Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра обчислювальної техніки

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

КОМП'ЮТЕРНА СИСТЕМА ПЛАНУВАННЯ ТА ОЦІНЮВАННЯ НАВЧАЛЬНОЇ ДІЯЛЬНОСТІ СТУДЕНТА

Виконав: студент 2 курсу, групи 1КІ-24м
спеціальності 123 — Комп'ютерна інженерія
Івасюк В.В.
Керівник: к.т.н., доц. каф. ОТ
Снігур А.В.
"12" 12 2025 р.
Опонент: к.т.н, доц. каф. ПЗ
Ракитянська Г.Б.
"12" 12 2025 р.

Допущено до захисту
Зав. кафедри ОТ
д.т.н., проф. Азаров О.Д.
«16» грудня 2025 р.

ВНТУ 2025

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Галузь знань — Інформаційні технології
Освітній рівень — магістр
Спеціальність — 123 Комп'ютерна інженерія
Освітньо-професійна програма — Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри обчислювальної техніки
д.т.н., проф. Азаров О.Д.
«25» 09 2025 р.

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

студенту Івасюку Вадиму Віталійовичу

1 Тема роботи: «Комп'ютерна система планування та оцінювання навчальної діяльності студента», керівник роботи Снігур А.В. к.т.н., доцент кафедри ОТ, затверджено наказом вищого навчального закладу від 24.09.2025 року №313.

2 Строк подання студентом роботи 4.12.2025 р.

3 Вихідні дані до роботи: засоби — середовище розробки Visual Studio Code, інтерфейс системи — застосунок для платформи Android/IOS, розроблений на мові програмування JavaScript з використанням фреймворку React Native, передбачено розробити — систему планування та оцінювання навчальної діяльності студента.

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): вступ, аналітичний огляд методів оцінювання навчальної діяльності студента, моделювання та проєктування системи планування та оцінювання навчальної діяльності студента, програмна реалізація системи планування та оцінювання навчальної діяльності студента, тестування розробленого програмного забезпечення, економічна частина.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): алгоритм процесу обліку результатів навчальної діяльності.

6 Консультанти розділів роботи приведені в таблиці 1.

Таблиця 1 — Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Снігур Анатолій Васильович, к.т.н., доцент каф. ОТ		
5	Ратушняк Ольга Георгіївна, к.т.н., доцент каф. ЕПВМ		
Нормо контроль	Швець Сергій Ілліч, асистент каф. ОТ		

7 Дата видачі завдання 25.09.2025 р.

8 Календарний план виконання МКР приведений в таблиці 2.

Таблиця 2 — Календарний план

№ з/п	Назва етапів МКР	Строк виконання	Примітка
1	Постановка задачі	25.09.2025-26.09.2025	виконано
2	Огляд існуючих рішень	27.09.2025-03.10.2025	виконано
3	Розробка структури системи	04.10.2025-09.10.2025	виконано
4	Вибір ПЗ для реалізації системи	10.10.2025-14.10.2025	виконано
5	Розробка інтерфейсу користувача	15.10.2025-17.10.2025	виконано
6	Програмна реалізація системи	18.10.2025-25.10.2025	виконано
7	Підготовка матеріалів для опису програмного забезпечення	26.10.2025-28.10.2025	виконано
8	Розрахунок економічної частини	29.10.2025-02.11.2025	виконано
9	Оформлення пояснювальної записки	03.11.2025-10.11.2025	виконано
10	Попередній захист	11.11.2025-12.11.2025	виконано
11	Підписи супроводжувальних документів	13.11.2025-20.11.2025	виконано
12	Перевірка якості виконання магістерської кваліфікаційної роботи та усунення недоліків	21.11.2025-3.12.2025	виконано

Студент _____ Івасюк Вадим Віталійович

Керівник _____ к.т.н., доц. каф. ОТ Снігур Анатолій Васильович

АНОТАЦІЯ

УДК 004.4:37

Івасюк В.В. Комп’ютерна система планування та оцінювання навчальної діяльності студента. Магістерська кваліфікаційна робота зі спеціальності 123 — Комп’ютерна інженерія, Вінниця: ВНТУ, 2025 — 132 с.

На укр. мові. Бібліогр.: 25 назв; рис: 16; табл.: 14.

У роботі розроблено комп’ютерну систему планування та оцінювання навчальної діяльності студента в умовах дистанційного й змішаного навчання. Проведено аналіз існуючих підходів до контролю знань і моделей автоматизованого оцінювання відкритих текстових відповідей на основі векторного представлення та семантичної подібності. Запропоновано структуру системи, розроблено модель процесу оцінювання із урахуванням часових штрафів та нормування результатів. Реалізовано мобільний застосунок на базі платформи React Native з підтримкою розподілу завдань, надсилання відповідей, автоматичного формування попередньої оцінки та ведення обліку результатів. Проведено тестування програмного забезпечення.

Ключові слова: комп’ютерна система, планування навчальної діяльності, автоматизоване оцінювання, семантична подібність, мобільний застосунок.

ABSTRACT

Ivasiuk V.V. Computer System for Planning and Evaluation of Student Learning Activity. Master's thesis in the specialty 123 — Computer Engineering, Vinnytsia: VNTU, 2025.

In Ukrainian language.

The thesis examines the principles of designing a computer system for planning and evaluating student learning activity in the context of distance and blended learning. Existing approaches to knowledge control and models of automated assessment of open-ended textual answers based on vector representations and semantic similarity are analysed. The system architecture is proposed, a model of the evaluation process with time penalties and algorithms for processing answers are developed. A mobile application based on the React Native platform is implemented, providing task distribution, submission of answers, automatic generation of a preliminary grade, and recording of results. Software testing has been carried out.

Keywords: computer system, learning activity planning, automated assessment, semantic similarity, mobile application.

ЗМІСТ

ВСТУП 8
1 АНАЛІТИЧНИЙ ОГЛЯД МЕТОДІВ ОЦІНЮВАННЯ НАВЧАЛЬНОЇ ДІЯЛЬНОСТІ СТУДЕНТА 10
1.1 Актуальність комп'ютерних систем у плануванні та оцінюванні навчальної діяльності 10
1.2 Аналіз підходів до оцінювання навчальної діяльності студентів 15
1.3 Аналіз існуючих моделей автоматизованого оцінювання навчальних завдань 18
1.3.1 Аналіз методів векторизації тексту 20
1.3.2 Аналіз методів обчислення міри семантичної подібності 24
1.4 Порівняльний аналіз існуючих програмних рішень і систем-аналогів 26
2 МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ СИСТЕМИ ПЛАНУВАННЯ ТА ОЦІНЮВАННЯ НАВЧАЛЬНОЇ ДІЯЛЬНОСТІ СТУДЕНТА 31
2.1 Побудова структури системи планування та оцінювання навчальної діяльності студента 31
2.2 Розробка моделі оцінювання завдань 36
2.3 Моделювання процесу обліку результатів навчання 40
2.4 Розробка алгоритму роботи програми 42
3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ПЛАНУВАННЯ ТА ОЦІНЮВАННЯ НАВЧАЛЬНОЇ ДІЯЛЬНОСТІ СТУДЕНТА 46
3.1 Вибір засобів розробки системи 46
3.1.1 Вибір інструментів розробки клієнтської частини 46
3.1.2 Вибір інструментів розробки серверної частини 50
3.2 Розробка модуля авторизації 53
3.3 Розробка користувацького інтерфейсу 58
3.4 Розробка серверної частини 69
4 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 76
4.1 Тестування роботи системи 76
4.2 Інструкція користувача 80
5 ЕКОНОМІЧНА ЧАСТИНА 86
5.1 Оцінювання комерційного потенціалу розробки 86
5.2 Прогнозування витрат на виконання науково-дослідної роботи 93

5.3 Розрахунок економічної ефективності науково-технічної розробки 99
5.4 Розрахунок ефективності вкладених інвестицій та періоду їх окупності 101
ВИСНОВКИ 104
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 106
ДОДАТОК А Технічне завдання 108
ДОДАТОК Б Протокол перевірки кваліфікаційної роботи 112
ДОДАТОК В Алгоритм процесу обліку результатів навчальної діяльності 113
ДОДАТОК Г Лістинг сторінки перегляду та оцінювання відповідей 114
ДОДАТОК Д Лістинг екрана створення нового завдання 122
ДОДАТОК Е Лістинг екрана статистики студента 125
ДОДАТОК Ж Лістинг модуля оцінювання на серверній частині 129

ВСТУП

Сучасна система освіти зазнає значних трансформацій під впливом цифрових технологій. Традиційні форми організації навчання, засновані на паперових матеріалах та аудиторних заняттях, уже не здатні повною мірою задовольнити потреби студентів та викладачів у гнучкості, інтерактивності та оперативності. У цих умовах важливу роль відіграють комп'ютерні навчальні системи, у яких інтегровано методи планування, контролю та оцінювання навчальної діяльності. Їх застосування призводить до оптимізації навчального процесу, робить стабільним доступ до матеріалів та підвищує об'єктивність оцінювання результатів.

Актуальність дослідження обумовлена потребою у створенні комплексної комп'ютерної системи, що поєднує функції планування, організації та оцінювання навчальної діяльності студентів. Система, що враховує сучасні методи автоматизованого та семантичного оцінювання, підвищує ефективність освітнього процесу, скорочує час перевірки завдань, зменшує суб'єктивний вплив викладача та підтримує гнучкість навчання.

Метою роботи є підвищення об'єктивності та ефективності оцінювання навчальної діяльності студента в умовах дистанційного навчання шляхом розробки комп'ютерної системи планування і автоматизованого оцінювання відкритих текстових відповідей.

Для досягнення поставленої мети поставлено наступні **задачі**:

— проаналізувати існуючі підходи до організації планування та оцінювання навчальної діяльності студента;

— розробити структуру системи планування та оцінювання навчальної діяльності студента;

— розробити модель автоматизованого оцінювання відповідей на основі векторного представлення тексту та обчислення семантичної подібності;

— реалізувати інтерфейс користувача та механізми здачі завдань;

— провести тестування системи.

Для досягнення поставленої в роботі мети використовуються такі **методи дослідження:**

— системний і порівняльний аналіз;

— методи інформаційного моделювання;

— експериментальні методи.

Об’єкт дослідження — процес планування та оцінювання результатів навчальної діяльності студентів у електронному освітньому середовищі.

Предмет дослідження — методи та засоби автоматизованого оцінювання текстових завдань на основі аналізу подібності текстів.

Наукова новизна роботи полягає в розширенні функціональних можливостей системи обліку результатів навчальної діяльності за рахунок додавання модуля автоматизованого оцінювання відповідей на основі семантичного порівняння з еталоном. На відміну від відомих систем, де облік результатів переважно зводиться до фіксації виставлених балів, у роботі запропоновано гібридний підхід, що поєднує семантичний аналіз відповідей із експертним контролем викладача.

Практичне значення роботи полягає в створенні прикладного рішення, яке підвищує ефективність організації навчальної діяльності та об’єктивність оцінювання. Інтеграція системи скорочує адміністративні витрати на узгодження завдань і критеріїв оцінювання, прискорює надання зворотного зв’язку студентам та зменшує навантаження на викладача за рахунок автоматичної попередньої перевірки відповідей і вибіркового експертного контролю у складних випадках.

Апробація результатів роботи здійснена в доповіді на Молодь в науці: дослідження, проблеми, перспективи, міжнародна науково-практична інтернет-конференція.

Матеріали роботи доповідались та опубліковувались [1]:

Івасюк В.В., Снігур А.В., Циркун В.В. Комп’ютерна система планування та оцінювання навчальної діяльності студента [Електронний ресурс] — Режим доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2026/paper/view/26272

1 АНАЛІТИЧНИЙ ОГЛЯД МЕТОДІВ ОЦІНЮВАННЯ НАВЧАЛЬНОЇ ДІЯЛЬНОСТІ СТУДЕНТА

1.1 Актуальність комп’ютерних систем у плануванні та оцінюванні навчальної діяльності

У сучасних умовах розвитку інформаційного суспільства комп’ютерні системи, такі як системи управління навчанням, електронні журнали успішності, платформи дистанційного та змішаного навчання, сервіси автоматизованого тестування й аналітики освітніх даних, є не просто допоміжним інструментом у сфері освіти, а ключовим елементом її функціонування. Вони забезпечують цілісну цифрову інфраструктуру, у межах якої реалізується процес управління навчанням, організація навчального контенту, взаємодія між учасниками освітнього процесу та моніторинг результатів. Застосування комп’ютерних систем у навчальному процесі формує нову дидактичну парадигму, у якій технологічна основа не лише підтримує педагогічну діяльність, але й модифікує саму структуру взаємодії між викладачем, студентом і навчальним середовищем.

Роль комп’ютерних систем полягає у створенні єдиного інформаційно-освітнього простору, який поєднує функції зберігання, обробки та відображення навчальної інформації, забезпечує адаптивність освітнього процесу, а також підвищує рівень прозорості та контрольованості освітніх результатів. Завдяки інтеграції програмних платформ, баз даних, мережевих ресурсів і аналітичних інструментів, такі системи стають інтелектуальним посередником між педагогічними методиками та технологічними можливостями сучасних засобів навчання.

Одним із головних аспектів застосування комп’ютерних систем у навчанні є їх здатність забезпечувати безперервність освітнього процесу незалежно від просторово-часових обмежень. Це особливо важливо в умовах дистанційної та змішаної освіти, де навчальні платформи виступають основним середовищем комунікації, зберігання та перевірки знань. Унаслідок впровадження таких рішень змінюється не лише форма відображення навчального матеріалу, а й структура

навчальної діяльності, яка набуває рис гнучкості, персоналізації та технологічної інтерактивності.

У загальному вигляді роль комп'ютерних систем полягає у виконанні чотирьох основних груп функцій: організаційних, освітніх, комунікаційних та оціночних [2].

Порівняно з традиційними формами організації освітнього процесу, цифрові системи демонструють значно вищу ефективність у забезпеченні зворотного зв'язку, моніторингу прогресу та підтримці індивідуальних траєкторій навчання. Класичне навчання переважно спирається на дискретний контроль, тобто проміжні і підсумкові тести. Сучасні освітні платформи реалізують безперервний процес оцінювання. Завдяки цьому фіксується не лише кінцевий результат, а й динаміка навчального розвитку, що відкриває можливість формувального оцінювання, орієнтованого не на покарання за помилки, а на їх аналітичне виявлення й подальше усунення.

Організаційні функції зосереджені на структуризації та координації навчального процесу. Вони дозволяють автоматизувати рутинні завдання, такі як формування розкладів занять, управління курсами та контроль строків здачі завдань. Системи розподіляють навчальні матеріали та ресурси між користувачами з різними ролями, серед яких студенти, викладачі або адміністратори. Наприклад, в університетських середовищах організаційна функція охоплює планування освітніх програм, моніторинг виконання навчального навантаження та координацію ресурсів на рівні кафедр чи цілих закладів. Це перетворює хаотичні процеси на єдину координаційну систему, де дані про доступність ресурсів оновлюються в реальному часі, дозволяючи адміністраторам оперативно реагувати на зміни. Ефективність роботи зростає, оскільки викладачі концентруються на креативних і методичних аспектах, а студенти отримують чіткі інструкції без інформаційної плутанини. Однак, для повної реалізації потрібна відповідна інфраструктура, яка забезпечує масштабованість і безпеку даних.

Освітні функції спрямовані на надання доступу до навчальних матеріалів у зручному та інтерактивному форматі, перетворюючи пасивне сприйняття на

активне засвоєння знань. Це включає мультимедійні курси, відеолекції, інтерактивні вправи, тестові завдання, моделі та симуляції, які формують інтегроване навчальне середовище. Системи підтримують диференціацію навчання, адаптуючи складність матеріалу до рівня підготовки кожного користувача через алгоритми адаптивного навчання. У вищій освіті це особливо корисно для спеціалізованих дисциплін, де симуляції дозволяють практикувати навички без реальних ризиків, як у медичних чи інженерних курсах. Подібна організація навчальних матеріалів істотно підвищує залученість студентів. Вони не просто читають текст, а взаємодіють із контентом, що забезпечує глибше розуміння матеріалу.

Комунікаційна функція спрямована на створення цифрового простору взаємодії між усіма учасниками навчального процесу. Вона реалізується через форуми, чати, коментарі до завдань, систему повідомлень, відеоконференції та спільні онлайн-документи. Це дозволяє підтримувати безперервний зворотний зв’язок, що є основою ефективного навчання. Завдяки цьому усувається комунікативний розрив між викладачем і студентом, який часто виникає у дистанційному навчанні. Крім того, такі засоби сприяють формуванню спільнот практики, де студенти можуть обмінюватися досвідом, обговорювати завдання та колективно вирішувати навчальні проблеми. За таких умов посилюється соціальна взаємодія, студенти відчувають підтримку, а викладачі отримують змогу відстежувати динаміку роботи групи. Комунікаційні функції перетворюють ізольоване навчання на спільне, сприяючи обміну знаннями та зменшенню відчуття самотності в онлайн-середовищі.

Оціночна функція фокусується на інструментах тестування, аналітики та моніторингу результатів у реальному часі, роблячи оцінювання об’єктивним і діагностичним. Вона автоматизує перевірку знань, формує статистику успішності, відстежує динаміку навчання та виявляє проблемні теми через алгоритми інтелектуального аналізу даних. Сучасні системи інтегрують алгоритми інтелектуального аналізу даних для прогнозування ризиків неуспішності та формування персоналізованих рекомендацій, перетворюючи оцінку з кінцевого

вердикту на інструмент розвитку. У вищій освіті це корисно для масштабних курсів, де ручна перевірка неможлива; аналітика виявляє закономірності в поведінці, як частота взаємодій, для раннього втручання. Використання таких інструментів підвищує об'єктивність оцінювання, перетворюючи його з суб'єктивного судження на індикатор, орієнтований на дані та освітнє зростання. Оціночні функції підвищують ефективність, дозволяючи викладачам фокусуватися на сильних сторонах студентів.

Комп'ютерні системи в освіті значно трансформують ролі викладача та студента. Для викладача комп'ютерні системи виступають насамперед інструментом організації та оптимізації праці [3]. Вони дозволяють:

— створювати електронні завдання, тести та опитування з використанням шаблонів і автоматизованих інструментів;

— відстежувати виконання завдань студентами в реальному часі через інформаційні панелі;

— отримувати автоматизовані звіти про успішність групи;

— коригувати навчальні плани на основі даних про ефективність студентів

— інтегрувати інструменти для спільної роботи.

Ці можливості зменшують адміністративне навантаження на викладача, дозволяючи приділяти більше уваги методичній роботі та індивідуальній підтримці студентів. Системи можуть автоматично виявляти студентів із низькою залученістю, дозволяючи викладачу запропонувати консультації чи додаткові ресурси. Персоналізація навчання через багаторівневі завдання та адаптивні рекомендації забезпечує індивідуалізований підхід, що складно реалізувати в традиційній аудиторії. Такі інструменти підвищують ефективність викладання, але вимагають від викладачів цифрової грамотності для повноцінного використання.

Для студента університетські системи управління навчанням, такі як Moodle чи Google Classroom, виконують мотиваційну та організаційну роль. Вони надають доступ до матеріалів у будь-який час, що особливо важливо для тих, хто поєднує навчання з роботою або має індивідуальний графік.

Основні переваги для студента:

— можливість працювати у власному темпі;

— миттєве отримання результатів тестування;

— доступ до додаткових ресурсів;

— зручність у комунікації з викладачем і групою;

— можливість бачити власний прогрес.

Окрему роль відіграє формування цифрових компетентностей. У процесі використання комп'ютерних систем студент опановує навички роботи з електронними ресурсами, управління інформацією, цифрової комунікації, що є необхідними у професійному середовищі будь-якого фахівця. Навчальна платформа у такій конфігурації функціонує не лише як інструмент передачі знань, а й як засіб формування інформаційної грамотності здобувачів освіти.

Додатковим мотиваційним чинником стає можливість бачити власний прогрес у реальному часі, коли результативність навчання подається у наочній формі.

Аналітична роль комп'ютерних систем, яка ґрунтується на аналітиці навчальних даних, трансформує освітній процес через систематичну обробку інформації про діяльність студентів. Кожна взаємодія студента з цифровою платформою, починаючи від перегляду матеріалів і закінчуючи виконанням завдань, генерує дані, які аналізуються для виявлення закономірностей у навчальній поведінці. Комп'ютерні системи дозволяють відстежувати активність, таку як частота доступу до ресурсів чи час, витрачений на завдання, і на основі цього надавати персоналізовані рекомендації. Наприклад, система може визначити, що студент має труднощі з певною темою, і запропонувати додаткові ресурси чи консультації. Аналітика також дає змогу викладачам оптимізувати навчальні програми, виявляючи теми, які систематично викликають труднощі, і вдосконалювати методи викладання. Застосування описаних механізмів аналізу навчальних даних підвищує ефективність освіти, робить її більш адаптивною та зорієнтованою на індивідуальні потреби [4].

1.2 Аналіз підходів до оцінювання навчальної діяльності студентів

Оцінювання навчальних результатів є фундаментальним елементом освітньої системи. Воно забезпечує зворотний зв’язок між процесом навчання та його результатом, визначає рівень засвоєння знань і формується як об’єктивний індикатор ефективності освітнього процесу. Рівень надійності та достовірності системи оцінювання безпосередньо впливає на якість навчання, точність прийняття педагогічних рішень і формування освітніх траєкторій.

Традиційне оцінювання базується на експертному аналізі результатів навчальної діяльності викладачем. Його основною характеристикою є глибина інтерпретації відповідей, можливість врахування змістовних аспектів, аргументації та логіки міркувань студента. Водночас цей підхід характеризується значними часовими витратами, низьким рівнем масштабованості та високим ступенем суб’єктивності [5].

Автоматизоване оцінювання передбачає застосування алгоритмів для порівняння отриманих відповідей з еталонними розв’язками або шаблонами. У таких системах часто використовуються методи тестування, аналіз синтаксичних структур, семантичне зіставлення та статистичні показники схожості. У порівнянні з традиційною перевіркою такі процедури працюють значно швидше і дають відтворювані результати, однак здатність коректно аналізувати творчі або розгорнуті відповіді для них залишається обмеженою [6].

Адаптивне оцінювання функціонує на принципах динамічного підбору складності завдань відповідно до поточного рівня підготовки студента. Такі системи використовують моделі ймовірнісного оцінювання, аналітичні функції успішності та параметри складності тестових елементів. Адаптивні методи підвищують точність визначення рівня знань, зменшуючи ймовірність випадкових результатів.

Комбіновані методи оцінювання поєднують алгоритмічні принципи автоматичної перевірки з елементами експертного контролю. Вони зберігають об’єктивність і водночас забезпечувати якісну інтерпретацію результатів у

завданнях відкритого типу. Комбіновані системи часто впроваджуються у платформах змішаного навчання, де необхідне поєднання автоматичного аналізу з викладацьким судженням. Порівняльна характеристика цих підходів подана в таблиці 1.1.

Таблиця 1.1 — Порівняльна характеристика підходів до оцінювання результатів навчального процесу

Критерій	Традиційне	Автоматизоване	Адаптивне	Комбіноване
Участь викладача	Повна	Мінімальна	Обмежена	Часткова
Швидкість	Низька	Висока	Висока	Середня
Об’єктивність	Суб’єктивна	Формальна	Висока	Збалансована
Гнучкість	Обмежена	Стандартна	Висока	Висока
Масштабованість	Низька	Висока	Висока	Висока
Типи завдань	Відкриті	Формалізовані	Змішані	Будь-які
Достовірність	Середня	Висока	Висока	Найвища
Зворотний зв’язок	Повільний	Миттєвий	Динамічний	Комбінований
Персоналізація	Відсутня	Низька	Висока	Висока
Трудомісткість	Висока	Низька	Середня	Помірна

Проведене порівняння показує, що кожен із розглянутих підходів до оцінювання навчальних результатів має специфічну сферу ефективного застосування, обумовлену характером навчальної діяльності, типом завдань та необхідним рівнем автоматизації процесу контролю знань.

Традиційні методи забезпечують глибину аналізу відповідей, однак характеризуються високою трудомісткістю, залежністю від людського фактору та обмеженою масштабованістю. Такий вид оцінювання залишається найефективнішими у випадках, коли важливо проаналізувати не лише кінцевий результат, а й логіку міркувань, аргументацію та послідовність мислення студента. Такі методи є оптимальними для гуманітарних дисциплін, творчих завдань, есе, проєктних робіт або усних виступів, де формальні критерії не здатні повною мірою охопити змістовну складову відповіді. Водночас значна залежність від суб’єктивного судження викладача та низька масштабованість обмежують

можливість їх використання у великих навчальних групах або дистанційних курсах. Їх використання є виправданим під час підсумкової атестації, коли пріоритетом є якість, а не швидкість перевірки.

Автоматизовані методи оцінювання найбільш доцільні в середовищах, де необхідна обробка великої кількості однотипних завдань у стислі терміни. У межах масових онлайн-курсів ці методи забезпечують оптимальне співвідношення швидкодії та точності. Їхнє застосування є оптимальним для тестових форм контролю, програмування, математичних дисциплін і лінгвістичних тренажерів, де можливо формалізувати правильну відповідь або набір допустимих рішень. Висока швидкість перевірки, стабільність критеріїв та відсутність людського чинника забезпечують об’єктивність і рівність умов для всіх студентів. Однак такі системи малоефективні у випадках відкритих завдань, що потребують інтерпретації контексту або аргументованих пояснень.

Адаптивні методи оцінювання є найбільш гнучкими у контексті персоналізації навчального процесу. Вони здатні враховувати додаткові параметри, такі як темп роботи, стабільність відповідей, часові затримки та забезпечують динамічне коригування рівня складності запитань залежно від поточних результатів студента. Унаслідок такого налаштування уникаються ситуації надмірного навантаження або, навпаки, занадто низької складності завдань. Такі системи застосовуються у діагностичних тестах, платформах формувального оцінювання та середовищах індивідуального навчання, де важливо не лише визначити рівень знань, а й відстежити динаміку прогресу. . За рахунок цих властивостей адаптивні схеми оцінювання точніше відображають рівень підготовки й індивідуалізують навчальний досвід, хоча їх реалізація потребує значних обчислювальних ресурсів і розвиненої аналітичної інфраструктури.

Комбіновані методи оцінювання поєднують автоматизовану перевірку об’єктивних критеріїв із експертним аналізом змістовних аспектів відповіді. Комбіновані системи є найефективнішими для комплексних дисциплін, що поєднують практичні, аналітичні та теоретичні елементи, наприклад у проєктному навчанні, технічних спеціальностях або при підсумковій атестації. Вони

створюють основу для впровадження аналітичних модулів, здатних відстежувати рівень засвоєння матеріалу в динаміці, що робить їх перспективними для інтелектуальних навчальних платформ.

Вибір методу оцінювання повинен базуватись на принципі функціональної відповідності між цілями навчального процесу та рівнем автоматизації контролю. Для базових дисциплін із чітко визначеними правильними відповідями оптимальними є автоматизовані системи, тоді як у сферах, де важливим є якісний аналіз і творчий підхід, більш доцільним є використання традиційних або комбінованих методів. Адаптивні підходи, у свою чергу, виступають універсальним інструментом персоналізації, який може бути інтегрований у будь-який із зазначених типів систем для підвищення точності та гнучкості навчального контролю.

1.3 Аналіз існуючих моделей автоматизованого оцінювання навчальних завдань

Ефективність систем дистанційного та змішаного навчання безпосередньо залежить від здатності механізмів контролю знань забезпечувати точну, стабільну та змістовно обґрунтовану оцінку результатів навчальної діяльності. Попередній аналіз показав, що традиційні, автоматизовані, адаптивні та комбіновані підходи мають суттєві переваги, але жоден із них не забезпечує одночасно глибину розуміння змісту, масштабованість і незалежність від людського чинника.

Традиційні методи забезпечують найвищу якість інтерпретації відповідей, однак їх використання обмежене трудомісткістю та суб’єктивністю. Автоматизовані системи, навпаки, характеризуються високою швидкістю й об’єктивністю, проте здатні оцінювати лише формальні або тестові відповіді, не аналізуючи логічні та семантичні зв’язки між поняттями. Адаптивні моделі частково розв’язують проблему індивідуалізації, але їхня ефективність значною мірою залежить від точності вихідних даних і можливостей алгоритмів класифікації.

Автоматизоване оцінювання відкритих відповідей у системах дистанційного навчання є складним завданням, оскільки потребує не лише формального розпізнавання правильних відповідей, а й глибокого аналізу їхнього змісту.

У найпростіших моделях текст відповіді розглядається як множина лексичних елементів, а ступінь відповідності між еталонною та студентською відповідями оцінюється за ступенем перекриття цих множин. Однією з найпростіших та найефективніших мір подібності є подібність Жаккара, яка кількісно визначає, наскільки дві множини перекриваються. У задачах аналізу текстів цей коефіцієнт використовується для оцінювання схожості документів, зокрема у системах виявлення текстової подібності та плагіату, де тексти попередньо подаються у вигляді множин токенів [7]. Він визначається як відношення перетину множин до їх об'єднання (1.1):

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}, \tag{1.1}$$

де A — множина змістовних елементів еталонної відповіді;

B — множина елементів відповіді студента.

Значення коефіцієнта змінюється від нуля до одиниці, нуль відповідає повній відсутності спільних змістових елементів, одиниця повному збігу множин. На відміну від простого підрахунку частки знайдених ключових слів, такий підхід одночасно враховує і відсутність важливих елементів, і наявність у відповіді великої кількості другорядних або випадкових лексичних одиниць.

Проте коефіцієнт Жаккара нечутливий до синонімічних замін, не враховує зміну значення слова залежно від контексту і не відображає логічну структуру міркування у відповіді. Дві відповіді, які передають однаковий зміст різними формулюваннями, можуть мати низьке значення коефіцієнта Жаккара, хоча з позиції викладача вони є змістовно еквівалентними. Тому для коректного аналізу відкритих відповідей потрібні моделі, здатні працювати із семантичними зв’язками між фрагментами тексту, а не лише з буквальними збігами окремих слів.

Основна проблема полягає в тому, що людська мова має складну семантичну структуру. Одна і та сама думка може бути сформульована різними словами, а слова можуть мати кілька значень залежно від контексту. Для подолання цієї проблеми сучасні системи оцінювання використовують методи векторного представлення тексту, які перетворюють лінгвістичні конструкції у числові вектори, що відображають зміст повідомлення в багатовимірному просторі. На основі цих векторів здійснюється обчислення міри семантичної подібності між еталонною та студентською відповідями, що забезпечує кількісне оцінювання змістовної близькості. Цей процес можна розділити на два ключові етапи: векторизація тексту та обчислення міри подібності [8].

1.3.1 Аналіз методів векторизації тексту

Першим етапом аналізу тексту є векторизація, тобто формалізація тексту у вигляді числового представлення. Одним із базових методів є модель «мішок слів», яка представляє текст як набір незалежних термінів без урахування порядку чи граматичних зв’язків. Процес перетворення тексту на вектор починається з побудови словника всіх унікальних слів у корпусі текстів, після чого кожен текст представляється як вектор, де кожна компонента відповідає кількості появи певного слова з словника в цьому тексті (1.2).

$$\vec{t} = (t_1, t_2, \dots, t_n), \tag{1.2}$$

де t_n — кількість появи і-го терміну в тексті.

Для врахування важливості слів у контексті колекції документів застосовується зважування за схемою TF-IDF. Цей підхід базується на двох компонентах: частота терміна в документі та зворотна частота документа.

Частота терміна в документі. Чим частіше слово зустрічається в тексті, тим воно важливіше для цього тексту.

Зворотна частота документа. Чим у більшій кількості документів корпусу

зустрічається слово, тим менш унікальним і значущим воно є.

Вага терміна обчислюється як добуток цих двох показників (1.3):

$$w_{i,j} = tf_{i,j} \times idf_i, \tag{1.3}$$

де $tf_{i,j}$ — частота терміна і у документі j;

idf_i — обернена документна частота терміна і.

Кожен документ представляється як вектор ваг TF-IDF у просторі термінів. Після такого перетворення подібність між документами можна оцінювати, зокрема, шляхом порівняння векторів еталонної та студентської відповідей і фіксації ключових термінів та базових збігів. Метод вирізняється простою реалізацією, невеликими обчислювальними витратами та придатністю для задач із великими обсягами даних, у яких критичною є саме лексична схожість текстів. Однак недоліки суттєві: модель повністю ігнорує семантику, контекст та синонімію, що призводить до низької точності для відкритих завдань, де різні формулювання можуть передавати той самий зміст. Кореляція з експертними оцінками для таких моделей не перевищує від 0,5 до 0,6 за шкалою Пірсона, що робить їх малоефективними для складних дисциплін, таких як гуманітарні науки чи інженерія, де потрібен аналіз логічних зв’язків.

Для уникнення ділення на нуль та врахування термінів, що є в усіх документах, у практичних реалізаціях використовується згладжування (1.4):

$$idf_i = log(\frac{N+1}{df_i+1}) + 1, \tag{1.4}$$

де N — загальна кількість документів у корпусі;

df_i — кількість документів корпусу, у яких зустрічається термін і.

Більш удосконаленим методом, що вирішує проблему синонімії, є латентно-семантичний аналіз. Він виявляє приховані семантичні структури у тексті, припускаючи, що слова зі схожим значенням зустрічатимуться в подібних

контекстах [9]. Процес перетворення тексту на вектор включає побудову матриці термін-документ, де рядки відповідають термінам, а стовпці — документам. Потім ця матриця розкладається за допомогою сингулярного розкладання для створення скороченого семантичного простору, де тексти представляються як вектори латентних тем (1.5):

$$A = U \Sigma V^T, \tag{1.5}$$

де U — ортогональна матриця лівих сингулярних векторів;

Σ — діагональна матриця сингулярних значень;

V^T — транспонована матриця правих сингулярних векторів.

Для виділення головних семантичних напрямів залишають лише k найбільших компонент, утворюючи зменшену матрицю (1.6):

$$A_k = U_k \Sigma_k V_k^T, \tag{1.6}$$

де U_k — матриця, що містить перші k стовпців матриці U;

Σ_k — діагональна матриця з k найбільших сингулярних значень;

V_k^T — матриця, що містить перші k рядків транспонованої матриці V^T.

Вектори документів у латентному просторі визначаються як (1.7):

$$d_j = \Sigma_k V_{k,j}, \tag{1.7}$$

де $V_{k,j}$ — стовпцевий вектор, що відповідає j-му документу у матриці V_k.

Це змушує модель узагальнювати, в результаті чого синонімічні слова, що вживаються в схожих контекстах, отримують близькі векторні представлення у низькорозмірному семантичному просторі. У цьому форматі стає можливим виявлення синонімії та тематичної спорідненості документів, причому кореляція

між автоматичними й експертними оцінками досягає приблизно 0.8. Однак LSA все ще погано справляється з полісемією і є обчислювально складнішим за TF-IDF.

Революційним кроком у представленні слів стали нейромережеві підходи, зокрема Word2Vec. Ця модель навчається на великих текстових корпусах і генерує для кожного слова щільний вектор фіксованої розмірності, який кодує його семантичні властивості. Існують дві основні архітектури: CBOW і Skip-gram [10].

CBOW (Continuous Bag-of-Words) прогнозує поточне слово на основі його контексту. Skip-gram прогнозує контекст на основі поточного слова. Ця архітектура зазвичай краще працює для рідкісних слів.

Слова, що трапляються в подібних контекстах, отримують близькі вектори, і над такими векторними поданнями можна виконувати операції векторної арифметики. Вектор речення або відповіді зазвичай формується шляхом усереднення векторів слів, що входять до нього (1.8):

$$\vec{s} = \frac{1}{n}\sum_{i=1}^{n} \overrightarrow{v(w_i)}, \tag{1.8}$$

де $v(w_i)$ — вектор і-го слова;

w_i — і-те слово у реченні;

n — кількість слів у реченні.

Цей підхід значно краще відображає семантичні зв’язки, але має два суттєві недоліки. Кожне слово має лише один вектор, що не вирішує проблему полісемії. Усереднення векторів призводить до втрати інформації про порядок слів та синтаксичну структуру речення.

Найвищу на сьогодні точність у задачах розуміння природної мови демонструють контекстуальні ембедінги на основі архітектури трансформерів, найвідомішим представником якої є BERT (Bidirectional Encoder Representations from Transformers). На відміну від попередніх моделей, BERT створює векторні представлення, що враховують повний контекст кожного слова у реченні завдяки механізму уваги. У різних контекстах, наприклад у виразах «старовинний замок на

горі» та «замок на дверях», слово «замок» описується різними векторами, і проблема полісемії фактично усувається. Отримане представлення речення виступає узагальненим вектором усієї відповіді й містить лексичні, синтаксичні та семантичні характеристики тексту. Така форма подання дає змогу кількісно оцінювати глибину змісту й логічну послідовність відповідей, а якість автоматичного оцінювання наближається до експертного рівня [11].

Для систематизації та наочного порівняння розглянутих підходів до векторизації тексту, їхні ключові характеристики зведено у таблицю 1.2.

Таблиця 1.2 — Порівняльний аналіз моделей векторизації тексту

Критерій	Мішок слів / TF-IDF	LSA	Word2Vec	BERT / Трансформери
Основний принцип	Підрахунок частоти слів	Матриця термінів і тем	Навчання нейромережі	Механізм уваги
Врахування контексту	Ні	На рівні документа	Локальне	Глибоке та динамічне
Вирішення синонімії	Ні	Так	Так	Так
Вирішення полісемії	Ні	Ні	Частково	Так
Обчислювальна складність	Низька	Середня	Висока	Дуже висока

1.3.2 Аналіз методів обчислення міри семантичної подібності

Другим ключовим етапом є кількісне порівняння векторів, отриманих на етапі векторизації. Для цього використовуються різні метрики відстані або подібності.

Найпоширенішою метрикою для текстових даних є косинусна подібність, яка вимірює косинус кута між вектором еталонної відповіді та вектором студентської відповіді (1.9). Вона нечутлива до довжини векторів, а лише до їхньої орієнтації у просторі. Отримане значення відображає ступінь узгодженості між двома текстами у семантичному просторі, незалежно від їхньої довжини чи синтаксичної структури. Значення цієї метрики змінюється від -1 до 1, де 1 означає повну семантичну тотожність. Косинусна подібність ефективна для порівняння змістовної схожості незалежно від довжини тексту.

$$Sim_{cos}(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\|\|\vec{b}\|}, \quad (1.9)$$

де $\vec{a}$, $\vec{b}$ — вектори ознак об’єктів, що порівнюються;

$\vec{a} \cdot \vec{b}$ — скалярний добуток векторів a та b;

$\|\vec{a}\|\|\vec{b}\|$ — евклідові норми векторів.

Іншою поширеною метрикою є евклідова відстань, яка вимірює пряму відстань між векторами у просторі ознак (1.10):

$$Dist_E(\vec{a}, \vec{b}) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}, \quad (1.10)$$

де n — розмірність векторів.

Ця міра є корисною для класифікаційних задач, однак чутлива до масштабу та розмірності простору. Також можливим є використання манхеттенської відстані (1.11). Вона ефективна для розріджених векторів, як у «мішку слів», але менш точна для контекстуальних ембедингів.

$$Dist_M(a, b) = \sum_{i=1}^{n}|a_i - b_i| \quad (1.11)$$

Серед сучасних підходів виділяються сіамські нейронні мережі, які навчаються на парах текстів для обчислення семантичної подібності [12]. Процес перетворення тексту на вектор здійснюється через дві ідентичні підмережі, що обробляють еталонну та студентську відповіді, генеруючи векторні представлення на основі шарів нейронної мережі.

Цей підхід характеризується високою адаптивністю до конкретних предметних областей, зокрема технічних дисциплін, а також можливістю тонкого налаштування на анотованих даних, що забезпечує кореляцію з експертними оцінками на рівні від 0,8 до 0,9. Наприклад, у задачах оцінювання коротких відповідей сіамські мережі інтегруються з механізмами уваги для фокусування на

ключових семантичних елементах. Однак недоліки включають потребу в значних обчислювальних ресурсах для навчання, ризик перенавчання на обмежених датасетах та чутливість до шуму в даних, що знижує узагальнюваність моделі.

Точність семантичного оцінювання значною мірою визначається якістю векторного представлення. Зокрема, на результати впливають розмірність простору ознак, глибина попереднього навчання моделі, репрезентативність навчального корпусу та здатність системи враховувати контекстні зв’язки між словами. Чим більш узагальненою є модель і чим ширше охоплення лінгвістичних структур, тим вища адекватність семантичного аналізу.

Порівняно з традиційними або тестовими методами, семантичне оцінювання має суттєву перевагу: воно не обмежується формальною відповідністю ключових слів, а аналізує глибинну логіку відповіді. Це дозволяє системі оцінити, наскільки студент розуміє причинно-наслідкові зв’язки, структуру аргументації та контекст завдання.

1.4 Порівняльний аналіз існуючих програмних рішень і систем-аналогів

Перед розробкою проектованої системи необхідно провести аналіз існуючих платформ для навчання, що дає змогу виокремити ефективні технічні рішення та мінімізувати ризик повторення типових помилок під час впровадження. На основі отриманих результатів можна буде сформувати вимоги до функціональних можливостей і сервісів, які мають бути реалізовані у системі.

Однією з базових платформ є Google Classroom — сервіс, орієнтований на організацію дистанційного навчання та управління навчальними матеріалами. Платформа дозволяє створювати курси, додавати студентів через код класу або шкільний домен, а також організовувати завдання у вигляді тестів та текстових робіт. Google Classroom підтримує базове автоматичне оцінювання тестових завдань та опитувань, але інструменти семантичного аналізу відкритих текстових відповідей залишаються обмеженими, оскільки автоматизовані коментарі формуються переважно за наперед заданими шаблонами чи ключовими словами.

Головна сторінка Google Classroom зображена на рисунку 1.1.

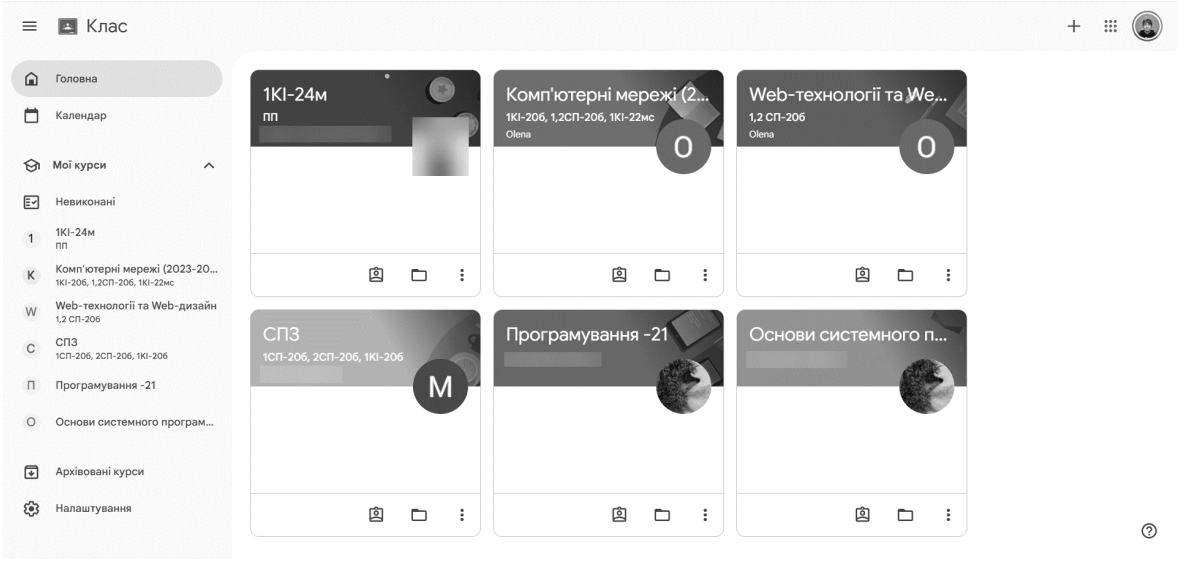

Рисунок 1.1 — Вигляд інтерфейсу Google Classroom

Переваги Google Classroom:

— зручна організація курсів та завдань;

— інтуїтивний інтерфейс для студентів та викладачів;

— інтеграція з іншими сервісами Google;

— базове автоматичне оцінювання тестових завдань.

Недоліки:

— обмежені можливості для автоматичного оцінювання відкритих текстових відповідей;

— відсутність відкритого коду та гнучких інструментів кастомізації;

— залежність від екосистеми Google для роботи з файлами та матеріалами.

Серед відкритих систем управління навчанням окрему увагу привертає платформа Moodle, яка підтримує комплексне створення курсів та інтеграцію додаткових модулів для автоматизації оцінювання. У Moodle можуть організовуватися різні типи завдань: тестові, текстові роботи, есе та проєктні завдання. Для автоматичної перевірки відкритих текстів використовується інтеграція з плагінами й сервісами, що виконують аналіз змісту, зокрема перевірку ключових слів, тематичних збігів та прості варіанти семантичного порівняння.

Система підтримує гнучке налаштування ваг оцінювання та правил автоматичного нарахування балів. Вигляд інтерфейсу Moodle наведено на рисунку 1.2.

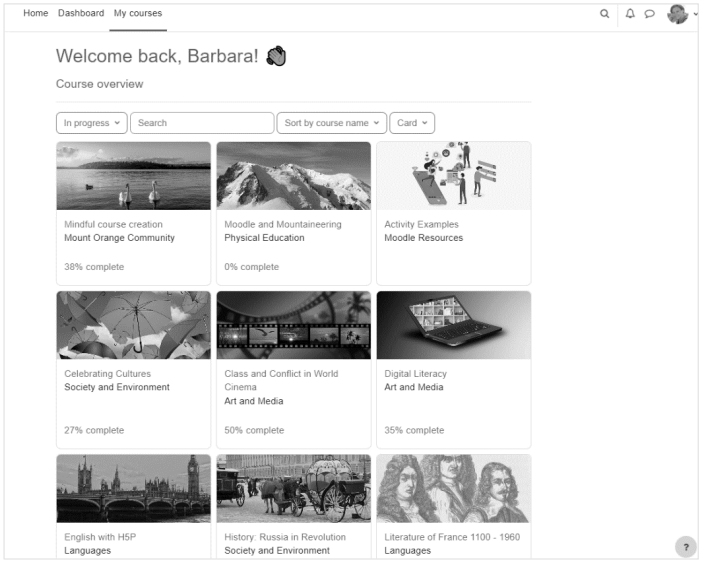

Рисунок 1.2 — Вигляд інтерфейсу Moodle

Переваги Moodle:

— відкритий код, що забезпечує можливість інтеграції сторонніх модулів;

— гнучкі налаштування завдань та системи оцінювання;

— підтримка різноманітних типів навчальних матеріалів та завдань;

— можливість автоматичного оцінювання текстових та тестових завдань через модулі.

Недоліки:

— складність початкового налаштування та адміністрування;

— не завжди інтуїтивний інтерфейс для студентів;

— необхідність додаткових модулів для семантичного оцінювання відкритих завдань.

Наступним аналогом розглядається система JetIQ — електронна система автоматизованого управління даними освітнього процесу та електронного документообігу Вінницького національного технічного університету. JetIQ дозволяє переглядати розклад занять, отримувати доступ до навчальних матеріалів за дисциплінами та контролювати успішність через електронні журнали і відомості.

Також система підтримує тестування з автоматичним підрахунком результатів, обмін повідомленнями та надсилання файлів викладачу. Вигляд головної сторінки JetIQ наведено на рисунку 1.3.

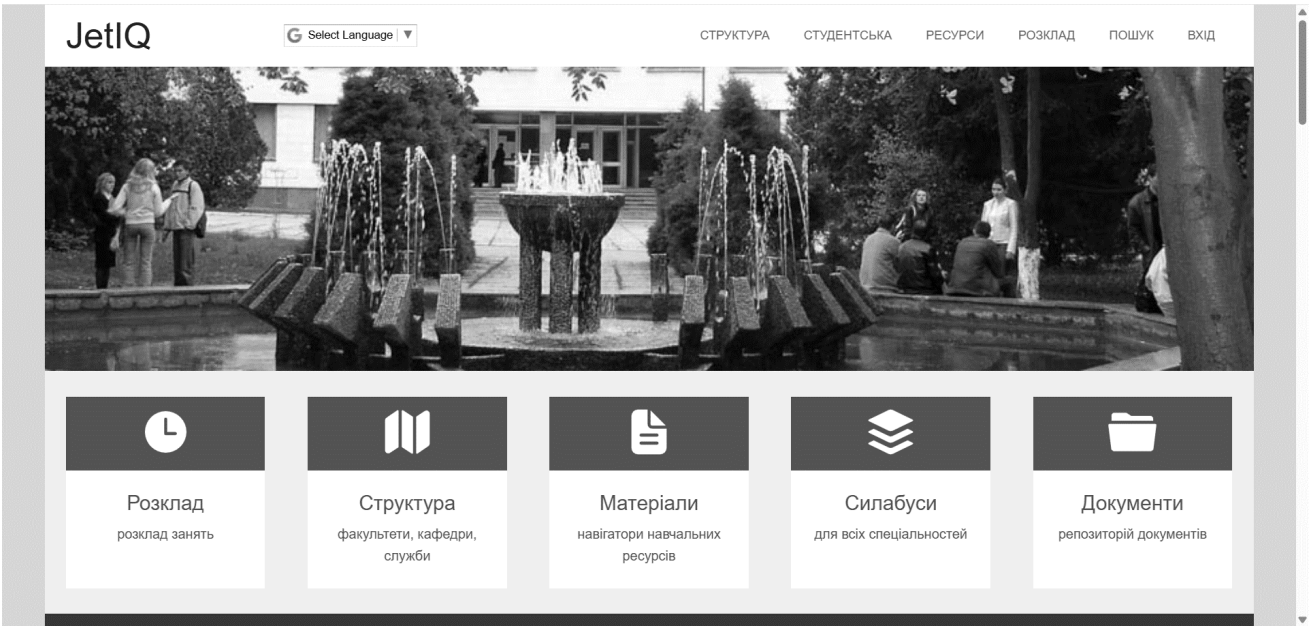

Рисунок 1.3 — Головна сторінка JetIQ

Переваги JetIQ:

— сильна інтеграція з навчальним закладом;

— доступ до навчальних матеріалів за дисциплінами;

— електронні інструменти контролю та перегляду успішності;

— детальний інтегрований розклад занять;

— підтримка тестового контролю знань із автоматичним отриманням результату та його фіксацією в електронному журналі;

— наявність вбудованих засобів комунікації та надсилання файлів викладачу.

Недоліки:

— орієнтація на внутрішню інфраструктуру ВНТУ, що обмежує універсальність;

— складна навігація та інтерфейс до яких необхідно звикнути.

Порівняльний аналіз платформ наведено у таблиці 1.3.

Таблиця 1.3 — Порівняльний аналіз аналогів з розроблюваною системою.

Характеристика	Google Classroom	Moodle	JetIQ	Система, що розробляється
Підтримка автоматичного оцінювання відкритих завдань	-	+	-	+
Навчальна аналітика	-	+	+	+
Масштабованість курсу та користувачів	+	+	+	+
Модульність і можливість розвитку	-	+	+	+
Зручність інтерфейсу для студентів	+	-	-	+
Простота налаштування для викладача	+	-	-	+

Таким чином, основні переваги проєктованої системи полягають у здатності автоматично оцінювати відкриті текстові завдання на основі семантичного порівняння, зручному інтерфейсі для студентів і викладачів, та підтримці навчальної аналітики для відстеження прогресу та результатів навчання, що разом забезпечує підвищення об’єктивності оцінювання та ефективності організації навчальної діяльності порівняно з існуючими платформами.

2 МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ СИСТЕМИ ПЛАНУВАННЯ ТА ОЦІНЮВАННЯ НАВЧАЛЬНОЇ ДІЯЛЬНОСТІ СТУДЕНТА

2.1 Побудова структури системи планування та оцінювання навчальної діяльності студента

Структура комп'ютерної системи планування та оцінювання навчальної діяльності визначається необхідністю забезпечення єдиного циклу взаємодії між викладачем і студентом у процесі виконання та контролю навчальних завдань. Її логічна побудова спрямована на узгоджене функціонування всіх елементів, що забезпечують збереження, оброблення, оцінювання та відображення навчальної інформації у формі, зручній для користувачів різних рівнів доступу. Система реалізує повний цикл управління навчальними завданнями від створення й розподілу до оцінювання результатів і формування підсумкової статистики.

Система має забезпечувати автоматизоване керування навчальними завданнями, об'єктивне оцінювання результатів студентів та формування зворотного зв'язку у зручному інтерфейсі. Система повинна забезпечувати стабільну роботу при збільшенні кількості користувачів, підтримувати точність обчислень і гарантувати узгодженість результатів на всіх етапах навчального процесу. З огляду на поставлені цілі визначимо основні функціональні вимоги до системи:

— забезпечення авторизації користувачів із розмежуванням ролей «студент» та «викладач»;

— можливість створення, редагування та видалення навчальних завдань;

— прийом та збереження відповідей студентів у єдиній базі даних із фіксацією часу здачі;

— автоматизоване оцінювання результатів;

— відображення результатів у вигляді таблиць або індивідуальних звітів;

— надання викладачу можливості перегляду та корекції оцінок;

— формування статистики успішності та зведених показників якості навчання.

Для системи також важливі нефункціональні вимоги, які визначають її поведінку в реальних умовах використання. До таких вимог належать надійність збереження даних, швидкість реакції на дії користувача, зручність роботи з інтерфейсом, захист навчальної інформації та здатність системи до подальшого розвитку.

Надійність означає, що створені завдання, збережені відповіді та виставлені оцінки не втрачаються через нестабільне з’єднання або збої на стороні клієнтських пристроїв. Операції створення завдання, надсилання відповіді і фіксації оцінки мають виконуватися як завершені дії. Якщо запис у базу даних не завершився успішно, попередній стан інформації зберігається без змін, а користувач отримує повідомлення про помилку. Для основних об’єктів варто передбачити регулярне резервне копіювання, яке дає змогу відновити роботу після відмови сервера без втрати відомостей про хід навчального процесу.

Швидкодія характеризує час, за який система виконує типові операції. Списки завдань, результати студентів і окремі роботи мають відкриватися без відчутних затримок навіть за умови роботи з великою групою. Під час автоматичного оцінювання користувач повинен бачити, що обробка відповіді триває, і мати змогу продовжувати роботу з іншими елементами інтерфейсу. Система має коректно працювати при одночасному зверненні багатьох студентів та викладачів в межах одного курсу.

Зручність використання стосується як студентського, так і викладацького інтерфейсу. Студент повинен мати впорядкований перелік поточних, майбутніх та виконаних завдань із зазначенням курсу, теми, граничної дати здачі та стану виконання. Викладачеві потрібні зручні засоби переходу від загального огляду групи до перегляду конкретної відповіді, її оцінки та пов’язаних записів. Підписи полів, елементи керування і текст системних повідомлень мають бути зрозумілими та відповідати прийнятій у навчальному процесі термінології.

Захист даних передбачає чітке розмежування прав доступу і безпечне передавання інформації між клієнтом і сервером. Студент бачить лише власні завдання та результати. Викладач працює з даними своїх курсів і не має доступу до

інших дисциплін без відповідних повноважень. Ручне коригування оцінки виконується лише в інтерфейсі викладача, а студент не має технічної можливості змінювати виставлений результат.

Масштабованість системи означає, наскільки легко вона може пристосовуватися до нових вимог без суттєвої зміни вже реалізованої логіки. Структура системи повинна дозволяти додавання нових типів завдань, підключення альтернативних моделей оцінювання або інтеграцію з іншими освітніми платформами без повної переробки вже реалізованих компонентів. Це спрощує підтримку та дає можливість поступово розширювати функціонал системи.

Сформульовані вимоги дозволяють спроєктувати структуру системи так, щоб кожен її елемент мав чітко визначену роль у загальному циклі роботи. Архітектура системи базується на принципах функціональної автономності модулів і централізованого управління потоками даних. Кожен модуль виконує визначену функцію, а їх взаємодія відбувається через стандартизовані інтерфейси обміну даними. У такій побудові окремі компоненти можуть змінюватися або модернізуватися без впливу на загальну логіку роботи, що важливо для довгострокової експлуатації системи. Внаслідок такої організації досягається цілісність інформаційних потоків і узгодженість усіх етапів роботи системи. Логічна структура системи представлена як послідовність взаємопов’язаних етапів оброблення навчальної інформації: створення завдань, їх розподіл між студентами, формування та надсилання відповідей, оцінювання результатів і збереження статистики успішності. Визначимо основні компоненти системи.

Загальну структурну схему системи подано на рисунку 2.1. Вона має трирівневу будову і включає користувацький інтерфейс у вигляді мобільного застосунку, серверну частину та базу даних. Користувач працює з системою через мобільний застосунок, який відповідає за відображення навчальної інформації, введення даних і надсилання відповідей на завдання. Застосунок реалізує два основні інтерфейси, що відрізняються складом доступних операцій, але користуються спільними сервісами серверної частини.

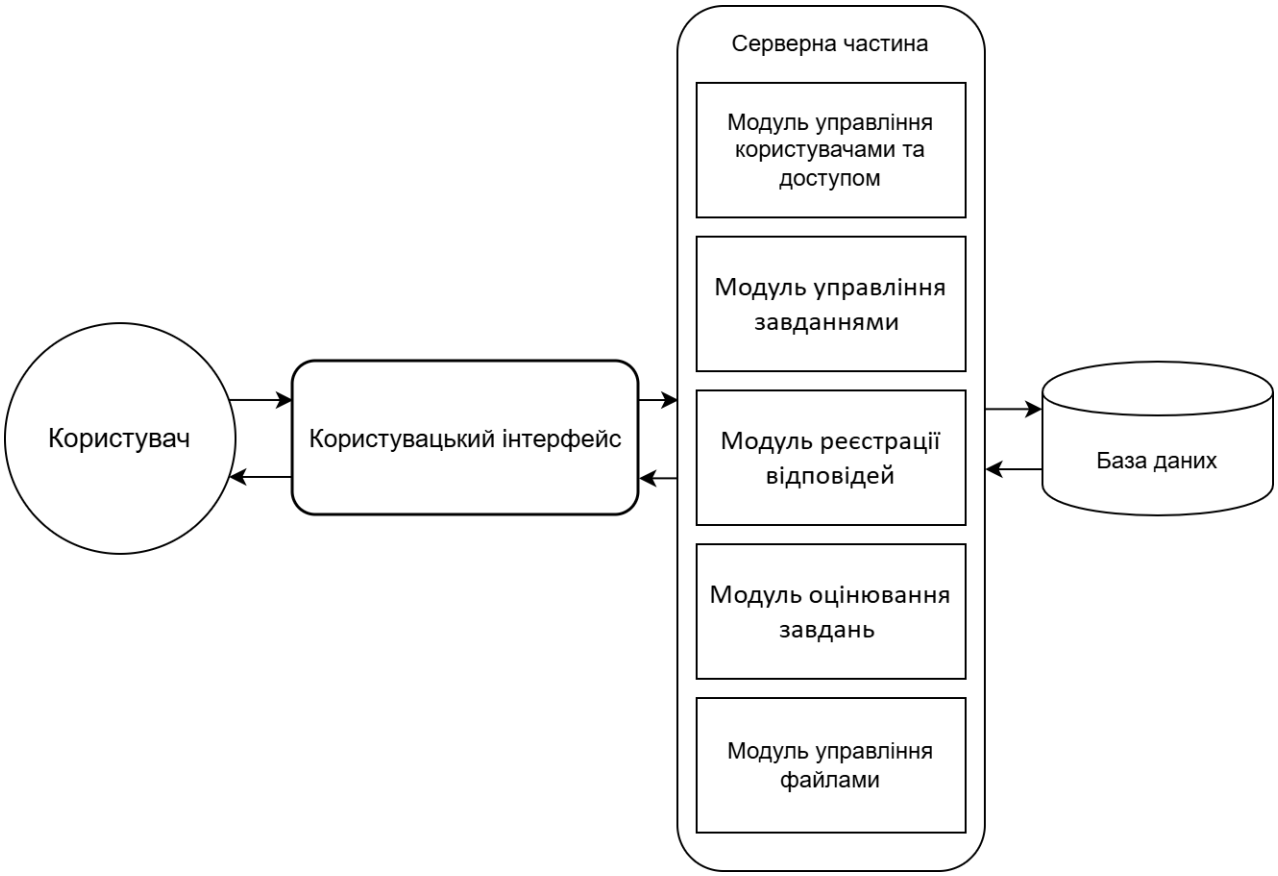

Рисунок 2.1 — Структура системи

На серверному рівні логіка системи поділена на п’ять взаємопов’язаних функціональних модулі: модуль управління користувачами та доступом, модуль управління завданнями, модуль реєстрації відповідей, модуль оцінювання завдань, а також модуль управління файлами. Обрана структуризація модулів формує чіткий розподіл відповідальності між підсистемами та спрощує подальший розвиток окремих компонентів без порушення роботи всієї системи.

Модуль управління користувачами та доступом відповідає за формування й підтримку облікових записів, збереження персональних даних та розмежування прав доступу. У ньому зберігаються відомості про студентів і викладачів, пов’язані з ними ролі, а також налаштування, що визначають, які операції дозволено виконувати конкретному користувачеві. Усі інші модулі серверної частини звертаються до цього модуля для перевірки прав доступу перед виконанням дій, пов’язаних зі створенням завдань, надсиланням відповідей або переглядом результатів.

Модуль управління завданнями реалізує повний життєвий цикл навчальних завдань. У його межах зберігаються структури курсів, їх прив’язка до викладачів, текстові формулювання завдань, терміни виконання та параметри штрафу за запізнення. Модуль формує перелік активних завдань для кожного студента, забезпечує їх публікацію, зміну статусу та закриття після завершення перевірки. Під час створення завдання модуль реєструє посилання на еталонний матеріал, отримане від модуля управління файлами, і пов’язує його з відповідним записом у базі даних.

Модуль реєстрації відповідей забезпечує реєстрацію виконання завдань студентами. Він приймає від користувацького інтерфейсу відомості про обране завдання, користувача та час відправлення, створює відповідний запис у базі даних і формує логічну структуру відповіді. У межах цього модуля фіксується факт відправки відповіді, поточний статус перевірки та зв’язок із прикріпленими файлами, які зберігаються у файловому сховищі.

Модуль оцінювання завдань виконує порівняння поданих відповідей з метою визначення їх відповідності еталонним розв’язкам. Він отримує з бази даних відомості про відповідь студента, параметри відповідного завдання та посилання на еталонний матеріал, після чого застосовує алгоритм оцінювання на основі порівняння змісту файлів. З урахуванням часу відправки модуль здійснює корекцію балів згідно з налаштованим штрафом за запізнення та зберігає результат у вигляді автоматичної оцінки. У разі потреби система дозволяє викладачу вручну скоригувати оцінку.

Модуль управління файлами забезпечує централізовану роботу з усіма документами, що супроводжують навчальні завдання та відповіді студентів. У його завдання входить приймання файлів з користувацького інтерфейсу, перевірка допустимих форматів і розмірів, розміщення матеріалів у файловому сховищі та формування стійких посилань для подальшого використання. Модуль надає уніфікований інтерфейс для інших компонентів серверної частини. Модуль управління завданнями використовує його для збереження еталонних рішень, модуль реєстрації відповідей використовує цей інтерфейс для реєстрації комплекту

файлів студентської роботи, а модуль оцінювання застосовує його для доступу до потрібних документів під час аналізу.

Центральна база даних об’єднує результати роботи всіх модулів. У ній зберігаються відомості про користувачів, курси, завдання, відповіді та пов’язані з ними файли. Для кожного завдання фіксується його зв’язок з курсом і викладачем, а окремі записи результатів містять інформацію про отримані відповіді із зазначенням часу надсилання, значення автоматичної оцінки і підсумкового бала. Така організація даних дає можливість відстежити історію роботи із завданнями і на цій основі формувати зведені показники успішності для студентів та навчальних груп.

Взаємодія між компонентами системи відбувається за принципом послідовної передачі інформації між основними процесами:

— створення та публікація завдань викладачем;
— отримання завдань студентом і надсилання відповіді;
— автоматизована обробка отриманих рішень;
— формування підсумкових оцінок і статистики.

2.2 Розробка моделі оцінювання завдань

Процес оцінювання результатів навчальної діяльності в системі є процедурою визначення ступеня семантичної подібності між еталонною відповіддю викладача та фактичною відповіддю студента з подальшим перетворенням отриманого показника у бальну шкалу. Метою побудови моделі є розробка такого алгоритму, який забезпечує відтворювану та об’єктивну перевірку відкритих завдань без безпосередньої участі експерта, з урахуванням змісту відповіді та дотримання встановлених термінів здачі.

У моделі оцінювання текст еталонної відповіді та текст відповіді студента перетворюються на числові вектори фіксованої розмірності. Кожному тексту ставиться у відповідність набір компонентів, що відображає використану лексику, контекстні зв’язки та типові поєднання слів. Утворені вектори розглядаються як

точки в багатовимірному просторі, де близькі за змістом тексти розташовані поблизу один одного. Для визначення ступеня відповідності між еталонною та студентською відповідями використовується косинусна подібність, яка обчислюється за кутом між відповідними векторами і не залежить від їхньої довжини. Завдяки цьому відповіді, у яких одна й та сама ідея викладена різним обсягом тексту, дають близькі значення показника, якщо збережено основні змістові елементи. Коли у відповіді відтворено ключові елементи еталонного розв'язку, орієнтація векторів майже збігається і значення показника наближається до одиниці. Якщо текст містить суттєві змістові відхилення або стосується іншої тематики, вектори орієнтовані під більшим кутом і значення косинусної подібності істотно зменшується.

На етапі попередньої обробки еталонний матеріал і відповідь студента отримуються з прикріплених файлів і перетворюються на текстові відображення. Якщо відповідь представлена у вигляді кількох файлів, їхній вміст об'єднується в один текстовий блок. Потім виконується очищення від службових символів, нормалізація регістру, та за потреби вилучаються дубльовані фрагменти або технічна інформація, що не несе змістового навантаження. На основі підготовленого тексту формується векторне представлення за допомогою моделі ембедингів, описаної в розділі 1.3. Нормований показник семантичної подібності визначається виразом (2.1):

$$S = \frac{1}{2}\left(1 + \frac{\overrightarrow{r_s} \cdot \overrightarrow{r_e}}{\|\overrightarrow{r_s}\| \|\overrightarrow{r_e}\|}\right), \tag{2.1}$$

де $\overrightarrow{r_s}$ — векторне представлення відповіді студента;

$\overrightarrow{r_e}$ — векторне представлення еталонної відповіді.

Таке нормування переносить результат косинусної подібності, що змінюється від мінус одиниці до одиниці, у проміжок від нуля до одиниці. Значення, близькі до верхньої межі цього проміжку, відповідають високій

змістовній близькості до еталона, а значення, близькі до нуля, свідчать про низький рівень відповідності поставленому завданню.

На концептуальному рівні в моделі враховуються такі аспекти якості відповіді, як точність відтворення ключових положень, повнота охоплення змісту, логічна послідовність викладення та відповідність контексту. Точність відображає збіг основних тверджень, визначень, формул і висновків з еталонним розв’язанням. Повнота характеризує охоплення усіх суттєвих елементів, передбачених завданням, включаючи проміжні кроки та обґрунтування. Логічна послідовність стосується внутрішньої узгодженості міркувань і коректності переходів між частинами відповіді. Відповідність контексту означає зосередженість саме на поставленій проблемі без зайвої інформації. У реалізованому алгоритмі ці характеристики не виділяються як окремі часткові показники з власними вагами, їхній внесок інтегрально відображається в нормованому значенні семантичної подібності завдяки властивостям простору ембедингів.

Для аналізу стабільності роботи алгоритму вводиться показник похибки оцінювання, який визначається як різниця між фактичним значенням нормованої семантичної подібності та ідеальним значенням, що відповідає повному збігу з еталоном (2.2).

$$\varepsilon = |S - S_{\text{ет}}|, \tag{2.2}$$

де S — фактичне отримане значення;

S_{em} — ідеальне значення при повному збігу з еталоном.

Цей параметр дає змогу оцінити, наскільки одержані результати наближені до теоретично очікуваних, і використовується для подальшого аналізу якості налаштування моделі.

Додатково модель враховує час відправки відповіді на завдання. Для кожного завдання встановлюється крайній термін здачі, а система фіксує фактичний час відправки. Якщо робота здана із запізненням, підсумковий бал зменшується згідно зі штрафним коефіцієнтом часу (2.3).

$$k_t = \begin{cases} 1, & T_s \leq T_d \\ 1 - \delta, & T_s > T_d \end{cases}, \quad (2.3)$$

де δ — коефіцієнт зниження оцінки за запізнення;

T_S — час надсилання відповіді;

T_d — встановлений граничний термін здачі.

Значення цього параметра визначається політикою курсу і може бути рівним нулю, якщо штрафи не передбачені. Урахування часової корекції дозволяє підвищити дисципліну виконання завдань без втручання викладача в процес оцінювання.

Після урахування семантичної складової та часової корекції отриманий результат переводиться у прийняту шкалу оцінювання за допомогою функції нормування (2.4):

$$O = O_{min} + (O_{max} - O_{min}) * S * k_t, \quad (2.4)$$

де O_{min} і O_{max} — межі шкали оцінювання.

При S = 1 студент отримує максимальний бал, а при S = 0 — мінімальний.

Загальний процес оцінювання можна подати як композицію трьох послідовних операторів, які відповідають за обчислення семантичної подібності, урахування своєчасності здачі та нормування до бальної шкали (2.5):

$$S_n = N(M(R_s, R_e), k_t), \quad (2.5)$$

де N — оператор нормування результатів до єдиної шкали;

M — оператор обчислення подібності;

k_t — коефіцієнт урахування своєчасності здачі відповіді.

Оператор M реалізується на основі алгоритмів семантичного аналізу, що забезпечують оцінку змістовної близькості текстів, оператор N виконує математичне перетворення отриманого результату у шкалу, зрозумілу користувачу.

Модель забезпечує цілісний підхід до автоматизованого оцінювання, у якому враховуються як змістовні, так і часові характеристики виконання завдання. Це дозволяє отримувати стабільні, порівнянні та об’єктивні результати незалежно від кількості перевірених робіт.

Отримане значення оцінювання може бути використане не лише для виставлення оцінки, а й для діагностики рівня компетентності студента. Для інтерпретації результатів виділяються порогові зони, наведені в таблиці 2.1.

Таблиця 2.1 — Інтерпретація отриманих балів до рівня засвоєння матеріалу

Інтервал	Рівень засвоєння	Інтерпретація
0.85–1.00	Високий	Повне засвоєння матеріалу
0.65–0.84	Достатній	Основні знання засвоєні, незначні неточності
0.45–0.64	Середній	Часткове розуміння, необхідна корекція
0.25–0.44	Низький	Поверхове знання, слабка аргументація
0–0.24	Незадовільний	Завдання не виконано або виконано неправильно

Побудована аналітична модель дозволяє не лише формально оцінювати результати, а й забезпечує основу для подальшої адаптації навчального процесу, наприклад для автоматичного підбору завдань відповідно до виявленого рівня компетентності.

2.3 Моделювання процесу обліку результатів навчання

У системі облік результатів навчання описується як послідовність операцій фіксації відомостей про виконання студентами навчальних завдань і подальшої перевірки цих результатів. У моделі фіксуються зв’язки між завданням, відповіддю студента та отриманою оцінкою, а також додаткова інформація про час відправки відповіді і спосіб формування бала, щоб можна було відокремити автоматичне оцінювання від підсумкового рішення викладача.

У центрі моделі обліку знаходиться зв’язок між навчальним завданням і конкретним студентом. Для кожного завдання формується окремий запис про виконання, який описує поточний стан роботи. У цьому записі зберігаються

ідентифікатори студента та завдання, час останнього надсилання відповіді, результати автоматичного оцінювання, наявність штрафу за запізнення, а також підсумковий бал, якщо його вже підтверджено викладачем.

У початковий момент завдання існує лише як елемент навчального плану і не має жодних пов’язаних результатів. Після створення викладачем воно містить формулювання, термін здачі та параметр штрафу за запізнення, який задається як відсоток зниження оцінки. До моменту активації завдання результати обліку для нього відсутні, а після публікації воно стає доступним студентам у списку призначених робіт.

Під час здачі відповіді студентом формується запис про виконання завдання. У ньому фіксуються ідентифікатор студента, ідентифікатор завдання, час здачі та поточний стан перевірки. Відповідь подається у вигляді одного або кількох файлів, що надалі використовуються для аналізу змісту. Після реєстрації факту здачі запускається процедура автоматичного оцінювання, а створений запис переходить у стан очікування результату.

Механізм автоматичного оцінювання представлено як окремий етап у процесі обліку результатів. Після реєстрації здачі система послідовно виконує три основні кроки. На початковому етапі здійснюється підготовка текстового представлення еталонного матеріалу та відповіді студента. Текстовий вміст вилучається з прикріплених файлів і приводиться до єдиного формату представлення. Далі виконується порівняння змісту еталонної відповіді та студентської роботи на основі векторного представлення тексту і обчислюється показник їх подібності. На завершальному кроці цей показник перетворюється у значення оцінки у прийнятій шкалі, додатково враховується факт відправки після кінцевого терміну здачі, та за наявності відповідного параметра завдання застосовується зниження бала.

Автоматичне оцінювання формує числовий результат, який зберігається разом із допоміжною службовою інформацією. Стан відповіді змінюється і позначається як такий, що очікує перевірки викладачем. На цьому етапі відповідь уже має автоматичну оцінку, але ще не вважається остаточно перевіреною.

Подальший етап стосується дій викладача. Для кожного завдання формується перелік зданих робіт із зазначенням студента, часу здачі та автоматичної оцінки. Після ознайомлення зі змістом файлів і результатом автоматичного оцінювання викладач може прийняти розрахований бал або змінити його, спираючись на власний аналіз відповіді.

У моделі обліку результатів навчання остаточна оцінка фіксується разом із часом її встановлення. Якщо викладач погоджується з автоматичним результатом, у записі фіксується, що підсумковий бал сформовано за результатами автоматичного оцінювання. Якщо викладач коригує значення, підсумковий бал набуває статусу встановленого вручну. Після збереження остаточного бала робота набуває статусу перевіреної, і цей стан використовується всіма компонентами системи для відображення результатів.

Блок-схема послідовності основних етапів автоматизованого обліку результатів наведена в додатку В. Спочатку викладач створює завдання, формує текст умови, задає кінцевий термін здачі, параметр штрафу та прикріплює еталонний файл. Після публікації студент отримує завдання у власному списку, готує відповідь і надсилає файли. Система реєструє відповідь, готує текстове представлення еталонного матеріалу та студентської роботи, обчислює міру їх подібності, застосовує зниження за запізнення і формує автоматичну оцінку. На наступному кроці викладач переглядає результат, ухвалює або коригує підсумковий бал, після чого відповідь остаточно вважається перевіреною, а інформація про оцінку стає доступною для студента.

2.4 Розробка алгоритму роботи програми

Алгоритм роботи системи планування та оцінювання навчальної діяльності визначає послідовність переходів між екранами мобільного застосунку та зміну інформаційного стану для користувача. Опис алгоритму потрібний для узгодження поведінки застосунку у різних сценаріях використання та для коректного відображення стану навчальних завдань і результатів їх виконання для ролей

викладача і студента.

У загальному вигляді робота програми поділяється на кілька логічних етапів:

— автентифікація користувача і визначення ролі;

— робота викладача з навчальними завданнями;

— робота студента з призначеними завданнями;

— автоматизоване оцінювання надісланих відповідей;

— оновлення записів про результати та відображення актуального стану в інтерфейсі.

Перший етап алгоритму пов’язаний із входом користувача до системи. Після запуску мобільного застосунку відображається форма автентифікації, де користувач вводить свої облікові дані. Під час реєстрації додатково задається роль студента чи викладача, яка зберігається у профілі користувача і надалі використовується для вибору гілки подальшої роботи. Після успішної перевірки облікових даних виконується розгалуження за роллю. Для викладача відкривається інтерфейс керування завданнями та перевіркою результатів. Для студента формується інтерфейс роботи із власним списком завдань і оцінок.

Алгоритм роботи викладача спрямований на створення, коригування і перевірку навчальних завдань. Після входу до системи стартовим є екран, де наведено перелік створених завдань із короткою інформацією про них. На цьому етапі реалізуються такі основні дії:

— створення нового завдання із зазначенням назви, тексту умови, дати здачі та параметрів оцінювання;

— редагування або актуалізація вже створених завдань;

— переведення завдання в активний стан, після чого воно стає доступним студентам;

— перегляд списку надісланих студентами відповідей;

— аналіз автоматично сформованих оцінок, їх підтвердження або коригування.

У процесі створення нового завдання викладач задає всі необхідні дані для завдання. Вказані дані зберігаються у модулі управління завданнями і надалі використовуються модулем оцінювання під час обчислення підсумкового бала. Після активації завдання система розповсюджує його серед відповідних студентів, а запис про завдання переходить у стан, який дозволяє приймати студентські відповіді.

Алгоритм роботи студента починається із завантаження списку актуальних завдань після успішного входу до системи. Студент отримує доступ до переліку призначених завдань із зазначенням основної інформації про них. Подальший сценарій містить такі основні кроки:

— вибір завдання зі списку та перегляд повного формулювання умови;

— підготовка та прикріплення матеріалів завдання;

— надсилання готової відповіді й підтвердження наміру здати роботу;

— очікування результату оцінювання та перегляд поточного статусу виконання;

— перегляд виставленої оцінки після її підтвердження викладачем;

— перегляд статистики по всім завданням.

До моменту, поки для конкретного завдання не зафіксовано остаточну оцінку, студент має право редагувати свою відповідь і повторно надсилати її на перевірку. Це допоможе уникнути проблем із помилковим відправленням робіт.

Ключовою частиною алгоритму є автоматизоване оцінювання відправлених студентами відповідей. Після надсилання чергової відповіді модуль управління завданнями фіксує факт здачі роботи та час отримання файлів. Далі модуль оцінювання завантажує еталонну відповідь і виконує обчислення узагальненого показника подібності студентської роботи до еталону. Якщо в політиці завдання передбачено штраф за запізнення, він застосовується відповідно до зафіксованого часу відправки відповіді відносно терміну здачі. Отриманий результат перетворюється у підсумкову оцінку за обраною шкалою оцінювання. Усі проміжні й підсумкові значення зберігаються в модулі обліку результатів. Після розрахунку

оцінки відповідь отримує статус «автоматично оцінено, очікує підтвердження викладача» і стає доступною в інтерфейсі перевірки для подальшого розгляду викладачем. Структура і функціонал сторінок системи наведено на рисунку 2.2.

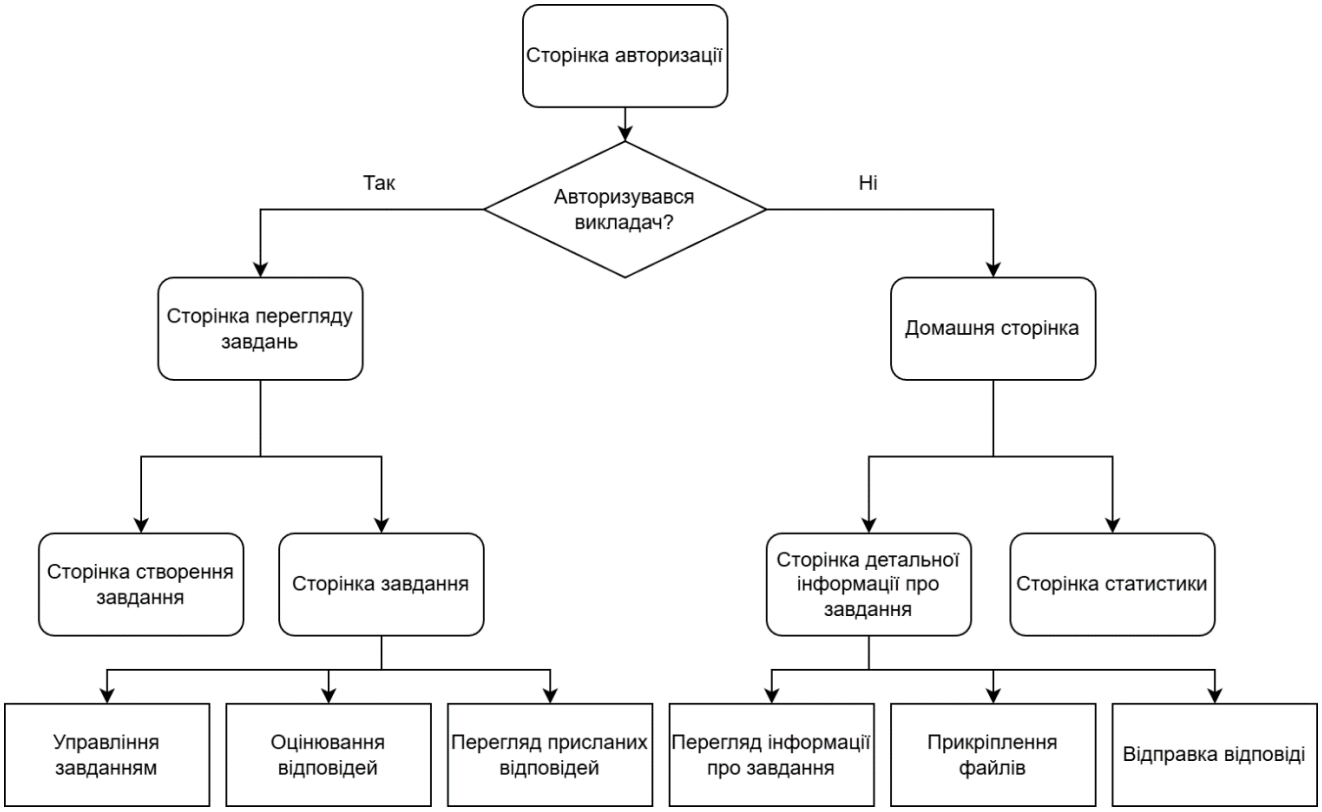

Рисунок 2.2 — Блок-схема функціональних можливостей сторінок системи

3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ПЛАНУВАННЯ ТА ОЦІНЮВАННЯ НАВЧАЛЬНОЇ ДІЯЛЬНОСТІ СТУДЕНТА

3.1 Вибір засобів розробки системи

3.1.1 Вибір інструментів розробки клієнтської частини

Клієнтська частина системи планування та оцінювання навчальної діяльності призначена для щоденного використання студентами та викладачами на смартфонах. Формат мобільного застосунку відповідає сучасній організації освітнього процесу, коли основні дії з навчальними завданнями, розкладом та результатами виконуються саме з мобільних пристроїв. До технологій клієнтської частини висуваються вимоги стабільної роботи, плавної взаємодії з інтерфейсом, можливості використання на широкому спектрі пристроїв та відсутності залежності від високопродуктивного апаратного забезпечення. Додатковою умовою є кросплатформеність, оскільки цільова аудиторія користується як Android, так і iOS.

Для розробки мобільного клієнта можна виокремити три основні підходи. Перший підхід передбачає нативну розробку для кожної платформи окремо з використанням рекомендованих мов програмування та стандартних наборів засобів створення програм. Другий підхід спирається на гібридні рішення, у яких інтерфейс реалізується у вигляді вебсторінки у вбудованому переглядачі. Третій підхід представлений кросплатформеними фреймворками нового покоління, де формується спільна кодова база для кількох платформ, а інтерфейс відтворюється через нативні компоненти або власний графічний рушій [13].

Нативна розробка забезпечує найтіснішу інтеграцію застосунку з операційною системою. Застосунок отримує повний доступ до системних прикладних інтерфейсів, апаратних ресурсів та стандартних елементів користувацького інтерфейсу. Нативна розробка має найвищу продуктивність і відповідає рекомендаціям розробників платформ, проте потребує підтримки окремих кодових баз для Android і iOS. Це збільшує витрати на розробку, тестування та супровід і подовжує час упровадження змін, що є суттєвим

обмеженням для навчальних систем з обмеженими ресурсами [14].

Гібридні фреймворки, зокрема Apache Cordova та Ionic, реалізують інтерфейс як вебсторінку у вбудованому браузерному компоненті операційної системи. У такому разі використовуються добре відомі вебтехнології, прискорюється створення прототипів і підтримується єдина кодова база для кількох платформ. Водночас інтерфейс часто поступається нативним аналогам за плавністю анімацій та швидкодією, а поведінка елементів може відрізнятися на різних платформах. Для навчальної системи, де важливими є передбачуваність інтерфейсу та стабільність роботи під час активної взаємодії з формами та списками, такі обмеження є критичними.

Кросплатформені фреймворки нового покоління поєднують переваги спільної кодової бази з використанням нативних компонентів інтерфейсу або власного високопродуктивного рушія рендерингу. До цієї групи належать React Native та Flutter, які орієнтовані на досягнення продуктивності, близької до нативної, за знижених витрат на розроблення і супровід. У порівнянні з нативним підходом такі рішення спрощують підтримку логіки, а порівняно з гібридними технологіями забезпечують природнішу взаємодію з інтерфейсом та кращу швидкодію на мобільних пристроях.

React Native базується на бібліотеці React і використовує декларативний опис користувацького інтерфейсу засобами JavaScript або TypeScript. Логіка застосунку виконується у власному середовищі JavaScript, а структура екранів задається у вигляді дерева компонентів. Під час виконання ця структура синхронізується з нативною підсистемою відображення, і елементи React Native перетворюються на стандартні елементи інтерфейсу Android та iOS [15]. Завдяки такому підходу зовнішній вигляд і поведінка застосунку повністю відповідають вимогам кожної платформи, а більша частина програмного коду залишається спільною для всіх мобільних пристроїв.

Архітектура React Native спирається на розділення середовища виконання JavaScript та нативної частини. Код, який описує логіку і стан інтерфейсу, виконується у окремому потоці, тоді як операції відображення і доступ до

системних ресурсів виконуються у нативних потоках Android та iOS. Обмін даними між цими частинами організовано через інтерфейс JavaScript Interface, який дає змогу викликати нативні функції без трудомісткої серіалізації даних і підтримує посилання на об’єкти у спільній пам’яті. У новій архітектурі застосовуються механізми Fabric та TurboModules, що зменшують затримки під час рендерингу інтерфейсу і прискорюють завантаження модулів, тому чутливість інтерфейсу до дій користувача зростає навіть на пристроях середнього рівня.

Компонентний підхід у React Native дає змогу розглядати кожен елемент інтерфейсу як самостійний будівельний блок. Базові компоненти мають відповідні аналоги у нативних елементах Android та iOS, тому під час відображення створюються повноцінні нативні представлення. Інтерфейс розбивається на ієрархію багаторазово придатних компонентів, кожен з яких поєднує у собі структуру відображення, локальний стан та логіку оброблення подій. Це спрощує розроблення складних інтерфейсів, дає змогу розділяти відповідальність між окремими частинами коду та полегшує подальшу підтримку. Розміщення елементів на екрані переважно реалізується за допомогою моделі Flexbox, завдяки чому забезпечується коректне розміщення компонентів на екранах з різними розмірами і пропорціями.

Серед ключових переваг React Native виокремлюються повторне використання коду та розвинена екосистема бібліотек. Значна частина бізнес логіки і компонентів інтерфейсу реалізується один раз і застосовується як для Android, так і для iOS, що знижує витрати на розробку, тестування і супровід. Оскільки компоненти React Native спираються на нативні елементи, застосунки зберігають звичний для користувача вигляд і швидкість реакції. Наявність великої спільноти розробників і широкого набору готових бібліотек для навігації, керування станом, роботи з формами та мережевими запитами прискорює реалізацію типових сценаріїв роботи програм.

Важливу роль у межах екосистеми React Native відіграє платформа Expo. Вона надає готове середовище для розробки, тестування і розгортання мобільних застосунків, механізми інтерактивного перегляду результатів на реальних

пристроях без додаткової ручної конфігурації проєкту та містить набір бібліотек для роботи з камерою, файловою системою, локальними сповіщеннями і мережевою взаємодією. Expo підтримує механізм віддалених оновлень, за якого нові версії програмного коду і ресурсів завантажуються безпосередньо на пристрої користувачів без повторної публікації застосунку у магазинах. Це дає змогу швидко виправляти помилки, оперативно змінювати навчальні сценарії та підтримувати актуальний стан клієнтської частини системи планування та оцінювання навчальної діяльності без значних затримок, пов’язаних із процедурою модерації у службових магазинах [16].

Альтернативним кросплатформеним рішенням є Flutter компанії Google, що базується на мові Dart. У Flutter інтерфейс відтворюється власним графічним рушієм, який працює безпосередньо з графічною підсистемою пристрою і забезпечує однаковий візуальний стиль на різних платформах [17]. Flutter забезпечує високу продуктивність та плавність анімацій, однак використання мови Dart потребує додаткового навчання, а перенесення напрацювань з вебсередовища ускладнюється, оскільки модель побудови інтерфейсу суттєво відрізняється від JavaScript та React.

Порівняння розглянутих підходів показує, що нативна розробка орієнтована на максимальну продуктивність, але потребує значних ресурсів, а гібридні рішення спрощують початковий етап, проте обмежують продуктивність і природність інтерфейсу. Кросплатформені фреймворки типу React Native та Flutter утворюють компроміс між вартістю, швидкістю розробки і якістю користувацького досвіду. Для системи планування та оцінювання навчальної діяльності, де переважає робота з формами, списками завдань і текстовими відповідями, а також важливе швидке розгортання на обох мобільних платформах, доцільно використати React Native у поєднанні з екосистемою Expo. Такий вибір забезпечує спільну кодову базу для Android і iOS, прийнятну продуктивність, розвинуті засоби доступу до функцій пристрою і інструменти для оперативного оновлення застосунку.

3.1.2 Вибір інструментів розробки серверної частини

Серверна частина системи планування та оцінювання навчальної діяльності виконує функції централізованого керування навчальними даними, збереження результатів оцінювання та надання єдиного інтерфейсу взаємодії для клієнтських застосунків. Вона відповідає за реєстрацію та автентифікацію користувачів, зберігання структури завдань, прийом і обробку відповідей, запуск алгоритму автоматизованого оцінювання та формування підсумкових результатів. На відміну від клієнтської частини, яка орієнтована на зручність взаємодії, серверна частина має забезпечувати цілісність даних, стійкість до помилок, передбачувану продуктивність та можливість поетапного розширення функціональності.

З урахуванням характеру задачі до серверної частини висуваються такі основні вимоги:

— підтримка роботи з текстовими даними;
— стандартизований прикладний інтерфейс для клієнтського застосунку;
— транзакційна цілісність записів у базі даних;
— можливість інтеграції з модулем автоматизованого оцінювання;
— використання поширених, добре документованих технологій.

Одним із можливих рішень для побудови вебзастосунків є використання мови Python разом із фреймворками Django або FastAPI. Django надає високорівневий каркас, механізм об’єктно реляційного відображення, вбудований адміністративний інтерфейс і засоби автентифікації, що дає змогу досить швидко будувати повноцінні вебсистеми та зменшувати обсяг шаблонного коду [18]. FastAPI орієнтований на створення високопродуктивних веб-API з використанням асинхронних обчислень та типізованих описів даних, автоматично формує документацію інтерфейсу і добре підходить для мікросервісної архітектури [19]. Мова Python має розвинену екосистему бібліотек для опрацювання тексту та машинного навчання, що зручно для реалізації алгоритмів семантичного оцінювання. Водночас використання окремої мови програмування для серверної частини у поєднанні з клієнтською частиною на JavaScript ускладнює спільне

використання моделей даних і змушує підтримувати два окремі набори інструментів.

Альтернативним підходом є застосування середовища Node.js як серверної платформи. Node.js реалізує модель неблокувального оброблення вводу-виводу та подієорієнтовану архітектуру, що робить його ефективним для серверів з великою кількістю одночасних підключень, орієнтованих на обмін даними у форматі JSON [20]. Платформа добре узгоджується з обміном короткими запитами мобільних клієнтів системи. Використання однієї мови програмування на клієнті та сервері спрощує перенесення перевірок даних, уніфікує моделі предметної області та зменшує витрати на навчання розробників.

На базі Node.js існує кілька популярних фреймворків для організації серверної логіки. Express надає мінімальний каркас для маршрутизації та оброблення HTTP запитів і залишає розробнику повну свободу у побудові архітектури [21]. Мінімалістична організація коду з використанням Express є зручною на ранніх етапах або у невеликих проєктах, проте вимагає вручну підтримувати структуру модулів, шари доступу до даних, механізми оброблення помилок та узгоджені підходи до організації коду. У системі з чітко виділеними модулями управління користувачами, завданнями, оцінюванням та результатами доцільно використати більш структурований фреймворк.

NestJS є фреймворком поверх Node.js, який використовує мову TypeScript, модульну структуру та підхід інверсії керування. Архітектура NestJS базується на розподілі застосунку на модулі, у межах яких описуються контролери, сервіси та провайдери, а взаємодія між ними організується через механізм залежностей [22]. Контролери відповідають за прийом і маршрутизацію запитів веб API, сервіси інкапсулюють бізнес логіку, модулі групують споріднені компоненти. Таке розділення спрощує масштабування системи, оскільки кожен модуль може розвиватися окремо без порушення роботи інших частин, а чітка структура коду полегшує довгострокову підтримку. У розроблюваній системі ця організація безпосередньо відповідає визначеним на етапі моделювання модулям та логічно продовжує закладену структуру.

Важливою складовою серверної частини є вибір системи керування базами даних. Дані про завдання, відповіді та оцінки мають чітку табличну структуру, між сутностями існують визначені зв’язки, а операції запису повинні виконуватись у транзакційному режимі. За таких умов доцільно використовувати реляційну систему керування базами даних. PostgreSQL є одним з найпоширеніших рішень цього класу і підтримує повну відповідність вимогам транзакційної цілісності, розвинуті механізми індексування та контролю узгодженості даних [23].

Для узгодження серверної логіки з базою даних у середовищі Node.js доцільно використати об’єктно реляційний відображувач. TypeORM та подібні бібліотеки дозволяють описувати структури таблиць і зв’язки між ними у вигляді типізованих класів, автоматично синхронізувати ці описи зі схемою бази та створювати міграції для поетапної зміни структури даних [24]. Застосування ORM зменшує кількість помилок під час ручного написання SQL запитів, забезпечує безпечний доступ до даних і полегшує підтримку узгодженості між моделями предметної області та фізичною схемою бази.

Модуль автоматизованого оцінювання, який виконує семантичне порівняння текстових відповідей з еталонними матеріалами, у структурі системи розглядається як окремий компонент. Серверна частина формує стандартизоване подання еталонної відповіді та відповіді студента, передає їх до модуля семантичного аналізу і отримує числову оцінку подібності. Після нормування цей результат записується до бази даних разом з іншими параметрами виконання завдання, а відповідний сервіс NestJS забезпечує узгоджене використання цього механізму у різних частинах системи.

З урахуванням проведеного аналізу для серверної частини системи планування та оцінювання навчальних завдань було вирішено використати середовище Node.js у поєднанні з фреймворком NestJS, реляційну систему керування базами даних PostgreSQL та ORM рішення TypeORM. Такий вибір узгоджується з клієнтською частиною на основі JavaScript, забезпечує можливість подальшого розширення функціональності та спирається на перевірені інструменти із розвиненою екосистемою підтримки.

3.2 Розробка модуля авторизації

Модуль авторизації є базовим елементом системи планування та оцінювання навчальної діяльності. Через нього здійснюється первинна ідентифікація користувача та встановлюється зв’язок між обліковим записом і даними про навчальні курси, завдання, відповіді й результати. Після входу в систему усі подальші операції виконуються від імені автентифікованого користувача, тому коректність роботи модуля визначає захищеність доступу до персональної інформації, а також можливість формування історії виконання завдань та індивідуальної статистики успішності.

Інтерфейс екрана авторизації побудований за принципом поділу на дві логічні області. У верхній частині відображається назва застосунку та допоміжні графічні елементи, нижня частина реалізована як панель, що висувається знизу екрана і містить форму входу або реєстрації. У режимі входу користувач вводить електронну пошту та пароль, у режимі реєстрації додатково вносить прізвище, ім’я та обирає роль, що визначає подальший сценарій роботи. Розмітка інтерфейсу сторінки авторизації наведено у лістингу 3.1.

Лістинг 3.1 — Розмітка сторінки авторизації

```
export const AuthScreen = () => {
  const [isRegister, setIsRegister] = useState(false);

  return (
    <View style={styles.root}>
      <View style={styles.heroArea}>
        <View style={styles.heroBgShapeA} />
        <View style={styles.heroBgShapeB} />
        <SafeAreaView edges={["top"]}>
          <View style={styles.topTabsContainer}>
            <Pill label="Вхід" active={!isRegister} onPress={() => setIsRegister(false)} />
            <Pill label="Реєстрація" active={isRegister} onPress={() => setIsRegister(true)} />
          </View>
        </SafeAreaView>
          <View style={styles.bottomBlock}>
            <Text style={styles.brand}>{BRAND_NAME}</Text>
          </View>
      </View>
      <KeyboardAvoidingView>
        <SafeAreaView edges={["bottom"]}>
```

```
        <ScrollView
          keyboardShouldPersistTaps="handled"
          contentContainerStyle={styles.sheetContent}
          showsVerticalScrollIndicator={false}
        >
          <AuthForm isRegister={isRegister} />
        </ScrollView>
      </SafeAreaView>
    </KeyboardAvoidingView>
  </View>
 );
};
```

Перемикання між режимами здійснюється за допомогою двох вкладок, пов’язаних зі станом isRegister: активний стан визначає, чи відображається форма входу чи форма реєстрації. Нижня частина екрана реалізована як область, обгорнута в контейнер, що коригує положення форми відносно екранної клавіатури, та додатково розміщена всередині прокручуваного представлення. Така організація забезпечує коректне відображення форми на екранах різного розміру та зручну взаємодію при введенні даних.

Компонент AuthForm визначає структуру і поведінку форми авторизації. У ньому описуються поля для введення електронної пошти, пароля, підтвердження пароля та імені користувача, а також правила валідації для кожного поля. Набір полів залежить від вибраного режиму, у режимі входу використовуються лише ідентифікаційні дані, у режимі реєстрації додаються додаткові поля. Той самий компонент форми застосовується на всіх варіантах екрана авторизації без дублювання розмітки, тому зміна структури форми потребує оновлення лише в одному місці програмного коду.

Поля форми авторизації інтегровані з логікою оброблення через компонент Controller, який пов’язує елементи введення з внутрішнім станом форми та механізмами валідації. Кожне поле описується як керований елемент, що працює з відповідним атрибутом форми. Для вибору ролі у режимі реєстрації використовується окремий блок, у якому елементи відображаються як варіанти з візуальним виділенням активного стану. Приклад підключення поля до контролера та користувацького компонента Input наведено в лістингу 3.2.

Лістинг 3.2 — Підключення поля форми до контролера та компонента введення

```
<Controller
  key={id}
  control={control}
  name={id}
  render={({ field: { onChange, onBlur, value }, fieldState: { error } }) => (
    <Input
      {...props}
      variant="filled"
      value={value}
      onChangeText={onChange}
      onBlur={onBlur}
      errorMessage={error?.message}
    />
  )}
/>
```

Контролер отримує з стану поточне значення поля, обробники зміни та втрати фокусу, а також інформацію про помилку валідації. У компонент Input передається значення, яке відображається в полі введення, обробник onChangeText, що оновлює стан форми, та повідомлення про помилку для виведення під полем у разі некоректних даних. Компоненти введення залишаються універсальними елементами інтерфейсу, а логіка керування станом і перевірки даних повністю зосереджена в шарі форми.

Логіка роботи форми зосереджена в окремому хуку, який відповідає за керування станом режиму роботи форми, вибір схеми валідації, початкові значення полів та обробку відправки форми. Його лістинг наведено у лістингу 3.3.

Лістинг 3.3 — Логіка обробки форми авторизації

```
export const useAuthForm = ({ isRegister }) => {
  const { register, login } = useAuth();
  const resolver = useMemo(
    () => (isRegister ? registerResolver : loginResolver),
    [isRegister]
  );

  const form = useForm({
    defaultValues: loginDefaults,
    mode: "onSubmit",
    resolver,
  });
```

```
  const { reset } = form;

  useEffect(() => {
    reset(isRegister ? registerDefaults : loginDefaults, {
      keepErrors: false,
      keepDirty: false,
      keepTouched: false,
    });
  }, [isRegister, reset]);

  const onSubmit = async (data) => {
    if (isRegister && isUserRegisterType(data)) {
      await register?.(data);
      return;
    }
    await login?.({ email: data.email, password: data.password });
  };

  return {
    form,
    onSubmit,
  };
};
```

За допомогою бібліотеки react-hook-form створюється об’єкт form, який відповідає за зберігання значень полів, стан валідації та запуск обробника відправки. Для перевірки коректності введення використовується бібліотека zod, а схеми валідації підключаються до react-hook-form через спеціальний адаптер. Схема авторизації містить правила щодо обов’язковості полів електронної пошти та пароля, а також перевірку формату електронної адреси і мінімальної довжини пароля. Схема реєстрації розширює ці правила додатковими вимогами до імені, прізвища та обов’язкового вибору ролі. Хук приймає значення isRegister, на основі якої обирається відповідна схема валідації та набір значень за замовчуванням. Під час зміни режиму виконується скидання поточного стану форми, включно з полями та повідомленнями про помилки.

Функція onSubmit інтегрує форму з контекстом авторизації. У режимі реєстрації вона ініціює створення нового облікового запису з використанням переданих користувачем даних, у режимі входу надсилає запит з електронною поштою і паролем для перевірки облікових даних. Обидві операції виконуються асинхронно і в разі успіху оновлюють глобальний стан авторизації. Контекст

автентифікації відповідає за відправку запитів на серверну частину, обробку відповідей та зберігання отриманого токена доступу з параметрами поточної сесії.

Після успішної авторизації користувачеві надається доступ до відповідної частини функціональності системи залежно від його ролі. Для цього використовується окремий компонент навігації верхнього рівня, який аналізує стан автентифікації та роль користувача і на основі цих даних обирає відповідний стек екранів. Фрагмент навігаційної логіки наведено в лістингу 3.4.

Лістинг 3.4 — Вибір навігаційного стеку залежно від статусу авторизації та ролі користувача

```
export const AppNav = () => {
  const { isLoading, userToken, userRole } = useAuth();

  if (isLoading) {
    return (
      <View style={{ flex: 1, justifyContent: "center", alignContent: "center" }}>
        <ActivityIndicator size="large" />
      </View>
    );
  }
  if (!userToken) {
    return <AuthStack />;
  }
  return userRole === "teacher" ? <TeacherStack /> : <StudentStack />;
};
```

Компонент AppNav використовує три значення з контексту автентифікації: isLoading, userToken та userRole. Змінна isLoading відображає стан ініціалізації авторизаційних даних. Поки система завантажує збережену інформацію про користувача, компонент не будує жоден із робочих стеків навігації, а повертає екран з індикатором очікування, що запобігає короткочасному мерехтінню неправильних екранів при запуску застосунку.

Змінна userToken містить токен доступу, який серверна частина повертає після успішної авторизації. Якщо значення токена відсутнє, користувач вважається неавторизованим і навігаційний компонент формує стек екранів авторизації AuthStack. Коли токен присутній, користувач розглядається як авторизований і переходить до основного робочого інтерфейсу.

Змінна userRole визначає тип облікового запису і використовується для вибору варіанта основної навігації. Для студента активується набір екранів StudentStack, пов’язаний з отриманням і виконанням завдань, для викладача формується інший набір TeacherStack, у якому зосереджені засоби створення завдань, перевірки відповідей і перегляду статистики. Розділення інтерфейсу за ролями запобігає змішуванню функцій і спрощує використання системи планування та оцінювання навчальних завдань.

3.3 Розробка користувацького інтерфейсу

Інтерфейс користувача системи реалізовано як набір екранів, розділених за ролями. Після проходження авторизації користувач переходить або до студентського, або до викладацького інтерфейсу. Для кожної ролі визначено окремі сценарії роботи. Студент переглядає призначені завдання і результати оцінювання та відправляє відповіді на перевірку. Викладач створює та редагує завдання, контролює їх виконання і аналізує показники успішності групи.

Інтерфейс студента складається з головного екрана із переліком активних завдань, екрана детального перегляду вибраного завдання та сторінки статистики. Центральним елементом головного екрана є компонент картки завдання, який повторно використовується у списку і відображає основні характеристики кожного завдання у компактному вигляді.

На головному екрані студент бачить перелік усіх доступних завдань із короткою інформацією: назвою дисципліни, формулюванням завдання, викладачем, кінцевим строком виконання та поточним станом завдання. Кожен запис цього переліку представлений окремим компонентом Task, який виконує функції відображення основних даних та ініціювання переходу до детального перегляду. Фрагмент реалізації компонента наведено у лістингу 3.5.

Лістинг 3.5 — Компонент картки навчального завдання для студента

```
export const Task = ({ id, course, title, expirationDate, mark, author, submitted }) => {
  const navigation = useNavigation();
```

```
  const handleNavigate = () => {
    navigation.navigate("Task", { id });
  };

  const status = useMemo(() => {
    if (!submitted)
      return { label: "Призначено", style: styles.statusAssigned };
    if (submitted && !mark)
      return { label: "Очікує", style: styles.statusPending };
    return { label: "Оцінено", style: styles.statusGraded };
  }, [submitted, mark]);

  return (
    <Pressable
      containerStyle={styles.container}
      pressableStyle={styles.pressable}
      onPress={handleNavigate}
      rippleColor="accent"
    >
      <View style={styles.headerRow}>
        <Text style={styles.course}>{course || ""}</Text>
        <View style={[styles.statusPill, status.style]}>
          <Text style={styles.statusText}>
            {status.label}
            {status.label === "Оцінено" && mark ? ` • ${mark}/100` : ""}
          </Text>
        </View>
      </View>
      <Text style={styles.titleText}>{title}</Text>
      {!!author && (
        <Text style={styles.authorText}>
          Викладач: <Text style={styles.authorValue}>{author}</Text>
        </Text>
      )}
      <View style={styles.metaRow}>
        <Text style={styles.deadline}>
          Термін здачі:{" "}
          <Text style={styles.deadlineValue}>
            {formatDateDisplay(expirationDate)}
          </Text>
        </Text>
      </View>
    </Pressable>
  );
};
```

Компонент Task приймає параметри id, course, title, expirationDate, mark, author і submitted, які задають вміст і стан елемента списку. Ідентифікатор id застосовується для переходу до екрана детального перегляду завдання. Значення

course відображає назву навчальної дисципліни. Параметр title містить формулювання завдання. Параметр expirationDate задає кінцевий термін виконання. Параметр mark містить підсумковий результат оцінювання. Параметр author зберігає ім’я викладача. Параметр submitted фіксує факт надсилання відповіді на завдання.

Внутрішня змінна status обчислюється на основі поєднання значень submitted і mark і задає поточний стан завдання для виведення у картці. Це значення визначає текст мітки стану та вибір стилю візуального індикатора. Обробник handleNavigate викликається під час натискання на картку і ініціює перехід до екрана з детальним описом завдання, для ідентифікації якого застосовується параметр id.

Детальний екран завдання показує повну інформацію про вибране завдання і забезпечує взаємодію студента з ним. У верхній частині екрана розміщуються назва курсу, формулювання завдання, кінцевий термін здачі, поточний статус перевірки та наявний результат. Нижче виводиться блок, який відображає залишок часу до завершення дозволеного інтервалу виконання, а також вкладки з додатковими деталями і описом завдання. У нижній частині екрана розташовано елемент для додавання файлів відповіді, що далі передаються на перевірку.

Обчислення залишку часу виконується у спеціальному компоненті, який формує відсотковий показник прогресу. Фрагмент коду, що реалізує розрахунок відсотка та його візуалізацію, наведено у лістингу 3.6.

Лістинг 3.6 — Обчислення та відображення залишку часу до терміну здачі

```
const progress = useMemo(() => {
  const start = new Date(publishDate).getTime();
  const end = new Date(expirationDate).getTime();
  if (end <= start) return 0;
  const now = Date.now();
  const pctElapsed = ((now - start) / (end - start)) * 100;
  const pctLeft = 100 - pctElapsed;
  return Math.max(0, Math.min(100, Math.round(pctLeft)));
}, [publishDate, expirationDate]);
<View style={styles.progressCard}>
  <View style={styles.progressHeader}>
    <Text style={styles.progressLabel}>Скільки залишилось</Text>
    <Text style={styles.progressValue}>{progress}%</Text>
  </View>
```

```
  <View style={styles.progressBar}>
    <View style={[styles.progressFill, { width: `${progress}%` }]} />
  </View>
  <View style={styles.progressFooter}>
    <Text style={styles.progressFootText}>
      {publishDate ? formatDateDisplay(publishDate) : "—"}
    </Text>
    <Text style={styles.progressFootText}>{formatDateDisplay(expirationDate)}</Text>
  </View>
</View>
```

Змінна progress описує відсотковий залишок часу до завершення терміну здачі. Значення дати публікації publishDate та кінцевої дати expirationDate перетворюються на числові часові мітки у мілісекундах. У разі коли кінцева дата не перевищує дату публікації, функція повертає нуль, що відповідає вичерпаному або некоректно заданому інтервалу. Далі обчислюється частка часу, що вже минула від моменту публікації до поточного часу, на основі відношення різниці між поточним часом та початком інтервалу до загальної тривалості між початковою та кінцевою датами. З отриманого значення визначається частка часу, що залишилась, після чого результат обмежується діапазоном від нуля до ста і заокруглюється до цілого числа.

Отримане значення progress використовується у трьох елементах інтерфейсу. У заголовку блоку воно відображається як текстовий відсотковий показник. У шкалі прогресу це значення керує шириною заповненої частини, що дає студенту наочне уявлення про наближення до терміну здачі. У нижній частині компонента демонструються початкова та кінцева дати у зручному для читання форматі, а у разі відсутності дати публікації виводиться спеціальне текстове повідомлення. Поєднання текстового та графічного відображення забезпечує користувача повною інформацією щодо обмежень у часі для конкретного завдання.

Екран статистики студента узагальнює інформацію про виконання навчальних завдань. У верхній частині екрана відображаються агреговані показники, серед яких кількість призначених завдань, кількість робіт, що очікують перевірки, кількість уже оцінених відповідей і середній бал. Нижче розміщено інформаційні блоки, які деталізують стан виконання завдань. Один блок описує

розподіл завдань за статусами, інший формує перелік завдань із найближчими термінами здачі. Така організація представлення даних допомагає студентові швидко оцінити власну ситуацію і визначити, які завдання потребують першочергової уваги.

Блок, що відповідає за відображення розподілу статусів опирається на попередньо підрахованій кількості завдань у станах призначено, очікують на перевірку та оцінено. Фрагмент розмітки цього блоку наведено у лістингу 3.7.

Лістинг 3.7 — Відображення розподілу статусів завдань

```
const gradedPercent = total ? Math.round((graded / total) * 100) : 0;
const pendingPercent = total ? Math.round((pending / total) * 100) : 0;
const assignedPercent = total ? Math.round((assigned / total) * 100) : 0;
<View style={styles.block}>
  <Text style={styles.blockTitle}>Розподіл статусів</Text>
  <View style={styles.row}>
    <Text style={styles.rowLabel}>Призначено</Text>
    <Text style={styles.rowValue}>{stats.assignedPercent}%</Text>
  </View>
  <ProgressBar value={stats.assignedPercent} color={theme.statusAssigned} />
  <View style={styles.row}>
    <Text style={styles.rowLabel}>Очікують</Text>
    <Text style={styles.rowValue}>{stats.pendingPercent}%</Text>
  </View>
  <ProgressBar value={stats.pendingPercent} color={theme.statusPending} />
  <View style={styles.row}>
    <Text style={styles.rowLabel}>Оцінено</Text>
    <Text style={styles.rowValue}>{stats.gradedPercent}%</Text>
  </View>
  <ProgressBar value={stats.gradedPercent} color={theme.statusGraded} />
</View>
```

Змінні assigned, pending та graded містять кількість завдань у відповідних станах. Змінна total зберігає сумарну кількість завдань. Для кожного стану обчислюється відсоткова частка від загальної кількості і відображається у вигляді числа у правій частині рядка. Ті самі значення використовуються для керування компонентом ProgressBar, який виводить горизонтальні індикатори з окремим кольором для кожного статусу. Студент за одним поглядом бачить розподілення завдань по статусам і пріоритетні до виконання задачі.

Блок нагадувань про завдання з найближчим терміном здачі формує стислий список актуальних елементів. На основі переліку призначених завдань кінцеві дати

перетворюються у числовий формат, список сортується за зростанням терміну здачі і вибираються перші кілька записів. Якщо для всіх завдань терміни вже минули, використовується загальний відсортований список, який все одно дає студенту змогу побачити найбільш актуальні за часом завдання. Для кожного елемента виводяться назва завдання та відформатована дата здачі. У лістингу 3.8 наведено фрагмент коду формування такого списку.

Масиви assignedList, tasksByDate і futureTasks відображають поетапну обробку даних. У assignedList зберігаються всі призначені завдання, tasksByDate містить їх копію, відсортовану за терміном здачі. Масив futureTasks зберігає лише завдання з датами, які ще не минули. Масив upcomingList містить перші три елементи futureTasks і передається до компонента відображення нагадувань.

Лістинг 3.8 — Формування переліку найближчих завдань

```
const tasksByDate = assignedList
  .map((task) => ({ task, date: parseDate(task.expirationDate) }))
  .filter((item) => item.date)
  .sort((a, b) => a.date!.getTime() - b.date!.getTime());
const futureTasks = tasksByDate.filter(
  (item) => item.date!.getTime() >= Date.now()
);
const upcomingList = (futureTasks.length ? futureTasks : tasksByDate)
  .slice(0, 3)
  .map((item) => item.task as TaskItem);
```

Інтерфейс викладача має власний набір екранів, адаптованих до завдань керування навчальним процесом. На головному екрані показується зведена інформація про створені та призначені завдання, нижче знаходиться список завдань із короткими характеристиками. На відміну від інтерфейсу студента, орієнтованого на індивідуальне виконання, викладацький інтерфейс зосереджений на роботі з усім набором завдань курсу.

Для спрощення навігації у списку завдань у верхній частині розміщено панель фільтрів. Кожен елемент панелі відповідає одному з режимів фільтрації, таких як перегляд всіх, тільки активних, тільки тих, що очікують перевірки, або вже завершених завдань. Логіку формування відфільтрованого списку наведено у лістингу 3.9.

Лістинг 3.9 — Формування відфільтрованого списку завдань у панелі викладача

```
const filters = [
  { value: "all", label: "Усі" },
  { value: "active", label: "Активні" },
  { value: "review", label: "Очікують перевірки" },
  { value: "completed", label: "Завершені" },
];
const [activeFilter, setActiveFilter] = useState("all");
const filteredTasks = useMemo(() => {
  if (activeFilter === "all") return tasks;
  if (activeFilter === "completed") {
    return tasks.filter((task) => task.status === "completed");
  }
  if (activeFilter === "active") {
    return tasks.filter((task) => task.status === "active");
  }
  if (activeFilter === "review") {
    return tasks.filter((task) => task.status === "review");
  }
  return tasks;
}, [activeFilter]);
```

У змінній activeFilter зберігається поточний режим фільтрації, що відповідає одному з доступних станів завдань. Масив tasks містить повний список завдань, отриманий із серверної частини. Функція, передана до useMemo, аналізує значення activeFilter і повертає відфільтрований масив. Механізм мемоізації обмежує кількість повторних обчислень і підтримує стабільну швидкодію інтерфейсу під час роботи з великими переліками завдань.

Екран створення нового завдання реалізовано у вигляді послідовно організованої форми. У верхній частині форми знаходяться поля вибору курсу, введення назви та текстового опису завдання. Далі встановлюється кінцевий термін здачі та відсоток штрафу за запізнення. Нижня частина форми призначена для прикріплення файлу еталонної відповіді до завдання. Цей файл надалі використовується під час автоматизованого семантичного аналізу на сервері.

Механізм вибору і видалення еталонного файлу реалізується через локальний стан компонента та допоміжні функції роботи з файловою системою пристрою. Відповідний фрагмент коду наведено у лістингу 3.10.

Лістинг 3.10 — Програмна реалізація вибору та скидання еталонного файлу

```
const [referenceAsset, setReferenceAsset] =
  useState(null);
const pickReference = async () => {
  try {
    const res = await DocumentPicker.getDocumentAsync({
      multiple: false,
      type: [
        "application/pdf",
        "text/plain",
        "application/msword",
        "application/vnd.openxmlformats-officedocument.wordprocessingml.document",
        "image/*",
      ],
      copyToCacheDirectory: true,
    });
    if (!res.canceled && res.assets?.length) {
      setReferenceAsset(res.assets[0]);
    }
  } catch (e) {
    console.log("reference file pick error", e);
  }
};
const removeReference = () => setReferenceAsset(null);
```

Змінна referenceAsset зберігає поточний стан прикріпленого еталонного файлу. У початковий момент вона має значення null, що відповідає відсутності обраного файлу. Після успішного вибору документа в цю змінну записується об’єкт, який містить шлях до файлу, його назву, тип та інші службові атрибути, необхідні для подальшого завантаження на сервер.

Функція pickReference виконує асинхронний запит до системного засобу вибору файлів. Параметри виклику обмежують користувача вибором одного документа та визначають допустимі типи файлів, серед яких текстові та офісні формати, а також зображення. Додатковий параметр копіює вибраний файл у кешову директорію застосунку, що спрощує подальше передавання даних. Після завершення операції результат аналізується: якщо користувач не скасував вибір і список вибраних файлів не порожній, у стан referenceAsset записується перший елемент масиву res.assets. У разі виникнення помилки вона виводиться в консоль.

Функція removeReference використовується для скидання вибраного файлу. Вона повертає стан referenceAsset до значення null, що інтерпретується

інтерфейсом як відсутність еталонної відповіді. На рівні користувацького інтерфейсу ця функціональність пов’язана з елементом керування видаленням файлу, який дозволяє викладачеві змінити або відмовитися від раніше обраного еталонного документа перед публікацією завдання.

Окремий екран викладача відповідає за перегляд відповідей студентів і виставлення підсумкових балів. У верхній частині відображається узагальнена інформація про завдання: статус, термін здачі, назва та курс, а також кнопки для редагування параметрів і закриття завдання. Нижче розташовано панель фільтрів за станом перевірки та список відповідей студентів. Для кожної відповіді показано прізвище та ім’я студента, дату й час надсилання, поточний статус перевірки, за потреби автоматично обчислену оцінку та кнопки для перегляду роботи й переходу до оцінювання.

Для кожної відповіді викладач може відкрити перегляд змісту роботи або перейти до встановлення чи корекції оцінки через окреме діалогове вікно. У цьому вікні викладач бачить ім’я студента, значення автоматичної оцінки, а також поле для введення власного підсумкового бала. Фрагмент коду, що описує логіку виведення діалогового вікна наведено у лістингу 3.11.

Лістинг 3.11 — Керування вибраною відповіддю та станом форми оцінювання

```
const [submissions, setSubmissions] = useState(mockSubmissions);
const [modalVisible, setModalVisible] = useState(false);
const [selectedSubmission, setSelectedSubmission] = useState(null);
const { control, handleSubmit, reset, formState: { errors } } = useForm({
  defaultValues: { score: undefined },
  resolver: zodResolver(scoreSchema),
  mode: "onSubmit",
});
const openEvaluateModal = (submission) => {
  setSelectedSubmission(submission);
  setModalVisible(true);
  reset({ score: submission.finalScore ?? undefined });
};
const closeEvaluateModal = () => {
  setModalVisible(false);
  setSelectedSubmission(null);
  reset({ score: undefined });
};
```

У змінній submissions зберігається масив об’єктів, кожен з яких представляє окрему відповідь студента. Змінна modalVisible вказує, чи відкрите діалогове вікно оцінювання, а selectedSubmission містить посилання на відповідь, з якою у даний момент працює викладач. Об’єкт форми, створений за допомогою useForm, визначає початкові значення полів, правила валідації та структуру для збереження можливих повідомлень про помилки. Поля control, handleSubmit, reset та errors використовуються для зв’язку елементів введення з формою, оброблення відправки форми, відновлення початкового стану та доступу до повідомлень про помилки.

Функція openEvaluateModal викликається під час вибору конкретної відповіді у списку. Вона записує об’єкт відповіді до selectedSubmission, відкриває діалогове вікно і встановлює початкове значення поля оцінки. Якщо для цієї відповіді вже збережено підсумковий бал, він підставляється у форму, інакше поле залишається порожнім. Функція closeEvaluateModal завершує роботу з поточною відповіддю, закриває діалогове вікно, очищує посилання на вибране завдання та повертає форму до початкового стану.

Перевірка коректності введеної оцінки та фіксація результатів у системі виконуються відповідно до схеми валідації. Оцінка повинна бути числовим значенням у діапазоні від нуля до ста, після чого вона може бути збережена як результат автоматичного або ручного оцінювання. Взаємодія з серверним інтерфейсом та оновлення локального списку відповідей реалізовані у функціях підтвердження автоматичної оцінки та збереження оцінки, введеної викладачем. Фрагмент валідації форми оцінювання наведено у лістингу 3.12.

Лістинг 3.12 — Перевірка оцінки та оновлення даних про відповідь

```
const scoreSchema = z.object({
  score: z.coerce
    .number({ message: "Обов'язкове поле" })
    .min(0, { message: "Введіть число від 0 до 100" })
    .max(100, { message: "Введіть число від 0 до 100" }),
});
const handleConfirmAutoScore = async () => {
  if (!selectedSubmission) return;
  if (selectedSubmission.autoScore === undefined) return;
  await api.updateSubmissionScore(selectedSubmission.id, {
    score: selectedSubmission.autoScore,
```

```
    source: "auto",
  });

  setSubmissions((prev) =>
    prev.map((submission) =>
      submission.id === selectedSubmission.id
        ? {
            ...submission,
            status: "graded",
            finalScore: selectedSubmission.autoScore,
          }
        : submission,
    ),
  );
  closeEvaluateModal();
};

const handleSaveManualScore = handleSubmit(async ({ score }) => {
  if (!selectedSubmission) return;
  const roundedScore = Math.round(score);
  await api.updateSubmissionScore(selectedSubmission.id, {
    score: roundedScore,
    source: "manual",
  });
  setSubmissions((prev) =>
    prev.map((submission) =>
      submission.id === selectedSubmission.id
        ? { ...submission, status: "graded", finalScore: roundedScore }
        : submission,
    ),
  );

  closeEvaluateModal();
});
```

Схема scoreSchema задає формат і обмеження для значення оцінки. Значення, введене у текстове поле, примусово перетворюється на число і перевіряється на заповненість та належність до інтервалу від нуля до ста. У разі порушення цих умов формується текст повідомлення, яке відображається під полем введення Функція handleConfirmAutoScore обробляє випадок, коли викладач погоджується з результатом автоматичного оцінювання. Вона надсилає автоматичну оцінку на сервер за допомогою методу updateSubmissionScore, оновлює локальний стан submissions і фіксує стан відповіді як перевірений запис з підсумковим балом. Функція handleSaveManualScore застосовується тоді, коли викладач задає бал вручну.

Після успішної валідації значення оцінки округлюється до найближчого цілого, надсилається до серверної частини як результат ручного оцінювання і записується до локального стану. У обох сценаріях після завершення операції діалогове вікно закривається, а на екрані відображається оновлена інформація про перевірені роботи.

3.4 Розробка серверної частини

Серверна частина системи планування та оцінювання навчальної діяльності відповідає за централізоване зберігання навчальних даних і керування всіма операціями з ними. У ній зосереджується логіка реєстрації та автентифікації користувачів, створення і супроводу курсів, формування завдань, приймання відповідей студентів та фіксації результатів оцінювання. Доступ клієнтських застосунків до цих операцій організовано через REST інтерфейс, у межах якого кожен HTTP запит обробляється як окрема операція створення, читання, оновлення або збереження записів, що описують виконання завдань.

Модель даних орієнтована на сутності навчального процесу. Структура даних базується на сутностях навчального процесу. Кожен користувач може виступати як студент або викладач. До користувачів прив’язуються навчальні курси і завдання. Оцінка зберігається у підсумковому вигляді і може бути скоригована викладачем. Структура бази даних наведена на рисунку 3.1.

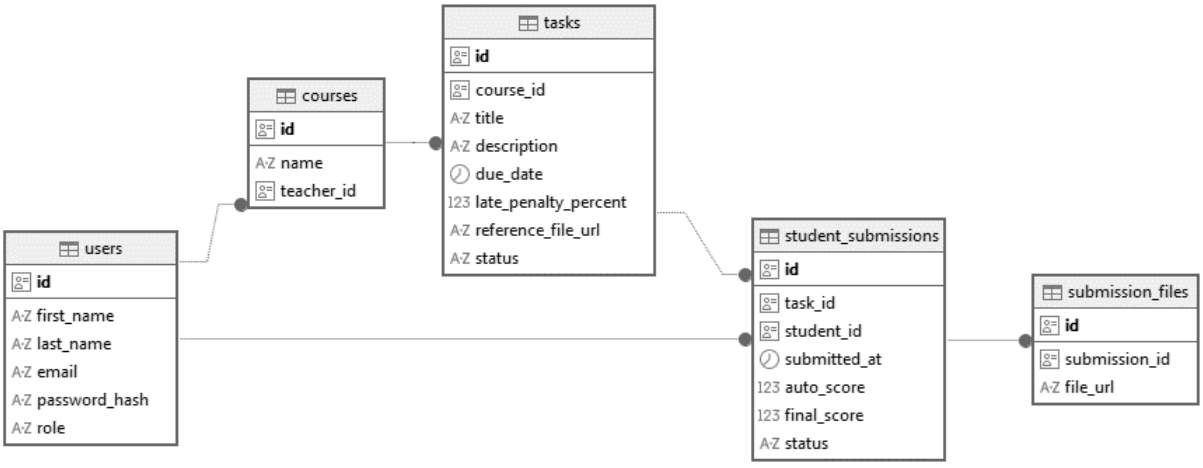

Рисунок 3.1 — Структура бази даних

Таблиця users містить облікові записи всіх користувачів з повним ім’ям, електронною поштою, хешованим паролем та роллю. Таблиця courses описує навчальні дисципліни та зберігає посилання на викладача, відповідального за їх ведення. У таблиці tasks описуються завдання, де зберігаються назва, опис, кінцевий термін виконання, відсоток штрафу за запізнення та шлях до еталонного файлу, який використовується для автоматичного оцінювання.. Таблиця student_submissions фіксує відповіді студентів, містить посилання на завдання і автора, дату надсилання, значення автоматичної оцінки і підсумковий бал. Допоміжна таблиця submission_files описує файли, пов’язані з конкретною відповіддю, і зберігає шлях до кожного файлу та його тип, що дає змогу працювати з текстовими документами, зображеннями та іншими форматами.

Доступ серверної частини до бази даних реалізовано через ORM бібліотеку TypeORM. Головний модуль серверної частини ініціалізує постійне з’єднання з системою керування базами даних PostgreSQL з використанням конфігурації, наведеної у лістингу 3.13.

Лістинг 3.13 — Конфігурація підключення серверної частини до бази даних

```
TypeOrmModule.forRoot({
  type: 'postgres',
  host: process.env.DB_HOST,
  port: parseInt(process.env.DB_PORT, 10),
  username: process.env.DB_USER,
  password: process.env.DB_PASSWORD,
  database: process.env.DB_NAME,
  autoLoadEntities: true,
  synchronize:false,
});
```

Параметр type визначає тип системи керування базами даних, що використовується серверною частиною. Значення postgres відповідає PostgreSQL як основній реляційній СКБД системи. Поля username і password містять облікові дані для автентифікації запитів. Назва бази даних задається у полі database. Опція synchronize забезпечує узгодження структури таблиць зі схемою сутностей під час запуску застосунку, що спрощує підтримку моделі даних на етапі розробки. Параметр autoLoadEntities активує автоматичне виявлення та реєстрацію сутностей

без необхідності їх ручного підключення.

Модуль автоматичного оцінювання відповідає за обчислення подібності між змістом відповіді студента та еталонним розв’язком, що прикріпляється викладачем під час створення завдання. Оцінювання здійснюється на основі векторного представлення тексту, сформованого за допомогою трансформерної моделі типу paraphrase-MiniLM, яка формує компактні семантичні ембединги фіксованої довжини. Для підвищення продуктивності ембединги еталонних відповідей попередньо обчислюються і зберігаються, тому під час перевірки великої кількості робіт сервер повторно використовує вже підготовлені представлення і виконує обробку тільки студентських матеріалів.

Автоматична оцінка представлена як початкове значення, яке при необхідності може бути скориговане викладачем у при ручному контролі результатів навчання. У разі порушення терміну здачі застосовується фіксований коефіцієнт зниження оцінки на основі параметра завдання. Процес перетворення тексту у векторний ембединг наведено у лістингу 3.14.

Лістинг 3.14 — Обчислення ембедингу тексту

```
async function embed(text) {
  const extractor = await this.ensurePipeline();
  const raw = await extractor(text, {
    pooling: 'none',
    normalize: false,
    return_attention_mask: false,
  });
  const lastHiddenState = this.extractLastHiddenState(raw);
  const pooled = this.meanPool(lastHiddenState);
  return { embedding: pooled };
}
```

Параметр text містить рядок з текстом, який необхідно оцінити. Змінна extractor зберігає проініціалізовану трансформерну модель з бібліотеки @xenova/transformers. Змінна raw містить сирий вихід моделі для заданого тексту, тобто багатовимірний тензор з прихованими станами по всіх токенах. Об’єкт параметрів виклику задає режим роботи моделі: опція pooling: 'none' вимикає автоматичне згортання, normalize: false залишає значення без додаткової

нормалізації, return_attention_mask: false означає, що маска уваги не потрібна. Функція extractLastHiddenState виділяє з сирого результату окремий тензор останнього прихованого шару мережі. Функція meanPool обчислює середнє значення по виміру токенів і повертає одновимірний масив, що є вектором фіксованої довжини. У підсумку формується одновимірний масив pooled з числами з плаваючою крапкою, який використовується як ембединг заданого тексту і передається до наступних етапів обробки.

Наступний крок полягає в обчисленні косинусної подібності між ембедингами еталонної відповіді та відповіді студента. Функція cosine з лістингу приймає два вектори a і b, знаходить для них скалярний добуток і окремо обчислює квадрати їхніх норм як суму квадратів компонент. Змінна len визначає кількість компонент, що беруть участь у розрахунку, і дорівнює мінімальній довжині з двох векторів. Змінна dot накопичує значення скалярного добутку векторів. Змінні normA та normB зберігають квадрати довжин кожного з векторів, що обчислюються як сума квадратів компонент. У циклі для кожного індексу обчислюються проміжні значення va і vb, які відповідають окремим координатам векторів, після чого оновлюються суми dot, normA та normB. Змінна denom містить добуток евклідових норм обох векторів і використовується у знаменнику формули косинусної подібності. Функція повертає значення косинусної подібності переданих в неї векторів. Її реалізація наведена в лістингу 3.15.

Лістинг 3.15 — Обчислення косинусної подібності векторів

```
function cosine(a, b) {
  const len = Math.min(a.length, b.length);
  let dot = 0;
  let normA = 0;
  let normB = 0;

  for (let i = 0; i < len; i++) {
    const va = a[i];
    const vb = b[i];
    dot += va * vb;
    normA += va * va;
    normB += vb * vb;
  }
```

```
  const denom = Math.sqrt(normA) * Math.sqrt(normB);
  if (denom === 0) return 0;
  return dot / denom;
}
```

Для узгодження косинусної подібності з прийнятою у системі стобальною шкалою використовується функція toScore, наведена в лістингу 3.16.

Лістинг 3.16 — Перетворення косинусної подібності у 100-бальну шкалу

```
function toScore(similarity) {
  const clamped = Math.max(-1, Math.min(1, similarity));
  const scaled = ((clamped + 1) / 2) * 100;
  return Math.round(Math.max(0, Math.min(100, scaled)));
}
```

Параметр similarity містить значення косинусної подібності, яке потрібно перетворити. На першому етапі значення подібності обмежується інтервалом від мінус одного до одного і зберігається в змінній clamped, що захищає від можливих числових похибок. Після цього проводиться лінійне масштабування до діапазону від нуля до ста, а результат додатково обмежується межами цього інтервалу і зберігається у змінній scaled. Після округлення отримується ціле число, яке розглядається як автоматична оцінка роботи у стобальній системі та може безпосередньо використовуватися в моделі обліку результатів.

На основі наведених допоміжних функцій реалізовано автоматичне оцінювання конкретної відповіді з урахуванням запізнення. Фрагмент сервісу наведено у лістингу 3.17.

Лістинг 3.17 — Автоматичне оцінювання відповіді з урахуванням штрафу за запізнення

```
async function autoEvaluate(submissionId) {
  const submission = await submissionsService.findById(submissionId);
  const referenceText = await resolveTextSource(
    { filePath: submission.task.referenceFileUrl },
    'reference'
  );
  const answerText = await extractSubmissionFilesText(submission);
  let score = await scoreTexts(referenceText, answerText);
  const due = submission.task.dueDate;
  const submitted = submission.submittedAt;
  const penaltyPercent = submission.task.latePenaltyPercent || 0;
```

```
  if (penaltyPercent > 0 && submitted.getTime() > due.getTime()) {
    const factor = Math.max(0, 1 - penaltyPercent / 100);
    score = Math.round(score * factor);
  }
  await submissionsService.updateAutoScore(submissionId, score);
  return score;
}
```

Параметр submissionId зберігає ідентифікатор відповіді студента, яка підлягає оцінюванню. Змінна submission після виклику findById містить повний об’єкт відповіді, включно з пов’язаним завданням та переліком прикріплених файлів. Змінна referenceText формується шляхом вилучення тексту з еталонного файлу, шлях до якого зберігається у полі referenceFileUrl об’єкта завдання. Змінна answerText містить об’єднаний текст усіх файлів, які надіслав студент. Функція scoreTexts обчислює векторні представлення обох текстів, знаходить між ними косинусну подібність і повертає бал, який зберігається у змінній score.

Далі використовуються часові параметри та відсоток штрафу за запізнення. Змінна due містить кінцевий термін здачі роботи. Змінна submitted зберігає фактичну дату та час надсилання відповіді. Змінна penaltyPercent зберігає фіксований відсоток зниження оцінки у разі запізнення. Якщо відповідь подана після встановленого терміну і для завдання задано штраф, оцінка множиться на коефіцієнт, який пропорційно зменшує результат. Оновлене значення передається до методу updateAutoScore, який зберігає результат автоматичного оцінювання у базі даних для відповідного запису з ідентифікатором submissionId. Функція повертає остаточну автоматичну оцінку, яка відображається в інтерфейсі викладача.

Обмін даними між клієнтською та серверною частинами реалізовано у вигляді REST-інтерфейсу. Для кожної сутності визначено набір URL-шляхів та HTTP-методів, які інкапсулюють окремі операції роботи із завданнями. Мобільний застосунок надсилає запити у форматі JSON, а сервер повертає структуровані відповіді з даними та службовими полями, що відображаються на відповідних екранах інтерфейсу. Приклад кінцевої точки, яка запускає автоматичне оцінювання конкретної відповіді, наведено у лістингу 3.18.

Лістинг 3.18 — REST-ендпоінт запуску автоматичного оцінювання відповіді

```
@Controller('submissions')
export class SubmissionsController {
  constructor(evaluationService) {
    this.evaluationService = evaluationService;
  }
  @Post(':id/auto-evaluate')
  async autoEvaluate(@Param('id') id) {
    const score = await this.evaluationService.autoEvaluate(id);
    return { submissionId: id, autoScore: score };
  }
}
```

Анотація @Controller('submissions') визначає базовий шлях для всіх операцій над відповідями студентів. Метод autoEvaluate обробляє HTTP-запит типу POST. Параметр id містить ідентифікатор відповіді у таблиці student_submissions. Декоратор @Param('id') вилучає значення цього параметра з URL-шляху і передає його у тіло методу у змінній id. Усередині методу викликається функція evaluationService.autoEvaluate, яка обчислює бал з урахуванням еталонного файлу та можливого запізнення. Результатом роботи кінцевої точки є об’єкт з полями submissionId та autoScore, що повертається клієнтському застосунку у форматі JSON і далі відображається в інтерфейсі викладача.

4 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Тестування роботи системи

Для перевірки працездатності системи вирішено провести ручне тестування. Воно передбачає відтворення послідовності дій кінцевого користувача, аналіз реакції інтерфейсу та перевірку отриманих результатів відносно очікуваної поведінки. У межах тестування оцінювалася робота програми в реальних сценаріях використання, включно з некоректним введенням даних, повторними діями та переходами між різними розділами мобільного застосунку.

Ручне тестування було обране як основний метод перевірки, оскільки розроблений застосунок орієнтований на безпосередню взаємодію користувача з інтерфейсом. Значна частина функціональності пов’язана з відображенням списків завдань, станів їх виконання, форм введення й підтвердження дій. У таких умовах особливо важливим є візуальний контроль, оцінка зручності і послідовності роботи інтерфейсу, що не може бути повноцінно відтворена засобами автоматизованих тестів. З огляду на обсяг системи та характер функціоналу ручне тестування за підготовленими сценаріями є достатнім і доцільним засобом перевірки працездатності системи.

Першим етапом є тестування механізмів авторизації. Перевіряються сценарії входу з коректними й некоректними даними, реакція форми на порожні поля, невірний пароль, неправильний формат електронної пошти, а також сценарій реєстрації нового користувача з вибором ролі. Для кожної ситуації аналізуються відображення повідомлень про помилки, блокування доступу до основних екранів у разі невірних даних і відкриття відповідного інтерфейсу після успішного входу. Вигляд екрана авторизації, який використовувався під час тестування наведено на рисунку 4.1.

Після перевірки авторизації проводиться тестування сценаріїв роботи викладача. На початковому етапі аналізується стартовий екран викладача, де відображається перелік створених завдань разом із базовою статистикою щодо

кількості призначених, зданих та перевірених робіт. Перевіряється коректне оновлення цих показників після появи нових відповідей студентів та після виставлення підсумкових оцінок. Проводиться перевірка створення нового завдання: введення назви, опису, кінцевого терміну здачі, параметрів штрафу за запізнення, прикріплення файлу еталонної відповіді. Моделюються ситуації частково заповнених форм, відсутності обов’язкових полів та спроби зберегти завдання без еталонного файла.

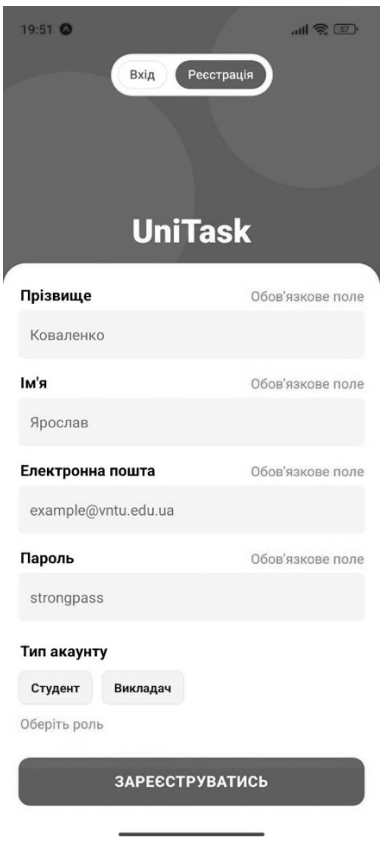

Рисунок 4.1 — Вигляд помилок валідації

Наступна група сценаріїв направлена на перевірку коректності роботи викладача з уже створеними завданнями. Перевіряється відкриття екрана детального перегляду завдання з домашньої сторінки, відображення всіх раніше заданих параметрів, можливість змінювати опис, термін здачі та параметри штрафу з подальшим збереженням змін. Окремо тестується закриття завдання для подальшої здачі та контроль того, що після цього студенти не можуть надсилати нові роботи, а відповідний стан відображається в інтерфейсі.

Розглянемо детальніше процес оцінювання робіт викладачем. Для тестового завдання було підготовлено декілька студентських відповідей з різним змістом та різним часом здачі. Проаналізуємо відображення списку зданих робіт разом з виводом інформації про студента, часу здачі, автоматичної оцінки та стану перевірки. Вигляд сторінки перегляду відповідей для конкретного завдання наведено на рисунку 4.2.

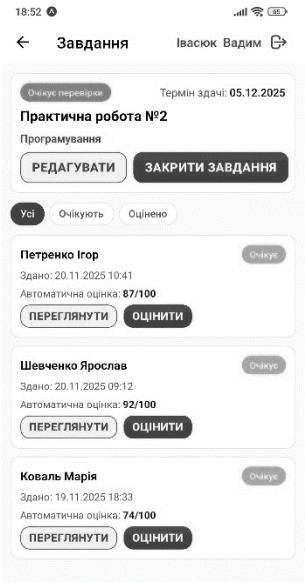

Рисунок 4.2 — Вигляд сторінки перегляду відповідей для завдання

Всі надіслані відповіді відображені в списку зі статусом очікування та мають поле з відображенням автоматичної оцінки. Проведемо перевірку працездатності оцінювання робіт: для однієї відповіді підтвердимо автоматичну оцінку, для іншої — призначимо свою. Для цього скористаємося спеціальним модальним вікном для оцінювання. Його вигляд наведено на рисунку 4.3.

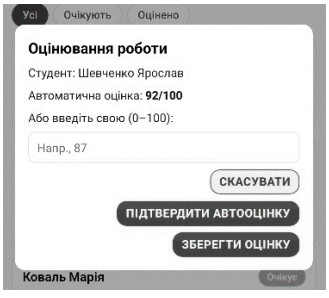

Рисунок 4.3 — Модальне вікно оцінювання роботи

Для підтвердження автоматичної оцінки достатньо натиснути відповідну кнопку. Для задання своєї оцінки потрібно вести бали в поле і підтвердити відправку. Переглянемо сторінку та перевіримо реакцію системи на проведені дії. Вигляд сторінки завдання після проведених дій показано на рисунку 4.4.

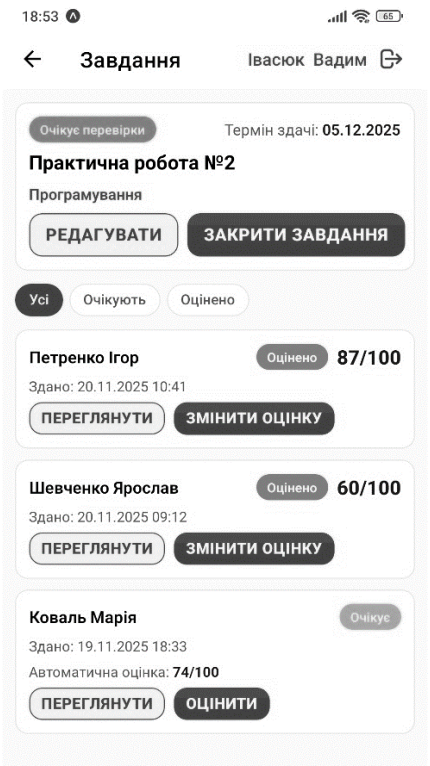

Рисунок 4.4 — Оновлена сторінка перегляду відповідей для завдання

Сторінка оновилась та показує актуальні статуси завдань. Збережені оцінки коректно виведені в правильному місці.

Окремий блок тестів присвячений сценаріям роботи студента. Після успішного входу перевіряється відображення списку призначених завдань із актуальними назвами, короткими описами, кінцевими термінами здачі та позначеннями поточного стану. Для перевірки правильності роботи інтерфейсу, завдання у тестовому наборі переводилися в різні стани етапів здачі. Для кожного стану контролюється коректність текстових позначень та відповідність відображених оцінок і термінів реальному стану навчального процесу.

Подальше тестування стосувалося екрана детальної роботи із завданням для студента. Перевірялися відображення повного формулювання умови, кінцевого

терміну здачі, інформації про поточну оцінку та стан перевірки. Послідовно відтворювалися сценарії здачі роботи без прикріплених файлів, здачі з одним файлом, здачі з кількома файлами, а також повторної здачі до моменту фіксації остаточної оцінки. У кожному випадку контролювалися зміна стану завдання, оновлення часу останньої здачі та відсутність суперечностей у подальшому відображенні результатів для студента і викладача.

Завершальний етап тестування був пов’язаний із перевіркою екрана статистики студента. Для цього створювалися завдання з різними оцінками та станами, після чого аналізувалося відображення загальної картини виконання: кількість призначених, зданих й оцінених завдань, узгодженість відображення оцінок у списку завдань і на сторінці статистики. Перевірялося, що після зміни оцінки викладачем або здачі нової роботи дані на екрані статистики оновлюються без затримок і суперечностей.

Тестування підтвердило, що реалізовані функції системи працюють узгоджено для обох ролей користувачів. Переходи між екранами відповідають спроєктованій логіці, стани навчальних завдань та оцінок відображаються коректно, а валідація введених даних забезпечує захист від помилок під час роботи з формами авторизації, створення завдань та здачі студентських робіт.

4.2 Інструкція користувача

При першому запуску системи користувач потрапляє на екран автентифікації. Для доступу до функцій програми потрібно або виконати вхід до вже створеного облікового запису, або зареєструвати новий. У режимі реєстрації заповнюються поля прізвища, імені, електронної пошти та пароля, а також обирається тип облікового запису студент чи викладач. Після успішної реєстрації користувач може перейти до режиму входу і використовувати зазначену електронну пошту та пароль. Якщо дані форми введено з помилками, біля відповідного поля з’являється коротке пояснювальне повідомлення. Вигляд екранів входу до системи наведено на рисунку 4.5.

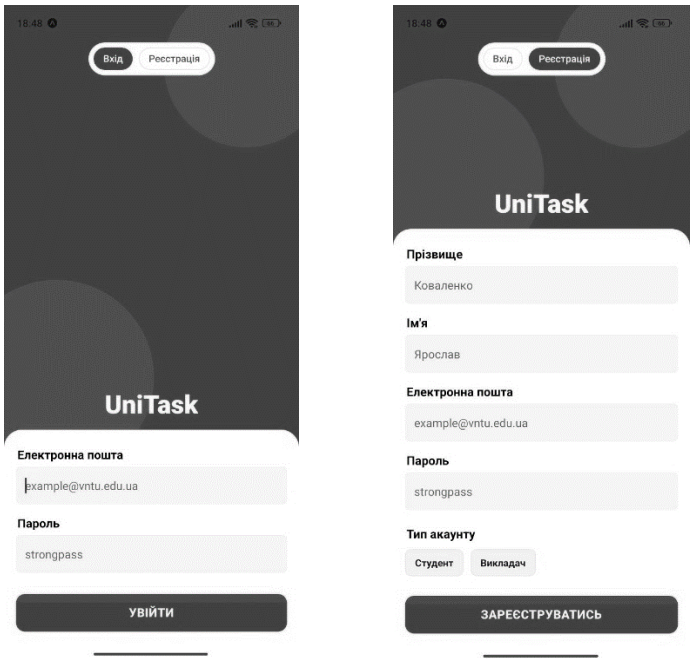

Рисунок 4.5 — Вигляд екранів входу

Після авторизації відкривається стартовий екран, який залежить від обраної ролі. У верхній частині розташований заголовок із назвою системи, ім’ям поточного користувача та кнопкою виходу з облікового запису. Натискання цієї кнопки завершує сеанс і повертає на екран входу.

У разі входу під роллю викладача стартовим є екран панелі керування завданнями. У верхній частині розміщено блок із коротким описом призначення панелі та кнопкою створення нового завдання. Нижче відображаються чотири показники кількості завдань із різними станами призначено, здано, очікують перевірки, оцінено. Під блоком статистики розташовано фільтри, які використовуються для відбору завдань за станом виконання. Основну частину екрана займає список завдань, згрупованих за курсами. Для кожного завдання показуються назва курсу, назва завдання, термін здачі та коротка статистика щодо кількості призначених, зданих, тих що очікують перевірки, і вже оцінених робіт. Вигляд панелі викладача наведено на рисунку 4.6.

Натиснувши на елемент завдання у списку, викладач переходить на сторінку керування конкретним завданням. У верхній частині екрана показано назву завдання, курс, термін здачі та поточний статусом завдання. Поруч розміщені

елементи керування, які дозволяють редагувати параметри або закрити завдання для подальшої здачі. У центральній частині екрана відображається список студентських робіт, поданих на перевірку. Для кожного запису вказуються прізвище та ім’я студента, час здачі, значення автоматичної оцінки та стан перевірки. Із цього списку можна відкрити сторінку перегляду відповіді або викликати вікно виставлення бала.

Вікно оцінювання відображається як модальне вікно, в якому відображається ім’я студента, автоматична оцінка та поле для введення власного значення бала. Дві окремі дії дозволяють або прийняти автоматичний результат без змін, або зберегти значення, введене викладачем. Після підтвердження оцінки стан роботи змінюється на оцінено, а нове значення бала відображається у списку відповідей та на інших екранах, пов’язаних з цим завданням.

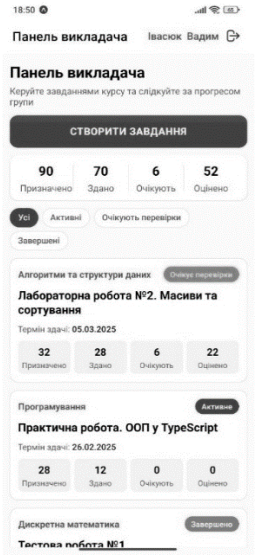

Рисунок 4.6 — Панель викладача

Кнопка створення завдання відкриває екран з формою, що містить поле вибору курсу, поля назви та опису завдання, поле введення кінцевого терміну здачі, а також поле для задання відсотка штрафу за запізнення (рис. 4.7). Окремий блок призначений для завантаження еталонної відповіді у вигляді файлу. У нижній частині екрана розміщені кнопки, за допомогою яких завдання можна зберегти як чернетку або одразу опублікувати для студентів.

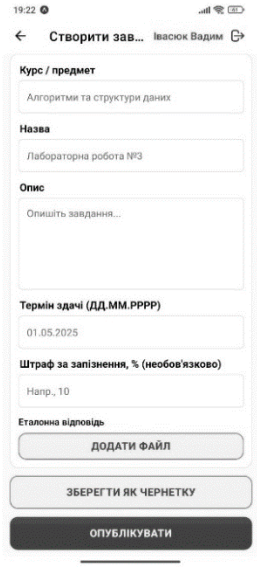

Рисунок 4.7 — Форма створення нового завдання

При вході під роллю студента відкривається домашня сторінка зі списком призначених завдань (рис. 4.8). У верхній частині розташована панель фільтрації, що дає змогу перемикатися між усіма завданнями за станом виконання Список завдань містить картки, у яких зазначені назва курсу, назва завдання, прізвище та ім’я викладача, кінцевий термін здачі, а також коротка текстова підказка щодо залишку часу або факту прострочення. Для завдань із підтвердженою оцінкою додатково показується поточний бал.

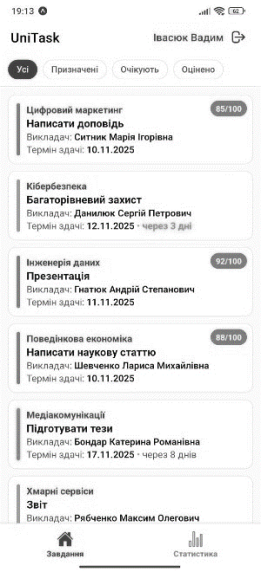

Рисунок 4.8 — Домашня сторінка студента

Натиснувши на завдання у списку, студент переходить на детальну сторінку. У верхньому блоці відображаються назва курсу, назва завдання, термін здачі, відносний показник часу до кінця виконання, а також статус завдання. Нижче розташована шкала, що відображає частку вже використаного часу між датами початку та кінцевого терміну. На вкладці з деталями виводяться структуровані відомості про курс, викладача, дати початку та завершення, тривалість виконання. На вкладці опис розміщується повний текст умови завдання.

У нижній частині сторінки знаходиться область, пов’язана з відповіддю студента. Спочатку вона містить кнопку для вибору файлів. Після її натискання відкривається стандартний засіб вибору файлів мобільної операційної системи, де можна відмітити один або кілька документів. Обрані файли відображаються списком над кнопкою, для кожного елемента передбачена можливість видалення перед відправкою. Кнопка надсилання відповіді розташована в нижній частині екрана і стає активною після вибору хоча б одного файла. Після успішного надсилання статус завдання змінюється, а оновлена інформація відображається як на детальній сторінці, так і у загальному списку завдань. Вигляд детальної сторінки завдання студента показано на рисунку 4.9.

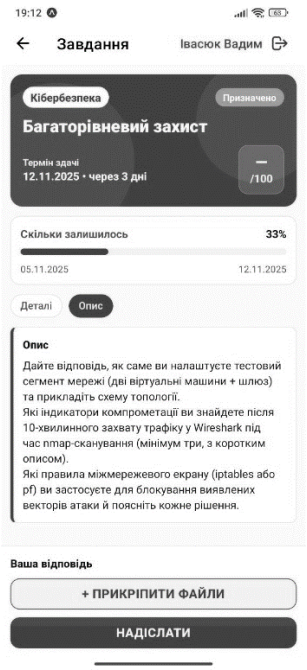

Рисунок 4.9 — Детальна сторінка інформації про завдання

У нижній панелі навігації студентського інтерфейсу доступний перехід до екрана статистики. Його вигляд наведено на рисунку 4.10.

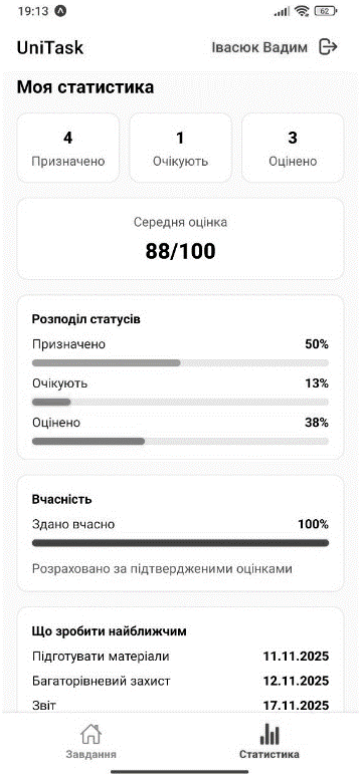

Рисунок 4.10 — Сторінка перегляду статистики

На цьому екрані відображається коротка статистика виконання завдань. У верхній частині виводяться кількість отриманих завдань та середній бал. Далі знаходиться графічне відображення розподілу завдань за статусами та показник своєчасності виконання. У нижній частині розміщується блок найближчих дій, де наведено декілька завдань із найближчими термінами здачі та датами виконання.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Оцінювання комерційного потенціалу розробки

Метою проведення комерційного та технологічного аудиту є підвищення об’єктивності та ефективності оцінювання навчальної діяльності студента в умовах дистанційного навчання шляхом розробки комп’ютерної системи планування і автоматизованого оцінювання відкритих текстових відповідей.

Для проведення технологічного аудиту було залучено 3-х незалежних експертів з ТОВ «Імперіал Енерго»: Швець Вадим Анатолійович, головний інженер; Урсул Олександр Сергійович, інженер, системний адміністратор; Козак Максим Володимирович, інженер.

Для проведення технологічного аудиту було використано таблицю 5.1, в якій за п’ятибальною шкалою використовуючи 12 критеріїв здійснено оцінку комерційного потенціалу [25].

Таблиця 5.1 — Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів

Продовження табл. 5.1

Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

В таблиці 5.2 наведено рівні комерційного потенціалу розробки залежно від суми балів отриманих в ході оцінювання експертами.

Таблиця 5.2 — Рівні комерційного потенціалу розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0-10	Низький
11-20	Нижче середнього
21-30	Середній
31-40	Вище середнього
41-48	Високий

В таблиці 5.3 наведено результати оцінювання експертами комерційного потенціалу розробки.

Таблиця 5.3 — Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта		
	Швець В. А.	Урсул О. С.	Козак М. В.
	Бали, виставлені експертами:		
1	3	3	3
2	3	4	3
3	4	4	4
4	3	4	4
5	4	3	4
6	4	4	4
7	3	3	3
8	3	4	4
9	4	4	3
10	3	3	3
11	4	4	4
12	3	3	3
Сума балів	$СБ_1$=38	$СБ_2$=40	$СБ_3$=39
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_1^3 СБ_i}{3} = \frac{38 + 40 + 39}{3} = 39$		

Середньоарифметична оцінка, отримана на основі експертних висновків, становить 39 балів, і згідно з таблицею 5.2, це вказує на рівень вище середнього комерційного потенціалу результатів проведених досліджень.

Розробка може бути реалізована у вигляді мобільного програмного засобу, інтегрованого в інформаційну інфраструктуру закладу освіти.

Користувачами системи будуть студенти і викладачі. Студенти використовують її для отримання і впорядкованого планування навчальних завдань, відправки виконаних робіт на перевірку та перегляду результатів оцінювання, а викладачі використовують її для формування завдань, автоматизованого оцінювання та аналізу успішності студентів.

Проведемо оцінку якості і конкурентоспроможності нової розробки порівняно з аналогом.

В якості аналога для розробки було обрано систему управління навчанням Moodle, яка забезпечує створення курсів, видачу та приймання завдань, ведення електронного журналу та базові засоби тестового контролю.

Основними недоліками аналога є відсутність вбудованого семантичного автоматизованого оцінювання відкритих текстових відповідей, складність адміністрування та перевантаженість інтерфейсу, що ускладнює використання системи.

Також до недоліків можна віднести залежність від сторонніх плагінів для розширення функціоналу та обмежену підтримку мобільних пристроїв.

У розробці дана проблема вирішується впровадженням модуля автоматизованого семантичного оцінювання відкритих відповідей у межах єдиного мобільного застосунку з простим інтерфейсом для студента і викладача. Проведено аналіз методів подання текстових відповідей і підходів до вимірювання семантичної подібності з урахуванням точності, стійкості до варіативності формулювань, чутливості до контексту та обчислювальних витрат. В результаті було обрано схему векторного подання з подальшим обчисленням косинусної подібності між еталоном і відповіддю студента з нормуванням оцінки до єдиної шкали та урахуванням часових штрафів, що забезпечує відтворюваність процедури і підвищує об’єктивність оцінювання.

Також система випереджає аналог за такими параметрами як ступінь автоматизації перевірки відкритих завдань, інтегрованість процесів планування, оцінювання та обліку результатів, а також зручність використання в умовах дистанційного і змішаного навчання.

В таблиці 5.4 наведені основні техніко-економічні показники аналога і нової розробки.

Проведемо оцінку якості продукції, яка є найефективнішим засобом забезпечення вимог споживачів та порівняємо її з аналогом.

Таблиця 5.4 — Основні параметри нової розробки та товару-конкурента

Показник	Варіанти		Відносний показник якості	Коефіцієнт вагомості параметра
	Базовий (товар-конкурент)	Новий (інноваційне рішення)		
1	2	3	4	5
Дизайн	3	5	1,67	10%
Зручність користування	3	5	1,67	25%
Підтримка мобільних пристроїв	4	5	1,25	20%
Легкість налаштування	3	5	1,67	15%
Підтримка автоматизації оцінювання	2	5	2,5	30%

Визначимо відносні одиничні показники якості по кожному параметру за формулами (5.1) та (5.2) і занесемо їх у відповідну колонку таблиці 5.5.

$$q_i = \frac{P_{Hi}}{P_{Бi}}, \tag{5.1}$$

або

$$q_i = \frac{P_{Бi}}{P_{Hi}}, \tag{5.2}$$

де P_{Hi}, $P_{Бi}$ — числові значення i-го параметру відповідно нового і базового виробів.

$$q_1 = \frac{5}{3} = 1{,}67;$$

$$q_2 = \frac{5}{3} = 1{,}67;$$

$$q_3 = \frac{5}{4} = 1{,}25;$$

$$q_4 = \frac{5}{3} = 1{,}67;$$

$$q_5 = \frac{5}{2} = 2{,}5.$$

Відносний рівень якості нової розробки визначаємо за формулою (5.3):

$$К_{я.в.} = \sum_{i=1}^{n} q_i \cdot \alpha_i \quad . \tag{5.3}$$

$$К_{я.в.} = 1{,}67 \cdot 0{,}1 + 1{,}67 \cdot 0{,}25 + 1{,}25 \cdot 0{,}2 + 1{,}67 \cdot 0{,}15 + 2{,}5 \cdot 0{,}3 = 1{,}84$$

Відносний коефіцієнт показника якості нової розробки більший одиниці, отже нова розробка якісніша базового товару-конкурента.

Наступним етапом є оцінка конкурентоспроможності продукції. Саме вона визначає здатність підприємства успішно конкурувати на ринку та є ключовою передумовою його прибуткової діяльності.

Одним із головних чинників вибору товару споживачем є відповідність основних ринкових властивостей виробу уявним характеристикам, які формують конкретну потребу покупця. До таких властивостей зазвичай відносять нормативні та технічні параметри, а також ціну придбання й витрати, пов'язані з використанням товару.

У таблиці 5.5 подано технічні й економічні показники, необхідні для визначення конкурентоспроможності нової розробки порівняно з товаром-аналогом. Технічні дані взято з попередніх розрахунків.

Загальний показник конкурентоспроможності інноваційного рішення (К) з урахуванням вищезазначених груп показників можна визначити за формулою (5.4):

$$K = \frac{I_{m.n.}}{I_{e.n.}}, \tag{5.4}$$

де $I_{m.n.}$ — індекс технічних параметрів;

$I_{e.n.}$ — індекс економічних параметрів.

Таблиця 5.5 — Нормативні, технічні та економічні параметри нової розробки і товару-виробника

Показники	Варіанти	
	Базовий (товар-конкурент)	Новий (інноваційне рішення)
1	2	3
1. Нормативно-технічні показники		
Дизайн	3	5
Зручність користування	3	5
Підтримка мобільних пристроїв	4	5
Легкість налаштування	3	5
Підтримка автоматизації оцінювання	2	5
2. Економічні показники		
Річна підписка на особу, грн	110	60

Індекс технічних параметрів є відносним рівнем якості інноваційного рішення. Індекс економічних параметрів визначається за формулою (5.5):

$$I_{e.n.} = \frac{\sum_{i=1}^{n} P_{Hei}}{\sum_{i=1}^{n} P_{Бei}}, \tag{5.5}$$

де P_{Hei}, $P_{Бei}$ — економічні параметри (ціна придбання та споживання товару) відповідно нового та базового товарів.

$$\mathrm{I_{e.п.}} = \frac{110}{60} = 1{,}83;$$

$$\mathrm{К} = \frac{1{,}84}{1{,}83} = 1{,}01.$$

Зважаючи на розрахунки, можна зробити висновок, що нова розробка буде конкурентоспроможніше, ніж конкурентний товар.

5.2 Прогнозування витрат на виконання науково-дослідної роботи

Витрати, пов'язані з проведенням науково-дослідної роботи групуються за такими статтями: витрати на оплату праці, витрати на соціальні заходи, матеріали, паливо та енергія для науково-виробничих цілей, витрати на службові відрядження, програмне забезпечення для наукових робіт, інші витрати, накладні витрати.

Зведемо сумарні розрахунки до таблиці 5.6. Основна заробітна плата кожного із дослідників $З_О$, якщо вони працюють в наукових установах бюджетної сфери визначається за формулою (5.6):

$$З_О = \frac{М}{Т_р} * t \text{ (грн)}, \quad (5.6)$$

де M — місячний посадовий оклад конкретного розробника, грн;

T_p — число робочих днів в місяці; приблизно $T_p \approx 21…23$ дні;

t — число робочих днів роботи дослідника.

Таблиця 5.6 — Заробітна плата дослідника в науковій установі бюджетної сфери

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату грн.
Керівник	16000	761,9	5	3810
Програміст	20000	952,4	30	28571
Всього				32381

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт розраховують за формулою (5.7):

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i, \quad (5.7)$$

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 (5.8):

$$C_i = \frac{М_м \cdot К_і \cdot К_с}{Т_р \cdot t_{зм}}, \quad (5.8)$$

де M_M — розмір прожиткового мінімуму працездатної особи або мінімальної місячної заробітної плати (залежно від діючого законодавства), грн;

K_i — коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду;

K_c — 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;

T_p — середня кількість робочих днів в місяці, приблизно T_p = 21…23 дні;

$t_{зм}$ — тривалість зміни, год.

Таблиця 5.7 — Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Погодинна тарифна ставка, грн	Величина оплати на робітника, грн
1. Підготовка	8	1	47,6	381,0
2. Серверна частина	118	5	81,0	9552,4
3. Клієнтська частина	74	4	71,4	5285,7
4. Розгортання	32	3	64,3	2057,1
5. Тестування	8	2	52,4	419,0
Всього				17695,2

Додаткова заробітна плата $З_д$ всіх розробників та робітників, які приймали устають в розробці нового технічного рішення розраховується як 10-12% від основної заробітної плати робітників (5.9).

На даному підприємстві додаткова заробітна плата начисляється в розмірі 11% від основної заробітної плати.

$$З_д = (З_о + З_р) * \frac{Н_{дод}}{100\%}. \quad (5.9)$$

$$З_д = 0{,}11 * (32381 + 17695{,}2) = 5508{,}38 \text{ (грн)}$$

Нарахування на заробітну плату $Н_{ЗП}$ дослідників та робітників, які брали участь у виконанні даного етапу роботи, розраховуються за формулою (5.10):

$$Н_{ЗП} = (З_о + З_р + З_д) * \frac{\beta}{100} \text{ (грн)}, \quad (5.10)$$

де $З_о$ — основна заробітна плата розробників, грн.;

$З_р$ — додаткова заробітна плата всіх розробників та робітників, грн.;

$З_д$ — основну заробітну плату робітників, грн.;

β — ставка єдиного внеску на загальнообов'язкове державне соціальне страхування, %.

Дана діяльність відноситься до бюджетної сфери, тому ставка єдиного внеску на загальнообов'язкове державне соціальне страхування буде складати 22%, тоді:

$$Н_{ЗП} = (32381 + 17695{,}2 + 5508{,}38) * \frac{22}{100} = 12228{,}6 \text{ (грн)}$$

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби й предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень за прямим призначенням згідно з нормами їх витрачання, а також витрачені придбані напівфабрикати, що підлягають монтажу або виготовленню й додатковій обробці в цій організації, чи дослідні зразки, що виготовляються виробниками за документацією наукової організації.

Витрати на матеріали (М) у вартісному вираженні розраховуються окремо для кожного виду матеріалів за формулою (5.11):

$$\text{М} = \sum_{\text{i}=1}^{n} H_j \cdot \text{Ц}_j \cdot \text{К}_j - \sum_{\text{i}=1}^{n} \text{В}_j \cdot \text{Ц}_{\text{в}j}, \quad (5.11)$$

де *Hj* — норма витрат матеріалу j-го найменування, кг;

n — кількість видів матеріалів;

Цj — вартість матеріалу j-го найменування, грн/кг;

Кj — коефіцієнт транспортних витрат, (*Кj* = 1,1 … 1,15);

Вj — маса відходів j-го найменування, кг;

Цвj — вартість відходів j-го найменування, грн/кг.

Проведені розрахунки зведені в таблицю 5.8.

Таблиця 5.8 — Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн	Норма витрат, шт	Вартість витраченого матеріалу, грн
Папір	195	1	195
Ручка	18	1	18
Блокнот	70	1	70
Флешка	310	1	310
З врахуванням коефіцієнта транспортування			652,3

Балансову вартість програмного забезпечення розраховують за формулою (5.12):

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i, \quad (5.12)$$

де $Ц_{iпрг}$ — ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.i}$ — кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_i — коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, (K_i = *1,10…1,12)*;

k — кількість найменувань програмних засобів.

Отримані результати необхідно звести до таблиці 5.9.

Таблиця 5.9 — Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Visual Studio Code	1	0	0
Android Studio	1	0	0
PostgreSQL Server	1	0	0
Postman	1	0	0
Всього з врахуванням налагодження			0

Усе програмне забезпечення, яке використовувалось при написані магістерської роботи є безкоштовним.

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, можуть бути розраховані з використанням прямолінійного методу амортизації (5.13). Проведені розрахунки необхідно звести до таблиці 5.10.

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \tag{5.13}$$

де $Ц_б$ — балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$T_в$ — строк корисного використання обладнання, програмних засобів, приміщень тощо, років;

$t_{вик}$ — термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців.

Таблиця 5.10 — Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Комп'ютер	40000	2	2	3333,33
Смартфон для тестування	5000	2	1	208,33
Всього				3541,67

До статті «Паливо та енергія для науково-виробничих цілей» відносяться витрати на всі види палива й енергії, що безпосередньо використовуються з технологічною метою на проведення досліджень (5.14).

$$В_е = \sum_{i=1}^{n} \frac{W_{yt} \cdot t_i \cdot Ц_е \cdot К_{впі}}{\eta_i}, \quad (5.14)$$

де W_{yt} — встановлена потужність обладнання на певному етапі розробки, кВт;

t_i — тривалість роботи обладнання на етапі дослідження, год;

$Ц_е$ — вартість 1 кВт-години електроенергії, грн;

$К_{впі}$ — коефіцієнт, що враховує використання потужності, $К_{впі} < 1$;

η_i — коефіцієнт корисної дії обладнання, $\eta_i < 1$.

Для написання магістерської роботи використовується персональний комп'ютер для якого розрахуємо витрати на електроенергію.

$$В_е = \frac{0{,}5 \cdot 220 \cdot 12{,}69 \cdot 0{,}5}{0{,}8} = 872{,}44$$

Витрати за статтею «Службові відрядження» розраховуються як 20…25% від суми основної заробітної плати дослідників та робітників (5.15).

$$В_{св} = (З_о + З_р) * \frac{Н_{св}}{100\%}, \quad (5.15)$$

де $Н_{св}$ — норма нарахування за статтею «Службові відрядження».

$$В_{св} = 0{,}2 * (32381 + 17695{,}2) = 10015{,}24$$

Накладні (загальновиробничі) витрати $В_{нзв}$ охоплюють: витрати на управління організацією, оплата службових відряджень, витрати на утримання, ремонт та експлуатацію основних засобів, витрати на опалення, освітлення, водопостачання, охорону праці тощо (5.16). Накладні (загальновиробничі) витрати

$В_{нзв}$ можна прийняти як (100…150)% від суми основної заробітної плати розробників та робітників, які виконували дану МКНР, тобто:

$$В_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (5.16)$$

де $Н_{НЗВ}$ — норма нарахування за статтею «Інші витрати».

$$В_{нзв} = (32381 + 17695{,}2) \cdot \frac{100}{100\%} = 50076{,}19 \text{ грн}$$

Сума всіх попередніх статей витрат дає витрати, які безпосередньо стосуються даного розділу МКНР:

$$В = 32381+17695{,}2+5508{,}38+12228{,}6+625{,}3+0+3541{,}67+872{,}44+$$
$$10015{,}24+50076{,}19 = 132971{,}01 \text{ грн}$$

Прогнозування загальних втрат ЗВ на виконання та впровадження результатів виконаної МКНР здійснюється за формулою (5.17):

$$ЗВ = \frac{В}{\eta}, \quad (5.17)$$

де η — коефіцієнт, який характеризує стадію виконання даної НДР.

Оскільки робота знаходиться на стадії науково-дослідних робіт, то коефіцієнт $\beta = 0{,}7$. Звідси:

$$ЗВ = \frac{132971{,}01}{0{,}7} = 189958{,}58 \text{ грн.}$$

5.3 Розрахунок економічної ефективності науково-технічної розробки

Кількісно спрогнозуємо, яку вигоду, зиск можна отримати у майбутньому від впровадження результатів виконаної наукової роботи. Розрахуємо збільшення

чистого прибутку підприємства $\Delta П_i$, для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, за формулою (5.18):

$$\Delta П_i = \sum_{1}^{n} \left(\Delta Ц_o \cdot N + Ц_o \cdot \Delta N \right)_i \cdot \lambda \cdot \rho \cdot \left(1 - \frac{\upsilon}{100} \right), \quad (5.18)$$

де $\Delta Ц_o$ — покращення основного оціночного показника від впровадження результатів розробки у даному році;

N — основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN — покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки;

$Ц_o$ — основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n — кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки:

λ — коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість дорівнює 20%, а коефіцієнт $\lambda = 0{,}8333$;

ρ — коефіцієнт, який враховує рентабельність продукту. $\rho = 0{,}25$;

υ — ставка податку на прибуток. У 2025 році — 18%.

Припустимо, що ціна зросте на 500 грн. Кількість одиниць реалізованої продукції також збільшиться: протягом першого року на 200 шт., протягом другого року — на 250 шт., протягом третього року на 350 шт. Реалізація продукції до впровадження розробки складала 1 шт., а її ціна до 2000 грн. Розрахуємо прибуток, яке отримає підприємство протягом трьох років.

$$\Delta П_1 = [20 \cdot 1 + (60 + 20) \cdot 12000] \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 + \frac{18}{100}\right) = 163996{,}86 \text{ грн.}$$

$$\Delta П_2 = [20 \cdot 1 + (60 + 20) \cdot (12000 + 13000)] \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 + \frac{18}{100}\right) = 341673 \text{ грн.}$$

$$\Delta П_З = [20 \cdot 1 + (60 + 20) \cdot (12000 + 13000 + 14000)] \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 + \frac{18}{100}\right) = 532998{,}68 \text{ грн.}$$

5.4 Розрахунок ефективності вкладених інвестицій та періоду їх окупності

Розрахуємо основні показники, які визначають доцільність фінансування наукової розробки певним інвестором, є абсолютна і відносна ефективність вкладених інвестицій та термін їх окупності.

Розрахуємо величину початкових інвестицій *PV*, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки (5.19).

$$PV = k_{\text{інв}} \cdot \text{ЗВ}, \tag{5.19}$$

де $k_{інв}$ — 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 ($k_{інв}$= 2…5).

$$PV = 2 \cdot 189958{,}58 = 379917{,}17$$

Розрахуємо абсолютну ефективність вкладених інвестицій $E_{абс}$ згідно формули (5.20):

$$E_{абс} = (ПП - PV), \tag{5.20}$$

де *ПП* — приведена вартість всіх чистих прибутків, що їх отримає підприємство від реалізації результатів наукової розробки, грн.

$$ПП = \sum_{1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \tag{5.21}$$

де $\Delta П_i$ — збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДЦКР, грн.;

T — період часу, протягом якою виявляються результати впровадженої НДДКР, роки;

τ — ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,2.

$$ПП = \frac{163996,86}{(1+0,2)^1} + \frac{341673}{(1+0,2)^2} + \frac{532998,68}{(1+0,2)^3} = 683819,92 \text{ грн.}$$

$$Е_{абс} = (683819,92 - 379917,17) = 303902,74 \text{ грн.}$$

Оскільки $E_{абс}>0$ то вкладання коштів на виконання та впровадження результатів НДДКР може бути доцільним.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$. Для цього користуються формулою (5.22):

$$E_в = \sqrt[T_ж]{1+\frac{E_{абс}}{PV}} - 1, \quad (5.22)$$

де $T_ж$ — життєвий цикл наукової розробки, роки.

$$Е_в = \sqrt[3]{1+\frac{303902,74}{379917,17}} - 1 = 0,38 = 38\%$$

Визначимо мінімальну ставку дисконтування, яка у загальному вигляді визначається за формулою (5.23):

$$\tau = d + f, \quad (5.23)$$

де d — середньозважена ставка за депозитними операціями в комерційних банках; в 2025 році в Україні d = (0,14…0,2);

f — показник, що характеризує ризикованість вкладень; зазвичай, величина *f* = (0,05...0,1).

$$\tau_{\min} = 0,18 + 0,05 = 0,23.$$

Так як $E_в > \tau_{min}$, то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Розрахуємо термін окупності вкладених у реалізацію наукового проекту інвестицій за формулою (5.24):

$$T_{ок} = \frac{1}{E_в}, \tag{5.24}$$

$$\mathrm{T_{ок}} = \frac{1}{0,38} = 2,7 \text{ роки}$$

Так як $T_{ок} \leq$ 3...5-ти років, то фінансування даної наукової розробки є доцільним.

Результати здійсненого технологічного аудиту вказують на рівень вище середнього комерційного потенціалу. У порівнянні з аналогічним виробом виявлено, що нова розробка вищої якості і більш конкурентоспроможна, як з технічних, так і економічних позначень.

Вкладені інвестиції в даний проєкт окупляться через 2,7 роки. Загальні витрати складають 189958,58грн. Прогнозований прибуток за три роки складе 683819,92грн.

ВИСНОВКИ

У магістерській кваліфікаційній роботі розроблено комп’ютерну систему планування та оцінювання навчальної діяльності студента, орієнтовану на підтримку дистанційного та змішаного навчання, автоматизацію роботи викладача та підвищення об’єктивності перевірки завдань.

Проведено аналіз підходів до організації планування та оцінювання навчальної діяльності, класифіковано традиційні, автоматизовані, адаптивні та комбіновані методи контролю, визначено їхні переваги та обмеження, що дало змогу сформувати вимоги до функцій та ступінь автоматизації системи.

Досліджено сучасні моделі автоматизованого оцінювання текстових завдань, проаналізовано методи векторизації та метрики семантичної подібності, на основі чого було обрано для використання семантичне порівняння відповідей студентів з еталоном як базовий механізм оцінювання у системі.

Виконано порівняльний аналіз платформ аналогів, виявлено їхні функціональні можливості й обмеження щодо підтримки автоматизованого семантичного оцінювання відкритих відповідей. На цій основі доведено доцільність розробки спеціалізованої системи, яка інтегрує модуль семантичного оцінювання.

Запропоновано структуру системи планування та оцінювання навчальної діяльності студента на основі трирівневої архітектури, що включає мобільний користувацький інтерфейс, серверну частину та базу даних. Виділено функціональні модулі керування користувачами, завданнями, реєстрацією відповідей, оцінюванням та роботою з файлами.

Розроблено математичну модель процесу оцінювання завдань на основі семантичної подібності між еталонною та студентською відповідями. У модель включено часовий штраф за порушення термінів здачі та нормування одержаного значення до єдиної шкали оцінювання.

Здійснено програмну реалізацію системи. Створено кросплатформений мобільний застосунок на базі платформи Expo та фреймворку React Native мовою

JavaScript. Серверна частина системи забезпечує приймання відповідей, взаємодію з базою даних, виконання алгоритмів семантичного оцінювання й застосування часових штрафів.

Проведено тестування програмного забезпечення, перевірено коректність реалізації основних функцій у типових сценаріях роботи студента й викладача. Розроблено інструкцію користувача.

Виконано економічне обґрунтування проєкту, оцінено витрати на розробку програмного продукту та показники ефективності його впровадження. Отримані результати підтверджують доцільність використання розробленої системи в закладах вищої освіти.

Наукова новизна роботи полягає у поєднанні функцій планування та обліку результатів навчальних завдань із модулем автоматизованого семантичного оцінювання відкритих відповідей у межах єдиної комп’ютерної системи, а також у використанні гібридного підходу, що інтегрує автоматичний аналіз із вибірковим контролем викладача.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Івасюк В.В., Снігур А.В., Циркун В.В. Комп’ютерна система планування та оцінювання навчальної діяльності студента [Електронний ресурс] — Режим доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2026/paper/view/26272

2. Sukmaindrayana, A., Yulianeu, A. The Roles of Information and Communication Technology (ICT) in Transforming Students Learning Outcomes at STMIK DCI Tasikmalaya. 2022. — 89-104.

3. Parveen, Dr & Ramsan, Shaikh. The Role of Digital Technologies in Education: Benefits and Challenges. International Research Journal on Advanced Engineering and Management. 2024. — 2029-2037.

4. Sharif H., Atif A. The Evolving Classroom: How Learning Analytics Is Shaping the Future of Education and Feedback Mechanisms. 2024. 176.

5. Manually Grading vs Automated Grading in the Classroom [Електронний ресурс] — Режим доступу: https://gradedplus.app/manually-grading-vs-automated-grading-in-the-classroom/

6. Traditional vs Automated Assessment: Key Differences [Електронний ресурс] — Режим доступу: https://qorrectassess.com/en/blog/traditional-vs-automated-assessment

7. Jaccard Similarity [Електронний ресурс] — Режим доступу: https://www.geeksforgeeks.org/python/jaccard-similarity

8. Mohler M., Mihalcea R. Text-to-text Semantic Similarity for Automatic Short Answer Grading. 2009. 567–575.

9. What is latent semantic analysis? [Електронний ресурс] — Режим доступу: https://www.ibm.com/think/topics/latent-semantic-analysis

10. Understanding Word2Vec: CBOW & Skip-Gram [Електронний ресурс] — Режим доступу: https://www.kaggle.com/code/ftaham/understanding-word2vec-cbow-skip-gram

11. BERT Model — NLP [Електронний ресурс] — Режим доступу: https://www.geeksforgeeks.org/nlp/explanation-of-bert-model-nlp/

12. Similarity Learning with Siamese Networks [Електронний ресурс] — Режим доступу: https://www.mygreatlearning.com/blog/siamese-networks/

13. The Different Types of Mobile Apps: Native, Hybrid, PWA [Електронний ресурс] — Режим доступу: https://swovo.com/blog/the-different-types-of-mobile-apps-native-hybrid-pwa

14. Native vs. Cross Platform Apps [Електронний ресурс] — Режим доступу: https://www.microsoft.com/en-us/power-platform/products/power-apps/topics/app-development/native-vs-cross-platform-apps

15. Core Components and Native Components. React Native [Електронний ресурс] — Режим доступу: https://reactnative.dev/docs/intro-react-native-components

16. Expo [Електронний ресурс] — Режим доступу: https://docs.expo.dev/

17. Flutter [Електронний ресурс] — Режим доступу: https://flutter.dev/

18. Django overview [Електронний ресурс] — Режим доступу: https://www.djangoproject.com/start/overview/

19. FastAPI [Електронний ресурс] — Режим доступу: https://fastapi.tiangolo.com/

20. NodeJS. Overview of Blocking vs Non-Blocking [Електронний ресурс] — Режим доступу: https://nodejs.org/en/learn/asynchronous-work/overview-of-blocking-vs-non-blocking

21. Express [Електронний ресурс] — Режим доступу: https://expressjs.com/

22. NestJS. Modules [Електронний ресурс] — Режим доступу: https://docs.nestjs.com/modules

23. PostgreSQL: About [Електронний ресурс] — Режим доступу: https://www.postgresql.org/about

24. Type ORM. Getting Started [Електронний ресурс] — Режим доступу: https://typeorm.io/docs/getting-started

25. 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

Технічне завдання

Міністерство освіти і науки України
Вінницький національних технічний університет
Факультет інформаційних технологій та комп’ютерної інженерії
Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
проф., д.т.н.. Азаров О.Д.

“ 03 ” 10 2025р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи
“Комп’ютерна система планування та оцінювання навчальної діяльності студента”

Науковий керівник: доцент к.т.н.
_____ Снігур А.В.
Студент групи 1КІ-24м
_____ Івасюк В.В

ВНТУ 2025

1 Підстава для виконання магістерської кваліфікаційної роботи (МКР)

1.1 Актуальність дослідження магістерської роботи обґрунтовується тим, що у закладах вищої освіти традиційні методи планування навчального навантаження і оцінювання відкритих текстових відповідей не забезпечують необхідної швидкості перевірки, що призводить до затримок у наданні зворотного зв’язку та суб’єктивних відмінностей у виставленні оцінок. Зростання обсягів навчальних матеріалів і кількості студентських робіт ускладнює організацію контролю знань, що вимагає застосування засобів для аналізу змісту відповідей, стандартизації критеріїв оцінювання та процедури узгодження автоматичних і експертних оцінок. Система дозволить зменшити навантаження на викладачів, покращити час і об'єктивність оцінювання та допоможе студентам слідкувати та своєчасно виконувати навчальні завдання.

1.2 Наказ про затвердження теми МКР.

2 Мета МКР і призначення розробки

2.1 Мета роботи — аналіз сучасного стану і тенденцій розвитку розробок у сфері організації навчання та розробка системи планування та оцінювання навчальної діяльності студента шляхом порівняння відповідей з еталоном;

2.2 Призначення розробки — система планування та оцінювання навчальної діяльності студента призначена для організації видачі та виконання навчальних завдань, перевірки відкритих відповідей з використанням автоматизованих методів аналізу та фіксації отриманих результатів з подальшим інформуванням студента про оцінку.

3 Вихідні дані для виконання МКР

3.1 Проведення аналізу існуючих методів оцінювання навчальної діяльності студента.

3.2 Моделювання та проєктування системи планування та оцінювання навчальної діяльності студента.

3.3 Програмна реалізація системи планування та оцінювання навчальної діяльності студента.

3.4 Тестування розробленої системи.

3.5 Виконання розрахунків для доведення доцільності нової розробки з економічної точки зору.

4 Вимоги до виконання МКР

Головна вимога — використати, як основний, метод оцінювання шляхом порівняння отриманої відповіді з еталоном використовуючи семантичну подібність векторів.

5 Етапи МКР та очікувані результати

Етапи роботи та очікувані результати приведено в Таблиці А.1.

Таблиця А.1 — Етапи МКР

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Аналіз сучасного стану досліджень	25.09.2025	01.10.2025	Аналітичний огляд літературних джерел, задачі досліджень
2	Аналітичний огляд існуючих рішень для організації планування та оцінювання навчальної діяльності студента	02.10.2025	08.10.2025	Розділ 1
3	Моделювання та проектування системи	09.10.2025	16.10.2025	Розділ 2
4	Практична реалізація та опис роботи системи	17.10.2025	25.10.2025	Розділ 3, програмний продукт
5	Тестування роботи і написання інструкції користувача	26.10.2025	30.10.2025	Розділ 4
6	Підготовка економічної частини	01.11.2025	04.11.2025	Розділ 5
7	Апробація та впровадження результатів дослідження	05.11.2025	12.11.2025	Тези доповідей

8	Оформлення пояснювальної записки, презентації	13.11.2025	24.11.2025	ПЗ, графічний матеріал і презентація
9	Підготовка і підпис супроводжуючих документів, нормоконтроль та тест на плагіат	25.11.2025	03.12.2025	Оформлені документи

6 Матеріали, що подаються до захисту МКР

До захисту подаються: пояснювальна записка МКР, графічні і ілюстративні матеріали, протокол попереднього захисту МКР на кафедрі, відгук наукового керівника, відгук опонента, анотації до МКР українською та іноземною мовами.

7 Порядок контролю виконання та захисту МКР

Виконання етапів графічної та розрахункової документації МКР контролюється науковим керівником згідно зі встановленими термінами. Захист МКР відбувається на засіданні Екзаменаційної комісії, затвердженої наказом ректора.

8 Вимоги до оформлювання та порядок виконання МКР

8.1 При оформлюванні МКР використовуються:

— ДСТУ 3008: 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302: 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— ГОСТ 2.104-2006 «Єдина система конструкторської документації. Основні написи»;

— методичні вказівки до виконання магістерських кваліфікаційних робіт зі спеціальності 123 — «Комп’ютерна інженерія»;

— документи на які посилаються у вище вказаних.

8.2 Порядок виконання МКР викладено в «Положення про кваліфікаційні роботи на другому (магістерському) рівні вищої освіти СУЯ ВНТУ-03.02.02-П.001.01:21».

ДОДАТОК Б

Протокол перевірки кваліфікаційної роботи

Назва роботи: ___Комп'ютерна система планування та оцінювання навчальної діяльності студента_____

Тип роботи: _____магістерська кваліфікаційна робота_____
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ _____кафедра обчислювальної техніки, ФІТКІ, 1КІ-24м_____
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism (КП1) __1__%

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

✓ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак академічного плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту.

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки академічного плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень. Робота до захисту не приймається.

Експертна комісія:

_____Азаров О.Д., д.т.н., зав. каф. ОТ_____ _____
(прізвище, ініціали, посада) (підпис)

_____Мартинюк Т.Б., д.т.н, проф. каф. ОТ_____ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ ____Захарченко С.М.___
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ _____к.т.н., доц. каф. ОТ Снігур А.В._____
(підпис) (прізвище, ініціали, посада)

Здобувач _____ _____Івасюк В.В._____
(підпис) (прізвище, ініціали)

ДОДАТОК В

Алгоритм процесу обліку результатів навчальної діяльності

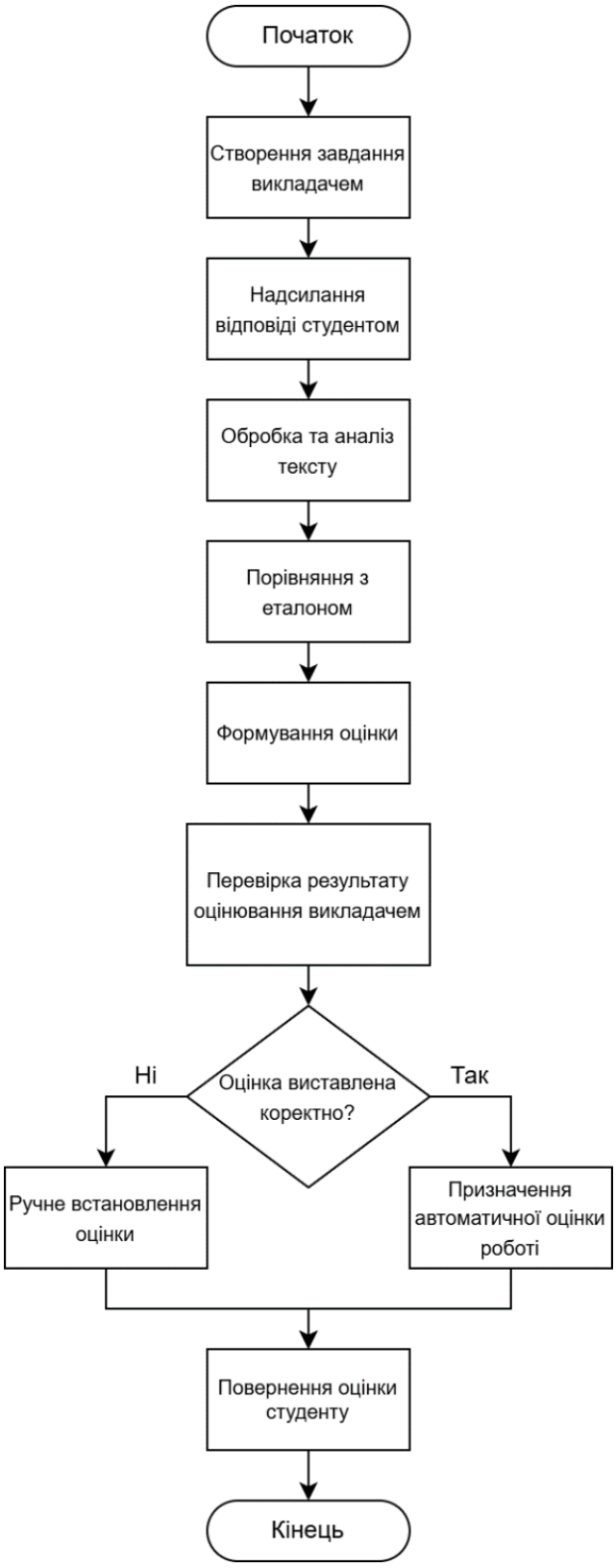

Рисунок В.1 — Алгоритм процесу обліку результатів навчальної діяльності

ДОДАТОК Г

Лістинг сторінки перегляду та оцінювання відповідей

```
import { zodResolver } from "@hookform/resolvers/zod";
import { useMemo, useState, useEffect } from "react";
import { Controller, useForm } from "react-hook-form";
import {
  Alert,
  FlatList,
  Modal,
  StyleSheet,
  Text,
  TextInput,
  View,
  ActivityIndicator,
} from "react-native";
import { z } from "zod";
import api from "../api/client";
import { Button } from "../components/Button/Button";
import { Pill } from "../components/Pill/Pill";
import { Pressable } from "../components/Pressable/Pressable";
import { theme } from "../styles/theme";
import { formatDateDisplay } from "../utils/formatDate";

const scoreSchema = z.object({
  score: z.coerce
    .number({ message: "Обов'язкове поле" })
    .min(0, { message: "Введіть число від 0 до 100" })
    .max(100, { message: "Введіть число від 0 до 100" }),
});

export default function TeacherTaskDetailsScreen({ route }) {
  const { taskId } = route.params;
  const [task, setTask] = useState(null);
  const [submissions, setSubmissions] = useState([]);
  const [loading, setLoading] = useState(true);
  const [modalVisible, setModalVisible] = useState(false);
  const [selectedSubmission, setSelectedSubmission] = useState(null);
  const [filter, setFilter] = useState("all");
  const {
    control,
    handleSubmit,
    reset,
    formState: { errors },
  } = useForm({
    defaultValues: { score: undefined },
    resolver: zodResolver(scoreSchema),
    mode: "onSubmit",
  });
  useEffect(() => {
```

```
  const loadData = async () => {
    try {
      setLoading(true);
      const taskRes = await api.getTaskById(taskId);
      const subsRes = await api.getTaskSubmissions(taskId);
      setTask(taskRes.data ?? null);
      setSubmissions(subsRes.data ?? []);
    } catch (e) {
      console.log("Error loading task details:", e);
    } finally {
      setLoading(false);
    }
  };
  loadData();
}, [taskId]);

const filterDefs = [
  { v: "all", l: "Усі" },
  { v: "pending", l: "Очікують" },
  { v: "graded", l: "Оцінено" },
];

const filteredSubmissions = useMemo(() => {
  if (filter === "all") return submissions;
  return submissions.filter((s) => s.status === filter);
}, [filter, submissions]);

const statusLabelMap = {
  draft: "Чернетка",
  active: "Активне",
  review: "Очікує перевірки",
  completed: "Завершено",
};

const openEvaluateModal = (submission) => {
  setSelectedSubmission(submission);
  setModalVisible(true);
  reset({ score: submission.finalScore ?? undefined });
};

const closeEvaluateModal = () => {
  setModalVisible(false);
  setSelectedSubmission(null);
  reset({ score: undefined });
};

const handleConfirmAutoScore = async () => {
  if (!selectedSubmission || selectedSubmission.autoScore == null) return;

  try {
    await api.scoreSubmission({
      submissionId: selectedSubmission.id,
```

```
      score: selectedSubmission.autoScore,
      source: "auto",
    });

    setSubmissions((prev) =>
      prev.map((s) =>
        s.id === selectedSubmission.id
          ? { ...s, status: "graded", finalScore: selectedSubmission.autoScore }
          : s
      )
    );

    closeEvaluateModal();
  } catch (e) {
    Alert.alert("Помилка", "Не вдалося зберегти оцінку");
  }
};

const handleSaveManualScore = handleSubmit(async ({ score }) => {
  if (!selectedSubmission) return;

  const rounded = Math.round(score);

  try {
    await api.scoreSubmission({
      submissionId: selectedSubmission.id,
      score: rounded,
      source: "manual",
    });

    setSubmissions((prev) =>
      prev.map((s) =>
        s.id === selectedSubmission.id
          ? { ...s, status: "graded", finalScore: rounded }
          : s
      )
    );

    closeEvaluateModal();
  } catch (e) {
    Alert.alert("Помилка", "Не вдалося зберегти оцінку");
  }
});

if (loading) {
  return (
    <View style={styles.loaderWrap}>
      <ActivityIndicator size="large" color={theme.accentColor} />
    </View>
  );
}
```

```
return (
  <View style={styles.container}>
    <View style={styles.headerCard}>
      <View style={styles.headerTop}>
        <View style={[styles.statusPill, styles[`status_${task.status}`]]}>
          <Text style={styles.statusText}>{statusLabelMap[task.status]}</Text>
        </View>
        <Text style={styles.deadline}>
          Термін здачі:{" "}
          <Text style={styles.deadlineValue}>
            {formatDateDisplay(task.deadline)}
          </Text>
        </Text>
      </View>
      <Text style={styles.title}>{task.title}</Text>
      <Text style={styles.course}>{task.course}</Text>
      <View style={styles.actions}>
        <Button title="Редагувати" type="secondary" />
        <Button title="Закрити завдання" />
      </View>
    </View>
    <View style={styles.filters}>
      {filterDefs.map((d) => (
        <Pill
          key={d.v}
          label={d.l}
          active={filter === d.v}
          onPress={() => setFilter(d.v)}
        />
      ))}
    </View>
    <FlatList
      data={filteredSubmissions}
      keyExtractor={(s) => s.id}
      contentContainerStyle={styles.listContent}
      ItemSeparatorComponent={() => <View style={{ height: 10 }} />}
      renderItem={({ item }) => (
        <Pressable
          containerStyle={styles.cardContainer}
          pressableStyle={styles.card}
          rippleColor="accent"
        >
          <View style={styles.cardRow}>
            <Text style={styles.student}>{item.student}</Text>
            <View style={styles.cardRight}>
              <View
                style={[
                  styles.subStatusPill,
                  item.status === "graded"
                    ? styles.subStatusGraded
                    : styles.subStatusPending,
                ]}
```

```
          >
            <Text style={styles.subStatusText}>
              {item.status === "graded" ? "Оцінено" : "Очікує"}
            </Text>
          </View>
          {item.status === "graded" && (
            <Text style={styles.headerScore}>{item.finalScore}/100</Text>
          )}
        </View>
      </View>
      <Text style={styles.meta}>
        Здано: {formatDateDisplay(item.submittedAt)}
      </Text>
      {item.status === "pending" && item.autoScore != null && (
        <Text style={styles.meta}>
          Автоматична оцінка:{" "}
          <Text style={styles.autoScore}>{item.autoScore}/100</Text>
        </Text>
      )}
      <View style={styles.rowActions}>
        <Button
          title="Переглянути"
          type="secondary"
          dense
          onPress={() => {
            Alert.alert(
              "Перегляд роботи",
              `Студент: ${item.student}\nДата: ${formatDateDisplay(
                item.submittedAt
              )}`
            );
          }}
        />
        <Button
          title={item.status === "pending" ? "Оцінити" : "Змінити оцінку"}
          dense
          onPress={() => openEvaluateModal(item)}
        />
      </View>
    </Pressable>
  )}
/>
<Modal
  visible={modalVisible}
  transparent
  animationType="fade"
  onRequestClose={closeEvaluateModal}
>
  <View style={styles.modalBackdrop}>
    <View style={styles.modalCard}>
      <Text style={styles.modalTitle}>Оцінювання роботи</Text>
      {selectedSubmission && (
```

```
<>
 <Text style={styles.modalText}>
  Студент: {selectedSubmission.student}
 </Text>
 {selectedSubmission.autoScore != null && (
  <Text style={styles.modalText}>
   Автоматична оцінка:
   <Text style={styles.modalScore}>
    {selectedSubmission.autoScore}/100
   </Text>
  </Text>
 )}
 <Text style={styles.modalText}>Або введіть свою (0–100):</Text>
 <Controller
  control={control}
  name="score"
  render={({ field: { onChange, onBlur, value } }) => (
   <TextInput
    keyboardType="number-pad"
    placeholder="Напр., 87"
    value={value === undefined ? "" : String(value)}
    onChangeText={onChange}
    onBlur={onBlur}
    style={[
     styles.modalInput,
     errors.score && styles.modalInputError,
    ]}
   />
  )}
 />
 {errors.score && (
  <Text style={styles.modalError}>{errors.score.message}</Text>
 )}
 <View style={styles.modalActions}>
  <Button
   title="Скасувати"
   type="secondary"
   dense
   onPress={closeEvaluateModal}
  />
  {selectedSubmission.autoScore != null && (
   <Button
    title="Підтвердити автооцінку"
    dense
    onPress={handleConfirmAutoScore}
   />
  )}
  <Button
   title="Зберегти оцінку"
   dense
   onPress={handleSaveManualScore}
  />
```

```
            </View>
          </>
        )}
      </View>
    </View>
  </Modal>
 </View>
 );
}
const styles = StyleSheet.create({
 container: {
  flex: 1,
  padding: 12,
  gap: 12,
  backgroundColor: theme.backgroundColor,
 },
 loaderWrap: { flex: 1, justifyContent: "center", alignItems: "center" },
 headerCard: {
  backgroundColor: theme.white,
  borderRadius: 12,
  padding: 12,
  gap: 8,
  borderWidth: 1,
  borderColor: theme.colorLines,
 },
 headerTop: {
  flexDirection: "row",
  justifyContent: "space-between",
  alignItems: "center",
 },
 title: { fontSize: 18, fontWeight: "700", lineHeight: 22 },
 course: { color: theme.accentColor, fontWeight: "600" },
 deadline: { color: theme.textSecondary },
 deadlineValue: { color: theme.textPrimary, fontWeight: "600" },
 actions: { flexDirection: "row", gap: 8 },
 statusPill: { paddingHorizontal: 10, paddingVertical: 4, borderRadius: 999 },
 statusText: { fontSize: 12, fontWeight: "600", color: theme.white },
 status_draft: { backgroundColor: theme.statusAssigned },
 status_active: { backgroundColor: theme.accentColor },
 status_review: { backgroundColor: theme.statusPending },
 status_completed: { backgroundColor: theme.statusGraded },
 filters: { flexDirection: "row", gap: 6, flexWrap: "wrap" },
 listContent: { paddingBottom: 28 },
 cardContainer: { borderRadius: 12, overflow: "hidden" },
 card: {
  backgroundColor: theme.white,
  borderRadius: 12,
  padding: 12,
  gap: 6,
  borderWidth: 1,
  borderColor: theme.colorLines,
 },
```

```
  cardRow: {
    flexDirection: "row",
    justifyContent: "space-between",
    alignItems: "center",
  },
  cardRight: { flexDirection: "row", alignItems: "center", gap: 8 },
  student: { fontSize: 15, fontWeight: "600" },
  meta: { color: theme.textSecondary, fontSize: 13 },
  autoScore: { fontWeight: "700", color: theme.textPrimary },
  rowActions: { flexDirection: "row", gap: 8 },
  subStatusPill: {
    paddingHorizontal: 10,
    paddingVertical: 4,
    borderRadius: 999,
  },
  subStatusPending: { backgroundColor: theme.statusPendingLight },
  subStatusGraded: { backgroundColor: theme.statusGraded },
  subStatusText: { color: theme.white, fontWeight: "600", fontSize: 12 },
  headerScore: { fontSize: 18, fontWeight: "800", color: theme.textPrimary },
  modalBackdrop: {
    flex: 1,
    backgroundColor: theme.overlayBlack35,
    justifyContent: "center",
    alignItems: "center",
    padding: 16,
  },
  modalCard: {
    width: "100%",
    borderRadius: 12,
    backgroundColor: theme.white,
    padding: 16,
    gap: 8,
  },
  modalTitle: { fontSize: 18, fontWeight: "700" },
  modalText: { color: theme.textPrimary },
  modalScore: { fontWeight: "800" },
  modalInput: {
    borderWidth: 1,
    borderColor: theme.colorLines,
    borderRadius: 8,
    paddingVertical: 8,
    paddingHorizontal: 10,
  },
  modalInputError: { borderColor: theme.red },
  modalError: { color: theme.red, marginTop: 4 },
  modalActions: {
    flexDirection: "row",
    flexWrap: "wrap",
    gap: 8,
    justifyContent: "flex-end",
  },
});
```

ДОДАТОК Д

Лістинг екрана створення нового завдання

```
import { Controller, useForm } from "react-hook-form";
import { useState } from "react";
import { z } from "zod";
import { zodResolver } from "@hookform/resolvers/zod";
import { Alert, ScrollView, StyleSheet, Text, View } from "react-native";
import * as DocumentPicker from "expo-document-picker";
import { Button } from "../components/Button/Button";
import { Input } from "../components/Input/Input";
import { theme } from "../styles/theme";
import { formatDateDisplay } from "../utils/formatDate";
import api from "../api/client";
const schema = z.object({
  course: z.string().min(1, { message: "Обов'язкове поле" }),
  title: z.string().min(1, { message: "Обов'язкове поле" }),
  body: z.string().min(1, { message: "Обов'язкове поле" }),
  deadline: z
    .string()
    .optional()
    .transform((v) => (v ?? "").trim())
    .refine((v) => v === "" || /^\d{4}-\d{2}-\d{2}$/.test(v) || /^\d{2}\.\d{2}\.\d{4}$/.test(v), {
      message: "Формат ДД.ММ.РРРР або YYYY-MM-DD",
    }),
  latePenalty: z
    .string()
    .transform((v) => (v ?? "").trim())
    .refine((v) => v === "" || /^\d{1,2}$/.test(v), { message: "0-99" })
    .optional(),
});
export const TeacherTaskCreateScreen = () => {
  const { control, handleSubmit, formState: { errors }, reset } = useForm({
    resolver: zodResolver(schema),
    defaultValues: {
      course: "",
      title: "",
      body: "",
      deadline: "",
      latePenalty: "",
    },
    mode: "onSubmit",
  });
  const onSubmit = async (data) => {
    const res = await api.createTask(data);
    if (!res) return;
    Alert.alert(
      "Завдання створено",
```

```
      `Курс: ${data.course}\nНазва: ${data.title}\nТермін здачі: ${data.deadline ?
formatDateDisplay(data.deadline) : "—"}${referenceAsset ? "\nЕталон: " + referenceAsset.name :
""}`,
      [{ text: "OK", onPress: () => reset() }]
    );
  };
  const [referenceAsset, setReferenceAsset] = useState<DocumentPicker.DocumentPickerAsset |
null>(null);
  const pickReference = async () => {
    try {
      const res = await DocumentPicker.getDocumentAsync({
        multiple: false,
        type: [
          "application/pdf",
          "text/plain",
          "application/msword",
          "application/vnd.openxmlformats-officedocument.wordprocessingml.document",
          "image/*",
        ],
        copyToCacheDirectory: true,
      });
      if (!res.canceled && res.assets?.length) {
        setReferenceAsset(res.assets[0]);
      }
    } catch (e) {
      console.log("reference file pick error", e);
    }
  };
  const removeReference = () => setReferenceAsset(null);
  return (
    <ScrollView contentContainerStyle={styles.container}>
      <View style={styles.card}>
        <Controller
          control={control}
          name="course"
          render={({ field: { onChange, onBlur, value } }) => (
            <Input label="Курс / предмет" placeholder="Алгоритми та структури даних" value={value}
onChangeText={onChange} onBlur={onBlur} errorMessage={errors.course?.message} />
          )}
        />
        <Controller
          control={control}
          name="title"
          render={({ field: { onChange, onBlur, value } }) => (
            <Input label="Назва" placeholder="Лабораторна робота №3" value={value}
onChangeText={onChange} onBlur={onBlur} errorMessage={errors.title?.message} />
          )}
        />
        <Controller
          control={control}
          name="body"
          render={({ field: { onChange, onBlur, value } }) => (
```

```
          <Input label="Опис" placeholder="Опишіть завдання..." value={value}
onChangeText={onChange} onBlur={onBlur} errorMessage={errors.body?.message} multiline
numberOfLines={6} style={styles.multiline} />
        )}
       />
       <Controller
        control={control}
        name="deadline"
        render={({ field: { onChange, onBlur, value } }) => (
         <Input label="Термін здачі (ДД.ММ.РРРР)" placeholder="01.05.2025" value={value}
onChangeText={onChange} onBlur={onBlur} errorMessage={errors.deadline?.message} />
        )}
       />
       <Controller
        control={control}
        name="latePenalty"
        render={({ field: { onChange, onBlur, value } }) => (
         <Input label="Штраф за запізнення, % (необов'язково)" placeholder="Напр., 10"
value={value} onChangeText={onChange} onBlur={onBlur}
errorMessage={errors.latePenalty?.message} />
        )}
       />
       <View style={styles.refBlock}>
        <Text style={styles.refLabel}>Еталонна відповідь</Text>
        {referenceAsset ? (
         <View style={styles.refFileRow}>
          <Text style={styles.refFileName} numberOfLines={2}>{referenceAsset.name}</Text>
          <Button title="Видалити" type="secondary" dense onPress={removeReference} />
         </View>
        ) : (
         <Button title="Додати файл" type="secondary" onPress={pickReference} />
        )}
       </View>
      </View>
      <Button title="Зберегти як чернетку" type="secondary" size="large"
onPress={handleSubmit(onSubmit)} />
      <Button title="Опублікувати" size="large" onPress={handleSubmit(onSubmit)} />
    </ScrollView>
  );
};
const styles = StyleSheet.create({
  container: { padding: 12, gap: 12 },
  card: { backgroundColor: theme.white, borderRadius: 12, padding: 12, gap: 10, borderWidth: 1,
borderColor: theme.colorLines },
  multiline: { height: 140, textAlignVertical: "top" },
  switchRow: { flexDirection: "row", alignItems: "center", justifyContent: "space-between",
paddingVertical: 4 },
  switchLabel: { fontSize: 14 },
  refBlock: { gap: 8 },
  refLabel: { fontWeight: "600" },
  refFileRow: { flexDirection: "row", alignItems: "center", justifyContent: "space-between", gap: 8 },
});
```

ДОДАТОК Е

Лістинг екрана статистики студента

```
import { useMemo } from "react";
import { ScrollView, StyleSheet, Text, View } from "react-native";
import tasksData from "../constants/mockdata.json";
import { theme } from "../styles/theme";
import { formatDateDisplay } from "../utils/formatDate";
const parseDate = (str) => {
  if (/^\d{2}\.\d{2}\.\d{4}$/.test(str)) {
    const [d, m, y] = str.split(".").map(Number);
    return new Date(y, m - 1, d);
  }
  if (/^\d{4}-\d{2}-\d{2}$/.test(str)) {
    return new Date(str);
  }
  return null;
};

const ProgressBar = ({
  value,
  color = theme.accentColor,
}) => (
  <View style={styles.progressOuter}>
    <View
      style={[
        styles.progressInner,
        {
          width: `${Math.max(0, Math.min(100, value))}%`,
          backgroundColor: color,
        },
      ]}
    />
  </View>
);

export const StudentAnalyticsScreen = () => {
  const stats = useMemo(() => {
    const gradedList = tasksData.filter((t) => t.submitted && !!t.mark);
    const pendingList = tasksData.filter((t) => t.submitted && !t.mark);
    const assignedList = tasksData.filter((t) => !t.submitted);
    const total = tasksData.length;
    const graded = gradedList.length;
    const pending = pendingList.length;
    const assigned = assignedList.length;
    const avg = graded? Math.round(gradedList.reduce((s, t) => s + (t.mark || 0), 0) / graded) : 0;
    const byDate = assignedList
      .map(task => ({ task, date: parseDate(t.expirationDate) }))
      .filter((task) => task.date)
      .sort((a, b) => a.date.getTime() - b.date.getTime());
```

```
  const future = byDate.filter((task) => task.date.getTime() >= Date.now());
  const upcomingList = (future.length ? future : byDate)
    .slice(0, 3)
    .map((item) => item.task);

  const gradedPercent = total ? Math.round((graded / total) * 100) : 0;
  const pendingPercent = total ? Math.round((pending / total) * 100) : 0;
  const assignedPercent = total ? Math.round((assigned / total) * 100) : 0;

  const onTimeCount = gradedList.reduce((acc, t) => {
    const sd = parseDate(t.submittedAt);
    const dd = parseDate(t.expirationDate);
    if (sd && dd && sd.getTime() <= dd.getTime()) return acc + 1;
    return acc;
  }, 0);
  const onTimePercent = graded ? Math.round((onTimeCount / graded) * 100) : 0;

  return {
    graded,
    pending,
    assigned,
    avg,
    upcomingList,
    gradedPercent,
    pendingPercent,
    assignedPercent,
    onTimePercent,
  };
}, []);

return (
  <ScrollView style={styles.scroll} contentContainerStyle={styles.container}>
    <Text style={styles.title}>Моя статистика</Text>
    <View style={styles.cards}>
      <View style={styles.card}>
        <Text style={styles.value}>{stats.assigned}</Text>
        <Text style={styles.label}>Призначено</Text>
      </View>
      <View style={styles.card}>
        <Text style={styles.value}>{stats.pending}</Text>
        <Text style={styles.label}>Очікують</Text>
      </View>
      <View style={styles.card}>
        <Text style={styles.value}>{stats.graded}</Text>
        <Text style={styles.label}>Оцінено</Text>
      </View>
    </View>
    <View style={styles.avgCard}>
      <Text style={styles.avgLabel}>Середня оцінка</Text>
      <Text style={styles.avgValue}>{stats.avg}/100</Text>
    </View>
    <View style={styles.block}>
```

```
        <Text style={styles.blockTitle}>Розподіл статусів</Text>
        <View style={styles.row}>
          <Text style={styles.rowLabel}>Призначено</Text>
          <Text style={styles.rowValue}>{stats.assignedPercent}%</Text>
        </View>
        <ProgressBar
          value={stats.assignedPercent}
          color={theme.statusAssigned}
        />
        <View style={styles.row}>
          <Text style={styles.rowLabel}>Очікують</Text>
          <Text style={styles.rowValue}>{stats.pendingPercent}%</Text>
        </View>
        <ProgressBar value={stats.pendingPercent} color={theme.statusPending} />
        <View style={styles.row}>
          <Text style={styles.rowLabel}>Оцінено</Text>
          <Text style={styles.rowValue}>{stats.gradedPercent}%</Text>
        </View>
        <ProgressBar value={stats.gradedPercent} color={theme.statusGraded} />
      </View>
      <View style={styles.block}>
        <Text style={styles.blockTitle}>Вчасність</Text>
        <View style={styles.row}>
          <Text style={styles.rowLabel}>Здано вчасно</Text>
          <Text style={styles.rowValue}>{stats.onTimePercent}%</Text>
        </View>
        <ProgressBar value={stats.onTimePercent} color={theme.accentColor} />
        <Text style={[styles.label, { marginTop: 6 }]}>
          Розраховано за підтвердженими оцінками
        </Text>
      </View>
      {!!stats.upcomingList.length && (
        <View style={styles.block}>
          <Text style={styles.blockTitle}>Що зробити найближчим</Text>
          {stats.upcomingList.map((t, idx) => (
            <View key={`${t.title}-${idx}`} style={styles.topRow}>
              <Text style={styles.topTitle} numberOfLines={1}>
                {t.title}
              </Text>
              <Text style={styles.topMark}>
                {formatDateDisplay(t.expirationDate)}
              </Text>
            </View>
          ))}
        </View>
      )}
    </ScrollView>
  );
};
const styles = StyleSheet.create({
  scroll: { flex: 1 },
  container: { padding: 16, gap: 16 },
```

```
  title: { fontSize: 20, fontWeight: "700" },
  cards: { flexDirection: "row", gap: 10 },
  card: {
    flex: 1,
    backgroundColor: theme.white,
    borderRadius: 12,
    borderWidth: 1,
    borderColor: theme.colorLines,
    padding: 14,
    alignItems: "center",
  },
  value: { fontSize: 18, fontWeight: "800" },
  label: { color: theme.textSecondary, marginTop: 4 },
  avgCard: {
    backgroundColor: theme.white,
    borderRadius: 12,
    borderWidth: 1,
    borderColor: theme.colorLines,
    padding: 16,
    alignItems: "center",
  },
  avgLabel: { color: theme.textSecondary, marginBottom: 6 },
  avgValue: { fontSize: 24, fontWeight: "800" },
  block: {
    backgroundColor: theme.white,
    borderRadius: 12,
    borderWidth: 1,
    borderColor: theme.colorLines,
    padding: 16,
    gap: 8,
  },
  blockTitle: { fontWeight: "700" },
  row: { flexDirection: "row", justifyContent: "space-between", alignItems: "center" },
  rowLabel: { color: theme.textPrimary },
  rowValue: { fontWeight: "700" },
  progressOuter: {
    height: 8,
    backgroundColor: theme.colorLines,
    borderRadius: 999,
    overflow: "hidden",
  },
  progressInner: { height: 8, borderRadius: 999 },
  upTitle: { fontWeight: "600", marginTop: 4 },
  topRow: { flexDirection: "row" justifyContent: "space-between", alignItems: "center" },
  topTitle: { flex: 1, marginRight: 10 },
  topMark: { fontWeight: "800" },
});
```

ДОДАТОК Ж

Лістинг модуля оцінювання на серверній частині

```
import {
  BadRequestException,
  Injectable,
  Logger,
  ServiceUnavailableException
} from '@nestjs/common';
import { isAbsolute, join } from 'node:path';
import { extractText } from '../utils/text-extraction.util';
@Injectable()
export class EvaluationService {
  logger = new Logger(EvaluationService.name);
  transformersModule;
  featureExtractor;
  referenceCache = new Map();
  constructor() { }
  async onModuleInit() {
    await this.ensurePipeline();
    await this.embed('warmup');
    this.logger.log(`Evaluation model ready: ${MODEL_ID}`);
  }
  async autoEvaluate(submissionId) {
    const submission = await this.submissionsService.findById(submissionId);
    if (!submission.task?.referenceFileUrl) {
      throw new BadRequestException('Task does not have a reference file');
    }
    const referenceText = await this.resolveTextSource(
      { filePath: submission.task.referenceFileUrl },
      'reference',
    );
    const answerText = await this.extractSubmissionFilesText(submission);
    let score = await this.scoreTexts(referenceText, answerText);
    const due = submission.task?.dueDate;
    const submitted = submission.submittedAt;
    const penaltyPercent = submission.task?.latePenaltyPercent ?? 0;
    if (penaltyPercent > 0) {
      if (submitted.getTime() > due.getTime()) {
        const factor = Math.max(0, 1 - penaltyPercent / 100);
        score = Math.round(score * factor);
      }
    }
    await this.submissionsService.updateAutoScore(submissionId, score);
    return score;
  }
  async score(reference, answer) {
    const ref = await this.resolveTextSource(reference, 'reference');
    const ans = await this.resolveTextSource(answer, 'answer');
    return this.scoreTexts(ref, ans);
```

```
}
async setFinalScore(submissionId, finalScore) {
  return this.submissionsService.updateFinalScore(submissionId, finalScore);
}

async ensurePipeline() {
  if (this.featureExtractor) return this.featureExtractor;
  this.transformersModule = await import('@xenova/transformers');
  const pipeline = await this.transformersModule.pipeline(
    'feature-extraction',
    MODEL_ID,
    { quantized: true },
  );
  if (typeof pipeline !== 'function') {
    throw new ServiceUnavailableException(
      'Cannot initialize feature-extraction pipeline',
    );
  }
  this.featureExtractor = pipeline;
  return this.featureExtractor;
}
async embed(text) {
  const extractor = await this.ensurePipeline();
  const raw = await extractor(text, {
    pooling: 'none',
    normalize: false,
    return_attention_mask: false,
  });
  const lastHiddenState = this.extractLastHiddenState(raw);
  const pooled = this.meanPool(lastHiddenState);
  return { embedding: pooled };
}
async scoreTexts(ref, ans) {
  const similarity = await this.similarity(ref, ans);
  return this.toScore(similarity);
}
async similarity(a, b) {
  const [refEmb, ansEmb] = await Promise.all([
    this.getReferenceEmbedding(a),
    this.embed(b).then((x) => x.embedding),
  ]);
  return this.cosine(refEmb, ansEmb);
}
cosine(a, b) {
  const len = Math.min(a.length, b.length);
  let dot = 0;
  let normA = 0;
  let normB = 0;
  for (let i = 0; i < len; i++) {
    const va = a[i];
    const vb = b[i];
    dot += va * vb;
```

```
    normA += va * va;
    normB += vb * vb;
  }
  const denom = Math.sqrt(normA) * Math.sqrt(normB);
  if (denom === 0) return 0;
  return dot / denom;
}
toScore(similarity) {
  const clamped = Math.max(-1, Math.min(1, similarity));
  const scaled = ((clamped + 1) / 2) * 100;
  return Math.round(Math.max(0, Math.min(100, scaled)));
}
async getReferenceEmbedding(text) {
  const cached = this.referenceCache.get(text);
  if (cached) return cached;
  const { embedding } = await this.embed(text);
  this.referenceCache.set(text, embedding);
  return embedding;
}
async extractSubmissionFilesText(submission) {
  const files = submission.files ?? [];
  if (!files.length) {
    throw new BadRequestException('Submission has no files');
  }
  const parts = [];
  for (const file of files) {
    parts.push(
      await this.resolveTextSource({ filePath: file.fileUrl }, 'answer'),
    );
  }
  return parts.join('\n\n');
}
async resolveTextSource(source, label = 'answer') {
  const inline = source.text?.trim();
  if (inline) return inline;
  if (!source.filePath) {
    throw new BadRequestException(`No ${label} text or file path provided`);
  }
  const absolutePath = this.resolveFilePath(source.filePath);
  try {
    const extracted = await extractText(absolutePath);
    return extracted;
  } catch (err) {
    throw new BadRequestException(
      `Unsupported ${label} file type: ${err.message}`,
    );
    this.logger.error(`Failed to extract ${label} text from ${absolutePath}`);
    throw new ServiceUnavailableException(`Failed to extract ${label} text`);
  }
}
resolveFilePath(filePath) {
  if (isAbsolute(filePath)) return filePath;
```

```
    return join(process.cwd(), filePath.replace(/^[\\/]+/, ''));
  }
  extractLastHiddenState(raw) {
    if (raw && typeof raw === 'object' && raw.last_hidden_state) {
      const t = raw.last_hidden_state;
      return this.ensureTensorLike(t);
    }
    if (Array.isArray(raw) && raw.length > 0) {
      return this.ensureTensorLike(raw[0]);
    }
    if (raw && typeof raw === 'object') {
      return this.ensureTensorLike(raw);
    }
    throw new ServiceUnavailableException(
      'Unexpected output from feature extractor',
    );
  }
  ensureTensorLike(value) {
    if (value && value.dims.length >= 2 && value.data) {
      return value;
    }
    throw new ServiceUnavailableException(
      'Invalid tensor format returned by model',
    );
  }
  meanPool(lastHiddenState) {
    const dims = lastHiddenState.dims;
    const data = lastHiddenState.data;
    let tokens, hidden;
    if (dims.length === 3) {
      tokens = dims[1];
      hidden = dims[2];
    } else {
      tokens = dims[0];
      hidden = dims[1];
    }
    const out = new Float32Array(hidden);
    for (let t = 0; t < tokens; t++) {
      const offset = t * hidden;
      for (let h = 0; h < hidden; h++) {
        out[h] += Number(data[offset + h]);
      }
    }
    const inv = tokens > 0 ? 1 / tokens : 1;
    for (let h = 0; h < hidden; h++) {
      out[h] *= inv;
    }
    return out;
  }
}
```