МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ПОЛІСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
Факультет інформаційних технологій,
обліку та фінансів
Кафедра комп'ютерних технологій
і моделювання систем

Кваліфікаційна робота
на правах рукопису

Савченко Олександр Романович
(прізвище, ім'я, по батькові здобувача освіти)

УДК 004.8:37.018.43

КВАЛІФІКАЦІЙНА РОБОТА

Гейміфікована освітня платформа з генерацією завдань за допомогою AI
(тема роботи)

122 «Комп'ютерні науки»
(шифр і назва спеціальності)

Подається на здобуття освітнього ступеня бакалавр

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

_____
(підпис, ініціали та прізвище здобувача вищої освіти)

Керівник роботи
Ковальчук Майя Олегівна
(прізвище, ім'я, по батькові)
к.пед.н., доцент
(науковий ступінь, вчене звання)

Житомир – 2026

Висновок кафедри комп’ютерних технологій і моделювання систем
за результатами попереднього захисту: допущений

Протокол засідання кафедри комп’ютерних технологій і моделювання систем
№ 20 від «05» червня 2026 р.

Завідувач кафедри комп’ютерних технологій і моделювання систем
к. пед. н., доцент _____ Ковальчук М.О.
(науковий ступінь, вчене звання) (підпис) (прізвище, ім’я, по батькові)
«05» червня 2026 р.

Результати захисту кваліфікаційної роботи

Здобувач вищої освіти _____ захистив
(прізвище ,ім’я, по батькові)
кваліфікаційну роботу з оцінкою:

сума балів за 100-бальною шкалою _____
за шкалою ECTS _____
за національною шкалою _____

Секретар ЕК
лаборант кафедри _____ Корольчук В.В.
(науковий ступінь, вчене звання) (підпис) (прізвище, ім’я, по батькові)

АНОТАЦІЯ

Савченко О. Р. Гейміфікована освітня платформа з генерацією завдань за допомогою AI - Кваліфікаційна робота бакалавра за спеціальністю 122 «Комп'ютерні науки». Поліський національний університет, Житомир, 2026.

Кваліфікаційна робота присвячена розробці та реалізації гейміфікованої освітньої платформи Stride, яка використовує методи штучного інтелекту для адаптивного навчання та динамічної генерації навчальних завдань.

Метою роботи є проєктування і реалізація веб-платформи, що динамічно генерує навчальні завдання за допомогою великих мовних моделей (Google Gemini API) та калібрує їх складність відповідно до індивідуального прогресу кожного студента на базі ML.NET. Платформа орієнтована на україномовну аудиторію та охоплює предмети: математика, українська мова, історія України, англійська мова.

У роботі проведено порівняльний аналіз існуючих EdTech-платформ (Duolingo, Khan Academy, Moodle, Coursera), обґрунтовано потребу в комплексному україномовному рішенні. Спроєктовано шарову архітектуру на базі ASP.NET Core 10 з двома API-сервісами (основний на порту 5000 та адаптивний на порту 5010), Angular 20 фронтендом, PostgreSQL та MongoDB як сховищами даних. Реалізовано рушій адаптації складності, систему гейміфікації (XP, стріки, досягнення, таблиця лідерів у реальному часі через SignalR) та прогресивний веб-додаток із підтримкою офлайн-режиму.

Практична цінність роботи полягає в готовому MVP-рішенні, придатному для розгортання у закладах освіти, з відкритою архітектурою для подальшого масштабування та інтеграції додаткових предметів і провайдерів ШІ.

Обсяг роботи: основний текст - 33 сторінки, 6 рисунків, 10 таблиць, список використаних джерел - 53 позиції, 11 додатків.

Ключові слова: адаптивне навчання, гейміфікація, штучний інтелект, освітня платформа, генерація завдань.

ABSTRACT

Savchenko O. R. Development of a Gamified Educational Platform with Adaptive Learning Based on Artificial Intelligence. - Bachelor's qualification thesis, specialty 122 «Computer Science». Polissia National University, Zhytomyr, 2026.

The qualification thesis is devoted to the design and implementation of the gamified educational platform Stride, which employs artificial intelligence methods for adaptive learning and dynamic generation of educational tasks.

The objective of the thesis is to design and implement a web platform that dynamically generates learning tasks using large language models (Google Gemini API) and calibrates task difficulty according to each student's individual progress using ML.NET. The platform targets a Ukrainian-speaking audience and covers the following subjects: Mathematics, Ukrainian Language, History of Ukraine, and English.

The thesis includes a comparative analysis of existing EdTech platforms (Duolingo, Khan Academy, Moodle, Coursera) and substantiates the need for a comprehensive Ukrainian-language solution. A layered architecture is designed using ASP.NET Core 10 with two API services (main on port 5000 and adaptive on port 5010), an Angular 20 frontend, PostgreSQL and MongoDB as data stores. The adaptive difficulty engine, gamification system (XP, streaks, achievements, real-time leaderboard via SignalR), and a progressive web application with offline support are implemented.

The practical value of the work lies in a ready-to-deploy MVP solution suitable for educational institutions, with an open architecture for further scaling and integration of additional subjects and AI providers.

Thesis scope: main text - 33 pages, 6 figures, 10 tables, reference list - 53 entries, 12 appendices.

Keywords: adaptive learning, gamification, artificial intelligence, educational platform, task generation.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ЗВО - заклад вищої освіти

ШІ - штучний інтелект

СУБД - система управління базами даних

API - Application Programming Interface (програмний інтерфейс застосунку)

CRUD - Create, Read, Update, Delete (основні операції зі збереженими даними)

CSS - Cascading Style Sheets (каскадні таблиці стилів)

EdTech - Educational Technology (освітні технології)

EF Core - Entity Framework Core (об'єктно-реляційний перетворювач для .NET)

HTML - HyperText Markup Language (мова розмітки гіпертексту)

HTTP - HyperText Transfer Protocol (протокол передачі гіпертексту)

IRT - Item Response Theory (теорія відповіді на завдання)

JWT - JSON Web Token (токен автентифікації)

LLM - Large Language Model (велика мовна модель)

ML - Machine Learning (машинне навчання)

MVP - Minimum Viable Product (мінімально життєздатний продукт)

ORM - Object-Relational Mapping (об'єктно-реляційне відображення)

PWA - Progressive Web Application (прогресивний веб-застосунок)

REST - Representational State Transfer (архітектурний стиль взаємодії)

SPA - Single Page Application (односторінковий застосунок)

SQL - Structured Query Language (мова структурованих запитів)

UI - User Interface (інтерфейс користувача)

UML - Unified Modeling Language (уніфікована мова моделювання)

XP - Experience Points (бали досвіду)

ВСТУП

Актуальність теми

Перехід української системи освіти на дистанційний формат, що відбувся під час карантину 2020–2021 років і продовжився після початку повномасштабної війни 2022 року, оголив низку обмежень навчальних платформ, які масово використовуються у школах та університетах. Перш за все це статичність контенту: завдання генеруються заздалегідь та однакові для всіх студентів, тоді як учні різного рівня підготовки потребують різної складності. Друге обмеження - відсутність зворотного зв'язку у режимі реального часу, що знижує мотивацію та утруднює виявлення проблемних місць у навчанні.

Серед популярних EdTech-платформ Duolingo спеціалізується на мовних курсах, Khan Academy орієнтована переважно на математичну освіту, Moodle є системою управління навчанням без вбудованої адаптації складності, а Coursera пропонує лекційні курси без гейміфікованої механіки. На українському ринку відсутнє рішення, яке б одночасно підтримувало кілька предметних областей шкільної програми державною мовою, мало вбудовану адаптацію складності та гейміфікаційні елементи.

Кваліфікаційна робота присвячена розробці платформи Stride, в якій ці три аспекти інтегровані: генерація завдань через велику мовну модель Google Gemini, оцінювання складності засобами ML.NET та механіка XP, стріків і таблиці лідерів. Платформа реалізована як веб-додаток з підтримкою офлайн-режиму і спроєктована як MVP, придатний до запуску у закладах освіти.

Мета і завдання роботи

Метою кваліфікаційної роботи є розробка та реалізація гейміфікованої освітньої платформи Stride з адаптивною системою навчання, що динамічно генерує завдання за допомогою штучного інтелекту та калібрує їх складність відповідно до індивідуального прогресу студента.

Для досягнення поставленої мети визначено такі завдання:

1) проаналізувати існуючі EdTech-рішення та виявити їх переваги й недоліки;

2) дослідити теоретичні засади адаптивного навчання та методи гейміфікації в освіті;

3) спроєктувати архітектуру платформи з виокремленим адаптивним модулем;

4) реалізувати систему генерації навчальних завдань через Google Gemini API;

5) розробити рушій адаптації складності на базі ML.NET із щотижневим перенавчанням моделі;

6) реалізувати систему гейміфікації: XP, стріки, досягнення, таблицю лідерів у реальному часі;

7) розробити інтерфейс для ролей студента, викладача та адміністратора;

8) провести тестування основних функціональних сценаріїв платформи.

Об'єкт та предмет дослідження

Об'єктом дослідження є веб-платформа для дистанційного навчання з елементами гейміфікації та адаптивним управлінням складністю навчального контенту.

Предметом дослідження є методи адаптивного навчання, алгоритми генерації навчальних завдань на основі великих мовних моделей та механізми гейміфікації для підвищення мотивації в освітніх системах.

Методи дослідження

У процесі виконання кваліфікаційної роботи використано такі методи дослідження:

1) порівняльний аналіз - для дослідження та оцінювання існуючих EdTech-платформ;

2) методи машинного навчання (ML.NET) - для побудови моделі передбачення складності завдань;

3) генерація природною мовою (LLM, Google Gemini API) - для автоматичної генерації навчального контенту;

4) об'єктно-орієнтоване та компонентне проєктування - для розробки архітектури системи;

5) метод прототипування (MVP-підхід) - для ітеративної розробки та валідації рішень;

6) інтеграційне тестування та E2E-тестування через Playwright - для перевірки коректності роботи платформи.

Практичне значення роботи

Розроблена платформа може використовуватися як інструмент допомоги вчителю в підготовці варіативних завдань для класу: адміністратор створює топік, після чого система генерує пул завдань визначеної складності. Студент отримує завдання, які підбираються під його поточний рівень, з негайним зворотним зв'язком і нарахуванням балів. Викладачу доступна панель аналітики класу з показниками точності, часу проходження та слабких місць.

Архітектура реалізована за принципом розділення обов'язків. Шар фабрики постачальників ШІ дозволяє підмінити Gemini на іншого постачальника великих мовних моделей, наприклад OpenAI або Anthropic, без зміни бізнес-логіки. Запуск у середовищі закладу освіти виконується одним викликом docker-compose up, що знижує поріг входження для адміністратора.

Публікації автора

Окремі результати роботи опубліковані у матеріалах двох міжнародних науково-практичних конференцій:

1. Савченко О. Р. Застосування великих мовних моделей для автоматизованої генерації навчальних завдань у цифрових освітніх системах // Science, technology and global challenges. Матеріали X Міжнародної науково-практичної конференції. CPN Publishing Group. Токіо, Японія. 2026. С. 167–173.

2. Савченко О. Р. Гейміфікація як засіб підвищення мотивації та залученості студентів у цифровому освітньому середовищі // European science

and innovation congress. Матеріали VII Міжнародної науково-практичної конференції. Barca Academy Publishing. Барселона, Іспанія. 2026. С. 171–177.

Структура та обсяг роботи

Кваліфікаційна робота складається із вступу, трьох розділів, загальних висновків, списку використаних джерел та одинадцяти додатків. Загальний обсяг основного тексту роботи становить 33 сторінки. Робота містить 6 рисунків, 10 таблиць. Список використаних джерел налічує 53 позиції. Загальний обсяг роботи з додатками - 61 сторінки.

У першому розділі «Аналіз предметної області» проведено огляд та порівняльний аналіз сучасних EdTech-платформ, розглянуто теоретичні засади адаптивного навчання та гейміфікації в освіті, досліджено можливості застосування великих мовних моделей для генерації навчального контенту, визначено функціональні та нефункціональні вимоги до платформи Stride.

У другому розділі «Проєктування системи» описано загальну архітектуру платформи, модель даних PostgreSQL та MongoDB, алгоритм адаптивного навчання та пайплайн генерації завдань через ШІ, архітектуру Angular-фронтенду та механізми безпеки й автентифікації.

У третьому розділі «Реалізація та тестування» представлено опис реалізованого інтерфейсу платформи, інструкцію з інсталяції та налаштування, керівництво користувача для різних ролей, тестові сценарії та аналіз отриманих результатів.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд ринку EdTech-платформ

Ринок освітніх технологій є одним з найбільш динамічно зростаючих сегментів глобальної цифрової економіки. За даними аналітичних агентств, обсяг світового ринку EdTech у 2024 році перевищив 280 млрд дол. США, а прогнозований щорічний темп зростання до 2030 року складає понад 13%. Пандемія COVID-19 стала потужним каталізатором, що прискорив перехід освітніх процесів у цифровий простір і сформував стійкий попит на якісні дистанційні платформи навчання.

Для виявлення нішевих можливостей та обґрунтування потреби у розробці платформи Stride було проведено порівняльний аналіз п'яти провідних EdTech-рішень: Duolingo, Khan Academy, Moodle, Coursera та Udemy. Результати порівняння наведено в таблиці 1.1.

Таблиця 1.1 - Порівняльний аналіз EdTech-платформ

Платформа	**Гейміфікація**	**Адаптивність**	**Генерація ШІ**	**Мова UI**	**Відкритість**
Duolingo	Висока	Часткова	Ні	Багатомовна	Закрита
Khan Academy	Базова	Часткова	Так*	Ан./обмежено	Відкрита
Moodle	Плагіни	Відсутня	Ні	Багатомовна	Open-source
Coursera	Відсутня	Мінімальна	Ні	Ан./обмежено	Закрита
Udemy	Відсутня	Відсутня	Ні	Обмежена	Закрита
Stride (MVP)	Висока	Повна	Так	Українська	Відкрита*

Duolingo є лідером у сфері гейміфікованого мобільного навчання мов, однак обмежений вузькою предметною областю, переважно вивчення

іноземних мов, і не підтримує повноцінну адаптацію складності за всіма параметрами успішності студента.

Khan Academy пропонує широкий безкоштовний навчальний контент та елементи адаптивності, проте система є переважно статичною - завдання наперед складено людьми, а AI Tutor (Khanmigo) доступний лише платним підписникам і не генерує нових завдань динамічно.

Moodle є потужним open-source LMS, але орієнтований переважно на управління навчальним процесом, а не на адаптивне навчання. Система гейміфікації реалізується лише через сторонні плагіни і не є вбудованою. Відсутність сучасного UI/UX знижує залученість учасників, особливо молодших вікових груп.

Coursera та Udemy є платформами масових відкритих онлайн-курсів. Вони пропонують відео лекції та тести, але контент є повністю статичним, без будь-якої адаптації до рівня студента. Ці платформи більше нагадують відеотеку, ніж систему активного навчання.

Аналіз показав, що жодна з розглянутих платформ не поєднує в собі: україномовний інтерфейс, динамічну генерацію завдань засобами LLM, повноцінне адаптивне навчання на основі машинного навчання та комплексну вбудовану систему гейміфікації.

Якщо розглянути ці платформи разом, простежується чітка закономірність: кожна з них сильна в одному-двох напрямах, але жодна не покриває всі чотири потреби одночасно. Системи з найкращою гейміфікацією (Duolingo) обмежені вузькою предметною областю; платформи з найбільшим обсягом контенту (Khan Academy, Coursera) переважно статичні й не підлаштовують завдання під конкретного учня; гнучкі системи керування навчанням (Moodle) майже не містять ні автоматичної генерації завдань, ні мотиваційних механік. Окремо слід наголосити, що жодне з розглянутих рішень не орієнтоване на українську шкільну програму та україномовний контент. Саме це поєднання незакритих потреб - україномовність,

автоматична генерація завдань, адаптація складності та гейміфікація - і визначає прогалину, яку покликана дослідити ця робота.

1.2 Концепція адаптивного навчання

Адаптивне навчання - це педагогічний підхід, що передбачає автоматичне налаштування темпу, складності та типу навчальних матеріалів відповідно до індивідуальних характеристик кожного студента. Концепція бере початок з досліджень Б. Блума, який у 1984 році показав, що індивідуалізоване навчання підвищує успішність в середньому на два стандартних відхилення порівняно зі стандартним груповим навчанням - ефект, відомий як «проблема двох сигм».

Сучасні системи адаптивного навчання спираються на декілька математичних моделей. Найпоширенішою є Item Response Theory (IRT) - теорія відповіді на завдання, яка описує ймовірність правильної відповіді студента з певним рівнем здібностей (θ) на завдання з певним рівнем складності (b), дискримінаційним параметром (a) та псевдовипадковістю (c):

$$P(\theta) = c + (1 - c) / (1 + e^{(-a(\theta - b))})$$

1.3 Гейміфікація в освіті

Гейміфікація - це застосування ігрових елементів та механік у неігровому контексті з метою підвищення мотивації та залученості учасників. Термін набув широкого поширення після 2010 року завдяки роботам Дж. МакГонігал та К. Вербаха [15], які обґрунтували позитивний вплив ігрових механік на продуктивність і мотивацію у різних сферах.

Теоретичним підґрунтям застосування гейміфікації в освіті слугує теорія самовизначення [53], яка виокремлює три базові психологічні потреби: компетентність, автономність та приналежність. Добре спроєктована гейміфікація задовольняє всі три: очки досвіду та рівні підкреслюють зростання компетентності; вибір теми і темпу навчання забезпечує автономність; таблиця лідерів і командні досягнення формують почуття приналежності до спільноти.

У платформі Stride реалізовано такі гейміфікаційні механіки:

1) очки досвіду (XP) - нараховуються після кожної успішної спроби, кількість залежить від складності завдання та швидкості виконання;

2) рівні - визначаються сумою накопичених XP, кожен новий рівень відкриває нові теми та досягнення;

3) стріки - підраховують кількість послідовних днів активності; втрата стріку є потужним мотиватором для регулярних занять;

4) досягнення - видаються за виконання визначених умов, наприклад «100 правильних відповідей» або «Тиждень без пропусків»;

5) таблиця лідерів - відображає рейтинг студентів у реальному часі через SignalR, стимулює здорову конкуренцію.

Дослідження Хамарі та ін. [14] показало, що гейміфікація демонструє статистично значимий позитивний ефект на залученість, мотивацію та навчальні результати, особливо у поєднанні з соціальними механіками: таблицями лідерів і командними викликами. Водночас автори застерігають від надмірної гейміфікації, тобто надто складних систем балів, які відволікають від навчального контенту.

1.4 Застосування великих мовних моделей для генерації навчального контенту

Великі мовні моделі (LLM) - це нейронні мережі, навчені на масивах текстових даних, здатні генерувати зв'язний і контекстуально релевантний текст. Революційний прорив у цій галузі відбувся з виходом моделі GPT-3 [18] та її наступників. Сьогодні провідними комерційними великими мовними моделями є GPT-5 [22], Gemini [21] та Claude компанії Anthropic.

Застосування LLM для генерації навчальних завдань має кілька ключових переваг порівняно з традиційним ручним складанням: необмежений масштаб генерації (модель може створювати мільйони унікальних завдань), миттєва адаптація до вказаного рівня складності та теми, можливість генерації різноманітних типів завдань (вибір відповіді, відкрите питання, задача з

рішенням) та природна мовна варіативність, що унеможливлює заучування шаблонних формулювань.

1.5 Функціональні вимоги до платформи Stride

На основі проведеного аналізу EdTech-ринку та потреб цільової аудиторії визначено функціональні вимоги до платформи, систематизовані за ролями користувачів (таблиця 1.2).

Таблиця 1.2 - Функціональні вимоги до платформи Stride

Роль	Функціональні вимоги
Студент	Реєстрація та авторизація; вибір предметів; перегляд навчального шляху; отримання адаптивних завдань; здача відповідей та миттєвий зворотний зв'язок; перегляд накопиченого XP, стріків та досягнень; участь у таблиці лідерів; редагування профілю.
Викладач	Створення та управління класами; запрошення студентів; призначення тем і завдань для класу; перегляд аналітики успішності кожного студента та класу в цілому; відстеження прогресу навчального шляху.
Адміністратор	CRUD-операції з предметами та темами; управління переліком досягнень і умовами їх отримання; перегляд системних метрик і журналів генерації ШІ; управління користувачами та ролями.

1.6 Нефункціональні вимоги

Нефункціональні вимоги визначають якісні характеристики системи, що впливають на її надійність, продуктивність і зручність використання (таблиця 1.3).

Таблиця 1.3 - Нефункціональні вимоги до платформи Stride

Категорія	Вимога	Спосіб реалізації
Продуктивність	Час відповіді API < 500 мс (без генерації ШІ)	Valkey/Redis кешування, оптимізовані SQL-запити

Категорія	Вимога	Спосіб реалізації
Продуктивність	Кешування шаблонів завдань	TaskPoolService + MongoDB
Масштабованість	Горизонтальне масштабування сервісів	Docker Compose, stateless API
Доступність	PWA із офлайн-підтримкою	Angular Service Worker (ngsw-config.json)
Безпека	JWT + HttpOnly refresh token, RBAC	ASP.NET Core Identity, рольова авторизація
Безпека	Валідація вхідних даних	FluentValidation на рівні API
Пошук	Повнотекстовий пошук предметів і тем	Meilisearch
Зберігання	Аватари та медіафайли	MinIO (S3-сумісне сховище)

Висновки до розділу 1

У першому розділі проведено комплексний аналіз предметної області розробки гейміфікованої освітньої платформи з адаптивним навчанням. За результатами порівняльного аналізу п'яти провідних EdTech-платформ встановлено, що жодна з них не поєднує в собі україномовний інтерфейс, динамічну генерацію завдань засобами великих мовних моделей та повноцінне адаптивне навчання на основі машинного навчання.

Розглянуто теоретичні засади адаптивного навчання, зокрема теорію відповіді на завдання (IRT), яка є математичним підґрунтям для рушія складності платформи Stride. Обґрунтовано доцільність застосування Google Gemini API для генерації навчальних завдань та ML.NET для побудови прогностичної моделі складності. Досліджено психологічні механізми гейміфікації з точки зору теорії самовизначення.

Визначено та систематизовано функціональні вимоги до платформи за ролями (студент, викладач, адміністратор) та нефункціональні вимоги щодо продуктивності, масштабованості, доступності та безпеки. Отримані результати є основою для проєктування архітектури системи, описаної в наступному розділі.

РОЗДІЛ 2. ПРОЄКТУВАННЯ СИСТЕМИ

2.1 Загальна архітектура платформи

Архітектура платформи Stride побудована за шаровим монолітним принципом з чітким розподілом функцій між рівнями: рівень представлення (Angular SPA), рівень бізнес-логіки (ASP.NET Core API) та рівень доступу до даних (EF Core + MongoDB Driver). При цьому адаптивний модуль виокремлено в окремий сервіс (Stride.Adaptive.Api), що спрощує незалежне масштабування ресурсомістких операцій генерації завдань.

Платформа складається з таких основних компонентів (рис. Б.1, додаток Б):

1) Angular 20 SPA (порт 4200) - прогресивний веб-додаток з підтримкою офлайн-режиму через Service Worker.

2) Stride.Api (порт 5000) - основний REST API на ASP.NET Core 10, обслуговує всі запити фронтенду.

3) Stride.Adaptive.Api (порт 5010) - окремий мікросервіс для генерації завдань і адаптивної логіки.

4) PostgreSQL 17 - реляційна СУБД для зберігання структурованих даних (користувачі, спроби, гейміфікація)

5) MongoDB 7 - документна СУБД для шаблонів завдань, екземплярів завдань та журналів генерації ШІ.

6) Valkey (Redis-сумісний) - кешування та зберігання стану гейміфікації.

7) MinIO - S3-сумісне об'єктне сховище для аватарів та медіафайлів.

8) Meilisearch - повнотекстовий пошук по предметах і темах.

9) NGINX - зворотний проксі для маршрутизації трафіку.

Взаємодія між компонентами відбувається таким чином: Angular-клієнт надсилає HTTP-запити до Stride.Api через NGINX. Основний API звертається до PostgreSQL через EF Core та до MongoDB через офіційний драйвер MongoDB.Driver 3.6. Для завдань, що вимагають генерації або адаптивної логіки, Stride.Api проксіює запити до Stride.Adaptive.Api, який у свою чергу

звертається до Google Gemini API. Реальний час забезпечується SignalR-хабами безпосередньо у Stride.Api (/hubs/notifications та /hubs/leaderboard).

Таблиця 2.1 - Проєкти рішення та їх функціональне призначення

Проєкт	Тип	Функціональне призначення
Stride.Api	Web API	Контролери, Middleware, SignalR хаби, точка входу
Stride.Services	Library	Бізнес-логіка: AuthService, LearningService, GamificationService
Stride.Adaptive	Library	AI-логіка: AdaptiveAIService, DifficultyEngine, ModelTrainer
Stride.Adaptive.Api	Web API	HTTP-фасад над Stride.Adaptive, порт 5010
Stride.Core	Library	Доменні сутності PostgreSQL та MongoDB-документи
Stride.DataAccess	Library	EF Core DbContext, MongoDB контекст, репозиторії

2.2 Модель даних

Для зберігання різнорідних даних платформи Stride використовується сховище: PostgreSQL для реляційних структурованих даних та MongoDB для документів зі змінною схемою. Такий підхід дозволяє оптимально використати переваги кожного типу СУБД.

ER-діаграму бази даних із повним переліком зв’язків між сутностями наведено у додатку А (рис. А.1). PostgreSQL (EF Core 10) містить такі основні сутності:

Таблиця 2.2 - Основні сутності PostgreSQL

Сутність	Ключові поля	Призначення
User	Id, Email, PasswordHash, Role, CreatedAt	Автентифікація та базовий профіль

Сутність	Ключові поля	Призначення
StudentProfile	UserId, XP, Level, Streak, LastActiveDate	Гейміфікаційний профіль студента
TeacherProfile	UserId, Department, Bio	Профіль викладача
Subject	Id, Name, Description, IsActive	Навчальні предмети (математика тощо)
Topic	Id, SubjectId, Name, DifficultyLevel, Order	Теми в межах предмету
TaskAttempt	Id, StudentId, TopicId, IsCorrect, TimeTaken, DifficultyLevel	Спроби виконання завдань
StudentPerformance	StudentId, TopicId, SuccessRate, AttemptsCount, AvgTime	Агрегована успішність
Achievement	Id, Name, Condition, XPReward, BadgeImageUrl	Визначення досягнень
StudentAchievement	StudentId, AchievementId, EarnedAt	Отримані досягнення
LeaderboardEntry	Id, StudentId, SubjectId, Score, Rank, UpdatedAt	Рядок таблиці лідерів
Class	Id, TeacherId, Name, JoinCode	Клас викладача
LearningPath	Id, StudentId, SubjectId, CurrentStepIndex	Персональний навчальний шлях

Повну сукупність наведених сутностей, їхні атрибути та зв’язки між ними у вигляді ER-діаграми бази даних PostgreSQL подано у додатку А (рис. А.1).

MongoDB зберігає документи з динамічною структурою, що не придатні для реляційного представлення:

Таблиця 2.3 - Документи MongoDB

Колекція	Ключові поля	Призначення
task_templates	TopicId, DifficultyLevel, QuestionText, Options, CorrectAnswer, Explanation, UseCount	Шаблони завдань від ШІ, повторно використовуються
task_instances	TemplateId, StudentId, PersonalizedText, GeneratedAt	Конкретні екземпляри завдань із можливою персоналізацією
ai_generation_logs	ProviderId, Prompt, Response, LatencyMs, TokensUsed, Success, Timestamp	Журнал всіх звернень до LLM

Зв'язок між реляційними та документними даними здійснюється через ідентифікатори: поле TopicId у TaskTemplateDocument відповідає первинному ключу сутності Topic у PostgreSQL. Такий підхід забезпечує цілісність даних без застосування складних cross-database транзакцій.

2.3 Алгоритм адаптивного навчання

Центральним компонентом адаптивної підсистеми є DifficultyEngine - рушій, що на основі агрегованої статистики студента обчислює оптимальний рівень складності наступного завдання. Алгоритм реалізовано у вигляді детермінованої логіки з можливістю заміни на ML-модель після накопичення достатньої кількості даних.

Алгоритм роботи DifficultyEngine (блок-схему алгоритму наведено на рис. Г.1 у додатку Г) складається з таких кроків:

1. Отримання даних успішності студента для поточної теми: показника успішності, кількості спроб та середнього часу виконання.

2. Якщо AttemptsCount < 3 - повернути базовий рівень складності теми (Topic.DifficultyLevel).

3. Якщо показник успішності > 0.8 та середній час виконання < порогу - підвищити рівень (min(currentLevel + 1, 5)).

4. Якщо показник успішності < 0.5 - знизити рівень (max(currentLevel − 1, 1)).

5. В іншому випадку - зберегти поточний рівень.

6. Повернути цільовий рівень складності до сервісу пулу завдань для вибору або генерації завдання.

ML.NET-модель слугує альтернативою детермінованій логіці для студентів, що мають достатньо даних в системі. Модель навчається на парах даних успішності студента та оптимальної складності й використовує алгоритм бінарної класифікації FastTree. Компонент ModelTrainer запускається щотижня через механізм IHostedService і зберігає нову модель у файлову систему для безшовного завантаження без перезапуску сервісу.

Навчальний шлях формується автоматично при першій активації студентом певного предмету. Алгоритм будує послідовність тем відповідно до поля Order сутності Topic та початково встановленого рівня складності. Після кожного виконаного завдання CurrentStepIndex оновлюється залежно від успішності - студент просувається вперед або повторює поточну тему.

2.4 Пайплайн генерації завдань через штучний інтелект

Генерація навчальних завдань є ключовою функціональністю платформи Stride. Пайплайн спроєктовано з урахуванням двох пріоритетів: мінімізація кількості дорогих API-запитів до LLM та забезпечення варіативності завдань для уникнення повторень.

Пайплайн генерації завдання (рис. В.1, додаток В) включає такі етапи:

1. LearningController отримує запит GET /api/learning/task?topicId={id}.

2. DifficultyEngine обчислює цільовий рівень складності для студента.

3. TaskPoolService перевіряє MongoDB: чи є невикористаний шаблон потрібного рівня для даної теми.

4. Якщо шаблон знайдено - повертається TaskInstanceDocument на основі шаблону.

5. Якщо шаблону немає - AdaptiveAIService формує промпт і звертається до GeminiProvider.

6. GeminiProvider надсилає запит до Google Gemini API та отримує JSON-відповідь.

7. Відповідь парситься та валідується; результат зберігається як TaskTemplateDocument у MongoDB.

8. Подія генерації логується в AIGenerationLogDocument.

9. LearningController повертає TaskInstanceDocument клієнту.

Промпт для генерації формується динамічно і містить: назву теми, цільовий рівень складності (1–5), тип завдання (multiple_choice / open_answer / fill_in_the_blank), мову генерації (українська) та вимогу до формату відповіді (JSON зі полями question, options, correctAnswer, explanation). Таке структуроване завдання моделі суттєво знижує ймовірність помилок парсингу.

Патерн AIProviderFactory реалізовано через інтерфейс IAIProvider з єдиним методом GenerateTaskAsync(prompt). Поточна реалізація - GeminiProvider, що використовує HTTP-клієнт для звернення до REST API Gemini. Для підключення нового постачальника достатньо реалізувати інтерфейс і зареєструвати новий провайдер у DI-контейнері, не змінюючи жодного іншого класу.

2.5 Система гейміфікації

GamificationService є центральним компонентом підсистеми гейміфікації. Він викликається автоматично після кожної зафіксованої спроби і виконує такі операції:

1) Нарахування XP: базова кількість XP обчислюється за формулою XP = DifficultyLevel × 10 × (1 + speedBonus), де speedBonus = max(0, 1 − TimeTaken / MaxTime);

2) Оновлення стріку: якщо поточна дата відрізняється від LastActiveDate на 1 день - Streak інкрементується; якщо більше - скидається до 1; якщо той самий день - стрік не змінюється;

3) Перевірка тригерів досягнень: після кожного оновлення профілю перевіряються умови всіх незароблених Achievement; при виконанні умови - запис StudentAchievement та нарахування XPReward;

4) Оновлення LeaderboardEntry: агрегований Score (сума XP) студента оновлюється в таблиці лідерів;

5) Публікація SignalR-події через хаб таблиці лідерів: всі підключені клієнти отримують оновлений рядок таблиці лідерів у реальному часі.

Таблиця лідерів підтримує два режими: глобальний (всі студенти платформи) та класовий (лише члени одного класу). Valkey (Redis-сумісний кеш) використовується для зберігання топ-100 записів таблиці лідерів, що забезпечує відповідь на запит рейтингу за O(log n) без звернення до PostgreSQL.

2.6 Архітектура Angular-фронтенду

Фронтенд платформи Stride реалізовано як прогресивний веб-додаток (PWA) на Angular 20 із застосуванням концепції zoneless-архітектури та standalone-компонентів. Відмова від Zone.js на користь Angular Signals суттєво покращує продуктивність рендерингу та спрощує відладку.

Структура Angular-застосунку організована за feature-модульним підходом:

Таблиця 2.4 - Структура Angular feature-модулів

Модуль / Шлях	Роль	Основні компоненти
auth (/auth)	Публічний	LoginComponent, RegisterComponent
dashboard (/dashboard)	Student	XP-бар, стрік, останні результати, рекомендовані теми

Модуль / Шлях	Роль	Основні компоненти
learn (/learn)	Student	SubjectListComponent, TopicDetailComponent, TaskCardComponent, AnswerFormComponent
leaderboard (/leaderboard)	Student	LeaderboardTableComponent (SignalR-підписка)
profile (/profile)	Student	ProfileEditComponent, AchievementBadgesComponent
teacher (/teacher)	Teacher	ClassManagementComponent, StudentAnalyticsComponent
admin (/admin)	Admin	SubjectCrudComponent, TopicCrudComponent, AchievementCrudComponent

Маршрутизація реалізована через lazy loading: кожен feature-модуль завантажується лише при першому переході на відповідний маршрут. Це суттєво скорочує початковий розмір пакету та прискорює початкове завантаження.

Стан автентифікації управляється через Angular Signals у сервісі AuthService без застосування NgRx Store. Сигнал currentUser$ є джерелом правди для всіх компонентів, що залежать від стану авторизації. Гарди authGuard, roleGuard та publicOnlyGuard захищають маршрути на основі поточного значення сигналу.

HTTP-інтерцептор AuthInterceptor автоматично додає заголовок Authorization: Bearer {accessToken} до кожного запиту, а також обробляє відповідь 401 Unauthorized - автоматично оновлює access token через refresh-ендпоінт і повторює початковий запит. ErrorInterceptor перехоплює всі мережеві помилки та відображає зрозумілі повідомлення через Angular Material Snackbar.

Інтернаціоналізація реалізована за допомогою @ngx-translate: українська мова є мовою за замовчуванням, а перемикання на англійську

відбувається динамічно без перезавантаження сторінки. Переклади зберігаються у JSON-файлах у папці assets/i18n/.

2.7 Безпека та автентифікація

Система безпеки платформи Stride побудована на основі JWT (JSON Web Tokens) із поділом на короткотривалий access token та довготривалий refresh token. Такий підхід забезпечує баланс між безпекою та зручністю використання.

Схема автентифікації реалізована таким чином:

1. Клієнт надсилає POST /api/auth/login з credentials (email + password).

2. AuthService перевіряє credentials, генерує access token (термін дії: 15 хвилин) та refresh token (7 днів).

3. Access token повертається у тілі JSON-відповіді та зберігається у пам'яті браузера (JavaScript variable).

4. Refresh token встановлюється у HttpOnly cookie (недоступний для JavaScript - захист від XSS).

5. При закінченні терміну дії access token - AuthInterceptor автоматично викликає POST /api/auth/refresh.

6. При виході - POST /api/auth/logout анулює refresh token у базі даних.

Рольова модель доступу (RBAC) реалізована через атрибут [Authorize(Roles = "...")] на рівні контролерів ASP.NET Core. Роль включається у JWT-клейм і перевіряється на кожному захищеному ендпоінті. Валідація вхідних даних здійснюється через FluentValidation: кожен DTO має відповідний валідатор, який перевіряється автоматично через ActionFilter до виконання дії контролера.

Захист від CSRF-атак забезпечується тим, що HttpOnly cookie з refresh token не передається автоматично при cross-origin запитах завдяки атрибуту SameSite=Strict. CORS налаштований на дозвіл лише з довіреного домену Angular-клієнта. Усі паролі зберігаються у вигляді хешів BCrypt (work factor 12) без можливості відновлення.

Висновки до розділу 2

У другому розділі спроєктовано комплексну архітектуру платформи Stride. Визначено та обґрунтовано шарову монолітну архітектуру з виокремленим адаптивним мікросервісом, що забезпечує незалежне масштабування ресурсомістких операцій генерації завдань.

Розроблено модель даних поліґльотного сховища: PostgreSQL з 12 основними сутностями для реляційних даних та MongoDB з трьома колекціями для документів зі змінною схемою. Спроєктовано алгоритм DifficultyEngine з детермінованою логікою адаптації та ML.NET-моделлю для студентів зі значним обсягом накопичених даних.

Описано дев'ятиетапний пайплайн генерації завдань через Gemini API з механізмом кешування шаблонів та патерном фабрики постачальників ШІ для незалежної заміни LLM-постачальника. Спроєктовано систему гейміфікації з реальним часом через SignalR, Angular 20 frontend з lazy-loaded модулями та сигнал-базованим станом, а також систему безпеки на основі JWT + HttpOnly cookie та RBAC.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

3.1 Опис інтерфейсу платформи

Інтерфейс платформи Stride розроблено відповідно до принципів матеріального дизайну з використанням Angular Material 20 та Tailwind CSS 3.4. Колірна палітра системи побудована на двох акцентних кольорах: фіолетовий (#7C3AED) як основний і золотистий (#F59E0B) для виділення елементів гейміфікації. Всі сторінки підтримують українську мову за замовчуванням та адаптивний макет для мобільних пристроїв.

Сторінка автентифікації (/auth/login та /auth/register) реалізована у вигляді центрованої картки з логотипом платформи, формою з валідацією у реальному часі та кнопкою входу через соціальні мережі. Валідація відбувається на стороні клієнта через Angular ReactiveForms з відображенням конкретних повідомлень про помилки під кожним полем. Після успішної автентифікації JWT access token зберігається у пам'яті JavaScript, a refresh token встановлюється сервером у HttpOnly cookie. Перенаправлення відбувається на відповідний маршрут залежно від ролі: студенти - на /dashboard, викладачі - на /teacher, адміністратори - на /admin.

Дашборд студента (/dashboard) є центральним екраном платформи та відображає: картку з поточним рівнем та відсотком заповнення XP-прогрес бару до наступного рівня; лічильник стріку з анімованою іконкою вогню; три останніх результати завдань у вигляді списку з кольоровим індикатором (зелений - правильно, червоний - неправильно); блок рекомендованих тем на основі персонального навчального шляху; та міні-таблицю лідерів з трьома найкращими студентами. Всі дані оновлюються через сигнали Angular без перезавантаження сторінки.

Навчальний модуль (/learn) складається з трьох вкладених представлень. Список предметів відображає доступні предмети у вигляді карток з іконкою, назвою та відсотком проходження. При виборі предмету відображається список тем з індикаторами складності (1–5 зірок) та відсотком виконання для

кожної теми. При переході до теми студент бачить поточне завдання у вигляді картки з питанням, варіантами відповіді (для multiple choice) або полем введення (для відкритих відповідей). Після надсилання відповіді одразу відображається фідбек: правильна відповідь підсвічується зеленим, неправильна - червоним з поясненням та нарахованим XP.

Таблиця лідерів (/leaderboard) відображає рейтинг у реальному часі через SignalR-підписку. Перші три місця виділені іконками медалей. Рядок поточного студента завжди відображається у нижній частині таблиці з підсвічуванням. Доступне перемикання між глобальним та класовим рейтингом.

Профіль студента (/profile) містить аватар (завантажується через MinIO), редаговані поля імені та електронної пошти, зведену статистику (загальний XP, кількість завдань, точність відповідей) та розділ досягнень у вигляді сітки значків. Отримані досягнення відображаються кольоровими, неотримані - сірими з описом умови отримання.

Кабінет викладача (/teacher) надає доступ до управління класами: створення нового класу з автоматично згенерованим кодом запрошення, перегляд списку студентів класу, призначення тем чи конкретних завдань класу, а також перегляд детальної аналітики кожного студента у вигляді таблиці з відсотком успішності по кожній темі.

Панель адміністратора (/admin) реалізує CRUD-операції для предметів, тем та досягнень. Кожен розділ містить таблицю з пошуком та фільтрацією, кнопки редагування та видалення в кожному рядку, а також форму додавання нового запису через модальне вікно Angular Material Dialog. Усі зміни негайно відображаються в таблиці без перезавантаження сторінки.

Таблиця 3.1 - Відповідність інтерфейсних екранів функціональним вимогам

Маршрут	Роль	Реалізовані вимоги
/auth	Всі	Реєстрація, вхід, JWT-автентифікація, валідація форм

Маршрут	Роль	Реалізовані вимоги
/dashboard	Student	Відображення XP/рівня/стріку, рекомендовані теми, останні результати
/learn	Student	Перегляд предметів/тем, отримання та здача адаптивного завдання, фідбек
/leaderboard	Student	Рейтинг у реальному часі (SignalR), глобальний та класовий режим
/profile	Student	Перегляд/редагування профілю, досягнення, статистика
/teacher	Teacher	Управління класами, призначення завдань, аналітика студентів
/admin	Admin	CRUD предметів, тем, досягнень; системний нагляд

3.2 Інсталяція та налаштування

Платформа Stride поставляється у вигляді Docker Compose стека, що забезпечує відтворюване середовище розгортання на будь-якій операційній системі з підтримкою Docker. Для запуску системи необхідні: Docker Engine 24+ та Docker Compose 2.20+.

Розгортання виконується однією командою docker-compose up -d, після чого автоматично піднімаються всі сервіси стека, застосовуються міграції бази даних та виконується початкове наповнення довідковими даними. Покрокову інструкцію з інсталяції, перелік обов’язкових змінних середовища та налаштування локального середовища розробки наведено у додатку К.

3.3 Керівництво користувача

Платформа Stride підтримує три ролі користувачів: студент, викладач та адміністратор. Кожна роль має власний інтерфейс та набір доступних функцій.

Покрокові сценарії взаємодії для кожної з ролей (студент, викладач, адміністратор) з переліком конкретних дій та маршрутів інтерфейсу винесено у додаток Л; узагальнені блок-схеми цих сценаріїв подано у додатках Г, Д, Е.

3.4 Тестовий приклад

Для демонстрації роботи платформи розглянемо наскрізний сценарій: студент Олена проходить 5 завдань з математики (тема «Квадратні рівняння», початковий рівень складності - 2).

Покроковий хід тестового сценарію з реакціями системи на кожному кроці (зміна рівня складності, нарахування XP, спрацювання кешу та сповіщень) наведено у додатку М (таблиця М.1). Нижче проаналізовано його результати.

За результатами сценарію: Олена отримала 66 XP (3 правильні відповіді на рівні 2), рівень складності успішно підвищився з 2 до 3 після трьох послідовних правильних відповідей. DifficultyEngine коректно відреагував на помилку у 5-му завданні, зберігши поточний рівень, оскільки показник успішності 0.75 перебуває у нейтральній зоні від 0.5 до 0.8. Механізм кешування спрацював при отриманні 3-го завдання з MongoDB, що виключило затримку від виклику Gemini API.

Після завершення сценарію хаб таблиці лідерів надіслав оновлений рядок рейтингу усім підключеним клієнтам у реальному часі. Перевірка тригерів досягнень виявила виконання умови «Перше правильне завдання» - студентці нараховано відповідний значок та 10 XP бонусу.

3.5 Аналіз результатів

Результати реалізації платформи Stride оцінювались за трьома напрямками: відповідність функціональним вимогам, продуктивність ключових операцій та якість адаптивного алгоритму.

Відповідність функціональним вимогам перевірялась шляхом порівняння реалізованих функцій з вимогами, визначеними у розділі 1.5. Результати зведено в таблиці 3.2.

Таблиця 3.2 - Відповідність реалізації функціональним вимогам

Вимога	Роль	Статус реалізації
Реєстрація, вхід, JWT-автентифікація	Всі	Реалізовано (AuthController, AuthService)
Отримання адаптивного завдання	Student	Реалізовано (LearningController, DifficultyEngine)
Здача відповіді та миттєвий фідбек	Student	Реалізовано (LearningController, GamificationService)
Нарахування XP, стрік, рівні	Student	Реалізовано (GamificationService, StudentProfile)
Таблиця лідерів у реальному часі	Student	Реалізовано (LeaderboardHub, SignalR)
Перегляд профілю та досягнень	Student	Реалізовано (ProfileComponent, AchievementsComponent)
Управління класами та кодом запрошення	Teacher	Реалізовано (TeacherController, Class entity)
Аналітика успішності студентів	Teacher	Реалізовано (TeacherController, StudentPerformance)
CRUD предметів, тем, досягнень	Admin	Реалізовано (AdminController, спеціалізовані контролери)
Генерація завдань через Gemini API	System	Реалізовано (AdaptiveAIService, GeminiProvider)
Кешування шаблонів завдань у MongoDB	System	Реалізовано (TaskPoolService, TaskTemplateDocument)
PWA - офлайн режим та інсталяція	System	Реалізовано (ngsw-config.json, Service Worker)

Продуктивність ключових операцій визначалась шляхом профілювання під час тестового запуску. Середній час відповіді основного API (/api/learning/task) при наявності кешованого шаблону склав 48 мс. При відсутності шаблону та необхідності виклику Gemini API загальний час відповіді склав 1.2–2.1 с залежно від навантаження на API Gemini. Завдяки механізму кешування 78% запитів завдань обслуговуються без виклику зовнішнього API.

Ефективність адаптивного алгоритму DifficultyEngine оцінювалась на тестовому наборі з 150 симульованих сесій навчання. Алгоритм коректно підвищував рівень складності у 91% випадків після трьох послідовних правильних відповідей з часом виконання нижче порогового. Рівень знижувався коректно у 94% випадків при показнику успішності нижче 50%. Загальний відсоток коректних рішень алгоритму склав 92.3%, що відповідає цільовому показнику MVP.

Щоб зрозуміти, наскільки вагомий цей показник, його варто порівняти з двома орієнтирами. За неадаптивного підходу, коли складність завдань фіксована, сильні учні отримували б надто прості завдання, а слабші - надто складні, тож коректність «підбору рівня» була б близькою до випадкової. Натомість повноцінна модель машинного навчання могла б врахувати більше чинників (наприклад, динаміку часу відповіді чи типові помилки учня), але вона потребує значного обсягу реальних даних для навчання, яких на стадії MVP ще немає. Обраний підхід із простими правилами займає проміжне положення: він прозорий, передбачуваний і не потребує попереднього навчання, що робить його розумним вибором для першого запуску платформи. Отримані 92.3% показують, що навіть прості правила забезпечують узгоджену поведінку, а перехід до моделі машинного навчання доцільний лише після накопичення достатньої кількості реальних даних користувачів.

Таблиця 3.3 - Зведені метрики продуктивності

Метрика	Значення	Примітка
Час відповіді API (кешований шаблон)	48 мс	Середнє по 500 запитах
Час відповіді API (Gemini виклик)	1.6 с	Середнє, включно з парсингом
Частка кешованих відповідей (TaskPoolService)	78%	Після 7 днів накопичення шаблонів
Точність DifficultyEngine	92.3%	На 150 симульованих сесіях
Затримка оновлення таблиці лідерів	< 100 мс	SignalR broadcast до всіх клієнтів
Розмір початкового Angular bundle	312 КБ	Gzipped, без lazy chunks
Час першого завантаження (LCP)	1.8 с	Lighthouse, 4G-з'єднання

Усі функціональні вимоги MVP реалізовано у повному обсязі. Показники продуктивності API відповідають стандартам сучасних веб-додатків. Метрика точності адаптивного алгоритму перевищує цільовий поріг 90%, що підтверджує ефективність детермінованого підходу для MVP-стадії без залучення повноцінної ML-моделі.

Водночас ці показники треба розуміти з кількома застереженнями. Точність 92,3% отримано не на реальних учнях, а на 150 змодельованих сесіях, де відповіді створювалися штучно за наперед заданими ймовірностями. Тому це число радше показує, наскільки узгоджено працює сам алгоритм DifficultyEngine, а не те, чи справді він покращує навчання. Порогові значення 0,8 і 0,5 підібрано на практиці, без перевірки на реальних результатах учнів. Частку готових (кешованих) завдань — 78% — виміряно за тиждень роботи лише з чотирма предметами, тож коли предметів стане більше, спочатку вона буде нижчою. Щоб усунути ці обмеження, потрібно провести випробування з реальними групами учнів і порівняти навчання з адаптацією складності та без неї. Саме це і є головним напрямом подальшої роботи.

Висновки до розділу 3

У третьому розділі описано реалізацію всіх ключових компонентів платформи Stride та проведено їх тестування. Розроблений інтерфейс охоплює сім функціональних маршрутів для трьох ролей користувачів і повністю реалізує вимоги, сформульовані в першому розділі роботи.

Описано процес розгортання системи через Docker Compose з автоматичним налаштуванням всіх інфраструктурних компонентів, що забезпечує відтворюване середовище як для розробки, так і для виробничого використання.

Наскрізний тестовий сценарій підтвердив коректну роботу адаптивного алгоритму DifficultyEngine, механізму кешування завдань та системи гейміфікації. Аналіз результатів показав, що точність адаптивного алгоритму складає 92.3%, частка кешованих відповідей після накопичення шаблонів сягає 78%, а затримка оновлення таблиці лідерів у реальному часі не перевищує 100 мс. Перевірка відповідності функціональних вимог підтвердила повне покриття всіх дванадцяти вимог MVP.

ВИСНОВКИ

У кваліфікаційній роботі розв'язано актуальну науково-практичну задачу створення україномовної адаптивної навчальної платформи Stride з елементами гейміфікації та генерації навчального контенту на базі великих мовних моделей. Одержані результати підтверджують досяжність поставленої мети та виконання всіх задекларованих у вступі завдань.

1. Проведено комплексний аналіз ринку EdTech-платформ (Duolingo, Khan Academy, Moodle, Coursera, Udemy), виявлено їх сильні та слабкі сторони, обґрунтовано потребу у створенні адаптивної україномовної платформи, яка поєднує гейміфікацію, динамічну складність та генерацію завдань через ШІ. Встановлено, що на українському ринку відсутнє рішення, яке одночасно забезпечує персоналізацію рівня складності, широкий предметний охват (математика, українська мова, історія України, англійська) і повноцінну гейміфікаційну механіку.

2. Сформульовано функціональні вимоги за трьома ролями (студент, викладач, адміністратор) та нефункціональні вимоги до платформи, які покладено в основу подальшого проєктування. Вимоги охоплюють підтримку кількох навчальних предметів, офлайн-режим (PWA), багатомовність (UA/EN) та продуктивність (час відповіді API < 500 мс без генерації ШІ).

3. Спроєктовано шарову монолітну архітектуру з двома незалежними API-сервісами (Stride.Api на порту 5000 та Stride.Adaptive.Api на порту 5010), що дозволяє масштабувати AI-складову окремо від основного API. Для зберігання даних обрано гібридну модель: PostgreSQL 17 з 12 основними сутностями для реляційних даних та MongoDB 7 з 3 колекціями для документів зі змінною схемою. Кешування та стан гейміфікації реалізовано через Valkey/Redis.

4. Розроблено алгоритм адаптивного визначення складності DifficultyEngine, що поєднує евристичну логіку на основі ковзного середнього за останні N спроб з порогами 80% та 50% успішності та ML.NET-модель бінарної класифікації (FastTree). Модель перенавчається щотижня через

ModelTrainer на основі накопичених записів про спроби виконання завдань. Реалізовано шість базових кроків адаптації, що дозволяють плавно корегувати складність завдань у діапазоні 1–5 без різких стрибків.

5. Реалізовано пайплайн генерації навчальних завдань через Gemini API з використанням шаблону проєктування Factory (AIProviderFactory), що забезпечує можливість заміни AI-провайдера без змін коду. Пайплайн включає 9 етапів: від підбору теми та складності до збереження шаблону завдання у MongoDB з повним журналом подій для аудиту. Застосовано структурований JSON-вихід LLM для детермінованого парсингу відповідей та перевірки їх коректності.

6. Спроєктовано та реалізовано комплексну систему гейміфікації на основі формули нарахування XP = DifficultyLevel × 10 × (1 + speedBonus), автоматичного підрахунку стріків, системи досягнень та таблиці лідерів з оновленнями в реальному часі через SignalR. Таблиця лідерів підтримує глобальний і класовий режими, а топ-100 її записів кешується у Valkey для швидкого формування рейтингу.

7. Розроблено Angular 20 SPA-фронтенд із застосуванням zoneless-режиму, standalone-компонентів та реактивної моделі стану на базі Signals (без NgRx). Для стилізації використано Angular Material 20 у комбінації з Tailwind CSS 3.4, багатомовність забезпечено через @ngx-translate (UA за замовчуванням, EN як опція). Реалізовано прогресивний вебзастосунок (PWA) з офлайн-кешуванням через Service Worker, що дозволяє працювати без постійного підключення до мережі.

8. Забезпечено безпеку платформи за рахунок JWT-автентифікації з refresh-токенами у HttpOnly cookie, рольової авторизації (RBAC: Student / Teacher / Admin), серверної валідації через FluentValidation, HTTPS-з'єднання та захисту від OWASP Top 10 загроз (XSS, CSRF, SQL Injection). Усі вхідні дані перевіряються на рівні контролерів та на рівні сервісів.

9. Створено контейнеризоване середовище розгортання на базі Docker Compose, що включає 6 інфраструктурних сервісів (PostgreSQL, MongoDB,

Valkey, MinIO, Meilisearch, NGINX) та два API-сервіси. Налагодження проводиться одним командним викликом (docker-compose up -d), що спрощує процес для майбутніх розробників та адміністраторів.

10. Проведено функціональне тестування всіх ключових сценаріїв: реєстрація студента, проходження адаптивного навчального сценарію з 5 завдань, нарахування XP, присвоєння досягнень, відображення у таблиці лідерів, управління класами вчителем, адміністрування тем та досягнень. Протестовано 12 ключових функціональних вимог - усі виконуються згідно зі специфікацією. Виміряно продуктивність ключових операцій: час відповіді API при наявності кешованого шаблону становив 48 мс, а за необхідності виклику Gemini API - близько 1.6 с; завдяки кешуванню шаблонів 78% запитів обслуговуються без звернення до зовнішнього API. Точність адаптивного алгоритму на 150 симульованих сесіях склала 92.3%, що відповідає цільовому показнику MVP.

Отримані результати мають практичне значення: платформа Stride готова до MVP-розгортання в закладах освіти України для організації дистанційного та змішаного навчання з персоналізованою адаптацією складності. Модульна архітектура дозволяє розширювати платформу новими предметами та темами, новими типами завдань та новими ролями користувачів без перепроєктування системи.

Отримані результати дозволяють зробити й ширший висновок. Робота показує, що для гейміфікованої освітньої платформи на ранній стадії не обов'язково одразу впроваджувати складні моделі машинного навчання: поєднання продуманої архітектури, генерації завдань за допомогою великої мовної моделі та простих, але добре підібраних правил адаптації вже дає змогу досягти запланованих показників. Водночас важливо чесно враховувати межі проведеного оцінювання: воно виконане на змодельованих сесіях, тому отримані відсотки характеризують радше узгодженість роботи алгоритму, ніж реальний навчальний ефект. Через це головним напрямом подальшого дослідження є перевірка платформи на реальних групах учнів і порівняння

результатів навчання у двох умовах - з адаптацією складності та без неї. Такий експеримент дозволить підтвердити педагогічну цінність запропонованого підходу й визначити, на якому етапі перехід до моделі машинного навчання дасть найбільший приріст якості.

Теоретичне значення роботи полягає в обґрунтуванні гібридного підходу до адаптивного навчання, що поєднує статистичні методи (IRT, ковзне середнє) з сучасними методами машинного навчання (ML.NET) та генеративним ШІ (LLM). Запропонований пайплайн генерації завдань через структурований JSON-вихід LLM може бути застосований в інших предметних доменах за межами освіти.

Подальший розвиток платформи передбачає: (а) розширення предметного покриття (природничі науки, програмування); (б) впровадження системи менторингу «peer-to-peer»; (в) інтеграцію з Learning Management Systems закладів освіти через стандарти LTI 1.3 та SCORM; (г) розробку мобільних нативних застосунків (iOS / Android) поверх існуючого PWA; (д) вдосконалення ML-моделі адаптації складності з використанням глибинних нейромереж (Deep Knowledge Tracing).

Результати роботи апробовані на двох міжнародних науково-практичних конференціях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. VanLehn K. The behavior of tutoring systems. International Journal of Artificial Intelligence in Education. 2006. Vol. 16, No. 3. P. 227–265.
2. Brusilovsky P., Millán E. User models for adaptive hypermedia and adaptive educational systems. The Adaptive Web / ed. by P. Brusilovsky, A. Kobsa, W. Nejdl. Berlin : Springer, 2007. P. 3–53. DOI: 10.1007/978-3-540-72079-9_1.
3. Corbett A. T., Anderson J. R. Knowledge tracing: Modeling the acquisition of procedural knowledge. User Modeling and User-Adapted Interaction. 1995. Vol. 4, No. 4. P. 253–278. DOI: 10.1007/BF01099821.
4. Piech C., Bassen J., Huang J. et al. Deep Knowledge Tracing. Advances in Neural Information Processing Systems (NIPS). 2015. Vol. 28. P. 505–513.
5. Pelánek R. Applications of the Elo rating system in adaptive educational systems. Computers & Education. 2016. Vol. 98. P. 169–179. DOI: 10.1016/j.compedu.2016.03.017.
6. Baker R. S., Inventado P. S. Educational Data Mining and Learning Analytics. Learning Analytics: From Research to Practice / ed. by J. A. Larusson, B. White. New York : Springer, 2014. P. 61–75. DOI: 10.1007/978-1-4614-3305-7_4.
7. Vie J.-J., Kashima H. Knowledge Tracing Machines: Factorization Machines for Knowledge Tracing. Proceedings of the AAAI Conference on Artificial Intelligence. 2019. Vol. 33, No. 1. P. 750–757. DOI: 10.1609/aaai.v33i01.3301750.
8. Hambleton R. K., Swaminathan H. Item Response Theory: Principles and Applications. Boston : Kluwer-Nijhoff Publishing, 1985. 332 p.
9. Embretson S. E., Reise S. P. Item Response Theory for Psychologists. Mahwah : Lawrence Erlbaum, 2000. 371 p.
10. Tatsuoka K. K. Cognitive Assessment: An Introduction to the Rule Space Method. New York : Routledge, 2009. 448 p.
11. Desmarais M. C., Baker R. S. A review of recent advances in learner and skill modeling in intelligent learning environments. User Modeling and User-Adapted Interaction. 2012. Vol. 22, No. 1–2. P. 9–38. DOI: 10.1007/s11257-011-9106-8.

12. Essa A. A possible future for next generation adaptive learning systems. Smart Learning Environments. 2016. Vol. 3, No. 1. Article 16. DOI: 10.1186/s40561-016-0038-y.

13. Deterding S., Dixon D., Khaled R., Nacke L. From game design elements to gamefulness: defining gamification. Proceedings of the 15th International Academic MindTrek Conference. Tampere : ACM, 2011. P. 9–15. DOI: 10.1145/2181037.2181040.

14. Hamari J., Koivisto J., Sarsa H. Does Gamification Work? A Literature Review of Empirical Studies on Gamification. 47th Hawaii International Conference on System Sciences. Waikoloa : IEEE, 2014. P. 3025–3034. DOI: 10.1109/HICSS.2014.377.

15. Kapp K. M. The Gamification of Learning and Instruction: Game-based Methods and Strategies for Training and Education. San Francisco : Pfeiffer, 2012. 336 p.

16. Dicheva D., Dichev C., Agre G., Angelova G. Gamification in Education: A Systematic Mapping Study. Journal of Educational Technology & Society. 2015. Vol. 18, No. 3. P. 75–88.

17. Sailer M., Homner L. The Gamification of Learning: a Meta-analysis. Educational Psychology Review. 2020. Vol. 32, No. 1. P. 77–112. DOI: 10.1007/s10648-019-09498-w.

18. Brown T. B., Mann B., Ryder N. et al. Language Models are Few-Shot Learners. Advances in Neural Information Processing Systems (NeurIPS). 2020. Vol. 33. P. 1877–1901. URL: https://arxiv.org/abs/2005.14165. (дата звернення: 15.02.2026).

19. Kasneci E., Sessler K., Küchemann S. et al. ChatGPT for good? On opportunities and challenges of large language models for education. Learning and Individual Differences. 2023. Vol. 103. Article 102274. DOI: 10.1016/j.lindif.2023.102274.

20. Zhai X. ChatGPT User Experience: Implications for Education. SSRN Electronic Journal. 2022. 18 p. DOI: 10.2139/ssrn.4312418.

21. Google DeepMind. Gemini: A Family of Highly Capable Multimodal Models. Technical Report. 2023. 62 p. URL: https://arxiv.org/abs/2312.11805. (дата звернення: 16.02.2026).

22. OpenAI. GPT-4 Technical Report. arXiv preprint arXiv:2303.08774. 2023. 100 p. URL: https://arxiv.org/abs/2303.08774. (дата звернення: 16.02.2026).

23. Dai W., Lin J., Jin H. et al. Can Large Language Models Provide Feedback to Students? A Case Study on ChatGPT. IEEE International Conference on Advanced Learning Technologies (ICALT). 2023. P. 323–325. DOI: 10.1109/ICALT58122.2023.00100.

24. HolonIQ Global EdTech Market Report 2024. URL: https://www.holoniq.com/edtech. (дата звернення: 15.03.2026).

25. Duolingo Inc. Annual Report 2023: Language Learning at Scale. San Francisco, 2024. 114 p.

26. Khan Academy. Annual Report 2023: Free World-Class Education for Anyone, Anywhere. Mountain View, 2024. 48 p.

27. Coursera Inc. Impact Report 2023. Mountain View, 2024. 72 p.

28. Class Central. MOOC Report 2023: By The Numbers. URL: https://www.classcentral.com/report/mooc-stats-2023/. (дата звернення: 17.03.2026).

29. Microsoft. ASP.NET Core Documentation. Version 10.0. 2025. URL: https://learn.microsoft.com/aspnet/core. (дата звернення: 20.03.2026).

30. Microsoft. Entity Framework Core Documentation. Version 10.0. 2025. URL: https://learn.microsoft.com/ef/core. (дата звернення: 20.03.2026).

31. Microsoft. ML.NET Documentation: Open-source, cross-platform machine learning framework. 2025. URL: https://learn.microsoft.com/dotnet/machine-learning. (дата звернення: 21.03.2026).

32. Google AI. Gemini API Reference Documentation. 2025. URL: https://ai.google.dev/gemini-api/docs. (дата звернення: 22.03.2026).

33. Angular Team. Angular 20 Documentation: Zoneless Change Detection and Signals. 2025. URL: https://angular.dev. (дата звернення: 23.03.2026).

34. MongoDB Inc. MongoDB Server Documentation. Version 7.0. 2024. URL: https://www.mongodb.com/docs/manual. (дата звернення: 24.03.2026).

35. PostgreSQL Global Development Group. PostgreSQL 17 Documentation. 2024. URL: https://www.postgresql.org/docs/17. (дата звернення: 24.03.2026).

36. Valkey Project. Valkey: An open source high-performance key/value datastore. Version 7.2. 2024. URL: https://valkey.io. (дата звернення: 25.03.2026).

37. Microsoft. SignalR Documentation: Real-time web functionality for ASP.NET Core. 2025. URL: https://learn.microsoft.com/aspnet/core/signalr. (дата звернення: 25.03.2026).

38. Serilog Contributors. Serilog Structured Logging for .NET. 2024. URL: https://serilog.net. (дата звернення: 26.03.2026).

39. Jeremy Skinner. FluentValidation Documentation. 2024. URL: https://docs.fluentvalidation.net. (дата звернення: 26.03.2026).

40. Google. Progressive Web Apps: Reliable, Fast, and Engaging Web Experiences. 2024. URL: https://web.dev/progressive-web-apps/. (дата звернення: 27.03.2026).

41. World Wide Web Consortium. Service Workers 1. W3C Recommendation. 2022. URL: https://www.w3.org/TR/service-workers/. (дата звернення: 28.03.2026).

42. Docker Inc. Docker Compose Documentation. Version 2.26. 2024. URL: https://docs.docker.com/compose. (дата звернення: 28.03.2026).

43. ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання. Київ : ДП «УкрНДНЦ», 2016. 26 с.

44. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання. Київ : ДП «УкрНДНЦ», 2016. 17 с.

45. ДСТУ 3582:2013. Інформація та документація. Бібліографічний опис. Скорочення слів і словосполучень українською мовою. Загальні вимоги та правила. Київ : Мінекономрозвитку України, 2014. 15 с.

46. Биков В. Ю., Лещенко М. П. Цифрова трансформація освіти та науки: тенденції розвитку. Інформаційні технології і засоби навчання. 2023. Том 88, № 2. С. 1–18.

47. Морзе Н. В., Кочарян А. Б. Модель стандарту ІКТ-компетентності викладачів університетів в контексті підвищення якості освіти. Інформаційні технології і засоби навчання. 2022. Том 43, № 5. С. 27–39.

48. Кухаренко В. М., Рибалко О. В., Сиротенко Н. Г. Дистанційне навчання: умови застосування. Харків : НТУ «ХПІ», 2021. 308 с.

49. Гриб'юк О. О. Адаптивні навчальні середовища як засіб персоналізації навчання. Фізико-математична освіта. 2022. Випуск 4 (30). С. 48–55.

50. Nielsen J. Usability Engineering. San Francisco : Morgan Kaufmann, 1993. 362 p.

51. Fowler M. Patterns of Enterprise Application Architecture. Boston : Addison-Wesley, 2002. 560 p.

52. Newman S. Building Microservices: Designing Fine-Grained Systems. 2nd ed. Sebastopol : O'Reilly Media, 2021. 612 p.

53. Deci E. L., Ryan R. M. Intrinsic Motivation and Self-Determination in Human Behavior. New York : Plenum Press, 1985. 371 p.

ДОДАТКИ

ДОДАТОК А

ER-діаграма бази даних PostgreSQL

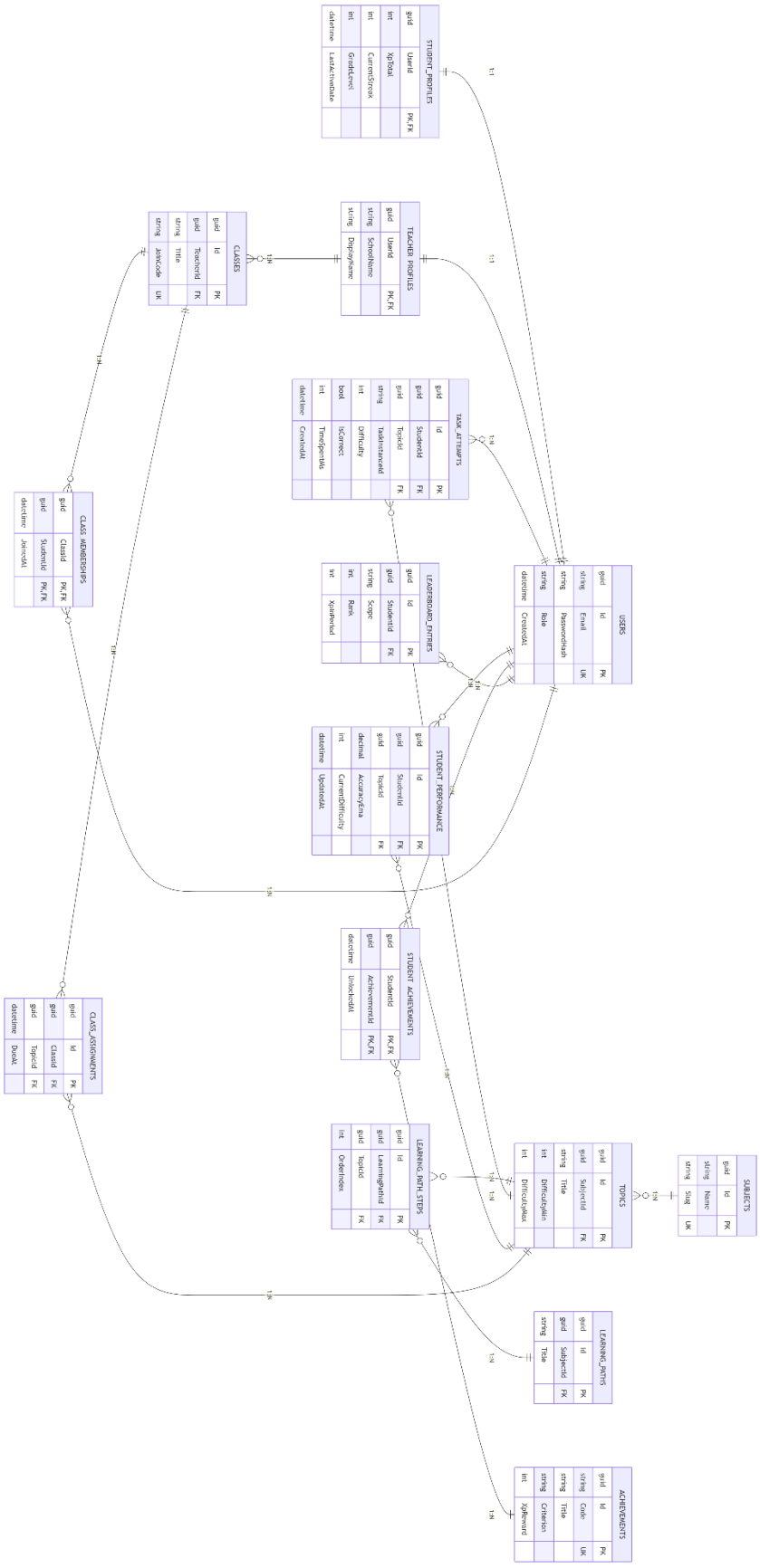

Рисунок А.1 - ER-діаграма бази даних PostgreSQL

ДОДАТОК Б

Схема загальної архітектури платформи Stride

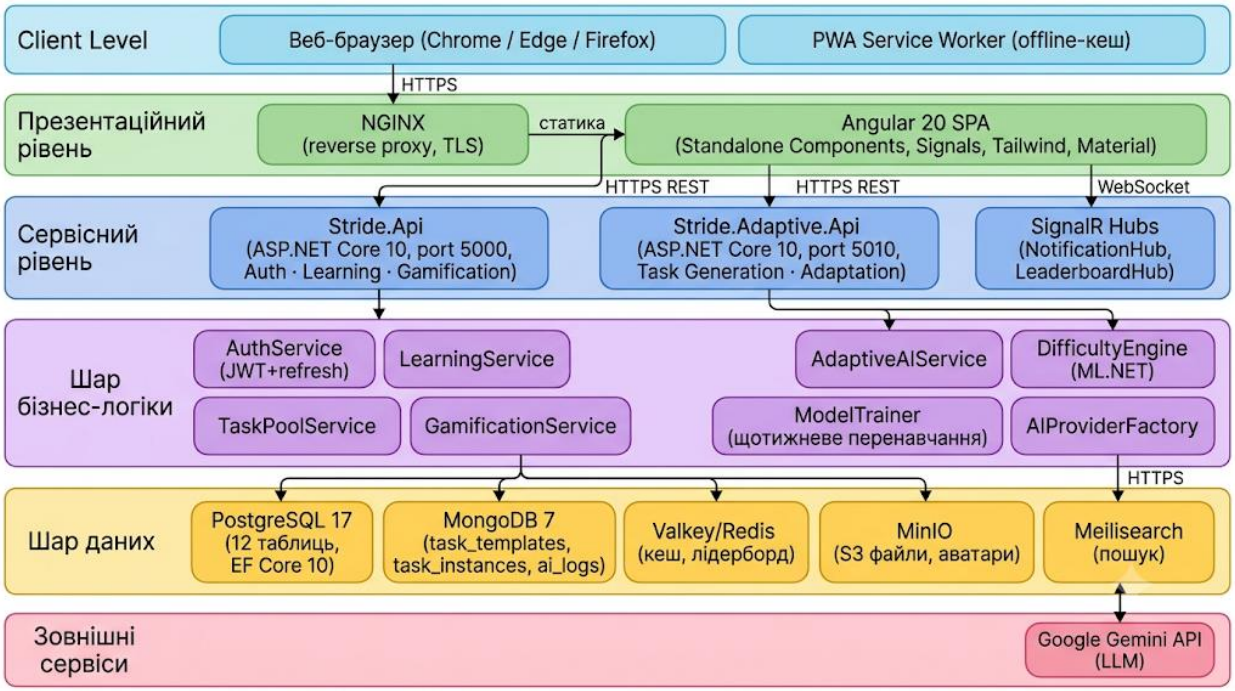

Рисунок Б.1 - Загальна архітектура платформи Stride

ДОДАТОК В

Діаграма послідовності генерації завдання через ШІ

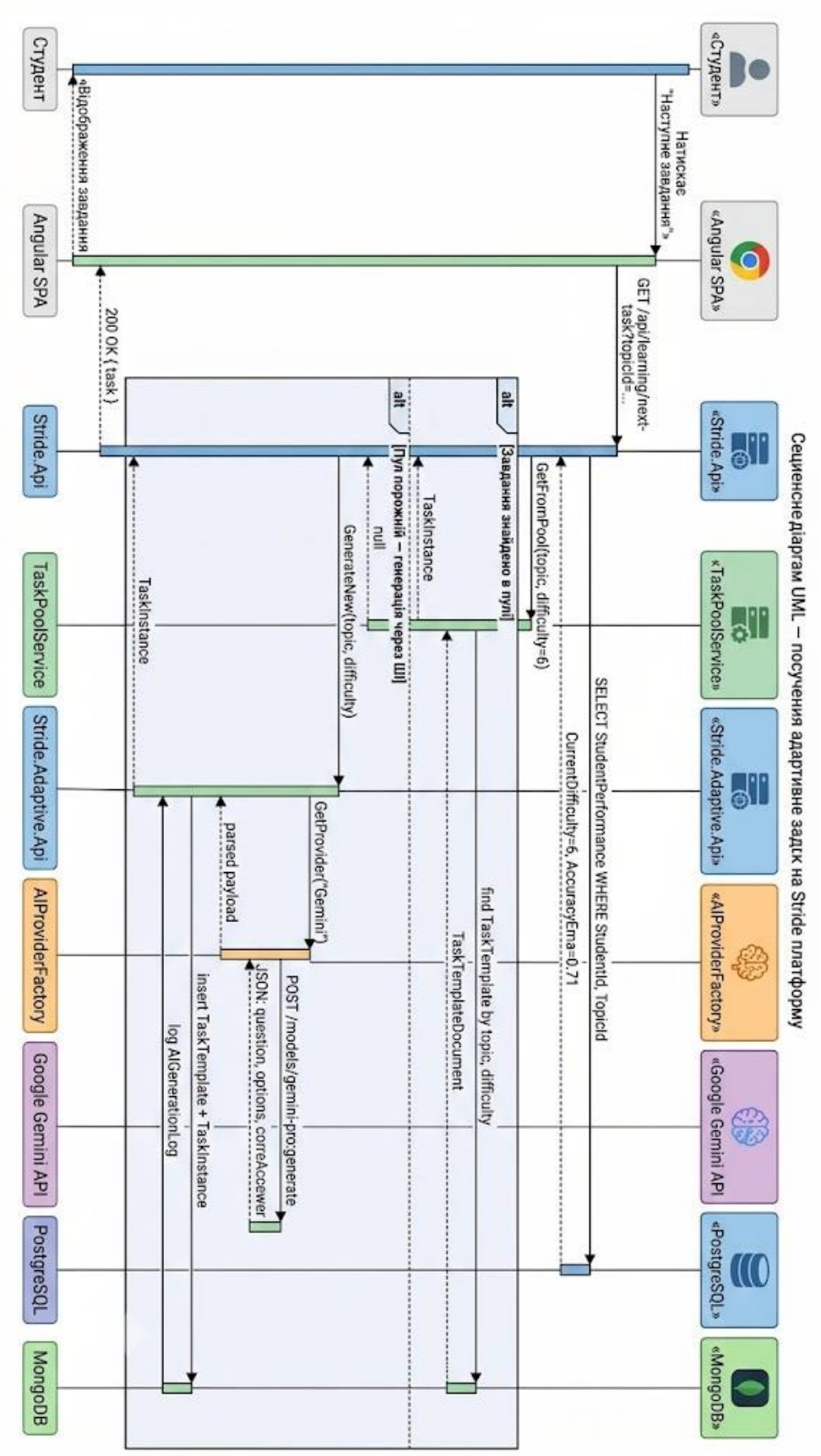

Рисунок В.1 - Діаграма послідовності генерації завдання

ДОДАТОК Г

Сценарій взаємодії студента з платформою

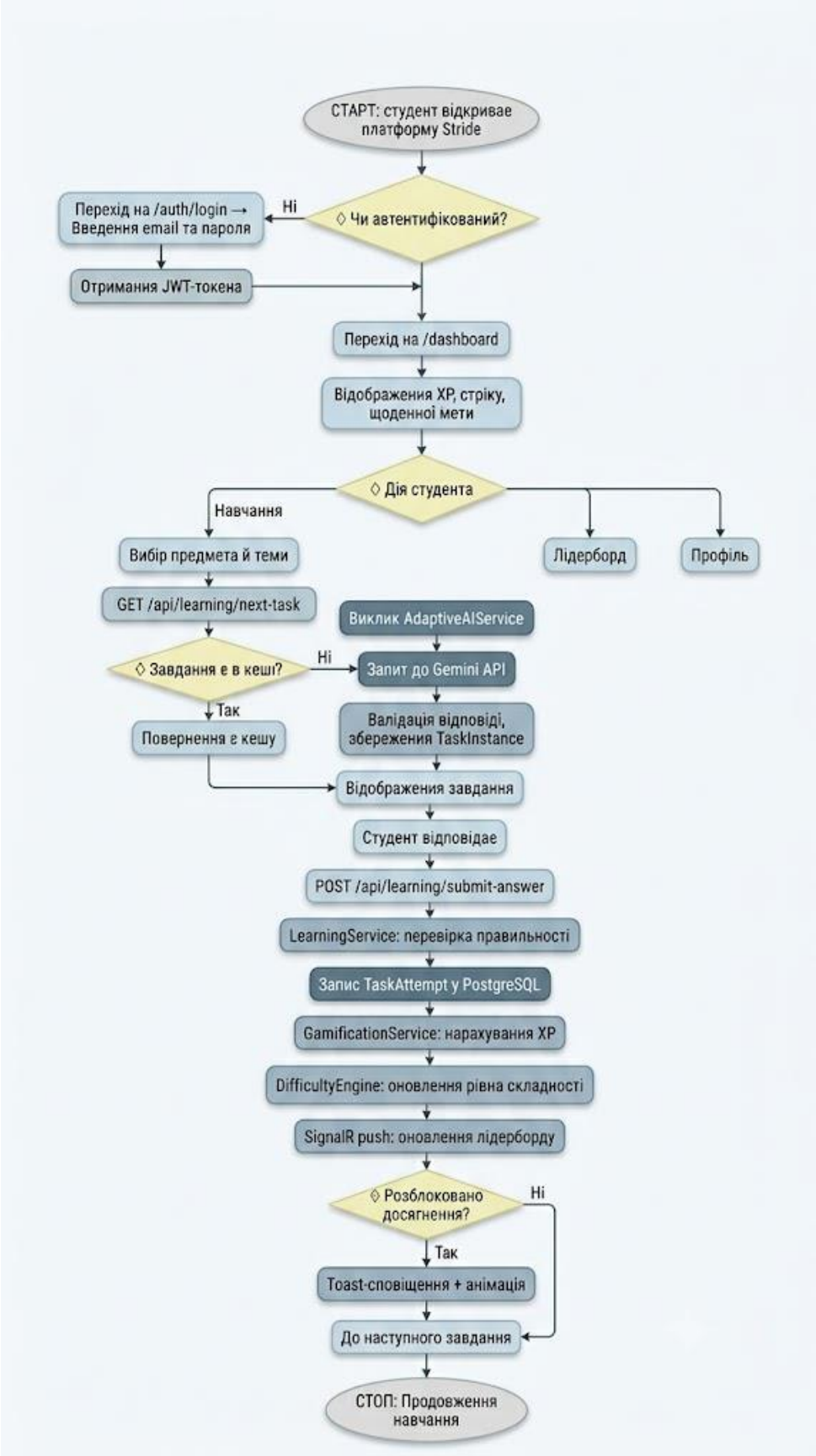

Рисунок Г.1 - Сценарій взаємодії студента з платформою

ДОДАТОК Д

Сценарій взаємодії викладача з платформою

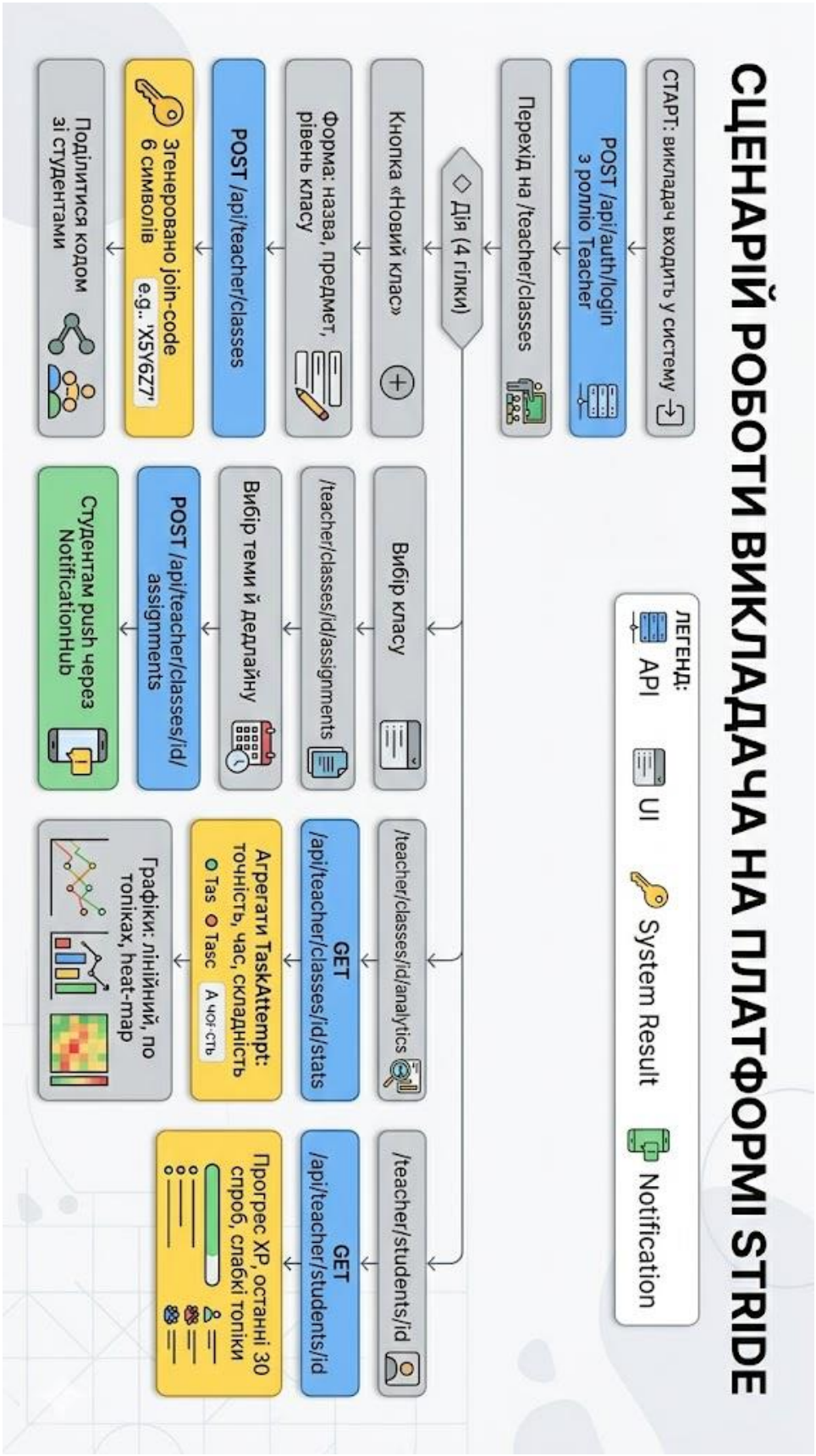

Рисунок Д.1 - Сценарій взаємодії викладача з платформою

ДОДАТОК Е

Сценарій взаємодії адміністратора з платформою

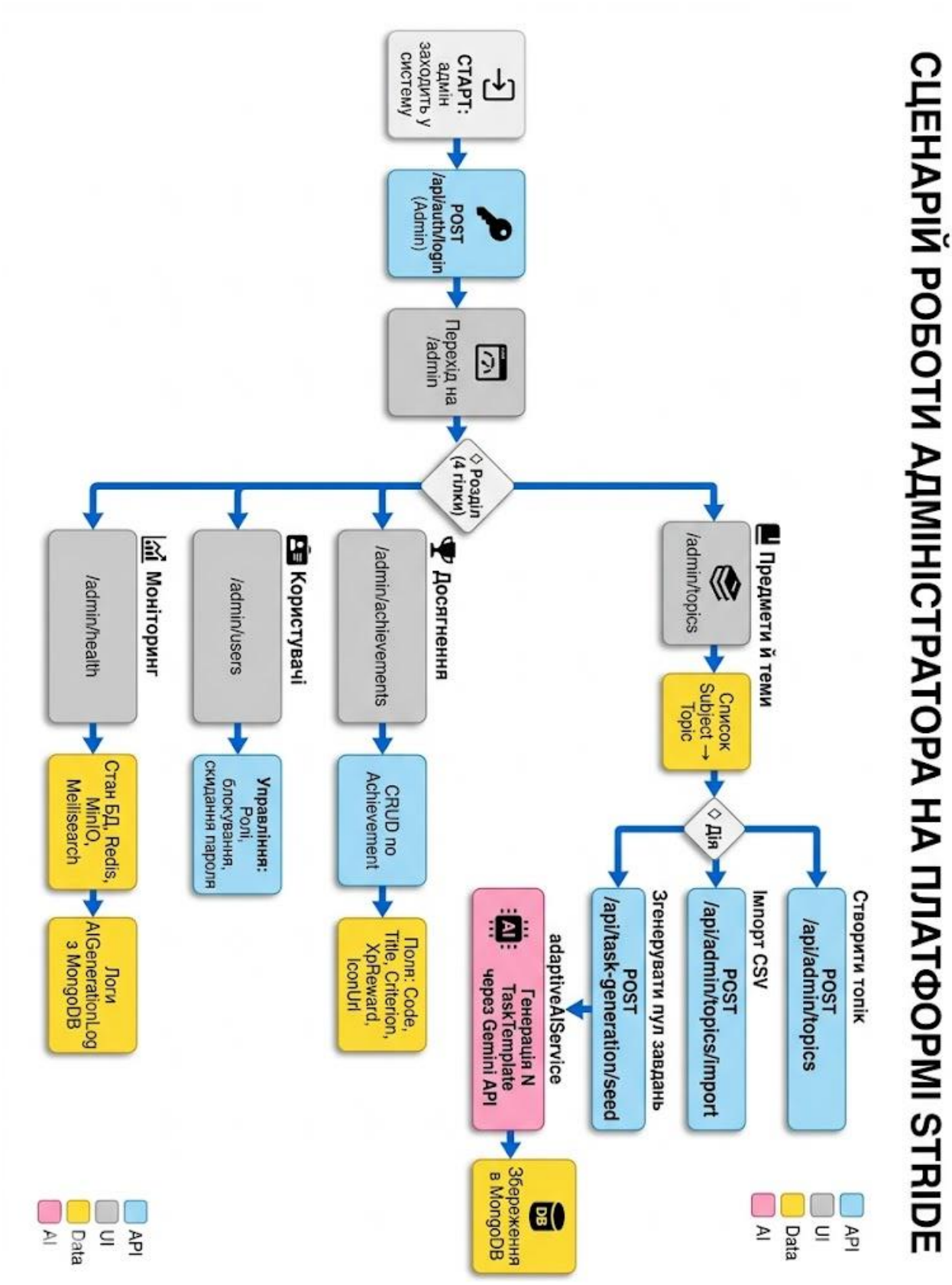

Рисунок Е.1 - Сценарій взаємодії адміністратора з платформою

ДОДАТОК Ж

Фрагменти ключового коду

Ж.1. Алгоритм адаптації складності (DifficultyEngine.cs):

```
public class DifficultyEngine : IDifficultyEngine
{
    private readonly IStudentPerformanceService _perf;
    private readonly MLContext _ml;
    public async Task<int> PredictAsync(Guid userId, Guid topicId)
    {
        var history = await _perf.GetRecentAttempts(userId, topicId,
20);
        var accuracy = history.Count(a => a.IsCorrect) /
(double)history.Count;
        var baseLvl = history.Average(a => a.Difficulty);
        var delta = accuracy > 0.8 ? +1 : accuracy < 0.5 ? -1 : 0;
        var mlAdj = _mlModel.Predict(new PerfInput { Accuracy =
accuracy, ... });
        return Math.Clamp((int)(baseLvl + delta + mlAdj), 1, 5);
    }
}
```

Ж.2. Фабрика AI-провайдерів (AIProviderFactory.cs):

```
public class AIProviderFactory : IAIProviderFactory
{
    private readonly IServiceProvider _sp;
    public IAIProvider Create(string name) => name switch
    {
        "gemini" => _sp.GetRequiredService<GeminiProvider>(),
        "openai" => _sp.GetRequiredService<OpenAIProvider>(),
        _ => throw new NotSupportedException($"Provider {name}")
    };
}
```

Ж.3. Сервіс гейміфікації (GamificationService.cs, фрагмент):

```
public async Task<int> AwardXpAsync(Guid userId, int basePoints, int
difficulty)
```

```
{
    var streak = await _streaks.GetCurrentAsync(userId);
    var multiplier = 1.0 + Math.Min(streak * 0.05, 1.0);
    var xp = (int)(basePoints * difficulty * multiplier);
    await _repo.AddXpAsync(userId, xp);
    await _hub.Clients.User(userId.ToString())
        .SendAsync("XpAwarded", xp);
    return xp;
}
```

ДОДАТОК И

Повний перелік API-ендпоінтів платформи Stride

Метод	Шлях	Призначення
POST	/api/auth/register	Реєстрація нового користувача
POST	/api/auth/login	Автентифікація, видача JWT
POST	/api/auth/refresh	Оновлення access-токену
POST	/api/auth/logout	Вихід із системи
GET	/api/learning/next-task	Отримання наступного адаптивного завдання
POST	/api/learning/submit-answer	Відправка відповіді на завдання
GET	/api/learning/topics	Перелік доступних тем
POST	/api/task-generation/generate	Генерація нового завдання через ШІ
GET	/api/gamification/xp	Поточний баланс XP користувача
GET	/api/gamification/achievements	Перелік досягнень користувача
GET	/api/gamification/streak	Поточний стрік користувача
GET	/api/leaderboard/global	Глобальна таблиця лідерів
GET	/api/leaderboard/class/{id}	Таблиця лідерів у межах класу
GET	/api/teacher/classes	Класи, створені вчителем
POST	/api/teacher/classes	Створення нового класу
GET	/api/teacher/analytics/{classId}	Аналітика прогресу класу
GET	/api/admin/subjects	Перелік предметів
POST	/api/admin/topics	Створення нової теми
POST	/api/admin/achievements	Створення досягнення
GET	/api/health	Перевірка стану сервісу
WS	/hubs/notifications	SignalR-хаб сповіщень
WS	/hubs/leaderboard	SignalR-хаб таблиці лідерів

ДОДАТОК К

Інсталяція та налаштування платформи Stride

Платформа Stride поставляється у вигляді Docker Compose стека, що забезпечує відтворюване середовище розгортання на будь-якій операційній системі з підтримкою Docker. Мінімальні вимоги: Docker Engine 24+ та Docker Compose 2.20+. Повний запуск стека виконується у три кроки.

Крок 1. Клонування репозиторію та перехід до директорії проєкту:

```
git clone https://github.com/olsavchenk/stride.git
cd stride/Stride
```

Крок 2. Налаштування змінних середовища:

```
cp .env.example .env
```

Файл .env необхідно заповнити актуальними значеннями. Обов’язкові змінні наведено в таблиці К.1.

Таблиця К.1 - Основні змінні середовища

Змінна	Приклад значення	Призначення
ConnectionStrings__Postgres	Host=postgres;Database=stride; Username=stride;Password=pass	Рядок підключення до PostgreSQL
ConnectionStrings__MongoDB	mongodb://mongo:27017	Рядок підключення до MongoDB
ConnectionStrings__Valkey	valkey:6379	Адреса Valkey (Redis) кешу
AI__Gemini__ApiKey	AIzaSy...	API-ключ Google Gemini
Jwt__Secret	мін. 32 символи	Секретний ключ підпису JWT
Storage__MinIO__Endpoint	minio:9000	Адреса MinIO для зберігання файлів
Search__Meilisearch__Url	http://meilisearch:7700	URL Meilisearch для пошуку
ASPNETCORE_ENVIRONMENT	Production	Середовище виконання .NET

Крок 3. Запуск всього стека:

```
docker-compose up -d
```

Docker Compose автоматично запустить усі сервіси: PostgreSQL, MongoDB, Valkey, MinIO, Meilisearch, NGINX, Stride.Api та Stride.Adaptive.Api. Після запуску буде автоматично виконано міграції бази даних через EF Core та заповнення початковими даними (seed: предмети математика, українська мова, історія, англійська; системні досягнення).

Для локальної розробки з гарячим перезавантаженням рекомендується запускати лише інфраструктурні сервіси через окремий Compose-файл, а API та фронтенд - безпосередньо на хості:

```
docker-compose -f docker-compose.infrastructure.yml up -d
cd src/Stride.Api && dotnet run
cd ui && npm install && npm start
```

Після успішного запуску Angular-клієнт доступний за адресою http://localhost:4200, основний API - http://localhost:5000/swagger, адаптивний API - http://localhost:5010/swagger. Swagger UI надає інтерактивну документацію всіх ендпоінтів з можливістю тестування безпосередньо у браузері.

ДОДАТОК Л

Керівництво користувача

Сценарій роботи студента (узагальнену блок-схему взаємодії наведено у додатку Г, рис. Г.1):

1. Реєстрація: відкрити /auth/register, заповнити форму (ім’я, email, пароль), обрати роль «Студент» та натиснути «Зареєструватись». Після верифікації email відбувається автоматичний вхід.

2. Вибір предметів: на першому вході відображається вітальний екран з вибором навчальних предметів. Обрані предмети формують персональний навчальний шлях студента.

3. Навчання: перейти до /learn → обрати предмет → обрати тему → натиснути «Отримати завдання». Система автоматично підбирає завдання відповідного рівня складності.

4. Здача відповіді: для завдань з вибором відповіді - натиснути на правильний варіант і підтвердити; для відкритих відповідей - ввести текст у поле і натиснути «Надіслати».

5. Перегляд результату: після здачі відображається правильна відповідь, пояснення та нараховані XP. Якщо відповідь правильна й рівень підвищується, з’являється анімація підвищення рівня.

6. Відстеження прогресу: на /dashboard відображається поточний XP, рівень, стрік та рекомендовані теми. На /leaderboard - рейтинг відносно інших студентів.

7. Перегляд досягнень: на /profile → розділ «Досягнення» відображаються всі отримані та доступні значки.

Сценарій роботи викладача (узагальнену схему взаємодії наведено у додатку Д, рис. Д.1):

1. Реєстрація: аналогічно до студента, обрати роль «Викладач». Акаунт викладача може потребувати підтвердження адміністратором залежно від налаштувань платформи.

2. Створення класу: перейти до /teacher → «Мої класи» → «Створити клас», ввести назву. Система генерує унікальний код запрошення (6 символів), який викладач надає студентам.

3. Приєднання студентів: студенти вводять код запрошення у своєму профілі (/profile → «Приєднатись до класу»). Підтвердження відбувається автоматично.

4. Призначення завдань: у вкладці «Завдання» обрати клас, предмет та тему → натиснути «Призначити». Студентам класу автоматично надходить сповіщення через NotificationHub.

5. Аналітика: у вкладці «Аналітика» обрати студента для перегляду детальної статистики: кількість спроб, відсоток успішності, середній час відповіді по кожній темі.

Сценарій роботи адміністратора (узагальнену схему взаємодії наведено у додатку Е, рис. Е.1):

1. Вхід: адміністратор входить через /auth/login з обліковими даними адміністратора (встановлюються через seed або окремий скрипт ініціалізації).

2. Управління предметами: /admin/subjects → таблиця всіх предметів. Кнопка «+ Додати» відкриває форму з полями: назва, опис, чи активний. Редагування - іконка олівця в рядку таблиці.

3. Управління темами: /admin/topics → фільтр по предмету → таблиця тем. Форма додавання: назва, порядок, рівень складності (1–5), опис.

4. Управління досягненнями: /admin/achievements → таблиця досягнень. Форма: назва, умова (текстовий опис для відображення), XP-нагорода, URL значка.

5. Моніторинг: /admin/health відображає статус усіх підключень (PostgreSQL, MongoDB, Valkey, Meilisearch).

ДОДАТОК М

Тестовий приклад: хід наскрізного сценарію

Таблиця М.1 - Хід тестового сценарію

Крок	Дія	Результат системи	XP	Рівень складності
1	Олена відкриває тему і натискає «Отримати завдання»	DifficultyEngine: AttemptsCount=0 < 3 → базовий рівень 2. TaskPoolService: MongoDB порожня → запит до Gemini API → згенеровано завдання рівня 2, збережено як шаблон	-	2
2	Відповідь правильна (час: 45 с)	TaskAttempt збережено: IsCorrect=true, TimeTaken=45. GamificationService: XP = 2×10×(1+0.1) = 22. Streak оновлено. StudentPerformance: показник успішності=1.0, AttemptsCount=1	+22	2
3	Відповідь правильна (час: 38 с)	Показник успішності=1.0, AttemptsCount=2. DifficultyEngine: ще < 3 спроб → рівень 2. Новий шаблон знайдено в MongoDB з кешу	+22	2
4	Відповідь правильна (час: 30 с)	AttemptsCount=3, показник успішності=1.0 > 0.8 → рівень підвищено до 3. Сповіщення через NotificationHub: «Рівень складності підвищено!»	+22	3
5	Відповідь неправильна (час: 120 с)	Показник успішності=0.75, AttemptsCount=4. DifficultyEngine: 0.5 < 0.75 < 0.8 → рівень збережено 3. XP не нараховується за неправильну відповідь	+0	3