

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «магістр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

**«Розробка он-лайн платформи з проходження курсів
військової підготовки з елементами штучного
інтелекту»**

Виконав студент групи ІПЗмп-23-3
КРАВЧЕНКО Олександр Іванович

Керівник роботи:
ЛОКТИКОВА Тамара Миколаївна

Рецензент:
ЛОБАНЧИКОВА Надія Миколаївна

Житомир – 2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення

Тетяна ВАКАЛЮК

«07» травня 2026 р

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **КРАВЧЕНКО Олександр Іванович**

Керівник роботи: **ЛОКТИКОВА Тамара Миколаївна**

Тема роботи: **«Розробка он-лайн платформи з проходження курсів військової підготовки з елементами штучного інтелекту»**, затверджена Наказом закладу вищої освіти від **«06» травня 2026 р., №205/с**

Вихідні дані для роботи: **об'єктом дослідження є процес організації та персоналізації он-лайн навчання військовослужбовців у цифровому середовищі, а предметом дослідження - методи та архітектурні принципи побудови он-лайн платформ, алгоритми семантичного ранжування навчального контенту з використанням векторних моделей представлення даних.**

Керівники / консультанти з магістерської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Керівник/ консультант	Завдання видав	Завдання прийняв
1	Локтікова Т.М.	08.05.2026р.	08.05.2026р.
2	Локтікова Т.М.	08.05.2026р.	08.05.2026р.
3	Локтікова Т.М.	08.05.2026р.	08.05.2026р.

РЕФЕРАТ

Магістерська атестаційна робота складається зі вступу, трьох розділів, висновків, списку використаних джерел, додатків. Текстова частина складає 88 сторінок. Робота містить 13 ілюстрацій та 14 таблиць. Список використаних джерел нараховує 40 найменувань та займає три сторінки. У роботі міститься 26 сторінок додатків.

Мета дослідження полягає у розробці архітектурних принципів, проектуванні та програмній реалізації он-лайн платформи для військової підготовки з використанням алгоритмів інтелектуального ранжування контенту для забезпечення персоналізованої освітньої траєкторії.

Було оглянуто й проаналізовано сучасні освітні платформи та ІІІ-технології, обґрунтовано необхідність їхньої адаптації до специфічних військових вимог; спроектовано архітектуру платформи з локальним ІІІ; розроблено алгоритм семантичного ранжування на основі векторних ембедінгів та косинусної схожості; реалізовано он-лайн платформу із системою моніторингу навчального прогресу в реальному часі.

Результати роботи можуть бути використані для організації дистанційної підготовки особового складу в навчальних центрах та військових закладах вищої освіти.

КЛЮЧОВІ СЛОВА: ОН-ЛАЙН ПЛАТФОРМА, ВІЙСЬКОВА ПІДГОТОВКА, ШТУЧНИЙ ІНТЕЛЕКТ, ПЕРСОНАЛІЗАЦІЯ НАВЧАННЯ, POSTGRESQL, NESTJS, REACT.

ABSTRACT

The master's thesis consists of an introduction, three chapters, conclusions, a list of references, and appendices. The main text comprises 88 pages. The thesis contains 13 illustrations and 14 tables. The list of references includes 40 entries and spans three pages. The thesis includes 26 pages of appendices.

The aim of the study is to develop architectural principles, design, and programmatically implement an online platform for military training using intelligent content ranking algorithms to ensure a personalized educational trajectory.

Modern educational platforms and AI technologies were reviewed and analyzed, and the need for their adaptation to specific military requirements was substantiated; The architecture of a platform with local AI was designed; a semantic ranking algorithm based on vector embeddings and cosine similarity was developed; and a software suite with a real-time learning progress monitoring system was implemented.

The results of this work can be used to organize remote training for personnel at training centers and military higher education institutions.

KEYWORDS: ONLINE PLATFORM, MILITARY TRAINING, ARTIFICIAL INTELLIGENCE, PERSONALIZED LEARNING, POSTGRESQL, NESTJS, REACT.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	6
ВСТУП	8
РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОБҐРУНТУВАННЯ АКТУАЛЬНОСТІ ТЕМИ РОБОТИ	11
1.1 Обґрунтування актуальності теми роботи та основні підходи до побудови он-лайн платформ навчання	11
1.2 Аналіз аналогів он-лайн платформ навчання.....	17
1.3 Постановка задачі та вимоги до платформи військової підготовки	25
Висновки до першого розділу	29
РОЗДІЛ 2 МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ ОН-ЛАЙН ПЛАТФОРМИ ВІЙСЬКОВОЇ ПІДГОТОВКИ З ЕЛЕМЕНТАМИ ШТУЧНОГО ІНТЕЛЕКТУ	31
2.1 Архітектурна модель платформи військової підготовки	31
2.2 Проектування підсистем управління навчальним процесом	38
2.3 Алгоритм інтелектуального ранжування курсів.....	43
Висновки до другого розділу.....	50
РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ДОСЛІДЖЕННЯ ФУНКЦІОНУВАННЯ ОН-ЛАЙН ПЛАТФОРМИ	52
3.1 Вибір та обґрунтування технологій та інструментів розробки.....	52
3.2 Реалізація он-лайн платформи.....	66
3.3 Інтерфейс користувача та його функціональні можливості.....	76
Висновки до третього розділу	81
ВИСНОВКИ	83
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	85
ДОДАТКИ	88

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – (англ. Application Programming Interface) – програмний інтерфейс застосунку, набір визначень, протоколів та інструментів, які дозволяють різним програмним компонентам взаємодіяти та обмінюватися даними між собою.

LLM – (англ. Large Language Model) – велика мовна модель, тип моделі штучного інтелекту, побудованої на архітектурі нейронних мереж, навченої на величезних масивах текстових даних для розуміння та генерації природної мови.

LMS – (англ. Learning Management System) – система управління навчанням, он-лайн платформа, призначена для адміністрування навчальних курсів, розподілу контенту, відстеження прогресу та оцінювання результатів навчання.

ORM – (англ. Object-Relational Mapping) – об’єктно-реляційне відображення; технологія програмування, що дозволяє взаємодіяти з реляційною базою даних за допомогою концепцій об’єктно-орієнтованої мови програмування без написання прямих SQL-запитів.

REST – (англ. Representational State Transfer) – архітектурний стиль взаємодії компонентів розподіленого застосунку в мережі, що базується на використанні стандартних методів протоколу HTTP для маніпуляції ресурсами.

SPA – (англ. Single Page Application) – односторінковий вебзастосунок. Архітектура вебсайту, при якій весь необхідний код завантажується одноразово, а оновлення контенту відбувається динамічно без повного перезавантаження сторінки в браузері.

UML – (англ. Unified Modeling Language) – уніфікована мова моделювання; стандартизована графічна мова для візуалізації, специфікації, проєктування та документування структури і поведінки програмних систем.

СУБД – (система управління базами даних) – спеціалізоване програмне забезпечення, призначене для створення, ведення, захисту та спільного використання баз даних багатьма користувачами або застосунками.

БД – (база даних) – структурована сукупність взаємопов'язаних даних, призначена для зберігання, пошуку та обробки інформації в інформаційній системі.

ШІ – (штучний інтелект) – сукупність методів і технологій, що забезпечують здатність комп'ютерних систем виконувати завдання, які потребують елементів людського мислення, аналізу та прийняття рішень.

ПЗ – (програмне забезпечення) – сукупність програм, процедур та документації, необхідних для функціонування комп'ютерної системи.

ВСТУП

Актуальність теми. Сучасний етап розвитку інформаційного суспільства та стрімка цифровізація всіх сфер людської діяльності зумовлюють необхідність трансформації підходів до професійної підготовки кадрів, зокрема у військовій сфері. В умовах динамічних змін безпекового середовища та необхідності швидкого опанування новітніми зразками озброєння і тактичними прийомами, традиційні методи навчання потребують доповнення гнучкими та інтелектуальними системами дистанційної освіти.

Особливої гостроти набуває питання конфіденційності та безпеки даних у військових навчальних системах. Використання популярних хмарних рішень на базі штучного інтелекту створює ризики витоку чутливої інформації про інтереси та рівень підготовки особового складу. Таким чином, розробка архітектурних принципів, що дозволяють виконувати інтелектуальний аналіз даних безпосередньо на пристрої користувача, є стратегічно важливим завданням.

На сьогодні розроблено багато платформ для організації дистанційного навчання. Проте, більшість існуючих рішень, а саме Moodle чи Coursera, або не мають вбудованих механізмів семантичного ранжування контенту, або покладаються на серверні обчислення, що не завжди прийнятно для військового сектору. Невирішеним залишається коло питань щодо створення легких, автономних алгоритмів персоналізації, які б поєднували високу точність рекомендацій із дотриманням суворих вимог до приватності.

Мета дослідження полягає у розробці архітектурних принципів, проєктуванні та програмній реалізації он-лайн платформи військової підготовки з використанням алгоритмів інтелектуального ранжування контенту для забезпечення персоналізованої освітньої траєкторії.

Для досягнення поставленої мети необхідно вирішити такі основні **завдання**:

1) проаналізувати сучасні підходи до побудови он-лайн платформ навчання та досвід застосування технологій ШІ в освітньому процесі;

2) визначити специфічні вимоги до систем військової підготовки, зокрема в контексті безпеки даних та структурованості контенту;

3) спроектувати архітектурну модель платформи, що забезпечує виконання нейромережових обчислень на стороні клієнта;

4) розробити математичну модель та алгоритм семантичного ранжування курсів на основі векторних ембедінгів та профілю компетенцій користувача;

5) реалізувати он-лайн платформу із використанням сучасного стеку технологій та інтегрувати підсистему управління навчальним прогресом.

Об’єктом дослідження є процес організації та персоналізації он-лайн навчання військовослужбовців у цифровому середовищі.

Предмет дослідження - методи та архітектурні принципи побудови он-лайн платформ, алгоритми семантичного ранжування навчального контенту з використанням векторних моделей представлення даних.

Наукова новизна отриманих результатів полягає у наступному:

- удосконалено алгоритм інтелектуального ранжування навчального контенту шляхом впровадження системи евристичних штрафів за завершені курси, що дозволяє динамічно оновлювати рекомендації залежно від актуального стану прогресу;

- удосконалена модель профілю компетенцій користувача через автоматичне виявлення слабких тем на основі аналізу результатів тестування, що слугує вхідним контекстом для семантичного пошуку релевантних курсів;

- запропоновано архітектурне рішення для військових навчальних платформ, яке базується на використанні клієнтського інтерфейсу для генерації ембедінгів, що забезпечує повну приватність профілю компетенцій курсанта.

Практичне значення отриманих результатів полягає у створенні програмного комплексу, який може бути використаний для організації дистанційної підготовки особового складу. Розроблені інструменти адміністратора дозволяють швидко наповнювати систему контентом та автоматично готувати його для

інтелектуального аналізу. Впровадження результатів дослідження дозволяє підвищити ефективність навчання за рахунок концентрації уваги курсанта на найбільш релевантних та проблемних для нього темах.

Апробація. За темою випускної кваліфікаційної роботи магістра були опубліковані:

1. Kravchenko O.I. Selection and Justification of Architecture for a Web-Based System for Military Training Courses. – Технічна творчість: Збірник наукових праць/ Укл.: Скиба М.Є., Поліщук О.С., Романець Т.П. – Хмельницький: ХНУ, 2023. – № 7. С. 112 – 114 [1];

2. Kravchenko O.I. Subject Area Research For Competitor Analysis. Технічна творчість: Збірник наукових праць/ Укл.: Скиба М.Є., Поліщук О.С., Романець Т.П. – Хмельницький: ХНУ, 2023. – № 7. С. 139 – 141 [2].

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОБҐРУНТУВАННЯ АКТУАЛЬНОСТІ ТЕМИ РОБОТИ

1.1 Обґрунтування актуальності теми роботи та основні підходи до побудови он-лайн платформ навчання

Сучасний етап розвитку інформаційних технологій характеризується стрімкою цифровізацією різних сфер діяльності людини, зокрема освіти. Активне впровадження мережових технологій, зростання доступності інтернету та розвиток програмного забезпечення сприяли формуванню нових форматів навчання, серед яких провідне місце займає електронне навчання. У цьому контексті он-лайн платформи навчання виступають ключовим інструментом організації освітнього процесу в цифровому середовищі.

Он-лайн платформи навчання являють собою комплексні інформаційні системи, що забезпечують створення, розповсюдження та засвоєння навчального контенту з використанням вебтехнологій. Вони дозволяють організувати повноцінний освітній процес незалежно від географічного розташування користувачів, що особливо актуально в умовах дистанційного або змішаного навчання. Такі системи забезпечують інтерактивну взаємодію між учасниками навчального процесу, автоматизацію оцінювання знань, а також накопичення і аналіз даних про навчальну діяльність [3].

У сучасних умовах он-лайн платформи навчання використовуються не лише в академічній освіті, але й у професійній підготовці, корпоративному навчанні та підвищенні кваліфікації. Особливої актуальності вони набувають у специфічних галузях, де необхідно поєднувати теоретичну підготовку з практичними навичками, а також забезпечувати постійне оновлення знань. Однією з таких сфер є військова підготовка, де важливу роль відіграє оперативність отримання знань, їх актуальність та можливість адаптації навчального процесу до різних категорій користувачів.

Ефективність функціонування он-лайн платформи навчання значною мірою визначається обраними архітектурними підходами, які регламентують організацію взаємодії між компонентами системи, розподіл відповідальності між ними, а також забезпечують масштабованість, надійність і зручність подальшого розвитку програмного продукту. У сучасних умовах розробки інформаційних систем домінуючим є клієнт-серверний підхід, який передбачає розділення системи на дві основні частини: клієнтську та серверну [4].

Клієнтська частина (frontend) відповідає за взаємодію з користувачем, відображення інтерфейсу та обробку користувацьких дій. Вона реалізується у вигляді вебдодатку, який працює у браузері та забезпечує доступ до функціональних можливостей платформи. Серверна частина (backend), у свою чергу, відповідає за обробку запитів, виконання бізнес-логіки, управління даними та забезпечення безпеки. Взаємодія між клієнтом і сервером здійснюється через програмний інтерфейс (API), найчастіше реалізований у вигляді REST-сервісів, що дозволяє стандартизувати обмін даними та забезпечити гнучкість інтеграції [4, 5].

Подальший розвиток клієнтської частини он-лайн платформ навчання привів до широкого використання підходу SPA, який передбачає побудову інтерфейсу у вигляді однієї вебсторінки з динамічним оновленням контенту без повного перезавантаження. Такий підхід забезпечує більш плавний та швидкий користувацький досвід, що є критично важливим для освітніх систем, де користувач постійно взаємодіє з навчальним матеріалом. SPA-додатки дозволяють ефективно реалізовувати складні інтерфейси, включаючи навігацію між курсами, перегляд уроків, проходження тестів та відображення прогресу [5].

Альтернативою є модульна архітектура, яка передбачає розподіл системи на окремі логічні компоненти або модулі, кожен з яких відповідає за певну частину функціональності. У контексті он-лайн платформ навчання такими модулями можуть виступати підсистеми автентифікації та авторизації користувачів, управління курсами, організації навчального процесу, оцінювання знань та

адміністративного керування. Такий підхід дозволяє ізолювати зміни в межах окремих модулів, підвищує гнучкість системи та спрощує її розвиток [6].

У сучасних системах також широко застосовується API-first підхід, який передбачає проектування серверної частини як набору незалежних сервісів, що надають функціональність через чітко визначені програмні інтерфейси. Це дозволяє використовувати один і той самий backend для різних клієнтських застосунків, зокрема вебверсії, мобільних додатків або інших інтегрованих систем. Такий підхід є особливо актуальним для навчальних платформ, які можуть використовуватися в різних середовищах і потребують універсального доступу до своїх функцій [6].

Не менш важливим аспектом є організація зберігання та обробки даних. Он-лайн платформи навчання оперують значними обсягами інформації, включаючи дані про користувачів, навчальні матеріали, результати тестування та історію навчальної активності. Для ефективної роботи системи необхідно використовувати реляційні або нереляційні бази даних, а також забезпечувати оптимізований доступ до інформації через механізми кешування та індексації. Це дозволяє зменшити затримки при обробці запитів і підвищити загальну продуктивність системи [6].

Архітектурні рішення безпосередньо впливають на способи реалізації функціональності системи, однак не менш важливим є підхід до організації самого навчального процесу в межах он-лайн платформи. Саме функціональна структура визначає, яким чином користувач взаємодіє з навчальним контентом, як відбувається засвоєння матеріалу та яким чином оцінюється результат навчання.

Одним із базових підходів до організації навчання є курсо-орієнтована модель. У межах цього підходу навчальний процес структурується у вигляді окремих курсів, кожен з яких присвячений певній тематиці або набору компетенцій. Така організація дозволяє логічно групувати навчальний матеріал, забезпечує послідовність його вивчення та спрощує навігацію для користувача. Курси можуть відрізнятися за рівнем складності, тривалістю, форматом подання матеріалу та цільовою аудиторією, що дає змогу формувати гнучкі освітні траєкторії [7].

Подальша деталізація курсу здійснюється через поділ на модулі та окремі навчальні одиниці, які часто називають уроками. Модуль представляє собою логічно завершений блок матеріалу, що охоплює певний підрозділ теми, тоді як урок є мінімальною структурною одиницею, яка містить конкретний навчальний контент. Такий підхід забезпечує поетапне засвоєння знань, дозволяє користувачеві поступово просуватися у навчанні та контролювати власний прогрес. Крім того, модульність сприяє повторному використанню навчальних матеріалів та спрощує їх оновлення [7].

Важливим компонентом функціональної організації навчальних платформ є система оцінювання знань. У більшості сучасних систем вона реалізується у вигляді тестування, яке дозволяє автоматично перевіряти рівень засвоєння матеріалу. Тести можуть містити різні типи завдань, включаючи вибір однієї або декількох правильних відповідей, відкриті питання, а також завдання на відповідність або впорядкування. Автоматизація процесу оцінювання значно підвищує ефективність навчання, дозволяє швидко отримувати результати та забезпечує об'єктивність перевірки [8].

Не менш важливою складовою є механізм відстеження навчального прогресу користувача. Сучасні он-лайн платформи фіксують інформацію про виконані завдання, переглянуті уроки, результати тестування та загальний рівень проходження курсу. На основі цих даних формується індивідуальний профіль навчальної активності, який дозволяє користувачеві оцінити власні досягнення, а системі - адаптувати подальший навчальний процес. Відображення прогресу у вигляді відсотків виконання, статусів проходження або інших індикаторів підвищує мотивацію користувачів і сприяє завершенню курсів [9].

Окрім базових функцій, сучасні платформи часто реалізують додаткові механізми взаємодії з користувачем, такі як сповіщення, рекомендації щодо подальшого навчання, а також можливість повернення до попереднього матеріалу. Це дозволяє створити більш гнучке та зручне навчальне середовище, яке враховує

індивідуальні особливості користувача. Важливою характеристикою таких систем є також підтримка асинхронного навчання, коли користувач самостійно визначає темп проходження курсу, що особливо актуально для дорослих користувачів та професійної підготовки [10].

Функціональна організація он-лайн платформ навчання також передбачає наявність різних ролей користувачів, кожна з яких має власний набір можливостей. Зокрема, можна виділити звичайних користувачів (студентів), які проходять навчання, та адміністраторів або викладачів, які створюють і редагують навчальний контент, керують курсами та аналізують результати навчання. Розмежування функцій між ролями дозволяє ефективно організувати процес управління платформою та забезпечити контроль якості навчального матеріалу [10].

З огляду на різноманітність користувачів он-лайн платформ навчання, їх рівень підготовки, інтереси та цілі, важливим напрямком розвитку таких систем є забезпечення персоналізації навчального процесу. На відміну від традиційних підходів, де навчання здійснюється за єдиною програмою для всіх учасників, сучасні цифрові платформи дозволяють адаптувати зміст і послідовність навчання відповідно до індивідуальних характеристик користувача.

Початкові підходи до персоналізації базуються на використанні фільтрації навчального контенту за заданими параметрами. Зокрема, користувачам можуть пропонуватися курси відповідно до обраної тематики, рівня складності або ключових слів. Такий підхід є відносно простим у реалізації, проте він не враховує динаміку навчального процесу та реальну поведінку користувача під час взаємодії з платформою. Подальший розвиток персоналізації пов'язаний із впровадженням адаптивного навчання, яке передбачає зміну структури або змісту навчального матеріалу залежно від результатів користувача. У цьому випадку система аналізує інформацію про проходження курсів, результати тестування, час взаємодії з контентом та інші показники, на основі яких формуються рекомендації щодо подальшого навчання. Наприклад, користувачеві можуть бути запропоновані

додаткові матеріали у випадку низьких результатів або, навпаки, більш складні теми при високому рівні засвоєння матеріалу.

Більш складним і ефективним підходом є використання рекомендаційних систем, які дозволяють автоматично підбирати навчальний контент для користувача. У загальному випадку можна виділити два основних типи таких систем: контентно-орієнтовані та колаборативні. Контентно-орієнтовані системи формують рекомендації на основі характеристик навчального матеріалу та попереднього досвіду користувача, тоді як колаборативні підходи використовують інформацію про поведінку інших користувачів зі схожими інтересами. Кожен із цих підходів має свої переваги та обмеження, тому на практиці часто застосовуються їх комбіновані варіанти [11].

У сучасних он-лайн платформах навчання дедалі більшого поширення набувають методи, що базуються на використанні штучного інтелекту та машинного навчання. Зокрема, для представлення навчального контенту та профілю користувача можуть використовуватися векторні моделі, які дозволяють формалізувати текстову інформацію у вигляді числових векторів. Такий підхід відкриває можливість обчислення семантичної схожості між різними об'єктами, наприклад між описом курсу та інтересами користувача, що значно підвищує якість рекомендацій [11].

Одним із ключових інструментів реалізації такого підходу є використання ембеддінгів (embeddings) - числових представлень тексту, які відображають його зміст у багатовимірному просторі. На основі цих представлень можна визначати ступінь близькості між різними навчальними матеріалами або між користувачем і курсом. Це дозволяє будувати рекомендаційні системи, які враховують не лише формальні характеристики контенту, але й його семантичний зміст [12].

Застосування методів штучного інтелекту в он-лайн платформах навчання не обмежується лише рекомендаційними системами. Вони також можуть використовуватися для аналізу поведінки користувачів, виявлення проблемних тем,

прогнозування результатів навчання та автоматизації створення навчального контенту. Наприклад, система може визначати теми, які викликають труднощі у користувачів, та пропонувати додаткові пояснення або альтернативні навчальні матеріали [12].

1.2 Аналіз аналогів он-лайн платформ навчання

У сучасних умовах цифровізації освіти он-лайн платформи навчання набули широкого поширення та застосовуються як у цивільній, так і у спеціалізованих сферах діяльності. Найбільш відомими є універсальні освітні платформи, такі як Coursera [13] або Prometheus [14], які забезпечують доступ до великої кількості курсів різного спрямування. Однак подібні системи орієнтовані на масову аудиторію та не враховують специфіку підготовки у вузькоспеціалізованих галузях, зокрема у сфері військової підготовки.

На відміну від універсальних рішень, у військовій сфері використовуються спеціалізовані он-лайн платформи, що призначені для підготовки військовослужбовців, підвищення їх кваліфікації та підтримки безперервного професійного навчання. Такі системи, як правило, мають обмежений доступ та функціонують у закритому середовищі, що ускладнює їх детальне дослідження. У зв'язку з цим аналіз базується переважно на відкритих джерелах, офіційних описах та загальнодоступних матеріалах.

Одним із найбільш відомих прикладів спеціалізованих он-лайн платформ військового навчання є система Joint Knowledge Online (JKO) [15], яка використовується Міністерством оборони США для підготовки військовослужбовців та цивільного персоналу. Дана платформа призначена для забезпечення безперервного навчання, підвищення кваліфікації та стандартизації знань у різних підрозділах. Основною функціональною можливістю JKO є надання доступу до великого каталогу навчальних курсів, що охоплюють широкий спектр тем. Користувачі можуть проходити курси у зручний для себе час, що забезпечує

гнучкість навчального процесу. Платформа також підтримує механізми відстеження прогресу, збереження результатів навчання та формування звітності, що дозволяє контролювати рівень підготовки персоналу.

На рис. 1.1 зображено головну сторінку он-лайн платформи ЖКО.



Рис. 1.1. Головна сторінка платформи ЖКО

До переваг даної системи належать її масштабованість та централізований характер. ЖКО забезпечує єдину точку доступу до навчальних матеріалів для великої кількості користувачів, що є важливим для організації навчання у масштабах збройних сил. Крім того, платформа сприяє уніфікації навчального контенту та стандартів підготовки.

Разом з тим, аналіз функціональних можливостей ЖКО дозволяє виявити ряд обмежень. Зокрема, система орієнтована переважно на надання стандартизованих курсів і не забезпечує достатнього рівня персоналізації навчального процесу. Вибір курсів здебільшого здійснюється користувачем або визначається службовими вимогами, без використання інтелектуальних рекомендаційних механізмів. Також відсутні засоби адаптивного формування індивідуальної траєкторії навчання на основі попередніх результатів користувача.

Ще одним прикладом спеціалізованої платформи військового навчання є NATO DEEP eAcademy [16], яка функціонує в межах програми НАТО з удосконалення оборонної освіти (Defence Education Enhancement Programme). Дана

платформа орієнтована на розвиток професійної військової освіти, підтримку навчальних закладів та обмін знаннями між країнами-партнерами. Основною метою NATO DEEP eAcademy є забезпечення доступу до навчальних матеріалів, курсів та методичних ресурсів для військових фахівців і викладачів. Платформа виконує роль централізованого освітнього середовища, що об'єднує навчальний контент, інструменти дистанційного навчання та ресурси для підвищення кваліфікації. Вона також сприяє інтеграції різних освітніх підходів, включаючи дистанційне та змішане навчання.

На рис. 1.2 зображено головну сторінку он-лайн платформи NATO DEEP eAcademy.

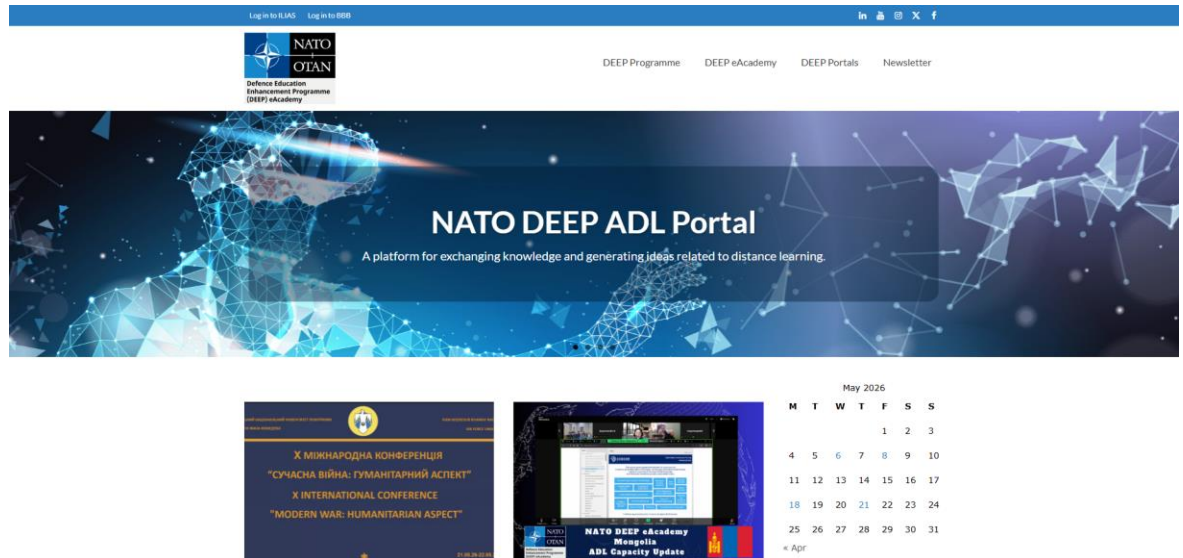


Рис. 1.2. Головна сторінка платформи NATO DEEP eAcademy

Серед ключових функціональних можливостей платформи можна виділити доступ до електронних курсів, навчальних програм, бібліотек матеріалів, а також підтримку взаємодії між учасниками навчального процесу. Особливістю даного рішення є орієнтація не лише на кінцевих користувачів, а й на викладачів та організаторів навчання, що дозволяє використовувати платформу як інструмент для розвитку освітніх програм у військовій сфері.

До переваг NATO DEEP eAcademy слід віднести її спрямованість на міжнародну співпрацю та розвиток системи професійної військової освіти. Платформа забезпечує єдине середовище для обміну досвідом, стандартизації підходів до навчання та підвищення якості освітніх програм.

Водночас, як і у випадку з іншими військовими платформами, дане рішення має певні обмеження. Зокрема, воно орієнтоване переважно на організацію навчального процесу та поширення освітніх матеріалів, а не на індивідуалізацію навчання. Механізми персоналізації та адаптивного підбору курсів для користувачів практично відсутні, а використання технологій штучного інтелекту в процесі навчання є обмеженим.

Прикладом корпоративної системи дистанційного навчання у військовій сфері є Defence Learning Network (DLN) [17], яка використовується Міністерством національної оборони Канади та Збройними силами Канади. Дана платформа призначена для організації навчального процесу, управління освітніми ресурсами та підтримки безперервного професійного розвитку військовослужбовців.

На рис. 1.3 зображено головну сторінку он-лайн платформи Defence Learning Network.

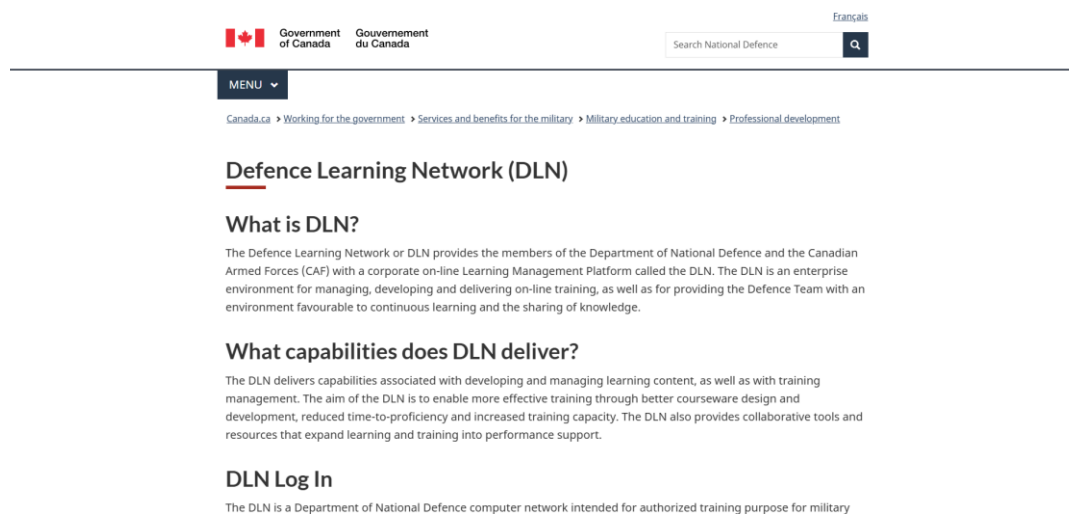


Рис. 1.3. Головна сторінка платформи Defence Learning Network

DLN реалізує класичну модель системи управління навчанням (Learning Management System, LMS), що забезпечує доступ до навчальних курсів, програм

підготовки та освітніх матеріалів. Користувачі мають можливість проходити курси дистанційно, отримувати навчальні завдання, складати тести та відслідковувати власний прогрес. Система також підтримує розмежування доступу відповідно до ролей користувачів, що дозволяє ефективно управляти навчальним процесом у межах організаційної структури. До основних функціональних можливостей DLN належать централізоване управління курсами, облік результатів навчання, формування звітності, а також інтеграція з внутрішніми системами управління персоналом. Це дозволяє використовувати платформу не лише як інструмент навчання, а й як складову частину загальної системи управління людськими ресурсами у військовій організації.

Серед переваг даної платформи слід відзначити її високу інтегрованість у структуру збройних сил, можливість контролю навчального процесу та забезпечення відповідності навчання встановленим стандартам. DLN дозволяє ефективно організувати навчання великої кількості користувачів та забезпечити централізований доступ до освітніх ресурсів.

Разом з тим, аналіз функціональних можливостей DLN виявляє певні обмеження, характерні для класичних LMS-рішень. Зокрема, система не передбачає використання інтелектуальних механізмів для аналізу поведінки користувачів та формування персоналізованих рекомендацій. Навчальний процес має переважно статичний характер, а вибір курсів визначається або користувачем, або адміністративними вимогами, без урахування індивідуальних особливостей навчання.

Ще одним прикладом сучасної військової навчальної платформи є ADELE (Australian Defence Education Learning Environment) [18], яка використовується Збройними силами Австралії для організації он-лайн та змішаного навчання. Дана система орієнтована на підтримку як дистанційної підготовки військовослужбовців, так і взаємодії між викладачами та слухачами у межах навчального процесу.

На рис. 1.4 зображено головну сторінку он-лайн платформи Australian Defence Education Learning Environment.

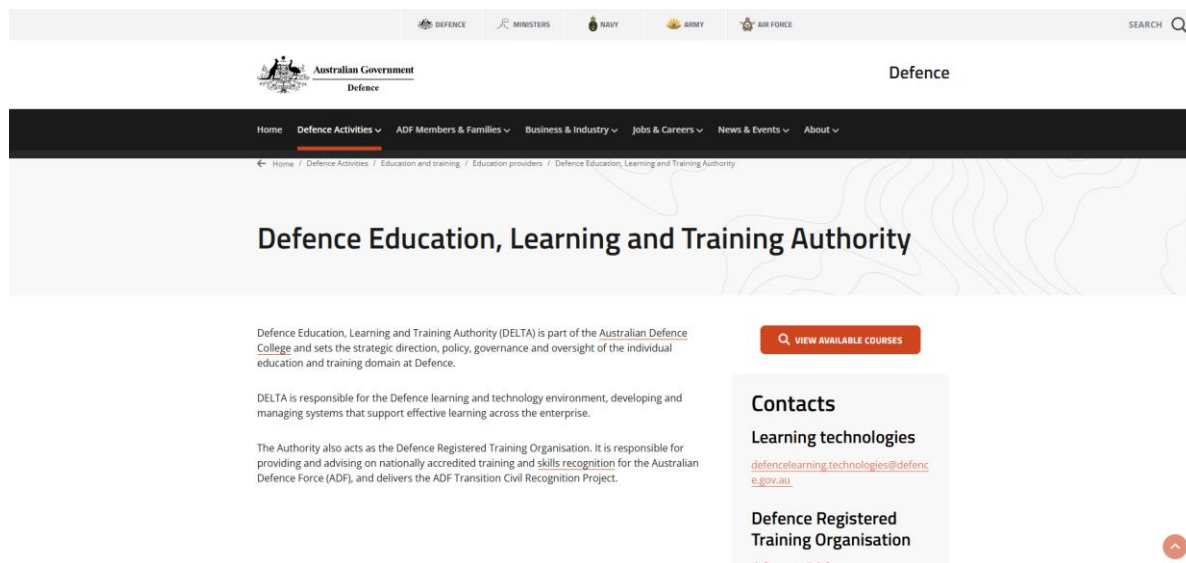


Рис. 1.4. Головна сторінка платформи Australian Defence Education Learning Environment

Платформа ADELE реалізує функціональність системи управління навчанням із розширеними можливостями для створення та адміністрування навчального контенту. Вона забезпечує доступ до курсів, навчальних матеріалів, тестів та інших освітніх ресурсів, а також підтримує механізми комунікації між учасниками навчального процесу. Особливістю даної системи є орієнтація на змішане навчання, що поєднує он-лайн формат із традиційними формами підготовки.

До основних переваг ADELE можна віднести гнучкість у створенні та організації навчального контенту, можливість адаптації курсів під потреби різних категорій користувачів, а також підтримку інструкторів у процесі розробки навчальних програм. Це дозволяє використовувати платформу не лише як інструмент для проходження курсів, а й як середовище для їх створення та модифікації.

Водночас, незважаючи на більш сучасний підхід до організації навчання, ADELE, як і інші розглянуті системи, базується на класичній LMS-архітектурі. Вибір навчального контенту здійснюється без використання інтелектуальних

алгоритмів аналізу даних, а механізми персоналізації навчання є обмеженими. Платформа не передбачає автоматизованого формування індивідуальної траєкторії навчання на основі рівня знань або попереднього досвіду користувача.

Проведений аналіз спеціалізованих он-лайн платформ військового навчання дозволяє зробити висновок, що сучасні рішення орієнтовані насамперед на організацію централізованого навчального процесу, забезпечення доступу до навчальних матеріалів та контроль результатів підготовки користувачів. Розглянуті системи, зокрема Joint Knowledge Online (JKO), NATO DEEP eAcademy, Defence Learning Network (DLN) та ADELE, демонструють подібні підходи до побудови навчальних платформ.

До спільних характеристик таких систем можна віднести наявність структурованого каталогу курсів, механізмів управління навчальним контентом, відстеження прогресу користувачів та формування звітності. Більшість платформ реалізують класичну модель систем управління навчанням (LMS), що забезпечує стандартизований підхід до підготовки військовослужбовців і дозволяє ефективно організувати навчання великої кількості користувачів.

Разом з тим, аналіз показав, що існуючі рішення мають ряд суттєвих обмежень. Зокрема, у більшості платформ відсутні механізми гнучкої персоналізації навчального процесу. Навчальний контент, як правило, є однаковим для всіх користувачів, незалежно від їх рівня підготовки, попереднього досвіду чи індивідуальних потреб. Вибір курсів здійснюється вручну або визначається адміністративними вимогами, що знижує ефективність навчання.

Крім того, у розглянутих системах практично не використовуються сучасні методи аналізу даних та технології штучного інтелекту для підтримки прийняття рішень у навчальному процесі. Відсутність рекомендаційних механізмів не дозволяє формувати індивідуальні траєкторії навчання, що є важливим фактором підвищення ефективності підготовки у складних та динамічних умовах військової діяльності.

Табл. 1.1 ілюструє переваги та недоліки кожної з платформ на основі їхнього функціоналу та користувацького досвіду.

Таблиця 1.1

Характеристика	JKO	NATO DEEP eAcademy	DLN	ADELE
Перелік активних та майбутніх курсів	+	+	+	+
Докладний опис курсу	+	+	+	+
Коректна робота сторонніх API	-	-	-	-
Коректна робота сповіщень	-	+	+	+
Необов'язкова авторизація для запису на курс	+	-	-	-
Якість мультимедійного контенту	-	+	+	+
Адаптивність до мобільних пристроїв	-	+	+	+
Наявність персоналізованих рекомендацій	-	-	-	-
Інтерфейс і зручність навігації	+	+	+	+
Наявність зворотного зв'язку	-	+	+	+
Відстеження прогресу навчання	+	+	+	+
Інтеграція з внутрішніми системами	+	+	+	+
Використання технологій ІІІ	-	-	-	-

Продовження табл. 1.1

Підтримка змішаного навчання	-	+	-	+
------------------------------------	---	---	---	---

Таким чином, можна зробити висновок, що існуючі військові он-лайн платформи ефективно вирішують задачі організації навчання та контролю результатів, однак не забезпечують достатнього рівня адаптивності та інтелектуалізації навчального процесу. Це створює передумови для розробки нових підходів до побудови навчальних систем, які поєднують традиційні можливості LMS із сучасними методами штучного інтелекту.

З урахуванням виявлених недоліків виникає необхідність створення он-лайн платформи військової підготовки, яка забезпечує персоналізований підбір навчального контенту, адаптацію до рівня підготовки користувача та використання рекомендаційних алгоритмів на основі аналізу даних. Саме формулювання цих вимог та постановка задачі дослідження розглядаються у наступному підрозділі.

1.3 Постановка задачі та вимоги до платформи військової підготовки

Проведений аналіз існуючих он-лайн платформ навчання, зокрема спеціалізованих військових рішень, показав, що сучасні системи здебільшого орієнтовані на організацію доступу до навчального контенту та контроль результатів підготовки користувачів. Більшість таких платформ надає стандартний набір можливостей, що включає розміщення навчальних матеріалів, проходження тестування та моніторинг успішності слухачів. Подібний підхід дозволяє автоматизувати окремі елементи освітнього процесу, проте не забезпечує достатнього рівня персоналізації навчання.

Для сфери військової підготовки ця проблема має особливе значення. На відміну від багатьох інших напрямів освіти, військове навчання охоплює широкий спектр дисциплін та передбачає підготовку користувачів із різним рівнем знань, практичного досвіду та професійної спеціалізації. Одні користувачі можуть

потребувати поглибленого вивчення тактичної медицини, інші – військової топографії, організації зв'язку або тактики дій підрозділів. За таких умов використання однакового набору навчальних матеріалів для всіх категорій слухачів не завжди забезпечує необхідний рівень ефективності навчального процесу.

Ключовою проблемою сучасних військових он-лайн платформ є відсутність механізмів адаптивного формування навчального процесу. У більшості випадків користувач отримує доступ до заздалегідь визначеної структури курсів, яка не змінюється залежно від його успішності, навчальної активності або попереднього досвіду. Навчальний контент подається у фіксованій послідовності, а система фактично не бере участі у виборі подальшої траєкторії навчання. У результаті користувач змушений самотійно визначати, які курси можуть бути корисними для його подальшого розвитку, що не завжди дозволяє обрати найбільш релевантні навчальні матеріали.

Окремої уваги потребує питання використання даних, які накопичуються в процесі роботи навчальної системи. Під час проходження курсів формується значний обсяг інформації про користувача, включаючи результати тестування, історію перегляду навчальних матеріалів, відомості про завершені курси та рівень активності в системі. У багатьох існуючих рішеннях ці дані використовуються виключно для формування статистичної звітності або контролю успішності. Водночас вони можуть слугувати основою для побудови інтелектуальних механізмів підтримки навчального процесу.

Розвиток технологій штучного інтелекту та методів аналізу даних відкриває нові можливості для вдосконалення систем дистанційного навчання. Особливий інтерес становлять підходи, пов'язані із семантичним аналізом текстової інформації та використанням векторних представлень даних. Такі технології дозволяють аналізувати зміст навчальних матеріалів не лише за наявністю певних ключових слів, а й з урахуванням їхнього смислового наповнення. Це створює можливість

визначати взаємозв'язки між навчальними курсами, інтересами користувача та результатами його навчання.

Одним із перспективних підходів є використання ембеддінгів – векторних представлень текстових даних. За допомогою цієї технології опис навчального курсу або профіль користувача можуть бути представлені у вигляді набору числових параметрів, що відображають їх змістовні характеристики. Порівняння таких векторів дозволяє визначати рівень семантичної близькості між навчальним контентом та потребами користувача. На практиці це дає змогу формувати персоналізовані рекомендації та пропонувати ті курси, які найбільше відповідають його інтересам і поточному рівню підготовки.

З огляду на виявлені недоліки існуючих рішень, у межах даної роботи ставиться задача розробки он-лайн платформи військової підготовки, яка поєднує функціональні можливості сучасних систем управління навчанням із механізмами інтелектуального аналізу даних. Розроблювана система повинна забезпечувати не лише доступ до навчальних матеріалів, але й підтримувати користувача на різних етапах навчання шляхом аналізу його діяльності та формування рекомендацій щодо подальшого проходження курсів.

На відміну від традиційних LMS-рішень, запропонована платформа передбачає використання рекомендаційного механізму, який враховує навчальну історію користувача, результати тестування, рівень проходження курсів та сформований профіль інтересів. Такий підхід дозволяє перейти від статичної моделі навчання до більш гнучкої моделі, орієнтованої на індивідуальні потреби кожного користувача.

Для досягнення поставленої мети необхідно забезпечити реалізацію низки функціональних можливостей системи.

Функціональні вимоги є такими:

- забезпечення реєстрації та автентифікації користувачів;
- підтримка ролей користувачів (студент, адміністратор);

- надання доступу до каталогу навчальних курсів із можливістю пошуку та фільтрації;
- забезпечення проходження навчальних матеріалів та тестування;
- відстеження прогресу користувача та збереження результатів навчання;
- ведення профілю користувача із накопиченням даних про його навчальну активність;
- реалізація рекомендаційної системи для підбору навчальних курсів на основі аналізу даних;
- забезпечення можливості адміністрування курсів, модулів та навчального контенту.

Крім функціональних можливостей, система повинна відповідати низці нефункціональних вимог, які визначають якість її роботи та можливість подальшого розвитку.

Нефункціональні вимоги виглядають таким чином:

- забезпечення масштабованості системи для обслуговування великої кількості користувачів;
- гнучкість та розширюваність програмної архітектури;
- забезпечення захисту персональних даних та контролю доступу до інформації;
- підтримка стабільної роботи системи під навантаженням;
- зручність та інтуїтивність користувацького інтерфейсу;
- мінімізація затримок при обробці запитів та формуванні рекомендацій;
- можливість подальшої інтеграції із зовнішніми сервісами та інтелектуальними модулями.

Отже, розробка он-лайн платформи військової підготовки з використанням сучасних вебтехнологій та елементів штучного інтелекту спрямована на підвищення ефективності навчального процесу шляхом персоналізації навчання та використання накопичених даних про користувача. Реалізація такого підходу

дозволяє створити основу для побудови адаптивної системи навчання, здатної враховувати індивідуальні особливості слухачів та підвищувати якість їх професійної підготовки.

Висновки до першого розділу

У першому розділі було проведено аналіз принципів побудови сучасних он-лайн платформ навчання та досліджено можливості застосування технологій штучного інтелекту в освітніх системах. Розглянуто основні підходи до організації навчального процесу в он-лайн середовищі, зокрема використання систем управління навчанням (LMS), що забезпечують доступ до навчального контенту, контроль проходження курсів та оцінювання результатів користувачів.

У ході аналізу існуючих рішень встановлено, що більшість сучасних спеціалізованих військових платформ, реалізують подібні функціональні можливості, зосереджені на централізованому управлінні навчанням та стандартизації освітнього процесу. Разом з тим, проведений аналіз дозволив виявити суттєві обмеження існуючих систем. Основною проблемою є відсутність адаптивності навчального процесу та недостатнє використання інтелектуальних методів аналізу даних. Більшість платформ використовують статичну модель навчання, яка не враховує індивідуальні особливості користувачів, їх рівень підготовки та попередній досвід. Це знижує ефективність навчання та не дозволяє оптимально формувати освітню траєкторію.

У зв'язку з цим було обґрунтовано доцільність використання сучасних методів штучного інтелекту, зокрема технологій побудови векторних представлень (ембеддінг), для аналізу навчального контенту та профілю користувача. Використання таких підходів дозволяє реалізувати рекомендаційні механізми, що забезпечують підбір найбільш релевантних курсів та формування індивідуальної траєкторії навчання.

На основі проведеного аналізу сформульовано задачу дослідження, яка полягає у проєктуванні он-лайн платформи військової підготовки з елементами штучного інтелекту, що поєднує можливості класичних систем управління навчанням із механізмами інтелектуального аналізу даних. Також визначено основні функціональні та нефункціональні вимоги до розроблюваної системи, які забезпечують її ефективність, гнучкість та масштабованість.

Отримані результати створюють теоретичну основу для подальшого етапу дослідження, що полягає у моделюванні структури он-лайн платформи, визначенні взаємодії її компонентів та розробці алгоритмів рекомендацій на основі аналізу даних, що буде розглянуто у наступному розділі.

РОЗДІЛ 2 МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ ОН-ЛАЙН ПЛАТФОРМИ ВІЙСЬКОВОЇ ПІДГОТОВКИ З ЕЛЕМЕНТАМИ ШТУЧНОГО ІНТЕЛЕКТУ

2.1 Архітектурна модель платформи військової підготовки

Проектування архітектури он-лайн платформи військової підготовки здійснюється з урахуванням вимог до масштабованості, гнучкості та ефективності обробки даних. З огляду на специфіку задачі та необхідність забезпечення інтерактивної взаємодії з користувачем, обрано дворівневу клієнт-серверну архітектуру, яка є одним із найбільш поширених і ефективних підходів до побудови сучасних вебсистем. У межах даного підходу система поділяється на два основні рівні: клієнтський та серверний. Клієнтський рівень відповідає за взаємодію з користувачем, відображення інтерфейсу та обробку подій, тоді як серверний рівень забезпечує виконання бізнес-логіки, обробку запитів та управління даними. Такий розподіл дозволяє ізолювати відповідальність між компонентами системи, спрощує її розвиток та забезпечує можливість незалежного масштабування окремих частин [19].

Одним з ключових архітектурних рішень, які планується впровадити, у межах розроблюваної системи є винесення інтерфейсу моделей штучного інтелекту на клієнтський рівень. Зокрема, операції, пов'язані з формуванням векторних представлень навчального контенту та профілю користувача, а також обчисленням схожості між ними, виконуються безпосередньо у середовищі клієнта. Такий підхід дозволяє зменшити навантаження на сервер, підвищити швидкість обробки даних та забезпечити більшу автономність системи, у разі різкого підвищення кількості користувачів [20].

Крім того, перенесення інтелектуальних обчислень на клієнтський рівень сприяє підвищенню рівня конфіденційності, оскільки частина даних користувача обробляється локально без передачі на сервер. Це є важливим фактором у контексті

навчальних систем, де обробляється інформація про навчальну діяльність та індивідуальні характеристики користувачів.

Загальна структура системи, що проєктується, відображається за допомогою діаграми компонентів, яка демонструє основні складові платформи та взаємозв'язки між ними (рис. 2.1). До ключових компонентів належать клієнтський застосунок, серверна частина та система зберігання даних. Взаємодія між клієнтом і сервером здійснюється через програмний інтерфейс, який забезпечує обмін даними та виклик функціональних можливостей системи. Окремо виділяється компонент, відповідальний за обробку інтелектуальних функцій на стороні клієнта, що підкреслює особливість обраної архітектури.

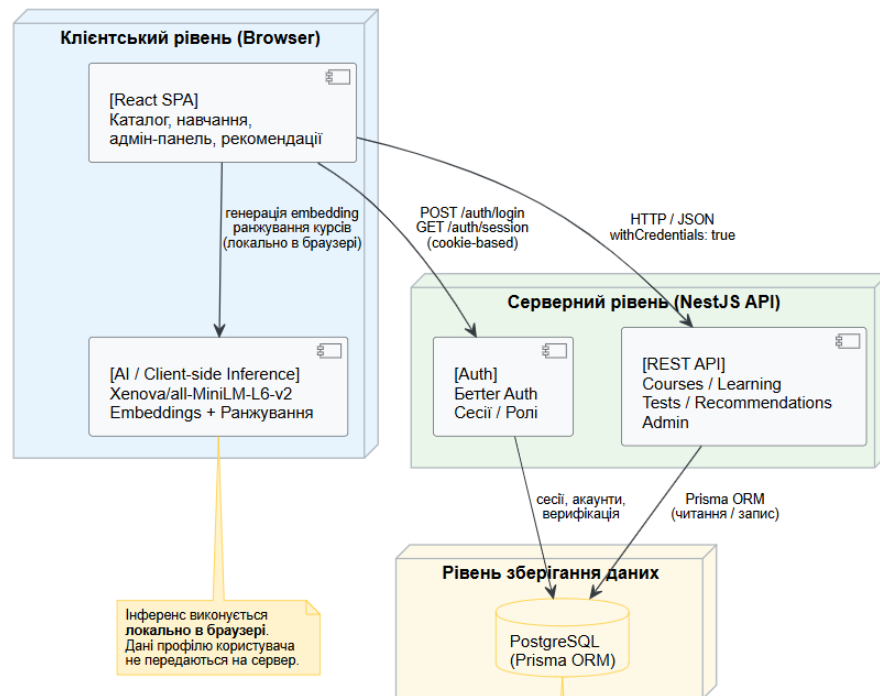


Рис. 2.1. UML-діаграма компонентів архітектури он-лайн платформи

Після визначення загальної архітектурної концепції доцільним є формалізація взаємодії користувачів із системою на рівні сценаріїв використання. Це дозволяє чітко окреслити функціональні межі системи, визначити ролі користувачів та структурувати їх взаємодію з основними компонентами платформи. Для цього використовується діаграма варіантів використання, яка відображає акторів системи та перелік їх можливих дій (Додаток А).

У межах розроблюваної платформи виділено дві основні категорії акторів: Student (користувач, який проходить навчання) та Admin (користувач, який здійснює управління навчальним контентом та системою в цілому).

Актор Student взаємодіє із системою з метою проходження навчання та отримання нових знань. Основні сценарії його використання включають перегляд доступних курсів, вибір та проходження навчального матеріалу, виконання тестових завдань, а також відстеження власного прогресу. Окремим сценарієм є отримання рекомендацій щодо подальшого навчання, що дозволяє користувачеві ефективніше планувати освітню діяльність.

Актор Admin виконує функції управління системою та навчальним контентом. До його основних сценаріїв використання належать створення та редагування курсів, формування структури навчальних матеріалів, управління тестами та контроль загального стану системи. Такий підхід дозволяє централізовано забезпечувати якість контенту та підтримувати актуальність навчальних матеріалів.

Обґрунтування розмежування ролей полягає у необхідності забезпечення безпеки та керованості системи. Розділення доступу дозволяє обмежити можливості користувачів відповідно до їх функцій, що є особливо важливим у навчальних системах, де необхідно забезпечити контроль над створенням та зміною навчального контенту. Крім того, такий підхід сприяє спрощенню інтерфейсу для кінцевого користувача, оскільки кожен актор отримує доступ лише до необхідного набору функцій [21].

Серверна частина он-лайн платформи військової підготовки проєктується з використанням принципу модульної декомпозиції, який передбачає поділ системи на незалежні логічні компоненти, кожен з яких відповідає за окрему функціональну область. Такий підхід забезпечує підвищення гнучкості системи, спрощує її супровід та дозволяє масштабувати окремі частини без впливу на інші компоненти [22].

Обрана архітектура серверної частини складається з набору функціональних модулів, які реалізують основні підсистеми платформи. До ключових модулів належать модуль автентифікації та авторизації, який забезпечує управління доступом користувачів до системи; модуль управління курсами, що відповідає за зберігання та обробку навчального контенту; модуль організації навчального процесу, який забезпечує відстеження прогресу користувачів; а також адміністративний модуль, призначений для управління структурою курсів і контролю функціонування системи.

Кожен із зазначених модулів є відносно автономним і взаємодіє з іншими через чітко визначені інтерфейси. Така організація дозволяє мінімізувати залежності між компонентами та забезпечує можливість незалежного розвитку окремих частин системи. Взаємозв'язки між модулями відображаються за допомогою діаграми залежностей модулів, яка демонструє напрямки взаємодії та дозволяє оцінити рівень зв'язаності компонентів (рис. 2.2).

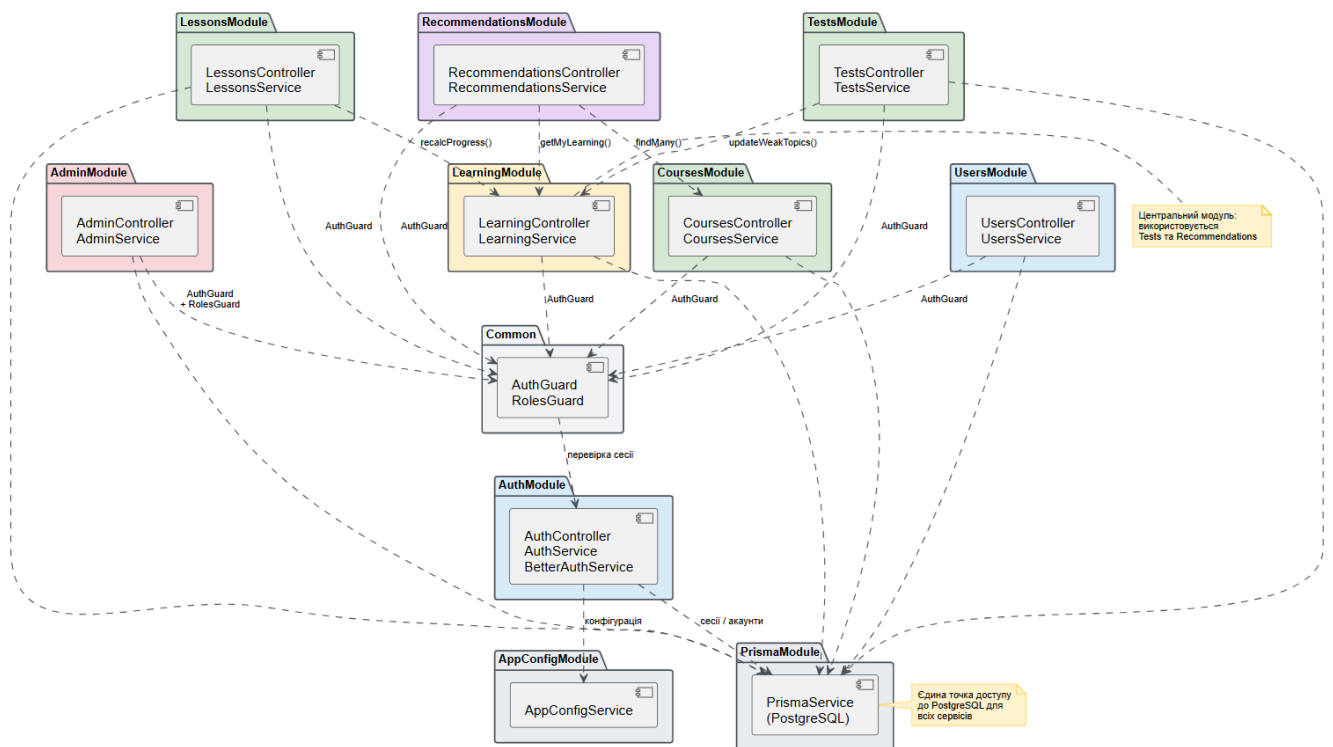


Рис. 2.2. UML-діаграма залежності модулів проектованої системи

У межах кожного модуля використовується багатошарова організація, яка передбачає розподіл функціональності між окремими рівнями. Основними шарами є рівень обробки запитів, рівень бізнес-логіки та рівень доступу до даних. Така структура забезпечує чітке розмежування відповідальностей, спрощує тестування та підвищує зрозумілість системи. Рівень обробки запитів відповідає за прийом та первинну обробку зовнішніх запитів, рівень бізнес-логіки реалізує основні функціональні можливості системи, а рівень доступу до даних забезпечує взаємодію зі сховищем інформації.

Клієнтська частина он-лайн платформи військової підготовки проєктується з врахуванням вимог до зручності використання, масштабованості та підтримки складних інтерактивних сценаріїв. Для цього застосовується підхід декомпозиції за функціональними можливостями: feature-based, який передбачає організацію інтерфейсу у вигляді окремих модулів, що відповідають за конкретні підсистеми.

У межах такого підходу клієнтський застосунок поділяється на функціональні модулі, серед яких можна виділити підсистему автентифікації користувачів, модуль роботи з курсами, підсистему навчального процесу, адміністративний інтерфейс, а також модуль рекомендацій. Кожен із цих модулів інкапсулює відповідну частину функціональності та відповідає за взаємодію з користувачем у межах визначеного набору задач [23].

Окрім функціональних модулів, архітектура клієнтської частини передбачає виділення окремого інфраструктурного рівня, який забезпечує уніфіковану взаємодію між компонентами застосунку та серверною частиною. Цей рівень є API-шаром, який інкапсулює всю логіку роботи з віддаленими ресурсами та виступає абстракцією над механізмами HTTP-взаємодії. API-шар реалізує централізовану обробку запитів до серверної частини, включаючи формування HTTP-запитів, обробку відповідей, нормалізацію помилок та управління параметрами запитів [24]. Таким чином, платформа буде мати ізольовану мережеву логіку від компонентів інтерфейсу, забезпечуючи повторне використання запитів та спрощуючи підтримку

змін у контрактах взаємодії між клієнтом і сервером. Додатково, інтеграція механізмів кешування та синхронізації стану дозволяє оптимізувати доступ до даних та мінімізувати кількість надлишкових запитів.

Паралельно з API-шаром формується рівень спільних типів даних, який забезпечує узгодженість структури даних у межах усього клієнтського застосунку. Використання централізованих типів дозволяє гарантувати коректність обробки даних, отриманих із серверної частини, та зменшує ризик виникнення помилок при зміні контрактів API.

Особливістю архітектури клієнтської частини є застосування стратегії відкладеного завантаження окремих функціональних модулів. Зокрема, модуль, відповідальний за інтелектуальні функції платформи, завантажується лише за необхідності, що дозволяє оптимізувати початковий час завантаження застосунку та зменшити навантаження на клієнтський пристрій. Такий підхід є особливо доцільним у випадках, коли певні функції використовуються не постійно, а лише в окремих сценаріях взаємодії.

Архітектурна модель платформи військової підготовки повинна спиратися на цілісну та логічно узгоджену модель даних, яка забезпечує збереження інформації про користувачів, навчальний контент та результати навчальної діяльності. З цією метою на етапі проєктування була сформована ER-діаграма, що відображає основні сутності системи, їх атрибути та зв'язки між ними (Додаток Б). Такий підхід дозволяє формалізувати структуру предметної області та створити основу для побудови узгодженої серверної частини системи.

З погляду логічної організації даних усі сутності доцільно об'єднати у три основні групи. До першої групи належать сутності, пов'язані з користувачами та розмежуванням доступу. Вона охоплює дані про облікові записи користувачів, їх ролі та службові атрибути, необхідні для ідентифікації та контролю доступу до функціональності системи. Друга група об'єднує сутності навчального контенту. До неї входять курси як базові одиниці організації навчання, їх структурні складові,

а також елементи контролю знань. У межах такої моделі курс розглядається як центральна сутність, з якою пов'язані модулі, уроки та засоби оцінювання. Третя група охоплює сутності, що відображають результати взаємодії користувача з навчальним контентом. Вона включає дані про проходження курсів, фіксацію завершення окремих етапів навчання, а також результати оцінювання. Сутності у межах зазначених груп поєднані відношеннями типу «один-до-багатьох» та «багато-до-одного», що відображають природну структуру предметної області. Один користувач може взаємодіяти з багатьма курсами, один курс може містити декілька структурних елементів, а результати навчальної активності формуються окремо для кожного користувача та кожної освітньої одиниці.

Окремої уваги потребує зберігання векторних представлень навчального контенту, що використовуються для реалізації інтелектуального ранжування курсів. У межах обраної моделі ембедінги доцільно пов'язувати безпосередньо з сутністю курсу, оскільки саме курс виступає цілісною одиницею рекомендації. Таке рішення дозволяє зберігати семантичне представлення узагальненого змісту курсу на рівні його метаданих без ускладнення структури даних додатковими аналітичними сутностями. Крім того, це спрощує подальше використання векторних ознак у рекомендаційних механізмах і не порушує загальну логіку предметної моделі.

Побудова он-лайн платформи військової підготовки вимагає врахування вимог до безпеки доступу та контролю використання функціональних можливостей системи. У межах архітектурної моделі ці вимоги реалізуються через поєднання механізмів автентифікації та авторизації, що забезпечують ідентифікацію користувачів і визначення їх прав доступу.

На концептуальному рівні система використовує дворівневу модель захисту:

1-й рівень – автентифікація, відповідає за підтвердження особи користувача під час взаємодії з платформою;

2-й рівень – авторизація, визначає перелік доступних функцій відповідно до ролі користувача в системі.

Розмежування доступу в системі здійснюється на основі ролей користувачів, визначених на етапі проєктування. Для кожної ролі формується окремий набір дозволених дій, що відображає її функціональне призначення. Це дозволяє обмежити доступ до критичних операцій, пов'язаних із управлінням навчальним контентом або конфігурацією системи, та водночас забезпечити користувачам необхідні можливості для проходження навчання.

На рівні архітектури клієнтської частини розмежування доступу проявляється у структурі маршрутів застосунку. Усі маршрути умовно поділяються на три категорії: публічні, доступні без попередньої автентифікації; маршрути, доступні лише авторизованим користувачам; та маршрути з розширеними правами доступу, призначені для адміністрування системи. Такий підхід дозволяє ще на рівні навігації забезпечити базовий контроль доступу до функціональності платформи.

Важливим архітектурним рішенням є використання сесійного підходу до управління станом автентифікації користувача. У межах клієнт-серверної взаємодії стан сесії зберігається у вигляді cookie, що автоматично передається під час виконання запитів до серверної частини.

2.2 Проєктування підсистем управління навчальним процесом

Ефективність функціонування он-лайн платформи військової підготовки безпосередньо залежить від науково обґрунтованого проєктування логічної структури взаємодії користувача з навчальним контентом. На етапі проєктування підсистем управління навчанням основна увага приділяється розробці моделей, що забезпечують безперервність освітнього процесу, точність відстеження результатів та гнучкість адміністрування в умовах ієрархічної структури військових знань.

Функціональна модель життєвого циклу навчання розробляється як базовий механізм керування станами взаємодії курсанта з платформою. У межах

проектування пропонується формалізувати процес навчання як модель станів та переходів, що дозволяє чітко визначити логіку переходів між етапами засвоєння матеріалу. Модель передбачає три детерміновані стани курсу для кожного профілю користувача: не розпочато – стан за замовчуванням для всіх доступних у каталозі курсів; у процесі – активна фаза навчання, що ініціюється після реєстрації користувача на курсі; завершено – фінальний стан, що досягається при виконанні встановлених критеріїв проходження всіх структурних одиниць курсу.

Проектне рішення щодо логіки переходів між цими станами, а також подій, що зумовлюють зміну статусу, представлено на діаграмі станів (Додаток В).

При проектуванні моделі навчального кроку закладається принцип атомарності: кожна дія курсанта: перегляд лекції чи вивчення блоку контенту, повинна верифікуватися системою [25]. Для кількісного оцінювання динаміки навчання буде використано алгоритм розрахунку прогресу. Математично рівень прогресу P для курсу C пропонується обчислювати як відношення кількості верифікованих пройдених уроків L_{comp} до загальної кількості уроків L_{total} , що визначені в структурі курсу:

$$P = \frac{L_{comp}}{L_{total}} \times 100\%.$$

Така модель забезпечує цілісність та інваріантність даних: навіть у разі переривання сесії або зміни пристрою, система дозволяє курсанту миттєво відновити навчання. Це реалізується шляхом збереження ідентифікатора останнього успішно завершеного або відвіданого уроку у профілі прогресу.

Логічним продовженням розробленої моделі життєвого циклу, що забезпечує перехід курсу до фінального стану «Завершено», є проектування підсистеми тестування та оцінювання знань. Ця підсистема розглядається як критичний контролюючий механізм, що дозволяє об'єктивно верифікувати рівень засвоєння навчального матеріалу перед наданням доступу до наступних етапів підготовки.

У межах проектування пропонується ієрархічна модель структури тестових завдань, де кожен модуль курсу завершується підсумковим тестом, що складається з множини питань та варіантів відповідей. Проектне рішення передбачає зберігання правильних відповідей безпосередньо в базі даних, що дозволяє проводити автоматизовану перевірку на стороні сервера без залучення інструктора. Процес взаємодії компонентів системи під час ініціалізації тесту, обробки відповідей та збереження результатів представлено на діаграмі послідовності (Додаток Д).

Алгоритм оцінювання знань проектується на основі аналізу об'єкта передачі даних. При отриманні відповідей від клієнтської частини, серверна підсистема TestsModule має виконувати компаративний аналіз надісланих ідентифікаторів із правильними значеннями в БД. Підсумковий бал S пропонується розраховувати за формулою:

$$S = \frac{N_{correct}}{N_{total}} \times 100\%,$$

де $N_{correct}$ – кількість правильних відповідей, а N_{total} – загальна кількість питань у тесті. Результат тестування фіксується в сутності TestResult, де, окрім балу, передбачено збереження повного логу відповідей у форматі JSON для подальшого ретроспективного аналізу.

Важливою науково-практичною особливістю проектування цієї підсистеми є реалізація механізму інтелектуального зворотного зв'язку. Пропонується, щоб підсистема не лише констатувала факт проходження тесту, а й ідентифікувала зони низької компетенції. Якщо під час тестування курсант припускається помилок у питаннях, що належать до певного тематичного тегу, ці дані мають автоматично трансформуватися в об'єкт слабких тем у профілі користувача.

Для забезпечення функціонування описаних механізмів навчання та контролю знань необхідним є проектування моделі управління контентом та адміністрування. Ця підсистема розробляється як інструментарій для створення структурованого освітнього середовища, що відповідає суворим вимогам до послідовності та повноти викладу військових дисциплін. В основу архітектури

контенту покладається ієрархічна модель сутностей, яка проєктується для забезпечення цілісності навчальних планів. Пропонована логічна структура передбачає декомпозицію курсу на модулі, кожен з яких є автономною навчальною одиницею, що містить набір уроків та фінальний тест. Детальна схема взаємозв'язків між сутностями Course, Module, Lesson та Test, а також їхні атрибути та типи відношень, представлені на діаграмі класів (Додаток Е). Таке проєктування дозволяє реалізувати гнучке управління контентом, де кожен елемент є частиною єдиного графа знань.

Особливе значення при проєктуванні адміністративної підсистеми надається підготовці даних для інтелектуального аналізу. На відміну від традиційних систем управління навчанням, у даному проєкті передбачено розширення моделі курсу специфічними AI-полями. Проєктне рішення включає впровадження атрибута `embeddingText` - семантичного опису курсу, який слугує основою для генерації векторного представлення.

Процес адміністрування проєктується таким чином, щоб мінімізувати навантаження на серверні потужності при збереженні високої точності рекомендацій. Для цього в інтерфейс адміністратора закладається механізм локального інференсу: при створенні або редагуванні курсу система ініціює генерацію ембеддінгу безпосередньо в браузері за допомогою моделі `Xenova/all-MiniLM-L6-v2` [26]. Згенерований вектор разом із метаданими про модель та розмірність передається на сервер для збереження в базі даних. Це дозволяє на етапі проєктування вирішити проблему швидкодії: під час ранжування курсів для користувача система оперуватиме вже готовими математичними векторами, уникаючи ресурсомістких обчислень у реальному часі.

Для забезпечення ефективної роботи платформи, що передбачає великий обсяг навчальних матеріалів та значну кількість користувачів, критично важливим є проєктування моделі оптимізації взаємодії з даними. Ця підсистема спрямована на мінімізацію навантаження на сервер, прискорення завантаження контенту та

покращення загального користувацького досвіду. У межах проєктування передбачено використання стратегії пагінації для відображення списків курсів та інших даних, що мають великий обсяг. Для цього, при реалізації API-ендпоінтів, використовується механізм передачі даних про загальну кількість записів, поточну сторінку та ліміт записів на сторінці. Такий підхід дозволяє уникнути перевантаження тіла відповіді, що позитивно впливає на швидкість завантаження сторінок.

На рівні клієнтської частини, для оптимізації взаємодії з API та зменшення кількості повторних запитів, планується використання моделі кешування серверного стану. Подібні інструменти мають вбудований механізм визначення параметрів життєвого циклу кешу `staleTime`, що дозволяє уникнути надлишкових запитів до API при навігації між сторінками та забезпечує швидкий доступ до даних, що часто використовуються. Для забезпечення актуальності даних, передбачено механізм автоматичного оновлення кешу при зміні даних на сервері.

Також, з метою покращення користувацького досвіду та зниження навантаження на базу даних, планується впровадження алгоритму «дебаунсингу» для пошукових запитів. Запит до API для фільтрації курсів ініціюється лише після паузи у введенні тексту користувачем. Це дозволяє уникнути надмірної кількості запитів під час введення пошукового терміну та забезпечує більш плавний інтерфейс.

Крім того, для оптимізації взаємодії з даними, передбачено використання механізмів лінивого завантаження для компонентів, що відображають великі обсяги даних, наприклад, для сторінки рекомендацій. Це дозволить завантажувати компоненти та дані лише тоді, коли вони стають видимими для користувача, що зменшує час початкового завантаження сторінки.

Завершальним етапом розробки логічної структури системи є формалізація профілю компетенцій користувача. У межах проєктування цей профіль розглядається не як статичний набір анкетних даних, а як динамічна інформаційна

модель, що акумулює результати навчальної діяльності та виявлені когнітивні прогалини військовослужбовця. Проєктне рішення передбачає структурування профілю компетенцій як багатокомпонентного об'єкта, що складається з двох основних блоків: дескриптивного та аналітичного. Дескриптивний блок включає короткий текстовий опис користувача. Аналітичний блок проєктується як результат роботи підсистем навчання та тестування, включаючи поточний статус проходження курсів та динамічний список слабких тем.

Науково-технічна новизна такого підходу до проєктування полягає у створенні механізму семантичної агрегації даних. Пропонується формалізувати стан компетенцій користувача через побудову інтегрованого текстового опису. Цей об'єкт формується шляхом конкатенації ключових маркерів профілю користувача, зокрема переліку пріоритетних напрямів підготовки, тем, за якими за результатами тестування виявлено низький рівень знань, а також даних про поточні курси та відсоток їх проходження.

Така формалізація дозволяє перетворити різномірну інформацію про навчання на уніфікований вхідний сигнал для інтелектуальної підсистеми. Проєктна модель передбачає, що будь-яка зміна у статусі проходження уроку або результаті тесту призводить до миттєвої регенерації профілю компетенцій. Це забезпечує високу адаптивність системи: освітнє середовище реагує на успіхи або труднощі курсанта в реальному часі.

2.3 Алгоритм інтелектуального ранжування курсів

У межах розробки інтелектуальної підсистеми платформи ключовим завданням є перехід від текстового опису навчальних об'єктів до їхнього математичного представлення, що дозволяє проводити автоматизований аналіз релевантності. Для розв'язання цієї задачі у підрозділі пропонується математична модель семантичного представлення об'єктів навчання, що базується на використанні векторних ембеддінгів.

Наукове обґрунтування вибору векторної моделі полягає в її здатності відображати семантичні зв'язки між словами та фразами у багатовимірному просторі, де близькість векторів відповідає смисловій схожості текстів [27]. Проектована модель передбачає трансформацію довільного текстового опису курсу (T_c) та агрегованого профілю компетенцій користувача (T_u) у вектори фіксованої розмірності $v \in R^n$.

Як базовий інструмент трансформації обрано модель all-MiniLM-L6-v2 архітектури Sentence-Transformers, яка оптимізована для роботи в умовах обмежених обчислювальних ресурсів, саме для Client-side AI.

Процес формування семантичного вектора формалізується як послідовність таких математичних операцій: розбиття вхідного тексту на атомарні одиниці (токени); проходження токенів через шари трансформера для отримання контекстуальних векторів кожного слова; розрахунок результуючого вектора речення шляхом обчислення середнього значення всіх токенів, що дозволяє отримати єдиний векторний відбиток тексту незалежно від його довжини; приведення вектора до одиничної довжини, що є критично важливим для стабільності подальших розрахунків косинусної схожості [27].

Математично результат роботи моделі для тексту T можна представити як:

$$v = \text{normalize} \left(\frac{1}{N} \sum_{i=1}^N e_i \right),$$

де e_i – вектор i -го токена, а N – загальна кількість токенів у тексті. Отримана розмірність вектора становить 384 компоненти, що є оптимальним балансом між семантичною місткістю та швидкістю обробки даних у браузері.

Проектована модель семантичного представлення дозволяє уніфікувати опис навчальних об'єктів різної природи. Текстовий опис курсу, що включає його назву, теги та короткий зміст, трансформується у статичний вектор, який зберігається в базі даних. Водночас динамічний профіль користувача трансформується у вектор у реальному часі. Таким чином, задача інтелектуального підбору курсів зводиться до

математичної задачі пошуку найближчих векторів у 384-вимірному просторі, що створює підґрунтя для розробки алгоритму ранжування.

На основі математичної моделі векторного представлення даних наступним кроком проєктування є формалізація алгоритму інтелектуального ранжування. Цей алгоритм розробляється як динамічний процес, що в реальному часі зіставляє актуальний стан компетенцій курсанта із наявним каталогом навчальних програм для формування персоналізованої освітньої траєкторії.

Проєктне рішення щодо архітектури алгоритму базується на концепції Client-side Inference. Такий підхід є науково обґрунтованим для платформ військової підготовки, оскільки він забезпечує високий рівень приватності. Чутливі дані про інтереси, слабкі сторони та прогрес військовослужбовця не потребують передачі на зовнішні сервери для обробки нейромережевими моделями. Весь процес векторизації та розрахунку релевантності відбувається безпосередньо в браузері користувача за допомогою бібліотеки `@huggingface/transformers` [28].

Логіка роботи алгоритму інтелектуального ранжування проєктується як послідовність детермінованих кроків, що відображені на діаграмі діяльності (Додаток Ж). Алгоритм передбачає виконання таких етапів: агрегація контексту, формування семантичного запиту, підготовка векторів порівняння, розрахунок семантичної близькості, сортування та вибірка, модифікація рангу.

На початковому етапі: «агрегація контексту» передбачається збір актуальних даних із профілю компетенцій. Наступний етап відповідає за побудову текстового опису профілю T_u та його трансформація у вектор v_u за допомогою моделі all-MiniLM-L6-v2. Підготовка векторів порівняння передбачає отримання ембеддінгів курсів v_c , спочатку використовуються вектори, збережені в базі даних, що згенеровані адміністратором, за їх відсутності - вектори з локального кешу сесії, і лише в крайньому разі ініціюється генерація вектора з `embeddingText` курсу в реальному часі. На наступному етапі, що названий «розрахунок семантичної близькості» відбувається обчислення математичної відстані між вектором

користувача та кожним вектором курсу. Далі відбувається застосування коригуючих коефіцієнтів. Для забезпечення актуальності рекомендацій проєкт передбачає накладання штрафу на курси, що вже мають статус «Завершено» у профілі курсанта. Це дозволяє зміщувати фокус системи на нові або незавершені напрямки підготовки. На завершальному етапі відбувається формування фінального списку рекомендацій, відсортованого за спаданням модифікованого показника схожості.

Така формалізація алгоритму дозволяє системі бути адаптивною: будь-яка зміна в профілі користувача (наприклад, отримання незадовільної оцінки за тест, що оновлює weakTopics) автоматично ініціює перерахунок ранжування при наступному зверненні до підсистеми рекомендацій. Це забезпечує динамічне підлаштування освітнього контенту під поточні потреби військовослужбовця. Проєктований алгоритм поєднує в собі методи семантичного аналізу тексту та класичні підходи до фільтрації контенту, що створює надійний інструментарій для інтелектуальної підтримки навчального процесу.

Для перетворення послідовності кроків алгоритму в обчислювальний процес необхідно визначити математичний апарат оцінки релевантності. У межах проєктування цей апарат базується на метриці косинусної схожості для визначення семантичної близькості та системі евристичних штрафів для врахування навчального контексту.

Основним критерієм близькості між вектором профілю користувача A та вектором курсу B у 384-вимірному просторі обрано косинусну схожість. Вибір цієї метрики обґрунтований тим, що вона оцінює саме орієнтацію векторів, а не їхню довжину, що дозволяє ефективно порівнювати тексти різного обсягу. Математично показник схожості $sim(A, B)$ обчислюється за формулою:

$$sim(A, B) = \cos(\theta) = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \cdot \sqrt{\sum_{i=1}^n B_i^2}},$$

де $n=384$ – розмірність векторного простору. Оскільки на етапі векторизації до векторів застосовується L2-нормалізація, знаменник формули дорівнює одиниці, що спрощує розрахунок до скалярного добутку векторів. Це суттєво знижує обчислювальну складність алгоритму при роботі на клієнтських пристроях.

Для забезпечення динамічності рекомендацій та уникнення надлишкового пропонування вже вивченого матеріалу, проєктне рішення передбачає впровадження системи модифікаторів ранжування. Пропонується використовувати штрафний коефіцієнт для курсів, які вже були успішно завершені курсантом. Величина штрафу P_{comp} встановлюється на рівні 0,06. Таке значення є евристично обґрунтованим: воно достатнє для того, щоб змістити завершений курс нижче в списку Top-N, але при цьому залишає можливість його відображення, якщо семантична відповідність є надзвичайно високою.

Підсумковий показник релевантності R для кожного курсу розраховується за формулою:

$$R = \max(0, \text{sim}(\mathbf{A}, \mathbf{B}) - \delta \cdot P_{comp}),$$

де δ – бінарний індикатор, що дорівнює 1, якщо курс має статус «Завершено» у профілі користувача, та 0 в іншому випадку.

Застосування такого математичного апарату дозволяє отримати нормований показник у діапазоні $[0, 1]$, де значення, близькі до одиниці, вказують на найвищу релевантність курсу поточним потребам військовослужбовця. Проєктована система штрафів гарантує, що платформа буде пріоритетно пропонувати нові знання або курси, спрямовані на усунення виявлених «слабких тем», забезпечуючи безперервний розвиток компетенцій.

Оцінка ефективності системи штрафів ($P_{comp} = 0,06$) також виступає критерієм якості. Проєктне значення штрафу було обрано таким чином, щоб забезпечити ротацію контенту: завершені курси не зникають з поля зору повністю (що важливо для повторення матеріалу), але поступаються місцем новим знанням, що відповідає принципу безперервності військової освіти.

Реалізація обчислювально складних алгоритмів ранжування безпосередньо у браузері користувача вимагає розробки спеціалізованих підходів до керування ресурсами. У зв'язку з цим важливим етапом розробки інтелектуальної підсистеми є проєктування механізму адаптивного завантаження ШІ-моделі, що дозволяє збалансувати продуктивність платформи та швидкість відгуку інтерфейсу.

Проєктне рішення базується на стратегії відкладеного завантаження. Оскільки нейромережева модель all-MiniLM-L6-v2 має значний обсяг артефактів (ваги моделі, токенизатор, конфігураційні файли), її ініціалізація під час старту застосунку є недоцільною. Пропонується активувати процес завантаження лише у моменти безпосередньої потреби: при переході курсанта на сторінку рекомендацій або при ініціації адміністратором генерації ембедінгу для нового курсу. Це дозволяє суттєво скоротити час початкового завантаження платформи та економити мережевий трафік користувача.

Для забезпечення стабільності та ефективності інференсу проєкт має передбачати дворівневу систему кешування. По-перше персистентне кешування - використання механізму `useBrowserCache` бібліотеки `Transformers.js`. Це дозволяє зберігати файли моделі в локальному сховищі браузера після першого завантаження. При повторних зверненнях модель зчитується з диска, що скорочує час готовності системи з десятків секунд до мілісекунд. По-друге оперативне кешування - на рівні програмного коду проєктується об'єкт, який утримує завантажену модель в пам'яті протягом поточної сесії. Це запобігає повторним ініціалізаціям обчислювальних конвеєрів при багаторазових викликах алгоритму ранжування.

Процес підготовки інтелектуальної підсистеми до роботи формалізується як модель станів, що дозволяє забезпечити прозорість процесу для користувача. Проєкт передбачає чотири детерміновані стани: Очікування, Завантаження, Готовність та Помилка. Очікування свідчить про те, що модель не завантажена, ресурси не споживаються. Завантаження передбачає активну фазу отримання

даних, під час якої через `progress_callback` обчислюється агрегований відсоток готовності та обсяг завантажених байтів. Готовність свідчить про те, що модель вже успішно ініціалізована у WASM-середовищі та готова до виконання інференсу. Помилка повідомляє про збій в мережі або обмеженнях пам'яті, передбачаючи механізми безпечного відновлення. Додатковим проєктним рішенням для забезпечення стабільності на мобільних пристроях та слабких робочих станціях є обмеження обчислювальних потоків. Це дозволяє уникнути блокування основного потоку інтерфейсу під час виконання математичних розрахунків.

Завершальним етапом проєктування інтелектуальної підсистеми є визначення критеріїв оцінки точності та оптимізації алгоритму. Оскільки розроблена модель ранжування функціонує безпосередньо на клієнтських пристроях, проєктні рішення повинні забезпечувати баланс між семантичною точністю рекомендацій та обчислювальною ефективністю системи.

Для оцінки адекватності роботи алгоритму пропонується використовувати метрику релевантності, що базується на підсумковому показнику. Основним критерієм точності проєктується здатність системи максимізувати цей показник для курсів, теги та опис яких семантично збігаються з ідентифікованими слабкими темами курсанта. Проєкт передбачає, що алгоритм вважається ефективним, якщо курси, спрямовані на усунення критичних прогалин у знаннях, тобто де $\text{confidence gap} > 0.3$, посідають перші три позиції у списку Top-N рекомендацій.

У межах оптимізації обчислювальної складності було прийнято стратегічне рішення щодо вибору розмірності векторного простору. Використання моделі з розмірністю 384 компоненти, на відміну від важчих моделей з 768 або 1024 компонентами, дозволяє досягти лінійної складності розрахунку схожості $O(N \cdot D)$, де N - кількість курсів у каталозі, а D - розмірність вектора. Це гарантує, що навіть при розширенні каталогу до тисяч одиниць контенту, час ранжування на середньостатистичному клієнтському пристрої не перевищуватиме 100-150 мс, що є непомітним для користувача.

Додатковим критерієм оптимізації, закладеним на етапі проектування, є стабілізація даних. Для зменшення «шуму» при порівнянні векторів та оптимізації обсягу переданих даних, проєкт передбачає округлення значень компонентів вектора до 10 знаків після коми. Це дозволяє зберегти високу семантичну роздільну здатність моделі, одночасно зменшуючи навантаження на підсистему серіалізації/десеріалізації даних при взаємодії між фронтом та базою даних.

Висновки до другого розділу

У другому розділі проведено комплексне моделювання та проектування он-лайн платформи військової підготовки, що дозволило сформувати цілісний технічний фундамент системи. Розроблена архітектурна модель базується на дворівневій структурі, яка поєднує клієнтську частину та серверну частину. Ключовим проєктним рішенням стало винесення інференсу штучного інтелекту на сторону клієнта, що дозволило забезпечити високий рівень приватності даних військовослужбовців та оптимізувати використання серверних ресурсів за рахунок розподілених обчислень.

Проектування підсистем управління навчальним процесом дозволило формалізувати життєвий цикл навчання у вигляді моделі станів та переходів, а також розробити ієрархічну модель контенту. Важливим результатом стало створення механізму інтелектуального зворотного зв'язку, який на основі результатів автоматизованого тестування ідентифікує прогалини у знаннях та формує профіль слабких тем користувача.

Математичну основу платформи побудовано на алгоритмі інтелектуального ранжування курсів, який застосовує векторні подання для глибшого розуміння змісту навчальних матеріалів. Використання косинусної схожості разом із системою евристичних штрафів за вже пройдені курси дало змогу сформувати адаптивну модель персоналізації навчання. У результаті система насамперед

пропонує найбільш доречний контент, орієнтований на подолання виявлених прогалин у знаннях військовослужбовця.

Крім того, у розділі спроектовано моделі оптимізації взаємодії з даними, включаючи механізми пагінації, стратегії кешування серверного стану та адаптивне завантаження нейромережових моделей. Сформовані проєктні рішення, математичні моделі та алгоритми створюють необхідний теоретичний фундамент для переходу до етапу практичної реалізації програмного комплексу та подальшого дослідження його функціонування в реальних умовах.

РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ДОСЛІДЖЕННЯ ФУНКЦІОНУВАННЯ ОН-ЛАЙН ПЛАТФОРМИ

3.1 Вибір та обґрунтування технологій та інструментів розробки

Практична реалізація програмного комплексу он-лайн платформи військової підготовки потребує вибору технологічного стеку, який би повною мірою забезпечував виконання проєктних рішень, сформованих у другому розділі. Методологія вибору інструментів розробки ґрунтується на необхідності створення високонадійної, масштабованої та безпечної системи, здатної функціонувати в умовах інтенсивного обміну даними та виконувати інтелектуальні обчислення безпосередньо на пристрої користувача. Основними критеріями при виборі технологій стали продуктивність обробки асинхронних операцій, модульність архітектури, підтримка сучасних стандартів безпеки та наявність розвинутої екосистеми бібліотек для роботи з нейромережевими моделями.

Фундаментальним проєктним рішенням стало використання мови програмування TypeScript як єдиного стандарту для розробки як серверної (backend), так і клієнтської (frontend) частин системи [29]. Вибір TypeScript обґрунтовується необхідністю впровадження суворої статичної типізації, що є критично важливим для систем військового спрямування. Використання типізованих структур дозволяє виявляти значну частину логічних помилок ще на етапі компіляції, забезпечуючи цілісність даних про навчальний прогрес, результати тестування та вектори ембедінгів.

Застосування TypeScript забезпечує наскрізну типізацію в межах усього програмного комплексу. Це дозволяє використовувати спільні інтерфейси та контракти даних DTO, що гарантує повну відповідність структур об'єктів, якими обмінюються клієнтська та серверна частини через REST API. Такий підхід мінімізує ризики виникнення помилок невідповідності форматів даних, спрощує процес рефакторингу коду та підвищує загальну якість програмного забезпечення.

Крім того, TypeScript забезпечує кращу підтримку інструментів розробки IDE, що прискорює процес написання та тестування складних алгоритмів інтелектуального ранжування, описаних у попередніх розділах.

Для реалізації серверної частини програмного комплексу обрано стек технологій, що забезпечує високу швидкість обробки запитів, надійність збереження даних та суворе дотримання архітектурних патернів. Обґрунтування вибору серверних технологій базується на необхідності створення стабільного REST API, здатного ефективно керувати навчальним контентом та профілями компетенцій користувачів. Центральним елементом серверної архітектури обрано фреймворк NestJS [30]. Вибір даного інструменту зумовлений його модульною структурою, яка дозволяє реалізувати принципи декомпозиції, описані у другому розділі. NestJS пропонує чітку архітектуру, засновану на використанні контролерів, сервісів та модулів, що спрощує підтримку та масштабування системи. Важливою перевагою є вбудована підтримка механізмів впровадження залежностей, а також використання декораторів для реалізації наскрізної логіки, такої як валідація даних, обробка винятків та розмежування прав доступу.

Для взаємодії з базою даних обрано Prisma ORM [31]. На відміну від традиційних підходів, Prisma забезпечує повну типізацію запитів на основі схеми даних, що виключає цілий клас помилок, пов'язаних із невідповідністю типів між кодом та базою даних. Використання Prisma дозволяє автоматизувати процес міграцій та забезпечує високу швидкість розробки завдяки інтуїтивно зрозумілому API для маніпуляції даними. Як систему управління базами даних обрано PostgreSQL [32]. Вибір цієї реляційної СУБД обґрунтований її високою продуктивністю, підтримкою складних запитів та гарантією цілісності даних, що є критично важливим для зберігання результатів тестування та прогресу військовослужбовців.

На рис. 3.1, 3.2 зображено ER-діаграму бази даних [40], спроектованої системи.

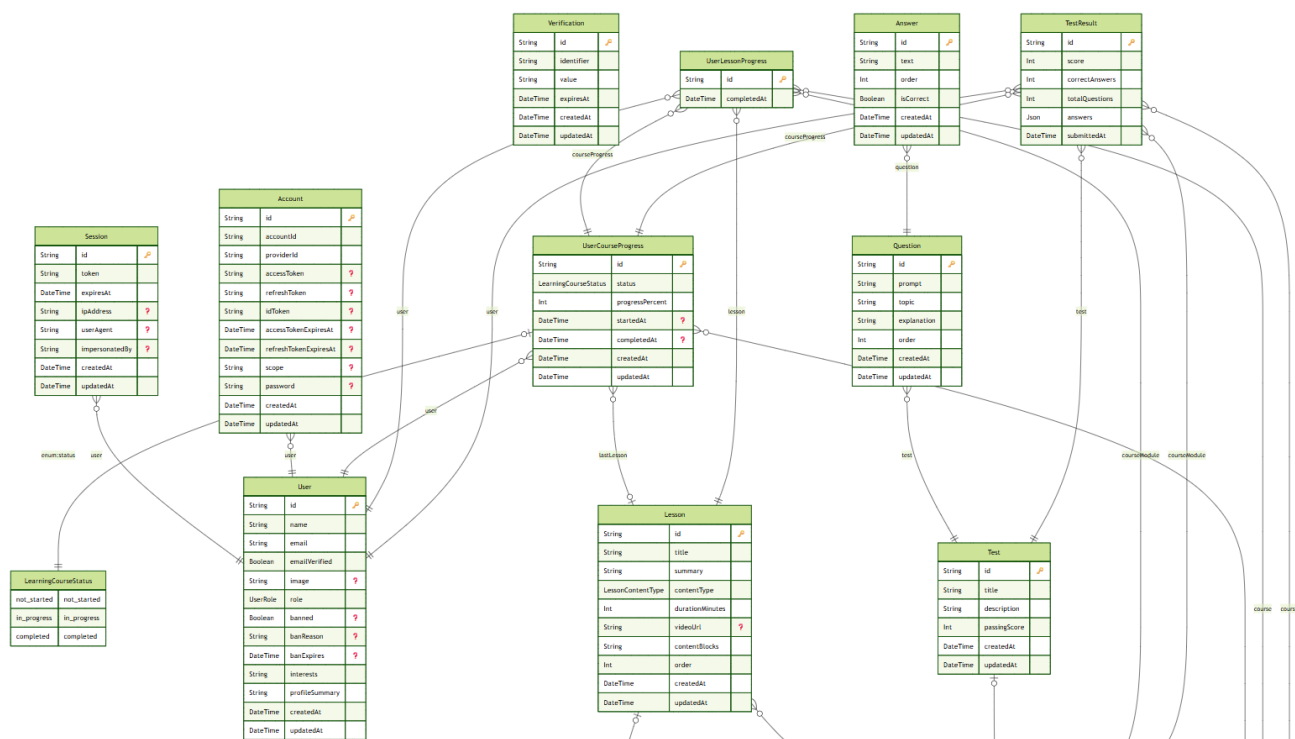


Рис. 3.1. ER-діаграма бази даних, спроектованої системи

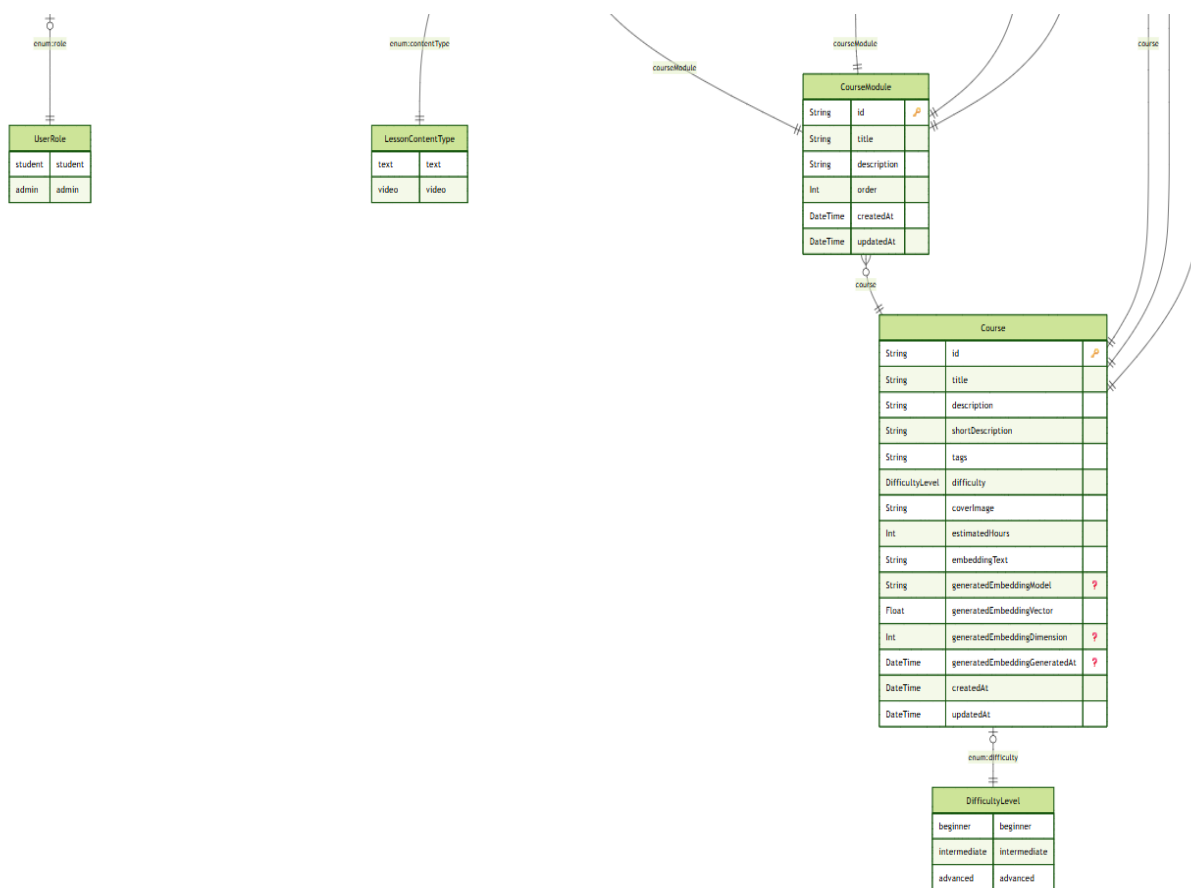


Рис. 3.2. ER-діаграма бази даних, спроектованої системи.

Табл. 3.1 User призначена для збереження інформації про користувачів онлайн-платформи військової підготовки. Вона містить персональні дані користувачів, інформацію про роль у системі, статус облікового запису, а також дані профілю та навчальної активності. Дана таблиця забезпечує можливість ідентифікації користувачів, керування доступом до функціоналу системи та персоналізації навчального процесу.

Таблиця 3.1

Структура таблиці User

Поле	Ключ	Тип	Призначення
id	PK	String	Унікальний ідентифікатор користувача
name	-	String	Ім'я користувача
email	UQ	String	Електронна пошта користувача
emailVerified	-	Boolean	Статус підтвердження електронної пошти
image	-	String	Посилання на зображення профілю
role	-	UserRole	Роль користувача у системі
banned	-	Boolean	Статус блокування користувача
banReason	-	String	Причина блокування користувача
banExpires	-	DateTime	Дата завершення блокування
interests	-	String[]	Інтереси користувача
profileSummary	-	String	Короткий опис профілю
createdAt	-	DateTime	Дата створення запису
updatedAt	-	DateTime	Дата оновлення запису

Табл. 3.2 Session призначена для збереження інформації про активні сесії користувачів у системі. Вона містить токени авторизації, час дії сесії, IP-адресу та інформацію про пристрій користувача. Дана таблиця забезпечує підтримку механізмів безпечної авторизації, контролю активних підключень та захисту облікових записів користувачів.

Таблиця 3.2

Структура таблиці Session

Поле	Ключ	Тип	Призначення
id	PK	String	Унікальний ідентифікатор сесії
token	UQ	String	Токен авторизації
userId	FK	String	Ідентифікатор користувача
expiresAt	-	DateTime	Дата завершення дії сесії
ipAddress	-	String	IP-адреса користувача
userAgent	-	String	Інформація про браузер та пристрій
impersonatedBy	-	String	Вхід від імені іншого користувача
createdAt	-	DateTime	Дата створення запису
updatedAt	-	DateTime	Дата оновлення запису

Табл. 3.3 Account призначена для збереження даних облікових записів користувачів, які використовуються для автентифікації та авторизації в системі. Вона містить інформацію про підключення до різних провайдерів (наприклад, email, OAuth), а також токени доступу та їх термін дії. Дана таблиця забезпечує можливість інтеграції з зовнішніми сервісами та підтримує механізми безпечного входу користувачів в систему онлайн-платформи військової підготовки.

Таблиця 3.3

Структура таблиці Account

Поле	Ключ	Тип	Призначення
id	PK	String	Унікальний ідентифікатор облікового запису
accountId	-	String	Ідентифікатор акаунта у зовнішньому сервісі
providerId	-	String	Назва провайдера автентифікації
userId	FK	String	Ідентифікатор користувача
accessToken	-	String	Токен доступу
refreshToken	-	String	Токен оновлення
idToken	-	String	Ідентифікаційний токен

Продовження табл. 3.3

accessTokenExpiresAt	-	DateTime	Дата завершення дії accessToken
refreshTokenExpiresAt	-	DateTime	Дата завершення дії refreshToken
scope	-	String	Рівень доступу
password	-	String	Пароль користувача
createdAt	-	DateTime	Дата створення запису
updatedAt	-	DateTime	Дата оновлення запису

Табл. 3.4 Verification призначена для збереження кодів підтвердження та токенів верифікації користувачів. Вона містить інформацію про коди підтвердження електронної пошти, відновлення паролю та інші механізми перевірки особи. Дана таблиця забезпечує підвищення рівня безпеки системи та підтримує процеси підтвердження автентичності користувачів.

Таблиця 3.4

Структура таблиці Verification

Поле	Ключ	Тип	Призначення
id	PK	String	Унікальний ідентифікатор запису
identifier	-	String	Ідентифікатор користувача або сервісу
value	-	String	Код підтвердження
expiresAt	-	DateTime	Дата завершення дії коду
createdAt	-	DateTime	Дата створення запису
updatedAt	-	DateTime	Дата оновлення запису

Табл. 3.5 Course призначена для збереження інформації про навчальні курси онлайн-платформи військової підготовки. Вона містить назву курсу, опис, рівень складності, тематичні теги та інші характеристики навчального матеріалу. Дана таблиця забезпечує структурування навчального контенту та підтримує організацію процесу дистанційного навчання користувачів.

Таблиця 3.5

Структура таблиці Course

Поле	Ключ	Тип	Призначення
id	PK	String	Унікальний ідентифікатор курсу
title	-	String	Назва курсу
description	-	String	Повний опис курсу
shortDescription	-	String	Короткий опис курсу
tags	-	String[]	Теги курсу
difficulty	-	DifficultyLevel	Рівень складності
coverImage	-	String	Зображення курсу
estimatedHours	-	Int	Тривалість курсу в годинах
embeddingText	-	String	Текст для embedding
generatedEmbeddingModel	-	String	Модель embedding
generatedEmbeddingVector	-	Float[]	Embedding-вектор
generatedEmbeddingDimension	-	Int	Розмірність embedding-вектора
generatedEmbeddingGeneratedAt	-	DateTime	Дата генерації embedding
createdAt	-	DateTime	Дата створення запису
updatedAt	-	DateTime	Дата оновлення запису

Табл. 3.6 CourseModule призначена для збереження інформації про модулі навчальних курсів. Вона містить назву модуля, опис та порядок його розташування у курсі. Дана таблиця забезпечує логічне структурування навчального матеріалу та поділ курсів на тематичні блоки для більш ефективного засвоєння інформації.

Таблиця 3.6

Структура таблиці CourseModule

Поле	Ключ	Тип	Призначення
id	PK	String	Унікальний ідентифікатор модуля
courseId	FK	String	Ідентифікатор курсу
title	-	String	Назва модуля
description	-	String	Опис модуля
order	-	Int	Порядок модуля
createdAt	-	DateTime	Дата створення запису

Продовження табл. 3.6

updatedAt	-	DateTime	Дата оновлення запису
-----------	---	----------	-----------------------

Табл. 3.7 Lesson призначена для збереження інформації про навчальні уроки, що входять до складу модулів курсів. Вона містить текстовий та мультимедійний контент, тип уроку, тривалість та порядок проходження. Дана таблиця забезпечує організацію навчального процесу та підтримує послідовне проходження навчальних матеріалів користувачами.

Таблиця 3.7

Структура таблиці Lesson

Поле	Ключ	Тип	Призначення
id	PK	String	Унікальний ідентифікатор уроку
courseModuleId	FK	String	Ідентифікатор модуля
title	-	String	Назва уроку
summary	-	String	Короткий опис уроку
contentType	-	LessonContentType	Тип контенту
durationMinutes	-	Int	Тривалість уроку
videoUrl	-	String	Посилання на відео
contentBlocks	-	String[]	Блоки контенту
order	-	Int	Порядок уроку
createdAt	-	DateTime	Дата створення запису
updatedAt	-	DateTime	Дата оновлення запису

Табл. 3.8 Test призначена для збереження інформації про тести, які використовуються для перевірки знань користувачів після проходження навчальних модулів. Вона містить назву тесту, опис та прохідний бал. Дана таблиця забезпечує можливість контролю рівня засвоєння навчального матеріалу та оцінювання результатів навчання користувачів.

Таблиця 3.8

Структура таблиці Test

Поле	Ключ	Тип	Призначення
id	PK	String	Унікальний ідентифікатор тесту
courseModuleId	FK	String	Ідентифікатор модуля

Продовження табл. 3.8

title	-	String	Назва тесту
description	-	String	Опис тесту
passingScore	-	Int	Прохідний бал
createdAt	-	DateTime	Дата створення запису
updatedAt	-	DateTime	Дата оновлення запису

Табл. 3.9 Question призначена для збереження тестових питань, що входять до складу тестів. Вона містить текст питання, тематику та пояснення правильної відповіді. Дана таблиця забезпечує формування структури тестових завдань та підтримує процес перевірки знань користувачів системи.

Таблиця 3.9

Структура таблиці Question

Поле	Ключ	Тип	Призначення
id	PK	String	Унікальний ідентифікатор питання
testId	FK	String	Ідентифікатор тесту
prompt	-	String	Текст питання
topic	-	String	Тематика питання
explanation	-	String	Пояснення правильної відповіді
order	-	Int	Порядок питання
createdAt	-	DateTime	Дата створення запису
updatedAt	-	DateTime	Дата оновлення запису

Табл. 3.10 Answer призначена для збереження варіантів відповідей на тестові питання. Вона містить текст відповіді та ознаку правильності варіанта. Дана таблиця забезпечує можливість проведення тестування користувачів та автоматичної перевірки правильності відповідей.

Таблиця 3.10

Структура таблиці Answer

Поле	Ключ	Тип	Призначення
id	PK	String	Унікальний ідентифікатор відповіді

Продовження табл. 3.10

questionId	FK	String	Ідентифікатор питання
text	-	String	Текст відповіді
order	-	Int	Порядок відповіді
isCorrect	-	Boolean	Правильність відповіді
createdAt	-	DateTime	Дата створення запису
updatedAt	-	DateTime	Дата оновлення запису

Табл. 3.11 UserCourseProgress призначена для збереження інформації про прогрес користувачів під час проходження навчальних курсів. Вона містить статус проходження, відсоток виконання курсу, дати початку та завершення навчання. Дана таблиця забезпечує моніторинг навчальної активності користувачів та контроль результатів проходження курсів.

Таблиця 3.11

Структура таблиці UserCourseProgress

Поле	Ключ	Тип	Призначення
id	PK	String	Унікальний ідентифікатор прогресу
userId	FK	String	Ідентифікатор користувача
courseId	FK	String	Ідентифікатор курсу
status	-	LearningCourseStatus	Статус проходження
progressPercent	-	Int	Відсоток проходження
startedAt	-	DateTime	Дата початку проходження
completedAt	-	DateTime	Дата завершення проходження
lastLessonId	FK	String	Останній переглянутий урок
createdAt	-	DateTime	Дата створення запису
updatedAt	-	DateTime	Дата оновлення запису

Табл. 3.12 UserLessonProgress призначена для збереження інформації про проходження користувачами окремих уроків. Вона містить дані про завершення уроків та зв'язок із загальним прогресом курсу. Дана таблиця забезпечує детальний

облік навчальної активності користувачів та контроль проходження окремих етапів навчання.

Таблиця 3.12

Структура таблиці UserLessonProgress

Поле	Ключ	Тип	Призначення
id	PK	String	Унікальний ідентифікатор запису
userId	FK	String	Ідентифікатор користувача
courseId	FK	String	Ідентифікатор курсу
courseModuleId	FK	String	Ідентифікатор модуля
lessonId	FK	String	Ідентифікатор уроку
courseProgressId	FK	String	Ідентифікатор прогресу курсу
completedAt	-	DateTime	Дата завершення уроку

Табл. 3.13 TestResult призначена для збереження результатів проходження тестів користувачами. Вона містить інформацію про отримані бали, кількість правильних відповідей та дату проходження тесту. Дана таблиця забезпечує збереження результатів оцінювання знань користувачів та підтримує аналіз ефективності навчального процесу.

Таблиця 3.13

Структура таблиці TestResult

Поле	Ключ	Тип	Призначення
id	PK	String	Унікальний ідентифікатор результату
userId	FK	String	Ідентифікатор користувача
courseId	FK	String	Ідентифікатор курсу
courseModuleId	FK	String	Ідентифікатор модуля
testId	FK	String	Ідентифікатор тесту
courseProgressId	FK	String	Ідентифікатор прогресу
score	-	Int	Отриманий бал
correctAnswers	-	Int	Кількість правильних відповідей
totalQuestions	-	Int	Загальна кількість питань

Продовження табл. 3.13

answers	-	Json	Відповіді користувача
submittedAt	-	DateTime	Дата проходження тесту

Питання безпеки та автентифікації вирішено шляхом інтеграції рішення Better Auth [33]. На етапі проєктування було прийнято рішення відмовитися від використання стандартних JWT-токенів на користь сесій на основі cookie. Better Auth забезпечує надійне керування сесіями, захист від поширених вразливостей, таких як CSRF чи XSS, та легку реалізацію рольової моделі доступу.

Реалізація клієнтської частини платформи потребує інструментарію, що забезпечує високу швидкість рендерингу, зручність керування станом та адаптивність інтерфейсу. Обґрунтування вибору клієнтських технологій базується на необхідності створення сучасного SPA-застосунку, який би надавав курсанту безперервний досвід взаємодії з навчальним контентом. Основним інструментом побудови інтерфейсу обрано бібліотеку React [34]. Такий вибір зумовлений його компонентно-орієнтованим підходом, що дозволяє декомпонувати складний інтерфейс платформи на повторно використовувані частини, такі як картки курсів, блоки уроків або форми тестування. Використання механізму Virtual DOM забезпечує високу продуктивність при оновленні інтерфейсу, що є критичним для динамічних сторінок, таких як каталог курсів із фільтрацією або режим проходження тестів. Для збірки проєкту та забезпечення швидкого циклу розробки використано інструмент Vite, який, на відміну від традиційних засобів, забезпечує миттєве оновлення модулів та оптимізовану підготовку продуктової збірки.

Ключовим елементом для реалізації стратегії оптимізації потоків даних, спроектованої у підрозділі 2.2, є бібліотека TanStack Query [35]. Вона обрана як основний інструмент для керування серверним станом. Бібліотека дозволяє автоматизувати складні процеси кешування, фонового оновлення даних та обробки станів завантаження і помилок. Це дозволяє мінімізувати кількість запитів до API, забезпечуючи миттєвий доступ до вже завантажених навчальних матеріалів при

навігації, що суттєво покращує користувацький досвід в умовах нестабільного мережевого з'єднання.

Візуальна складова та адаптивність платформи реалізовані за допомогою фреймворку TailwindCSS [36]. Вибір утилітарного підходу до стилізації дозволяє створювати легкі та швидкі інтерфейси, уникаючи накопичення надлишкового CSS-коду. TailwindCSS забезпечує консистентність дизайну та легкість адаптації платформи під різні типи пристроїв: планшети, смартфони, десктопи, що є важливим для мобільності військовослужбовців під час навчання.

Взаємодія з серверною частиною здійснюється через HTTP-клієнт Axios [37]. Його вибір обґрунтований підтримкою механізму interceptors, які використовуються для глобальної обробки помилок, автоматичного додавання заголовків авторизації та парсингу пагінаційних даних із відповідей сервера.

Реалізація інтелектуальних функцій ранжування курсів та семантичного аналізу профілів користувачів вимагає використання спеціалізованого інструментарію для роботи з моделями глибокого навчання в реальному часі. Обґрунтування інструментарію інтелектуальної підсистеми базується на архітектурному рішенні щодо виконання ІІІ-обчислень на стороні клієнта, що є ключовим для забезпечення приватності та швидкодії системи.

Основним інструментом для впровадження нейромережевих моделей у браузерне середовище обрано бібліотеку Transformers.js. Вибір цієї бібліотеки обґрунтований її здатністю забезпечувати інференс сучасних моделей обробки природної мови NLP за допомогою технологій WebAssembly та ONNX Runtime. Це дозволяє перетворити клієнтський пристрій на автономний обчислювальний вузол, здатний генерувати векторні ембедінги без необхідності звернення до зовнішніх хмарних сервісів або використання високовартісних серверних GPU-ресурсів. Для платформи військової підготовки такий підхід є критично важливим, оскільки він гарантує повну приватність даних: семантичний аналіз інтересів, слабких сторін та

навчального прогресу військовослужбовця відбувається локально, що виключає ризики витоку чутливої інформації під час її передачі на сторонні сервери обробки.

Як базову модель для генерації семантичних векторів обрано all-MiniLM-L6-v2. Вибір цієї конкретної моделі є результатом аналізу співвідношення між обчислювальною складністю та точністю семантичного представлення. Модель має компактний розмір, що дозволяє ефективно завантажувати її в кеш браузера, та генерує вектори розмірністю 384 компоненти. Така розмірність є оптимальною для задач ранжування в реальному часі, оскільки вона забезпечує достатню семантичну глибину для точного розрізнення військових дисциплін та тематик, не створюючи при цьому надмірного навантаження на центральний процесор при розрахунку косинусної схожості для великих масивів даних.

Для забезпечення стабільного циклу розробки, тестування та розгортання програмного комплексу обрано набір допоміжних інструментів, що гарантують інфраструктурну цілісність та ідентичність середовищ. Обґрунтування інструментів розробки та інфраструктури базується на принципах автоматизації рутинних процесів та забезпечення високої швидкості ітерацій при написанні коду.

Основним середовищем виконання для серверної та клієнтської частин обрано рантайм Node.js [38]. Його вибір зумовлений використанням подієво-орієнтованої, неблокуючої моделі введення-виведення, що ідеально підходить для розробки масштабованих REST API. Node.js забезпечує високу швидкість обробки асинхронних запитів, що є критичним для платформи, де велика кількість курсантів може одночасно проходити тестування або завантажувати навчальні матеріали. Крім того, використання Node.js дозволяє підтримувати єдину екосистему інструментів та пакетів, через менеджер пакетів npm, що спрощує керування залежностями проєкту.

Важливим елементом інфраструктури розробки та розгортання є технологія контейнеризації Docker [39]. У межах проєкту Docker застосовується для повної ізоляції та стандартизації середовища функціонування всіх компонентів системи:

серверної частини на NestJS, клієнтського застосунку на React та бази даних PostgreSQL. Використання інструменту docker-compose дозволяє оркеструвати роботу цих сервісів як єдиного цілісного програмного комплексу, що гарантує абсолютну ідентичність конфігурацій на етапах розробки, тестування та експлуатації. Це є стандартом якості для сучасної інженерії програмного забезпечення, забезпечуючи надійність та передбачуваність роботи платформи незалежно від особливостей хост-системи.

3.2 Реалізація он-лайн платформи

Практична реалізація он-лайн платформи для військової підготовки базується на сучасній дворівневій архітектурі, що передбачає повне розділення клієнтської та серверної частин. Такий підхід дозволяє реалізувати концепцію «товстого клієнта», де значна частина обчислювальної логіки, зокрема інференс нейромережових моделей, перенесена на сторону браузера користувача.

Серверна частина (Backend) реалізована на базі фреймворку NestJS, що використовує архітектурний патерн «Модульність». Структура вихідного коду в директорії backend/src (рис. 3.3) організована таким чином, щоб забезпечити високу ступінь інкапсуляції бізнес-логіки.

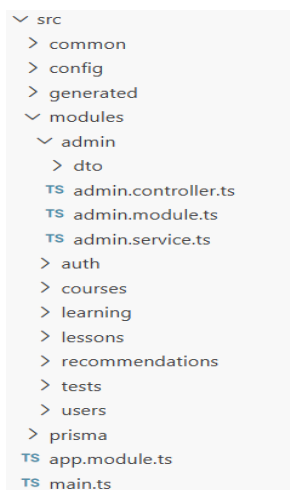


Рис. 3.3. Структура директорії backend/src

Кожен модуль у папці `modules` є автономною одиницею, що містить:

- контролери (`*.controller.ts`), які відповідають за обробку вхідних HTTP-запитів, визначення маршрутів та повернення відповідей;
- сервіси (`*.service.ts`), де зосереджена основна бізнес-логіка та взаємодія з базою даних через Prisma ORM;
- об'єкти передачі даних (DTO), які забезпечують сувору валідацію вхідних payload за допомогою `class-validator`.

Особливу увагу приділено шару `common`, який реалізує наскрізну функціональність. У ньому зосереджені `auth.guard.ts` та `roles.guard.ts`, що забезпечують безпеку на рівні кожного окремого ендпоїнта, та кастомні декоратори `@CurrentUser` та `@Roles`, які спрощують доступ до даних сесії. Шар `config` реалізує централізоване управління конфігураціями застосунку з використанням `AppConfigService`, що забезпечує валідацію змінних середовища `.env` при старті системи.

Клієнтська частина (`frontend`) побудована за принципом фіча-орієнтованої архітектури, що дозволяє масштабувати застосунок без втрати керованості кодом. Логічні рівні директорії `frontend/src` представлені на рис. 3.4.

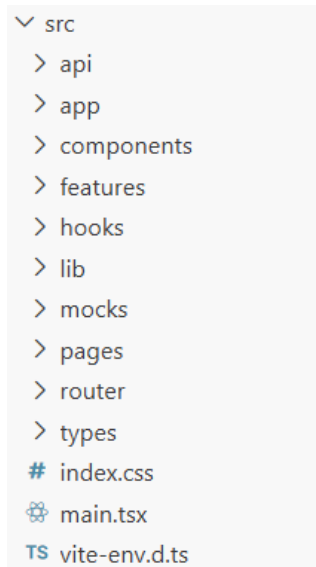


Рис. 3.4. Структура директорії `frontend/src`

Структура розділена на наступні рівні:

- features/ – основний рівень, де функціонал розбитий на домени: auth, courses, learning, ai, recommendations. Кожна фіча містить власні UI-компоненти, кастомні React-хуки для управління станом та внутрішні бібліотеки. Наприклад, фіча ai інкапсулює всю логіку роботи з Transformers.js, приховуючи складність завантаження моделей від решти застосунку;

- api/ – шар мережевої взаємодії, де реалізовано централізований клієнт Axios. Тут зосереджені контракти даних та методи запитів, що відповідають модулям бекенду;

- app/ – точка збірки застосунку, де конфігуруються провайдери, включаючи QueryClientProvider для TanStack Query та AuthProvider для управління станом автентифікації Better Auth;

- components/ui/ – бібліотека спільних компонентів (кнопки, інпути, модальні вікна), побудована на базі TailwindCSS, що забезпечує візуальну консистентність платформи.

Фундаментом для стабільної роботи описаної архітектури є шар збереження даних, оскільки саме на рівні бази даних забезпечується цілісність навчального контенту та надійність відстеження прогресу курсантів. У межах даного проєкту розробка об'єктно-реляційного відображення виконана з використанням ORM Prisma, що дозволило поєднати гнучкість реляційної СУБД PostgreSQL із перевагами суворої типізації TypeScript. Центральним елементом цієї підсистеми є файл декларативного опису схеми schema.prisma (Додаток II). Вибір Prisma обґрунтований можливістю автоматичної генерації типізованого клієнта Prisma Client, який надає готові методи для маніпуляції даними, виключаючи помилки ручного написання SQL-запитів. Логічна структура бази даних розділена на кілька функціональних кластерів.

Перший кластер – кластер навчального контенту, де реалізовано сувору ієрархію сутностей, що відображає структуру військових курсів. Сутність Course

виступає кореневим об'єктом, який через відношення «один-до-багатьох» пов'язаний із модулями CourseModule. Кожен модуль, у свою чергу, містить впорядкований набір уроків Lesson та підсумковий тест Test. Така структура дозволяє реалізувати складну логіку навігації, де кожен елемент має поле order для визначення послідовності вивчення.

Наступним є кластер відстеження прогресу та результатів. Для реалізації динамічного навчання розроблено модель UserCourseProgress, яка виконує роль сполучної ланки між користувачем та курсом. Вона зберігає агреговані дані: поточний статус, відсоток завершення та посилання на останній відвіданий урок. Деталізація прогресу забезпечується через таблицю UserLessonProgress, де фіксується факт завершення кожного окремого уроку. Результати тестування зберігаються в сутності TestResult, яка містить не лише підсумковий бал, а й повний лог відповідей у форматі Json, що дозволяє проводити глибокий аналіз помилок курсанта.

Окрему увагу приділено кластеру інтелектуальних метаданих, унікальною особливістю якого є розширення моделі Course полями для підтримки ШІ-функцій. У базі даних передбачено зберігання векторних даних у таких полях: embeddingText – текстова база для генерації вектору; generatedEmbeddingVector – масив чисел типу Float[], що зберігає 384-вимірний ембедінг; generatedEmbeddingModel – ідентифікатор моделі, використаної для генерації.

Це дозволяє системі оперувати математичними векторами безпосередньо при виконанні запитів до БД, що значно прискорює роботу алгоритмів ранжування. Також присутній кластер автентифікації та профілювання. Оскільки платформа використовує рішення Better Auth, у схемі реалізовано необхідні сутності для управління сесіями, обліковими записами та верифікацією. Сутність User розширена специфічними полями, такими як interests та profileSummary, які є критично важливими для формування контексту персоналізованих рекомендацій.

Завершується розробка цього шару генерацією клієнта в директорію `src/generated/prisma`, що дозволяє використовувати в сервісах бекенду автоматично згенеровані типи та гарантує безпеку роботи з вкладеними даними. Застосування механізму транзакцій Prisma при виконанні операцій завершення уроків та обробки тестів забезпечує атомарність змін, запобігаючи розсинхронізації даних та гарантуючи цілісність навчального прогресу в системі.

Програмна реалізація бізнес-логіки сервера зосереджена в модулі `LearningModule`, де ключову роль відіграє сервіс `LearningService`, який інкапсулює складні алгоритми управління освітньою траєкторією та забезпечує цілісність даних про успішність курсантів. Одним із центральних елементів реалізації даного сервісу є метод `recalculateCourseProgress`, який виконує динамічний розрахунок прогресу навчання (Додаток К). На відміну від лінійних моделей, даний алгоритм враховує як факт завершення уроків, так і успішність складання модульних тестів. Програмно це реалізовано через обчислення зваженого знаменника, що включає загальну кількість уроків та тестів у курсі, що дозволяє отримати об'єктивний показник засвоєння матеріалу у відсотковому еквіваленті.

Процедура перевірки знань реалізована у методі `submitTest`. Програмне рішення базується на використанні механізму транзакцій Prisma, що гарантує атомарність оновлення результатів тесту та загального прогресу курсу (Додаток Л). Важливою особливістю реалізації є попередня валідація стану проходження модуля: система блокує можливість обробки результатів тесту до моменту завершення всіх уроків, що входять до відповідного структурного підрозділу курсу. Алгоритм автоматично порівнює індекси обраних відповідей із еталонними значеннями, що зберігаються в базі даних, та формує об'єкт `TestResult`.

Для забезпечення інтелектуальної підтримки навчання реалізовано алгоритм виявлення проблемних зон курсанта у методі `buildWeakTopics` (Додаток М). Програмна логіка передбачає ретроспективний аналіз усіх спроб тестування користувача. Алгоритм агрегує дані про помилки, групує їх за тематичними

тегами, що визначені для кожного питання. На основі цих даних розраховується показник `confidenceGap` - відношення кількості неправильних відповідей до загальної кількості питань у конкретній темі. Це дозволяє системі кількісно оцінити дефіцит знань та виділити до восьми найбільш критичних тем для подальшого опрацювання.

Фінальним етапом підготовки даних для інтелектуальної підсистеми є робота сервісу `RecommendationsService`. У ньому реалізовано метод `buildUserProfileText`, який виконує семантичну агрегацію профілю користувача (Додаток Н). Алгоритм конвертує аналітичні дані про слабкі теми, задекларовані інтереси та поточний статус навчання у єдиний структурований текстовий рядок. Цей рядок виступає «семантичним відбитком» курсанта і передається на клієнтську частину для генерації вектора профілю. Таким чином, серверна логіка не лише керує навчальним процесом, а й виконує роль інтелектуального аналізатора, що готує математичну базу для персоналізації навчання.

Програмна реалізація клієнтської частини платформи базується на створенні високопродуктивного SPA-застосунку, який забезпечує безшовну взаємодію з сервером та локальне виконання інтелектуальних функцій. Реалізація фронтенд-рівня зосереджена на забезпеченні цілісності стану застосунку, оптимізації мережевих запитів та захисті конфіденційних даних користувача. Центральним вузлом мережевої взаємодії є клієнт `apiClient` на базі бібліотеки `Axios`. У файлі `apiClient.ts` реалізовано механізм перехоплювачів, які автоматизують обробку помилок та нормалізацію відповідей сервера.

Для управління серверним станом та синхронізації даних використано бібліотеку `TanStack React Query`. Програмна реалізація цього шару базується на створенні кастомних хуків, таких як `useCourses` та `useMyLearning`. У реалізації хука `useCourses` застосовано стратегію `placeholderData: keepPreviousData`, що дозволяє зберігати попередній список курсів під час оновлення фільтрів або пошуку. Це усуває ефект «мерехтіння» інтерфейсу та забезпечує плавний UX, що було

спроєктовано у другому розділі. Взаємодія з API типізована через контракти даних, що гарантує відповідність структур об'єктів між клієнтом та сервером.

Управління сесією користувача та автентифікація реалізовані через хук `useAuth`, який інтегрує клієнтську частину `Better Auth`. Програмна логіка передбачає збереження стану автентифікації в контексті застосунку та автоматичне відновлення сесії при завантаженні сторінки. Такий підхід дозволяє іншим компонентам системи миттєво отримувати доступ до даних поточного користувача, таких як роль та ідентифікатор, що є критичним для персоналізації контенту та роботи ШІ-модуля.

Безпека доступу до функціональних можливостей платформи реалізована на рівні системи роутингу. У файлі `index.tsx` (router) визначено ієрархію маршрутів, де приватні зони захищені спеціалізованим компонентом `PrivateRoute` (Додаток П). Цей компонент виконує роль гварда на стороні клієнта: він аналізує стан автентифікації та, у разі відсутності активної сесії, виконує редірект на сторінку входу. Для адміністративної панелі реалізовано додаткову перевірку ролі користувача, що гарантує ізоляцію інструментів управління контентом від звичайних курсантів.

Найбільш технологічно складною частиною програмного комплексу є реалізація інтелектуального модуля, який забезпечує виконання нейромережових обчислень безпосередньо на стороні клієнта. Програмне втілення цієї підсистеми базується на використанні бібліотеки `Transformers.js` та реалізації математичних алгоритмів векторної алгебри для аналізу семантичної близькості навчальних об'єктів.

Процес управління життєвим циклом ШІ-моделі реалізовано у файлі `model-loader.ts`. Програмна логіка базується на використанні патерна `Singleton`, що дозволяє утримувати екземпляр моделі `all-MiniLM-L6-v2` в пам'яті браузера протягом усієї сесії (Додаток Р). Реалізовано складну систему відстеження стану завантаження через механізм `progress_callback`, який транслює події завантаження

ваг моделі у візуальні індикатори прогресу для користувача. Програмно передбачено використання браузерного кешування, що дозволяє уникнути повторного завантаження даних при наступних візитах на платформу.

Функціонал векторизації тексту реалізовано у модулі `generate-embedding.ts`, логіка роботи якого передбачає асинхронний виклик моделі з чітко визначеними параметрами конвеєра: використано метод `pooling`: “mean” для отримання усередненого вектора речення та активовано L2-нормалізацію (рис. 3.5). Це гарантує, що на виході система отримує вектор розмірністю 384 компоненти, готовий для коректного порівняння без додаткової математичної обробки.

```
import { loadEmbeddingModel } from '@features/ai/lib/model-loader';

export async function generateEmbedding(text: string) {
  const normalizedText = text.trim();

  if (!normalizedText) {
    return [];
  }

  const model = await loadEmbeddingModel();
  const output = await model(normalizedText, {
    pooling: 'mean',
    normalize: true,
  });

  return Array.from(output.data);
}
```

Рис. 3.5. Програмна реалізація генерації embedding-векторів

Математичний апарат порівняння векторів реалізовано у `cosine-similarity.ts`. Функція обчислення косинусної схожості програмно втілює розрахунок скалярного добутку двох масивів чисел, нормалізований за їхньою евклідовою довжиною (рис. 3.6). Реалізація містить захисні механізми для обробки нульових векторів або невідповідності розмірностей, що забезпечує відмовостійкість інтелектуального модуля при роботі з динамічним контентом.

```

export function cosineSimilarity(a: number[], b: number[]) {
  if (!a.length || !b.length || a.length !== b.length) return 0;

  let dot = 0;
  let aNorm = 0;
  let bNorm = 0;

  for (let index = 0; index < a.length; index += 1) {
    dot += a[index] * b[index];
    aNorm += a[index] * a[index];
    bNorm += b[index] * b[index];
  }

  if (!aNorm || !bNorm) return 0;
  return dot / (Math.sqrt(aNorm) * Math.sqrt(bNorm));
}

```

Рис. 3.6. Алгоритм обчислення косинусної схожості векторів

Фінальна логіка інтелектуального підбору курсів зосереджена у функції rankCoursesByProfile (Додаток С). Цей алгоритм виконує багатопотоковий збір ембедінгів усіх курсів каталогу. Алгоритм використовує багаторівневу стратегію отримання векторів: пріоритет надається даним, отриманим від бекенду, після чого перевіряється локальний кеш сесії, і лише за відсутності готових даних ініціюється генерація вектора в реальному часі.

Особливістю реалізації ранжування є програмне впровадження системи штрафів за завершене навчання. Алгоритм знаходить ідентифікатори курсів зі статусом completed та автоматично віднімає коефіцієнт 0.06 від їхнього показника схожості. Це дозволяє системі динамічно оновлювати список рекомендацій, пропонуючи курсанту нові напрямки підготовки, що семантично відповідають його слабким темам. Результатом роботи модуля є масив об'єктів RecommendationResult, який містить не лише ранг курсу, а й сформоване пояснення релевантності для відображення в інтерфейсі.

Завершальним етапом розробки програмного комплексу є реалізація адміністративного інтерфейсу та інструментів підготовки контенту. Ця підсистема розроблена для забезпечення повного циклу управління навчальними матеріалами та формування математичного базису для роботи інтелектуальних алгоритмів.

Серверна частина адміністративного модуля зосереджена в `AdminService`, який забезпечує виконання CRUD-операцій над сутностями навчального контенту, використовуючи складні запити `Prisma` для збереження ієрархічних структур. Реалізація методів створення та оновлення курсів передбачає атомарний запис даних про модулі, уроки та тести в межах однієї транзакції. Безпека адміністративних операцій гарантується на рівні контролера `AdminController` за допомогою декоратора `@Roles('admin')`, який активує `RolesGuard` для перевірки прав доступу користувача перед виконанням будь-якої адміністративної дії.

Клієнтська частина адміністративного інтерфейсу реалізована через набір спеціалізованих сторінок та форм, ключовою з яких є `CourseEditorForm` (Додаток Т). Форма дозволяє адміністратору не лише редагувати текстові атрибути курсу, а й динамічно змінювати структуру модулів, додавати контентні блоки до уроків та конструювати тестові питання з варіантами відповідей. Валідація введених даних здійснюється на основі схем `Zod`, що забезпечує відповідність `payload` очікуваним контрактам API.

Унікальною функціональною можливістю адміністративного інтерфейсу є інтегрований інструмент підготовки III-метаданих. Програмно це реалізовано через кастомний хук `useGenerateEmbedding`, який взаємодіє з раніше описаним модулем `ai` (Додаток У). При натисканні кнопки генерації система зчитує вміст поля `embeddingText`, виконує локальний інференс моделі та отримує 384-вимірний вектор. Програмне рішення передбачає автоматичне округлення значень компонентів вектора до 10 знаків після коми для оптимізації обсягу даних, що передаються на сервер.

Фіксація інтелектуальних даних у системі відбувається через передачу об'єкта `generatedEmbedding` у складі загального `payload`-а курсу. На стороні сервера реалізовано мапер даних, який трансформує вхідний DTO у формат моделі `Prisma`, заповнюючи поля `generatedEmbeddingVector`, `generatedEmbeddingModel` та `generatedEmbeddingDimension` (Лістинг 3.16). Це дозволяє адміністратору

самостійно формувати семантичний відбиток курсу, який згодом буде миттєво використаний алгоритмом ранжування без додаткових обчислень.

3.3 Інтерфейс користувача та його функціональні можливості

Фінальним етапом розробки програмного комплексу є створення графічного інтерфейсу користувача, який виступає сполучною ланкою між складними алгоритмами обробки даних та курсантом. Загальна концепція та візуальна стилістика інтерфейсу платформи розроблена з урахуванням специфіки військової підготовки, де пріоритетними є чіткість викладу інформації, сувора ієрархічність та мінімізація когнітивного навантаження на користувача.

Візуальна стилістика системи базується на принципах сучасного мінімалізму з використанням утилітарного підходу до дизайну. Основна кольорова гама платформи поєднує нейтральні світлі тони для фонових елементів та акцентні кольори для виділення важливих статусів і прогресу навчання. Таке рішення забезпечує високу контрастність тексту та зручність тривалої роботи з навчальними матеріалами. Для побудови інтерфейсу використано бібліотеку TailwindCSS, що дозволило реалізувати консистентну систему відступів, типографіки та візуальних ефектів у межах усього застосунку. Загальний вигляд головного екрана платформи та структура навігації представлені в додатку Ф.

Архітектурно інтерфейс побудований навколо центрального макета AppShell, який забезпечує постійний доступ до ключових розділів системи. Макет включає бічну панель навігації, де зосереджені посилання на каталог курсів, особистий кабінет, інтелектуальні рекомендації та додаткові адміністративні функції, а також інформацію про поточну сесію користувача. Основна область контенту проєктується як динамічна зона, що адаптується під конкретний функціональний модуль. Важливою особливістю інтерфейсу є його повна адаптивність. Завдяки використанню гнучких сіток grid та флекс-контейнерів, елементи керування та

навчальний контент коректно відображаються на пристроях з різною діагоналлю екрана.

Взаємодія користувача з платформою розпочинається з підсистеми автентифікації, яка забезпечує безпечний доступ до навчальних матеріалів та персоналізацію освітнього простору. Функціонал автентифікації та налаштування профілю реалізовано у вигляді інтерфейсів, що інтегрують складну логіку Better Auth у прості та зрозумілі форми взаємодії. Сторінки реєстрації та входу проєктуються з використанням централізованих компонентів LoginForm та RegisterForm, які забезпечують валідацію введених даних на стороні клієнта, запобігаючи некоректному заповненню полів. Візуальне представлення інтерфейсу входу до системи та форми створення нового облікового запису наведено в додатку X.

На платформі наявна сторінка профілю користувача, де є можливість базового налаштування анкетних параметрів. На цій сторінці користувач може змінити свій логін, адресу електронної пошти чи пароль. Інтерфейс особистого кабінету відображено на додатку Ц. Програмна реалізація профілю забезпечує миттєве збереження змін та оновлення стану автентифікації без необхідності повного перезавантаження сторінки. Використання реактивних форм дозволяє користувачу бачити актуальний стан своїх даних та отримувати підтвердження про успішне оновлення інформації через систему спливаючих повідомлень.

Центральним місцем платформи є каталог навчальних курсів. Інтерфейс каталогу курсів та механізми пошуку розроблені для забезпечення швидкої навігації серед великої кількості навчальних дисциплін. Сторінка CoursesPage представляє собою динамічну сітку карток курсів, кожна з яких містить вичерпну інформацію про предмет: назву, короткий опис, рівень складності, тривалість та тематичні теги. Інтерфейс каталогу курсів представлений на додатку Ф. Для ефективного пошуку необхідного контенту реалізовано панель фільтрації CourseFilters, яка дозволяє шукати курси за складністю та тегами. Також, наявний

функціонал пошуку з використанням алгоритму дебаунсингу, який забезпечує оптимізацію арі-запитів завдяки відсіканню зайвих запитів до сервера. Кожна картка курсу також відображає поточний статус взаємодії користувача з ним - наприклад, індикатор «В процесі» або відсоток прогресу, що дозволяє курсанту легко ідентифікувати пріоритетні для продовження навчання програми. Приклад відображення картки курсу з детальними метаданими та статусом проходження наведено на додатку Ш.

Основним функціональним блоком платформи, що забезпечує безпосередню взаємодію з навчальним матеріалом, є режим навчання та підсистема контролю знань. Проектування цього розділу спрямоване на створення комфортного когнітивного середовища, де курсант може послідовно засвоювати знання та отримувати миттєву верифікацію своїх компетенцій. Центральним елементом навчального процесу є сторінка уроку `LearningLessonPage`, інтерфейс якої оптимізовано для тривалої концентрації на контенті. Основна область сторінки підтримує відображення текстових блоків та вбудованих відеоматеріалів, забезпечуючи мультимодальність навчання. Візуалізація інтерфейсу уроку з навігаційною панеллю та контентною областю представлена на додатку Щ.

Важливою складовою підсистеми контролю знань є інтерфейс тестування, реалізований через компонент `ModuleTestForm`. Форма тестування проектується таким чином, щоб забезпечити однозначність сприйняття питань та варіантів відповідей. Після завершення тесту система ініціює анімацію обробки даних, після чого виводить екран результатів. На цьому екрані курсант бачить свій підсумковий бал, кількість правильних відповідей та інформацію про те, чи було досягнуто прохідний поріг. У разі успішного складання, інтерфейс автоматично оновлює статус модуля, відкриваючи доступ до наступних розділів. Приклад інтерфейсу модульного тестування та вікна з результатами перевірки знань наведено на додатку Ю.

Для комплексного відстеження освітньої траєкторії реалізовано сторінку «Моє навчання», яка виступає аналітичним центром для курсанта. На цій сторінці у вигляді карток представлені всі активні навчальні програми. Кожна картка містить динамічний прогрес-бар, що візуалізує відсоток завершення курсу, розрахований на основі пройдених уроків та тестів. Окремим блоком на сторінці винесено панель «Слабкі теми», де у вигляді списку бейджів відображаються тематики, в яких користувач припустився найбільшої кількості помилок. Це дозволяє курсанту швидко перейти до повторення складного матеріалу, підвищуючи загальну ефективність підготовки. Інтерфейс сторінки представлено на додатку Я.

Особливе місце в структурі платформи посідає інтерфейс інтелектуальних рекомендацій, який є візуальним втіленням розробленого алгоритму семантичного ранжування. Цей розділ спроектований як інтерактивний помічник, що допомагає курсанту орієнтуватися в навчальному контенті на основі його індивідуальних потреб, виявлених слабких тем та задекларованих інтересів.

Оскільки архітектурне рішення передбачає виконання ІІІ-обчислень безпосередньо в браузері користувача, інтерфейс сторінки RecommendationsPage реалізує специфічний стан очікування, пов'язаний з ініціалізацією нейромережевої моделі. При першому відвідуванні розділу користувач бачить інформаційну панель із прогрес-баром, який у реальному часі відображає етапи завантаження ваг моделі all-MiniLM-L6-v2. Це забезпечує прозорість роботи системи та інформує про готовність інтелектуального модуля до виконання розрахунків. Після завершення ініціалізації моделі інтерфейс автоматично трансформується у список персоналізованих карток рекомендацій. Кожна картка проєктується таким чином, щоб надати курсанту не лише посилання на курс, а й ступінь його релевантності. Ключовим елементом дизайну тут є індикатор семантичної схожості, представлений у відсотковому еквіваленті. Інтерфейс результатів інтелектуального ранжування з відображенням відсотка релевантності наведено на додатку АА.

Функціональні можливості сторінки рекомендацій також включають механізм динамічного оновлення, тобто після успішного складання тесту, що оновлює список слабких тем, інтерфейс пропонує користувачу перерахувати рекомендації. Такий функціонал дозволяє підтримувати актуальність освітніх пропозицій у реальному часі. Картки рекомендацій візуально відрізняються від стандартних карток каталогу акцентним оформленням та наявністю рангу релевантності.

Завершальним компонентом графічної оболонки системи є адміністративна панель та інструменти управління контентом, які призначені для інструкторів-адміністраторів платформи. Цей розділ спроектований як професійне робоче середовище для створення, структурування та інтелектуальної підготовки навчальних матеріалів, забезпечуючи повний контроль над освітнім контентом без необхідності прямого втручання в базу даних.

Основним екраном адміністративного модуля є сторінка управління списком курсів AdminCoursesPage. Вона реалізована у вигляді інтерактивної таблиці, що надає адміністратору швидкий доступ до всіх наявних програм, відображаючи їхній статус, кількість модулів та дату останнього оновлення. Інтерфейс передбачає зручні механізми фільтрації та швидкого пошуку, а також функціональні кнопки для переходу до редагування або видалення контенту. Загальний вигляд панелі управління курсами для адміністратора представлено на додатку АБ.

Найбільш функціонально насиченим елементом адміністративної зони є форма редагування курсу CourseEditorForm. Вона проектується як багаторівневий інтерфейс, що дозволяє керувати не лише загальними метаданими, такими як: назва, опис та складність, а й внутрішньою ієрархією об'єкта. Редактор структури курсу дозволяє адміністратору динамічно додавати до нього модулі, змінювати порядок уроків та конструювати тестові завдання безпосередньо в межах однієї форми. Кожен розділ форми логічно відокремлений, що спрощує роботу з великими

обсягами інформації. Інтерфейс редактора структури курсу та модулів представлено на додатку АВ.

Унікальною особливістю адміністративного інтерфейсу є вбудований інструментарій для підготовки ІІІ-даних. У формі редагування передбачено спеціалізований блок «Інтелектуальні метадані», де адміністратор формує семантичний опис курсу. Поруч із текстовим полем розташована кнопка ініціації генерації ембедінгу. При її натисканні інтерфейс відображає стан локального інференсу, а після завершення розрахунків виводить параметри згенерованого вектора (модель, розмірність). Це дозволяє адміністратору верифікувати готовність курсу до роботи в системі інтелектуальних рекомендацій ще до моменту його публікації. Візуалізація інструменту генерації семантичних ембеддінгів в адмін-панелі наведена на рис. 3.7.

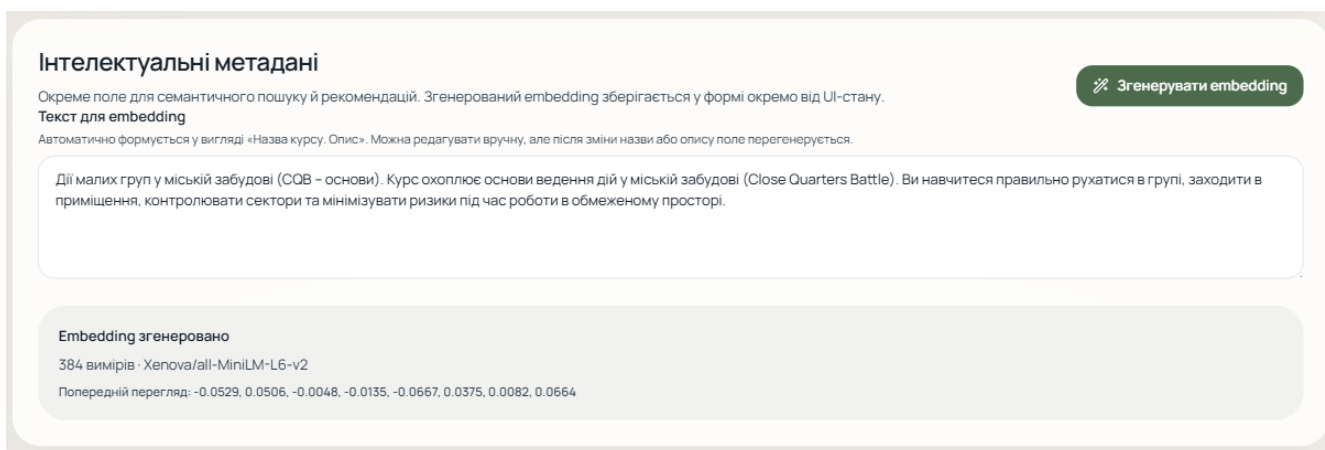


Рис. 3.7. Інтерфейс генерації семантичних embedding-векторів в адміністративній панелі

Висновки до третього розділу

У третьому розділі було виконано практичну реалізацію програмного комплексу он-лайн платформи військової підготовки. На основі проведеного аналізу обґрунтовано вибір технологічного стеку, що включає фреймворки NestJS та React, мову програмування TypeScript та СУБД PostgreSQL. Особливу увагу

приділено вибору інструментарію для реалізації інтелектуальних функцій, зокрема бібліотеки Transformers.js, що дозволило впровадити концепцію виконання неймережових обчислень на стороні клієнта для забезпечення максимальної приватності даних та швидкодії системи.

Програмна реалізація серверної частини дозволила втілити модульну архітектуру, де кожен функціональний блок інкапсулює специфічну бізнес-логіку управління навчанням, тестуванням та формуванням аналітичного профілю курсанта. Завдяки використанню ORM Prisma було реалізовано надійний шар збереження даних із суворою типізацією. Розроблено та впроваджено алгоритми динамічного перерахунку прогресу та виявлення слабких тем, які стали основою для формування семантичного контексту персоналізованих рекомендацій.

На етапі проєктування бази даних створено структуру таблиць, яка дозволяє зберігати всі необхідні дані: курси, тести, запитання, відповіді, а також інформацію про користувачів і їхню діяльність. ER-діаграма та розроблені таблиці забезпечують логічну організацію даних, що сприяє їхній ефективній обробці.

Клієнтська частина платформи реалізована як сучасний SPA-застосунок із фіча-орієнтованою структурою. Програмно втілено механізми асинхронного завантаження ІІІ-моделей, векторизації текстів та розрахунку семантичної близькості курсів у реальному часі за допомогою метрики косинусної схожості. Також, реалізована ізоляція адміністративних функцій на основі захисту маршрутів та управління сесіями Better Auth.

Завершальним етапом стала розробка графічного інтерфейсу користувача та адміністративної панелі. Створено адаптивне та структуроване освітнє середовище, яке забезпечує зручну взаємодію з навчальним контентом, прозору візуалізацію роботи ІІІ-алгоритмів та потужний інструментарій для управління освітніми програмами. Результати реалізації підтверджують повну працездатність спроектованих моделей та готовність програмного комплексу до експлуатації в умовах реальної військової підготовки.

ВИСНОВКИ

У магістерській атестаційній роботі проведено комплексне дослідження принципів побудови та проєктування он-лайн платформи для військової підготовки з інтеграцією елементів штучного інтелекту. У ході виконання роботи було розв'язано актуальну науково-практичну задачу створення персоналізованого освітнього середовища, яке враховує специфіку військової служби та забезпечує високу ефективність засвоєння знань.

Основним результатом роботи, відповідно до поставленої мети, є розробка архітектурних принципів, проєктування та програмна реалізація он-лайн платформи військової підготовки, яка забезпечує автоматизовану персоналізацію навчання на основі семантичного аналізу компетенцій користувача.

У процесі дослідження було отримано такі наукові та практичні результати.

Проаналізовано сучасні підходи до побудови он-лайн платформ навчання та досвід застосування технологій штучного інтелекту в освітньому процесі. Виявлено, що існуючі рішення потребують адаптації для військової сфери в частині безпеки даних та інтелектуального супроводу курсанта.

Визначено специфічні вимоги до систем військової підготовки, серед яких ключовими є сувороструктурованість контенту, тобто ієрархія курсів, модулів і уроків, та забезпечення конфіденційності профілю навчання. На основі цих вимог сформовано технічне завдання на розробку системи.

Спроектовано архітектурну модель платформи, яка базується на концепції товстого клієнта та забезпечує виконання нейромережевих обчислень безпосередньо на стороні браузера користувача. Це дозволило реалізувати Client-side AI інференс, що гарантує автономність інтелектуальних функцій та захист персональних даних.

Розроблено математичну модель та алгоритм семантичного ранжування курсів, що базується на використанні векторних ембедінгів моделі all-MiniLM-L6-v2. Алгоритм враховує динамічний профіль компетенцій користувача та

використовує метрику косинусної схожості з авторською системою штрафів за завершені курси.

Реалізовано он-лайн платформу із використанням сучасного стеку технологій: NestJS для серверної частини, React для клієнтського інтерфейсу та Transformers.js для інтеграції ШІ. У систему інтегровано підсистему управління навчальним прогресом, яка в реальному часі відстежує проходження уроків та результати тестування.

Практична цінність отриманих результатів полягає у створенні готового до експлуатації інструментарію, який дозволяє автоматизувати процес військової підготовки, роблячи його більш персоналізованим та ефективним. Впровадження розробленої платформи сприяє швидшому засвоєнню спеціалізованих знань за рахунок інтелектуального підбору контенту, що відповідає актуальному рівню підготовки кожного конкретного військовослужбовця.

Напрямами подальшого наукового дослідження та вдосконалення програмного комплексу є: інтеграція більш складних мовних моделей (LLM) для створення інтелектуальних чат-ботів-репетиторів, здатних надавати розгорнуті пояснення до навчального матеріалу; розробка мобільного застосунку з підтримкою повноцінного офлайн-режиму навчання та локального збереження ШІ-моделей; розширення алгоритму ранжування за рахунок врахування соціального контексту та часових факторів забування інформації; дослідження можливостей використання ONNX Runtime для прискорення інтерфейсу на пристроях із графічними прискорювачами.

Таким чином, розроблена он-лайн платформа є завершеним науково-технічним рішенням, яке демонструє ефективність поєднання сучасних вебтехнологій та штучного інтелекту в галузі військової освіти.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Kravchenko O. I. Selection and Justification of Architecture for a Web-Based System for Military Training Courses // Технічна творчість : зб. наук. праць. Хмельницький : ХНУ, 2023. № 7. С. 112–114.
2. Kravchenko O. I. Subject Area Research For Competitor Analysis // Технічна творчість : зб. наук. праць. Хмельницький : ХНУ, 2023. № 7. С. 139–141.
3. Биков В. Ю. Цифрова трансформація суспільства і розвиток комп'ютерно-технологічної платформи освіти і науки України : монографія. Київ : ІТЗН НАПН України, 2019. 80 с.
4. Бичков О. С., Сєверінов О. В. Веб-технології: проектування та розробка : навч. посіб. Київ : Видавничо-поліграфічний центр «Київський університет», 2021. 240 с.
5. Архітектура вебзастосунків: що це, типи та рівні. Robot Dreams. URL: <https://robotdreams.cc/uk/blog/567-arhitektura-vebzastosunkiv>.
6. Сучасні технології розробки Front-end та Back-end частин веб-проектів : метод. вказівки / уклад. С. В. Чопоров, С. О. Гнітій, Г. В. Гнітій. Запоріжжя : НУ «Запорізька політехніка», 2023. 62 с.
7. Морзе Н. В. Моделі навчання у цифровому середовищі : монографія. Київ : НУБіП України, 2022. 190 с.
8. Спірін О. М. Інформаційно-комунікаційні технології в освіті та науці : навч. посіб. Житомир : Вид-во ЖДУ ім. І. Франка, 2021. 210 с.
9. Яцина О. Ф. Формування професійної компетентності студентів в умовах дистанційного навчання // Інноваційна педагогіка. 2021. Вип. 31, т. 2. С. 84–92.
10. Литвинова С. Г. Проектування хмаро орієнтованого навчального середовища загальноосвітнього навчального закладу : монографія. Київ : Компринт, 2016. 354 с.
11. Гриценчук О. В. Використання штучного інтелекту в освіті: тенденції та перспективи в Україні та за кордоном // Вісник кафедри ЮНЕСКО «Неперервна

- професійна освіта XXI століття». 2024. Т. 2, № 10. С. 152–161. DOI: 10.35387/ucj.2(10).2024.0012.
12. Величко Я. І. Адаптивне навчання та персоналізація з використанням штучного інтелекту // Сучасні тенденції розвитку інформаційних технологій в освіті та науці : матеріали конф. Харків : Харківський національний автомобільно-дорожній університет, 2024. С. 67–70.
 13. Coursera: навчальна онлайн-платформа. URL: <https://www.coursera.org>.
 14. Prometheus: найбільша платформа масових відкритих онлайн-курсів в Україні. URL: <https://prometheus.org.ua>.
 15. Official Website of the Joint Chiefs of Staff. URL: <https://www.jcs.mil>.
 16. DEEP Portal – NATO DEEP ADL Portal. URL: <https://deepportal.hq.nato.int/eacademy>.
 17. DLN Learning Management System. URL: <https://dln-rad.forces.gc.ca/login-lien/index-eng.html>.
 18. ADELE (O). URL: <https://www.adele.edu.au/login/index.php>.
 19. Бойко О. В. Стан та перспективи розвитку дистанційного навчання у системі військової освіти України // Збірник наукових праць Центру воєнно-стратегічних досліджень НУОУ ім. Івана Черняхівського. 2019. № 2. С. 115–120.
 20. Хаустова В. Є., Решетняк О. І., Хаустов М. М., Зінченко В. А. Напрямки розвитку технологій штучного інтелекту в забезпеченні обороноздатності країни // Бізнес Інформ. 2022. № 3. С. 17–26.
 21. Серверні web-технології : навч. посіб. / уклад. О. С. Бунке. Київ : КПІ ім. Ігоря Сікорського, 2023. 109 с.
 22. Гнатюк С. О. Кібербезпека в системах дистанційної військової підготовки // Захист інформації. 2021. Т. 23, № 4. С. 201–210.
 23. Як перетворити feature-based структуру на повноцінну фронтенд-архітектуру. DOU. URL: <https://dou.ua/forums/topic/58333/>.

24. Які є конвенції в REST API та для чого їх дотримуватись. DOU. URL: <https://dou.ua/forums/topic/34550/>.
25. Ткачук В. В. Штучний інтелект у вищій освіті: сучасний стан та перспективи // Фізико-математична освіта. 2021. Вип. 2(28). С. 114–120.
26. Xenova. all-MiniLM-L6-v2 ONNX model for in-browser inference. Hugging Face Model Hub. URL: <https://huggingface.co/Xenova/all-MiniLM-L6-v2>.
27. Кучерук В. Ю., Глушко М. В. Покращення алгоритму «Item to Item» методу колаборативної фільтрації для розробки рекомендаційних систем на основі косинусної міри шляхом оцінки релевантності // ScienceRise. — 2018. — № 1 (42). — С. 20–24.
28. Документація Transformers.js: запуск моделей Hugging Face в браузері. Hugging Face. URL: <https://huggingface.co/docs/transformers.js>.
29. TypeScript Documentation. URL: <https://www.typescriptlang.org/docs>.
30. NestJS Documentation. URL: <https://docs.nestjs.com>.
31. Prisma Documentation. URL: <https://www.prisma.io/docs>.
32. PostgreSQL Documentation. URL: <https://www.postgresql.org/docs>.
33. Better Auth Documentation. URL: <https://www.better-auth.com/docs>.
34. React Documentation. URL: <https://react.dev> (дата звернення: 15.06.2026).
35. TanStack Query Documentation. URL: <https://tanstack.com/query/latest/docs/framework/react/overview>.
36. Tailwind CSS Documentation. URL: <https://tailwindcss.com/docs>.
37. Axios Docs. URL: <https://axios-http.com/docs/intro>.
38. Node.js Documentation. URL: <https://nodejs.org/docs>.
39. Docker Documentation. URL: <https://docs.docker.com>.
40. Босько В. В., Константинова Л. В., Поліщук Л. І., Коноплицька-Слободенюк О. К. Бази даних : навч. посіб. Кропивницький : ЦНТУ, 2024. 226 с.

ДОДАТКИ

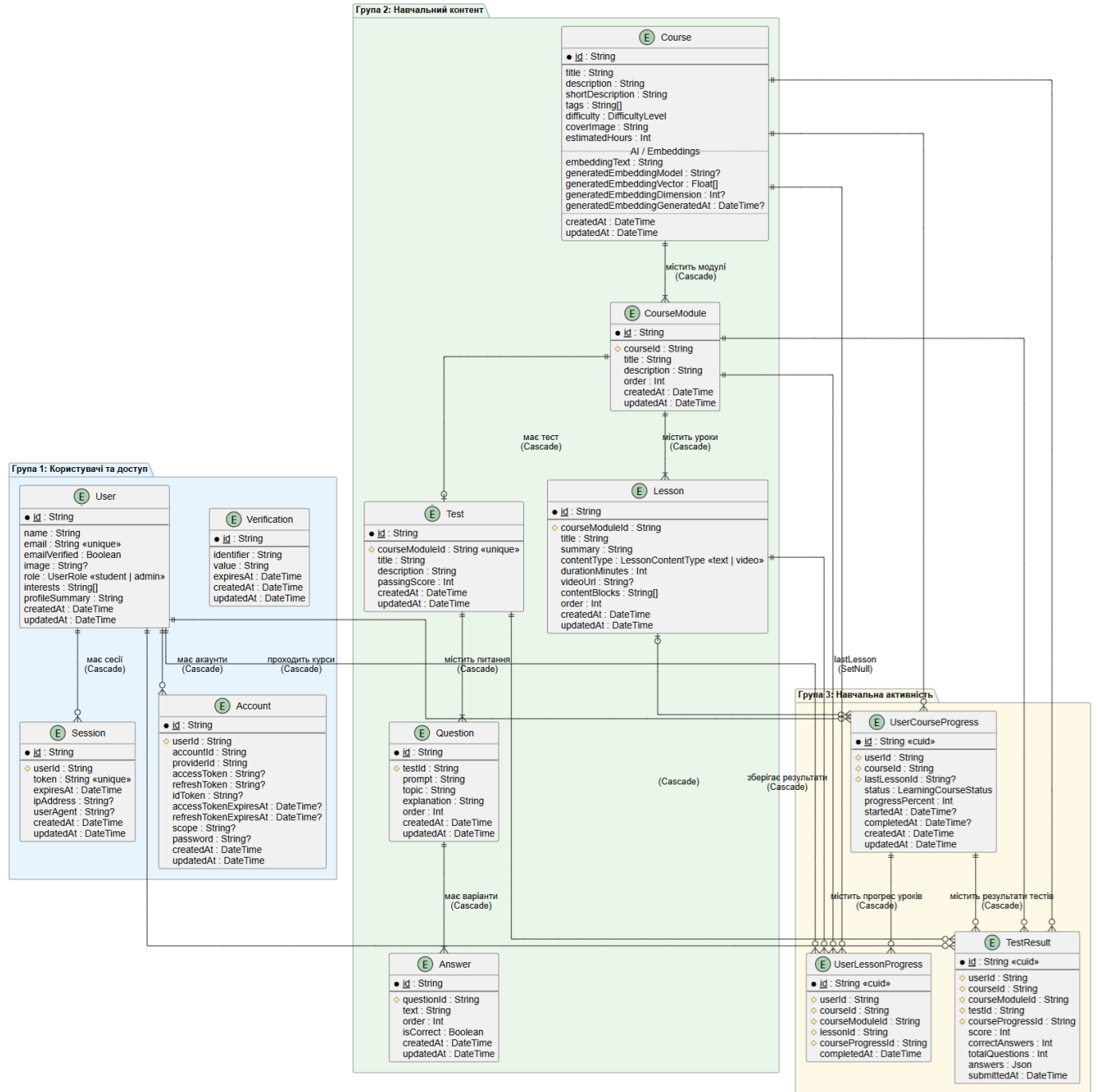
Додаток А

UML-діаграма варіантів використання платформи військової підготовки



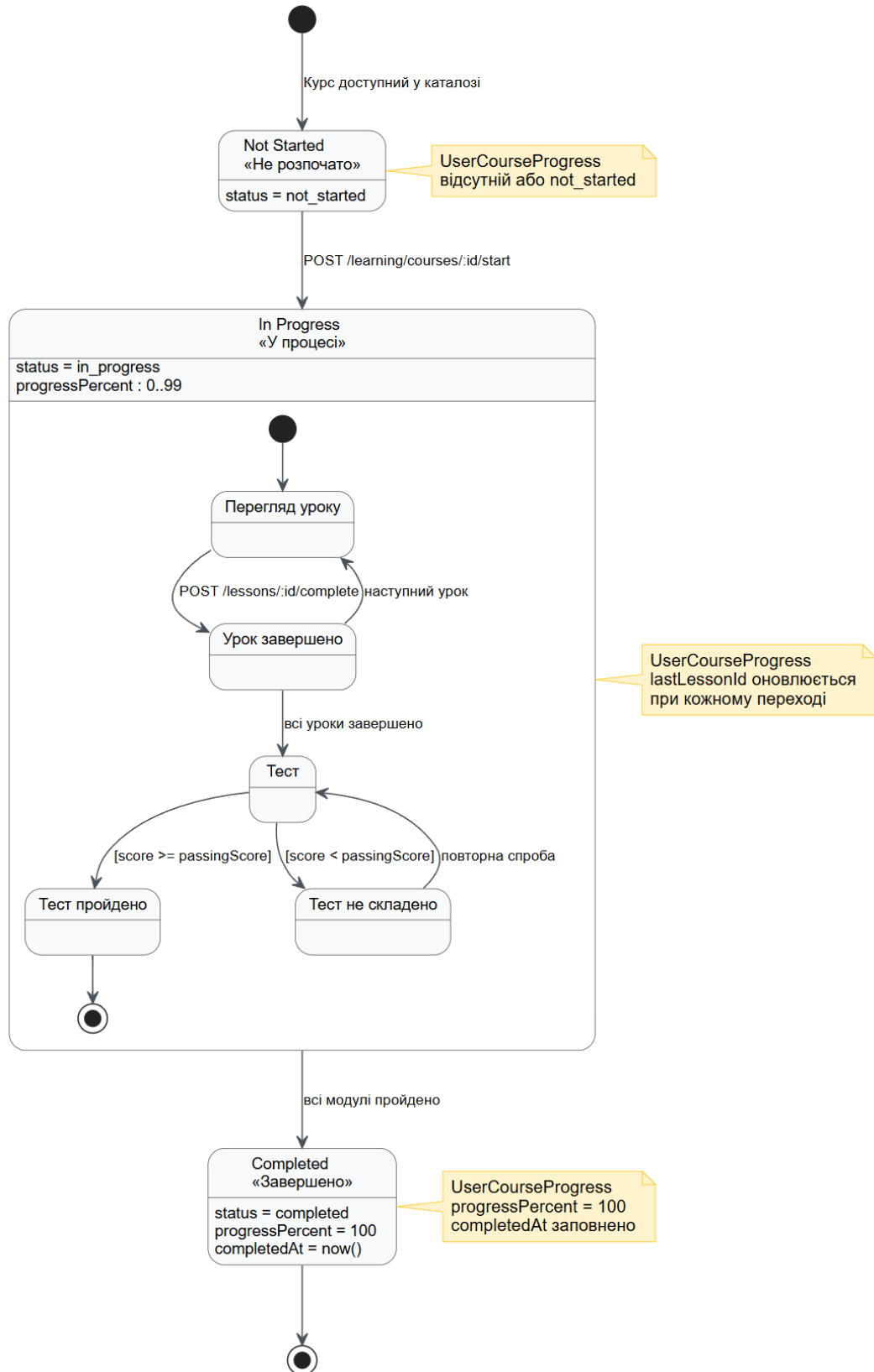
Додаток Б

ER-діаграма моделі даних платформи військової підготовки



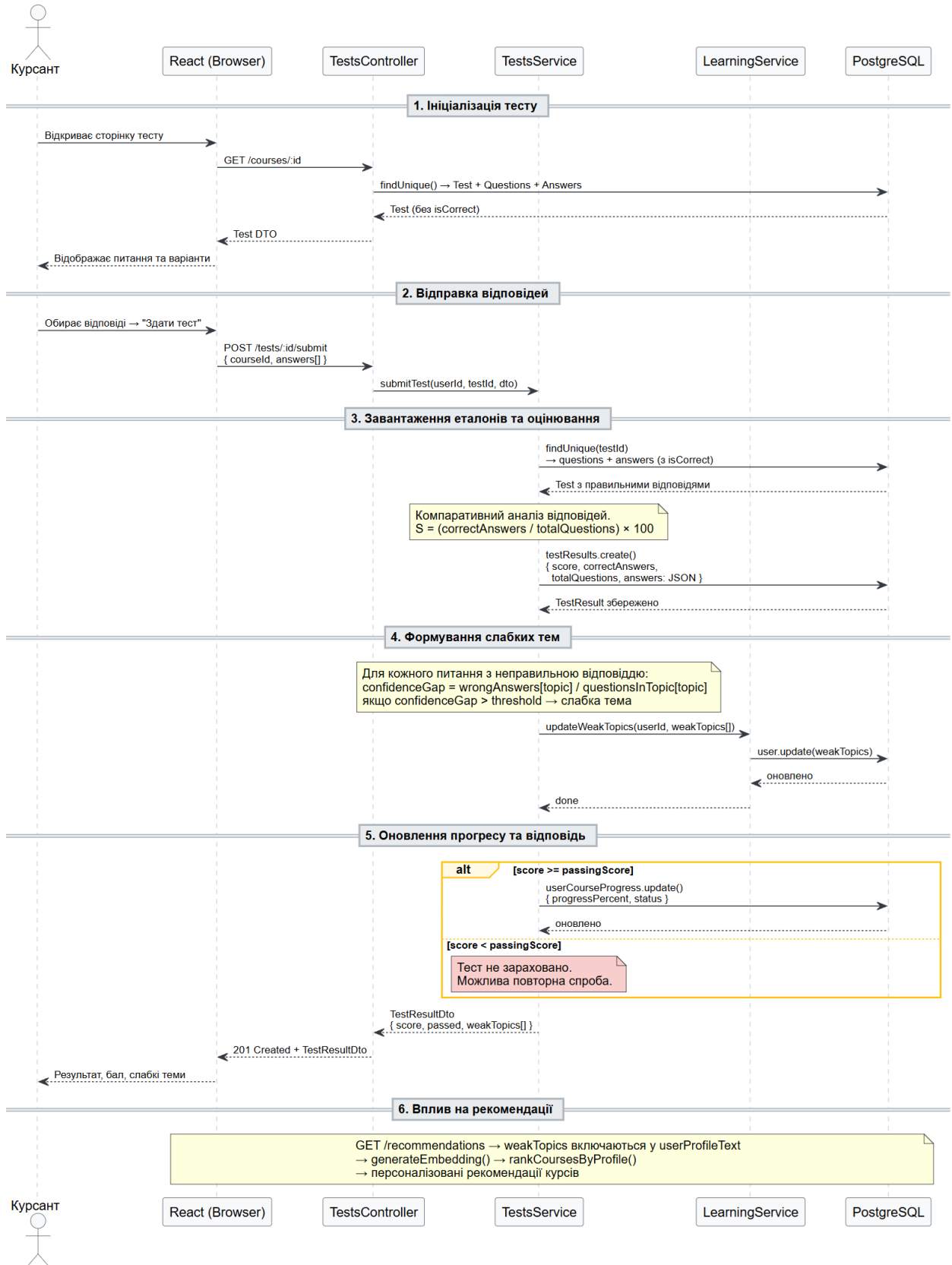
Додаток В

UML-діаграма станів навчального процесу користувача



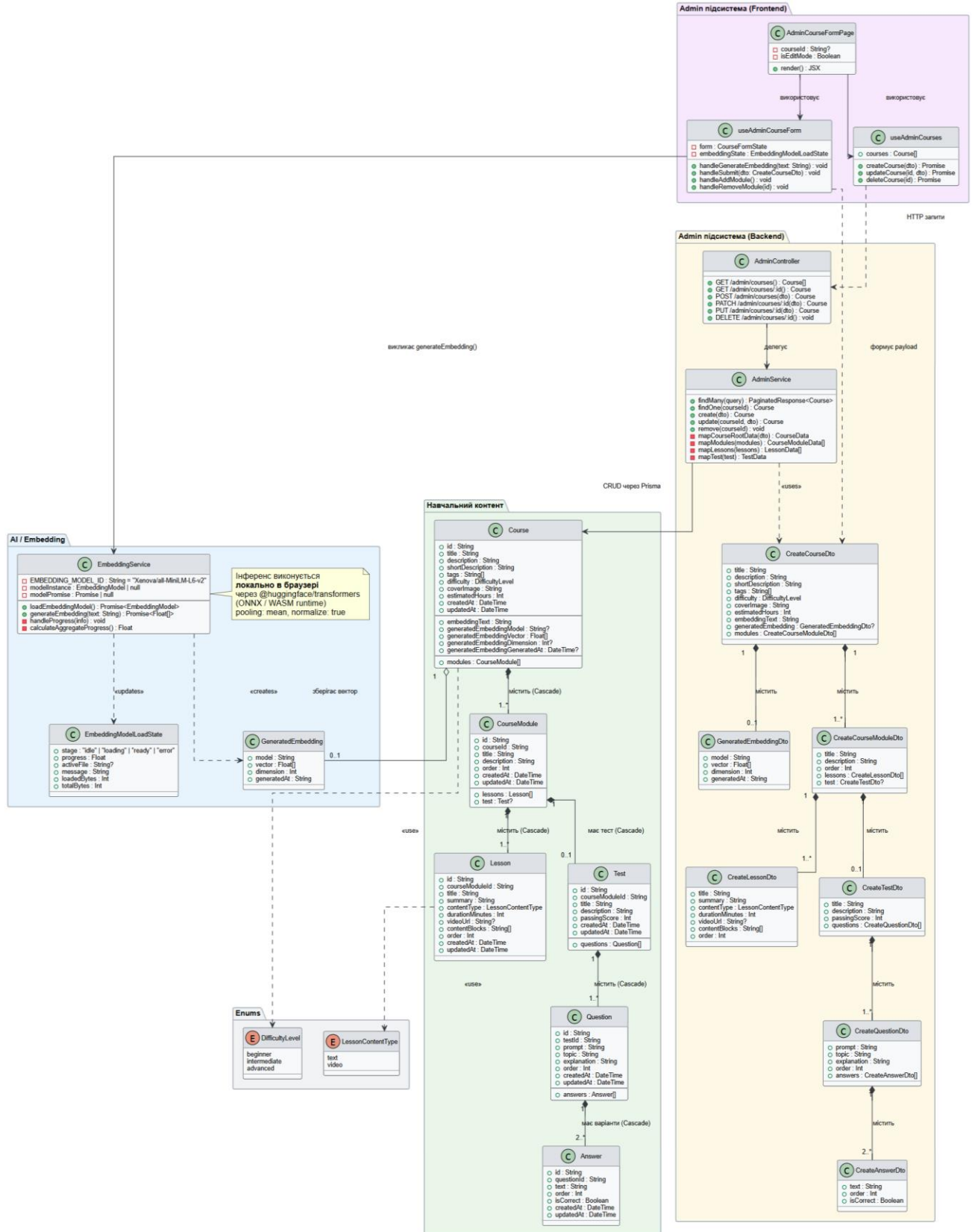
Додаток Д

UML-діаграма послідовності процесу проходження тестування



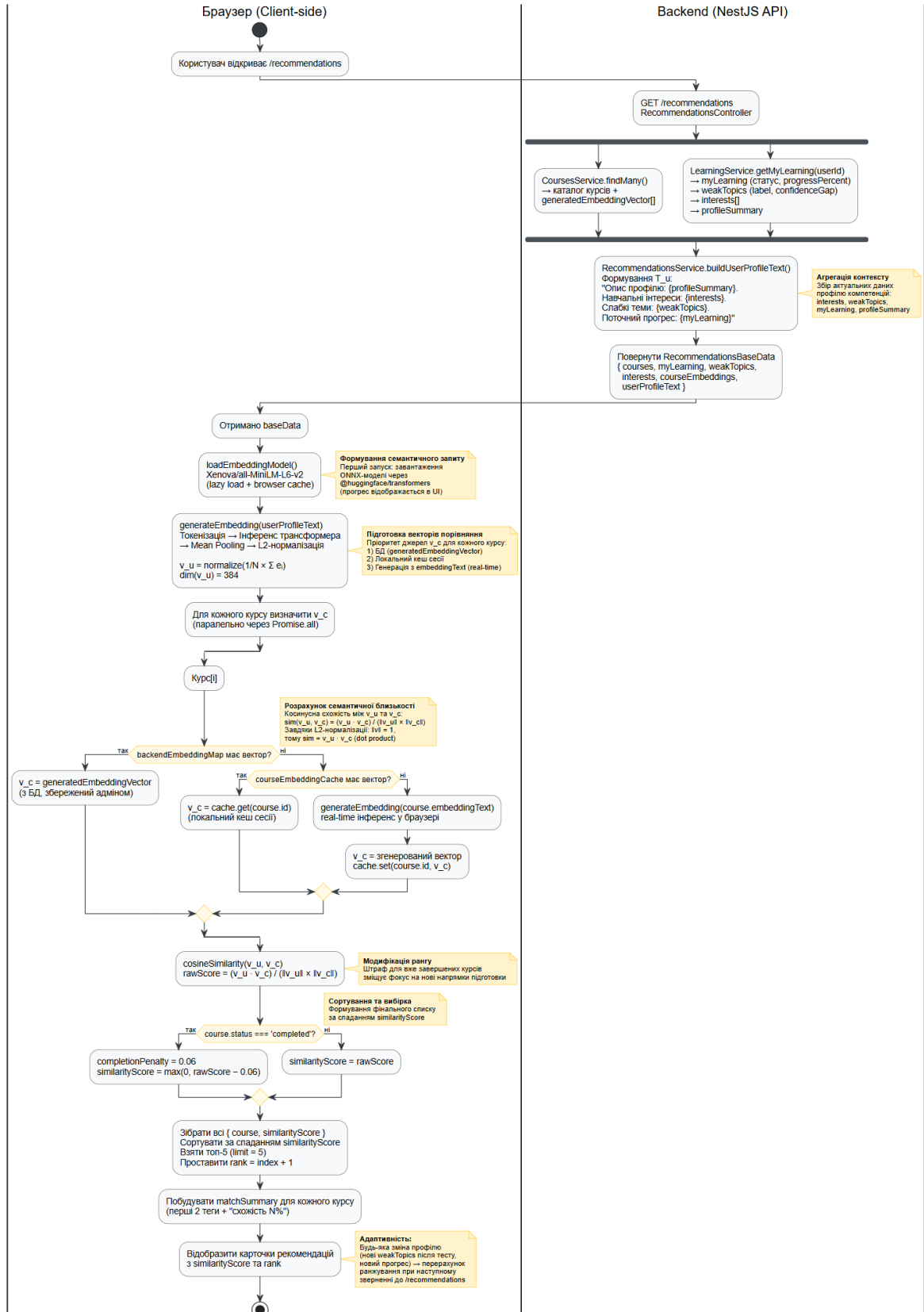
Додаток Е

UML-діаграма класів структури навчального контенту



Додаток Ж

UML-діаграма діяльності алгоритму інтелектуального ранжування курсів



Додаток И

Фрагмент схеми бази даних у файлі schema.prisma

```
generator client {
  provider      = "prisma-client"
  output        = "../src/generated/prisma"
  moduleFormat  = "cjs"
}

datasource db {
  provider = "postgresql"
}

model User {
  id          String          @id
  name        String
  email       String          @unique
  emailVerified Boolean       @default(false)
  image       String?
  role        UserRole        @default(student)
  interests   String[]        @default([])
  profileSummary String      @default("")
  createdAt   DateTime        @default(now())
  updatedAt   DateTime        @updatedAt
  accounts    Account[]
  sessions    Session[]
  courseProgresses UserCourseProgress[]
  lessonProgresses UserLessonProgress[]
  testResults TestResult[]
}

model CourseModule {
  id          String          @id
  courseId    String
  title       String
  description  String
  order       Int
  createdAt   DateTime        @default(now())
  updatedAt   DateTime        @updatedAt
  course      Course          @relation(fields: [courseId], references: [id], onDelete: Cascade)
  lessons     Lesson[]
  test        Test?
  lessonProgresses UserLessonProgress[]
  testResults TestResult[]

  @@unique([courseId, order])
  @@index([courseId])
}
```

Додаток К

Програмна реалізація алгоритму розрахунку прогресу навчання

```
private async recalculateCourseProgress(
  transaction: Prisma.TransactionClient,
  course: CourseWithRelations,
  userId: string,
  courseId: string,
  lastLessonId?: string,
) {
  const progressRecord = await transaction.userCourseProgress.findUnique({
    where: {
      userId_courseId: {
        userId,
        courseId,
      },
    },
    include: {
      lessonProgresses: true,
      testResults: true,
    },
  });

  if (!progressRecord) {
    throw new NotFoundException(
      `Learning progress for course "${courseId}" was not found.`,
    );
  }

  const totalLessons = course.modules.reduce(
    (count, courseModule) => count + courseModule.lessons.length,
    0,
  );
  const totalTests = course.modules.length;
  const passedTests = progressRecord.testResults.filter((testResult) => {
    const matchingModule = course.modules.find(
      (courseModule) => courseModule.id === testResult.courseModuleId,
    );
    return (
      matchingModule?.test?.passingScore !== undefined &&
      testResult.score >= matchingModule.test.passingScore
    );
  }).length;
  const denominator = totalLessons + totalTests;
  const progressPercent =
    denominator === 0
      ? 0
      : Math.min(
          100,
          Math.round(
            ((progressRecord.lessonProgresses.length + passedTests) /
              denominator) *
              100,
          ),
        );
  const hasActivity =
    Boolean(progressRecord.startedAt) ||
    progressRecord.lessonProgresses.length > 0 ||
    progressRecord.testResults.length > 0;
  const status =
    progressPercent >= 100
      ? 'completed'
      : hasActivity
      ? 'in_progress'
      : 'not_started';

  await transaction.userCourseProgress.update({
    where: {
      id: progressRecord.id,
    },
    data: {
      status,
      progressPercent,
      startedAt: progressRecord.startedAt ?? new Date(),
      completedAt:
        status === 'completed'
          ? (progressRecord.completedAt ?? new Date())
          : null,
      lastLessonId:
        lastLessonId !== undefined
          ? lastLessonId
          : progressRecord.lastLessonId,
    },
  });
}
```

Додаток Л

Програмна реалізація алгоритму перевірки результатів тестування

```

async submitTest(
  userId: string,
  courseId: string,
  testOrModuleId: string,
  answers: Record<string, number>,
) {
  const course = await this.coursesService.getCourseEntityById(courseId);
  const moduleAndTest = this.resolveModuleAndTest(course, testOrModuleId);

  this.validateAnswersPayload(answers);

  const result = await this.prismaService.$transaction(
    async (transaction) => {
      const progress = await this.ensureCourseProgress(transaction, userId, courseId);
      const completedLessons = await transaction.userLessonProgress.findMany({
        where: {
          userId, courseId, lessonId: {
            in: moduleAndTest.module.lessons.map((lesson) => lesson.id),
          },
        },
      });

      if (completedLessons.length !== moduleAndTest.module.lessons.length) {
        throw new ConflictException('All module lessons must be completed before submitting the test.');
      }

      const correctAnswers = moduleAndTest.test.questions.reduce(
        (score, question) => {
          const selectedOption = answers[question.id];
          const correctOptionIndex = question.answers.findIndex(
            (answer) => answer.isCorrect,
          );

          return score + Number(selectedOption === correctOptionIndex);
        },
        0,
      );

      const totalQuestions = moduleAndTest.test.questions.length;
      const percentageScore = totalQuestions === 0 ? 0 : Math.round((correctAnswers / totalQuestions) * 100);

      const savedResult = await transaction.testResult.upsert({
        where: {
          userId_testId: { userId, testId: moduleAndTest.test.id },
        },
        create: {
          userId,
          courseId,
          courseModuleId: moduleAndTest.module.id,
          courseProgressId: progress.id,
          testId: moduleAndTest.test.id,
          score: percentageScore,
          correctAnswers,
          totalQuestions,
          answers,
        },
        update: {
          courseId,
          courseModuleId: moduleAndTest.module.id,
          courseProgressId: progress.id,
          score: percentageScore,
          correctAnswers,
          totalQuestions,
          answers,
          submittedAt: new Date(),
        },
        include: { test: true },
      });

      await this.recalculateCourseProgress(transaction, course, userId, courseId);

      return savedResult;
    },
  );

  const learningEntry = await this.getCourseLearningEntry(userId, courseId);

  return {
    result: this.mapTestResult(result),
    learningEntry,
    passingScore: moduleAndTest.test.passingScore,
  };
}

```

Додаток М

Реалізація алгоритму виявлення проблемних тем навчання

```

private buildWeakTopics(
  progressRecords: UserCourseProgressWithRelations[],
): WeakTopicResponse[] {
  const aggregate = new Map<
    string,
    {
      wrong: number;
      total: number;
      relatedCourseIds: Set<string>;
    }
  >();

  for (const progressRecord of progressRecords) {
    for (const testResult of progressRecord.testResults) {
      const parsedAnswers = this.parseAnswersRecord(testResult.answers);

      for (const question of testResult.test.questions) {
        const correctOptionIndex = question.answers.findIndex(
          (answer) => answer.isCorrect,
        );
        const selectedOptionIndex = parsedAnswers[question.id];
        const topicStats = aggregate.get(question.topic) ?? {
          wrong: 0,
          total: 0,
          relatedCourseIds: new Set<string>(),
        };

        topicStats.total += 1;

        if (selectedOptionIndex !== correctOptionIndex) {
          topicStats.wrong += 1;
        }

        topicStats.relatedCourseIds.add(progressRecord.courseId);
        aggregate.set(question.topic, topicStats);
      }
    }
  }

  return [...aggregate.entries()]
    .filter(([_, topicStats]) => topicStats.wrong > 0)
    .map(([label, topicStats]) => ({
      id: `weak-${this.toTopicSlug(label)}`,
      label,
      confidenceGap: Number((topicStats.wrong / topicStats.total).toFixed(2)),
      relatedCourseIds: [...topicStats.relatedCourseIds],
    })))
    .sort((left, right) => right.confidenceGap - left.confidenceGap)
    .slice(0, 8);
}

```

Додаток Н

Реалізація алгоритму формування семантичного профілю користувача

```
private buildUserProfileText(input: {
  myLearning: {
    courseId: string;
    status: string;
    progressPercent: number;
  }[];
  weakTopics: {
    label: string;
    confidenceGap: number;
  }[];
  interests: string[];
  profileSummary: string;
}) {
  const learningSnapshot = input.myLearning
    .map(
      (course) =>
        `${course.courseId}:${course.status}:${course.progressPercent}%`,
    )
    .join('; ');
  const weakTopics = input.weakTopics
    .map((topic) => `${topic.label}:${topic.confidenceGap}`)
    .join('; ');
  const interests = input.interests.join(', ');

  return [
    `Опис профілю: ${input.profileSummary}`,
    `Навчальні інтереси: ${interests}`,
    `Слабкі теми: ${weakTopics}`,
    `Поточний прогрес: ${learningSnapshot}`,
  ].join('. ');
}
```

Додаток П

Реалізація механізму захисту приватних маршрутів

```
export function PrivateRoute({
  children,
  requireAdmin = false,
}: PrivateRouteProps) {
  const { isAuthenticated, isAdmin, isReady, error, refetchSession } = useAuth();

  if (!isReady) {
    return (
      <LoadingState
        title="Відновлюємо сесію"
        description="Перевіряємо поточну Better Auth сесію та cookies."
      />
    );
  }

  if (error) {
    return (
      <ErrorState
        title="Не вдалося перевірити сесію"
        description={error.message}
        onRetry={() => {
          void refetchSession();
        }}
      />
    );
  }

  if (!isAuthenticated) {
    return <Navigate to="/login" replace />;
  }

  if (requireAdmin && !isAdmin) {
    return <Navigate to="/courses" replace />;
  }

  return <>{children}</>;
}
```

Додаток Р

Реалізація механізму завантаження та кешування ШІ-моделі

```
export async function loadEmbeddingModel() {
  if (embeddingModelInstance) {
    emitModelLoadState({
      stage: 'ready',
      progress: 100,
      activeFile: null,
      loadedBytes: null,
      totalBytes: null,
      message: 'Embedding-модель взято з кешу браузерної сесії',
    });
    return embeddingModelInstance;
  }

  if (embeddingModelPromise) {
    return embeddingModelPromise;
  }

  fileProgress.clear();
  emitModelLoadState({
    stage: 'loading',
    progress: 0,
    activeFile: null,
    loadedBytes: null,
    totalBytes: null,
    message: 'Починаємо ініціалізацію embedding-моделі',
  });

  embeddingModelPromise = (async () => {
    const { env, pipeline } = await import('@huggingface/transformers');

    env.allowLocalModels = false;
    env.useBrowserCache = true;

    if (env.backends?.onnx?.wasm) {
      env.backends.onnx.wasm.numThreads = 1;
    }

    const extractor = (await pipeline('feature-extraction', EMBEDDING_MODEL_ID, {
      progress_callback: (progressInfo) => handleProgress(progressInfo as TransformersProgressInfo),
    })) as unknown as EmbeddingModel;

    embeddingModelInstance = extractor;

    return extractor;
  })();

  try {
    return await embeddingModelPromise;
  } catch (error) {
    emitModelLoadState({
      stage: 'error',
      progress: 0,
      activeFile: null,
      loadedBytes: null,
      totalBytes: null,
      message: error instanceof Error ? error.message : 'Не вдалося завантажити embedding-модель.',
    });
    embeddingModelPromise = null;
    throw error;
  }
}
```

Додаток С

Реалізація алгоритму інтелектуального ранжування курсів

```

export async function rankCoursesByProfile({
  baseData,
  limit = 5,
  courseEmbeddingCache,
}: RankCoursesOptions): Promise<RecommendationResult[]> {
  const cache = courseEmbeddingCache ?? new Map<string, number[]>();
  const userProfileText = buildUserProfileText(baseData);
  const userEmbedding = await generateEmbedding(userProfileText);
  const backendEmbeddingMap = new Map(
    baseData.courseEmbeddings.map((entry) => [
      entry.courseId,
      entry.generatedEmbedding?.vector ?? null,
    ]),
  );
  const completedCourseIds = new Set(
    baseData.myLearning.filter((course) => course.status === 'completed').map((course) => course.courseId),
  );
  const rankedCourses = await Promise.all(
    baseData.courses.map(async (course) => {
      const cachedEmbedding = cache.get(course.id);
      const courseEmbedding =
        backendEmbeddingMap.get(course.id) ??
        course.generatedEmbedding?.vector ??
        cachedEmbedding ??
        (await generateEmbedding(course.embeddingText));

      if (courseEmbedding) {
        cache.set(course.id, courseEmbedding);
      }

      const rawScore = cosineSimilarity(userEmbedding, courseEmbedding);
      const completionPenalty = completedCourseIds.has(course.id) ? 0.06 : 0;
      const similarityScore = Math.max(0, rawScore - completionPenalty);

      return {
        course,
        similarityScore,
        rank: 0,
        matchSummary: buildMatchSummary(course, similarityScore),
      };
    }),
  );

  return rankedCourses
    .sort((left, right) => right.similarityScore - left.similarityScore)
    .slice(0, limit)
    .map((course, index) => ({
      ...course,
      rank: index + 1,
    })));
}

```

Додаток Т

Реалізація форми керування навчальними курсами

```
export function CourseEditorForm({ initialValues, submitLabel, onSubmit }: CourseEditorFormProps) {
  const initialFormValues = initialValues ?? createEmptyCourseFormValues()
  const [values, setValues] = useState<AdminCourseFormValues>(initialFormValues)
  const [isSubmitting, setIsSubmitting] = useState(false)
  const [tagsInput, setTagsInput] = useState(initialFormValues.tags.join(', '))
  const [lessonContentInputs, setLessonContentInputs] = useState<Record<string, string>>(
    createLessonContentInputs(initialFormValues.modules),
  )
  const generateEmbeddingMutation = useGenerateEmbeddingMutation()

  useEffect(() => {
    const nextValues = initialValues ?? createEmptyCourseFormValues()
    setValues(nextValues)
    setTagsInput(nextValues.tags.join(', '))
    setLessonContentInputs(createLessonContentInputs(nextValues.modules))
  }, [initialValues])

  useEffect(() => {
    const nextEmbeddingText = buildEmbeddingText(values.title, values.description)
    setValues((current) =>
      current.embeddingText === nextEmbeddingText
        ? current
        : {
            ...current,
            embeddingText: nextEmbeddingText,
          },
    )
  }, [values.description, values.title])

  async function handleSubmit(event: FormEvent<HTMLFormElement>) {
    event.preventDefault()
    setIsSubmitting(true)

    try {
      await onSubmit(values)
    } finally {
      setIsSubmitting(false)
    }
  }
}
```

Додаток У

Механізм генерації семантичних embedding-векторів у адміністративному інтерфейсі

```
export function useGenerateEmbeddingMutation() {
  const [modelLoadState, setModelLoadState] = useState(getEmbeddingModelLoadState);
  const mutation = useMutation({
    mutationFn: async (text: string): Promise<GeneratedEmbedding> => {
      const { generateEmbedding } = await import('@features/ai');
      const rawVector = await generateEmbedding(text);
      const vector = Array.from(rawVector, (value) =>
        | Number(Number(value).toFixed(10)),
      );

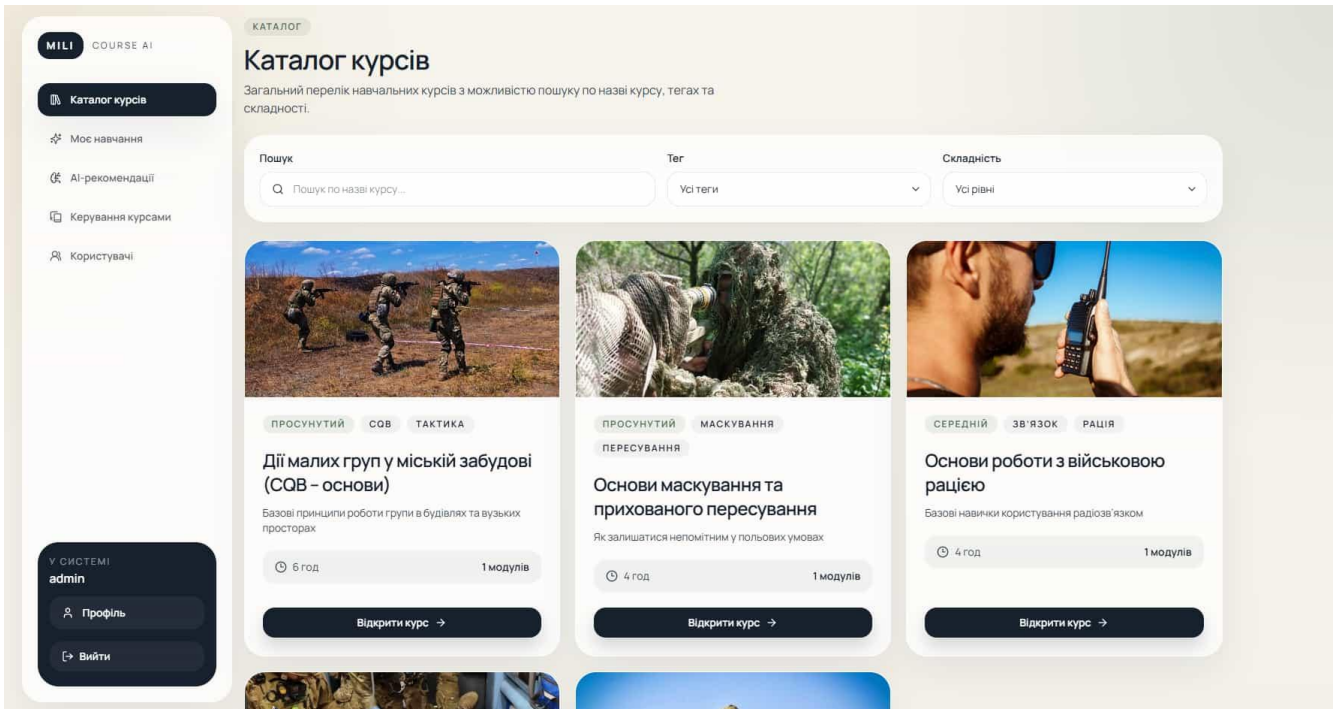
      return {
        model: 'Xenova/all-MiniLM-L6-v2',
        vector,
        dimension: vector.length,
        generatedAt: new Date().toISOString(),
      };
    },
  });

  useEffect(() => {
    return subscribeToEmbeddingModelLoad((nextState) => {
      | setModelLoadState(nextState);
    });
  }, []);

  return {
    ...mutation,
    modelLoadState,
  };
}
```

Додаток Ф

Головна сторінка каталогу курсів та система навігації платформи



Додаток X

Інтерфейс автентифікації та реєстрації користувачів платформи

The image displays two side-by-side login and registration forms for 'Mili Course AI'. Both forms are set against a light beige background with a dark blue header bar at the top of each section.

Left Form: Увійти в систему (Login)

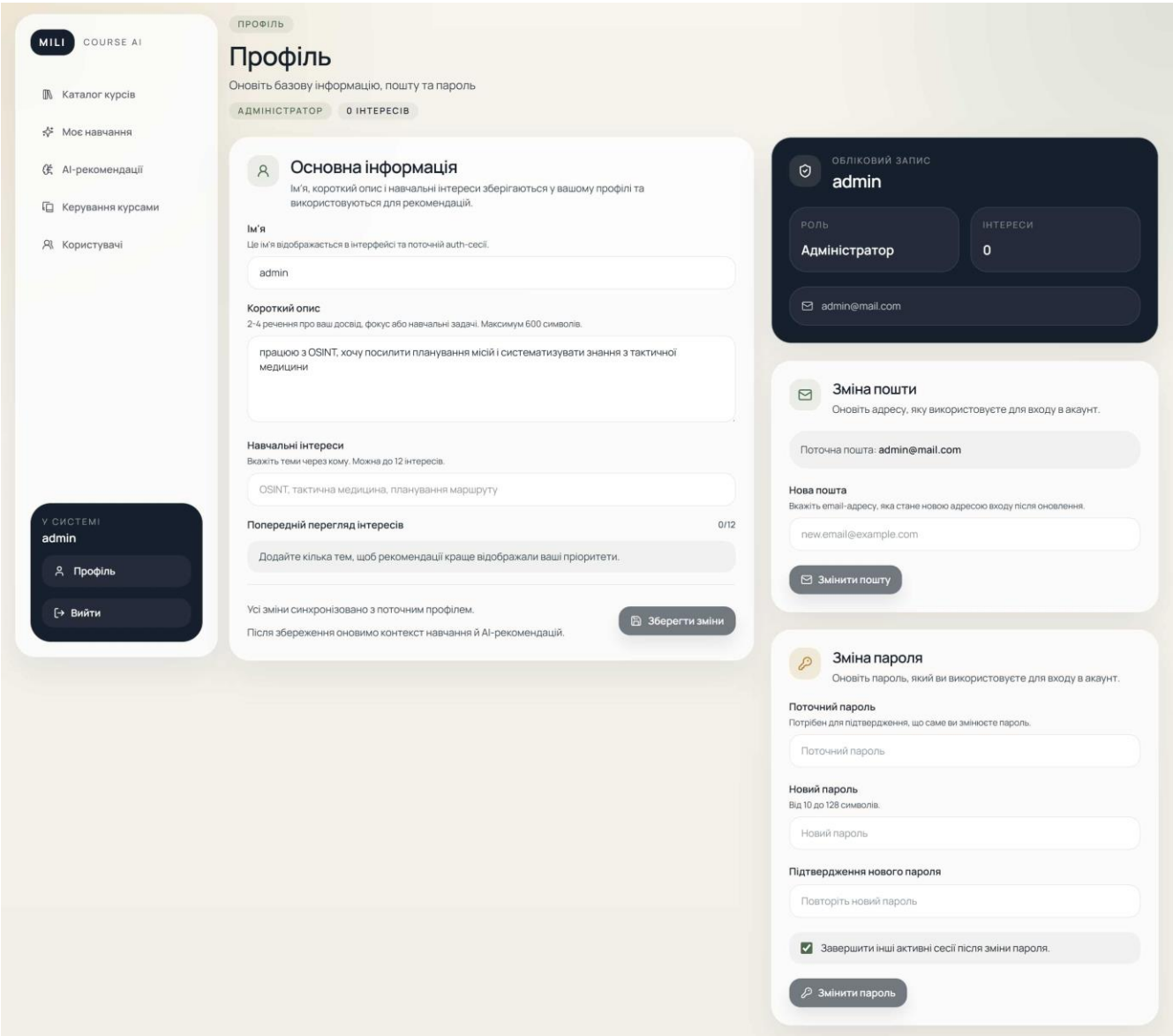
- Header:** Mili Course AI
- Title:** Увійти в систему
- Email:** Input field containing 'admin@mili.local'.
- Пароль (Password):** Input field with masked characters '.....'.
- Button:** Увійти (Login).
- Link:** Немає акаунту? Створити (Don't have an account? Create).

Right Form: Створити акаунт (Create Account)

- Header:** Mili Course AI
- Title:** Створити акаунт
- Ім'я (Name):** Input field containing 'Ім'я'.
- Email:** Input field containing 'email@example.com'.
- Пароль (Password):** Input field with masked characters '.....'.
- Button:** Створити акаунт (Create Account).
- Link:** Вже є акаунт? Увійти (Already have an account? Login).

Додаток Ц

Інтерфейс особистого кабінету користувача



Додаток Ш

Інтерфейс сторінки навчального курсу з метаданими та статусом проходження

MILI

COURSE AI

Каталог курсів

Мое навчання

AI-рекомендації

Керування курсами

Користувачі

У СИСТЕМІ

admin

Профіль

Вийти

ДЕТАЛІ КУРСУ

Тактична медицина

Курс охоплює основи надання першої допомоги в умовах бойових дій відповідно до принципів TCCC (Tactical Combat Casualty Care). Учасники навчатимуться діяти під вогнем, оцінювати стан поранених та застосовувати базові медичні процедури.

ПОЧАТКОВИЙМЕДИЦИНАTCCCPЕРША ДОПОМОГА

Продовжити курс

МОДУЛІ

2

ГОДИНИ

4

СТАТУС

У процесі

Навчальний прогрес

ПРОГРЕС

50%

Що всередині

Вступ до тактичної медицини

2 уроків + тест

Контроль кровотечі

2 уроків + тест

До активного уроку →

МОДУЛЬ 1

Вступ до тактичної медицини

2 уроків

Уроки завершено

Ознайомлення з принципами TCCC та фазами допомоги

УРОК 1

Що таке TCCC

Огляд концепції Tactical Combat Casualty Care

Переглянути ще раз >

УРОК 2

Фази надання допомоги

Детальний розбір фаз

Переглянути ще раз >

Тест модуля

Ще не проходили тест

Почати тест

Перевірка базових знань

МОДУЛЬ 2

Контроль кровотечі

2 уроків

Тест відкриється після всіх уроків

Методи зупинки критичних кровотеч

УРОК 1

Типи кровотеч

Методи зупинки критичних кровотеч

Переглянути ще раз >

УРОК 2

Використання турнікета

Короткий опис уроку.

Відкрити урок >

Тест модуля

Ще не проходили тест

Завершіть уроки модуля

Перевірка засвоєння матеріалу модуля.

Додаток Щ

Інтерфейс сторінки навчального уроку

MILI COURSE AI

Каталог курсів

Моє навчання

AI-рекомендації

Керування курсами

Користувачі

У СИСТЕМІ
admin

Профіль

Вийти

ВІДЕОУРОК

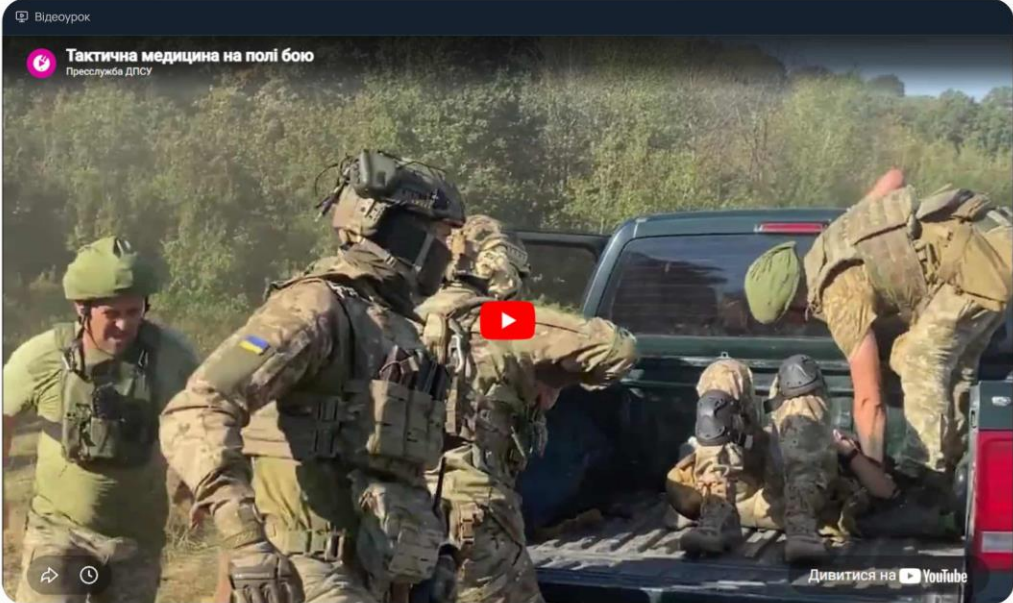
Фази надання допомоги

Детальний розбір фаз

Відеоурок

Тактична медицина на полі бою

Пресслужба ДПСУ



Дивитися на YouTube

Відповідно до протоколів тактичної медицини (TCCC), надання допомоги пораненим на полі бою поділяється на три основні фази, залежно від загрози:

Допомога під вогнем (Care Under Fire, CUF) — червона зона, пріоритет — вогонь у відповідь та зупинка критичних кровотеч.

Допомога в польових умовах (Tactical Field Care, TFC) — жовта зона, надання допомоги в укритті.

Допомога під час евакуації (Tactical Evacuation Care, TACEVAC) — зелена зона, допомога під час транспортування

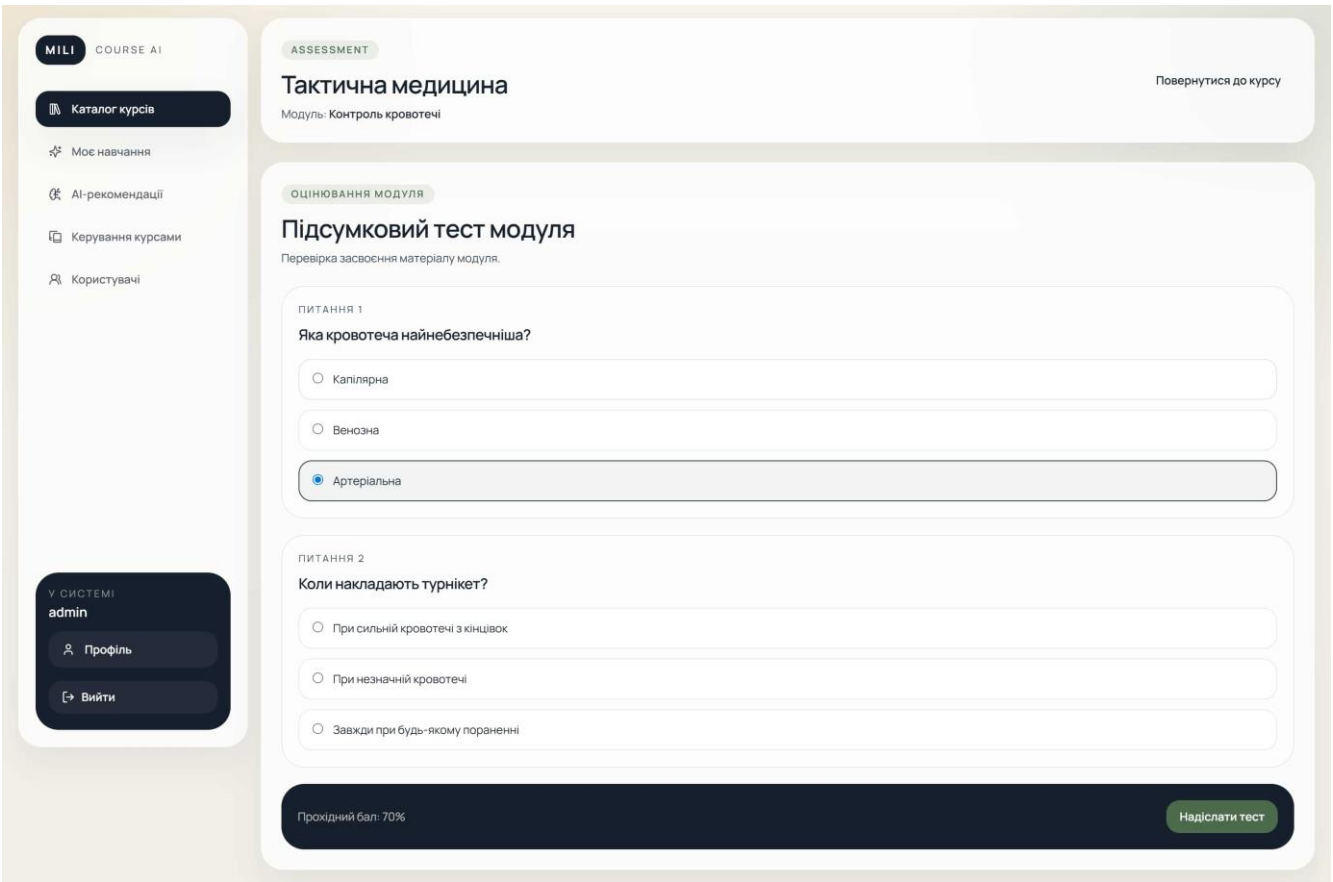
Урок уже завершено. Можна рухатись далі.

← Попередній

Далі →

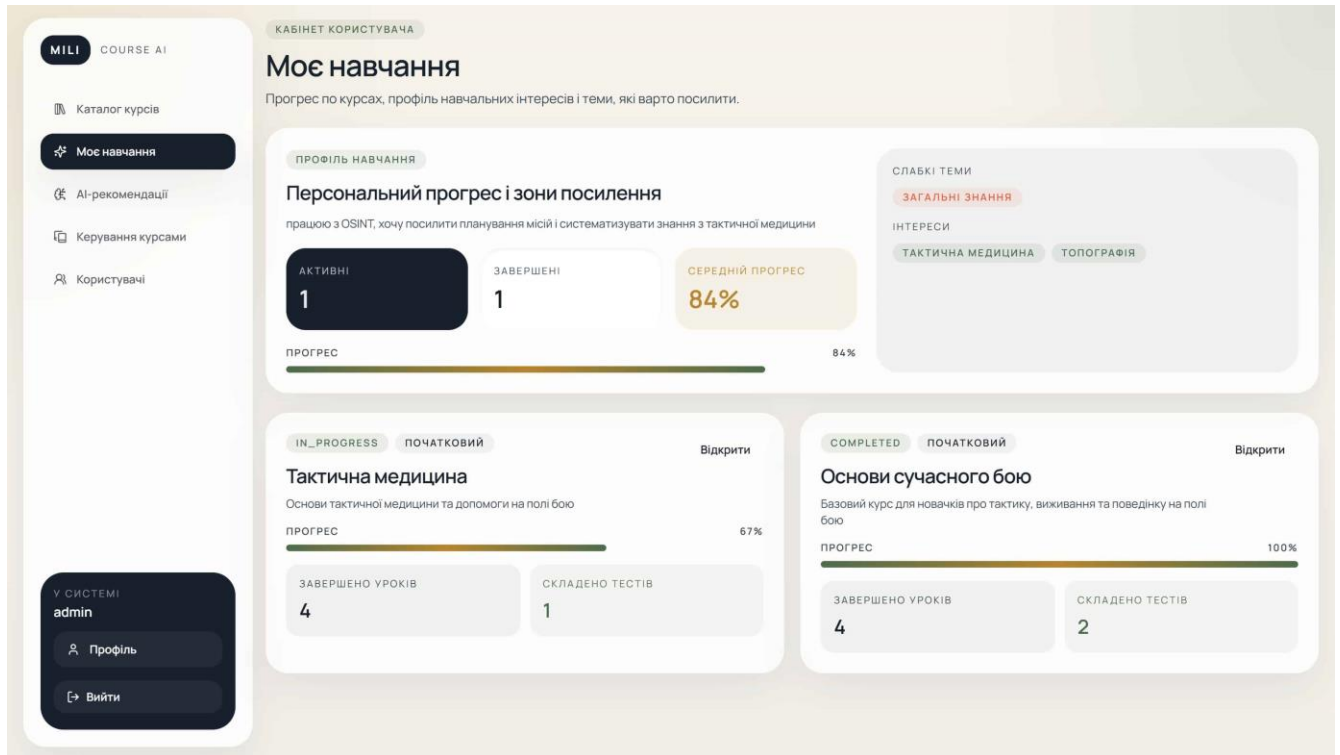
Додаток Ю

Інтерфейс сторінки контролю знань користувача



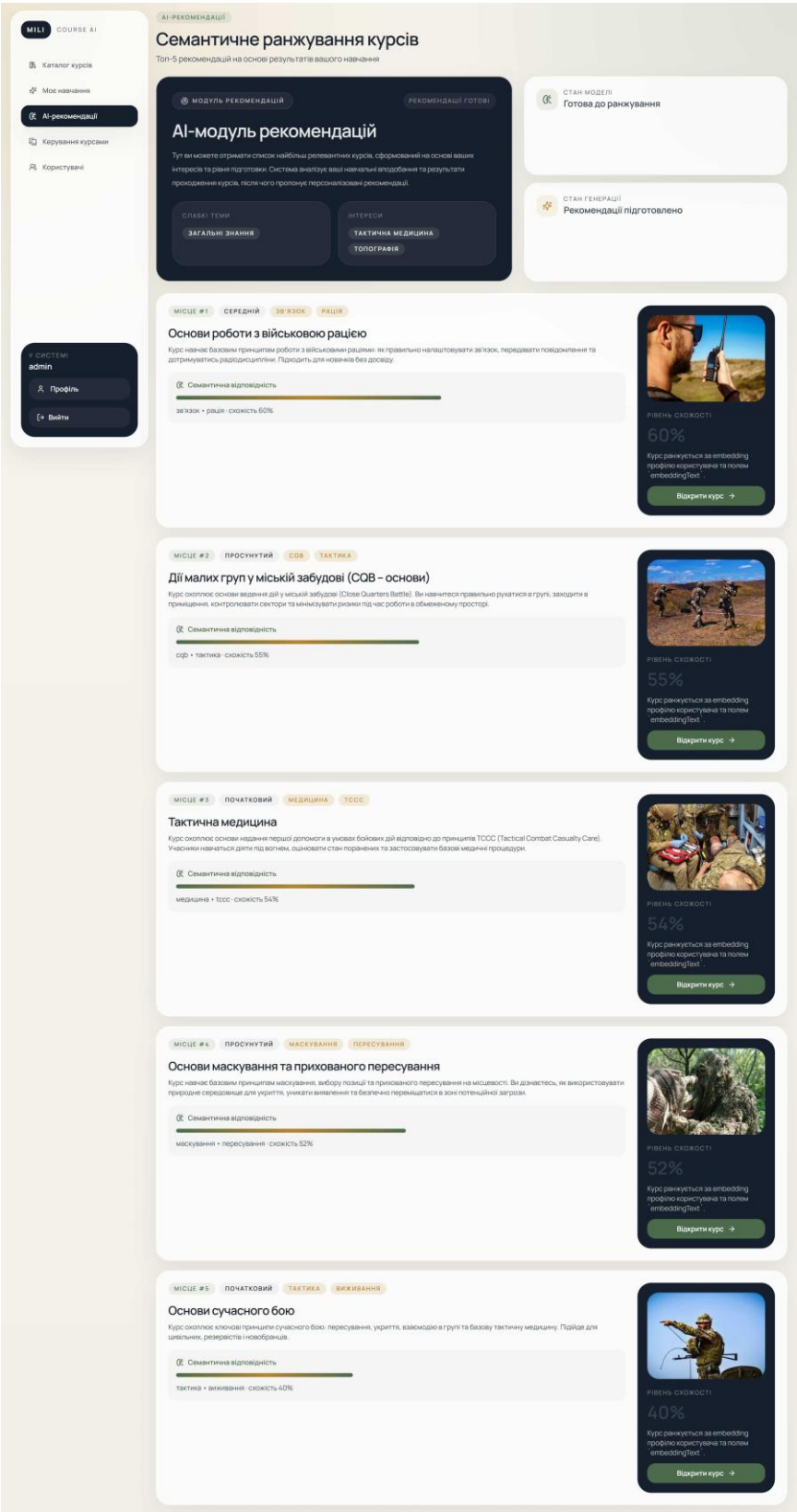
Додаток Я

Інтерфейс сторінки «Моє навчання» та аналітики прогресу



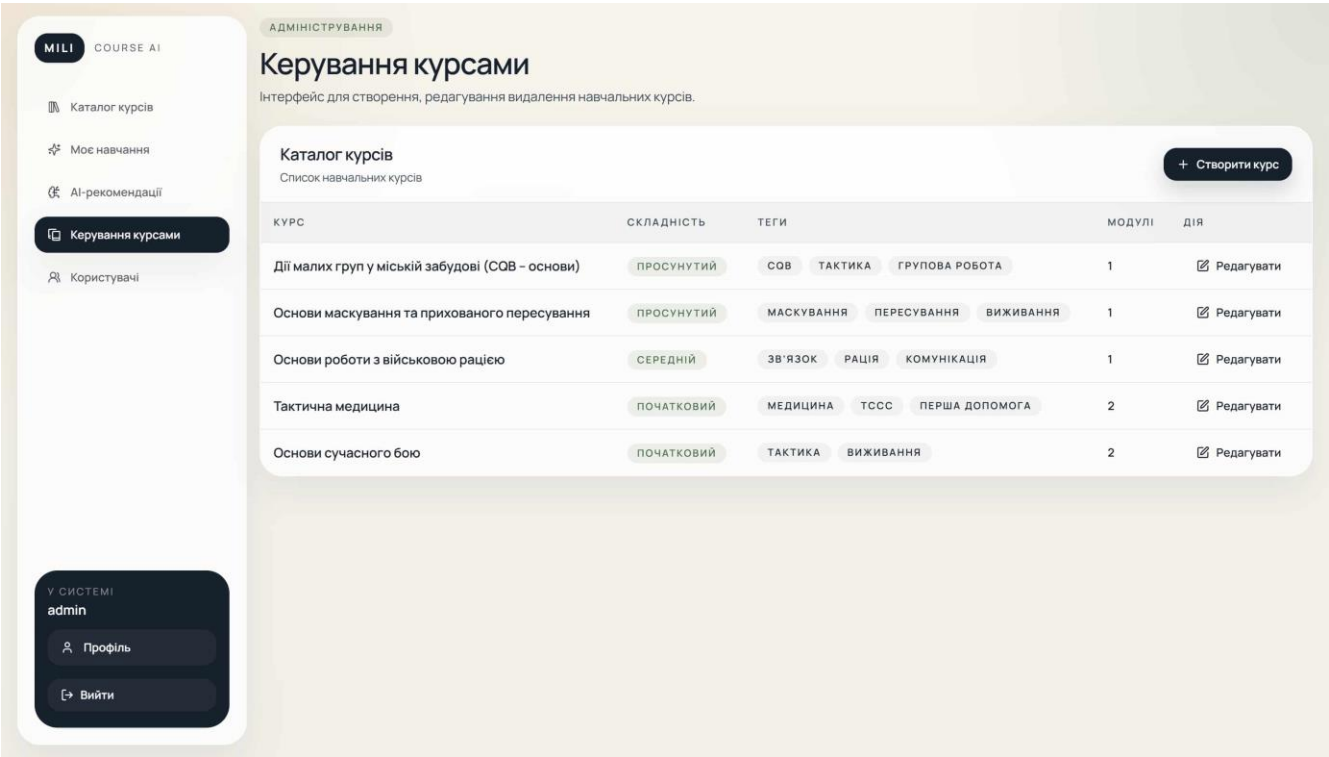
Додаток АА

Інтерфейс сторінки рекомендацій із відображенням релевантності курсів



Додаток АБ

Інтерфейс сторінки управління навчальними курсами



Додаток АВ

Інтерфейс редактора навчального курсу

MLI

COURSE AI

Каталог курсів

Моя навчання

AI-рекомендації

Керування курсами

Користувачі

СИСТЕМА

admin

Профіль

Вийти

АДМІНІСТРУВАННЯ

Редагування курсу

Редактор навчальних курсів, їх модулів та уроків.

Назва

Дії малих груп у міській забудові (СОВ - основи)

Складність

Просунутий

Короткий опис

Короткий текст для картки курсу й адмін-списку.
Базові принципи роботи групи в будівлях та вузьких просторах

URL обкладинки

https://nova.net.ua/wp-content/uploads/2020/06/99.jpg

Опис

Повний опис курсу на сторінці деталей.
Курс охоплює основи ведення дій у міській забудові (Close Quarters Battle). Ви навчитеся правильно рухатися в групі, заходити в приміщення, контролювати сектори та мінімізувати ризики під час роботи в обмеженому просторі.

Теги

Через кому, без дублювання.
срб, тактика, групова робота

Орієнтовна тривалість

6

Текст для embedding

Окремі поля для семантичного пошуку й рекомендацій. Згенерований embedding зберігається у формі окремо від UI-стану.
Текст для embedding
Автоматично формується у вигляді «назва курсу - Опис». Можна редагувати вручну, але після зміни назви або опису поле перегенерується.
Дії малих груп у міській забудові (СОВ - основи). Курс охоплює основи ведення дій у міській забудові (Close Quarters Battle). Ви навчитеся правильно рухатися в групі, заходити в приміщення, контролювати сектори та мінімізувати ризики під час роботи в обмеженому просторі.

Згенерувати embedding

Embedding згенеровано

384 виміри - Xopenai/gpt-4o-mini-L6-v2
Початковий перегляд: [-0.0929, 0.0506, -0.0948, -0.0335, -0.0687, 0.0375, 0.0582, 0.0664]

МОДУЛЬ 1

Виділити модуль

Назва модуля

Основи роботи в приміщенні

Опис модуля

Принципи входу, руху та взаємодії в малих групах

Уроки

Додати урок

Урок 1

Виділити

Назва уроку

Ролі в групі

Короткий опис

розподіл обов'язків

Тип контенту

text

Тривалість у хвиликах

10

Блоки контенту

Один абзац на рядок.
Перший номер – заходити і перевіряє сектор
Другий номер – прикриває інший сектор
Включив постійне координати між боїцями

Урок 2

Виділити

Назва уроку

Вхід у приміщення

Короткий опис

базові техніки входу

Тип контенту

text

Тривалість у хвиликах

10

Блоки контенту

Один абзац на рядок.
Вхід має бути швидким і скоординованим
Кожна відповідь за свій сектор
Не можна зупинятись у двірному проїзді