МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інженерії
програмного забезпечення
_____ Євген ДАВИДЕНКО
«___»_____ 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ МАГІСТРА

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ТРЕНУВАННЯ ШВИДКОСТІ ВВОДУ ДАНИХ НА КЛАВІАТУРІ З ВИКОРИСТАННЯМ ШТУЧНОГО ІНТЕЛЕКТУ

Спеціальність 121 Інженерія програмного забезпечення

Освітня програма «Інженерія програмного забезпечення»

Здобувач _____Костянтин БЄКТІН
«___»_____2024 р.

Керівник канд. техн. наук, доцент _____Євген ДАВИДЕНКО
«___»_____ 2024 р.

Миколаїв – 2024

Чорноморський національний університет імені Петра Могили

(повне найменування закладу вищої освіти)

Факультет	Комп'ютерних наук
Кафедра	Інженерії програмного забезпечення
Рівень вищої освіти	Другий (магістерський)
Освітній ступень	Магістр
Спеціальність	121 Інженерія програмного забезпечення
Освітня програма	Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри інженерії
програмного забезпечення
_____Євген ДАВИДЕНКО
«____»_____2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу здобувача

_____Бєктіна Костянтина Олександровича_____
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи
Програмне забезпечення тренування швидкості вводу даних на клавіатурі з використанням штучного інтелекту.

Затверджена наказом ЧНУ ім. Петра Могили від «04» вересня 2024 р. № 220

2. Строк представлення кваліфікаційної роботи «18» грудня 2024 р.

3. Очікуваний результат роботи та початкові дані якщо такі потрібні
Очікуваним результатом є програмне забезпечення тренування швидкості вводу даних на клавіатурі з використанням штучного інтелекту

4. Перелік питань, що підлягають розробці:
- проведення аналізу доступних систем ШІ для інтеграції та дослідження предметної області;
- формування вимог до розроблюваного програмного забезпечення;
- проєктування програмного забезпечення, створення відповідних моделей;
- розробка програмного забезпечення;
- тестування роботи розробленої системи;
- аналіз зібраних даних щодо результатів розробки.

5. Перелік графічних матеріалів
презентація

Керівник роботи	_____	*Євген ДАВИДЕНКО*
	Особистий підпис	*Власне ім'я ПРІЗВИЩЕ*
Здобувач	_____	*Костянтин БЄКТІН*
	Особистий підпис	*Власне ім'я ПРІЗВИЩЕ*

Дата видачі завдання «30» червня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної роботи

Тема: Програмне забезпечення тренування швидкості вводу даних на клавіатурі з використанням штучного інтелекту _____

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КМР	10.09.2024 р.	11.09.2024 р.	виконано
2.	Огляд літератури за темою роботи	13.09.2024 р.	19.09.2024 р.	виконано
3.	Складання календарного плану КМР	20.09.2024 р.	21.09.2024 р.	виконано
4.	Аналіз предметної області	24.09.2024 р.	26.09.2024 р.	виконано
5.	Розробка проєктних рішень	28.09.2024 р.	30.09.2024 р.	виконано
6.	Моделювання та конструювання ПЗ	01.10.2024 р.	06.10.2024 р.	виконано
7.	Кодування, тестування та апробація розробленого ПЗ, аналіз результатів тестування, розробка керівництва користувача	11.10.2024 р.	07.11.2024 р.	виконано
8.	Оформлення КМР та презентації	10.11.2024 р.	20.11.2024 р.	виконано
9.	Відгук керівника КМР	15.12.2024 р.	15.12.2024 р.	виконано
10.	Попередній захист	28.11.2024 р.	28.11.2024 р.	виконано
11.	Завершення оформлення КМР та презентації	01.12.2024 р.	14.12.2024 р.	виконано
12.	Рецензування	17.12.2024 р.	18.12.2024 р.	виконано
13.	Захист кваліфікаційної роботи	18.12.2024 р.	23.12.2024 р.	виконано

Здобувач _____ **Костянтин БЄКТІН**

«10» __жовтня__ 2024 р.

Керівник роботи

канд. техн. наук,

доцент _____ **Євген ДАВИДЕНКО**

«__» _____ 2024 р.

АНОТАЦІЯ

до кваліфікаційної магістерської роботи

«Програмне забезпечення тренування швидкості вводу даних на клавіатурі з використанням штучного інтелекту»

Здобувач 608м гр.: Бєктін Костянтин

Керівник: канд. техн. наук, доцент Давиденко Євген

Дана робота присвячена вдосконаленню існуючого вебзастосунку, спрямованого на покращення навичок користувачів з володіння клавіатурою, зокрема зменшення кількості помилок та збільшення швидкості вводу даних. Основним нововведенням є інтеграція технологій штучного інтелекту, що дозволяє динамічно підлаштовувати тренувальні завдання під потреби користувачів.

Об'єкт роботи: процес вдосконалення швидкості вводу даних на клавіатурі.

Предмет роботи: програмні засоби для реалізації інтерактивного тренажеру швидкості вводу даних.

Мета: створення сучасного програмного рішення для тренувань швидкості набору тексту з інтеграцією штучного інтелекту, що автоматизує процеси генерації завдань і покращує продуктивність системи.

Для досягнення визначеної мети необхідно виконати наступний перелік завдань:

– аналіз існуючих рішень для інтеграції штучного інтелекту;

– перегляд архітектури застосунку для впровадження даної інтеграції;

– проєктування нових алгоритмів бізнес-процесів застосунку шляхом побудови блок-схем та UML діаграм;

– доповнення макету користувацького інтерфейсу;

– дослідження нових версій використовуваних програмних бібліотек, впровадження нововведень у систему;

– налаштування інтеграції зі штучним інтелектом;

– побудова нових елементів користувацького інтерфейсу за допомогою бібліотеки React;

– оптимізація серверної частини для відповідності новим вимогам

– проведення тестування.

Кваліфікаційна магістерська робота складається з вступу, чотирьох розділів, висновків та переліку джерел посилання.

У вступі визначено актуальність теми, об’єкт і предмет дослідження, а також завдання, необхідні для досягнення мети роботи.

У першому розділі проведено аналіз предметної області, розглянуто аналогічні застосунки, їхні функціональні особливості, переваги й недоліки.

Другий розділ присвячений моделюванню системи. Побудовано UML-діаграми, які демонструють структуру, алгоритми роботи й взаємодію між компонентами системи.

У третьому розділі виконано проєктування системи з урахуванням нових технологій. Описано обраний стек технологій, зокрема використання React для фронтенду, ASP.NET Core для бекенду, Microsoft SQL Server для роботи з даними та інтеграцію штучного інтелекту через OpenAI API.

Четвертий розділ містить опис реалізації функціоналу системи, розгортання інфраструктури на базі Azure та оптимізації серверної частини. Виконано тестування застосунку для перевірки його функціональності, продуктивності та надійності.

Результатом роботи є вдосконалений вебзастосунок, який пропонує інтерактивний тренувальний процес з новими можливостями, заснованими на сучасних технологіях.

КРМ викладена на 81 сторінці, вона містить 4 розділи, 32 ілюстрації, 13 таблиць, 20 джерел в переліку посилань.

Ключові слова: *вебзастосунок, тренування швидкості вводу даних, React, ASP.NET Core, штучний інтелект, Microsoft Azure, OpenAI API, платформа .NET, ChatGPT.*

ABSTRACT

of the Master's Thesis

«Software for typing speed training on keyboard using artificial intelligence»

Student: Biektin Kostiantyn

Supervisor: Candidate of Technical Sciences (PhD), Associate Professor

Davydenko Yevhen

This work is devoted to the improvement of the existing web application, aimed at improving the keyboard skills of users, in particular, reducing the number of errors and increasing the speed of data entry. The main innovation is the integration of artificial intelligence technologies, which allows you to dynamically adjust training tasks to the needs of users.

The object of work: the process of improving the speed of data entry on the keyboard.

The subject of work: software tools for the implementation of an interactive data entry speed simulator.

Goal: creation of a modern software solution for typing speed training with the integration of artificial intelligence, which automates task generation processes and improves system performance.

To achieve the defined goal, the following list of tasks must be completed:

– analysis of existing solutions for the integration of artificial intelligence;

– revision of the application architecture for the implementation of this integration;

– designing new algorithms of application business processes by building block diagrams and UML diagrams;

– supplementing the layout of the user interface;

– research of new versions of used software libraries, introduction of innovations into the system;

– setting integration with artificial intelligence;

– construction of new user interface elements using the React library;

– optimization of the server part to meet new requirements

– testing.

The qualifying master's thesis consists of an introduction, four chapters, conclusions and a list of reference sources.

The introduction defines the relevance of the topic, the object and subject of the research, as well as the tasks necessary to achieve the goal of the work.

In the first chapter, an analysis of the subject area was carried out, similar applications, their functional features, advantages and disadvantages were considered.

New requirements for the functionality and specification of the system being developed are also defined.

The second section is devoted to system modeling. UML-diagrams were built, which demonstrate the structure, work algorithms and interaction between system components.

In the third section, the system is designed taking into account new technologies. A selected technology stack is described, including the use of React for the frontend, ASP.NET Core for the backend, Microsoft SQL Server for working with data, and the integration of artificial intelligence through the OpenAI API.

The fourth section contains a description of the implementation of the system's functionality, the deployment of infrastructure based on Azure, and the optimization of the server part. The application was tested to verify its functionality, performance and reliability.

The result of the work is an improved web application that offers an interactive training process with new capabilities based on modern technologies.

The qualification work is presented on 81 pages, it contains 4 chapters, 32 illustrations, 13 tables, 20 sources in the list of references.

Keywords: *web system, web application, keyboard input speed training, input speed, development using ASP.NET Core, development using React, .NET platform.*

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ПЗ – Програмне Забезпечення

LLM – Large Language Model
UML – Unified Modeling Language
GPT – Generative Pre-trained Transformer
AI – Artificial Intelligence
API – Application Programming Interface
SQL – Structured Query Language
REST – Representational State Transfer
CORS – Cross-Origin Resource Sharing
DI – Dependency Injection
HTTP – Hypertext Transfer Protocol
CRUD – Create, Read, Update, Delete
DOM – Document Object Model
SDK – Software Development Kit
MFA – Multi-Factor Authentication
AWS – Amazon Web Services
ID – Identifier
PaaS – Platform as a Service
CDN – Content Delivery Network
DNS – Domain Name System
CSS – Cascading Style Sheets
XSS – Cross-Site Scripting
HTML – Hyper Text Markup Language
HTTPS – Hyper Text Transfer Protocol Secure
CI – Continuous Integration
CD – Continuous Deployment (Continuous Delivery)

FE – Front-End

M2M – Machine-to-Machine

JWT – JSON Web Token

JSON – JavaScript Object Notation

ORM – Object-Relational Mapping

LINQ – Language Integrated Query

EF – Entity Framework

URL – Uniform Resource Locator

IDE – Integrated Development Environment

DDoS – Distributed Denial-of-Service

WAF – Web Application Firewall

ВСТУП

Актуальність теми кваліфікаційної магістерської роботи обумовлена швидким розвитком сучасних технологій, особливо у сфері штучного інтелекту та машинного навчання. За останні кілька років такі поняття, як «нейронна мережа» та «штучний інтелект», отримали небачений розвиток і поширення, як серед професіоналів галузі інформаційних технологій, так і серед звичайних користувачів. Завдяки впровадженню великих мовних моделей (LLM) і нейронних мереж відбулася революція у багатьох сферах: автоматизація рутинних завдань, оптимізація процесів і покращення продуктивності стали набагато доступнішими та ефективнішими[1].

Зокрема, великі мовні моделі надають унікальні можливості для розробки застосунків, що адаптуються до індивідуальних потреб користувача, генерують персоналізований контент і підвищують ефективність навчання. У контексті тренування швидкості введення тексту на клавіатурі це дозволяє не лише динамічно підлаштовувати завдання під рівень користувача, але й усувати обмеження фіксованих словників, що значно розширює можливості застосунку.

Клавіатура залишається ключовим інструментом взаємодії з комп'ютером, і тренування швидкості та точності її використання є важливим аспектом у підвищенні продуктивності користувачів у різних професійних і побутових сферах. Хоча сучасні інтерфейси користувача еволюціонують, клавіатура не втрачає своєї актуальності, особливо в роботі з текстом.

За основу даної роботи взято попередньо розроблену систему «typro», яка не втрачає актуальності і по сьогоднішній день, оскільки клавіатура лишається одним з провідних пристроїв взаємодії з комп’ютером. Застосування штучного інтелекту дозволить не лише збільшити продуктивність системи, а й спростити її супровід і зменшити обсяг кодової бази. Це важливо з огляду на сучасні вимоги до розробки програмного забезпечення, де оптимізація коду і підтримка довготривалих проектів відіграють ключову роль у забезпеченні їх надійності та ефективності.

Об’єкт роботи: процес вводу даних на клавіатурі.

Предмет роботи: програмні засоби реалізації тренажеру тренування вводу даних на клавіатурі.

Мета: покращення системи тренувань швидкості вводу даних на клавіатурі шляхом делегації виконання певних задач нейронній мережі.

Для досягнення визначеної мети необхідно виконати наступний **перелік завдань**:

– аналіз існуючих рішень для інтеграції штучного інтелекту;

– перегляд архітектури застосунку для впровадження даної інтеграції;

– проєктування нових алгоритмів бізнес-процесів застосунку шляхом побудови блок-схем та UML діаграм;

– доповнення макету користувацького інтерфейсу;

– дослідження нових версій використовуваних програмних бібліотек, впровадження нововведень у систему;

– налаштування інтеграції зі штучним інтелектом;

– побудова нових елементів користувацького інтерфейсу за допомогою бібліотеки React[2];

– оптимізація серверної частини для відповідності новим вимогам;

– проведення тестування.

Апробація результатів КМР відбулась під час XXVII Всеукраїнської науково-практичної конференції «Могилянські читання – 2024», Миколаїв, 06-10 листопада 2024 р. (Додаток Б).

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз існуючих рішень для інтеграції штучного інтелекту

Сучасний ринок засобів штучного інтелекту стрімко розвивається і залишається одним із найдинамічніших напрямків технологічного прогресу. Щотижня, а іноді й щодня, світ дізнається про нові досягнення провідних компаній, стартапів та незалежних дослідників. Вони охоплюють широкий спектр галузей – від обробки природної мови та зображень до робототехніки, медицини та автоматизації бізнес-процесів. Розробники штучного інтелекту невпинно вдосконалюють існуючі моделі, знаходячи нові застосування для машинного навчання та нейронних мереж.

Завдяки цьому ми спостерігаємо появу великого різноманіття мовних моделей, кожна з яких розроблена для виконання певних завдань. Моделі штучного інтелекту відіграють ключову роль у багатьох сферах, включаючи генерацію тексту, аналіз даних, автоматизацію рутинних процесів та персоналізацію користувацького досвіду. Особливо важливим є те, що ці рішення вже не є ексклюзивними для великих корпорацій — доступ до штучного інтелекту стає дедалі ширшим завдяки відкритим платформам, таким як Hugging Face[3], OpenAI, та численним API, що спрощують інтеграцію AI у вебсистеми та бізнес-продукти.

У такому різноманітті моделей та рішень важливо розуміти, які з них найкраще задовільнять ваші вимоги. Аналіз наявних варіантів дає змогу вибрати оптимальну модель для вирішення таких задач, як генерація тексту, класифікація, прогнозування та інші[4].

З вибірки лідерів на момент написання даної роботи було обрано п’ять моделей для порівняння:

1. GPT-4o;
2. BERT;
3. Bloom;
4. LLaMA;

5. Gemini.

Порівняльну характеристику представлено у таблиці 1.1.

Таблиця 1.1 – Порівняльна характеристика існуючих рішень для інтеграції штучного інтелекту

	GPT-4o mini	**BERT**	**Bloom**	**LLaMA**	**Gemini**
Розробник	OpenAI	Google	BigScience	Meta (Facebook)	Google DeepMind
Тип моделі	Генеративна модель (GPT)	Трансформер (енкодер)	Генеративна модель (GPT-подібна)	Генеративна модель (GPT-подібна)	Мультикомпонентна модель AI
Архітектура	Трансформер, оптимізований GPT-4	Трансформер, тільки енкодер	Трансформер	Трансформер	Трансформер з інтеграцією інших технологій
Призначення	Генерація тексту, адаптивна до завдань	Контекстне розуміння, класифікація тексту	Генерація тексту	Генерація тексту, моделювання	Генерація тексту, мультимодальні завдання
Особливості	Оптимізована версія GPT-4, швидка	Глибоке розуміння контексту	Масштабована модель, багатомовність	Висока точність з меншими обчислювальними витратами	Інтеграція мультимодальних функцій
Навчальні дані	Тренована на великих текстових корпусах	Величезний корпус текстів	Багатомовні дані, 176B параметрів	Мільярди параметрів, оптимізовані для точності	Великі мультимодальні корпуси (зображення, текст)
Параметри	Менше параметрів у версії mini	110M до 345M параметрів	176B параметрів	Від 7B до 65B параметрів	Не розкрито, але великі

Кінець таблиці 1.1

Підтримувані мови	Багатомовна	Англійська (може бути налаштована)	46 мов	Багатомовна	Мультимовна підтримка
Швидкість	Висока продуктивність, оптимізована	Висока для класифікації	Менш оптимізована за швидкістю	Оптимізована для продуктивності	Висока продуктивність
Основне застосування	Генерація тексту, адаптація до користувача	Класифікація, заповнення пропусків	Генерація тексту	Генерація, моделювання	Генерація тексту, мультимодальні завдання
Переваги	Швидка генерація, висока точність	Висока точність у контекстних завданнях	Підтримка багатьох мов	Висока продуктивність	Інтеграція різних AI-компонентів
Обмеження	Менше параметрів у mini версії	Не генерує текст	Вимагає великих ресурсів	Високі вимоги до ресурсів	Новий продукт, обмежена доступність
Масштабованість	Висока	Добре масштабується	Дуже добре масштабується	Дуже масштабована	Очікується висока
Ліцензія	Приватна	Відкрита (Apache 2.0)	Відкрита (Apache 2.0)	Доступно для досліджень	Приватна
Наявність API для інтеграції	Так (через OpenAI API)	Ні (локальна реалізація)	Так (через Hugging Face API)	Ні (локальний запуск)	Так
Легкість інтеграції	Висока (через OpenAI API)	Середня (потребує налаштування)	Середня (через Hugging Face)	Низька (потребує локального запуску)	Очікується висока
Наявність офіційного .NET пакету	Так	Ні	Ні	Ні	Ні

В результаті порівняння, з врахуванням всіх переваг та недоліків було обрано варіант GPT-4o mini.

1.2 Аналіз системи, що розробляється

У порівнянні з початковим станом вебзастосунку, нова версія доповнюється функціоналом, пов’язаним зі штучним інтелектом. Це впровадження значно підвищує гнучкість і адаптивність тренувального процесу, дозволяючи системі пропонувати користувачам більш динамічні сценарії тренувань.

Система зберігає основні принципи роботи, однак тепер вона здатна ефективніше підлаштовуватися під задані користувацькі параметри, що сприяє більш інтерактивному і продуктивному навчальному процесу.

Розширення функціоналу також вносить зміни до основних сценаріїв використання системи, відкриваючи нові можливості для її користувачів. Це включає більшу варіативність і адаптивність у виборі завдань, що сприяє підвищенню ефективності тренувального процесу.

Нова версія вебзастосунку значно покращує існуючу інфраструктуру та відкриває нові перспективи для покращення користувацького досвіду завдяки інтеграції сучасних рішень штучного інтелекту. Опис розроблюваної системи бере за основу відповідну таблицю з попередньої роботи та містить в собі нові пункти, що обумовлені нововведеннями. Таким чином повністю передається очікувана функціональність системи (див. табл. 1.2).

Таблиця 1.2 – Опис розроблюваної системи

Основні задачі	1. введення даних на клавіатурі з одночасним відображенням продуктивності вводу, помилок, автоматичним генеруванням тексту, за потреби; 2. надання користувачу переліку різних режимів та налаштувань тренування; 3. надання користувачу статистики щодо успішності його тренувань як у різних режимах так і за обрані періоди часу (день, тиждень, місяць тощо);

Кінець таблиці 1.2

Основні задачі	4. проведення групових тренувань, коли декілька користувачів суперничають між собою в наборі обраного обсягу тексту; 5. відображення списку лідерів за різними критеріями (наприклад, за режимами); 6. реєстрація нових користувачів; 7. авторизація зареєстрованих користувачів; 8. налаштування теми інтерфейсу; 9. налаштування даних профілю.
Користувачі системи	1. гість; 2. зареєстрований користувач.
Сценарії роботи системи	1. користувач проходить тренування з введення тексту; 2. користувач додає пунктуацію до генерованого тексту; 3. користувач додає цифри до генерованого тексту; 4. користувач змінює мову генерованого тексту; 5. користувач обирає режим тренування (на час чи на кількість слів); 6. користувач виконує налаштування свого профілю; 7. користувач переглядає списки лідерів серед користувачів за режимами. 8. користувач надає побажання щодо генерації тексту для тренування шляхом використання штучного інтелекту.
Засоби апаратної та програмної реалізації	1. ASP.NET Core[5]; 2. React; 3. SQL Server; 4. ChatGPT-4o mini LLM.
Вихідні дані	1. графіки успішності користувача за різними режимами; 2. текстове представлення даних про найуспішніші тренування; 3. текстове представлення про кількість проведених тренувань за режимами.

Усі пункти таблиці 1.2, що виділені жирним шрифтом підлягають змінам враховуючи нововведення. Це більше ніж половина початкових сценаріїв роботи системи.

Рисунок 1.1 – Макет оновленого інтерфейсу застосунку «typro»

На рис. 1.1 зображено макет нових елементів користувацького інтерфейсу. До них входить іконка нового режиму з поміткою «beta» (виділено зеленим кольором), поле для вводу запиту до штучного інтелекту та блок з отриманою відповіддю від нього. Додаткові елементи інтерфейсу було виключено з початкового дизайну.

1.3 Специфікація вимог до розроблюваної системи

Призначення системи, для якої розробляється програмне забезпечення

Призначення розроблюваної системи полягає у автоматизації тренування швидкості вводу даних на клавіатурі.

Погодження, ухвалені в програмній документації

Було узгоджено, що для покращення розроблюваного програмного забезпечення будуть використані фреймворк ASP.NET Core для побудови веб API, бібліотека для побудови елементів користувацького інтерфейсу React та база даних Microsoft SQL Server[6].

Межі проєкту ПЗ

Крайня дата завершення робіт над проєктом – 18.12.2024р.

Сфера застосування

Це програмне забезпечення не має жорстких обмежень щодо сфери використання, що робить його універсальним інструментом для широкого кола користувачів. У сучасному технологічному середовищі комп'ютери є невід'ємною частиною майже кожної професійної галузі — від офісної роботи, програмування та дизайну до наукових досліджень, освіти й медицини. Використання комп'ютерів значно підвищує продуктивність у багатьох сферах діяльності, де ефективне введення даних на клавіатурі є критично важливим навиком.

Характеристики користувачів

Користувачеві потрібні кілька ключових ресурсів для повноцінного використання програмного забезпечення. По-перше, необхідний надійний і стабільний доступ до Інтернету, оскільки більшість функцій програми залежить від постійного підключення до мережі. По-друге, потрібна клавіатура, яка може бути як фізичною, так і віртуальною (екранною), залежно від типу пристрою, що використовується.

Крім цього, користувачеві знадобиться відповідний пристрій для роботи з програмою. Це може бути смартфон, ноутбук або персональний комп'ютер з монітором, що дозволяє забезпечити комфортне введення тексту та взаємодію з інтерфейсом програми. Незалежно від типу гаджета, програмне забезпечення адаптоване до різних платформ, що дозволяє користувачам обирати найбільш зручний для них спосіб тренування.

Загальна структура та склад системи

Система включає у себе наступні компоненти:

1. клієнт (застосунок на базі бібліотеки React);
2. сервер (сукупність застосунків на базі фреймворку ASP.NET Core);
3. сховище даних (база даних SQL Server).

Обмеження

Головним обмеженням є наявність доступу до мережі Інтернет у користувача.

Функції вебзастосунку тренування швидкості вводу даних на клавіатурі

Функція обробки користувацького вводу під час тренування

Опис функції

Функція обробки користувацького вводу є провідною у роботі даної системи. За допомогою неї користувач отримує візуальний відгук щодо правильності введених даних під час тренування.

Вхідна та вихідна інформація

Вхідна інформація – код натиснутої користувачем клавіши на клавіатурі.

Вихідна інформація – графічне відображення відповідності надісланого коду клавіши до очікуваного.

Функціональні вимоги

Сховище даних зі словами для тренувань, доступ до мережі Інтернет.

Функція генерації слів для тренування за словесним запитом користувача

Опис функції

Функція генерації слів для тренування за запитом користувача є ключовим елементом в системі, що забезпечує користувачам індивідуально підібраний набір слів для тренування швидкості набору тексту. Ця функція дозволяє динамічно створювати набори слів на основі заданого користувачем словесного запиту.

Вхідна та вихідна інформація

Вхідна інформація – словесний запит користувача.

Вихідна інформація – набір слів для проходження тренування.

Функціональні вимоги

Необхідний стабільний доступ до мережі Інтернет.

Функція перегляду користувацької статистики пройдених тренувань

Опис функції

Ця функція надає зареєстрованим користувачам можливість отримати детальну інформацію про результати своїх тренувань, що допомагає ефективніше відслідковувати прогрес. Вона спрямована на аналіз досягнень користувача і відображення відповідної статистики.

Вхідна та вихідна інформація

Вхідна інформація – Ідентифікатор користувача та інформація про тренування, які він пройшов.

Вихідна інформація – Текстова та графічна статистика, що демонструє рівень успішності користувача.

Джерела та зміст вхідної інформації (даних)

У цьому програмному забезпеченні основним джерелом інформації виступає користувач. Своїми діями він створює нові дані для збереження у базі даних, такі як реєстрація облікового запису чи завершене тренування.

Нормативно-довідкова інформація (класифікатори, довідники тощо)

Вимоги до даного пункту відсутні.

Вимоги до технічного забезпечення

Технічні вимоги до обладнання в сучасних умовах є досить лояльними і не передбачають серйозних обмежень. Для комфортної роботи з програмним забезпеченням пристрій повинен мати стабільний доступ до Інтернету та бути обладнаний екраном для відображення інтерфейсу. Крім того, пристрій має підтримувати фізичну або екранну клавіатуру для введення тексту. Важливою умовою є наявність сучасного браузера, такого як Google Chrome, Mozilla Firefox, Opera або аналогічного, що підтримує актуальні вебстандарти.

Вимоги до програмного забезпечення

Архітектура програмної системи

До складу системи входять клієнтський застосунок, низка REST API вебсерверів та база даних.

Системне програмне забезпечення

Клієнтська частина застосунку буде розроблена з використанням бібліотеки React, що забезпечує швидкий і гнучкий інтерфейс користувача. Вебсервери для обробки запитів та обміну даними через REST API будуть реалізовані на базі фреймворку ASP.NET Core, що гарантує високу продуктивність і надійність у роботі з сервісами. В якості провайдера баз даних обрано Microsoft SQL Server, який забезпечить ефективне зберігання та управління даними, дозволяючи легко масштабувати систему при зростанні обсягу даних.

Мережне програмне забезпечення

Для розробки програмного забезпечення вибрано операційну систему Windows 11, яка забезпечує стабільну платформу для роботи з сучасними інструментами розробки. Редактор коду Visual Studio Code буде використовуватися для написання і налагодження клієнтської частини, тоді як інтегроване середовище розробки (IDE) Visual Studio 2022 буде використовуватися для розробки серверної частини на базі ASP.NET Core. Для тестування і перегляду результатів застосовується браузер Google Chrome, що підтримує всі необхідні стандарти та інструменти для веброзробки.

Програмне забезпечення ведення інформаційної бази

У якості сховища даних обрано Microsoft SQL Server, що надає потужні засоби для управління великими обсягами інформації. Він забезпечує надійне зберігання, швидкий доступ до даних та підтримку складних запитів, що є важливим для забезпечення ефективної роботи системи. Додатково, вибір Microsoft SQL Server гармонійно узгоджується із загальним використанням платформи .NET у розробці. Завдяки цьому досягається максимальна інтеграція між сервером бази даних та бекендом, що реалізований за допомогою ASP.NET Core.

Microsoft SQL Server повністю підтримує Entity Framework — ORM-фреймворк, інтегрований у .NET, який дозволяє розробникам працювати з даними у вигляді об’єктів, не створюючи вручну складні SQL-запити. Така інтеграція спрощує роботу з базою даних, скорочує час розробки та зменшує кількість

помилок у коді. Крім того, використання LINQ (Language Integrated Query), який також підтримується .NET, забезпечує додаткову зручність і гнучкість у написанні запитів до бази даних.

Використання Microsoft SQL Server у поєднанні з платформою .NET забезпечує не лише продуктивність і надійність, але й оптимізує процес розробки завдяки єдиній екосистемі технологій Microsoft. Такий підхід дозволяє розробникам ефективно використовувати можливості платформи, інтегрувати різні компоненти проєкту без додаткових складнощів та легко підтримувати систему у майбутньому.

Мова і технологія розробки ПЗ

Розробка програмного забезпечення буде виконана з використанням сучасних технологій: бібліотека React забезпечить високопродуктивний інтерфейс користувача, тоді як фреймворк ASP.NET Core стане основою для серверної частини, відповідальної за обробку серверних запитів і бізнес-логіку. Основними мовами програмування будуть TypeScript для клієнтської частини і C#[7] для серверної частини, що дозволить забезпечити високий рівень продуктивності та надійності програмного забезпечення.

Вимоги до зовнішніх інтерфейсів

Інтерфейс користувача

Інтерфейс користувача має бути розроблений відповідно до сучасних стандартів для забезпечення зручного та інтуїтивного користувацького досвіду. Основна структура інтерфейсу повинна бути логічно поділена на кілька ключових частин: навігаційну панель, контейнер для відображення основного контенту та нижню панель з додатковою інформацією чи функціями. Це дозволить користувачеві легко орієнтуватися в системі, а головний контент залишатиметься динамічним, змінюючись відповідно до дій користувача, при цьому навігаційна та нижня панелі залишатимуться статичними. Такий підхід забезпечує максимальну зручність використання і зменшує навантаження на користувача під час роботи із застосунком.

Апаратний інтерфейс

Апаратний інтерфейс буде представлений пристроєм, який користувач використовує для взаємодії з клієнтським додатком. Це може бути будь-який сучасний гаджет, такий як смартфон, планшет, ноутбук або стаціонарний комп'ютер, який забезпечує доступ до Інтернету та підтримує веббраузери для роботи зі застосунком.

Програмний інтерфейс

Програмний інтерфейс системи базується на бібліотеці React та фреймворку ASP.NET Core, які забезпечують високоякісне середовище для розробки. Завдяки інструментам відлагодження та моніторингу, що надаються цими фреймворками, усі помилки або пропозиції щодо покращення коду оперативно фіксуються та передаються розробникам для швидкого усунення. Це дозволяє підтримувати високу якість коду, швидко реагувати на проблеми та вносити необхідні зміни, що сприяє стабільній роботі застосунку та постійному вдосконаленню його функціоналу.

Властивості програмного забезпечення

Безпека

Аутентифікація користувачів здійснюватиметься за допомогою комбінації JWT токенів доступу та токенів оновлення, що забезпечує надійний захист від несанкціонованого доступу та спроб підміни даних. Використання цих токенів дозволяє контролювати сесії користувачів, а також оперативно реагувати на підозрілу активність, блокуючи доступ за допомогою токену оновлення.

Окрім цього, система гарантує конфіденційність та безпеку взаємодії зі штучним інтелектом, оскільки не зберігає жодних даних про запити користувача до AI. Це означає, що вся інформація, яка надсилається для обробки, залишається анонімною та не зберігається на серверах після завершення запиту. Такий підхід виключає ризики пов'язані з витоком чи зловживанням особистими даними користувачів, забезпечуючи їхню повну конфіденційність.

Простота використання

Інтерфейс користувача розроблений у мінімалістичному стилі та не перевантажений зайвими елементами. Це дозволяє користувачам швидко освоїти доступний функціонал, буквально за кілька хвилин, що сприяє інтуїтивній взаємодії з програмою.

Доступність

Система має низький поріг для початку використання, що робить її легкою у доступі для нових користувачів. Для початку роботи не потрібно виконувати додаткові дії, що значно спрощує процес взаємодії з програмою.

Функціональність

Незважаючи на простий і мінімалістичний дизайн, застосунок пропонує широкий набір функцій, орієнтованих на вдосконалення швидкості набору тексту з клавіатури, що дозволяє повноцінно задовольнити потреби кінцевого користувача.

Продуктивність

Обсяг даних, що передаються між клієнтським застосунком і REST API, є незначним, тому навіть при низькій швидкості інтернет-з'єднання не повинно виникати суттєвих затримок у роботі системи.

Висновки до розділу 1

У першому розділі кваліфікаційної магістерської роботи проведено детальний аналіз доступних варіантів інтеграції моделей штучного інтелекту на ринку. На основі цього було прийнято рішення щодо розширення архітектури застосунку та складено план його реалізації. Також виконано аналіз системи, що розробляється. Окреслено основний функціонал, визначено користувачів, робочі сценарії та технології, необхідні для реалізації проєкту. Крім того, було надано специфікацію вимог до системи з детальним описом її поведінки, функціональних та нефункціональних вимог, а також можливих обмежень.

2 МОДЕЛЮВАННЯ СИСТЕМИ

2.1 Створення варіантів використання системи

Короткий use case

Користувач прагне покращити свої навички швидкого набору тексту на клавіатурі. Для цього він заходить на вебсайт та проходить етапи аутентифікації та авторизації, що дозволяють зберігати дані про його тренування. Після входу в систему він переходить у спеціальний режим, де штучний інтелект генерує текст на основі введених користувачем критеріїв, таких як рівень складності чи тематика. Виконавши тренування, користувач переглядає свою статистику у відповідному розділі сайту, де відображаються його результати та прогрес, що допомагає йому відслідковувати власні досягнення та вдосконалювати свої навички.

Поверхневий use case

Головний сценарій (успішний):

Користувач прагне покращити свої навички швидкого набору тексту на клавіатурі. Для цього він заходить на вебсайт та проходить етапи аутентифікації та авторизації, що дозволяють зберігати дані про його тренування. Після входу в систему він переходить у спеціальний режим, де штучний інтелект генерує текст на основі введених користувачем критеріїв, таких як рівень складності чи тематика.

Альтернативні сценарії:

1. у випадку технічних проблем на вебсервері, де розміщений застосунок, вебсайт стає недоступним для користувачів. Доступ до системи буде тимчасово припинений до усунення проблем;

2. якщо користувач забув пароль від свого облікового запису, він може скористатися функцією відновлення пароля, підтвердивши свою особу через електронну пошту або інший метод ідентифікації;

3. якщо користувач не зареєстрований у системі, йому не буде доступний перегляд результатів тренувань. Хоча тренування можливе, прогрес і статистика не зберігатимуться;

4. якщо під час тренування пристрій користувача відключається від Інтернету на понад 30 секунд, тренування отримає статус «не завершене», і його результати не будуть враховані;

5. у випадку, якщо користувач закриває вкладку з вебсайтом під час тренування, це тренування також матиме статус «не завершене» у статистиці через передчасне завершення сесії.

Повний use case

Таблиця 2.1 – Повний use case

Scope	Вебзастосунок для тренування швидкості вводу даних на клавіатурі
Level	Мета користувача (user-goal)
Primary Actor	Користувач
Preconditions	Стабільне підключення до мережі Інтернет
Stakeholders and interests	1. користувач: прагне до максимально комфортного та ефективного тренування своїх навичок швидкого набору тексту на клавіатурі; 2. адміністратор: відповідає за безперебійну роботу застосунку та інформування користувачів про нові функції й оновлення системи.
Main Success Scenario	1. користувач відвідує вебсайт з метою розпочати тренування зі швидкості введення тексту на клавіатурі; 2. користувач проходить процес реєстрації для створення облікового запису; 3. користувач успішно виконує вхід у систему через аутентифікацію та авторизацію; 4. користувач відкриває режим налаштування тренування із використанням штучного інтелекту;

Продовження таблиці 2.1

Main Success Scenario	5. користувач надає штучному інтелекту необхідні параметри для проведення тренування; 6. після закінчення тренування користувач успішно завершує сесію; 7. користувач переходить до розділу статистики для оцінки своїх результатів та продуктивності.
Result	Користувач покращив свої навички введення даних на клавіатурі та проаналізував свою продуктивність за останній період.
Extensions	1. вебсервер, на якому розміщено застосунок, тимчасово недоступний, що унеможливлює доступ до системи та всіх її функці; 2. користувач не був авторизований на момент початку тренування: 2.1. користувач створює новий обліковий запис або виконує вхід за допомогою раніше створених облікових даних; 2.2. після успішної авторизації користувач переходить у багатокористувацький режим для тренування з іншими учасниками; 2.3. користувач налаштовує параметри тренування, вибираючи бажані опції, такі як складність, тривалість тощо; 2.4. після налаштування користувач приєднується до тренувальної сесії, де змагається з іншими користувачами; 2.5. по завершенню тренування користувач успішно завершує сесію; 2.6. користувач переходить до розділу статистики, де може проаналізувати свої результати та прогрес. 3. користувач забув пароль від свого облікового запису: 3.1. користувач натискає на кнопку «Forgot password?» на сторінці входу в систему; 3.2. на електронну пошту користувача надсилається лист із посиланням для відновлення паролю;

Кінець таблиці 2.1

Extensions	3.3. користувач відкриває лист та переходить за вказаним у ньому посиланням для початку процесу відновлення паролю; 3.4. користувач вводить новий пароль двічі для підтвердження та натискає кнопку «Change password», щоб завершити процес зміни; 3.5. після відновлення доступу користувач знову входить у систему та переходить у багатокористувацький режим; 3.6. користувач налаштовує параметри тренування за власними потребами; 3.7. після цього користувач бере участь у тренуванні разом з іншими користувачами; 3.8. після завершення тренування користувач успішно завершує сесію; 3.9. користувач переглядає результати свого тренування у розділі статистики для аналізу ефективності та подальшого покращення. 4. користувач закриває вкладку з вебсайтом під час тренування: 4.1. у цьому випадку поновлення тренувальної сесії неможливе, і тренування автоматично отримує статус «Not completed» у системі; 5. пристрій втратив зв’язок з мережею Інтернет більш ніж на 30 секунд: 5.1. тренування матиме статус «Not completed».
Special Requirements	1. будь який пристрій для вводу даних (бажано фізична клавіатура); 2. доступ до мережі Інтернет.
Frequency of Occurrence	Система здатна функціонувати практично без перерв.

Діаграми прецедентів дозволяють відобразити загальний функціонал системи та її область застосування. Вони також наочно ілюструють взаємозв'язки між акторами і системою, визначаючи загальний контекст та вимоги до неї.

На рис. 1 у додатку А представлена діаграма використання для вебзастосунку, що слугує для тренування швидкості введення тексту на клавіатурі.

2.2 Побудова діаграм взаємодії

Діаграми взаємодії є однією з ключових складових у проєктуванні програмного забезпечення, особливо в контексті моделювання поведінки системи. Вони дозволяють наочно відобразити процес комунікації між компонентами системи протягом виконання певного сценарію. Основна мета таких діаграм — продемонструвати, як окремі елементи обмінюються повідомленнями для виконання функціональних вимог.

Діаграми взаємодії допомагають розробникам глибше зрозуміти динаміку системи, деталізувати бізнес-логіку та виявити потенційні проблеми в процесі проєктування[9]. Вони показують послідовність викликів методів, обробку даних і взаємодію між компонентами, що є важливими для забезпечення коректної реалізації вимог. Також вони допомагають проаналізувати процеси, що відбуваються між різними частинами системи, і є важливим інструментом для виявлення та усунення можливих проблем на ранніх етапах розробки.

В порівнянні з попередньою реалізацією процесу аутентифікації та авторизації користувачів складність обробки даних значно зменшено. Такого ступеня покращення досягнуто шляхом застосування фреймворку аутентифікації Auth0. Більше того, завдяки делегації відповідальності за видачу токенів авторизації на сторонній сервіс оптимізовано навантаження на розроблювану систему. На рис. 2.1 продемонстровано поточну діаграму послідовності.

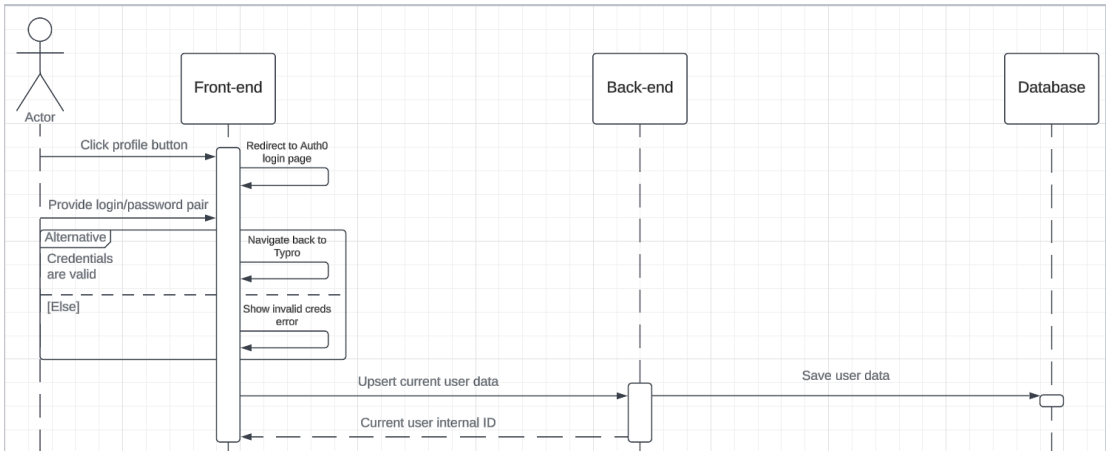

Рисунок 2.1 – Оновлена діаграма взаємодії аутентифікації користувача

Найбільшого впливу зазнав безпосередньо процес проходження тренування. Спершу розглянемо зміни внутрішніх процесів системи, що не додають нового функціоналу для користувача. Як можна побачити на рис. 2.2, етап генерації тексту тренування повністю делеговано штучному інтелекту. Сукупність системних елементів позначених маркером «Back-end» складається з двох REST API, на відміну від попередньої версії системи. Другий вебсервер відповідальний за спілкування зі моделлю штучного інтелекту шляхом застосування OpenAI API. Таким чином, запити до бази даних для отримання слів усунено.

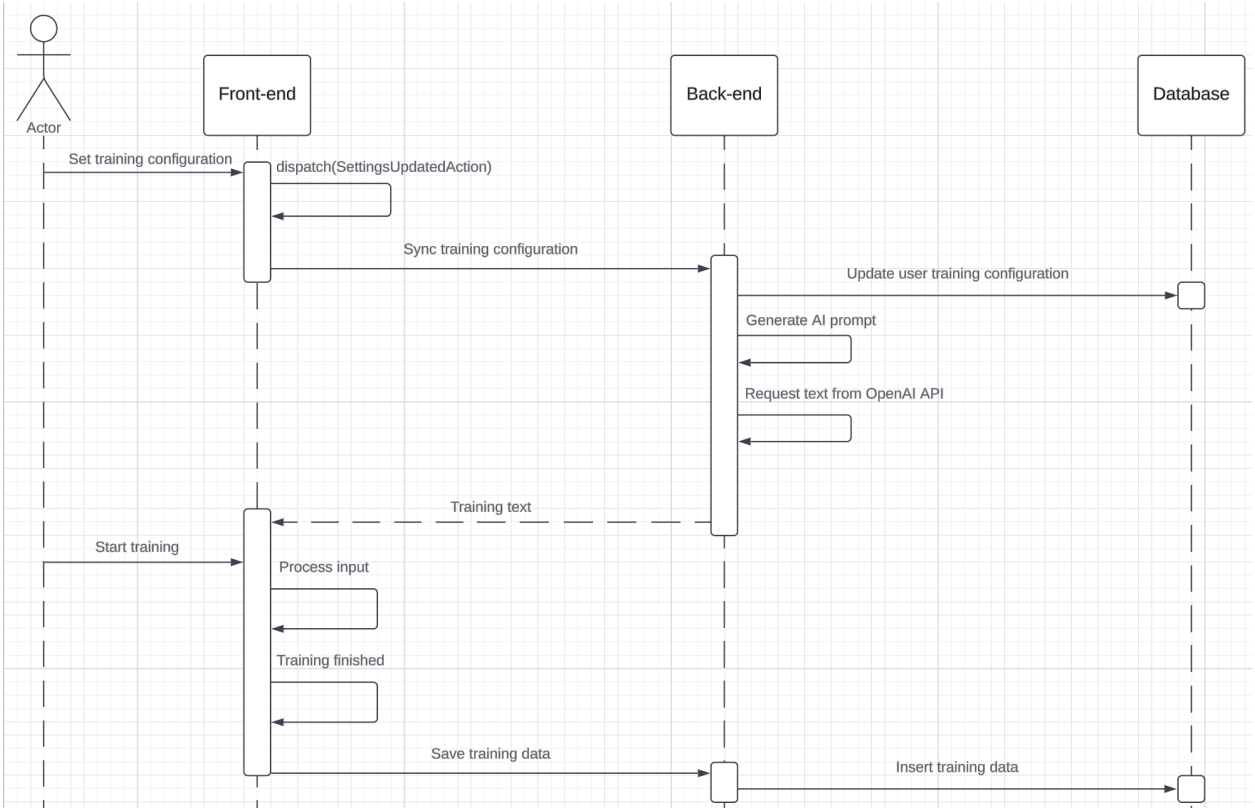

Рисунок 2.2 – Оновлена діаграма взаємодії проходження тренування в режимі для одного користувача з обраною кількістю слів

Іншим вагомим нововведенням для користувача є можливість самостійно корегувати текст для тренування шляхом відправки запитів до моделі штучного інтелекту напряму. Це відкриває потужний спектр можливостей персоналізації досвіду тренування шляхом визначення тем, обсягів, формату тексту тощо. На рис. 2.3 зображено діаграму взаємодії для цього режиму.

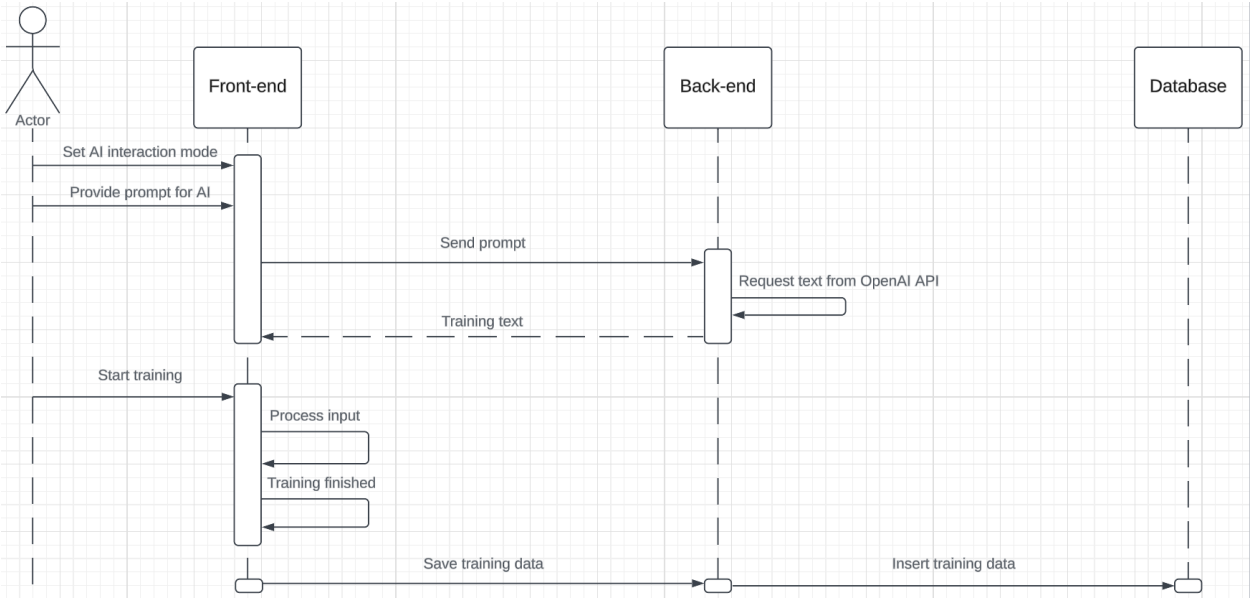

Рисунок 2.3 – Діаграма взаємодії проходження тренування в режимі штучного інтелекту

Звертаючи увагу на надані діаграми можна зробити висновок, що загальне навантаження на систему значно зменшене. Делегація відповідальності окремим сервісам полегшує розробку та функціонування системи загалом. Це позитивний крок в сторону спрощення інфраструктури, полегшення процесу розробки та підтримки кодової бази.

2.3 Побудова діаграм діяльності

Діаграми діяльності є одним із ключових типів діаграм у моделюванні програмного забезпечення за допомогою UML[8] (Unified Modeling Language). Вони використовуються для візуалізації потоків управління у межах певного процесу. Основна мета діаграм діяльності - продемонструвати послідовність дій, що виконуються під час досягнення виконання бізнес-процесу.

Ці діаграми дозволяють чітко описати логіку виконання, включаючи послідовність операцій, паралельні дії, розгалуження та обробку виключень. Вони використовуються для відображення як внутрішніх, так і зовнішніх бізнес-процесів, а також для моделювання потоку даних між компонентами системи.

Діаграми діяльності допомагають зрозуміти, як саме функціонує система під час виконання завдань, а також дають змогу виявити можливі проблеми в потоці виконання. Вони є корисними інструментами для планування алгоритмів і розуміння поведінки системи у відповідь на певні події або дії користувача.

Завдяки своїй простоті та наочності, діаграми діяльності часто використовуються для комунікації між різними учасниками проєкту — від бізнес-аналітиків до розробників, сприяючи кращому розумінню процесів і допомагаючи у розробці більш структурованого програмного забезпечення.

На рис. 2.4 зображено діаграму діяльності, яка відображає процес генерації тексту для тренувань із використанням заданих параметрів. Ця діаграма демонструє поетапний хід виконання операцій, починаючи з налаштування конфігурації тренування і закінчуючи поверненням згенерованого результату.

На початковому етапі відбувається налаштування конфігурації тренування, що визначає параметри майбутнього тексту. Після цього здійснюється запит до API для створення початкового тексту на основі цих параметрів. Додатково формується спеціальний запит (prompt), який передається до AI API для генерації відповідного тексту.

Діаграма містить кілька ключових етапів перевірки і модифікації згенерованого тексту. Зокрема:

– перевіряється режим роботи ("words mode") та правильність кількості слів. Якщо кількість слів не відповідає заданим параметрам, їх кількість коригується;

– у разі потреби додається пунктуація до тексту, що робить завдання більш наближеним до реальних сценаріїв набору тексту;

– за запитом користувача, випадкові слова у тексті можуть бути замінені на числа, що створює додаткову складність для тренування.

На фінальному етапі текст перетворюється у формат, готовий для передачі фронтенду — масив символів для покращення обробки під час відображення на екрані. Завершення процесу позначається поверненням результату до користувача.

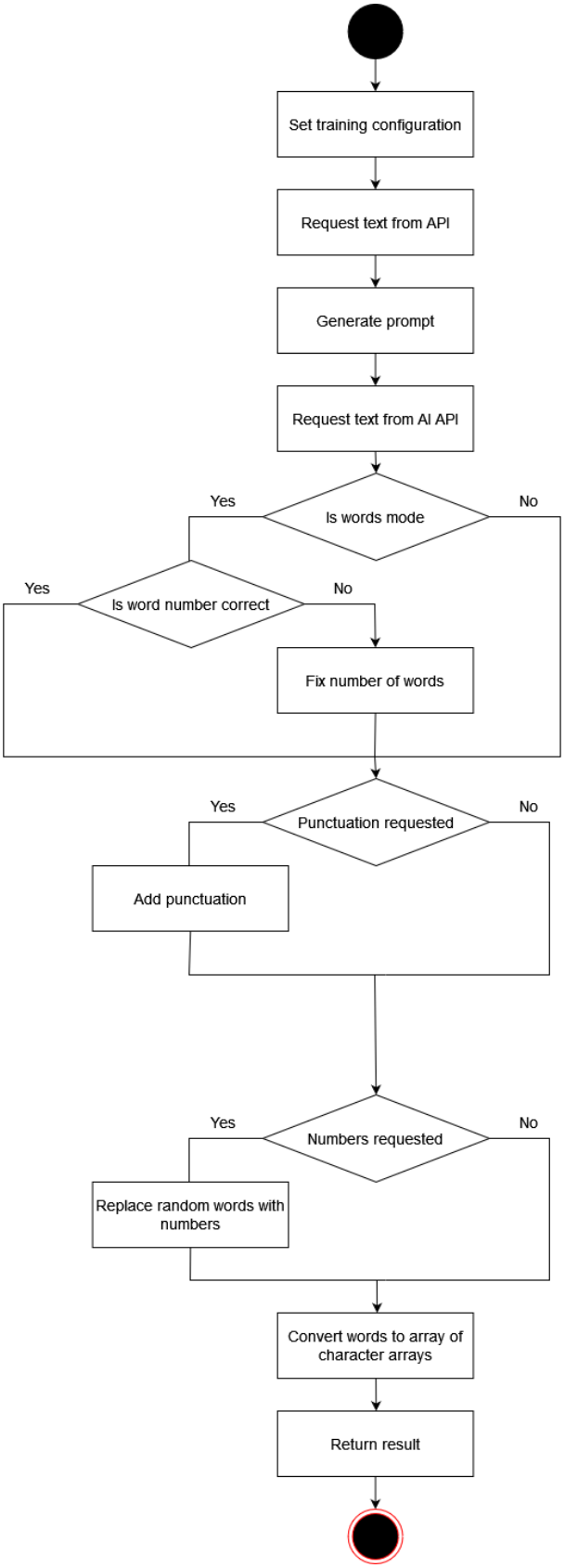

Рисунок 2.4 – Діаграма діяльності процесу генерації тексту для тренування зі заданням параметрів

Додаткова можливість взаємодії зі штучним інтелектом напряму має схожу послідовність дій, виключаючи етапи обробки чітких критеріїв режиму гри (див. рис. 2.5).

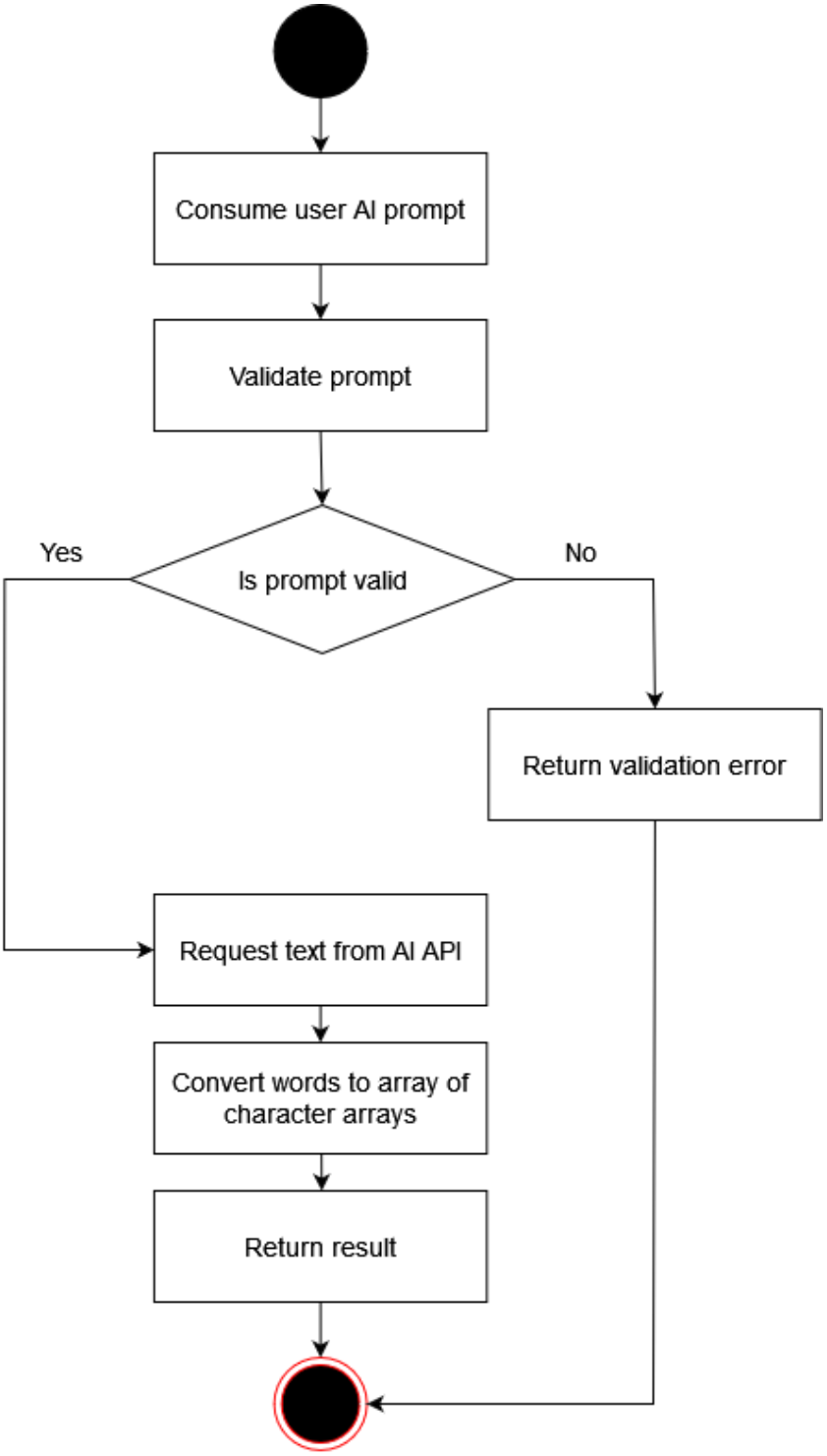

Рисунок 2.5 – Діаграма діяльності процесу генерації тексту з передачею запиту від користувача напряму

Не менш важливою зміною функціонування застосунку є нова система аутентифікації та авторизації Auth0. На рис. 2.6 зображено оновлений принцип ідентифікації користувачів.

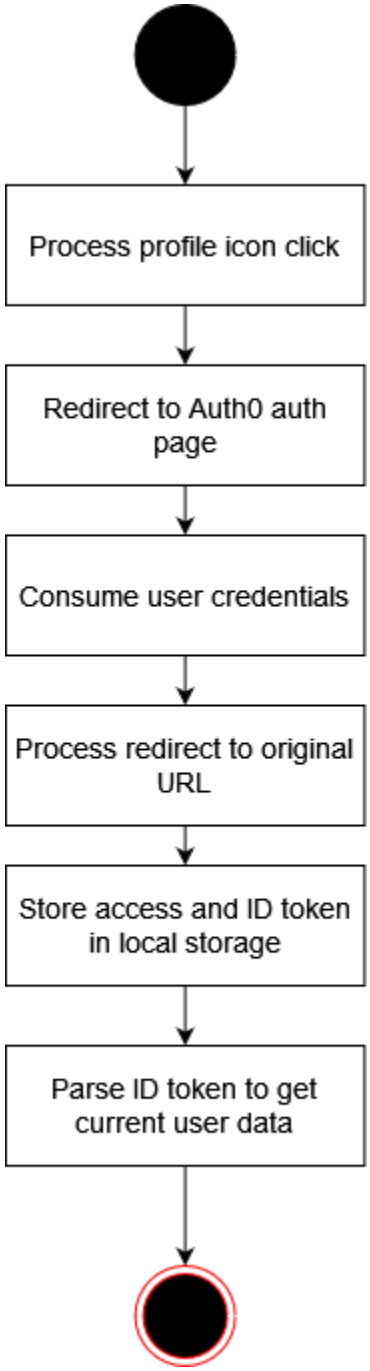

Рисунок 2.6 – Діаграма діяльності процесу аутентифікації та авторизації користувача з використанням Auth0

Отже, на внутрішньому рівні розроблювана система зазнала значних змін. Усі нововведення вносять позитивну динаміку до процесу розробки та впровадження застосунку. Описані процеси містять достатній рівень деталізації та наочність для самостійного використання нетехнічними спеціалістами для отримання глибшого сприйняття та розуміння нюансів роботи продукту.

2.4 Побудова діаграми розгортання

Діаграми розгортання (Deployment diagrams) є важливим інструментом у проєктуванні програмного забезпечення, який використовується для моделювання фізичної архітектури системи. Вони відображають, як програмні компоненти будуть розгорнуті на фізичних або віртуальних пристроях, таких як сервери, комп'ютери тощо. Основна мета діаграм розгортання — показати, як компоненти програмного забезпечення (наприклад, модулі, сервіси, бази даних) співіснують і взаємодіють на рівні інфраструктури[11].

Даний тип діаграм наочно демонструє розподіл програмних компонентів на різні фізичні вузли (сервери, робочі станції тощо), що допомагає розуміти, як система функціонує в реальному середовищі. Вони також відображають мережеві підключення між вузлами, що дозволяє моделювати структуру мережі, типи з'єднань, а також протоколи взаємодії між компонентами.

Вони є особливо важливими на етапах планування розгортання програмного забезпечення у продакшн-середовищі або під час створення складних розподілених систем. Завдяки діаграмам розгортання можна чітко побачити, як компоненти системи будуть інтегровані в інфраструктуру, де кожен з них буде розміщений, і як вони будуть взаємодіяти один з одним.

На рис. 2.7 зображено діаграму розгортання з врахуванням інфраструктурних нововведень.

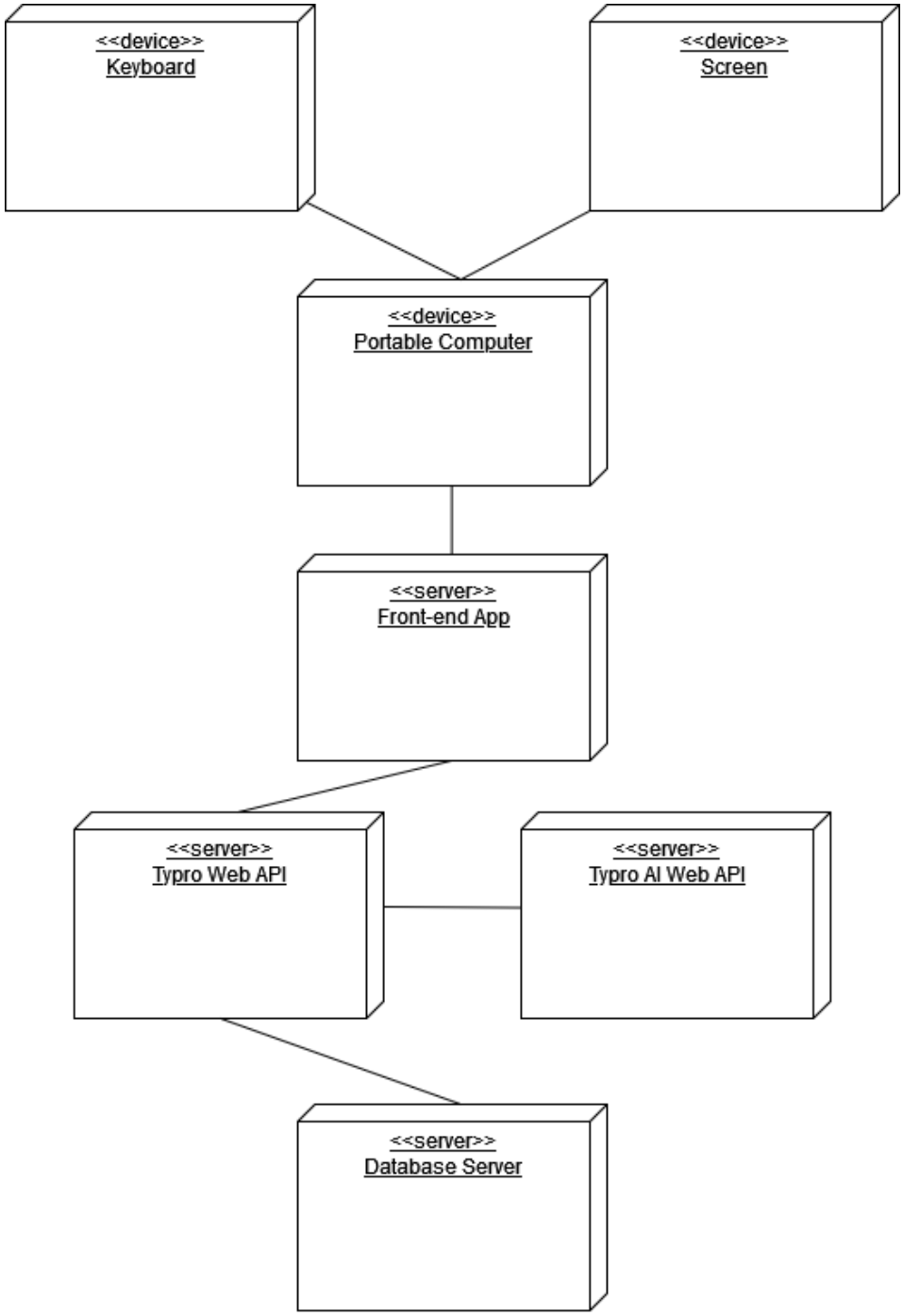

Рисунок 2.7 – Оновлена діаграма розгортання вебзастосунку тренування швидкості набору даних на клавіатурі з використанням штучного інтелекту

На діаграмі розгортання представлено архітектуру програмного забезпечення для тренування швидкості введення тексту на клавіатурі з використанням штучного інтелекту. Вузли на діаграмі відображені у вигляді прямокутних паралелепіпедів і поділяються на обчислювальні та програмні.

До обчислювальних вузлів належать:

– клавіатура (Keyboard) – пристрій вводу, який забезпечує взаємодію користувача із системою через введення тексту;

– екран (Screen) – пристрій виводу, який відображає інтерфейс застосунку та результати тренування;

– комп'ютер (Portable Computer) – основний пристрій для роботи клієнтського застосунку, що забезпечує фізичні ресурси для запуску програмного забезпечення;

До програмних вузлів належать:

– клієнтський застосунок (Front-end App) – програмний компонент, який відповідає за інтерфейс користувача та взаємодію з бекендом через API;

– API-сервер (Typro Web API) – серверний компонент, що обробляє запити від клієнтського застосунку, виконує бізнес-логіку та взаємодіє з базою даних;

– AI API-сервер (Typro AI Web API) – програмний сервіс для взаємодії зі штучним інтелектом, який генерує текстові завдання на основі заданих параметрів.

– сервер бази даних (Database Server) – компонент, який зберігає дані користувачів, результати тренувань і конфігурації системи;

Обчислювальні вузли виконують роль фізичної інфраструктури, на якій розгорнуті програмні вузли. Наприклад, Portable Computer забезпечує апаратну основу для запуску клієнтського застосунку (Front-end App), а сервери API працюють на виділених обчислювальних ресурсах.

Програмні вузли реалізують ключові функції системи, забезпечуючи взаємодію між різними компонентами. Typro Web API та Typro AI Web API виконують обробку бізнес-логіки та запитів, забезпечуючи динамічний обмін даними між клієнтською частиною та базою даних, а також інтеграцію зі штучним інтелектом.

Загальна структура діаграми демонструє, як обчислювальні вузли забезпечують підтримку програмних компонентів, які виконують основні логічні та сервісні процеси. Це дозволяє створити масштабовану систему, яка забезпечує

ефективне використання ресурсів та підтримує основний функціонал вебзастосунку.

Висновки до розділу 2

Другий розділ кваліфікаційної магістерської роботи присвячений моделюванню програмного забезпечення тренування швидкості вводу даних на клавіатурі з використанням штучного інтелекту. У цьому розділі, за допомогою попередньо розробленої діаграми використання, детально описані основні сценарії роботи системи, які спрямовані на задоволення ключових потреб користувача. Ці сценарії допомагають зрозуміти, як користувач взаємодіє із системою в різних режимах.

Крім того, побудовані діаграми діяльності, які детально описують алгоритми функціонування застосунку на різних етапах його роботи. Вони наочно демонструють послідовність виконання бізнес-процесів, що відбуваються у системі.

Також, представлено діаграми взаємодії, які використовуються для опису процесу обміну даними між клієнтською частиною та сервером під час виконання базових задач. Ці діаграми дозволяють краще зрозуміти, як відбувається комунікація між різними компонентами системи, підкреслюючи їхню роль у забезпеченні ефективної роботи вебзастосунку.

Здебільшого, у розділі представлено діаграму розгортання вебзастосунку, яка ілюструє фізичну архітектуру системи.

3 ПРОЄКТУВАННЯ СИСТЕМИ

3.1 Побудова діаграм класів

Діаграма класів є одним із ключових інструментів у процесі моделювання програмного забезпечення та слугує основою для подальшої розробки системи засобами об'єктно-орієнтованого програмування. Цей тип діаграм дозволяє візуалізувати структуру майбутньої програми, деталізуючи її компоненти, атрибути та методи, зв’язки між ними. Діаграма класів виконує роль містка між абстрактним проєктуванням та реалізацією у вигляді коду.

Кожен представлений клас, має чітко визначену структуру. Його поля відображають стан, зберігаючи внутрішню інформацію, тоді як методи описують поведінку класу та взаємодію з іншими компонентами системи. На даному етапі моделювання важливо звернути особливу увагу на найменування програмних компонентів. Імена мають бути інформативними, чітко відображаючи сутність і призначення, що сприяє кращому розумінню моделі не лише розробниками, але й іншими учасниками проєкту[12].

У цьому підрозділі представлені діаграми класів, які охоплюють ключові частини застосунку, деталізуючи окремі компоненти системи. На рисунках 3.1–3.5 наведено графічне відображення класів, які забезпечують реалізацію основної функціональності. Ці діаграми дозволяють простежити, як кожен клас взаємодіє з іншими елементами системи, формуючи загальну архітектуру.

Крім того, у таблицях 3.1–3.5 представлено докладний опис кожного з компонентів.

Такий підхід дозволяє отримати цілісне уявлення про внутрішню логіку застосунку, відображає принципи об’єктно-орієнтованого програмування, використані під час розробки. Це забезпечує не лише зручність для розробників у підтримці системи, але й створює базу для подальшого вдосконалення застосунку.

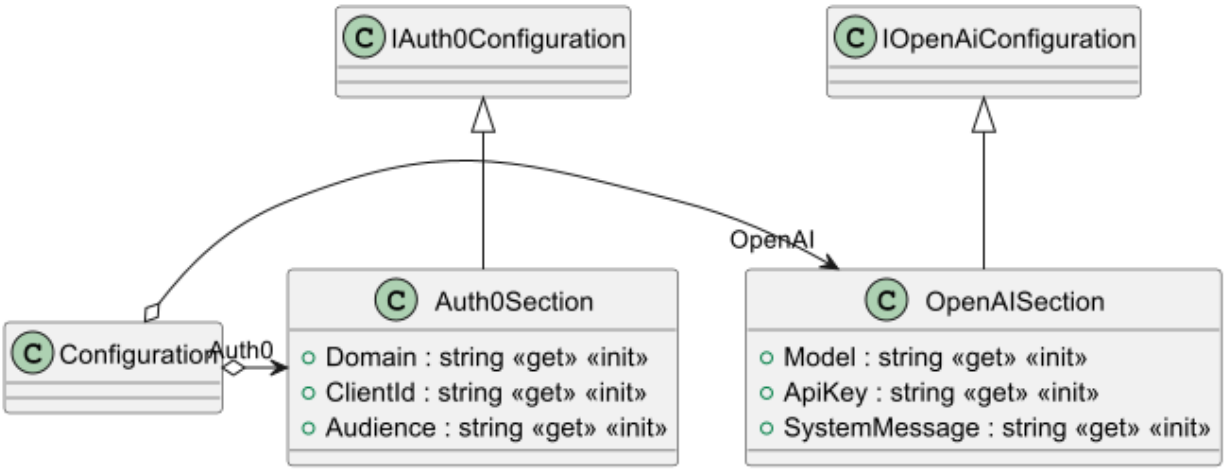

Рисунок 3.1 – Діаграма класів конфігурації сервісу комунікації зі штучним інтелектом

У таблиці 3.1 наведено опис класів представлених на рисунку 3.1.

Таблиця 3.1 – Характеристика класів задіяних у конфігурації сервісу комунікації зі штучним інтелектом

Назва класу/інтерфейсу	**Поля**	**Методи**	**Короткий опис**
Configuration	Auth0 OpenAI	-	Загальна конфігурація сервісу
Auth0Section	Domain ClientId Audience	-	Конфігурація аутентифікації за допомогою Auth0
OpenAISection	Model ApiKey SystemMessage	-	Налаштування пов’язані з OpenAI

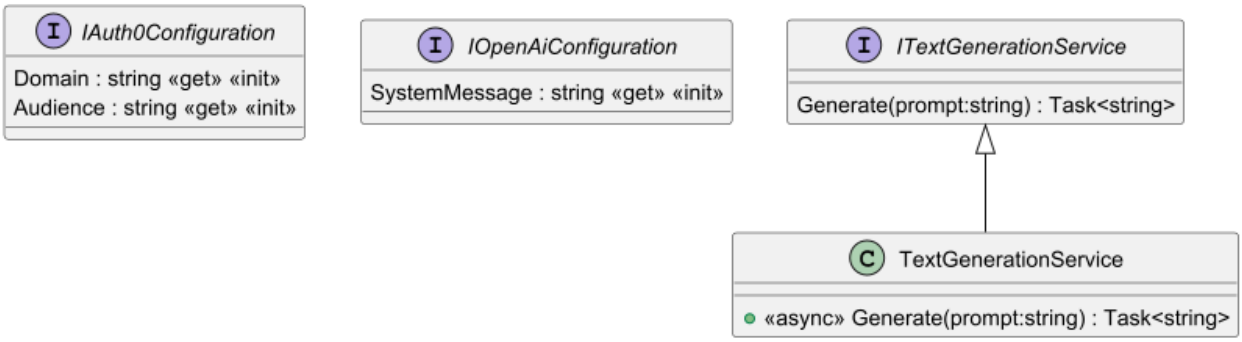

Рисунок 3.2 – Діаграма класів, що містять бізнес-логіку комунікації зі штучним інтелектом

У таблиці 3.2 наведено опис класів представлених на рисунку 3.2.

Таблиця 3.2 – Характеристика класів, що містять бізнес-логіку комунікації зі штучним інтелектом

Назва класу	**Поля**	**Методи**	**Короткий опис**
TextGenerationService	-	Generate	Відповідає за формування запиту до моделі штучного інтелекту та отримання відповіді

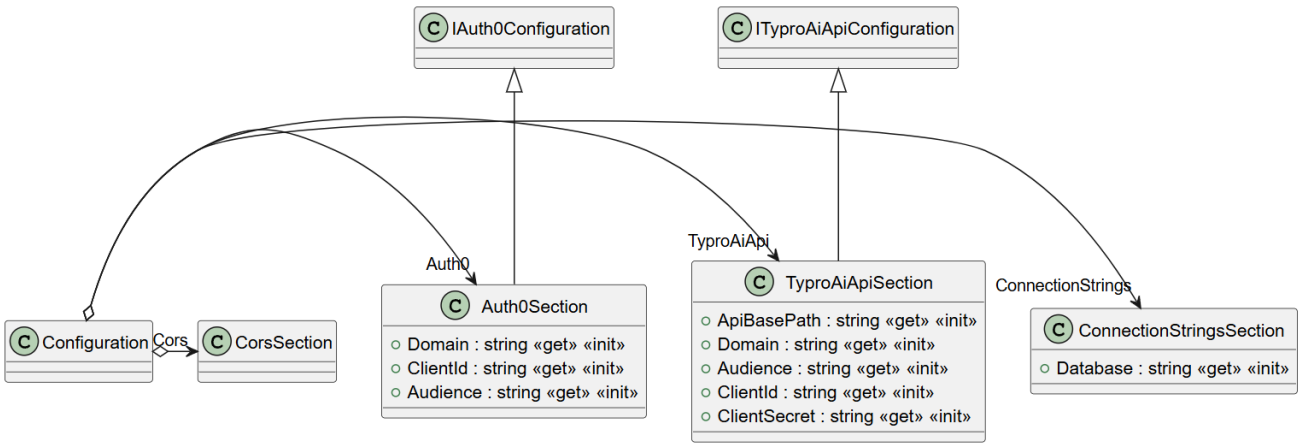

Рисунок 3.3 – Діаграма класів конфігурації основного сервісу

У таблиці 3.3 наведено опис класів представлених на рисунку 3.3.

Таблиця 3.3 – Характеристика класів задіяних у конфігурації основного сервісу

Назва класу/інтерфейсу	**Поля**	**Методи**	**Короткий опис**
Configuration	Cors Auth0 TyproAiApi ConnectionStrings	-	Загальна конфігурація сервісу
CorsSection	AllowedOrigins	-	Конфігурація CORS
Auth0Section	Domain ClientId Audience	-	Конфігурація аутентифікації за допомогою Auth0
TyproAiApiSection	ApiBasePath Domain Audience ClientId ClientSecret	-	Конфігурація для комунікації з сервісом комунікації зі штучним інтелектом
ConnectionStringsSection	Database	-	Конфігурація для рядків з’єднання

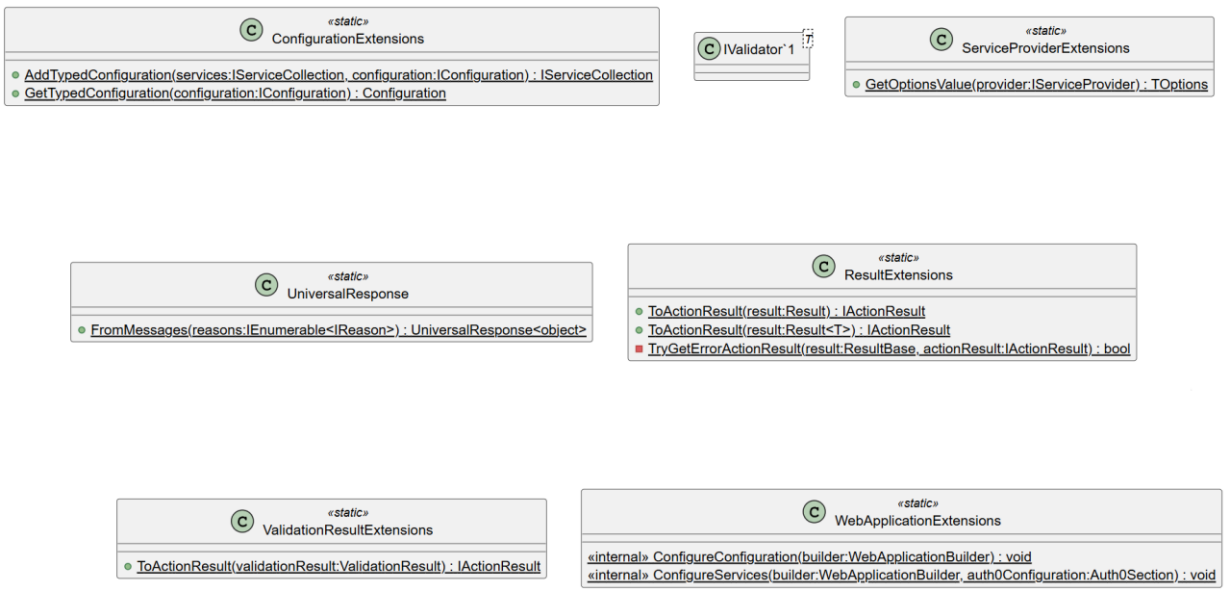

Рисунок 3.4 – Діаграма класів налаштування роботи Web API

У таблиці 3.4 наведено опис класів представлених на рисунку 3.4.

Таблиця 3.4 – Характеристика класів задіяних у налаштування роботи Web API

Назва класу/інтерфейсу	**Поля**	**Методи**	**Короткий опис**
ConfigurationExtensions	-	AddTypedConfiguration GetTypedConfiguration	Методи додавання конфігурації до DI-контейнера та отримання її напряму
ServiceProviderExtensions	-	GetOptionsValue	Містить метод для отримання значення напряму від IOptions інтерфейсу
UniversalResponse	-	FromMessages	Формує відповідь сервера на основі отриманих повідомлень
ResultExtensions	-	ToActionResult (2 перевантаження) TryGetErrorActionResult	Методи для роботи з об’єктами результатів від сервісів бізнес логіки для формування відповіді сервера
ValidationResultExtensions	-	ToActionResult	Формування відповіді серверу на основі результату перевірки правильності запиту
WebApplicationExtensions	-	ConfigureConfiguration ConfigureServices	Загальні методи, що містять всі виклики для налаштування конфігурації та сервісів у DI-контейнері

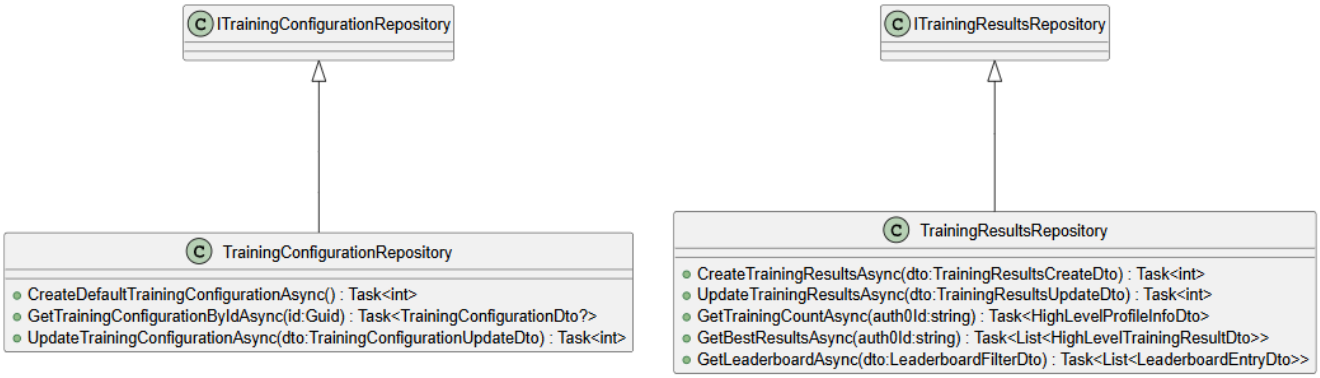

Рисунок 3.5 – Діаграма класів-репозиторіїв для конфігурації тренувань та результатів

У таблиці 3.5 наведено опис класів представлених на рисунку 3.5.

Таблиця 3.5 – Характеристика класів-репозиторіїв

Назва класу/інтерфейсу	**Поля**	**Методи**	**Короткий опис**
TrainingResultsRepository	dbContext	CreateDefaultTrainingConfigurationAsync GetTrainingConfigurationByIdAsync UpdateTrainingConfigurationAsync	Робота з конфігураціями тренувань у базі даних
TrainingResultsRepository	dbContext	CreateTrainingResultsAsync UpdateTrainingResultsAsync GetTrainingCountAsync GetBestResultsAsync GetLeaderboardAsync	Робота з результатами тренувань у базі даних

3.2 Побудова діаграм станів та переходів

Діаграми станів і переходів є важливим інструментом для моделювання поведінки системи та наочної демонстрації можливих сценаріїв її роботи. Вони дозволяють деталізувати, як об'єкти системи змінюють свій стан у відповідь на внутрішні чи зовнішні події, забезпечуючи чітке розуміння логіки роботи програмного забезпечення. Такий підхід особливо корисний для складних систем, де кожен компонент взаємодіє з іншими і має свій набір можливих станів.

Для програмного забезпечення, призначеного для тренування швидкості введення тексту на клавіатурі з використанням штучного інтелекту, було створено три діаграми станів і переходів.

Діаграма процесу аутентифікації на рисунку 3.6 описує логіку зміни станів під час виконання процесу входу в систему. Вона охоплює перевірку облікових даних користувача, обробку помилок аутентифікації, таких як невірний пароль, та сценарій успішного входу. Ця діаграма допомагає побачити, як система обробляє різні сценарії аутентифікації, забезпечуючи плавність переходів і захист від потенційних помилок.

Діаграма перегляду списку лідерів на рисунку 3.7 демонструє, як система переходить між станами під час взаємодії користувача з розділом рейтингу. Вона описує, як відбувається запит до сервера, отримання даних про лідерів, обробка цих даних і реакція системи на можливі помилки, наприклад, у разі недоступності сервера. Така деталізація дозволяє оптимізувати процес, зробити його більш надійним для кінцевого користувача.

Діаграма отримання статистики тренувань на рисунку 3.8 моделює сценарій, у якому користувач запитує статистику своїх тренувань. У цій діаграмі описані ключові етапи: підготовка системою відповідного запиту, надсилання запиту до сервера, обробка отриманих даних і виведення результатів на екран. Крім того, враховані можливі ситуації, коли запит не може бути виконаний через проблеми із сервером або відсутність відповідних даних.

Кожна з цих діаграм надає структурований опис станів і переходів, що відображають ключові процеси системи. Завдяки їм можна краще зрозуміти логіку роботи програмного забезпечення, забезпечити його відповідність функціональним вимогам і виявити потенційні точки для оптимізації. Також вони допомагають виявляти можливі проблеми, наприклад, неправильне оброблення помилок чи неефективні переходи, що сприяє вдосконаленню якості системи. Такі діаграми є важливим інструментом для подальшого розвитку та підтримки проєкту.

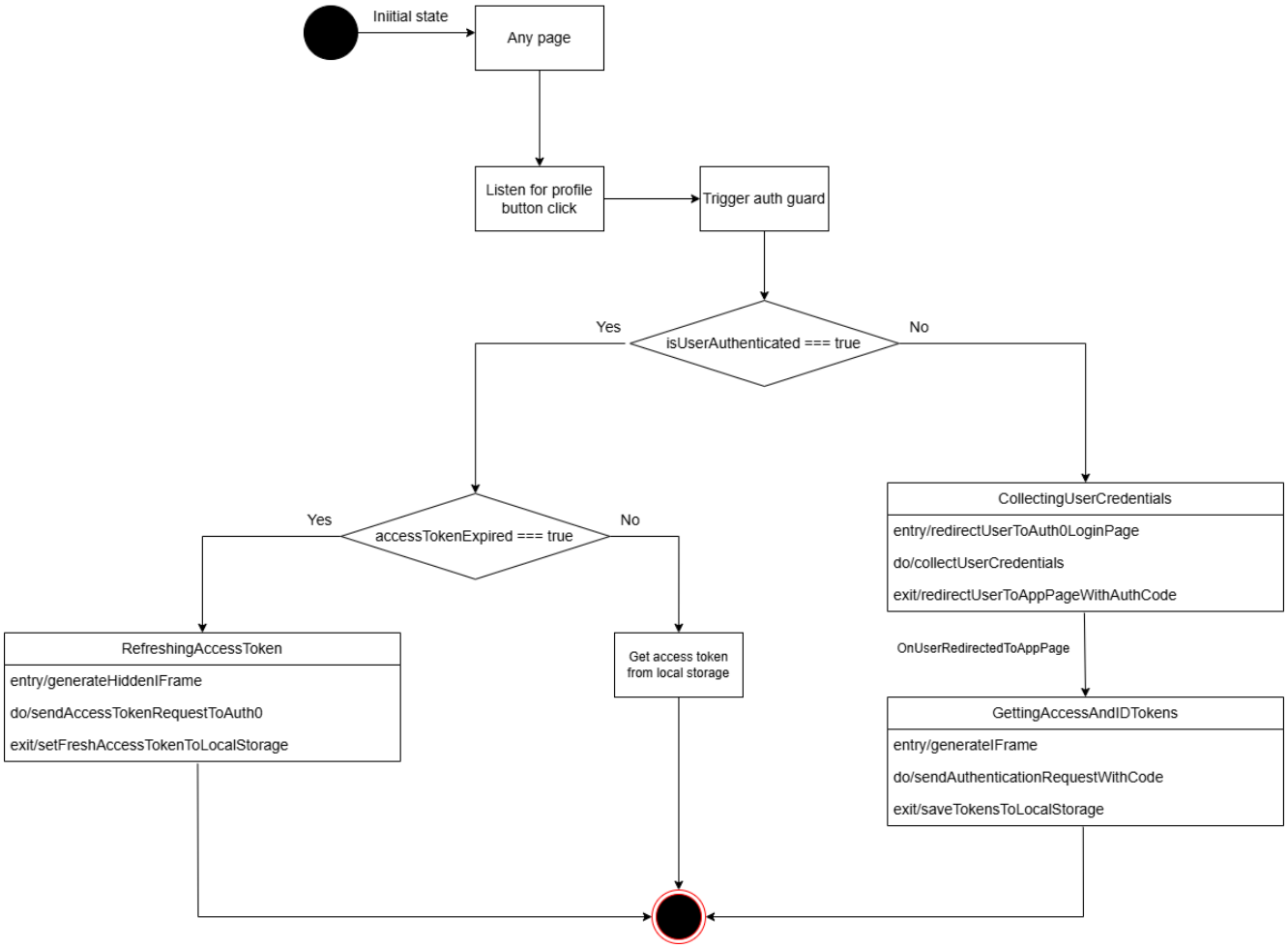

Рисунок 3.6 – Діаграма станів процесу аутентифікації

Діаграма станів, представлена на рисунку 3.6, ілюструє процес автентифікації користувача в системі з використанням Auth0. Вона охоплює ключові етапи взаємодії користувача із системою, включаючи перевірку стану автентифікації, обробку закінчення терміну дії токенів доступу, а також процес збору облікових даних для входу.

Процес починається з будь-якої сторінки застосунку, де система чекає на натискання кнопки профілю. Після цього активується перевірка через Auth Guard, яка визначає, чи автентифікований користувач. Якщо користувач не автентифікований, система ініціює збір облікових даних, перенаправляючи користувача на сторінку входу Auth0. У разі успішного введення даних система отримує токени доступу та ідентифікації.

Якщо користувач автентифікований, але термін дії токену доступу закінчився, система виконує процес його оновлення через прихований iframe, надсилаючи запит до Auth0 та зберігаючи новий токен. Якщо токен залишається дійсним, система отримує його з локального сховища, забезпечуючи безперервний доступ користувача.

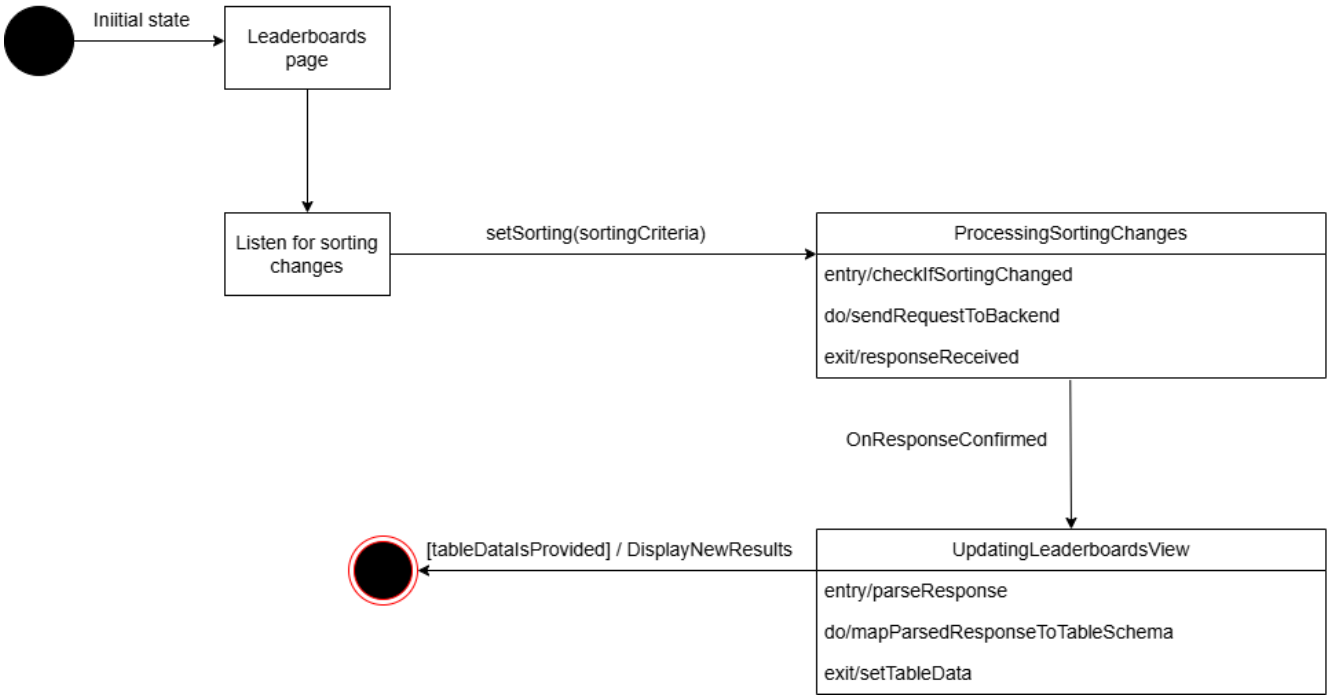

Рисунок 3.7 – Діаграма станів при перегляді списку лідерів

Діаграма на рисунку 3.7 демонструє процес обробки змін сортування на сторінці лідерів у вебзастосунку. Процес починається зі сторінки лідерів, де користувач може взаємодіяти із таблицею, змінюючи критерії сортування. Після цього відбувається низка станів, які відповідають за обробку введених змін і оновлення інтерфейсу.

Першим кроком система очікує на зміну сортування від користувача. Після натискання користувачем кнопки сортування або вибору іншого критерію активується подія setSorting(sortingCriteria), яка ініціює стан ProcessingSortingChanges. У цьому стані система перевіряє, чи дійсно критерій сортування змінився, формує запит до бекенду для отримання оновлених даних і очікує на відповідь.

Коли відповідь від бекенду отримано (подія responseReceived), система переходить у стан UpdatingLeaderboardsView, де відбувається обробка даних. На цьому етапі система аналізує відповідь (парсить дані), зіставляє їх із потрібною схемою таблиці та оновлює дані для відображення.

Заключним етапом є оновлення відображення на екрані користувача. Якщо нові дані таблиці успішно передані (умова [tableDataIsProvided]), система повертається до базового стану з відображенням оновленої таблиці лідерів.

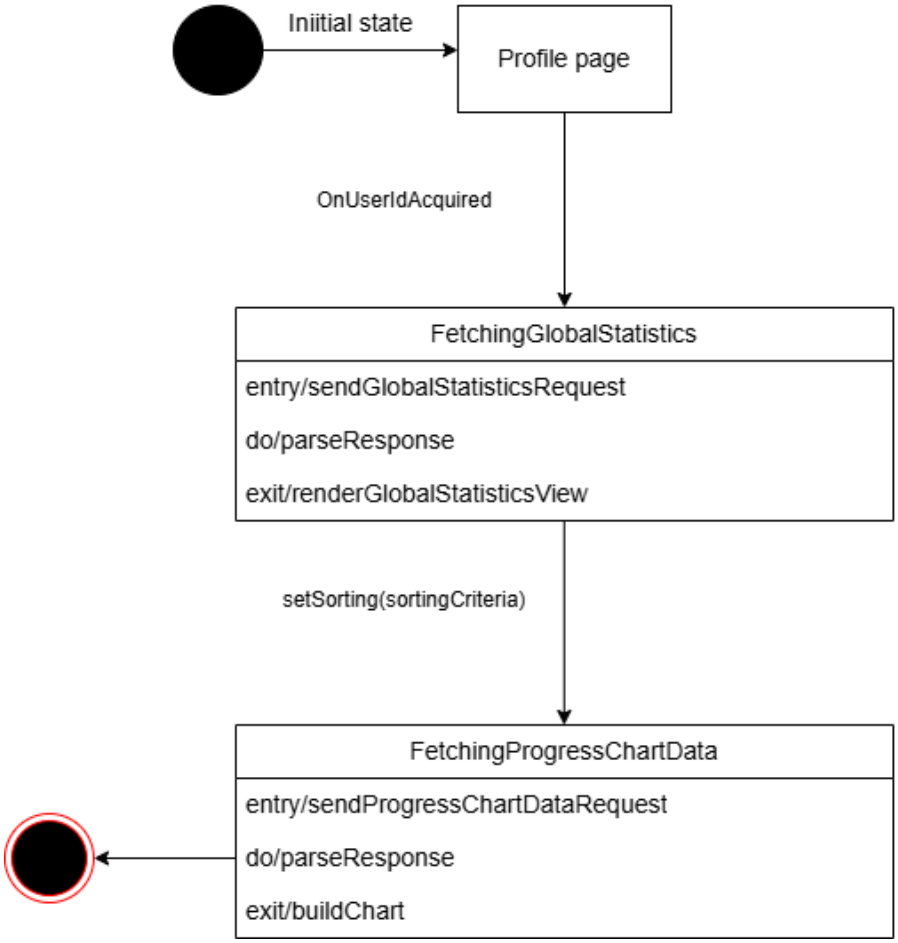

Рисунок 3.8 – Діаграма станів процесу перегляду статистики профілю

Діаграма на рисунку 3.8 демонструє процес отримання та обробки статистики на сторінці профілю користувача. Початковий стан передбачає завантаження сторінки профілю, після чого відбувається збір необхідних даних для відображення статистики.

Першим етапом є стан FetchingGlobalStatistics, у якому система виконує запит до бекенду для отримання глобальної статистики. Під час цього стану:

– вхідна дія sendGlobalStatisticsRequest ініціює запит;

– відповідь від сервера обробляється (дія parseResponse);

– після обробки даних виконується відображення глобальної статистики у вигляді відповідного інтерфейсу (вихідна дія renderGlobalStatisticsView).

Далі, якщо користувач змінює критерії сортування (подія setSorting(sortingCriteria)), система переходить до стану FetchingProgressChartData. У цьому стані відбувається:

– вхідна дія sendProgressChartDataRequest, що запускає запит до бекенду для отримання детальних даних прогресу;

– відповідь обробляється через дію parseResponse;

– на виході будується графік прогресу (buildChart), який відображається користувачу.

3.3 Побудова діаграм компонентів

Діаграми компонентів є важливим інструментом у проєктуванні програмного забезпечення, що дозволяє розділити всю об'єктно-орієнтовану систему на менші частини. Це сприяє підвищенню керованості окремих елементів, а також полегшує аналіз і розуміння залежностей між різними частинами системи. Компоненти можуть бути як логічними, так і фізичними елементами, що беруть участь у виконанні певних процесів. Наприклад, вони можуть включати класи, модулі, бібліотеки або інтерфейси, а також сервери або бази даних.

Основною метою побудови діаграм компонентів є ідентифікація основних функціональних блоків системи та їх взаємозв'язків. Вони допомагають виявити залежності між компонентами, забезпечити модульність системи та чітко визначити, як різні частини співпрацюють одна з одною для досягнення загальних цілей[13].

Першим кроком у побудові діаграми компонентів є виділення основних функціональних частин системи на основі попередньо створеної діаграми класів. Для вебзастосунку тренування швидкості введення тексту на клавіатурі було виділено чотири ключові шари (див. рис. 3.9).

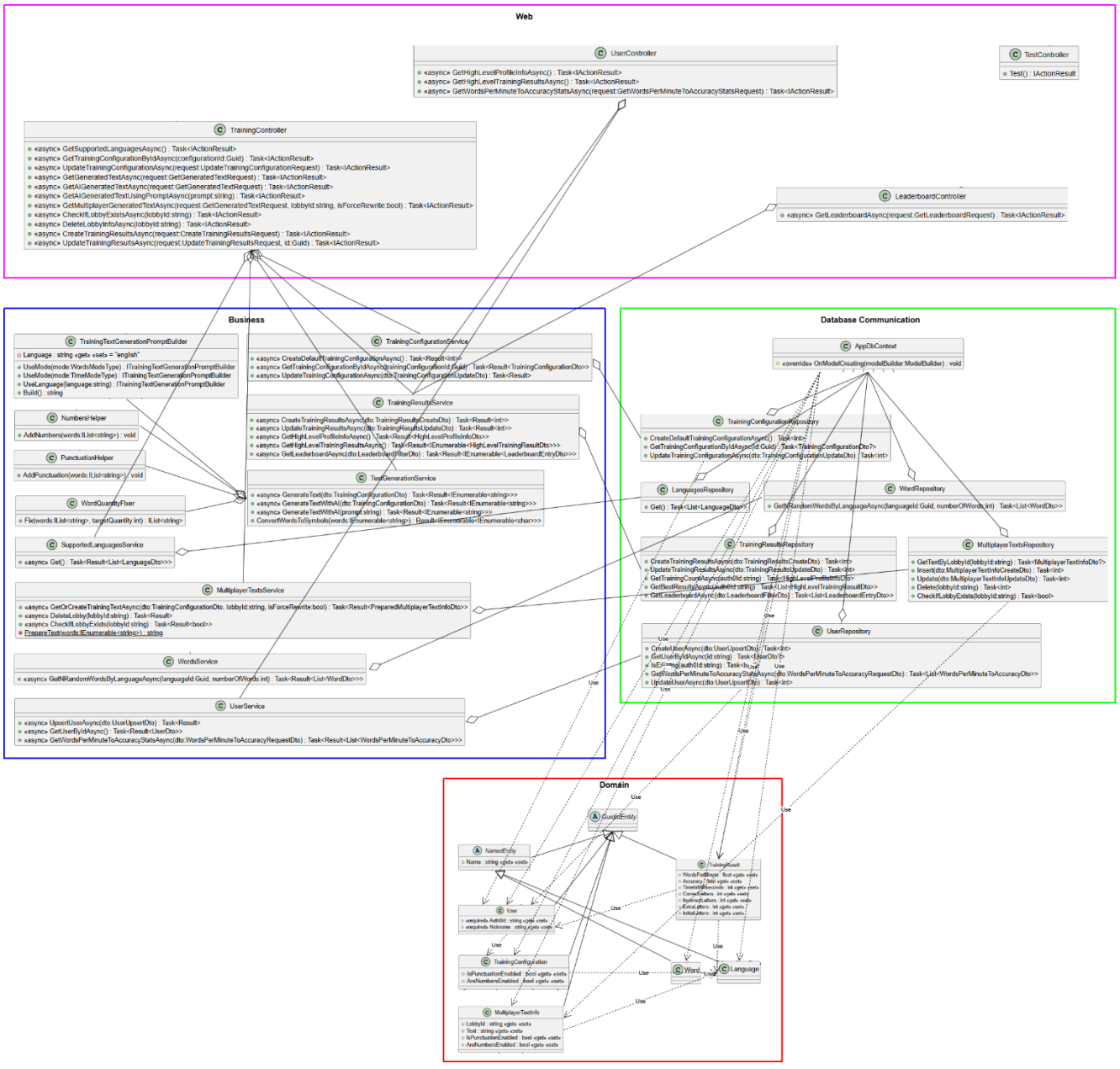

Рисунок 3.9 – Діаграма класів із чотиришаровою архітектурою

Перший шар, позначений пурпурним кольором у верхній частині діаграми, представляє веб-шар системи, відповідальний за взаємодію з клієнтами. Він включає класи-контролери, такі як UserController, TrainingController та

LeaderboardController, які приймають зовнішні HTTP-запити від клієнтів. Ці контролери відповідають за маршрутизацію запитів, перевірку вхідних даних, виклик відповідних сервісів бізнес-логіки та формування HTTP-відповідей. Наприклад, UserController реалізує функції управління користувачами, такі як збереження та отримання користувацьких даних, а TrainingController — обробляє запити, пов'язані з тренуваннями. Цей шар забезпечує зв'язок між клієнтським застосунком і серверною частиною, а також виступає точкою входу до системи.

Другий шар, позначений синім кольором у лівій частині діаграми, відповідає за бізнес-логіку, яка є ядром системи. Цей шар включає класи, які реалізують алгоритми та правила роботи застосунку. Наприклад, TrainingConfigurationService забезпечує налаштування тренувань на основі вхідних параметрів, MultyplayerTextsService реалізує логіку для багатокористувацького режиму, дозволяючи організувати взаємодію між гравцями. У цьому шарі реалізується основна логіка обробки запитів, така як генерація тексту для тренувань, обробка результатів користувача, управління мовами, і навіть управління правилами гри. Завдяки цьому шар бізнес-логіки відокремлює складні обчислення та алгоритми від інтерфейсу користувача, роблячи код більш структурованим і зручним для підтримки.

Третій шар, позначений зеленим кольором у правій частині діаграми, фокусується на взаємодії з базою даних. Цей шар включає класи-репозиторії, такі як TrainingResultRepository, LanguagesRepository, UserRepository, які забезпечують доступ до даних у сховищі. Кожен репозиторій відповідає за операції створення, читання, оновлення та видалення (CRUD) для конкретної сутності бази даних, наприклад, результати тренувань, тренувальні сесії або інформація про користувачів. Шар комунікації з базою даних ізолює бізнес-логіку від деталей зберігання даних, що сприяє модульності та гнучкості архітектури.

Четвертий шар, позначений червоним кольором у нижній частині діаграми, містить доменні моделі, такі як User, TrainingResult, Word та Language, які є основою для всієї системи. Ці моделі відображають структури даних, що

представляють ключові сутності вебзастосунку. Наприклад, модель User містить інформацію про користувача, а модель TrainingResult — про результати його тренувань. Ці моделі використовуються в усіх шарах системи: у бізнес-логіці для обробки даних, у репозиторіях для зберігання та у веб-шарі для передачі даних клієнтам. Доменні моделі визначають чіткі контракти для взаємодії між компонентами, що забезпечує узгодженість і надійність даних у системі.

Узагальнюючи, дана архітектура забезпечує чіткий розподіл відповідальності між компонентами системи. Перший шар служить інтерфейсом для клієнтів, другий реалізує основну логіку роботи системи, третій забезпечує зберігання та доступ до даних, а четвертий визначає структури даних, якими оперує система. Такий підхід не лише сприяє модульності, гнучкості та масштабованості системи, але й полегшує її розвиток та підтримку в довгостроковій перспективі. Це особливо важливо для забезпечення стабільності роботи системи в умовах зміни вимог або збільшення навантаження.

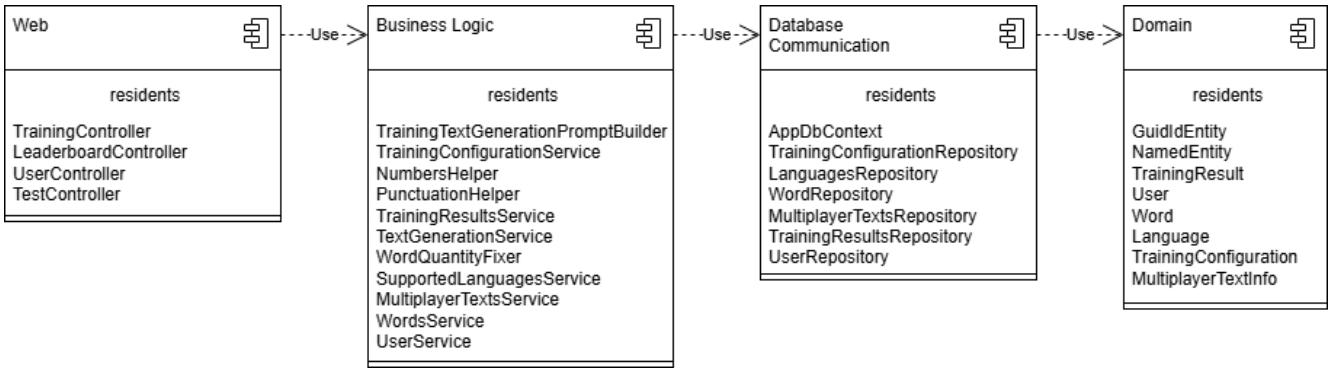

Рисунок 3.10 – Діаграма компонентів

Згідно з діаграмою класів, яка відображає чотиришарову архітектуру системи, було побудовано діаграму компонентів для вебзастосунку тренування швидкості введення даних на клавіатурі (див. рис. 3.10). Діаграма компонентів деталізує структуру системи, розділяючи її на окремі функціональні блоки, кожен з яких виконує специфічну роль у загальній архітектурі. Такий підхід дозволяє забезпечити модульність, підвищити зрозумілість архітектури та полегшити розробку і підтримку системи.

На основі виділених чотирьох шарів — презентаційного, бізнес-логіки, доступу до даних та доменних моделей — кожен компонент представлено окремою сутністю з чіткими залежностями.

Презентаційний компонент відповідає за взаємодію з клієнтами та забезпечує прийом і обробку HTTP-запитів. Цей компонент включає функції маршрутизації та передачі даних між клієнтом і внутрішніми компонентами системи. Наприклад, такі модулі, як UserController або TrainingController, є частиною цього компонента.

Компонент бізнес-логіки реалізує основні алгоритми та правила роботи системи. У цьому компоненті обробляються ключові запити, такі як генерація тексту для тренувань, обробка результатів користувача та управління багатокористувацькими режимами. Його побудова забезпечує абстрагування бізнес-логіки від деталей реалізації нижчих рівнів системи, таких як робота з базою даних.

Компонент доступу до даних відповідає за взаємодію з базою даних. Він ізолює бізнес-логіку від низькорівневих операцій, таких як збереження, отримання чи оновлення даних. Репозиторії цього компонента чітко визначають правила роботи з кожною сутністю, такою як користувачі, результати тренувань або мовні налаштування.

Доменний компонент служить основою для всієї системи, забезпечуючи визначення основних сутностей і структур даних. Цей компонент використовують усі інші шари для обміну даними, що забезпечує їх узгодженість і цілісність.

Дана діаграма підкреслює переваги чотиришарової архітектури, яка дозволяє досягти високої гнучкості, структурованості та ефективного розподілу завдань між частинами системи. Це забезпечує стійкість системи до змін і значно полегшує її подальший розвиток і підтримку.

3.4 Побудова діаграми пакетів

Діаграми пакетів слугують спрощеним способом представлення діаграм класів, забезпечуючи більш високий рівень абстракції. Кожен пакет на діаграмі

представляє логічну групу програмних елементів, таких як класи, інтерфейси або інші структури, що об'єднані спільною функціональністю роллю. Такий підхід дозволяє ефективно організувати компоненти системи, полегшуючи її аналіз.

Головною метою діаграм пакетів є побудова ієрархії залежностей між логічними групами елементів. Вони дають змогу чітко побачити, які пакети взаємодіють між собою, яким чином побудована їхня структура та які залежності існують між ними. Це дозволяє розробникам оцінити загальний рівень зв'язності системи, що є важливим аспектом у процесі проєктування архітектури.

Висока зв'язність між пакетами може ускладнювати розробку та підтримку системи, оскільки зміни в одному пакеті можуть вплинути на інші. Використовуючи діаграми пакетів, можна виявити ці залежності та почати розробляти рішення для її зниження. Наприклад, шляхом впровадження чітких інтерфейсів взаємодії, розбиття великих пакетів на менші або перерозподілу відповідальностей між ними[14].

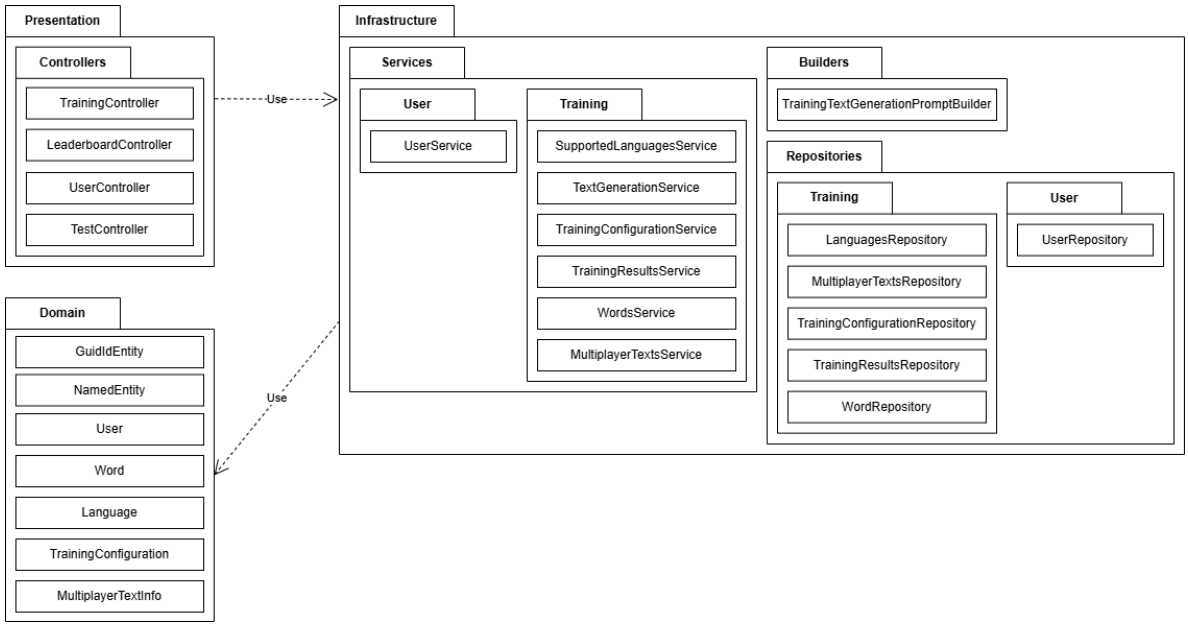

Рисунок 3.11 – Діаграма пакетів

Діаграма пакетів вебзастосунку для тренування швидкості введення тексту на клавіатурі (рис. 3.11) показує логічну організацію системи, поділену на кілька основних груп:

1. пакет «Web» – містить усі компоненти, що відповідають за взаємодію із зовнішнім світом, зокрема контролери, які обробляють HTTP-запити;

2. пакет «Business» – представляє бізнес-логіку застосунку, включаючи класи, які реалізують основні функції, такі як обробка тренувань, генерація текстів і управління статистикою;

3. пакет «Data Access» – об’єднує репозиторії та інші компоненти, що забезпечують доступ до бази даних і роботу з нею;

4. пакет «Domain» – включає доменні моделі, які є фундаментальними сутностями системи.

3.5 Вибір стеку технологій

3.5.1 Front-end

У початковій версії вебзастосунку, розробленій у рамках кваліфікаційної роботи бакалавра, для фронтенду було обрано JavaScript-бібліотеку React. Цей вибір був зроблений через його високу популярність, зручність у використанні та оптимізацію для створення інтерактивних вебінтерфейсів. Реалізація на основі React дозволила створити стабільний, продуктивний і гнучкий інтерфейс, що забезпечив чудовий користувацький досвід для тренування швидкості введення тексту на клавіатурі. Ця бібліотека не перестає бути одним з лідерів за популярністю на ринку (див рис. 3.12)[15].

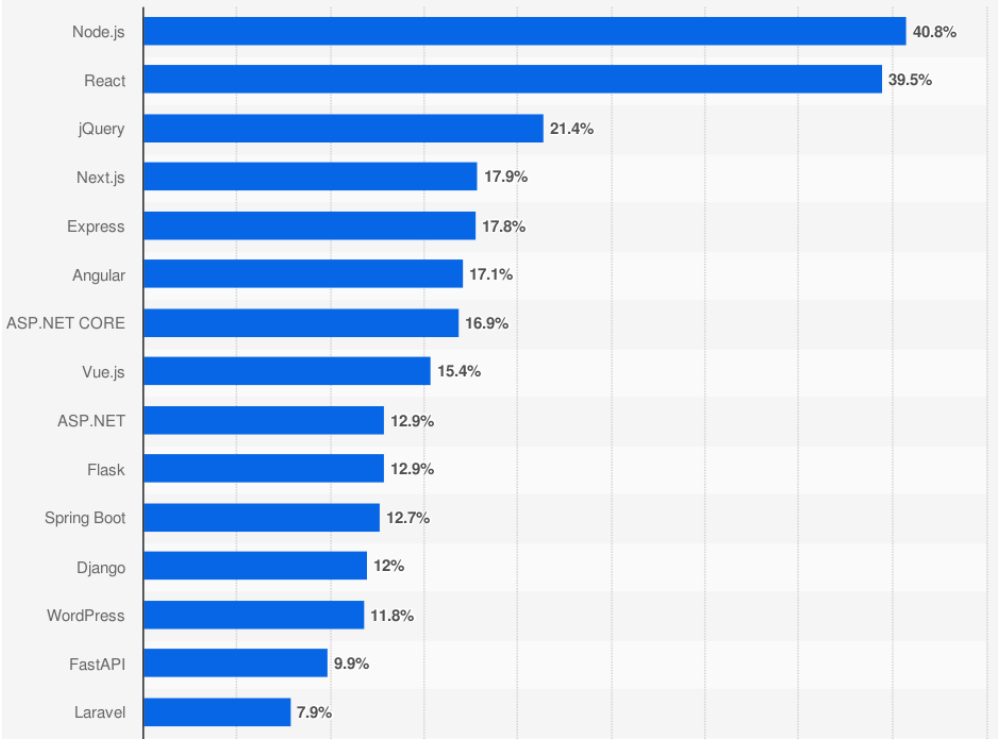

Рисунок 3.12 – Топ 15 вебфреймворків у 2024 році за версією Stack Overflow по кількості репозиторіїв

У новій версії вебзастосунку, що розробляється в рамках магістерської кваліфікаційної роботи, рішення використовувати React залишається незмінним. Це обґрунтовується наступними причинами:

1. React забезпечує плавну роботу інтерфейсу навіть за умов постійного оновлення компонентів, що критично важливо для відображення результатів у реальному часі під час тренувань;

2. використання React у новій версії дозволяє зберегти структуру та ідеї початкової розробки, що спрощує процес міграції та розвитку проєкту;

3. React активно розвивається і підтримується широкою спільнотою, що надає доступ до великої кількості бібліотек, інструментів і готових рішень, які можна легко інтегрувати в проєкт;

4. React добре оптимізований для високонавантажених систем, завдяки механізму Virtual DOM, який мінімізує реальні маніпуляції з DOM, що прискорює роботу застосунку;

5. React дозволяє налаштувати архітектуру проєкту відповідно до потреб, що робить його універсальним інструментом для розробки.

3.5.2 Back-end

Для розробки Web API у початковій версії вебзастосунку було обрано фреймворк ASP.NET Core. Це рішення зберігається і в новій версії, розроблюваній у рамках магістерської кваліфікаційної роботи. ASP.NET Core займає четверту позицію серед бекенд вебфреймворків (див. рис. 3.12) і його вибір має чітке обґрунтування, враховуючи сучасний розвиток платформи .NET та її унікальні переваги.

До появи .NET Core фреймворк ASP.NET був орієнтований виключно на операційну систему Windows, що обмежувало його універсальність і популярність серед розробників, які віддають перевагу кросплатформним рішенням. Однак із виходом .NET Core і подальшою еволюцією до .NET 5 і вище платформа набула повної кросплатформності, що значно розширило її можливості та сферу використання. Завдяки цьому ASP.NET Core став популярним вибором для створення високопродуктивних і масштабованих API-систем[16].

У порівнянні з іншими фреймворками, такими як Express.js, Spring Boot або Django, ASP.NET Core має свої унікальні переваги, які зумовлюють його використання:

1. ASP.NET Core підтримує розробку і розгортання на різних операційних системах, таких як Windows, macOS і Linux, що робить його універсальним рішенням для різних середовищ;

2. завдяки оптимізації під .NET Runtime, ASP.NET Core забезпечує високу швидкість обробки запитів і зменшене споживання ресурсів, що особливо важливо для вебзастосунку, який обробляє значну кількість одночасних запитів під час тренувань;

3. ASP.NET Core інтегрований у велику екосистему інструментів і технологій Microsoft, таких як Visual Studio, Azure, Entity Framework та інші, що спрощує

налаштування, розробку та розгортання. Особливо зручними є хмарні рішення Azure, які легко інтегруються із застосунками, побудованими на ASP.NET Core;

4. ASP.NET Core дозволяє створювати легкі і модульні додатки завдяки системі middleware, що дає змогу налаштовувати обробку запитів під потреби конкретного проєкту;

5. платформа постійно оновлюється, додаючи новий функціонал, покращення продуктивності та сумісність із сучасними стандартами.

Збереження ASP.NET Core у новій версії вебзастосунку також сприяє спадкоємності архітектури проєкту. Це дозволяє використовувати напрацювання попередньої версії, що зменшує витрати на перенесення логіки та спрощує розширення функціональності.

3.5.3 Аутентифікація та авторизація

Перехід на Auth0 був обумовлений низкою причин, що стосуються як технічних, так і організаційних аспектів розробки. Початкова реалізація аутентифікації на основі токенів доступу та оновлення була ефективною, але мала свої обмеження, особливо в контексті зростаючих вимог до безпеки, масштабованості та спрощення підтримки.

Причини переходу:

1. реалізація і підтримка власного механізму потребує постійного оновлення для відповідності сучасним стандартам безпеки. Це включає управління терміном дії токенів, ротацію ключів, виявлення шахрайства та захист від атак, таких як повторне використання токенів. Все це створює значне навантаження на команду розробки;

2. сучасні загрози, такі як атаки на сесію, підробка токенів та інші вразливості, вимагають складних рішень, які часто важко реалізувати самостійно;

3. із зростанням кількості користувачів потреба в масштабуванні системи аутентифікації стала очевидною. Власне рішення було недостатньо гнучким для обробки великого обсягу запитів або інтеграції з іншими сервісами;

4. все більше користувачів віддають перевагу входу через сторонні платформи, такі як Google, Facebook, Discord тощо. Реалізація такої інтеграції вручну є складною та часозатратною;

5. залучення готового рішення дозволяє команді зосередитися на основній функціональності застосунку, а не на розробці механізму аутентифікації;

Переваги переходу на Auth0:

1. Auth0 надає готове рішення, що усуває необхідність самостійно підтримувати механізми аутентифікації, управління токенами та безпеки;

2. Auth0 забезпечує підтримку сучасних стандартів OAuth 2.0[17] та OpenID Connect, багатофакторну аутентифікацію (MFA), захист від шахрайства та автоматичну ротацію ключів шифрування;

3. Auth0 легко масштабується для роботи з великою кількістю користувачів і підтримує глобальний доступ через свої інфраструктури;

4. Auth0 дозволяє легко додати підтримку входу через Google, Facebook, GitHub, Microsoft та інші платформи;

5. через Auth0 Dashboard можна керувати дозволами, політиками безпеки, часом життя токенів та іншими параметрами;

6. Auth0 надає SDK та бібліотеки для популярних платформ, таких як ASP.NET Core і React, що значно скорочує час інтеграції в існуючу систему.

Таким чином, перехід на Auth0 забезпечив не лише зручність управління ідентифікацією користувачів, але й значно підвищив безпеку та масштабованість вебзастосунку. Це рішення дозволяє розробникам зосередитися на основному функціоналі проєкту, довіривши складні аспекти аутентифікації спеціалізованій платформі.

3.5.4 Хостинг

В якості хостингу обрано постачальник хмарних послуг Microsoft Azure. Він пропонує низку переваг для хостингу вебзастосунків, що робить його одним із найкращих варіантів на ринку хмарних платформ. У порівнянні з іншими

провайдерами, такими як AWS та Google Cloud, Azure має свої унікальні особливості, які підвищують його привабливість, особливо для застосунків, розроблених із використанням екосистеми Microsoft.

Azure забезпечує високу інтеграцію з продуктами Microsoft, такими як Visual Studio, .NET, Entra ID та Office 365, що робить його природним вибором для розробників, які працюють у середовищі Microsoft. Це значно спрощує розробку, тестування та розгортання програм, оскільки всі інструменти та сервіси вже інтегровані один з одним. У цьому аспекті Azure має перевагу над AWS, який хоч і пропонує більшу кількість сервісів, але не забезпечує такого рівня синхронізації із програмами Microsoft[18].

Однією з головних переваг Azure є його зручність у використанні завдяки фокусуванню на платформах як послугах (PaaS). Сервіси, такі як Azure App Service, надають розробникам готове середовище для розгортання та підтримки вебзастосунків без необхідності займатися управлінням інфраструктурою. AWS і Google Cloud пропонують схожі сервіси, але їх реалізація часто вимагає більше налаштувань і технічних знань.

Azure SQL є ще однією сильною стороною платформи. Це повністю керована реляційна база даних із підтримкою масштабування та високої доступності. Даний сервіс особливо привабливий для застосунків, які вже використовують SQL Server у локальній інфраструктурі, оскільки перехід до хмари є інтуїтивно зрозумілим. У цьому аспекті Azure SQL може бути більш зрозумілим для розробників порівняно з аналогами, такими як Amazon RDS чи Google Cloud SQL, які хоч і підтримують схожі функції, але не мають такого рівня інтеграції з екосистемою Microsoft.

Щодо App Service Plan, Azure забезпечує зручний механізм керування ресурсами, дозволяючи розробникам налаштовувати параметри продуктивності, залежно від вимог проєкту. Інші провайдери, такі як AWS Elastic Beanstalk чи Google App Engine, також надають схожі можливості, але Azure виділяється своєю зручністю для розробників, що вже знайомі з інструментами Microsoft.

Ще однією важливою перевагою Azure є його глобальна присутність і широка мережа дата-центрів. Хоча AWS має найбільшу кількість дата-центрів, Azure пропонує конкурентоспроможне покриття та дозволяє розробникам легко розгортати застосунки в регіонах, наближених до кінцевих користувачів, що мінімізує затримки.

3.5.5 Безпека

Кожен публічний продукт має бути надійно захищений від зловмисників. В якості мережевого захисту обрано платформу Cloudflare.

Використання Cloudflare для вебзастосунків пропонує численні переваги, особливо завдяки його багатофункціональності, доступності безкоштовного плану та широкому спектру вбудованих сервісів. У поєднанні із сучасними вимогами до безпеки та продуктивності Cloudflare виступає не лише як засіб оптимізації роботи вебзастосунків, але і як потужний інструмент для захисту мережевої інфраструктури, зокрема через сценарії використання Zero Trust.

Cloudflare надає такі базові сервіси навіть на безкоштовному плані, як захист від DDoS-атак, оптимізація продуктивності через глобальну CDN, DNS-менеджмент та автоматична мінімізація ресурсів (наприклад, JavaScript, CSS). Завдяки глобальній мережі дата-центрів, Cloudflare дозволяє знизити затримки доступу до вебзастосунку, забезпечуючи блискавичну доставку контенту користувачам незалежно від їхнього місця розташування. У цьому контексті Cloudflare виступає унікальним гравцем, адже пропонує потужний набір функцій навіть безкоштовно, у той час як подібні сервіси у конкурентів, таких як AWS CloudFront або Akamai, вимагають платної підписки[19].

На безкоштовному плані Cloudflare також забезпечує:

1. захист від атак на прикладному рівні. Завдяки Web Application Firewall (WAF) можна налаштувати базові правила захисту, які блокують спроби експлуатації відомих вразливостей, як-от SQL-ін'єкцій чи XSS;

2. автоматичну оптимізацію для зменшення розміру файлів і забезпечення швидшого завантаження, наприклад, через мінімізацію HTML, JavaScript і CSS, а також стискання зображень;

3. автоматичне встановлення сертифікатів для забезпечення HTTPS-з'єднань, що спрощує захист даних користувачів;

4. швидкий і безпечний DNS, що дозволяє значно знизити час очікування запитів;

5. розподіл трафіку завдяки інтелектуальному маршрутизаційному механізму. Cloudflare може автоматично спрямовувати запити через найоптимальніші маршрути.

Cloudflare також містить сервіс Zero Trust, який стає все більш популярним у сфері кібербезпеки. Zero Trust базується на принципі "не довіряй нічому і перевіряй усе", забезпечуючи максимальний захист ресурсів навіть усередині корпоративної мережі.

Cloudflare надає рішення, які дозволяють забезпечити захищений доступ до внутрішніх систем і даних. Наприклад, через Cloudflare Access (одну з функціональних можливостей Zero Trust) можна обмежити доступ до ресурсів за допомогою аутентифікації через сторонні провайдери ідентифікації, такі як Google, Microsoft чи GitHub. Це дозволяє створити гнучку систему доступу до ресурсів, яка залежить від ідентифікації та контексту (наприклад, географічного розташування, пристрою, IP-адреси).

Cloudflare Access, як частина Zero Trust, інтегрується з Cloudflare Gateway, щоб забезпечити безпечну маршрутизацію трафіку, блокування небезпечних доменів і застосування політик безпеки на рівні DNS та HTTP. Це дозволяє організаціям створити гнучку, масштабовану та безпечну архітектуру доступу до своїх ресурсів.

У порівнянні з конкурентами, такими як Zscaler або Palo Alto Prisma Access, Cloudflare надає подібні функції на доступнішому рівні, що робить його привабливим для малих і середніх бізнесів. Завдяки легкості інтеграції та

налаштування, Cloudflare стає відмінним вибором для проєктів будь-якого масштабу, забезпечуючи високу продуктивність і надійність без необхідності великих інвестицій у хмарну інфраструктуру.

Висновки до розділу 3

У підсумку, поєднання React, ASP.NET Core, Microsoft Azure і Cloudflare забезпечує створення сучасної, масштабованої архітектури вебзастосунку, яка відповідає вимогам високої продуктивності, безпеки та гнучкості. Використання React дозволяє створювати динамічний інтерфейс користувача, оптимізуючи взаємодію з клієнтом, а ASP.NET Core забезпечує потужну серверну основу, здатну ефективно обробляти запити та реалізовувати бізнес-логіку.

Хмарна платформа Microsoft Azure додає цій архітектурі додаткову надійність, завдяки використанню сервісів Azure App Service, Azure SQL і App Service Plan. Вона дозволяє легко керувати ресурсами, забезпечувати високу доступність системи та адаптуватися до зростаючих вимог у разі розширення користувацької бази. Інтеграція Cloudflare додатково підсилює систему через глобальну мережу доставки контенту (CDN), автоматичну оптимізацію продуктивності, захист від кібератак та впровадження сучасного підходу Zero Trust для забезпечення безпеки доступу.

Така архітектура спрощує подальший розвиток і підтримку застосунку, дозволяючи легко інтегрувати нововведення. Завдяки цьому система не тільки відповідає потребам, але й готова до адаптації в умовах постійного технологічного прогресу. Крім того, вибір технологій з єдиної екосистеми (Microsoft .NET і Azure) спрощує процеси розробки, тестування та розгортання, знижуючи ризики та час впровадження оновлень.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ

4.1 Підготовка інфраструктури

Для полегшення процесу розробки та подальшого впровадження необхідно підготувати інфраструктуру. Це є стратегічно важливим кроком, який позитивно впливає на впровадження продукту завдяки підвищенню його стабільності, продуктивності та безпеки. Створення стандартизованої інфраструктури на основі Microsoft Azure та Cloudflare забезпечує передбачуваність процесів розробки, розгортання та тестування.

Платформи на кшталт Azure Portal спрощують керування ресурсами та забезпечують гнучкість у масштабуванні, що особливо важливо для динамічно розвиваються застосунків. Наприклад, Azure App Service та Azure SQL дозволяють безперебійно адаптуватися до зростання кількості користувачів і навантаження, забезпечуючи високу доступність і продуктивність. У той же час Cloudflare сприяє оптимізації продуктивності та підвищенню рівня безпеки, захищаючи систему від кіберзагроз і забезпечуючи швидку доставку контенту.

4.1.1 Доменне ім’я

Першочергово, для підтримки брендингу та полегшеного налаштування всіх необхідних сервісів у подальшому необхідно орендувати доменне ім’я.

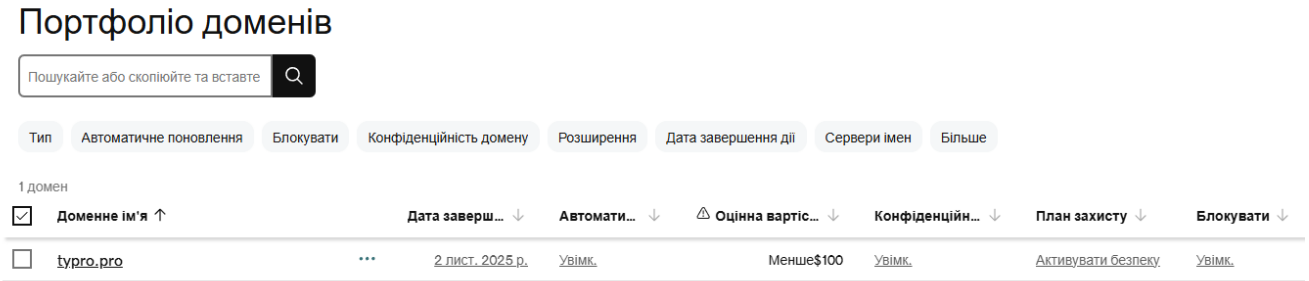

Рисунок 4.1 – Панель керування доменами на GoDaddy

Однією з найпопулярніших платформ є GoDaddy. В ході оцінки витрат та відповідності до критерії бренду обрано домен typro.pro (див. рис. 4.1).

4.1.2 Cloudflare

Наступним кроком виступає налаштування мережевої безпеки та регулювання доступу до застосунку.

По перше, треба змінити імена DNS серверів на стороні реєстратора. Для цього отримуємо нові імена надані Cloudflare (див. рис. 4.2).

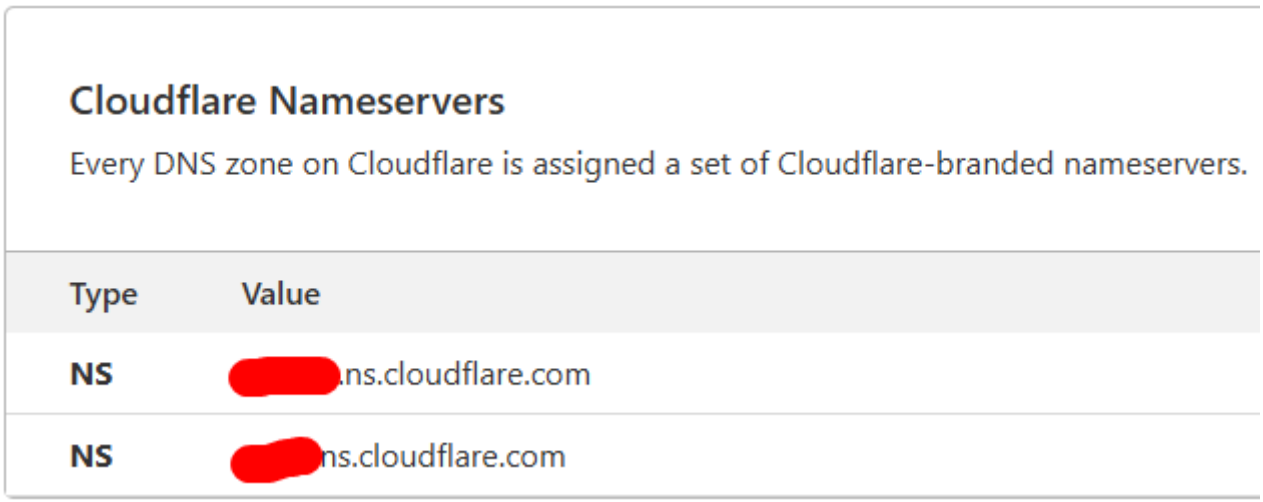

Рисунок 4.2 – Назви DNS серверів виданих Cloudflare

Далі переходимо на GoDaddy та встановлюємо їх (див. рис. 4.3).

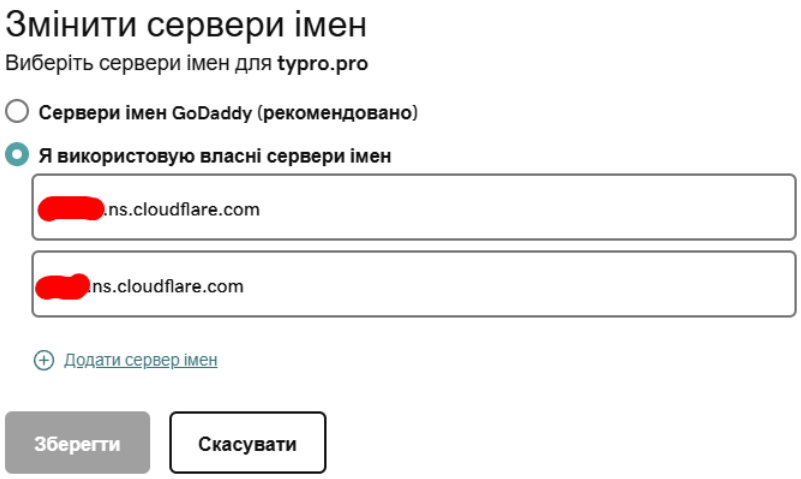

Рисунок 4.3 – Налаштування DNS на стороні реєстратора

Порядок встановлення імен серверів не має значення.

4.1.3 Azure

Інфраструктура вебсистеми розгорнута у постачальнику хмарних послуг Azure. На рисунку 4.4 зображено перелік усі залучених сервісів.

Name	Type	Resource group	Location	Subscription
Application Insights Smart Detection	Action group	PreProd	Global	TyproDiploma Subscripti...
Clawfang	Azure DevOps organization	VisualStudioOnline-B73B...	West Europe	Basic Subscription
DefaultWorkspace-5385e38a-2b9b-4ea2-8fe9-c...	Log Analytics workspace	DefaultResourceGroup-PLC	Poland Central	TyproDiploma Subscripti...
Dev	Application Insights	PreProd	Poland Central	TyproDiploma Subscripti...
Dev-1	App Service plan	PreProd	Poland Central	TyproDiploma Subscripti...
typro	SQL server	PreProd	Poland Central	TyproDiploma Subscripti...
typro (typro/Typro)	SQL database	PreProd	Poland Central	TyproDiploma Subscripti...
Typro-Api	App Service	PreProd	Poland Central	TyproDiploma Subscripti...
Typro-FE	App Service	PreProd	Poland Central	TyproDiploma Subscripti...
TyproAI-Api	App Service	PreProd	Poland Central	TyproDiploma Subscripti...

Рисунок 4.4 – Хмарна інфраструктура вебзастосунку Typro

Кожен із сервісів відіграє важливу роль у функціонуванні вебзастосунку, забезпечуючи управління, моніторинг і роботу ключових компонентів.

Application Insights Smart Detection (Action group)

Цей ресурс відповідає за виявлення аномалій у роботі застосунку. Він автоматично аналізує дані телеметрії, зібрані з вебзастосунку, і надсилає попередження у разі виявлення проблем, таких як високий час відгуку або часті помилки.

Clawfang (Azure DevOps organization)

Це організація в Azure DevOps, яка інтегрується з розробкою застосунку, забезпечуючи управління вихідним кодом, автоматизацію CI/CD процесів і моніторинг змін у репозиторіях. Вона дозволяє налаштувати автоматичне розгортання та тестування застосунку.

DefaultWorkspace (Log Analytics workspace)

Цей простір для логів забезпечує збір, аналіз і зберігання даних телеметрії. Він дозволяє отримувати інформацію про продуктивність, безпеку та події, які відбуваються у системі. Це важливий компонент для моніторингу та діагностики.

Dev (Application Insights)

Application Insights використовується для збору даних про роботу застосунку в середовищі розробки (Dev). Це включає метрики, такі як швидкість відповіді, кількість запитів, рівень помилок тощо, що допомагає виявляти проблеми ще на етапі розробки.

Dev-1 (App Service Plan)

Цей сервіс планує ресурси для розгортання вебзастосунків. Він визначає потужність обчислювальних ресурсів, які виділяються для App Services, таких як кількість ядер CPU, оперативна пам'ять та інші параметри. "Dev-1" відповідає за обслуговування застосунків у середовищі розробки.

typro (SQL server)

SQL Server забезпечує зберігання та управління даними вебзастосунку. Він виконує роль сервера для обробки запитів до бази даних, яка зберігає інформацію про користувачів, результати тренувань, налаштування тощо.

typro (typro/Typro) (SQL database)

Ця база даних є реляційним сховищем, яке безпосередньо використовується застосунком. Вона інтегрована з бекендом для роботи з даними, забезпечуючи швидке та надійне зберігання й отримання інформації.

Typro-Api (App Service)

Це сервіс для розгортання бекенд API вебзастосунку. Він обробляє запити клієнтської частини, виконує бізнес-логіку та забезпечує взаємодію з базою даних через SQL Server.

Typro-FE (App Service)

Front-end застосунок, розгорнутий за допомогою цього сервісу, відповідає за взаємодію з кінцевим користувачем. Typro-FE працює як інтерфейс користувача, обробляючи запити до API та відображаючи результати роботи системи.

TyproAI-Api (App Service)

Це сервіс, що відповідає за інтеграцію зі штучним інтелектом. Він забезпечує функціонал для генерації текстів або іншої логіки, пов'язаної з AI, обробляючи відповідні запити від клієнтської частини чи бекенду.

4.1.4 Auth0

Враховуючи простоту застосування системи, вона не потребує витончених рішень автентифікації та авторизації користувачів. На рисунку 4.5 зображено перелік Auth0-застосунків системи.

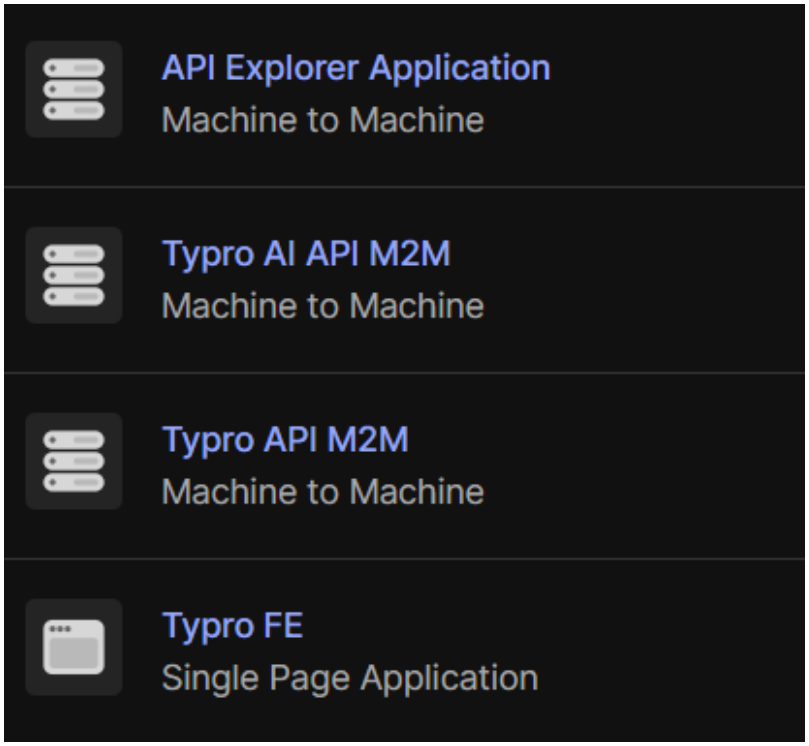

Рисунок 4.5 – Auth0-застосунки системи

Додатково, є поняття Auth0-APIs, що визначає API-сервіси системи, налаштування видачі токенів для них, параметрів audience, ролей тощо (див. рис. 4.6).

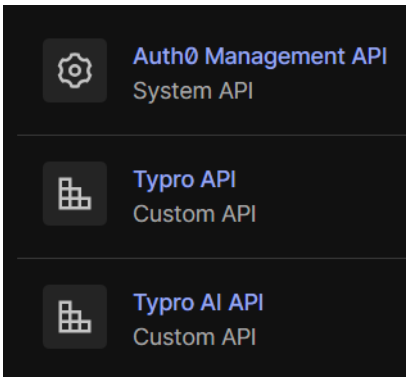

Рисунок 4.6 – Auth0-APIs системи

Окремо варто звернути увагу на можливість використання власного домену для сторінки аутентифікації (див. рис. 4.7).

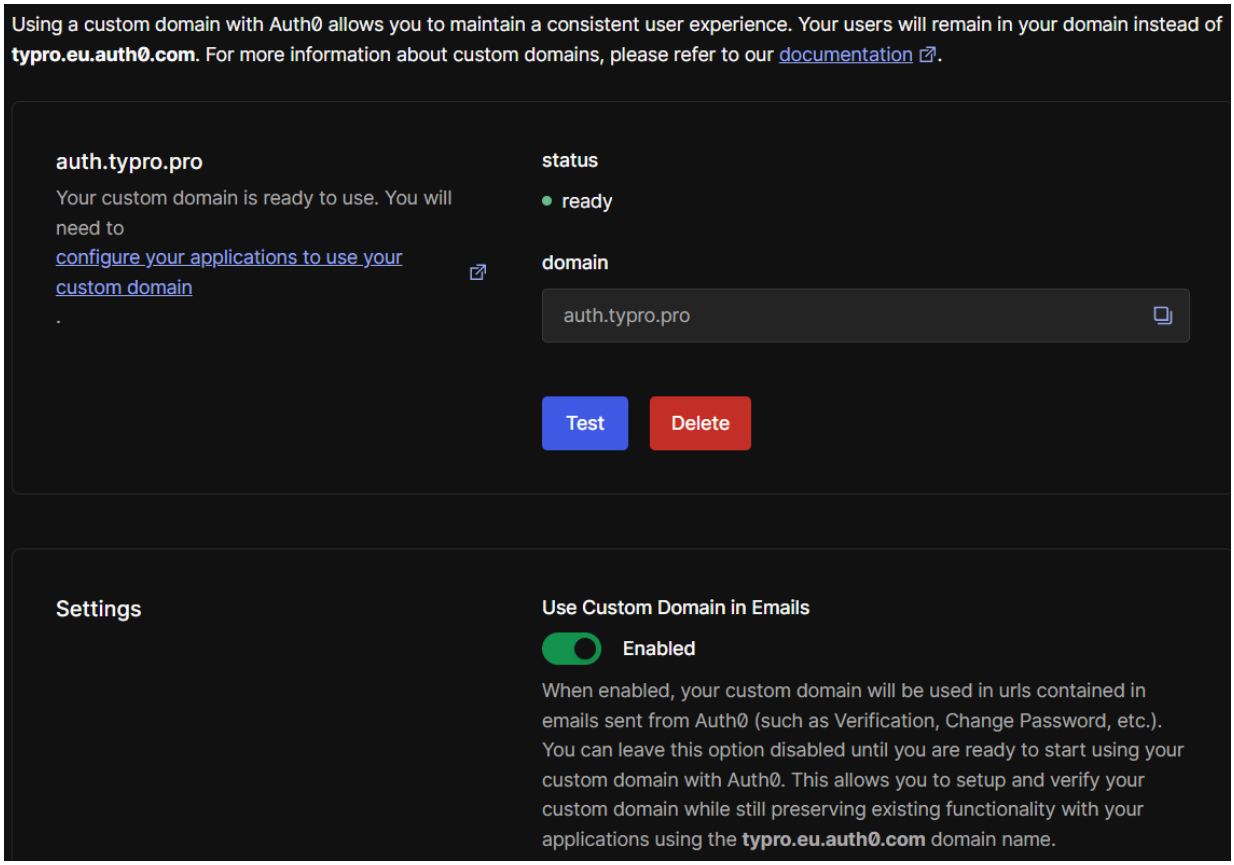

Рисунок 4.7 – Налаштування користувацького домену

Таким чином досягається більша залученість зовнішніх сервісів до бренду користувача.

4.2 Реалізація функціоналу вебзастосунку

4.2.1 Інтеграція штучного інтелекту

Основним нововведенням у цій версії застосунку є інтеграція зі штучним інтелектом Як вже було зазначено раніше, в рамках системи цю роль відіграє окремий web API.

Ключовим аспектом роботи цього сервісу є TextGenerationService, який максимально абстрагований він зовнішньої бізнес-логіки, що робить його ідеальним кандидатом для повторного використання в майбутньому:

```
public class TextGenerationService(
    IOpenAiConfiguration openAiConfiguration,
    ChatClient chatClient) : ITextGenerationService
{
    public async Task<string> Generate(string prompt)
    {
        List<ChatMessage> inputPrompts = [
            new SystemChatMessage(openAiConfiguration.SystemMessage),
            new UserChatMessage(prompt)
        ];

        var options = new ChatCompletionOptions
        {
            FrequencyPenalty = 1,
            PresencePenalty = 1,
            MaxOutputTokenCount = 700
        };

        ClientResult<ChatCompletion> result = await
chatClient.CompleteChatAsync(inputPrompts, options);

        string[] messages = result.Value.Content
            .ToList()
            .Select(e => e.Text)
            .ToArray();

        return string.Join("", messages);
    }
}
```

В той же час, додаткової конфігурації потребує ChatClient – одна із залежностей цього сервісу. Для легкої інтеграції шляхом агрегації, необхідно

задати фабричний метод для створення об’єктів цього класу, використовуючи заздалегідь надану конфігурацію, а саме GPT модель та API ключ:

```
services.AddScoped<ChatClient>(_ => new(openAiConfiguration.Model,
openAiConfiguration.ApiKey));
```

4.2.2 Налагодження комунікації між сервісами

Для автоматизації обробки повідомлень формату запит-відповідь HTTP-клієнтів в рамках спілкування різних API між собою прийнято рішення використати бібліотеку-автогенератор, яка дозволяє зручно працювати з REST API через типізовані HTTP-клієнти - Refit. Вона оптимізує процес написання запитів завдяки використанню інтерфейсів і автоматично генерує код для взаємодії з API. Основною перевагою є простота налаштування: розробнику потрібно лише описати інтерфейс з методами, що відповідають конкретним запитам. Завдяки цьому код стає більш читабельним, що сприяє легшій підтримці й масштабуванню.

Refit забезпечує підтримку асинхронності, інтегрується з Dependency Injection та надає гнучкість у налаштуванні параметрів, таких як базовий URL або політики повтору. Це робить її ідеальною для проектів, де необхідна регулярна взаємодія з REST API, а також у випадках, коли важливі чистота й структурованість коду[20].

Зазвичай, такого роду клієнти представлені у вигляді окремих NuGet пакетів, для можливості інтеграції одночасно в декілької проєктах. Випадок Typro – не виключення. Типову структуру такого пакету представлено на рисунку 4.8.

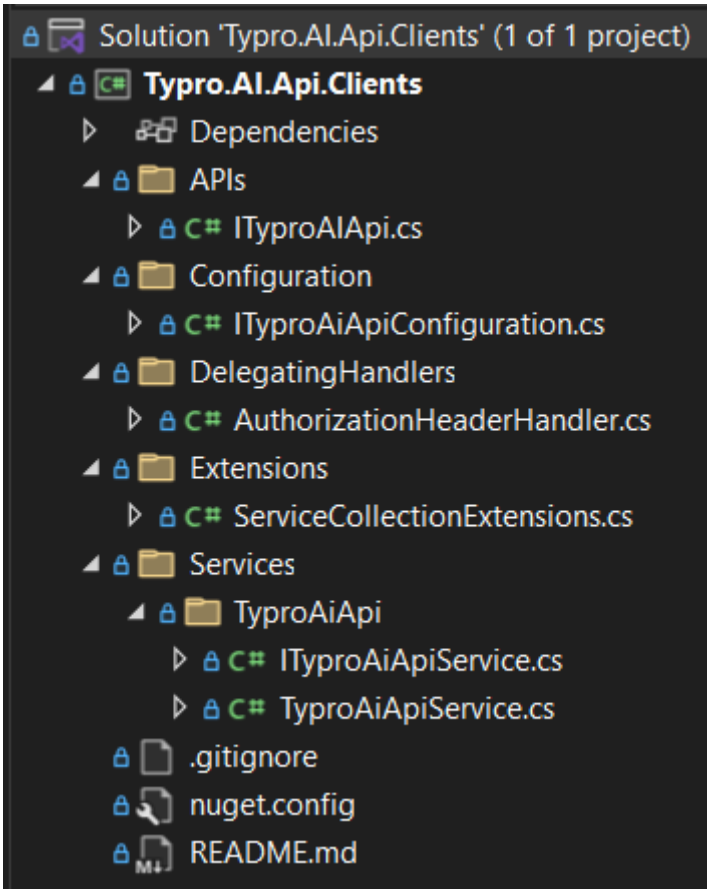

Рисунок 4.8 – Типова структура бібліотеки-клієнту з використанням Refit

Ключові програмні компоненти знаходяться у директорії «APIs». Це і є ті самі інтерфейси, які використовує Refit для генерації HTTP-клієнтів:

```
public interface ITyproAiApi
{
    [Post("/ai/text")]
    Task<ApiResponse<string>>
GenerateText([Body(BodySerializationMethod.Serialized)] string prompt);
}
```

Оскільки бібліотека потребує конфігурування зі сторони користувача, для цього визначається абстракція, яка відповідає за тип даних з необхідними значеннями:

```
public interface ITyproAiApiConfiguration : IAuth0ClientSecretConfiguration
{
    string ApiBasePath { get; init; }

    string Audience { get; init; }
}
```

Авторизація – невід'ємна частина M2M спілкування. Для таких випадків Refit надає можливість інтеграції middleware для модифікації запиту під час відправки. В конкретній реалізації для Typro додаткового залучено постачальник токенів з використанням кешування, для зменшення навантаження для сервіс авторизації:

```
internal class AuthorizationHeaderHandler(
    ITyproAiApiConfiguration typroAiApiConfiguration,
    ITokenProvider tokenProvider) : DelegatingHandler
{
    private const string TyproAiApiIdentifier = "TyproAiApi";

    protected override async Task<HttpResponseMessage>
SendAsync(HttpRequestMessage request, CancellationToken cancellationToken)
    {
        string accessToken = await tokenProvider.GetTokenAsync(
            TyproAiApiIdentifier,
            typroAiApiConfiguration.Audience,
            typroAiApiConfiguration.ClientId,
            typroAiApiConfiguration.ClientSecret);

        request.Headers.Add("Authorization", $"Bearer {accessToken}");

        return await base.SendAsync(request, cancellationToken);
    }
}
```

Сам постачальник токенів містить наступну логіку:

```
public class TokenProvider(
    IAuth0AuthService authService,
    [FromKeyedServices(ServiceKeys.Cache)] IFusionCache fusionCache) :
ITokenProvider
{
    public async ValueTask<string> GetTokenAsync(string identifier, string
audience, string clientId, string clientSecret)
    {
        MaybeValue<string> cachedToken = await
fusionCache.TryGetAsync<string>(identifier);

        if (cachedToken.HasValue)
        {
            return cachedToken.Value;
        }

        AccessTokenResponse tokenResponse = await
authService.GetAccessTokenAsync(audience, clientId, clientSecret);

        var cacheEntryOptions = new FusionCacheEntryOptions
        {
            Duration = TimeSpan.FromSeconds(tokenResponse.ExpiresIn - 10)
        };
```

```
        await fusionCache.SetAsync(identifier, tokenResponse.AccessToken,
cacheEntryOptions);

        return tokenResponse.AccessToken;
    }
}
```

4.2.3 Перехід від Dapper до Entity Framework

Entity Framework є повноцінним ORM[10], який автоматично керує сутностями, їхніми зв'язками та життєвим циклом, що значно спрощує роботу з даними в порівнянні з Dapper. Він автоматично створює таблиці та налаштовує зв'язки між ними на основі визначень у класах, дозволяючи розробникам зосередитися на бізнес-логіці, а не на налаштуваннях чи написанні SQL-запитів. EF підтримує використання LINQ для написання запитів, що робить їх більш читабельними та інтегрованими з кодом, а також знижує ризик помилок, пов'язаних з некоректними SQL-запитами.

Завдяки вбудованій підтримці міграцій база даних легко синхронізується зі змінами в коді, тоді як у Dapper ці процеси доводиться виконувати вручну. EF спрощує роботу зі складними реляційними запитами через механізми завантаження зв'язаних даних, що особливо зручно при роботі з ієрархічними структурами. Він значно скорочує обсяг шаблонного коду для CRUD-операцій, що важливо у великих проєктах.

Прийнято рішення відійти від використання Dapper на користь Entity Framework. Початково, використання першої бібліотеки аргументувалось швидкодією та простотою SQL-запитів, перспективою застосування більш складних скриптів.

За цей час EF зазнав оновлень, які неодноразово були спрямовані на покращення швидкодії. Враховуючи цей факт та те, що він є повноцінною ORM, а не micro ORM, як Dapper, це рішення стало очевидним.

Повертаючись назад до діаграми класів (див. рис. 3.9), структура шару роботи з даними зазнала значних змін. Робота з транзакційністю набула більш

зрозумілої форми, стало легше відмовитись від патерну Unit of Work, взаємодія з даними перетворилась у прямолінійний алгоритм.

Приклад застосування нової бібліотеки:

```
public Task<int> UpdateTrainingConfigurationAsync(TrainingConfigurationUpdateDto
dto)
{
    return dbContext.TrainingConfigurations
        .Where(e => e.Id == dto.Id)
        .ExecuteUpdateAsync(e => e
            .SetProperty(p => p.LanguageId, dto.LanguageId)
            .SetProperty(p => p.AreNumbersEnabled, dto.AreNumbersEnabled)
            .SetProperty(p => p.IsPunctuationEnabled, dto.IsPunctuationEnabled)
            .SetProperty(p => p.TimeModeType, dto.TimeMode)
            .SetProperty(p => p.WordsModeType, dto.WordsMode));
}
```

4.2.4 Зміни у користувацькому інтерфейсі

Першими змін зазнали сторінки входу та реєстрації. Їх замінили сторінки, що надаються зі сторони Auth0. Додатково, вдалось під'єднати сторонні сервіси для входу, такі як Google, Discord та GitHub. На риснуку 4.9 зображено нову сторінку входу в систему.

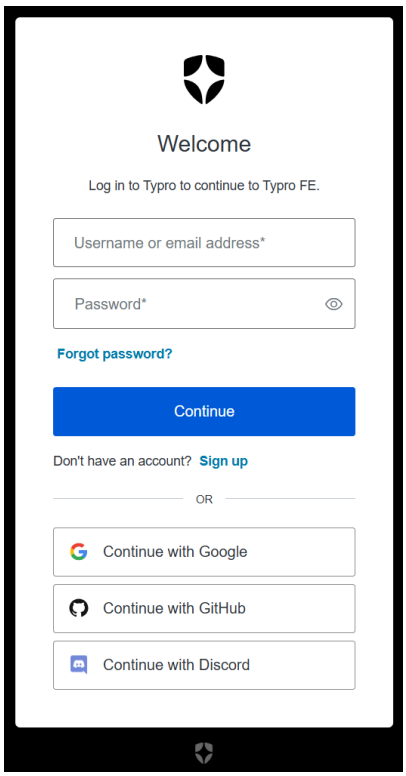

Рисунок 4.9 – Сторінка входу

А на рисунку 4.10 нову сторінку реєстрації.

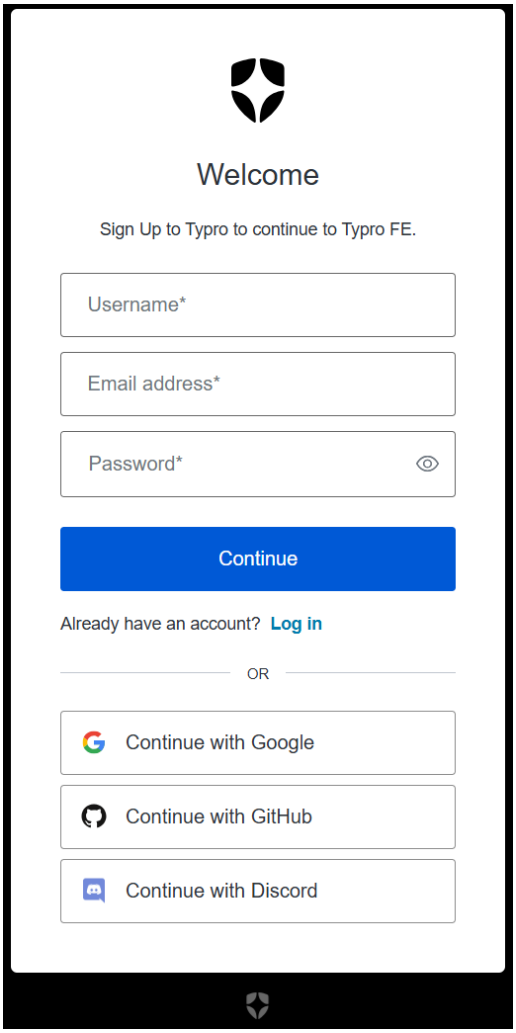

Рисунок 4.10 – Сторінка реєстрації

Також, відбулась модифікація існуючої сторінки застосунку. На головному екрані з’явилась нова кнопка, що вмикає режим взаємодії зі штучним інтелектом. Користувачу пропонується ввести запит, який буде передано до моделі та згенеровано перелік слів на основі нього. До уваги також беруться налаштування, у вигляді кількості слів, наявності пунктуації і т. д. На рисунку 4.11 зображено приклад налаштування тренування для генерації 10 слів на шкільну тематику.

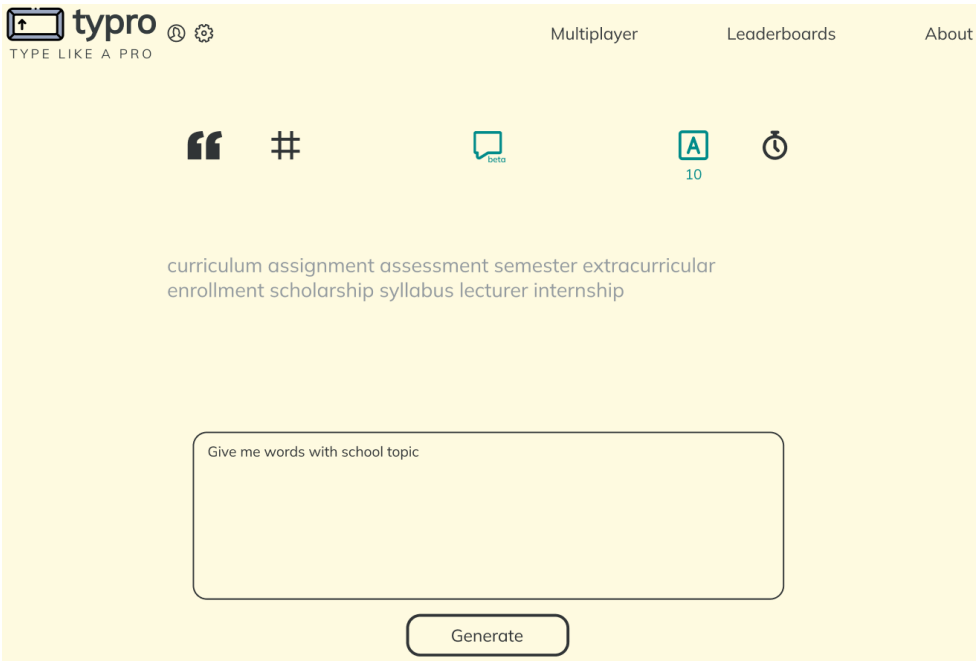

Рисунок 4.11 – Приклад використання нового режиму тренування

На рисунку 4.12 надано схожий приклад, але на тематику слів популярних пісень в розмірі двадцяти п’яти з пунктуацією.

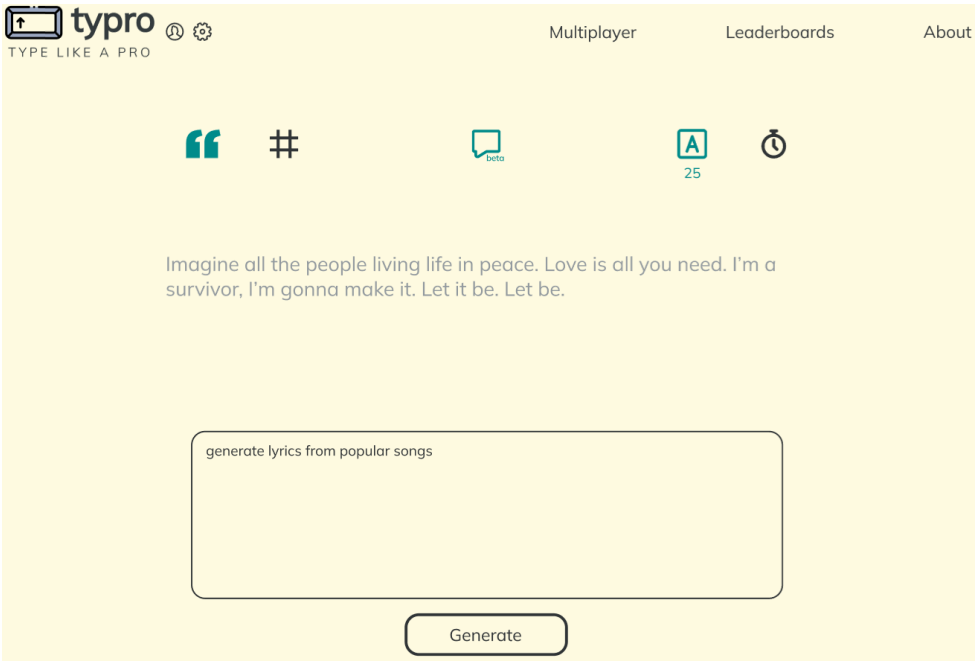

Рисунок 4.12 – Приклад використання нового режиму тренування

Хоч зовнішньо застосунок не зазнав значних змін, але всередині він був значно покращений.

4.3 Тестування роботи вебзастосунку

Будь-яка система потребує списку тест кейсів для проведення тестування. В цьому підрозділі розглядається певний перелік з цього списку з детальним описом.

Таблиця 4.1 – Реєстрація облікового запису

Діючі особи	Користувач, система
Мета	Створення нового облікового запису
Передумова	Користувач виконує дії будучи неавторизованим
1. Успішний сценарій: 1. користувач натискає на іконку профілю; 2. користувача перенаправлено на сторінку аутентифікації Auth0; 3. користувач натискає на текст «Sign up»; 4. користувач вводить необхідні дані; 5. Auth0 створює обліковий запис; 6. користувача перенаправлено назад на сторінку застосунку разом з кодом аутентифікації в URL; 7. проведено обмін коду аутентифікації на access та ID токени користувача.	
Сценарій успішний. Створено новий обліковий запис.	
Розширення	
1a	Користувач заповнив не всі необхідні поля. Форма не проходить перевірку на правильність введених даних та відображає відповідні помилки. Результат: обліковий запис не створено.
2a	Користувач вводить пошту вже існуючого облікового запису. Auth0 повертає відповідь з відповідним повідомленням та виводить його на екран. Результат: обліковий запис не створено.
3a	Стається неочікувана помилка серверу. Результат: обліковий запис не створено.
Всі сценарії розширення успішно виконані.	

Тест кейс у таблиці 4.1 призначений для перевірки процесу створення нового облікового запису через систему аутентифікації Auth0, включаючи перевірку коректності введених даних, обробку можливих помилок і передачу токенів для подальшої автентифікації користувача в застосунку.

Таблиця 4.2 – Вхід до облікового запису

Діючі особи	Користувач, система
Мета	Успішно пройти аутентифікацію
Передумова	Користувач виконує дії будучи неавторизованим
Успішний сценарій: 1. користувач натискає на іконку профілю; 2. користувач вводить необхідні дані (пошта та пароль) і натискає кнопку «Continue»; 3. Auth0 перевіряє відповідність переданих даних до існуючого облікового запису; 4. Auth0 перенаправляє користувача назад на сторінку застосунку з кодом аутентифікації в URL; 5. відбувається отримання токенів шляхом обміну коду аутентифікації.	
Сценарій успішний. Пройдено процес аутентифікації в системі.	
Розширення	
1a	Користувач заповнив не всі необхідні поля. Auth0 виводить відповідне повідомлення на екран. Результат: процес аутентифікації не пройдено.
2a	Користувач вводить пошту неіснуючого облікового запису. Auth0 повертає помилку про неіснуючий акаунт. Результат: процес аутентифікації не пройдено.
3a	Auth0 повертає неочікувану помилку під час відправки запиту. Результат: процес аутентифікації не пройдено.
Всі сценарії розширення успішно виконані.	

Цей тест кейс спрямований на перевірку процесу аутентифікації користувача через Auth0, включаючи введення облікових даних, перевірку їхньої коректності, отримання токенів для доступу, а також обробку можливих помилок, таких як неповне заповнення полів, помилкові дані чи технічні збої.

Таблиця 4.3 – Проходження тренування в режимі для одного користувача

Діючі особи	Користувач, система
Мета	Проходження тренування в режимі використання штучного інтелекту
Передумова	Стабільний доступ до мережі інтернет

Кінець таблиці 4.3

Успішний сценарій: 1. користувач переходить на головну сторінку; 2. користувач обирає режим зі штучним інтелектом; 3. користувач виконує налаштування генерації тексту; 4. користувач починає вводити текст; 5. користувач завершує тренування; 6. користувача перенаправляє на сторінку з результатами; 7. результати тренування відправляються на backend сервер та зберігаються в базу даних.	
Сценарій успішний. Тренування пройдено та результати збережено.	
Розширення	
1а	Користувач перериває роботу застосунку. Результат: тренування не завершено та результати не отримано.
2а	Користувач втрачає доступ до мережі Інтернет до або під час відправки результатів на сервер. Результат: тренування не завершено та результати не отримано.
3а	Backend сервер недоступний. Результат: дані про результати тренування не збережено.
Всі сценарії розширення успішно виконані.	

Цей тест кейс спрямований на перевірку роботи режиму тренування з використанням штучного інтелекту, включаючи генерацію текстів за допомогою AI, дотримання заданих параметрів тренування та коректну взаємодію системи з користувачем за умови стабільного доступу до мережі інтернет.

Таблиця 4.4 – Оновлення псевдоніму користувача

Діючі особи	Користувач, система
Мета	Оновлення псевдоніму користувача
Передумова	Користувач пройшов аутентифікацію у системі

Кінець таблиці 4.4

Успішний сценарій: 1. користувач переходить на сторінку профілю; 2. користувач натискає кнопку «Edit profile»; 3. користувач надає новий псевдонім; 4. користувач натискає кнопку «Save»; 5. відправляється запит на backend сервер для збереження змін; 6. користувач отримує попередження про необхідність перезаходу в акаунт для відображення змін; 7. користувач автоматично виходить з акаунту; 8. користувач знову заходить у свій обліковий запис.	
Сценарій успішний. Псевдонім користувача оновлено.	
Розширення	
1а	Користувач перейшов на іншу сторінку. Результат: все залишається без змін.
2а	Користувач натиснув кнопку «Save» лишивши поле вводу пустим. Результат: виведено повідомлення про некоректність уведеного значення.
3а	Користувач втрачає доступ до мережі Інтернет до або в ході відправки запиту на оновлення даних. Результат: псевдонім не оновлено.
4а	Backend сервер недоступний. Результат: псевдонім не оновлено.
Всі сценарії розширення успішно виконані.	

Цей тест кейс перевіряє процес оновлення псевдоніму користувача у системі. Він включає перевірку введення нового значення, відправлення запиту на сервер для збереження змін, обробку відповіді сервера, а також коректне завершення процесу через перезавантаження сеансу користувача. Додатково враховано сценарії обробки помилок, таких як некоректне введення даних, втрату підключення до мережі або недоступність сервера.

Таблиця 4.5 – Відновлення паролю

Діючі особи	Користувач, система
Мета	Відновити пароль від облікового запису
Передумова	Користувач забув пароль від облікового запису

Кінець таблиці 4.5

<table>
<tr><td colspan="2">Успішний сценарій:
1. користувач натискає на кнопку профілю;
2. Auth0 перенаправляє користувача на сторінку аутентифікації;
3. користувач натискає на текст «Forgot password»;
4. користувач вводить свою пошту;
5. користувач переходить до своєї пошти;
6. користувач натискає на посилання в листі;
7. користувач надає новий пароль до свого облікового запису;
Користувач натискає кнопку «Reset password».</td></tr>
<tr><td colspan="2">Сценарій успішний. Псевдонім користувача оновлено.</td></tr>
<tr><td colspan="2">Розширення</td></tr>
<tr><td>1a</td><td>Користувач допустив помилку в назві пошти. Результат: лист надійшов на іншу пошту. Пароль не відновлено.</td></tr>
<tr><td>2a</td><td>Користувач не отримав лист на пошту. Результат: Пароль не відновлено.</td></tr>
<tr><td>3a</td><td>Користувач неправильно вводить пароль для підтвердження. Результат: Виведено повідомлення про не співпадіння.</td></tr>
<tr><td>4a</td><td>Auth0 сервер недоступний. Результат: Пароль не відновлено.</td></tr>
<tr><td colspan="2">Всі сценарії розширення успішно виконані.</td></tr>
</table>

Цей тест кейс спрямований на перевірку процесу відновлення пароля від облікового запису через систему аутентифікації Auth0. Він охоплює введення користувачем електронної пошти для отримання посилання на відновлення, виконання дій через лист для створення нового пароля та перевірку коректності завершення процесу. Також враховуються сценарії обробки помилок, таких як введення некоректної пошти, технічні проблеми з сервером або невідповідність підтвердження нового пароля, забезпечуючи перевірку надійності та стабільності процесу.

Висновки до розділу 4

Виконання тестів продемонструвало високу функціональність і надійність системи у відповідності до поставлених вимог. Були успішно перевірені ключові сценарії взаємодії користувача із системою, включаючи процеси автентифікації,

відновлення пароля, оновлення профілю, створення нового облікового запису та проходження тренувань із використанням штучного інтелекту. Усі сценарії завершилися очікуваними результатами, підтверджуючи коректність реалізованої бізнес-логіки.

Розширені сценарії виявили належну обробку помилок, таких як некоректне введення даних, втрата підключення до мережі чи недоступність сервера. Це свідчить про готовність системи до реальних умов експлуатації, де користувач може зіткнутися з непередбачуваними ситуаціями.

Результати тестування дозволили ідентифікувати можливі точки для оптимізації, зокрема вдосконалення повідомлень про помилки та покращення часу обробки певних запитів. Загалом, проведені тести підтвердили, що система відповідає встановленим вимогам і готова до впровадження у реальне середовище з високим рівнем стабільності, безпеки та продуктивності.

Окрім перевірки функціональності та стабільності системи, тестування також дозволило оцінити її зручність для користувача. Особливу увагу було приділено сценаріям, що передбачають взаємодію користувача з інтерфейсом, наприклад, введення даних, отримання повідомлень про помилки та перенаправлення на різні сторінки. Результати показали, що система забезпечує інтуїтивно зрозумілий користувацький досвід, мінімізуючи кількість дій для досягнення мети. Це підтверджує, що обрані методи розробки ефективно сприяли створенню зручного, адаптивного рішення, яке може задовольнити потреби кінцевих користувачів.

ВИСНОВКИ

В ході виконання кваліфікаційної магістерської роботи вдосконалено програмне забезпечення для тренування швидкості введення тексту на клавіатурі шляхом інтеграції нових технологій, включаючи використання штучного інтелекту для генерації текстів, покращення інфраструктури хостингу та забезпечення високого рівня безпеки й продуктивності.

Для досягнення визначеної на початку роботи мети виконано наступні завдання:

– проведено аналіз існуючої системи для розуміння можливостей інтеграції штучного інтелекту.

– проведено аналіз наявних рішень зі штучного інтелекту для визначення можливих кандидатів для використання;

– виявлено та детально описано недоліки початкової версії системи, запропоновано ефективні способи їх вирішення;

– визначено нові функціональні та нефункціональні вимоги до вдосконаленої версії вебзастосунку;

– створено відповідні UML-діаграми, зокрема діаграми класів, компонентів, станів і розгортання, для деталізації структури та функціонування системи;

– удосконалено серверну частину системи за допомогою ASP.NET Core, впроваджено сучасні принципи автентифікації з використанням Auth0;

– покращено фронтенд, розроблений на React, для підвищення динамічності та інтерактивності роботи застосунку;

– впроваджено хмарну інфраструктуру на базі Microsoft Azure з використанням App Service, App Service Plan і Azure SQL;

– інтегровано Cloudflare для забезпечення оптимізації продуктивності, захисту від атак і впровадження сучасної моделі Zero Trust для безпечного доступу до ресурсів.

Актуальність вдосконалення системи підтверджено через аналіз предметної області, що виявив необхідність у розширенні функціональності та покращенні її технічних характеристик. Дослідження аналогічних рішень дозволило визначити сильні та слабкі сторони конкурентних продуктів, що стало основою для уточнення вимог до вдосконаленого застосунку.

Відповідно до поставленого технічного завдання було обрано стек технологій, який відповідає сучасним стандартам продуктивності, гнучкості та безпеки. Інтеграція хмарних рішень Microsoft Azure і Cloudflare дозволила побудувати стабільну інфраструктуру, здатну адаптуватися до майбутніх потреб проєкту.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Eloundou T., Manning S., Mishkin P., Rock D. Gpts are gpts: An early look at the labor market impact potential of large language models. *arXiv preprint arXiv:2303.10130*. 2023. pp. 4–6.
2. Rawat P., Mahajan A. N. ReactJS: A modern web development framework. *International Journal of Innovative Science and Research Technology*. 2020. Vol. 5, № 11. pp. 698–702.
3. Castaño J., Silverio M.-F., Franch X., Bogner J. Analyzing the evolution and maintenance of ml models on hugging face. 2024. pp. 3–5.
4. Tamkin A., Brundage M., Clark J., Ganguli D. Understanding the capabilities, limitations, and societal impact of large language models. *arXiv preprint arXiv:2102.02503*. 2021. pp. 1–4.
5. Lock A. ASP. NET core in Action. Simon and Schuster, 2021. pp. 19–25.
6. Ilić M., Kopanja L., Zlatković D., Trajković M., Ćurguz D. Microsoft sql server and oracle: Comparative performance analysis. 2021. pp. 34–38.
7. Albahari J. C# 10 in a Nutshell. « O'Reilly Media, Inc.», 2022. pp. 1–73.
8. Bell D. UML basics: The sequence diagram. *Retrieved July*. 2004. Vol. 17. C. 2015. pp. 3–63.
9. Kurniawan T. A., Lam-Son L., Priyambadha B. Challenges in developing sequence diagrams (UML). *Journal of Information Technology and Computer Science*. 2020. Vol. 5, № 2. pp. 221–234.
10. Güvercin A. E., Avenoglu B. Performance Analysis of Object-Relational Mapping (ORM) Tools in. Net 6 Environment. *Bilişim Teknolojileri Dergisi*. 2022. Vol. 15, № 4. pp. 453–465.
11. Bhatt B., Nandu M. An Overview of Structural UML Diagrams. *International Research Journal of Engineering and Technlogy (IRJET)*. 2021. pp. 1577–1580.
12. Sergievskiy M., Kirpichnikova K. Optimizing UML class diagrams. 2018. pp. 1–5.

13. Ermel G., Farias K., Gonçales L. J., Bischoff V. Supporting the composition of UML component diagrams. 2018. pp. 442–444.
14. Koç H., Erdoğan A. M., Barjakly Y., Peker S. UML diagrams in software engineering research: a systematic literature review. 2021. pp. 1–8.
15. Nikolov N. Modern Technologies for Building Graphical User Interfaces On The Internet. *HR Technol.* 2023. Vol. 2. pp. 90–104.
16. Cvijić B., Ranilović P. From. NET Core to. NET 8: A Comprehensive Analysis of Performance, Features, and Migration Pathways. *JITA-APEIRON*. 2024. Vol. 27, № 1. pp. 69–77.
17. Singh J., Chaudhary N. K. OAuth 2.0: Architectural design augmentation for mitigation of common security vulnerabilities. *Journal of Information Security and Applications*. 2022. Vol. 65. C. 103091. pp. 1–5.
18. Gupta B., Mittal P., Mufti T. A review on amazon web service (aws), microsoft azure & google cloud platform (gcp) services. 2021. pp. 2–5.
19. Nadeem M., Arshad A., Riaz S., Zahra S. W., Rashid M., Band S. S., Mosavi A. Preventing Cloud Network from Spamming Attacks Using Cloudflare and KNN. *Computers, Materials & Continua*. 2023. Vol. 74, № 2. pp. 2642–2646.
20. Lukeš S. API for C# code generation. 2020. pp. 5–6.

ДОДАТОК А

Діаграма використання

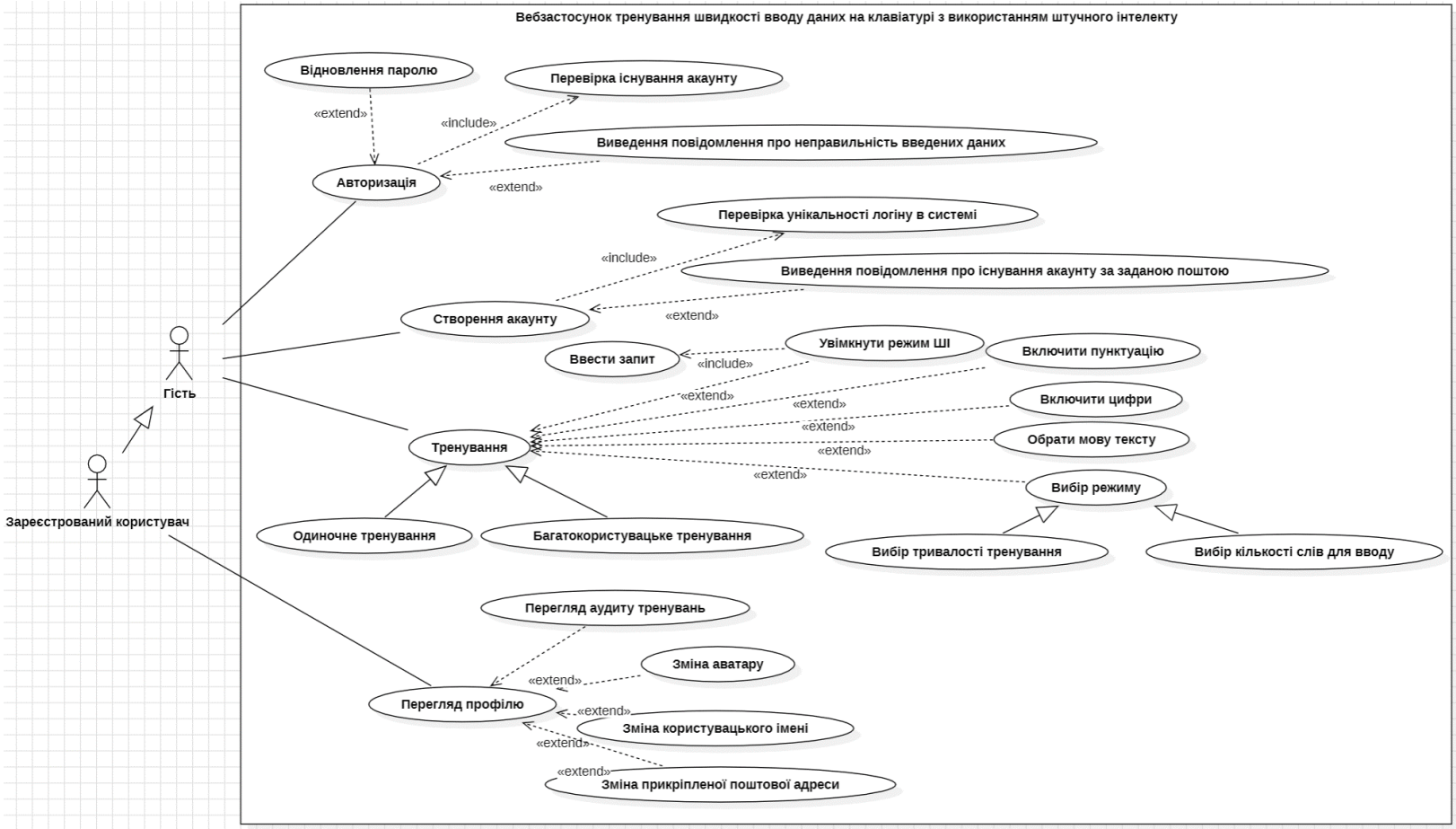

Рисунок А.1 – Діаграма використання

ДОДАТОК Б

Апробація кваліфікаційної роботи

Результати досліджень були представлені на двох конференціях:

– Могилянські читання – 2024, : Відстеження активності користувачів засобами логування в сучасних вебистемах : XXVII Всеукр. наук.-практ. конф. : тези доповідей : Комп’ютерні науки. Технічні науки, Миколаїв, 6-10 листоп. 2024 р. / ЧНУ ім. Петра Могили. – Миколаїв : Вид-во ЧНУ ім. Петра Могили, 2024.

Міністерство освіти і науки України
Чорноморський національний університет імені Петра Могили
ДНУ «Інститут модернізації змісту освіти»
Південний науковий центр НАН та МОН
Інститут української археографії та джерелознавства
імені М. С. Грушевського НАН України
Первинна профспілкова організація ЧНУ ім. Петра Могили

ЧНУ

**«МОГИЛЯНСЬКІ ЧИТАННЯ – 2024:
досвід та тенденції розвитку суспільства в Україні:
глобальний, національний та регіональний аспекти»**

XXVII Всеукраїнська науково-практична конференція

ТЕЗИ ДОПОВІДЕЙ

ТЕХНІЧНІ НАУКИ

Миколаїв, 6–10 листопада 2024 року

Миколаїв – 2024

Рисунок Б.1 – Обкладинка збірника тез конференції Могилянські читання – 2024