ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Система штучного інтелекту для підвищення ефективності онлайн-освіти за допомогою персональних рекомендацій»

на здобуття освітнього ступеня магістра
зі спеціальності 126 Інформаційні системи та технології
освітньо-професійної програми Інформаційні системи та технології

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ _____

Виконав:
здобувач вищої освіти Шаповалов Богдан Дмитрович
група ІСДМ-61

Керівник: Сагайдак Віктор Анатолійович
науковий ступінь, вчене звання доктор філософії

Рецензент:
науковий ступінь, вчене звання _____
Ім’я, ПРІЗВИЩЕ

Київ – 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра Інформаційних систем та технологій

Ступінь вищої освіти Магістр

Спеціальність Інформаційні системи та технології

Освітньо-професійна програма Інформаційні системи та технології

ЗАТВЕРДЖУЮ
Завідувач кафедрою ІСТ

_____Каміла СТОРЧАК
«____» _____20___ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ СТУДЕНТУ

Шаповалов Богдан Дмитрович

1.Тема кваліфікаційної роботи: «Система штучного інтелекту для підвищення ефективності онлайн-освіти за допомогою персональних рекомендацій»
керівник кваліфікаційної роботи Сагайдак Віктор Анатолійович доктор філософії,
затверджені наказом вищого навчального закладу від «15» жовтня 2024 року № 320.

2. Строк подання кваліфікаційної роботи: 27 грудня 2024 року.

3. Вихідні дані до кваліфікаційної роботи:
Специфікації та вимоги до функціональності проєктованої ШІ-системи, отримані під час попереднього аналізу освітніх платформ
Опрацьовані та систематизовані вимоги користувачів (викладачів, студентів, адміністраторів) щодо функціоналу адаптивного навчання
Науково-технічна література з питань, пов’язаних з наукою про дані.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Обґрунтування вибору методів і алгоритмів ШІ для побудови адаптивної освітньої системи (аналіз моделей, критерії вибору)
2. Проєктування програмної архітектури (модульна структура, взаємодія компонентів, протоколи передачі даних)

5. Перелік ілюстративного матеріалу: *презентація*

6. Дата видачі завдання: 15 жовтня 2024 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	15.10 – 23.10.24	
2	Визначення методів та інструментів дослідження, формування вимог до ШІ-системи	24.10 – 01.11.24	
3	Проєктування архітектури прототипу і підготовка середовища розробки	02.11 – 07.11.24	
4	Реалізація (програмування) основних модулів ШІ-системи та первинне тестування	08.11 – 16.11.24	
5	Аналіз та оптимізація результатів тестування; доробка і налагодження прототипу	17.11 – 21.11.24	
6	Порівняльний аналіз з існуючими рішеннями; узагальнення проміжних результатів	22.11 – 11.12.24	
7	Оформлення роботи: вступ, висновки, реферат	12.12 – 20.12.24	
8	Розробка демонстраційних матеріалів	21.12 – 25.12.24	

Здобувач вищої освіти _____ Богдан Шаповалов
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник роботи
кваліфікаційної роботи _____ Віктор Сагайдак
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 80 стор., 15 рис., 11 джерел.

Мета роботи – обґрунтувати концепцію та розробити прототип системи з елементами штучного інтелекту, спрямованої на підвищення ефективності освітніх онлайн-платформ. Особливий акцент зроблено на формуванні індивідуальної траєкторії навчання, персоналізації контенту та використанні аналітики для вдосконалення освітнього процесу.

Об'єкт дослідження – сучасні цифрові освітні середовища, зокрема платформи для онлайн-навчання, що інтегрують різноманітні навчальні матеріали, інструменти оцінювання та засоби комунікації.

Предмет дослідження – методи та алгоритми штучного інтелекту (рекомендаційні системи, машинне навчання, модулі аналізу даних), а також архітектурні рішення для побудови адаптивної освітньої платформи з можливістю інтелектуальної обробки великих обсягів навчальної інформації.

Короткий зміст роботи: У роботі викладено теоретичні й практичні аспекти використання ШІ у сфері дистанційної освіти. Перші розділи присвячено аналізу науково-технічної літератури та огляду сучасних платформ, визначенню вимог до адаптивних систем. Окрему увагу зосереджено на проектуванні прототипу з мікросервісною архітектурою, де основні компоненти системи реалізовано із застосуванням node.js-фреймворків та MongoDB для збереження даних. Результати свідчать про перспективність впровадження ШІ-механізмів у процес навчання. Окреслено потенційні напрями подальшої розробки, включно з переходом до формату SaaS і розширенням функціоналу за допомогою більш складних рекомендаційних моделей.

КЛЮЧОВІ СЛОВА: штучний інтелект, освітні платформи, адаптивне навчання, персоналізація, машинне навчання, мікросервісна архітектура, MongoDB, рекомендаційні системи, SaaS, аналітика навчальних даних.

ВІДГУК НАУКОВОГО КЕРІВНИКА

РЕЦЕНЗІЯ

ЗМІСТ

ВСТУП 1
Актуальність теми 1
Об’єкт дослідження 2
Предмет дослідження 2
Методи дослідження 3
Апробація результатів та публікації ..... **Ошибка! Закладка не определена.**
Теоретична, методична та практична значущість отриманих результатів 3
РОЗДІЛ 1. ТЕОРЕТИЧНИЙ АНАЛІЗ ТА ВИЗНАЧЕННЯ ВИМОГ 5
1.1 Аналіз існуючих освітніх платформ 5
1.2 Технологічні тенденції у веб-розробці 6
1.3 Інтеграція штучного інтелекту в освітні програми 8
1.4 Формулювання системних вимог 10
1.5 Аналіз цільової аудиторії 12
1.6 Діаграми та моделювання 13
1.7 Вибір технологій та інструментів 14
РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ПРОГРАМИ 17
2.1 Архітектура програми 17
2.2 Моделювання бази даних 19
2.3 Дизайн RESTful API 20
2.4 Асистент зі штучним інтелектом 22
2.5 Огляд структури проекту 23
2.6 Структура API 26
2.7 Огляд реалізації проекту 28
РОЗДІЛ 3. ТЕСТУВАННЯ ТА ОЦІНКА РЕЗУЛЬТАТІВ 50
3.1 Підходи до тестування 50
3.2 Інструменти тестування 52
3.3 Досягнення поставлених цілей 55
3.4 Переваги та обмеження системи 57
3.5 Майбутні напрямки розвитку 60
ВИСНОВКИ 64
ПЕРЕЛІК ПОСИЛАНЬ 69

ВСТУП

Актуальність теми

Останніми роками стрімкий розвиток технологій штучного інтелекту (ШІ) спричинив глибокі зміни в тому, як ми розуміємо, надаємо та вимірюємо освітні результати. Освітні платформи, що колись покладалися на статичні ресурси й універсальні підходи, дедалі частіше використовують динамічні алгоритми для персоналізації навчального процесу. Такий перехід не є простою технологічною тенденцією — він відображає визнання того, що студенти розрізняються за рівнем підготовки, темпами засвоєння матеріалу й освітніми потребами. Традиційні лекції й стандартизовані тести часто ігнорують учнів, які або випереджають програму, або навпаки, потребують додаткової допомоги. Зважаючи на це, якість та ефективність навчання у змішаних чи повністю онлайн-середовищах набувають високої пріоритетності для навчальних закладів і освітніх політиків.

Системи, що базуються на ШІ, обіцяють докорінно змінити всю освітню екосистему. Використовуючи складні методи машинного навчання (від алгоритмів обробки природної мови до адаптивних механізмів рекомендацій), платформи можуть пропонувати індивідуальний підхід до навчання, розпізнавати прогалини у знаннях студентів, надавати додаткові ресурси та динамічно змінювати складність завдань. Крім того, ШІ-платформи зменшують навантаження на викладачів, забезпечуючи їх актуальною статистикою щодо прогресу групи, а студентам надають відчуття активної залученості у власний освітній процес. Такий підхід, що поєднує автоматизацію й гнучкість навчальних траєкторій, здатен підвищити не лише ефективність, а й мотивацію та загальну якість навчання.

Мета і завдання дослідження

Головна мета цієї роботи полягає в концептуалізації, розробці та оцінюванні прототипу системи штучного інтелекту, спрямованої на підвищення ефективності освітніх платформ. Під «ефективністю» тут розуміють не лише зменшення витрат

часу й ресурсів, а й формування більш глибокої взаємодії з учнем та підтримку викладачів у прийнятті рішень, що базуються на аналітиці даних.

Для досягнення цієї мети окреслено такі завдання:

Визначити ключові показники ефективності освітньої платформи (рівень залученості студентів, актуальність контенту, відсоток завершення курсу, рівень засвоєння основних понять).

Розробити архітектуру ШІ-системи, включно з вибором алгоритмів, механізмами збору та обробки даних і точками інтеграції з наявними навчальними платформами.

Впровадити адаптивні механізми персоналізації, які б динамічно налаштовували освітній контент під індивідуальні потреби кожного учня (попередній досвід, стиль навчання, швидкість опрацювання матеріалу).

Створити методику оцінювання впливу інтегрованої системи ШІ на результати навчання, задоволеність користувачів і ефективність використання ресурсів.

Провести експериментальний аналіз для порівняння розробленого прототипу з традиційними освітніми платформами без ШІ-функцій.

Об’єкт дослідження

Об’єктом дослідження виступає сучасний ландшафт освітніх платформ, що динамічно розвивається та включає різні формати онлайн-навчання: масові відкриті онлайн-курси (МВОК), системи управління навчанням (LMS), програми для професійного розвитку тощо. Всі ці платформи формують складну екосистему, яка охоплює контент, інструменти для оцінювання, обговорення в спільноті та аналітичні модулі.

Предмет дослідження

Предметом роботи є спеціалізована ШІ-система й алгоритмічні рішення, що впливають на підвищення результативності освітніх платформ. Сюди входять: методи машинного навчання для рекомендацій, стратегії адаптації навчальних матеріалів у реальному часі, а також елементи аналітики, спрямовані на моніторинг і

прогнозування навчальних досягнень. Дослідження зосереджується не на окремому компоненті, а на їхній цілісній інтеграції в рамках єдиної платформи.

Методи дослідження

Аналіз літератури та теоретичне обґрунтування. У межах початкового етапу було здійснено вивчення наукових і прикладних джерел для визначення актуальних тенденцій та прогалин.

Системне проектування та створення прототипу. На базі зібраних теоретичних даних було розроблено прототип системи штучного інтелекту з використанням гнучких методологій розробки та вибраних технологічних стеків (Unicorn Application Framework, MongoDB, бібліотеки для машинного навчання).

Методи статистичної обробки. Для підтвердження гіпотез та інтерпретації результатів застосовувалися статистичні методи аналізу даних.

Теоретична, методична та практична значущість отриманих результатів

Теоретична значущість полягає в розвитку наукового підходу до інтеграції ШІ в освітні середовища. Зокрема, робота доповнює вже існуючі моделі адаптивного навчання розширеними функціями з гнучким комбінуванням рекомендаційних алгоритмів і механізмів відстеження прогресу.

Методична значущість виявляється у формуванні чітких критеріїв і методів оцінювання результативності впровадження ШІ-систем у навчальний процес. Узгоджені метрики, описані в межах дослідження, допомагають викладачам та адміністрації в систематичному аналізі успішності курсів та окремих учнів.

Практична цінність проявляється в можливості безпосереднього застосування розроблених прототипів і підходів у реальних освітніх платформах. Результати дослідження демонструють, що автоматизований підхід до керування навчальними шляхами, адаптації контенту й генерування рекомендацій здатен підвищити мотивацію та результативність навчання. Надалі, переведення системи в формат SaaS

створить ринкові умови для масштабного впровадження, надаючи закладам освіти інструменти аналізу й покращення навчального процесу на основі даних.

Апробація результатів магістерської роботи

Основні положення і результати магістерської роботи були представлені та обговорені на II Всеукраїнській науково-технічній конференції “Технологічні горизонти: дослідження та застосування інформаційних технологій для технологічного прогресу України і світу” – “Застосування штучного інтелекту в розробці ігор: вплив на розвиток індустрії та перспективи”, “Застосування штучного інтелекту в розробці ігор: вплив на розвиток індустрії та перспективи” (18 листопада 2024 року, Київ: Державний університет інформаційно-комунікаційних технологій).

РОЗДІЛ 1. ТЕОРЕТИЧНИЙ АНАЛІЗ ТА ВИЗНАЧЕННЯ ВИМОГ

1.1 Аналіз існуючих освітніх платформ

За останнє десятиліття діджиталізація освіти значно розширила спектр платформ для онлайн-навчання. Ці платформи, які спочатку виникли як сховища контенту або додаткові ресурси для традиційних класів, перетворилися на складні, багатофункціональні екосистеми, які підтримують учнів на різних етапах - від початкової освіти до професійного розвитку та навчання впродовж життя. Яскравими прикладами є провайдери масових відкритих онлайн-курсів (МВОК), такі як Coursera, edX та Udemy, а також системи управління навчанням (LMS), що використовуються в академічному середовищі, такі як Moodle, Canvas та Blackboard. Крім того, численні спеціалізовані платформи орієнтовані на нішеві навички та галузі, пропонуючи курси в таких сферах, як програмування, дизайн, охорона здоров'я та управління бізнесом.

Однією з визначальних характеристик цих сучасних платформ є їхня універсальність. Спочатку багато систем мали на меті лише надання статичних матеріалів курсу - записаних лекцій, PDF-файлів для читання та тестових завдань з множинним вибором - для широкої аудиторії. З часом розвиток педагогіки і технологій дозволив цим платформам включати більше інтерактивних елементів. Дискусійні форуми, вебінари в прямому ефірі, віртуальні лабораторні симуляції та процеси експертного оцінювання додали соціальний і динамічний вимір до асинхронного онлайн-навчання. Крім того, деякі платформи почали впроваджувати аналітику даних для відстеження залучення студентів, вимірювання результатів навчання і забезпечення зворотного зв'язку, який інформує як викладачів, так і студентів. Однак, незважаючи на ці покращення, більшість існуючих освітніх платформ все ще борються з уніфікованою подачею контенту та стандартизацією, що

не дозволяє адекватно задовольнити різноманітні потреби, досвід та прагнення окремих учнів. [1, 2]

Іншим важливим аспектом є глобальне охоплення та доступність цих платформ. Багато з них розроблені таким чином, щоб виходити за межі географічних кордонів, об'єднуючи учнів і викладачів з усього світу. Хоча таке глобальне підключення заслуговує на похвалу, воно створює нові виклики, такі як забезпечення доступності для учнів з обмеженими можливостями, надання локалізованого і культурно релевантного контенту, а також збереження конфіденційності та безпеки даних. Взаємодія цих факторів створює середовище, що дозріло для інновацій - інновацій, які можуть допомогти платформам вийти за межі своїх обмежень і стати дійсно орієнтованим на учнів середовищем.

1.2 Технологічні тенденції у веб-розробці

Стрімкий розвиток технологій веб-розробки суттєво вплинув на можливості та функціональність освітніх платформ. Сучасні веб-технології наголошують на оперативності, масштабованості, інтероперабельності та безпеці, що є важливими для платформ, які обслуговують великі та різноманітні бази користувачів. Освітні платформи все частіше використовують новітні фронтенд-фреймворки (такі як React, Angular та Vue.js) для створення плавних та інтуїтивно зрозумілих користувацьких інтерфейсів. Ці фреймворки роблять акцент на компонентній архітектурі, що дозволяє розробникам створювати модульні та багаторазові елементи інтерфейсу, які можна легко адаптувати в міру зростання та диверсифікації платформи.

На серверній стороні архітектура мікросервісів набула популярності, розбиваючи монолітні додатки на незалежно розгорнуті, слабко пов'язані між собою сервіси. Такий підхід дозволяє освітнім платформам масштабувати певні функціональні

можливості - наприклад, механізми рекомендацій, автентифікацію користувачів чи модулі оцінювання - не впливаючи на всю систему. У поєднанні з контейнеризацією (за допомогою таких інструментів, як Docker) та технологіями оркестрування (наприклад, Kubernetes), ці архітектури гарантують, що освітні платформи залишатимуться стійкими, підтримуватимуть високу доступність та адаптуватимуться до сплесків користувацького трафіку.

Більше того, повсюдне поширення мобільних пристроїв і поява прогресивних веб-додатків (PWA) [5] означає, що учні тепер очікують безперешкодного доступу до освітніх ресурсів у будь-який час і в будь-якому місці. Адаптивний дизайн і мобільні інтерфейси забезпечують однаковий користувацький досвід на різних пристроях, усуваючи бар'єри, пов'язані з різним розміром екрану або обмеженням пропускної здатності. Крім того, заходи безпеки, такі як впровадження протоколів HTTPS, регулярна оцінка вразливостей і надійне шифрування даних, стали неодмінною умовою, особливо з огляду на чутливі дані користувачів (наприклад, оцінки, особиста інформація), які часто обробляються на освітніх платформах.

Важливо, що екосистема веб-розробки також охопила широку інтеграцію API, що дозволяє освітнім платформам підключатися до зовнішніх інструментів і сервісів. Незалежно від того, чи інтегруються спеціалізовані механізми оцінювання, сервіси перевірки або зовнішні бібліотеки цифрового контенту, така модульність дозволяє освітнім екосистемам бути більш гнучкими, багатими і здатними задовольняти унікальні потреби учнів. Гнучкість, яку забезпечують ці сучасні тенденції веб-розробки, є основною вимогою до наступного покоління освітніх технологій - технологій, які прагнуть бути не лише зручними для користувача, але й інтелектуальними, адаптивними та глибоко інтегрованими в ширші навчальні екосистеми.

1.3 Інтеграція штучного інтелекту в освітні програми

Оскільки освітні платформи прагнуть диференціювати себе і покращити навчальний процес, інтеграція штучного інтелекту (ШІ) стала трансформаційною тенденцією. ШІ, що охоплює широкий спектр методів - машинне навчання, обробку природної мови (NLP), навчання з підкріпленням і комп'ютерний зір - пропонує потенціал для радикального поліпшення способів доставки, персоналізації та оцінювання освітнього контенту.

Одне з найбільш безпосередніх застосувань ШІ в освіті пов'язане з інтелектуальними системами навчання (ІСН). Ці системи імітують індивідуальне навчання, відстежуючи успішність учня і адаптуючи складність, темп і зміст матеріалу, що подається. Аналізуючи час відгуку, типи помилок і відгуки користувачів, ІТС можуть виявити прогалини в знаннях і надати цілеспрямовану підтримку. Наприклад, програми для вивчення мов використовують алгоритми НЛП для надання зворотного зв'язку в режимі реального часу щодо вимови, граматики і використання словника, тоді як інструменти для навчання математики і природничих наук можуть розбивати складні кроки розв'язання проблем і виділяти області, які потребують додаткової уваги.

Механізми персоналізації, вбудовані в освітні платформи, також завдячують своєю функціональністю ШІ. Замість того, щоб представляти однаковий прогрес курсу для всіх учнів, ці платформи використовують алгоритми рекомендацій - подібні до тих, що застосовуються в електронній комерції або потокових сервісах, - щоб запропонувати відповідні шляхи навчання. Вони можуть враховувати попередні курсові роботи, результати оцінювання, вподобання користувача і навіть поведінкові підказки, як-от час, витрачений на певні види діяльності. З часом ці системи рекомендацій стають дедалі точнішими, уможливлюючи високоіндивідуалізовані навчальні траєкторії, які підтримують залученість учнів і покращують результати.

Окрім персоналізації, аналітика на основі штучного інтелекту відіграє важливу роль у підвищенні якості освіти. Удосконалені моделі машинного навчання можуть обробляти величезні обсяги даних - оцінки за іспити, спроби пройти вікторину, участь у форумах і використання навчальних ресурсів - і генерувати дієві висновки для викладачів і адміністраторів. Виявляючи закономірності та взаємозв'язки, така аналітика може сприяти розробці навчальних планів, допомагати викладачам визначати пріоритетні напрямки і навіть прогнозувати, хто з учнів ризикує залишити навчання поза увагою. Після цього можна впроваджувати стратегії раннього втручання, надаючи підтримку до того, як прогалини в знаннях стануть нездоланним бар'єром.

Обробка природної мови виявилася особливо цінною як для автоматизованого оцінювання, так і для вилучення знань. Автоматизовані системи оцінювання есе можуть оцінювати письмові завдання зі зростаючою надійністю, пропонуючи детальний зворотний зв'язок щодо зв'язності, структури та якості аргументації. Аналогічно, інструменти вилучення контенту на основі штучного інтелекту можуть обробляти великі бібліотеки академічних ресурсів, семантично індексуючи їх, щоб студенти могли легко знаходити потрібні матеріали. Таке поєднання інтелектуального аналізу контенту та пошуку значно зменшує когнітивне навантаження як на учнів, так і на викладачів, дозволяючи їм зосередитися на більш складних когнітивних завданнях.

Однак інтеграція ШІ в освіту також пов'язана з низкою викликів і етичних міркувань. [10] Забезпечення справедливості та прозорості оцінювання за допомогою штучного інтелекту, захист приватного життя студентів і збереження людського нагляду - все це є критично важливими питаннями, які необхідно вирішити. Мета полягає не в тому, щоб замінити викладачів, а в тому, щоб розширити їхні можливості, звільнити

від повторюваних завдань і дати їм змогу зосередитися на обов'язках більш високого рівня, таких як наставництво, розвиток критичного мислення і заохочення творчості. Балансування цих етичних імперативів з технологічними можливостями є делікатним, але необхідним завданням.

Підсумовуючи, ландшафт освітніх технологій динамічно розвивається під впливом зростаючої складності практик веб-розробки, глобального поширення онлайн-освіти та зростаючої популярності штучного інтелекту. Сучасні освітні платформи перетворилися зі статичних сховищ контенту на динамічні навчальні екосистеми, проте залишається багато викликів, особливо у сферах персоналізації, доступності та залучення. Оскільки технологічні тенденції, описані в цьому розділі, продовжують зближуватися, інтеграція штучного інтелекту стає ключовим елементом освітніх інновацій наступного покоління. Вона відкриває можливості для створення адаптивних, орієнтованих на учня середовищ, здатних задовольнити різноманітні навчальні потреби, і, потенційно, перетворити онлайн-освіту з моделі пасивного споживання на справді інтерактивний, персоналізований і ефективний досвід. [7]

1.4 Формулювання системних вимог

Початковим кроком у розробці системи зі штучним інтелектом для підвищення ефективності освітньої платформи є чітке формулювання функціональних і нефункціональних вимог. Ці вимоги слугують відправною точкою для прийняття архітектурних рішень і подальших етапів реалізації.

З функціональної точки зору, система повинна дозволяти освітнім платформам автоматично адаптуватися до різних стилів навчання, рівнів кваліфікації та вподобань користувачів. Це передбачає можливість відстежувати та аналізувати показники успішності учнів, виявляти сфери, де виникають труднощі або непорозуміння, а також рекомендувати персоналізований контент, завдання або

додаткові ресурси. Крім того, вона повинна включати інтелектуальні функції навчання, такі як підказки в режимі реального часу, автоматизоване оцінювання та механізми негайного зворотного зв'язку, які працюють разом для підвищення ефективності навчання.

Окрім цієї основної функціональності, система повинна легко інтегруватися з існуючими екосистемами освітнього програмного забезпечення. Зокрема, вона повинна підтримувати стандартизовані API для обміну даними з різними системами управління навчанням (LMS), електронними бібліотеками та інструментами зовнішнього оцінювання. Протоколи конфіденційності та безпеки даних повинні бути вбудовані на кожному рівні, щоб гарантувати, що конфіденційна інформація про учнів, така як особисті дані, академічні записи та аналітика успішності, залишатиметься конфіденційною та захищеною від порушень. Крім того, система повинна підтримувати високу доступність і масштабованість, щоб пристосуватися до різних груп користувачів і мінливого робочого навантаження, особливо в пікові моменти, наприклад, під час реєстрації на курс або іспитів.

Нефункціональні вимоги включають зручність використання, підтримуваність, інтероперабельність та обмеження продуктивності. Інтерфейс системи має бути інтуїтивно зрозумілим і вимагати мінімальної підготовки для викладачів чи адміністраторів. З технічної точки зору, підтримуваність передбачає чисту, модульну кодову базу та вичерпну документацію, яка дозволяє майбутні вдосконалення та оновлення. Інтероперабельність вимагає використання відкритих стандартів і протоколів для безперешкодної інтеграції в різноманітні навчальні середовища. Нарешті, міркування продуктивності гарантують, що алгоритми рекомендацій, модулі аналізу даних і компоненти адаптивного навчання реагують з прийнятними затримками, зберігаючи таким чином плавність і безперервність навчального процесу.

1.5 Аналіз цільової аудиторії

Щоб розробити систему, яка справді підвищить ефективність освітніх платформ, важливо розуміти демографічні та психографічні профілі основних зацікавлених сторін. Хоча учні стоять у центрі цієї інновації, інші важливі учасники, такі як викладачі, адміністратори, розробники навчальних програм і навіть інституційні політики, роблять свій внесок в екосистему і мають різні очікування від системи.

Учні: Ця група є дуже різноманітною, включаючи учнів шкільного віку, студентів, працюючих професіоналів, які шукають можливості для підвищення кваліфікації, а також тих, хто самостійно навчається впродовж усього життя. Кожна підгрупа має різний рівень цифрової грамотності, мотивації та часових обмежень. Пропонуючи індивідуальні навчальні траєкторії, персоналізований зворотний зв'язок і доступні інтерфейси, сумісні з різними пристроями, система повинна бути привабливою для широкої аудиторії, забезпечуючи адаптивність та інклюзивність.

Викладачі та репетитори: Викладачі покладаються на надійну аналітику для оцінки ефективності своїх педагогічних підходів. Їм потрібні дієві дані про прогрес учнів, проблемні зони та моделі залучення. Система повинна пропонувати інформаційні панелі звітності, сповіщення для учнів з групи ризику та гнучкість у динамічному коригуванні матеріалів курсу. Таким чином, система дає можливість викладачам вдосконалювати свої підходи, зосереджуючись на більш складних завданнях, таких як фасилітація групових дискусій, наставництво та розвиток критичного мислення.

Адміністратори та розробники навчальних програм: Зацікавлені сторони, включаючи адміністраторів та розробників навчальних програм, використовують аналітику системи для прийняття рішень на основі даних. Розподіляючи ресурси,

вдосконалюючи пропозиції курсів або формуючи інституційні стратегії, ці користувачі потребують всебічного аналізу тенденцій ефективності, рівня завершення курсів і статистики залучення користувачів. Система повинна надавати їм масштабовані та безпечні інтерфейси для роботи з даними, щоб вони могли визначати та задовольняти ширші організаційні потреби.

Технічний та допоміжний персонал: Особи, відповідальні за підтримку та модернізацію технічної інфраструктури освітньої платформи, повинні вважати архітектуру системи прозорою та керованою. Вони покладатимуться на добре задокументовані API, модульні структури коду та конвеєри безперервної інтеграції/безперервної доставки (CI/CD). Завдяки спрощенню процесу системного адміністрування платформа залишатиметься стабільною, сучасною та легкою для розвитку.

1.6 Діаграми та моделювання

Систематичний підхід до моделювання прояснює архітектуру системи, її взаємодію з різними компонентами та потік даних між модулями. У цьому розділі ми розглянемо високорівневі діаграми проектування для інкапсуляції модулів системи, конвеєрів даних та взаємодії з користувачем. Ці діаграми включатимуть

Діаграми варіантів використання: Ілюструють різні ролі користувачів - учнів, викладачів, адміністраторів - і функціональні можливості, до яких вони можуть отримати доступ. Наприклад, учень може взаємодіяти з системою рекомендацій і автоматизованими інструментами зворотного зв'язку, тоді як викладач може мати доступ до панелей детальної аналітики і кастомізації.

Діаграми компонентів: Відображають основні модулі системи, такі як рівень введення даних, аналітичний механізм ШІ, підсистема рекомендацій, рівень користувацького інтерфейсу та служби автентифікації. Кожен компонент взаємодіє через чітко визначені інтерфейси, забезпечуючи модульність і простоту обслуговування.

Діаграми потоків даних (DFD): Показують, як дані переміщуються між компонентами системи. Наприклад, DFD може показувати, як необроблені журнали взаємодії з користувачем обробляються моделями ШІ для створення персоналізованих рекомендацій щодо курсу, і як ці рекомендації потім передаються в інтерфейс користувача.

Діаграми класів або сутностей-зв'язків (ERD): Надають концептуальне уявлення про основні структури даних, включаючи профілі студентів, метадані курсу, журнали активності та результати аналітики.

Створюючи ці діаграми, ми встановлюємо візуальну основу, яка керує розробниками, зацікавленими сторонами та іншими учасниками співпраці. Хоча ми не будемо вбудовувати ці діаграми в даний текст, план дозволяє кожному зрозуміти складність системи, виявити потенційні точки інтеграції та визначити області, які можуть потребувати доопрацювання. Такий підхід допомагає гарантувати, що остаточна архітектура системи буде не лише надійною та ефективною, але й прозорою та зрозумілою для всіх залучених сторін.

1.7 Вибір технологій та інструментів

Вибір правильних технологій та інструментів є ключовим кроком, який безпосередньо впливає на швидкість розробки, продуктивність системи та

довгострокову підтримуваність. Спираючись на сучасні тенденції веб-розробки, обраний стек повинен сприяти стабільності, масштабованості та сумісності з різноманітними освітніми платформами. Незважаючи на те, що існує безліч фреймворків та інструментів, ми прагнемо оптимізувати стек на основі перевірених, гнучких технологій, які поєднують простоту використання з надійністю.

Для бекенду ми використовуємо фреймворк Unicorn Application Framework (UAF). UAF - це сучасний, високомодульний фреймворк, який полегшує швидку побудову масштабованих, орієнтованих на мікросервіси архітектур. Акцент на стандартизовані інтерфейси, чіткі шаблони конфігурації та надійні засоби безпеки роблять його особливо придатним для запропонованої системи. Використовуючи UAF, розробники можуть легше інтегрувати можливості ШІ, оскільки фреймворк надає гнучкі конвеєри даних і точки підключення, які можуть легко вмістити моделі машинного навчання та аналітичні механізми.

Для інтеграції ШІ ми можемо використовувати широко розповсюджені бібліотеки та фреймворки, такі як TensorFlow, PyTorch або scikit-learn. Ці інструменти дозволяють експериментувати з різними алгоритмами - від обробки природної мови та методів кластеризації до моделей навчання з підкріпленням - таким чином гарантуючи, що наша система може розвиватися та вдосконалюватися з часом. Крім того, хмарні платформи (такі як AWS, Azure або GCP) дозволяють інтегрувати масштабовані обчислювальні ресурси, керовані бази даних і послуги з оркестрування контейнерів. [8]

З точки зору інтерфейсу, використання сучасного фреймворку JavaScript (наприклад, React або Vue.js) дозволить створювати інтерактивні дашборди, інтуїтивно зрозумілі аналітичні інтерфейси та адаптивний користувацький досвід. Поєднання цих фреймворків з надійними API-шлюзами та протоколами автентифікації (OAuth2,

JWT) гарантує безперебійний та безпечний зв'язок між фронтенд- та бекенд-компонентами.

Нарешті, для забезпечення якості та надійності рекомендується застосовувати практики CI/CD за допомогою таких інструментів, як Jenkins, GitLab CI або GitHub Actions. Фреймворки для автоматизованого тестування, лістери якості коду та інструменти для бенчмаркінгу продуктивності допоможуть підтримувати високий рівень якості програмного забезпечення та мінімізувати ризик регресій або вразливостей.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ПРОГРАМИ

2.1 Архітектура програми

Архітектура освітньої системи зі штучним інтелектом повинна відображати складність її основних функцій - адаптивні навчальні траєкторії, інтелектуальні рекомендації, надійний аналіз даних - і при цьому залишатися зручною в обслуговуванні, масштабованою та відмовостійкою. Для досягнення цих цілей ми покладаємося на модульну, сервіс-орієнтовану архітектуру, яка використовує Unicorn Application Framework (UAF) як основу системи. Вибір UAF гарантує, що кожен модуль - чи то навчальний аналітичний движок, служба рекомендацій, чи шлюз користувацького інтерфейсу - може працювати як незалежний, але взаємопов'язаний сервіс.

На найвищому рівні систему можна уявити як набір взаємопов'язаних шарів:

Рівень представлення: Інтерфейсні компоненти, побудовані з використанням сучасного фреймворку JavaScript (наприклад, React), слугують точкою входу для всіх взаємодій користувачів. Учні та викладачі підключаються через свої браузери або мобільні пристрої, отримуючи доступ до інформаційних панелей, матеріалів курсу та персоналізованих рекомендацій. Цей рівень взаємодіє з внутрішніми сервісами за допомогою RESTful API, захищених стандартизованими протоколами автентифікації.

Рівень сервісів (мікросервіси): Цей рівень містить набір спеціалізованих сервісів, кожен з яких інкапсулює певну частину логіки. Наприклад, служба профілю користувача займається автентифікацією учня, налаштуваннями облікового запису та зберіганням налаштувань. Служба вмісту курсу зберігає і керує матеріалами курсу - відеолекціями, PDF-файлами, вправами і зовнішніми посиланнями - забезпечуючи при цьому контроль версій і доступність. Служба рекомендацій використовує

алгоритми штучного інтелекту, щоб відповідати потребам учнів і підбирати відповідний контент. Служба аналітики агрегує та обробляє дані про взаємодію з учнями, щоб генерувати показники ефективності та прогностичні висновки. Ці сервіси взаємодіють через чітко визначені інтерфейси, як правило, через легкі RESTful кінцеві точки або асинхронні черги повідомлень, що забезпечує ефективний розподіл навантаження і легке масштабування.

Рівень штучного інтелекту і обробки даних: На цьому рівні знаходяться моделі машинного навчання, компоненти обробки природної мови та конвеєри предиктивної аналітики. Працюючи на виділених обчислювальних ресурсах, ці модулі безперервно отримують нові дані з рівня сервісів, оновлюють свої моделі та надсилають результати - наприклад, персоналізовані навчальні траєкторії, коригування складності тестів або автоматизований зворотний зв'язок - назад до системи.

Рівень доступу до даних: Рівень збереження даних спирається на комбінацію механізмів зберігання, включаючи реляційні бази даних для структурованих даних про учнів і курси, бази даних NoSQL або озера даних для неструктурованих журналів і трас взаємодії, а також рівень розподіленого кешу для швидкого отримання ресурсів, до яких часто звертаються. Завдяки впровадженню чітких політик доступу до даних за допомогою шаблонів конфігурації UAF, цей рівень підтримує цілісність, безпеку та відповідність нормам конфіденційності даних.

Рівень безпеки та інтеграції: В основі архітектури лежить спеціальна служба автентифікації та авторизації, що реалізує стандартні галузеві протоколи, такі як OAuth2 та токени JWT. Шлюзи API та сервісні сітки забезпечують балансування навантаження, обмеження швидкості та можливості ведення журналів. Зовнішня інтеграція - наприклад, підключення до сторонніх LMS, бібліотек контенту або платформ оцінювання - відбувається через чітко визначені служби адаптерів, що

гарантує, що систему можна розширювати за потреби без шкоди для її внутрішньої узгодженості.

Цей багаторівневий, сервіс-орієнтований підхід в рамках UAF гарантує, що кожен компонент залишається модульним і взаємозамінним. В міру розвитку моделей ШІ можна безперешкодно впроваджувати нові послуги. Зі збільшенням навантаження на користувача послуги можуть бути репліковані і масштабовані незалежно один від одного. Результатом є архітектура, в якій пріоритетами є гнучкість, продуктивність і ремонтопридатність, що дозволяє освітній системі розвиватися разом з технологічним і педагогічним розвитком.

2.2 Моделювання бази даних

Модель даних системи повинна охоплювати різноманітний набір сутностей: учнів, курси, завдання, оцінки, взаємодію користувачів та складну аналітику. Враховуючи динамічну природу освітніх даних - від структурованих профілів користувачів до журналів взаємодії - MongoDB є ідеальним вибором завдяки своїм гнучким схемам, документ орієнтованому підходу та масштабованості.

Основні колекції та схеми:
Фундаментальні сутності домену, такі як користувачі, курси та реєстрації, можна зберігати у вигляді документів MongoDB. Наприклад, колекція Користувачі може містити документи з такими полями, як user_id, ім'я, електронна пошта, роль і вбудований об'єкт для налаштувань. Аналогічно, колекція Courses може зберігати course_id, назву, опис і масив пов'язаних ресурсів або модулів. Організовуючи дані в такі добре структуровані, але гнучкі за схемою колекції, ми можемо легко розвивати і адаптувати ці сутності без складних міграцій схем.

Дані взаємодії та аналітики:

Великі обсяги даних, такі як журнали активності учнів, спроби оцінювання та метрики взаємодії, природно вписуються в модель документів MongoDB. Такі колекції, як LearnerActivityLogs або AssessmentResults, можуть зберігати документи з позначкою часу, які фіксують події, відповіді та показники ефективності. Такий підхід спрощує зберігання різнорідних даних і дозволяє швидко змінювати схеми, що дуже важливо в системі, де нові метрики або навчальні модулі можуть вводитися безперервно. Можливість горизонтального розбиття колекцій на частини гарантує, що рівень бази даних може легко масштабуватися в міру розширення бази користувачів і збільшення пропускної здатності даних.

Використання MongoDB в якості центрального рішення для зберігання даних дає системі переваги масштабованості, адаптивності та високої доступності. Документно-орієнтована модель разом зі стратегічним індексуванням та кешуванням гарантує, що рівень бази даних може гнучко реагувати на мінливі вимоги до даних. Своєю чергою, ця надійна і гнучка модель даних закладає основу для розширеної аналітики, персоналізованих рекомендацій і безперешкодної інтеграції нових функцій, керованих штучним інтелектом.

2.3 Дизайн RESTful API

Зрозумілий і послідовний RESTful API має велике значення для зв'язку між інтерфейсами, внутрішніми сервісами і будь-якими зовнішніми інтеграціями. Дизайн API відповідає принципам ресурсно-орієнтованої архітектури, що означає, що кінцеві точки представляють сутності, а дії виконуються за допомогою стандартних методів HTTP (GET, POST, PUT, DELETE).

Деякі ключові міркування включають:

Іменування та структура ресурсів: Використання читабельних, передбачуваних імен кінцевих точок.

Стандартизовані методи HTTP:
GET: Отримати дані (наприклад, деталі курсу або профілі користувачів).
POST: Створення нових сутностей (наприклад, реєстрація користувача на курс).
PUT/PATCH: Оновлення існуючих сутностей (наприклад, зміна налаштувань користувача).
DELETE: Видалення сутностей (наприклад, видалення користувача з курсу).

Аутентифікація та авторизація: API повинен вимагати безпечні токени доступу для захищених кінцевих точок. Система повинна перевіряти токени, забезпечувати відповідність дозволів користувачів запитуваним діям і повертати відповідні коди статусу для несанкціонованих або заборонених запитів (401 і 403 відповідно).

Обробка помилок та зворотній зв'язок: Всі кінцеві точки повинні надавати змістовні відповіді на помилки та стандартизовані структури JSON, включаючи коди помилок, повідомлення та підказки щодо їх усунення. Наприклад, відповідь на 400 Bad Request може містити інформацію про відсутні обов'язкові поля, що допоможе розробникам або кінцевим користувачам виправити запит.

Дотримуючись конвенцій RESTful, система забезпечує інтероперабельність, підтримуваність і полегшує процес входження в систему для нових розробників і партнерів. Така філософія дизайну дозволяє поступово розширювати функціональність з часом і спрощує як тестування інтеграції, так і моніторинг.

2.4 Асистент зі штучним інтелектом

В основі адаптивних можливостей платформи лежить асистент штучного інтелекту: інтелектуальний модуль, який безперервно навчається на основі дій користувача, змісту курсу та моделей успішності, щоб надавати персоналізовану підтримку. Асистента ШІ можна уявити як спеціалізований сервіс, що працює на перетині аналітики, рекомендаційних систем та інтерфейсів природної мови.

Адаптивний механізм рекомендацій: спираючись на взаємодію з користувачем - результати тестів, навчальні звички, показники завершення - асистент ШІ визначає контент, який найкраще відповідає профілю кожного учня. Це може означати, що він запропонує додаткову літературу учням, які мають проблеми з певними поняттями, або порекомендує поглиблені матеріали для тих, хто постійно перевищує базові показники. Асистент також допомагає регулювати рівні складності в режимі реального часу, гарантуючи, що навчання залишається складним, але не непосильним.

Інтелектуальні функції репетитора: Асистент може пропонувати підказки, пояснення та покрокові інструкції для вирішення проблем. Використовуючи моделі НЛП для розуміння запитів користувачів, він надає контекстно-чутливі відповіді на питання, пояснює складні концепції більш простими словами та роз'яснює непорозуміння, які можуть виникнути під час роботи над складними завданнями.

Прогностична аналітика та раннє втручання: Аналізуючи довгострокові тенденції, ШІ-асистент передбачає, коли студенти ризикують припинити навчання або провалити курс. Запуск автоматичних сповіщень для викладачів або надсилання мотиваційних повідомлень учням може допомогти запобігти падінню успішності. Ці проактивні втручання створюють більш сприятливе і чуйне навчальне середовище.

Взаємодія природною мовою: Інтеграція чат-ботів або голосових інтерфейсів дозволяє учням взаємодіяти з системою більш природно. Незалежно від того, чи запитуєте ви роз'яснення, чи ставите практичні завдання, чи вивчаєте нові теми, розмовний ШІ сприяє більш захоплюючому користувацькому досвіду. Крім того, така взаємодія може генерувати цінні дані про поширені помилкові уявлення або запитання, які часто ставлять, що сприятиме постійному вдосконаленню стратегій викладання та дизайну контенту.

Таким чином, асистент зі штучним інтелектом втілює суть цілей платформи: підвищення ефективності, персоналізація навчального процесу і, зрештою, покращення освітніх результатів. Завдяки ітеративному вдосконаленню моделей машинного навчання та циклів зворотного зв'язку за участю учнів та викладачів, асистент ШІ може перетворитися на незамінного помічника в освітній екосистемі.

2.5 Огляд структури проекту

Зі скріншотів видно, що проект організовано у декілька ключових каталогів та файлів, які відображають багаторівневий підхід до розробки бекенд-версії. Нижче наведено високорівневий опис структури:

abl/: У цьому каталозі зберігається бізнес-логіка додатку, зазвичай розділена на підкаталоги (наприклад, курс, урок, модуль, питання, тест). Кожна підпапка, містить логіку, специфічну для певної функції або предметної області - наприклад, маніпуляції з даними, перевірку або оркестровку операцій. Тут також можуть знаходитися файли спільних утиліт (наприклад, constants.js або загальні точки входу в бізнес-логіку).

api/: Містить рівень, пов'язаний з API, включаючи контролери, обробку помилок, типи валідації та визначення попереджень. Саме тут визначаються маршрути або кінцеві точки, які зіставляються з основною бізнес-логікою.
components/: Місце для багаторазового використання модулів або будівельних блоків програми (наприклад, загальносистемні допоміжні програми або спільні сервіси). Наявність таких файлів, як components.js і module.js, свідчить про те, що ці скрипти можуть розкривати або реєструвати загальну функціональність, яку можуть використовувати інші частини програми.

dao/: Чіткий рівень «об'єктів доступу до даних», що містить специфічні для MongoDB операції для різних частин домену (наприклад, читання та запис до колекцій). Кожен файл DAO може відповідати окремій функції, наприклад, модулю, уроку або питанню. Загальний файл DAO (common-mongo.js) передбачає спільну логіку та утиліти, що використовуються окремими файлами DAO.

helpers/
Містить різні допоміжні скрипти, включаючи утиліти перевірки, логіку відкату та інші наскрізні питання. Ці допоміжні скрипти надають зручні методи, які можуть бути використані у різних частинах програми для послідовної, DRY (не повторюйся) розробки.

repository/
Папка репозиторію яка містить абстракції або класи, які представляють бізнес-об'єкти та їхню взаємодію з рівнем DAO. Ці файли визначають, як програма запитує і структурує дані перед тим, як повернути їх на верхні рівні (наприклад, ABL або API). Вони слугують мостом між необробленими даними та більш предметно-орієнтованою логікою.

Файли та папки верхнього рівня

env/: Потенційно може зберігати специфічні для середовища конфігурації (розробка, постановка, виробництво) або змінні середовища.

node_modules/: Стандартна папка залежностей Node.js.
public/, target/, test/: Додаткові директорії для публічних ресурсів, артефактів збірки або коду, пов'язаного з тестуванням, відповідно.

Конфігураційні файли: Конфігурації ESLint (.eslintrc.js), менеджери пакетів (.pnpmfile.cjs) та інші файли кореневого рівня, які керують тим, як додаток лінкується, збирається або тестується.

Як частини поєднуються між собою
Загалом, ця структура передбачає чіткий розподіл завдань:

Бізнес-логіка (в app/abl) фокусується на організації робочих процесів та інкапсуляції специфічних для домену правил.
Рівень API (в app/api) забезпечує точки входу для зовнішніх запитів, перетворюючи HTTP-виклики на логічні виклики в межах рівня ABL.
Рівень доступу до даних (у dao) інкапсулює безпосередню взаємодію з MongoDB, тоді як рівень репозиторію слугує посередницькою абстракцією, додаючи, зокрема, додаткову логіку обробки або перетворення даних.
Допоміжні модулі та компоненти уніфікують загальну функціональність і зменшують дублювання в різних модулях.
Конфігурація (у файлах config/ та .json верхнього рівня) уніфікує налаштування середовища, деталі збереження бази даних та профілі безпеки.

Така багаторівнева, модульна структура робить кодову базу більш зручною в обслуговуванні та масштабованою. Чітке призначення кожної папки для конкретного домену допомагає ефективно орієнтуватися та взаємодіяти з кодом, а також забезпечує гнучкість для додавання або модифікації нових функцій без заплутування не пов'язаних між собою логічних зв'язків.

2.6 Структура API

Цей підрозділ описує кінцеві точки API та їх конфігурацію. Система визначає набір варіантів використання (кінцевих точок) для різних функцій домену, таких як курси, модулі, уроки, тести та запитання, а також відповідні методи HTTP, відображення контролерів та вимоги до доступу. Крім того, конфігурація сховища даних зіставляє кожну сутність домену з об'єктом доступу до даних (DAO) на основі MongoDB. Нарешті, типи перевірки даних (DtoIn) забезпечують контракт для вхідних запитів, гарантуючи узгодженість і коректність на кожній кінцевій точці. [3,6]

1. Карта варіантів використання та прив'язки контролерів
Об'єкт useCaseMap визначає логічні кінцеві точки (часто звані «варіантами використання») програми. Кожен запис включає в себе

реалізація: Вказує на фактичний метод контролера, який обробляє запит (наприклад, api/controllers/CourseController.create).
httpMethod: Метод HTTP (наприклад, GET, POST), що використовується кінцевою точкою.
тип: Класифікація запиту, зазвичай CMD (командна операція, яка часто змінює дані) або UVE (візуальна кінцева точка користувача, яка часто обслуговує файли інтерфейсу користувача, такі як index.html).
Нижче наведено огляд основних кінцевих точок:

Кінцеві точки курсу

course/create (POST, CMD)

course/get (GET, CMD)

course/list (GET, CMD)

course/update (POST, CMD)

Вони керують операціями CRUD (створення, читання, оновлення, список) для сутностей курсу.

Кінцеві точки модулів

module/create (POST, CMD)

module/get (GET, CMD)

module/list (GET, CMD)

module/update (POST, CMD)

Ці команди обробляють операції на рівні модуля, часто пов'язані з курсами.

Кінцеві точки уроку

lesson/create (POST, CMD)

lesson/get (GET, CMD)

lesson/list (GET, CMD)

lesson/update (POST, CMD)

Кожен урок зазвичай знаходиться в межах модуля і може посилатися на тести або інші ресурси.

Кінцеві точки тесту

test/create (POST, CMD)

test/get (GET, CMD)

test/list (GET, CMD)

test/submit (POST, CMD)

test/update (POST, CMD)

Тести пов'язані з уроками або модулями, а кінцева точка «submit» дозволяє учням надсилати свої відповіді.

Кінцеві точки запитань

question/create (POST, CMD)

question/get (GET, CMD)

question/list (GET, CMD)

question/update (POST, CMD)

question/generate (GET, CMD)

question/answer (POST, CMD)

Ці кінцеві точки керують окремими питаннями, в тому числі генерують нові або надсилають/перевіряють відповіді.

Кінцева точка результатів

results/generateStatistics (GET, CMD)

Генерує статистику на рівні системи або користувача, наприклад, агреговані бали та аналітику продуктивності.

2.7 Огляд реалізації проекту

Нижче наведено структурований аналіз того, як сутність Question і пов'язані з нею блоки коду працюють в освітній системі, керованій ШІ. Це пояснення фокусується на логіці класу CreateAbl (зокрема, на обробці варіанту використання «створити»), визначенні сутності Question і допоміжних модулях, таких як CommonEntity, DtoInValidationHelper і NeuraLearn. Мета - прояснити обов'язки, взаємодію і потік даних при створенні або управлінні документами типу «Питання» в MongoDB.

1. Клас CreateAbl (основна логіка створення)

Файл: create-abl.js

Призначення: Цей клас ABL (Application Business Logic) інкапсулює процес створення нового Запитання в системі. Кожен крок позначений як «HDS» (High-level Design Step), що вказує на структуровану послідовність операцій.

```
"use strict";
const { NeuraLearn, Question } = require("../../repository");
const Errors = require("../../api/errors/question-error.js").Create;
const { DtoInValidationHelper, RollbackHelper } = require("../../helpers");
const { States } = require("../constants.js");

class CreateAbl {
  async create(uri, dtoIn, session) {
    const awid = uri.getAwid();

    // HDS 1
    const uuAppErrorMap = DtoInValidationHelper.validateDtoIn(uri, dtoIn, Errors.UC_CODE);

    // HDS 2
    await NeuraLearn.ensureInstanceAndState(awid, Errors.UC_CODE, States.NeuraLearn.expectedStates, uuAppErrorMap);

    // HDS 3
    const questionCreateDtoIn = this.#prepareQuestionCreateDtoIn(dtoIn, session);

    // HDS 4
    const rollbackHelper = new RollbackHelper();
    const errorConfig = { rollbackHelper, UC_CODE: Errors.UC_CODE, uuAppErrorMap };
    let question = await Question.create(awid, questionCreateDtoIn, errorConfig);

    // HDS 5
    const questionUpdateDtoIn = { state: States.Question.active };
    question = await Question.update(awid, question.id, questionUpdateDtoIn, errorConfig);

    // HDS 6
    return {
      ...question,
      uuAppErrorMap,
    };
  }

  #prepareQuestionCreateDtoIn(dtoIn, session) {
    const createDtoIn = { ...dtoIn };
    createDtoIn.creator = session.getIdentity().getUuIdentity();

    return createDtoIn;
  }
}

module.exports = new CreateAbl();
```

Рис. 2.1 Question create ABL

Валідація (DtoInValidationHelper.validateDtoIn)

Перевіряє відповідність вхідних даних (dtoIn) очікуваній схемі (наприклад, обов'язкові поля, довжини рядків, зчислення). Якщо перевірка не вдається, процес достроково переривається з помилкою.

Перевірка екземпляра та стану

Система посилається на батьківський екземпляр «NeuraLearn» (загальний запис програми). Вона перевіряє існування та відповідність дозволеним станам перед тим, як продовжити роботу.

Підготовка даних

Приватний метод #prepareQuestionCreateDtoIn додає ідентифікатор творця (session.getIdentity().getUuIdentity()) до корисного навантаження. Це гарантує, що кожен запис питання зберігає аудиторський слід того, хто його створив.

Створення з підтримкою відкату

Фактична вставка до бази даних виконується за допомогою Question.create. Об'єкт RollbackHelper відстежує зміни, дозволяючи програмі повернути частково завершені операції, якщо будь-який наступний крок не вдасться.

Оновлення стану

Після успішного створення поле стану питання стає активним. Цей крок фіналізує запис для використання в системі.

Результат

Фінальний об'єкт питання повертається разом з картою помилок (uuAppErrorMap), яка акумулює всі попередження або часткові помилки.

2. Об'єкт питання

Файл: question.js

Призначення: Клас Question розширює базовий клас CommonEntity, визначаючи специфічну для сутності логіку і посилаючись на MongoDB DAO. Він використовує визначення Schemas.question для збереження даних.

```
const { DaoFactory } = require("uu_appg01_server").ObjectStore;
const { Schemas : {...}  } = require("../abl/constants.js");
const CommonErrors = require("../api/errors/question-error.js").CommonQuestionErrors;
const CommonEntity : CommonEntity | {...}  = require("./common-entity.js");

class Question extends CommonEntity {
  constructor() {
    super();
    this.dao = DaoFactory.getDao(Schemas.question);
    this.generateErrors = CommonErrors;
  }

  // Entity-specific methods
}

module.exports = new Question();
```

Рис. 2.2 Question repository

Успадкування від CommonEntity

Розширюючи CommonEntity, клас Question автоматично отримує методи create, update, get та list. Цей патерн запобігає дублюванню коду в декількох сутностях (наприклад, Курс, Модуль, Урок).

Ініціалізація DAO

Об'єкт доступу до даних (DAO) отримується через DaoFactory.getDao(Schemas.question). Цей DAO відповідає за CRUD-операції над колекцією «question» в MongoDB.

Обробка помилок

generateErrors посилається на спеціальний набір помилок для питань (наприклад, CreateDaoFailed, UpdateDaoFailed тощо). Кожен метод може генерувати або обробляти специфічні для домену винятки, якщо щось піде не так під час операцій DAO.

3. Базовий клас CommonEntity

Файл: common-entity.js

Призначення: Надає загальні методи створення, оновлення, отримання та списку для дочірніх сутностей, таких як Питання, Тест, Урок тощо.

```
const { States :  } = require("../abl/constants");

class CommonEntity {
  constructor() {
    this.dao = null; // !!! must be set in child class
    this.generateErrors = null; // !!! must be set in child class
  }

  async create(awid, dtoIn, errorConfiguration) : Promise<any>  {...}

  async update(awid, id, uuObject, errorConfiguration) : Promise<any>  {...}

  async get(awid, dtoIn, uuAppErrorMap, UC_CODE, customError) : Promise<any>  {...}

  async list(awid, dtoIn) : Promise<any>  {...}

  async listByIdList(awid, idList, pageInfo) : Promise<any>  {...}
}

module.exports = CommonEntity;
```

Рис. 2.3 Common entity

```
async create(awid, dtoIn, errorConfiguration) : Promise<any>  {
  const { rollbackHelper, UC_CODE, uuAppErrorMap } = errorConfiguration;
  const { state : string  = States.created } = dtoIn;
  const Errors = this.generateErrors(UC_CODE);

  let entity;
  try {
    entity = await this.dao.create({ ...dtoIn, state, awid });
  } catch (e) {
    if (rollbackHelper) await rollbackHelper.runRollback();
    throw new Errors.CreateDaoFailed({ uuAppErrorMap }, e);
  }

  if (rollbackHelper) {
    rollbackHelper.addRollback( func: async () : Promise<void>  => {
      await rollbackHelper.rollbackDeleteWithWarning();
    });
  }

  return entity;
}
```

Рис. 2.4 Common entity create function

```
async update(awid, id, uuObject, errorConfiguration) : Promise<any>  {
  const { rollbackHelper, UC_CODE, uuAppErrorMap } = errorConfiguration;
  const Errors = this.generateErrors(UC_CODE);

  let entity;
  try {
    entity = await this.dao.update({ id, awid }, uuObject);
  } catch (e) {
    if (rollbackHelper) await rollbackHelper.runRollback();
    throw new Errors.UpdateDaoFailed({ uuAppErrorMap }, e);
  }

  if (rollbackHelper) {
    rollbackHelper.addRollback( func: async () : Promise<void>  => {
      await rollbackHelper.rollbackUpdateWithWarning();
    });
  }

  return entity;
}
```

Рис. 2.5 Common entity update function

```
async get(awid, dtoIn, uuAppErrorMap, UC_CODE, customError) : Promise<any>  {
  const Errors = this.generateErrors(UC_CODE);
  const { id, code } = dtoIn;
  const errorParamMap : {}  = {};

  let entity;
  if (id) {
    entity = await this.dao.get(awid, { id });
    errorParamMap.id = id;
  }

  if (!entity && code) {
    entity = await this.dao.get(awid, { code });
    errorParamMap.code = code;
  }

  if (!entity) {
    if (customError) throw new customError({ uuAppErrorMap }, errorParamMap);
    throw new Errors.EntityDoesNotExist({ uuAppErrorMap }, errorParamMap);
  }

  return entity;
}
```

Рис. 2.6 Common entity get function

```
async list(awid, dtoIn) : Promise<any>  {
  const { pageInfo, ...filters : Omit<any, "pageInfo">  } = dtoIn;
  return this.dao.list(awid, filters, pageInfo);
}

async listByIdList(awid, idList, pageInfo) : Promise<any>  {
  return this.dao.listByIdList(awid, idList, pageInfo);
}
```

Рис. 2.7 Common entity list and list by idList functions

Ключові моменти:

Методи create() та update() містять логіку відкату. Якщо виникає виняток, система може повернути частково виконані зміни.

Абстракція DAO: Кожна сутність посилається на свій власний DAO, гарантуючи, що бізнес-логіка не є тісно пов'язаною з логікою бази даних.
Ін'єкція помилок: Об'єкт errorConfiguration вводить UC_CODE (код варіанту використання), мапи помилок і допоміжні засоби для відкату, узгоджуючи обробку помилок з більш широким підходом програми.

Збираємо все воєдино
Коли надходить запит на створення запитання (наприклад, POST question/create):
Перевірка даних - запит проходить через DtoInValidationHelper.validateDtoIn.
Перевірка екземпляра - система підтверджує, що основний екземпляр «NeuraLearn» є дійсним і знаходиться в допустимому стані.
Construct DTO - до об'єкта даних додаються додаткові поля (ідентифікатор творця).
Операції з базою даних
Створити: Запис вставляється в колекцію питань за допомогою методу Question.create з CommonEntity.
Оновлення стану: новостворене питання переходить до стану «активне».
Результат - кінцевий об'єкт і будь-яка інформація про помилки або попередження (uuAppErrorMap) повертається користувачеві.
Така багаторівнева конструкція гарантує, що кожна частина логіки є ізольованою:
Перевірка знаходиться у DtoInValidationHelper.
Персистентність даних відбувається всередині Question DAO (через CommonEntity).
Високорівнева оркестровка (тобто кроки «HDS») обробляється класом CreateAbl, що відображає послідовний шаблон в інших сутностях (Курс, Модуль, Урок і т.д.).

Клас GetAbl (основна логіка «Get»)
Файл: get-abl.js

Призначення: Клас GetAbl інкапсулює послідовність кроків («HDS»), необхідних для отримання запису Question. Він організовує валідацію, перевірку стану та отримання даних з об'єкта доступу до даних Question DAO (Data Access Object).

```
"use strict";
const { Question : Question | {...} , NeuraLearn : NeuraLearn | {...}  } = require("../../repository");
const Errors : {UC_CODE: string}  = require("../../api/errors/question-error.js").Get;
const { DtoInValidationHelper : DtoInValidationHelper | {...}  } = require("../../helpers");
const { States :  } = require("../constants.js");

class GetAbl {
  async get(uri, dtoIn) : Promise<{...}>  {
    const awid : String  = uri.getAwid();

    // HDS 1
    const uuAppErrorMap = DtoInValidationHelper.validateDtoIn(uri, dtoIn, Errors.UC_CODE);

    // HDS 2
    await NeuraLearn.ensureInstanceAndState(awid, Errors.UC_CODE, States.NeuraLearn.expectedStates, uuAppErrorMap);

    // HDS 3
    const question = await Question.get(awid, dtoIn, uuAppErrorMap, Errors.UC_CODE);

    // HDS 4
    delete question.correctAnswer;
    return {
      ...question,
      uuAppErrorMap,
    };
  }
}

module.exports = new GetAbl();
```

Рис. 2.8 Question get ABL

Перевірка (HDS 1)

Викликає DtoInValidationHelper.validateDtoIn для перевірки відповідності вхідного dtoIn очікуваній схемі (наприклад, дійсний ідентифікатор або код).

Збирає будь-які попередження про валідацію або помилки у uuAppErrorMap.

Перевірка робочої області (HDS 2)

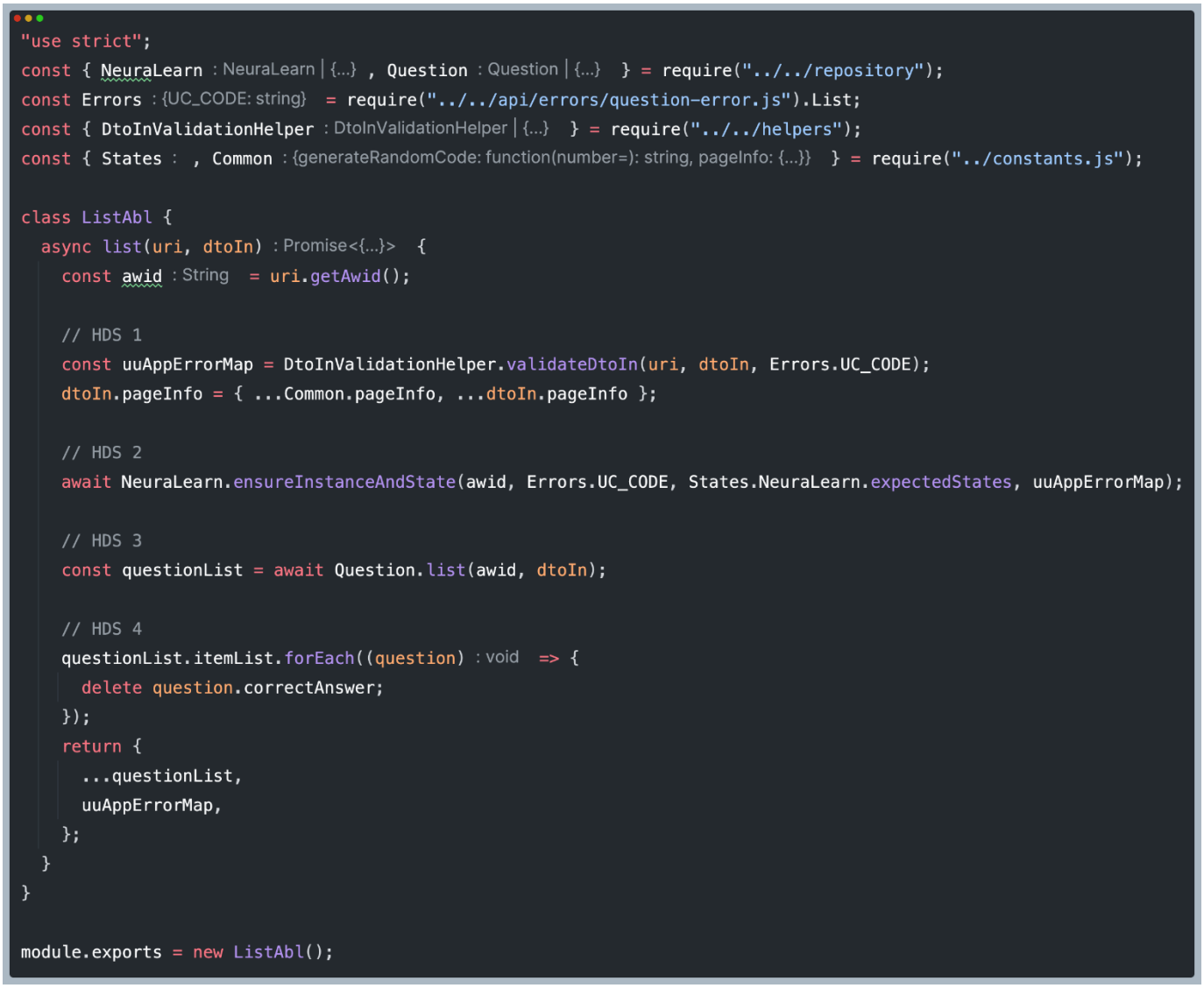
Викликає NeuraLearn.ensureInstanceAndState для перевірки того, що робоча область програми (ідентифікована за допомогою awid) існує і знаходиться у прийнятному стані.
Якщо робоча область не існує або знаходиться у неправильному стані (наприклад, «закрита» замість «активна»), метод генерує помилку.

Отримання даних (HDS 3)
Використовує метод Question.get зі сховища для запиту до MongoDB.
Цей виклик також передає uuAppErrorMap і унікальний UC_CODE (код варіанту використання) для послідовної обробки помилок.

Чутливість фільтрації (HDS 4)
Видаляє властивість correctAnswer з отриманого об'єкта питання перед його поверненням. Цей крок може бути заходом безпеки або для покращення користувацького досвіду, запобігаючи передчасному баченню правильної відповіді учнями.

Результат
Повертає об'єкт, що містить дані питання (за вирахуванням правильної відповіді) та uuAppErrorMap. Будь-які попередження або часткові помилки, що виникають під час перевірки або пошуку, включаються до карти помилок.

Клас ListAbl (основна логіка «списку»)
Файл: list-abl.js
Призначення: Клас ListAbl обробляє потік для отримання декількох сутностей Question у посторінковому режимі. Дотримуючись структурованого підходу (кроки «HDS»), він перевіряє вхідні дані, забезпечує належний стан робочої області програми та отримує записи питань через рівень репозиторію.



```
"use strict";
const { NeuraLearn : NeuraLearn | {...} , Question : Question | {...}  } = require("../../repository");
const Errors : {UC_CODE: string}  = require("../../api/errors/question-error.js").List;
const { DtoInValidationHelper : DtoInValidationHelper | {...}  } = require("../../helpers");
const { States :  , Common : {generateRandomCode: function(number=): string, pageInfo: {...}}  } = require("../constants.js");

class ListAbl {
  async list(uri, dtoIn) : Promise<{...}>  {
    const awid : String  = uri.getAwid();

    // HDS 1
    const uuAppErrorMap = DtoInValidationHelper.validateDtoIn(uri, dtoIn, Errors.UC_CODE);
    dtoIn.pageInfo = { ...Common.pageInfo, ...dtoIn.pageInfo };

    // HDS 2
    await NeuraLearn.ensureInstanceAndState(awid, Errors.UC_CODE, States.NeuraLearn.expectedStates, uuAppErrorMap);

    // HDS 3
    const questionList = await Question.list(awid, dtoIn);

    // HDS 4
    questionList.itemList.forEach((question) : void  => {
      delete question.correctAnswer;
    });
    return {
      ...questionList,
      uuAppErrorMap,
    };
  }
}

module.exports = new ListAbl();
```

Рис. 2.9 Question list ABL

Перевірка (HDS 1)

Викликає DtoInValidationHelper.validateDtoIn, щоб переконатися, що корисне навантаження запиту (dtoIn) відповідає необхідним схемам (наприклад, параметрам пагінації).

Об'єднує стандартну пагінацію (Common.pageInfo) з будь-якою інформацією про пагінацію, наданою користувачем.

Перевірка стану NeuraLearn (HDS 2)

Перевіряє, чи існує головний екземпляр «NeuraLearn» для поточного awid і чи перебуває робоча область у прийнятному стані (наприклад, «активна»).
Накопичує будь-які помилки або попередження у uuAppErrorMap.

Виклик сховища (HDS 3)
Виконує Question.list для виконання посторінкового запиту до колекції MongoDB.
Результат включає масив запитань (itemList), об'єкт pageInfo та будь-які інші релевантні метадані.

Фільтрація чутливості (HDS 4)
Перебирає знайдені запитання, видаляючи поле correctAnswer з кожного запису.
Гарантує, що правильні відповіді залишаться прихованими від учнів, надаючи лише основну інформацію у відповіді API.
Метод повернення
Об'єднує посторінкові результати з оновленою картою помилок uuAppErrorMap.
Клієнт отримує очищений список запитань, кожен з яких позбавлений ключа до відповіді.

Клас UpdateAbl (основна логіка «оновлення»)
Файл: update-abl.js
Призначення: Цей клас організовує кроки («HDS»), необхідні для оновлення існуючого документа Question. Процес гарантує, що стан робочої області є дійсним, отримує запис і застосовує внесені користувачем зміни через сховище запитань.

```
"use strict";
const { NeuraLearn : NeuraLearn | {...} , Question : Question | {...}  } = require("../../repository");
const Errors : {UC_CODE: string}  = require("../../api/errors/lesson-error.js").Update;
const { DtoInValidationHelper : DtoInValidationHelper | {...}  } = require("../../helpers");
const { States :  } = require("../constants.js");

class UpdateAbl {
  async update(uri, dtoIn) : Promise<{...}>  {
    const awid : String  = uri.getAwid();

    // HDS 1
    const uuAppErrorMap = DtoInValidationHelper.validateDtoIn(uri, dtoIn, Errors.UC_CODE);

    // HDS 2
    await NeuraLearn.ensureInstanceAndState(awid, Errors.UC_CODE, States.NeuraLearn.expectedStates, uuAppErrorMap);

    // HDS 3
    const question = await Question.get(awid, dtoIn, uuAppErrorMap, Errors.UC_CODE);

    // HDS 4
    const updateDtoIn = { ...dtoIn };

    // HDS 5
    const errorConfig : {UC_CODE: string, uuAppErrorMap: any}  = { UC_CODE: Errors.UC_CODE, uuAppErrorMap };
    const updatedQuestion = await Question.update(awid, question.id, updateDtoIn, errorConfig);

    // HDS 6
    return {
      ...updatedQuestion,
      uuAppErrorMap,
    };
  }
}

module.exports = new UpdateAbl();
```

Рис. 2.10 Question update ABL

Перевірка (HDS 1)

Використовує DtoInValidationHelper.validateDtoIn для підтвердження відповідності корисного навантаження запиту (dtoIn) необхідній схемі (наприклад, допустимі поля, типи).

Агрегує попередження або часткові помилки у uuAppErrorMap.

Перевірка екземпляру та стану (HDS 2)

Викликає NeuraLearn.ensureInstanceAndState для перевірки того, що робоча область програми (ідентифікована за допомогою awid) існує і їй дозволено виконувати операції оновлення (наприклад, активний стан).
Записує будь-які проблеми у uuAppErrorMap.

Отримання записів (HDS 3)
Викликає функцію Question.get для отримання поточної сутності Question за dtoIn.id (або кодом, якщо схема це підтримує).
Якщо запис не існує або виклик DAO завершився невдало, він видає відповідну помилку.

Підготовка даних (HDS 4)
Копіює перевірені дані (dtoIn) до updateDtoIn. Це гарантує, що для оновлення будуть використані лише перевірені поля (хоча за потреби можна виконати подальшу фільтрацію).

Операція оновлення (HDS 5)
Створює об'єкт errorConfig з поточною картою помилок і UC_CODE.
Викликає Question.update з ідентифікатором запису і новими даними.
Цей метод використовує базовий клас CommonEntity для стандартизованої підтримки відкату та генерування помилок у разі збою операції DAO.

Результат виконання (HDS 6)
Об'єднує оновлений об'єкт Question з uuAppErrorMap.
Повертає користувачеві фінальний запис, що відображає будь-які нові або змінені властивості.

Клас GenerateAbl (основна логіка «Generate»)

Файл: generate-abl.js

Призначення: Цей клас обробляє створення нового питання через OpenAI API. Він бере тему, надану користувачем, надсилає її до моделі чату та очікує відповідь у форматі JSON, що містить опис питання, варіанти відповідей та правильну відповідь (відповіді).

```
"use strict";
const { Config } = require("uu_appg01_server").Utils;
const { NeuraLearn : NeuraLearn | {...}  } = require("../../repository");
const Errors : {UC_CODE: string}  = require("../../api/errors/question-error.js").Get;
const { DtoInValidationHelper : DtoInValidationHelper | {...}  } = require("../../helpers");
const { States :  } = require("../constants.js");
const OpenAI : OpenAI |  = require("openai");

class GenerateAbl {
  async generate(uri, dtoIn) : Promise<{...}>  {
    const awid : String  = uri.getAwid();

    // HDS 1
    const uuAppErrorMap = DtoInValidationHelper.validateDtoIn(uri, dtoIn, Errors.UC_CODE);

    // HDS 2
    await NeuraLearn.ensureInstanceAndState(awid, Errors.UC_CODE, States.NeuraLearn.expectedStates, uuAppErrorMap);

    // HDS 3
    const apiKey = Config.get("openaiApiKey");
    const openai : OpenAI  = new OpenAI( {baseURL, apiKey, organization, project, ...opts}: { apiKey });

    const completion : ChatCompletion & {...}  = await openai.chat.completions.create( body: {
      model: "gpt-4o-mini",
      messages: [
        { role: "system", content: "You are a helpful assistant that create test question based on topic." },
        {
          role: "user",
          content: `Generate a new question about: ${dtoIn.text}.`,
        },
        {
          role: "system",
          content: `Return response in the following JSON parsable format:

        {
           "type": "String (one of 'multiple_answers', 'single_answer')",
           "content": "String (text of the question)",
           "options": ["String"] (list of answer options),
           "correctAnswer": "String or [String] (correct answer or answers)",
           "difficulty": "Sting (one of 'easy', 'medium', 'hard')",
         }
        `,
        },
      ],
    });

    // HDS 4
    return {
      ...JSON.parse(completion.choices[0].message.content),
      uuAppErrorMap,
    };
  }
}

module.exports = new GenerateAbl();
```

Рис. 2.11 Question generate ABL

Перевірка (HDS 1)

Викликає DtoInValidationHelper.validateDtoIn, щоб переконатися, що вхідні дані відповідають схемі (перевіряє, що dtoIn.text є непорожнім рядком).

Перевірка стану NeuraLearn (HDS 2)

Перевіряє, що основний екземпляр NeuraLearn у системі (ідентифікований за допомогою awid) присутній і перебуває у прийнятному стані.

Якщо робоча область програми налаштована неправильно (наприклад, не «активна»), процес зупиняється.

Виклик OpenAI API (HDS 3)

Отримує openaiApiKey з конфігурації за допомогою Config.get(«openaiApiKey»).

Викликає метод openai.chat.completions.create з власним системним повідомленням, запитом користувача та системною інструкцією, що визначає формат виводу JSON.

Обрана модель - «gpt-4o-mini».

Фінальна інструкція системи створює відповідь у форматі JSON, що містить деталі питання (тип, зміст, варіанти, правильна відповідь, складність).

Розбір і повернення відповіді (HDS 4)

Розбирає згенероване ШІ корисне навантаження JSON з completion.choices[0].message.content.

Об'єднує всі наявні повідомлення про попередження/помилки у uuAppErrorMap у кінцевий об'єкт.

Повернутий об'єкт інкапсулює новостворену схему питання, придатну для подальшого зберігання або візуалізації.

Клас AnswerAbl (основна логіка «відповіді»)

Файл: answer-abl.js

Призначення: Цей клас забезпечує логіку обробки відповіді на конкретне Питання. Він слідує стандартному підходу високорівневого проектування (HDS), який використовується у всіх ABL-модулях програми.

```
"use strict";
const { NeuraLearn : NeuraLearn | {...} , Question : Question | {...}  } = require("../../repository");
const Errors : {UC_CODE: string}  = require("../../api/errors/question-error.js").Create;
const { DtoInValidationHelper : DtoInValidationHelper | {...}  } = require("../../helpers");
const { States :  } = require("../constants.js");
const Results : Results | {...}  = require("../../components/results.js");

class AnswerAbl {
  async answer(uri, dtoIn, session) : Promise<{...}>  {
    const awid : String  = uri.getAwid();

    // HDS 1
    const uuAppErrorMap = DtoInValidationHelper.validateDtoIn(uri, dtoIn, Errors.UC_CODE);

    // HDS 2
    await NeuraLearn.ensureInstanceAndState(awid, Errors.UC_CODE, States.NeuraLearn.expectedStates, uuAppErrorMap);

    // HDS 3
    await Question.get(awid, dtoIn, uuAppErrorMap, Errors.UC_CODE);

    // HDS 4
    const checkResultDtoIn :{...}  = {
      questionId: dtoIn.id,
      answer: dtoIn.answer,
      session,
    };

    const result = await Results.checkResultAndStoreByQuestionId(awid, checkResultDtoIn,  errorConfig: {
      uuAppErrorMap,
      UC_CODE: Errors.UC_CODE,
    });

    // HDS 6
    return {
      isCorrect: result.isCorrect,
      ...(!result.isCorrect && { correctAnswer: result.correctAnswer }),
      uuAppErrorMap,
    };
  }
}

module.exports = new AnswerAbl();
```

Рис. 2.12 Question answer ABL

Перевірка (HDS 1)

Використовує DtoInValidationHelper.validateDtoIn, щоб переконатися, що тіло запиту (dtoIn) відповідає схемі (наприклад, ідентифікатор питання, масив відповідей або рядок).

Будь-які попередження перевірки або часткові помилки потрапляють до uuAppErrorMap.

Перевірка робочої області (HDS 2)

Перевіряє, що екземпляр NeuraLearn для цього awid є дійсним і знаходиться у дозволеному стані (наприклад, «активний»).

Якщо екземпляр недійсний, генерується помилка, в іншому випадку дані зберігаються у uuAppErrorMap.

Отримання питання (HDS 3)

Викликає метод Question.get, щоб підтвердити, що потрібне питання існує.

Якщо питання відсутнє, метод видає помилку, специфічну для домену (наприклад, «EntityDoesNotExist»).

Підготовка та зберігання результатів (HDS 4)

Створює checkResultDtoIn з основними даними: questionId, відповідь і поточний сеанс користувача.

Передає ці дані до Results.checkResultAndStoreByQuestionId, делегуючи перевірку коректності та зберігання даних.

Метод повернення (HDS 6)

Метод повертає прапорець isCorrect.

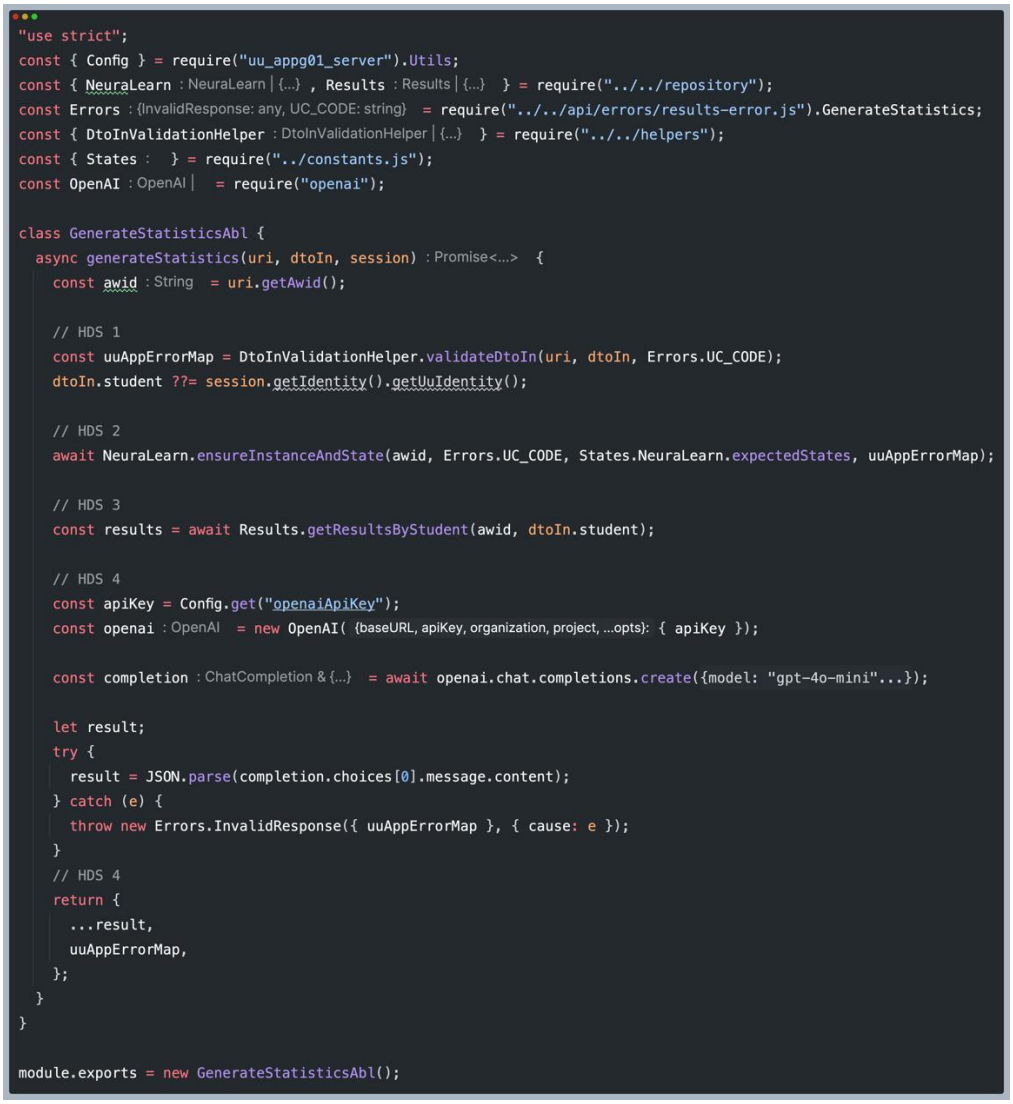
Якщо користувач помилився, він також включає у відповідь correctAnswer.

Включає uuAppErrorMap для будь-яких помилок або попереджень на цьому шляху.

Клас GenerateStatisticsAbl (основна логіка «генерації статистики»)

Файл: generate-statistics-abl.js

Призначення: Забезпечує високорівневий процес збору та узагальнення даних, пов'язаних з учнями. Зокрема, він збирає всі результати, отримані певним користувачем, а потім передає ці дані моделі штучного інтелекту, яка виробляє всебічний аналіз і рекомендації у форматі JSON



```
"use strict";
const { Config } = require("uu_appg01_server").Utils;
const { NeuraLearn : NeuraLearn | {...} , Results : Results | {...}  } = require("../../repository");
const Errors : {InvalidResponse: any, UC_CODE: string}  = require("../../api/errors/results-error.js").GenerateStatistics;
const { DtoInValidationHelper : DtoInValidationHelper | {...}  } = require("../../helpers");
const { States :  } = require("../constants.js");
const OpenAI : OpenAI |  = require("openai");

class GenerateStatisticsAbl {
  async generateStatistics(uri, dtoIn, session) : Promise<...>  {
    const awid : String  = uri.getAwid();

    // HDS 1
    const uuAppErrorMap = DtoInValidationHelper.validateDtoIn(uri, dtoIn, Errors.UC_CODE);
    dtoIn.student ??= session.getIdentity().getUuIdentity();

    // HDS 2
    await NeuraLearn.ensureInstanceAndState(awid, Errors.UC_CODE, States.NeuraLearn.expectedStates, uuAppErrorMap);

    // HDS 3
    const results = await Results.getResultsByStudent(awid, dtoIn.student);

    // HDS 4
    const apiKey = Config.get("openaiApiKey");
    const openai : OpenAI  = new OpenAI( {baseURL, apiKey, organization, project, ...opts}: { apiKey });

    const completion : ChatCompletion & {...}  = await openai.chat.completions.create({model: "gpt-4o-mini"...});

    let result;
    try {
      result = JSON.parse(completion.choices[0].message.content);
    } catch (e) {
      throw new Errors.InvalidResponse({ uuAppErrorMap }, { cause: e });
    }
    // HDS 4
    return {
      ...result,
      uuAppErrorMap,
    };
  }
}

module.exports = new GenerateStatisticsAbl();
```

Рис. 2.13 Generate statistics ABL

```
{
  "studentId": "String (unique identifier for the student, e.g., '4120-9691-2145-0000')",
  "overallStatistics": {
    "totalQuestions": "Integer (total number of questions attempted)",
    "correctAnswers": "Integer (total number of correct answers)",
    "incorrectAnswers": "Integer (total number of incorrect answers)",
    "score": "String (percentage score, e.g., '72.73%')"
  },
  "testStatistics": [
    {
      "testId": "String (unique identifier for the test, e.g., '675dd4790b963f430c23820f')",
      "testName": "String (name of the test, e.g., 'Basics of arithmetic')",
      "totalQuestions": "Integer (number of questions in this test)",
      "correctAnswers": "Integer (correct answers in this test)",
      "incorrectAnswers": "Integer (incorrect answers in this test)",
      "score": "String (percentage score for this test, e.g., '66.67%')",
      "questions": [
        {
          "questionId": "String (unique identifier for the question, e.g., '675dd423dceb60e235601c54')",
          "type": "String (one of 'multiple_answers', 'single_answer')",
          "content": "String (text of the question, e.g., 'What is the sum of 27 and 45?')",
          "options": ["String (list of answer options, e.g., ['62', '72', '52', '82'])"],
          "correctAnswer": "String or [String] (correct answer or answers, e.g., '72')",
          "difficulty": "String (one of 'easy', 'medium', 'hard')",
          "answer": "String or [String] (answer provided by the student, e.g., '72')",
          "isCorrect": "Boolean (true if the student's answer is correct, false otherwise)"
        }
      ]
    }
  ],
  "questionStatistics": [
    {
      "questionId": "String (unique identifier for the question, e.g., '675dd423dceb60e235601c54')",
      "content": "String (text of the question, e.g., 'What is the sum of 27 and 45?')",
      "type": "String (one of 'multiple_answers', 'single_answer')",
      "options": ["String (list of answer options, e.g., ['62', '72', '52', '82'])"],
      "correctAnswer": "String or [String] (correct answer or answers, e.g., '72')",
      "difficulty": "String (one of 'easy', 'medium', 'hard')",
      "attempts": "Integer (total number of attempts for this question)",
      "correctAttempts": "Integer (number of correct attempts)",
      "incorrectAttempts": "Integer (number of incorrect attempts)",
      "accuracy": "String (percentage of correct attempts, e.g., '57.14%')"
    }
  ],
  "examStatistics": {
    "examAttempts": "Integer (total number of questions attempted in exams)",
    "correctAnswers": "Integer (total correct answers in exams)",
    "incorrectAnswers": "Integer (total incorrect answers in exams)",
    "score": "String (percentage score for exams, e.g., '66.67%')"
  },
  "practiceStatistics": {
    "practiceAttempts": "Integer (total number of questions attempted in practice)",
    "correctAnswers": "Integer (total correct answers in practice)",
    "incorrectAnswers": "Integer (total incorrect answers in practice)",
    "score": "String (percentage score for practice, e.g., '75%')"
  }
  "suggestions": [
    {
      "category": "String (e.g., 'overall', 'test', 'question', 'exam', 'practice')",
      "target": "String or Object (specific ID or broader context, e.g., testId or questionId)",
      "message": "String (personalized suggestion for improvement)",
      "recommendationType": "String (e.g., 'review_material', 'practice_more', 'focus_area', 'positive_feedback')"
    }
  ]
}
```

Рис. 2.14 Statistics structure

Перевірка та скидання за замовчуванням (HDS 1)
Використовує DtoInValidationHelper.validateDtoIn для перевірки дійсності dtoIn.
Якщо dtoIn.student не надано, за замовчуванням використовується ідентифікатор поточного користувача (session.getIdentity().getUuIdentity()).

Перевірка стану NeuraLearn (HDS 2)
Переконується, що робоча область (awid) є дійсною і знаходиться у прийнятному стані (наприклад, «активна»).
Якщо робоча область не налаштована належним чином, виникає помилка.

Отримати результати (HDS 3)
Викликає Results.getResultsByStudent, який, імовірно, запитує колекцію «results» в MongoDB для збору всіх спроб або відповідей, пов'язаних з певним студентом.
Повернута структура даних передається до ШІ-моделі на наступному кроці.

Аналіз на основі ШІ (HDS 4)
Зчитує openaiApiKey з конфігурації програми (Config.get(«openaiApiKey»)).
Ініціює запит на отримання відповіді від моделі OpenAI («gpt-4o-mini»). Включає в себе
Системне повідомлення з інструкціями для AI генерувати статистику та пропозиції.
Повідомлення користувача, що містить серіалізовані дані результатів.
Ще одне системне повідомлення з детальним описом необхідного формату відповіді у JSON.
Відповідь AI розбирається за допомогою JSON.parse. Якщо розбір не вдається, код видає специфічну для домену помилку InvalidResponse.

Метод повернення

Об'єднує об'єкт результату, згенерований ШІ, з будь-якими накопиченими повідомленнями про помилки (uuAppErrorMap).

Остаточна структура JSON відповідає схемі, запитаній у підказці системи, і містить такі дані, як testStatistics, questionStatistics та пропозиції щодо покращення.

Приклад згенерованої статистики:


```
{
  "studentId": "4120-9691-2145-0000",
  "overallStatistics": {
    "totalQuestions": 10,
    "correctAnswers": 6,
    "incorrectAnswers": 4,
    "score": "60.00%"
  },
  "testStatistics": [
    {
      "testId": "675dd4790b963f430c23820f",
      "testName": "Basics of arithmetic",
      "totalQuestions": 3,
      "correctAnswers": 2,
      "incorrectAnswers": 1,
      "score": "66.67%",
      "questions": [
        {
          "questionId": "675dd423dceb60e235601c54",
          "type": "single_answer",
          "content": "What is the sum of 15 and 28?",
          "options": [...],
          "correctAnswer": "43",
          "difficulty": "easy",
          "answer": "43",
          "isCorrect": true
        },
        {"questionId": "675dd417dceb60e235601c4d"...},
        {"questionId": "675dcc343f008d37ec5756e6"...}
      ]
    }
  ],
  "questionStatistics": [
    {
      "questionId": "675dcc343f008d37ec5756e6",
      "content": "What is the sum of 27 and 45?",
      "type": "single_answer",
      "options": [...],
      "correctAnswer": "72",
      "difficulty": "easy",
      "attempts": 5,
      "correctAttempts": 2,
      "incorrectAttempts": 3,
      "accuracy": "40.00%"
    },
    {"questionId": "675dd423dceb60e235601c54"...},
    {"questionId": "675dd417dceb60e235601c4d"...}
  ],
  "examStatistics": {
    "examAttempts": 5,
    "correctAnswers": 2,
    "incorrectAnswers": 3,
    "score": "40.00%"
  },
  "practiceStatistics": {
    "practiceAttempts": 5,
    "correctAnswers": 4,
    "incorrectAnswers": 1,
    "score": "80.00%"
  },
  "suggestions": [
    {
      "category": "overall",
      "target": "student",
      "message": "Consider reviewing basic addition principles to improve overall accuracy.",
      "recommendationType": "review_material"
    },
    {
      "category": "question",
      "target": "675dcc343f008d37ec5756e6",
      "message": "Try practicing similar addition questions to build confidence.",
      "recommendationType": "practice_more"
    },
    {
      "category": "test",
      "target": "675dd4790b963f430c23820f",
      "message": "Focus on exam settings to help simulate test conditions and reduce errors.",
      "recommendationType": "focus_area"
    }
  ],
  "uuAppErrorMap": {}
}
```


Рис. 2.15 Generated statistics example

РОЗДІЛ 3. ТЕСТУВАННЯ ТА ОЦІНКА РЕЗУЛЬТАТІВ

3.1 Підходи до тестування

Створення надійної освітньої платформи зі штучним інтелектом вимагає системного підходу до тестування, який гарантує, що кожен рівень - від обробки даних до алгоритмів машинного навчання - функціонує послідовно і правильно. Наявність логіки персоналізації, динамічної генерації контенту та складних взаємодій з даними вимагає більшого, ніж поверхнева перевірка. Нижче наведено ключові підходи до тестування, які відповідають архітектурі та функціям системи:

Модульне тестування

Гранулярний фокус: Тести ізолюють невеликі частини коду (наприклад, окремі функції або методи), такі як допоміжні засоби перевірки даних, утиліти або DTO-перетворення.

Імітація зовнішніх залежностей: Можна імітувати компоненти, які звертаються до служб ШІ або баз даних, що дозволяє тесту перевірити логіку ізольовано, без накладних витрат на мережу або введення/виведення.

Типові інструменти: Jest або Mocha (для Node.js) добре працюють тут. Вони дозволяють перевіряти отримані результати, повідомлення про помилки та загальний стан системи.

Тестування інтеграції

Міжкомпонентна перевірка: Інтеграційні тести підтверджують, що модулі, мікросервіси та зовнішні API (наприклад, OpenAI) взаємодіють належним чином.

Взаємодія з базами даних: Кожен тест зазвичай використовує локальний екземпляр MongoDB або базу даних в пам'яті (наприклад, MongoDB Memory Server), щоб забезпечити коректне функціонування рівня DAO (Data Access Object).

Кінцеві точки API: Тестування кінцевих точок, таких як POST /question/create або GET /lesson/list, допомагає переконатися, що RESTful шаблони, потоки автентифікації та механізми перевірки даних узгоджуються належним чином.

Функціональне тестування

Сценарії, орієнтовані на користувача: Ці тести імітують історії користувачів або наскрізні робочі процеси. Наприклад, створення нового курсу, додавання модулів та уроків, а потім призначення тестів і надсилання відповідей.

Кілька профілів: Перевірки системи на основі ролей (наприклад, «Авторизація», «Публічний», «Аутентифікація») вимагають перевірки того, що певні кінцеві точки є обмеженими або відкритими, як визначено в sysStateList і profileList.

Аналіз результатів: Зокрема, в середовищі зі штучним інтелектом функціональні тести підтверджують, що персоналізовані рекомендації та зворотний зв'язок у реальному часі відображають правильну логіку.

Регресійне тестування

Безперервна перевірка: Після впровадження нових функцій або виправлень повторний запуск регресійних тестів гарантує, що існуючі функціональні можливості залишаться недоторканими.

Автоматизація в CI/CD: Інтеграція регресійних тестів в конвеєр безперервної інтеграції (GitHub Actions, GitLab CI) допомагає виявити ненавмисні поломки на ранніх стадіях.

Тестування продуктивності та навантаження

Перевірка масштабованості: Зі збільшенням кількості користувачів платформа може обробляти великі обсяги заявок на участь у вікторинах, запитів до штучного інтелекту та аналітики даних. Такі інструменти, як JMeter або Artillery, можуть імітувати одночасні сеанси для вимірювання часу відгуку і використання ресурсів.

Виявлення вузьких місць: Стрес-тести можуть виявити проблеми з продуктивністю при взаємодії мікросервісів, запитах до баз даних або кінцевих точок, керованих штучним інтелектом, що допоможе оптимізувати кешування та архітектуру.

Тестування на безпеку та проникнення
Доступ на основі ролей: Перевіряє, що тільки авторизовані ролі можуть змінювати курси, модулі або системні налаштування.
Захист даних: Перевіряє, чи належним чином зашифровані конфіденційні дані користувача (наприклад, показники продуктивності, особиста інформація) і чи не розкриваються вони в журналах або повідомленнях про помилки.
Зміцнення API: Гарантує, що неправильно сформовані або зловмисні запити не призведуть до небажаних наслідків або порушень безпеки.
Використовуючи ці підходи в унісон, можна систематично перевіряти надійність, масштабованість і безпеку системи. Таким чином, освітнє середовище захищене від регресії, несанкціонованих маніпуляцій з даними та необроблених помилок - факторів, які є критично важливими для довіри студентів та інституційного прийняття.

3.2 Інструменти тестування

Ефективне тестування та налагодження залежить від правильного вибору інструментів. Поєднання сучасних тестових фреймворків, рішень для автоматизації та спеціалізованих утиліт забезпечує швидкі цикли зворотного зв'язку та стабільну якість для різноманітних функцій. Наступні інструменти зазвичай використовуються в проектах на основі Node.js, інтегрованих з сервісами штучного інтелекту та MongoDB:

Jest

Призначення: Основний фреймворк модульного тестування для середовищ JavaScript/TypeScript.

Переваги: Вбудовані функції імітації, тестування знімків та режиму спостереження. Добре поєднується зі структурою коду мікросервісів.

Варіанти використання: Тестування окремих класів (наприклад, CreateAbl, GetAbl), допоміжних засобів перевірки та функцій перетворення даних.

Mocha & Chai

Призначення: Ще одна популярна комбінація для тестування в стилі BDD/TDD в Node.js.

Переваги: Гнучка структура, багата бібліотека тверджень (Chai) та екосистема плагінів для покриття, імітації тощо.

Варіанти використання: Інтеграційні тести, які вимагають більшої конфігурації або певного стилю, якому надає перевагу команда розробників.

Supertest

Призначення: Бібліотека, спеціально розроблена для тестування кінцевих точок HTTP.

Переваги: Спрощує надсилання запитів до експрес-додатків, дозволяючи здійснювати наскрізні перевірки без ручного запуску серверів.

Варіанти використання: Перевірка кінцевих точок RESTful, включаючи входи/виходи API, обробку помилок і коди відповідей.

MongoDB Memory Server

Призначення: Екземпляр MongoDB в пам'яті, який використовується під час тестування, щоб уникнути зовнішніх залежностей.

Переваги: Ізолює тести від виробничих даних, прискорює виконання тестів і скидає стан бази даних перед кожним тестом.
Варіанти використання: Інтеграційні тести, які потребують реалістичної взаємодії DAO без накладних витрат на налаштування локальних або віддалених екземплярів MongoDB.

Artillery / JMeter
Призначення: Тестування навантаження і продуктивності.
Переваги: Генерування одночасного трафіку користувачів і збір показників продуктивності за різними сценаріями (імітуючи реальні піки використання).
Варіанти використання: Забезпечення здатності кінцевих точок і мікросервісів штучного інтелекту масштабуватися і залишатися швидкодіючими під час пікових навантажень (наприклад, великі онлайн-оцінювання).

SonarQube / ESLint
Призначення: Якісне сканування та лінтування.
Переваги: Виявляє запахи коду, вразливості в системі безпеки та забезпечує дотримання узгоджених стильових конвенцій.
Варіанти використання: Автоматизовані перевірки в конвеєрах CI/CD для підтримки високих стандартів цілісності та читабельності коду.

Інструменти для імітації OpenAI
Призначення: Оскільки система взаємодіє із зовнішнім ШІ (наприклад, моделями GPT-4), спеціалізовані рішення для імітації або ручні дублери тестів можуть відтворювати відповіді OpenAI.
Переваги: Дозволяє проводити офлайн-тестування, отримувати детерміновані результати та запобігає витратам на API під час кожного запуску тесту.

Варіанти використання: Кінцеві точки, такі як «generateStatistics» або «generateQuestion», можна протестувати на правильність обробки відповідей, не покладаючись на виклики OpenAI в реальному часі.

Отже, тестування та налагодження освітньої платформи зі штучним інтелектом за своєю суттю є багатогранним процесом, що охоплює все - від базових перевірок блоків до складного навантажувального тестування та перевірки реакції ШІ. Використовуючи добре зарекомендували себе інструменти (Jest, Mocha, Supertest), а також спеціалізовані рішення (MongoDB Memory Server, макети OpenAI), ми гарантуємо, що кожен рівень - база даних, логіка програми, інтеграція ШІ та інтерфейсні API - залишатиметься стабільним за різних сценаріїв. Такий суворий підхід до тестування не тільки підвищує якість програмного забезпечення, але й сприяє зміцненню довіри серед студентів, викладачів та адміністраторів, які покладаються на послідовний, точний і безпечний навчальний процес.

3.3 Досягнення поставлених цілей

Під час розробки прототипу освітньої платформи зі штучним інтелектом ми прагнули досягти кількох основних цілей, а саме: покращити персоналізацію, автоматизувати рутинні завдання та надати можливість приймати рішення на основі даних як учням, так і викладачам. Аналізуючи наш прогрес, ми можемо зробити висновок, що наступні ключові цілі були досягнуті:

Розробка функціонального прототипу:
Ми успішно створили прототип системи, яка інтегрує механізм рекомендацій на основі ШІ, динамічне відстеження прогресу та модулі адаптивного оцінювання. Цей прототип демонструє, що поєднання алгоритмів машинного навчання з надійною архітектурою бази даних і добре продуманими API дозволяє освітнім платформам пропонувати більш цікавий і гнучкий навчальний процес. Навіть на цьому етапі

прототип вирішує багато проблем, з якими стикаються традиційні платформи, такі як статичні навчальні матеріали, універсальні оцінки та обмежена видимість успішності учнів.

Впровадження основних компонентів штучного інтелекту:
Наше рішення використовує можливості штучного інтелекту в декількох сферах, включаючи персоналізовані пропозиції контенту, інтелектуальну генерацію запитань та аналітику ефективності. Кожен з цих компонентів посилює адаптивність платформи, допомагаючи викладачам зосередитися на завданнях вищого рівня, таких як розробка навчальних програм і наставництво, а не на ручному оцінюванні та повторюваному зворотному зв'язку.

Перевірка основних варіантів використання:
За допомогою початкових тестових сценаріїв та симуляцій користувачів ми підтвердили, що система може керувати ключовими освітніми робочими процесами: створювати та оновлювати контент курсу, генерувати тести та запитання, фіксувати відповіді учнів та надавати негайний зворотній зв'язок. Ці перевірки демонструють потенціал платформи для зменшення адміністративних витрат та оптимізації навчального процесу.

Масштабованість:
Розробляючи прототип з архітектурою, орієнтованою на мікросервіси (на основі Unicorn Application Framework та MongoDB), ми врахували майбутні потреби в масштабуванні. Система може обробляти зростаючу кількість користувачів і наборів даних, підкреслюючи, що інфраструктура є одночасно економічно ефективною і легко розширюваною.

Фундамент для розширеної аналітики:

Ми створили механізми для реєстрації даних про взаємодію (наприклад, відповіді студентів, метрики використання), що створює платформу для розширеної аналітики. Хоча ці конвеєри даних ще перебувають на ранніх стадіях, вони можуть сприяти глибшому розумінню поведінки учнів, уможливлюючи прогностичне моделювання та проактивні втручання для підвищення рівня утримання та завершення навчання.

3.4 Переваги та обмеження системи

Вивчаючи наш прототип освітньої платформи зі штучним інтелектом, важливо виділити сильні сторони, які позиціонують її як трансформаційне рішення, а також визнати обмеження, з якими вона наразі стикається. Ці фактори визначають дорожню карту майбутнього розвитку та допомагають зацікавленим сторонам оцінити готовність платформи до розгортання в реальних умовах.

Переваги

Адаптивне навчання

В основі системи лежать передові механізми рекомендацій та персоналізованої доставки контенту. Постійно аналізуючи дані про успішність учнів і моделі взаємодії, платформа адаптує рівні складності, пропонує додаткові матеріали та надає цільовий зворотний зв'язок. Такий підхід, орієнтований на учня, значно підвищує його зацікавленість і може призвести до покращення результатів навчання.

Оптимізоване адміністрування

Традиційні освітні процеси часто включають в себе трудомісткі завдання, такі як ручне виставлення оцінок, повторний зворотний зв'язок або створення оцінок. Наші модулі штучного інтелекту автоматизують ці завдання, дозволяючи викладачам зосередитися на більш важливих обов'язках, таких як розробка навчальних планів і

наставництво. Модульна архітектура системи ще більше спрощує розгортання, обслуговування та поступове оновлення функцій.

Розширювана архітектура

Орієнтований на мікросервіси дизайн, що базується на Unicorn Application Framework (UAF) та MongoDB, дозволяє здійснювати як горизонтальне, так і вертикальне масштабування. Нові функціональні можливості, такі як додаткові типи запитань, розширені аналітичні модулі або інтеграція зі стороннім програмним забезпеченням, можуть бути легко інтегровані. Така розширюваність робить прототип адаптованим до широкого спектру інституційних потреб та освітніх моделей.

Аналітика на основі даних

Функції реєстрації та аналітики платформи фіксують детальні показники використання, дозволяючи викладачам та адміністраторам виявляти прогалини в знаннях студентів, виявляти нові тенденції та точно налаштовувати навчальні траєкторії. Узагальнені дані допомагають вдосконалити дизайн курсів, виділити популярні або складні теми та спрямовувати стратегічні рішення на інституційному рівні.

Потенціал для монетизації SaaS

Враховуючи його модульну природу та потужний набір функцій, прототип може перетворитися на повноцінний SaaS-продукт. Цей перехід дозволить іншим освітнім платформам або установам використовувати функції, керовані штучним інтелектом, на основі підписки, зменшуючи бар'єри для входу та надаючи передові технології штучного інтелекту ширшій аудиторії.

Обмеження

Залежність від зовнішніх сервісів штучного інтелекту

Такі компоненти, як генерація запитань на основі OpenAI, створюють залежність від сторонніх API та інфраструктури. Ця залежність може спричинити витрати на використання, потенційні простої або проблеми з дотриманням нормативних вимог, особливо при масштабуванні системи в глобальному масштабі або в регіонах із суворими вимогами до управління даними. Для безперебійного обслуговування або більш суворого контролю конфіденційності можуть знадобитися автономні або локальні рішення зі штучним інтелектом.

Обмежені навчальні дані для адаптивних моделей

Хоча платформа вже може адаптувати контент на основі профілів учнів, глибина персоналізації залежить від обсягу та якості даних. Невеликі заклади або нещодавно запущені курси можуть не генерувати достатню кількість логів взаємодії для повноцінного навчання просунутих моделей рекомендацій. Це обмеження підкреслює необхідність постійного збору та обробки даних для вдосконалення можливостей штучного інтелекту.

Набір функцій на ранніх стадіях

Незважаючи на реалізацію основних функцій, таких як створення запитань, адаптивне тестування та аналітика, система залишається прототипом. Основні функції масштабу підприємства, такі як багатомовна підтримка, офлайн-доступ, надійні інформаційні панелі звітності та глибока інтеграція з системами управління навчанням (LMS), можуть бути неповними або розробленими лише частково.

Потреба в ресурсах

Складні обчислення ШІ - особливо ті, що забезпечують зворотний зв'язок у режимі реального часу - вимагають масштабованих обчислювальних ресурсів і надійної інфраструктури. Невеликі організації або організації з обмеженим IT-бюджетом

можуть зіткнутися з труднощами, пов'язаними з підтримкою такого середовища, що підкреслює потребу в керованому хостингу або хмарних сервісах.

3.5 Майбутні напрямки розвитку

Хоча прототип вже продемонстрував значні можливості та очевидний потенціал як освітній інструмент зі штучним інтелектом, його поточний стан слугує лише основою для подальших розробок. Подальші розробки можуть розширити сферу його застосування, покращити його адаптивність і створити монетизоване, готове до ринку програмне забезпечення як послугу (SaaS). Нижче наведено основні напрямки, в яких може розвиватися платформа:

Перехід до повної моделі SaaS

Багатокористувацька архітектура: Адаптація прототипу для підтримки декількох клієнтських установ в рамках єдиної інфраструктури. Кожен клієнт матиме ізольоване сховище даних, що забезпечить безпеку та відповідність вимогам, а також спільний доступ до основних послуг на основі штучного інтелекту.

Плани підписки та ліцензування: Розробити рівні цін на основі наборів функцій (наприклад, базова аналітика проти розширених рекомендацій ШІ). Така структура дозволить освітнім провайдерам легко збільшувати або зменшувати масштаби відповідно до їхніх бюджетів і потреб.

Спрощений онбордінг: Впровадження автоматизованих робочих процесів реєстрації та забезпечення, що дозволить новим школам або платформам реєструватися онлайн, налаштовувати початкові курси та інтегрувати платформу з мінімальними технічними накладними витратами.

Розширена персоналізація за допомогою штучного інтелекту

Безперервне навчання та перепідготовка моделей: У міру накопичення більшої кількості взаємодій з користувачами та даних оцінювання, перенавчання моделей ШІ на цьому зростаючому наборі даних може підвищити точність прогнозування та персоналізацію. Система може поступово адаптуватися до інституційного контексту, предметних нюансів і регіональних навчальних уподобань.

Адаптивна рекомендація контенту: Розширте механізм рекомендацій, щоб врахувати більш складні критерії - наприклад, стилі навчання студентів, поведінку в Інтернеті в минулому або успішність одногрупників - щоб точно визначити, які ресурси або вправи оптимізують засвоєння знань.

Інтелектуальні цикли зворотного зв'язку: Надавайте контекстно-залежні пропозиції в режимі реального часу не лише учням, а й викладачам. Наприклад, попередження викладачів про концепції, які часто неправильно розуміються, або рекомендації щодо коригування змісту в середині курсу, якщо клас відчуває труднощі.

Інтеграція зі сторонніми інструментами та екосистемами LMS

Розробка плагінів: Створення плагінів або модулів, які вбудовують функції штучного інтелекту в широко використовувані системи управління навчанням (LMS), такі як Moodle, Canvas і Blackboard. Цей підхід знижує бар'єр для впровадження для установ, які вже інвестували в існуюче програмне забезпечення LMS.

Екосистема API: Публікація комплексних RESTful API та документації для розробників для розширення чи кастомізації платформи. Партнери можуть створювати спеціалізовані інформаційні панелі або додаткові додатки, розширюючи екосистему системи, зберігаючи при цьому основні функції, керовані штучним інтелектом.

Покращена аналітика та візуалізація даних

Інтерактивні дашборди: Розробка розширених інтерфейсів звітності для адміністраторів, викладачів та учнів, що візуалізують тенденції продуктивності,

показники залученості та відсоток завершення навчання. Ці дашборди можуть допомогти у прийнятті рішень на основі даних та вдосконаленні політики.
Прогностичне втручання: Розширте аналітичний механізм, щоб передбачати учнів з групи ризику, надсилаючи автоматизовані повідомлення раннього попередження або персоналізовані рекомендації щодо навчання. Ця функція має на меті покращити показники утримання та успішності на масштабних онлайн-курсах.

Спеціальна експертиза домену та графіки знань
Бази знань для конкретних доменів: Для спеціалізованих галузей (наприклад, медична освіта, інженерія) створіть спеціальні онтології або графи знань. Моделі ШІ, навчені на цих наборах даних для конкретної галузі, можуть надавати точніші рекомендації щодо контенту та генерувати запитання.
Динамічне створення контенту: Покращення генерації запитань і контенту на основі ШІ шляхом інтеграції релевантних домену корпусів. З часом система може створювати все більш точні та контекстно-багаті матеріали, ще більше зменшуючи навантаження на викладачів, які мають створювати контент вручну.

Масштабування та операційна надійність
Розгортання в хмарі: Використання Kubernetes або інших платформ оркестрування контейнерів для автоматичного масштабування та розгортання без простоїв, що гарантує, що система зможе впоратися з піковими навантаженнями, такими як екзаменаційні тижні або зарахування на новий семестр.
Офлайн/локальні варіанти ШІ: Для клієнтів із суворими вимогами щодо конфіденційності даних або обмеженим доступом до інтернету розгортайте моделі висновків ШІ локально. Такий підхід може вирішити проблеми з дотриманням нормативних вимог і пом'якшити перебої в роботі сервісів, спричинені зовнішніми залежностями від API.

Покращена конфіденційність даних та етика

Тонкий контроль доступу: Впровадження дозволів на рівні функцій, інструментів анонімізації даних і шифрування в режимі спокою для захисту конфіденційних даних учнів.

Етичні настанови щодо штучного інтелекту: Постійно вдосконалювати політику управління для обробки даних, зменшення упередженості та прозорості. У міру зростання системи ці рекомендації гарантують, що персоналізація залишається справедливою і поважає права учнів.

Ці майбутні напрямки підкреслюють здатність платформи перетворитися з експериментального прототипу на багатофункціональну хмарну SaaS-пропозицію. Вдосконалюючи свої моделі штучного інтелекту за допомогою реальних даних, розширюючи можливості рекомендацій та аналітики, а також безперешкодно інтегруючись у ширшу екосистему освітніх технологій, платформа може перетворитися на складну послугу. Такий розвиток не лише задовольнить зростаючий попит на інтелектуальні рішення для електронного навчання, але й створить основу для довгострокових інновацій, персоналізації та глобальної доступності онлайн-освіти.

ВИСНОВКИ

Розробка прототипу освітньої платформи зі штучним інтелектом знаменує собою значущу конвергенцію передових технологій, продуманого навчального дизайну та перспективної архітектури. На кожному етапі - від концептуалізації системних вимог до реалізації модульної інфраструктури, орієнтованої на мікросервіси, - ми демонстрували, що онлайн-освіта стоїть на порозі нової ери: ери персоналізації, розуміння в реальному часі та динамічної адаптивності. Цей перехід виходить за рамки традиційної моделі «завантаж та оціни», яка довгий час домінувала в цифровій освіті, натомість зосереджуючись на створенні індивідуальних навчальних маршрутів та глибоких аналітичних можливостях.

Що ми досягли

Прототип адаптивного навчання

В основі цього рішення лежить функціональний прототип, який надає пріоритет адаптивному навчанню. Автоматизуючи повторювані завдання, такі як створення тестів і оцінювання, та використовуючи моделі штучного інтелекту для персоналізованих рекомендацій, платформа дозволяє зменшити адміністративний тягар і підвищити рівень індивідуальної залученості учнів. Це безпосередньо стосується поширених пасток у традиційних цифрових класах, де статичні ресурси та знеособлені оцінки часто не можуть утримати увагу різних груп учнів.

Інтеграція штучного інтелекту в ключові процеси

Наша робота підкреслює стратегічну інтеграцію штучного інтелекту в різні точки взаємодії системи. Від генерації контекстно-залежних запитань за допомогою сервісів на основі НЛП до аналізу продуктивності в режимі реального часу як для учнів, так і для викладачів, результатом є значне покращення користувацького досвіду. Цей рівень штучного інтелекту допомагає проактивно виявляти потенційні перешкоди в навчанні, гарантуючи, що студенти отримають своєчасну та релевантну

підтримку. Викладачі, тим часом, отримують цінну інформацію про прогрес і вузькі місця в класі, що дозволяє їм більш ефективно коригувати навчальну програму.

Розширюваність і масштабованість

Використовуючи архітектуру мікросервісів (за підтримки Unicorn Application Framework) та гнучку модель зберігання даних MongoDB, ми забезпечили готовність платформи до майбутнього розширення. Додавання нових функцій, таких як спеціалізовані навчальні модулі на основі штучного інтелекту, бібліотеки оцінювання для конкретних галузей або навіть інтеграція з новими освітніми стандартами, стає більш плавним процесом. Увага до масштабованості не обмежується технічною сферою, вона поширюється і на стратегічні моделі розгортання. Незалежно від того, чи йдеться про запуск екземплярів для невеликих академій, керованих волонтерами, чи великих університетів, інфраструктуру можна легко налаштувати відповідно до змінних вимог до навантаження.

Перевірка основних освітніх робочих процесів

Попереднє моделювання та функціональне тестування показали, що система ефективно справляється з основними функціями, такими як управління контентом, автоматизований зворотний зв'язок та поглиблене відстеження прогресу. Цей крок відкриває шлях до подальшого вдосконалення та проведення реальних пілотних програм для збору ширшого зворотного зв'язку з різних предметів, вікових груп та освітніх контекстів.

Конвеєри даних та аналітичний фонд

Центральним елементом дизайну платформи є прихильність до ретельної реєстрації даних. Взаємодія кожного користувача - від часу завершення уроку до виконання тесту - реєструється у структурованому, придатному для аналізу форматі. Цей конвеєр даних, хоч і на ранній стадії, відкриває вікно в майбутнє, наприклад, для

предиктивного моделювання. У довгостроковій перспективі алгоритми розпізнавання образів можуть призвести до появи таких функцій, як попередження про ранній відсів, міжпредметні знання або вдосконалені системи рекомендацій для корекції чи поглибленого навчання.

Майбутнє онлайн-освіти

Якщо вийти за межі цього прототипу, то траєкторія розвитку онлайн-освіти готова до трансформаційних змін, оскільки технології штучного інтелекту та можливості хмарних обчислень продовжують розвиватися. Наступні ключові тенденції та можливості ілюструють майбутні перспективи:

Ширша доступність і глобальне охоплення

Системи на основі штучного інтелекту можуть поширюватися по всьому світу як платформи «програмне забезпечення як послуга» (SaaS), заповнюючи прогалини для навчальних закладів, які не мають необхідної інфраструктури або досвіду. Така демократизація засобів навчання може стати суттєвим фактором у вирівнюванні глобального освітнього поля, пропонуючи рівний доступ до передових педагогічних методів незалежно від місцезнаходження чи соціально-економічних обмежень.

Персоналізація та втручання, збагачені даними

Оскільки ці системи збирають більше даних про взаємодію з користувачами, моделі машинного навчання, що лежать в їх основі, ставатимуть все більш адаптивними. Прогностична аналітика на основі великих даних може передбачати тенденції успішності студентів, ще більш детально підлаштовуючи їхні навчальні траєкторії. Платформа може автоматично групувати учнів у сегменти на основі стилю навчання або історичної успішності, сприяючи справді індивідуалізованому навчанню, яке виходить далеко за межі традиційних методів.

Інтеграція в цілісну екосистему

Наступна хвиля розвитку, ймовірно, буде пов'язана з тісною інтеграцією з популярними системами управління навчанням (LMS), мобільними додатками та постачальниками освітнього контенту. Такий підхід сприяє створенню безперервного навчального середовища, в якому рекомендації на основі штучного інтелекту будуть всюдисущими: від мобільних нагадувань про навчання до складних інформаційних панелей, які об'єднують формальну навчальну роботу із зовнішніми відкритими освітніми ресурсами. Оскільки освіта стає все більш взаємопов'язаною, ці навчальні екосистеми переосмислять те, як навчальні заклади курують і надають контент, керують обліковими даними і відстежують прогрес учнів.

Етичні та регуляторні міркування

Поряд з обіцянками персоналізації та широкомасштабної аналітики, етичні питання, пов'язані з конфіденційністю, безпекою даних і потенційними упередженнями в моделях ШІ, стають все більш виразними. Такі закони, як GDPR (у Європі) або FERPA (у США), встановлюють суворі правила щодо зберігання та обробки даних про студентів. Забезпечення постійного узгодження передових освітніх систем з такою політикою має велике значення для збереження довіри користувачів і довіри до сектору в цілому. Системи управління ШІ, покликані зменшити алгоритмічну упередженість і забезпечити прозорість, стануть фундаментом для дотримання та прийняття етичних норм.

Безперервні інновації та розвиток галузі

Освіта, яка традиційно повільно впроваджує нові технології, все частіше визнається сферою з високим потенціалом для інновацій у галузі ШІ. Оскільки бізнес та академічні дослідники співпрацюють над новими методами ШІ (наприклад, федеративним навчанням або розширеним розумінням природної мови), синергія між алгоритмами, що розвиваються, та реальними освітніми потребами створить

безпрецедентні можливості. У наступному десятилітті, ймовірно, з'являться гіперперсоналізовані цифрові репетитори, навчальні модулі з ефектом занурення VR/AR, а також специфічні для конкретних галузей рішення ШІ, що співіснуватимуть в рамках універсальних архітектур.

Заключні думки

Шлях від концептуальної основи до функціонуючого прототипу висвітлив не лише технічну можливість застосування штучного інтелекту в онлайн-освіті, а й його ширші наслідки для моделей викладання та навчання. Інтегрувавши надійну внутрішню архітектуру з передовою аналітикою та продуманим дизайном, орієнтованим на користувача, ми заклали основу для інструменту, який може стати каталізатором значного переходу до справді адаптивного електронного навчання.

Але це лише початок. Зі збільшенням обсягів даних і вдосконаленням алгоритмів платформа перетвориться на основу пропозиції SaaS, яка дасть викладачам можливість надавати висококваліфіковані знання, а студентам - по-справжньому розвиватися завдяки індивідуальним, динамічним навчальним траєкторіям. У поєднанні з ретельним дотриманням вимог і постійним діалогом про справедливе використання даних, ці інтелектуальні системи обіцяють переосмислити охоплення та ефективність цифрової освіти для учнів по всьому світу. Попереду нас чекають безперервні ітерації, масштабування та інновації, але вплив на майбутнє онлайн-освіти буде суттєвим і довгостроковим.

ПЕРЕЛІК ПОСИЛАНЬ

1. **Sullins, J. P.** The Ethics of Artificial Intelligence in Education: An Integrative Review // *AI & Society*. – 2021. – Vol. 36(1). https://www.researchgate.net/publication/367420600_Artificial_Intelligence_in_Education_and_Ethics
2. **Russell, S., & Norvig, P.** *Artificial Intelligence: A Modern Approach*. – 4th ed. – New York: Pearson, 2020. https://aima.cs.berkeley.edu/
3. **Anderson, T., & Dron, J.** Three Generations of Distance Education Pedagogy // *International Review of Research in Open and Distributed Learning*. – 2011. – Vol. 12(3). https://www.irrodl.org/index.php/irrodl/article/view/890
4. **Mayer, R. E.** *Multimedia Learning*. – 3rd ed. – Cambridge: Cambridge University Press, 2021. https://www.jsu.edu/online/faculty/MULTIMEDIA%20LEARNING%20by%20Richard%20E.%20Mayer.pdf
5. **Ramírez-Montoya MS, Loaiza-Aguirre MI, Zúñiga-Ojeda A, Portuguez-Castro M.** Characterization of the teaching prole within the framework of education 4.0. Future Internet. 2021; https://doi.org/10.3390/13040091
6. **García-Peñalvo, F. J., Llorens, F., Vidal, J.** *The new reality of education in the face of advances in generative artificial intelligence* // RIED Revista Iberoamericana de Educación a Distancia**. –** 2024**. –** 27(1). https://www.researchgate.net/publication/372788949_The_new_reality_of_education_in_the_face_of_advances_in_generative_artificial_intelligence
7. **Rivera, E., Garden, C.** *Gamification for student engagement: a framework* // Journal of Further and Higher Education**. –** 2021**. –** 45(4)**.** https://www.researchgate.net/publication/348954951_Gamification_for_student_engagement_a_framework
8. **Information Resources Management Association (IRMA)** *Intelligent Systems: Concepts, Methodologies, Tools, and Applications* // June 2018.

https://www.researchgate.net/publication/330481555_Intelligent_Systems_Concepts_Methodologies_Tools_and_Applications

9. **Awad, M., Khanna, R.** *Deep Neural Networks* // Efficient Learning Machines**. –** April 2015. https://www.researchgate.net/publication/300719034_Deep_Neural_Networks
10. **Шаповалов, Б. Д., Якименко, О. В.** Етичні аспекти використання штучного інтелекту в освіті // *ІІ Всеукраїнська науково-технічна конференція "Технологічні горизонти: дослідження та застосування інформаційних технологій для технологічного прогресу України і світу"*, 18 листопада 2024 року. – Київ: Державний університет інформаційно-комунікаційних технологій, 2024. https://duikt.edu.ua/uploads/p_2661_83654085.pdf
11. **Якименко, О. В., Шаповалов, Б. Д.** Застосування штучного інтелекту в розробці ігор: вплив на розвиток індустрії та перспективи // *ІІ Всеукраїнська науково-технічна конференція "Технологічні горизонти: дослідження та застосування інформаційних технологій для технологічного прогресу України і світу"*, 18 листопада 2024 року. – Київ: Державний університет інформаційно-комунікаційних технологій, 2024. https://duikt.edu.ua/uploads/p_2661_83654085.pdf

Кваліфікаційна робота
спеціальність 126 «Інформаційні системи та технології»
освітньо-професійна програма «Інформаційні системи та технології»

СИСТЕМА ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ОНЛАЙН-ОСВІТИ ЗА ДОПОМОГОЮ ПЕРСОНАЛЬНИХ РЕКОМЕНДАЦІЙ

ВИКОНАВ: ШАПОВАЛОВ БОГДАН ДМИТРОВИЧ, ІСДМ-61
НАУЧНИЙ КЕРІВНИК: САГАЙДАК ВІКТОР АНАТОЛІЙОВИЧ, ДОКТОР ФІЛОСОФІЇ

МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- Мета роботи: Обґрунтувати концепцію та розробити експериментальний прототип системи з елементами штучного інтелекту, спрямованої на підвищення ефективності освітніх онлайн-платформ.
- Об’єкт дослідження: Сучасні цифрові освітні середовища (платформи для онлайн-навчання), що інтегрують різні навчальні матеріали, інструменти оцінювання та засоби комунікації.
- Предмет дослідження: Методи й алгоритми штучного інтелекту (рекомендаційні системи, машинне навчання, аналітика даних) та архітектурні рішення для побудови адаптивної освітньої платформи з інтелектуальною обробкою великих обсягів даних.
- Методи дослідження:
  - Огляд і порівняння існуючих освітніх платформ та AI-технологій
  - Моделювання та розробка експериментального прототипу
  - Функціональне й автоматизоване тестування на основі згенерованих сценаріїв
  - Аналіз отриманих результатів і формування рекомендацій для подальших поліпшень

ТЕОРЕТИЧНИЙ АНАЛІЗ ТА ВИЗНАЧЕННЯ ВИМОГ

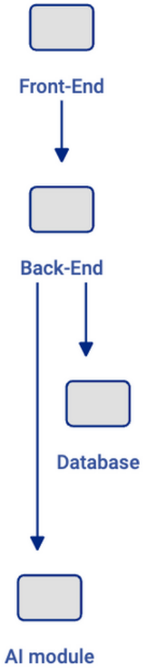
- Аналіз освітніх платформ, технологічних тенденцій
  - Дослідження функціоналу популярних ресурсів (Coursera, Udemy тощо)
  - Оцінка рівня персоналізації, інтерфейсів та способів взаємодії з користувачами
- Інтеграція ШІ в освітні програми
  - Використання алгоритмів рекомендацій для формування індивідуальних завдань
- Системні вимоги та цільова аудиторія
  - Забезпечення надійності, масштабованості та безпеки даних
  - Орієнтація на студентів, які прагнуть персоналізованого навчального контенту
- Вибір технологій та інструментів
  - MERN-стек для створення веб-застосунку (MongoDB, Express, React, Node.js)
  - Вбудовані аналітичні та AI-сервіси для обробки навчальних даних
  - Підхід, що дозволяє швидко розгортати й оновлювати систему

ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ПРОГРАМИ

- Архітектура програми, бази даних
  - Використання MERN-стеку (MongoDB, Express, React, Node.js) для створення інтерактивного та масштабованого застосунку
  - Структурна модель з розподілом ролей «клієнт-сервер» і зберігання даних у NoSQL-базі (MongoDB)
  - Забезпечення гнучкої схеми для зберігання інформації про користувачів і навчальні матеріали
- RESTful API та асистент зі штучним інтелектом
  - Проєктування методів API для обміну даними між фронт-ендом та бек-ендом
  - Застосування інструментів OpenAI/ChatGPT (або інших AI-сервісів) як допоміжного модуля
  - Можливість швидкого розширення функціоналу та впровадження нових рекомендаційних алгоритмів
- Структура проєкту та ключові компоненти
  - Компоненти інтерфейсу користувача (React) і модулі візуалізації освітнього контенту
  - Серверна частина на Express/Node.js для реалізації бізнес-логіки та інтеграції зі сторонніми сервісами

СТРУКТУРНА СХЕМА ТА ОСНОВНІ МОДУЛІ СИСТЕМИ

- Інтерфейс користувача (React):
  - Забезпечує візуалізацію навчальних матеріалів і рекомендацій
  - Опрацьовує запити та відправляє їх на сервер
- Серверна частина (Express / Node.js):
  - Обробляє RESTful-запити, керує бізнес-логікою
  - Взаємодіє з базою даних і модулем ШІ
- База даних (MongoDB):
  - Зберігає відомості про користувачів, їхній прогрес і навчальні матеріали
  - Підтримує гнучку схему для різноманітних структур даних
- Модуль штучного інтелекту:
  - Формує персональні рекомендації
  - Аналізує дані про активність користувача



ТЕСТУВАННЯ ТА ОЦІНКА РЕЗУЛЬТАТІВ

- Підходи та інструменти тестування
  - Функціональне тестування з використанням згенерованих сценаріїв
  - Перевірка працездатності ключових маршрутів RESTful API і базових алгоритмів рекомендацій
  - Автоматизоване тестування основних модулів для мінімізації ризику регресії
- Досягнення поставлених цілей, аналіз ефективності
  - Оцінка відповідності прототипу визначеним вимогам і функціональним критеріям
  - Аналіз даних логування для розуміння коректності обробки користувацьких запитів
  - Попередня валідація механізмів рекомендацій на основі тестових даних
- Переваги, обмеження та майбутні напрямки розвитку
  - Масштабованість і гнучкість архітектури, можливість простої інтеграції з іншими платформами
  - Необхідність залучення реальних користувачів для отримання релевантної статистики
  - Потенціал удосконалення алгоритмів ШІ та розширення можливостей аналітики

БЛОК-СХЕМА ВЗАЄМОДІЇ КОРИСТУВАЧА ІЗ СИСТЕМОЮ

1. Користувач заходить на платформу та проходить авторизацію/реєстрацію
2. Система збирає інформацію про попередні сесії та рівень підготовки (за наявності)
3. За запитом формуються попередні рекомендації
4. Користувач обирає курс або завдання для проходження
5. Після виконання завдання система оновлює прогрес у базі даних
6. Модуль ШІ аналізує результати та підказує нові матеріали

ПРИКЛАДИ СЦЕНАРІЇВ ВИКОРИСТАННЯ

- Новий користувач:
  - Реєстрація та аналіз первинних інтересів
  - Формування стартової програми з урахуванням рівня підготовки
  - Поступове надання рекомендацій та адаптація системи
- Поглиблене навчання:
  - Користувач бажає розширити знання у певній галузі
  - Система пропонує складніші матеріали та практичні завдання
  - Рекомендації оновлюються залежно від результатів проміжних тестів
- Викладач або адміністратор:
  - Перегляд статистики й прогресу студентів
  - Налаштування параметрів рекомендаційних алгоритмів
  - Оптимізація освітньої траєкторії для різних груп користувачів

ВИСНОВКИ

- Підсумок дослідження
  - Розроблено концепцію та прототип системи з елементами штучного інтелекту
  - Здійснено аналіз сучасних освітніх платформ і визначено методи інтеграції ШІ
- Загальні результати
  - Реалізовано базову архітектуру (MERN), механізм рекомендацій і RESTful API
  - Проведено функціональне тестування зі згенерованими сценаріями
  - Отримано попередні підтвердження життєздатності концепції персоналізованого навчання

ПЕРСПЕКТИВИ РОЗВИТКУ СИСТЕМИ

- Залучення реальних користувачів і збір зворотного зв’язку
  - Проведення пілотних впроваджень у навчальних закладах
  - Аналіз коментарів та відгуків для вдосконалення алгоритмів
- Удосконалення алгоритмів ШІ
  - Поглиблені методи машинного навчання для більш точних рекомендацій
  - Використання метрик успішності, врахування індивідуальних темпів навчання
- Масштабування й інтеграція з іншими платформами
  - Створення універсального API для обміну даними з LMS-системами
  - Розширення користувацької бази, розраховане на різні рівні освіти
- Розширення функціоналу та контенту
  - Додавання нових форматів навчальних матеріалів і гейміфікації
  - Модулі для оцінювання навичок і створення адаптивних тестів

АПРОБАЦІЯ

Основні ідеї та здобуті результати було представлено на II Всеукраїнській науково-технічній конференції «Технологічні горизонти: дослідження та застосування інформаційних технологій для технологічного прогресу України і світу» (18 листопада 2024 року, м. Київ). Зокрема, у збірнику матеріалів конференції опубліковано:

- Шаповалов Б. Д., Якименко О. В. «Етичні аспекти використання штучного інтелекту в освіті»
- Якименко О. В., Шаповалов Б. Д. «Застосування штучного інтелекту в розробці ігор: вплив на розвиток індустрії та перспективи»

ДЯКУЮ ЗА УВАГУ