

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

## **ПОЯСНЮВАЛЬНА ЗАПИСКА**

до кваліфікаційної роботи освітнього ступеня «бакалавр»  
за спеціальністю 121 «Інженерія програмного забезпечення»  
(освітня програма «Інженерія програмного забезпечення»)  
на тему:  
«Застосунок для тренування швидкості друку»

Виконав студент групи ПЗ-21-1  
УКРАЇНЕЦЬ Дмитро Валентинович

Керівник:  
МОРОЗОВ Андрій Васильович

Рецензент:  
ЧИЖМОТРЯ Олена Геннадіївна

Житомир – 2025

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

ЗАТВЕРДЖУЮ  
Зав. кафедри інженерії  
програмного забезпечення  
Тетяна Вакалюк  
«26» лютого 2025 р.

**ЗАВДАННЯ**  
**на кваліфікаційну роботу**

Здобувач вищої освіти: Українець Дмитро Валентинович

Керівник роботи: Морозов Андрій Васильович

Тема роботи: «Застосунок для тренування швидкості друку»,  
затверджена Наказом закладу вищої освіти від 14.02.2025р., №61/с

Вихідні дані для роботи: процес розробки програмного засобу, що забезпечує можливість індивідуального тренування навичок сліпого набору тексту з метою підвищення швидкості та точності друку. Основну увагу приділено створенню зручного веб-інтерфейсу, реалізації системи збереження результатів, підтримці багатомовності та інтеграції елементів гейміфікації, таких як рейтинги, досягнення і турніри.

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

| Розділ | Консультант       | Завдання видав | Завдання прийняв |
|--------|-------------------|----------------|------------------|
| 1      | Андрій<br>МОРОЗОВ | 02.03.2025     | 02.03.2025       |
| 2      | Андрій<br>МОРОЗОВ | 02.03.2025     | 02.03.2025       |
| 3      | Андрій<br>МОРОЗОВ | 02.03.2025     | 02.03.2025       |

## РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з програмного застосунку для тренування швидкості друку та пояснювальної записки. Пояснювальна записка до роботи містить 73 сторінки, 26 ілюстрацій та 18 таблиць.

Метою роботи є розробка інтуїтивно зрозумілого та ефективного застосунку, який дозволить користувачам покращувати навички сліпого методу набору тексту, розвивати точність і збільшувати швидкість друку. Застосунок орієнтований на широке коло користувачів — від школярів до офісних працівників — і спрямований на підвищення загальної комп'ютерної грамотності.

Об'єктом дослідження є процес навчання та вдосконалення навичок швидкого друку за допомогою комп'ютерних технологій.

Предметом дослідження є методи реалізації інтерактивних тренувальних програм з використанням сучасних засобів розробки програмного забезпечення, а також принципи побудови ефективного навчального інтерфейсу.

У роботі застосовано методи об'єктно-орієнтованого програмування, веб-розробки, інтерфейсного дизайну та тестування програмного забезпечення. Особливу увагу приділено реалізації адаптивних тренувальних режимів, збору статистики та зворотному зв'язку для користувача з метою забезпечення ефективного навчального процесу.

**КЛЮЧОВІ СЛОВА:** ШВИДКІСТЬ ДРУКУ, НАВЧАЛЬНИЙ ЗАСТОСУНОК, СЛІПІЙ МЕТОД НАБОРУ, ІНТЕРАКТИВНЕ НАВЧАННЯ, ВЕБ-ТЕХНОЛОГІЇ.

|              |                |          |        |      |                                                                   |                                            |      |         |  |
|--------------|----------------|----------|--------|------|-------------------------------------------------------------------|--------------------------------------------|------|---------|--|
|              |                |          |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ                                          |                                            |      |         |  |
|              |                |          |        |      |                                                                   |                                            |      |         |  |
| Змн.         | Арк.           | № докум. | Підпис | Дата |                                                                   |                                            |      |         |  |
| Розроб.      | Українець Д.В. |          |        |      | Застосунок для тренування швидкості друку<br>Пояснювальна записка | Лім.                                       | Арк. | Аркушів |  |
| Керівник     | Морозов А.В.   |          |        |      |                                                                   |                                            | 3    | 73      |  |
|              |                |          |        |      |                                                                   | Житомирська політехніка,<br>група ІПЗ-21-3 |      |         |  |
| Рецензент    | Чижмoтря О.Г   |          |        |      |                                                                   |                                            |      |         |  |
| Зав. Кафедри | Вакалюк Т.А.   |          |        |      |                                                                   |                                            |      |         |  |

## ABSTRACT

The bachelor's final qualification project consists of a software application for typing speed training and an explanatory note. The explanatory note contains 76 pages, 26 illustrations, and 18 tables.

The aim of the work is to develop a user-friendly and efficient application that helps users improve their touch typing skills, increase typing speed, and enhance accuracy. The application is designed for a wide audience, including students, office workers, and anyone interested in boosting their typing proficiency, ultimately contributing to improved digital literacy and productivity.

The object of the research is the process of learning and improving typing speed through the use of digital tools.

The subject of the research is the implementation of modern programming technologies and user interface design principles to create an interactive training tool that provides personalized practice, performance feedback, and progress tracking.

This work applies methods of object-oriented programming, web development, user interface design, and software testing. Special attention is given to developing adaptive training modes, collecting user performance statistics, and ensuring an engaging and effective learning experience.

**KEYWORDS:** TYPING SPEED, TRAINING APPLICATION, TOUCH TYPING, INTERACTIVE LEARNING, WEB TECHNOLOGIES.

|      |      |                |        |      |                          |      |
|------|------|----------------|--------|------|--------------------------|------|
|      |      | Українець Д.В. |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      | Морозов А.В.   |        |      |                          |      |
| Змн. | Арк. | № докум.       | Підпис | Дата |                          | 4    |

## ЗМІСТ

|                                                                                                    |    |
|----------------------------------------------------------------------------------------------------|----|
| РЕФЕРАТ .....                                                                                      | 0  |
| ABSTRACT .....                                                                                     | 4  |
| ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ .....                                                                    | 6  |
| ВСТУП.....                                                                                         | 7  |
| РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ<br>ТЕХНОЛОГІЙ.....                           | 9  |
| 1.1 Постановка задачі .....                                                                        | 9  |
| 1.2. Аналіз аналогів програмного продукту.....                                                     | 10 |
| 1.3. Вибір архітектури веб-додатку.....                                                            | 15 |
| 1.4. Інструментальні засоби реалізації програмного продукту та вимоги до<br>апаратних засобів..... | 17 |
| Висновки до першого розділу .....                                                                  | 20 |
| РОЗДІЛ 2. ПРОЕКТУВАННЯ ЗАСТОСУНКУ ДЛЯ ТРЕНУВАННЯ<br>ШВИДКОСТІ ДРУКУ .....                          | 22 |
| 2.1 Визначення варіантів використання та структури застосунків .....                               | 22 |
| 2.2 Розробка бази даних системи .....                                                              | 29 |
| 2.3 Проектування та реалізація алгоритмів роботи системи .....                                     | 33 |
| 2.4 Реалізація функціоналу платформи застосунку для тренування швидкості<br>друку .....            | 36 |
| Висновок до другого розділу.....                                                                   | 44 |
| РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ ДОДАТКУ                                           | 45 |
| 3.1 Порядок встановлення та налаштування системи.....                                              | 45 |
| 3.2 Структура інтерфейсу та порядок взаємодії з застосунком для тренування<br>швидкості друку..... | 46 |
| 3.3 Тестування роботи програмного продукту .....                                                   | 59 |
| Висновки до третього розділу. ....                                                                 | 61 |
| ВИСНОВКИ .....                                                                                     | 62 |
| СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....                                                                    | 63 |
| ДОДАТКИ .....                                                                                      | 66 |

## ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – прикладний програмний інтерфейс

БД – база даних

ПЗ – програмне забезпечення

JS – JavaScript (мова програмування)

HTML – мова розмітки гіпертексту

CSS – каскадні таблиці стилів

UX – користувацький досвід

UI – користувацький інтерфейс

|      |      |                |        |      |                          |      |
|------|------|----------------|--------|------|--------------------------|------|
|      |      | Українець Д.В. |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      | Морозов А.В.   |        |      |                          |      |
| Змн. | Арк. | № докум.       | Підпис | Дата |                          | 6    |

## ВСТУП

**Актуальність теми.** У сучасному світі, де цифрові технології стають невід’ємною частиною повсякденного життя, навичка швидкого та ефективного друку є критично важливою для багатьох професій. Програмісти, копірайтери, секретарі та інші фахівці, які працюють з текстами, часто стикаються з необхідністю підвищити свою продуктивність за рахунок швидкості друку. Однак багато існуючих веб-додатків для тренування швидкості друку мають обмежений функціонал або застарілий інтерфейс, що ускладнює процес навчання. Вони часто не враховують індивідуальні потреби користувачів, не надають достатньо мотивації для регулярних тренувань і не мають сучасних механізмів аналітики прогресу. Створення веб-додатку, який поєднує інтерактивність, персоналізацію та зручний інтерфейс, є актуальним завданням для покращення ефективності навчання швидкості друку.

**Мета випускної роботи.** Розробка програмного застосунку для тренування швидкості друку, який забезпечить користувачам ефективний інструмент для вдосконалення навичок сліпого методу набору тексту, підвищення точності та збільшення швидкості друку. Застосунок орієнтований на широку аудиторію та спрямований на розвиток комп’ютерної грамотності й продуктивності користувачів шляхом створення інтерактивного середовища навчання.

Основні завдання включають:

1. Аналіз існуючих програм для тренування швидкості друку та виявлення їхніх недоліків.
2. Визначення вимог до функціональності, зручності та адаптивності застосунку.
3. Проєктування архітектури системи та вибір технологій для реалізації застосунку.
4. Розробка функціоналу для тренувань, обліку помилок, підрахунку результатів і відстеження прогресу.
5. Створення простого та інтуїтивного користувацького інтерфейсу.

|      |      |                |        |      |                          |      |
|------|------|----------------|--------|------|--------------------------|------|
|      |      | Українець Д.В. |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      | Морозов А.В.   |        |      |                          |      |
| Змн. | Арк. | № докум.       | Підпис | Дата |                          | 7    |

6. Проведення тестування застосунку з метою перевірки його стабільності, ефективності та відповідності поставленим вимогам.

Реалізація зазначених завдань дозволить створити зручний навчальний інструмент, який можна використовувати як у процесі самонавчання, так і в освітніх установах для формування навичок швидкого друку.

**Об'єкт дослідження.** Об'єктом дослідження є процес створення веб-додатку для тренування швидкості друку, який охоплює аналіз потреб користувачів, проектування інтерфейсів, розробку функціоналу та інтеграцію зі зовнішніми сервісами.

**Предмет дослідження.** Предметом дослідження є сучасні веб-технології, фреймворки та інструменти, необхідні для реалізації ефективного додатку для тренування швидкості друку, зокрема механізми гейміфікації, аналітики прогресу, персоналізації завдань та інші функціональні компоненти.

**Практичне значення одержаних результатів.** Практичне значення одержаних результатів полягає у створенні функціонального веб-додатку, який може бути впроваджений для масового використання користувачами. Реалізовані рішення сприятимуть покращенню навичок друку, підвищенню мотивації до навчання та забезпеченню зручного користувацького досвіду. Додаток також може бути використаний для освітніх цілей, наприклад, у школах або курсах комп'ютерної грамотності.

|      |      |                |        |      |                          |      |
|------|------|----------------|--------|------|--------------------------|------|
|      |      | Українець Д.В. |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      | Морозов А.В.   |        |      |                          |      |
| Змн. | Арк. | № докум.       | Підпис | Дата |                          | 8    |

# РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

## 1.1 Постановка задачі

Метою випускної кваліфікаційної роботи бакалавра є створення веб-додатку для тренування швидкості друку, який поєднує зручний інтерфейс, широкий функціонал та можливість ефективного навчання користувачів. Такий додаток покликаний задовольнити зростаючу потребу в розвитку навичок швидкого друку, що є важливою складовою комп'ютерної грамотності у сучасному світі. Ураховуючи, що багато людей не мають доступу до якісних та ефективних інструментів для тренування цієї навички, створення спеціалізованої платформи дозволить підвищити продуктивність користувачів, а також сприятиме загальному поліпшенню їх цифрової компетентності. Призначенням цього додатку є як самостійне навчання, так і використання в освітньому середовищі.

Платформа передбачає реалізацію декількох функціональних рівнів доступу. Для всіх користувачів передбачається можливість вибору тесту швидкості, перегляду лідерборду з результатами інших учасників, а також виконання вправ на основі підготовлених текстів. Зареєстровані користувачі отримують доступ до розширеного функціоналу: проходження тестів із подальшою генерацією детальної статистики, перегляд зведених даних про власну активність, участь у змаганнях із іншими учасниками системи та шанс потрапити до лідерборду. Для адміністраторів реалізовано окремий рівень керування — вони мають змогу додавати нові тексти та мови для тренувань, позначати результати, що викликають підозру, блокувати користувачів у разі виявлення порушень і надавати адміністративні права іншим учасникам.

Процес створення веб-додатку охоплює кілька ключових етапів. Насамперед здійснюється розробка концептуального дизайну, UI-елементів та вебінтерфейсів. Після цього формується архітектура клієнтської частини, яка включає вибір технологій та створення інтерфейсних компонентів. Далі будується серверна частина, для якої визначається набір технологій і

|      |      |                |        |      |                          |      |
|------|------|----------------|--------|------|--------------------------|------|
|      |      | Українець Д.В. |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      | Морозов А.В.   |        |      |                          |      |
| Змн. | Арк. | № докум.       | Підпис | Дата |                          | 9    |

створюється API для взаємодії з клієнтською стороною. Значну увагу приділено системі реєстрації, авторизації та ролей, що забезпечує контроль доступу та персоналізацію взаємодії. Основна функціональність платформи включає реалізацію механізмів проведення тестів швидкості друку, модулів для збору та візуалізації статистики, інтеграцію функціоналу для організації змагань та інтерфейсу для адміністрування.

Завершальним етапом є тестування: перевірка працездатності всіх компонентів, аналіз користувацького досвіду, а також удосконалення системи на основі отриманих результатів. У результаті створюється інноваційна веб-платформа, яка надає користувачам можливість покращити свої навички друку, оцінити власні результати, змагатися з іншими користувачами та брати участь у навчальному процесі, тим самим сприяючи підвищенню загального рівня цифрової грамотності.

## **1.2. Аналіз аналогів програмного продукту**

Для тренування швидкості друку існує велика кількість веб-додатків, які пропонують різні підходи до тренування цієї навички. Однак, кожен з цих додатків має свої особливості, переваги та недоліки, які впливають на ефективність навчання. Проведення детального аналізу існуючих аналогів є важливим кроком у процесі розробки власного веб-додатку, оскільки воно дозволяє виявити ключові функціональні вимоги та запропонувати користувачам оптимальний досвід.

Аналіз популярних платформ для тренування швидкості друку, таких як Monkeytype, 10FastFingers та TypeRacer, дає змогу зрозуміти, які механізми найбільше сприяють успіху таких додатків. Деякі з них, наприклад, фокусуються на мінімалістичному дизайну та широкому наборі режимів тренувань, тоді як інші надають перевагу соціальній взаємодії та змаганням у реальному часі. Кожен з цих підходів має свої сильні сторони, але також і обмеження, які можуть заважати користувачам досягти максимальних результатів. Тому важливо зрозуміти, які функції є найбільш затребуваними серед користувачів, а які можуть бути покращені чи доповнені у нашому майбутньому додатку.

|      |      |                |        |      |                          |      |
|------|------|----------------|--------|------|--------------------------|------|
|      |      | Українець Д.В. |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      | Морозов А.В.   |        |      |                          |      |
| Змн. | Арк. | № докум.       | Підпис | Дата |                          | 10   |

У цьому розділі ми детально розглянемо три провідні веб-додатки для тренування швидкості друку: Monkeytype, 10FastFingers та TypeRacer. Ми проаналізуємо їх основні функціональні можливості, переваги та недоліки, а також визначимо, які аспекти їхньої роботи можуть бути використані як основа для розробки нашого веб-додатку. Цей аналіз допоможе нам створити продукт, який поєднає усі переваги існуючих аналогів, усуне їх недоліки та запропонує користувачам унікальний та ефективний інструмент для тренування швидкості друку.

## 1. Monkeytype

Monkeytype — це сучасний веб-додаток для тренування швидкості друку, який вирізняється мінімалістичним дизайном та широким функціоналом. Його інтерфейс інтуїтивно зрозумілий і зручний для користувачів будь-якого рівня підготовки. Додаток надає можливість обирати серед різноманітних режимів тестування: з часовими обмеженнями (15 секунд, 30 секунд, 1 хвилина тощо), за кількістю слів (10, 20, 50 і більше), а також у вільному режимі без обмежень. Окрім цього, користувачі можуть переглядати детальну статистику, що включає графіки прогресу, показники точності та середню швидкість друку.

Серед головних переваг Monkeytype — лаконічний і зручний інтерфейс, велика кількість доступних режимів тестування, а також наявність розгорнутої статистики результатів, що дозволяє відстежувати особистий прогрес. Додаток також підтримує кастомізацію — користувачі можуть змінювати тексти для тренувань і обирати теми оформлення на власний смак.

Водночас Monkeytype має і певні обмеження. У ньому відсутня функція змагань із іншими користувачами в реальному часі, що знижує елемент гейміфікації. Крім того, платформа не передбачає взаємодії між користувачами, оскільки не містить спільноти, форумів або чатів. Також у додатку бракує мотиваційних механізмів — таких як досягнення, нагороди чи щоденні завдання — які могли б додатково стимулювати користувача до регулярних тренувань.

|      |      |                |        |      |                          |      |
|------|------|----------------|--------|------|--------------------------|------|
|      |      | Українець Д.В. |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      | Морозов А.В.   |        |      |                          |      |
| Змн. | Арк. | № докум.       | Підпис | Дата |                          | 11   |

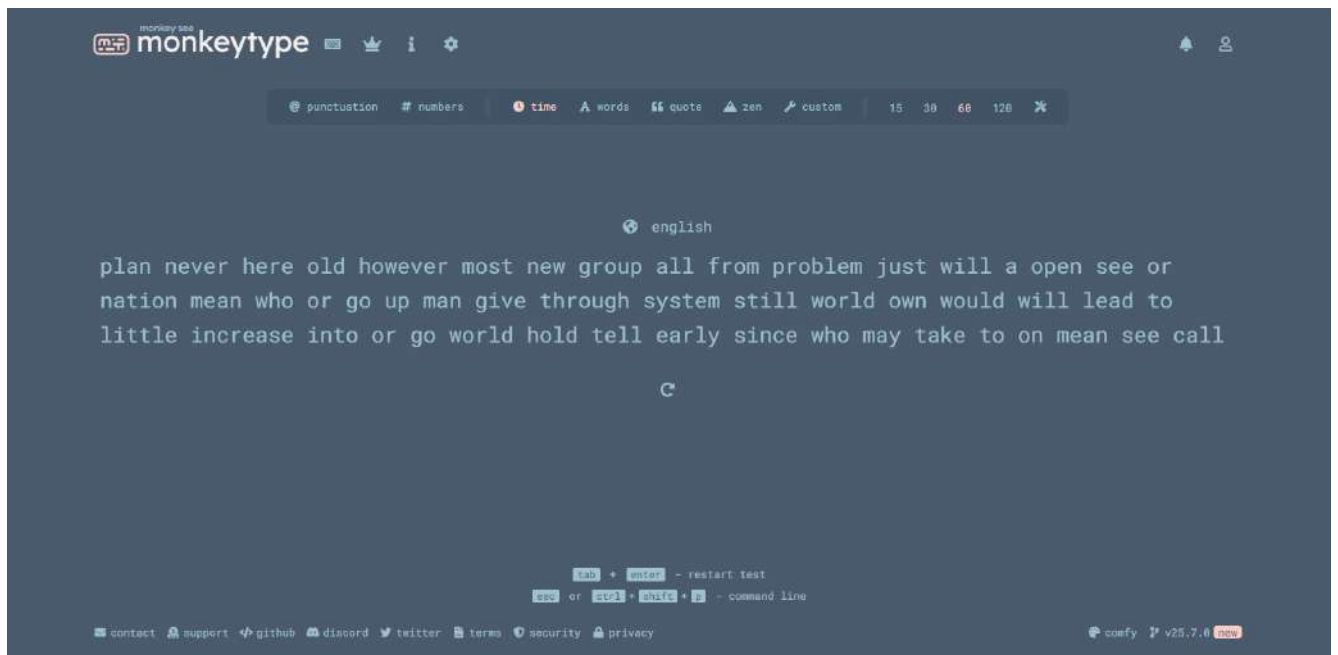


Рис. 1.1 — Інтерфейс Monkeytype

## 2. 10FastFingers

10FastFingers — це класичний веб-додаток для тренування швидкості друку, який вже тривалий час залишається популярним серед користувачів завдяки своїй простоті. Основна функція сервісу полягає у проведенні тестів друку тривалістю одну хвилину з можливістю порівняти свої результати з іншими учасниками через глобальний рейтинг. Додаток підтримує декілька мов, що робить його доступним для міжнародної аудиторії, і має інтуїтивно зрозумілий інтерфейс, з яким легко працювати навіть новачкам.

Серед його сильних сторін — простота використання, наявність глобального лідерборду та підтримка кількох мов, що дозволяє тренуватися в комфортному мовному середовищі. Проте, незважаючи на свою функціональність, платформа має і певні недоліки. Її дизайн виглядає застарілим і не відповідає сучасним вимогам до естетики та зручності. Функціонал досить обмежений — відсутні альтернативні режими тестування або налаштування рівня складності, а також бракує детальної аналітики результатів, що ускладнює відстеження прогресу користувача в динаміці.

|      |      |                |        |      |                          |      |
|------|------|----------------|--------|------|--------------------------|------|
|      |      | Українець Д.В. |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      | Морозов А.В.   |        |      |                          |      |
| Змн. | Арк. | № докум.       | Підпис | Дата |                          | 12   |

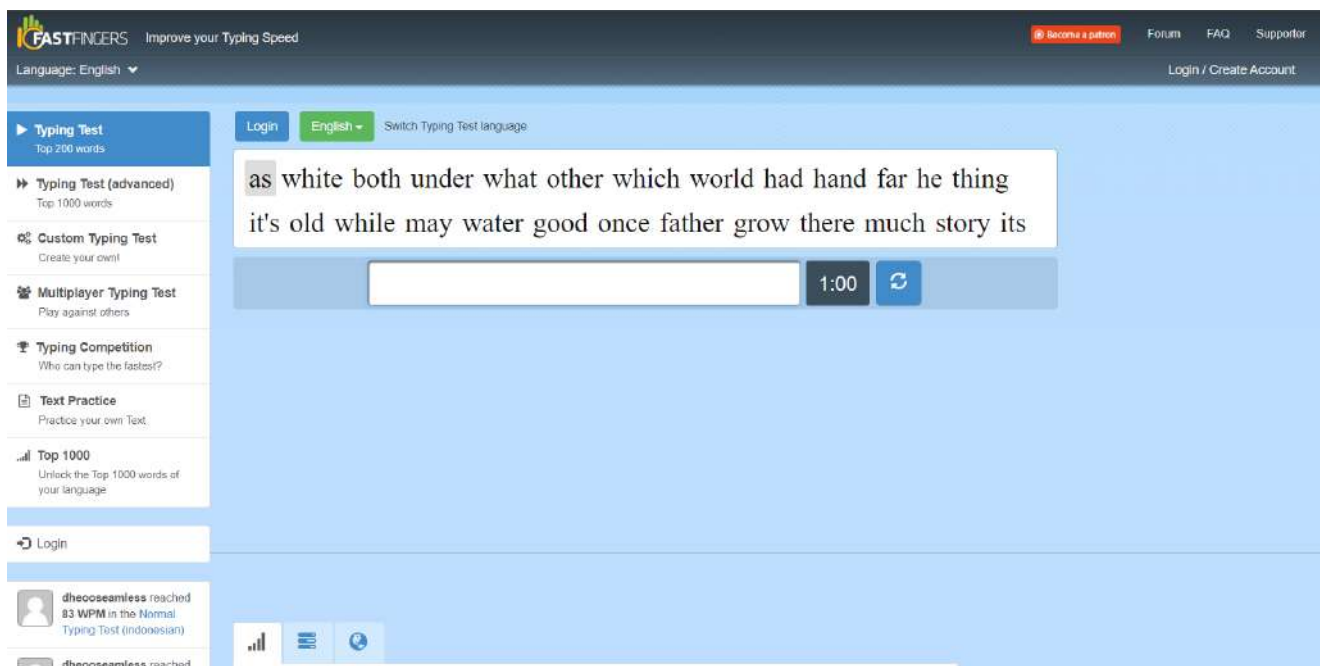


Рис. 1.2 — Інтерфейс 10FastFingers

### 3. TypeRacer

TypeRacer — це інтерактивний веб-додаток, орієнтований на змагання між користувачами у реальному часі. Його головною особливістю є механіка гонок, у яких учасники змагаються, намагаючись якомога швидше набрати заданий текст. Завдяки цій особливості платформа вирізняється динамічністю та азартом, залучаючи користувачів до регулярних тренувань через елемент суперництва. Додаток також передбачає соціальні можливості — користувачі можуть спілкуватися в чаті, створювати особисті кімнати для змагань із друзями та переглядати глобальні рейтинги, що формують відчуття спільноти та взаємодії.

Інтерфейс TypeRacer є інтерактивним і візуально привабливим, що робить процес друку схожим на гру. Утім, платформа має певні обмеження. Оскільки головний акцент зроблено на змагальність, це може створювати стрес для новачків, які лише починають тренуватися. Крім того, система не надає детальної статистики прогресу користувача, а кількість доступних режимів тренування залишається досить обмеженою. Незважаючи на це, TypeRacer залишається привабливим рішенням для тих, хто шукає драйвову альтернативу стандартним вправам на друк.

|      |      |                |        |      |                          |      |
|------|------|----------------|--------|------|--------------------------|------|
|      |      | Українець Д.В. |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      | Морозов А.В.   |        |      |                          |      |
| Змн. | Арк. | № докум.       | Підпис | Дата |                          | 13   |

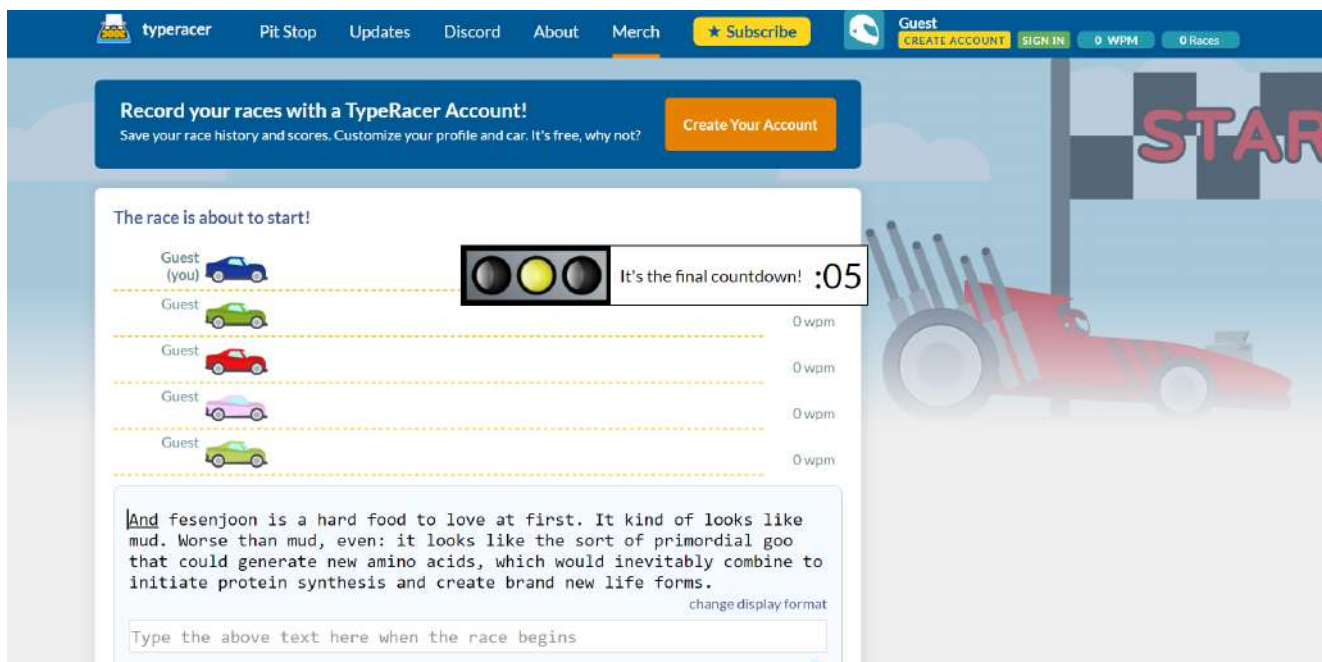


Рис. 1.3 — Інтерфейс TypeRacer

Аналіз аналогів показує, що кожен із наявних веб-додатків має свої сильні сторони, проте жоден із них не забезпечує повного набору функцій, необхідних для створення ідеального користувацького досвіду. Monkeytype відзначається найбільш розвиненим функціоналом для індивідуальних тренувань, однак у ньому повністю відсутні соціальні механізми, що обмежує можливість взаємодії між користувачами. У свою чергу, 10FastFingers є простим та доступним інструментом, проте його застарілий дизайн і обмежений набір функцій роблять його менш привабливим для сучасного користувача. TypeRacer орієнтований на змагання між користувачами в реальному часі, що є безперечною перевагою, однак такий підхід може викликати дискомфорт у новачків, а також не передбачає глибокої аналітики прогресу.

Враховуючи ці особливості, сформульовано основні вимоги до майбутнього веб-додатку. Насамперед, він має пропонувати різноманітні режими тренувань, зокрема за часом, кількістю слів або у вільному форматі. Важливо забезпечити користувачам доступ до детальної статистики прогресу, включаючи графіки, показники точності та швидкості. Значну роль відіграє також можливість змагань у реальному часі, що сприяє залученню користувачів і підтримці мотивації. Соціальні функції, такі як глобальний рейтинг, дозволяють відчувати

|      |      |                |        |      |                          |      |
|------|------|----------------|--------|------|--------------------------|------|
|      |      | Українець Д.В. |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      | Морозов А.В.   |        |      |                          |      |
| Змн. | Арк. | № докум.       | Підпис | Дата |                          | 14   |

себе частиною спільноти та порівнювати власні результати з іншими. Завершує перелік вимог сучасний дизайн, що відповідатиме актуальним стандартам юзабіліті та візуальної привабливості.

Ці вимоги стануть основою для розробки веб-додатку, який поєднає усі переваги існуючих аналогів, усуне їх недоліки та забезпечить користувачам комфортний та ефективний досвід навчання.

### 1.3. Вибір архітектури веб-додатку

Для реалізації сучасного веб-додатку з тренування швидкості друку обрано архітектуру на основі фреймворку Next.js [1] з використанням server actions, що дозволяє поєднувати переваги як серверної, так і клієнтської обробки даних. Такий підхід забезпечує високу продуктивність, інтерактивність, зручність підтримки та розширення додатку.

На відміну від традиційної серверно-орієнтованої архітектури (MPA), де кожна нова дія викликає повне перезавантаження сторінки, підхід із server actions у Next.js дозволяє виконувати бізнес-логіку на сервері без потреби в окремому REST API, передаючи запити напряму з клієнта через React-компоненти. Це значно спрощує архітектуру, зменшує кількість шарів взаємодії та підвищує швидкість обробки запитів.

Клієнтська частина побудована за принципами Single Page Application (SPA), що дає змогу реалізувати плавні переходи між сторінками, миттєве оновлення інтерфейсу та мінімізувати затримки в роботі з користувацькими подіями. Наприклад, результати тестування, графіки прогресу чи участь у змаганнях будуть оновлюватися без повного перезавантаження сторінки завдяки React-стану [13] та серверним екшенам, що викликаються безпосередньо з компонентів.

Форма взаємодії з базою даних реалізується через server actions, які напряму виконуються на сервері та мають доступ до Prisma ORM [6] для взаємодії з PostgreSQL [5] або іншою СУБД. Це дозволяє уникнути посередницького API-шару, зменшити кількість запитів до сервера та підвищити безпеку, оскільки обробка даних відбувається на сервері, а не на клієнті.

|      |      |                |        |      |                          |      |
|------|------|----------------|--------|------|--------------------------|------|
|      |      | Українець Д.В. |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      | Морозов А.В.   |        |      |                          |      |
| Змн. | Арк. | № докум.       | Підпис | Дата |                          | 15   |

Наприклад, під час збереження результатів тесту або реєстрації нового користувача виконується server action, який створює або оновлює відповідний запис у базі даних.

Крім того, Next.js підтримує гібридний рендеринг — як server-side rendering (SSR), так і static generation (SSG) [2], що важливо для забезпечення SEO-оптимізації. Головні публічні сторінки (наприклад, домашня сторінка або сторінка опису) можуть бути згенеровані наперед, тоді як приватні розділи (тестування, профіль, статистика) оброблятимуться динамічно на сервері, що забезпечує баланс між продуктивністю та актуальністю даних.

Інтерфейс реалізується з використанням Tailwind CSS [14] для створення сучасного, адаптивного та мінімалістичного дизайну, що підвищує привабливість продукту для користувачів. Для візуалізації статистики використовуються бібліотеки на кшталт Recharts або Chart.js, що дозволяє реалізувати графіки точності, прогресу та швидкості. Підтримка завантаження зображень реалізується через Next.js image-оптимізацію, що дозволяє додати до профілю користувача фотографію та забезпечити адаптивну обробку зображень з високою продуктивністю.

У результаті обраний підхід на основі Next.js з server actions дозволяє створити масштабований, безпечний і зручний для користувача веб-додаток, який відповідає сучасним стандартам розробки. Така архітектура забезпечує високу швидкість роботи, ефективну взаємодію з базою даних, простоту реалізації адміністративних функцій та оптимізовану підтримку SEO.

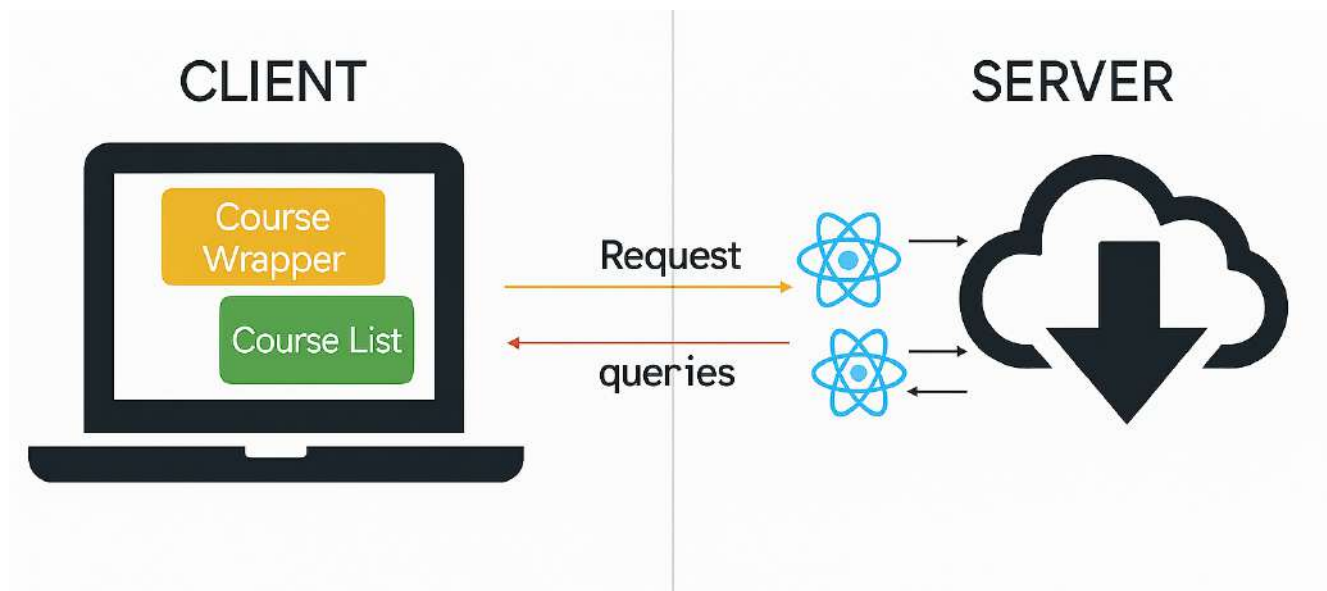


Рис.1.4 — Архітектура веб-додатку на основі Next.js із server actions

Таким чином, архітектурне рішення, засноване на Next.js із використанням server actions, є найбільш оптимальним для реалізації веб-додатку з тренування швидкості друку. Воно забезпечує сучасний підхід до розробки, який відповідає потребам як користувачів, так і розробників, дозволяючи створити швидкий, інтерактивний, масштабований і безпечний продукт з можливістю розширення функціональності в майбутньому.

#### 1.4. Інструментальні засоби реалізації програмного продукту та вимоги до апаратних засобів

Для розробки програмного продукту було обрано сучасний технологічний стек, орієнтований на забезпечення високої продуктивності, безпеки та гнучкого масштабування. Основною мовою програмування став TypeScript, який завдяки статичній типізації зменшує кількість помилок під час розробки, спрощує рефакторинг та інтегрується з такими фреймворками, як Next.js, Prisma та AI SDK.

Клієнтська частина реалізована на базі React 19 з використанням Next.js 15, що забезпечує оптимізований рендеринг, вбудовану маршрутизацію, підтримку SSR/SSG та Server Actions — сучасного підходу до реалізації серверної логіки без потреби створювати окремі API-ендпоінти. Для стилізації інтерфейсу застосовується Tailwind CSS, що дозволяє будувати адаптивний дизайн швидко

та зручно. Інтерфейс доповнюється за допомогою бібліотеки lucide-react, яка забезпечує сучасні іконки, що легко кастомізуються.

Аутентифікація та авторизація реалізовані за допомогою NextAuth.js [3] з адаптером Prisma, що підтримує автентифікацію через email, паролі та OAuth, а також надає зручну роботу із сесіями. Для збереження паролів використовується bcrypt, що гарантує безпечне хешування чутливих даних.

Для взаємодії з базою даних використовується ORM Prisma, що забезпечує типобезпечну взаємодію з реляційними базами даних. Як основну СУБД використано PostgreSQL, яка забезпечує транзакційність (ACID), підтримку складних SQL-запитів, високу надійність, індексацію, а також гнучке масштабування для великих обсягів інформації.

Також у застосунку використовується AI SDK [7], що інтегрує сучасні мовні моделі OpenAI та Google Gemini. Штучний інтелект застосовується виключно для генерації текстових завдань у процесі тренування, що дозволяє зробити контент більш різноманітним, динамічним та індивідуалізованим.

У процесі вибору СУБД було проведено порівняння декількох сучасних систем — PostgreSQL, MongoDB і MySQL — за ключовими критеріями: тип даних, транзакційна модель, підтримка запитів, масштабованість, продуктивність, рівень безпеки, інструменти адміністрування та зручність для розробників. Нижче наведено порівняльні таблиці.

Основні структурні особливості СУБД, що впливають на їхню придатність до різних типів застосунків, наведено в таблиці 1.1.

Таблиця 1.1.

| Критерій          | PostgreSQL | MongoDB                                                              | MySQL         |
|-------------------|------------|----------------------------------------------------------------------|---------------|
| Тип СУБД          | Реляційна  | Документоорієнтована                                                 | Реляційна     |
| Модель транзакцій | ACID       | ЧасткПроцесор:<br>Мінімум 2 ядра,<br>рекомендовано 4<br>ядрМІНдсутні | Підтримуються |

|               |                              |               |          |
|---------------|------------------------------|---------------|----------|
| Масштабування | Вертикальне та горизонтальне | Горизонтальне | Обмежене |
|---------------|------------------------------|---------------|----------|

Критичною для платформи є також швидкість пошуку та масштабованість рішення в умовах зростання кількості даних (табл. 1.2).

Таблиця 1.2.

| Критерій                 | PostgreSQL | MongoDB | MySQL   |
|--------------------------|------------|---------|---------|
| Обробка складних запитів | Висока     | Середня | Середня |
| Масштабованість          | Середня    | Висока  | Середня |
| Масштабованість          | Висока     | Висока  | Висока  |

У зв'язку з обробкою персональних даних користувачів важливим аспектом вибору СУБД є наявність механізмів контролю доступу та захисту інформації (табл. 1.3).

Таблиця 1.3.

| Критерій                    | PostgreSQL | MongoDB  | MySQL    |
|-----------------------------|------------|----------|----------|
| Аутентифікація              | Так        | Так      | Так      |
| Модель ролей                | Гнучка     | Обмежена | Базова   |
| Шифрування на рівні таблиць | Так        | Частково | Частково |

Із урахуванням обраної архітектури, функціонального навантаження платформи та перспективи масштабування, були сформульовані мінімальні та рекомендовані вимоги як до серверної частини, так і до клієнтських пристроїв. Ці вимоги дозволяють забезпечити стабільну та безпечну роботу системи, включно з обробкою запитів, роботою з базою даних, генерацією текстів і підтримкою багатокористувацького доступу.

Із урахуванням обраної архітектури, функціонального навантаження платформи та перспективи масштабування, були сформульовані мінімальні та рекомендовані вимоги як до серверної частини, так і до клієнтських пристроїв. Ці вимоги дозволяють забезпечити стабільну та безпечну роботу системи,

включно з обробкою запитів, роботою з базою даних, генерацією текстів і підтримкою багатокористувацького доступу.

1. Вимоги до сервера:

- Процесор: Мінімум 2 ядра, рекомендовано 4 ядра.
- Оперативна пам'ять: Мінімум 2 ГБ, рекомендовано 4 ГБ.
- Диск: SSD об'ємом мінімум 20 ГБ для зберігання бази даних та файлів додатку.
- Операційна система: Linux (Ubuntu, CentOS) або Windows Server.
- Мережеве з'єднання: Стабільне з'єднання з пропускнуою здатністю не менше 100 Мбіт/с.

2. Вимоги до клієнтських пристроїв:

- Операційна система: Windows, macOS, Linux або будь-яка мобільна платформа (iOS, Android).
- Браузер: Останні версії Google Chrome, Mozilla Firefox, Microsoft Edge або Safari.
- Оперативна пам'ять: Мінімум 1 ГБ для комфортної роботи з додатком.
- Інтернет-з'єднання: Стабільне з'єднання з пропускнуою здатністю не менше 5 Мбіт/с.

Вимоги до апаратних засобів враховують як поточні потреби додатку, так і можливість масштабування у майбутньому. Це дозволить підтримувати стабільну та ефективну роботу веб-застосунку навіть при значному зростанні кількості одночасних користувачів і обсягу даних.

### Висновки до першого розділу

У першому розділі було проведено комплексний аналіз передумов, засад та технічних рішень, необхідних для створення веб-додатку для тренування швидкості друку. Обґрунтовано актуальність теми дослідження, проаналізовано наявні аналоги й визначено основні функціональні вимоги до системи. Розглянуто сучасні архітектурні підходи до розробки веб-застосунків, у

|      |      |                |        |      |                          |      |
|------|------|----------------|--------|------|--------------------------|------|
|      |      | Українець Д.В. |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      | Морозов А.В.   |        |      |                          |      |
| Змн. | Арк. | № докум.       | Підпис | Дата |                          | 20   |

результаті чого було обрано архітектуру на основі фреймворку Next.js із Server Actions, що забезпечує високу продуктивність, інтегровану обробку серверних запитів та спрощену взаємодію між клієнтом і сервером.

Також було визначено доцільність використання технологій TypeScript, React, Tailwind CSS, NextAuth та Prisma, що дозволяють реалізувати надійну, масштабовану та зручну у підтримці платформу. Особливу увагу приділено аналізу систем управління базами даних — порівняльний аналіз показав перевагу PostgreSQL у контексті функціональності, безпеки, продуктивності та сумісності з обраним стеком. На завершення були сформульовані апаратні вимоги до клієнтських і серверних пристроїв, що враховують як поточне навантаження, так і можливості подальшого масштабування системи.

Таким чином, у результаті виконання першого розділу сформовано концептуальну та технічну основу для реалізації веб-додатку, що дозволяє перейти до проектування його структури та реалізації функціонального ядра.

|      |      |                |        |      |                          |      |
|------|------|----------------|--------|------|--------------------------|------|
|      |      | Українець Д.В. |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      | Морозов А.В.   |        |      |                          |      |
| Змн. | Арк. | № докум.       | Підпис | Дата |                          | 21   |

## РОЗДІЛ 2. ПРОЕКТУВАННЯ ЗАСТОСУНКУ ДЛЯ ТРЕНУВАННЯ ШВИДКОСТІ ДРУКУ

### 2.1 Визначення варіантів використання та структури застосунків

Застосунок для тренування швидкості друку надає вебінтерфейс, через який користувачі можуть проходити тести, переглядати лідерборд, тренуватися на наданих текстах та переглядати статистику результатів. У залежності від ролі користувача функціональність розширюється. У системі передбачено три основні категорії користувачів: гість, авторизований користувач та адміністратор. Гість має базовий доступ до тренувань і тестування. Авторизовані користувачі додатково можуть брати участь у змаганнях, переглядати детальну статистику та пропонувати власні тексти. Адміністратори відповідають за наповнення бази даних, модерацію результатів та керування користувачами. Для наочної демонстрації взаємодії між ролями користувачів та функціональними можливостями застосунку було створено діаграму варіантів використання (рис. 2.1).

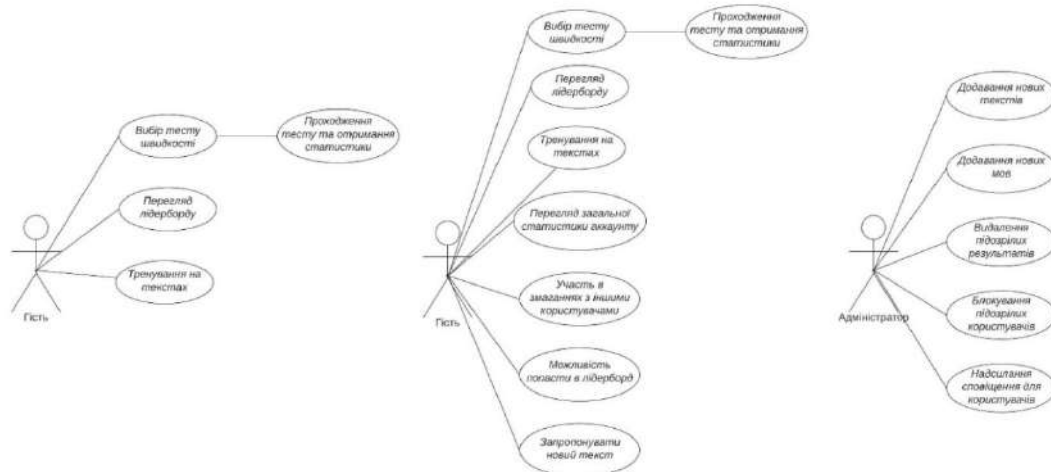


Рис 2.1 – Діаграма варіантів використання

Опис основних варіантів використання наведено нижче (табл. 2.1–2.10).

Таблиця 2.1

Опис прецеденту «Вибір тесту швидкості»

| Поле           | Опис                  |
|----------------|-----------------------|
| Назва варіанту | Вибір тесту швидкості |

Продовження таблиці 2.1

|                |                                                |
|----------------|------------------------------------------------|
| Основні актори | Гість, Користувач                              |
| Короткий опис  | Користувач обирає текст для проходження тесту. |

Таблиця 2.2

Опис прецеденту «Проходження тесту та отримання статистики»

|                |                                                                 |
|----------------|-----------------------------------------------------------------|
| Поле           | Опис                                                            |
| Назва варіанту | Проходження тесту та отримання статистики                       |
| Основні актори | Гість, Користувач                                               |
| Короткий опис  | Користувач проходить тест на швидкість друку та отримує оцінку. |

Таблиця 2.3

Опис прецеденту «Перегляд лідерборду»

|                |                                                 |
|----------------|-------------------------------------------------|
| Поле           | Опис                                            |
| Назва варіанту | Перегляд лідерборду                             |
| Основні актори | Гість, Користувач                               |
| Короткий опис  | Перегляд рейтингу користувачів за результатами. |

Таблиця 2.4

Опис прецеденту «Тренування на текстах»

|                |                                                     |
|----------------|-----------------------------------------------------|
| Поле           | Опис                                                |
| Назва варіанту | Тренування на текстах                               |
| Основні актори | Гість, Користувач                                   |
| Короткий опис  | Користувач тренується, не проходячи офіційний тест. |

Таблиця 2.5

Опис прецеденту «Перегляд загальної статистики»

|                |                               |
|----------------|-------------------------------|
| Поле           | Опис                          |
| Назва варіанту | Перегляд загальної статистики |

|                |                                                                   |
|----------------|-------------------------------------------------------------------|
| Основні актори | Користувач                                                        |
| Короткий опис  | Користувач переглядає свою історію тестів і динаміку результатів. |

Таблиця 2.6

## Опис прецеденту «Участь у змаганнях»

|                |                                                           |
|----------------|-----------------------------------------------------------|
| Поле           | Опис                                                      |
| Назва варіанту | Участь у змаганнях з іншими користувачами                 |
| Основні актори | Користувач                                                |
| Короткий опис  | Користувач бере участь у реальному змаганні на швидкість. |

Таблиця 2.7

## Опис прецеденту «Запропонувати новий текст»

|                |                                                               |
|----------------|---------------------------------------------------------------|
| Поле           | Опис                                                          |
| Назва варіанту | Запропонувати новий текст                                     |
| Основні актори | Користувач                                                    |
| Короткий опис  | Користувач надсилає власний текст для включення в базу даних. |

Таблиця 2.8

## Опис прецеденту «Додавання нових текстів»

|                |                                             |
|----------------|---------------------------------------------|
| Поле           | Опис                                        |
| Назва варіанту | Додавання нових текстів                     |
| Основні актори | Адміністратор                               |
| Короткий опис  | Адміністратор додає нові тексти до системи. |

Таблиця 2.9

## Опис прецеденту «Блокування порушників»

|                |                       |
|----------------|-----------------------|
| Поле           | Опис                  |
| Назва варіанту | Блокування порушників |

|                |                                                           |
|----------------|-----------------------------------------------------------|
| Основні актори | Адміністратор                                             |
| Короткий опис  | Адміністратор блокує користувачів, які порушують правила. |

Таблиця 2.10

## Опис прецеденту «Надсилення сповіщень»

|                |                                                                           |
|----------------|---------------------------------------------------------------------------|
| Поле           | Опис                                                                      |
| Назва варіанту | Надсилення сповіщень для користувачів                                     |
| Основні актори | Адміністратор                                                             |
| Короткий опис  | Адміністратор надсилає повідомлення про події, оновлення чи попередження. |

Застосунок має реалізовувати функціональні можливості відповідно до ролей користувачів: гість, авторизований користувач і адміністратор. Кожна з ролей має доступ до визначеного переліку дій, що відповідає її призначенню в системі. Основними функціональними вимогами є забезпечення доступу до тестів, обліку результатів, участі в змаганнях, адміністрування контенту та підтримки взаємодії з користувачами.

## Функціональні вимоги до застосунку

Застосунок повинен забезпечувати реалізацію функцій відповідно до ролей користувачів: гість, авторизований користувач і адміністратор. Основними функціональними вимогами є:

1. Реєстрація та авторизація користувачів із призначенням відповідної ролі.
2. Вхід у систему через особистий обліковий запис.
3. Створення та редагування профілю, включаючи зміну імені, аватару та мови інтерфейсу.
4. Вибір і проходження тестів, з фіксацією результатів.

|      |      |                |        |      |                          |      |
|------|------|----------------|--------|------|--------------------------|------|
|      |      | Українець Д.В. |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      | Морозов А.В.   |        |      |                          |      |
| Змн. | Арк. | № докум.       | Підпис | Дата |                          | 25   |

5. Перегляд власної статистики.
6. Подача заявок на участь у змаганнях, участь у реальному режимі з таймером і фіксацією результату.
7. Формування та перегляд лідерборду, з фільтрацією за періодом або категоріями.
8. Пропозиція користувачем власного тексту до бази через форму.
9. Додавання, редагування та видалення текстів адміністраторами.
10. Блокування користувачів, що порушують правила.
11. Видалення підозрілих результатів для підтримання чесної статистики.
12. Надсилання сповіщень адміністраторами до користувачів щодо оновлень або порушень.

#### Нефункціональні вимоги

##### 1. Зручність користування:

- Інтерфейс повинен бути інтуїтивно зрозумілим.
- Час навчання для новачків — до 2 годин, для досвідчених — до 1 години.

##### 2. Продуктивність:

- Час відповіді системи на типові дії не повинен перевищувати 2 секунд.
- Завантаження сторінки має тривати не більше 3 секунд при стандартному з'єднанні.

##### 3. Надійність:

- Доступність сервісу — не менше 98% робочого часу.
- Середній час безвідмовної роботи — щонайменше 30 днів.

##### 4. Масштабованість:

- Система повинна підтримувати одночасну роботу до 1000 активних користувачів без зниження продуктивності.
- Підтримка горизонтального масштабування серверної частини.

##### 5. Безпека:

|      |      |                |        |      |                          |      |
|------|------|----------------|--------|------|--------------------------|------|
|      |      | Українець Д.В. |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      | Морозов А.В.   |        |      |                          |      |
| Змн. | Арк. | № докум.       | Підпис | Дата |                          | 26   |

- Захищене зберігання паролів (хешування через bcrypt).
- Розмежування прав доступу за ролями.
- Щоденне резервне копіювання основних даних.

6. Адаптивність:

- Коректне відображення інтерфейсу на ПК, планшетах та мобільних пристроях.
- Підтримка актуальних версій браузерів: Chrome, Firefox, Safari, Edge.

Для розробки застосунку були використані такі технології: фреймворк Next.js для побудови клієнтської та серверної частини, база даних PostgreSQL для зберігання інформації про користувачів, результати тестів та тексти, а також Gemini AI API у зв’язці з AI SDK від Vercel для реалізації функції генерації навчальних текстів. Застосунок побудований на основі функціонального підходу, що дозволяє структурувати бізнес-логіку у вигляді окремих серверних функцій без використання класів або об’єктно-орієнтованих конструкцій.

Діаграма структури компонентів застосунку (рис. 2.2) демонструє основні функціональні блоки системи та потоки даних між ними. Вона відображає взаємодію між інтерфейсом користувача, серверними функціями, базою даних та штучним інтелектом, який генерує тексти для тренувань.

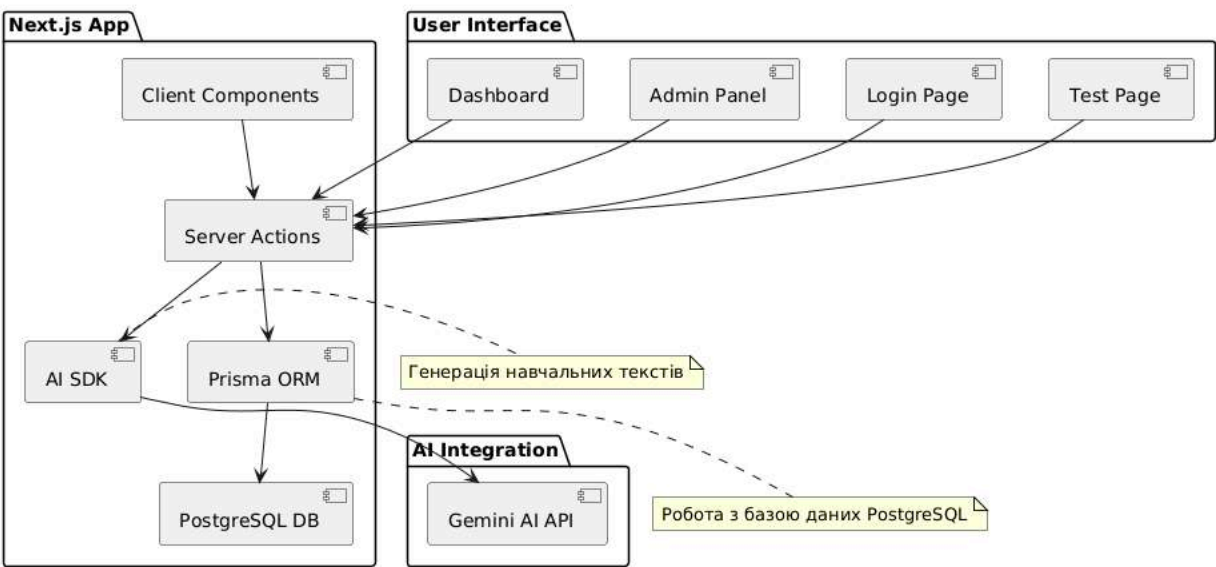


Рис. 2.2 – Діаграма структури компонентів застосунку

Функціональність застосунку охоплює кілька ключових напрямів, які забезпечують повноцінну роботу сервісу для тренування швидкості друку та аналізу результатів. Усі основні компоненти реалізовано в архітектурі Next.js із використанням серверних функцій, які взаємодіють із базою даних PostgreSQL і сторонніми AI-сервісами.

Клієнтська частина створена за допомогою компонентів Next.js і забезпечує інтерфейс користувача. Вона відповідає за відображення списку тестів, введення тексту, показ результатів, навігацію та сторінки профілю. Усі дії, які потребують збереження чи обробки даних, відправляються на сервер.

Серверні функції виконують основну бізнес-логіку: ініціалізацію тестів, фіксацію результатів користувача, підрахунок точності та швидкості, зберігання історії тренувань. Вони працюють безпосередньо з базою даних через систему доступу до даних.

База даних PostgreSQL є центральним сховищем даних про користувачів, їх сесії тренувань, результати тестів, тексти для друку. Система забезпечує надійну та структуровану роботу з даними.

Штучний інтелект Gemini інтегровано у систему для генерації навчальних текстів. Застосунок дозволяє адміністраторам або авторизованим користувачам надсилати запити на створення нового тексту, який формується на основі заданих критеріїв. Це дозволяє автоматично розширювати контентну базу.

Модуль статистики відповідає за обробку результатів тестів та формування аналітики: середня швидкість, кількість помилок, рівень прогресу. Статистику можна переглядати у вигляді числових значень або графіків, що допомагає користувачеві відстежувати власні успіхи.

Адміністративна панель — окремий функціональний модуль, доступний лише адміністраторам. Через неї можна керувати списком текстів, модерувати надіслані користувачами тексти, додавати нові мови інтерфейсу, контролювати підозрілі результати та блокувати порушників.

|      |      |                |        |      |                          |      |
|------|------|----------------|--------|------|--------------------------|------|
|      |      | Українець Д.В. |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      | Морозов А.В.   |        |      |                          |      |
| Змн. | Арк. | № докум.       | Підпис | Дата |                          | 28   |

Взаємодія між компонентами забезпечує безперервний обмін даними між клієнтом, сервером, базою даних та AI-сервісом. Це формує цілісну інфраструктуру для ефективного та зручного тренування друку.

## 2.2 Розробка бази даних системи

Провівши ретельний аналіз, було виявлено, що для збереження даних веб-додатку для тренування швидкості друку найкраще підходить реляційна система керування базами даних (СКБД) PostgreSQL. Ця база даних забезпечує надійне та ефективне зберігання інформації про користувачів, текстові матеріали, результати тестів та інші сутності системи.

База даних системи складається із наступних таблиць, наведених на діаграмі (рис. 2.7).

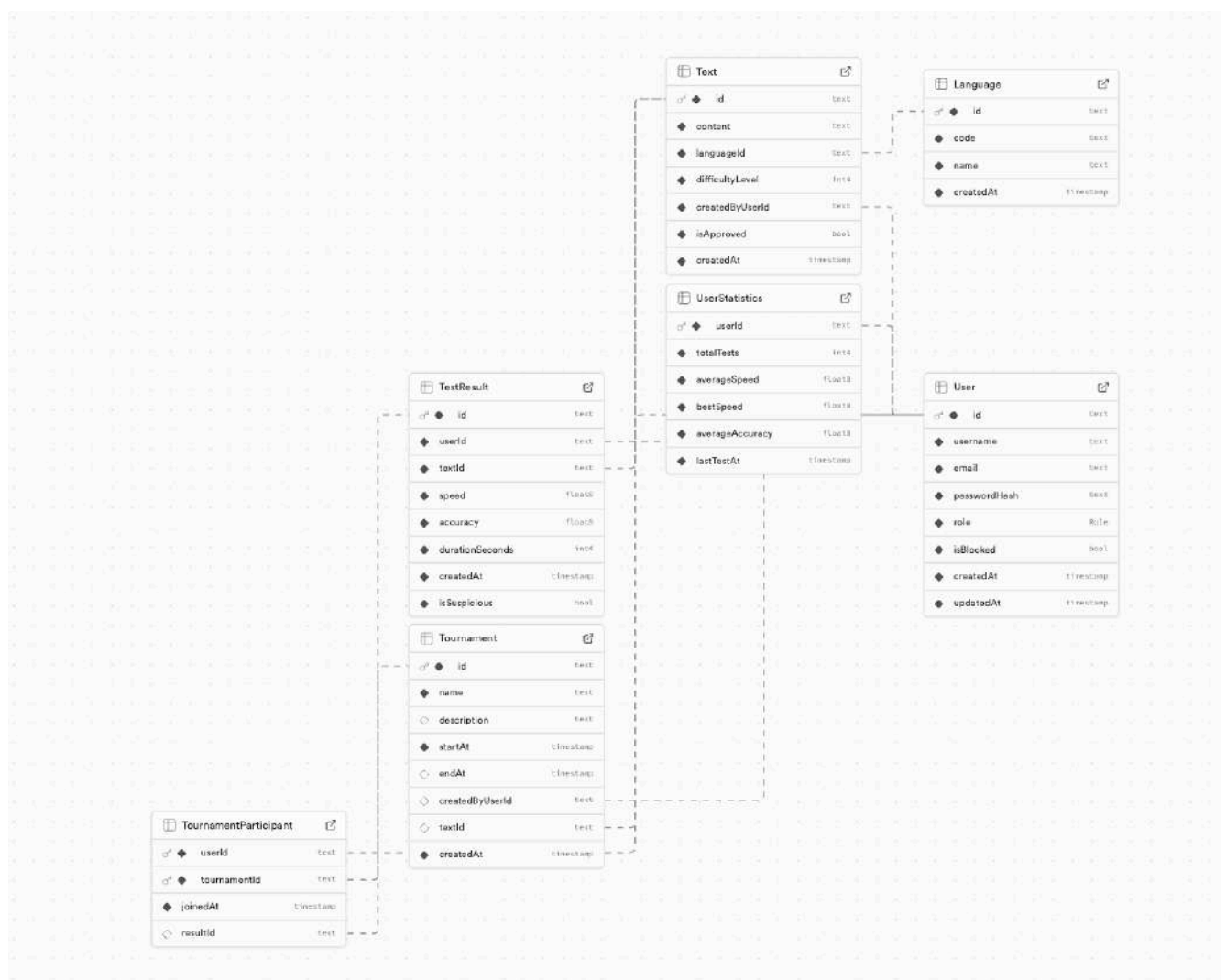


Рис 2.3. Діаграма бази даних

Таблиця User зберігає основну інформацію про користувачів системи. Поле id є первинним ключем, що генерується автоматично. Поля username та email є унікальними. Пароль зберігається у полі passwordHash у зашифрованому вигляді. Поле role вказує на тип користувача (звичайний або адміністратор), а isBlocked визначає, чи обмежено користувача. Поля createdAt та updatedAt фіксують час створення та останнього оновлення запису. Зв'язки включають відношення до результатів тестів, текстів, статистики та участі в турнірах.

Таблиця 2.11

Структура таблиці User

| Назва поля   | Тип даних | PK | FK | Опис                                 |
|--------------|-----------|----|----|--------------------------------------|
| id           | String    | +  | -  | Унікальний ідентифікатор користувача |
| username     | String    | -  | -  | Унікальне ім'я користувача           |
| email        | String    | -  | -  | Унікальна електронна адреса          |
| passwordHash | String    | -  | -  | Хешований пароль                     |
| role         | Role      | -  | -  | Роль користувача (USER, ADMIN)       |
| isBlocked    | Boolean   | -  | -  | Статус блокування користувача        |
| createdAt    | DateTime  | -  | -  | Дата створення                       |
| updatedAt    | DateTime  | -  | -  | Дата останнього оновлення            |

Таблиця Language використовується для збереження мов, якими доступні тексти. Кожна мова має унікальний код (code) і назву (name). Таблиця пов'язана з текстами, написаними цією мовою.

Таблиця 2.12

Структура таблиці Language

| Назва поля | Тип даних | PK | FK | Опис                          |
|------------|-----------|----|----|-------------------------------|
| id         | String    | +  | -  | Унікальний ідентифікатор мови |
| code       | String    | -  | -  | Код мови                      |
| name       | String    | -  | -  | Назва мови                    |
| createdAt  | DateTime  | -  | -  | Дата додавання мови           |

Таблиця Text зберігає навчальні тексти для тренування друку. Містить посилання на автора, мову, рівень складності, дату створення, а також прапорець затвердження адміністрацією.

Таблиця 2.13

Структура таблиці Text

| Назва поля      | Тип даних | PK | FK | Опис                            |
|-----------------|-----------|----|----|---------------------------------|
| id              | String    | +  | -  | Унікальний ідентифікатор тексту |
| content         | String    | -  | -  | Вміст тексту                    |
| languageId      | String    | -  | +  | Посилання на мову тексту        |
| difficultyLevel | Int       | -  | -  | Рівень складності тексту        |
| createdByUserId | String    | -  | +  | Автор тексту (користувач)       |
| isApproved      | Boolean   | -  | -  | Статус затвердження тексту      |
| createdAt       | DateTime  | -  | -  | Дата створення тексту           |

Таблиця TestResult зберігає результати проходження тестів. Містить посилання на текст, користувача, швидкість, точність, тривалість та відмітку про підозрілість.

Таблиця 2.14

Структура таблиці TestResult

| Назва поля      | Тип даних | PK | FK | Опис                                   |
|-----------------|-----------|----|----|----------------------------------------|
| id              | String    | +  | -  | Унікальний ідентифікатор результату    |
| userId          | String    | -  | +  | Посилання на користувача               |
| textId          | String    | -  | +  | Посилання на текст                     |
| speed           | Float     | -  | -  | Швидкість друку (знаків/хв)            |
| accuracy        | Float     | -  | -  | Точність (%)                           |
| durationSeconds | Int       | -  | -  | Тривалість проходження тесту (секунди) |
| createdAt       | DateTime  | -  | -  | Дата створення запису                  |
| isSuspicious    | Boolean   | -  | -  | Ознака підозрілого результату          |

Таблиця UserStatistics акумулює статистику користувача. Вона є зв'язаною один до одного з таблицею користувача.

Таблиця 2.15

Структура таблиці UserStatistics

| Назва поля      | Тип даних | PK | FK | Опис                                |
|-----------------|-----------|----|----|-------------------------------------|
| userId          | String    | +  | +  | Посилання на користувача            |
| totalTests      | Int       | -  | -  | Загальна кількість пройдених тестів |
| averageSpeed    | Float     | -  | -  | Середня швидкість                   |
| bestSpeed       | Float     | -  | -  | Найкраща швидкість                  |
| averageAccuracy | Float     | -  | -  | Середня точність                    |
| lastTestAt      | DateTime  | -  | -  | Дата останнього тестування          |

Таблиця Tournament містить інформацію про створені турніри — назву, опис, дати проведення, автора, текст для змагання.

Таблиця 2.16

Структура таблиці Tournament

| Назва поля      | Тип даних | PK | FK | Опис                                   |
|-----------------|-----------|----|----|----------------------------------------|
| id              | String    | +  | -  | Унікальний ідентифікатор турніру       |
| name            | String    | -  | -  | Назва турніру                          |
| description     | String?   | -  | -  | Опис турніру (необов'язково)           |
| startAt         | DateTime  | -  | -  | Дата початку                           |
| endAt           | DateTime? | -  | -  | Дата завершення (необов'язково)        |
| createdByUserId | String?   | -  | +  | Створено користувачем                  |
| textId          | String?   | -  | +  | Текст, який використовується у турнірі |
| createdAt       | DateTime  | -  | -  | Дата створення                         |

Таблиця TournamentParticipant зберігає зв'язки між користувачами та турнірами, а також результати участі.

Таблиця 2.17

Структура таблиці TournamentParticipant

| Назва поля   | Тип даних | PK | FK | Опис                     |
|--------------|-----------|----|----|--------------------------|
| userId       | String    | +  | +  | Посилання на користувача |
| tournamentId | String    | +  | +  | Посилання на турнір      |

|          |          |   |   |                                     |
|----------|----------|---|---|-------------------------------------|
| joinedAt | DateTime | - | - | Дата приєднання до турніру          |
| resultId | String?  | - | + | Посилання на результат (якщо вже є) |

Використовуючи такі таблиці, як User, Language, Text та TestResult, було побудовано логіку для зберігання основної інформації про користувачів, навчальні тексти та результати проходження тестів, із розподіленням доступу відповідно до ролей користувачів. Таблиці UserStatistics та Tournament забезпечують збереження аналітичних даних і реалізацію механізму подій, у яких користувачі можуть брати участь. Таблиця TournamentParticipant виконує роль зв'язуючої ланки між учасниками та турнірами. Завдяки чітко визначеним зв'язкам між сутностями та використанню Prisma ORM, реалізовано цілісну і масштабовану модель даних, що відповідає функціональним потребам веб-застосунку.

### 2.3 Проєктування та реалізація алгоритмів роботи системи

Для реалізації веб-застосунку для тренування швидкості друку було обрано клієнт-серверну архітектуру з використанням сучасного фреймворку Next.js, що забезпечує ефективний розподіл обчислювального навантаження між клієнтською та серверною частинами. Такий підхід дозволяє створювати високопродуктивні динамічні інтерфейси, підтримує серверний рендеринг і статичну генерацію сторінок, що позитивно впливає на швидкодію застосунку та комфорт користувача.

Ключовим технічним рішенням стало впровадження Server Actions — функціональності, яка дає змогу викликати серверні обробники безпосередньо з клієнтських компонентів. Це дозволило значно спростити взаємодію між інтерфейсом та серверною логікою, уникаючи потреби в окремих API-ендпоінтах і скорочуючи кількість проміжних запитів. В обробку даних були інтегровані обчислення швидкості, точності, тривалості виконання вправ, а також збереження результатів у базу даних безпосередньо з серверної частини, що забезпечило стабільність, безпеку та цілісність інформації.

Уся логіка реалізована з використанням Prisma ORM, яка забезпечує типізовану взаємодію з PostgreSQL-базою даних. Це дозволило легко реалізувати алгоритми фільтрації результатів, обчислення середніх значень, формування статистики користувача та підрахунок рейтингу учасників. Алгоритми взаємодії з базою даних побудовані таким чином, щоб зменшити навантаження на сервер і забезпечити швидку відповідь навіть при великій кількості одночасних користувачів.

Загалом застосування клієнт-серверної архітектури, сучасних інструментів Next.js і Prisma, а також акцент на обробку логіки на сервері дозволили побудувати надійний, безпечний і масштабований веб-застосунок для онлайн-тренування швидкості друку.

Загальна логіка роботи застосунку визначається статусом авторизації та роллю користувача. Якщо користувач не авторизований, він має доступ лише до перегляду рейтингів, проходження тестів швидкості та тренування на доступних текстах. Після авторизації користувач отримує розширений функціонал: зокрема, можливість переглядати персональну статистику, брати участь у турнірах і надсилати власні тексти. Створення турнірів дозволено виключно адміністраторам, що забезпечує контроль за їх наповненням і структурою. Крім того, адміністратори можуть модерувати тексти та користувачів, керувати мовами інтерфейсу й блокувати порушників. Уся обробка даних виконується на сервері за допомогою Server Actions, що забезпечує безпеку та спрощує архітектуру. Для взаємодії з базою даних використовується Prisma ORM, яка дозволяє ефективно працювати з усіма сутностями системи. Загальну схему логіки застосунку наведено на рисунку 2.4.

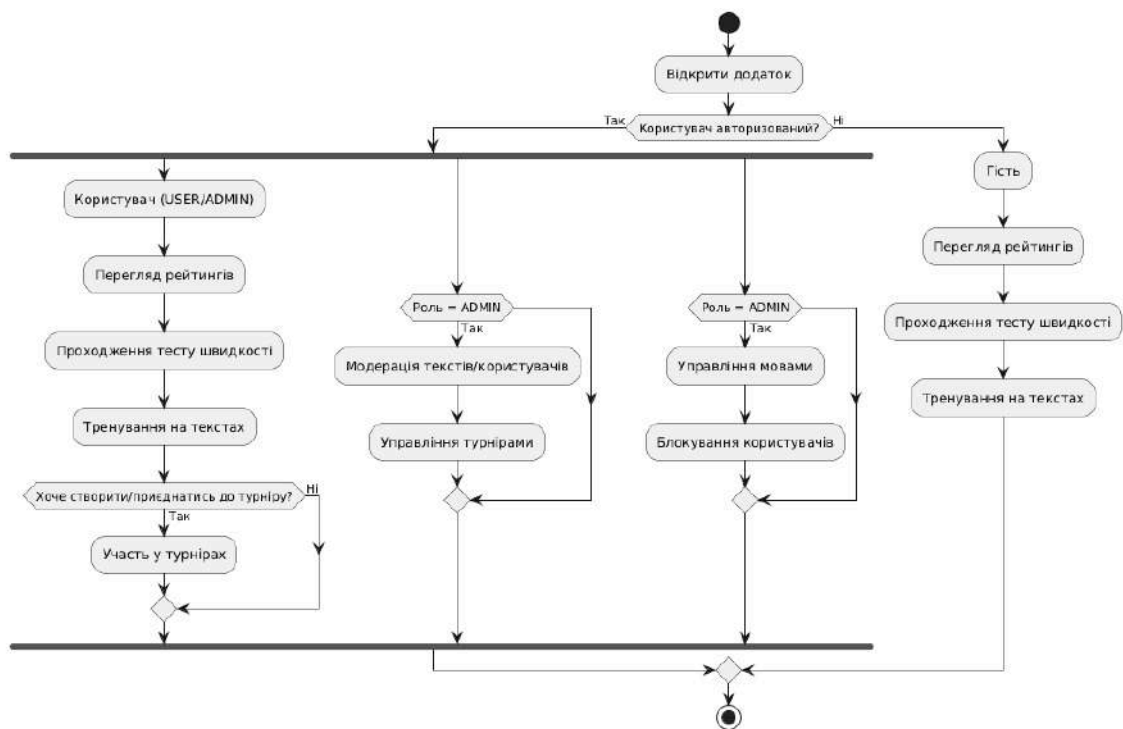


Рис. 2.4 – Загальна логіка роботи веб-застосунку для тренування швидкості друку

Склавши загальний алгоритм роботи застосунку, доцільно також деталізувати окремі ключові процеси за допомогою діаграм діяльності. Такі діаграми дозволяють наочно представити логіку взаємодії користувача з системою, а також внутрішні серверні процеси, що значно полегшує реалізацію, тестування й подальшу підтримку функціоналу. Один із центральних сценаріїв — процес проходження тесту на швидкість друку, який поєднує взаємодію з клієнтським інтерфейсом, обробку даних на сервері та збереження результатів.

Процес починається з відкриття користувачем сторінки тестування та натискання кнопки «Почати тест». У відповідь система генерує випадковий текст і відображає його для набору. Після цього користувач починає вводити текст, а система фіксує час початку. По завершенню введення або натисканні кнопки «Завершити тест», система зупиняє таймер, підраховує швидкість (у знаках на хвилину) та точність (відсоток правильних символів). Якщо користувач авторизований, результат зберігається до бази даних і враховується у загальній статистиці. Після обробки результату користувач отримує можливість переглянути свої показники. Уся логіка забезпечує коректну фіксацію

результатів і гарантує об'єктивність оцінювання. Візуалізацію процесу наведено на рисунку 2.5.

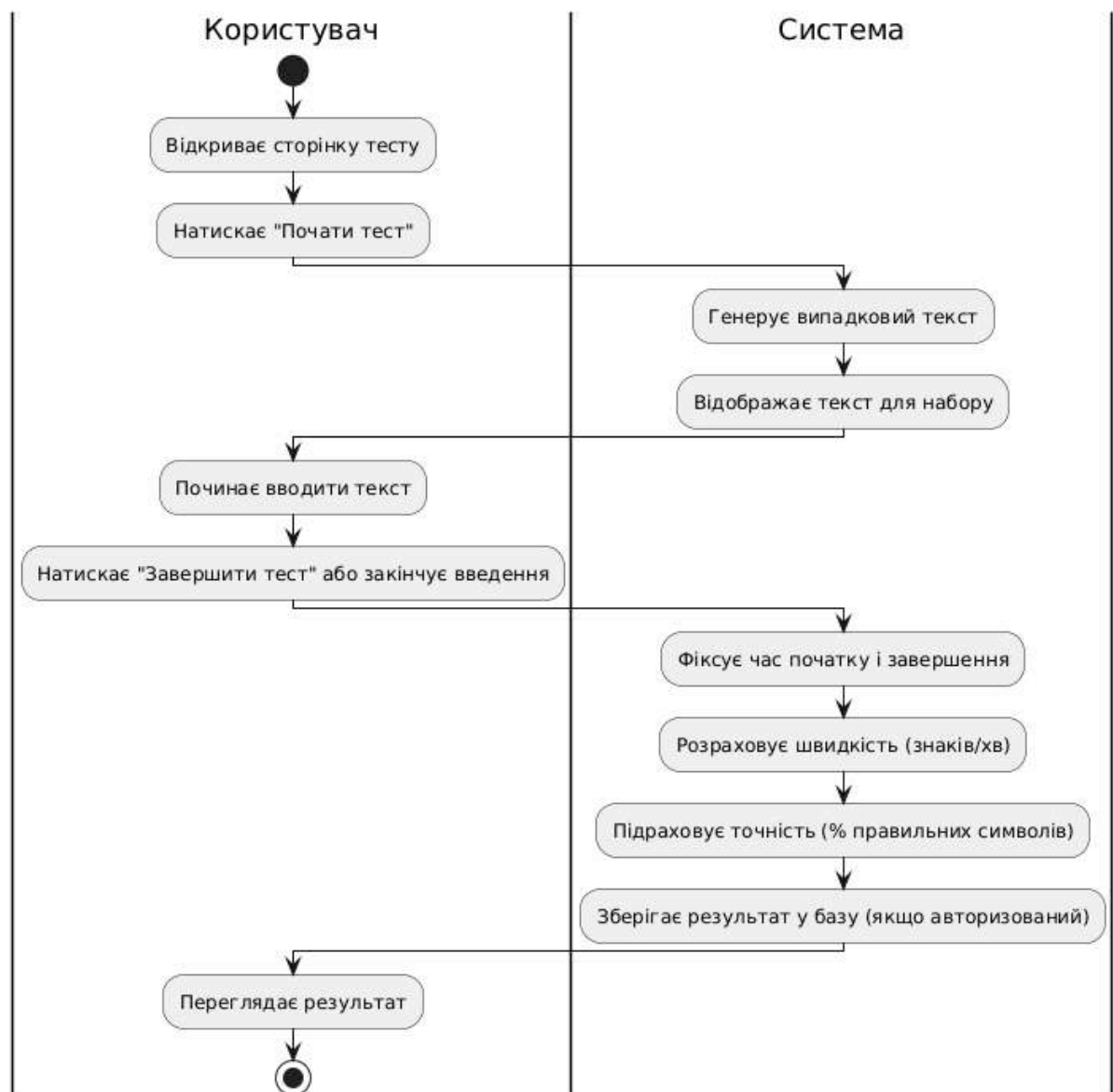


Рис.2.5 – Діаграма діяльності процесу проходження тесту на швидкість друку

## 2.4 Реалізація функціоналу платформи застосунку для тренування швидкості друку

Застосунок складається з клієнтської частини, серверної логіки та адміністративного інтерфейсу, всі елементи якого реалізовані в одному проєкті на основі фреймворку Next.js. Такий підхід дозволяє забезпечити єдність

структури, спрощує маршрутизацію та забезпечує ефективну інтеграцію між різними модулями системи.

Серверна логіка побудована за допомогою Server Actions — спеціального механізму Next.js, що дозволяє викликати функції на сервері безпосередньо з клієнтського коду. Це спрощує архітектуру проєкту, зменшує потребу в окремих REST- або GraphQL-інтерфейсах і підвищує безпеку, оскільки критичні операції виконуються у контрольованому середовищі.

Вхідними точками додатку є файли компонування layout.tsx, які визначають структуру інтерфейсу, та серверні функції, позначені директивою "use server", які реалізують логіку збереження результатів, обробки статистики, роботи з текстами та перевірки прав доступу.

Для взаємодії з базою даних використовується PostgreSQL у поєднанні з ORM Prisma. Ініціалізація клієнта Prisma реалізована з використанням патерна сінглтон, що дозволяє уникнути створення кількох з'єднань до бази даних під час розробки. Це особливо важливо для середовищ із підтримкою гарячого перезапуску, таких як Next.js. Код реалізації сінглтона наведено нижче:

```
import { PrismaClient } from "@generated/prisma";

const globalForPrisma = global as unknown as { prisma: PrismaClient };
export const prisma = globalForPrisma.prisma || new PrismaClient();
if (process.env.NODE_ENV !== "production") globalForPrisma.prisma = prisma;
```

Цей механізм забезпечує повторне використання одного екземпляра PrismaClient у всьому застосунку, що дозволяє знизити навантаження на систему та уникнути конфліктів при багатьох викликах до бази даних. Такий підхід гарантує стабільність і ефективність роботи серверної частини застосунку.

Для забезпечення безпечного доступу до захищених сторінок веб-застосунку використовується middleware — проміжна функція, яка виконується до завантаження сторінки. Вона перевіряє наявність токена авторизації, збереженого після входу користувача. Якщо токен відсутній, користувач

|      |      |                |        |      |                          |      |
|------|------|----------------|--------|------|--------------------------|------|
|      |      | Українець Д.В. |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      | Морозов А.В.   |        |      |                          |      |
| Змн. | Арк. | № докум.       | Підпис | Дата |                          | 37   |

перенаправляється на сторінку входу з параметром повернення, що дозволяє після авторизації повернутись до запитуваної сторінки. У реалізації також передбачено перевірку прав доступу. Якщо користувач намагається потрапити до маршруту з префіксом /admin, але не має ролі адміністратора, він автоматично перенаправляється на головну сторінку. Це дозволяє обмежити доступ до конфіденційного функціоналу лише уповноваженим особам. Список маршрутів, до яких застосовується перевірка, вказано в параметрі matcher, що охоплює адміністративну панель, особистий кабінет та сторінку результатів. Нижче наведено код реалізації middleware:

```
import { NextResponse } from "next/server";
import type { NextRequest } from "next/server";
import { getToken } from "next-auth/jwt";

export const config = {
  matcher: ["/admin/:path*", "/dashboard/:path*", "/results/:path*"],
};

export async function middleware(req: NextRequest) {
  const token = await getToken({ req, secret: process.env.NEXTAUTH_SECRET });
  const { pathname } = req.nextUrl;

  if (!token) {
    const loginUrl = new URL("/auth/signin", req.url);
    loginUrl.searchParams.set("callbackUrl", req.nextUrl.pathname);
    return NextResponse.redirect(loginUrl);
  }

  if (pathname.startsWith("/admin") && token.role !== "ADMIN") {
    const homeUrl = new URL("/", req.url);
    return NextResponse.redirect(homeUrl);
  }

  return NextResponse.next();
}
```

Для автентифікації користувачів у системі застосовується бібліотека NextAuth, яка забезпечує зручну інтеграцію з базою даних через адаптер PrismaAdapter. Система використовує стратегію jwt, що дозволяє зберігати

|      |      |                |        |      |                          |      |
|------|------|----------------|--------|------|--------------------------|------|
|      |      | Українець Д.В. |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      | Морозов А.В.   |        |      |                          |      |
| Змн. | Арк. | № докум.       | Підпис | Дата |                          | 38   |

інформацію про користувача у вигляді токена без необхідності створення сесій на сервері. У якості провайдера автентифікації використовується CredentialsProvider, який реалізує вхід за логіном і паролем. Під час авторизації облікові дані користувача перевіряються у базі даних, а пароль порівнюється з хешованим значенням за допомогою бібліотеки bcrypt. У разі успішної авторизації користувач отримує JWT, у якому зберігається не лише його ідентифікатор, а й роль. Це дозволяє надалі використовувати рольову модель доступу в усій системі. У колбек-функціях jwt і session здійснюється збереження додаткової інформації про користувача у токен і сесію відповідно. Таким чином, застосунок отримує гнучкий механізм управління правами доступу, який тісно інтегрований із middleware для перевірки прав на рівні маршрутів. Нижче наведено код налаштування NextAuth, який реалізує описану логіку:

```
import NextAuth from "next-auth";
import CredentialsProvider from "next-auth/providers/credentials";
import { PrismaAdapter } from "@next-auth/prisma-adapter";
import { prisma } from "@lib/prisma";
import bcrypt from "bcrypt";

export const authOptions = {
  adapter: PrismaAdapter(prisma),
  session: {
    strategy: "jwt" as const,
  },
  providers: [
    CredentialsProvider({
      name: "Credentials",
      credentials: {
        email: {
          label: "Email",
          type: "email",
          placeholder: "you@example.com",
        },
        password: { label: "Password", type: "password" },
      },
      async authorize(credentials) {
        if (!credentials?.email || !credentials?.password) {
          throw new Error("Email and password required");
        }
      },
    })
  ],
}
```

```

    }
    const user = await prisma.user.findUnique({
      where: { email: credentials.email },
    });
    if (!user || user.isBlocked) {
      throw new Error("No access");
    }
    const isValid = await bcrypt.compare(
      credentials.password,
      user.passwordHash
    );
    if (!isValid) {
      throw new Error("Invalid credentials");
    }
    return {
      id: user.id,
      name: user.username,
      email: user.email,
      role: user.role,
    };
  },
  }),
],
callbacks: {
  async jwt({ token, user }) {
    if (user) {
      token.id = user.id;
      token.role = user.role;
    }
    return token;
  },
  async session({ session, token }) {
    session.user = {
      ...session.user!,
      id: token.id as string,
      role: token.role as string,
    };
    return session;
  },
},
pages: {

```

|      |      |                |        |      |                          |      |
|------|------|----------------|--------|------|--------------------------|------|
|      |      | Українець Д.В. |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      | Морозов А.В.   |        |      |                          |      |
| Змн. | Арк. | № докум.       | Підпис | Дата |                          | 40   |

```

    signIn: "/signin",
    error: "/auth/error",
  },
  secret: process.env.NEXTAUTH_SECRET,
};

const handler = NextAuth(authOptions);
export { handler as GET, handler as POST };

```

Завдяки цій конфігурації система авторизації легко масштабується, підтримує перевірку ролей користувачів та забезпечує безпечний і гнучкий механізм доступу до функціоналу застосунку.

У даному проєкті застосовано репозиторний патерн для абстрагування доступу до бази даних. Це дозволяє розділити бізнес-логіку застосунку та логіку зберігання/отримання даних, спрощує тестування та підтримку коду. Кожен репозиторій відповідає за одну сутність і надає набір методів для взаємодії з відповідною таблицею у базі даних через Prisma ORM. Для управління текстами, які використовуються у тестах на швидкість друку, реалізовано клас `TextRepository`. Нижче подано стислу структуру ключових методів цього класу:

```

export class TextRepository {
  async createText(data: CreateTextDto) { ... }

  async getTextById(id: string) { ... }

  async getAllTexts() { ... }

  async getApprovedTexts() { ... }

  async updateText(id: string, data: UpdateTextDto) { ... }

  async deleteText(id: string) { ... }

  async getRandomText(languageId?: string, difficultyLevel?: number) { ... }
}

```

Кожен з цих методів інкапсулює взаємодію з ORM, що дозволяє зручно використовувати `TextRepository` в інших частинах застосунку без дублювання

|      |      |                |        |      |                          |      |
|------|------|----------------|--------|------|--------------------------|------|
|      |      | Українець Д.В. |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      | Морозов А.В.   |        |      |                          |      |
| Змн. | Арк. | № докум.       | Підпис | Дата |                          | 41   |

запитів до бази. Це підвищує читаність, модульність та підтримуваність коду. Повна реалізація репозиторіїв наведена у додатку А.

Для обробки серверної логіки в застосунку використано Server Actions — сучасний підхід, який дозволяє викликати серверні функції безпосередньо з клієнтського коду без необхідності створення окремих API-маршрутів. Це суттєво спрощує структуру проєкту та покращує продуктивність. Server Actions взаємодіють із базою даних не безпосередньо, а через спеціальні репозиторії, що реалізують репозиторний патерн. Такий підхід дозволяє чітко розділити бізнес-логіку та логіку доступу до даних, що сприяє кращій масштабованості та підтримуваності проєкту.

Нижче наведено приклад реалізації серверних функцій, які використовують методи репозиторію TextRepository:

```
"use server";
import { TextRepository } from "@lib/repositories/TextRepository";
const textRepo = new TextRepository();
export async function createTextAction(content: string, languageId: string,
difficultyLevel: number, createdByUserId: string) { ... }
export async function getTextByIdAction(id: string) { ... }
export async function getAllTextsAction() { ... }
export async function getApprovedTextsAction() { ... }
export async function updateTextAction(id: string, data: Partial<{ content: string;
difficultyLevel: number; isApproved: boolean; }>) { ... }
export async function deleteTextAction(id: string) { ... }
export async function getRandomTextAction(languageId?: string, difficultyLevel?: number)
{ ... }
```

Повний код реалізації всіх функцій, наведено у Додатку А.

У застосунку для тренування швидкості друку реалізовано функціонал генерації текстів за допомогою штучного інтелекту. Це дозволяє створювати навчальні матеріали динамічно, відповідно до обраної мови та рівня складності, що значно розширює можливості системи й робить процес тренування більш персоналізованим. Для реалізації цієї функціональності використовується модель Gemini від Google, яка інтегрована за допомогою бібліотеки @ai-sdk/google. Користувач вказує бажану мову та рівень складності (від 1 до 5), після

|      |      |                |        |      |                          |      |
|------|------|----------------|--------|------|--------------------------|------|
|      |      | Українець Д.В. |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      | Морозов А.В.   |        |      |                          |      |
| Змн. | Арк. | № докум.       | Підпис | Дата |                          | 42   |

чого система формує відповідний запит до моделі. У відповіді штучний інтелект генерує зв'язний суцільний текст обсягом від 300 до 600 символів. Особливістю реалізації є те, що текст не містить списків, діалогів або структурованих елементів — лише послідовний абзац для тренування друку. Нижче наведено фрагмент реалізації серверної функції, яка відповідає за генерацію навчального тексту:

```
import { streamText } from "ai";
import { google } from "@ai-sdk/google";

export const maxDuration = 30;

export async function POST(req: Request) {
  const { language, level }: { language: string; level: number } =
    await req.json();

  const levelDescription = {
    1: "дуже простий текст (короткі прості речення, загальна лексика)",
    2: "простий текст (буденна тематика, короткі речення)",
    3: "середній рівень (звичайна мова для щоденного вжитку, трохи складніші конструкції)",
    4: "складний текст (трохи технічна або академічна мова, більше складнопідрядних речень)",
    5: "дуже складний текст (науковий, офіційний стиль, термінологія)",
  };

  const messages = [
    {
      role: "system",
      content: `Ти генератор навчальних текстів для тренування швидкості друку.
      Генеруй тексти українською мовою або іншою мовою, яку вкаже користувач.
      Рівень складності тексту повинен відповідати опису рівня.
      Не додавай заголовок чи пояснення — лише сам текст для друку (від 300 до 600 символів).
      Важливо — не генеруй списків, питань, діалогів, коду або маркування — лише суцільний текст.`,
    },
    {
      role: "user",
      content: `Згенеруй текст для тренування друку.`
    }
  ]
```

|      |      |                |        |      |                          |      |
|------|------|----------------|--------|------|--------------------------|------|
|      |      | Українець Д.В. |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      | Морозов А.В.   |        |      |                          | 43   |
| Змн. | Арк. | № докум.       | Підпис | Дата |                          |      |

```

    мова: "${language}".
    Рівень складності: ${level} (${levelDescription[level] ?? "середній"}).
    Формат: один суцільний текст від 300 до 600 символів.`
  },
];

const result = streamText({
  model: google("models/gemini-2.0-flash-exp"),
  messages,
});

return result.toDataStreamResponse();
}

```

Завдяки такому підходу система може адаптувати контент під різні рівні користувачів без необхідності вручну створювати навчальні тексти, що робить застосунок значно гнучкішим і ефективнішим для масового використання.

### Висновок до другого розділу

У другому розділі було здійснено повний цикл проєктування та реалізації веб-застосунку для тренування швидкості друку. Розглянуто вибір технологій, серед яких ключовими стали фреймворк Next.js, база даних PostgreSQL та Prisma ORM. Обґрунтовано використання клієнт-серверної архітектури із застосуванням Server Actions, що дозволило досягти високої продуктивності та спростити обробку запитів. Було спроектовано структуру бази даних, описано таблиці та їхні зв'язки, реалізовано репозиторний патерн доступу до даних.

Окрему увагу приділено реалізації автентифікації з розмежуванням доступу на основі ролей користувачів, а також використанню middleware для контролю доступу до захищених маршрутів. Описано логіку ключових процесів системи, зокрема алгоритм проходження тесту на швидкість друку. Також реалізовано функціонал генерації навчальних текстів за допомогою моделі Gemini AI, що підвищує адаптивність і корисність платформи.

Усі реалізовані рішення спрямовані на створення зручного, ефективного й безпечного інструменту для навчання швидкому набору тексту, що легко масштабується та підтримується.

## РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ ДОДАТКУ

### 3.1 Порядок встановлення та налаштування системи

Діаграма розгортання (рис. 3.1) демонструє загальну інфраструктуру веб-застосунку для тренування швидкості друку, показуючи логічну взаємодію між основними апаратними вузлами та програмними компонентами у процесі розгортання та використання системи. Така діаграма дозволяє візуалізувати архітектуру системи у реальному середовищі виконання та зрозуміти, як окремі компоненти взаємодіють між собою.

Користувач працює із застосунком через сучасний веб-браузер, який взаємодіє з веб-сервером на базі Next.js. Веб-сервер одночасно виконує роль клієнтської частини (інтерфейсу), серверної логіки (обробка запитів), а також містить окремі Server Actions, що забезпечують обробку запитів до бази даних та взаємодію з іншими сервісами. Також до складу веб-серверу входить модуль авторизації (NextAuth), middleware для контролю доступу, а також модуль інтеграції з сервісом генерації навчальних текстів на базі Google Gemini AI.

Дані зберігаються у реляційній базі даних PostgreSQL, з якою застосунок взаємодіє через ORM Prisma. Під час запуску сервер встановлює з'єднання з базою даних, після чого зберігає його у вигляді сінглтона, що дозволяє уникнути зайвих підключень і покращити продуктивність. У базі зберігаються користувачі, мовні тексти, результати тестів, турнірна статистика та інша інформація.

Інтеграція із Google Gemini API виконується через серверну частину, що дозволяє безпечно отримувати навчальні тексти для тренування друку. Ці тексти генеруються відповідно до вибраної мови та рівня складності. Такий підхід забезпечує інтелектуальну адаптацію контенту до рівня користувача та робить процес навчання більш ефективним.

|      |      |                |        |      |                          |      |
|------|------|----------------|--------|------|--------------------------|------|
|      |      | Українець Д.В. |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      | Морозов А.В.   |        |      |                          |      |
| Змн. | Арк. | № докум.       | Підпис | Дата |                          | 45   |

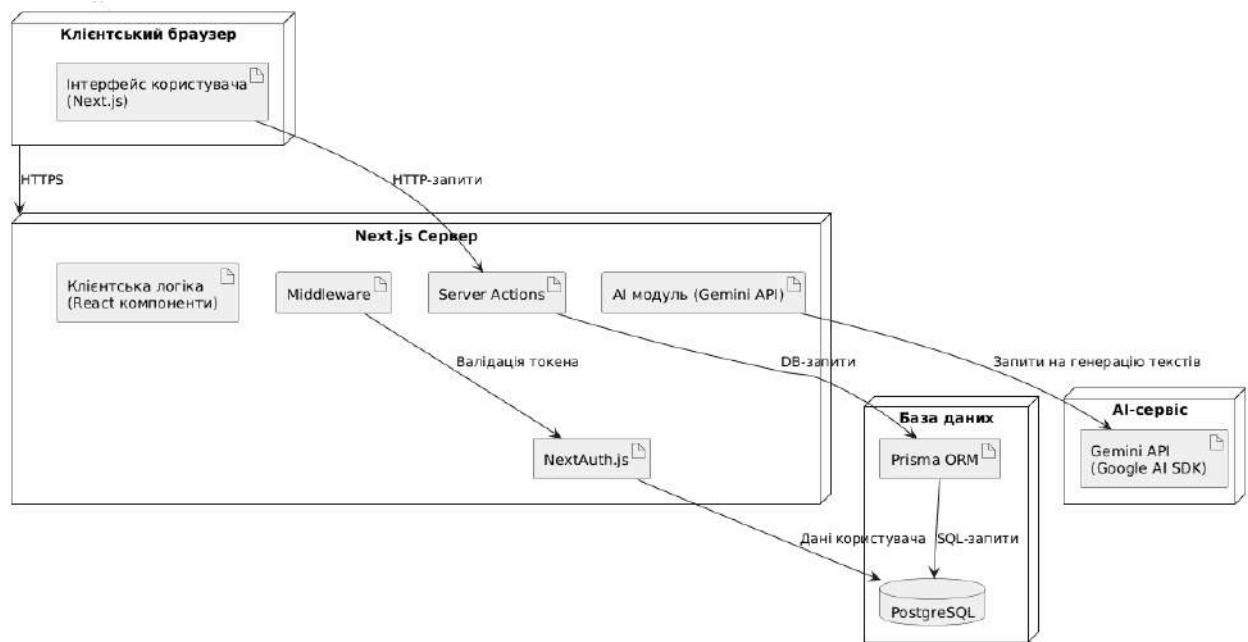


Рис. 3.1 – Діаграма розгортання веб-застосунку для тренування швидкості друку

### 3.2 Структура інтерфейсу та порядок взаємодії з застосунком для тренування швидкості друку

Для користування застосунком із тренування швидкості друку достатньо пристрою з доступом до Інтернету та сучасного браузера. Інтерфейс реалізовано з урахуванням принципів адаптивного дизайну, що забезпечує комфортну роботу як на настільних комп'ютерах, так і на мобільних пристроях. Головні елементи навігації інтуїтивно зрозумілі, що дозволяє користувачу швидко зорієнтуватися та розпочати тренування без додаткових інструкцій.

Після відкриття застосунку користувач одразу отримує можливість почати проходження тесту без авторизації. У такому режимі дані про швидкість та точність введення не зберігаються, однак користувач може ознайомитися з інтерфейсом та протестувати свої навички друку. Для отримання повного функціоналу, включно зі збереженням результатів, доступом до особистої статистики, участю в турнірах та можливістю запропонувати власний текст, необхідно увійти до акаунту або зареєструватися. Це дозволяє системі персоналізувати досвід взаємодії та відображати індивідуальний прогрес користувача.

Користувач також має можливість обрати мову тексту для тренування. Інтерфейс підтримує кілька мовних варіантів, серед яких українська, англійська та німецька. Зміна мови відображається миттєво — система автоматично підбирає відповідний навчальний текст згідно з обраним мовним параметром. Це дозволяє тренувати швидкість друку не лише рідною мовою, а й удосконалювати навички друку іноземними мовами. Така гнучкість робить застосунок універсальним інструментом як для новачків, так і для користувачів з різним мовним досвідом. Загальний вигляд інтерфейсу сторінки тренування подано на рисунку 3.2.

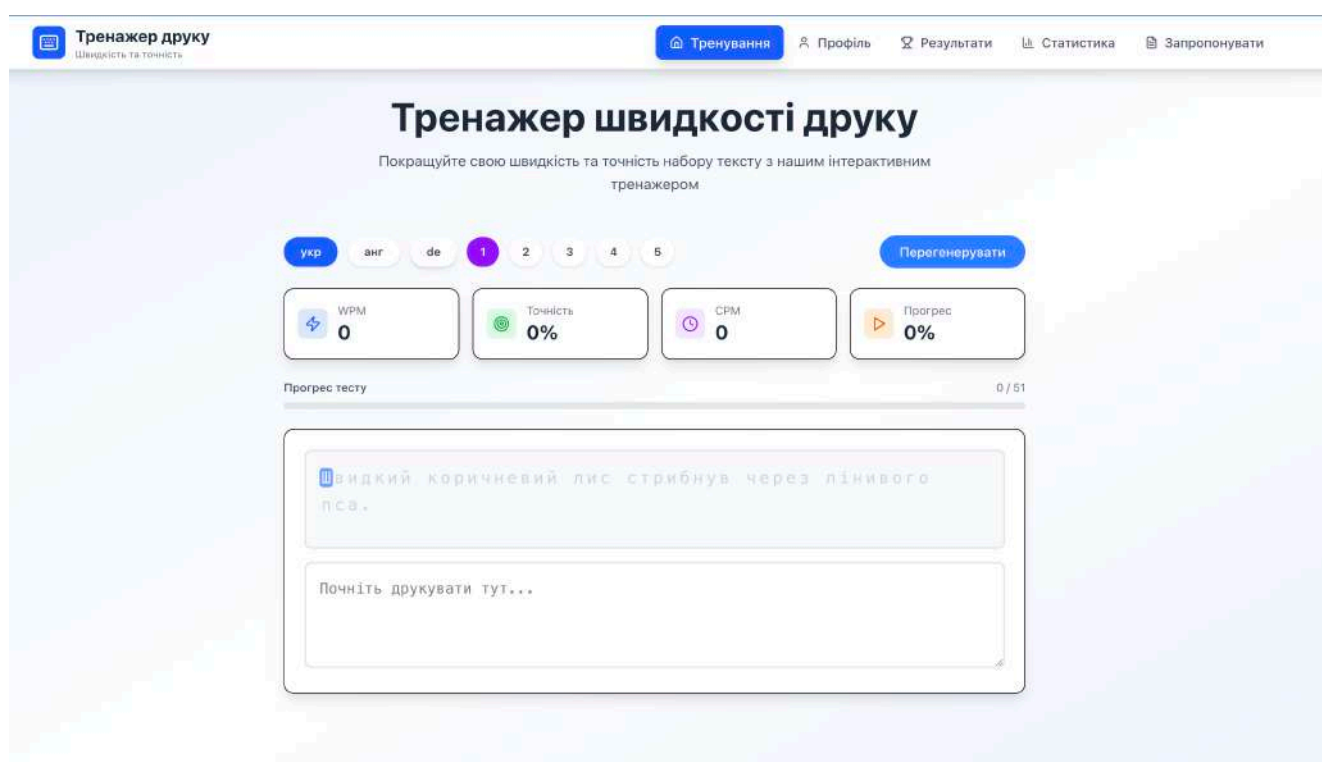


Рис. 3.2 – Інтерфейс тренажера швидкості друку

Для початку проходження тесту на швидкість друку користувач має обрати мову тексту зі списку доступних варіантів, а також встановити бажаний рівень складності — від дуже простого до максимально складного. Після цього необхідно почати вводити текст у спеціальне поле, що автоматично запускає відлік часу. Під час проходження тесту система у реальному часі аналізує точність введення, швидкість друку (в словах і символах за хвилину), а також показує загальний прогрес. Після завершення тексту надається короткий звіт про результати, як представлено на рисунку 3.3.

|      |      |                |        |      |                          |      |
|------|------|----------------|--------|------|--------------------------|------|
|      |      | Українець Д.В. |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      | Морозов А.В.   |        |      |                          |      |
| Змн. | Арк. | № докум.       | Підпис | Дата |                          | 47   |

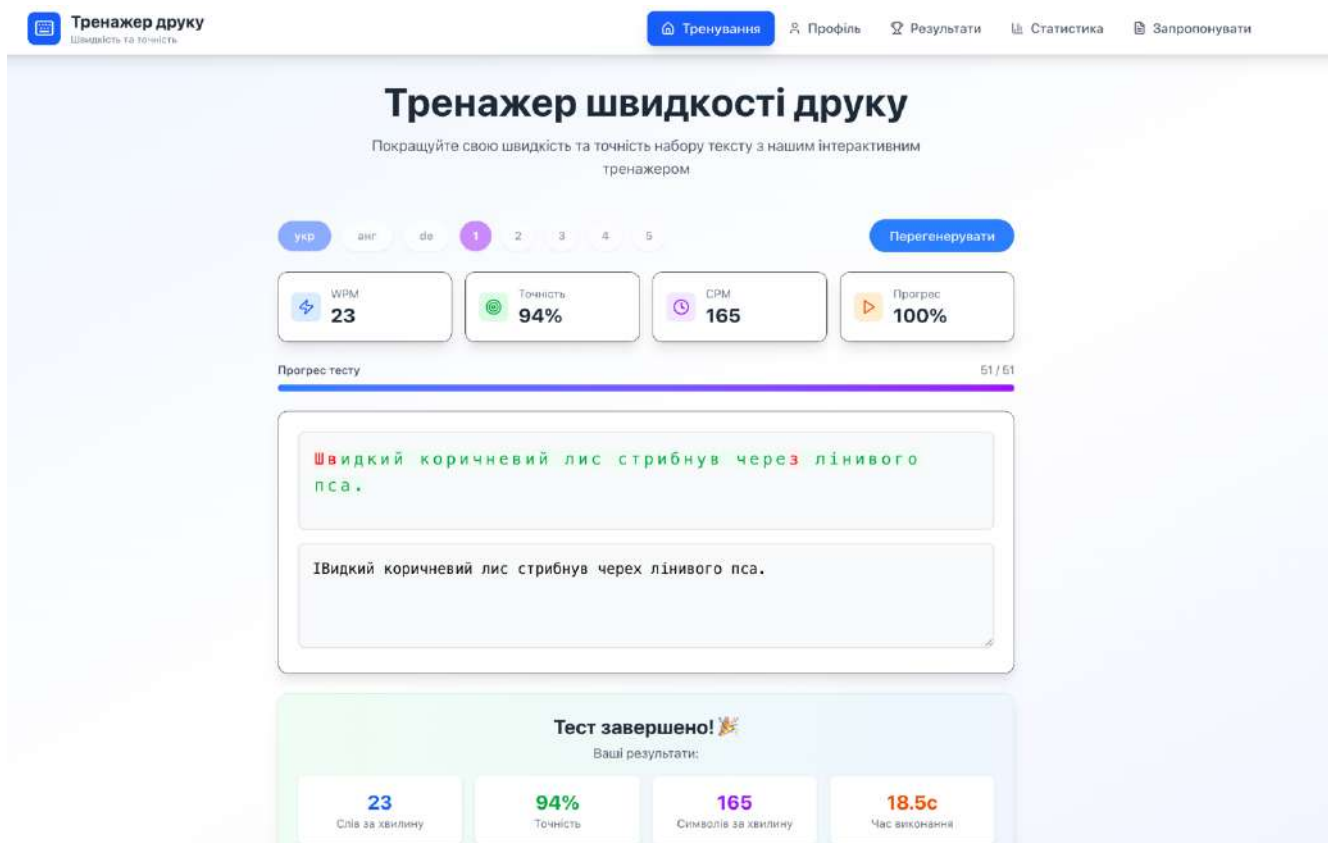


Рис. 3.3 – Виведення результатів після проходження тесту на швидкість друку

Для входу в систему користувач може скористатися відповідною кнопкою у верхньому меню навігації, яка відкриває сторінку авторизації. На цій сторінці необхідно ввести електронну пошту та пароль, після чого натиснути кнопку «Увійти». У разі відсутності акаунта користувач має можливість перейти до форми реєстрації та створити новий обліковий запис, заповнивши основні поля: ім'я користувача, електронну пошту та пароль. Процес реєстрації простий та не вимагає додаткової верифікації. Вигляд сторінок авторизації та реєстрації наведено на рисунках 3.4 та 3.5 відповідно.

|      |      |                |        |      |                          |      |
|------|------|----------------|--------|------|--------------------------|------|
|      |      | Українець Д.В. |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      | Морозов А.В.   |        |      |                          | 48   |
| Змн. | Арк. | № докум.       | Підпис | Дата |                          |      |

**Тренажер друку**  
Швидкість та точність

Тренування

Профіль

Результати

Статистика

Запропонувати

### Вхід до системи

Увійдіть до свого акаунта для продовження

Електронна пошта

Введіть ваш email

Пароль

Введіть ваш пароль

☐ Запам'ятати мене

Забули пароль?

Увійти

Немає акаунта? Зареєструватися

Рис. 3.4 – Сторінка входу в систему

**Тренажер друку**  
Швидкість та точність

Тренування

Профіль

Результати

Статистика

Запропонувати

### Створити акаунт

Приєднуйтеся до тренажера швидкості друку

Ім'я користувача

Введіть ім'я користувача

Електронна пошта

Введіть email

Пароль

Введіть пароль

Підтвердіть пароль

Повторіть пароль

Вимоги до пароля:

- Мінімум 6 символів
- Паролі співпадають

Створити акаунт

Вже маєте акаунт? Увійти

Рис. 3.5 – Сторінка створення нового акаунта

Також неавторизовані користувачі мають можливість ознайомитися з результатами тестів інших учасників системи. У розділі «Результати тестів» відображається загальна аналітика, включаючи кількість виконаних тестів, середню швидкість друку, найвищий результат та середню точність усіх

|      |      |                |        |      |                          |      |
|------|------|----------------|--------|------|--------------------------|------|
|      |      | Українець Д.В. |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      | Морозов А.В.   |        |      |                          | 49   |
| Змн. | Арк. | № докум.       | Підпис | Дата |                          |      |

користувачів. Завдяки зручним фільтрам можна переглядати результати за мовою, текстом або іменем користувача. У нижній частині сторінки подано рейтинг найкращих виконавців на основі їхніх показників. Це створює елемент мотивації та змагання серед користувачів. Інтерфейс цього розділу зображено на рисунку 3.6.

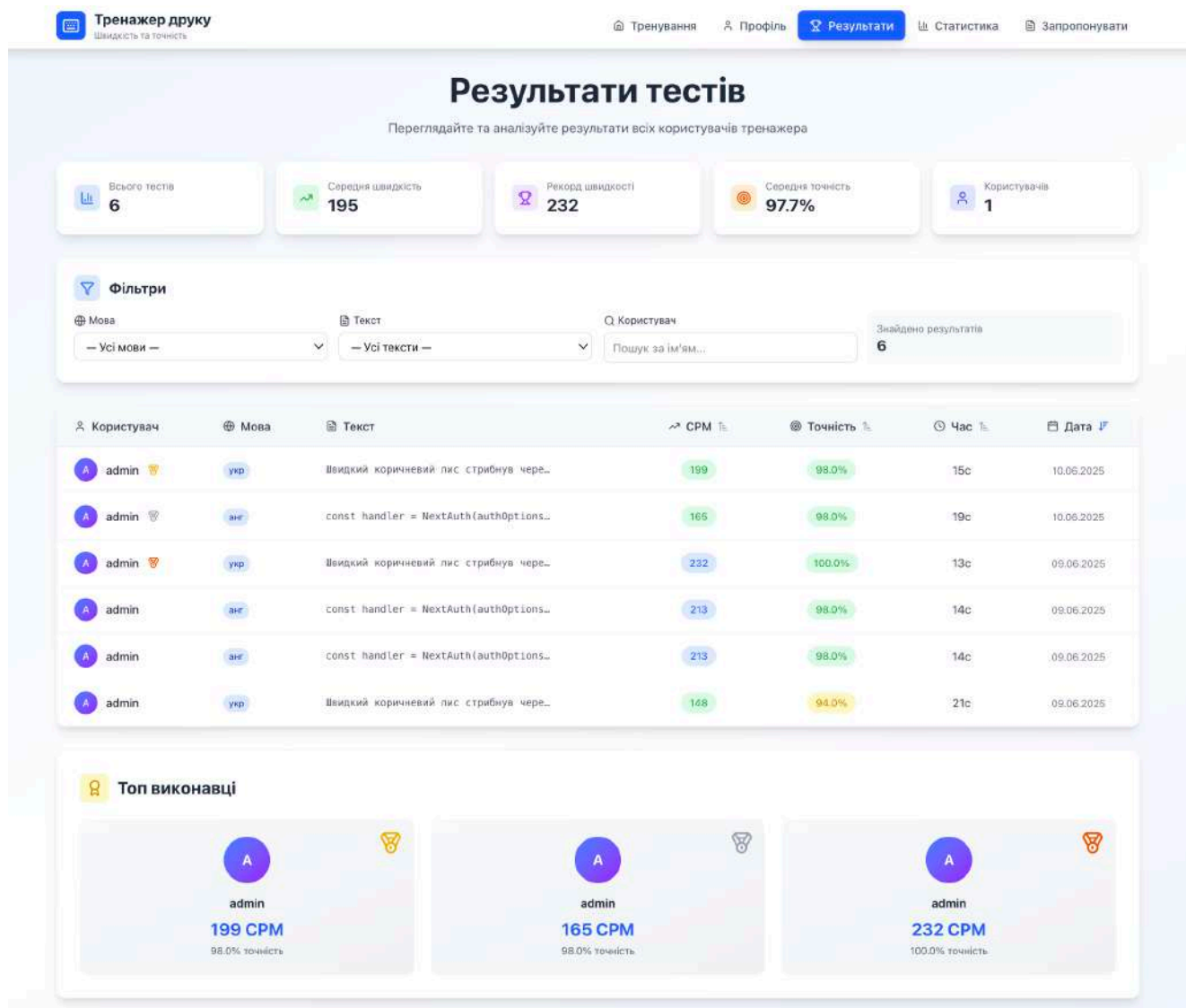


Рис. 3.6 – Сторінка результатів тестів із фільтрацією та рейтингом користувачів

Крім результатів тестів, користувачі можуть переглядати узагальнену статистику в спеціальному розділі «Статистика користувачів». Тут відображаються досягнення всіх учасників системи — середня швидкість, найвищий результат, середня точність і загальна кількість тестів. Сторінка також містить рейтинг найкращих виконавців, що дозволяє швидко оцінити прогрес

найактивніших користувачів. Для зручності передбачено фільтрацію за іменем користувача. Інтерфейс цього модуля зображено на рисунку 3.7.

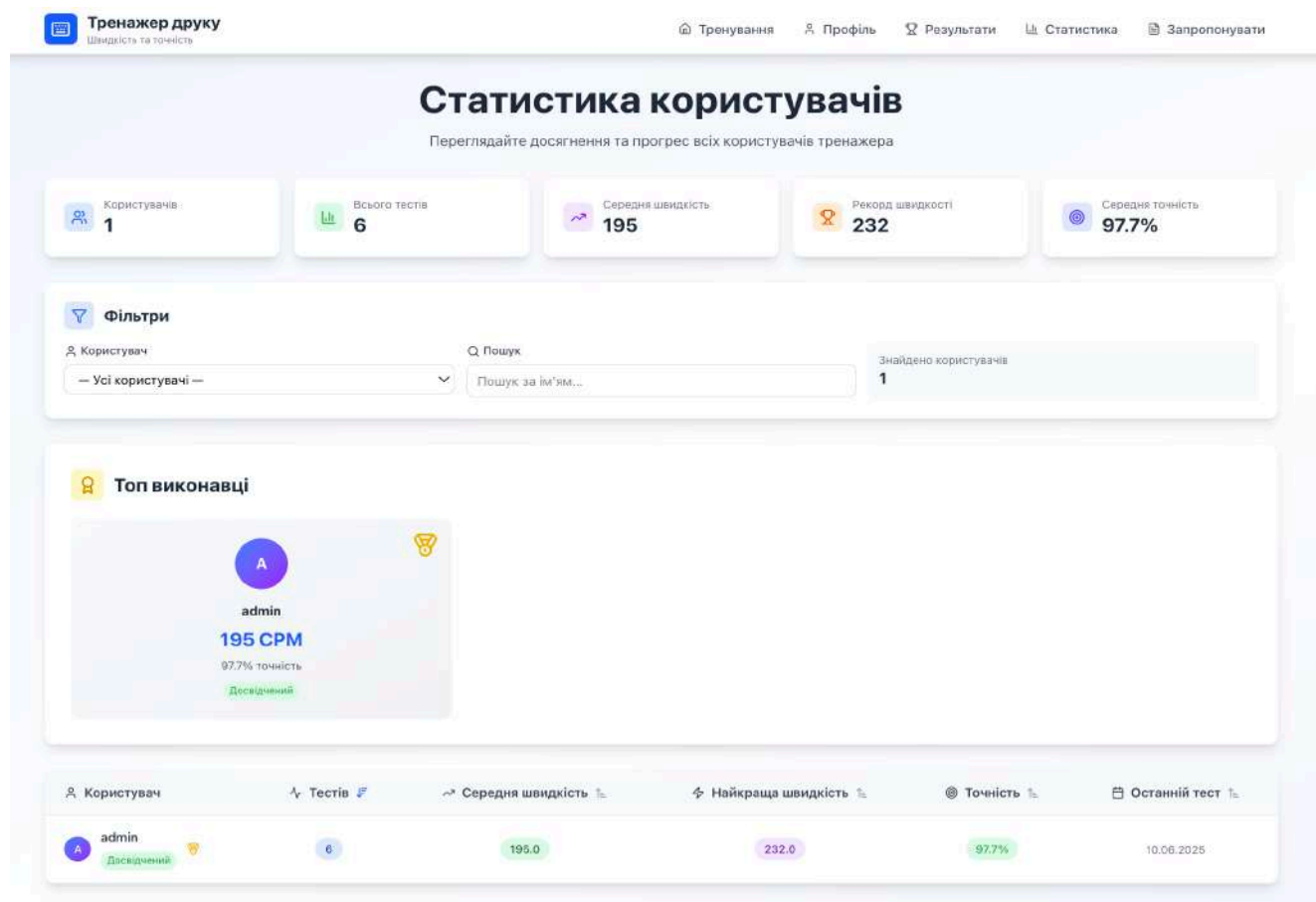


Рис. 3.7 – Розділ загальної статистики користувачів тренажера

Після входу до системи користувач отримує доступ до особистого кабінету, в якому відображається основна статистика активності. У профілі можна переглянути загальну кількість пройдених тестів, середню швидкість друку (у символах за хвилину), найвищий результат, середню точність, а також дату останнього проходження тесту. Крім того, доступна таблиця з історією всіх нещодавніх результатів, де наведено час виконання, точність, швидкість та дату кожного тесту. Завдяки цьому користувач має змогу відстежувати свій прогрес і аналізувати динаміку покращення навичок. Загальний вигляд сторінки профілю представлено на рисунку 3.8.

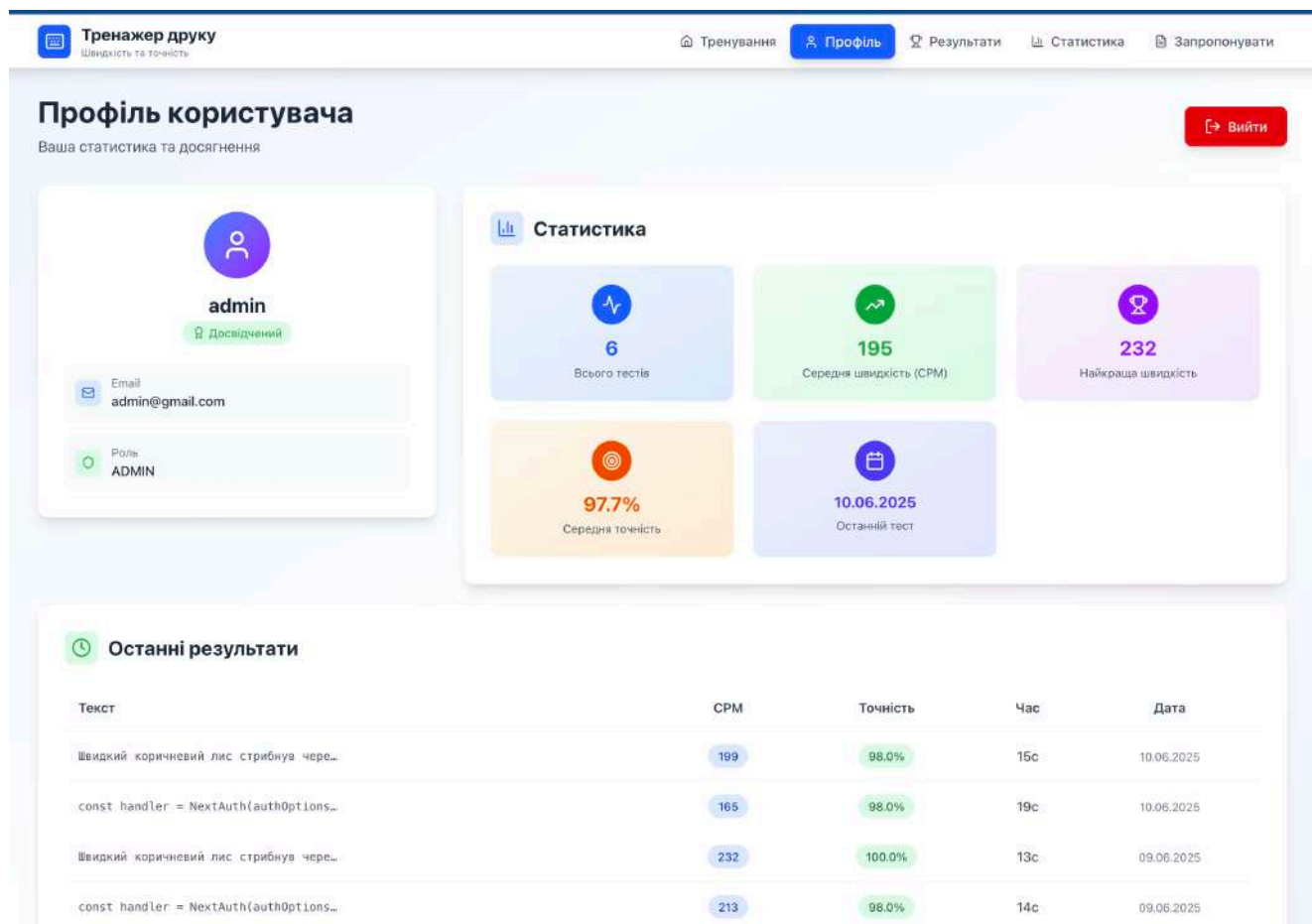


Рис. 3.8 – Профіль користувача із загальною статистикою та історією результатів

Авторизовані користувачі мають можливість покращувати платформу, пропонуючи власні тексти для тренування. У спеціальному розділі «Запропонувати» доступна форма, де можна ввести текст, вибрати мову та зазначити рівень складності. Система також надає підказки щодо якості контенту — рекомендовано уникати спеціальних символів і використовувати освітні, цікаві тексти. Такий функціонал доступний лише після входу в акаунт і дає змогу розширювати базу тренувальних матеріалів. Інтерфейс цієї сторінки зображено на рисунку 3.9.

Тренажер друку

Швидкість та точність

Тренування

Профіль

Результати

Статистика

Запропонувати

?

Запропонувати новий текст

Допоможіть покращити тренажер, додавши нові тексти для тренування швидкості друку

Новий текст

Текст для тренування

Введіть текст для тренування швидкості друку. Рекомендується використовувати цікавий та освітній контент...

Символів: 0 Слів: 0

Мова

укр — Українська

Рівень складності

1 — Початковий

Початковий

Запропонувати текст

Рекомендації

Використовуйте цікавий та освітній контент

Мінімум 50 символів для ефективного тренування

Перевірте правопис та пунктуацію

Уникайте спеціальних символів

Рівні складності

1 Початковий

Прості слова, базова лексика

2 Легкий

Звичайні речення, знайомі слова

3 Середній

Складніші конструкції

Рис. 3.9 – Сторінка додавання нового тексту авторизованим користувачем

Авторизований користувач також має можливість брати участь у турнірах із тренування швидкості друку. У відповідному розділі (рис. 3.10) відображаються як рекомендовані, так і всі створені турніри. Користувач може ознайомитися з назвою турніру, його рівнем складності, мовою тексту, періодом проведення та кількістю учасників. Після натискання кнопки «Приєднатися» турнір буде додано до списку активних, а користувач отримає доступ до спеціальної сторінки з текстом турніру. Там він може розпочати змагання, результати якого фіксуються у загальній статистиці. Такий підхід дозволяє створити елемент гейміфікації та мотивувати користувачів покращувати свої навички у конкурентному середовищі.

|      |      |                |        |      |                          |      |
|------|------|----------------|--------|------|--------------------------|------|
|      |      | Українець Д.В. |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      | Морозов А.В.   |        |      |                          | 53   |
| Змн. | Арк. | № докум.       | Підпис | Дата |                          |      |

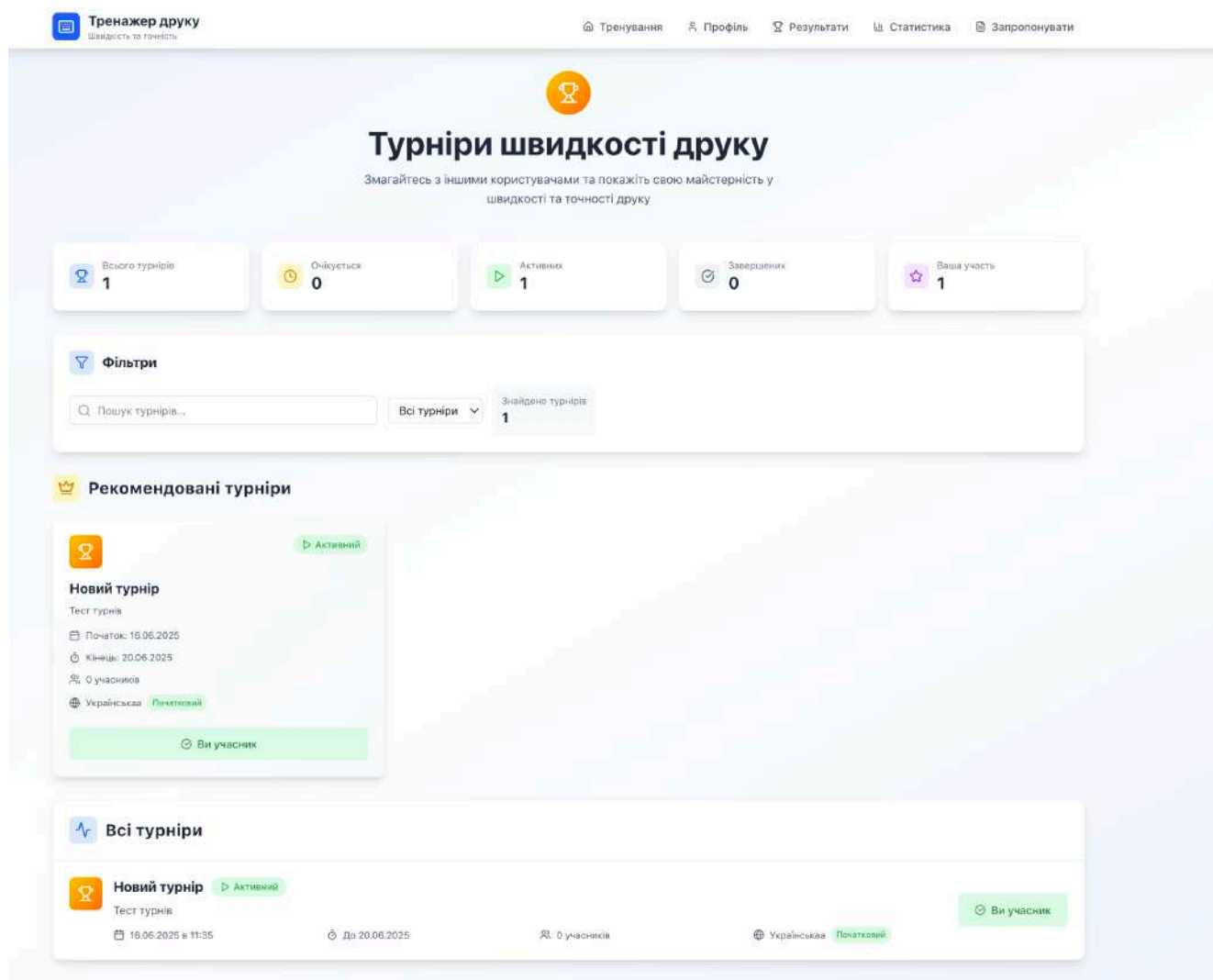


Рис. 3.10 – Інтерфейс перегляду та участі в турнірах

Для користувачів з роллю адміністратора доступна окрема адміністративна панель керування (рис. 3.11). Вона дозволяє переглядати загальну статистику системи, включно з кількістю користувачів, тестів, середніми показниками швидкості та точності. Крім того, адміністратор може аналізувати розподіл тестів за мовами та рівнями складності, що сприяє прийняттю обґрунтованих рішень щодо подальшого розвитку застосунку. З адміністративної панелі також відкривається доступ до управління мовами, користувачами, текстами для тренування та результатами, що робить цей розділ ключовим для підтримки функціональності системи.

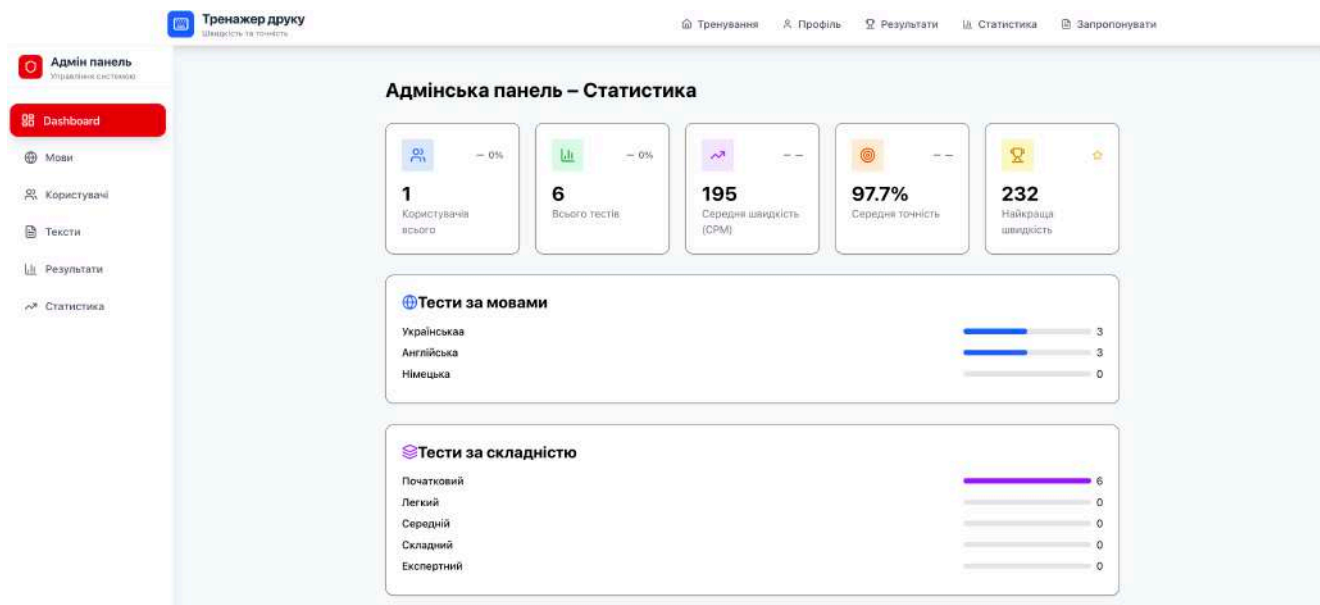


Рис. 3.11 – Інтерфейс адміністративної панелі

У межах адміністративної панелі також реалізовано розділ для управління підтримуваними мовами інтерфейсу та текстів (рис. 3.12). Адміністратор може переглядати список активних мов, а також редагувати або видаляти кожен з них за допомогою відповідних кнопок дій. Крім того, передбачено можливість додавання нових мов до системи через інтерфейс введення коду та назви мови. Такий функціонал дозволяє гнучко налаштовувати мовну підтримку залежно від потреб користувачів та розширення цільової аудиторії.

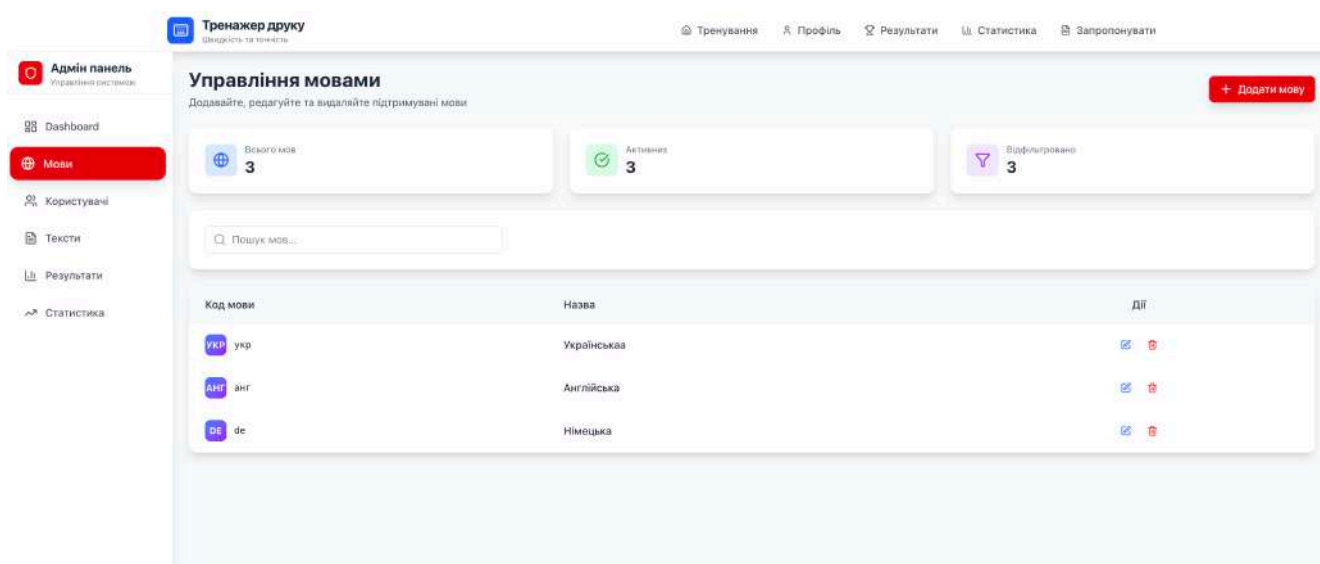


Рис. 3.12 – Інтерфейс керування мовами в адміністративній панелі

У розділі «Тексти» адміністративної панелі адміністратору надається можливість повноцінного управління навчальними матеріалами для тренування

швидкості друку. На відповідній сторінці відображається перелік усіх наявних текстів із зазначенням їх рівня складності, статусу модерації, дати додавання та доступних дій. Тексти можуть бути у статусі «Схвалено» або «На розгляді», що дозволяє контролювати якість контенту перед його появою у відкритому доступі. Кожен текст можна переглянути, відредагувати, затвердити або видалити, а також скористатися фільтрами для швидкого пошуку за мовою, складністю чи статусом.

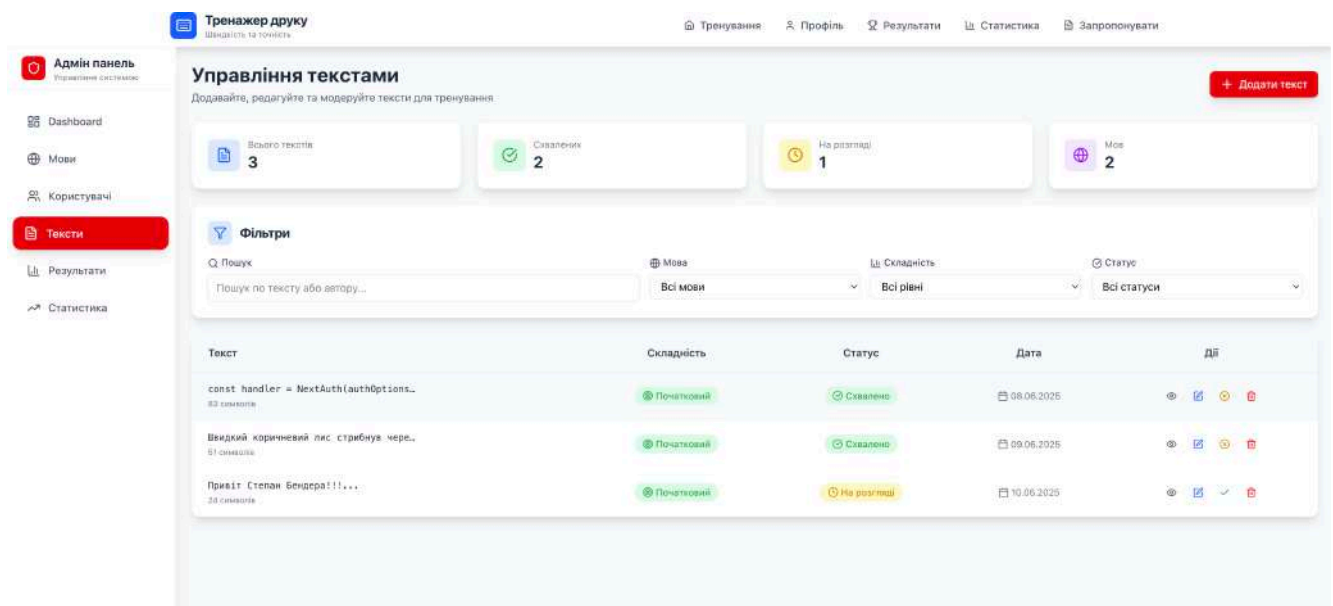


Рис. 3.13 – Інтерфейс модерації текстів адміністративною панеллю

У вкладці «Користувачі» адміністративної панелі реалізовано механізм керування обліковими записами. Адміністратор має змогу переглядати список усіх зареєстрованих користувачів, фільтрувати їх за роллю або статусом, а також здійснювати ключові дії — змінювати роль, блокувати, розблоковувати або видаляти облікові записи. В таблиці також виводиться ім'я користувача, електронна пошта, дата реєстрації та поточний статус. Такий функціонал дозволяє ефективно управляти доступом до системи, запобігати зловживанням та підтримувати порядок у спільноті користувачів.

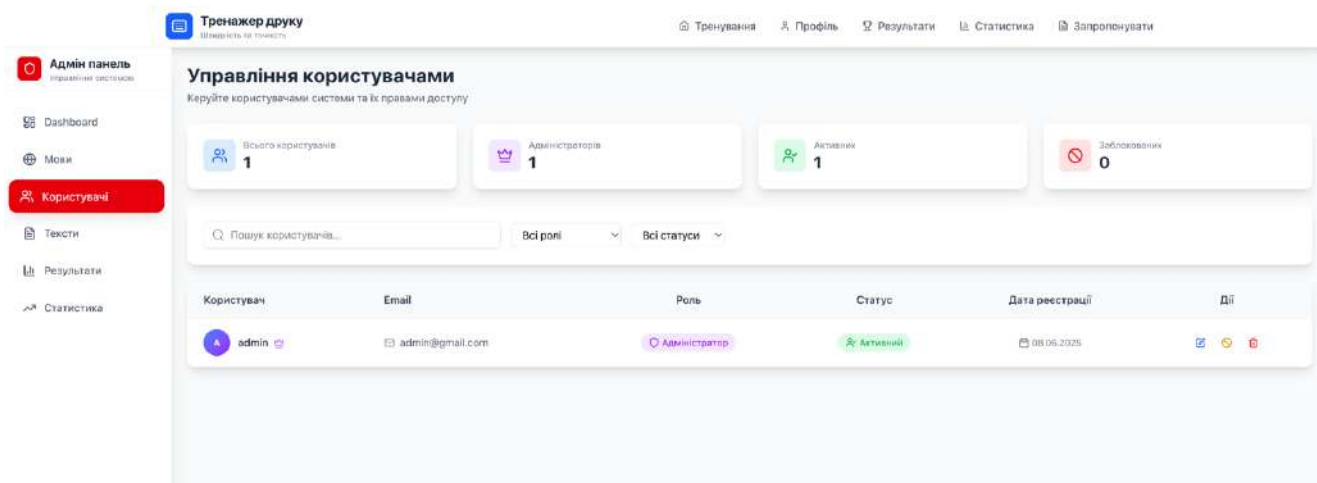


Рис. 3.14 – Інтерфейс керування користувачами адміністративною панеллю

У вкладці «Результати» адміністративної панелі адміністратор має змогу переглядати, фільтрувати та модерувати результати тестів, пройдених користувачами. Доступна аналітика включає середню швидкість, точність, рекордні показники та кількість підозрілих спроб. Інтерфейс також надає список усіх результатів із детальною інформацією про користувача, мову, текст, швидкість (CPM), точність, статус тесту, дату проходження та можливість перегляду або видалення. Такий функціонал дозволяє не лише контролювати активність у системі, а й виявляти потенційні аномалії або зловживання.

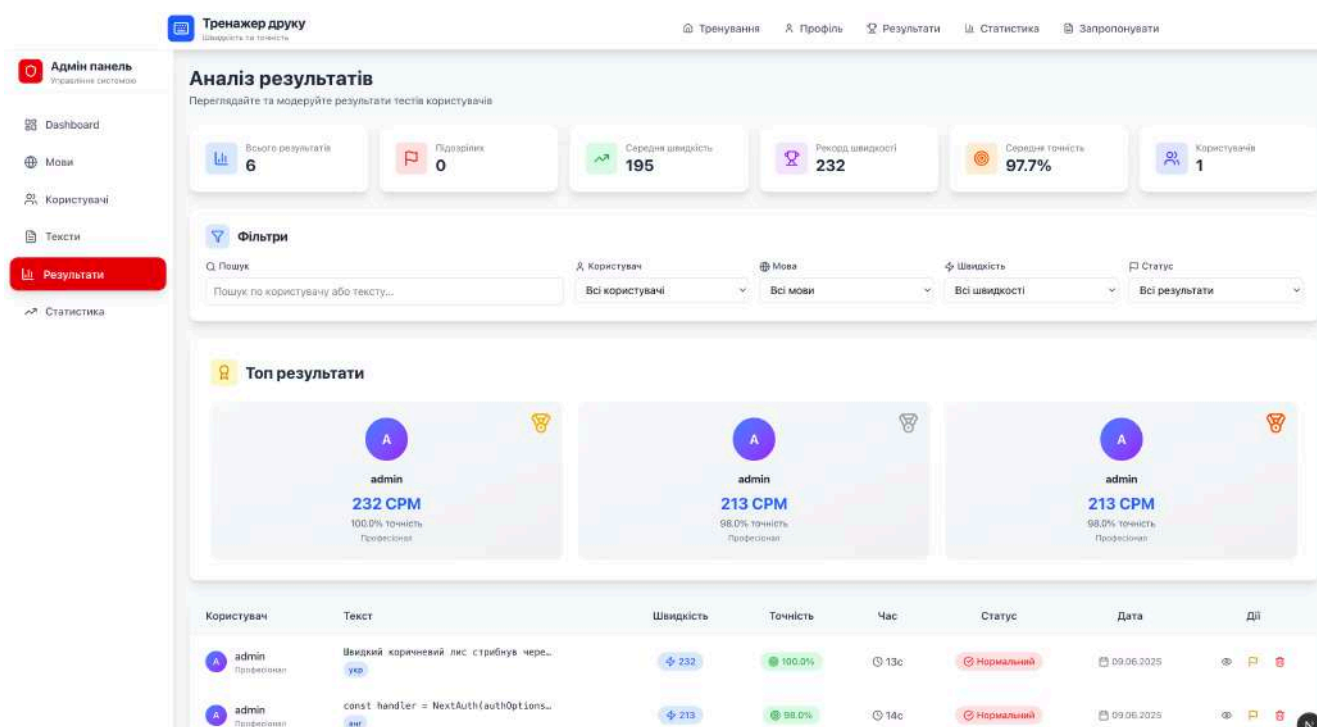


Рис. 3.15 – Інтерфейс аналізу результатів у панелі адміністратора

У розділі «Статистика» адміністративної панелі відображається аналітика по кожному користувачу, включно з кількістю пройдених тестів, середньою швидкістю (CPM), точністю введення, найкращим результатом та датою останнього тестування. Система автоматично визначає найкращих виконавців, показуючи їх у спеціальному блоці. Також реалізовано зручну фільтрацію за іменем користувача, що полегшує навігацію при великій кількості учасників. Цей функціонал дозволяє адміністраторам ефективно відслідковувати прогрес та продуктивність користувачів платформи.

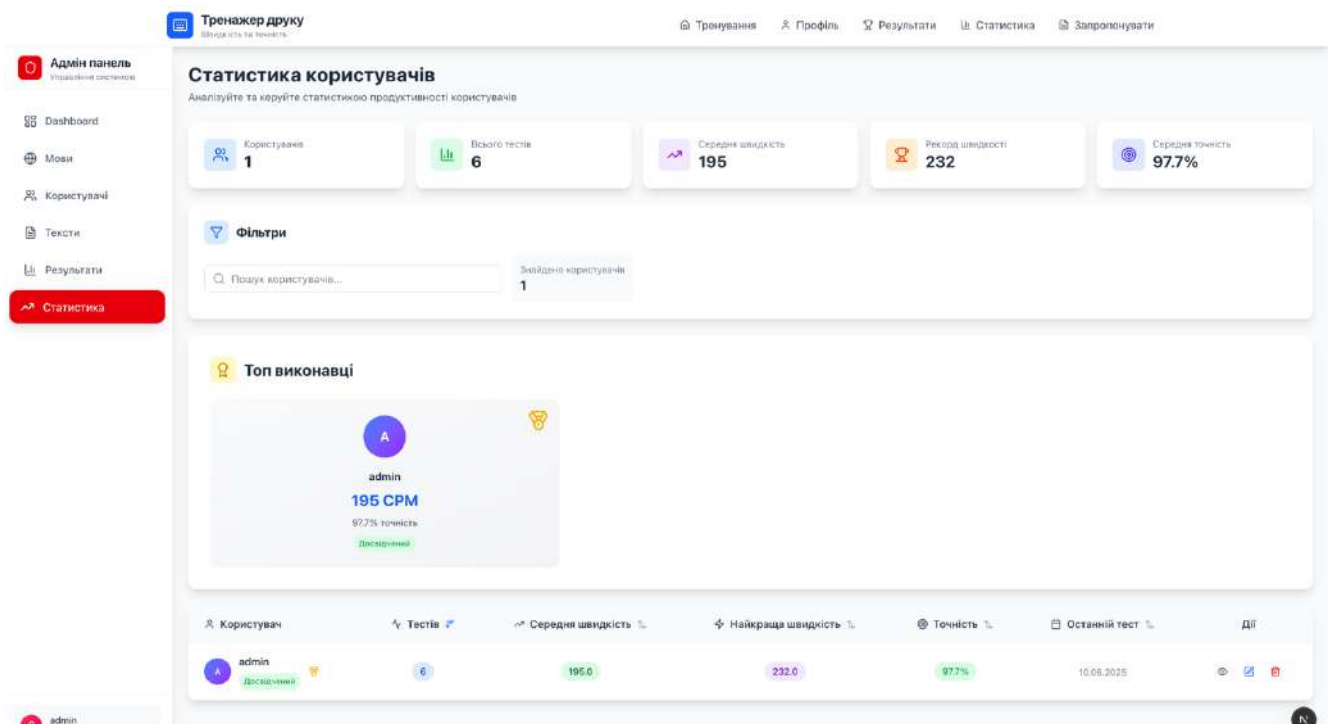


Рис. 3.16 – Інтерфейс перегляду статистики користувачів для адміністратора

У вкладці «Турніри» адміністративної панелі адміністратор може створювати нові змагання з тренування швидкості друку, керувати їх статусом, переглядати кількість учасників, обрані тексти та період проведення. Інтерфейс надає змогу швидко знаходити потрібний турнір за допомогою фільтрації та пошуку, а також редагувати або видаляти існуючі записи. Це дозволяє ефективно організовувати та контролювати навчальний процес у гейміфікованому форматі.

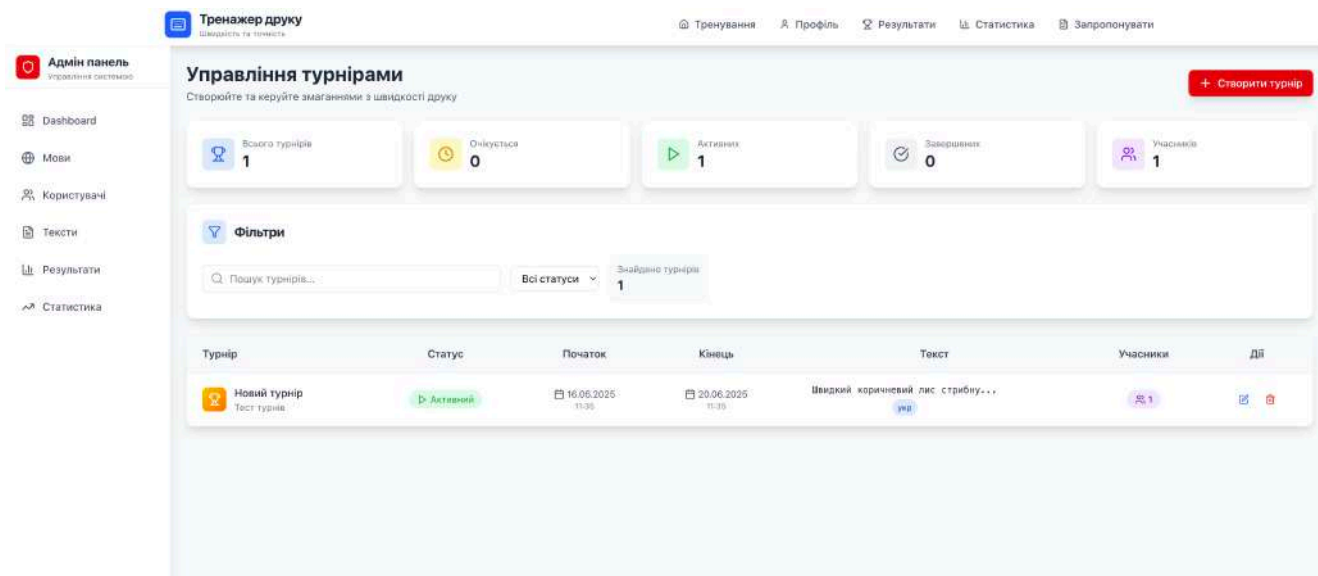


Рис. 3.17 – Інтерфейс керування турнірами в адмін-панелі

### 3.3 Тестування роботи програмного продукту

Після завершення розробки основного функціоналу застосунку для тренування швидкості друку було проведено комплексне ручне тестування. Його метою було виявлення помилок у логіці роботи, перевірка відповідності реалізованих функцій до вимог проєкту та забезпечення стабільної роботи системи в реальних умовах. Особливу увагу приділяли зручності використання інтерфейсу та точності обробки дій користувача під час проходження тестів на друк.

Функціональні перевірки охопили всі ключові сценарії: реєстрацію й вхід користувачів, вибір мови й рівня складності, запуск тесту, підрахунок статистики (швидкість, точність, кількість символів), а також збереження результатів. Було протестовано як користувацький інтерфейс, так і серверну логіку, зокрема взаємодію із базою даних через Prisma та виконання Server Actions. Окремо перевірявся механізм авторизації та захисту доступу до особистого кабінету, турнірів і модерації контенту.

Особлива увага приділялася перевірці роботи генератора навчальних текстів на базі штучного інтелекту (Gemini AI). Було оцінено релевантність та складність згенерованих текстів відповідно до заданого рівня, а також стабільність обробки запитів через стрімінг AI-відповіді. Усі згенеровані

фрагменти тексту були перевірені на відповідність умовам: без списків, діалогів, заголовків та технічного контенту.

Тестування проводилося вручну за попередньо підготовленими чеклістами, які включали всі основні дії користувача в системі. Також перевірялися адаптивність дизайну на мобільних пристроях, коректне відображення інтерфейсу в різних браузерях та стабільність роботи адміністративної панелі. Результати тестів зафіксовано у вигляді таблиці, яка містить перелік функцій, кроки користувача, очікувані результати та фактичний статус виконання.

Таблиця 3.1

Результати тестування основних функцій застосунку

| № тесту | Функціональність                 | Кроки                                                                    | Очікуваний результат                                   | Статус   |
|---------|----------------------------------|--------------------------------------------------------------------------|--------------------------------------------------------|----------|
| 1       | Реєстрація користувача           | Перейти на сторінку реєстрації, ввести дані, натиснути "Зареєструватися" | Користувача створено, відбувається перехід до кабінету | Пройдено |
| 2       | Авторизація користувача          | Ввести email і пароль, натиснути "Увійти"                                | Вхід до особистого кабінету                            | Пройдено |
| 3       | Створення тексту адміністратором | Увійти як адміністратор, перейти до "Тексти", створити новий текст       | Текст з'являється у списку                             | Пройдено |

|   |                              |                                                               |                                             |          |
|---|------------------------------|---------------------------------------------------------------|---------------------------------------------|----------|
| 4 | Проходження тесту на друк    | Увійти як користувач, натиснути "Почати тест", завершити його | Результат зберігається, доступна статистика | Пройдено |
| 5 | Генерація тексту III         | Обрати мову і рівень, натиснути "Згенерувати текст"           | Виводиться згенерований текст без помилок   | Пройдено |
| 6 | Перевірка ролей у middleware | Увійти як звичайний користувач, перейти на сторінку /admin    | Редирект на головну сторінку                | Пройдено |

### Висновки до третього розділу.

У третьому розділі було проаналізовано архітектуру розгортання веб-застосунку для тренування швидкості друку, зокрема логіку взаємодії між клієнтською частиною, серверними компонентами та базою даних. Детально описано порядок користування системою з урахуванням ролей користувача, таких як гість, зареєстрований користувач і адміністратор. Проведено огляд основних елементів інтерфейсу, серед яких модулі тренування, перегляду результатів, статистики, турнірів і адміністративної панелі. В результаті ручного тестування було перевірено основні функції застосунку, включно з авторизацією, проходженням тестів, участю в турнірах та керуванням даними. Отримані результати засвідчили коректну роботу системи, зручність інтерфейсу та відповідність реалізованого функціоналу запланованим вимогам.

## ВИСНОВКИ

У ході виконання випускної кваліфікаційної роботи було спроектовано та реалізовано веб-застосунок для тренування швидкості друку з адаптивним інтерфейсом і підтримкою багатомовних текстів. Проведено аналіз потреб користувачів, вивчено існуючі рішення для навчання друку, після чого сформовано функціональні вимоги до системи. На основі цих вимог було визначено архітектуру застосунку, розроблено структуру інтерфейсу та бази даних, а також реалізовано ключову логіку роботи користувачів з різними рівнями доступу.

Застосунок надає можливість проходити тести на швидкість і точність друку, переглядати результати, аналізувати статистику та брати участь у турнірах. Реалізовано функції генерації навчальних текстів за допомогою штучного інтелекту, зокрема моделі Gemini AI, що дозволяє динамічно формувати матеріал відповідно до вибраної мови та рівня складності. Адміністратори мають окрему панель керування, де можуть модерувати тексти, керувати користувачами, мовами та турнірами.

У роботі розглянуто інтерфейс усіх основних сторінок, наведено приклади дій користувача та описано логіку взаємодії з системою. Проведене ручне тестування підтвердило стабільну роботу всіх модулів, відповідність функціоналу заявленим вимогам і зручність користування інтерфейсом.

Результатом виконаної роботи є повнофункціональний веб-застосунок, який може бути використаний для самостійного тренування навичок друку, освітніх цілей або організації онлайн-змагань. Система легко масштабовується, має гнучку архітектуру та потенціал для подальшого розвитку, зокрема в напрямках персоналізованих рекомендацій, адаптивного навчання та гейміфікації.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. NextJS [Електронний ресурс] – Режим доступу: <https://nextjs.org>
2. NextJS Server Actions [Електронний ресурс] – Режим доступу: <https://nextjs.org/docs/app/building-your-application/data-fetching/server-actions-and-mutations>
3. NextAuth.js Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://next-auth.js.org/getting-started/introduction>
4. MongoDB [Електронний ресурс] – Режим доступу: <https://www.mongodb.com>
5. PostgreSQL- [Електронний ресурс] - Режим доступу до ресурсу: <https://www.postgresql.org/>
6. Prisma [Електронний ресурс]. – Режим доступу: <https://www.prisma.io>
7. Vercel AI SDK [Електронний ресурс] – Режим доступу: <https://vercel.com/docs/ai/vercel-ai-sdk>
8. Vercel Blob [Електронний ресурс] – Режим доступу: <https://vercel.com/docs/storage/vercel-blob>
9. NextJS SSR [Електронний ресурс] – Режим доступу: <https://nextjs.org/docs/pages/building-your-application/rendering/server-side-rendering>
10. NextJS SSG [Електронний ресурс] – Режим доступу: <https://nextjs.org/docs/pages/building-your-application/rendering/static-site-generation>
11. NextJS CSR [Електронний ресурс] – Режим доступу: <https://nextjs.org/docs/pages/building-your-application/rendering/client-side-rendering>
12. TypeScript [Електронний ресурс] – Режим доступу: <https://www.typescriptlang.org>
13. ReactJS [Електронний ресурс] – Режим доступу: <https://react.dev/reference/react>

14. Tailwind CSS [Електронний ресурс] – Режим доступу: <https://tailwindcss.com/docs>
15. Роберт М. Чиста архітектура: Мистецтво створення програмного забезпечення. / Роберт М. [Видання друге] – Харків: Фабула, 2023 – 368 с
16. Learning React: Modern Patterns for Developing React Apps, 2nd edition / Alex Banks, Eve Porcello, 2020 – 307 p.
17. React Explained: Your Step-by-Step Guide to React, 1st edition / Zac Gordon, 2019 – 366 p.
18. Документація JavaScript [Електронний ресурс] – Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide>
19. Документація HTML [Електронний ресурс] – Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Web/HTML>
20. Н. J. How HTML, CSS, and JavaScript work [Електронний ресурс] Н. J. Scrimba blog – Режим доступу до ресурсу: <https://scrimba.com/articles/html-css-javascript/>.
21. What is Client Server Architecture? - Differences, Types, Example [Електронний ресурс] Intellipaat Blog – Режим доступу до ресурсу: <https://intellipaat.com/blog/what-is-client-server-architecture/?US>.
22. Web Application Architecture: Principles, Protocols, and Practices / Leon Shklar and Rich Rosen, 2003. – 608 с.
23. JavaScript – JavaScript. Work with DOM elements [Електронний ресурс] – Режим доступу: <https://www.educative.io/answers/how-to-remove-a-dom-event-listener-using-javascript>
24. CSS Basic concepts of flexbox. How to use flexbox [Електронний ресурс] – Режим доступу: [https://developer.mozilla.org/enUS/docs/Web/CSS/CSS\\_flexible\\_box\\_layout/Basic\\_concepts\\_of\\_flexbox](https://developer.mozilla.org/enUS/docs/Web/CSS/CSS_flexible_box_layout/Basic_concepts_of_flexbox)
25. NextAuth.js Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://next-auth.js.org/getting-started/introduction>

26. Веб-технології та веб-дизайн / О. Г. Трофименко, О. Б. Козін, О. В. Задерейко, О. Є. Плачінда., 2019. – 284 с.
27. Building your own hooks [Електронний ресурс] - Режим доступу до ресурсу: <https://legacy.reactjs.org/docs/hooks-custom.html>
28. Сесіл Мартін Р. Чиста архітектура / Роберт Сесіл Мартін., 2019. – 368 с.
29. Single-page application [Електронний ресурс] – Режим доступу до ресурсу: [https://en.wikipedia.org/wiki/Single-page\\_application](https://en.wikipedia.org/wiki/Single-page_application).
30. What is middleware? [Електронний ресурс] – Режим доступу до ресурсу: <https://www.redhat.com/en/topics/middleware/what-is-middleware>
31. Prisma Documentation - Офіційна документація Prisma [Електронний ресурс]. – Режим доступу до ресурсу: <https://www.prisma.io/docs/>

ДОДАТКИ

|      |      |                |        |      |                          |      |
|------|------|----------------|--------|------|--------------------------|------|
|      |      | Українець Д.В. |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      | Морозов А.В.   |        |      |                          |      |
| Змн. | Арк. | № докум.       | Підпис | Дата |                          | 66   |

## Лістинг authOption.ts:

```

import NextAuth from "next-auth";
import CredentialsProvider from "next-auth/providers/credentials";
import { PrismaAdapter } from "@next-auth/prisma-adapter";
import { prisma } from "@lib/prisma";
import bcrypt from "bcrypt";

export const authOptions = {
  adapter: PrismaAdapter(prisma),
  session: {
    strategy: "jwt" as const,
  },
  providers: [
    CredentialsProvider({
      name: "Credentials",
      credentials: {
        email: {
          label: "Email",
          type: "email",
          placeholder: "you@example.com",
        },
        password: { label: "Password", type: "password" },
      },
      async authorize(credentials) {
        if (!credentials?.email || !credentials?.password) {
          throw new Error("Email and password required");
        }
        const user = await prisma.user.findUnique({
          where: { email: credentials.email },
        });
        if (!user || user.isBlocked) {
          throw new Error("No access");
        }
        const isValid = await bcrypt.compare(
          credentials.password,
          user.passwordHash
        );
        if (!isValid) {
          throw new Error("Invalid credentials");
        }
        return {
          id: user.id,
          name: user.username,
          email: user.email,
          role: user.role,
        };
      },
    }),
  ],
  callbacks: {
    async jwt({ token, user }) {
      if (user) {

```

|      |      |                |        |      |                          |      |
|------|------|----------------|--------|------|--------------------------|------|
|      |      | Українець Д.В. |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      | Морозов А.В.   |        |      |                          |      |
| Змн. | Арк. | № докум.       | Підпис | Дата |                          | 67   |

```

        token.id = user.id;
        token.role = user.role;
    }
    return token;
},
async session({ session, token }) {
    session.user = {
        ...session.user!,
        id: token.id as string,
        role: token.role as string,
    };
    return session;
},
},
pages: {
    signIn: "/signin",
    error: "/auth/error",
},
secret: process.env.NEXTAUTH_SECRET,
};

const handler = NextAuth(authOptions);
export { handler as GET, handler as POST };

```

#### Лістинг TextRepository.ts

```

import { prisma } from "@lib/prisma";
import type { Prisma, Text } from "@generated/prisma";

export type CreateTextDto = {
    content: string;
    languageId: string;
    difficultyLevel: number;
    createdByUserId: string;
};

export type UpdateTextDto = Partial<{
    content: string;
    difficultyLevel: number;
    isApproved: boolean;
}>;

export class TextRepository {
    async create(data: CreateTextDto): Promise<Text> {
        const lang = await prisma.language.findUnique({
            where: { id: data.languageId },
        });
        if (!lang) {
            throw new Error(`Language with id "${data.languageId}" not found.`);
        }
        const user = await prisma.user.findUnique({
            where: { id: data.createdByUserId },
        });
    }
}

```

|      |      |                |        |      |                          |      |
|------|------|----------------|--------|------|--------------------------|------|
|      |      | Українець Д.В. |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      | Морозов А.В.   |        |      |                          |      |
| Змн. | Арк. | № докум.       | Підпис | Дата |                          | 68   |

```

    if (!user) {
      throw new Error(`User with id "${data.createdByUserId}" not found.`);
    }
    return prisma.text.create({ data });
  }

  async findById(id: string): Promise<Text | null> {
    return prisma.text.findUnique({ where: { id } });
  }

  async findAll(): Promise<Text[]> {
    return prisma.text.findMany();
  }

  async findAllApproved(): Promise<Text[]> {
    return prisma.text.findMany({ where: { isApproved: true } });
  }

  async update(id: string, data: UpdateTextDto): Promise<Text> {
    const existing = await this.findById(id);
    if (!existing) {
      throw new Error(`Text with id "${id}" not found.`);
    }
    return prisma.text.update({
      where: { id },
      data,
    });
  }

  async delete(id: string): Promise<Text> {
    const existing = await this.findById(id);
    if (!existing) {
      throw new Error(`Text with id "${id}" not found.`);
    }
    return prisma.text.delete({ where: { id } });
  }

  async findRandom(
    languageId?: string,
    difficultyLevel?: number
  ): Promise<Text | null> {
    const where: Prisma.TextWhereInput = { isApproved: true };
    if (languageId) where.languageId = languageId;
    if (typeof difficultyLevel === "number")
      where.difficultyLevel = difficultyLevel;

    const count = await prisma.text.count({ where });
    if (count === 0) return null;

    const skip = Math.floor(Math.random() * count);
    const [text] = await prisma.text.findMany({
      where,
      take: 1,
      skip,
    });
  }

```

```

    });
    return text;
  }
}

```

### Лістинг UserRepository.ts

```

import { prisma } from "@lib/prisma";
import type { User, Role } from "@generated/prisma";

export type CreateUserDto = {
  username: string;
  email: string;
  passwordHash: string;
  role?: Role;
};

export type UpdateUserDto = Partial<{
  username: string;
  email: string;
  passwordHash: string;
  role: Role;
  isBlocked: boolean;
}>;

export class UserRepository {
  async create(data: CreateUserDto): Promise<User> {
    const existingByEmail = await prisma.user.findUnique({
      where: { email: data.email },
    });
    if (existingByEmail) {
      throw new Error(`User with email "${data.email}" already exists.`);
    }
    const existingByUsername = await prisma.user.findUnique({
      where: { username: data.username },
    });
    if (existingByUsername) {
      throw new Error(`Username "${data.username}" is already taken.`);
    }
    return prisma.user.create({
      data: {
        username: data.username,
        email: data.email,
        passwordHash: data.passwordHash,
        role: data.role ?? "USER",
      },
    });
  }

  async findAll(): Promise<User[]> {
    return prisma.user.findMany();
  }

  async findById(id: string): Promise<User | null> {

```

|      |      |                |        |      |                          |      |
|------|------|----------------|--------|------|--------------------------|------|
|      |      | Українець Д.В. |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      | Морозов А.В.   |        |      |                          |      |
| Змн. | Арк. | № докум.       | Підпис | Дата |                          | 70   |

```

    return prisma.user.findUnique({ where: { id } });
  }

  async findByEmail(email: string): Promise<User | null> {
    return prisma.user.findUnique({ where: { email } });
  }

  async findByUsername(username: string): Promise<User | null> {
    return prisma.user.findUnique({ where: { username } });
  }

  async update(id: string, data: UpdateUserDto): Promise<User> {
    const existing = await this.findById(id);
    if (!existing) {
      throw new Error(`User with id "${id}" not found.`);
    }
    return prisma.user.update({
      where: { id },
      data,
    });
  }

  async delete(id: string): Promise<User> {
    const existing = await this.findById(id);
    if (!existing) {
      throw new Error(`User with id "${id}" not found.`);
    }
    return prisma.user.delete({ where: { id } });
  }

  async toggleBlock(id: string, block: boolean): Promise<User> {
    const existing = await this.findById(id);
    if (!existing) {
      throw new Error(`User with id "${id}" not found.`);
    }
    return prisma.user.update({
      where: { id },
      data: { isBlocked: block },
    });
  }
}

```

#### Лістинг textAction.ts

"use server";

```

import { TextRepository } from "@lib/repositories/TextRepository";

const textRepo = new TextRepository();

export async function createTextAction(
  content: string,
  languageId: string,
  difficultyLevel: number,
  createdByUserId: string

```

|      |      |                |        |      |                          |      |
|------|------|----------------|--------|------|--------------------------|------|
|      |      | Українець Д.В. |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      | Морозов А.В.   |        |      |                          | 71   |
| Змн. | Арк. | № докум.       | Підпис | Дата |                          |      |

```

) {
  return textRepo.create({
    content,
    languageId,
    difficultyLevel,
    createdByUserId,
  });
}

export async function getTextByIdAction(id: string) {
  return textRepo.findById(id);
}

export async function getAllTextsAction() {
  return textRepo.findAll();
}

export async function getApprovedTextsAction() {
  return textRepo.findAllApproved();
}

export async function updateTextAction(
  id: string,
  data: Partial<{
    content: string;
    difficultyLevel: number;
    isApproved: boolean;
  }>
) {
  return textRepo.update(id, data);
}

export async function deleteTextAction(id: string) {
  return textRepo.delete(id);
}

export async function getRandomTextAction(
  languageId?: string,
  difficultyLevel?: number
) {
  const text = await textRepo.findRandom(languageId, difficultyLevel);
  if (!text) throw new Error("Text not found");
  return text;
}

```

#### Лістинг userAction.ts

```

"use server";

import { UserRepository } from "@lib/repositories/UserRepository";
import type {
  CreateUserDto,
  UpdateUserDto,
} from "@lib/repositories/UserRepository";

```

|      |      |                |        |      |                          |      |
|------|------|----------------|--------|------|--------------------------|------|
|      |      | Українець Д.В. |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      | Морозов А.В.   |        |      |                          |      |
| Змн. | Арк. | № докум.       | Підпис | Дата |                          | 72   |

```

import bcrypt from "bcrypt";

const userRepo = new UserRepository();

export async function registerUserAction(
  username: string,
  email: string,
  password: string
) {
  if (!username || !email || !password) {
    throw new Error("Username, email and password are required");
  }

  const passwordHash = await bcrypt.hash(password, 10);

  const data: CreateUserDto = {
    username,
    email,
    passwordHash,
  };

  return userRepo.create(data);
}

export async function getUserByIdAction(id: string) {
  return userRepo.findById(id);
}

export async function getAllUsersAction() {
  return userRepo.findAll();
}

export async function updateUserAction(id: string, data: UpdateUserDto) {
  return userRepo.update(id, data);
}

export async function deleteUserAction(id: string) {
  return userRepo.delete(id);
}

export async function toggleUserBlockAction(id: string, block: boolean) {
  return userRepo.toggleBlock(id, block);
}

```

#### Лістинг prisma.ts

```

import { PrismaClient } from "@generated/prisma";

const globalForPrisma = global as unknown as { prisma: PrismaClient };
export const prisma = globalForPrisma.prisma || new PrismaClient();
if (process.env.NODE_ENV !== "production") globalForPrisma.prisma = prisma;

```

|      |      |                |        |      |                          |      |
|------|------|----------------|--------|------|--------------------------|------|
|      |      | Українець Д.В. |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      | Морозов А.В.   |        |      |                          |      |
| Змн. | Арк. | № докум.       | Підпис | Дата |                          | 73   |