandler
```

```
    {
        private const string ModuleName = nameof(CommandHandler);

        private readonly ISessionManager _session;
        private readonly ILogManager _logManager;

        public CommandHandler(ISessionManager sessionManager, ILogManager logManager)
        {
            _session = sessionManager;
            _logManager = logManager;
        }

        public async Task HandleStart(ITelegramBotClient bot, Message message,
CancellationToken token)
        {
            var sFuncName = nameof(HandleStart);
            var session = _session.GetOrCreateSession(message.From.Id, out var
source);

            switch (source)
            {
                case EntrySource.Source:
                    break;
                case EntrySource.Cache:
                    return;
                case EntrySource.NotFound:
                    _logManager.Log(MsgLevel.INFO, ModuleName, $"{sFuncName} - New
user message received. Initiate user blind registration.",
message.From.Id.ToString());
                    var user = RegisterUser(message);
                    session = _session.CreateSession(user);
                    _logManager.Log(MsgLevel.INFO, ModuleName, $"{sFuncName} - New
user was registered. Session created.", session.UserInfo.Id.ToString(),
session.Id.ToString());
                    _session.GetOrCreateSession(message.From.Id, out _, () =>
session);
                    var localizedMessage =
LocalizationManager.GetMessage("WelcomeMessage", user.LanguageCode);
                    var localizedAbout = LocalizationManager.GetMessage("AboutBot",
user.LanguageCode);
                    var formattedAbout = string.Format(localizedAbout, Consts.BotTag);
                    var formattedMessage = string.Format(localizedMessage,
Consts.BotTag, formattedAbout);
                    var sentMsg = await bot.SendMessage(session.UserInfo.Id,
formattedMessage, cancellationToken: token);
                    session.SessionData.LastMsg = sentMsg.Id;
                    await Task.Delay(1000, token);
                    break;
                case EntrySource.Failed:
                    _logManager.Log(MsgLevel.WARNING, ModuleName, $"{sFuncName} -
Failed to get or add user session to cache.", message.From.Id.ToString());
                    await Tg.SendFailureMessageToClient(message.From.Id,
message.From.LanguageCode, "FailedRegisterMessage", bot, token);
                    return;
                default:
                    return;
            }

            await HandleMenu(bot, message, token);
        }

        public async Task HandleMenu(ITelegramBotClient bot, Message message,
CancellationToken token)
        {
```

```
            var sFuncName = nameof(HandleMenu);

            if (!_session.EstablishSession(bot, message, token, out var session))
            {
                return;
            }

            await InternalHandleMenu(bot, session, token);
        }

        public async Task HandleMenu(ITelegramBotClient bot, Chat chat, string
langCode, CancellationToken token)
        {
            var sFuncName = nameof(HandleMenu);

            if (!_session.EstablishSession(bot, chat, langCode, token, out var
session))
            {
                return;
            }

            await InternalHandleMenu(bot, session, token);
        }

        public async Task HandleAbout(ITelegramBotClient bot, Message message,
CancellationToken token)
        {
            var sFuncName = nameof(HandleAbout);

            if (!_session.EstablishSession(bot, message, token, out var session))
            {
                return;
            }

            var localizedAbout = LocalizationManager.GetMessage("AboutBot",
session.UserInfo.LanguageCode);
            var formattedAbout = string.Format(localizedAbout, Consts.BotTag);
            var sentMsg = await bot.SendMessage(message.From.Id, formattedAbout,
cancellationToken: token);
            session.SessionData.LastMsg = sentMsg.Id;
        }

        public async Task HandleEndTask(ITelegramBotClient bot, Message message,
CancellationToken token)
        {
            var sFuncName = nameof(HandleEndTask);

            if (!_session.EstablishSession(bot, message, token, out var session))
            {
                return;
            }

            await InternalEndTask(bot, session, token);
            await HandleMenu(bot, message, token);
        }

        public async Task HandleEndTask(ITelegramBotClient bot, Chat chat, string
langCode, CancellationToken token)
        {
            var sFuncName = nameof(HandleEndTask);

            if (!_session.EstablishSession(bot, chat, langCode, token, out var
session))
            {
```

```
                return;
            }

            await InternalEndTask(bot, session, token);
            await HandleMenu(bot, chat, langCode, token);
        }

        private EduUser RegisterUser(Message message)
        {
            Enum.TryParse(message.From.LanguageCode, true, out Locale locale);
            var user = new EduUser()
            {
                Id = message.From.Id,
                Name = message.From.FirstName,
                Tag = message.From.Username,
                Privilege = UserPrivileges.Default,
                LanguageCode = locale,
            };

            //Add user to DB
            return user;
        }

        private async Task InternalHandleMenu(ITelegramBotClient bot, ISession
session, CancellationToken token)
        {
            var localizedMenu = LocalizationManager.GetMessage("MenuText",
session.UserInfo.LanguageCode);
            var menuButtons = Tg.GetMenuReplyMarkup(Consts.MainMenuStartIndex,
session.UserInfo.LanguageCode);
            if (session.SessionData.MenuMsg == Consts.NoMessage)
            {
                await Tg.SendMenu(bot, session, localizedMenu, token, menuButtons);
            }
            else if (session.SessionData.MenuMsg != session.SessionData.LastMsg)
            {
                await Tg.ResendMenu(bot, session, localizedMenu, token, menuButtons);
            }
            else
            {
                await Tg.UpdateMenu(bot, session, localizedMenu, token, menuButtons);
            }

            session.SessionData.MenuOption = Consts.MainMenuStartIndex;
            session.State = UserAction.Menu;
        }

        private async Task InternalEndTask(ITelegramBotClient bot, ISession session,
CancellationToken token)
        {
            switch (session.State)
            {
                case UserAction.AdminPanel:
                    return;
                case UserAction.Menu:
                    session.SessionData.MenuMsg = Consts.NoMessage;
                    break;
                case UserAction.TestPassing:
                    session.SessionData.TestId = Guid.Empty;
                    //_testMgr.EndTestPassing();
                    break;
                case UserAction.TestCreation:
                    //_testMgr.CancelCreation(session.SessionData.TestId);
                    session.SessionData.TestId = Guid.Empty;
```

```
                    break;
                case UserAction.ClassCreation:
                    //_classMgr.CancelCreation(session.SessionData.ClassId);
                    break;
            }
            session.NextUserInput = ExpectedInput.None;

            await bot.DeleteMessage(session.UserInfo.Id, session.SessionData.LastMsg,
cancellationToken: token);
            session.SessionData.LastMsg = session.SessionData.MenuMsg;
        }
    }
}

using Common;
using Common.Helpers;
using Common.Interfaces;
using Telegram.Bot;
using Telegram.Bot.Types;
using Telegram.Bot.Types.ReplyMarkups;

namespace TestingBot.Handlers
{
    public class MenuHandler : IMenuHandler
    {
        private const string ModuleName = nameof(MenuHandler);

        private readonly ISessionManager _session;
        private readonly ICommandHandler _commandHandler;
        private readonly ILogManager _logManager;

        public MenuHandler(ISessionManager session, ICommandHandler commandHandler,
ILogManager logManager)
        {
            _session = session;
            _commandHandler = commandHandler;
            _logManager = logManager;
        }

        public async Task HandleMenuCallback(ITelegramBotClient bot, List<string>
parsedData, CallbackQuery callback, CancellationToken token)
        {
            var sFuncName = nameof(HandleMenuCallback);

            if (!_session.EstablishSession(bot, callback, token, out var session))
            {
                return;
            }

            if (session.SessionData.MenuMsg == Consts.NoMessage)
            {
                session.SessionData.MenuMsg = callback.Message.Id;
                session.SessionData.LastMsg = callback.Message.Id;
            }

            var isBack = false;
            if (!int.TryParse(parsedData[1], out int chosenOption))
            {
                _logManager.Log(MsgLevel.WARNING, ModuleName, $"{sFuncName} - Cannot
parse chosen menu option from the callback. Data: '{parsedData[1]}'",
session.UserInfo.Id.ToString(), session.Id.ToString());
                return;
            }
            if (chosenOption == Consts.BackMenuOptionId)
```

```
            {
                _logManager.Log(MsgLevel.INFO, ModuleName, $"{sFuncName} - Received
'Back' option.", session.UserInfo.Id.ToString(), session.Id.ToString());
                isBack = true;
            }
            else if (chosenOption == Consts.CancelMenuOptionId)
            {
                _logManager.Log(MsgLevel.INFO, ModuleName, $"{sFuncName} - Received
'Cancel' option.", session.UserInfo.Id.ToString(), session.Id.ToString());
                await _commandHandler.HandleEndTask(bot, callback.Message.Chat,
session.UserInfo.LanguageCode.ToString(), token);
                return;
            }
            else if (session.NextUserInput != ExpectedInput.None)
            {
                return;
            }

            if (!int.TryParse(parsedData[0], out int menuProfile))
            {
                _logManager.Log(MsgLevel.WARNING, ModuleName, $"{sFuncName} - Cannot
parse menu profile from the callback. Data: '{parsedData[0]}'",
session.UserInfo.Id.ToString(), session.Id.ToString());
                return;
            }

            var startIndex = GetMenuStartIndex(menuProfile, chosenOption, isBack);
            _logManager.Log(MsgLevel.DEBUG, ModuleName, $"{sFuncName} - {chosenOption}
unique menu option was chosen on menu profile {menuProfile}.",
session.UserInfo.Id.ToString(), session.Id.ToString());

            if (startIndex.DigitsNumber() < Consts.MaxMenuDepth || isBack)
            {
                await Tg.UpdateMenuLayer(startIndex, bot, token, session);
            }
            else
            {
                await UpdateInnerMenuOption(startIndex, menuProfile, chosenOption,
bot, token, session);
            }
        }

        private async Task UpdateInnerMenuOption(int startIndex, int menuProfile, int
chosenOption, ITelegramBotClient bot, CancellationToken token, ISession session)
        {
            var sFuncName = nameof(UpdateInnerMenuOption);

            switch (menuProfile)
            {
                case 10:
                    switch (chosenOption)
                    {
                        case 0:
                            await Tg.UpdateMenuLayer(startIndex, bot, token, session);
                            break;

                        case 1:
                            var msg =
LocalizationManager.GetMessage("CurrentlyUnavailable", session.UserInfo.LanguageCode);
                            _ = await bot.SendMessage(session.UserInfo.Id, msg,
cancellationToken: token);
                            break;

                        default:
```

```
                            _logManager.Log(MsgLevel.WARNING, ModuleName,
$"{sFuncName} - No handler for chosen option '{chosenOption}' for menu profile
'{menuProfile}'", session.UserInfo.Id.ToString(), session.Id.ToString());
                            break;
                    };
                    break;

                case 100:
                    switch (chosenOption)
                    {
                        case 0:
                            await Tg.UpdateMenuLayer(startIndex, bot, token, session);
                            break;

                        default:
                            _logManager.Log(MsgLevel.WARNING, ModuleName,
$"{sFuncName} - No handler for chosen option '{chosenOption}' for menu profile
'{menuProfile}'", session.UserInfo.Id.ToString(), session.Id.ToString());
                            break;
                    };
                    break;

                case 1000:
                    switch (chosenOption)
                    {
                        case 0:
                            session.NextUserInput = ExpectedInput.UserAlias;
                            var msg = GetMenyReplyText(menuProfile + chosenOption,
session.UserInfo.LanguageCode);
                            await Tg.UpdateMenuLayer(Consts.CancelMenuOptionId, bot,
token, session);
                            var sentMsg = await bot.SendMessage(session.UserInfo.Id,
msg, cancellationToken: token);
                            session.SessionData.LastMsg = sentMsg.Id;
                            break;

                        case 1:
                            session.UserInfo.Alias = string.Empty;
                            //Update user in Db
                            var newStartIndex = GetMenuStartIndex(menuProfile,
chosenOption, true);
                            await Tg.UpdateMenuLayer(Consts.MainMenuStartIndex, bot,
token, session);
                            await Tg.RespondWithOk(bot, token, session.UserInfo.Id,
session.UserInfo.LanguageCode);
                            break;

                        default:
                            _logManager.Log(MsgLevel.WARNING, ModuleName,
$"{sFuncName} - No handler for chosen option '{chosenOption}' for menu profile
'{menuProfile}'", session.UserInfo.Id.ToString(), session.Id.ToString());
                            break;
                    };
                    break;
            };
        }

        private string GetMenyReplyText(int menuOption, Locale locale)
        {
            return LocalizationManager.GetMessage($"MenuReplyText{menuOption}",
locale);
        }

        private int GetMenuStartIndex(int menuProfile, int chosenOption, bool isBack)
```

```
        {
            var startIndex = 0;
            if (!isBack)
            {
                if (menuProfile < Consts.MenuOptions)
                {
                    startIndex = chosenOption + menuProfile + 1;
                }
                else
                {
                    startIndex = chosenOption + menuProfile;
                }
                startIndex *= 10;
            }
            else
            {
                startIndex = (menuProfile / Consts.MenuOptions).RoundOff();
            }
            return startIndex;
        }
    }
}

using Telegram.Bot.Types.Enums;
using Telegram.Bot.Types;
using Telegram.Bot.Types.ReplyMarkups;
using Telegram.Bot;
using Common.Interfaces;
using Common.Models;

namespace Common.Helpers
{
    public static class Tg
    {
        public static string GetHtmlUserTag(string text, long id)
        {
            return $"<a href=\"tg://user?id={id}\">{text}</a>";
        }

        public static bool IsIncomingMessageCommand(Message message)
        {
            if (message.Entities != null && message.Entities.Length != 0)
            {
                var entity = message.Entities.FirstOrDefault();
                if (entity != null && entity.Type == MessageEntityType.BotCommand)
                {
                    return true;
                }
            }
            return false;
        }

        public static BotCommands DetectCommand(string message)
        {
            var editedMsg = message.Remove(0, 1).Replace($"@{Consts.BotTag}",
string.Empty);
            Enum.TryParse(editedMsg, true, out BotCommands command);
            return command;
        }

        public static IReplyMarkup GetMenuReplyMarkup(int startIndex, Locale locale,
bool backButton = false, bool cancelButton = false)
        {
            var inlineKeyboardButtons = new List<InlineKeyboardButton>();
```

```
            if (startIndex != Consts.BackMenuOptionId && startIndex !=
Consts.CancelMenuOptionId)
            {
                for (int i = startIndex; i < startIndex + Consts.MenuOptions; i++)
                {
                    var localizationButtonTextKey = $"{Consts.MenuOptionsPrefix}{i}";
                    try
                    {
                        var localizedMenuOption =
LocalizationManager.GetMessage(localizationButtonTextKey, locale);
                        if (string.IsNullOrEmpty(localizedMenuOption))
                        {
                            break;
                        }
                        var menuOption = new InlineKeyboardButton(localizedMenuOption)
                        {
                            CallbackData =
Key.GetCallbackId($"{Consts.MenuOptionsPrefix}#{startIndex}#{i - startIndex}")
                        };
                        inlineKeyboardButtons.Add(menuOption);
                    }
                    catch
                    {
                        break;
                    }
                }
            }

            if (backButton)
            {
                var localizedMenuOption =
LocalizationManager.GetMessage($"MenuOption{Consts.BackMenuOptionId}", locale);
                var menuOption = new InlineKeyboardButton(localizedMenuOption)
                {
                    CallbackData =
Key.GetCallbackId($"{Consts.MenuOptionsPrefix}#{startIndex}#{Consts.BackMenuOptionId}"
)
                };

                inlineKeyboardButtons.Add(menuOption);
            }

            if (cancelButton)
            {
                var localizedMenuOption =
LocalizationManager.GetMessage($"MenuOption{Consts.CancelMenuOptionId}", locale);
                var menuOption = new InlineKeyboardButton(localizedMenuOption)
                {
                    CallbackData =
Key.GetCallbackId($"{Consts.MenuOptionsPrefix}#{startIndex}#{Consts.CancelMenuOptionId
}")
                };

                inlineKeyboardButtons.Add(menuOption);
            }

            var markup = new InlineKeyboardMarkup();

            foreach (var markupOption in inlineKeyboardButtons)
            {
                markup.AddNewRow(markupOption);
            }
            return markup;
```

```
        }

        public static async Task SendFailureMessageToClient(long clientId, string
langCode, string failedMessageKey, ITelegramBotClient bot, CancellationToken token)
        {
            Enum.TryParse(langCode, true, out Locale locale);
            var localizedMessage = LocalizationManager.GetMessage(failedMessageKey,
locale);
            await bot.SendMessage(clientId, localizedMessage, cancellationToken:
token);
        }

        public static async Task SendMenu(ITelegramBotClient bot, ISession session,
string message, CancellationToken token, IReplyMarkup menuButtons)
        {
            var sentMsg = await bot.SendMessage(session.UserInfo.Id, message,
cancellationToken: token, replyMarkup: menuButtons);
            session.SessionData.MenuMsg = sentMsg.Id;
            session.SessionData.LastMsg = sentMsg.Id;
        }

        public static async Task ResendMenu(ITelegramBotClient bot, ISession session,
string message, CancellationToken token, IReplyMarkup menuButtons)
        {
            await bot.DeleteMessage(session.UserInfo.Id, session.SessionData.MenuMsg,
token);
            await SendMenu(bot, session, message, token, menuButtons);
        }

        public static async Task UpdateMenu(ITelegramBotClient bot, ISession session,
string message, CancellationToken token, IReplyMarkup menuButtons, bool isHtml =
false)
        {
            var updatedMarkup = menuButtons as InlineKeyboardMarkup;
            if (!isHtml)
            {
                await bot.EditMessageText(session.UserInfo.Id,
session.SessionData.MenuMsg, message, cancellationToken: token);
            }
            else
            {
                await bot.EditMessageText(session.UserInfo.Id,
session.SessionData.MenuMsg, message, cancellationToken: token, parseMode:
ParseMode.Html);
            }
            await bot.EditMessageReplyMarkup(session.UserInfo.Id,
session.SessionData.MenuMsg, updatedMarkup, cancellationToken: token);
        }

        public static async Task RespondWithOk(ITelegramBotClient bot,
CancellationToken token, long uid, Locale locale)
        {
            var respond = LocalizationManager.GetMessage("SaveData", locale);
            _ = await bot.SendMessage(uid, respond, cancellationToken: token);
        }

        public static async Task UpdateMenuLayer(int startIndex, ITelegramBotClient
bot, CancellationToken token, ISession session)
        {
            var localizedMenu = GetMenuText(startIndex, session.UserInfo);
            var backBtn = false;
            var cancelBtn = false;
            if (startIndex == Consts.CancelMenuOptionId) { cancelBtn = true; }
            else if (startIndex != Consts.MainMenuStartIndex) { backBtn = true; }
```

```
            var newMenuButtons = Tg.GetMenuReplyMarkup(startIndex,
session.UserInfo.LanguageCode, backButton: backBtn, cancelButton: cancelBtn);
            await Tg.UpdateMenu(bot, session, localizedMenu, token, newMenuButtons,
true);
            session.SessionData.MenuOption = startIndex == Consts.CancelMenuOptionId ?
Consts.MainMenuStartIndex : startIndex;
        }

        private static string GetMenuText(int startIndex, EduUser userInfo)
        {
            string localizedUnformatted = string.Empty;
            try
            {
                localizedUnformatted =
LocalizationManager.GetMessage($"MenuText{startIndex}", userInfo.LanguageCode);
            }
            catch
            {
                return LocalizationManager.GetMessage("MenuText",
userInfo.LanguageCode);
            }

            return FormatMenuText(localizedUnformatted, startIndex, userInfo);
        }

        private static string FormatMenuText(string unformattedText, int startIndex,
EduUser userInfo)
        {
            string formattedMenu = unformattedText;
            switch (startIndex)
            {
                case 10:
                    var userTag = Tg.GetHtmlUserTag(userInfo.Name, userInfo.Id);
                    var alias = string.IsNullOrEmpty(userInfo.Alias) ?
LocalizationManager.GetMessage("NoAlias", userInfo.LanguageCode) : userInfo.Alias;
                    var lang =
LocalizationManager.MapLanguageCode(userInfo.LanguageCode);
                    formattedMenu = string.Format(unformattedText, userTag, alias,
lang, userInfo.EverPassedTests.Count, userInfo.JoinedClassrooms.Count);
                    break;
            };

            return formattedMenu;
        }
    }
}

using System.Globalization;
using System.Resources;

namespace Common
{
    public static class LocalizationManager
    {
        private static readonly ResourceManager ResourceManager =
            new ResourceManager("Common.Locales.Messages",
typeof(LocalizationManager).Assembly);

        public static string GetMessage(string key, Locale locale)
        {
            var culture = new CultureInfo(locale.ToString());
            return ResourceManager.GetString(key, culture) ?? string.Empty;
        }
```

```
        public static string MapLanguageCode(Locale locale)
        {
            var culture = new CultureInfo(locale.ToString());
            return culture.NativeName;
        }
    }
}

using Common.Interfaces;

namespace Common.Helpers
{
    public static class Key
    {
        public static string GetCacheKey<TType>(string id)
        {
            if (typeof(TType).Name == typeof(ISession).Name)
            {
                return $"{Consts.SessionCachePrefix}{id}";
            }
            return $"{Consts.ObjectCachePrefix}{id}";
        }

        public static string GetCallbackId(string data)
        {
            return $"{Consts.CallbackPrefix}{data}";
        }
    }
}

namespace Common.Helpers
{
    public class ThreadSafeProp<TType>
    {
        private readonly ReaderWriterLockSlim _lock = new();
        private TType _value;
        public TType Value
        {
            get
            {
                _lock.EnterReadLock();
                var safeValue = _value;
                _lock.ExitReadLock();
                return safeValue;
            }
            set
            {
                _lock.EnterWriteLock();
                _value = value;
                _lock.ExitWriteLock();
            }
        }
    }
}
```

Додаток В (обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ МОНІТОРИНГУ ПРОЦЕСУ НАВЧАННЯ

Виконав: студент 2-го курсу групи 2КН-23м
спеціальності 122 «Комп'ютерні науки»
(шифр і назва напряму підготовки, спеціальності)
Вишневський А.В.
(прізвище та ініціали)

Керівник: д.т.н., проф. каф. КН
Іванчук Я.В.
(прізвище та ініціали)
«05» 12 2024 р.

Вінниця ВНТУ - 2024 рік

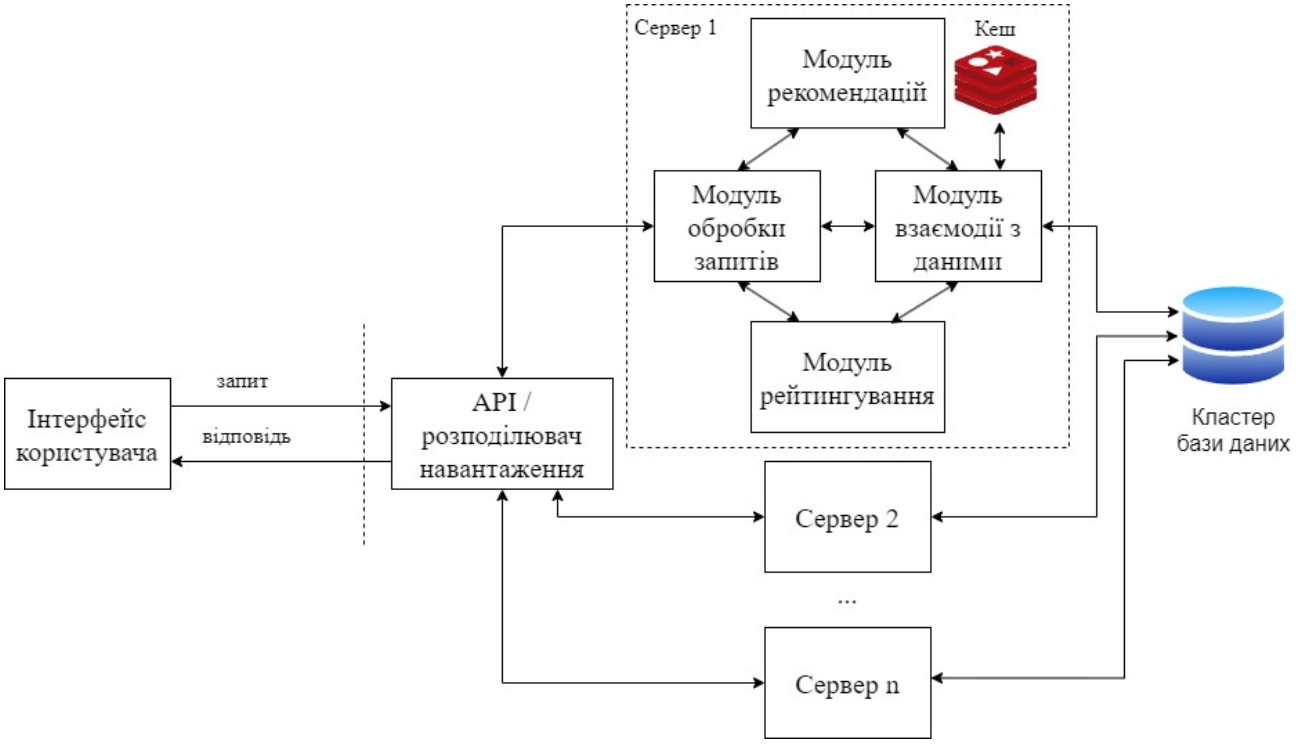

Рисунок В.1 – Схема функціонування інформаційної системи моніторингу прогресу навчання

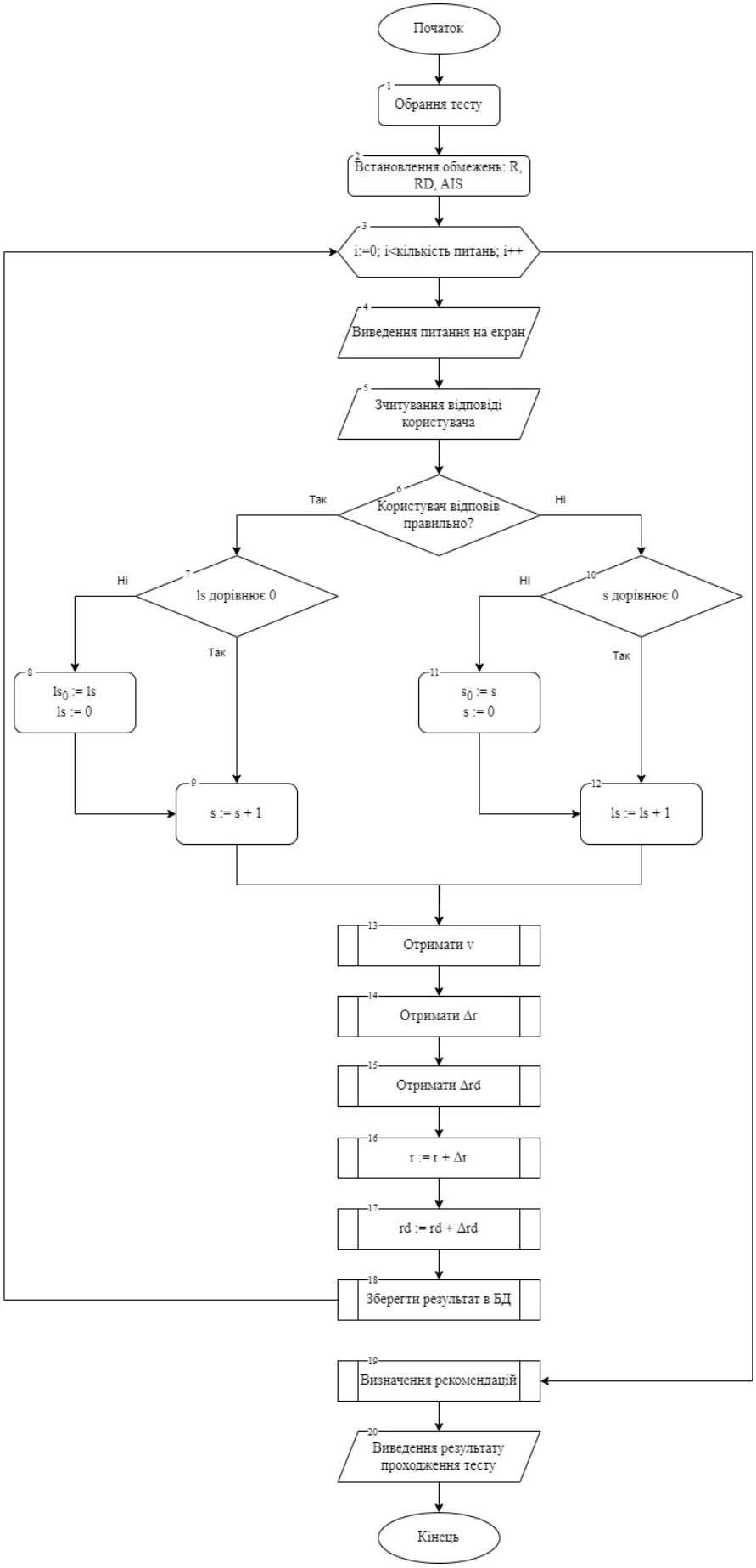

Рисунок В.2 – Схема алгоритму рейтингування при проходженні тесту

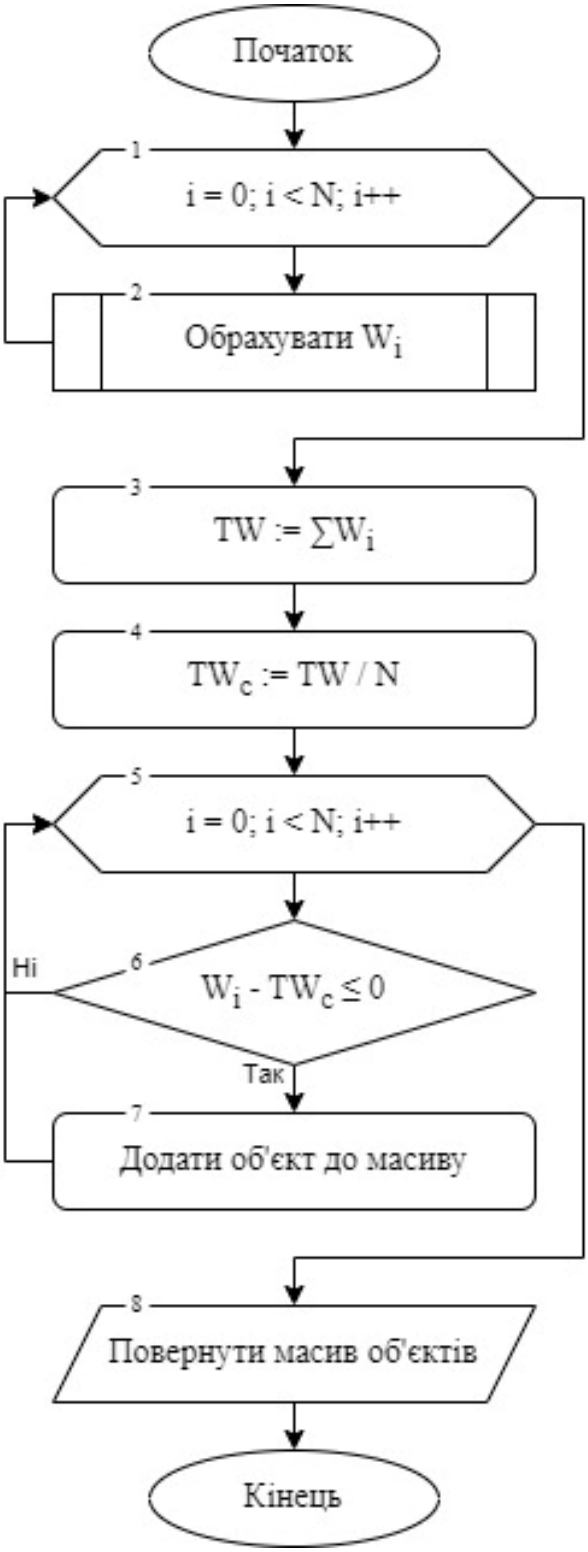

Рисунок В.3 – Схема алгоритму формування рекомендацій

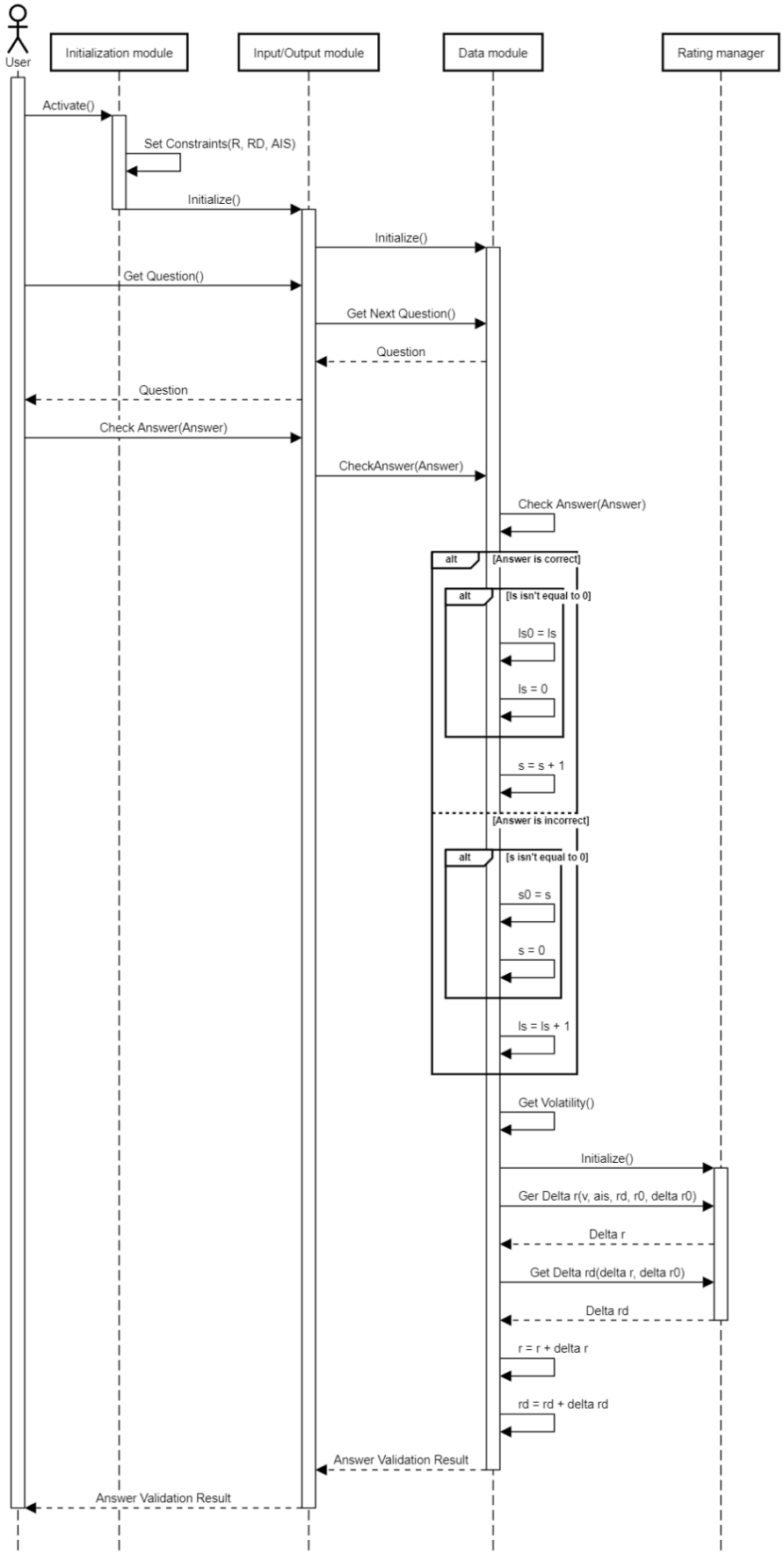

Рисунок В.4 – UML-діаграма послідовностей алгоритму рейтингування для тестів

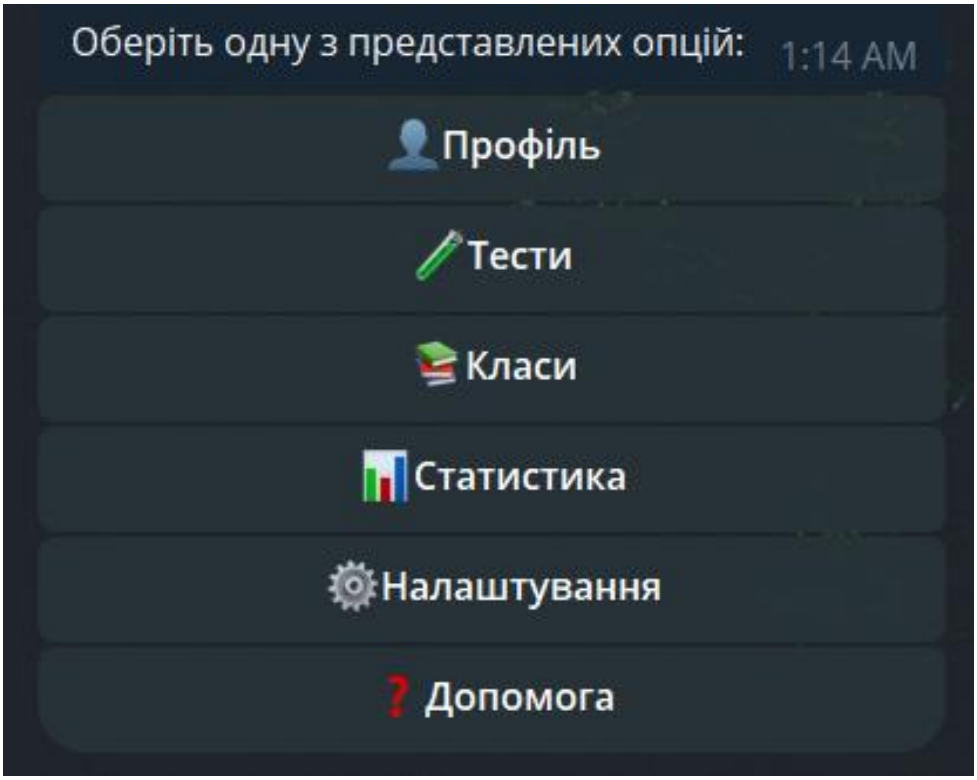

Рисунок В.5 – Головне меню телеграм бота для взаємодії з системою

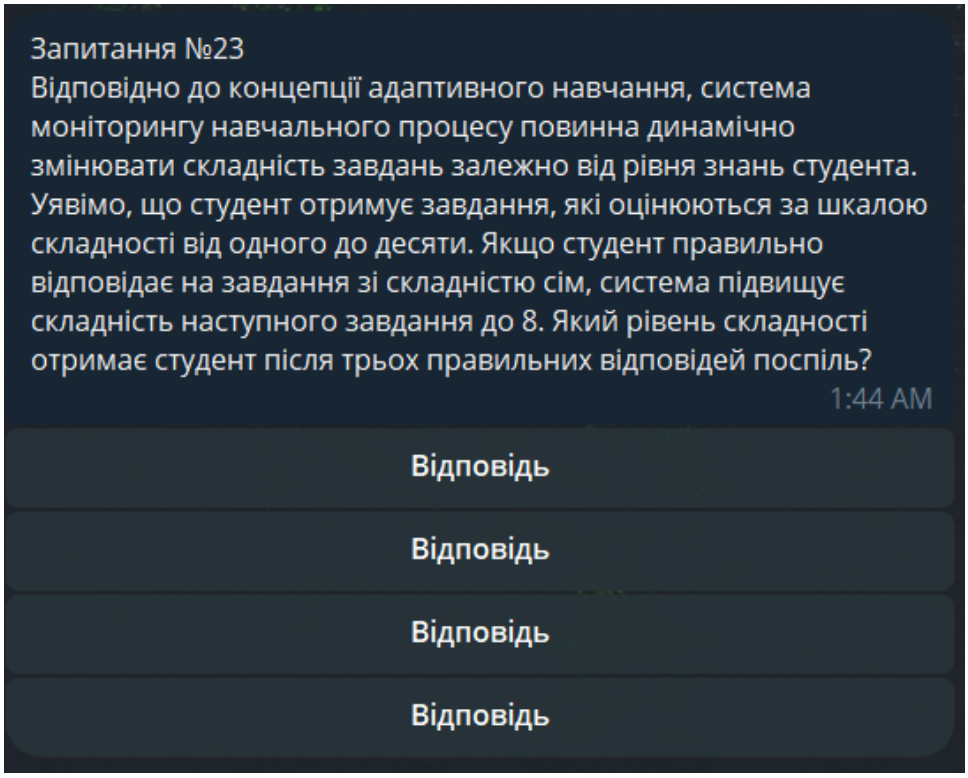

Рисунок В.6 – Процес проходження тесту користувачем

Додаток Г (довідниковий)
Інструкція користувача

Інтерфейс користувача представлений ботом в соціальній мережі Телеграм.

Для того щоб почати роботу з ботом, натисніть кнопку «Розпочати» у його вікні або ж надіслати команду «/start». Після запуску вам відкриється головне меню, яке містить основні функції (рис. Г.1). Ви можете обирати необхідні розділи, просто натискаючи на відповідні кнопки.

У розділі «Профіль» (рис. Г.2) ви можете переглянути інформацію про себе, наприклад, ім'я, рейтинг чи рівень активності. Якщо потрібно внести зміни, скористайтеся кнопкою «Редагувати профіль», щоб оновити свої дані. Для оновлення доступна опція «Псевдонім». Ваш псевдонім буде показаний усім користувачам в усіх класах, до яких ви доєднаєтесь. Будучи викладачем (власником класу) ви можете задавати власний аліас для кожного учня, який буде відображатись лише для вас у всіх наявних класах. «Мій рейтинг» показує поточний рейтинг користувача в певний момент часу серед усіх активних користувачів системи.

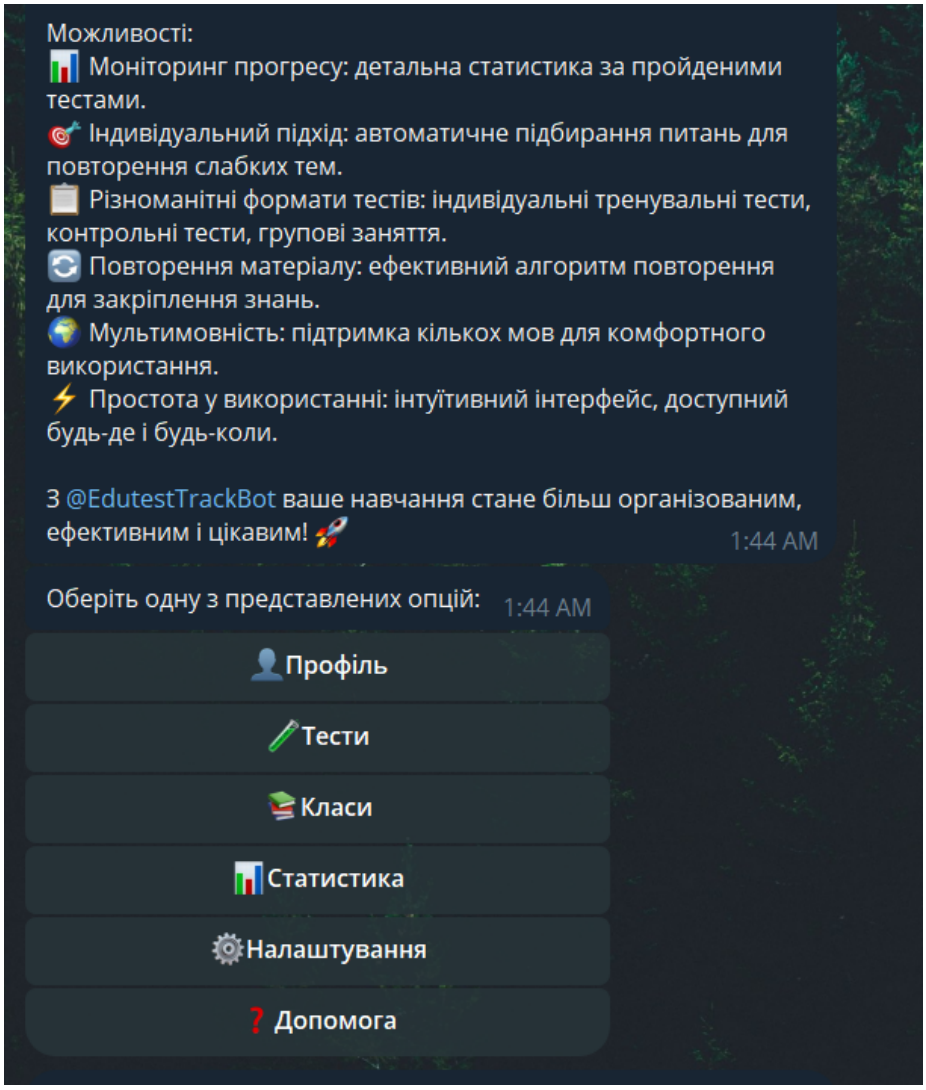

Рисунок Г.1 – Головне меню розробленого телеграм бота

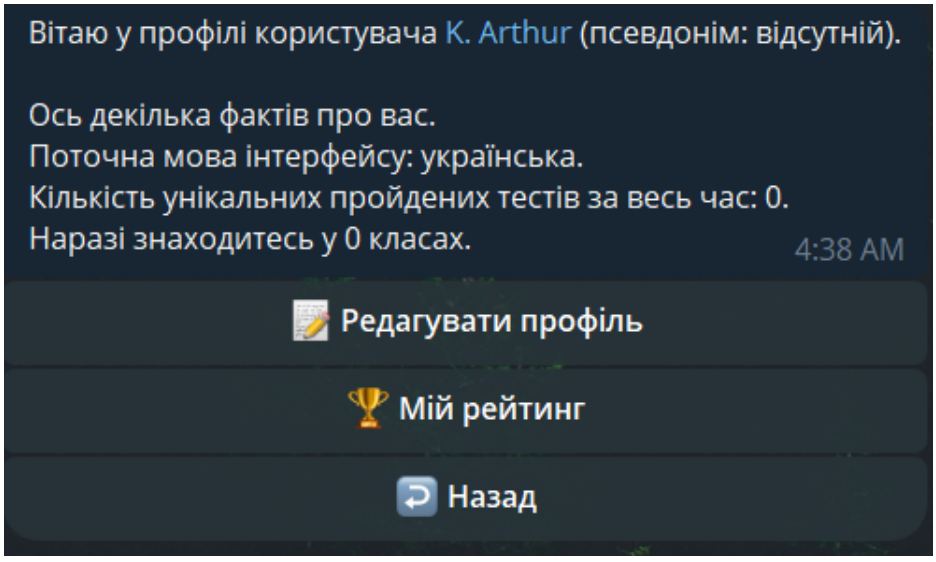

Рисунок Г.2 – Меню «Профіль» розробленого телеграм бота

Щоб знайти та пройти тести, відкрийте розділ «Тести» (рис. Г.3). Використовуйте функцію «Пошук тесту» для знаходження потрібного матеріалу за ключовими словами. Після вибору тесту почніть проходження, а по завершенню перегляньте свої результати у вікні статистики. Ви також можете створювати власні тести у цьому розділі, додаючи запитання та відповіді.

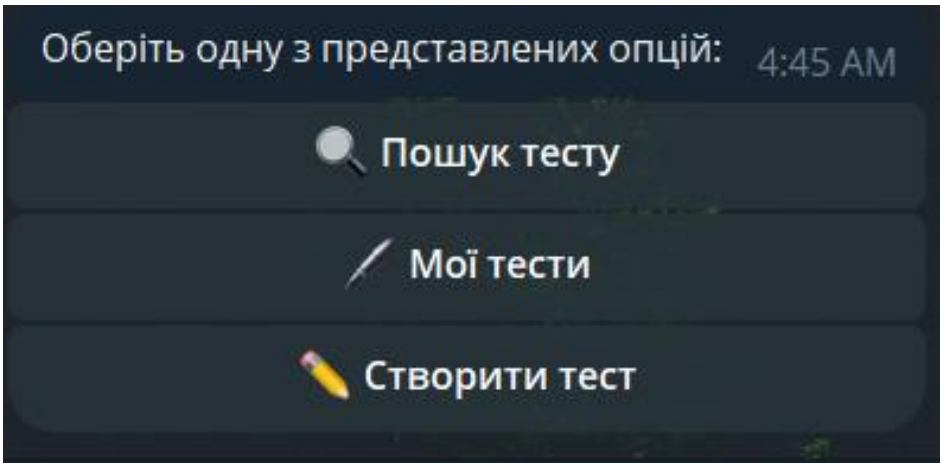

Рисунок Г.3 – Меню «Тести» розробленого телеграм бота

Для роботи з групами користувачів зайдіть у розділ «Класи» (рис. Г.4). Ви можете створити новий клас для своїх учнів, натиснувши «Створити клас», або приєднатися до існуючого за допомогою коду. У вкладці «Мої класи» доступний повний список класів, якими ви керуєте чи до яких ви приєднані.

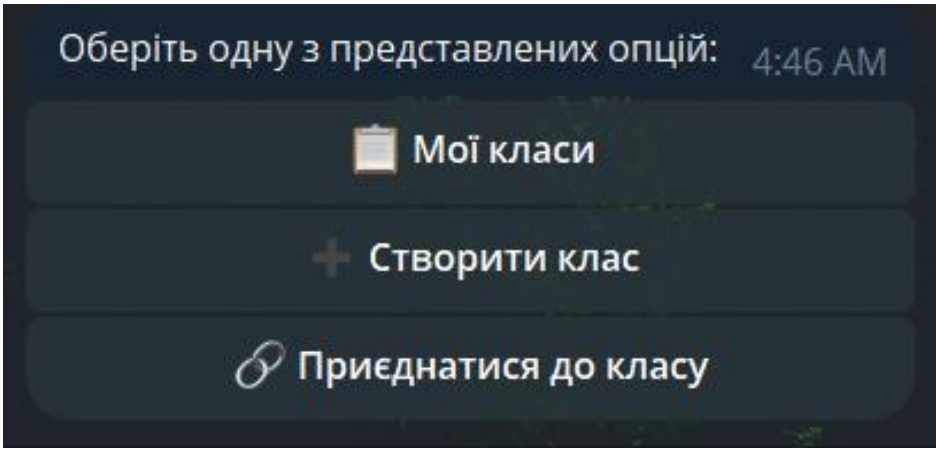

Рисунок Г.4 – Меню «Класи» розробленого телеграм бота

Розділ «Статистика» (рис. Г.5) дозволяє переглядати ваш прогрес у навчанні. Ви можете подивитися загальні результати у вкладці «Моя статистика». Якщо ви є викладачем, для вас також доступна функція «Статистика учнів», де відображається успішність усіх членів вашого класу.

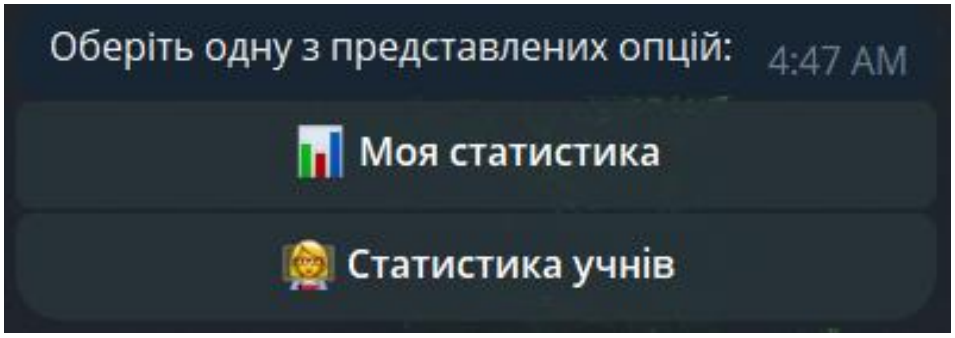

Рисунок Г.5 – Меню «Статистика» розробленого телеграм бота

Щоб змінити мову інтерфейсу, відкрийте розділ «Налаштування» та виберіть пункт «Зміна мови». Оберіть потрібну мову зі списку, і бот автоматично оновить усі тексти.

Якщо у вас виникли запитання або проблеми, скористайтеся розділом «Допомога». Там ви знайдете короткий посібник з основних функцій бота та зможете залишити відгук чи пропозицію у вкладці «Зворотній зв'язок».

Ця інструкція допоможе вам освоїти всі можливості бота та максимально ефективно використовувати його для навчання.

Додаток Д (довідниковий)

Акт впровадження результатів магістерської кваліфікаційної роботи

АСОЦІАЦІЯ ІТ КОМПАНІЙ ВІННИЦІ

ЗАТВЕРДЖУЮ
Виконавчий директор
Дар'я НИШПОРСЬКА
«1» грудня 2024 р

АКТ

впровадження результатів магістерської кваліфікаційної роботи
Вишневського Артура В'ячеславовича на тему:
«Інформаційна технологія моніторингу процесу навчання»

Впроваджені результати магістерської кваліфікаційної роботи, а саме програмний засіб для моніторингу процесу успішно використовувався для внутрішнього навчання співробітників Асоціації ІТ компаній Вінниці. Його впровадження дозволило підвищити ефективність працівників, завдяки персоналізованим рекомендаціям щодо навчальних матеріалів, які потребують подальшого вивчення.

Практична цінність програмного засобу полягає у його здатності оптимізувати процес навчання та підвищити швидкість засвоєння знань, що має значний потенціал для подальшого впровадження в освітні та корпоративні системи.

Виконавчий директор
Дар'я НИШПОРСЬКА