МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВОЛИНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ЛЕСІ УКРАЇНКИ

Кафедра загальної математики та методики навчання інформатики

На правах рукопису

МАЛАЩУК ВЛАДИСЛАВ АНДРІЙОВИЧ

ЗАСТОСУВАННЯ ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ ОРГАНІЗАЦІЇ ІНДИВІДУАЛІЗОВАНОГО НАВЧАННЯ УЧНІВ У ШКІЛЬНОМУ КУРСІ ІНФОРМАТИКИ

Спеціальність: 014 «Середня освіта (Інформатика)»
Освітньо-професійна програма «Середня освіта. Інформатика»
Робота на здобуття освітнього ступеня «Магістр»

Науковий керівник:
ХОМЯК МАРІЯ ЯРОСЛАВІВНА,
кандидат фізико-математичних наук,
доцент кафедри загальної математики
та методики навчання інформатики

РЕКОМЕНДОВАНО ДО ЗАХИСТУ
Протокол № ____
засідання кафедри загальної математики
та методи навчання інформатики
від _____ 2025 року
Завідувач кафедри:
_____ _____
(підпис) ПІБ

ЛУЦЬК – 2025

Анотація

Малащук В.А. Застосування штучного інтелекту для організації індивідуалізованого навчання учнів у шкільному курсі інформатики – Рукопис.

Випускна кваліфікаційна робота за спеціальністю 014 Середня освіта (Інформатика). – Волинський національний університет імені Лесі Українки. Луцьк, 2025 р.

У магістерській роботі досліджено можливості застосування штучного інтелекту для підтримки індивідуалізованого навчання учнів у шкільному курсі інформатики.

На основі аналізу наявних підходів та практичних потреб учителя інформатики сформовано вимоги до програмного засобу та реалізовано вебзастосунок MentorAI, орієнтований на підтримку навчального сценарію: «предмет, тема, короткий урок, тестування».

Практична частина роботи включає проєктування та розробку клієнтської багатосторінкової вебплатформи з сучасним мінімалістичним інтерфейсом, реалізованої на базі React та TypeScript із застосуванням Redux для керування станом, маршрутизації й контролю доступу за ролями «учень» і «вчитель».

Ключовою особливістю MentorAI є механізм генерації тестових завдань на основі ШІ з резервним використанням локального банку питань у разі недоступності зовнішнього провайдера. Для вчителя передбачено перегляд узагальненої аналітики успішності та керування доступом учнів до предметів. Апробацію результатів здійснено під час педагогічної та переддипломної практик на базі опорного закладу загальної середньої освіти «Камінь-Каширський ліцей №1 ім. Євгена Шабліовського» Волинської області.

Ключові слова: штучний інтелект, індивідуалізоване навчання, адаптивне тестування, генерація завдань, шкільний курс інформатики, вебзастосунок, MentorAI.

Abstract

Malashchuk V.A. Application of Artificial Intelligence to Organize Individualized Learning of Students in the School Informatics Course – Manuscript.

Graduate qualification thesis in the specialty 014 Secondary Education (Informatics). – Lesya Ukrainka Volyn National University. Lutsk, 2025.

This master's thesis explores the possibilities of applying artificial intelligence to support individualized learning of students in the school informatics course.

Based on an analysis of existing approaches and the practical needs of an informatics teacher, the requirements for a software tool were formulated and the MentorAI web application was developed, aimed at supporting the learning scenario: "subject, topic, short lesson, assessment."

The practical part of the work includes the design and development of a client-side multi-page web platform with a modern minimalist interface, implemented using React and TypeScript, with Redux applied for state management, routing, and role-based access control for the "student" and "teacher" roles.

A key feature of MentorAI is an AI-based mechanism for generating assessment tasks, with a fallback to a local question bank in case an external provider is unavailable. For teachers, the system provides access to aggregated performance analytics and tools for managing students' access to subjects. The results were piloted during teaching and pre-graduation practice at the supporting general secondary education institution "Kamin-Kashyrskyi Lyceum No. 1 named after Yevhen Shabliovskyi" in the Volyn region.

Keywords: artificial intelligence, individualized learning, adaptive assessment, task generation, school informatics course, web application, MentorAI.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному освітньому середовищі значна увага приділяється забезпеченню індивідуального підходу до кожного учня. Шкільний курс інформатики вимагає не лише надання базових знань із програмування, алгоритмів та комп'ютерних систем, а й адаптації цих знань під індивідуальні потреби й темп засвоєння учням. Зважаючи на різний рівень підготовки, мотивації та стилю навчання, традиційні методи викладання часто не забезпечують бажаного рівня залученості та ефективності.

Штучний інтелект (ШІ) відкриває нові можливості для адаптації освітнього процесу: він здатний аналізувати дані про успішність учня, його поведінку при виконанні завдань та моделювати оптимальні траєкторії навчання. Системи, що базуються на алгоритмах машинного навчання, можуть автоматично підбирати навчальні матеріали, пропонувати вправи диференційного рівня, оцінювати відповіді та коригувати подальшу стратегію викладання.

Попри наявність окремих комерційних рішень (наприклад, DreamBox Learning, Smart Sparrow, Squirrel AI Education), адаптація їх під шкільний курс інформатики в умовах української школи обмежена, а чимало сервісів не відповідають вимогам Державного стандарту базової та повної загальної середньої освіти або не підтримують специфічні локалізовані навчальні програми.

Тому розробка власної системи з елементами ШІ, орієнтованої саме на індивідуалізацію навчання у шкільному курсі інформатики, є нагальною освітньою потребою.

Об'єкт дослідження – процес організації індивідуалізованого навчання у шкільному курсі інформатики з використанням технологій штучного інтелекту. Предмет дослідження – методологічні та технологічні підходи до проєктування програмно-апаратного комплексу ШІ-системи, що забезпечує адаптивну траєкторію навчання, персоналізований добір завдань та зворотний зв'язок.

Мета дослідження – розробити теоретико-методичні засади та програмний комплекс штучного інтелекту для організації індивідуалізованого навчання учнів у шкільному курсі інформатики, забезпечити реалізацію, протестувати систему та оцінити її ефективність.

Завдання дослідження:

1. Проаналізувати поняття, принципи та компоненти індивідуалізованого навчання в контексті шкільного курсу інформатики.

2. Дослідити роль і можливості штучного інтелекту в адаптивних освітніх системах та механізми автоматизованої оцінки.

3. Класифікувати сучасні ШІ-інструменти для шкільної інформатики (супервізивні й ненадзглядові моделі, NLP-модулі, генеративні алгоритми, комп'ютерне зір).

4. Визначити педагогічні й психологічні аспекти індивідуалізованого навчання з використанням ШІ (компетентнісний підхід, мотивація, інклюзивність).

5. Оглянути методи персоналізації контенту на основі машинного навчання (динамічні траєкторії, колаборативна фільтрація, мультимодальні рекомендації).

6. Провести аналіз існуючих закордонних і вітчизняних шкільних ШІ-платформ з акцентом на адаптивні механізми, локалізацію, відповідність ДСТУ та доступність.

7. Дослідити принципи управління навчальним контентом із використанням ШІ (автоматична класифікація, генерація й оновлення матеріалів, контроль якості, інклюзивні формати).

8. Обґрунтувати необхідність розробки вітчизняної ШІ-системи для інформатики з урахуванням мовної локалізації, відповідності державним стандартам, інклюзивності, інтеграції з «Е-щоденником» і захисту даних.

Наукова новизна дослідження полягає у:

- розробці комплексного підходу до проєктування та реалізації ШІ-системи, що суголосно поєднує алгоритми машинного навчання з навчальною програмою з інформатики для україномовних шкіл;
- створенні алгоритмів динамічної адаптації навчальних траєкторій, що враховують психолого-педагогічні особливості учнів;
- апробації комплексного рішення в реальному шкільному середовищі та комплексному оцінюванні його педагогічної ефективності.

Практичне значення полягає у можливості використання розробленої системи у навчальних закладах для підвищення мотивації та якості засвоєння навчального матеріалу з інформатики, оптимізації роботи вчителів завдяки автоматизації аналізу успішності та рекомендацій щодо корекції освітнього процесу.

Методи дослідження. У роботі застосовано комплекс методів:

- теоретичний аналіз і синтез наукових джерел щодо індивідуалізованого навчання та ШІ;
- експертне опитування вчителів інформатики для визначення педагогічних вимог;
- методи системного аналізу та проєктування (UML-моделі, ER-діаграми);
- алгоритмічні методи машинного навчання (дерева рішень, кластеризація, методи колективного навчання);
- експериментальне дослідження (пілотне впровадження та порівняльний аналіз статистики);
- методи математичної статистики для оцінювання результатів.

Апробація результатів дослідження відбулася на базі опорного закладу загальної середньої освіти «Камінь-Каширський ліцей №1 ім. Євгена Шабліовського» Камінь-Каширської міської ради Волинської області під час педагогічної та переддипломної практик. Методика була впроваджена у навчальний процес учнів 10–11 класів.

Результати дослідження були представлені на Всеукраїнській науково-практичній конференції студентів, аспірантів та молодих вчених «Сучасна освіта і наука Волині» (тези); XIV Міжнародній науково-практичній конференції «Математика. Інформаційні технології. Освіта» (тези).

1. Хомяк А., Стельмащук Р., Малащук В., Хомяк М. Цифрові технології в освіті: сучасні виклики та перспективи. Математика. Інформаційні технології. Освіта: збірник тез доп. XIV міжнар. наук.-практ. конф. (м. Луцьк, 13-15 червн. 2025 р.). Луцьк, 2025. С. 262-265.

2. Малащук В., Хомяк М. Застосування штучного інтелекту для організації індивідуалізованого навчання учнів у шкільному курсі інформатики. Математика. Інформаційні технології. Освіта: збірник тез доп. XIV міжнар. наук.-практ. конф. (м. Луцьк, 13-15 червн. 2025 р.). Луцьк, 2025. С. 213-215.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ ЗАСТОСУВАННЯ ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ ІНДИВІДУАЛІЗАЦІЇ НАВЧАННЯ

1.1. Індивідуалізоване навчання: поняття, принципи, компоненти

Індивідуалізоване навчання виступає ключовим підходом у сучасній освіті, спрямованим на врахування індивідуальних особливостей кожного учня – пізнавальних, мотиваційних, емоційних та соціально-психологічних. У контексті шкільного курсу інформатики індивідуалізація означає не лише диференціацію змісту та форм роботи, а й можливість кожному учню вчитися у власному темпі, розвивати власні навчальні стратегії та використовувати засоби, що найбільш відповідні саме його стилю пізнання.

По першому читанню, індивідуалізоване навчання можна визначити як сукупність педагогічних умов, які забезпечують адаптацію освітнього процесу до унікальних потреб кожного учня. Така адаптація охоплює зміст матеріалів, методи викладання, темп опанування, зворотний зв’язок та дидактичні засоби, що мають відповідати особистим можливостям і мотивації учня. У межах шкільного курсу інформатики індивідуалізація реалізується через можливість вибору різних рівнів складності завдань, варіативних технологій подачі (наприклад, текстові інструкції, інструктивні відео, інтерактивні симуляції) і зворотного зв’язку у режимі реального часу.

Одним із фундаментальних понять, яким оперує методологія педагогіки, є “індивідуальна освітня траєкторія”. Учень, який навчається індивідуалізовано, отримує змогу проходити курс залежно від власного темпу, початкових знань і навичок, що дозволяє уникнути як переобтяження, так і відчуття бракованої мотивації через надмірну простоту чи складність матеріалу. Тому індивідуалізація прагне забезпечити баланс між викликом і доступністю, надаючи учневі можливість самостійно відстежувати власні успіхи та коригувати стратегію навчання за підтримки вчителя та інформаційних систем.

Принципи індивідуалізованого навчання спираються на загальнонаукові й педагогічні позиції, серед яких можна виокремити кілька ключових. Перший принцип – врахування індивідуальних особливостей учня, що передбачає попередню діагностику рівня знань, стилю навчання, мотиваційного профілю та емоційно-психологічного стану. На практиці це означає, що для кожного учня вчитель або система автоматичного оцінювання спочатку встановлює базовий рівень, за яким формує початковий набір завдань. Другий принцип – адаптивність освітнього середовища, коли зміст, методи й форми роботи змінюються залежно від успішності учня: якщо він швидко опановує матеріал, отримує завдання підвищеної складності, а якщо стикається з труднощами, – додаткові пояснення й вправи. Третій принцип – підтримка мотивації та саморегуляції, коли учень залучається до постановки власних навчальних цілей, рефлексії над результатами та планування подальшої роботи. Четвертий принцип – забезпечення безперервного зворотного зв’язку, що дає змогу своєчасно коригувати траєкторію: учень отримує оперативні коментарі, рекомендації, результати автоматичних тестів і завдань. П’ятий принцип – співпраця учителя та технологій, коли педагог виступає не єдиним джерелом знань, а наставником, який координує роботу автоматизованих систем, аналізує отримані дані та коригує процес навчання, звертаючи особливу увагу на ті аспекти, які машина поки що не здатна обробити: емоційні прояви, мотиваційні коливання, особливості взаємодії в групі.

За структурою компоненти індивідуалізованого навчання можна представити у вигляді чотирьох взаємопов’язаних елементів. Перший – діагностичний компонент, який включає інструменти створення профілю учня: тести, опитування, завдання попереднього рівня, методи психологічного обстеження. На його основі формується персональна модель учня, що відображає його пізнавальні ресурси, стиль навчання, слабкі й сильні сторони. Другий – навчальний компонент, що складається з диференційованого контенту: вправ різного рівня складності, мультимедійних матеріалів, симуляційних вправ

і теоретичних блоків, які систематизовані за складністю та напрямками. Третій – механізм корекції, що передбачає формування зворотного зв'язку, обробку результатів роботи учня, аналіз помилок, рекомендації для повторного опрацювання теми. Частиною цього механізму є адаптивні алгоритми, які можуть змінювати послідовність і складність завдань у реальному часі. Четвертий – методичний супровід, що реалізується через роль учителя як експерта: саме педагог формулює додаткові пояснення, організовує індивідуальні консультації, формує групові та індивідуальні завдання, проводить рефлексію разом із учнем і коригує загальний план.

У практиці шкільного курсу інформатики індивідуалізація має свої особливості, пов'язані з природою навчального матеріалу й методологією викладання. По-перше, інформатика традиційно базується на поєднанні теоретичних понять (алгоритми, структури даних, моделі обчислювальних процесів) та практичної діяльності (мови програмування, середовища розробки, практичні завдання з написання коду). Щоб кожен учень міг працювати у власному темпі, диференціація здійснюється за кількома напрямками: складність алгоритму, кількість рядків коду, обсяг теоретичного матеріалу, тип інструменту (текстовий редактор, графічне середовище, інтерактивний тренажер). По-друге, важливим компонентом індивідуалізації в інформатиці є розвиток алгоритмічного та критичного мислення, яке включає уміння аналізувати проблему, формалізувати її у вигляді алгоритму, оцінювати ефективність різних підходів і прогнозувати результати. Учень, що навчається індивідуалізовано, отримує можливість виконувати спочатку спрощені вправи, які формують базу, а потім – більш складні завдання, у яких поєднуються різні алгоритмічні структури (наприклад, вкладені цикли, рекурсія). Таке поступове ускладнення створює сприятливі умови для поступового нарощування компетенцій і водночас формує відчуття власного успіху.

Інший аспект, на якому акцентує індивідуалізація, – цифрова грамотність та уміння самоорганізовуватися. Оскільки сучасні цифрові платформи

мамонтують масив функцій – від інтерактивних лекцій до систем підрахунку результатів – учень повинен вміти самостійно планувати свій час, користуватися різними типами навчальних ресурсів (відео, текст, інтерактив), а також аналізувати власні прогалини. У цьому контексті роль індивідуалізованого навчання стає одночасно і навчальною, і виховною: учень набуває навичок ефективного планування, контролю результатів, самооцінювання й коригування власної діяльності.

Підсумовуючи зміст поняття, можна сказати, що індивідуалізоване навчання – це не суто дидактична операція диференціювання завдань. Це комплексний педагогічний підхід, що поєднує попередню діагностику, створення персональної освітньої траєкторії, безперервний зворотний зв’язок та методичний супровід учителя. Його метою є забезпечення максимально ефективного та комфортного освітнього середовища для кожного учня, що вчиться у шкільному курсі інформатики.

Одним із завдань індивідуалізації є розширення можливостей кожного учня в самостійному пошуку інформації й рішення нестандартних питань. Наприклад, якщо під час вивчення теми «Масиви» учень вільно справляється зі створенням одновимірних масивів і маніпуляціями з ними, йому пропонують перейти до вправ із двовимірними масивами та алгоритмами обробки матриць. Водночас учню, який ще не опанував базову операцію ініціалізації одновимірного масиву, надають додаткові візуалізаційні матеріали та інтерактивні тренажери, які допомагають вибудувати алгоритмічне мислення «знизу вгору».

У цілому, застосування індивідуалізації в шкільній інформатиці сприяє формуванню ключових компетентностей: цифрової грамотності, алгоритмічного мислення, здатності до співпраці і саморегуляції. Завдяки цьому учні не лише опановують окремі теми курсу, а й набувають умінь планувати власний навчальний процес, аналізувати інформацію, аргументувати вибір способу

розв’язання задачі та працювати в команді, використовуючи цифрові інструменти.

У наступних розділах буде продемонстровано, як методи індивідуалізації доповнюються інструментами штучного інтелекту – від модулів автоматичного оцінювання до генерації персоналізованих навчальних маршрутів, що разом формують ефективну модель адаптивного навчання у шкільному курсі інформатики.

1.2 Роль ШІ в адаптивному навчанні

Адаптивне навчання базується на принципі, коли освітня система підлаштовується під індивідуальні потреби учня, змінюючи зміст, темп і форму подачі матеріалу залежно від поточної успішності та особливостей конкретного учня. Штучний інтелект у цьому процесі виступає рушійною силою, оскільки надає можливість автоматизованого аналізу великих обсягів даних про поведінку учнів і приймати рішення щодо коригування траєкторії навчання в режимі реального часу.

По-перше, ключова роль ШІ в адаптивному навчанні полягає в діагностиці рівня і стилю навчання учня на підставі різних параметрів: швидкості відповіді на тестові запитання, кількості помилок у практичних вправах, частоти звернення до підказок і навіть часу перебування в кожному навчальному модулі. Алгоритми машинного навчання можуть автоматично аналізувати цей масив даних і формувати модель учня, що враховує його сильні та слабкі сторони. Така модель слугує відправною точкою для подальших коригувань: якщо система виявляє, що учень швидко опановує прості вправи, вона пропонує завдання підвищеного рівня складності, а коли фіксується затримка або збільшена кількість помилок, їй доступні додаткові роз’яснення або спрощені матеріали для повторного опрацювання.

По-друге, ШІ-системи виконують роль безперервного монітора змін навчального процесу. Замість того щоб чекати на чергову контрольну роботу або оцінку від учителя, адаптивна платформа відстежує успіхи учня постійно: аналізує виконання вправ, вираховує середні показники часу на завдання, виявляє типові помилки і тенденції у динаміці навчання. Завдяки цьому учитель отримує попередження про учнів, які опинилися в зоні ризику, та своєчасно організовує додаткові консультації. На практиці це зменшує ймовірність того, що учень наздожене відставання надто пізно, коли компенсувати прогалини буде складніше.

По-третє, складовою адаптивного навчання є модуль підбору контенту, який на основі рекомендаційних алгоритмів пропонує учневі саме ті матеріали, які відповідають його поточному рівню та інтересам. Наприклад, під час вивчення теми «Масиви» система може запропонувати комусь вправи на маніпуляцію одновимірними масивами, тоді як іншому учневі, котрий опанував цю тему раніше, вона автоматично підготує завдання з динамічними структурами даних або двовимірними масивами (рис. 1.1). Таке динамічне підбирання контенту підвищує ефективність засвоєння навчального матеріалу, бо учень працює з тим, що йому наразі потрібно, а не з набором шаблонних вправ.

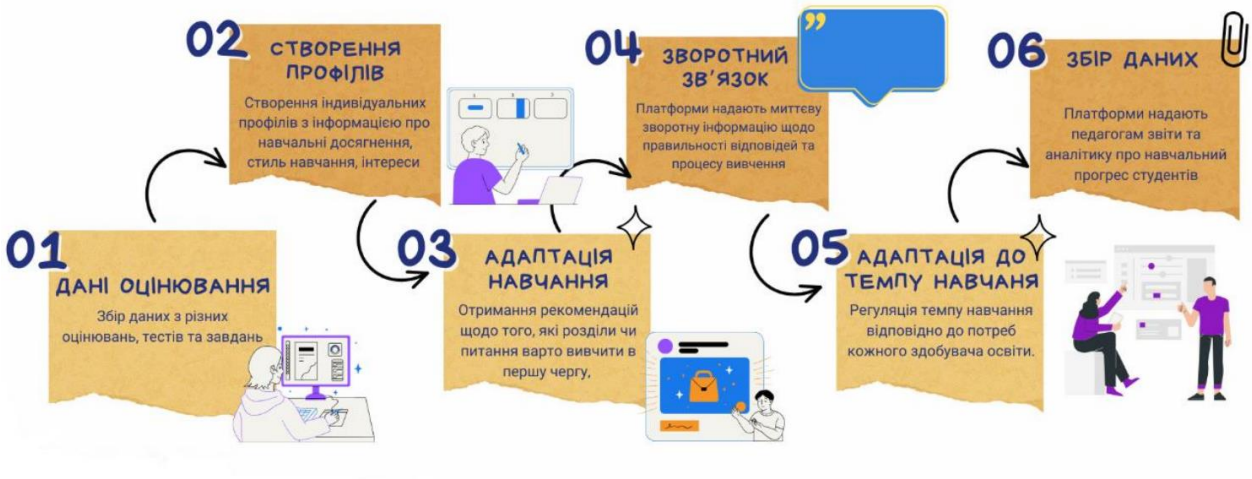


Рис. 1.1. Приклад функціонування адаптивної системи на основі ШІ

По-четверте, застосування ШІ в адаптивному навчанні спрямоване також на автоматизоване оцінювання. Інтегровані системи автогрейдингу дозволяють перевіряти практичні завдання учнів – наприклад, програмний код – без участі вчителя. Алгоритми не лише оцінюють правильність синтаксису, а й аналізують логіку виконання програми, порівнюючи результати з очікуваними. Якщо учень припустився помилки, система миттєво надає коментар чи підказку з поясненням, яка частина коду є хибною і як її виправити. Це робить навчальний процес більш інтерактивним: учень отримує «живий» зворотний зв'язок, може самостійно виправляти помилки й одразу перевіряти виправлений варіант. Таке прискорене оцінювання сприяє глибшому розумінню алгоритмічних конструкцій, запобігає накопиченню помилок у мисленні та підвищує мотивацію до самостійної роботи.

По-п'яте, ШІ забезпечує розробку інтелектуальних тьюторів – віртуальних помічників, які взаємодіють із учнями через чат-інтерфейс або голосовий модуль. Застосовуючи моделі обробки природної мови, такі тьютори здатні аналізувати запити учнів, відповідати на запитання щодо концептів інформатики, пояснювати помилки в коді або пропонувати приклади рішень. Наприклад, якщо учень запитує: «Чому мій цикл не завершується?», модель розпізнає помилку в логіці умови виходу й видає конкретну пораду: перевірити параметр умови чи змінну, що експлуатується в лічильнику. Інтелектуальний тьютор працює 24/7, тому навіть поза уроками учень має змогу оперативно отримати допомогу.

По-шосте, додатковою роллю ШІ у адаптивному навчанні є генерація контенту. Сучасні великі мовні моделі можуть створювати варіанти тестових запитань, готувати приклади коду в різних мовах програмування, формулювати пояснювальні тексти та опис задач на основі теми уроку. Це дозволяє швидко формувати нові банківські запитання та вправи, які враховують актуальні програмні середовища або специфіку навчальної програми. Учитель натомість лише коригує або додає локальні приклади, але більший обсяг підготовчої

роботи виконує штучний інтелект, що значно економить час на планування уроків.

Врешті, важливо зазначити, що безпосередня роль ШІ в адаптивному навчанні виходить за межі виключно технічної автоматизації. Адже для створення ефективної системи необхідне поєднання технологічних рішень зі зваженим педагогічним дизайном. Інженери й методисти спільно визначають, які алгоритми машинного навчання найкраще відповідають освітнім цілям, а вчитель виступає фасилітатором, що аналізує результати й коригує систему. Таким чином, ШІ стає невід'ємним елементом модернізації освітнього процесу, але успіх адаптивного навчання залежить від взаємодії технологій і педагогічних практик.

1.3 Класифікація ШІ-інструментів для шкільної інформатики

Штучний інтелект стає дедалі важливішим компонентом освітніх платформ, особливо у навчанні інформатики, де велика увага приділяється алгоритмізації, програмуванню та аналітиці даних. Інструменти ШІ можна класифікувати за функціональною ознакою, типом алгоритмів та особливостями застосування в освітньому процесі. Нижче наведено основні категорії ШІ-інструментів, які найчастіше використовуються у шкільному курсі інформатики.

Інструменти супервізивного навчання у шкільній інформатиці відзначаються здатністю навчатись на вже розмічених даних. Зазвичай використовують дерева рішень, Random Forest, XGBoost та нейронні мережі. У навчальному контексті ці алгоритми навчаються на показниках учнів: кількість правильних відповідей на попередніх тестах, час, витрачений на вправи, кількість звернень до підказок. Після навчання модель може прогнозувати ймовірність того, що конкретний учень успішно розв'яже завдання з певної теми, наприклад «масиви» чи «цикли». Це дозволяє вчасно виявити учнів, які

потребують додаткової підтримки, перш ніж помилки накопичаться й призведуть до серйозного відставання.

Коли обсяг розмічених даних обмежений, або потрібно знайти приховані зв'язки без явних міток, доцільно застосовувати алгоритми ненадзглядового навчання. Метод кластеризації, зокрема K-means чи DBSCAN, дає змогу згрупувати учнів із подібними поведінковими профілями в інтерактивному тренажері. Наприклад, можна виділити групу учнів, які систематично витрачають занадто багато часу на вправи з циклів, а також тих, хто швидко переходить до вправ підвищеного рівня складності. Така класифікація допомагає адаптувати методику викладання, оскільки вчитель або система автоматично генерує різні набори вправ для різних кластерів.

У разі потреби аналізу багатовимірних даних про учнів і виявлення аномалій найчастіше використовують методи зниження розмірності, наприклад PCA (метод головних компонент). Застосовуючи його до вхідних даних, що складаються з результатів тестів, часу на завдання, кількості помилок і частоти звернень до підказок, можна отримати візуалізацію в двовимірному просторі. Учитель одразу помічає учнів «вінґлів» – тих, які суттєво відрізняються від решти класу – і може оперативно реагувати: проводити індивідуальну консультацію чи призначати додаткові матеріали.

Звичайно, підкріплене навчання застосовують рідше безпосередньо в шкільній інформатиці, але воно має значний потенціал при створенні гейміфікованих середовищ. Використання агента підкріпленого навчання у симуляторі допомагає динамічно змінювати рівень складності ігрових вправ відповідно до успіхів учня. Наприклад, коли агент фіксує, що учень успішно розв'язує поточні завдання, наступний раунд може містити більш складні алгоритмічні проблеми. Це підвищує мотивацію, оскільки учень бачить, що гра дійсно «підтримує» його прогрес і виклики зростають разом із його навичками.

Наступними є NLP-модулі, які дають змогу аналізувати відкриті відповіді та допомагають створювати діалогових помічників. Моделі на кшталт BERT чи

GPT-серії обробляють текстові коментарі учня в коді, порівнюють їх із базою еталонних пояснень і автоматично оцінюють коректність термінології. Наприклад, якщо учень пояснює алгоритм сортування, – система може перевірити, чи правильно використовуються терміни «компаратор», «обмін», «побудова піраміди» тощо, і вказати на неточності.

Коли створюють інтелектуального тьютора, зазвичай використовують ті самі NLP-модулі зі спеціалізацією на освітній діяльності. Такий тьютор може працювати як чат-бот, який відповідає на запитання учня в режимі реального часу. Наприклад, учень може запитати: «Чому мій цикл не завершується?» – і тьютор миттєво аналізує умову виходу в коді, виявляє логічну помилку і пропонує пояснення: «Перевірте значення лічильника – воно ніколи не стає меншим за n».

Паралельно з NLP активно використовують генеративні моделі для створення навчального контенту. Сучасні системи, зокрема ті, що базуються на GPT-4, можуть генерувати тести з багатоваріантними запитаннями, формулювати кілька варіантів практичних завдань і навіть складати «код-челленджі» для самостійної практики. Наприклад, учитель може задати системі: «Згенеруй п’ять прикладів завдань із множинним вибором на тему «масиви в Python» для учнів 10 класу», а система має повернути готовий перелік запитань із поясненнями до правильних відповідей. Завдяки цій можливості педагоги економлять значну кількість часу, проте, як правило, додають свої локалізовані приклади, щоб контент відповідав українській програмі.

Не слід ігнорувати роль рекомендаційних систем, які в освітньому середовищі діють аналогічно до тих, що використовуються в онлайн-магазинах. Використовуючи матрицю «учень–завдання» з оцінками успішності, алгоритм колаборативної фільтрації і методи факторизації матриць визначають, які вправи будуть найбільш релевантними для конкретного учня. Якщо учень А і учень Б мають подібну історію взаємодії з вправами, але А успішно справився із певними завданнями, система пропонує ці самі вправи учню Б. Таким чином формується

персоналізований набір вправ, орієнтований на максимальний розвиток компетенцій.

Ще однією категорією інструментів є сервіси комп'ютерного зору. Вони дозволяють розпізнавати рукописні алгоритми в блок-діаграмах або ж аналізувати фотографії завдань, що учні виконали на папері. Алгоритми OCR (Optical Character Recognition) сканують та розпізнають текст чи графічні елементи, після чого система конвертує рукописний код у цифровий формат для подальшого автоматизованого оцінювання. Завдяки цьому під час контрольних робіт учні мають можливість писати алгоритми рукою, а система швидко перетворює їх у код і перевіряє на коректність.

Усі зазначені інструменти об'єднуються в єдину адаптивну систему, де кожен клас алгоритмів виконує свою роль (рис. 1.2). Супервізивні моделі постійно навчаються на нових даних учнів, ненадглядові алгоритми допомагають виявляти приховані закономірності, NLP-модулі надають текстову підтримку та формують інтерактивні діалоги, генеративні моделі розширюють банк вправ, рекомендаційні системи спрямовують учня саме на ті завдання, які в даний момент є найбільш корисними, а сервіси комп'ютерного зору знижують бар'єр між традиційним форматом «папір–ручка» і цифровим середовищем.

Рис. 1.2. Популярні ШІ-інструменти, які можуть використовуватись для вивчення шкільної інформатики

Комбіноване використання цих інструментів створює ефективне адаптивне середовище. Наприклад, під час уроку на тему «цикл for» система

може спочатку запустити короткий діагностичний тест (супервізивна модель), потім запропонувати персоналізовані вправи (рекомендаційна система), обробити та оцінити результати (автогрейдер), а далі, за необхідності, інтелектуальний тьютор дає детальні пояснення. У кінці уроку система формує звіт для вчителя, в якому показано динаміку успіхів, час вирішення завдань, типові помилки та рекомендації щодо подальших кроків. Така інтеграція дає змогу не лише підвищити якість засвоєння матеріалу, але й розвантажити вчителя від рутинних операцій, залишаючи педагогічно важливі завдання – мотиваційний супровід, аналіз творчих здобутків учнів і педагогічні консультації.

З огляду на викладене, класифікація ШІ-інструментів для шкільної інформатики вимагає розуміння їхніх функціональних ролей та особливостей застосування. Лише інтегрований підхід, що поєднує різні алгоритми та сервіси, забезпечує справжню адаптивність навчання та створює передумови для глибокого розкриття потенціалу кожного учня.

1.4 Педагогічні та психологічні аспекти індивідуалізованого навчання

Педагогічні та психологічні аспекти індивідуалізованого навчання спрямовані на створення таких умов, за яких кожен учень отримує можливість розвиватися у власному темпі та відповідно до своїх індивідуальних особливостей. Це дозволяє враховувати не лише рівень підготовки й темп засвоєння матеріалу, а й мотиви, інтереси та емоційний стан дитини. Упровадження ШІ-інструментів у цей контекст відкриває нові можливості для персоналізації процесу: системи можуть аналізувати результати учнів у реальному часі, адаптувати завдання та надавати своєчасний коригувальний зворотний зв’язок. Крім того, педагогічна підтримка формується у взаємодії вчителя й технологій, що дозволяє оптимізувати навчальне середовище й забезпечити психологічний комфорт учням різного рівня та соціально-

психологічних потреб. Таким чином, поєднання педагогічних підходів із потенціалом ШІ сприяє формуванню цілісного підходу до індивідуалізації навчання, де враховуються як освітні, так і психоемоційні аспекти.

1.4.1 Компетентнісний підхід до впровадження ШІ

Компетентнісний підхід передбачає цілеспрямоване формування в учнів не лише знань і навичок, а й умінь критичного мислення, аналізу та самостійного пошуку інформації. Залучення ШІ-інструментів у навчальний процес сприяє розвитку таких компетенцій, адже технології пропонують адаптивні завдання залежно від рівня підготовки та індивідуальних потреб учня. Наприклад, системи адаптивного навчання аналізують відповіді кожного учня, виявляють прогалини та автоматично пропонують матеріали для самостійного опрацювання. Це створює умови для цілеспрямованого розвитку ключових компетенцій: цифрової грамотності, інформаційної компетентності, комунікативних та соціальних навичок, а також умінь співпрацювати в інтерактивному середовищі.

Щоб компетентнісний підхід був ефективним, необхідно правильно організувати інтеграцію ШІ-інструментів у навчальну стратегію. Педагог має визначити цільові компетенції, які формуються у межах конкретного уроку чи проєкту, та обрати оптимальні інструменти (наприклад, інтелектуальні системи тьюторства або віртуальних тренажерів). Важливо також забезпечити зворотний зв’язок: штучний інтелект може автоматично оцінювати прогрес учня, але вчитель має інтерпретувати ці дані й коригувати стратегію навчання. Таким чином, компетентнісний підхід із підтримкою ШІ сприяє формуванню в учнів умінь, необхідних для успішної взаємодії у сучасному цифровому суспільстві.

1.4.2 Мотиваційні чинники та самоорганізація учнів

Індивідуалізація навчання, підсилена ШІ-інструментами, відкриває нові можливості для підвищення мотивації учнів. Внутрішня мотивація формується тоді, коли учень бачить власний прогрес, отримує швидкий зворотний зв'язок і відчуває контроль над процесом навчання. Адаптивні платформи, що використовують алгоритми машинного навчання, надають персоналізовані завдання з урахуванням інтересів та рівня складності, що стимулює учня долати нові виклики.

Зовнішні мотиваційні чинники можуть бути реалізовані через елементи гейміфікації: рейтинги, бали, бейджі за досягнення. ШІ-асистенти можуть відстежувати успіхи та автоматично нагадувати про невиконані завдання або нові виклики, сприяючи формуванню навички самоорганізації. Самоорганізація учнів полягає в умінні планувати власний час, розставляти пріоритети й оцінювати свої слабкі й сильні сторони. Використання інструментів із можливістю відслідковування прогресу – електронних портфоліо, звітів системи – навчає учня самостійно встановлювати короткострокові та довгострокові цілі.

Ключова роль учителя полягає в підтримці та наставництві: він ознайомлює учнів із доступними ШІ-ресурсами, навчає ефективно використовувати їх для самонавчання й підтримує мотивацію шляхом індивідуальних консультацій. Важливо також враховувати, що різні учні реагують на визначені мотиваційні стимули по-різному. Тому завдання педагога – гнучко підходити до вибору інструментів та методів мотивації, моніторити динаміку інтересу й своєчасно коригувати підтримку.

1.4.3 Інклюзивність та відповідність потребам учнів із особливими освітніми потребами

Для учнів із особливими освітніми потребами (ООП) індивідуалізація є особливо важливою, оскільки традиційні методи навчання часто не враховують їхніх фізичних, сенсорних чи когнітивних особливостей. ШІ-інструменти можуть автоматично адаптувати формат презентації матеріалу – наприклад, перетворювати текст у аудіо, збільшувати шрифти, підбирати кольорові схеми для учнів із порушеннями зору. Для учнів із дислексією такі засоби можуть запропонувати спеціальні шрифти або фон, що полегшують читання.

Крім того, інтелектуальні системи можуть налаштовувати темп і складність вправ на основі швидкості обробки інформації кожного учня: це дає змогу уникнути перевантаження чи, навпаки, втрати інтересу через недостатній рівень стимулювання. Учні з порушеннями уваги можуть отримувати коротші блоки інформації чи перерви, які визначає система на основі показників втрати концентрації.

Соціально-психологічна підтримка також не менш важлива: віртуальні помічники або чат-боти можуть стати джерелом емоційної підтримки, надавати рекомендації щодо саморегуляції стресу або тривожності. ШІ-платформи часто містять модулі для відстеження емоційного стану учня через аналіз взаємодії (час відповіді, кількість помилок), що дозволяє педагогу своєчасно втрутитися.

Таким чином, завдяки інклюзивним ШІ-інструментам кожен учень отримує навчальне середовище, адаптоване до його індивідуальних фізичних, когнітивних та психологічних особливостей, що забезпечує рівний доступ до якісної освіти та сприяє формуванню позитивного ставлення до навчання.

1.5 Методи персоналізації контенту на основі машинного навчання

Методи персоналізації контенту на основі машинного навчання спираються на можливості алгоритмів аналізувати великі масиви даних про навчальну активність учнів і пропонувати кожному саме ті ресурси, які максимально відповідають його рівню, стилю навчання та інтересам. Насамперед виділяють підхід, коли створюється набір вправ із різними рівнями складності: від базового до поглибленого. Зазвичай учень спочатку проходить діагностичний тест, у якому відповідає на запитання або виконує прості практичні завдання з теми «цикл», «умовний оператор» чи «масиви». Алгоритм машинного навчання обробляє результати цього тесту, фіксує час виконання вправ, кількість помилок, кількість звернень до підказок і на основі цих даних робить висновок, чи готовий учень працювати на середньому рівні складності чи слід залишитися на базовому. Якщо учень демонструє високий рівень знань і швидко дає правильні відповіді, система пропонує йому завдання підвищеного рівня, у яких поєднуються декілька тем або вводиться додатковий контекст. Якщо ж аналіз показує, що учень затримується надто довго або часто помиляється у простих завданнях, система повертає його до початкових вправ із детальними поясненнями та мультимедійними ілюстраціями.

Ще один метод персоналізації пов’язаний із використанням моделей прогнозування успішності, зокрема таких алгоритмів, як Random Forest чи логістична регресія. У цих моделях на вході використовуються ознаки (features) учня – кількість спроб на попередніх вправах, середній час виконання коду, частота звернення до цифрових підказок, загальний бал у тематичних тестах, а також метрики поведінки в інтерактивних середовищах (наприклад, наскільки активно учень взаємодіє з відеоінструкціями чи графічними тренажерами). Навчившись на даних про велику кількість учнів, модель прогнозує ймовірність

того, що наступна вправа буде успішно виконана, або виявляє, що у конкретного учня є ризик відставання. Відповідно до таких прогнозів система може коригувати послідовність тем: наприклад, якщо модель вказує на високий ризик провалу в темі «функції», учню запропонують повторення попередніх тем у стислому форматі або додаткові вправи на закріплення базових понять. Завдяки цьому підхід мінімізується імовірність того, що учень раптово опиниться перед надскладними завданнями без необхідного базису.

Однією з ефективних методик є колаборативна фільтрація, у якій система аналізує матрицю «учень–вправа» з оцінками успішності та виділяє кластери учнів із подібними поведінковими профілями. Наприклад, якщо учень А і учень Б обоє показали схожу динаміку у темах «масиви» та «цикли», але учень А уже успішно виконав вправу на редагування файлів, система може рекомендувати цю вправу учню Б як наступну. Такий підхід враховує не лише поточний рівень знань, а й поведінку однолітків із аналогічною історією навчання. За рахунок цього персоналізовані рекомендації стають більш точними, оскільки система орієнтується на успішний досвід реальних учнів.

Додатково до суто статистичних методів, застосовують мультимодальні рекомендації, які враховують, як саме учень сприймає інформацію. Якщо під час вступного опитування і роботи з матеріалами в інтерактивній системі зафіксовано, що учень переважно вивчає матеріал через відео та демонстрації алгоритмів, доцільно продовжувати пропонувати йому відео-уроки з інтерактивними прикладами. Якщо ж учень краще оперує текстовими умовними системами, платформа наголошує на текстових прикладах коду та покрокових поясненнях. У разі потреби до мультимодального контенту додають інтерактивні тренажери, де учень «перетасовує» блоки коду або використовує графічний конструктор для ілюстрації алгоритмів. Цей підхід виходить за межі простих «якщо–то» рекомендацій на основі правил і забезпечує більш індивідуалізовану взаємодію.

Не менш важливим є метод динамічного формування адаптивних траєкторій навчання. Алгоритм аналізує дані про кожну спробу учня упродовж усього навчального циклу: він фіксує, як учень працює з окремими вправами, наскільки швидко вчиться новим поняттям, скільки часу витрачає на повторення. Використовуючи підхід reinforcement learning, система може «винагороджувати» учня за успішні спроби переходом до складніших завдань і вилучати занадто прості вправи з поточного набору. Таким чином формується унікальна траєкторія: якщо учень добре розуміє тему «масиви», траєкторія може перейти до «рекурсії» чи «об'єктно-орієнтованого програмування», але якщо виявлено слабке розуміння базової структури даних, система поверне до вправ на роботу з одновимірними масивами з різними типами даних.

Для більшості шкільних платформ важливо відзначити, що обчислювальні ресурси обмежені, і тому часто використовують легкі версії моделей, які можна виконати у веб-браузері або на локальному сервері школи. Наприклад, прості регресійні моделі чи дерева рішень можуть працювати навіть на ноутбуках із середніми технічними характеристиками. Це дає змогу ефективно впроваджувати персоналізацію в регіонах із нестабільним інтернет-з'єднанням або за обмежених ресурсів.

У результаті вжиття вищезазначених методів персоналізації контенту на основі машинного навчання навчальний процес стає більш ефективним та гнучким. Учні працюють у комфортному для себе режимі, отримують саме ті матеріали, які сприяють їхньому розвитку, а вчитель може зосередитися на аналітичному та методичному супроводі, зводячи до мінімуму рутинну перевірку та підготовку матеріалів. Така система дає змогу поєднувати індивідуальні потреби учнів із загальними освітніми стандартами, підвищуючи загальну якість викладання інформатики.

1.6. Аналіз існуючих шкільних ШІ-платформ

Аналіз сучасних шкільних платформ на базі штучного інтелекту дозволяє виявити ключові переваги та обмеження у реалізації адаптивного навчання. Насамперед варто зазначити, що глобальні рішення здебільшого розроблені з урахуванням універсальних підходів і можуть адаптуватися під різні навчальні програми, але часто не враховують локальні освітні стандарти чи мовні особливості. У той же час українські проєкти прагнуть забезпечити повну відповідність Державному стандарту, проте їм бракує ресурсів для розробки високотехнологічних алгоритмів і масового введення у школи. Порівняльний аналіз функціоналу, локалізації й доступності показує, що жодне рішення поки не є універсально оптимальним: глобальні платформи мають потужні адаптивні механізми, але вартість їхніх ліцензій та брак локалізованих матеріалів обмежують їх застосування, тоді як локальні платформи більш доступні мовно, але не завжди мають повний функціонал. У наступних підпунктах розглянемо особливості закордонних та українських рішень окремо, а в кінці проведемо порівняльну оцінку.

1.6.1 Закордонні рішення: особливості адаптивних механізмів

Сучасні закордонні платформи, такі як DreamBox Learning, Smart Sparrow та Khan Academy, відзначаються високим рівнем автоматизації й використанням алгоритмів машинного навчання для динамічного підбору матеріалів. DreamBox Learning, наприклад, використовує адаптивний двигун, який в реальному часі аналізує дії учня: час відповіді на кожне питання, кількість помилок і шляхи розв’язання. На основі цих даних система миттєво змінює складність наступного завдання, переходячи від простих вправ до вправ із розширеним контекстом.

Smart Sparrow пропонує конструктор «розумних підручників», що дає змогу педагогам створювати гнучкі сценарії з гілками адаптації: залежно від відповіді учня, платформа перенаправляє його до додаткових пояснень або нових завдань. У Khan Academy адаптивні механізми працюють за алгоритмом модульної перевірки кожної теми: якщо учень демонструє високий рівень знань, то система пропонує просунуті питання, а якщо є труднощі – повертає до базових пояснень.

У всіх цих рішеннях загальною рисою є використання механізму непрямого навчання: платформа не просто пропонує завдання, а аналізує поведінкові метрики – час на питання, кількість спроб, звернення до гіперпосилань і підказок. Педагог у таких системах виступає як координатор та наставник: він може налаштовувати базу запитань, додавати мультимедійні матеріали та корегувати рівні складності, але основна роль адаптації покладається на ШІ. Загалом закордонні рішення відрізняються високими затратами на підтримку інфраструктури, складними алгоритмами, які вимагають регулярного перенавчання моделей на великій кількості даних, а також ліцензійними обмеженнями, що робить їхнє впровадження в українських школах складнішим (рис. 1.3).

Рис. 1.3. Порівняльні адаптивні механізми закордонних платформ освіти.

1.6.2 Українські проєкти та їхня відповідність державним стандартам

Вітчизняні розробки, наприклад UkAIClass, ШІ-Учитель та EduMe, створені з урахуванням специфіки Державного стандарту з інформатики. Їхній контент розроблено українською мовою, він включає програмові теми 7–11 класів, і платформи орієнтовані на інтеграцію з локальними електронними щоденниками. UkAIClass, наприклад, забезпечує базову адаптацію: система пропонує учням комплекс питань за темою «масиви» з урахуванням їхніх результатів у попередніх тестах, однак механізм адаптації побудовано на правилах «якщо-те» (рівень успіху нижче 60% – повернутися до теоретичного блоку). ШІ-Учитель у вигляді чат-бота відповідає на запитання учнів і дає короткі пояснення термінів, але не аналізує великі масиви поведінкових даних. EduMe пропонує відеоуроки з елементами адаптації: Йякщо учень переглянув пояснення кілька разів і повернувся до нього, платформа рекомендує перегляд іншого формату матеріалу – інфографіки або інтерактивного тренажера.

Усі українські рішення мають чітку відповідність навчальній програмі, стандартизовані теми та список компетентностей, які повинні сформуватися у кінці курсу. Це дозволяє вчителям слідувати методичним рекомендаціям Міністерства освіти і уникати розбіжностей із базовими і профільними рівнями. Проте українські платформи не завжди можуть забезпечити глибоку адаптивність, оскільки бракує алгоритмів машинного навчання високої складності. Найчастіше вони використовують прості правила та фіксовані пороги для різних рівнів складності, а не багатовимірні моделі прогнозування. Незважаючи на це, такі платформи є доступними, адже їх розробляли з орієнтацією на бюджетні можливості шкіл, і вони не вимагають дорогих серверних рішень.

Однією з особливостей українських проєктів є прагнення забезпечити інклюзивність: платформи передбачають зміну кольорових схем для учнів із порушенням зору, озвучення тексти за допомогою TTS-движків для учнів із

дислексією, а також додаткові підказки для учнів із когнітивними особливостями. У багатьох випадках українські розробники співпрацюють із фахівцями з НаУОА чи педагогічних університетів, що дає змогу швидко оновлювати контент відповідно до змін стандарту. Незважаючи на обмежений функціонал адаптації, системи забезпечують повноцінний супровід усіх темніх блоків курсу, що компенсує відсутність «просунутих» алгоритмів машинного навчання.

1.6.3 Порівняльна оцінка функціоналу, локалізації та доступності

Порівнюючи глобальні і локальні рішення, можна виокремити кілька критеріїв: функціонал, локалізація, відповідність стандартам, доступність. За функціоналом закордонні платформи значно випереджають українські аналоги: вони мають розгалужені адаптивні механізми, здатні працювати з великими даними, використовують NLP для аналізу відкритих відповідей і генерують контент на основі сучасних моделей GPT. Водночас українські рішення здебільшого обмежуються rule-based механізмами та простими рекомендованими сценаріями. Тому, якщо школа прагне максимального технічного рівня, варто розглядати платформи на кшталт DreamBox чи Smart Sparrow.

Локалізація є значним плюсом українських розробок. Вони забезпечені матеріалами, які точно відповідають програмі 7–11 класів з інформатики, розробленою Міністерством освіти, та враховують особливості типового електронного щоденника. Закордонні рішення потребують додаткового перекладу інтерфейсу й адаптації контенту до ДСТУ, що іноді призводить до втрати частини навчальних блоків або дублювання контенту без урахування локального контексту. Через це українські платформи виграють у сегменті «повна відповідність стандартам».

Доступність включає питання вартості ліцензій, технічних вимог до обладнання та швидкості інтернету. Більшість закордонних платформ працює за підписною схемою з досить високою абонплатою, що непідйомно для шкіл із обмеженим бюджетом. Крім того, вони вимагають стабільного інтернету та потужних серверів. Українські платформи розроблено з урахуванням технічних обмежень: вони можуть працювати у локальному мережевому режимі, на недорогому обладнанні й навіть у разі повільного інтернету (за рахунок кешування контенту). Це робить їх доступними для більшості шкіл, особливо у віддалених регіонах.

У результаті порівняльної оцінки можна констатувати, що універсального рішення поки не існує. Якщо школа може інвестувати у потужну інфраструктуру та платити за підписку, закордонні платформи забезпечують найвищий рівень адаптивності й технологічності. Якщо ж пріоритетом є локалізація, відповідність стандартам і доступність, українські рішення є оптимальними, хоча й з обмеженим функціоналом. Вибір залежить від пріоритетів закладу: технологічне лідерство чи мовна та методична відповідність.

1.7. Управління навчальним контентом із використанням ШІ

Управління навчальним контентом із застосуванням штучного інтелекту передбачає не лише зберігання матеріалів, а й їхню автоматизовану класифікацію, адаптацію, генерацію нових ресурсів та постійний контроль якості. Спершу освітня платформа формує єдиний репозиторій, де всі навчальні одиниці мають метадані (тему, складність, формат). Потім алгоритми машинного навчання аналізують ці метадані: за допомогою токенізації та векторизації текстових матеріалів вони визначають ключові слова й групують схожі за змістом ресурси. У результаті кожен текст, відео чи інтерактивна вправа отримує теги, що спрощують пошук і подальшу персоналізацію.

Коли відбувається оновлення стандартів або змінюється навчальна програма, генеративні моделі на кшталт GPT можуть швидко створити нові пояснювальні блоки та тестові завдання. Наприклад, вчитель вказує заголовок теми та ключові пункти програмної вимоги – система генерує конспект із підзаголовками, прикладами коду й питаннями для самоперевірки. Якщо Державний стандарт доповнює розділ “Об’єктно-орієнтоване програмування”, платформа сама оновлює контент, додаючи відповідні вправи й тестові запитання. Це значно скорочує час на підготовку до уроків, адже вручну створювати аналогічну кількість матеріалів довго й трудомістко.

Персоналізація контенту реалізується через рекомендаційні механізми, що ґрунтуються на аналізі поведінкових даних учнів: успішні та помилкові спроби, час виконання вправ і взаємодія з мультимедійними елементами. Система може пропонувати додаткові вправи тим, хто має підтверджені прогалини, або одразу переключати на більш складні завдання учнів, які демонструють високий рівень опанування матеріалу. У разі, коли учень віддає перевагу відео, платформа автоматично відфільтровує текстові уроки та пропонує саме ті відеоматеріали, які краще відповідають його стилю навчання.

Аналітичні модулі ШІ регулярно збирають дані про взаємодію кожного учня з контентом. За допомогою цих даних формуються звіти, які вчитель отримує у вигляді графіків і таблиць: наприклад, скільки разів конкретне відео переглянули, який відсоток учнів успішно виконав вправи, скільки часу було витрачено на ту чи іншу тему. Завдяки цьому вчитель може коригувати навчальний план: якщо видно, що значна частина класу не засвоїла тему «масиви», доцільно додати додаткові приклади або організувати міні-лекцію. Аналітика також допомагає вчасно виявити учнів із ризиком відставання – платформа сигналізує, коли хтось тривалий час не покращує результати в конкретному блоці.

Інклюзивність є невід’ємною складовою управління контентом із ШІ. Автоматичне створення альтернативних форматів – текстових субтитрів для

відео або озвучення текстів – дозволяє учням із порушеннями зору чи слуху комфортно працювати з матеріалами. Вбудовані TTS (text-to-speech)-движки читають текст уроків учням із дислексією, а модуль розпізнавання голосу дає змогу тим, хто має труднощі з моторикою, взаємодіяти з платформою без клавіатури. Налаштування часових обмежень для виконання завдань гарантує, що кожен учень матиме достатньо часу, враховуючи індивідуальні потреби.

Окрім аналізу та генерації, ШІ виконує автоматичний контроль якості контенту. Навчальні модулі скануються на коректність метаданих, відповідність стандартам SCORM/xAPI й наявність «битих» посилань. Алгоритми перевіряють синтаксис кодових фрагментів, які вбудовані в пояснення, – платформа виконує тестовий запуск кожного фрагмента, щоб упевнитися в його працездатності. Помилки або застарілі елементи відразу передаються методисту або адміністратору LMS для виправлення.

Завдяки застосуванню ШІ та автоматизації управління контентом відпадає необхідність у постійному ручному оновленні матеріалів (рис. 1.4). Учитель може зосередитися на методичних питаннях, а платформа забезпечить постійну актуалізацію, персоналізацію й контроль якості.

Рис. 1.4. Архітектура ШІ для управління контентом.

1.8. Обґрунтування необхідності розробки вітчизняної ШІ-системи

Попри наявність численних закордонних адаптивних платформ, потреба у власній вітчизняній ШІ-системі обумовлена низкою специфічних обставин. Передусім, закордонні рішення часто не враховують мовно-культурні особливості української школи. Інтерфейси й контент можуть бути доступними лише англійською мовою, що створює бар'єр для багатьох учнів. Переклад інтерфейсу та матеріалів у більшості випадків здійснюється вручну, що забирає час і призводить до невідповідностей із чинними Державними стандартами. Крім того, закордонні платформи можуть не включати саме ті теми й компетентності, які передбачені українською програмою з інформатики; це загрожує тим, що учні отримуватимуть розширені чи обмежені версії курсу без урахування профільних вимог.

З іншого боку, використання глобальних рішень супроводжується значними фінансовими витратами на ліцензії, особливо якщо мова йде про цілу школу чи мережу закладів. Для більшості українських шкіл із обмеженим бюджетом підписка на закордонні платформи вища за реальні можливості. Брак фінансування призводить до того, що школи змушені шукати альтернативи або йти на компроміси, обмеживши доступ учнів до повного набору функцій.

Основні аргументи на користь розробки власного рішення:

- Відповідність Державному стандарту з інформатики. Українська програма має чітко визначені теми, компетентності й очікувані результати навчання. Власна ШІ-система може бути побудована так, щоб усі навчальні модулі й вправи розроблялися на основі чинних програмних документів, що гарантує повну відповідність державним вимогам.

- Мовна локалізація та культурний контекст. Важливо розробити контент українською мовою, із прикладами, актуальними для наших учнів (наприклад, обробка даних про річні врожаї, локалізація інтерфейсу відповідно до

українських шкільних термінів). Це створить більш комфортне середовище та підвищить мотивацію.

- Доступність і вартість. Вітчизняне рішення можна розповсюджувати за демократичними цінами або взагалі відкривати у форматі відкритого коду для шкіл із низьким бюджетом. Це забезпечить ширше покриття й дозволить усім категоріям учнів мати рівний доступ до інновацій.

- Інтеграція з державними інформаційними системами. Українські школи користуються «Електронним щоденником» і Moodle-сервісами, які вже мають свої дані про учнів, оцінювання й звітність. Власна ШІ-система може бути спроєктована таким чином, щоб безшовно обмінюватися даними з цими системами, що спрощує адміністрування й зводить до мінімуму дублювання інформації.

- Підтримка інклюзивності. Вітчизняне рішення може передбачати локальні особливості – наприклад, процедури роботи з учнями, які мають порушення зору чи слуху, зокрема озвучування уроків українською TTS-движком, субтитри до відеоматеріалів, адаптовані форми роботи для учнів з особливими

- освітніми потребами. Конфіденційність та захист даних. Використання закордонних сервісів може бути зумовлене ризиком витоку персональних даних учнів за кордон. Власна система гарантує зберігання даних у межах держави, що відповідає вимогам законодавства про захист інформації та забезпечує безпеку учнівської бази.

Додаткові фактори важливості.

По-перше, розробка власного рішення стимулює Україну до формування національної екосистеми EdTech-продуктів. Це створить робочі місця для IT-фахівців, педагогів-методистів та аналітиків, а також розвиватиме партнерство між університетами та школами у сфері інноваційних технологій. Більшість університетів уже мають центри прикладних досліджень, які можуть стати майданчиком для тестування нових алгоритмів та моделей.

По-друге, власна ШІ-система спрощує поширення ефективних практик та методик навчання. Вчителі зможуть обмінюватися шаблонами вправ, алгоритмами адаптації та сценаріями уроків у межах єдиної платформи, де ШІ аналізує практики найуспішніших педагогів і рекомендує їх колегам. Це підвищує загальну якість освіти та сприяє професійному розвитку вчителів.

По-третє, співпраця з державою у створенні вітчизняної ШІ-системи дозволяє залучити державні кошти та гранти на розвиток освітніх інновацій. Такі проєкти можуть отримати фінансування з держбюджету або Європейського Союзу, що зменшить фінансове навантаження для окремих шкіл і сприятиме довгостроковому розвитку платформи.

У підсумку, обґрунтування розробки вітчизняної ШІ-системи включає мовну локалізацію, відповідність стандарту, доступність, інклюзивність й інтеграцію з державними інформаційними системами. Лише власне комплексне рішення дозволить забезпечити якісне індивідуалізоване навчання інформатики в українських школах, підвищити рівень захисту даних і створити умови для сталого розвитку національної EdTech-екосистеми.

РОЗДІЛ 2. РОЗРОБКА ТА РЕАЛІЗАЦІЯ ОСВІТНЬОГО ВЕБЗАСТОСУНКУ ДЛЯ ІНДИВІДУАЛІЗОВАНОГО НАВЧАННЯ З ВИКОРИСТАННЯМ ШТУЧНОГО ІНТЕЛЕКТУ

2.1. Постановка задачі та формування вимог до освітнього вебзастосунку MentorAI

Основною метою розробки освітнього вебзастосунку MentorAI є створення цифрового середовища для індивідуалізованого навчання учнів закладів середньої освіти з опорою на інструменти штучного інтелекту. Актуальність такого підходу зумовлена потребою підвищення ефективності навчання шляхом адаптації навчальних завдань до рівня підготовки, темпу засвоєння матеріалу та типових помилок конкретного учня. У межах обраної теми магістерської роботи MentorAI розглядається як практичний приклад застосування ШІ для персоналізації освітньої траєкторії, де ключовим механізмом виступає автоматичне формування тестових завдань і їх подальше коригування залежно від результатів.

Під час реалізації було обрано підхід створення клієнтського вебзастосунку на базі React з використанням TypeScript та Redux для централізованого керування станом. З огляду на навчально-демонстраційний характер роботи дані зберігаються локально у браузері. Такий спосіб є тестовим варіантом, який дозволяє швидко перевірити проєктні рішення та логіку взаємодії користувачів без розгортання серверної інфраструктури. Водночас для реального впровадження у закладі освіти необхідним є перехід до повноцінного збереження даних у базі даних та реалізація серверної частини, що забезпечить керування користувачами, захист інформації та надійність зберігання результатів навчання.

Після аналізу призначення системи та очікуваних сценаріїв використання було сформовано ключові вимоги до MentorAI.

До основних функціональних вимог було віднесено такі:

– система має забезпечувати автентифікацію та авторизацію користувачів, аби розмежувати доступ до функцій і навчальних даних;

– передбачити розподіл ролей користувачів на «учень» і «вчитель», оскільки ці ролі виконують різні дії в системі;

– забезпечити відображення для учня навчального каталогу, який включає предмети та теми, а також доступ до навчальних матеріалів для обраної теми;

– реалізувати механізм тестування з автоматичною перевіркою відповідей та підрахунком результату;

– забезпечити генерацію тестових завдань засобами ШІ та підтримку резервного сценарію у випадку недоступності генерації, щоб система залишалася працездатною;

– реалізувати адаптивний принцип формування наступних тестів, коли система посилює ті концепти, у яких учень допускає помилки, і зменшує частку завдань з уже засвоєних елементів;

– забезпечити накопичення історії спроб у межах теми та можливість перегляду деталей конкретної спроби з відображенням правильних і обраних відповідей;

– надати вчителю інструменти перегляду узагальненої успішності учнів та елементів аналітики навчальних результатів, а також можливість підключати учнів до предметів;

– передбачити елементи мотивації у вигляді винагороди за успішне проходження тестів та системи досягнень, що сприяє підтримці навчальної активності.

На етапі формування вимог було визначено, що система має забезпечувати повний базовий цикл взаємодії в межах навчального процесу. Учень після входу отримує доступ до переліку предметів, обирає тему, опрацьовує короткий навчальний матеріал та переходить до тестування. За результатами проходження тесту учень отримує оцінку, а також може переглянути детальну розбірку

відповідей і власну історію спроб. Адаптаційний компонент передбачає, що наступні добірки завдань будуть орієнтовані на проблемні концепти, які визначаються за статистикою помилок.

Вчитель, у свою чергу, виконує функції організації та контролю. Він має доступ до інтерфейсів для перегляду узагальненої інформації щодо результатів учнів, а також може керувати підключенням учнів до предметів. Такий розподіл забезпечує модель, де учень концентрується на навчальній діяльності, а вчитель отримує інструменти для моніторингу та педагогічних рішень.

Серед нефункціональних вимог ключовими є простота та зрозумілість користувацького інтерфейсу, сучасний мінімалістичний дизайн і коректна робота на різних розмірах екранів. Окремо враховано вимоги до читабельності та візуальної гармонійності інтерфейсу, зокрема можливість підготовки скріншотів для друкованих матеріалів. Також важливими є стабільність роботи в межах браузера, передбачуваність навігації між сторінками та можливість подальшого розширення функціоналу. Зокрема, система проєктувалась так, щоб у перспективі можна було додати серверну частину, перейти до централізованої бази даних, підсилити механізми безпеки, а також розширити перелік предметів, тем і аналітичних інструментів.

Отже, завданням розробки MentorAI є створення доступного та функціонального вебзастосунку, що демонструє практичні можливості використання штучного інтелекту для індивідуалізованого навчання. Сформовані вимоги стали основою для подальшого проєктування структури застосунку, вибору технологій та реалізації ключових модулів системи у практичній частині магістерської роботи.

2.2. Вибір моделі та методології розробки програмного забезпечення

Сучасний процес розробки програмного забезпечення передбачає обґрунтований вибір моделі життєвого циклу та методології, що забезпечують

структурованість, керованість змін, стабільність і можливість подальшого розвитку програмного продукту. З огляду на специфіку MentorAI як освітнього вебзастосунку з елементами адаптивного навчання та інтеграцією ШІ, під час виконання практичної частини магістерської роботи було визначено підхід до організації розробки, який дозволяє швидко перевіряти гіпотези, уточнювати вимоги та поступово нарощувати функціональність без втрати працездатності системи.

Для реалізації MentorAI було обрано ітераційну модель розробки програмного забезпечення. Її сутність полягає в поетапному створенні функціональних модулів із регулярним тестуванням, аналізом результатів і внесенням коригувань у наступних ітераціях. Такий підхід є доцільним для вебзастосунків, де вимоги уточнюються в процесі розробки, а значна частина рішень залежить від зручності інтерфейсу та реальних сценаріїв використання. У MentorAI підтвердженням доцільності ітеративного підходу стала поступова побудова системи від базових компонентів до складніших механізмів індивідуалізації. Спочатку формувалась базова структура застосунку та маршрутизація, після чого послідовно додавалися модулі навчального каталогу, контент тем, тестування, збереження результатів, а також інструменти для ролі вчителя і механізми аналітики. На кожному етапі виконувалась перевірка працездатності, коректності навігації та взаємодії компонентів між собою, що дозволяло зменшити ризик накопичення критичних помилок наприкінці роботи.

Паралельно з ітераційною моделлю використовувалися принципи гнучкої методології розробки Agile, орієнтованої на швидке отримання робочого результату та адаптацію до змін. У межах цієї методології процес розробки умовно поділявся на короткі цикли, у яких реалізовувався конкретний набір функцій, що мав чіткі критерії завершення (рис. 2.1). Після завершення кожного циклу проводилась перевірка функціональності, усунення неточностей у логіці та удосконалення користувацького інтерфейсу. Такий підхід є особливо актуальним для систем, що пов’язані зі штучним інтелектом, оскільки якість

взаємодії користувача із системою, формат підказок, логіка адаптації та механізми генерації завдань потребують неодноразового уточнення і часто коригуються після практичного тестування.

Рис. 2.1. Принцип роботи ітераційної моделі розробки

Важливою особливістю процесу розробки MentorAI стало поєднання модульного підходу та прототипування. Модульність означала, що ключові підсистеми застосунку проєктувалися як відносно незалежні компоненти з чіткими межами відповідальності. Зокрема, окремо були виділені модулі авторизації та ролей користувачів, навчального каталогу, тестування та обліку результатів, а також підсистема генерації питань на основі ШІ з резервним сценарієм. Це підвищило керованість коду та спростило подальше розширення функціоналу. Прототипування застосовувалося там, де необхідно було швидко перевірити, чи буде зручною обрана логіка для учня або вчителя, наприклад під час формування сценарію проходження теми або перегляду історії спроб.

Окремої уваги у межах методології було приділено тестуванню та налагодженню після кожної ітерації. Перевірка включала оцінювання коректності маршрутизації, стабільності авторизації та розмежування доступу за ролями, правильності автоматичної перевірки тесту, коректності збереження спроб і відтворення їх перегляду в інтерфейсі. Таким чином забезпечувалася безперервна підтримка працездатного стану застосунку, що є ключовим принципом Agile.

Слід зазначити, що для демонстраційної реалізації MentorAI було використано локальне збереження даних у браузері як проміжне рішення, що спрощує розгортання і дає змогу зосередитися на логіці індивідуалізації. У методологічному плані це відповідало підходу мінімально достатнього прототипу. Водночас обрана ітераційно-гнучка організація розробки забезпечує можливість подальшого переходу до повноцінної клієнт–серверної архітектури з централізованою базою даних без необхідності повного перепроєктування всього застосунку. Такий розвиток є перспективним для практичного впровадження, оскільки дозволить забезпечити довготривале збереження навчальних результатів, керування користувачами та підвищений рівень безпеки.

Отже, вибір ітераційної моделі життєвого циклу та використання принципів Agile забезпечили ефективну організацію розробки MentorAI, можливість швидко реагувати на зміни, послідовно нарощувати функціональність і підтримувати стабільність вебзастосунку. Обраний підхід створив передумови для подальшого масштабування системи та її адаптації до вимог реального освітнього середовища.

2.3. Загальний опис проєкту та архітектури вебзастосунку

MentorAI реалізовано як клієнтський вебзастосунок, що працює безпосередньо у браузері та побудований на основі бібліотеки React з

використанням TypeScript. Такий підхід дозволяє створити швидкий, інтерактивний інтерфейс із чіткою компонентною структурою та підтримкою багатосторінкової навігації. На відміну від класичних клієнт–серверних рішень, поточна реалізація MentorAI не потребує розгортання окремого серверного бекенду. Це спрощує демонстрацію функціональних можливостей і дає змогу зосередитися на логіці індивідуалізації навчання та інтеграції ШІ. Водночас у межах практичного застосування в освітньому середовищі доцільним буде перехід до серверної архітектури з централізованою базою даних, що забезпечить довготривале збереження результатів, керування користувачами та підвищений рівень безпеки.

Загальна архітектура MentorAI базується на принципах модульності та розмежування відповідальності. Застосунок умовно поділено на кілька логічних рівнів, що взаємодіють між собою через визначені інтерфейси та функції доступу (рис. 2.2). Такий підхід наближає структуру проєкту до концепції шарової архітектури на клієнтській стороні, де окремо виділяються рівень представлення, рівень стану, модулі даних і сервісні модулі для зовнішніх інтеграцій.

Рівень представлення відповідає за інтерфейс користувача та взаємодію з ним. Він реалізований через систему сторінок і компонентів React та включає окремі макети для ролей учня і вчителя. У застосунку передбачено багатосторінкову навігацію, яка організовує послідовний користувацький сценарій навчання: вхід у систему, перехід до списку предметів, вибір теми, перегляд навчального матеріалу, проходження тесту, перегляд результатів і історії спроб. Окремо реалізовано інтерфейси для вчителя, які забезпечують доступ до аналітичних даних та керування навчальними підключеннями учнів. Для обмеження доступу використовується механізм перевірки ролі, що запобігає відкриттю розділів, не призначених для відповідного типу користувача.

Рівень керування станом реалізовано за допомогою Redux. Він слугує центральним сховищем даних про поточний стан застосунку, зокрема містить інформацію про авторизацію, поточного користувача та його роль.

Використання Redux підвищує передбачуваність поведінки інтерфейсу та спрощує синхронізацію даних між різними сторінками. Завдяки централізованому підходу компоненти отримують потрібні дані через селектори, а зміни виконуються через визначені дії та редюсери, що мінімізує ризик неузгодженого стану під час навігації.

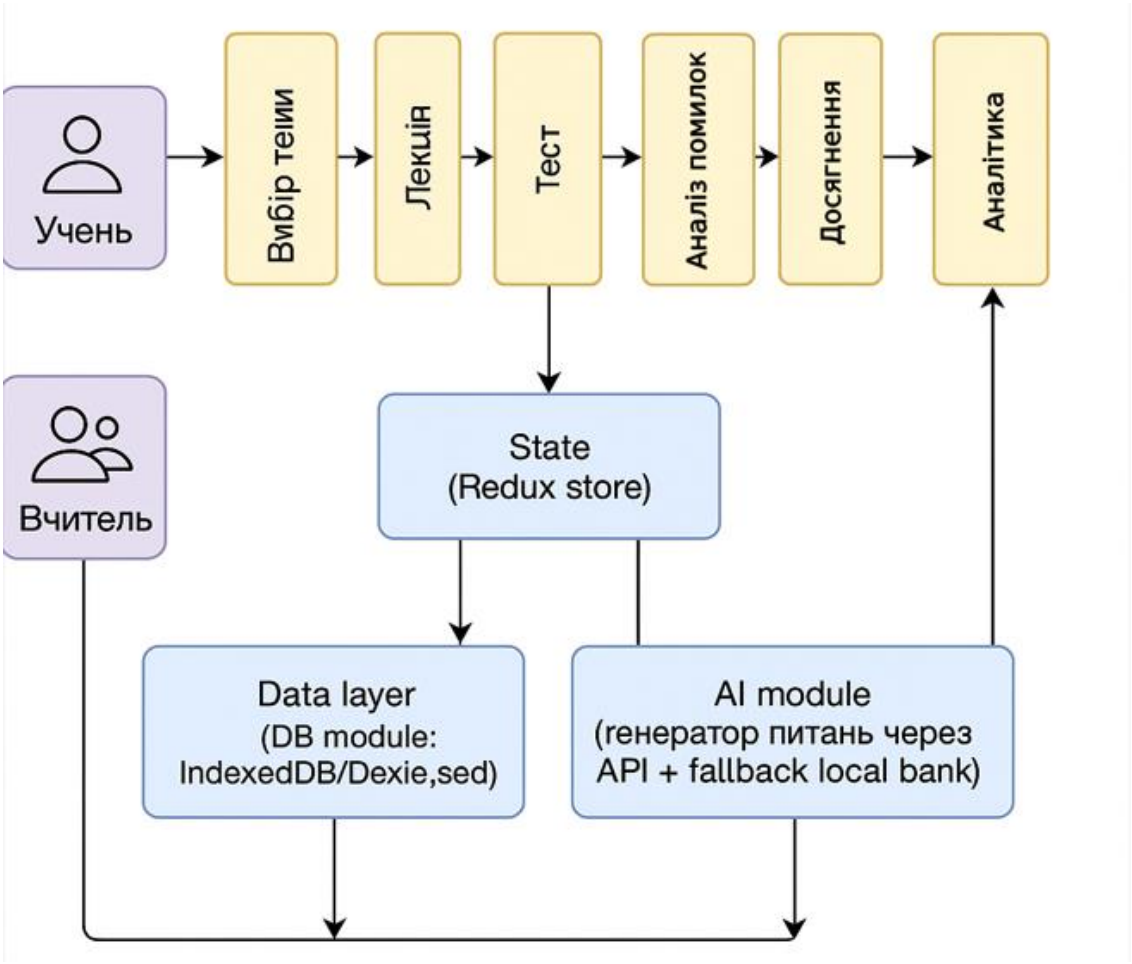


Рисунок 2.2 Узагальнена архітектура взаємодії модулів MentorAI

Модуль даних у MentorAI відповідає за локальне збереження та отримання навчальної інформації. Для демонстраційної реалізації дані зберігаються на стороні клієнта, а саме у браузері користувача. У цьому модулі зосереджено роботу з каталогом предметів і тем, навчальними матеріалами, банком питань, спробами проходження тестів, а також показниками навчального прогресу. Використання локального сховища є тестовим варіантом, який забезпечує

автономність застосунку та дозволяє швидко перевіряти логіку персоналізації. У перспективі цей модуль може бути замінений або розширений клієнтськими сервісами доступу до серверного API та бази даних без суттєвого впливу на інші частини застосунку завдяки виділенню доступу до даних у вигляді окремих функцій.

Окремим важливим компонентом є модуль інтеграції зі штучним інтелектом. Він відповідає за формування тестових завдань на основі інформації про тему та типові помилки учня. Ключовою вимогою стало забезпечення працездатності застосунку навіть у випадку недоступності генерації. Тому використовується комбінований підхід, коли генеровані питання за можливості доповнюють або замінюють частину локального банку, а за відсутності ШІ застосовується резервний сценарій. Даний модуль також підтримує принцип індивідуалізації через орієнтацію на слабкі концепти, що визначаються на основі статистики неправильних відповідей у попередніх спробах. Це дозволяє реалізувати адаптаційний механізм, коли наступні завдання з високою імовірністю включають питання з проблемних аспектів теми.

З погляду взаємодії між модулями загальна логіка роботи MentorAI полягає в тому, що дії користувача в інтерфейсі ініціюють запити до модуля даних та, за необхідності, до модуля генерації питань. Після отримання результатів відбувається оновлення стану застосунку та відображення змін у інтерфейсі. Наприклад, після проходження тесту обчислюється результат, формується запис спроби з «знімком» питань, оновлюються показники прогресу, а користувачу стає доступним екран результатів та навігація до перегляду відповідей і історії спроб. Для ролі вчителя доступними є узагальнені дані щодо активності учнів і результатів, що дозволяє виконувати базовий моніторинг успішності.

Крім того, структура проєкту в середовищі Visual Studio Code (рис. 2.3) демонструє модульний поділ на сторінки, макети, модулі стану, даних і інтеграції ШІ, що спрощує супровід та масштабування застосунку.

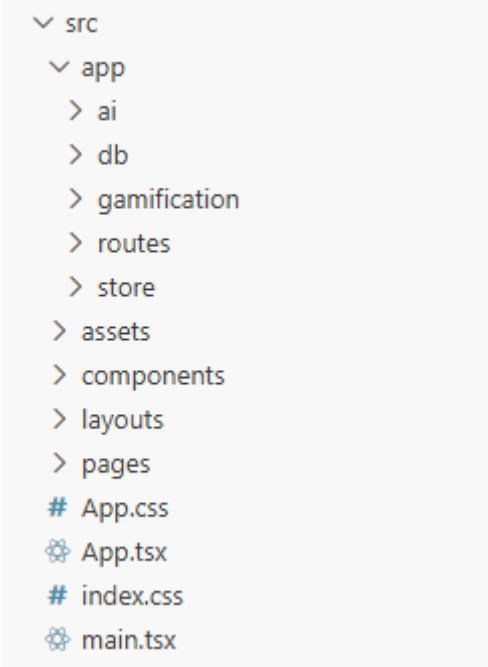


Рисунок 2.3 Структура проєкту MentorAI у середовищі Visual Studio Code

Запропонована архітектура MentorAI має суттєві переваги з позиції підтримки та розвитку. Компонентна структура React спрощує повторне використання елементів інтерфейсу, Redux забезпечує керованість стану та зменшує кількість логічних помилок у навігації, а виділення модулів доступу до даних і ШІ формує основу для подальшого масштабування. У майбутньому архітектуру доцільно розширити серверною частиною з базою даних, системою реєстрації користувачів, керуванням навчальними планами та підвищеними вимогами до безпеки. Таким чином, MentorAI у поточній реалізації виступає як працездатний прототип, який демонструє основні механізми індивідуалізованого навчання на основі ШІ та створює підґрунтя для переходу до повноцінної освітньої платформи.

2.4. Обґрунтування вибору інструментальних засобів розробки

Розробка сучасного програмного продукту потребує застосування надійних та актуальних інструментальних засобів, які забезпечують

ефективність роботи, якість коду, зручність супроводу та можливість подальшого розвитку системи. Під час створення освітнього вебзастосунку MentorAI було обрано набір технологій, що відповідає вимогам до інтерактивного клієнтського застосунку, підтримує компонентний підхід, забезпечує керованість стану та дозволяє реалізувати інтеграцію зі штучним інтелектом. Окремо враховано навчально-демонстраційний характер проєкту, тому частина рішень спрямована на спрощення розгортання і тестування без серверної інфраструктури, з можливістю подальшого переходу до клієнт–серверної архітектури.

Основні інструментальні засоби та технології, використані в процесі розробки MentorAI, включають:

– інтегроване середовище розробки Visual Studio Code;

– мову програмування TypeScript та бібліотеку React для побудови користувацького інтерфейсу;

– систему керування станом Redux Toolkit для централізованого зберігання ключових даних застосунку;

– інструмент компоновки та запуску проєкту Vite, що забезпечує швидку збірку, гаряче оновлення та зручний цикл розробки;

– бібліотеку Dexie як обгортку над IndexedDB для локального збереження навчальних даних у браузері в межах прототипу;

– React Router для організації багатосторінкової навігації та доступу до розділів відповідно до ролі користувача;

– ESLint та Prettier для підтримки єдиного стилю коду та зменшення кількості типових помилок;

– інтеграцію з провайдерами ШІ через HTTP-запити для генерації тестових завдань, а також механізм резервного сценарію на основі локального банку питань.

Використання зазначеного набору інструментів дало змогу реалізувати ключові функції MentorAI, зокрема рольову модель користувачів, навчальний

каталог, механізм тестування з автоматичною перевіркою, збереження історії спроб та адаптацію завдань з урахуванням помилок. Обраний стек є поширеним у сучасній веброзробці, має широку підтримку спільноти та документації, а також дозволяє надалі розширити проєкт за рахунок підключення серверної частини, централізованої бази даних і додаткових аналітичних можливостей. Сформований набір засобів став основою для деталізації вибору окремих інструментів, що буде розглянуто у відповідних підпунктах цього розділу.

2.4.1. Вибір Visual Studio для розробки веборієнтованої платформи

Інтегроване середовище розробки є одним із ключових інструментів під час створення сучасних вебзастосунків, оскільки воно забезпечує зручність у написанні коду, налагодженні, запуску, тестуванні та підтримці програмного продукту. Під час реалізації MentorAI основним середовищем розробки було обрано Visual Studio Code, яке є одним із найпоширеніших кросплатформних редакторів коду для веброзробки. Вибір саме цього середовища зумовлений його легкістю, швидкою роботою, гнучкістю налаштувань, широкою екосистемою розширень і зручністю інтеграції з технологіями, використаними у проєкті.

Важливим фактором вибору Visual Studio Code стала його якісна підтримка розробки на базі JavaScript та TypeScript. У межах MentorAI застосовується TypeScript для підвищення надійності коду, тому потреба в статичній перевірці типів, підказках під час написання, швидкому переході до визначень і навігації між файлами має суттєве значення. Visual Studio Code забезпечує інтелектуальне автодоповнення коду, підказки щодо типів і параметрів функцій, а також швидке виявлення помилок ще на етапі написання. Це сприяє зменшенню кількості синтаксичних і логічних неточностей та підвищує якість реалізації компонентів і модулів застосунку.

Окремою перевагою є підтримка вбудованого термінала, що є важливим для вебпроєктів на базі Node.js. У процесі роботи над MentorAI середовище

дозволяло виконувати основні операції без перемикання між програмами, зокрема встановлення залежностей, запуск локального сервера розробки, збирання проєкту та виконання допоміжних команд. Це пришвидшує цикл розробки, оскільки запуск застосунку та перевірка змін відбуваються оперативно, а завдяки механізмам гарячого оновлення під час розробки інтерфейс оновлюється практично миттєво.

Важливою складовою роботи над MentorAI стало налагодження та перевірка поведінки застосунку в браузері, зокрема під час маршрутизації, авторизації за ролями, проходження тестів та перегляду історії спроб. Visual Studio Code має зручні засоби для інтеграції з інструментами розробника браузера, а також підтримує запуск і налагодження JavaScript та TypeScript коду з використанням відповідних розширень. Така можливість дозволяє аналізувати стан застосунку, перевіряти значення змінних, відстежувати помилки у консолі та швидко знаходити причини некоректної роботи окремих елементів.

Суттєвим аргументом на користь Visual Studio Code є розвинена система розширень. Під час розробки MentorAI корисними є інструменти для контролю стилю коду та форматування, зокрема розширення для ESLint і Prettier. Вони дозволяють підтримувати єдиний стиль написання коду, уникати випадкових синтаксичних відхилень і спрощують читання та супровід проєкту. Також середовище забезпечує зручну роботу з файловою структурою, що є важливим за умови модульної організації застосунку з окремими папками для сторінок, макетів, стану, модуля даних і модуля інтеграції ШІ.

Додатковою перевагою Visual Studio Code є вбудована підтримка системи контролю версій Git. Це дозволяє вести історію змін проєкту, фіксувати результати окремих етапів реалізації, повертатися до попередніх версій у разі помилок, а також підтримувати дисципліну розробки за допомогою логічних комітів. Для освітнього проєкту це також корисно як з позиції організації роботи, так і з позиції демонстрації процесу створення програмного продукту.

Приклад структури проєкту MentorAI, відкритого у Visual Studio Code, подано у вигляді ілюстрації, що демонструє модульний поділ застосунку на основні складові (рис. 2.4).

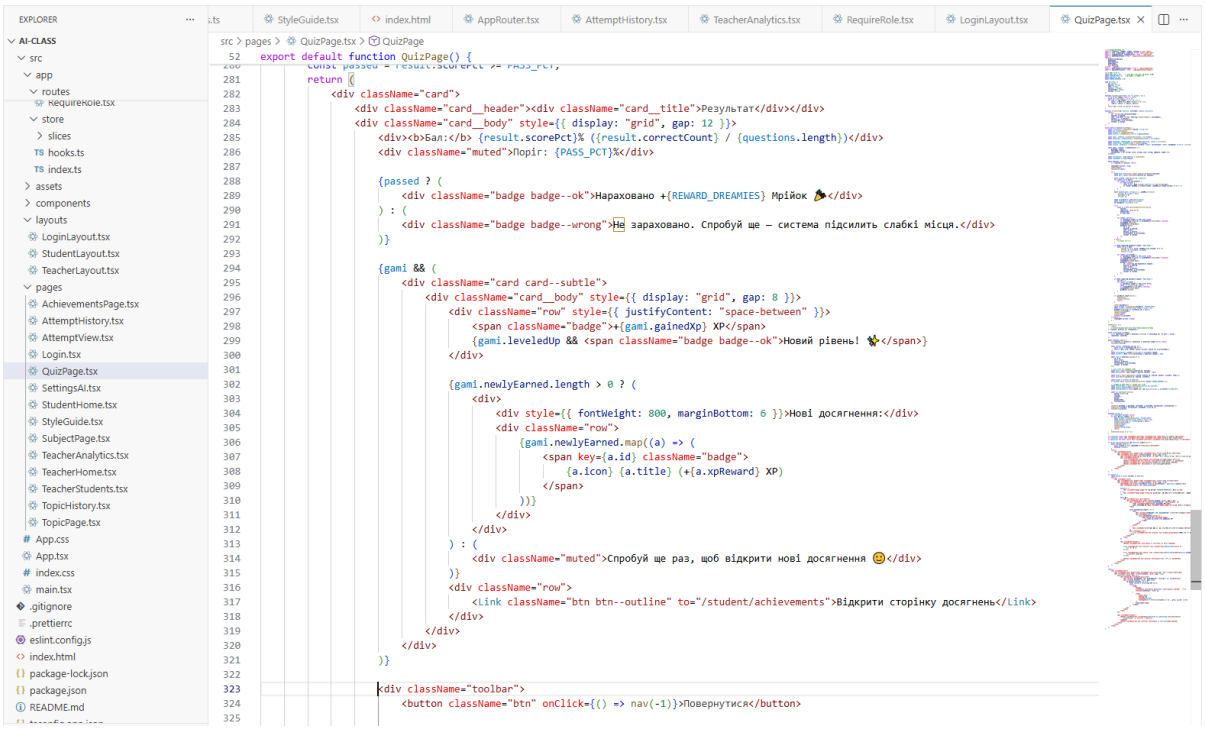


Рисунок 2.4 Робоче середовище Visual Studio Code під час розробки MentorAI

Отже, вибір Visual Studio Code як основного середовища розробки для MentorAI є обґрунтованим, оскільки воно забезпечує зручний цикл створення вебзастосунку – від написання і форматування коду до запуску, налагодження, контролю версій та організації структури проєкту. Використання цього середовища сприяло підвищенню якості реалізації, скороченню часу розробки та забезпеченню керованості змін у процесі створення освітнього вебзастосунку.

2.4.2. Застосування мови програмування TypeScript та бібліотеки React для розробки вебзастосунку MentorAI

Вибір мови програмування та технологічної платформи для реалізації вебзастосунку є одним із ключових етапів розробки, оскільки саме він визначає продуктивність, зручність супроводу, стабільність, можливість масштабування та швидкість подальшого розвитку програмного продукту. Для створення освітнього вебзастосунку MentorAI базовими інструментами реалізації клієнтської частини було обрано TypeScript та бібліотеку React. Таке поєднання є типовим для сучасної веброзробки та дає змогу будувати інтерфейс як набір незалежних компонентів, що легко повторно використовуються, тестуються і розширюються.

TypeScript є надбудовою над JavaScript і забезпечує статичну типізацію, що позитивно впливає на надійність коду та зменшує кількість помилок під час розробки. Для освітнього застосунку, у якому важливо коректно обробляти дані тестів, результати спроб, потоки навігації та рольові обмеження, типізація є суттєвою перевагою. Вона дозволяє встановлювати чіткі контракти між компонентами, описувати структури даних, зокрема сутності користувача, питання, варіанти відповідей та спроби, а також забезпечує контроль правильності використання функцій доступу до даних. Завдяки цьому підвищується стабільність системи під час розширення функціональності, адже значна частина помилок виявляється ще на етапі компіляції.

React використано для побудови інтерфейсу користувача як односторінкового застосунку з багатосторінковою навігацією. Компонентний підхід React дозволив спроєктувати MentorAI як набір логічно відокремлених елементів – сторінок, макетів і повторно використовуваних UI-компонентів. Для сценаріїв навчання це є особливо важливим, оскільки інтерфейс постійно змінюється залежно від стану користувача. Наприклад, учень переходить від

вибору предмета до теми, далі до лекційного матеріалу, тесту, результатів та перегляду історії спроб, і на кожному етапі застосунок має відображати актуальні дані без перезавантаження сторінки. React забезпечує таку інтерактивність завдяки керуванню станом компонентів, реактивному оновленню інтерфейсу та підтримці життєвого циклу.

Для централізованого керування станом застосунку використано Redux Toolkit. Це дозволяє зберігати ключові дані, пов’язані з авторизацією, роллю користувача та іншими глобальними налаштуваннями, в одному передбачуваному сховищі. У контексті MentorAI це важливо для коректного розмежування доступу між учнем і вчителем, стабільної навігації між сторінками, а також збереження поточного стану при переходах. Використання Redux Toolkit спрощує реалізацію типових сценаріїв через дії та редюсери, зменшує кількість дублювання логіки в компонентах і полегшує супровід коду.

Окремо слід виділити значення обраного стеку для реалізації інтеграції зі штучним інтелектом. У MentorAI передбачено генерацію тестових завдань через зовнішні API, тому важливими є підтримка асинхронних операцій, робота з мережевими запитами та контроль коректності отриманих даних. TypeScript у цьому випадку допомагає описувати контракт відповідей і структуру згенерованих питанням, а React забезпечує коректну взаємодію користувача з процесом завантаження, відображення і перевірки тесту. Додатково реалізовано резервний сценарій, що дозволяє застосунку працювати навіть за відсутності генерації, використовуючи локальний банк питань. Це підтримує стабільність системи та забезпечує безперервність навчального процесу.

Таким чином, застосування TypeScript та React у проєкті MentorAI є обґрунтованим рішенням як з технічної, так і з практичної точки зору. TypeScript підвищує надійність коду завдяки типізації та контролю структури даних, а React забезпечує швидкий і гнучкий користувацький інтерфейс, придатний для інтерактивних навчальних сценаріїв. У поєднанні з Redux Toolkit це створює основу для керованої, модульної та розширюваної реалізації, яка в майбутньому

може бути доповнена серверною частиною та централізованою базою даних без потреби повного перепроєктування клієнтського застосунку.

2.4.3. Застосування локального сховища IndexedDB як тестового рішення та перспективи переходу до серверної бази даних (SQL Server)

Одним із ключових компонентів будь-якої сучасної інформаційної системи є механізм зберігання даних, який забезпечує збереження навчального контенту, результатів користувачів, зв'язків між сутностями та можливість подальшої аналітики. Для освітнього вебзастосунку MentorAI потреба у збереженні даних проявляється насамперед у фіксації результатів тестування, історії спроб, інформації про доступ учнів до предметів, а також у зберіганні каталогу тем і банку питань. У межах виконання практичної частини магістерської роботи було прийнято рішення використати локальне сховище даних у браузері як тестовий варіант, що дозволяє зробити застосунок автономним і спрощує демонстрацію його функціональних можливостей без розгортання серверної інфраструктури.

Для реалізації локального збереження в MentorAI застосовано IndexedDB, що є вбудованим механізмом браузера для зберігання структурованих даних. На практиці IndexedDB дозволяє організувати сховище з логікою, близькою до роботи з базою даних, зокрема виконувати запис, вибірку, фільтрацію та сортування сутностей. Щоб спростити роботу з IndexedDB і уникнути надмірної кількості низькорівневого коду, використано бібліотеку Dexie, яка надає зрозумілий API для опису таблиць, індексів і типових операцій доступу до даних. Завдяки цьому в MentorAI було реалізовано збереження каталогу предметів і тем, навчальних матеріалів, питань, спроб проходження тестів і показників прогресу, а також виконано механізм ініціалізації початкових даних, необхідних для роботи прототипу.

Використання локального сховища має низку практичних переваг у контексті навчального проєкту. По-перше, це забезпечує просте розгортання системи, адже застосунок не потребує серверної частини та може бути запущений на будь-якому комп’ютері з браузером. По-друге, це дозволяє швидко тестувати логіку індивідуалізації, алгоритм формування адаптивних тестів, механізми навігації та відображення результатів. По-третє, локальна модель корисна для прототипування, коли основною метою є перевірка коректності сценаріїв і взаємодії модулів, а не організація повноцінного багатокористувацького середовища.

Водночас локальне збереження даних у браузері має обмеження, що впливають на можливість практичного використання застосунку в реальних умовах. Дані зберігаються лише на пристрої користувача, не синхронізуються між різними пристроями та можуть бути втрачені, наприклад, після очищення кешу або налаштувань браузера. Крім того, такий підхід не забезпечує централізованого керування обліковими записами, правами доступу та політиками безпеки, що є критично важливим для освітніх систем, які працюють з персональними даними учнів.

З урахуванням зазначеного, для подальшого розвитку MentorAI доцільним є перехід до повноцінної клієнт–серверної архітектури з використанням серверної бази даних. У такому випадку можливим варіантом може бути застосування реляційної системи керування базами даних, зокрема SQL Server, що дозволить централізовано зберігати облікові записи користувачів, структуру навчальних предметів і тем, історію спроб, показники прогресу та дані для аналітики. Використання серверної бази даних забезпечить надійність збереження, контроль доступу, резервне копіювання, масштабованість та можливість розгортання MentorAI для роботи з багатьма користувачами одночасно в межах навчального закладу.

Отже, у межах поточної реалізації MentorAI збереження даних організовано локально як тестове рішення, що відповідає завданням

прототипування та демонстрації індивідуалізованого навчання з використанням ШІ. Разом із тим, для практичного впровадження у закладах освіти необхідним є подальший розвиток системи із переходом до централізованої серверної бази даних, що стане основою для безпечної, стабільної та масштабованої експлуатації застосунку.

2.5. Особливості програмної реалізації

Розробка вебзастосунку MentorAI вимагала продуманого підходу до організації структури коду, уважного розподілу відповідальностей між модулями та застосування сучасних практик побудови клієнтських застосунків. Основною метою було створити не лише працездатний прототип для демонстрації можливостей індивідуалізованого навчання з використанням штучного інтелекту, а й рішення, яке можна надалі розвивати та масштабувати. Зокрема, у перспективі MentorAI може бути доповнений серверною частиною, централізованим збереженням даних у базі даних, розширеним керуванням користувачами та поглибленою аналітикою успішності. Водночас у поточній версії локальне збереження використовується як тестовий варіант для демонстрації логіки та сценаріїв роботи.

Архітектурно MentorAI побудовано як клієнтський вебзастосунок на React з використанням TypeScript. У застосунку реалізовано багатосторінкову навігацію, розділення інтерфейсів за ролями, централізоване керування станом через Redux, локальний шар даних на базі IndexedDB (з використанням Dexie) та модуль інтеграції ШІ для генерації тестових завдань із резервним сценарієм, що гарантує працездатність навіть за відсутності доступу до зовнішнього API. Така організація дозволяє підтримувати керованість коду, зменшує дублювання логіки та формує основу для подальшого розширення без критичного втручання в уже реалізовані частини.

Однією з ключових особливостей програмної реалізації MentorAI є організація маршрутизації, макетів сторінок та контролю доступу за ролями. У застосунку передбачено дві базові ролі користувачів – учень та вчитель. Кожна роль має власний набір доступних сторінок, що відповідає їхнім задачам: учень працює з навчальним контентом, тестуванням та переглядом результатів, тоді як вчитель отримує інструменти для моніторингу успішності та керування підключенням учнів до предметів. Для забезпечення логічності сценаріїв та захисту від несанкціонованого доступу реалізовано рольовий контроль на рівні маршрутів.

Маршрутизація побудована за допомогою React Router і організована як дерево маршрутів із вкладеними гілками для кожної ролі. Центральним файлом, який описує структуру навігації, виступає AppRouter.tsx (рис. 2.5). Саме в ньому визначаються основні шляхи, вкладені сторінки, а також поведінка застосунку для невідомих адрес. Структурно логіка виглядає таким чином: кореневий шлях відповідає сторінці входу, гілка /student містить сторінки кабінету учня, гілка /teacher – сторінки кабінету вчителя, а для всіх інших адрес передбачено перенаправлення до стартового екрану. Це забезпечує передбачувану навігацію та запобігає появі «порожніх» або некоректних сторінок.

Окремо в MentorAI реалізовано контроль доступу до розділів за ролями, що є критично важливим для коректної логіки роботи. Навіть якщо користувач вручну введе адресу іншого розділу в адресному рядку, застосунок не повинен відкривати сторінки, які не відповідають його ролі. Для цього застосовано компонент-захисник маршрутів RequireRole, який обгортає гілки /student і /teacher. Компонент отримує цільову роль, звертається до глобального стану (Redux) та визначає, чи є користувач авторизованим і чи має він потрібні права доступу. Якщо користувач не авторизований, система блокує доступ і виводить повідомлення про необхідність авторизації або виконує перенаправлення на сторінку входу. Якщо користувач авторизований, але його роль не збігається з роллю розділу, застосунок не дозволяє перейти далі та запобігає відкриттю

невідповідного інтерфейсу. Такий підхід зменшує ризик помилок у сценаріях, підтримує цілісність даних і забезпечує коректний користувацький досвід.

```
const router = createBrowserRouter([
    {
        path: "/",
        element: <LoginLayout />,
        children: [{ index: true, element: <Login /> }],
    },

    // settings доступний і учню, і вчителю (якщо не хочеш — можна потім обмежити)
    { path: "/settings-ai", element: <SettingsAI /> },

    {
        path: "/student",
        element: (
            <RequireRole role="student">
                <StudentLayout />
            </RequireRole>
        ),
        children: [
            { index: true, element: <StudentHome /> },
            { path: "achievements", element: <AchievementsPage /> },

            { path: "subject/:subjectId", element: <SubjectPage /> },
            { path: "topic/:topicId", element: <TopicPage /> },

            { path: "topic/:topicId/quiz", element: <QuizPage /> },
            { path: "topic/:topicId/history", element: <AttemptHistory /> },
            { path: "topic/:topicId/attempt/:attemptId", element: <AttemptView /> },
        ],
    },

    {
        path: "/teacher",
        element: (
            <RequireRole role="teacher">
                <TeacherLayout />
            </RequireRole>
        ),
        children: [
            { index: true, element: <TeacherHome /> },
            { path: "students", element: <TeacherStudents /> },
            { path: "analytics", element: <TeacherAnalytics /> },
        ],
    },

    { path: "*", element: <Navigate to="/" replace /> },
]);
```

Рисунок 2.5 Структура маршрутів у файлі AppRouter.tsx

Щоб інтерфейс залишався цілісним та однаково оформленим у межах кожного кабінету, застосовано підхід із макетами сторінок. Для учня використано StudentLayout (рис. 2.6), для вчителя – TeacherLayout (рис. 2.7), а для сторінки входу – окремий макет стартового екрану. Макет у цьому контексті виконує роль контейнера з повторюваними елементами інтерфейсу: верхньою панеллю, брендом, пунктами навігації, інформацією про користувача та кнопкою виходу з акаунту. Усередині макету відображається поточна сторінка відповідного розділу, що забезпечується механізмом <Outlet /> у React Router.

Таким чином, при переходах між сторінками в межах одного кабінету змінюється лише основний контент, тоді як навігаційна частина залишається сталою. Це підвищує зручність використання, зменшує когнітивне навантаження на користувача та одночасно спрощує підтримку коду, оскільки навігаційні елементи не дублюються у кожній сторінці.

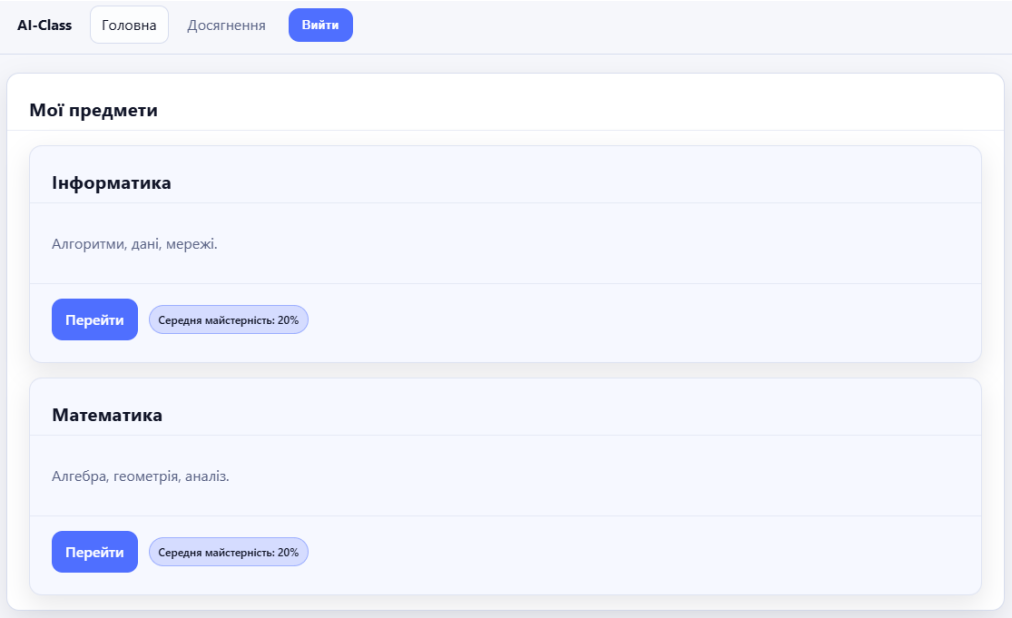


Рисунок 2.6 Макет кабінету учня з верхньою панеллю навігації

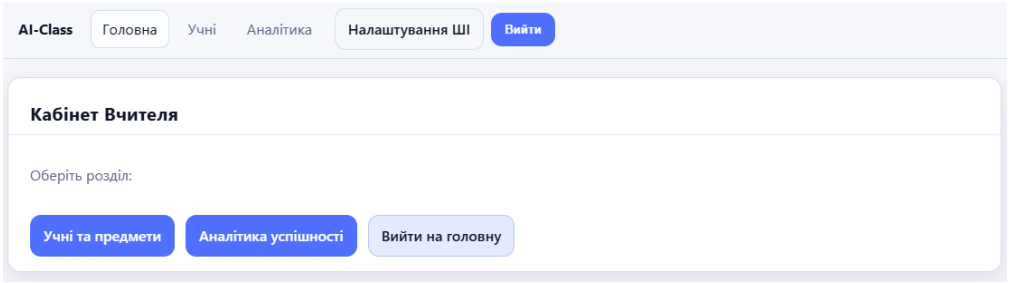


Рисунок 2.7 Макет кабінету вчителя з верхньою панеллю навігації

У підсумку можна зазначити, що реалізована маршрутизація, система макетів та рольовий контроль доступу формують основу архітектури MentorAI на рівні взаємодії користувача з платформою. Це рішення забезпечує чіткий поділ на кабінети учня і вчителя, логічну навігацію між сторінками, єдність

інтерфейсу в межах кожного розділу та захист від доступу до функцій, не передбачених роллю користувача.

Суттєвою частиною програмної реалізації MentorAI є централізоване керування станом застосунку, оскільки в системі необхідно підтримувати узгодженість інтерфейсу під час переходів між сторінками, коректно працювати з даними авторизації та забезпечувати рольову логіку доступу. Для цього використано Redux Toolkit, що дозволяє організувати єдине сховище стану та керувати ним через передбачувані дії. Такий підхід особливо важливий для застосунків із розділенням на ролі, адже роль і профіль користувача впливають на маршрутизацію, доступні сторінки, елементи навігації та поведінку захищених компонентів.

Ключовим модулем у цьому контексті виступає authSlice.ts, у якому описано стан авторизації та базові операції, пов'язані з користувачем. У стані зберігаються дані поточного профілю, роль, ознака авторизованості та допоміжні параметри, необхідні для коректного відображення інтерфейсу. Логіка побудована так, щоб після входу користувача система мала доступ до його ідентифікатора, імені та ролі. Це дозволяє, з одного боку, показувати персоналізовану інформацію (наприклад, привітання або індикатори прогресу), а з іншого – коректно застосовувати обмеження доступу до сторінок через механізм перевірки ролі.

У authSlice.ts реалізовано набір дій для зміни стану, зокрема сценарії входу та виходу, а також допоміжні асинхронні операції, що можуть взаємодіяти з локальним сховищем даних (рис. 2.8). Наприклад, під час входу може виконуватися пошук користувача в локальній базі або створення нового профілю, якщо користувач входить уперше. Вихід із системи очищує дані поточного профілю у стані, що автоматично призводить до повернення до стартової сторінки та блокування доступу до захищених розділів. Таким чином забезпечується повний цикл авторизації без потреби дублювати логіку на багатьох сторінках.

```
export const addDreamiesThunk = createAsyncThunk(
    "auth/addDreamies",
    async ({ amount }: { amount: number }, { getState }) => {
        const state = getState() as { auth: AuthState };
        const p = state.auth.profile;
        if (!p) throw new Error("No profile in auth state");

        const newVal = (p.dreamies ?? 0) + amount;
        await updateUser(p.id, { dreamies: newVal });
        const fresh = await getUser(p.id);
        if (!fresh) throw new Error("Failed to read updated user");

        return { id: fresh.id, name: fresh.name, role: fresh.role as Role, dreamies: fresh.dreamies ?? newVal };
    }
);

const authSlice = createSlice({
    name: "auth",
    initialState,
    reducers: {
        loginAs(state, action: { payload: { name: string; role: Role } }) {
            state.name = action.payload.name.trim();
            state.role = action.payload.role;
            state.profile = null; // заповнюється після пошуку/створення у IndexedDB
        },
        logout(state) {
            state.name = null;
            state.role = null;
            state.id = null;
            state.profile = null;
        },
        setId(state, action: { payload: string | null }) {
            state.id = action.payload;
        },
        setProfile(state, action: { payload: Profile | null }) {
            state.profile = action.payload;
            if (action.payload) {
                state.id = action.payload.id;
                state.name = action.payload.name;
                state.role = action.payload.role;
            }
        },
    },
```

Рисунок 2.8 Фрагмент файлу authSlice.ts: структура стану та дії (signIn, signOut, thunks)

Для зручного використання Redux у React-компонентах у MentorAI застосовано типізовані «хуки» доступу до сховища, які зазвичай визначаються в модулі store або окремому файлі hooks. Використання таких хуків дозволяє уникнути повторюваного опису типів для вибірки стану (useSelector) та для відправлення дій (useDispatch). Це підвищує читабельність компонентів і зменшує ймовірність помилок під час роботи зі станом, оскільки TypeScript контролює відповідність типів та підказує правильні поля. У результаті компоненти сторінок можуть зосереджуватися на бізнес-логіці інтерфейсу, а не на технічних деталях доступу до Redux.

Сценарій входу користувача реалізовано на стартовій сторінці логіну (рис. 2.9). На цьому етапі користувач вводить ім’я або обирає відповідний профіль, після чого визначає роль входу. З педагогічної точки зору це спрощує демонстрацію ролей і сценаріїв під час тестування, а з технічної – дозволяє швидко перемикатися між режимами учня та вчителя без складної процедури реєстрації, що є доцільним для прототипу. Після виконання дії входу дані записуються до Redux-стану, і застосунок автоматично перенаправляє користувача до відповідного кабінету.

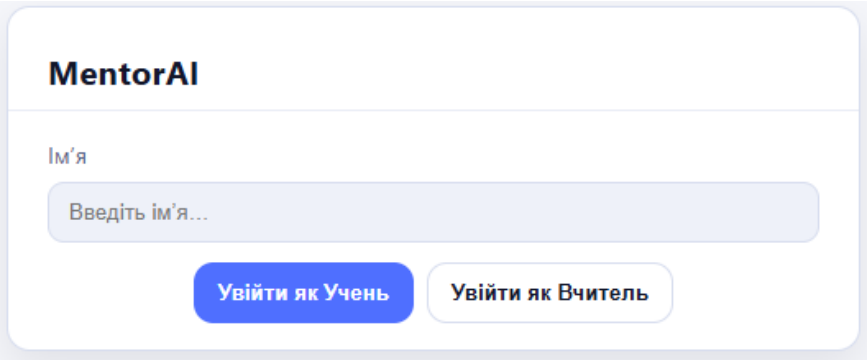


Рисунок 2.9 Сторінка логіну MentorAI

Після авторизації змінюється інтерфейс та доступні елементи навігації. Наприклад, учень бачить розділи, пов’язані з навчанням, темами, тестуванням, історією спроб, прогресом і досягненнями. Вчитель натомість отримує доступ до сторінок аналітики та керування підключенням учнів до предметів. Ця зміна відбувається не через приховані умови у кожній сторінці, а як наслідок зміни централізованого стану у Redux та роботи рольової маршрутизації. Практично це означає, що одна інформація – роль користувача – визначає поведінку значної частини інтерфейсу в усьому застосунку. Саме таким чином досягається узгодженість роботи системи, зменшується дублювання перевірок та підвищується надійність логіки.

Отже, застосування Redux у MentorAI забезпечило централізований підхід до керування станом, насамперед у частині авторизації та ролей. Реалізація

authSlice, типізованих хуків та сценарію входу дозволила сформувати основу для коректної роботи маршрутизації, розмежування прав доступу та стабільної поведінки інтерфейсу при переходах між сторінками. Це рішення є важливою складовою архітектури MentorAI та створює передумови для подальшого розвитку платформи, зокрема при переході до серверної авторизації та централізованого зберігання даних.

Навчальний сценарій учня у MentorAI побудовано як послідовний ланцюжок дій: вибір предмета, вибір теми, ознайомлення з коротким навчальним матеріалом та проходження тесту. Така структура відповідає логіці навчального процесу в середній школі, коли теоретичний блок передує контролю знань, а результати фіксуються для подальшого аналізу прогресу.

Після входу в систему учень потрапляє на головну сторінку кабінету, де відображаються доступні йому предмети. Перехід до предмета відкриває сторінку SubjectPage, яка завантажує список тем, прив'язаних до обраного предмета, та відображає їх у вигляді упорядкованого переліку (рис. 2.10). Кожна тема виступає точкою входу до конкретної навчальної одиниці, а навігація реалізована через маршрути з параметрами, що дозволяє відкривати конкретну тему за її ідентифікатором.

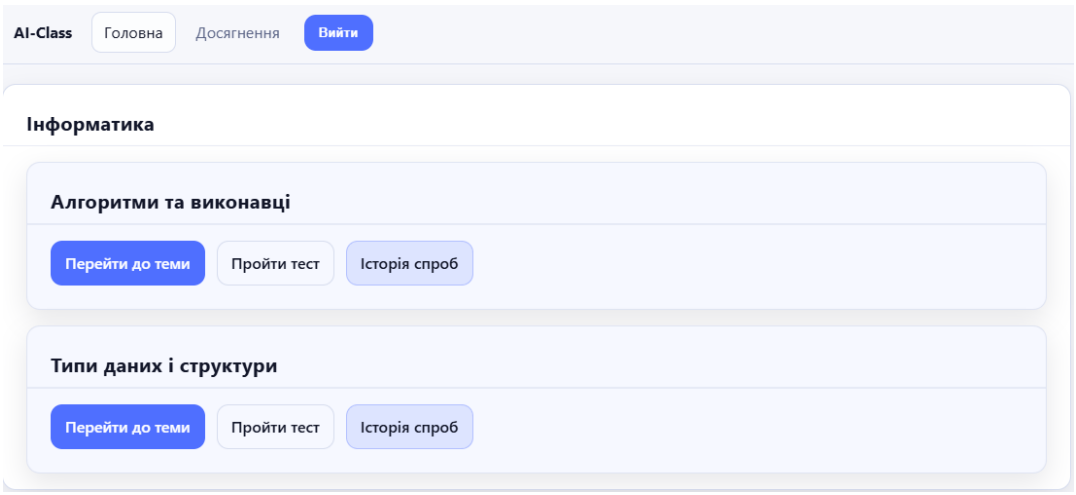


Рисунок 2.10 Сторінка предмета (SubjectPage)

Після вибору теми відкривається TopicPage. Її задача – сформувати «вузол теми», де учень бачить короткий опис, переходить до лекційного матеріалу та запускає тестування. На сторінці виконується завантаження:

– даних уроку, прив’язаного до теми;

– базової інформації про тему;

– елементів навігації назад до предмета та вперед до тесту.

З погляду програмної реалізації важливо, що TopicPage використовує topicId з URL-параметрів і на цій основі підтягує потрібний контент (рис. 2.11). Це забезпечує коректність навігації та дозволяє прямо відкривати конкретну тему через адресний рядок.

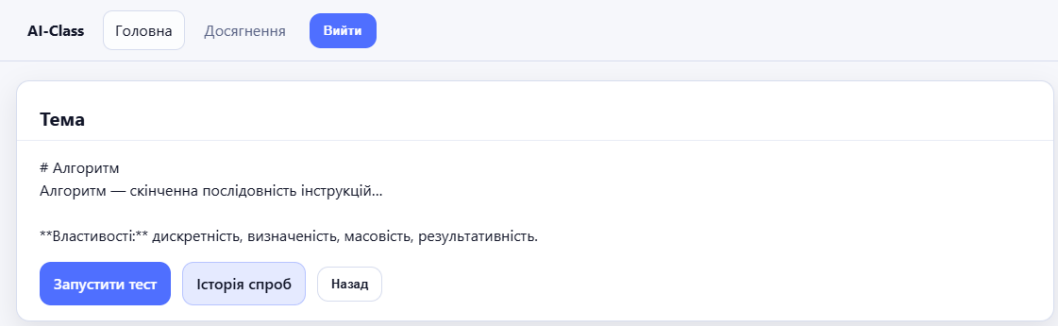


Рисунок 2.11 – Сторінка теми (TopicPage), відображення теми та кнопки переходу до уроку і тесту

Лекційний матеріал реалізовано як короткий текстовий блок у форматі Markdown. Такий формат обраний через його простоту для створення навчального контенту та можливість додавати структуровані елементи: заголовки, списки, акценти, формули у спрощеному вигляді. На сторінці теми або в окремому компоненті (LessonView/Markdown) текст уроку відображається з базовим форматуванням, після чого учень має можливість перейти до тесту. Таким чином, теорія і практика логічно пов’язані в одному сценарії та не розриваються переходами на сторонні ресурси.

Ключова частина логіки TopicPage полягає в завантаженні матеріалу за topicId та формуванні правильних посилань для переходів (рис. 2.12). У коді це реалізовано через: отримання параметра маршруту, виклик функції отримання уроку з локального сховища та умовне відображення контенту залежно від наявності даних. Також у TopicPage визначено навігаційні кнопки, які ведуть до сторінки тесту для цієї теми, що важливо для цілісності сценарію «урок → контроль знань».

```
export default function TopicPage() {
    const { topicId } = useParams<{ topicId: string }>();
    const dispatch = useAppDispatch();
    const nav = useNavigate();

    const lesson = useAppSelector((s) => (topicId ? s.catalog.lessonByTopic[topicId] : undefined));
    const loading = useAppSelector((s) => s.catalog.loading);

    useEffect(() => {
        if (topicId && !lesson) {
            dispatch(loadLesson(topicId));
        }
    }, [dispatch, topicId, lesson]);

    if (!topicId) return <div className="card"><div className="card__body">Тема не вказана.</div></div>;

    return (
        <div className="card">
            <div className="card__header">
                <div className="card__title">Тема</div>
            </div>
            <div className="card__body" style={{ display: "grid", gap: 16 }}>
                {loading && <div className="badge">Завантаження…</div>}
                {!loading && !lesson && <div className="muted">Лекційний матеріал ще не додано.</div>}
                {!loading && lesson && (
                    <div>
                        <div style={{ whiteSpace: "pre-wrap" }}>{lesson.markdown}</div>
                    </div>
                )}

                <div className="toolbar">
                    <Link className="btn" to={`/student/topic/${topicId}/quiz`}>Запустити тест</Link>
                    <Link className="btn btn--subtle" to={`/student/topic/${topicId}/history`}>Історія спроб</Link>
                    <button className="btn btn--outline" onClick={() => nav(-1)}>Назад</button>
                </div>
            </div>
        </div>
    );
}
```

Рисунок 2.12 – Фрагмент коду TopicPage.tsx: завантаження уроку за topicId та навігація до тесту

Ключовою відмінністю MentorAI від звичайного тестового тренажера є механізм індивідуалізації перевірки знань. Його мета – не лише оцінити поточний рівень засвоєння теми, а й адаптувати наступні тестові завдання під слабкі місця учня. Реалізаційно це досягається поєднанням генерації питань за

допомогою ШІ, резервного сценарію (фолбек) на основі локального банку питань, фіксації кожної спроби з «знімком» питань, а також можливості перегляду історії та детального розбору відповідей. У сукупності ці механізми утворюють замкнений цикл: учень проходить тест → система аналізує помилки → наступний тест підсилює слабкі концепти → результати накопичуються для моніторингу прогресу.

Початковою точкою цього процесу є сторінка QuizPage.tsx (рис. 2.13), яка відповідає за підготовку тесту, відображення питань, збір відповідей учня, автоматичну перевірку та збереження результатів. Перед початком тестування формується пул питань для заданої теми. Далі з цього пулу обирається певна кількість питань випадковим чином, щоб кожне проходження тесту було різним і не перетворювалося на механічне запам’ятовування порядку відповідей. У навчальному сенсі це підвищує валідність перевірки, оскільки учень змушений щоразу відтворювати знання, а не згадувати «де була правильна кнопка».

Основний пріоритет під час формування пулу – використання згенерованих ШІ питань. Модуль генерації отримує інформацію про тему та так звані «слабкі концепти», після чого повертає набір завдань у стандартизованому форматі (текст питання, варіанти відповіді, правильний індекс, тег концепту). Додатково передбачено контроль якості на рівні клієнта: відсіювання дублікатів за ідентифікатором або текстом та підтримка мінімальної різноманітності запитань. Це потрібно, щоб уникати ситуацій, коли користувач бачить однакові формулювання в межах одного проходження, що знижує користь тестування.

```
74    async function load() {
75        if (!topicId || !profile) return;
76
77        loadingRef.current = true;
78        setPool(null);
79        setQuestions(null);
80
81        try {
82            const bank: Question[] = await getQuestionsByTopic(topicId);
83            const past = await getAttempts(profile.id, topicId);
84
85            const weakMap = new Map<string, number>();
86            for (const att of past) {
87                for (const ans of att.answers) {
88                    if (!ans.correct) {
89                        const found = bank.find((b: Question) => b.id === ans.qid);
90                        if (found) weakMap.set(found.concept, (weakMap.get(found.concept) ?? 0) + 1);
91                    }
92                }
93            }
94            const weakConcepts: string[] = [...weakMap.entries()]
95                .sort((a, b) => b[1] - a[1])
96                .map(([k]) => k)
97                .slice(0, 3);
98
99            const uniqueByText = new Set<string>();
100           const uniqueById = new Set<string>();
101           let poolBuilt: QuizItem[] = [];
102
103           try {
104               const ai = await generateQuestionsForTopic({
105                   topicId,
106                   topicTitle: "Тема курсу",
107                   weakConcepts,
108                   n: POOL_SIZE,
109               });
110
111               for (const q of ai) {
112                   if (poolBuilt.length >= POOL_SIZE) break;
113                   if (uniqueById.has(q.id) || uniqueByText.has(q.text)) continue;
114                   uniqueById.add(q.id);
115                   uniqueByText.add(q.text);
116                   poolBuilt.push({
117                       id: q.id,
118                       topicId: q.topicId,
119                       text: q.text,
120                       options: q.options,
```

Рисунок 2.13 Фрагмент коду QuizPage.tsx: формування пулу питань, випадковий вибір для проходження, перевірка та запис результату

Разом із тим, MentorAI має працювати стабільно навіть тоді, коли генерація недоступна (наприклад, відсутній ключ API, немає мережі або провайдер повернув помилку). Для цього застосовано фолбек-логіку: якщо модуль ШІ не повернув достатньо питань (рис. 2.14), система добирає їх із локального банку, який зберігається в IndexedDB і ініціалізується на старті застосунку. Таким чином, тестування не блокується, а користувач завжди може продовжити навчання. Локальні питання також мають теги концептів, тому навіть у режимі фолбеку зберігається можливість «підсилення» слабких місць,

хоча гнучкість і різноманітність завдань у цьому випадку менша, ніж при повноцінній генерації.

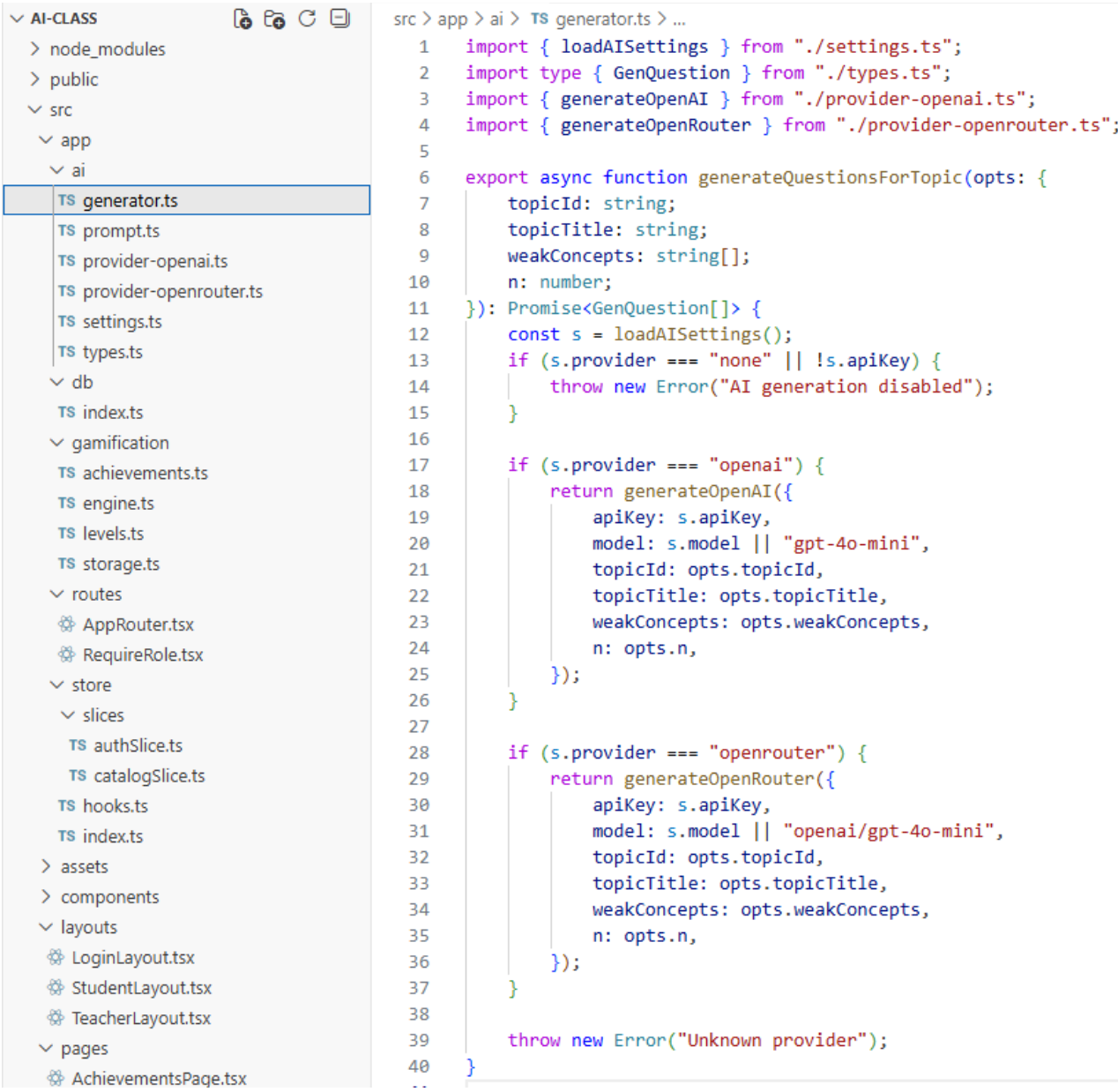


Рисунок 2.14 Фрагмент модуля генерації ШІ, формування запиту, обробка відповіді та звернення до локального банку

Механізм «підсилення слабких концептів» базується на аналізі попередніх спроб. Після кожного тесту система зберігає інформацію про те, на які питання була дана неправильна відповідь. Оскільки кожне питання має додаткову ознаку

– concept (тег концепту), MentorAI може агрегувати помилки за концептами. На наступному проходженні тесту ці концепти передаються в модуль генерації як пріоритетні, щоб ШІ сформував більше питань саме з проблемних місць. Якщо ж працює фолбек, то локальний добір також віддає пріоритет питанням із концептів, де помилок було найбільше.

Після формування тесту користувачу відображається інтерфейс проходження: питання, варіанти відповіді (у форматі радіокнопок) та кнопка завершення (рис. 2.15). Завершення стає доступним лише тоді, коли учень дав відповіді на всі показані питання. Це є важливою умовою для коректного підрахунку результату та формування зворотного зв’язку. Перевірка виконується автоматично: система порівнює вибраний варіант із правильним індексом та обчислює відсоток правильних відповідей.

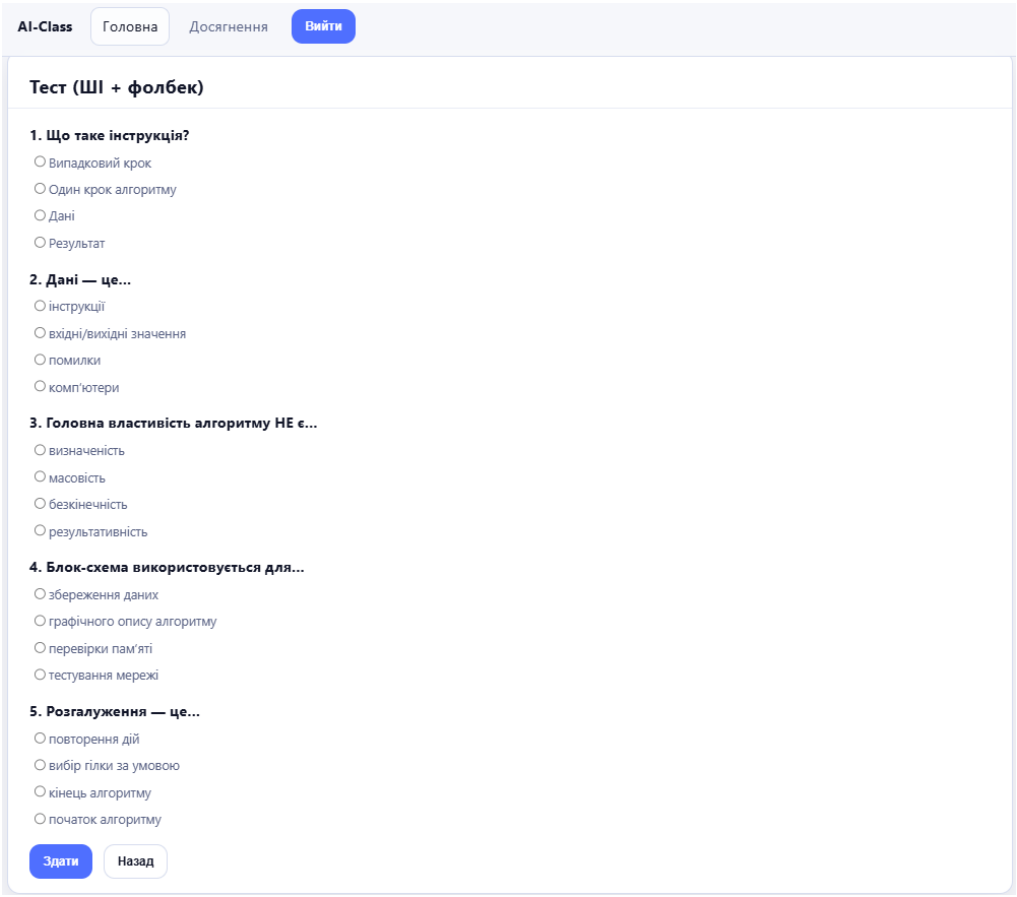


Рисунок 2.15 Екран тесту: відповіді на питання та завершення проходження

Після здачі тесту MentorAI показує підсумок: відсотковий результат, кількість правильних відповідей та статус проходження (пройдено або рекомендовано повторити) (рис. 2.16). Додатково реалізовано логіку повторного проходження, при якій система повторно формує набір питань, враховуючи результат попередньої спроби та підсилюючи слабкі місця. Це створює безперервний навчальний цикл, де повтор не є «тим самим тестом», а стає уточнюючим тренуванням.

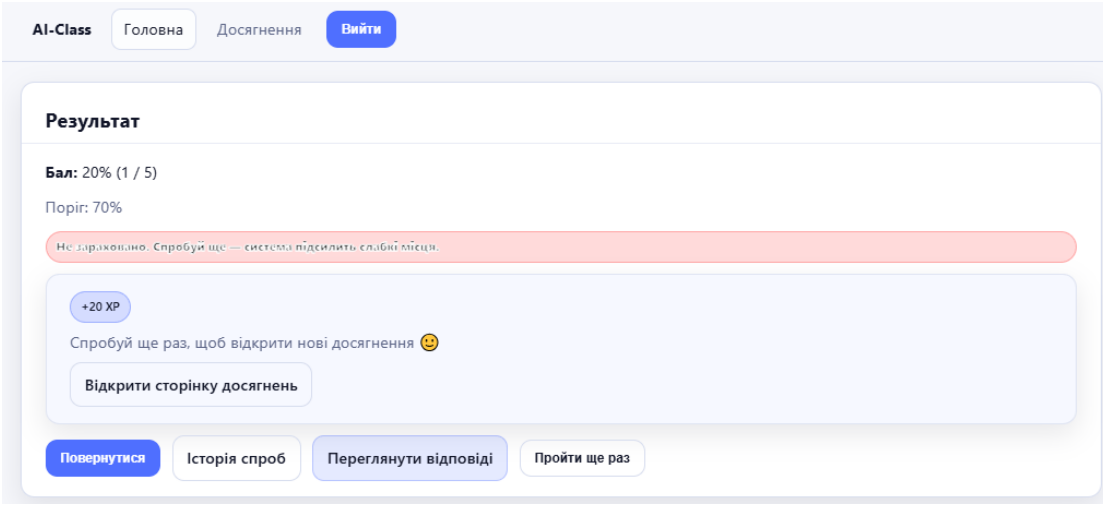


Рисунок 2.16 Екран результату: підсумковий бал і кнопки повтору та перегляду відповіді

Важливою частиною реалізації є фіксація спроби. Після завершення тесту дані зберігаються в локальному сховищі: які відповіді обрав учень, які з них правильні, підсумковий бал. Додатково MentorAI зберігає «знімок» питань, які були показані саме в цій спробі. Це принципово важливо, оскільки при генерації ШІ питання можуть бути унікальними і згодом не повторюватися. Завдяки знімку стає можливим коректний перегляд минулих спроб і пояснення помилок навіть тоді, коли в базі вже з’явився інший набір запитань для теми.

Перегляд конкретної спроби реалізовано у AttemptView.tsx. На цій сторінці для кожного питання відображається: текст завдання, позначка «правильно/неправильно», варіант відповіді учня, правильний варіант та коротке

пояснення або підказка (рис. 2.17). Пояснення прив’язане до концепту і подається у лаконічній формі, щоб не дублювати правильну відповідь і водночас підсилювати розуміння помилки. Такий формат є придатним для самоконтролю, оскільки учень бачить не лише факт помилки, а й мінімальне методичне пояснення, на яке правило або ідею слід звернути увагу.

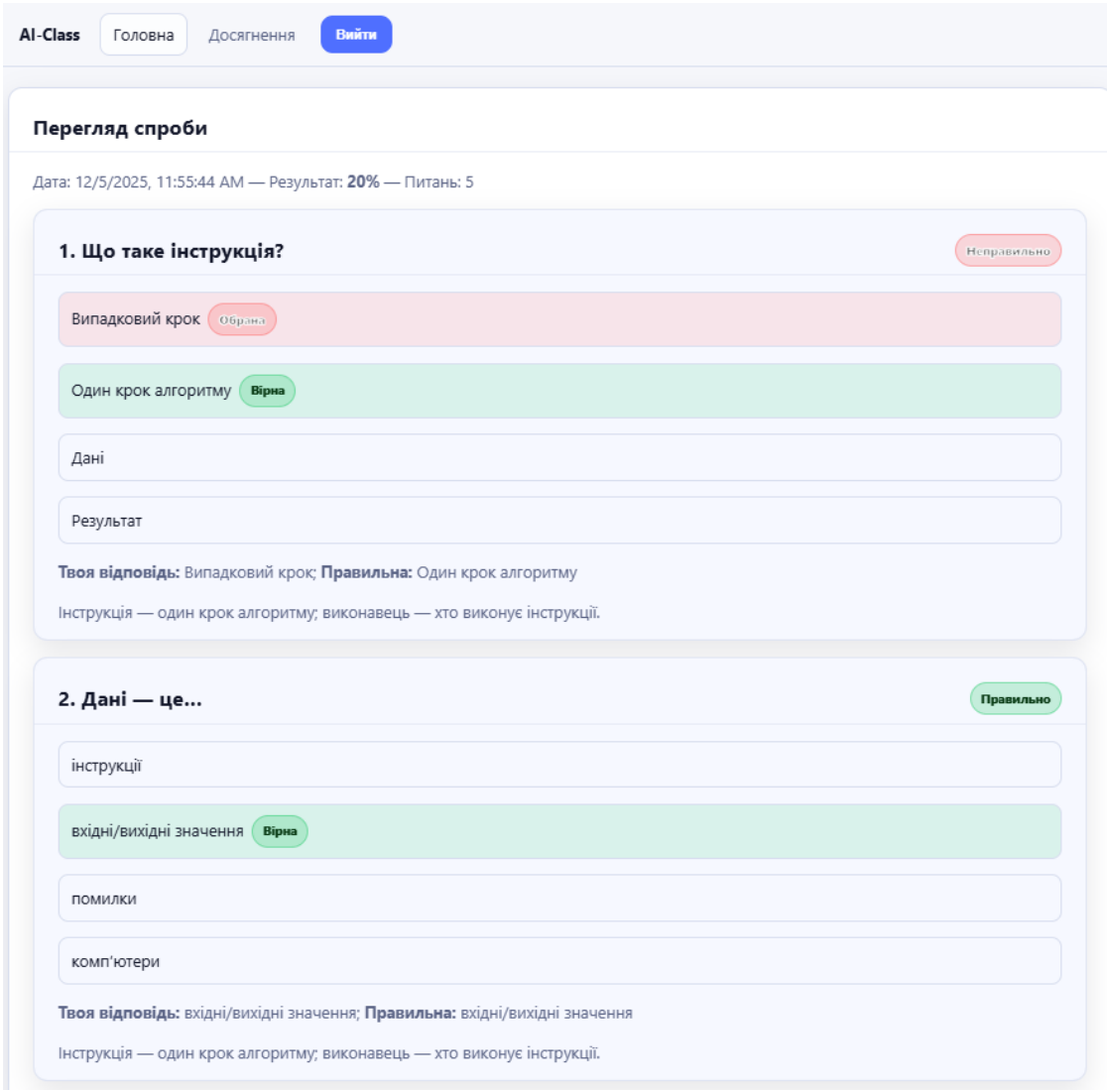


Рисунок 2.17 Перегляд спроби (AttemptView), розбір питань, позначення правильності та пояснення

Окремо передбачено сторінку історії спроб (AttemptHistory), що дає змогу швидко переглядати попередні результати по конкретній темі. У навчальному

плані це підтримує рефлексію: учень може бачити динаміку змін, порівнювати спроби та усвідомлювати, як саме повторення і підсилення слабких місць вплинули на прогрес.

Таким чином, індивідуалізація в MentorAI реалізована як практичний ланцюжок механізмів, безпосередньо пов'язаних із тестуванням: генерація питань ШІ як основний режим, фолбек до локального банку для стабільності, аналіз помилок через концепти для адаптації наступного тесту, збереження спроби зі знімком питань та інтерфейси перегляду спроб і історії. Ця сукупність рішень забезпечує не лише оцінювання знань, а й кероване повторення, спрямоване на усунення конкретних прогалин учня.

2.6. Організація тестування та налагодження програмного засобу

Тестування та налагодження MentorAI виконувалося як безперервний процес упродовж усіх ітерацій розробки, оскільки вебзастосунок містить декілька взаємопов'язаних рівнів – маршрутизацію та рольовий доступ, керування станом у Redux, локальний шар даних на базі IndexedDB (Dexie), навчальний контент (уроки) і логіку тестування, включно з інтеграцією генерації питань за допомогою ШІ. З огляду на те, що поточна версія MentorAI є прототипом без серверної частини, основний акцент тестування було зроблено на перевірці коректності клієнтської логіки, збереження даних у браузері та відтворюваності сценаріїв користувача.

Налагодження застосунку здійснювалося в середовищі Visual Studio Code із використанням можливостей DevTools у браузері та інструментів Vite для live-reload. Після створення або зміни кожного важливого модуля виконувалася перевірка цілісного сценарію: вхід у систему, відкриття відповідного кабінету за роллю, перехід між сторінками, виконання операцій із даними та контроль правильності відображення результатів в інтерфейсі. Окремо перевірялася поведінка програми при перезавантаженні сторінки, що є важливим для SPA-

застосунків, оскільки стан має відновлюватися коректно, а маршрутизація – не руйнуватися.

З точки зору функціональних перевірок першочергово тестувався механізм авторизації та рольового контролю доступу. Перевірка включала такі ситуації: успішний вхід як учень та як вчитель, відображення відповідних пунктів навігації, коректний перехід у розділ, який відповідає ролі, а також блокування/перенаправлення при спробі відкрити «чужий» розділ через пряме введення адреси. Під час налагодження додатково перевірялося, що вихід із системи очищує стан авторизації та повертає користувача на стартову сторінку. Цей блок тестування був критичним, оскільки роль визначає не лише інтерфейс, а й допустимі сценарії роботи з даними.

Наступним напрямом перевірок була робота з локальним сховищем даних. Оскільки дані MentorAI в прототипі зберігаються в IndexedDB, важливо було переконатися, що ініціалізація бази відбувається стабільно, а тестовий каталог предметів, тем, уроків і базовий банк питань коректно «підсаджується» при першому запуску. Для цього використовувалися: перевірки через DevTools (Application → IndexedDB), контроль наповнення таблиць, а також практичні сценарії у застосунку – відкриття списків предметів і тем, відображення лекційного матеріалу та доступність тестування. Якщо під час тестування виникали некоректні стани через попередні дані, застосовувався контрольований «скид» або відновлення локального банку, щоб підтвердити працездатність механізму ініціалізації та узгодженість структури даних.

Окрему увагу було приділено тестуванню навчального сценарію учня: предмети → теми → урок → тест. Перевірялися коректність переходів між сторінками, правильність завантаження уроку за ідентифікатором теми та відображення короткого матеріалу. Далі тестувався блок тестування: відображення необхідної кількості питань, можливість вибору відповіді, блокування кнопки завершення до моменту надання відповідей на всі питання та автоматичний підрахунок результату. Важливим аспектом було підтвердження,

що після завершення тесту результат зберігається як окрема спроба й одразу доступний для перегляду.

Тестування модуля індивідуалізації включало перевірку інтеграції ШІ та резервного сценарію. Для цього перевірялася поведінка у двох режимах: коли генерація доступна і коли вона недоступна. У випадку недоступності ШІ-провайдера MentorAI повинен сформувати тест із локального банку питань, забезпечивши працездатність без зовнішніх сервісів. У випадку доступної генерації перевірялася коректність отримання питань, фільтрація дублікатів та формування тесту з урахуванням слабких концептів. Додатково тестувалося повторне проходження тесту як основний інструмент підсилення слабких місць: після помилок у першій спробі наступне проходження повинно було містити більшу частку питань зі споріднених концептів.

Важливою частиною налагодження стала перевірка сторінок історії та перегляду спроб. На практиці перевірялося, що для кожної спроби збережено повний «знімок» питань, включно з варіантами відповідей і правильним індексом. Це дозволяє переглядати минулі результати навіть тоді, коли питання були згенеровані ШІ та не повторюються. Перевірялися коректність відображення правильності/неправильності, відображення відповіді учня та правильного варіанту, а також змістовність короткого пояснення за концептом без дублювання однакових фраз. Окремо перевірялася робота навігаційних кнопок між сторінками теми, тесту, результату та історії спроб.

У процесі налагодження використовувалися стандартні інструменти діагностики: повідомлення у консолі браузера, логування ключових етапів, перегляд стану Redux за допомогою Redux DevTools, а також перезапуск TypeScript Server у VS Code у випадках, коли редактор зберігав застарілі типи або некоректно індексував імпорти. Значна частина помилок на ранніх етапах була пов’язана з типізацією (особливо у поєднанні з налаштуваннями TypeScript), коректністю експорту/імпорту модулів, а також з узгодженістю версії схеми локальної бази даних. Виправлення таких проблем виконувалося

через уніфікацію експортів, уточнення типів параметрів, приведення до єдиного стилю імпортів та контрольоване оновлення структури Dexie зі збереженням логіки ініціалізації.

Таким чином, тестування MentorAI було організовано як послідовність практичних перевірок реальних сценаріїв користувача з одночасним технічним контролем станів, даних та логіки. Це дозволило підтвердити працездатність ключових модулів: авторизації та ролей, навігації, локального збереження, навчального контенту, тестування з індивідуалізацією, перегляду результатів і аналітики. Результати роботи основних функцій можуть бути проілюстровані скріншотами інтерфейсу застосунку та фрагментами коду, наведеними у відповідних підрозділах і додатках.

2.7. Рекомендації по використанню та впровадженню програмного засобу

Вебзастосунок MentorAI призначений для використання в закладах середньої освіти як допоміжний інструмент для індивідуалізованого навчання з інформатики та математики. Програмний засіб підтримує базові сценарії навчального процесу: рольовий вхід (учень – вчитель), перегляд навчального матеріалу за темами, проходження тестування з автоматичною перевіркою, накопичення результатів та аналіз прогресу. Окремою особливістю MentorAI є можливість формування тестових завдань із застосуванням штучного інтелекту із резервним механізмом підбору питань із локального банку, що забезпечує працездатність прототипу навіть коли зовнішній сервіс генерації тимчасово недоступний.

Поточна версія MentorAI реалізована як локальний клієнтський вебзастосунок і орієнтована на демонстрацію концепції. Дані зберігаються у браузері користувача (IndexedDB), що є зручним тестовим рішенням для прототипування та навчальної апробації, але не є оптимальним для повноцінного

впровадження у навчальному закладі. У разі реального використання з кількома учнями та вчителями рекомендовано перейти до серверної архітектури з централізованим збереженням у базі даних, щоб забезпечити керованість облікових записів, цілісність даних, резервне копіювання та роботу з кількох пристроїв.

Для коректної роботи MentorAI рекомендується використовувати сучасний веббраузер (Google Chrome, Microsoft Edge або Mozilla Firefox). Якщо передбачається використання генерації питань через ШІ, необхідне стабільне підключення до Інтернету та наявність дійсного API-ключа відповідного сервісу. Якщо ж генерація недоступна або не використовується, застосунок залишається працездатним завдяки локальному банку питань, що дає змогу застосовувати MentorAI у навчальному процесі в режимі обмеженого доступу до мережі.

Рекомендований сценарій використання MentorAI в освітньому процесі передбачає поєднання уроку та практики. Учень після входу обирає предмет і тему, ознайомлюється з коротким матеріалом, після чого проходить тест. Після завершення тестування учень отримує результат і може переглянути детальний розбір помилок та історію попередніх спроб. Рекомендується використовувати повторні проходження тестів як механізм закріплення, оскільки саме повтори дозволяють системі підсилювати слабкі місця за концептами й формувати індивідуальну траєкторію тренування. Для вчителя MentorAI доцільно застосовувати як інструмент моніторингу: перегляд успішності, аналіз типових помилок учнів, а також керування доступом учнів до предметів.

Для впровадження MentorAI на новому пристрої у форматі локального запуску необхідно встановити середовище розробки Visual Studio Code, платформу Node.js та менеджер пакетів npm, після чого виконати запуск проєкту стандартними командами середовища React/Vite. Такий спосіб підходить для апробації, демонстрації та методичного використання в межах навчального курсу. Для розгортання у навчальному закладі більш доцільним є варіант публікації застосунку на локальному сервері закладу або на хостингу з

налаштованою серверною частиною та базою даних, що дозволить організувати централізований доступ і контроль даних.

Практичні рекомендації щодо впровадження MentorAI включають поетапний підхід. На першому етапі доцільно використовувати застосунок у форматі пілотного проєкту на окремій групі учнів, оцінити зручність інтерфейсу, якість тестових завдань, логіку адаптації під слабкі місця та загальну методичну придатність. На другому етапі можна розширювати набір тем і предметів, удосконалювати банк завдань, а також інтегрувати централізоване збереження даних. У подальшому MentorAI може бути впроваджений як елемент цифрового освітнього середовища закладу, доповнюючи традиційні форми навчання, сприяючи формуванню стійкої навчальної мотивації та розвитку цифрових компетентностей учнів.

Таким чином, MentorAI рекомендовано використовувати як інструмент підтримки змішаного навчання, самостійної роботи та формувального оцінювання, з акцентом на індивідуалізацію через адаптивне тестування. Для повноцінного масштабування і впровадження в реальних умовах експлуатації доцільно передбачити перехід від локального збереження до серверної архітектури з базою даних, що забезпечить надійність, переносимість даних та розширені можливості керування навчальним процесом.

ВИСНОВКИ

У магістерській роботі розглянуто актуальну для сучасної школи проблему забезпечення індивідуального підходу до учнів у межах вивчення інформатики. З огляду на різний рівень підготовки, темп засвоєння, мотивацію та навчальні стратегії учнів, традиційні підходи до організації навчання часто не дають змоги забезпечити стійке формування предметних компетентностей у всіх здобувачів освіти. Штучний інтелект, зокрема генеративні моделі та інструменти аналізу навчальних даних, відкриває можливості для побудови адаптивних траєкторій навчання, персоналізованого добору завдань і надання розгорнутого зворотного зв'язку, що визначає практичну значущість обраної теми.

У теоретичній частині дослідження проаналізовано поняття, принципи та компоненти індивідуалізованого навчання у шкільному курсі інформатики, а також розглянуто роль штучного інтелекту в сучасних адаптивних освітніх системах. Узагальнено підходи до персоналізації навчального контенту, окреслено педагогічні й психологічні умови ефективного використання ШІ, зокрема з позицій компетентнісного підходу, мотивації й інклюзивності. Проведено огляд і порівняння існуючих ШІ-платформ, що дало змогу сформувати вимоги до розробки програмного рішення для українського освітнього контексту, з урахуванням мовної локалізації, навчальної програми та потреб учителів і учнів.

Практична частина роботи спрямовувалася на реалізацію програмного комплексу MentorAI, що демонструє можливості застосування ШІ для індивідуалізації навчання. У процесі розробки створено вебзастосунок на базі React і TypeScript із компонентною структурою, маршрутизацією та розмежуванням ролей користувачів (учень і вчитель). Реалізовано керування станом за допомогою Redux, що забезпечило передбачуваність логіки авторизації, коректність інтерфейсу після входу та контроль доступу до функцій відповідно до ролей. Навчальний сценарій учня побудовано у логічному ланцюгу

«предмет – тема – урок – тест», що відповідає типовій структурі навчального процесу та дає змогу організувати систематичне опрацювання матеріалу.

Ключовим результатом роботи є впровадження механізму індивідуалізації через адаптивне тестування. У MentorAI реалізовано генерацію тестових завдань за допомогою ШІ з резервним сценарієм добору питань із локального банку для забезпечення стабільності роботи. На основі аналізу попередніх спроб і помилок учня система визначає слабкі концепти та підсилює їх у наступних проходженнях, що наближує цифровий інструмент до логіки індивідуального педагогічного супроводу. Додатково реалізовано збереження історії спроб, перегляд деталізованого розбору відповідей і відстеження прогресу, що формує умови для рефлексії та самоконтролю учнів.

Важливим аспектом практичної реалізації є використання локального збереження даних у браузері як прототипного рішення, що спрощує апробацію та демонстрацію роботи системи без серверної інфраструктури. Водночас встановлено, що для повноцінного впровадження в масштабі навчального закладу доцільним є перехід на централізоване зберігання у базі даних із серверною частиною, що забезпечить цілісність даних, багатокористувацький режим, резервне копіювання та уніфіковане керування обліковими записами.

Тестування і налагодження MentorAI проводилися ітеративно, із перевіркою реальних користувацьких сценаріїв для обох ролей, коректності маршрутизації, авторизації, роботи локального сховища, формування тестів, підрахунку результатів та перегляду історії спроб. Отримані результати підтвердили працездатність ключових модулів і відповідність програмного комплексу поставленим функціональним вимогам.

Таким чином, мети дослідження досягнуто: розроблено теоретико-методичні засади застосування штучного інтелекту для індивідуалізованого навчання у шкільному курсі інформатики та створено програмний комплекс MentorAI, реалізацію якого протестовано в межах пілотної апробації. Практичне значення роботи полягає у можливості використання MentorAI як інструменту

підтримки змішаного навчання, формувального оцінювання та індивідуальної підготовки учнів, а також як основи для подальшого розвитку вітчизняних україномовних ШІ-рішень у сфері освіти.

Перспективними напрямами подальшого розвитку MentorAI є: розгортання серверної частини та бази даних, розширення навчального контенту й банку завдань відповідно до державних стандартів, удосконалення процедур контролю якості згенерованих ШІ-матеріалів, інтеграція з електронними щоденниками та впровадження розширеної аналітики для вчителя. Також доцільним є впровадження методик педагогічного експерименту з кількісним порівнянням результатів навчання учнів із використанням MentorAI та без нього, із застосуванням методів математичної статистики для оцінювання ефекту персоналізації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Биков В. Ю., Буров О. Ю. Цифрове навчальне середовище: нові технології та вимоги до здобувачів знань. *Сучасні інформаційні технології та інноваційні методики навчання в підготовці фахівців, методологія, теорія, досвід, проблеми* : Збірник наукових праць. Вінниця : ТОВ «Друк плюс», 2020. Вип. 55. С.11-22.

2. Васильченко Л., Шацька Н. Дистанційне (онлайн) навчання у закладах зсо: проблеми, виклики, рішення. Education. Innovation. Practice. 2022. Т. 10, № 7. С. 19-24.

3. Гайтан О. М. Порівняльний аналіз можливостей використання інструментарію вебінарорієнтованих платформ Zoom, Google Meet та Microsoft Teams в онлайн-навчання. *Інформаційні технології і засоби навчання.* 2022. Т. 87. №1. С. 33-67.

4. Белан Т., Ющенко В., Овдієнко В. Переваги і недоліки електронного навчання в закладі вищої освіти. Вісник Національного університету «Чернігівський колегіум» імені Т. Г. Шевченка. 2023. Вип. 178(22). С. 97–101. DOI: 10.58407/232216.

5. Чекрій І. І. Особливості навчання з використанням електронних навчальних матеріалів (за матеріалами ЮНЕСКО). Інноваційна педагогіка. 2020. № 3(22). С. 192–195.

6. Drobin A. Classification of Digital Educational Resources as a Means of Specifying Their Practical Purpose. Academic Notes. Series: Pedagogical Sciences. 2022. Vol. 1(201). DOI: 10.36550/2415-7988-2021-1-201-77-81.

7. Гончарук Л. В. Інформаційні системи та технології в освіті. Київ : Професійний розвиток. 2023. 160 с.

8. Дмитрієва Н. Б., Крикляс В. Г., Крикляс К. В., Чернякова Ж. Ю., Пєнов В. В. Методи активного навчання: особливості використання в дистанційній освіті. *Вісник науки та освіти*. 2024. № 6(24). С. 596-610.

9. Друшляк М. Г., Семеног О. М., Грона Н. В., Пономаренко Н. П., Семеніхіна О. В. Типологія інтернет-ресурсів для розвитку інфомедійної грамотності молоді. *Інформаційні технології і засоби навчання*. 2022. Т. 88. №2. С. 1-22.

10. Карабін О. Й., Поморський Д. В. Формування основ веб орієнтованих технологій на уроках інформатики в учнів старших класів закладів загальної середньої освіти. *Інноваційна педагогіка*. Одесса : Гельветика, 2020. Вип. 30. Т. 1. С. 53-57.

11. Хомяк А., Стельмащук Р., Малащук В., Хомяк М. Цифрові технології в освіті: сучасні виклики та перспективи. Математика. Інформаційні технології. Освіта: збірник тез доп. XIV міжнар. наук.-практ. конф. (м. Луцьк, 13-15 червн. 2025 р.). Луцьк, 2025. С. 262-265.

12. Стельмащук Р., Хомяк М. Використання гейміфікації та ігрових програм для розвитку компетентностей учнів базової школи в процесі навчання інформатики. Математика. Інформаційні технології. Освіта: збірник тез доп. XIV міжнар. наук.-практ. конф. (м. Луцьк, 13-15 червн. 2025 р.). Луцьк, 2025. С. 251-254.

13. Малащук В., Хомяк М. Застосування штучного інтелекту для організації індивідуалізованого навчання учнів у шкільному курсі інформатики. Математика. Інформаційні технології. Освіта: збірник тез доп. XIV міжнар. наук.-практ. конф. (м. Луцьк, 13-15 червн. 2025 р.). Луцьк, 2025. С. 213-215.

14. Коваль О., Умрик М. Практичний досвід впровадження авторської онлайн платформи в освітній процес закладу загальної середньої освіти. *Фізико-математична освіта*. 2025. Том 40, №1. С.34-41

15. Крупко С. Електронні освітні ресурси: реалії сучасного освітнього середовища. Problems of Education. 2022. № 2(97). С. 226–238. DOI: 10.52256/2710-3986.2-97.2022.13.

16. Мар'єнко М. В., Шишкіна М. П., Коновал О. А. Методологічні засади формування хмаро орієнтованих систем відкритої науки у закладах вищої

педагогічної освіти. *Інформаційні технології і засоби навчання.* 2022. Том 89 № 3. С. 209-232.

17. Морзе Н., Нанаєва Т., Пасічник О. Стан та перспективи навчання інформатики в закладах загальної середньої освіти України. *Інформаційні технології і засоби навчання*. 2022. Том 92. №6. С. 1-20

18. Мякшин А. С. Роль інформаційно-комунікаційних технологій у сучасній освіті. Матеріали VIII міжнародної науково-практичної інтернет-конференції (м. Київ, 26 січн. 2022 р.). Київ, 2022. С. 67-69.

19. Орлова А. А. Цифрові освітні технології: переваги, недоліки, перспективи впровадження. *Інновації та інтеграція цифрових трендів освітянського простору в економіку знань*. 2023. С.160-164 URL: https://doi.org/10.36059/978-966-397-332-6-46

20. Платонова О. Використання інтернет додатків та платформ для організації освітнього процесу у закладах вищої освіти. *Вісник Національного університету «Чернігівський колегіум» імені Т. Г. Шевченка*. 2024. № 29-30 (185-186). С.125-130

21. Стечишин І. Системи управління навчанням: сучасні тенденції та перспективи розвитку. *Вісник науки та освіти*. 2025. № 12(30). С. 1203-1214.

22. Zebua W., Gulo S., Gulo S., Bawamenewi A., Gulo M. K. Developing Interactive Learning with PowerPoint via the Canva Application. Journal of International Inspire Education Technology. 2024. Т. 3(1). С. 132–138. DOI: 10.55849/jiiet.v3i1.654.

ДОДАТКИ

Додаток А

Технічне завдання

1. Вступ

Технічне завдання описує практичну частину магістерської роботи – вебзастосунок MentorAI, призначений для індивідуалізації навчання учнів у шкільному курсі інформатики з використанням технологій штучного інтелекту. Застосунок орієнтований на підтримку очного, дистанційного та змішаного форматів навчання, надання персоналізованого контенту, проведення тестування з адаптацією під рівень учня, а також забезпечення аналітики для вчителя.

2. Підстави для розробки

Розробка виконана відповідно до індивідуального завдання магістерської роботи та є практичним продовженням досліджень у сфері адаптивного навчання й застосування ШІ в освіті. Необхідність створення MentorAI обумовлена потребою у доступних інструментах, які дозволяють організувати індивідуальну траєкторію навчання, формувати персоналізовані завдання, збирати дані про результати тестування та забезпечувати вчителю зручні засоби контролю й аналізу успішності.

Апробація окремих сценаріїв використання проводилася під час педагогічної та переддипломної практик, що підтвердило актуальність обраного напряму та практичну доцільність реалізації програмного засобу.

3. Призначення розробки

Функціональне призначення MentorAI полягає у створенні вебплатформи, яка забезпечує керування навчальним контентом з інформатики, організацію проходження уроків і тестів та індивідуалізацію навчання на основі результатів учня. Платформа призначена для використання двома основними ролями –

учнем та вчителем. MentorAI забезпечує:

- авторизацію користувачів та розмежування доступу за ролями;
- перегляд навчального каталогу: предмети, теми, уроки;
- проходження тестів з автоматичною перевіркою;
- формування банку питань із використанням ШІ з наявністю запасного сценарію (fallback) на локальний банк питань у разі недоступності ШІ;
- збереження історії спроб, перегляд відповідей та пояснень;
- аналітику результатів для вчителя за темами та учнями;
- механізми мотивації через досягнення, бейджі та рівні.

Експлуатаційне призначення системи – надання єдиного інструменту для організації персоналізованого навчального процесу з інформатики та підвищення ефективності засвоєння матеріалу за рахунок адаптації тестових завдань і накопичення статистики результатів.

4. Вимоги до програмного продукту

4.1. Вимоги до функціональних характеристик

Для MentorAI визначено наступний функціональний набір:

1. Авторизація та ідентифікація користувачів. Система повинна забезпечувати вхід користувача за ім’ям із вибором ролі, збереженням активної сесії та можливістю виходу.

2. Розмежування доступу за ролями. Учень повинен мати доступ лише до навчального сценарію та власних результатів, а вчитель – до аналітики та адміністрування доступів.

3. Каталог навчального контенту. Система повинна відображати предмети, теми та навчальні матеріали з можливістю переходів між рівнями.

4. Перегляд уроку. Учень повинен мати можливість переглянути матеріал уроку у форматі розмітки та перейти до тестування.

5. Тестування з автоматичною перевіркою. Система повинна формувати набір тестових питань, дозволяти вибір відповідей, обчислювати результат та

відображати його користувачу.

6. Індивідуалізація тестів. Під час формування тесту система повинна враховувати слабкі місця учня на основі попередніх помилок та підсилювати відповідні концепти, при цьому у разі недоступності ШІ використовувати локальний банк питань як резервний варіант.

7. Історія спроб і перегляд відповідей. Система повинна зберігати спроби, відображати список спроб, а також детальний перегляд із позначенням правильних та неправильних відповідей і коротким поясненням.

8. Аналітика для вчителя. Вчитель повинен мати можливість переглянути результативність учнів за темами, частоту спроб, середні результати та інші узагальнені показники.

9. Досягнення, бейджі, рівні. Система повинна нараховувати прогрес користувача на основі виконаних дій та відображати отримані досягнення.

10. Налаштування ШІ. Система повинна містити сторінку налаштувань провайдера та ключа доступу до API, з можливістю вимкнути ШІ та працювати лише з локальним банком питань.

Примітка: функціонал «профіль учня» та «магазин кастомізацій» у поточній версії свідомо пропущено.

4.2. Вимоги до надійності

1. Система повинна коректно працювати при багаторазових переходах між сторінками без циклічних перезавантажень.
2. Система повинна обробляти помилки інтеграції з ШІ та автоматично переходити до резервного сценарію формування питань без зупинки роботи.
3. Під час завершення тесту має гарантуватися запис результатів і знімка питань, щоб забезпечити коректний перегляд спроб у майбутньому.
4. Платформа повинна забезпечувати стабільну роботу в типових умовах використання навчального кабінету без критичних збоїв.

4.3. Умови експлуатації

Застосунок повинен запускатися локально або на сервері навчального закладу та бути доступним через сучасний веббраузер. Для використання інтеграції з ШІ потрібне підключення до Інтернету та наявність валідного API-ключа.

4.4. Вимоги до технічних засобів

Для клієнтської частини достатньо персонального комп’ютера або мобільного пристрою з веббраузером Chrome, Firefox, Edge або Safari.

Для запуску середовища розробки та локального розгортання потрібні:

- Node.js та менеджер пакетів npm;
- інструменти фронтенд-розробки (Vite, TypeScript);
- сучасна ОС сімейства Windows або аналогічна з підтримкою веброзробки.

4.5. Вимоги до інформаційної сумісності

1. Підтримка основних сучасних браузерів.
2. Збереження коректності відображення інтерфейсу на різних розмірах екрану.
3. Підтримка локалізації українською мовою в інтерфейсі користувача.

4.6. Вимоги до збереження даних

Поточна реалізація допускає використання локального збереження даних у браузері як тестового варіанту для швидкої апробації функціоналу. У перспективі система повинна бути перенесена на серверне збереження даних у повноцінній базі даних із реалізацією механізмів резервного копіювання, захисту персональних даних та централізованого адміністрування.

5. Вимоги до програмної документації

Документація повинна містити:

- опис архітектури застосунку та структури проєкту;
- опис маршрутизації та ролей доступу;
- опис модулів роботи з контентом, тестуванням і аналітикою;
- пояснення взаємодії з ШІ та резервного сценарію;
- інструкцію користувача для ролей учня і вчителя;
- опис вимог до розгортання та запуску.

6. Техніко-економічні показники

Розробка MentorAI не потребує значних фінансових ресурсів, оскільки базується на безкоштовних інструментах і технологіях веброзробки. Потенційні витрати пов’язані з використанням API сторонніх ШІ-сервісів, які можуть бути платними. Очікувана ефективність полягає у підвищенні якості засвоєння матеріалу з інформатики завдяки персоналізованим тестам та зменшенню навантаження на вчителя через автоматизацію перевірки й аналітики.

7. Стадії та етапи розробки

1. Формування вимог та сценаріїв використання (учень, вчитель).
2. Проєктування структури застосунку, маршрутизації та макетів сторінок.
3. Реалізація базових модулів: авторизація, доступ за ролями, каталог контенту.
4. Реалізація навчального сценарію: уроки та тестування.
5. Інтеграція ШІ для генерації питань та реалізація резервного сценарію з локальним банком.
6. Реалізація історії спроб і перегляду відповідей.
7. Реалізація модулю аналітики для вчителя.
8. Додавання механізмів мотивації: досягнення, рівні, бейджі.

9. Тестування, налагодження, виправлення помилок та покращення інтерфейсу.

8. Необхідні сторінки та структурні елементи сайту MentorAI (у вигляді таблиці)

Назва сторінки / модуля	Призначення та детальний опис
LoginLayout	Публічний шаблон для неавторизованої частини. Містить мінімальну шапку, короткий опис застосунку та область для відображення сторінки входу.
StudentLayout	Базовий шаблон для ролі учня. Містить шапку з навігацією, інформацію про активний профіль, перемикач теми (за наявності), кнопку виходу та область контенту.
TeacherLayout	Базовий шаблон для ролі вчителя. Містить шапку з навігацією вчителя, доступ до списку учнів/аналітики та область контенту.
Login	Сторінка входу з вибором ролі (учень/вчитель) та створенням/вибором локального профілю. Забезпечує повторний вхід із відновленням останнього профілю.
StudentDashboard (StudentHome)	Головна сторінка учня. Відображає список предметів, коротку статистику (mastery, останні бали), графік прогресу та швидкі переходи до тем.
Subjects (Каталог предметів)	Сторінка зі списком доступних предметів/курсів. Дозволяє перейти до сторінки конкретного предмета.
SubjectPage (Сторінка предмета)	Відображає теми вибраного предмета, прогрес по темах, доступ до уроку та тесту для кожної теми.

TopicPage (Сторінка теми)	Центральна сторінка теми з блоком прогресу, кнопками відкриття уроку та запуску тестування, короткими рекомендаціями щодо повторення (за наявності).
LessonView (Урок / лекція)	Відображення навчального матеріалу у форматі Markdown із підтримкою форматування, блоків коду, зображень і формул (KaTeX – за наявності).
TestRunner	Модуль проходження тесту. Підтримує типи завдань: single-choice, multiple, numeric, free-text; забезпечує перевірку відповідей і підрахунок балу.
ResultView (Результат тесту)	Сторінка/екран після завершення тесту. Показує отриманий бал, правильні/неправильні відповіді, пояснення помилок і рекомендації (у т.ч. через ШІ – за наявності).
AttemptsHistory (Історія спроб)	Перегляд попередніх спроб тестування по темі/предмету, динаміки результатів та типових помилок.
TeacherDashboard (TeacherHome)	Головна сторінка вчителя. Відображає коротку аналітику по класу/групі: загальні показники, проблемні теми, активність.
StudentsList (Список учнів)	Таблиця учнів із фільтрами та основними показниками (mastery, середній бал, кількість спроб). Дозволяє відкрити профіль учня.
StudentProfile (Профіль учня)	Детальна аналітика по конкретному учню: прогрес по предметах/темах, типові помилки (errorTags), історія результатів.
Help / FAQ	Довідковий розділ з інструкціями для учня та вчителя, короткими правилами відповідального використання ШІ.

NotFound / Error	Службова сторінка для неіснуючих маршрутів та відображення помилок застосунку.

9. Порядок контролю і прийняття програмного засобу

Контроль якості та прийняття MentorAI здійснюється поетапно відповідно до вимог технічного завдання та індивідуального завдання магістерської роботи. Поточний контроль включає перевірку працездатності модулів авторизації, коректності переходів між сторінками, доступу за ролями, роботи навчального сценарію (предмети, теми, урок, тест), збереження та відображення історії спроб, а також правильності відображення аналітики в кабінеті вчителя.

Фінальне прийняття здійснюється після комплексного тестування: функціональної перевірки, перевірки сценаріїв помилок інтеграції ШІ та працездатності резервного механізму, а також оцінювання зручності інтерфейсу. Програмний засіб вважається прийнятим у разі:

- відповідності реалізованого функціоналу вимогам технічного завдання;
- відсутності критичних помилок, що блокують навчальний сценарій;
- стабільної роботи в режимі багаторазового використання;
- позитивних результатів апробації під час педагогічної та переддипломної практик.

Додаток Б

Інструкція користувача

1. Загальна інформація про програмне забезпечення

Вебзастосунок MentorAI призначений для підтримки індивідуалізованого навчання учнів у шкільному курсі інформатики. Програмний засіб забезпечує доступ до навчального контенту, проходження тестів, аналіз результатів та формування персоналізованих завдань з урахуванням попередніх помилок. Окремі функції можуть використовувати інтеграцію зі штучним інтелектом для генерації тестових питань, а у разі недоступності ШІ система використовує резервний локальний банк питань.

На поточному етапі MentorAI функціонує як вебзастосунок, що може запускатися локально в середовищі розробки. Локальне збереження даних використано як тестовий варіант, який у перспективі має бути замінено на повноцінне серверне збереження у базі даних.

2. Вимоги користувача та технічні умови

Для запуску й коректної роботи MentorAI необхідно:

- операційна система Windows або macOS;
- встановлений Node.js (рекомендовано актуальну LTS-версію);
- менеджер пакетів npm (встановлюється разом із Node.js);
- середовище розробки Visual Studio Code;
- сучасний веббраузер (Google Chrome, Microsoft Edge або аналогічний);
- стабільне підключення до Інтернету у разі використання генерації питань ШІ;
- API-ключ провайдера (за потреби) для доступу до генеративної моделі.

3. Запуск програмного засобу

Для запуску MentorAI необхідно:

- відкрити папку проєкту у Visual Studio Code;
- відкрити термінал у корені проєкту;
- встановити залежності командою npm install;
- запустити застосунок командою npm run dev;
- відкрити у браузері адресу локального сервера, яку відобразить термінал (зазвичай http://localhost:5173).

Після запуску застосунок працює у браузері через вебінтерфейс.

4. Вхід до системи та вибір ролі

Для початку роботи користувач переходить на сторінку входу та вводить ім’я. Далі обирається роль: учень або вчитель. Після входу користувач отримує доступ до функціоналу відповідно до ролі.

У межах демонстраційної версії доступ учня до предметів може визначатися вчителем через інтерфейс керування, тому можливе повідомлення про відсутність підключених предметів, якщо доступ не надано.

5. Функціональні можливості вчителя

Після входу з роллю вчителя доступні такі можливості:

- перегляд аналітики успішності учнів;
- аналіз результатів за темами, спробами та загальними показниками;
- перегляд списку учнів та керування доступом до предметів і тем (у межах реалізованого сценарію);
- контроль активності через історію тестових спроб та узагальнення статистики.

6. Функціональні можливості учня

Після входу з роллю учня здійснюється навчальний сценарій:

- перегляд доступних предметів;
- перехід до обраного предмета та перегляд списку тем;
- відкриття теми та перегляд короткого навчального матеріалу уроку;
- перехід до тестування;
- отримання результату та рекомендацій після завершення тесту;
- перегляд історії спроб для теми;
- детальний перегляд відповідей у конкретній спробі з позначенням правильних і неправильних відповідей та поясненнями.

7. Робота з тестуванням та індивідуалізацією

Тестування у MentorAI виконує дві функції: перевірку засвоєння матеріалу та підсилення слабких місць учня. Після завершення тесту система:

- автоматично перевіряє відповіді;
- обчислює результат та зберігає спробу;
- оновлює показники прогресу за темою;
- у разі використання ШІ може формувати нові питання відповідно до слабких концептів;
- якщо ШІ недоступний або вимкнений, використовується локальна база питань.

8. Налаштування ШІ

У застосунку передбачено сторінку налаштувань ШІ, де можна:

- вказати провайдера;
- додати API-ключ;
- увімкнути або вимкнути генерацію питань;
- перевірити роботу ШІ через запуск тесту.

Якщо ключ не задано або генерація вимкнена, застосунок працює у режимі локального банку питань.

9. Правила безпечної роботи

Користувачам рекомендовано:

- не розголошувати API-ключ стороннім особам;
- не зберігати ключ у публічних репозиторіях;
- використовувати лише перевірені пристрої для роботи із застосунком;
- завершувати сесію виходом із системи після завершення роботи;
- у разі перенесення системи на серверне середовище забезпечити налаштування прав доступу та захист персональних даних відповідно до вимог безпеки.

Додаток В

Анкета для оцінювання вебзастосунку MentorAI (для вчителів інформатики)

Шановний(а) вчителю,

Прошу Вас оцінити вебзастосунок MentorAI, розроблений для підтримки індивідуалізованого навчання учнів з інформатики, та відповісти на запитання. Ваші відповіді допоможуть удосконалити функціонал платформи, покращити методику використання в освітньому процесі та підвищити якість навчальних матеріалів.

1. Ваш досвід викладання інформатики:

□ менше 3 років

□ 3–10 років

□ більше 10 років

2. Як Ви оцінюєте зручність використання MentorAI (інтерфейс, навігація, зрозумілість кнопок і сторінок)?

□ Дуже зручно

□ Зручно

□ Частково зручно

□ Незручно

3. Як Ви оцінюєте логіку навчального сценарію учня (предмети – теми – урок – тест)?

□ Дуже логічно

□ Логічно

□ Частково логічно

□ Нелогічно

4. Наскільки зрозумілим є навчальний матеріал на сторінці уроку для учнів різного рівня підготовки?

□ Дуже зрозумілий

□ Зрозумілий

 Частково зрозумілий

 Незрозумілий

5. Чи сприяє MentorAI підвищенню мотивації учнів до вивчення інформатики?

 Так, значно підвищує

 Частково підвищує

 Не впливає

 Знижує мотивацію

6. Які елементи MentorAI, на Вашу думку, є найбільш ефективними? (можна обрати кілька варіантів)

 Структура “предмети – теми – урок”

 Тестування після уроку

 Перегляд помилок та пояснення після тесту

 Історія спроб і можливість повторного проходження

 Персоналізація питань за результатами попередніх спроб

 Аналітика для вчителя

 Інше (вкажіть): _____

7. Як Ви оцінюєте корисність розбору відповідей після тесту (правильно/неправильно, пояснення, виділення правильної відповіді)?

 Дуже корисно

 Корисно

 Частково корисно

 Некорисно

8. Чи є, на Вашу думку, доцільним використання тестів із можливістю генерації питань за темою (ШІ) або використання локального банку питань?

 Так, доцільно

 Скоріше так

 Скоріше ні

 Ні

9. Чи достатньо, на Вашу думку, інформації для вчителя в розділі аналітики (результати учнів, спроби, теми)?

□ Цілком достатньо

□ Загалом достатньо

□ Частково недостатньо

□ Недостатньо

10. Чи стикалися Ви з труднощами під час роботи з MentorAI?

□ Ні

□ Так (вкажіть): _____

11. Які функції або можливості Ви б додали у MentorAI в майбутньому?

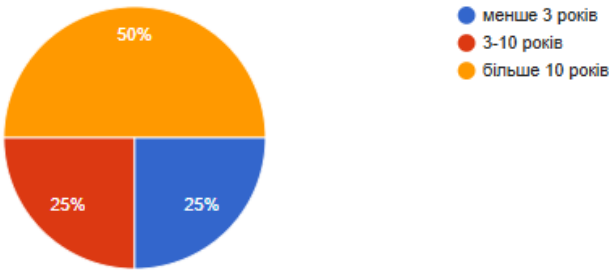
Додаток Г

Результати опитування вчителів щодо використання вебзастосунку MentorAI на уроках інформатики

Ваш досвід викладання інформатики:

Копіювати діаграму

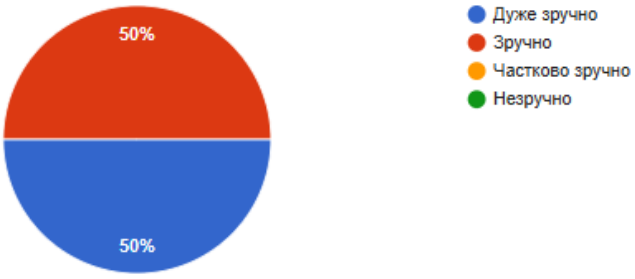
4 відповіді



Як Ви оцінюєте зручність використання MentorAI (інтерфейс, навігація, зрозумілість кнопок і сторінок)?

Копіювати діаграму

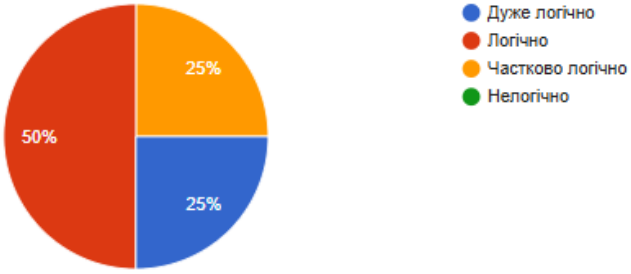
4 відповіді



Як Ви оцінюєте логіку навчального сценарію учня (предмети – теми – урок – тест)?

Копіювати діаграму

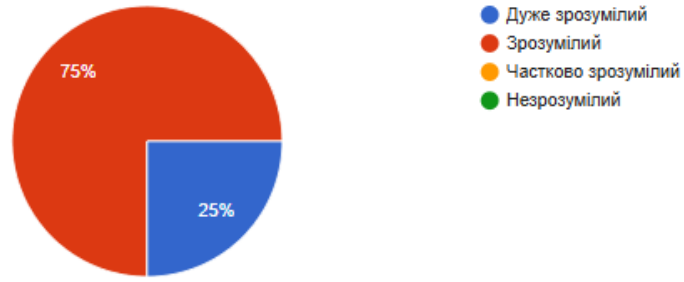
4 відповіді



Наскільки зрозумілим є навчальний матеріал на сторінці уроку для учнів різного рівня підготовки?

Копіювати діаграму

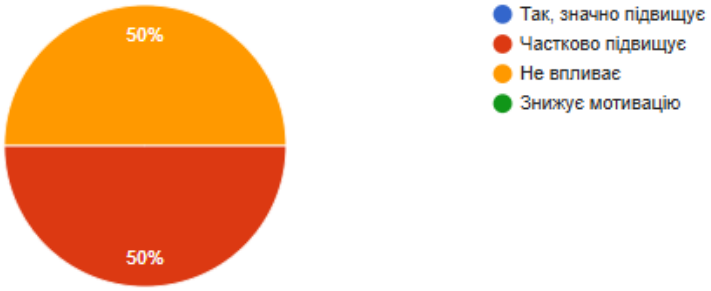
4 відповіді



Чи сприяє MentorAI підвищенню мотивації учнів до вивчення інформатики?

Копіювати діаграму

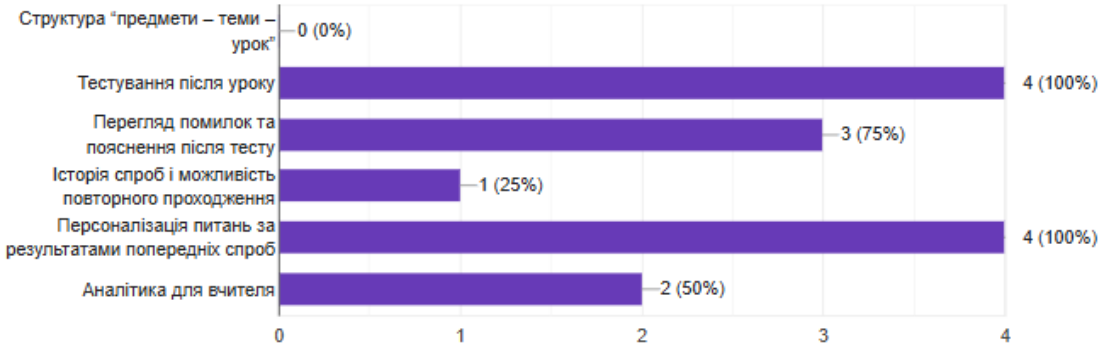
4 відповіді



Які елементи MentorAI, на Вашу думку, є найбільш ефективними? (можна обрати кілька варіантів)

Копіювати діаграму

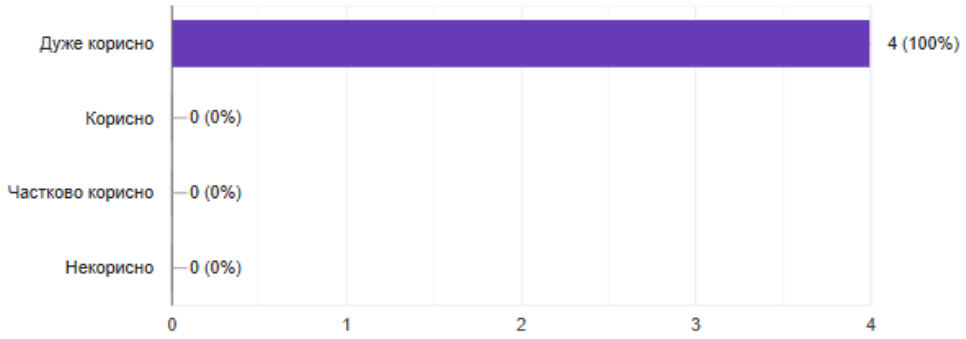
4 відповіді



Як Ви оцінюєте корисність розбору відповідей після тесту (правильно/неправильно, пояснення, виділення правильної відповіді)?

Копіювати діаграму

4 відповіді



Чи є, на Вашу думку, доцільним використання тестів із можливістю генерації питань за темою (ШІ) або використання локального банку питань?

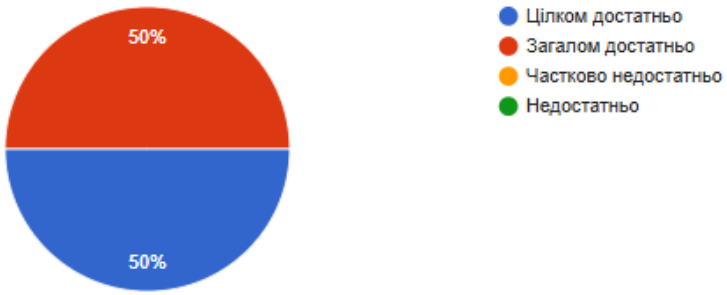
Копіювати діаграму

4 відповіді

Чи достатньо, на Вашу думку, інформації для вчителя в розділі аналітики (результати учнів, спроби, теми)?

Копіювати діаграму

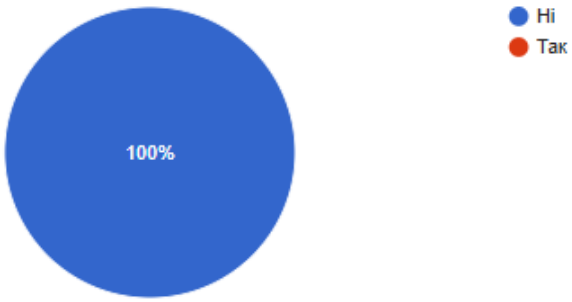
4 відповіді



Копіювати діаграму

Чи стикалися Ви з труднощами під час роботи з MentorAI?

4 відповіді



Які функції або можливості Ви б додали у MentorAI в майбутньому?

Додати можливість вчителю редагувати та доповнювати банк питань і створювати власні тести для тем.

2 відповіді

Реалізувати персональний профіль учня з чіткою статистикою прогресу по темах і рекомендаціями, що повторити.

1 відповідь

Додати режим "домашнє завдання": вчитель задає тему/тест із дедлайном, а система зберігає результати й показує виконання.

1 відповідь