МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет імені Петра Могили
Факультет комп'ютерних наук
Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри, канд. техн. наук,
доцент _____ Є. О. Давиденко
«___»___2023 р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

ВЕБЗАСТОСУНОК ТРЕНУВАННЯ ШВИДКОСТІ ВВОДУ ДАНИХ НА КЛАВІАТУРІ

Спеціальність «Інженерія програмного забезпечення»

121 – КРБ.1 – 409.21910901

Студент _____ К. О. Бєктін
«___»___2023 р.

Керівник завідувач кафедри ІПЗ, канд. техн. наук, доцент _____ Є. О. Давиденко
«___»___2023 р.

Консультант канд. техн. наук, доцент _____ А. О. Алексєєва
«___»___2023 р.

Миколаїв 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет імені Петра Могили
Факультет комп'ютерних наук
Кафедра інженерії програмного забезпечення

ЗАТВЕРДЖУЮ

Зав. кафедри _____ Є. О. Давиденко

«_____» _____ 2023 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи бакалавра

Видано студенту групи 409 факультету комп'ютерних наук

_____Бєктіну Костянтину Олександровичу_____

(прізвище, ім'я, по батькові студента)

1. Тема кваліфікаційної роботи

Вебзастосунок тренування швидкості вводу даних на клавіатурі_____

_____

Затверджена наказом по ЧНУ від «17» _березня____ 2023 р. № ____60____

2. Строк представлення кваліфікаційної роботи «____» __червня_____ 2023 р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні

Очікуваним результатом є вебзастосунок тренування швидкості вводу даних на клавіатурі_____

4. Перелік питань, що підлягають розробці:

- проведення аналізу аналогічних систем 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.

6. Завдання до спеціальної частини

Аналіз охорони праці на робочих місцях фахівців з інформаційних технологій.

7. Консультанти:

Консультант	Кафедра (організація)	Частина роботи
Алексєєва А. О.	Кафедра екології	Спеціальна частина з охорони праці

Керівник роботи _канд. техн. наук, доцент Давиденко Євген Олександрович_

(посада, прізвище, ім'я, по батькові)

_____

(підпис)

Завдання прийнято до виконання

_____Бєктін Костянтин Олександрович_____

(прізвище, ім'я, по батькові студента)

_____

(підпис)

Дата видачі завдання « 20 » __березня_____ 2023 р.

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної роботи

Тема: Вебзастосунок тренування швидкості вводу даних на клавіатурі

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КРБ	20.03.2023 р.	21.03.2023 р.	виконано
2.	Огляд літератури за темою роботи	23.03.2023 р.	27.03.2023 р.	виконано
3.	Складання календарного плану КРБ	28.03.2023 р.	29.03.2023 р.	виконано
4.	Аналіз предметної області	01.04.2023 р.	03.04.2023 р.	виконано
5.	Розробка проєктних рішень	05.04.2023 р.	07.04.2023 р.	виконано
6.	Моделювання та конструювання ПЗ	10.04.2023 р.	14.04.2023 р.	виконано
7.	Кодування, тестування та апробація розробленого ПЗ, аналіз результатів тестування, розробка керівництва користувача	15.04.2023 р.	31.05.2023 р.	виконано
8.	Розробка спеціальної частини з охорони праці	08.05.2023 р.	15.05.2023 р.	виконано
9.	Оформлення КРБ та презентації	22.05.2023 р.	01.06.2023 р.	виконано
10.	Відгук керівника КРБ	05.06.2023 р.	05.06.2023 р.	виконано
11.	Попередній захист	06.06.2023 р.	07.06.2023 р.	виконано
12.	Завершення оформлення КРБ та презентації	08.06.2023 р.	12.06.2023 р.	виконано
13.	Рецензування	16.06.2023 р.	16.06.2023 р.	виконано
14.	Захист кваліфікаційної роботи	26.06.2023 р.	26.06.2023 р.	виконано

Розробив студент ____Бєктін Костянтин Олександрович_____ _____

(прізвище, ім'я, по батькові) *(підпис)*

«__» _____ 2023 р.

Керівник роботи __канд. техн. наук, доцент Давиденко Є. О.__ _____

(посада, прізвище, ім'я, по батькові) *(підпис)*

«__» _____ 2023 р.

АНОТАЦІЯ

до кваліфікаційної роботи бакалавра

«Вебзастосунок тренування швидкості вводу даних на клавіатурі»

Студент 409 гр.: Бєктін Костянтин Олександрович

Керівник: канд. техн. наук, доцент Давиденко Є. О.

Дана робота присвячена розробці програмного забезпечення спрямованого на покращення навичок користувачів з володіння клавіатурою, тобто, зменшення кількості помилок та збільшення швидкості вводу даних.

Об’єкт роботи: процес покращення швидкості вводу даних на клавіатурі.

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

Мета: автоматизація тренувань зі швидкості вводу даних на клавіатурі шляхом реалізації програмного забезпечення відповідного тренажеру.

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

У вступі описані актуальність обраної теми, об’єкт і предмет роботи та перелік основних задач, що потребують виконання для досягнення поставленої мети.

У першому розділі описується процес аналізу предметної області. Проводиться огляд існуючих аналогів, виконується порівняння їх функціоналу, переваг і недоліків. Також, на основі отриманих результатів надається специфікація вимог до розроблюваного програмного забезпечення.

У другому розділі описується процес моделювання системи згідно до попередньо визначених вимог шляхом побудови діаграм, що представляють різні аспекти роботи системи на даному етапі.

У третьому розділі описується проєктування системи, засноване на результатах отриманих у попередньому розділі. Сюди включено перелік обраних технологій для реалізації продукту, а саме: мови програмування,

фреймворки, провайдери баз даних тощо. Також, UML-діаграми, що представляють внутрішню будову застосунку, взаємодію між компонентами.

У четвертому розділі проводиться огляд результатів безпосереднього програмування та послідуючого тестування розроблюваного застосунку.

У висновках описується аналіз проведеної роботи та фінальних результатів.

КРБ викладена на 62 сторінки, вона містить 4 розділи, 41 ілюстрацію, 16 таблиць, 20 джерел в переліку посилань.

Ключові слова: *вебсистема, вебзастосунок, тренування швидкості вводу даних на клавіатурі, швидкість вводу даних, розробка на ASP.NET Core, розробка на React, платформа .NET.*

ABSTRACT

of the Bachelor's Thesis

«Web application for typing speed training on keyboard»

Student: Biektin Kostiantyn

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

Davydenko Y. O.

This work is devoted to the development of software aimed at improving the skills of users in using a keyboard, to be more preciese, reducing the number of errors and increasing the speed of entering data.

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

The subject of work: software tools for the implementation of the simulator for training data input on the keyboard.

Objective: automation of training of the speed of entering data on a keyboard by implementing the software of the corresponding simulator.

The qualification work consists of an introduction, four chapters, conclusions and a list of reference sources.

The introduction describes the relevance of the chosen topic, the object and subject of the work, and a list of the main tasks that need to be performed to achieve the goal.

The first chapter describes the process of domain analysis. A review of existing analogues is conducted, a comparison of their functionality, advantages and disadvantages is performed. Also, based on the obtained results, a specification of requirements for the developed software is provided.

The second section describes the process of modelling the system according to predefined requirements by constructing diagrams representing various aspects of the system's operation at this stage.

The third section describes the system design based on the results obtained in the previous section. A list of selected technologies for product implementation,

namely: programming languages, frameworks, database providers, etc. Also, UML diagrams representing the internal structure of the application, interaction between components.

The fourth chapter reviews the results of actual programming and subsequent testing of the developed application.

The conclusions describe the analysis of the work carried out and the final results.

The qualification work is presented on 62 pages, it contains 4 sections, 41 illustrations, 16 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
КРБ	–	кваліфікаційні робота бакалавра
UML	–	unified modeling language
SQL	–	structured query language
REST	–	representational state transfer
API	–	application programming interface
IDE	–	integrated development environment
JWT	–	JSON web token
HTTP	–	hypertext transfer protocol
JSON	–	javascript object notation
UI	–	user interface

ВСТУП

Актуальність теми кваліфікаційної роботи бакалавра обумовлена сучасними технологічними реаліями. Кожного дня люди стикаються зі задачами, виконання яких потребує використання комп’ютера для взаємодії з яким застосовуються пристрої для вводу даних. Серед такого роду пристроїв першість займає усім відома пара – клавіатура та миша. Багато хто недооцінює важливість ефективного користування саме клавіатурою [1]. Користувач без хороших навичок швидкого сліпого набору тексту приречений на постійне згадування розміщення потрібних клавіш. Замість того, аби зосередитись виключно на вирішенні поставленої задачі людина вимушена думати про абсолютно не пов’язані з нею речі.

В момент, коли користувач все ж таки вирішує зайнятись покращенням своїх умінь в даній сфері постає наступна проблема — потрібне джерело тексту для набору, адже вигадування чогось на ходу унеможливлює зосередження на головній проблемі. Саме для цього і використовується тренажер для набору тексту. На нього покладається відповідальність з його генерації. До того ж, існує можливість налаштовувати процес генерації шляхом внесення, наприклад, пунктуації, цифр тощо. Існує, також, можливість позмагатись з іншими користувачами у багатокористувацькому режимі та перевірити успіхи інших на загальній дошці лідерів. Фактор змагання за першість виступає додатковим стимулом для покращення своїх результатів.

Об’єкт роботи: процес вводу даних на клавіатурі.

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

Мета: автоматизація тренувань швидкості вводу даних на клавіатурі шляхом реалізації програмного забезпечення відповідного тренажеру.

Досягнення визначеної мети передбачає вирішення наступних **завдань**:

1. аналіз аналогічних застосунків;

2. виокремлення існуючих проблем та формування шляхів їх вирішення;

3. визначення майбутнього функціоналу розроблюваної системи;

4. проєктування алгоритмів процесів застосунку за допомогою побудови блок-схем та UML-діаграм;

5. створення макету користувацького інтерфейсу;

6. побудова елементів користувацького інтерфейсу за допомогою фреймворку React;

7. створення серверної частини на базі фреймворку ASP.NET Core та провайдера баз даних SQL Server;

8. підключення клієнтської частини до серверної.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз прикладів аналогічних систем

На початку розробки будь якого продукту вкрай важливо приділити увагу конкурентам на ринку, місце на якому ви плануєте зайняти. На основі проведеного аналізу, в перспективі, можна буде виділити основні недоліки існуючих пропозицій, для того щоб в майбутньому використати ці знання для покращення свого продукту, спрямувати зусилля на привнесення нововведень, небачених у аналогів. Таким чином, залучення нових користувачів буде лише питанням часу. В якості прямих конкурентів було виділено наступні застосунки:

1. Type Racer (див. табл. 1.1 та рис. 1.1);
2. Monkeytype (див. табл. 1.2 та рис. 1.2);
3. TypingTest (див. табл. 1.3 та рис. 1.3).

Type Racer

Таблиця 1.1 – Аналіз системи Type Racer

Виробник	Addicting Games, Inc.
Архітектура	3-tier application [2, 4]
Мова реалізації	JavaScript
Основні функції	1. проведення одиночних тренувань набору тексту; 2. проведення багатокористувацьких тренувань набору тексту в режимі суперництва; 3. перегляд таблиці лідерів за різними критеріями; 4. вибір різних варіантів тематики текстів, мов; 5. налаштування користувацького профілю (фото, аватар для відображення під час багатокористувацького тренування, персональні дані тощо).

Кінець таблиці 1.1

Переваги	1. гарна підтримка сайту; 2. регулярні оновлення; 3. проведення різного роду подій для користувачів; 4. наявність світлої та темної тем.
Недоліки	1. нелаконічний дизайн, місцями доволі грубий; 2. часткове привнесення нерелевантних нововведень; 3. відсутність налаштування інтерфейсу для тренувань.
Вебсайт	https://play.typeracer.com/

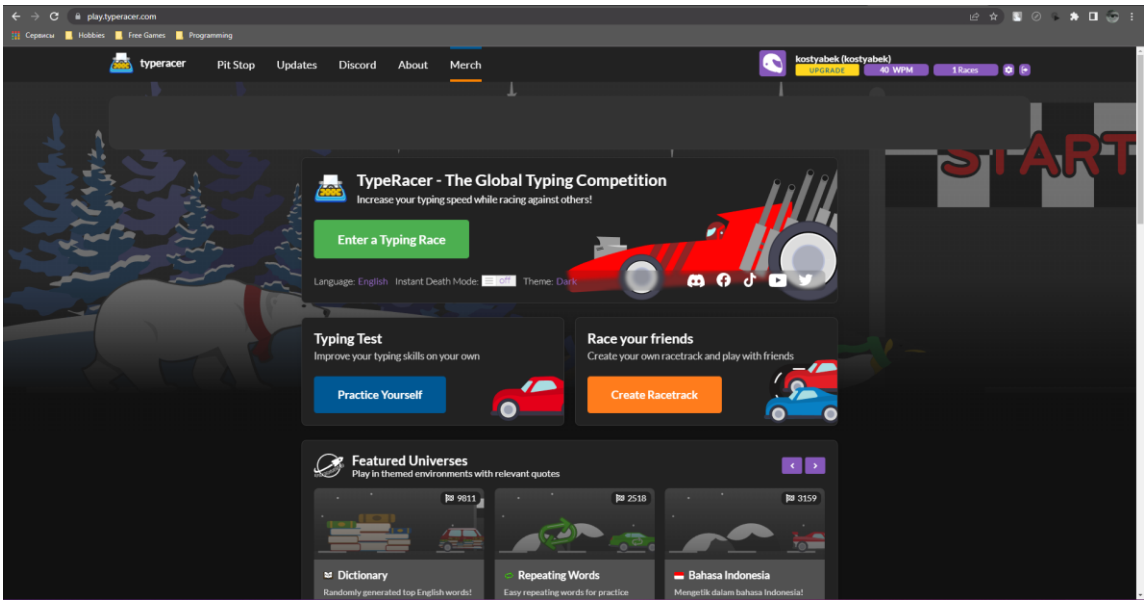

Рисунок 1.1 – Вигляд інтерфейсу застосунку Type Racer

Monkeytype

Таблиця 1.2 – Аналіз системи Monkeytype

Виробник	Miodec
Архітектура	3-tier application
Мова реалізації	TypeScript [3]
Основні функції	1. тренування набору тексту; 2. вибір різних варіантів тексту для набору (людські мови за різними типами складності слів, мови програмування тощо);

Кінець таблиці 1.2

Основні функції	3. вибір модифікаторів для набору тексту (пунктуація, цифри); 4. вибір модифікаторів для набору тексту (пунктуація, цифри); 5. відображення детальної статистики (з графіками) успішності проходження тренувань; 6. дошка лідерів у різного роду режимах; налаштування загального вигляду інтерфейсу.
Переваги	1. гарна підтримка сайту; 2. регулярні оновлення; 3. open source; 4. наявність детального налаштування різних аспектів, від різноманітних кольорових тем до інформації та її стилю під час введення даних, що дозволяє адаптувати процес тренувань для отримання найбільш високих результатів.
Недоліки	1. відсутність багатокористувацького режиму; 2. відсутність подій та взаємодії між користувачами; 3. обмеженість деяких режимів.
Вебсайт	https://monkeytype.com/

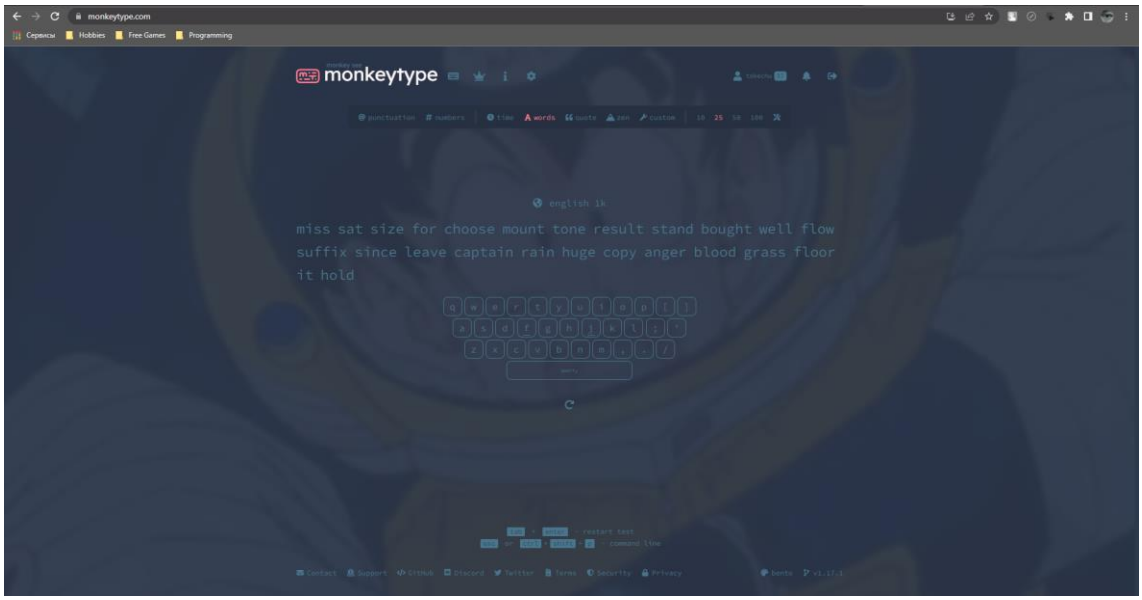

Рисунок 1.2 – Вигляд інтерфейсу застосунку Monkeytype

TypingTest

Таблиця 1.3 – Аналіз системи TypingTest

Виробник	Cloud Kayak Labs, Inc
Архітектура	3-tier application
Мова реалізації	JavaScript
Основні функції	1. тренування швидкості вводу даних за допомогою клавіатури; 2. надання інтерактивних курсів для покращення навичок набору; 3. можливість пограти у інтерактивні ігри, для досягнення перемоги в яких задіяний набір тексту; 4. тренування набору слів із літерами, з якими наявна найбільша кількість помилок під час вводу.
Переваги	1. містить велику кількість підлеглих ресурсів (наприклад курси щодо підвищення швидкості набору у ігровій формі, тренування для школярів тощо); 2. можливість отримання сертифікату, що підтверджує ваші здібності; 3. простий у користуванні.
Недоліки	1. відсутність багатокористувацького режиму; 2. старий дизайн; 3. відсутність перспективи оновлень ресурсу.
Вебсайт	https://www.typingtest.com/

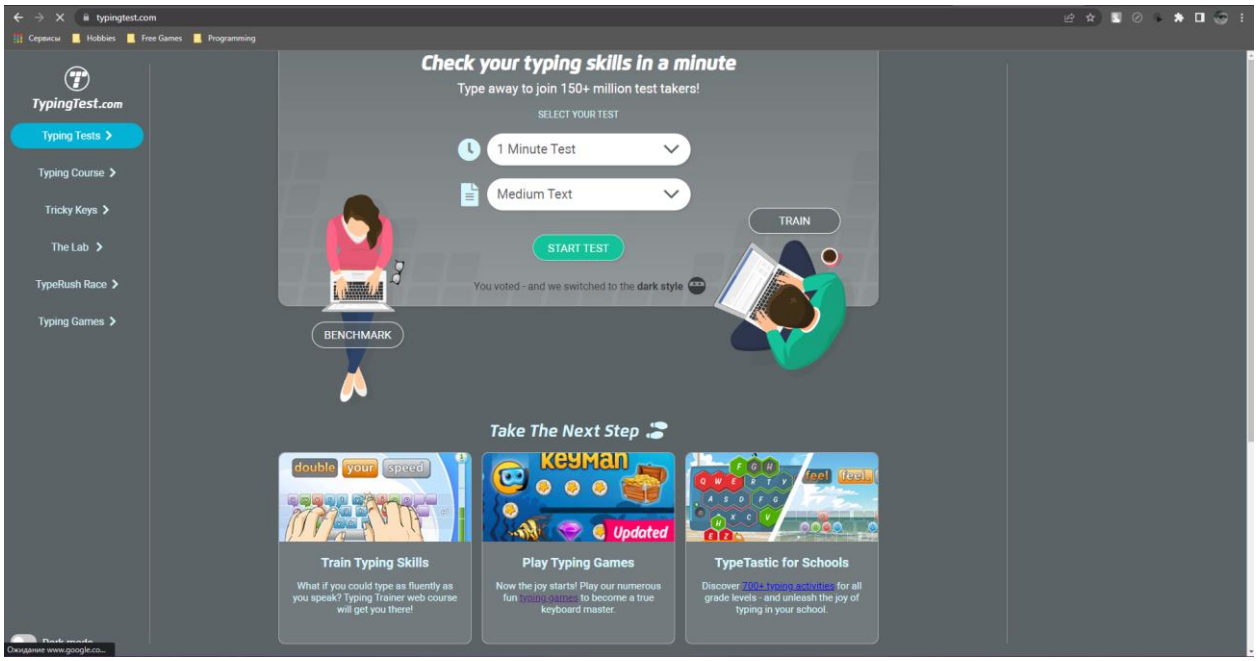

Рисунок 1.3 – Вигляд інтерфейсу застосунку TypingTest

Як можна бачити на рисунку 1.3, вебсайт дійсно містить велику кількість розділів, спрямованих на ігрову модель взаємодії.

1.2 Аналіз системи, що розробляється

Вебзастосунок тренування швидкості вводу даних на клавіатурі «typro» розробляється для покращення користувачами своїх вмінь з введення тексту. За допомогою регулярних тренувань можна суттєво полегшити виконання задач на комп’ютері, що потребують тривалого використання клавіатури. Після проходження декількох тренувань користувач має змогу переглянути свої успіхи у форматі графіків та текстових даних. Також, декілька користувачів можуть проходити тренування з набору одного й того самого тексту в багатокористувацькому режимі.

Таблиця 1.4 – Опис системи, що розробляється

Основні задачі	1. введення даних на клавіатурі з одночасним відображенням продуктивності вводу, помилок, автоматичним генеруванням тексту, за потреби;

Кінець таблиці 1.4

Основні задачі	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); 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; 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.
Засоби апаратної та програмної реалізації	1. ASP.NET Core [5, 6] 2. React [7] 3. SQL Server
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.

Рисунок 1.4 – Макет інтерфейсу застосунку «typro»

Mock-up було створено за допомогою програмного забезпечення «Figma» [8]. Кольорова гама була підібрана згідно зі стандартами сумісності кольорів [9]. Загалом користувачу надаються наступні розділи застосунку:

1. головна сторінка (вона відповідає за одиночні тренування);
2. профіль (інформація про користувача, налаштування пов’язані зі нею);
3. налаштування (налаштування застосунку);
4. багатокористувацький режим (проходження тренувань у групі);
5. дошка лідерів (таблиця де відображаються кращі результати зареєстрованих користувачів у різних режимах);
6. про застосунок (короткий опис застосунку).

1.3 Специфікація вимог до розроблюваної системи

Призначення системи, для якої розробляється програмне забезпечення

Призначенням застосунку є автоматизація тренування швидкості вводу даних на клавіатурі.

Погодження, ухвалені в програмній документації

Було узгоджено, що для створення програмного забезпечення будуть використовуватись фреймворки React та ASP.NET Core.

Межі проєкту ПЗ

Крайня дата завершення робіт над проєктом – 12.06.2023р.

Сфера застосування

Дане ПЗ не має чітких обмежень щодо сфери застосування. У нашому технологічному сьогоденні комп’ютери використовуються майже у всіх професійних напрямках. До того ж, більшість має персональні комп’ютери, якими користуються вдома.

Характеристики користувачів

Користувачу необхідні стабільний доступ до мережі Інтернет, клавіатура (екранна або фізична) та гаджет, на кшталт смартфону, ноутбуку чи персонального комп’ютеру з монітором.

Загальна структура та склад системи

Система включає у себе наступні компоненти:

1. клієнт (застосунок на базі фреймворку React);
2. сервер (застосунок на базі фреймворку ASP.NET Core);
3. сховище даних (база даних SQL Server) [10].

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 перегляду користувацької статистики пройдених тренувань

Опис функції

Дана функція дозволяє зареєстрованим користувачам переглядати детальну статистику щодо пройдених тренувань, що, в свою чергу, допомагає краще відслідковувати свій прогрес.

Вхідна та вихідна інформація

Вхідна інформація – ідентифікатор користувача, дані про пройдені ним тренування.

Вихідна інформація – текстові та графічні дані щодо успішності користувача.

Функціональні вимоги

Попередньо, користувач має бути зареєстрованим у системі, для того, аби інформація про пройдені тренування зберігалась у базі даних. Також, необхідний стабільний доступ до мережі Інтернет.

Джерела та зміст вхідної інформації (даних)

У даному програмному забезпеченні головним джерелом інформації є користувач. Він своїми діями генерує нову інформацію для збереження у базі даних (наприклад, після успішної реєстрації облікового запису, пройденого тренування тощо).

Нормативно-довідкова інформація (класифікатори, довідники тощо)

Вимоги до даного пункту відсутні.

Вимоги до технічного забезпечення

Вимоги до технічного забезпечення не мають серйозних обмежень у сучасних реаліях. Гаджет, з якого буде виконуватись робота повинен мати доступ до мережі Інтернет, мати екран, підтримку фізичної або екранної клавіатури та мати підтримку роботи зі сучасними браузерами, на кшталт, Google Chrome, Mozilla Firefox, Opera тощо.

Вимоги до програмного забезпечення

Архітектура програмної системи

До складу системи входять клієнтський застосунок, REST [11] API сервер та база даних.

Системне програмне забезпечення

Клієнтський застосунок має бути побудований за допомогою фреймворку React, REST API сервер за допомогою фреймворку ASP.NET Core та провайдером баз даних буде Microsoft SQL Server.

Мережне програмне забезпечення

Для створення програмного забезпечення обрано операційну систему Windows 11, редактор коду Visual Studio Code, IDE JetBrains Rider та браузер Google Chrome.

Програмне забезпечення ведення інформаційної бази

В якості сховища даних обрано Microsoft SQL Server.

Мова і технологія розробки ПЗ

Програмне забезпечення буде розроблюватися за допомогою фреймворків React і ASP.NET Core та мов програмування TypeScript та C# [12].

Вимоги до зовнішніх інтерфейсів

Інтерфейс користувача

Інтерфейс користувача має бути побудований за усіма стандартами для найкращого досвіду користування. Головний каркас має бути поділений на навігаційну панель, контейнер для відображення головного контенту та нижню панель. Таким чином, під час роботи застосунку, змінним є тільки контейнер для головного контенту.

Апаратний інтерфейс

Апаратним інтерфейсом буде гаджет використовуваний користувачем для роботи зі клієнтським застосунком.

Програмний інтерфейс

Обидва фреймворки React та ASP.NET Core мають чудове оточення для розробки. Таким чином усі помилки та елементи для покращення одразу відсилаються розробнику для перегляду для внесення необхідних змін.

Властивості програмного забезпечення

Доступність

Поріг початку користування системою доволі низький. Ввід користувача не потрібні жодні додаткові дії.

Безпека

Аутентифікація користувачів буде відбуватись за допомогою поєднання JWT токену доступу та токену оновлення. Таким чином унеможливлюється підміна даних користувача зловмисниками та можливість швидкого реагування на підозрілі дії з використанням токену оновлення.

Продуктивність

Об’єми даних, що пересилаються під час роботи системи між клієнтським застосунком та REST API малі, тож навіть із низькою швидкістю передачі даних не має виникнути серйозних затримок.

Простота використання

Користувацький інтерфейс виконаний у мінімалістичному стилі, не перевантажений елементами взаємодії. Таким чином, користувач буквально після перших кількох хвилин зможе майже повністю оволодіти доступним функціоналом.

Функціональність

Будучи доволі мінімалістичним застосунок пропонує широкий спектр функціоналу, який спрямований на повноцінне задоволення потреб з покращення швидкості вводу даних з клавіатури кінцевого користувача.

Висновки до розділу 1

В першому розділі кваліфікаційної роботи бакалавра було проведено детальний аналіз існуючих конкурентів на даному ринку. Таким чином, виокремлено основні недоліки та отримано шляхи до їх вирішення. Крім того, було проаналізовано систему, що розробляється. Сформовано список основного функціоналу системи, користувачів, сценаріїв роботи, технологій для реалізації тощо. Надано специфікацію вимог до розроблюваної системи з повним описом поведінки, функціональних та нефункціональних вимог, обмежень тощо.

2 МОДЕЛЮВАННЯ СИСТЕМИ

2.1 Створення варіантів використання системи

Короткий use case

Користувач має бажання відточити свої навички зі швидкості вводу даних за допомогою клавіатури. Він заходить на сайт, проходить етап аутентифікації та авторизації для збереження даних про його тренування, виставляє потрібні налаштування, заходить в багатокористувацький режим та змагається з іншими користувачами на швидкість. Після проходження тренування він переглядає свою статистику у відповідному розділі вебсайту.

Поверхневий use case

Головний сценарій (успішний):

Користувач має бажання відточити свої навички зі швидкості вводу даних за допомогою клавіатури. Він заходить на сайт, проходить етап аутентифікації та авторизації, виставляє потрібні налаштування, заходить в багатокористувацький режим та змагається з іншими користувачами на швидкість.

Альтернативні сценарії:

1. Користувач не зареєстрований у системі. Перегляду результатів тренування для нього недоступний.

2. Пристрій з якого користувач зайшов на вебсайт від’єднався від мережі Інтернет під час тренування більш ніж на 30 секунд. Це тренування в статистиці матиме статус «не завершене».

3. Користувач забув пароль від облікового запису.

4. Вебсервер на якому відбувається хостинг застосунку має технічні негаразди. Вебсайт недоступний.

5. Користувач закриває вкладку з вебсайтом під час тренування. Це тренування в статистиці матиме статус «не завершене».

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	Наявність стабільного підключення до інтернету
Stakeholders and interests	1. Користувач: зацікавлений у найбільш зручному та продуктивному тренуванні навичок вводу даних на клавіатурі. 2. Адміністратор: зацікавлений у підтримці злагодженої роботи застосунку, наданні повідомлень користувачам про нововведення.
Main Success Scenario	1. Користувач заходить на вебсайт з метою початку тренування швидкості набору даних на клавіатурі. 2. Користувач проходить процес реєстрації. 3. Користувач проходить процес аутентифікації та авторизації. 4. Користувач переходить у багатокористувацький режим. 5. Користувач обирає потрібні налаштування для тренування. 6. Користувач починає пошук суперників. 7. Користувач успішно завершує сесію тренування. 8. Користувач переходить в розділ статистики для аналізу своєї продуктивності.
Result	користувач відточив свої навички з вводу даних на клавіатурі, проаналізував свою продуктивність за останній час.

Продовження таблиці 2.1

Extensions	1. Вебсервер на якому відбувається хостинг застосунку недоступний. 2. Користувач не був авторизований на момент проходження тренування: 2.1. Користувач реєструє новий обліковий запис, або авторизується за допомогою існуючого. 2.2. Користувач переходить у багатокористувацький режим. 2.3. Користувач виставляє необхідні налаштування для тренування. 2.4. Користувач проходить тренування з іншими користувачами. 2.5. Користувач успішно завершує сесію тренування. 2.6. Користувач переглядає результати тренування у розділі статистики. 3. Користувач забув пароль від облікового запису: 3.1. Користувач натискає кнопку з написом «Forgot password?». 3.2. Користувачу надсилається лист на пошту з посиланням для відновлення паролю. 3.3. Користувач переходить за посиланням з листа. 3.4. Користувач двічі вводить новий пароль та натискає кнопку з написом «Change password». 3.5. Користувач переходить у багатокористувацький режим. 3.6. Користувач виставляє необхідні налаштування для тренування. 3.7. Користувач проходить тренування з іншими користувачами. 3.8. Користувач успішно завершує сесію тренування. 3.9. Користувач переглядає результати тренування у розділі статистики. 4. Користувач закриває вкладку з вебсайтом під час тренування: 4.1. Поновлення сесії неможливе. Дане тренування зберігається зі статусом «Not completed».

Кінець таблиці 2.1

Extensions	5. Пристрій з якого користувач зайшов на вебсайт від'єднався від мережі Інтернет під час тренування більш ніж на 30 секунд: 5.1. Це тренування в статистиці матиме статус «не завершене».
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 надають змогу описати загальний функціонал системи разом із областю її застосування. До того ж, вони наочно демонструють зв'язки між акторами та розроблюваною системою, визначаючи загальний контекст, на ряду зі вимогами. На рисунку 1 у додатку А представлено діаграму використання для вебзастосунку тренування швидкості вводу даних на клавіатурі.

2.2 Побудова діаграм взаємодії

Unified Modeling Language [13] (UML) – це набір правил та визначень для специфікації системи програмного забезпечення, що керується Object Management Group. Дана система визначень надає набір графічних елементів для моделювання частин системи. В наступних підрозділах даного розділу використовуються UML-діаграми для моделювання розроблюваної системи.

В процесі проведення аналізу розроблюваної системи необхідно продумувати варіанти взаємодії користувачів із застосунком. Для моделювання цієї поведінки використовуються діаграми взаємодії (послідовності) [14]. Так, наприклад, на діаграмі №1 (рис. 2.1) демонструє взаємодію користувача зі застосунком при проходженні аутентифікації.

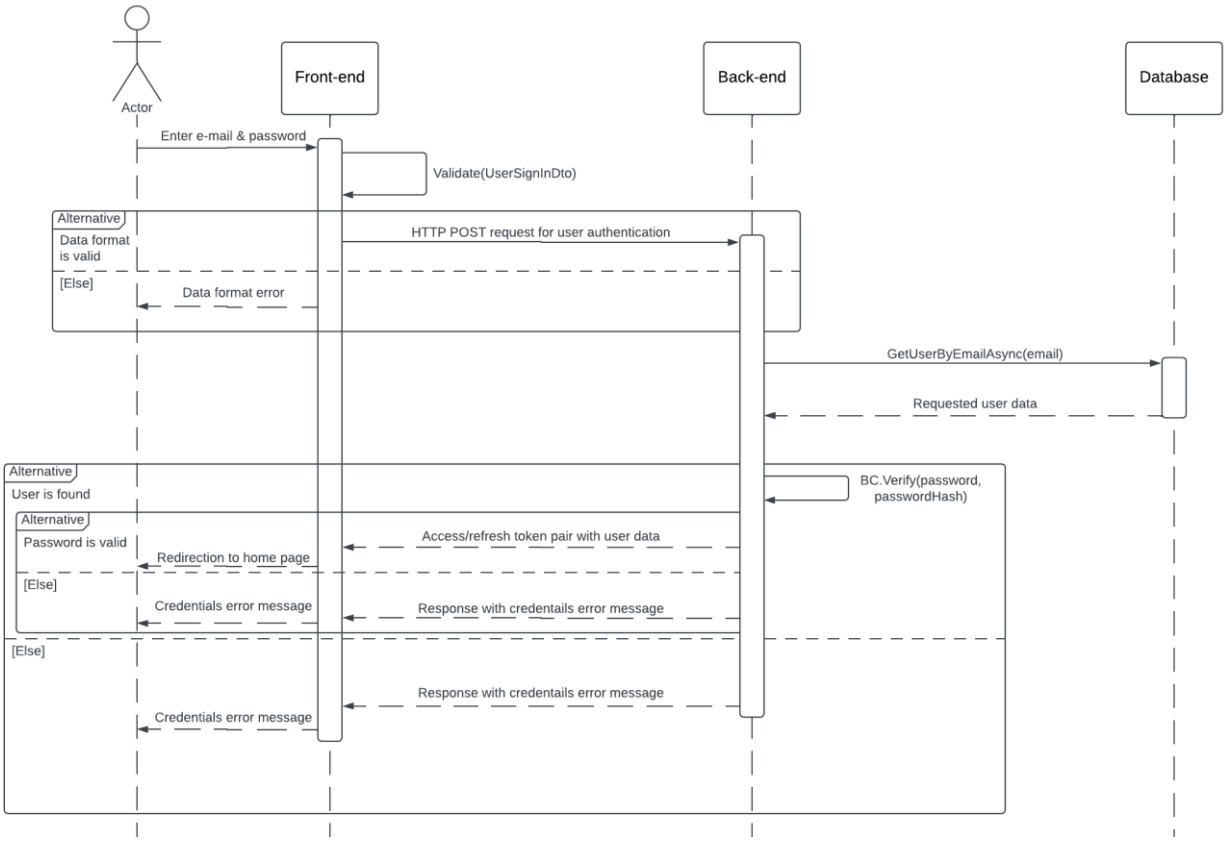

Рисунок 2.1 – Діаграма взаємодії аутентифікації користувача

Демонстрація взаємодії користувача зі застосунком для отримання даних про профіль показана на діаграмі №2 (рис. 2.2)

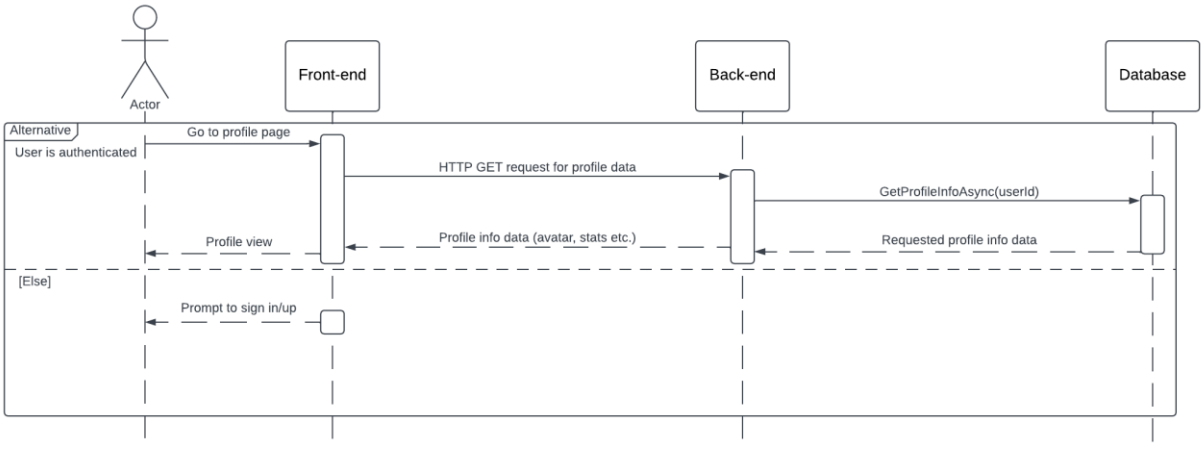

Рисунок 2.2 – Діаграма взаємодії показу даних профілю

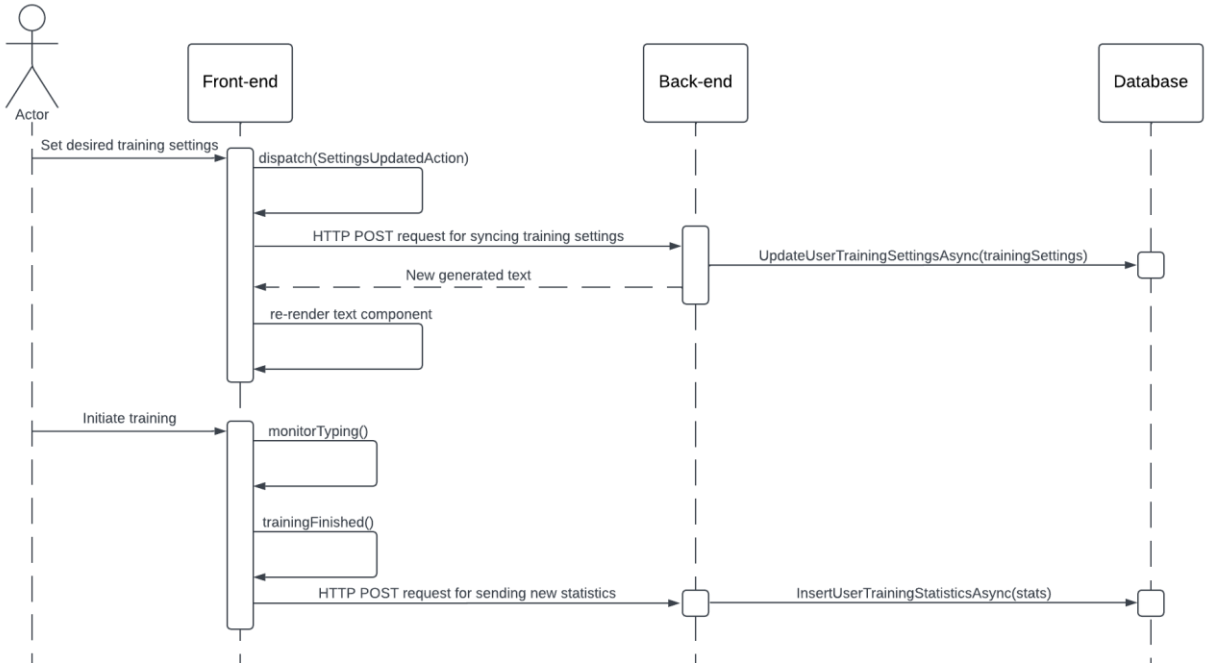

Рисунок 2.3 – Діаграма взаємодії проходження тренування в режимі для одного користувача з обраною кількістю слів

Діаграма №3 (рис. 2.3) передбачає собою демонстрацію одного з варіантів проходження тренування, а саме у режимі для одного користувача та з обраною кількістю слів.

2.3 Побудова діаграм діяльності

Діаграми діяльності схожі за своєю семантикою до діаграм станів та переходів, але несуть більш загальний характер. По суті, вони є аналогами блок-схем, що описують ті чи інші можливі процеси всередині розроблюваної системи. Зображуються у вигляді орієнтовних графів, де вершини та ребра – це дії та переходи між діями відповідно.

Для вебзастосунку тренування швидкості вводу даних на клавіатурі було розроблено 3 діаграми діяльності:

1. для процесу обробки користувацького вводу під час тренування (рис. 2.4 – 2.6). Повний варіант діаграми зображено на рисунку 2 у додатку А;

2. для процесу оновлення псевдоніму користувача (рис. 2.7);

3. для процесу обробки результатів (рис. 2.8).

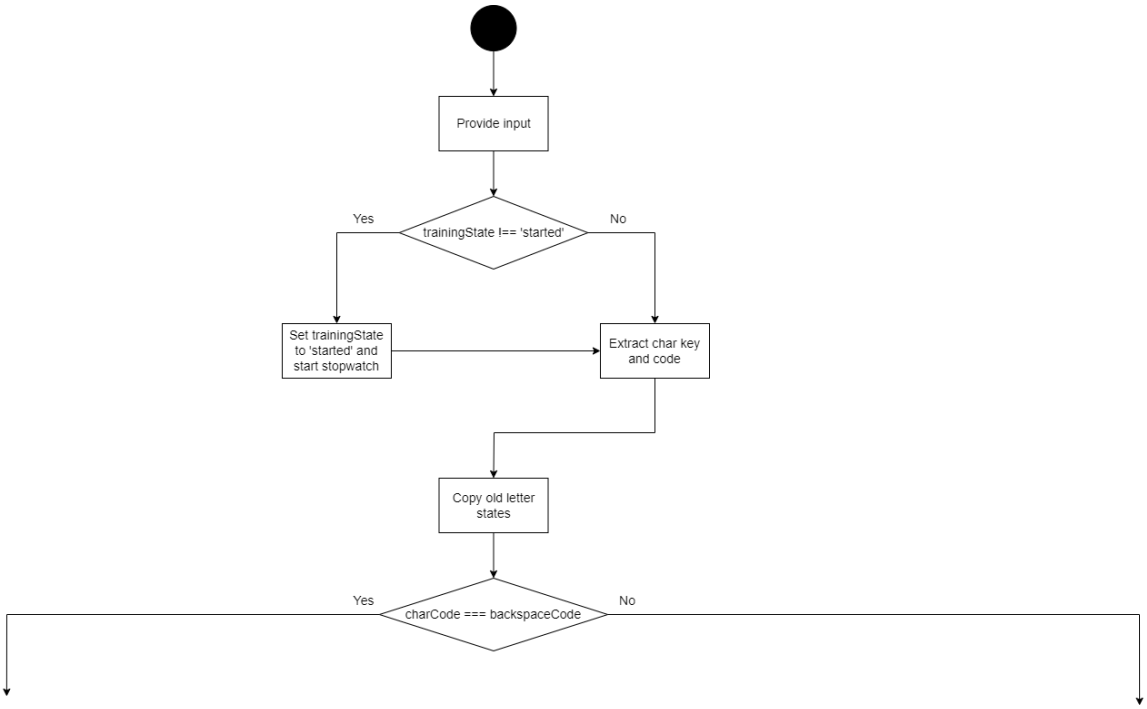

Рисунок 2.4 – Початковий етап обробки користувацького вводу

Рисунок 2.5 – Стезя обробки за умови, що було натиснуто клавішу «backspace»

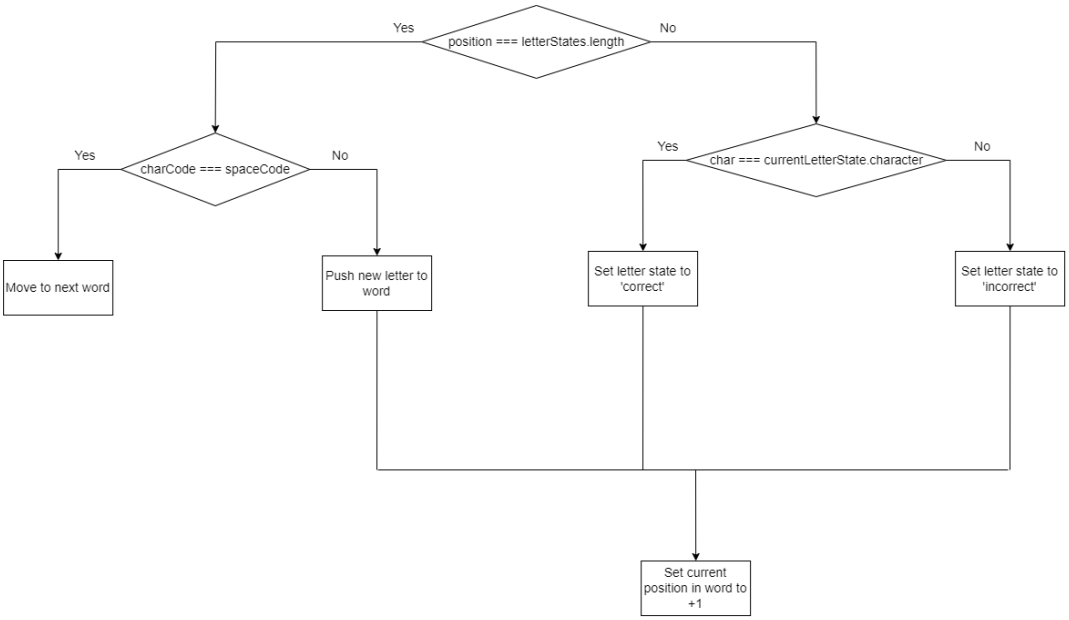

Рисунок 2.6 – Шлях обробки за умови, що було натиснуто клавішу відмінну від «backspace»

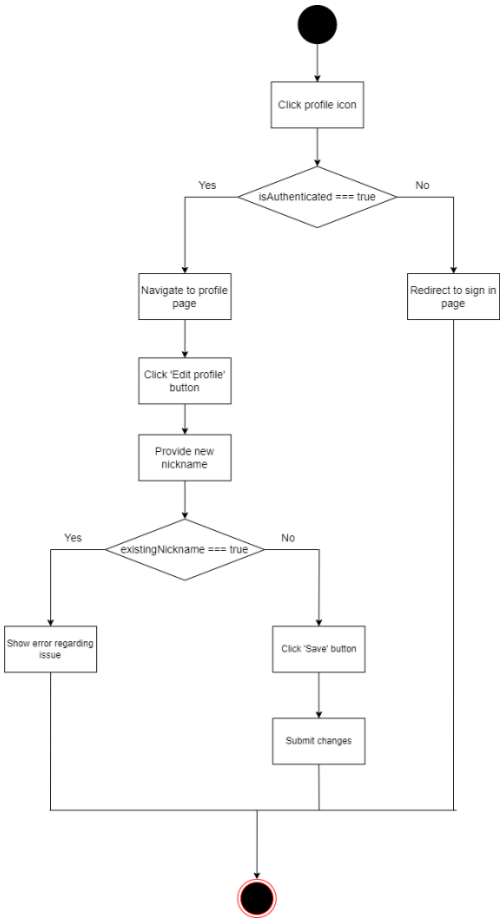

Рисунок 2.7 – Діаграма діяльності для процесу оновлення псевдоніму користувача

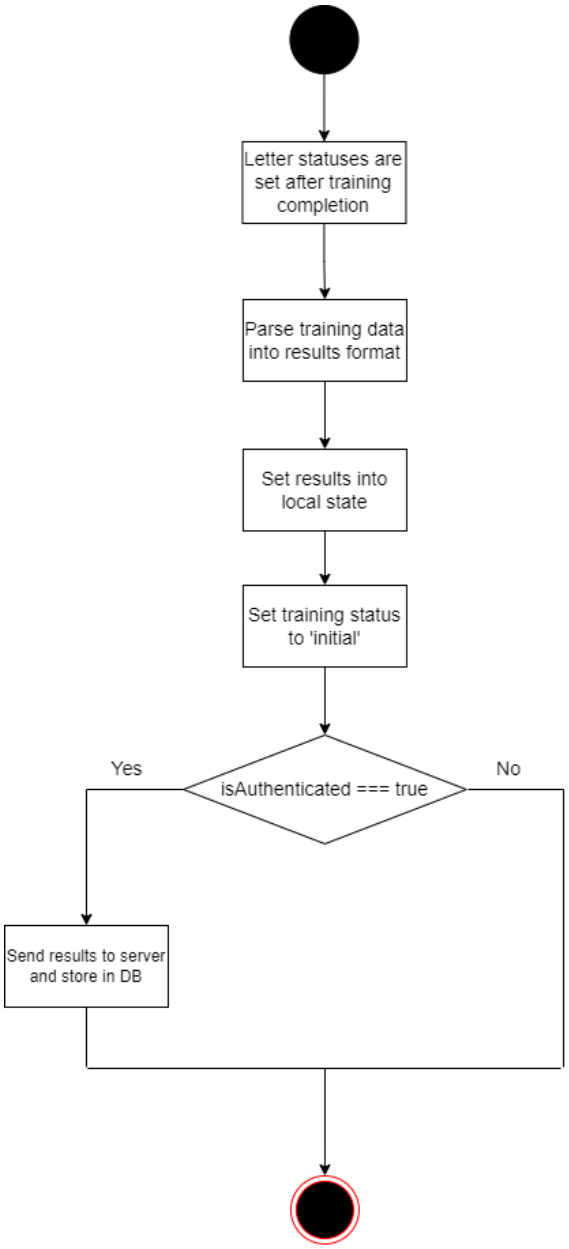

Рисунок 2.8 – Діаграма діяльності для процесу обробки результатів

Варто зазначити, що діаграми даного виду також використовуються під час моделювання бізнес-процесів для відображення послідовності дій. Для цього використовується додатковий елемент, що має назву swimlane (в перекладі з англійської – доріжка басейну). Кожна з таких «доріжок» представляє з себе акторів (осіб, відділів, організацій тощо).

2.4 Побудова діаграми розгортання

Діаграма розгортання надає змогу графічно представити розроблюване програмне забезпечення. Її зображено на рисунку 2.9. Це UML діаграма, що відображає обчислювальні вузли, задіяні під час роботи системи. Також, об’єкти та компоненти.

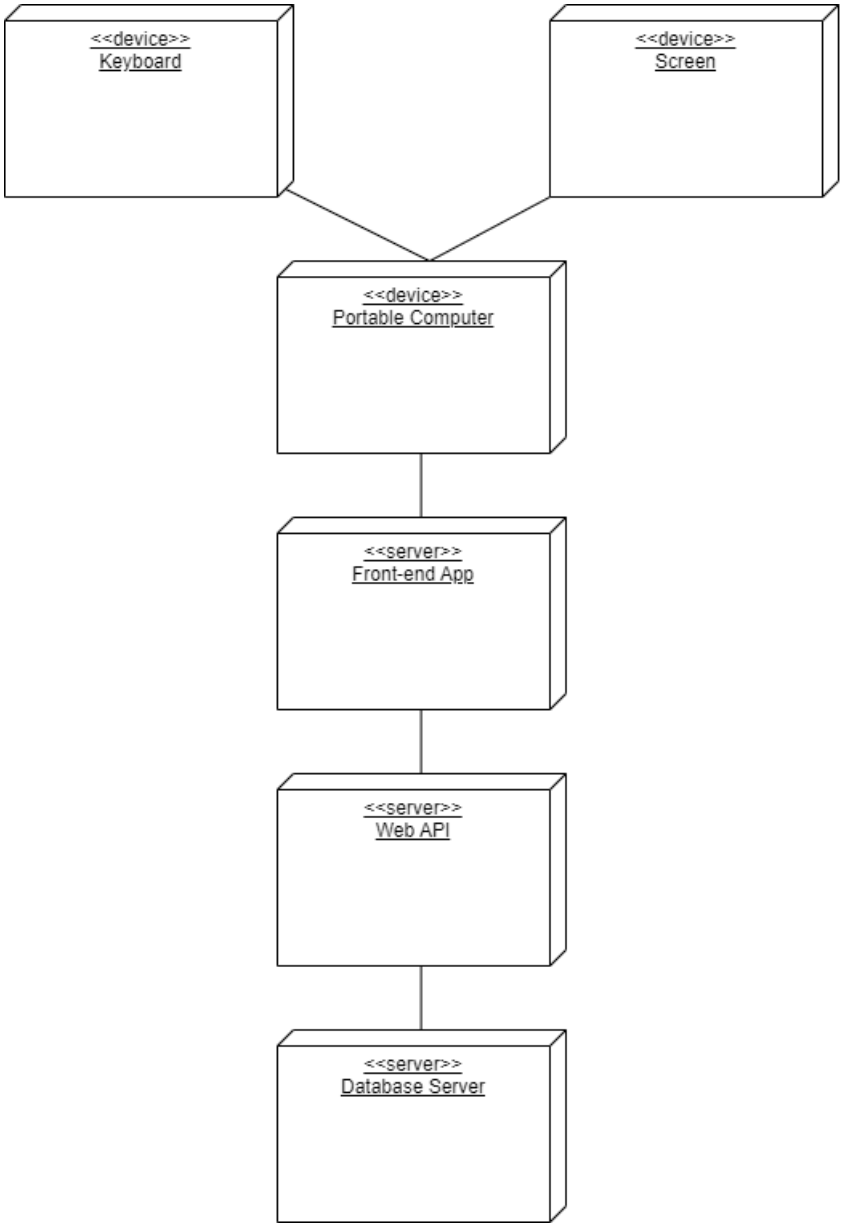

Рисунок 2.9 – Діаграма розгортання вебзастосунку тренування швидкості набору даних на клавіатурі

На діаграмі розгортання вебзастосунку тренування швидкості набору даних на клавіатурі вузли (nodes) зображені прямокутними паралелепіпедами. Вони представляють наступне:

1. клавіатуру (keyboard);
2. екран (screen);
3. комп’ютер (portable computer);
4. клієнтський застосунок (front-end app);
5. API-сервер (web API);
6. сервер бази даних (database server).

Загально вузли поділяються на дві категорії:

1. обчислювальні;
2. програмні.

Обчислювальні вузли представляють пристрої, що наділені власними фізичними ресурсами (процесор, пам’ять тощо) для обчислення та сервісами для виконання ПЗ. Програмні ж, у свою чергу, працюють всередині зовнішніх вузлів. Вони являють собою сервіси, що застосовують інші програмні елементи.

Висновки до розділу 2

Другий розділ кваліфікаційної роботи бакалавра присвячений моделюванню вебзастосунку тренування швидкості вводу даних на клавіатурі. За допомогою попередньо розробленої діаграми використання представлено основні сценарії роботи системи, спрямовані на задоволення ключових потреб користувача. Побудовано діаграми діяльності для висвітлення алгоритмів функціонування застосунку. Також, за допомогою діаграм взаємодії описано принципи спілкування між клієнтом та сервером під час виконання тривіальних задач.

3 ПРОЄКТУВАННЯ СИСТЕМИ

3.1 Побудова діаграм класів

Діаграма класів [15] виступає фундаментом для подальшої розробки системи засобами об’єктно-орієнтованого програмування. Її використовують для проєктування структури майбутньої програми з подальшою реалізацією результатів у вигляді коду. Кожен зображений клас містить ряд параметрів у вигляді методів, що описують його поведінку та полів – його стан. Важливим аспектом на даному етапі є їх найменування. Воно має чітко відображати сутність того чи іншого компоненту або члену.

Діаграми класів деяких частин застосунку представлено на рисунках 3.1 та 3.2. Також, у таблицях 3.1 та 3.2 можна знайти пояснення до них. Варто зазначити, що клас «UnitOfWork» використовується в обох діаграмах, тож повторно на рисунку 3.1 та у таблиці 3.2 його не було зазначено.

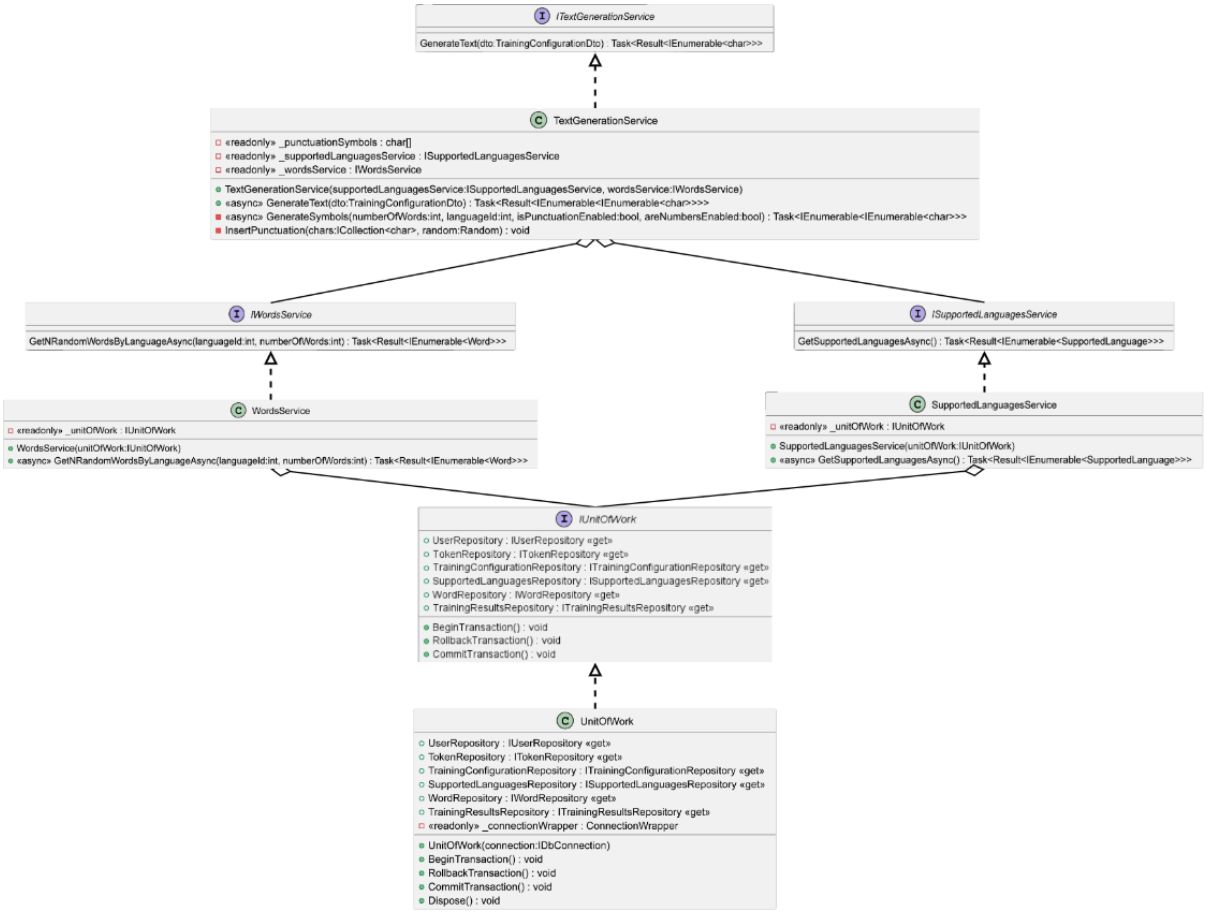

Рисунок 3.1 – Діаграма класів задіяних у генерації тексту

Таблиця 3.1 – Характеристика класів задіяних у генерації тексту

Назва класу	**Поля**	**Методи**	**Короткий опис**
UnitOfWork	UserRepository TokenRepository TrainingConfigurationRepository SupportedLanguagesRepository WordRepository TrainingResultsRepository _connectionWrapper	BeginTransaction() RollbackTransaction() CommitTransaction() Dispose()	Взаємодія з базою даних
WordsService	_unitOfWork	GetNRandomWordsByLanguageAsync()	Отримання набору випадкових слів з бази даних
SupportedLanguagesService	_unitOfWork	GetSupportedLanguagesAsync()	Отримання списку підтримуваних мов
TextGenerationService	_punctuationSymbols _supportedLanguagesService _wordsService	GenerateText() GenerateSymbols() InsertPunctuation()	Генерація тексту для тренувань

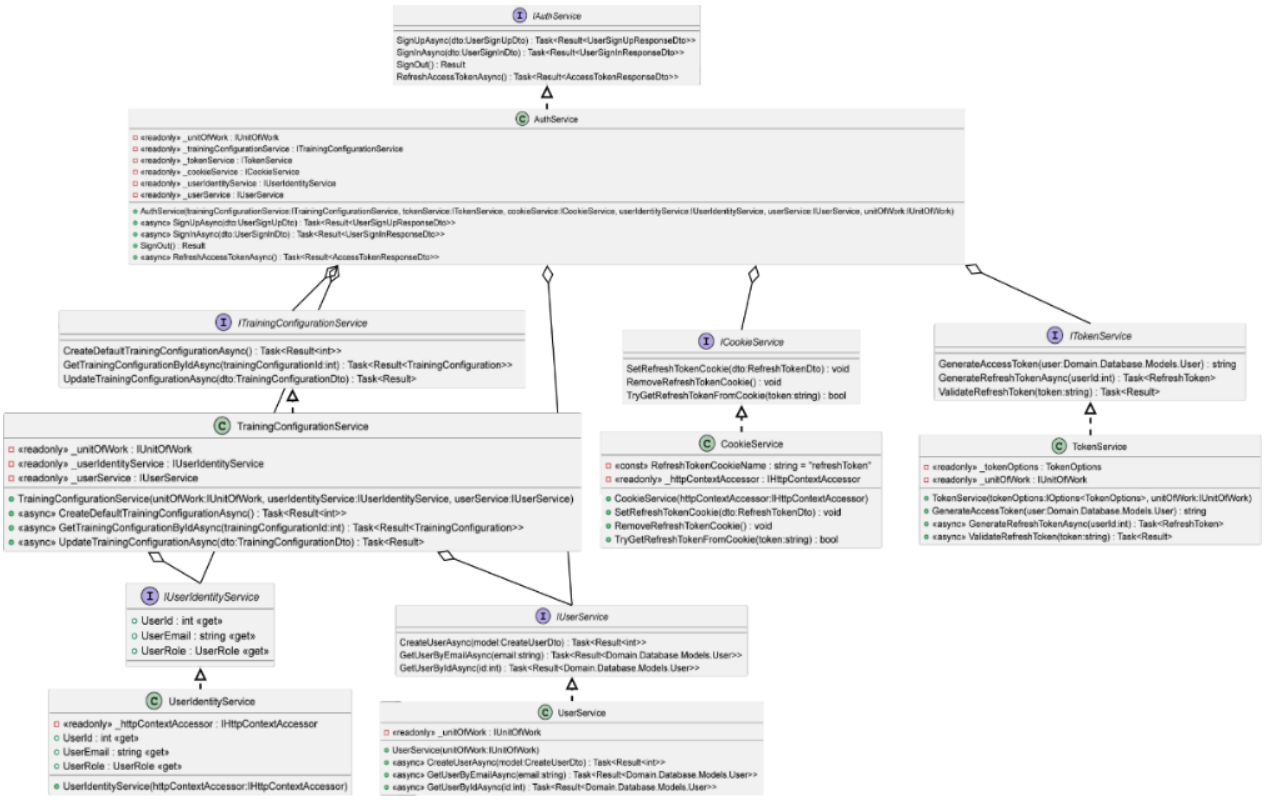

Рисунок 3.2 – Діаграма класів задіяних у процесі аутентифікації та авторизації користувача

Таблиця 3.2 – Характеристика класів задіяних у процесі аутентифікації та авторизації користувача

Назва класу	**Поля**	**Методи**	**Короткий опис**
UserIdentityService	_httpContextAccessor UserId UserEmail UserRole	–	Отримання даних про поточного користувача з токену авторизації
UserService	_unitOfWork	CreateUserAsync() GetUserByEmailAsync() GetUserByIdAsync()	Взаємодія з сутністю користувача на рівні системи

Кінець таблиці 3.2

TrainingConfigurationService	_unitOfWork _userIdentityService _userService	CreateDefaultTrainingConfigurationAsync() GetTrainingConfigurationByIdAsync() UpdateTrainingConfigurationAsync()	Взаємодія з сутністю налаштування тренування на рівні системи
CookieService	RefreshTokenCookieName _httpContextAccessor	SetRefreshTokenCookie() RemoveRefreshTokenCookie() TryGetRefreshTokenFromCookie()	Взаємодія з cookie
TokenService	_tokenOptions _unitOfWork	GenerateAccessToken() GenerateRefreshTokenAsync() ValidateRefreshToken()	Взаємодія з refresh та access токенами
AuthService	_unitOfWork _trainingConfigurationService _tokenService _cookieService _userIdentityService userService	SignUpAsync() SignInAsync() SignOut() RefreshAccessTokenAsync()	Взаємодія з аспектами авторизації користувача

3.2 Побудова діаграм станів та переходів

Діаграми станів та переходів слугують наочною демонстрацією можливих сценаріїв роботи системи.

Для вебзастосунку тренування швидкості вводу даних на клавіатурі було розроблено 3 діаграми станів та переходів:

1. для усієї системи (рис. 3.3);

2. для проходження тренування (рис. 3.4);

3. для збереження результатів після успішного завершення тренування (рис. 3.5).

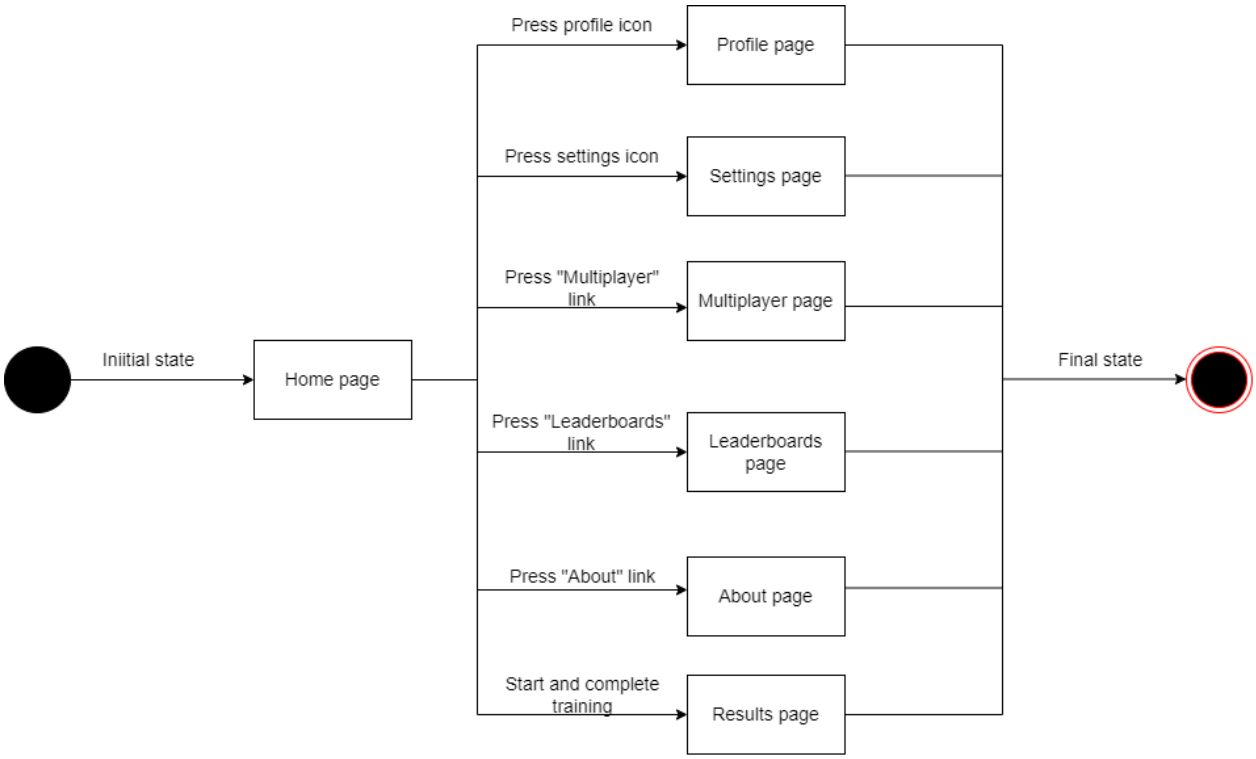

Рисунок 3.3 – Діаграма станів і переходів з головної сторінки до інших

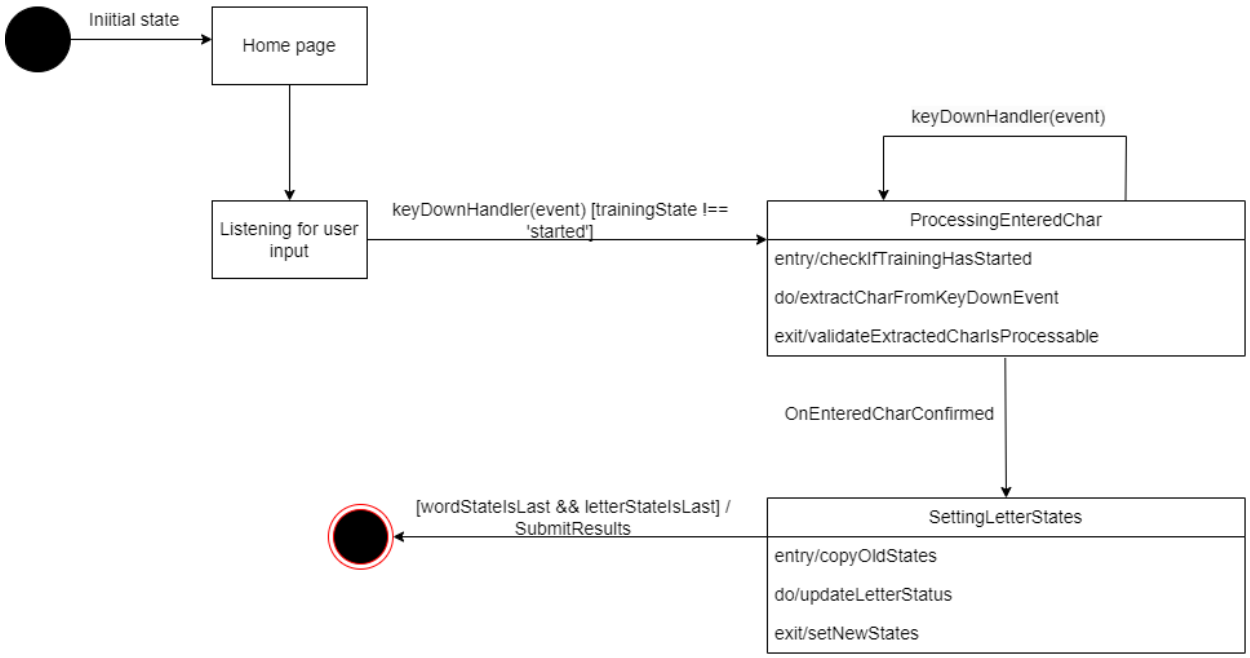

Рисунок 3.4 – Діаграма станів при проходженні тренування

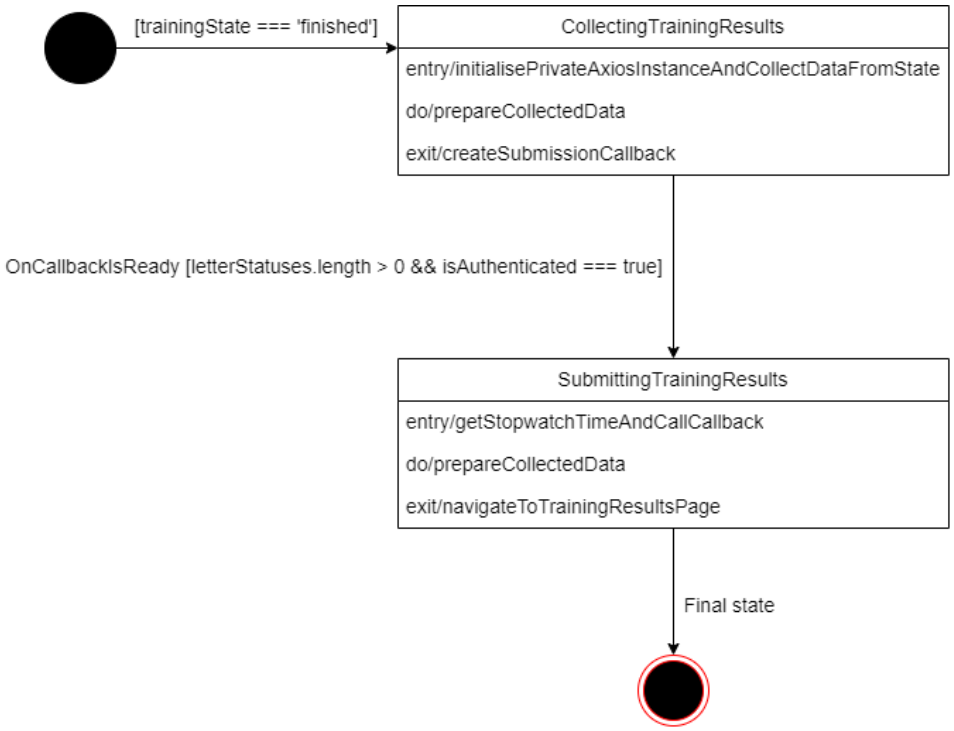

Рисунок 3.5 – Діаграма станів процесу обробки та відправки результатів

Отже, за допомогою такого роду діаграм можна прекрасно описувати основні послідовності можливих станів і переходів розроблюваної системи. Таким чином, ми маємо детальну характеристику поведінки системи впродовж її життєвого циклу виконання доступну для розуміння спеціалістів різного профілю, як бізнес аналітиків, наприклад, так і розробників.

3.3 Побудова діаграми компонентів

Діаграма компонентів використовуються для розділення всієї об’єктно-орієнтованої системи на декілька менших частин. Таким чином, можна досягти підвищення їх керованості та наочно відображати залежності. Компоненти можуть представляти як логічні, так і фізичні елементи, що задіяні у процесах.

Першим кроком у побудові діаграми компонентів є розділення діаграми класів на 4 шари (див. рис. 3.6).

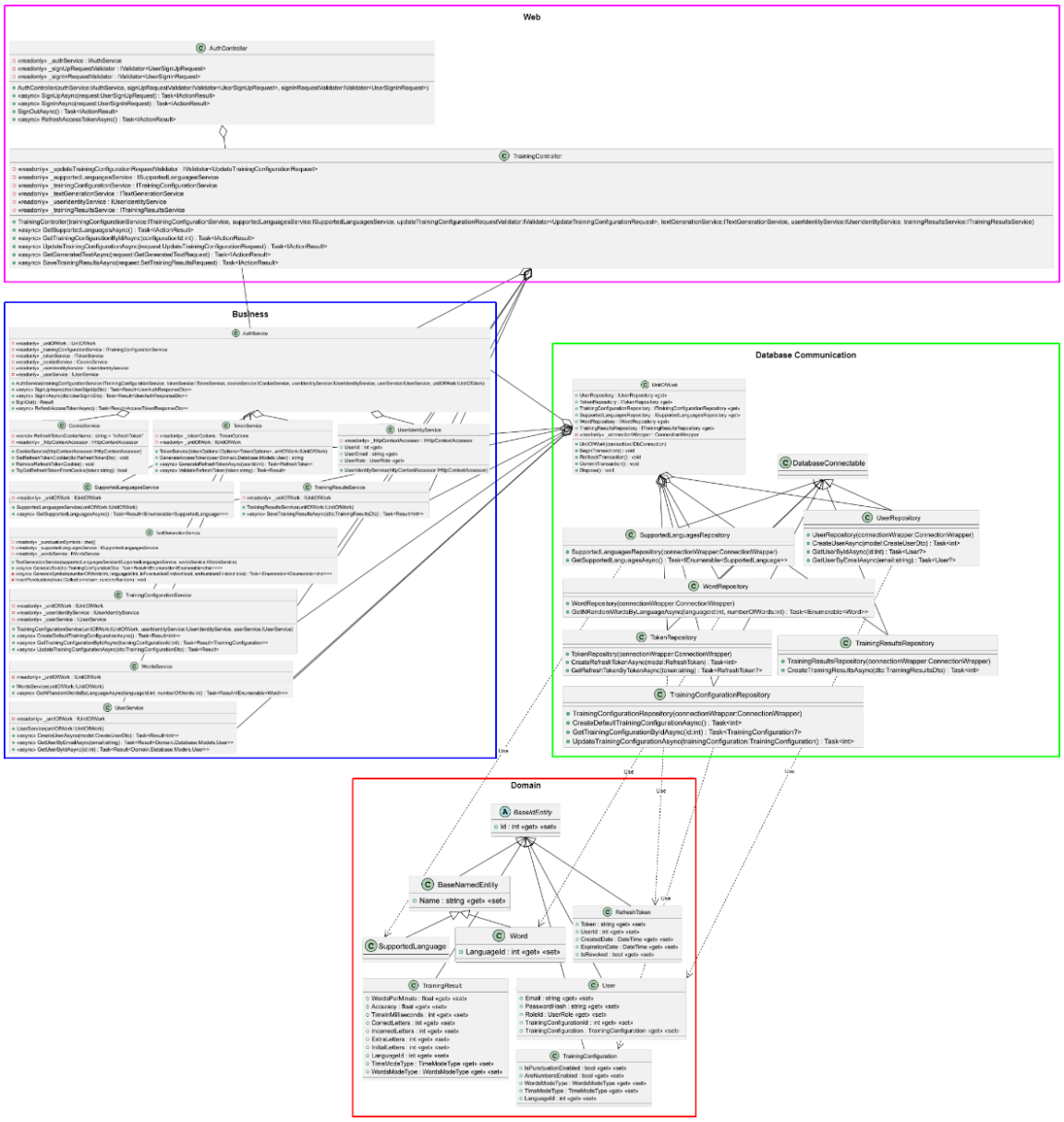

Рисунок 3.6 – Діаграма класів із чотиришаровою архітектурою

Перший шар (позначений пурпурним кольором у верхній частині діаграми) складається з набору класів (контролерів), що відповідають за прийом зовнішніх HTTP-запитів [16] від клієнтів.

Другий шар (позначений синім кольором у лівій частині діаграми) включає у себе сукупність класів, що включають у себе класи з алгоритмами, що засновуються на бізнес-логіці (генерація тексту, обробка даних користувача, обробка результатів тренувань тощо).

Третій шар (позначений зеленим кольором у правій частині діаграми) відповідає за збереження та отримання даних зі сховища. Він включає у себе класи (репозиторії), кожен з яких містить логіку для роботи за окремою сутністю (таблицею) у базі даних.

Четвертий шар (позначений червоним кольором у нижній частині діаграми) містить доменні моделі, які використовуються повсюдно у коді.

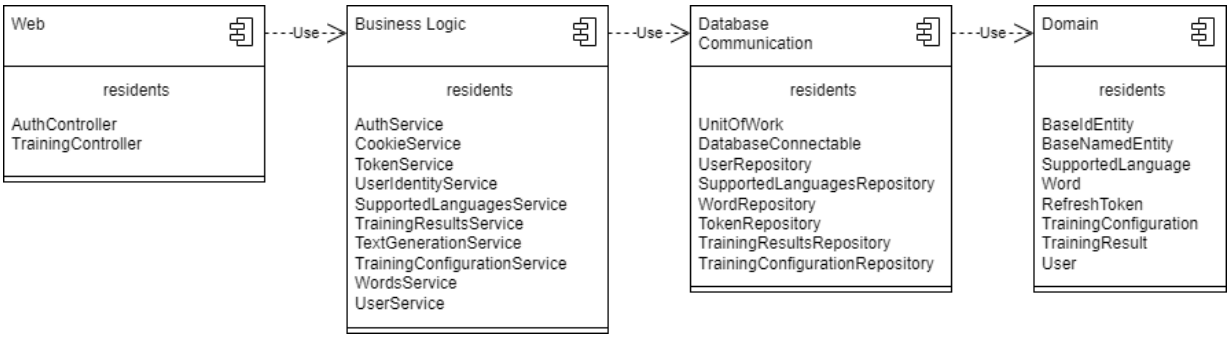

Рисунок 3.7 – Діаграма компонентів

Згідно зі діаграмою класів з чотиришаровою архітектурою побудовано діаграму компонентів для вебзастосунку тренування швидкості вводу даних на клавіатурі (див. рис. 3.7).

3.4 Побудова діаграми пакетів

Діаграми пакетів мають на меті спрощене представлення діаграм класів. Таким чином, кожен пакет представляє зі себе логічну групу програмних елементів. Ці елементи надалі використовуються для побудови ієрархії залежностей між собою, що в свою чергу дозволяє зрозуміти загальний рівень зв’язності системи та почати розробляти рішення щодо її зниження. Діаграму пакетів вебзастосунку тренування швидкості вводу даних на клавіатурі представлено на рисунку 3.8.

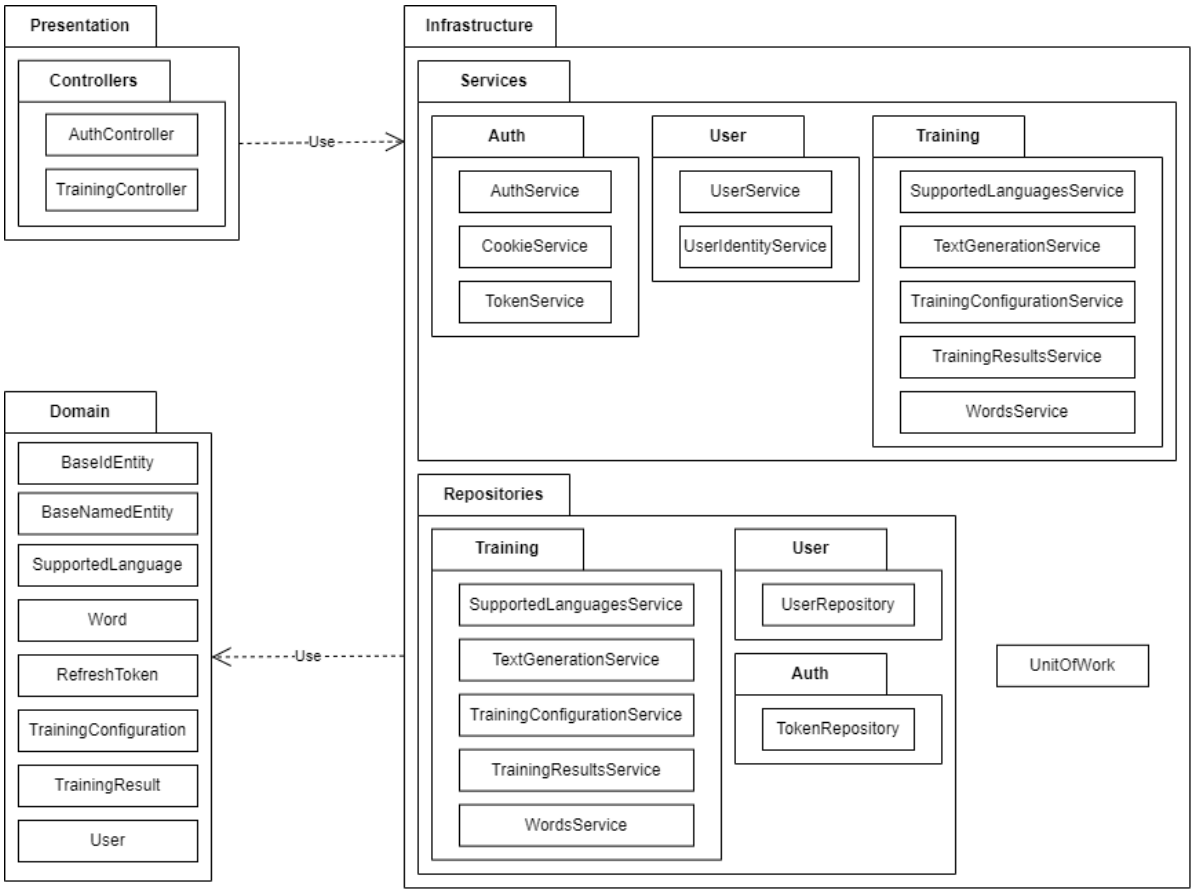

Рисунок 3.8 – Діаграма пакетів

Пакет «Presentation» містить у собі пакет «Controllers», який в свою чергу включає набір класів-контролерів, що приймають HTTP-запити від клієнтів. Він використовує пакет «Infrastructure», описаний нижче.

Пакет «Infrastructure» містить у собі пакети «Services» та «Repositories». Вони включають в себе набори пакетів розбитих за семантикою класів, що розміщені (Auth, User тощо). Усі класи з даного пакету відповідають за бізнес-логіку системи. Він використовує пакет «Domain», описаний нижче.

Пакет «Domain» включає у себе класи-сутності, що представляють з себе доменні моделі.

3.5 Вибір стеку технологій

Front-end

В рамках сучасних реалій неймовірний ріст популярності набувають саме JavaScript фреймворки. На рисунку 3.9 зображено статистику використання відповідних тегів на платформі stackoverflow, починаючи з 2008 року. Виходячи з отриманих даних можна зробити висновок, що починаючи з 2019 року React почав лідирувати серед користувачів. І це не дивно, адже тоді команда розробки фреймворку випустила версію 16.8, яка містила ключове нововведення – React Hooks. Воно спростило та прискорило роботу з ним, тим самим спровокувавши велику кількість користувачів перейти на його використання.

В порівнянні з альтернативами на ринку саме React є найбільш релевантним для розробки вебзастосунку тренування швидкості вводу даних на клавіатурі, оскільки він краще оптимізований для частих оновлень компонентів, що критично важливо для процесу проходження тренування. До того ж, він наділений купою переваг у вигляді колосальної підтримки як зі сторони розробників, так і суспільства, містить дуже широкий спектр готових рішень щодо елементів користувацького інтерфейсу, регулярні оновлення та високий ступінь налаштування.

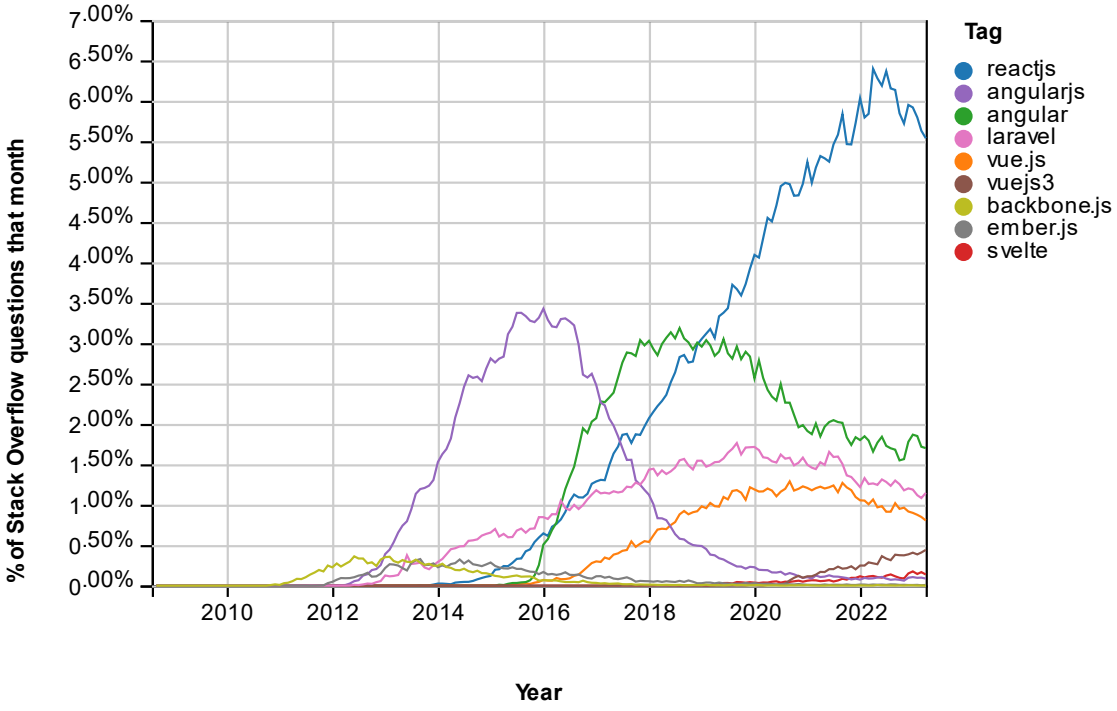

Рисунок 3.9 – Відсоток питань з тегами front-end фреймворків на платформі stackoverflow з місячним розподілом починаючи з 2008 року

Back-end

Для розробки Web API було обрано фреймворк ASP.NET Core. На відміну від React, він не займає провідного місця серед аналогів (див. рис. 3.10), але цьому є пояснення – до виходу .NET Core ASP.NET був націлений лише на операційну систему Windows, що доволі сильно зменшувало його універсальність. Після набуття платформою .NET кросплатформності цей фреймворк повільно, але стабільно почав набувати популярності. Розробники потроху починають переходити на сторону Microsoft, адже вони створили повністю свою екосистему, від IDE до хмарних рішень, які дуже легко та швидко налаштовуються для взаємодії.

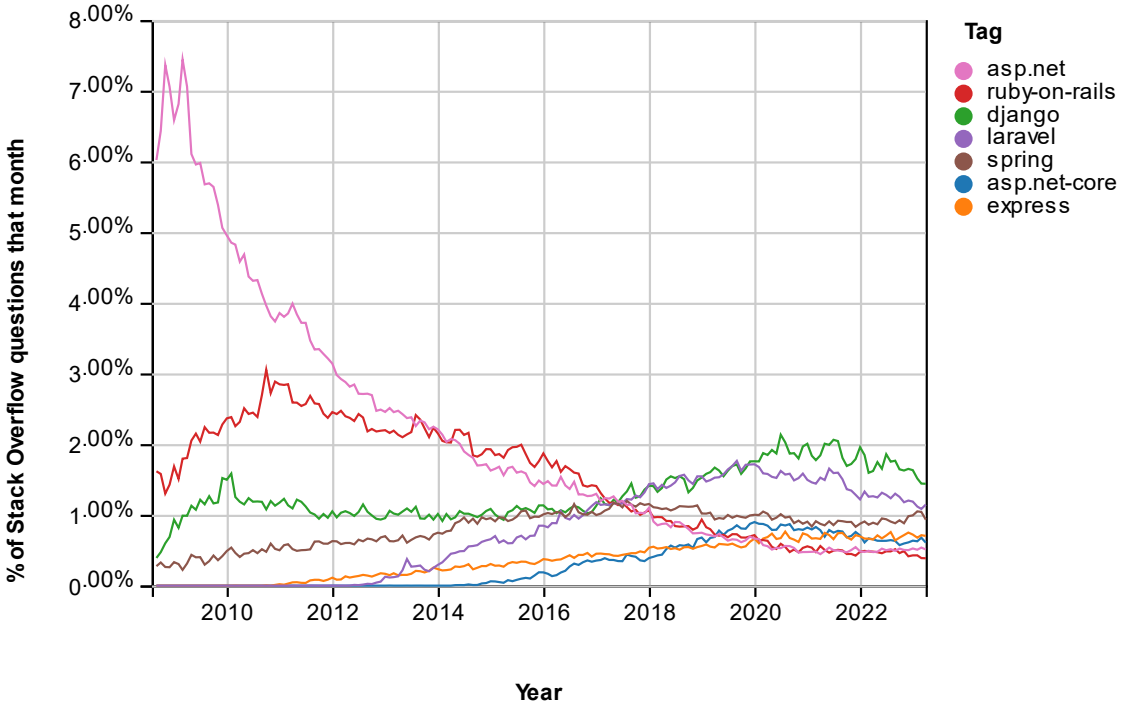

Рисунок 3.10 – Відсоток питань з тегами back-end фреймворків на платформі stackoverflow з місячним розподілом починаючи з 2008 року

Серед переваг ASP.NET Core можна виділити дуже детальну документацію, що постійно підтримується, стрімкі оновлення, що привносять багато дійсно корисних покращень, підтримку Docker та прекрасну екосистему для подальшої інтеграції.

Висновки до розділу 3

Третій розділ даної роботи присвячений проєктуванню вебзастосунку тренування швидкості вводу даних на клавіатурі. Під час роботи над ним отримано додаткові навички з побудови UML-діаграм класів, пакетів, станів та компонентів. Завдяки даним діаграмам зображено загальну структуру системи. Також, проведено аналіз стеку технологій для визначення найкращих кандидатів для використання на етапі розробки.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ

4.1 Огляд UI

Проєктування користувацького інтерфейсу здійснювалось за допомогою програмного забезпечення «Figma». Обраний шрифт – Mulish. Легкість сприйняття великих об’ємів тексту написаних шрифтом такого типу задовільна і не буде слугувати причиною перенапруження очей користувачів [17]. Будь-який користувач може зайти на головну сторінку і отримати доступ до її повного функціоналу, а саме – проходження тренування в режимі для одного користувача (див. рис. 4.1). Як можна побачити з переліку UI-елементів, наявні налаштування генерації пунктуації, чисел, вибір режиму на кількість слів або на час та мови. Також, можна виконати повторну генерацію тексту шляхом натискання на кнопку з іконкою круглої стрілки під текстом.

Рисунок 4.1 – Головна сторінка вебзастосунку

Перейдемо до сторінки профілю користувача. Якщо користувач не пройшов процес аутентифікації, то його буде переадресовано на сторінку входу, де він має ввести свою пошту, що використовувалась під час реєстрації та пароль (див. рис. 4.2).

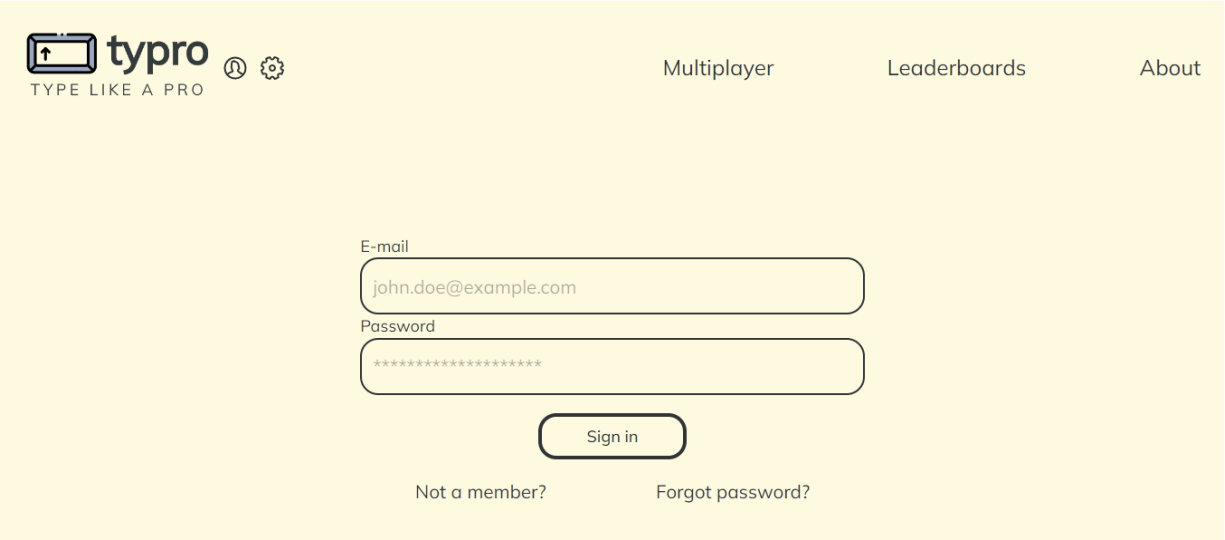

Рисунок 4.2 – Сторінка входу до облікового запису

Успішне проходження аутентифікації дозволяє отримати доступ до сторінки профілю (див. рис. 4.3 та 4.4). На ній зображені такі дані, як псевдонім користувача, кількість початих та завершених тренувань, відображення найкращих результатів по кожному з доступних режимів та графік зміни швидкості та точності вводу за найкращими результатами за кожен день з фільтрацією по мові, режиму та часовому проміжку.

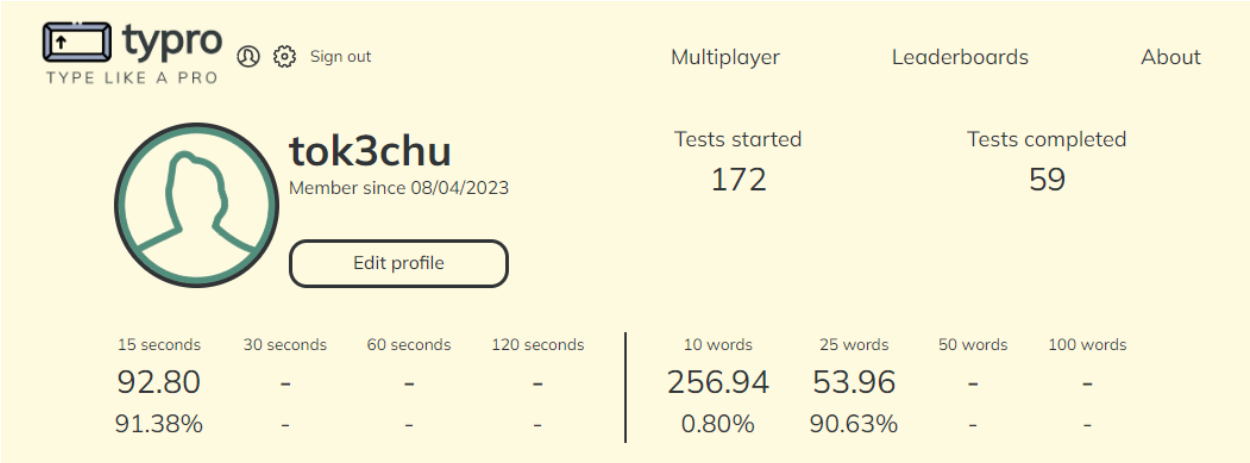

Рисунок 4.3 – Частина сторінки профілю користувача з загальною статистикою

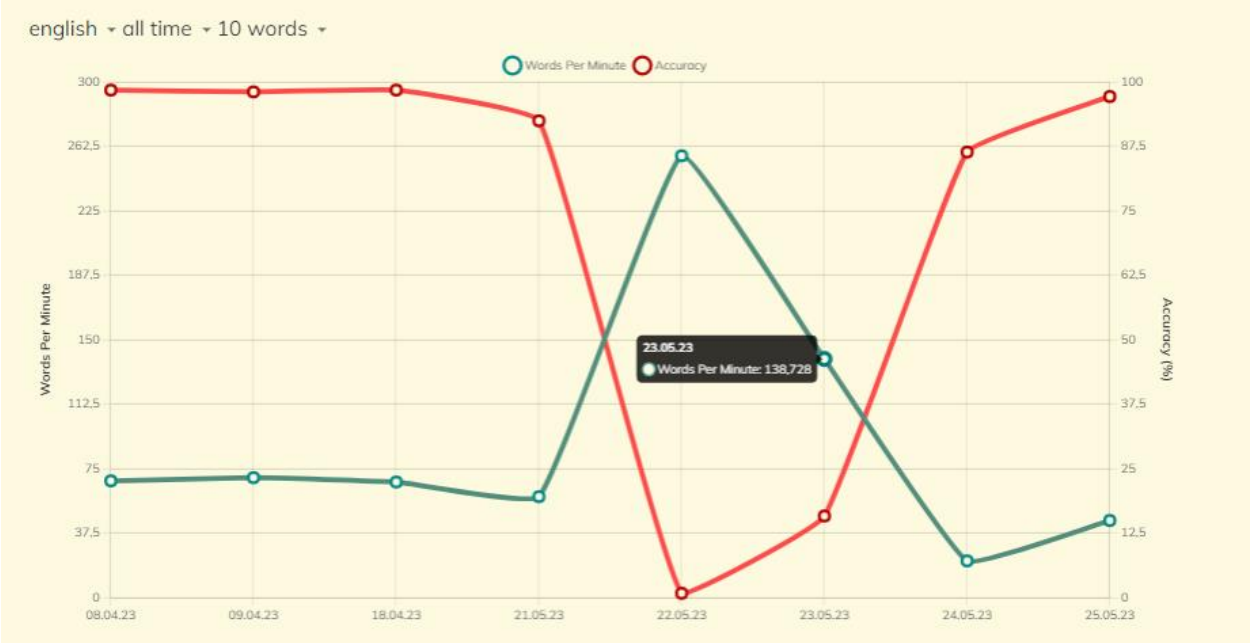

Рисунок 4.4 – Частина сторінки профілю користувача з графіком зміни швидкості та точності вводу

Наступною є сторінка налаштувань, яка дозволяє змінювати тему застосунку (див. рис. 4.5). Всього доступно дві теми – світла та темна.

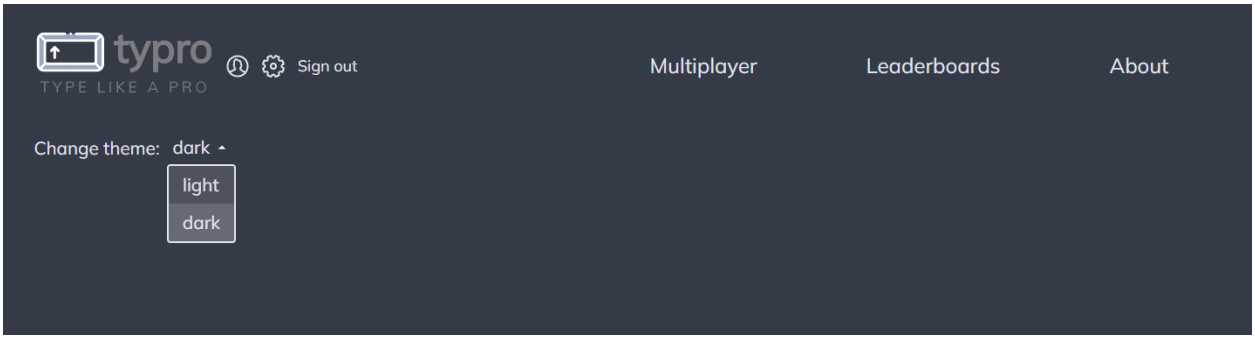

Рисунок 4.5 – Сторінка з налаштуваннями застосунку

Сторінка для багатокористувацького режиму передбачає дві варіації. Перша – якщо користувач створив групу (див. рис. 4.6), та друга – якщо користувач приєднався до групи (див. рис. 4.7). Обидві вони передбачають можливість скопіювати код запрошення, перегляд користувачів у групі з їх прогресом проходження тренування, інтерфейс налаштування тренування та, відповідно, згенерований текст. Головними відмінностями між ними є наявність кнопки «Start», яка стає активною після наявності хоча б двох

користувачів та можливість виставляти налаштування тренування у того, хто створив групу. Інші ж лише споглядають зміни у реальному часі.

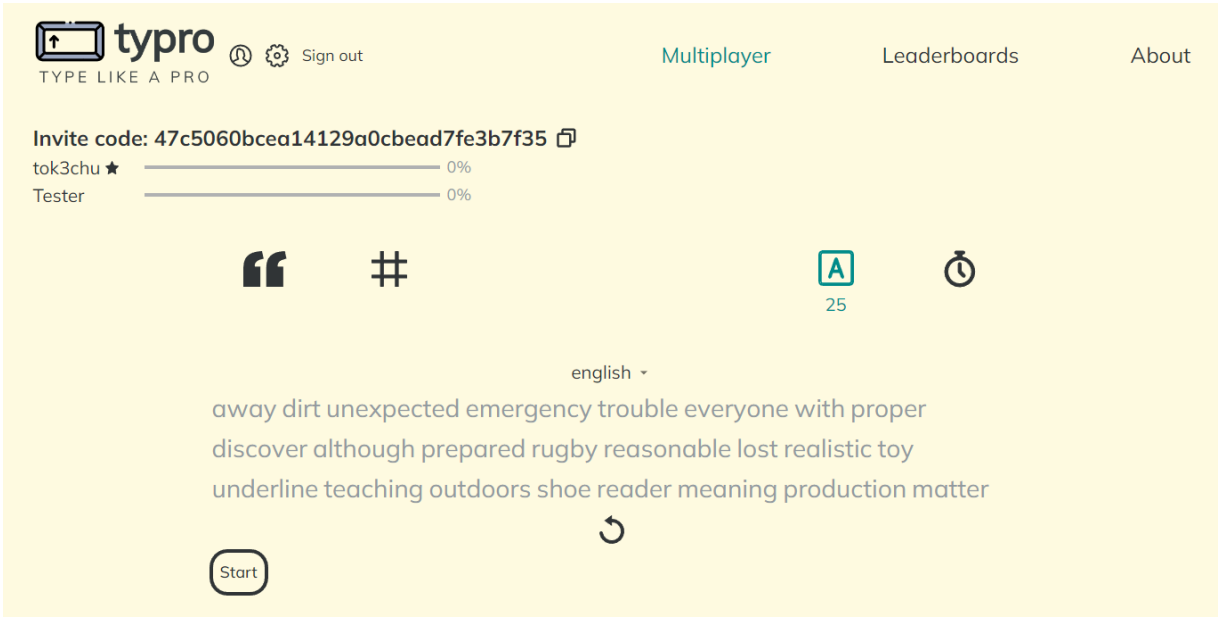

Рисунок 4.6 – Сторінка для багатокористувацького режиму з виглядом для того, хто створив групу

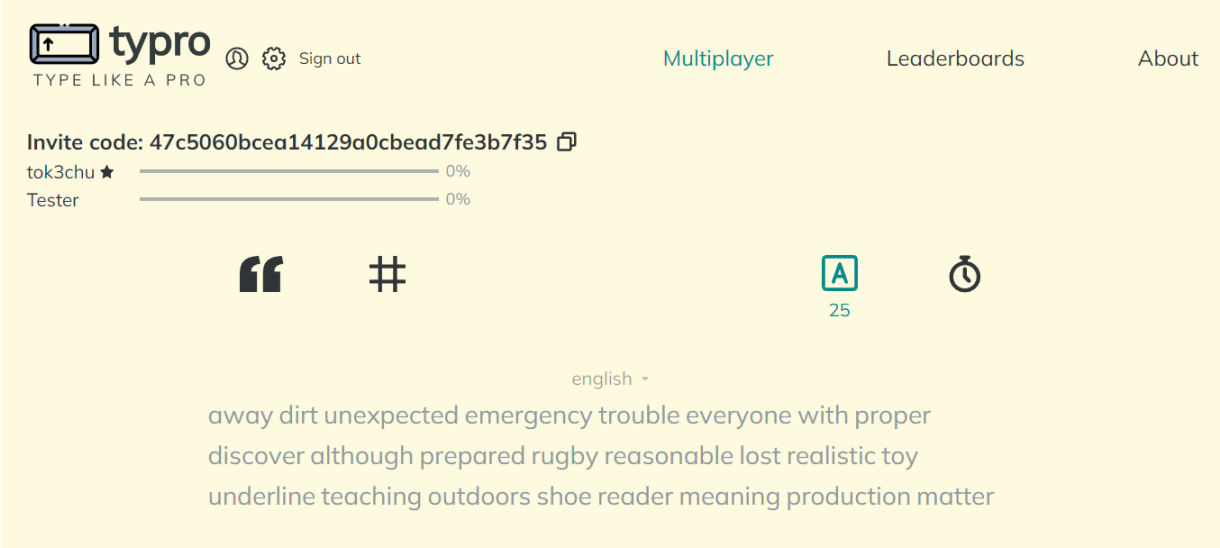

Рисунок 4.7 – Сторінка для багатокористувацького режиму з виглядом для того, хто доєднався до групи

Сторінка зі списками лідерів зображена на рисунку 4.8. Як і у випадку з графіком у профілі користувача, тут так само наявні фільтри по мові, часовому проміжку та режимах тренувань. Самі ж записи містять інформацію про місце

користувача в рейтингу, його псевдонім, кількість слів у хвилину, точність та дату закінчення проходження тренування. Також, дана таблиця містить пагінацію з можливістю обрати кількість записів на одну сторінку.

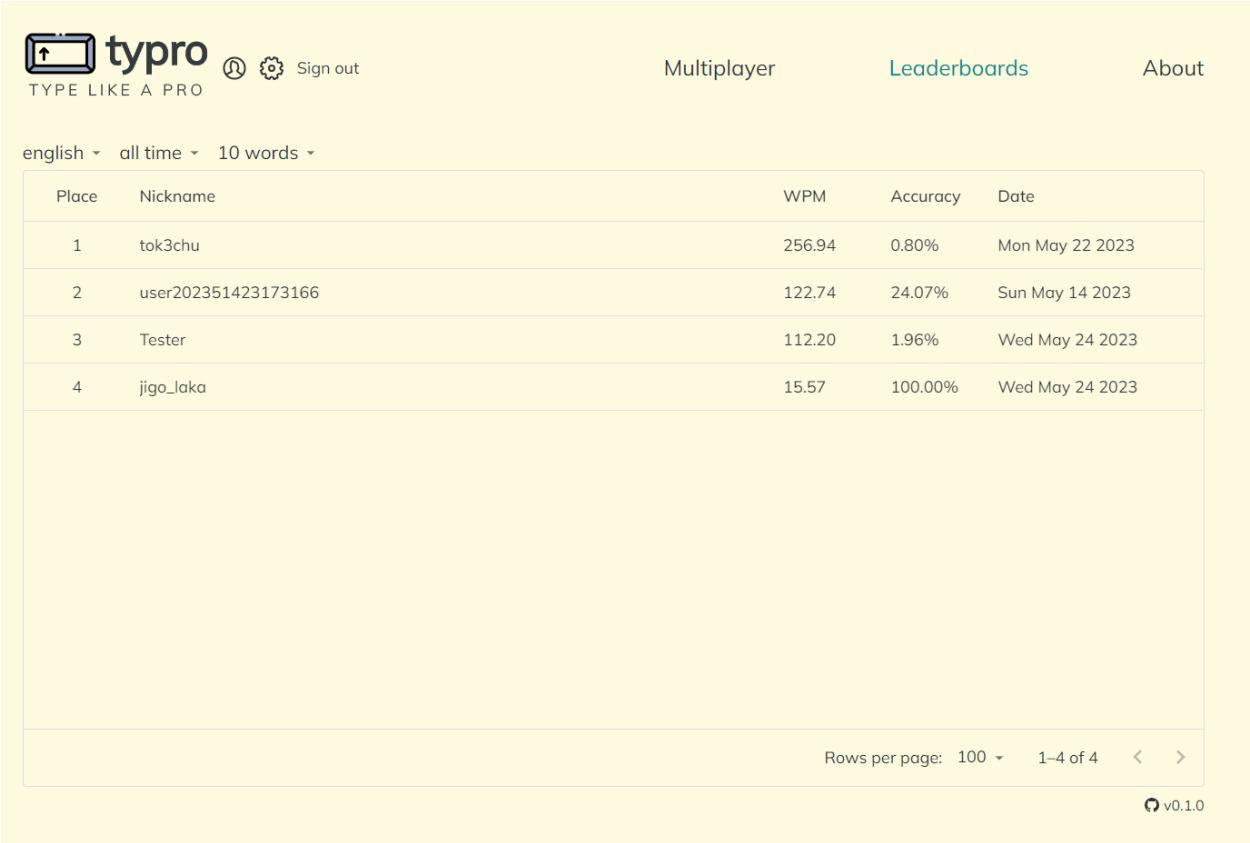

Рисунок 4.8 – Сторінка зі списками лідерів

Останньою є сторінка «About» з інформацією про застосунок (див. рис. 4.9).

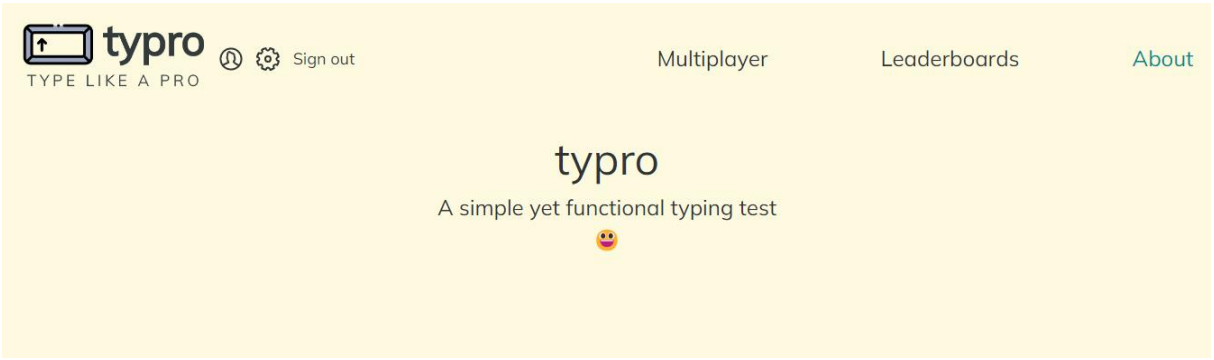

Рисунок 4.9 – Сторінка з інформацією про застосунок

Таким чином, було розглянуто основні елементи користувацького інтерфейсу на головних сторінках розроблюваного вебзастосунку. Надалі буде приведено деталі реалізації з більш наочними прикладами їх практичного застосування.

4.2 Реалізація функціоналу вебзастосунку тренування швидкості вводу даних на клавіатурі

4.2.1 Структура front-end застосунку

Перед тим як перейти до фактичного огляду програмного коду варто приділити увагу загальному підходу до структуризації проєкту. Таким чином спрощується майбутній перегляд кодової бази та розуміння принципів, використаних під час розробки. На рисунку 4.10 представлено загальну структуру front-end проєкту.

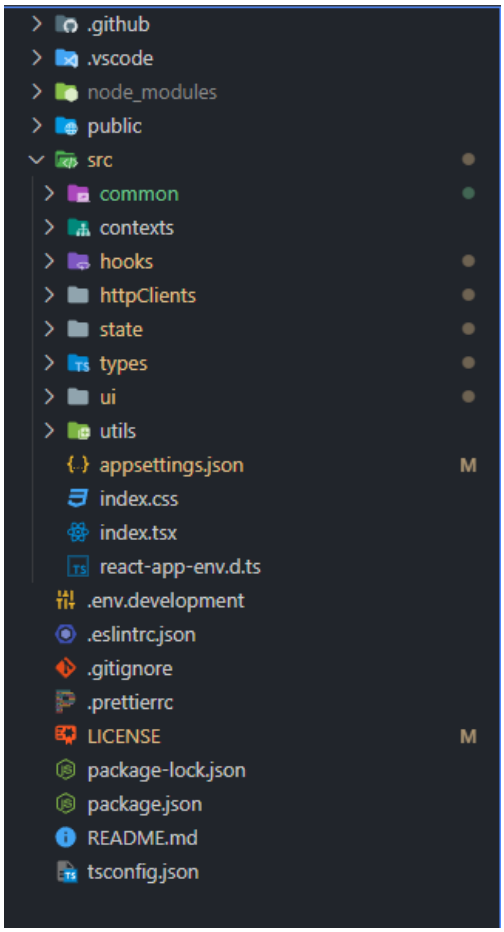

Рисунок 4.10 – Загальна структура front-end проєкту

Кожна з директорій у директорії «src» має наступне значення:

1. common – містить спільні програмні елементи для всього проєкту (в нашому випадку – налаштування теми);

2. contexts – містить React Contexts;

3. hooks – містить користувацькі React Hooks;

4. httpClients – містить HTTP-клієнти для звернення на back-end сервер;

5. state – містить налаштування React Redux [18];

6. types – містить користувацькі типи, що застосовуються у системі;

7. ui – містить React-компоненти, що відповідають за користувацький інтерфейс;

8. utils – містить методи-утиліти (на кшталт роботи з local storage).

Основу застосунку складає вміст директорії «ui», тому розглянемо її більш детально. Вона також містить директорію «common», яка включає в себе спільні програмні компоненти для UI-частини. Новою є папка «groups». Вона, в свою чергу, вміщає в собі директорії, що містять React-компоненти для кожного з розділів вебсайту, допоміжні методи-утиліти та налаштування маршрутизації.

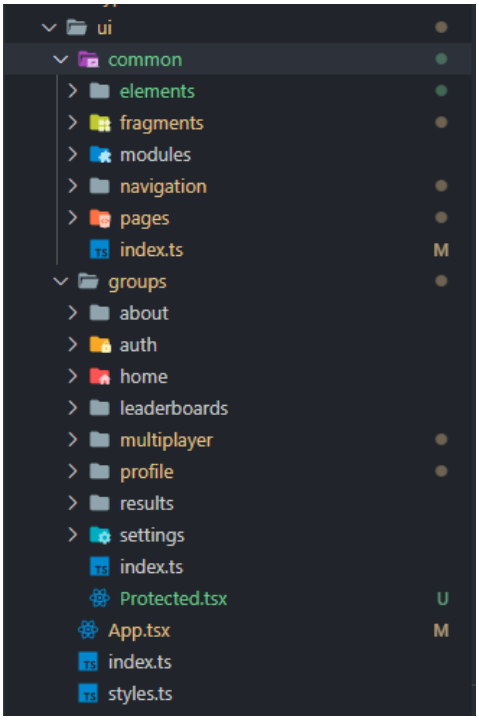

Рисунок 4.11 – Структура директорії src/ui

4.2.2 Структура back-end застосунку

Сервер web API також вартий окремого огляду. В третьому розділі вже зачіпалась тема пакетів, з яких складається система. Тепер настав час переглянути цю структуру більш детально (див. рис. 4.12).

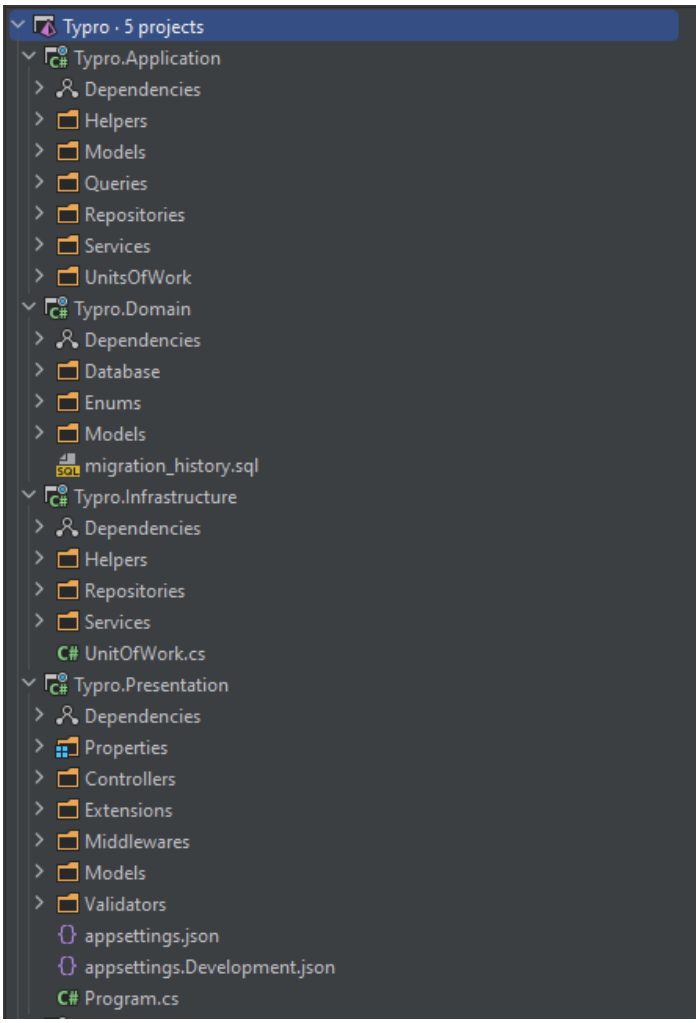

Рисунок 4.12 – Структура back-end проєкту

Як можна побачити, проєкт «Domain» містить моделі, що єдині для всього домену, разом із історією міграцій бази даних. Проєкт «Application» містить абстракції, використовувані ними моделі та SQL-запити. У свою ж чергу «Infrastructure» наповнений класами, що є конкретними реалізаціями абстракцій з «Application». Останній проєкт під назвою «Presentation» охоплює класи-контролери, моделі запитів, файли налаштування тощо. Він містить точку входу програми.

4.2.3 Головна сторінка

Головна сторінка вебзастосунку швидкості тренування вводу даних на клавіатурі представляє собою інтерфейс для проходження тренування в режимі для одного користувача. Основні елементи UI було розглянуто у попередньому розділі. Перейдемо до деталей їх реалізації.

Першим компонентом є панель налаштування тренування (див. рис. 4.13).

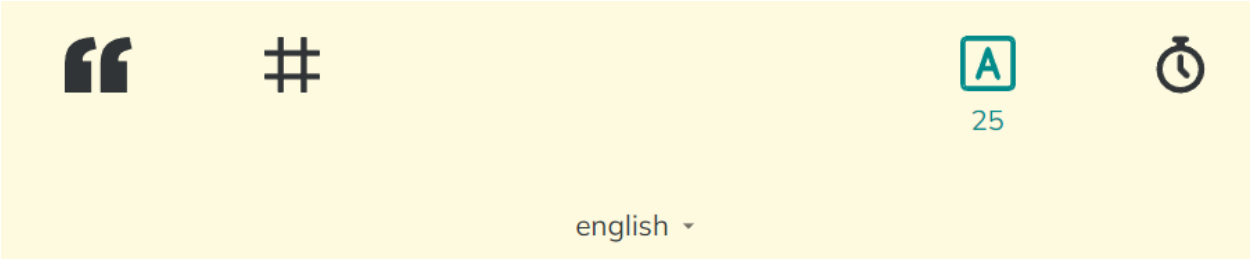

Рисунок 4.13 – Панель налаштування тренування

У наступному фрагменті коду представлено компонент, який відповідає за відображення даної панелі. Вона містить у собі елементи «IconSwitchesElement» та «LanguageSelectElement», що отримують дані через вхідні параметри:

```
interface TrainingConfiguration {
  areNumbersGenerated: boolean;
  isPunctuationGenerated: boolean;
  wordsMode: WordsModeType;
  timeMode: TimeModeType;
  languageInfo: LanguageInfo;
}

interface Props {
  languagesInfo: LanguageInfo[];
  trainingConfiguration: TrainingConfiguration;
}

const TrainingConfigurationFragment = (props: Props): JSX.Element => {
  const { state: trainingState } = useAppSelector((store) =>
store.data.trainingState);

  return (
    <Fade in={trainingState !== 'started'}>
      <Box sx={styles.mainContainer}>
        <IconSwitchesElement {...props.trainingConfiguration} />
        <LanguageSelectElement
          languagesInfo={props.languagesInfo}
          preferredLanguageInfo={props.trainingConfiguration.languageInfo}
        />
```

```
      </Box>
    </Fade>
  );
};

export default memo(TrainingConfigurationFragment);
```

Інформація про доступні для вибору мови надходить з web API. Код методу, що містить бізнес-логіку даного запиту наведено нижче:

```
public async Task<Result<IEnumerable<SupportedLanguageDto>>>
GetSupportedLanguagesAsync()
{
    IEnumerable<SupportedLanguage> supportedLanguages =
        await
_unitOfWork.SupportedLanguagesRepository.GetSupportedLanguagesAsync();

    IEnumerable<SupportedLanguageDto> dtos = supportedLanguages.Select(e =>
        new SupportedLanguageDto(e.Id, e.Name,
            e.Name.Equals("english",
StringComparison.InvariantCultureIgnoreCase)));

    return Result.Ok(dtos);
}
```

Наступний компонент відповідає за генерацію тексту. Його основним функціоналом є отримання тексту зі серверу та конвертування його у стани кожного окремого слова для відображення. Логіка для формування запиту на сервер:

```
const { data } = useQuery({
  queryKey: ['generatedText', trainingConfiguration],
  queryFn: async () => {
    const data = await trainingHttpClient.getGeneratedText({
      ...trainingConfiguration,
      languageId: ensure(trainingConfiguration.languagesInfo.find((e) =>
e.isActive)).id
    });

    await sleep(100);
    setRestartScheduledStatus(false);
    dispatch(trainingStateActions.setWordsTyped(0));
    dispatch(trainingStateActions.setState('initial'));

    return data;
  },
  enabled: isRestartScheduled && trainingConfiguration.languagesInfo.length >
0
});
```

Отримання станів слів виконується наступним чином:

```
const generatedText: string[][] = data ?? [];
useEffect(() => {
  if (generatedText.length > 0) {
    setWordStates(
      generatedText.map<WordProps>((wordChars, wordCharsIndex) => {
        return {
          letters: wordChars,
          isActive: wordCharsIndex === 0,
```

```
            isCounted: false,
            onMoveToAnotherWord: moveOnToAnotherWordHandler,
            onTrainingStart: trainingStartHandler,
            onWordModeTrainingEnd: letterStatusesSubmissionHandler
          };
        })
    );
  }
}, [generatedText, setWordStates]);
```

Код генерації тексту на стороні back-end можна знайти у додатках Б та В. Варто зазначити, що перелік підтримуваних слів зберігається в базі даних. Для обміну інформацією з нею було обрано бібліотеку Dapper, оскільки операції зчитування даних найчастіші у системі і саме вона найкраща з конкурентів у цьому [19].

4.2.4 Обробка користувацького вводу

Мабуть найголовнішим аспектом роботи розроблюваного тренажеру є обробка користувацького вводу. Кожне слово, що відображається на екрані є окремим компонентом, який містить у собі слухач подій зі сторони клавіатури:

```
const keyDownHandler = (event: KeyboardEvent<HTMLDivElement>): void => {
  const char = event.key;
  const code = event.code;

  if (invalidCharCodes.find((e) => e === code) !== undefined) {
    return;
  }

  if (trainingState !== 'started') {
    props.onTrainingStart();
  }

  setLetterStates((oldStates) => {
    let newStates = [...oldStates];

    if (code === backspaceCode) {
      if (event.ctrlKey) {
        newStates = newStates.filter((e) => e.status !== 'extra');
        newStates.forEach((s) => (s.status = 'initial'));
        setPosition(0);
        return newStates;
      }

      const previousLetterPosition = position - 1;

      if (previousLetterPosition === -1) {
        props.onMoveToAnotherWord(false);
        return newStates;
      }

      setPosition(previousLetterPosition);
      const currentLetterState = newStates[previousLetterPosition];
```

```
      if (currentLetterState !== undefined) {
        if (currentLetterState.status === 'extra') {
          return newStates.slice(0, newStates.length - 1);
        }

        newStates[previousLetterPosition] = {
          character: currentLetterState.character,
          status: 'initial',
          position: currentLetterState.position
        };
      }

      return newStates;
    }

    if (position === letterStates.length) {
      if (code === spaceCode) {
        props.onMoveToAnotherWord(true);
        return newStates;
      } else {
        newStates.push({ character: char, position, status: 'extra' });
      }
    } else {
      const currentLetterState = newStates[position];
      const newLetterStatus: LetterStatus =
        char === currentLetterState.character ? 'correct' : 'incorrect';
      newStates[position] = {
        character: currentLetterState.character,
        status: newLetterStatus,
        position: currentLetterState.position
      };
    }

    setPosition(position + 1);

    return newStates;
  });
};
```

В процесі обробки події перевіряється відповідність коду введеного символу на можливість обробки, на необхідність виконання стирання останнього введеного символу та досягання останнього символу у слові.

Оскільки на сторінці відображається одночасно дуже багато слів, кожне з них використовує захват фокусу користувацького вводу, якщо дане слово є поточним.

Також, кожен символ у слові представляє окремий компонент зі своїм внутрішнім станом для збереження даних про позицію, свій стан (правильність або неправильність введеного користувачем символу по відношенню до даного символу) та сам символ.

Більш детальний програмний код щодо обробки користувацького вводу можна знайти у додатку В.

4.2.5 Аутентифікація та авторизація користувачів

Система аутентифікації користувачів базується на використанні пошти та паролю. Під час реєстрації облікового запису необхідно ввести даний набір даних (див. рис. 4.14). Після успішного проходження даного процесу користувачу надається комбінація JWT-токену доступу та токену оновлення (див. рис. 4.15 та 4.16), які він використовує для авторизації під час запитів до захищених ресурсів back-end серверу.

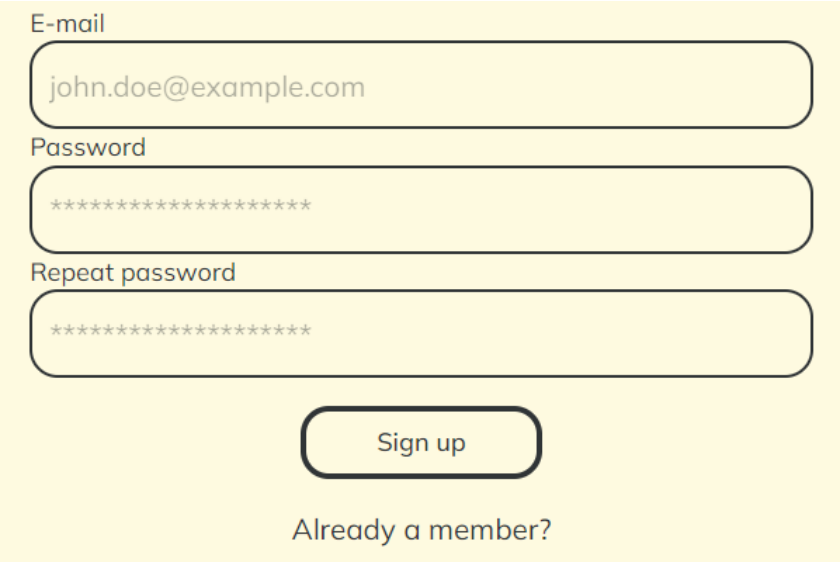

Рисунок 4.14 – Форма реєстрації нового облікового запису

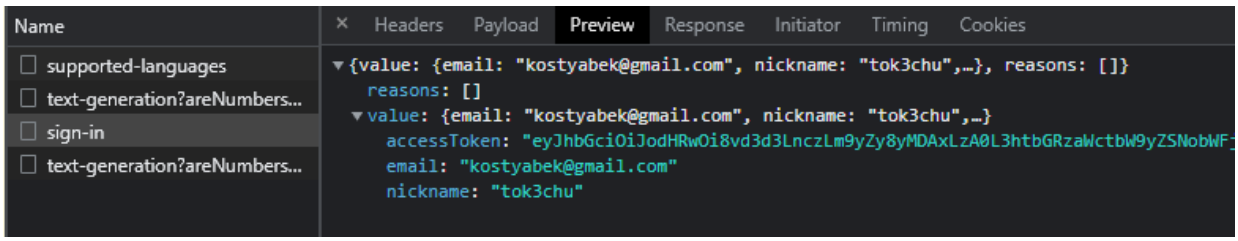

Рисунок 4.15 – Тіло відповіді серверу з JWT-токеном доступу

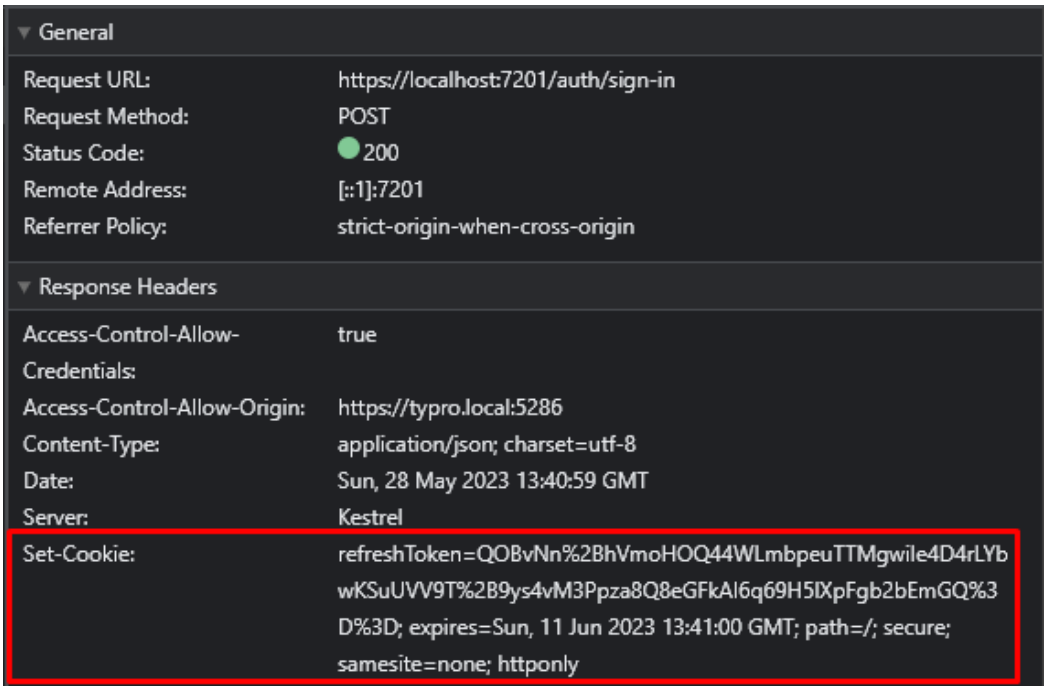

Рисунок 4.16 – Заголовки відповіді back-end серверу на запит щодо аутентифікації користувача

Токен доступу зберігається у local storage браузеру, а токен оновлення в якості httponly cookie.

4.2.6 Багатокористувацький режим

Реалізація багатокористувацького режиму впроваджена за принципом створення приватних груп із кодом запрошення, тобто, користувачі проходять тренування лише з тими, з ким вважатимуть за потрібне.

При переході на відповідну сторінку є можливість натиснути кнопку «Create lobby» для створення групи та «Join lobby» для підключення до існуючої (див. рис. 4.17 та 4.18).

Рисунок 4.17 – Кнопки для вибору створення або підключення до групи

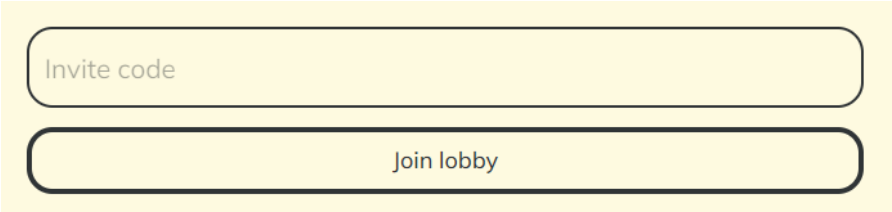

Рисунок 4.18 – Форма для вводу коду запрошення до групи

Для впровадження багатокористувацької інфраструктури було використано сервіс «Ably» та бібліотеку «@ably-labs/react-hooks». Він надає API для налагодження спілкування клієнтів у реальному часі через протокол WebSockets [20].

Завдяки своїй простоті використання «Ably» дуже сильно економить час та зусилля, необхідні для створення бажаного функціоналу. Концепція груп вже реалізована за допомогою так званого «presence», тобто кожен клієнт, що підключається до каналу має можливість утримувати дані про себе і синхронізувати ці дані з усіма іншими клієнтами, розсилаючи відповідні повідомлення. Приклад використання наведено нижче:

```
const [presence, updatePresence] =
usePresence<AppPresenceData>(lobbyInfo.channelId, {
  isCreator,
  indicatorValue,
  place
});
const [channel] = useChannel(lobbyInfo.channelId, 'restart-scheduled', () =>
{
  updatePresence({
    isCreator,
    indicatorValue: 0,
    place: 1
  });
});

useChannel(lobbyInfo.channelId, 'increment-place', (message) => {
  const myPresenceData = ensure(presence.find((e) => e.clientId ===
message.clientId)).data;
  const newPresenceData: AppPresenceData = {
    isCreator: myPresenceData.isCreator,
    indicatorValue: myPresenceData.indicatorValue,
    place: message.data.incrementedPlace
  };

  updatePresence(newPresenceData);
  dispatch(multiplayerActions.setPlace(newPresenceData.place));
});
```

Спочатку шляхом виклику хуку «usePresence» поточний клієнт у каналі оголошує свої початкові дані, передаючи їх другим аргументом. Хук, в свою чергу, повертає дані усіх клієнтів з каналу та метод для оновлення своїх даних у вигляді масиву. В наступних двох викликах хуку «useChannel» для визначення обробників подій «restart-scheduled» та «increment-place» зображено приклад використання методу «updatePresence».

Оновлення даних про прогрес проходження тренування відбувається після кожного введеного користувачем слова:

```
if (trainingConfiguration.wordsMode !== WordsModeType.TurnedOff) {
  const completedLettersQty = completedWords.flatMap((e) =>
e.letters).length;
  const allLettersQty = newStates.flatMap((e) => e.letters).length;

  newPresenceData = {
    isCreator,
    indicatorValue: (completedLettersQty / allLettersQty) * 100,
    place
  };
  updatePresence(newPresenceData);
} else {
  newPresenceData = {
    isCreator,
    indicatorValue: numberOfCompletedWords,
    place
  };
  updatePresence(newPresenceData);
}
```

Отже, шляхом використання готової інфраструктури спілкування клієнтів у реальному часі досягається мінімізація помилок, оскільки організація такого роду функціоналу з нуля потребує немалих зусиль та розуміння кожної дрібниці, що може призвести до небажаних наслідків.

4.3 Тестування вебзастосунку

Розробка жодної системи не може вважатись завершеною без її ретельного тестування. Отримані результати представлено у таблицях 4.1-4.5.

Таблиця 4.1 – Реєстрація нового облікового запису

Діючі особи	Користувач, система
Мета	Створення нового облікового запису
Передумова	Користувач виконує дії будучи неавторизованим

Кінець таблиці 4.1

<table>
<tr><td colspan="2">Успішний сценарій:
1. Користувач переходить на сторінку реєстрації.
2. Користувач вводить необхідні дані (пошта, двічі один і той самий пароль, що відповідає заданим критеріям).
3. Користувач натискає кнопку «Sign up».
4. Back-end сервер створює обліковий запис та повертає у відповідь клієнту комбінацію JWT-токену доступу та токену оновлення.
5. Front-end застосунок перенаправляє користувача на головну сторінку.</td></tr>
<tr><td colspan="2">Сценарій успішний. Створено новий обліковий запис.</td></tr>
<tr><td colspan="2">Розширення</td></tr>
<tr><td>1a</td><td>Користувач заповнив не всі необхідні поля. Front-end застосунок виводить відповідне повідомлення на екран. Спроба відправки некоректного запиту на back-end сервер завершується відповіддю з повідомленням про неправильність запиту. Результат: обліковий запис не створено.</td></tr>
<tr><td>2a</td><td>Користувач вводить пошту вже існуючого облікового запису. Back-end сервер повертає відповідь з відповідним повідомленням та front-end застосунок виводить його на екран. Результат: обліковий запис не створено.</td></tr>
<tr><td>3a</td><td>Back-end сервер недоступний на момент відправки запиту на реєстрацію. Результат: обліковий запис не створено.</td></tr>
<tr><td colspan="2">Всі сценарії розширення успішно виконані.</td></tr>
</table>

Таблиця 4.2 – Аутентифікація в системі

<table>
<tr><td>Діючі особи</td><td>Користувач, система</td></tr>
<tr><td>Мета</td><td>Успішно пройти аутентифікацію</td></tr>
<tr><td>Передумова</td><td>Користувач виконує дії будучи неавторизованим</td></tr>
<tr><td colspan="2">Успішний сценарій:
1. Користувач переходить на сторінку аутентифікація.
2. Користувач вводить необхідні дані (пошта та пароль).Користувач натискає кнопку «Sign in».</td></tr>
</table>

Кінець таблиці 4.2

3. Back-end сервер перевіряє відповідність переданих даних до існуючого облікового запису та повертає у відповідь клієнту комбінацію JWT-токену доступу та токену оновлення. 4. Front-end застосунок перенаправляє користувача на головну сторінку.	
Сценарій успішний. Пройдено процес аутентифікації в системі.	
Розширення	
1a	Користувач заповнив не всі необхідні поля. Front-end застосунок виводить відповідне повідомлення на екран. Спроба відправки некоректного запиту на back-end сервер завершується відповіддю з повідомленням про неправильність запиту. Результат: процес аутентифікації не пройдено.
2a	Користувач вводить пошту неіснуючого облікового запису. Back-end сервер повертає відповідь з відповідним повідомленням та front-end застосунок виводить його на екран. Результат: процес аутентифікації не пройдено.
3a	Back-end сервер недоступний на момент відправки запиту на аутентифікацію. Результат: процес аутентифікації не пройдено.
Всі сценарії розширення успішно виконані.	

Таблиця 4.3 – Проходження тренування в режимі для одного користувача

Діючі особи	Користувач, система
Мета	Проходження тренування в режимі одного користувача
Передумова	Стабільний доступ до мережі інтернет
Успішний сценарій: 1. Користувач переходить на головну сторінку. 2. Користувач виконує налаштування генерації тексту для тренування. 3. Користувач починає вводити текст. 4. Користувач завершує тренування. 5. Користувача перенаправляє на сторінку з результатами. 6. Результати тренування відправляються на back-end сервер та зберігаються в базу даних.	

Кінець таблиці 4.3

Сценарій успішний. Тренування пройдено та результати збережено.	
Розширення	
1а	Користувач закриває вкладку браузеру з вебсайтом під час проходження тренування. Результат: тренування не завершено та результати не отримано.
2а	Користувач втрачає доступ до мережі Інтернет і результати тренування не відправляються на сервер. Результат: тренування не завершено та результати не отримано.
3а	Back-end сервер недоступний на момент відправки запиту про результати. Результат: дані про результати тренування не збережено.
Всі сценарії розширення успішно виконані.	

Таблиця 4.4 – Проходження тренування в багатокористувацькому режимі

Діючі особи	Користувачі, система
Мета	Проходження тренування у багатокористувацькому режимі
Передумова	Користувачі пройшли процес аутентифікації перед початком тренування
Успішний сценарій: 1. Користувач створює групу. 2. Він поширює код запрошення до групи з іншими користувачами. 3. Користувачі з якими було поширено код підключаються до групи. 4. Користувач виконує налаштування генерації тексту. 5. Користувач, що створив групу натискає кнопку «Start». 6. Користувач завершує тренування. 7. Користувач отримує результати тренування з місцем, яке він посів. 8. Результати тренування відправляються на back-end сервер та зберігаються в базу даних.	
Сценарій успішний. Тренування успішно пройдено та результати збережено в базі даних.	
Розширення	
1а	Хтось з користувачів, які мали доєднатись до групи не пройшов процес аутентифікації. Результат: тренування не було розпочато.

Кінець таблиці 4.4

2а	Користувач закриває вкладку браузеру з вебсайтом під час проходження тренування. Результат: тренування не завершено та результати не отримано.
3а	Користувач втрачає доступ до мережі Інтернет і результати тренування не відправляються на сервер. Результат: тренування не завершено та результати не отримано.
4а	Back-end сервер недоступний на момент відправки запиту про результати. Результат: дані про результати тренування не збережено.
Всі сценарії розширення успішно виконані.	

Таблиця 4.5 – Оновлення псевдоніму користувача

Діючі особи	Користувач, система
Мета	Оновлення псевдоніму користувача
Передумова	Користувач пройшов аутентифікацію у системі
Успішний сценарій: 1. Користувач переходить на сторінку профілю. 2. Користувач натискає кнопку «Edit profile». 3. Користувач вводить новий псевдонім у поле вводу. 4. Користувач натискає кнопку «Save». 5. Front-end застосунок відправляє запит на back-end сервер для збереження змін та отримує збережений псевдонім у відповідь.	
Сценарій успішний. Псевдонім користувача оновлено.	
Розширення	
1а	Користувач натиснув кнопку «Discard». Результат: все залишається без змін.
2а	Користувач натиснув кнопку «Save» не надавши новий псевдонім. Результат: виведено повідомлення про некоректність уведеного значення.
3а	Користувач втрачає доступ до мережі Інтернет до відправки запиту на збереження даних на сервер. Результат: псевдонім не оновлено.
4а	Back-end сервер недоступний на момент відправки запиту на зміну псевдоніму. Результат: псевдонім не оновлено.
Всі сценарії розширення успішно виконані.	

Таблиця 4.6 – Зміна теми застосунку

Діючі особи	Користувач, система
Мета	Зміна поточної теми користувацького інтерфейсу на бажану
Передумова	–
Успішний сценарій: 1. Користувач переходить на сторінку налаштувань. 2. Користувач натискає випадаюче меню з переліком назв тем. 3. Користувач обирає бажану тему шляхом натискання на назву. 4. Тема користувацького інтерфейсу та іконка сайту у вкладці браузеру змінюють свої кольори на відповідні.	
Сценарій успішний. Тему користувацького інтерфейсу змінено.	
Розширення	
1а	Під час завантаження ресурсів сайту не було завантажено додаткові іконку сайту та логотип. Результат: іконка сайту та логотип не відображаються правильно, але усі інші елементи інтерфейсу змінюють свій колір.
Всі сценарії розширення успішно виконані.	

Висновки до розділу 4

В четвертому розділі кваліфікаційної роботи бакалавра демонструються результати проведеної роботи з кодування. Наведено структуру front-end та back-end проєктів з роз’ясненнями щодо обраного підходу розподілу директорій та їх наповнення.

Представлено деталі реалізації основних частин розроблюваного функціоналу зі наведенням фрагментів програмного коду та скріншотів з відповідними поясненнями.

Проведено тестування системи. Базуючись на отриманих результатах оформлено опис основних функцій з успішними сценаріями та їх альтернативними неуспішними варіаціями. Завдяки даному аналізу системи знайдено дефекти, виправлено їх та оптимізовано деякі компоненти.

ВИСНОВКИ

В ході виконання кваліфікаційної роботи бакалавра було автоматизовано процес тренування швидкості вводу даних на клавіатурі шляхом розробки програмного забезпечення – вебзастосунку швидкості тренування вводу даних на клавіатурі.

Для досягнення визначеної на початку роботи мети виконано наступні завдання:

– проведено аналіз предметної області та аналогічних застосунків на ринку;

– знайдено потенційні проблеми та сформовано шляхи до їх вирішення;

– зазначено специфікацію вимог до розроблюваного програмного забезпечення;

– створено відповідні UML-діаграми для демонстрації алгоритмів роботи за структури системи;

– розроблено дизайн користувацького інтерфейсу;

– проведено кодування back-end частини системи за допомогою фреймворку ASP.NET Core та SQL Server;

– проведено кодування front-end частини системи з використанням фреймворку React.

Актуальність розробки даної системи та визначення її функціоналу було отримано шляхом проведення аналізу предметної області та застосунків-аналогів. Отримані дані щодо переваг та недоліків також допомогли у формуванні вимог до застосунку.

Згідно до поставленого технічного завдання обрано найбільш підходящий стек технологій для впровадження припустимого рівня комфорту користування системою та її подальшого супроводження.

В результаті отримано готову систему для проведення тренувань зі швидкості вводу даних на клавіатурі «typro».

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Helander M. G. Handbook of human-computer interaction. Elsevier, 2014. pp. 475–492 (last accessed 25.03.2023).

2. Liu X., Heo J., Sha L. Modeling 3-tiered web applications. 2005. pp. 1–7 (last accessed 27.03.2023).

3. Cherny B. Programming TypeScript: making your JavaScript applications scale. O'Reilly Media, 2019 pp. 1–7 (last accessed 31.03.2023).

4. Richards M. Software architecture patterns. O'Reilly Media, Incorporated 1005 Gravenstein Highway North, Sebastopol, CA~…, 2015 pp. 1–7 (last accessed 05.04.2023).

5. Lock A. ASP. NET core in Action. Simon and Schuster, 2021 pp. 65–66 (last accessed 05.04.2023).

6. Kronis K., Uhanova M. Performance Comparison of Java EE and ASP. NET Core Technologies for Web API Development. *Appl. Comput. Syst.* 2018. Vol. 23, № 1. pp. 37–44. (last accessed 15.04.2023).

7. 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 (last accessed 19.04.2023).

8. Staiano F. Designing and Prototyping Interfaces with Figma: Learn essential UX/UI design principles by creating interactive prototypes for mobile, tablet, and desktop. Packt Publishing Ltd, 2022. pp. 107–113 (last accessed 22.04.2023).

9. Holtzschue L. Understanding color: an introduction for designers. John Wiley & Sons, 2012. pp. 144–148 (last accessed 25.04.2023).

10. Ilić M., Kopanja L., Zlatković D., Trajković M., Ćurguz D. Microsoft sql server and oracle: Comparative performance analysis. 2021. pp. 33-38 (last accessed 28.04.2023).

11. Masse M. REST API design rulebook: designing consistent RESTful web service interfaces. « O'Reilly Media, Inc.», 2011. pp. 23-33 (last accessed 30.04.2023).

12. Albahari J. C# 10 in a Nutshell. « O'Reilly Media, Inc.», 2022. pp. 1–73 (last accessed 01.05.2023).

13. Sugrue J. Getting Started with UML. *Dzone Inc. Cary NC*. 2009. pp. 1–33 (last accessed 04.05.2023).

14. Bell D. UML basics: The sequence diagram. *Retrieved July*. 2004. Vol. 17. C. 2015. pp. 3–63 (last accessed 07.05.2023).

15. McGill M. J. UML Class Diagram Syntax: An Empirical Study of Comprehension. 2001. pp. 2–43 (last accessed 09.05.2023).

16. Gourley D., Totty B., Sayer M., Aggarwal A., Reddy S. HTTP: the definitive guide. « O'Reilly Media, Inc.», 2002. pp. 14-27 (last accessed 12.05.2023).

17. Minakata K., Beier S. The effect of font width on eye movements during reading. *Applied ergonomics*. 2021. Vol. 97. C. 103523. pp. 4–6 (last accessed 15.05.2023).

18. Lee J., Wei T., Mukhiya S. K. Redux Quick Start Guide: A beginner's guide to managing app state with Redux. Packt Publishing Ltd, 2019. pp. 31–35 (last accessed 16.05.2023).

19. GÜVERC\.IN 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. (last accessed 18.05.2023).

20. Pimentel V., Nickerson B. G. Communicating and displaying real-time data with websocket. *IEEE Internet Computing*. 2012. Vol. 16, № 4. pp. 45–53. (last accessed 25.05.2023).

ДОДАТОК А

Діаграма використання

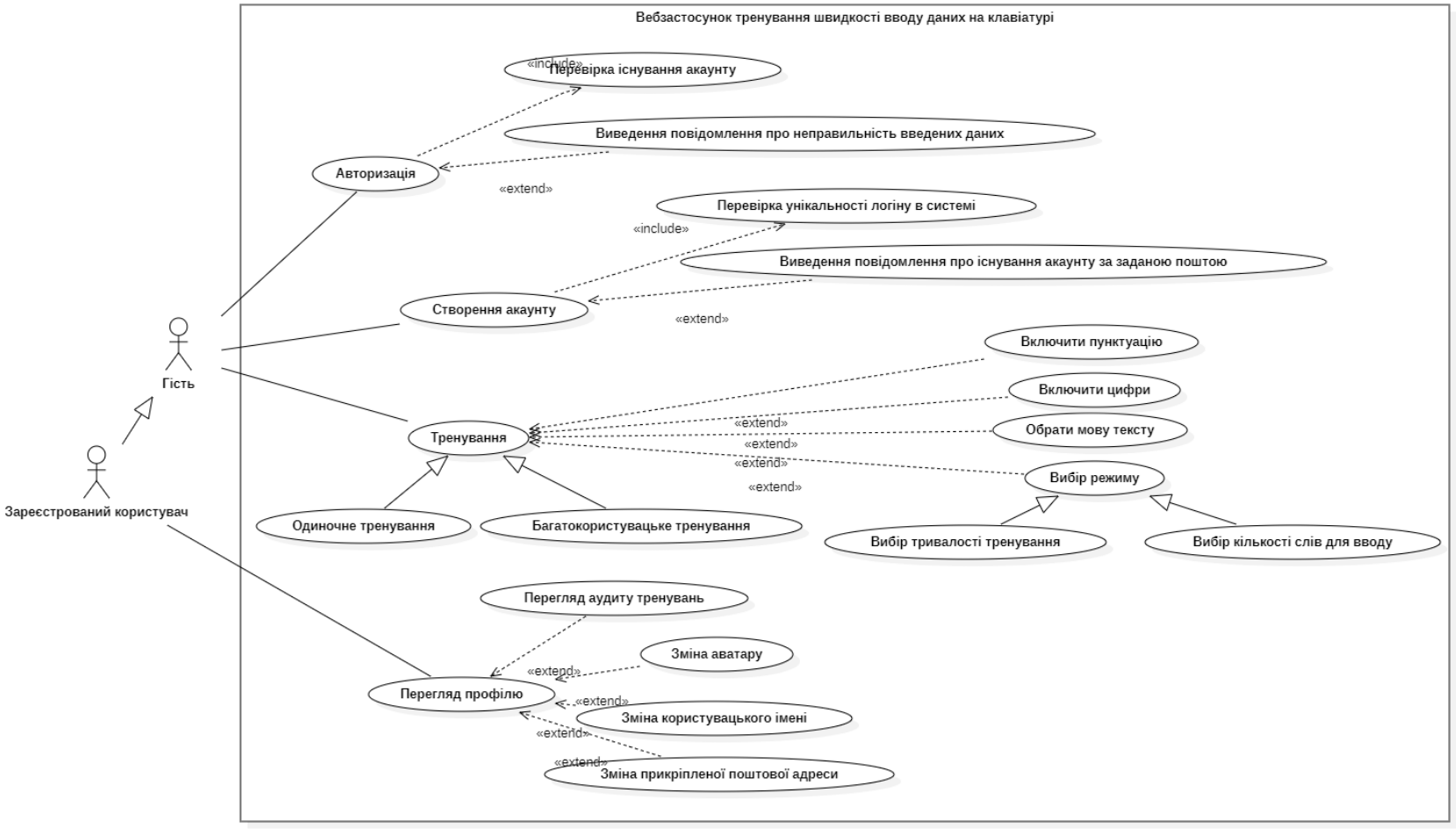

Рисунок А.1 – Діаграма використання

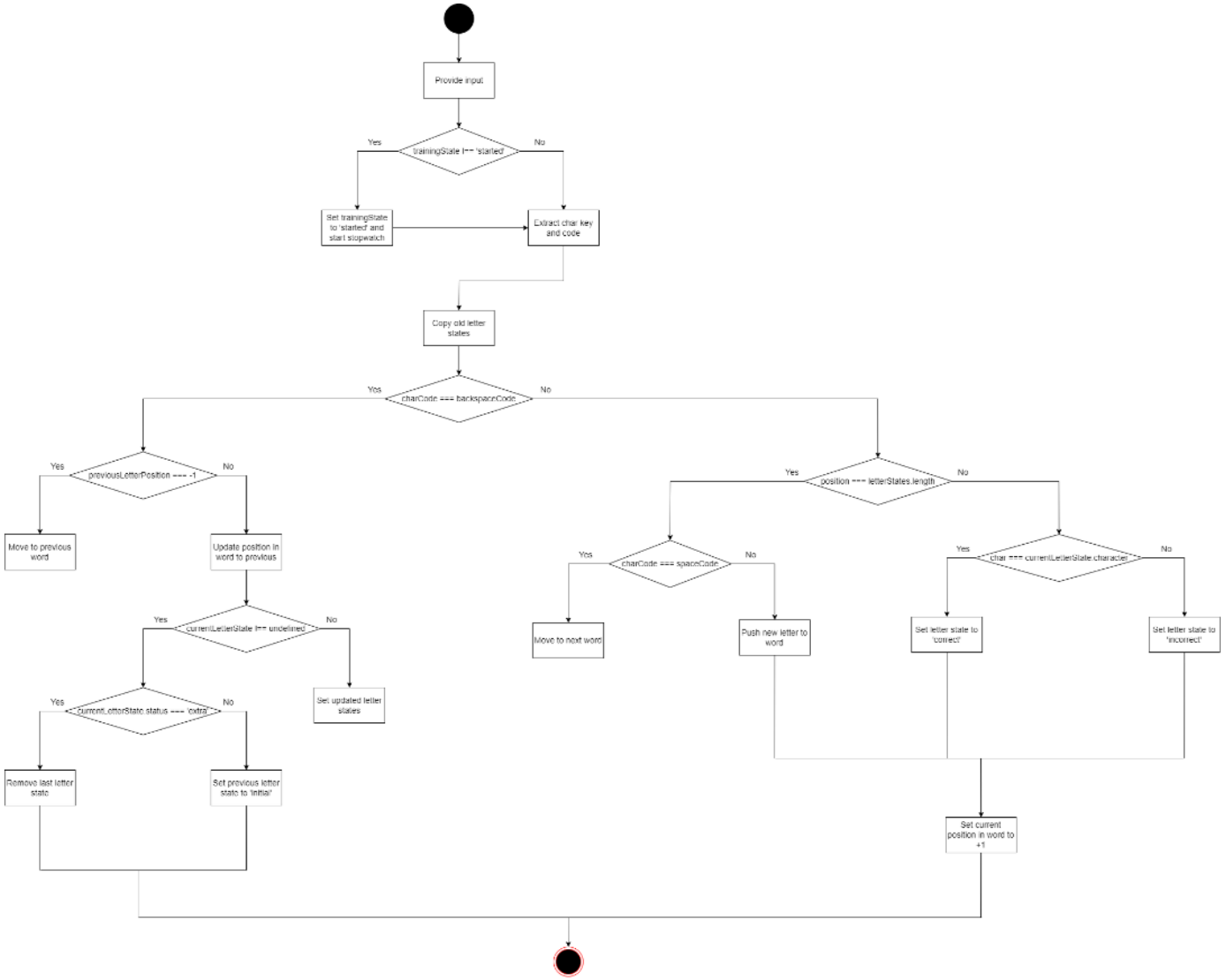

Рисунок А.2 – Повна діаграма діяльності для процесу обробки користувацького вводу під час тренування

ДОДАТОК Б

Програмний код обробника HTTP-запитів для генерації тексту

```
[HttpGet("text-generation")]
public async Task<IActionResult> GetGeneratedTextAsync([FromQuery]
GetGeneratedTextRequest request)
{
    var dto = new TrainingConfigurationDto(
        request.IsPunctuationGenerated,
        request.AreNumbersGenerated,
        request.WordsMode,
        request.TimeMode,
        request.LanguageId);

    Result<IEnumerable<string>> wordsResult = await
_textGenerationService.GenerateText(dto);
    if (wordsResult.IsFailed)
    {
        return wordsResult.ToActionResult();
    }

    Result<IEnumerable<IEnumerable<char>>> symbolsResult =
        _textGenerationService.ConvertWordsToSymbols(wordsResult.Value);

    return symbolsResult.ToActionResult();
}
```

ДОДАТОК В

Програмний код методу для генерації тексту

```
public async Task<Result<IEnumerable<string>>>
GenerateText(TrainingConfigurationDto dto)
{
    Result<IEnumerable<SupportedLanguageDto>> supportedLanguagesResult =
        await _supportedLanguagesService.GetSupportedLanguagesAsync();
    if (supportedLanguagesResult.IsFailed)
    {
        return Result.Fail(supportedLanguagesResult.Errors);
    }

    IEnumerable<SupportedLanguageDto>? supportedLanguages =
supportedLanguagesResult.Value;
    SupportedLanguageDto? targetLanguage = supportedLanguages.SingleOrDefault(e =>
e.Id == dto.LanguageId);
    if (targetLanguage is null)
    {
        return Result.Fail(new NotFoundError("Could not find specified
language."));
    }

    int numberOfWords;
    if (dto.WordsMode != WordsModeType.TurnedOff)
    {
        numberOfWords = (int)dto.WordsMode;
    }
    else
    {
        numberOfWords = 150;
    }

    Result<IEnumerable<Word>> wordsResult =
        await _wordsService.GetNRandomWordsByLanguageAsync(dto.LanguageId,
numberOfWords);

    List<string> words = wordsResult
        .Value
        .Select(w => w.Name)
        .ToList();

    var random = new Random();
    if (dto.AreNumbersEnabled)
    {
        for (var i = 0; i < words.Count; i++)
        {
            int index = random.Next(0, words.Count);
            bool shouldInsertNumber = random.Next(0, 10) is >= 3 and <= 6;
            if (!shouldInsertNumber)
            {
                continue;
            }

            int randomNumber = random.Next(0, 101);
            words[index] = randomNumber.ToString();
        }
    }
```

```
    if (dto.IsPunctuationEnabled)
    {
        var stringBuilder = new StringBuilder();
        for (var i = 0; i < words.Count; i++)
        {
            stringBuilder.Clear();

            bool shouldInsertPunctuation = random.Next(0, 10) is >= 0 and < 2 or
7;
            if (!shouldInsertPunctuation)
            {
                continue;
            }

            char punctuationSymbol = _punctuationSymbols[random.Next(0,
_punctuationSymbols.Length)];
            stringBuilder.Append(words[i]).Append(punctuationSymbol);
            words[i] = stringBuilder.ToString();
        }
    }

    return Result.Ok(words.AsEnumerable());
}
```

ДОДАТОК Г

Програмний код компонента для обробки користувацького вводу

```
const stopwatchTaskName = 'training';

export const GeneratedTextAreaContainer = (): JSX.Element => {
  const { isRestartScheduled, setRestartScheduledStatus } =
useContext(RestartContext);

  const dispatch = useAppDispatch();
  const trainingState = useAppSelector((store) => store.data.trainingState.state);
  const trainingConfiguration = useAppSelector((store) =>
store.data.trainingConfiguration);

  const [wordStates, setWordStates] = useState<WordProps[]>([]);

  const stopwatch = useStopwatch(stopwatchTaskName);
  useTrainingResults(stopwatch);

  const { data } = useQuery({
    queryKey: ['generatedText', trainingConfiguration],
    queryFn: async () => {
      const data = await trainingHttpClient.getGeneratedText({
        ...trainingConfiguration,
        languageId: ensure(trainingConfiguration.languagesInfo.find((e) =>
e.isActive)).id
      });

      await sleep(100);
      setRestartScheduledStatus(false);
      dispatch(trainingStateActions.setWordsTyped(0));
      dispatch(trainingStateActions.setState('initial'));

      return data;
    },
    enabled: isRestartScheduled && trainingConfiguration.languagesInfo.length > 0
  });

  const trainingStartHandler = (): void => {
    dispatch(trainingStateActions.setState('started'));
    if (trainingConfiguration.wordsMode !== WordsModeType.TurnedOff) {
      stopwatch.start();
    }
  };

  useEffect(() => {
    if (
      trainingState === 'finished' &&
      trainingConfiguration.wordsMode !== WordsModeType.TurnedOff
    ) {
      stopwatch.stop();
    }
  }, [trainingState]);

  const letterStatusesSubmissionHandler = (letterStatuses: LetterStatus[]): void
=> {
    if (trainingConfiguration.wordsMode !== WordsModeType.TurnedOff) {
      dispatch(trainingResultsActions.addLetterStatuses(letterStatuses));
    } else {
```

```
      const nonInitialLetterStatuses = letterStatuses.filter((e) => e !==
'initial');

dispatch(trainingResultsActions.addLetterStatuses(nonInitialLetterStatuses));
    }
  };

  const requestAdditionalWords = async (): Promise<void> => {
    const data = await trainingHttpClient.getGeneratedText({
      ...trainingConfiguration,
      languageId: ensure(trainingConfiguration.languagesInfo.find((e) =>
e.isActive)).id
    });

    setWordStates((prevStates) => {
      const newStates = data.map<WordProps>((wordChars, wordCharsIndex) => {
        return {
          letters: wordChars,
          isActive: false,
          isCounted: false,
          onMoveToAnotherWord: moveOnToAnotherWordHandler,
          onTrainingStart: trainingStartHandler,
          onWordModeTrainingEnd: letterStatusesSubmissionHandler
        };
      });
      return [...prevStates, ...newStates];
    });
  };

  const moveOnToAnotherWordHandler = (isForward: boolean): void => {
    setWordStates((oldStates) => {
      const activeWordIndex = oldStates.findIndex((s) => s.isActive);
      if (activeWordIndex === 0 && !isForward) {
        return oldStates;
      }

      if (isForward && trainingConfiguration.timeMode !== TimeModeType.TurnedOff)
{
        const shouldRequestWords = oldStates.filter((e) => !e.isCounted).length <
21;
        if (shouldRequestWords) {
          void requestAdditionalWords();
        }
      }

      if (activeWordIndex === oldStates.length - 1 && isForward) {
        dispatch(trainingStateActions.setState('finished'));
        return oldStates;
      }

      const newStates = [...oldStates];
      const newWordState = newStates[activeWordIndex];
      newWordState.isActive = false;
      if (isForward && !newWordState.isCounted) {
        newWordState.isCounted = true;
      }

      const wordToMoveOnToIndex = isForward ? activeWordIndex + 1 :
activeWordIndex - 1;
      newStates[wordToMoveOnToIndex].isActive = true;
```

```
      const numberOfCompletedWords = newStates.filter((s) => s.isCounted).length;
      dispatch(trainingStateActions.setWordsTyped(numberOfCompletedWords));

      return newStates;
    });
  };

  const generatedText: string[][] = data ?? [];
  useEffect(() => {
    if (generatedText.length > 0) {
      setWordStates(
        generatedText.map<WordProps>((wordChars, wordCharsIndex) => {
          return {
            letters: wordChars,
            isActive: wordCharsIndex === 0,
            isCounted: false,
            onMoveToAnotherWord: moveOnToAnotherWordHandler,
            onTrainingStart: trainingStartHandler,
            onWordModeTrainingEnd: letterStatusesSubmissionHandler
          };
        })
      );
    }
  }, [generatedText, setWordStates]);

  const words = wordStates.map((wordState, index) => <Word key={`word_${index}`}
{...wordState} />);

  return (
    <Fade in={!isRestartScheduled}>
      <Box>
        <GeneratedTextAreaFragment words={words} />
      </Box>
    </Fade>
  );
};
```

ДОДАТОК Е

Програмний код компонента, що відповідає за слова

```
export interface WordProps {
  isActive: boolean;
  isCounted: boolean;
  letters: string[];
  onMoveToAnotherWord: (isForward: boolean) => void;
  onTrainingStart: () => void;
  onWordModeTrainingEnd: (letterStatuses: LetterStatus[]) => void;
}

export const Word = (props: WordProps): JSX.Element => {
  const trainingState = useAppSelector((store) => store.data.trainingState.state);

  const [letterStates, setLetterStates] = useState<LetterProps[]>([]);
  const [position, setPosition] = useState(0);

  useEffect(() => {
    setLetterStates(
      props.letters.map<LetterProps>((l, i) => {
        return { status: 'initial', character: l, position: i };
      })
    );
    setPosition(0);
  }, [props.letters]);

  useEffect(() => {
    if (trainingState === 'finished') {
      const letterStatuses = letterStates.map((e) => e.status);
      props.onWordModeTrainingEnd(letterStatuses);
    }
  }, [trainingState, letterStates]);

  const keyDownHandler = (event: KeyboardEvent<HTMLDivElement>): void => {
    const char = event.key;
    const code = event.code;

    if (invalidCharCodes.find((e) => e === code) !== undefined) {
      return;
    }

    if (trainingState !== 'started') {
      props.onTrainingStart();
    }

    setLetterStates((oldStates) => {
      let newStates = [...oldStates];

      if (code === backspaceCode) {
        if (event.ctrlKey) {
          newStates = newStates.filter((e) => e.status !== 'extra');
          newStates.forEach((s) => (s.status = 'initial'));
          setPosition(0);
          return newStates;
        }

        const previousLetterPosition = position - 1;
```

```
        if (previousLetterPosition === -1) {
          props.onMoveToAnotherWord(false);
          return newStates;
        }

        setPosition(previousLetterPosition);
        const currentLetterState = newStates[previousLetterPosition];
        if (currentLetterState !== undefined) {
          if (currentLetterState.status === 'extra') {
            return newStates.slice(0, newStates.length - 1);
          }

          newStates[previousLetterPosition] = {
            character: currentLetterState.character,
            status: 'initial',
            position: currentLetterState.position
          };
        }

        return newStates;
      }

      if (position === letterStates.length) {
        if (code === spaceCode) {
          props.onMoveToAnotherWord(true);
          return newStates;
        } else {
          newStates.push({ character: char, position, status: 'extra' });
        }
      } else {
        const currentLetterState = newStates[position];
        const newLetterStatus: LetterStatus =
          char === currentLetterState.character ? 'correct' : 'incorrect';
        newStates[position] = {
          character: currentLetterState.character,
          status: newLetterStatus,
          position: currentLetterState.position
        };
      }

      setPosition(position + 1);

      return newStates;
    });
  };

  const letters = letterStates.map((letterState, index) => (
    <Letter
      key={`char_${index}`}
      character={letterState.character}
      status={letterState.status}
      position={letterState.position}
    />
  ));
  const lettersBox = (
    <Box sx={styles.word} tabIndex={-1} onKeyDown={keyDownHandler}>
      {letters}
    </Box>
  );
  return props.isActive ? <FocusLock>{lettersBox}</FocusLock> : lettersBox;
};
```

ДОДАТОК Ж

Програмний код компонента, що відповідає за символи у словах

```
import { SxProps, Typography, useTheme } from '@mui/material';
import { useMemo } from 'react';
import { createStyles } from './styles';
import { LetterStatus } from '../../../../../../types';

export interface LetterProps {
  position: number;
  character: string;
  status: LetterStatus;
}

export const Letter = (props: LetterProps): JSX.Element => {
  const theme = useTheme();
  const styles = useMemo(() => createStyles(theme), [theme]);

  const getStylesBasedOnState = (): SxProps => {
    if (props.status === 'correct') {
      return styles.correct;
    }
    if (props.status === 'incorrect') {
      return styles.incorrect;
    }
    if (props.status === 'extra') {
      return styles.extra;
    }
    return styles.initial;
  };

  return <Typography sx={{ ...getStylesBasedOnState()
}}>{props.character}</Typography>;
```