Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка веб-застосунку для навчання «сліпому» друку з ігровими елементами

Виконав: студент IV курсу, групи СН-41
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

_____ Гумен Ю. П.
(підпис) (прізвище та ініціали)

Керівник _____ Матійчук Л. П.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Шимчук Г.В.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ _____
(підпис) (прізвище та ініціали)

Тернопіль
2025

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«____» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)
за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)
Студенту Гумену Юрію Петровичу
(прізвище, ім'я, по батькові)
1. Тема роботи Розробка веб-застосунку для навчання «сліпому» друку з ігровими елементами

Керівник роботи Матійчук Любомир Павлович, д. е. н., професор кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
Затверджені наказом ректора від «*07*» *травня* 2025 року № *4/7-444*
2. Термін подання студентом завершеної роботи 24 червня 2025 р.
3. Вихідні дані до роботи базуються на основі досліджених літературних та інтернет-джерелах
4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1 Аналіз предметної області та постановка завдання. 1.1 Аналіз предметної області. 1.2 Використання ігрових практик в освітніх веб-застосунках.1.3 Огляд існуючих рішень для навчання друку наосліп. 1.4 Формулювання вимог до веб-застосунку. 1.4.1 Функціональні вимоги. 1.4.2 Нефункціональні вимоги. 1.5 Висновок до першого розділу. 2 Проектування веб-застосунку. 2.1 Обґрунтування вибору технологій розробки. 2.2 Вибір та обґрунтування архітектури веб-застосунку. 2.3 Моделювання варіантів використання. 2.4 Проектування структури бази даних. 2.5 Висновок до другого розділу. 3 Розробка веб-застосунку для навчання «сліпому» друку. 3.1 Початок розробки та структура файлів веб-застосунку. 3.2 Розробка серверної частини. 3.3 Розробка клієнтської частини. 3.4 Висновок до третього розділу. 4 Безпека життєдіяльності, основи охорони праці. Висновки. Перелік джерел. Додатки. 5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Сенчишин В. С.	09.06.2025	13.06.2025

7. Дата видачі завдання 27 січня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	28.01.2025	*Виконано*
2.	Підбір та опрацювання літературних джерел по темі	08.05.2025-15.05.2025	*Виконано*
	кваліфікаційної роботи		
3.	Виконання дослідження щодо розробки веб-	16.05.2025-23.05.2025	*Виконано*
	застосунку для навчання «сліпому» друку з		
	ігровими елементами		
	Розробка веб-застосунку		
4.	Оформлення розділу «Аналіз предметної області та	26.05.2025-13.06.2025	*Виконано*
	постановка завдання»		
5.	Оформлення розділу «Проектування веб-застосунку»	26.05.2025-13.06.2025	*Виконано*
6.	Оформлення розділу «Розробка веб-застосунку для	26.05.2025-13.06.2025	*Виконано*
	навчання «сліпом» друку		
7.	Виконання завдання до підрозділу «Безпека	09.06.2025-13.06.2025	*Виконано*
	життєдіяльності»		
8.	Виконання завдання до підрозділу «Основи охорони	09.06.2025-13.06.2025	*Виконано*
	праці»		
9.	Оформлення кваліфікаційної роботи	10.06.2025-12.06.2025	*Виконано*
10.	Нормоконтроль	13.06.2025-16.06.2025	*Виконано*
11.	Перевірка на плагіат	18.06.2025	*Виконано*
12.	Попередній захист кваліфікаційної роботи	20.06.2025	*Виконано*
13.	Захист кваліфікаційної роботи	25.06.2025	

Студент _____ Гумен Ю. П
(підпис) (прізвище та ініціали)

Керівник роботи _____ Матійчук Л. П.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка веб-застосунку для навчання «сліпому» друку з ігровими елементами // Кваліфікаційна робота освітнього рівня «Бакалавр» // Гумен Юрій Петрович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СН-41 // Тернопіль, 2025 // С. 64, рис. – 16, табл. – 13, слайди – , додат. – 14, бібліогр. – 51.

Ключові слова: сліпий друк, веб-застосунок, гейміфікація, навчання, монорепозиторій, метрики друку, React, tRPC.

Кваліфікаційна робота присвячена створенню веб-застосунку, який допомагає користувачам навчитися друкувати на клавіатурі нвосліп. У першому розділі розглянуто, як зараз навчаються цьому методу, які є труднощі, а також як ігрові елементи можуть зробити навчання цікавішим. Аналіз існуючих рішень дозволив визначити, які функції варто реалізувати у власному застосунку.

У другому розділі кваліфікаційної роботи обґрунтовано вибір технологій і архітектури, спроектовано базу даних, визначено ролі користувачів і описано загальну структуру застосунку у форматі монорепозиторію з трирівневою архітектурою.

У третьому розділі кваліфікаційної роботи описано реалізацію веб-застосунку, включаючи розробку клієнтської і серверної частин, збереження прогресу користувача, а також реалізацію ігрових елементів.

Об'єкт дослідження: процес розробки веб-застосунку для навчання «сліпому» друку з ігровими елементами.

Предмет дослідження: методи та засоби реалізації гейміфікованих навчальних інтерфейсів із використанням сучасних веб-технологій.

ANNOTATION

Development of a Web Application for Touch Typing Training with Gamification Elements // Qualification work of the educational level «Bachelor» // Humen Yurii // Ternopil Ivan Pulyu National Technical University, Computer and Information Systems and Software Engineering Faculty, Computer Sciences Department, group SN-41 // Ternopil, 2025 // P. 64, fig. – 16, tabl. – 13, slides – , annexes. – 14, references – 51.

Keywords: touch typing, web application, gamification, education, monorepo, typing metrics, React, tRPC.

The qualification work is dedicated to the development of a web application that helps users learn touch typing. The first section of the qualification work analyzes current methods of learning this skill, highlights typical difficulties, and explores how gamification elements can make the learning process more engaging. The analysis of existing solutions helped define which features should be implemented in the developed application.

In the second section of the qualification work, the choice of technologies and system architecture is justified. The database structure is designed, user roles are defined, and the overall architecture of the application is described, using a monorepo format and three-tier structure.

The third section describes the implementation of the web application, including the development of the client and server parts, tracking user progress, and the integration of gamification elements.

The object of the qualification work research: the process of developing a web application for touch typing training with gamification elements.

The subject of the qualification work research: methods and tools for implementing gamified educational interfaces using modern web technologies.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API (англ. Application Programming Interface) – програмний інтерфейс прикладного програмування.

ORM (англ. Object-Relational Mapping) – об’єктно-реляційне відображення.

WPM (англ. Words Per Minute) – слів за хвилину.

XP (англ. Experience Points) – очки досвіду.

БД – база даних.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному світі, де швидкість взаємодії з ПК відіграє велику роль, володіння «сліпим» методом друку є надзвичайно цінною навичкою. Оволодіння цим методом дозволяє значно підвищити продуктивність роботи за комп'ютером. Проте саме навчання методу друкування наосліп часто супроводжується низькою мотивацією, через монотонність та повторенню вправ. Водночас додавання ігрових елементів до навчальних застосунків довело свою ефективність. Впровадження гейміфікації в навчальний процес є ефективним рішенням, яке може трансформувати рутинний процес навчання у цікавий та захоплюючий досвід, допомагаючи утримувати високу мотивацію та стабільний прогрес у користувачів.

Мета і задачі дослідження. Ця кваліфікаційна робота бакалавра присвячена розробці веб-застосунку для навчання «сліпому» друку з ігровими елементами. Метою дослідження є створення зрозумілого, інтерактивного та мотивуючого веб-застосунку, який забезпечить ефективне опанування «сліпого» друку за рахунок гейміфікації навчального процесу.

Для досягнення поставленої мети потрібно виконати наступні завдання:

– провести аналіз предметної області та дослідити застосування ігрових практик в освітніх веб-застосунках;

– дослідити та порівняти існуючі веб-платформи для навчання «сліпому» друку;

– сформулювати функціональні та нефункціональні вимоги до майбутнього веб-застосунку;

– обґрунтувати вибір технологічного стеку та архітектури веб-застосунку;

– спроектувати структуру бази даних, враховуючи функціональні та нефункціональні вимоги;

– озробити веб-застосунок згідно з визначеними вимогами та архітектурою;

Практичне значення одержаних результатів. Розроблений веб-застосунок стане корисним інструментом для всіх, хто хоче опанувати метод «сліпого» друку у зручному та інтерактивному форматі. Він може бути використаний: в освітніх закладах для уроків інформатики, людьми, що працюють за ПК і хочуть покращити свої особисті навички, а також у різних корпоративних навчальних платформах.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Аналіз предметної області

«Сліпий» метод друку, або сенсорний друк є методом набору тексту на клавіатурі, не дивлячись на неї. Цей метод спирається на м'язову пам'ять, а не на зорову та передбачає використання всіх десяти пальців. Основна ідея – поділ клавіатури на рівні групи клавіш, які натискаються закріпленими до цих груп пальцями. Такий підхід забезпечує рівномірний розподіл навантаження на руки та пальці, знижуючи загальну напругу в кистях.

Метод ґрунтується на виробленні правильних рефлексів, щоб пальці самі «знали», на яку клавішу натиснути [1]. Ключовою відмінністю друку наосліп від традиційного є постійний фокус погляду на моніторі, а не на клавіатурі. Це дозволяє користувачеві відстежувати текст на моніторі, виправляти помилки та зосередитися на змісті, а не на клавіатурі.

У сучасному світі, де робота з комп'ютером є невід'ємною частиною повсякденної та професійної діяльності опанування «сліпого» друку є надзвичайно цінною навичкою. Ця навичка корисна практично в будь-якій професії, що передбачає роботу з комп'ютером, зокрема для копірайтерів, програмістів, журналістів, перекладачів, офісних працівників. Однією з основних переваг є підвищення швидкості набору тексту [2]. Цей метод друку може заощадити до 21 дня в рік. Це досягається завдяки значному збільшенню швидкості друку. Люди, які друкують звичним способом, зазвичай досягають швидкості 30-40 слів за хвилину, тоді як при друці наосліп – це вже 60-70 слів за хвилину і більше.

Незважаючи на численні переваги сліпого методу друку, сам процес навчання вимагає часу, терпіння та високої мотивації. Однією з головних проблем є монотонність та однотипність вправ, що часто призводить до втрати інтересу й зниження мотивації. Оскільки основний принцип друку наосліп

базується на розвитку м'язової пам'яті через численні повторення, такі завдання з часом сприймаються користувачем як нудні та безглузді. Це призводить до зниження залученості та припинення тренувань.

Крім того, труднощі з утриманням уваги та концентрації становлять ще один виклик. Недбала праця може призвести до формування звички бути неуважним. Нудьга, в свою чергу, може викликати обмежену тривалість уваги та труднощі з концентрацією. Страх зробити помилку також може бути причиною нудьги або відсутності залученості в діяльність, оскільки учні можуть уникати спроб, побоюючись невдачі.

Взаємозв'язок між монотонністю, нудьгою та демотивацією формує суттєвий бар'єр для засвоєння навичок, які потребують тривалої та регулярної практики, таких як «сліпий» друк. Це не просто «незручність», а психологічний механізм, що активно заважає навчанню. Коли монотонність і нудьга призводять до зниження залученості та мотивації, це безпосередньо підриває регулярність і стійкі зусилля, необхідні для розвитку м'язової пам'яті та майстерності. Це створює порочне коло, в якому відсутність задоволення веде до меншої практики, що, в свою чергу, призводить до повільнішого прогресу та подальшої демотивації [3].

1.2 Використання ігрових практик в освітніх веб-застосунках

Гейміфікація – це застосування ігрових практик у неігрових сферах. У контексті освіти гейміфікація використовується для підвищення залученості, мотивації та ефективності навчання. Мета використання ігрових практик в освітніх веб-застосунках перетворити процес навчання з рутинної діяльності на захоплюючий досвід [4]. Гейміфікація працює, використовуючи ігрові механіки для навчання, перетворюючи звичайні завдання на захоплюючі виклики. Основні ігрові механіки, такі як бали, значки, рівні, лідерборди і прогрес-бари, створюють у користувача відчуття досягнення, змагання та просування вперед.

Одним із яскравих прикладів успішної інтеграції ігрових практик у навчальний додаток є «Duolingo». Дослідження показують, що конкретні елементи гейміфікації, такі як серії, рівні, очки досвіду (XP), лінготи, серця, значки та нагороди, значно підвищують залученість та мотивацію користувачів. Серії та значки виявилися особливо ефективними для підтримки послідовних навчальних звичок та досягнення різних етапів у навчанні [5].

Гейміфікація має сильний психологічний вплив на навчання та запам’ятовування інформації. Вона активізує внутрішнє бажання досягати результатів і конкурувати, що збільшує залученість учнів до 60% і покращує засвоєння інформації. Додавання таких механік у навчання в середньому підвищує мотивацію на 22%. Гейміфікація позитивно впливає на когнітивні функції й нейропсихологічне навчання, підкреслюючи важливість чітких цілей, негайного зворотного зв'язку та збалансованих викликів. Це допомагає учням активніше та довше залишатись в процесі навчання, що призводить до більшого обсягу засвоєного матеріалу за менший проміжок часу [6].

Приклад «Duolingo» демонструє, що впровадження ігрових елементів в навчальні системи є потужним інструментом для подолання психологічних бар'єрів, таких як монотонність та низька мотивація, характерних для навчання «сліпому» друку. Гейміфікація перетворює повторювані завдання на захоплюючі виклики, надаючи негайні винагороди, чітке відстеження прогресу та відчуття досягнення. Проте, дослідження також вказують на необхідність балансу. Якщо користувачі більше уваги приділяють нагородам, а не самому процесу навчання, знання з часом можуть не закріплюватися.

1.3 Огляд існуючих рішень для навчання друку наосліп

Сьогодні існує багато онлайн-застосунків, що допомагають опанувати навички сліпого друку та легко почати навчання без особливих труднощів. Більшість платформ є безкоштовними, а їхній інтерфейс зрозумілий та доступний для широкої аудиторії. Регулярна практика є основою для успіху, і

більшість платформ пропонують структуровані вправи та тексти для тренування, що сприяють формуванню м'язової пам'яті.

Одним із найбільш відомих сервісів є «TypingClub». Він є однією з провідних веб-платформ для навчання «сліпому» друку, доступною через веб-браузер та додаток для iOS. Платформа пропонує віртуальну клавіатуру та посібники для рук, які візуально демонструють правильне положення пальців на клавіатурі для кожної клавіші. Це особливо важливо для початківців, оскільки допомагає їм формувати правильні звички. «TypingClub» підтримує різні розкладки клавіатури, такі як «QWERTY» та «Dvorak», а також пропонує різні плани уроків, включаючи навчання друку однією рукою та програму «Jungle Junior» для молодших учнів [7].

Додаткові функції, такі як голосовий супровід та анімовані історії, роблять процес навчання більш захоплюючим. Платформа також надає можливість навчатись друку наосліп різними мовами. «TypingClub» працює за моделлю «Freemium», надаючи безкоштовний доступ до основної програми, а також пропонуючи необов’язкову платну версію для шкіл.

На рисунку 1.1 представлено інтерфейс веб-платформи для навчання «TypingClub».

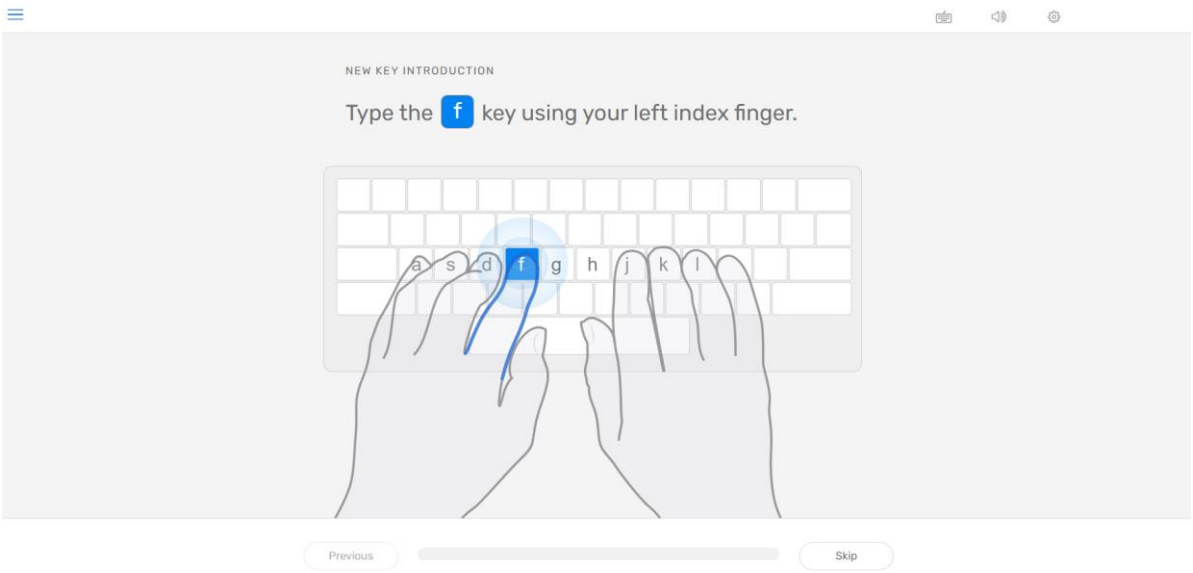

Рисунок 1.1 – Інтерфейс веб-платформи для навчання «TypingClub»

Підхід «TypingClu»b до навчання є гнучким, що дозволяє користувачам навчатися у власному темпі з будь-якого місця, де є доступ до Інтернету. Платформа активно використовує елементи гейміфікації, такі як: значки, рівні та досягнення, що підвищує задоволення від навчання та підтримує мотивацію. Система рівнів, значків та зірок слугує не лише для мотивації, але й для візуального підтвердження прогресу, що допомагає відслідковувати прогрес навчання.

«Monkeytype» – це інструмент, що виділяється мінімалістичним і сучасним підходом до навчання. Користувач має можливість налаштовувати весь застосунок під себе, включаючи вибір теми оформлення, стилів курсору, звукових ефектів, фонових зображень, а також активувати або деактивувати таймер, підказки та інші елементи інтерфейсу. Така персоналізація сприяє створенню максимально комфортного середовища для друку відповідно до особистих вподобань.

На рисунку 1.2 представлено інтерфейс веб-застосунку «Monkeytype».

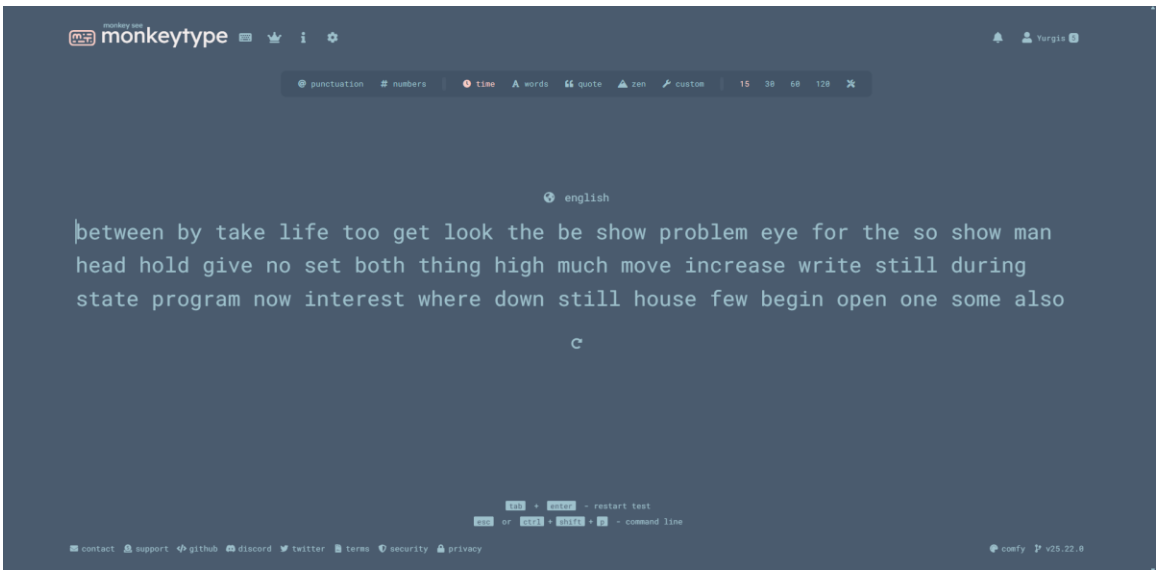

Рисунок 1.2 – Інтерфейс веб-застосунку «Monkeytype»

«Monkeytype» надає широкий спектр можливостей для тренування, що впливають на процес навчання. Доступні режими: «Час», «Слова», «Цитата», «Дзен» та «Користувацький», а також розважальні «Funbox Modes». Застосунок

пропонує різні рівні складності, зокрема «Normal», «Expert», де помилка у слові зупиняє тест, і «Master», що потребує абсолютної точності, зупиняючи тестування на будь-якій помилці. Функції «Stop on Error» – зупинка при помилці в літері або слові та «Confidence Mode» – відсутність можливості виправлення помилок, дозволяють користувачам адаптувати рівень складності відповідно до власних потреб та цілей. Застосунок підтримує численні мови та різні списки слів [8].

Всі результати навчання зберігаються в акаунті користувача. Після завершення тесту на екрані результатів відображаються основні метрики: слова за хвилину (WPM), точність, статистика символів, тривалість тесту та інформація про лідерборди. Графіки демонструють зміни WPM протягом тесту, що дозволяє візуалізувати динаміку продуктивності. Лідерборди, загальні та щоденні, додають змагальний елемент, спонукаючи користувачів покращувати свої результати. Крім того, існує функція перегляду запису процесу тестування, де можна побачити, як саме користувач натискав на клавіші та які помилки було допущено.

На відміну від попередніх рішень, «Keybr» – це інструмент, що вирізняється ретельною статистикою набору тексту та оригінальним підходом до адаптивної генерації слів. Застосунок ретельно відстежує всі дані набору тексту з високою деталізацією. Користувачі можуть переглядати інформацію про точність і швидкість набору для кожної літери окремо. Це дає змогу виокремити слабкі сторони та зосередитись на їх покращенні.

Ключова особливість «Keybr» – це адаптивна генерація слів, яка базується на поточному рівні навичок користувача. Цей метод дає змогу звернути увагу на слабкі сторони учня, пропонуючи слова, що містять літери чи сполучення, які викликають труднощі. Також «Keybr» підтримує велику кількість мов, що розширює його сферу застосування. Застосунок також пропонує функцію, яка дозволяє користувачам змагатися з іншими у реальному часі, відточуючи швидкість друку. Преміум акаунт дає змогу видалити рекламу. Статистика кожної літери є ключовим засобом зворотного зв'язку з

користувачем, надаючи можливість незалежно відстежувати власний прогрес і змінювати процес навчання відповідно до отриманих результатів [9].

На рисунку 1.3 представлено інтерфейс веб-застосунку для навчання «Keybr».

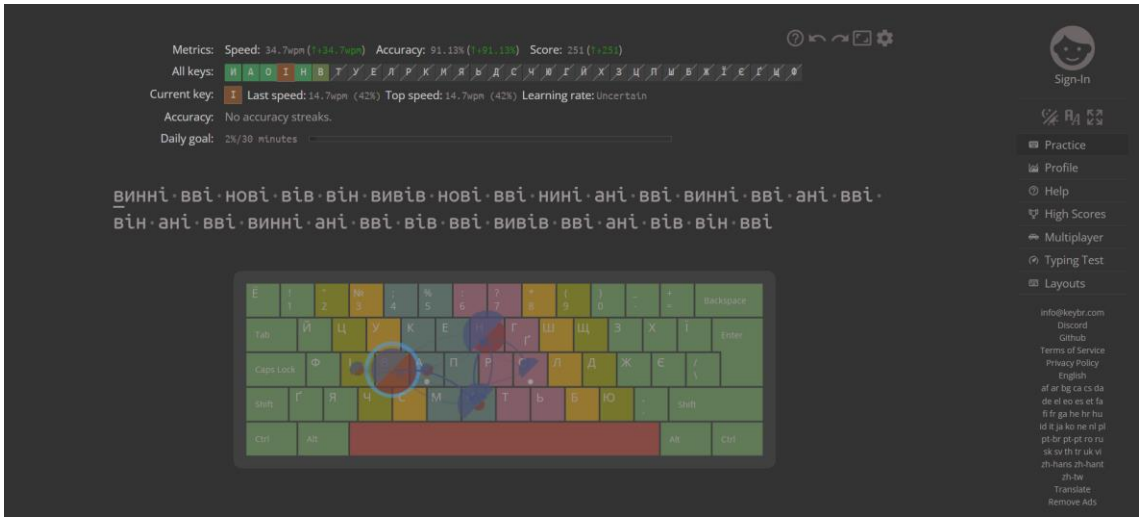

Рисунок 1.3 – Інтерфейс веб-застосунку для навчання «Keybr»

Усі розглянуті системи спільні у тому, що вони надають користувачам зручні інструменти для самостійного навчання «сліпому» друку, а також пропонують зворотний зв’язок у реальному часі та елементи мотивації для підтримання інтересу до занять.

1.4 Формулювання вимог до веб-застосунку

На основі аналізу предметної області та наявних рішень були сформульовані вимоги до системи. Ці вимоги поділяються на функціональні та нефункціональні.

1.4.1 Функціональні вимоги

Веб-застосунок повинен забезпечувати наступні функціональні вимоги. Користувач має мати можливість проходити інтерактивні заняття, які

складаються з окремих екранів з різноманітними режимами. Заняття поділяються на три рівні складності. Під час навчання система повинна надавати підказки, зокрема відображати віртуальну клавіатуру з підсвічуванням клавіш, які потрібно натиснути. В екранах уроку відображаються метрики у реальному часі залежно від дій користувача. Після завершення кожного заняття користувач отримуватиме загальні метрики у вигляді показників ефективності, таких як: середня швидкість друку, середня точність, кількість помилок, загальний час та інші метрики. Усі результати уроків мають зберігатися, а користувач повинен мати змогу повернутись на останній завершений етап уроку.

Для відкриття доступу до нових уроків потрібно проходити попередні заняття. Під час навчання нараховуються XP, які сприяють підвищенню рівня користувача. При цьому XP не можуть нараховуватись повторно за один і той самий урок, що дозволяє уникнути зловживань і забезпечує баланс.

Також передбачено наявність особистого профілю, де відображається загальна статистика, слабкі літери, отримані досягнення та візуальна карта активності як у GitHub. Авторизація має відбуватися через сторонні сервіси, зокрема Google і GitHub, оскільки більшість користувачів вже мають облікові записи в цих системах, що спрощує процес реєстрації. Крім того, система повинна підтримувати взаємодію як зареєстрованими користувачами, так і з гістьми.

1.4.2 Нефункціональні вимоги

Веб-застосунок повинен відповідати наступним нефункціональним вимогам. Він має забезпечувати високу швидкість завантаження та плавну роботу інтерфейсу, особливо під час набору тексту та відображення живих метрик. Інтерфейс він має бути простим, мінімалістичним, інтуїтивно зрозумілим та приємним у використанні. Повинен бути доступним через

сучасні веб-браузери на настільних пристроях. Мобільна адаптація не є обов’язковою.

Усі дані користувачів, такі як: прогрес навчання, показники ефективності та досягнення, мають зберігатися надійно та без втрат. Застосунок повинний забезпечувати їх цілісність, доступність та стійкість до збоїв. Архітектура має бути спроектована таким чином, щоб забезпечити можливість подальшого розширення функціоналу та ефективної роботи при значному збільшенні кількості користувачів. Це передбачає здатність системи адаптуватися до зростаючих навантажень без суттєвого зниження продуктивності.

Код веб-застосунку повинен бути структурованим, зрозумілим та легким для модифікації. Це забезпечить можливість ефективного оновлення функціоналу, виправлення помилок та подальшого розвитку.

1.5 Висновок до першого розділу

У першому розділі було проаналізовано особливості навчання методу друку наосліп, зокрема труднощі, пов’язані з втомою, монотонністю та втратою мотивації. Було обґрунтовано доцільність використання гейміфікації як засобу підвищення залученості та ефективності навчання. Розглянуто приклади успішного застосування ігрових механік в освітніх платформах, таких як «Duolingo», та проаналізовано їх вплив на користувацьку мотивацію.

Також проведено огляд існуючих рішень: «TypingClub», «Keybr», «Monkeytype», з акцентом на їх функціональні можливості, сильні сторони та недоліки. На основі цього аналізу сформульовано вимоги до власного веб-застосунку для навчання «сліпому» друку, що поєднує освітній контент з ігровими елементами, які стали основою подальшого проектування системи.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ВЕБ-ЗАСТОСУНКУ

2.1 Обґрунтування вибору технологій розробки

Вибір технологічного стеку визначає архітектуру та структуру майбутнього веб-застосунку, оскільки від обраних технологій залежить масштабованість, продуктивність і безпека. Тому стек підбирається з врахуванням поставлених раніше функціональних і нефункціональних вимог.

Для створення користувацького інтерфейсу було обрано React. Ця бібліотека набула широкої популярності завдяки своїй компонентній структурі, яка дає змогу ділити інтерфейс застосунку на невеликі, незалежні частини. Такий метод значно спрощує процес розробки, так як дає можливість використовувати компоненти декілька разів. React дає змогу створювати динамічні інтерфейси, які оновлюються без перезавантаження сторінки, що підходить для швидкого оновлення метрик користувача [10-11].

Основною мовою програмування було обрано TypeScript, щоб збільшити зручність та надійність написання коду. TypeScript додає статичну типізацію поверх JavaScript, що дозволяє уникати більшості помилок у коді. Завдяки статичній типізації код стає зрозумілішим та легшим для підтримки у майбутньому [12].

Оскільки у застосунку для навчання друку наосліп важливо моментально реагувати на зміни метрик, прогресу користувача, зміни екранів тощо, зберігаючи при цьому актуальний стан. Для цих завдань було обрано бібліотеку керування станом застосунку – Zustand. Невеликий розмір пакета дозволяє клієнтській частині завантажуватись швидше, що підвищує продуктивність застосунку [13]. Також Zustand легко інтегрувати без зайвого шаблонного коду, на відміну від інших бібліотек для керування станом, а його структура полегшує розширення функціоналу [14].

Для швидкої розробки клієнтської частини та зменшення витрат часу на створення базових елементів інтерфейсу було обрано Shadcn UI у поєднанні з Tailwind CSS.

Tailwind CSS дає набір спеціальних класів, що прискорює створення адаптивних компонентів без потреби зміни каскадних таблиць стилів [15]. В свою чергу Shadcn UI, побудований за допомогою Tailwind CSS і бібліотеки Radix, надає готові компоненти, які можна підключати окремо до проекту в залежності від потреб [16]. Це дозволяє гарантувати єдиний стиль у всьому застосунку. Завдяки такому підходу розробка й підтримка інтерфейсу стають легшими, а дизайн відповідає сучасним стандартам доступності та зручності.

Для кращого досвіду використання веб-застосунку користувачем, обрано бібліотеку Framer Motion, для створення плавних та красивих анімацій [17]. Framer Motion є бібліотекою написаною для React, яка дозволяє легко, без зайвих зусиль, створювати складні анімації. Плавні анімації роблять інтерфейс приємнішими для взаємодії [18].

Щоб взаємодіяти з сервером, отримувати та відправляти запити, буде використовуватись бібліотека React Query. Ця бібліотека автоматично кешує дані, які надходять з сервера, це допомагає зменшити кількість запитів на сервер. Також React Query слідкує за змінами та автоматично оновлює та синхронізує дані сервера з клієнтом. Завдяки цьому немає потреби окремо виділяти місце для зберігання даних з сервера, тому що бібліотека сама відповідає за їхню актуальність та кешування [19-20].

Сучасний збірник для клієнтської частини Vite був обраний для швидкої та оптимізованої збірки проекту. Vite підтримує функцію Hot Module Replacement (HMR), що дозволяє миттєво та точно застосовувати зміни без повного перезавантаження сторінки та втрати стану веб-застосунку [21]. Це збільшує швидкість розробки, зменшуючи час очікування для відображення зроблених змін у браузері. Ще одним плюсом є підтримка TypeScript з «коробки», що зменшує час налаштування проекту [22].

Для створення серверної частини веб-застосунку було обрано платформу Node.js у парі з фреймворком Express [23-24]. Node.js дозволяє виконувати JavaScript скрипти на сервері, що дає можливість писати клієнтську та серверну частини з використанням однієї мови програмування . Express, у свою чергу, надає гнучку та просту основу для побудови API . Для збільшення зручності розробки обрано монорепозиторій на базі Turborepo. Він дозволяє зберігати клієнтський і серверний код у єдиному репозиторії. Це дає змогу використовувати спільні пакети як для серверної частини, так і для клієнтської. Також стає зручніше керувати залежностями застосунку [25].

Використання монорепозиторію та TypeScript в обох частинах застосунку, дозволяє легко інтегрувати tRPC. Це фреймворк, за допомогою якого можна створювати повністю типізовані API. Це можливо, тому що обидві частини проекту знаходяться поруч, у одному репозиторію і діляться одними й тими самими TypeScript-типами. Також tRPC значно зменшує вірогідність виникнення помилок зв’язаних з неправильним передаванням даних між сервером і клієнтом. Як результат, розробка стає швидшою та зручнішою, оскільки програміст отримує підказки під час написання коду [26].

Дані для уроків, метрик і користувачів, потрібно десь зберігати, для цього було обрано NoSQL базу даних MongoDB. Ця БД є документо-орієнтованою, тобто вона зберігає дані у JSON-подібному форматі, що дає велику гнучкість для застосунків, де структура даних може часто мінятися [27].

Для зручнішої роботи з БД буде використано Prisma ORM. Ця бібліотека надає зручний інтерфейс, який спрощує взаємодію з базою даних. Prisma сама генерує TypeScript-типи на основі описаною схеми таблиць БД [28].

Також для валідації даних на сервері та клієнті буде використовуватись бібліотека Zod. Вона створена для опису правил валідації даних, що дає уникнути повторювання логіки перевірки введених даних і гарантує, що дані, які надсилаються від клієнта до сервера відповідають однаковим правилам [29].

Крім того, у веб-застосунку використовується бібліотека better-auth, яка дає змогу просто додати авторизації через сторонні сервіси, такі як Google або

GitHub. Вона також повністю підтримує TypeScript та Prisma ORM, що дозволяє безпечно працювати з сесіями користувача та мінімізує кількість помилок при роботі з аутентифікацією [30].

У результаті, завдяки суміші всіх обраних раніше технологій весь застосунок стає «типізованим». Це означає, що вся інформація від БД до клієнта має чітко встановлений тип, що зменшує кількість помилок і робить проект надійнішим і легшим для масштабування.

2.2 Вибір та обґрунтування архітектури веб-застосунку

Архітектура веб-застосунку для навчання «сліпому» друку є основою на якій будується вся система. Вона визначає структуру, компоненти та принципи взаємодії між елементами системи. Вдало спроектована архітектура гарантує ефективність, масштабованість, надійність і гнучкість програмного продукту.

Для реалізації веб-застосунку навчання «сліпому» друку з ігровими елементами було обрано трирівневу архітектуру, що є класичним методом розробки, який був перевіреним часом. Ця архітектура ділить застосунок на три логічні, взаємопов’язані та незалежні компоненти: рівень представлення (клієнт), рівень логіки (сервер) та рівень даних (БД). Такий поділ дає чітко розмежувати відповідальність, що спрощує розробку, тестування та подальшу підтримку проекту [31].

Веб-застосунок буде складатися з трьох ключових компонентів:

– Клієнтська частина (Рівень представлення) – це компонент, з яким взаємодіє користувач через свій браузер. Вона буде реалізована як односторінковий застосунок за допомогою бібліотеки React. Клієнтська частина буде відповідати за відображення всього інтерфейсу.

– Серверна частина (Рівень логіки) – це ядро застосунку, що містить всю бізнес-логіку. Для її побудови буде використана платформа Node.js та фреймворк Express.js. Сервер оброблятиме HTTP-запити від клієнта,

проводитиме автентифікацію користувачів, керуватиме прогресом навчання, видачою досягнень та буде взаємодіяти з БД.

– База даних (Рівень даних) – компонент, що відповідає за зберігання інформації. Взаємодія з БД буде здійснюватися через Prisma ORM, що забезпечить безпеку та типізацію запитів.

На рисунку 2.1 зображено схема трирівневої архітектури в контексті веб-застосунку для навчання «сліпому» друку.

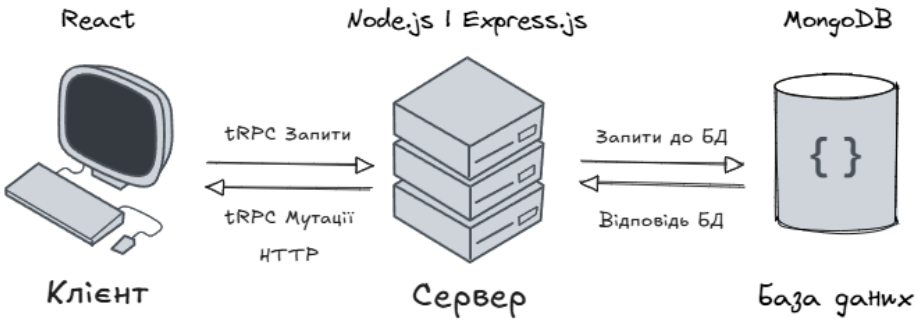

Рисунок 2.1 – Схема трирівневої архітектури

Для ефективної реалізації описаної вище трирівневої архітектури в одному проекті буде застосовано структуру монорепозиторію. Такий підхід дозволяє об’єднати рівень представлення, рівень логіки, рівень даних та допоміжні спільні модулі в одному спільному репозиторії. Це значно спрощує взаємодію між рівнями та полегшує розробку [32-33].

Монорепозиторій дуже ефективний у проектах, що використовують TypeScript, оскільки дає змогу спільно використовувати типи між клієнтом і сервером. Структура застосунку буде поділена на додатки та спільні пакети. У додатках будуть знаходитись клієнтська та серверна частини. У спільних пакетах, у свою чергу, буде знаходитись пакет для роботи з БД, винесені UI-компоненти та пакет з tRPC роутерами, що буде мати опис маршрутів, запитів і відповідей, а також типи, що використовуються клієнтом і сервером.

На рисунку 2.2 зображено схему структури монорепозиторію.

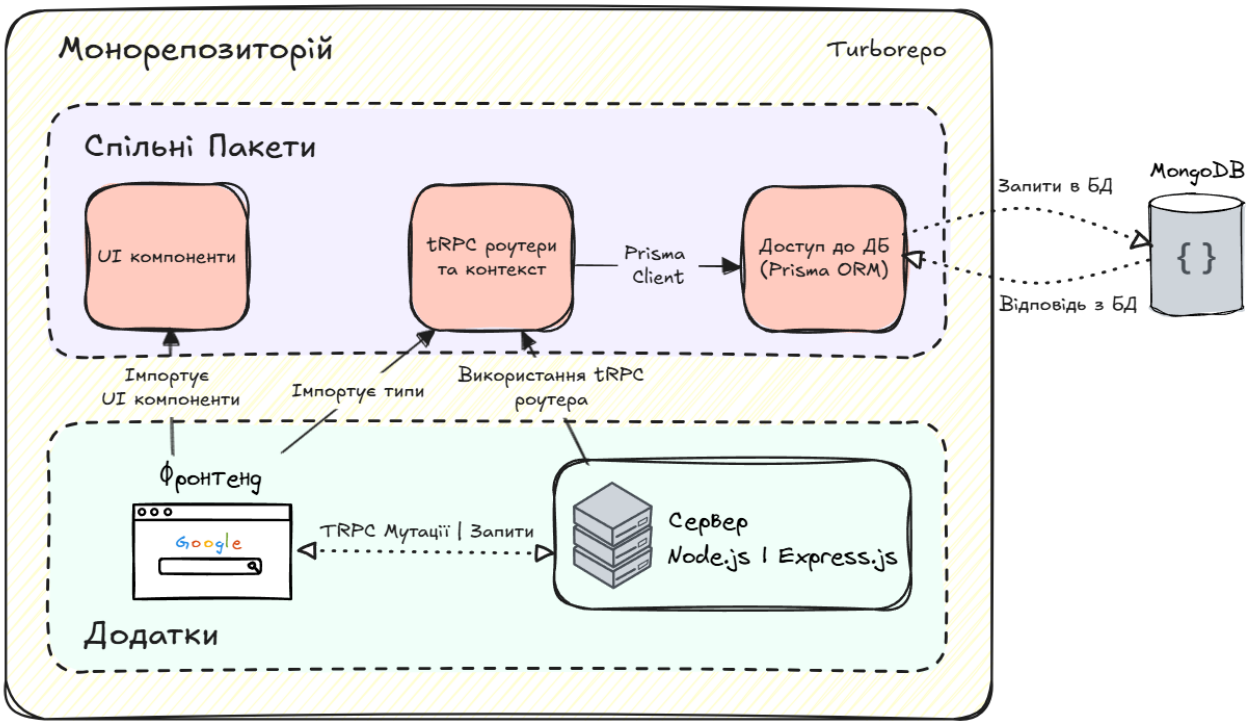

Рисунок 2.2 – Схема структури репозиторію

Таким чином, поєднання класичної трирівневої архітектури з сучасним підходом у вигляді монорепозиторію дозволяє забезпечити як технологічну чистоту веб-застосунку для навчання «сліпому» друку, так і зручність у його підтримці та масштабуванні.

2.3 Моделювання варіантів використання

Моделювання варіантів використання є основним етапом у проектуванні будь-якого програмного забезпечення, що дозволяє показати функціональні вимоги до системи з точки зору її користувачів.

Для веб-застосунку, що призначений для навчання «сліпому» друку з ігровими елементами, моделювання варіантів використання дозволяє чітко визначити, як користувачі взаємодіють із системою. До таких цілей належать проходження уроків, відстеження прогресу, отримання досвіду та досягнень, а також управління своїм профілем. Це гарантує, що всі функціональні вимоги, сформульовані раніше, будуть повноцінно реалізовані у поведінці застосунку.

Для моделювання варіантів використання потрібно визначити акторів застосунку. Основна ціль пошуку акторів полягає в тому, щоб знайти типи користувачі, які взаємодіють із застосунком, і знайти різницю в їхній поведінці та доступному функціоналі [34].

Для веб-застосунку для навчання «сліпому» друку було визначено наступних акторів: зареєстрований користувач та гість.

На рисунку 2.3 зображено акторів веб-застосунку.

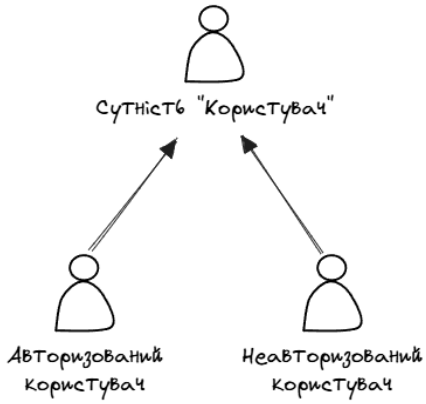

Рисунок 2.3 – Актори веб-застосунку

Зареєстрований користувач є основним користувачем системи, який має повний доступ до всього функціоналу застосунку. Гість, в свою чергу, немає доступу до всіх функцій. Далі було спроектовано діаграми використання для кожного актора. На рисунку 2.4 зображено діаграму використання для неавторизованого користувача.

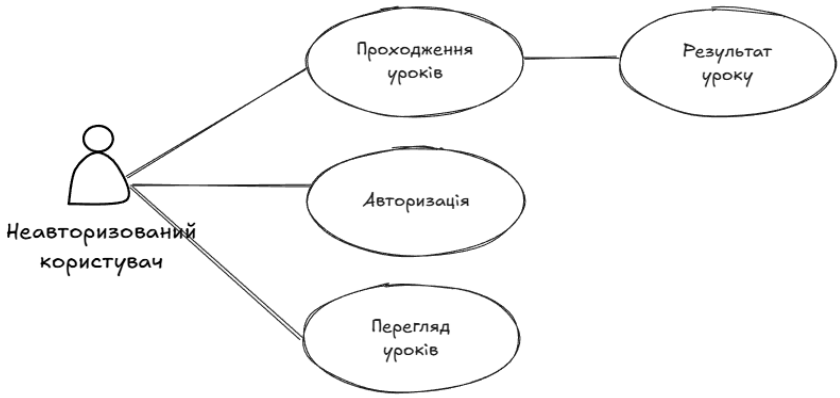

Рисунок 2.4 – Діаграма використання для неавторизованого користувача

На рисунку 2.5 зображено діаграму використання для авторизованого користувача.

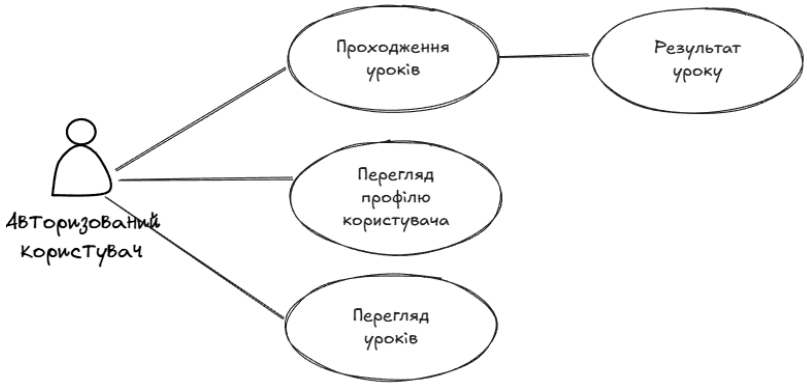

Рисунок 2.5 – Діаграма використання для авторизованого користувача

Оскільки для авторизованого користувача авторизація вже пройдена, цей актор має доступ до розширених можливостей: перегляд особистого профілю з детальною аналітикою: heatmap активності, досвід, рівень та отримані досягнення. Всі інші можливості цього актора залишаються тими самими.

Повну діаграму варіантів використання інтернет-магазину зображено на рисунку 2.6.

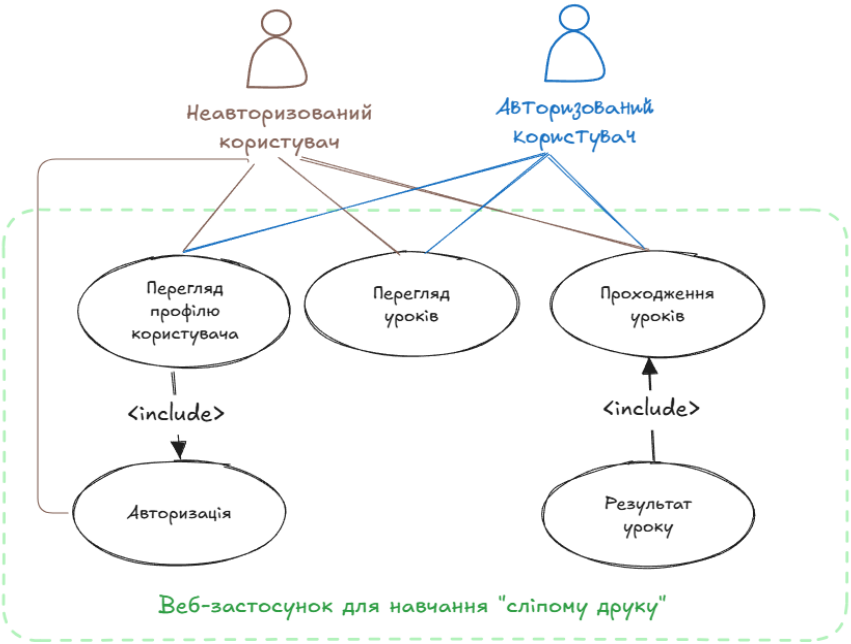

Рисунок 2.6 – Діаграма варіантів використання

Створення цих взаємодій на діаграмі варіантів використання допомагає чітко представити функціональні межі веб-застосунку та ролі акторів у ній.

2.4 Проектування структури бази даних

Оскільки для розробки застосунку було обрано MongoDB, документо-орієнтовану БД, а не реляційну, вона не використовує «таблиці» в їх звичайному розумінні, а працює з колекціями. У кожному документі дані організовано у вигляді пар «ключ-значення», які в MongoDB називають полями або атрибутами.

Дивлячись на функціональні вимоги, було спроектовано БД веб-застосунку для навчання друку наосліп з ігровими елементами. Для зручності розуміння структури БД, колекції було поділено на логічні групи за їх призначенням і використанням:

– Облікові записи та автентифікація – це колекції, які містять інформацію про користувача, автентифікацію, записи сесій та зовнішні акаунти.

– Уроки та прогрес – колекції, які зберігають уроки, екрани і також записують метрики, та прогрес користувача.

– Активність та досягнення – це колекції для зберігання різної статистики, активності та досягнень користувача.

Загалом БД буде складатись з одинадцяти таблиць: User, Session, Account, Verification, Lesson, UserLessonProgress, ScreenMetrics, UserStats, UserAchievement, UserDailyActivity, CharacterMetric.

Проектування структури колекцій було почато з групи «Облікові записи та автентифікація».

Колекція User зберігає загальну інформацію про користувача. Вона має інформацію про ім'я користувача, адресу електронної пошти, статус підтвердження електронної пошти, посилання на зображення, роль, статус блокування та причини блокування, а також час створення та оновлення запису.

В таблиці 2.1 наведено атрибути колекції User.

Таблиця 2.1 – Атрибути колекції User

Назва атрибуту	**Тип даних**	**Опис**
id	String	Унікальний ідентифікатор користувача
name	String	Ім’я користувача
email	String	Адреса електронної пошти користувача
emailVerified	Boolean	Статус підтвердження електронної пошти
image	String	URL зображення профілю користувача
role	String	Роль користувача у системі
banned	Boolean	Статус блокування користувача
banReason	String	Причина блокування
banExpires	DateTime	Дата закінчення блокування
createdAt	DateTime	Дата та час створення запису
updatedAt	DateTime	Дата та час останнього оновлення запису

Ця колекція є центральною у веб-застосунку та забезпечує логічний зв’язок із іншими колекціями, які зберігають дані про сесії, акаунти, прогрес у навчанні та активність.

Колекція Session слугує для зберігання даних про сесії користувачів, що використовуються з метою аутентифікації. Кожен документ у цій колекції містить інформацію про унікальну сесію, зокрема токен сесії, термін її дії, час створення, а також дані про клієнтський браузер або додаток.

В таблиці 2.2 наведено атрибути колекції Session.

Таблиця 2.2 – Атрибути колекції Session

Назва атрибуту	**Тип даних**	**Опис**
1	**2**	**3**
Id	String	Унікальний ідентифікатор сесії
expiresAt	DateTime	Дата та час закінчення дії сесії
token	String	Унікальний токен сесії
createdAt	DateTime	Дата та час створення сесії
updatedAt	DateTime	Дата та час останнього оновлення сесії

Продовження таблиці 2.2

1	2	3
ipAddress	String	IP-адреса пристрою користувача
userAgent	String	Інформація про браузер або пристрій
userId	String	Ідентифікатор користувача, що належить сесії
impersonatedBy	String	Ідентифікатор користувача, від імені якого здійснюється сесія

Колекція Session зберігає інформацію як про активні, так і про минулі сесії користувачів, що дозволяє реалізувати контроль доступу до веб-застосунку, відстежувати активність користувача та забезпечувати безпеку.

Колекція Verification містить документи, які зберігають унікальні ідентифікатори, значення для верифікації та терміни дії таких записів.

У таблиці 2.3 наведено атрибути колекції Verification.

Таблиця 2.3 – Атрибути колекції Verification

Назва атрибута	Тип даних	Опис
id	String	Унікальний ідентифікатор запису
identifier	String	Ідентифікатор для верифікації
value	String	Значення для верифікації
expiresAt	DateTime	Дата та час закінчення терміну дії запису

Колекція Verification зберігає тимчасову інформацію для підтвердження електронної пошти або скидання паролю.

Колекція Account містить дані про акаунти користувачів, які прив'язані до зовнішніх сервісів, на кшталт Google або GitHub. Також саме тут зберігаються відомості, необхідні для керування OAuth-токенами.

В таблиці 2.4 наведено атрибути колекції Account.

Таблиця 2.4 – Атрибути колекції Account

Назва атрибуту	Тип даних	Опис
1	2	3
id	String	Унікальний ідентифікатор запису акаунту

Продовження таблиці 2.4

1	2	3
accountId	String	Ідентифікатор акаунту у провайдера
providerId	String	Ідентифікатор провайдера
userId	String	Ідентифікатор користувача
accessToken	String	Токен доступу OAuth
refreshToken	String	Токен оновлення Oauth
idToken	String	ID-токен Oauth
accessTokenExpiresAt	DateTime	Дата і час закінчення терміну дії токена доступу
refreshTokenExpiresAt	DateTime	Дата і час закінчення терміну дії токена оновлення
scope	String	Область дії (scope) токена
password	String	Хеш пароля
createdAt	DateTime	Дата і час створення запису
updatedAt	DateTime	Дата і час останнього оновлення запису

Колекція Account слугує для зберігання інформації про способи аутентифікації користувача через зовнішні сервіси, підтримки OAuth-механізму, та забезпечує можливість керування токенами доступу, підтримуючи безперебійну роботу авторизації.

Наступною було спроектовано групу колекцій «Уроки та прогрес». Колекція Lesson зберігає загальну інформацію про уроки. Вона містить назву, опис, рівень складності та масив екранів кожного уроку.

У таблиці 2.5 наведено атрибути колекції Lesson.

Таблиця 2.5 – Атрибути колекції Lesson

Назва атрибуту	Тип даних	Опис
id	String	Унікальний ідентифікатор уроку
title	String	Назва уроку
difficulty	String	Рівень складності
order	Int	Порядковий номер уроку
screens	Array<Object>	Масив екранних блоків уроку, кожен описує окремий екран

Колекція UserLessonProgress відстежує прогрес кожного користувача в конкретному уроці. Ця колекція допомагає слідкувати, на якому етапі навчання зупинився користувач.

У таблиці 2.6 наведено атрибути колекції UserLessonProgress.

Таблиця 2.6 – Атрибути колекції UserLessonProgress

Назва атрибуту	**Тип даних**	**Опис**
id	String	Унікальний ідентифікатор прогресу користувача в уроці
userId	String	Ідентифікатор користувача
lessonId	String	Ідентифікатор уроку
currentScreenOrder	Int	Порядковий номер поточного екрану, на якому зупинився користувач
isCompleted	Boolean	Статус завершення уроку користувачем
completedAt	DateTime	Дата та час завершення уроку
totalRawWPM	Float	Загальна сира швидкість друку
totalAdjustedWPM	Float	Загальна скоригована швидкість друку
totalAccuracy	Float	Загальна точність друку у відсотках
totalBackspaces	Int	Загальна кількість натискань клавіші Backspace
totalErrors	Int	Загальна кількість допущених помилок
totalTimeTaken	Int	Загальний час, витрачений на урок
totalTypedCharacters	Int	Загальна кількість набраних символів
totalCorrectCharacters	Int	Загальна кількість правильно набраних символів
createdAt	DateTime	Дата та час створення запису
updatedAt	DateTime	Дата та час останнього оновлення запису

Далі колекція ScreenMetrics зберігає детальні метрики друку для кожного екрану пройденого користувачем уроку. Ця колекція є основним місцем для збору статистики про продуктивність користувача на кожному екрані заняття.

У таблиці 2.7 наведено атрибути колекції ScreenMetrics.

Таблиця 2.7 – Атрибути колекції ScreenMetrics

Назва атрибуту	Тип даних	Опис
id	String	Унікальний ідентифікатор метрики
userId	String	Ідентифікатор користувача
lessonId	String	Ідентифікатор уроку
screenId	String	Ідентифікатор екрану уроку
rawWPM	Number	Сира швидкість набору (слова на хвилину)
adjustedWPM	Number	Відкоригована швидкість набору
accuracy	Number	Точність (%)
errors	Number	Кількість помилок
backspaces	Number	Кількість видалень символів
timeTaken	Number	Час проходження екрану у секундах
createdAt	DateTime	Дата та час запису метрики

Фінальною є група «Статистика та активність». Колекція UserStats об’єднує загальну статистику користувача в одне місце, таку як: загальний досвід, поточний рівень, досвід до наступного рівня, кількість завершених уроків, найвищу швидкість та точність друку для різних рівнів складності та нинішню та найдовшу серію щоденної активності. Таблицю атрибутів колекції UserStats розміщено в додатку А.

Колекція UserDailyActivity відстежує та зберігає щоденну активність користувача, включаючи кількість завершених уроків, пройдених екранів та зароблено досвіду в конкретний день.

У таблиці 2.8 наведено атрибути колекції UserDailyActivity.

Таблиця 2.8 – Атрибути колекції UserDailyActivity

Назва атрибуту	Тип даних	Опис
_id	String	Унікальний ідентифікатор щоденної активності
userId	String	Ідентифікатор користувача
date	DateTime	Дата активності
lessonsCompleted	Int	Кількість уроків, завершених за день
screensCompleted	Int	Кількість екранів, пройдених за день
xpEarnedToday	Int	Кількість досвіду, зробленого за день

Колекція CharacterMetric зберігає детальні метрики для кожного символу, щоб пізніше користувач мав змогу бачити свої слабкі місця.

У таблиці 2.9 наведено атрибути колекції UserDailyActivity.

Таблиця 2.9 – Атрибути колекції UserDailyActivity

Назва атрибуту	**Тип даних**	**Опис**
id	String	Унікальний ідентифікатор
userId	String	Ідентифікатор користувача
character	String	Символ, для якого збирається метрика
correctCount	Int	Кількість правильних натискань цього символу
errorCount	Int	Кількість помилок при натискані цього символу

Колекція UserAchievement реєструє досягнення, які розблокував користувач. Кожен запис містить отримене досягнення, час його розблокування та ідентифікатор досягнення.

У таблиці 2.10 наведено атрибути колекції UserAchievement.

Таблиця 2.10 – Атрибути колекції UserAchievement

Назва атрибуту	**Тип даних**	**Опис**
id	String	Унікальний ідентифікатор
userId	String	Ідентифікатор користувача
achievementId	String	Ідентифікатор розблокованого досягнення
unlockedAt	DateTime	Дата та час розблокування досягнення

Тут варто зазначити, що самі унікальні досягнення та умови їх отримання будуть зберігатися на сервері, а в БД зберігатимуться тільки записи, яке саме досягнення було отримано користувачем та коли.

Після визначення та опису усіх сутностей веб-застосунку для навчання «сліпому» друку, можна розробити повну діаграму БД.

На рисунку 2.7 зображено діаграму класів БД.

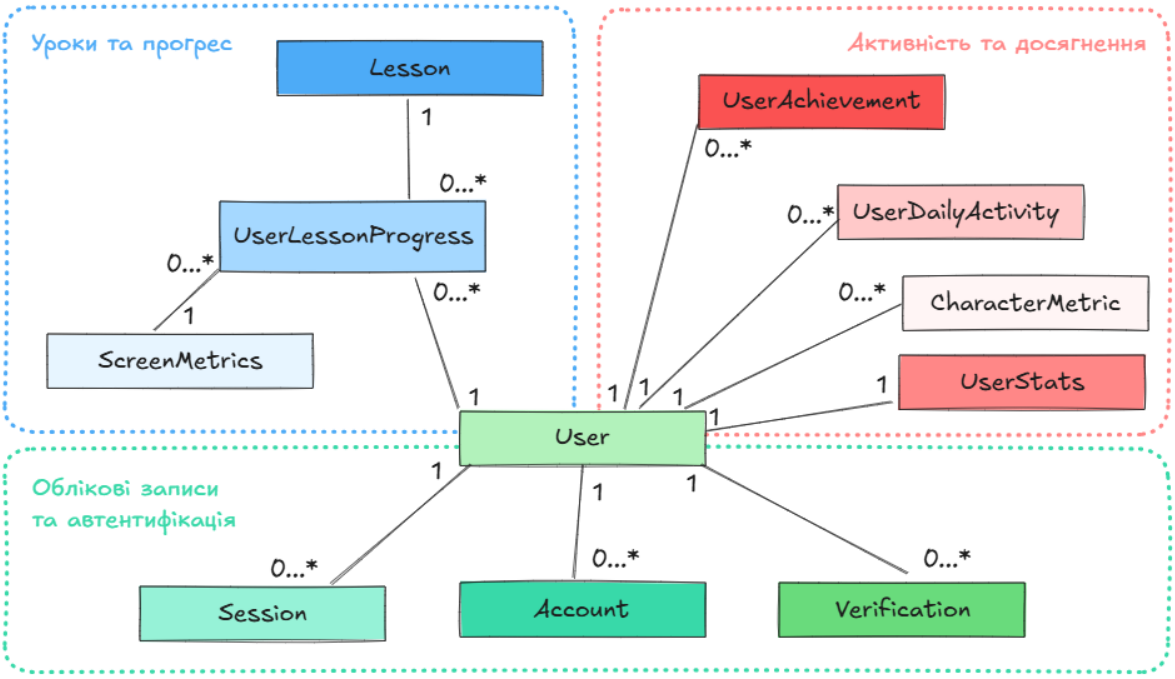

Рисунок 2.7 – Діаграма БД

З наведеної вище діаграми БД можна побачити такі відносини між сутностями:

– Кожен користувач може мати багато активних або завершених сесій, що створює відношення «один до багатьох» між сутностями User та Session.

– Кожен користувач може прив'язувати багато зовнішніх акаунтів, що формує відношення «один до багатьох» між сутностями User та Account.

– Один користувач може мати один або більше записів для верифікації, що встановлює відношення «один до багатьох» між сутностями User та Verification.

– Кожен користувач може мати записи прогресу для багатьох уроків, що утворює відношення «один до багатьох» між сутностями User та UserLessonProgress.

– Один урок може бути пройдений багатьма користувачами, кожен з яких матиме свій запис прогресу, що також створює відношення «один до багатьох» між сутностями Lesson та UserLessonProgress.

– Прогрес користувача в уроці зберігається через метрики для кожного окремого екрану. Тому між UserLessonProgress і ScreenMetrics існує зв’язок «один до багатьох».

– Кожен користувач має лише один агрегований запис статистики, що створює відношення «один до одного» між сутностями User та UserStats.

– Кожен користувач може получити багато досягнень, що встановлює відношення «один до багатьох» між сутностями User та UserAchievement.

– Користувачі можуть мати багато записів про щоденну активність (по одному на кожен день), що створює відношення «один до багатьох» між сутностями User та UserDailyActivity.

– Кожен користувач має метрики для багатьох окремих символів, що утворює відношення «один до багатьох» між сутностями User та CharacterMetric.

2.5 Висновок до другого розділу

У другому розділі було виконано проектування веб-застосунку для навчання «сліпому» друку з ігровими елементами. Було описано та обґрунтовано вибір технологічного стеку проекту, до якого входять: React, TypeScript, Express, MongoDB, Prisma, tRPC та інші сучасні інструменти.

Було розглянуто архітектурну модель веб-застосунку, де основну роль відіграє трирівнева архітектура. Також для організації кодової бази було обрано структуру монорепозиторію, яка дозволяє об’єднати всі частини проекту в єдиному середовищі розробки, забезпечивши спільне використання типів і конфігурацій.

Окремий підрозділ було присвячено моделюванню варіантів використання, де було описано акторів системи та їхню взаємодію з веб-застосунком. Також було детально описано структуру БД. Було спроектовано вміст кожної колекції, її атрибути та призначення, що стане основою для зберігання прогресу користувача, метрик, досягнень та інших даних.

РОЗДІЛ 3. РОЗРОБКА ВЕБ-ЗАСТОСУНКУ ДЛЯ НАВЧАННЯ «СЛІПОМУ» ДРУКУ

3.1 Початок розробки та структура файлів веб-застосунку

Розробку веб-застосунку для навчання «сліпому» друку з ігровими елементами було почато з ініціалізації монорепозиторію на базі Turborepo. Після генерації початкової структури застосунку на Turborepo, були видалені всі непотрібні додатки та спільні пакети, крім конфігураційних.

Далі відповідно до раніше описаної структури монорепозиторію було створено дві основні директорії:

– «apps» – для клієнтської та серверної частин застосунку;

– «packages» – для спільних пакетів, які використовуються як клієнтом, так і сервером;

На рисунку 3.1 зображено файлову структуру веб-застосунку для навчання «сліпому» друку.

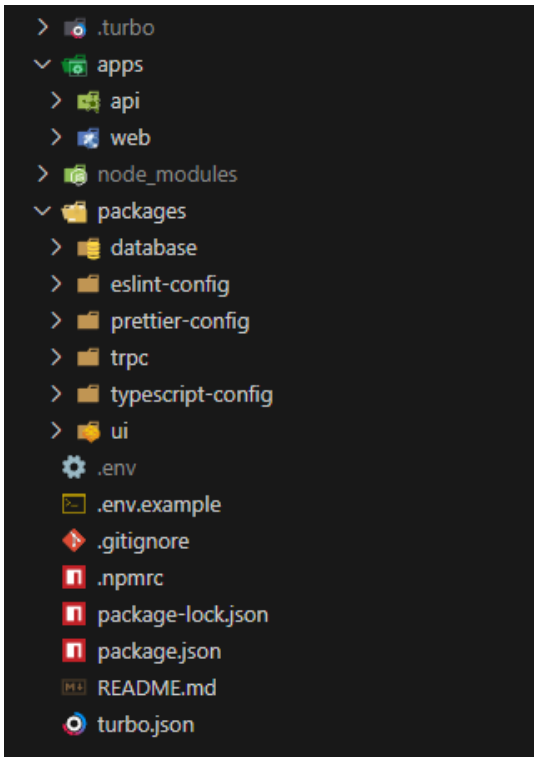

Рисунок 3.1 – Файлова структура веб-застосунку

У таблиці 3.1 наведено опис основних додатків і пакетів, що входять до складу веб-застосунку.

Таблиця 3.1 – Опис основних застосунків та пакетів

Назва дирикторії	**Тип**	**Призначення**
apps/web	Додаток	Клієнтська частина застосунку, яка реалізована на React та Vite.
apps/api	Додаток	Серверна частина застосунку, побудована на Node.js з Express.
packages/ui	Пакет	Набір спільних простих UI-компонентів, створених з використанням Shadcn UI і Tailwind CSS.
packages/database	Пакет	Містить схему бази даних, та клієнт Prisma ORM.
packages/trpc	Пакет	Містить tRPC-роутери, типи, схеми валідації (Zod) та допоміжні сервіси для бізнес-логіки.
packages/tsconfig	Пакет	Типова конфігурація TypeScript (спільна для всіх частин проєкту)
packages/eslint-config	Пакет	Конфігурація ESLint для уніфікації стилю коду в межах усього репозиторію.
packages/prettier-config	Пакет	Загальний конфіг prettier для утримання коду в одному стилі

Після створення директорій для кожного застосунку та пакету було налаштовано файли «packages.json». Для кожного пакету було вказано унікальну назву, наприклад, «@repo/ui» для пакету з UI-компонентами.

Також, для того щоб всі модулі мали доступ до спільних пакетів застосунку та працювали разом, було налаштовано файл «package.json» у самому корені проекту та вказані робочі простори, а саме папки «apps» та «packages».

Завдяки створенню початкової структури та налаштуванню монорепозиторію, тепер можна переходити до розробки серверної та клієнтської частин застосунку.

3.2 Розробка серверної частини

Розробку застосунку було вирішено почати з серверної частини, щоб описати основну бізнес логіку, а після вже переходити до клієнтської частини.

Для початку було ініціалізовано сервер з використанням Express, встановлено необхідні залежності та додано конфігураційний файл авторизації.

Вхідна точка сервера знаходиться у файлі «apps/api/src/index.ts». Цей файл налаштовує Express-сервер, підключає проміжне програмне забезпечення для обробки CORS-запитів та JSON-даних, а також додає систему авторизації та tRPC. У лістингу 3.1 наведено код файлу «index.ts».

Лістинг 3.1 – Код файлу «index.ts»

```
import { createContext } from "@repo/trpc/context";
import { appRouter } from "@repo/trpc/routers";
import * as trpcExpress from "@trpc/server/adapters/express";
import { toNodeHandler } from "better-auth/node";
import cors from "cors";
import express from "express";
import { auth } from "./lib/auth";
import { env } from "./lib/config";
const app = express();
app.use(
  cors({
    origin: env.FRONTEND_URL,
    credentials: true,
  }),
);
app.all("/api/auth/*", toNodeHandler(auth));
app.use(express.json());
app.use(
  "/trpc",
  trpcExpress.createExpressMiddleware({
    router: appRouter,
    createContext,
  }),
);
app.listen(env.PORT, () => {
  console.log(`API ready at http://localhost:${env.PORT}`);
});
```

Далі у файлі «apps/api/src/lib/auth.ts» було описано налаштування для авторизації користувачів через сторонні сервіси, зокрема GitHub та Google,

використовуючи бібліотеку better-auth. Ця бібліотека спрощує процес інтеграції OAuth-провайдерів та керування сесіями. У лістингу 3.2 наведено код файлу «auth.ts».

Лістинг 3.2 – Код файлу конфігурації авторизації «auth.ts»

```
import { prisma } from "@repo/database/prisma";
import { betterAuth } from "better-auth";
import { prismaAdapter } from "better-auth/adapters/prisma";
import { admin } from "better-auth/plugins";

import { env } from "./config";

export const auth = betterAuth({
  secret: env.BETTER_AUTH_SECRET,
  database: prismaAdapter(prisma, {
    provider: "mongodb",
  }),
  socialProviders: {
    google: {
      clientId: env.GOOGLE_CLIENT_ID,
      clientSecret: env.GOOGLE_CLIENT_SECRET,
    },
    github: {
      clientId: env.GITHUB_CLIENT_ID,
      clientSecret: env.GITHUB_CLIENT_SECRET,
    },
  },
  trustedOrigins: [env.FRONTEND_URL],
  plugins: [admin()],
});
```

Оскільки обрана бібліотека авторизації, better-auth підтримує Prisma, було згенеровано початкову схему БД, що мала основні таблиці для роботи з автентифікацією: User, Account, Verification та Session. Цю згенеровану схему було перенесено в пакет «@repo/database», який відповідає за опис та доступ до бази даних.

Після додавання схеми БД для авторизації, було додано нові таблиці, які зберігають уроки, метрики та різну статистику користувачів. Також таблицю User було змінено, щоб додати нові поля та відношення з створеними раніше таблицями.

З пакета «@repo/database» експортується Prisma-клієнт, який дозволяє іншим пакетам та застосункам легко взаємодіяти з БД, а також всі згенеровані Prisma-типи для зменшення помилок при роботі з даними.

Серверна частина веб-застосунку була розроблена як API, що відповідає за обробку основної бізнес-логіки. Вона забезпечує збереження та отримання даних з БД, авторизацію користувачів, збереження метрик, рівнів, досвіду, досягнень, а також формування статистики активності користувачів.

Для створення цієї бізнес-логіки та забезпечення типізованої взаємодії між клієнтом та сервером, було використано tRPC. У пакеті «@repo/trpc» створено роутери, що ізолюють основні функції застосунку.

Було реалізовано два основних роутера:

– Роутер для уроків (lessonRouter) – відповідає за отримання даних про уроки та їх екрани.

– Роутер для прогресу та статистики (userProgressRouter) – відповідає за збереження метрик, отримання різної статистики та досягнень.

Для підтримки чистоти коду, спрощення роутерів та уникнення дублювання, складна бізнес-логіка та обчислення були винесені в окремі допоміжні сервіси. Ці сервіси інкапсулюють спеціальні функції, дозволяючи tRPC роутерам зосереджуватися на маршрутизації запитів та валідації вхідних даних, тоді як основна обробка відбувається в цих сервісах.

До основних сервісів належать:

– AchievementService – сервіс, який відповідає за перевірку та нарахування досягнень користувачам. Варто зазначити, що самі унікальні досягнення та умови їх отримання зберігаються на сервері у вигляді константи, а не в БД. Такий підхід був обраний, оскільки будь-які зміни в ігровій логіці, наприклад, зміна умови отримання досягнення не потребують складних міграцій БД та спрощують розробку застосунку.

– DailyActivityService – сервіс, який керує щоденною активністю користувача та оновленням щоденної активності. Він оновлює записи та

підтримує актуальність поточної і найдовшої серій користувача. Також перетворює збережені дані в теплову карту активності.

– UserProgressService – основний сервіс для збереження та оновлення прогресу користувача в уроках та екранах. Він обробляє логіку нарахування досвіду за проходження екранів та завершення уроків, враховуючи, чи це перше проходження, чи покращення попередніх результатів. Цей сервіс взаємодіє з іншими сервісами для оновлення загальної статистики та досягнень.

– UserStatsService – сервіс, який відповідає за об’єднання та оновлення загальної статистики користувача, включаючи: загальний досвід, рівень, кількість завершених уроків, найвищі показники швидкості та точності для різних рівнів складності.

Далі для забезпечення безпеки API та обмеження доступу до певних роутів tRPC використовується механізм «контексту» та спеціальна захищена процедура – «protectedProcedure». Функція «createContext» відповідає за формування контексту для кожного запиту. Вона включає в себе Prisma-клієнт для роботи з БД, а також інформацію про сесію користувача, яку отримують за допомогою бібліотеки better-auth. Це дозволяє в процесі обробки запиту на сервер мати доступ до необхідних ресурсів і перевіряти чи користувач авторизований (див. лістинг 3.3).

Лістинг 3.3 – Код файлу з створенням tRPC контексту

```
import { prisma } from "@repo/database/prisma";
import type { CreateExpressContextOptions } from
"@trpc/server/adapters/express";
import { fromNodeHeaders } from "better-auth/node";
import { auth } from "../../../apps/api/src/lib/auth";

export type Context = {
  prisma: typeof prisma;
  session: Awaited<ReturnType<typeof auth.api.getSession>>;
};

export const createContext = async ({ req }:
CreateExpressContextOptions): Promise<Context> => {
  const session = await auth.api.getSession({
    headers: fromNodeHeaders(req.headers),
  });
```

```
  return {
    prisma,
    session,
  };
};
```

Після створення функції «createContext» та відповідного типу «Context», цей тип використовується для ініціалізації tRPC. Ініціалізація виконується з передачею контексту, що забезпечує типізацію та доступ до Prisma-клієнта й сесії користувача у всіх процедурах і роутерах.

Це налаштування дозволяє створювати різні види процедур для роботи з API, зокрема:

– «publicProcedure» – для публічних, відкритих запитів, які не потребують аутентифікації;

– «protectedProcedure» – для захищених процедур, які перевіряють наявність валідної сесії користувача і при відсутності сесії повертають помилку.

У застосунку «protectedProcedure» використовується для запитів, коли потрібно отримати дані про користувача, його метрики, досягнення або активність. У свою чергу «publicProcedure» призначена для відкритих запитів, які доступні без авторизації, наприклад, для отримання списку уроків чи інформації про екрани. У лістингу 3.4 наведено код реалізації ініціалізації tRPC із застосуванням типу «Context» та створюються «publicProcedure» та «protectedProcedure».

Лістинг 3.4 – Код файлу з ініціалізацією tRPC

```
import { initTRPC, TRPCError } from "@trpc/server";
import superjson from "superjson";
import { ZodError } from "zod";
import { Context } from "./context";
import { AppRouter } from "./routers";

const t = initTRPC.context<Context>().create({
  transformer: superjson,
  errorFormatter({ shape, error }) {
    return {
      ...shape,
```

```
      data: {
        ...shape.data,
        zodError: error.cause instanceof ZodError ?
error.cause.flatten() : null,
      },
    };
  },
});

export const router = t.router;
export const publicProcedure = t.procedure;

export const protectedProcedure = t.procedure.use(async ({ ctx,
next }) => {
  if (!ctx.session || !ctx.session.user) {
    throw new TRPCError({ code: "UNAUTHORIZED" });
  }
  return next({
    ctx: {
      ...ctx,
      session: { ...ctx.session, userId: ctx.session.user.id },
    },
  });
});

export type RouterInputs = inferRouterInputs<AppRouter>;
export type RouterOutputs = inferRouterOutputs<AppRouter>;
```

Серверна частина веб-застосунку для навчання «сліпому» друку була успішно реалізована. Далі можна переходити до розробки клієнтської частини застосунку, яка буде взаємодіяти з цим API для реалізації функціоналу користувацького інтерфейсу.

3.3 Розробка клієнтської частини

Розробка клієнтської частини веб-застосунку розпочалася з ініціалізації проекту на базі React та Vite. Були встановлені всі необхідні залежності, а також додані компоненти Shadcn UI до пакету «@repo/ui». Це дозволило швидко створювати інтерфейс, використовуючи готові та стилізовані елементи, такі як: «alert», «avatar», «badge», «button», «card», «carousel», «dialog», «hover-card» та «tabs». Крім того, було реалізовано базову функціональність для

підтримки світлої та темної тем інтерфейсу, що підвищує зручність використання застосунку.

Для коректної роботи авторизації на клієнтській стороні було підключено та налаштовано бібліотеку better-auth (див. лістинг 3.5). Це дозволило легко інтегрувати механізми входу через сторонні сервіси та приєднати клієнтську частину до серверної.

Лістинг 3.5 – Код налаштування better-auth на клієнті

```
import { createAuthClient } from 'better-auth/react';
import { env } from './config';

export const authClient = createAuthClient({
  baseURL: env.VITE_API_URL
});

export const { signIn, signUp, useSession, signOut } = authClient;
```

Також було налаштовано клієнт tRPC, який забезпечує типізовану взаємодію з серверними роутерами, визначеними у пакеті «@repo/trpc». Це гарантує, що дані, які надсилаються між клієнтом і сервером, відповідають визначеним типам.

Оскільки основна застосунку для навчання «сліпому» друку базується на взаємодії з клавіатурою, клієнтська частина повинна миттєво реагувати та реєструвати кожне натискання клавіш користувачем. Для цього було створено спеціальний хук «useKeyboardHandler», який «ловить» події клавіатури та записує їхній стан у Zustand-стор (сховище стану) «useKeyboardStore». Це дозволяє всьому застосунку мати доступ до інформації про натиснуті клавіші в реальному часі. Навколо цієї можливості було побудовано ряд інших хуків та сторів, які обробляють натискання, обчислюють метрики, зберігають прогрес та виконують інші функції.

У таблиці 3.2 наведено опис основних Zustand-сторів та React-хуків, які використовуються для керування станом та логікою клієнтської частини.

Таблиця 3.2 – Опис основних Zustand-сторів та React-хуків

Назва стору/хука	Призначення
useKeyboardStore	Відстежує стан клавіатури та зберігає інформацію про натиснуті клавіші.
useKeyboardHandler	Обробляє події клавіатури, оновлює стан натиснутих клавіш та активує гарячі клавіші.
useTypingMetricsStore	Збирає та обчислює метрики друку в реальному часі (WPM, точність, помилки) для екранів та уроків.
useTypingStore	Керує цільовим текстом для друку та поточним введеним текстом користувача.
useCurrentLessonStore	Відстежує поточний активний урок та екран, на якому зупинився користувач.
useThemeStore	Керує темою застосунку (темна або світла) та зберігає її.
useApplyTheme	Динамічно застосовує обрану тему до кореневого HTML-елементу документа.
useLessonsScreensHandler	Надає основну логіку для керування екранами уроків, включаючи завантаження даних, переходи між екранами та збереження прогресу.
useTypingHandler	Керує введенням тексту користувачем, обробляє натискання клавіш, виправлення помилок та оновлює пов'язані метрики.

Після цього було створено перелік різних специфічних саме для веб-застосунку для навчання «сліпому» друку компонентів.

Одним із ключових елементів став компонент віртуальної клавіатури, який відіграє важливу роль під час проходження уроків. Цей компонент відображає клавіатуру на екрані, підсвічуючи клавіші, які потрібно натиснути, та реагує на реальні натискання, що дозволяє користувачу краще орієнтуватися під час тренування.

На сторінці зі списком уроків відображаються спеціальні картки, кожна з яких містить назву уроку, рівень складності та індикатор прогресу. Щоб забезпечити логічну послідовність навчання, доступ до уроків обмежується: нові уроки відкриваються лише після проходження попередніх.

Окрім основних елементів, було створено також низку допоміжних компонентів.

Наступним кроком було розпочато розробку сторінок застосунку. Налаштування маршрутизації було виконано у файлі «main.tsx» за допомогою React Router (див. лістинг 3.6).

Лістинг 3.6 – Код файлу «main.tsx» з налаштуванням маршрутизації застосунку

```
import { QueryClientProvider } from '@tanstack/react-query';
import { StrictMode } from 'react';
import { createRoot } from 'react-dom/client';
import { BrowserRouter, Navigate, Route, Routes } from 'react-
router';

import MainLayout from './layouts/MainLayout.tsx';
import { Achivments, Lesson, LessonResults, Lessons, Profile }
from './pages;
import { queryClient } from './utils/trpc.ts';

createRoot(document.getElementById('root')!).render(
  <StrictMode>
    <QueryClientProvider client={queryClient}>
      <BrowserRouter>
        <Routes>
          <Route element={<MainLayout />}>
            <Route path="/" element={<Navigate to="/lessons"
replace />} />
            <Route path="/lessons" element={<Lessons />} />
            <Route path="/lesson/:lessonId" element={<Lesson />}
/>
            <Route path="/lesson/:lessonId/results"
element={<LessonResults />} />
            <Route path="/profile" element={<Profile />} />
            <Route path="/profile/achievements"
element={<Achivments />} />
            <Route path="*" element={<div>404 Not Found</div>} />
          </Route>
        </Routes>
      </BrowserRouter>
    </QueryClientProvider>
  </StrictMode>
);
```

У таблиці 3.3 наведено короткий опис сторінок веб-застосунку.

Таблиця 3.3 – Опис сторінок веб-застосунку

Назва сторінки	Призначення
Головний макет «MainLayout.tsx»	Визначає загальну структуру сторінок
Сторінка уроків «Lessons.tsx»	Відображає список доступних уроків, дозволяє фільтрувати за складністю та показує прогрес користувача.
Сторінка конкретного уроку «Lesson.tsx», «LessonScreen.tsx»	Центральний елемент навчання, відображає поточний екран уроку, віртуальну клавіатуру та метрики друку.
Сторінка результатів уроку «LessonResults.tsx»	Відображає детальну статистику продуктивності користувача після завершення уроку.
Сторінка профілю користувача «Profile.tsx»	Показує загальну статистику користувача, рівень, досвід, теплову карту активності та статистику символів.
Сторінка досягнень «Achivments.tsx»	Відображає всі розблоковані та ще не отримані досягнення користувача.

Головний макет застосунку – компонент визначає загальну структуру всіх сторінок застосунку. Він включає верхню панель (шапку) з перемикачем тем, заголовком та кнопкою для входу/виходу з системи. «MainLayout.tsx» також глобально використовує хук «useKeyboardHandler» для обробки гарячих клавіш та «useApplyTheme» для динамічного застосування обраної користувачем теми.

Після входу на сайт користувач потрапляє на сторінку уроків, яка є початковою точкою взаємодії з веб-застосунком (див. лістинг Б.1). Тут відображається список доступних уроків, що можна фільтрувати за рівнем складності. Кожен урок представлений у вигляді картки з назвою, складністю та індикатором прогресу. Незареєстровані користувачі можуть проходити уроки без облікового запису, отримуючи базову статистику після завершення, тоді як зареєстровані – бачать вже пройдені екрани й можуть продовжити з останнього місця (див. рисунок 3.2).

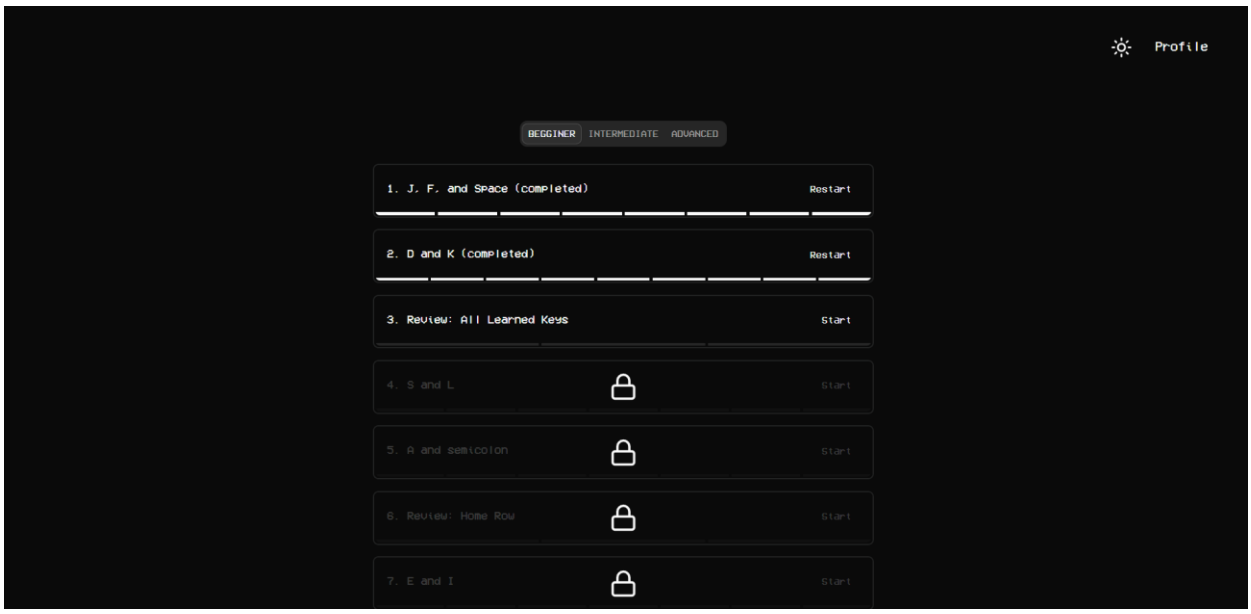

Рисунок 3.2 – Сторінка уроків

При виборі уроку користувач переходить на сторінку навчання, де послідовно проходить навчальні екрани з візуальною клавіатурою та метриками друку в реальному часі (див. лістинг Б.2). Екрани завантажуються динамічно в межах однієї сторінки, а їх порядок і логіка перемикання керуються хуком «useLessonsScreensHandler». На рисунку 3.3 зображено сторінку конкретного уроку.

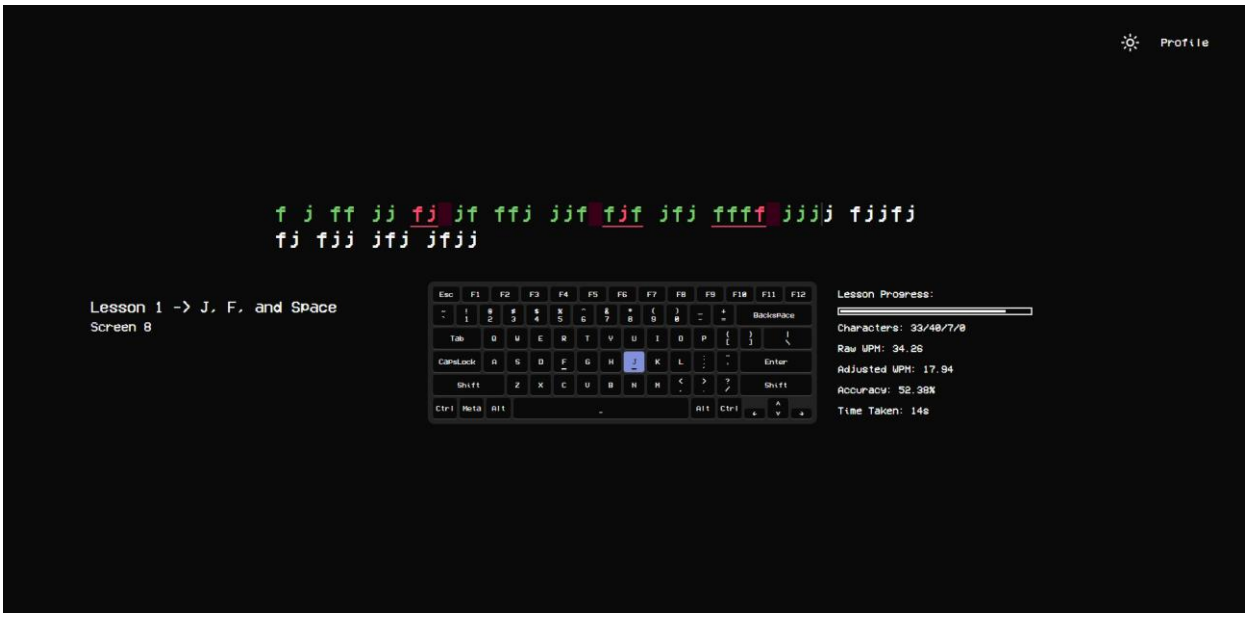

Рисунок 3.3 – Сторінка конкретного уроку

Після завершення всіх екранів користувача автоматично перенаправляє на сторінку результатів (див. лістинг Б.3). Тут відображаються загальні метрики і детальна статистика по кожному екрану, а також можливість порівняти результат з попередніми спробами. На рисунку 3.4 зображено сторінку результатів уроку.

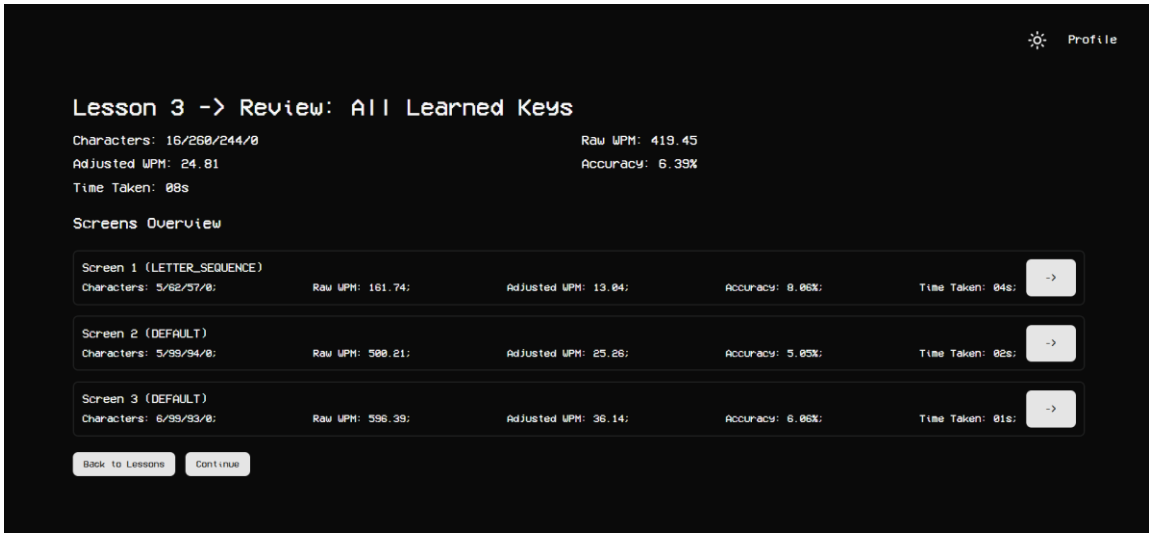

Рисунок 3.4 – Сторінка результатів уроку

Також з сторінки результатів користувачі можуть перейти на наступний урок або вернутись на сторінку зі всіма заняттями. Зареєстровані користувачі мають доступ до сторінки профілю (див. лістинг Б.4). На рисунку 3.5 зображено сторінку профілю.

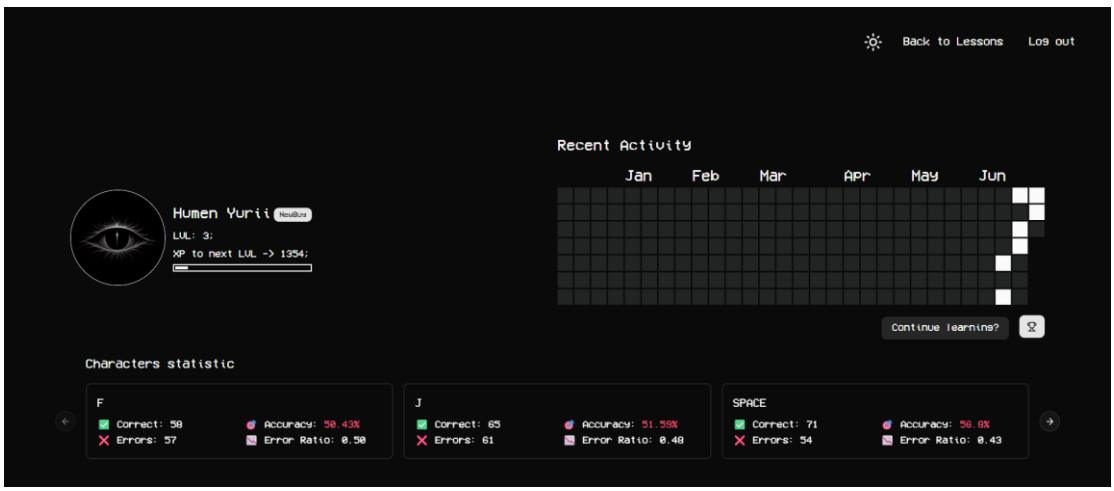

Рисунок 3.5 – Сторінка профілю

На цій сторінці відображається: аватар, ім’я, поточний рівень, досвід, теплову карту активності, а також статистику за кожною буквою – точність, кількість помилок, коефіцієнт помилок Також можна перейти на сторінку з досягненнями або продовжити навчання перейшовши на останній пройдений урок.

На сторінці досягнень користувач бачить повний перелік усіх досягнень – як здобутих, так і заблокованих (див. лістинг Б.5). Розблоковані досягнення мають візуальні відзнаки, що створює ефект колекціонування та мотивації до подальшого проходження. На рисунку 3.6 зображено сторінку досягнень.

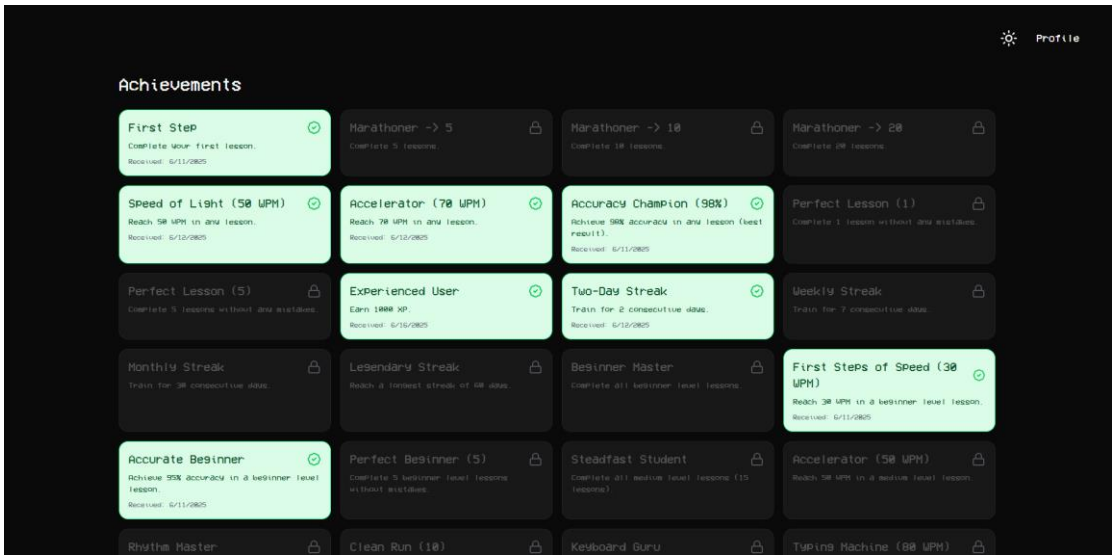

Рисунок 3.6 – Сторінка досягнень

Інтерфейс веб-застосунку розроблений з акцентом на користувацький досвід, забезпечуючи інтуїтивну навігацію та візуально привабливе представлення інформації.

3.4 Висновок до третього розділу

У третьому розділі розглянуто розробку веб-застосунку для навчання «сліпому» друку з ігровими елементами. Описано реалізацію архітектури на основі React, Node.js, MongoDB та tRPC, включно з налаштуванням бази даних,

реалізацією маршрутів, обробкою бізнес-логіки та інтеграцією авторизації через сторонні сервіси. Особливу увагу приділено обробці користувацьких метрик, організації стану на клієнті за допомогою Zustand, а також забезпеченню безпеки даних і модульності коду.

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Вплив шуму та вібрацій на концентрацію користувача під час навчання «сліпому» друку

Під час навчання сліпому друку за допомогою веб-застосунку важливо забезпечити комфортні умови роботи за комп’ютером, зокрема дотримання норм рівня шуму та вібрацій. Навіть фонові звуки, які можуть видавати системний блок, вентилятори або зовнішні пристрої, впливають на здатність користувача зосередитися. Шум заважає сприйняттю інформації, знижує швидкість реакції, викликає втому, дратівливість і знижує мотивацію до навчання. У середовищі, де потрібна підвищена концентрація, наприклад, під час друку без погляду на клавіатуру, надмірний шум може бути критичним чинником, що погіршує результати навчання.

Нормативно допустимий рівень шуму для користувачів персональних комп’ютерів становить 50 дБА. Ця межа регламентується державними санітарними нормами ДСН 3.3.6.037-99 [43]. Для операторів набору тексту або працівників, які перебувають у залах обробки даних, допустиме значення становить до 65 дБА. Для приміщень, де розміщено шумні агрегати або серверне обладнання, допускається до 75 дБА, але в освітньому або офісному середовищі рівень шуму не повинен перевищувати 50 дБА, згідно з Наказом МОЗ № 463 від 22.02.2019. Щоб дотриматися цих показників, використовують спеціальні звукопоглинаючі матеріали, килимки або техніку з пониженим рівнем шуму [44].

Окрім шуму, варто звертати увагу і на вібрації, які хоч і рідко відчутні, але теж мають вплив на самопочуття та ефективність користувача. Джерелами вібрацій можуть бути системний блок, монітор, клавіатура або миша. Якщо техніка стоїть на твердих поверхнях без амортизації, вона може створювати

вібрації, що передаються на руки, очі або спину користувача. Це може викликати втому м’язів, головний біль, біль у шиї та навіть зниження точності друку.

Допустимі значення локальної вібрації для робочого місця регламентуються нормами ДСН 3.3.6.039-99 [45]. Згідно з ними, максимальне допустиме віброприскорення для 8-годинної зміни становить 5 м/с^2, а бажане значення – 2,5 м/с^2. Щоб уникнути впливу вібрації, рекомендується розміщувати системний блок на м’якій основі, використовувати антивібраційні килимки або ковролін, а також обирати якісну клавіатуру та мишу, які не створюють додаткових навантажень на руки.

Забезпечення тиші та зменшення вібрацій на робочому місці є необхідною умовою для успішного проходження навчального процесу. Всі ці заходи дозволяють підвищити комфорт, зменшити втомлюваність та покращити результати при навчанні сліпому друку.

4.2 Психологія безпеки праці в цифровому середовищі: вплив гейміфікації на увагу, мотивацію та стан користувача

Ігрові елементи дуже добре допомагають у навчанні, роблячи його цікавішим і мотивуючим. Завдяки гейміфікації нудні завдання перетворюються на захоплюючі виклики, що змушує людину хотіти досягати результатів. Дослідження показують, що впровадження ігрових елементів може збільшити успішність учнів на 34,75%. Учні, які пройшли гейміфікацію на основі завдань, покращили свою успішність на 89,45% порівняно з тими, хто лише виконував традиційні завдання. Також впровадження ігрових елементів у навчальні застосунки допомагає краще мислити, бути уважнішим і краще запам'ятовувати [46].

Однак, якщо довго повторювати одні й ті самі вправи, навіть з ігровими елементами, це може знову стати нудним, і людина втратить інтерес. Коли завдання стає занадто звичним, увага може розсіюватися. Постійні змагання та

таблиці лідерів, хоч і мотивують, можуть викликати стрес або зневіру у тих, хто завжди внизу списку. Це може призвести до тривоги або навіть «синдрому самозванця». Крім того, якщо занадто зосереджуватися на ігрових нагородах, а не на самому навчанні, це може відволікати від головної мети і навіть викликати залежність від технологій або вигорання.

Щоб уникнути цих проблем, навчальний веб-застосунок повинен бути добре продуманий. Для боротьби з нудьгою він має пропонувати різні види тренувань, підлаштовувати слова під слабкі місця користувача. Щоб зменшити стрес від змагань, застосунок повинен показувати, як користувач покращує свої власні результати, а не тільки порівнювати його з іншими. Можна дозволити приховувати таблиці лідерів або зосереджуватися на особистих цілях. Щоб уникнути вигорання та залежності, програма може нагадувати про перерви, радити, скільки часу краще тренуватися і показувати, скільки часу користувач провів у програмі. Важливо підкреслювати цінність реальних навичок, а не тільки ігрових балів.

Закон України «Про охорону праці» говорить, що роботодавець повинен створювати безпечні та здорові умови праці, а також враховувати психологічний стан працівника (Стаття 4). Це означає, що психологічні фактори важливі у будь-якій роботі. Роботодавець також повинен розповідати про можливі шкідливі фактори, а для небезпечних робіт може знадобитися психологічна перевірка (Стаття 5) [47].

Гігієнічна класифікація праці (Наказ Міністерства охорони здоров'я України № 528 від 27.12.2001) допомагає оцінювати умови праці. Вона визначає «напруженість праці» як навантаження на нервову систему, органи чуття та емоції. Сюди входять розумові, сенсорні та емоційні навантаження, а також одноманітність роботи та режим праці (Пункт 3.7.2 Наказу № 528). Цей документ дозволяє оцінити умови праці, включаючи психологічні аспекти, за різними класами: від оптимальних до небезпечних [48].

Хоча в українських законах немає прямої згадки про «стрес від гейміфікації», існуючі закони про охорону праці все одно охоплюють

психологічну безпеку, говорячи про «напруженість праці» та «психологічний стан». Це означає, що загальні правила безпеки застосовуються і до онлайн-навчання. Навіть якщо розробники програми не є прямими роботодавцями, вони несуть відповідальність за те, щоб враховувати ці психологічні фактори. Розробляючи програму так, щоб зменшити розумове навантаження, одноманітність та стрес від змагань, вони дотримуються духу цих законів.

4.3 Вимоги до освітлення робочого місця користувача ПК для зменшення навантаження на зір

Правильне освітлення робочого місця користувача ПК має надзвичайно важливе значення для мінімізації навантаження на зір та запобігання втоми очей. Згідно з українськими нормами, для приміщень, де відбувається робота з ВДТ, передбачено забезпечення природного та штучного освітлення відповідно до ДСанПіН 3.3.2.007-98 [49]. Природне світло має надходити через вікна, що розташовані переважно на північ або північний схід, і його слід регулювати за допомогою жалюзі або завіс, щоб уникнути прямих сонячних променів і відблисків на екрані. Робоче місце має бути розташоване таким чином, щоб світло падало збоку – зазвичай зліва для правшів, з правого боку для лівшів.

Штучне освітлення забезпечується за допомогою системи рівномірної освітленості загального призначення, часто у вигляді суцільних або переривчастих ліній світильників, що розташовані паралельно лінії зору з боку, як правило, ліворуч. Рекомендовані джерела освітлення – люмінесцентні лампи для загального освітлення та додаткові настільні світильники для робочого місця, розміщені так, щоб не утворювати сліпучих відблисків на екрані.

Рівень освітленості на робочій поверхні в зоні розташування документів або клавіатури має становити від 300 до 500 люкс, що відповідає ДБН В.2.5-28:2018 [50] та підтверджується ДСанПіН 3.3.2.007-98. При цьому рівень освітленості самого екрану не рекомендується перевищувати 300 люкс, щоб уникнути контрасту, що спричиняє напругу зору. Крім цього, коефіцієнт

мерехтіння світла має бути не вище ніж 5 %, що забезпечує комфорт для очей та запобігає головним болям і сухості очей. Нерівномірність освітленості не повинна перевищувати співвідношення 3:1, аби уникнути зорового дискомфорту та зменшити навантаження на центральну нервову систему.

Формування оптимального освітлення на робочому місці можна підтримувати програмними засобами веб-застосунку. Наприклад, він може нагадувати користувачу підтримувати комфортну відстань до монітора (40-75 см), регулювати яскравість екрану відповідно до освітлення приміщення й пропонувати піти на перерву або зробити вправи для очей, такі як «правило 20–20–20» – кожні 20 хвилин дивитися вдалину на 20 секунд. Такі повідомлення допомагають знизити напругу на зір, підвищити продуктивність та довго зберігати здоров’я очей під час навчання «сліпому» друку [51].

4.4 Висновок до четвертого розділу

У четвертому розділі кваліфікаційної роботи було досліджено аспекти безпеки життєдіяльності та охорони праці, пов’язані з використанням веб-застосунку для навчання «сліпому» друку. Було проаналізовано психологічний вплив гейміфікації, визначено можливі ризики емоційного вигорання, перевантаження та стресу внаслідок ігрових елементів, а також наведено шляхи їх мінімізації.

Розглянуто вплив шуму, вібрацій і освітлення на фізичний стан і концентрацію користувача. Було висвітлено вимоги до організації робочого місця, зокрема, щодо рівня шуму та освітленості відповідно до чинного українського законодавства.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи освітнього рівня «Бакалавр» було розроблено веб-застосунок для навчання «сліпому» друку з використанням ігрових елементів.

У першому розділі кваліфікаційної роботи:

– Розглянуто особливості навчання методу друку наосліп та основні труднощі, з якими стикаються користувачі.

– Проаналізовано вплив ігрових механік на мотивацію у навчальному процесі.

– Проведено огляд існуючих платформ та визначено ключові функціональні можливості.

– Сформульовано функціональні та нефункціональні вимоги до власного застосунку.

У другому розділі кваліфікаційної роботи:

– Обґрунтовано вибір технологій, описано архітектуру клієнт-серверної системи та структуру монорепозиторію.

– Спроектовано базу даних та визначено логіку взаємодії між основними модулями застосунку.

– Розроблено діаграми, які відображають загальну структуру та сценарії використання.

У третьому розділі кваліфікаційної роботи:

– Реалізовано серверну та клієнтську частини.

– Реалізовано облік метрик друку в реальному часі, накопичення досвіду, досягнення, рівні та теплову карту активності.

У розділі «Безпека життєдіяльності, основи охорони праці» було розглянуто вплив роботи з веб-застосунком на фізичний і психоемоційний стан користувача. Описано основні ризики та способи їх мінімізації відповідно до чинних норм.

ПЕРЕЛІК ДЖЕРЕЛ

1 Діана Сологуб. Сліпий метод друку – корисна навичка для будь-якого копірайтера. Content Writer [Електронний ресурс]. Режим доступу до ресурсу: https://contentwriter.com.ua/slipyi-metod-druku/ (дата звернення: 10.06.2025).

2 Wen Shan. How To Save 21 Days Per Year By Typing Fast. LifeHack [Електронний ресурс]. Режим доступу до ресурсу: https://www.lifehack.org/560218/how-save-21-days-per-year-typing-fast (дата звернення: 10.06.2025).

3 Abdul Aziz Mohamed Mohamed Ali El Deen. Students' boredom in English language classes: Voices from Saudi Arabia. Frontiers in Psychology [Електронний ресурс]. Режим доступу до ресурсу: https://www.frontiersin.org/journals/psychology/articles/10.3389/fpsyg.2023.1108372/full (дата звернення: 10.06.2025).

4 Sara Heegaard. What is Gamification in E-Learning? Articulate [Електронний ресурс]. Режим доступу до ресурсу: https://www.articulate.com/blog/what-is-gamification-in-e-learning/ (дата звернення: 10.06.2025).

5 The Influence of Duolingo's Gamified Design Elements: A study on maintaining user engagement [Електронний ресурс]. Режим доступу до ресурсу: https://hj.diva-portal.org/smash/record.jsf?pid=diva2%3A1866808&dswid=8403 (дата звернення: 10.06.2025).

6 Psico-smart Editorial Team. What are the psychological impacts of gamification on adult learning, and how can games be designed to boost retention rates based on recent research findings? Blogs.psico Smart [Електронний ресурс]. Режим доступу до ресурсу: https://blogs.psico-smart.com/blog-what-are-the-psychological-impacts-of-gamification-on-adult-learning-a-188798 (дата звернення: 10.06.2025).

7 Staff Writer. TypingClub vs Traditional Typing Courses: Which is More Effective? *Ask.com* [Електронний ресурс]. Режим доступу до ресурсу: https://www.ask.com/news/typingclub-vs-traditional-typing-courses-effective (дата звернення: 10.06.2025).

8 About. Monkeytype [Електронний ресурс]. Режим доступу до ресурсу: https://monkeytype.com/about (дата звернення: 10.06.2025).

9 Keybr review – what is this site and is it worth practicing on? [Електронний ресурс]. Режим доступу до ресурсу: https://typingdonewell.com/blog/keybr-review-what-is-this-site-and-is-it-worth-practicing-on/ (дата звернення: 10.06.2025).

10 React Component-based Architecture: Build Modern UIs Easily. eSparkInfo [Електронний ресурс]. Режим доступу до ресурсу: https://www.esparkinfo.com/software-development/technologies/reactjs/component-based-architecture (дата звернення: 10.06.2025).

11 Vikas Singh. React Components Explained: A 2025 Guide for Developers. DEV Community [Електронний ресурс]. Режим доступу до ресурсу: https://dev.to/brilworks/react-components-explained-a-2025-guide-for-developers-4dhe (дата звернення: 10.06.2025).

12 Paul Bratslavsky. Top 6 Benefits of Implementing TypeScript. *Strapi* [Електронний ресурс]. Режим доступу до ресурсу: https://strapi.io/blog/benefits-of-typescript (дата звернення: 10.06.2025).

13 Iftekhar Ali. Zustand - A lightweight state-management solution for React applications. Younode [Електронний ресурс]. Режим доступу до ресурсу: https://younode.com/articles/zustand (дата звернення: 10.06.2025).

14 Alejandro Ramírez. Zustand: A Lightweight State Management Library for React. Sonatafy Technology [Електронний ресурс]. Режим доступу до ресурсу: https://sonatafy.com/zustand-a-lightweight-state-management-library-for-react/ (дата звернення: 10.06.2025).

15 Hassan Djirdeh. A Primer on Tailwind CSS: Pros, Cons and Real-World Use Cases. Telerik blogs [Електронний ресурс]. Режим доступу до ресурсу:

https://www.telerik.com/blogs/primer-tailwind-css-pros-cons-real-world-use-cases (дата звернення: 10.06.2025).

16 Zack Proser. What is the difference between Radix and shadcn-ui? WorkOS [Електронний ресурс]. Режим доступу до ресурсу: https://workos.com/blog/what-is-the-difference-between-radix-and-shadcn-ui (дата звернення: 10.06.2025).

17 Nipuni Arunodi. Top 7 React Animation Libraries. Syncfusion Blogs [Електронний ресурс]. Режим доступу до ресурсу: https://www.syncfusion.com/blogs/post/top-react-animation-libraries (дата звернення: 10.06.2025).

18 Kesar Bhimani. React Spring vs. Framer Motion: A Detailed Guide. DhiWise [Електронний ресурс]. Оновлено: 10.05.2025. Режим доступу до ресурсу: https://www.dhiwise.com/post/react-spring-vs-framer-motion-a-detailed-guide-to-react (дата звернення: 10.06.2025).

19 GoodSoft. Advantages of using React Query. GoodSoft [Електронний ресурс]. Режим доступу до ресурсу: https://goodsoft.pl/advantages-of-react-query/ (дата звернення: 10.06.2025).

20 Abhay Singh Kathayat. React Query (TanStack Query): Efficient Data Fetching and State Management for React. DEV Community [Електронний ресурс]. Режим доступу до ресурсу: https://dev.to/abhay_yt_52a8e72b213be229/react-query-tanstack-query-efficient-data-fetching-and-state-management-for-react-1c18 (дата звернення: 10.06.2025).

21 Features. Vite [Електронний ресурс]. Режим доступу до ресурсу: https://vite.dev/guide/features (дата звернення: 10.06.2025).

22 Le Do Nghiem. Integrate TypeScript into a React Project with Vite. DEV Community [Електронний ресурс]. Режим доступу до ресурсу: https://dev.to/nghiemledo/integrate-typescript-into-a-react-project-with-vite-12ah (дата звернення: 10.06.2025).

23 Hrithik Das K. Why Use Node.js as Your Backend: A Comprehensive Guide for 2025. Webandcrafts [Електронний ресурс]. Режим доступу до ресурсу: https://webandcrafts.com/blog/node-js-backend (дата звернення: 10.06.2025).

24 Nusrat Sarmin. Express.js: A Battle-tested Backend Framework for Node.js. StaticMania [Електронний ресурс]. Режим доступу до ресурсу: https://staticmania.com/blog/details-of-express-js (дата звернення: 10.06.2025).

25 Understanding Code Management with Monorepos and Turborepo. GeeksforGeeks [Електронний ресурс]. Режим доступу до ресурсу: https://www.geeksforgeeks.org/mern/understanding-code-management-with-monorepos-and-turborepo/ (дата звернення: 10.06.2025).

26 Jagadhiswaran Devaraj. tRPC: A Simple and Type-Safe Way to Build APIs in TypeScript. Peerlist [Електронний ресурс]. Режим доступу до ресурсу: https://peerlist.io/jagss/articles/trpc-a-simple-and-typesafe-way-to-build-apis-in-typescript (дата звернення: 10.06.2025).

27 Advantages Of MongoDB. MongoDB [Електронний ресурс]. Режим доступу до ресурсу: https://www.mongodb.com/resources/compare/advantages-of-mongodb (дата звернення: 10.06.2025).

28 Jesse Hall - codeSTACKr. How Prisma Transforms MongoDB Development (And Why You Need It). DEV Community [Електронний ресурс]. Режим доступу до ресурсу: https://dev.to/mongodb/how-prisma-transforms-mongodb-development-and-why-you-need-it-34bk (дата звернення: 10.06.2025).

29 Stanley Ulili. A Complete Guide to Zod. Better Stack Community [Електронний ресурс]. Режим доступу до ресурсу: https://betterstack.com/community/guides/scaling-nodejs/zod-explained/ (дата звернення: 10.06.2025).

30 Comparison. Better Auth [Електронний ресурс]. Режим доступу до ресурсу: https://www.better-auth.com/docs/comparison (дата звернення: 10.06.2025).

31 Five Tops Benefits of Using a 3-Tier Architecture. *Insightsoftware* [Електронний ресурс]. Режим доступу до ресурсу:

https://insightsoftware.com/blog/5-benefits-of-a-3-tier-architecture/ (дата звернення: 10.06.2025).

32 Greg Foster. Monorepo Pros and Cons. Graphite [Електронний ресурс]. Режим доступу до ресурсу: https://graphite.dev/guides/monorepo-pros-and-cons (дата звернення: 10.06.2025).

33 Kristian Dupont. Type Safety in TypeScript with tRPC for Enhanced Code Reliability. Newline [Електронний ресурс]. Режим доступу до ресурсу: https://www.newline.co/@kristiandupont/draft-5-type-safety-in-typescript-with-trpc-for-enhanced-code-reliability--b3dafc7e (дата звернення: 10.06.2025).

34 Six Principles For Success with Use Case Adoption. Ivar Jacobson International [Електронний ресурс]. Режим доступу до ресурсу: https://www.ivarjacobson.com/publications/blog/6-principles-success-use-case-adoption (дата звернення: 10.06.2025).

35 Харченко, О. Г., Галай, І. О., Бондарчук, І. О., & Яцишин, В. В. (2010). Проектування архітектури WEB-застосування на основі моделі якості проектування. Інженерія Програмного Забезпечення, 4(4), 26 (дата звернення: 10.06.2025).

36 Боднарчук, І., Харченко, О., Хоміцький, Б., & Шимчук, Г. (2019). Проектування архітектури програмних систем в проектах з гнучкими методами управління. Матеріали XXI Наукової Конференції Тернопільського Національного Технічного Університету Імені Івана Пулюя, 46–48 (дата звернення: 10.06.2025).

37 Harchenko, A., Bodnarchuk, I., & Yatcyshyn, V. (2012). The modeling and optimization of software engineering processes. Proceedings of International Conference on Modern Problem of Radio Engineering, Telecommunications and Computer Science, 326 (дата звернення: 10.06.2025).

38 Kharchenko, A., Halay, I., Zagorodna, N., & Bodnarchuk, I. (2015). Trade-off optimal decision of the problem of software system architecture choice. 2015 Xth International Scientific and Technical Conference" Computer Sciences and Information Technologies"(CSIT), 198–205 (дата звернення: 10.06.2025).

39 Ihor, B., Oleksii, D., Aleksandr, K., Nataliia, K., Oleksandr, M., & Volodymyr, P. (2019). Multicriteria Choice of Software Architecture Using Dynamic Correction of Quality Attributes. Advances in Computer Science for Engineering and Education II. ICCSEEA 2019. Advances in Intelligent Systems and Computing, 938, 419–427 (дата звернення: 10.06.2025).

40 Kharchenko, A., Raichev, I., Bodnarchuk, I., & Matsiuk, O. (2021). The Survey of Global Software Design Processes. 2021 IEEE 8th International Conference on Problems of Infocommunications, Science and Technology (PIC S&T), 291–294 (дата звернення: 10.06.2025).

41 Orobchuk, B., Buniak, O., Sysak, I., Babiuk, S., Bodnarchuk, I., & Koval, V. (2024). Development of Software for the Implementation of Automated Reserve Input Modes Operation. 2nd International Workshop on Computer Information Technologies in Industry, 4, 12–14 (дата звернення: 10.06.2025).

42 Bodnarchuk, I., Skorenkyy, Y., Kramar, T., Duda, O., & Nykytyuk, V. (2022). Use of Analytical Hierarchy Process in Scenarios Design for a Digital Museum with XR components. ITTAP, 414–425 (дата звернення: 10.06.2025).

43 Санітарні норми виробничого шуму, ультразвуку та інфразвуку (ДСН 3.3.6.037-99) [Електронний ресурс]. Режим доступу до ресурсу: https://ips.ligazakon.net/document/MOZ641 (дата звернення: 10.06.2025).

44 Міністерство охорони здоров’я. Наказ від 22.02.2019 № 463 Про затвердження Державних санітарних норм допустимих рівнів шуму в приміщеннях житлових та громадських будинків і на території житлової забудови [Електронний ресурс]. Режим доступу до ресурсу: https://online.budstandart.com/ua/catalog/doc-page.html?id_doc=82742 (дата звернення: 10.06.2025).

45 Державні санітарні норми виробничої загальної та локальної вібрації (ДСН 3.3.6.039-99) від 01.12.1999 [Електронний ресурс]. Режим доступу до ресурсу: https://ips.ligazakon.net/document/MOZ639 (дата звернення: 10.06.2025).

46 Побризгаєва Валентина. Гейміфікація навчання у початкових класах в умовах дистанційного навчання [Електронний ресурс]. Режим доступу до

ресурсу: https://www.researchgate.net/profile/Oleksii-Nalyvaiko/publication/380183877_GEJMIFIKACIA_NAVCANNA_U_POCATKOVIH_KLASAH_V_UMOVAH_DISTANCIJNOGO_NAVCANNA/links/6630765206ea3d0b7419a9ea/GEJMIFIKACIA-NAVCANNA-U-POCATKOVIH-KLASAH-V-UMOVAH-DISTANCIJNOGO-NAVCANNA.pdf (дата звернення: 19.06.2025).

47 ЗАКОН УКРАЇНИ Про охорону праці. Верховна Рада Україн [Електронний ресурс]. Режим доступу до ресурсу: http://www.nbuv.gov.ua/sites/default/files/msd/zupop_vdkd.pdf (дата звернення: 10.06.2025).

48 Міністерство охорони здоров'я України. Наказ № 528 від 27.12.2001 Про затвердження Гігієнічної класифікації праці за показниками шкідливості та небезпечності факторів виробничого середовища, важкості та напруженості трудового процесу [Електронний ресурс]. Оновлено: 27.12.2001. Режим доступу до ресурсу: https://zakononline.com.ua/documents/show/44478___44478 (дата звернення: 10.06.2025).

49 Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПІН 3.3.2.007-98 [Електронний ресурс]. Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/v0007282-98#Text (дата звернення: 10.06.2025).

50 ДБН В.2.5-28:2018 Природне і штучне освітлення [Електронний ресурс]. Режим доступу до ресурсу: https://online.budstandart.com/ua/catalog/doc-page.html?id_doc=79885 (дата звернення: 10.06.2025).

51 Рекомендації щодо збереження зору під час роботи з комп’ютером [Електронний ресурс]. Режим доступу до ресурсу: https://www.amelsmart.com/rekomendacziyi-shhodo-zberezhennya-zoru-pid-chas (дата звернення: 10.06.2025)

ДОДАТКИ

Додаток А

Таблиця сутностей колекції UserStats

Назва атрибуту	Тип даних	Опис
id	String	Унікальний ідентифікатор
userId	String	Ідентифікатор користувача
totalExperience	Int	Загальна кількість отриманого досвіду
currentLevel	Int	Поточний рівень користувача
xpToNextLevel	Int	Кількість досвіду до наступного рівня
totalLessonsCompleted	Int	Загальна кількість завершених уроків
totalPerfectLessons	Int	Загальна кількість уроків, завершених без помилок
highestOverallWPM	Float	Найвища загальна швидкість друку (слів за хвилину)
highestOverallAccuracy	Float	Найвища загальна точність друку у відсотках
beginnerLessonsCompleted	Int	Кількість завершених уроків початкового рівня
beginnerPerfectLessons	Int	Кількість уроків початкового рівня, завершених без помилок
highestBeginnerWPM	Float	Найвища швидкість друку для початкового рівня
highestBeginnerAccuracy	Float	Найвища точність друку для початкового рівня
mediumLessonsCompleted	Int	Кількість завершених уроків середнього рівня
mediumPerfectLessons	Int	Кількість уроків середнього рівня, завершених без помилок
highestMediumWPM	Float	Найвища швидкість друку для середнього рівня
highestMediumAccuracy	Float	Найвища точність друку для середнього рівня
advancedLessonsCompleted	Int	Кількість завершених уроків високого рівня
advancedPerfectLessons	Int	Кількість уроків високого рівня, завершених без помилок

highestAdvancedWPM	Float	Найвища швидкість друку для високого рівня
highestAdvancedAccuracy	Float	Найвища точність друку для високого рівня
currentStreak	Int	Поточна серія щоденної активності (кількість днів поспіль)
longestStreak	Int	Найдовша серія щоденної активності (кількість днів поспіль)
createdAt	DateTime	Дата та час створення запису
updatedAt	DateTime	Дата та час останнього оновлення запису

Додаток Б

Лістинги сторінок веб-застосунку для навчання «сліпому» друку

Лістинг Б.1 – Код сторінки уроків

```
import { Tabs, TabsList, TabsTrigger } from
'@repo/ui/components/ui/tabs';
import { useQuery } from '@tanstack/react-query';

import LessonCard from '@/components/LessonCard';
import { trpc } from '@/utils/trpc';

const Lessons = () => {
  const { data: lessons, isLoading: isLoadingLessons } =
useQuery({
    ...trpc.lesson.getAll.queryOptions(),
    staleTime: 1000 * 60 * 10
  });

  const { data: allUserProgress, isLoading: isLoadingProgress } =
useQuery(
    trpc.userProgress.getAllUserProgress.queryOptions()
  );

  const { data: lastCompletedLessonProgress, isLoading:
isLoadingLastLesson } = useQuery(
    trpc.userProgress.getLastLessonByOrder.queryOptions()
  );

  if (isLoadingLessons || isLoadingProgress ||
isLoadingLastLesson) {
    return <div>Loading lessons...</div>;
  }

  const lastCompletedLessonOrder =
lastCompletedLessonProgress?.lesson?.order ?? 0;

  return (
    lessons && (
      <div className="no-scrollbar h-full max-h-screen overflow-
auto py-40">
        <Tabs className="mb-6 flex w-full items-center justify-
center" defaultValue="BEGGINER">
          <TabsList>
            <TabsTrigger value="BEGGINER">BEGGINER</TabsTrigger>
            <TabsTrigger
value="INTERMEDIATE">INTERMEDIATE</TabsTrigger>
            <TabsTrigger value="ADVANCED">ADVANCED</TabsTrigger>
          </TabsList>
        </Tabs>
        <ul className="flex flex-col gap-4">
```

```
          {lessons.map((lesson, index) => {
            const lessonProgress =
allUserProgress?.find((progress) => progress.lessonId ===
lesson.id);

            const isLessonAvailable = lessonProgress?.isCompleted
|| lesson.order <= lastCompletedLessonOrder + 1;

            return (
              <LessonCard
                key={lesson.id}
                lesson={lesson}
                index={index}
                lessonProgress={lessonProgress}
                isLessonAvailable={isLessonAvailable}
              />
            );
          })}
        </ul>
      </div>
    )
  );
};

export default Lessons;
```

Лістинг Б.2 – Код конкретної сторінки уроків

```
import { Button } from '@repo/ui/components/ui/button';
import { AnimatePresence, LayoutGroup, motion } from 'framer-
motion';
import { useEffect, useState } from 'react';

import Keyboard from '@/components/Keyboard';
import KeyIntroduction from '@/components/KeyIntroduction';
import RealTimeMetrics from '@/components/RealTimeMetrics';
import SequenceOfLetters from '@/components/SequenceOfLetters';
import TypingText from '@/components/TypingText';
import { useLessonsScreensHandler } from
'@/hooks/useLessonsScreensHandler';
import { useTypingMetricsStore } from
'@/stores/useTypingMetricsStore';
import { useTypingStore } from '@/stores/useTypingStore';
import { LearningMode, Screen } from '@/utils/types';

const Lesson = () => {
  const { lesson, currentScreen, handleScreenComplete } =
useLessonsScreensHandler();

  return (
    <motion.div layout className="relative w-full">
      {lesson && currentScreen && (
```

```
        <LessonScreen currentScreen={currentScreen}
onScreenComplete={handleScreenComplete} lesson={lesson} />
      )}
    </motion.div>
  );
};

export default Lesson;

const screenVariants = {
  hidden: { clipPath: 'polygon(0 0, 100% 0, 100% 0, 0 0)' },
  visible: { clipPath: 'polygon(0 0, 100% 0, 100% 100%, 0% 100%)'
},
  exit: { clipPath: 'polygon(0 100%, 100% 100%, 100% 100%, 0%
100%)' }
};

export function LessonScreen({
  currentScreen,
  onScreenComplete,
  lesson
}: {
  currentScreen: Screen;
  onScreenComplete: () => void;
  lesson: Lesson;
}) {
  const isEndOfInputText = useTypingStore((s) =>
s.isEndOfInputText);
  const screenStartTime = useTypingMetricsStore((s) =>
s.screenStartTime);
  const updateCalculatedScreenMetrics = useTypingMetricsStore((s)
=> s.updateCalculatedScreenMetrics);
  const [isButtonVisible, setIsButtonVisible] = useState(false);
  const [isExiting, setIsExiting] = useState(false);

  const onContinueButtonClick = () => {
    setIsExiting(true);
  };

  useEffect(() => {
    if (isEndOfInputText) {
      setIsButtonVisible(true);
    }
  }, [isEndOfInputText]);

  useEffect(() => {
    if (!screenStartTime) return;

    const interval = setInterval(() => {
      updateCalculatedScreenMetrics();
    }, 1000);

    if (isEndOfInputText) {
```

```
      clearInterval(interval);
    }

    return () => clearInterval(interval);
  }, [isEndOfInputText, screenStartTime,
updateCalculatedScreenMetrics]);

  return (
    <LayoutGroup>
      <motion.div className="relative flex w-full flex-col items-
center justify-center gap-6">
        <AnimatePresence mode="wait">
          {!isExiting && (
            <motion.div
              key={currentScreen.order}
              layout="position"
              layoutId="lesson-screen"
              initial="hidden"
              animate="visible"
              exit="exit"
              variants={screenVariants}
              transition={{ duration: 0.4 }}
              onAnimationComplete={(definition) => {
                if (definition === 'exit') {
                  onScreenComplete();
                  setIsExiting(false);
                  setIsButtonVisible(false);
                }
              }}
              className="flex w-full items-center justify-center
py-1"
            >
              {(() => {
                switch (currentScreen.type) {
                  case LearningMode.KEY_INTRODUCTION:
                    return <KeyIntroduction />;
                  case LearningMode.DEFAULT:
                    return <TypingText />;
                  case LearningMode.LETTER_SEQUENCE:
                    return <SequenceOfLetters />;
                  default:
                    return null;
                }
              })()}
            </motion.div>
          )}
        </AnimatePresence>

        <motion.div layoutId="keyboard" className="relative flex
items-center justify-center">
          <Keyboard size="full" />
          <div className="absolute left-full ml-8 w-3/6">
```

```
            <RealTimeMetrics
screensInLesson={lesson.screens.length}
currentScreenOrder={currentScreen.order} />
          </div>
          <div className="absolute top-6 -left-full ml-20 w-full">
            <h1 className="text-2xl">
              Lesson {lesson.order} -&gt; {lesson.title}
            </h1>
            <p className="text-xl">Screen
{currentScreen.order}</p>
          </div>
        </motion.div>

        <AnimatePresence mode="wait">
          {isButtonVisible && !isExiting && (
            <motion.div
              layoutId="continue-button"
              initial="hidden"
              animate="visible"
              exit="exit"
              variants={screenVariants}
              transition={{ duration: 0.4 }}
            >
              <Button
onClick={onContinueButtonClick}>Continue</Button>
            </motion.div>
          )}
        </AnimatePresence>
      </motion.div>
    </LayoutGroup>
  );
}
```

Лістинг Б.3 – Код сторінки результатів уроку

```
import { Button } from '@repo/ui/components/ui/button';
import { useQuery } from '@tanstack/react-query';
import { useCallback } from 'react';
import { useNavigate, useParams } from 'react-router';

import { useCurrentLessonStore } from
'@/stores/useCurrentLessonStore';
import { useTypingMetricsStore } from
'@/stores/useTypingMetricsStore';
import { formatTime } from '@/utils/metrics';
import { trpc } from '@/utils/trpc';

const LessonMetricsScreen = () => {
  const { lessonId } = useParams<{ lessonId: string }>();
  const { setCurrentLessonId, setCurrentScreenOrder } =
useCurrentLessonStore();
  const navigate = useNavigate();
```

```
  const currentLessonMetrics = useTypingMetricsStore((state) =>
state.currentLessonMetrics);

  const { data: lesson } = useQuery({
    ...trpc.lesson.getById.queryOptions(lessonId || ''),
    staleTime: 1000 * 60 * 10
  });

  const { refetch: fetchNextLesson } = useQuery({
    ...trpc.lesson.getNextLessonById.queryOptions(lessonId || ''),
    staleTime: 1000 * 60 * 10,
    enabled: false
  });

  const handleNextLesson = useCallback(async () => {
    if (!lessonId) return;

    const { data: nextLesson } = await fetchNextLesson();

    if (nextLesson) {
      setCurrentLessonId(nextLesson.id);
      navigate(`/lesson/${nextLesson.id}`);
    } else {
      navigate('/lessons');
    }
  }, [fetchNextLesson, lessonId, navigate, setCurrentLessonId]);

  const handleRestartScreen = (order: number) => {
    if (!currentLessonMetrics) return;

    setCurrentScreenOrder(order);
    navigate(`/lesson/${currentLessonMetrics.lessonId}`);
  };

  if (!currentLessonMetrics) {
    return <div>No metrics available.</div>;
  }

  return (
    <div className="w-full">
      <div className="mb-6">
        <h2 className="mb-4 text-4xl">
          Lesson {lesson?.order} -&gt; {lesson?.title}
        </h2>
        <div className="grid grid-cols-2 gap-2 text-xl">
          <p>
            Characters:
{currentLessonMetrics.totalCorrectCharacters}/{currentLessonMetric
s.totalTypedCharacters}/

{currentLessonMetrics.totalErrors}/{currentLessonMetrics.totalBack
spaces}
```

```
          </p>
          <p>Raw WPM: {currentLessonMetrics.totalRawWPM}</p>
          <p>Adjusted WPM:
{currentLessonMetrics.totalAdjustedWPM}</p>
          <p>Accuracy: {currentLessonMetrics.totalAccuracy}%</p>
          <p>Time Taken:
{formatTime(currentLessonMetrics.totalTimeTaken)}</p>
        </div>
      </div>

      <div className="flex flex-col gap-4">
        <h3 className="mb-2 text-2xl">Screens Overview</h3>
        {currentLessonMetrics.screenMetrics.map((screen) => (
          <div
            key={screen.order}
            className="border-border min-h-[80px] overflow-hidden
rounded-md border-2 transition-all duration-200 hover:shadow-(--
key-shadow)"
          >
            <div className="flex items-center justify-between px-3
py-3">
              <div className="h-full flex-1">
                <h2 className="mb-1 text-lg">
                  Screen {screen.order} ({screen.type})
                </h2>
                <div className="flex justify-between gap-4">
                  <p>
                    Characters:
{screen.correctCharacters}/{screen.typedCharacters}/{screen.errors
}/{screen.backspaces};
                  </p>
                  <p>Raw WPM: {screen.rawWPM};</p>
                  <p>Adjusted WPM: {screen.adjustedWPM};</p>
                  <p>Accuracy: {screen.accuracy}%;</p>
                  <p>Time Taken:
{formatTime(screen.timeTaken)};</p>
                </div>
              </div>
              <div className="ml-3 flex flex-[0_0_5%] self-
stretch">
                <Button size={null} className="h-full w-full"
onClick={() => handleRestartScreen(screen.order)}>
                  -&gt;
                </Button>
              </div>
            </div>
          </div>
        ))}
      </div>
      <div className="flex gap-4">
        <Button className="mt-6" onClick={() =>
navigate('/lessons')}>
          Back to Lessons
```

```
        </Button>
        <Button className="mt-6" onClick={handleNextLesson}>
          Continue
        </Button>
      </div>
    </div>
  );
};

export default LessonMetricsScreen;
```

Лістинг Б.4 – Код сторінки профілю користувача

```
import { Avatar, AvatarFallback, AvatarImage } from
'@repo/ui/components/ui/avatar';
import { Badge } from '@repo/ui/components/ui/badge';
import { Button } from '@repo/ui/components/ui/button';
import {
  Carousel,
  CarouselContent,
  CarouselItem,
  CarouselNext,
  CarouselPrevious
} from '@repo/ui/components/ui/carousel';
import { useQuery } from '@tanstack/react-query';
import { Trophy } from 'lucide-react';
import { useEffect } from 'react';
import CalendarHeatmap from 'react-calendar-heatmap';
import { useNavigate } from 'react-router';

import ProgressBar from '@/components/ProgressBar';
import { authClient } from '@/lib/auth';
import { LEVEL_THRESHOLDS } from '@/lib/constansts';
import { useCurrentLessonStore } from
'@/stores/useCurrentLessonStore';
import { trpc } from '@/utils/trpc';

const Profile = () => {
  const { data: session } = authClient.useSession();
  const { data: heatmap } =
useQuery(trpc.userProgress.getUserActivityHeatmap.queryOptions());
  const { data: experienceData } =
useQuery(trpc.userProgress.getUserXpAndLevel.queryOptions());
  const { data: charactersData } =
useQuery(trpc.userProgress.getCharacterMetrics.queryOptions());
  const { refetch } = useQuery({
    ...trpc.userProgress.getLastLessonByOrder.queryOptions(),
    enabled: false
  });

  const navigate = useNavigate();
```

```
  const { setCurrentLessonId, setCurrentScreenOrder } =
useCurrentLessonStore();

  const startLastLesson = async () => {
    const { data: lastLesson } = await refetch(); // remake later,
make it redirect to first lesson forst screen

    if (!lastLesson) return;

    setCurrentLessonId(lastLesson.lessonId);
    setCurrentScreenOrder(lastLesson.currentScreenOrder + 1);
    navigate(`/lesson/${lastLesson.lessonId}`);
  };

  const currentLevel = experienceData?.currentLevel ?? 1;
  const totalExperience = experienceData?.totalExperience ?? 0;

  const prevLevelXp = LEVEL_THRESHOLDS[currentLevel - 1] ?? 0;

  const xpOnThisLevel = totalExperience - prevLevelXp; // remake
on server side, so we can use it directly

  useEffect(() => {
    if (!session) {
      navigate('/lessons');
    }
  }, [navigate, session]);

  return (
    <div className="min-w-[1200px] pt-20">
      <div className="flex items-center justify-between">
        <div>
          <div className="flex items-center gap-3">
            <Avatar className="border-foreground size-[150px]
border-1">
              <AvatarImage src={session?.user.image ?? undefined}
alt="User Avatar" />

<AvatarFallback>{session?.user.name.charAt(0)}</AvatarFallback>
            </Avatar>
            <div className="flex flex-col items-start gap-1">
              <div className="flex items-center gap-1">
                <h1 className="mb-1 text-2xl">{session?.user.name}
</h1>
                <Badge variant="default" color="">
                  NewBuy
                </Badge>
              </div>
              <p>LVL: {experienceData?.currentLevel};</p>
              <p>XP to next LVL -&gt;
{experienceData?.xpToNextLevel ?? 'error'};</p>
              <ProgressBar current={xpOnThisLevel}
max={experienceData?.xpToNextLevel} />
```

```
            </div>
          </div>
        </div>
        {heatmap && session && (
          <div className="flex w-[50%] flex-col">
            <h1 className="mb-4 text-2xl">Recent Activity</h1>
            <CalendarHeatmap
              endDate={new Date(Date.now() - 24 * 60 * 60 * 1000)}
              values={
                heatmap?.heatmapData
                  ?
Object.entries(heatmap.heatmapData).map(([date, value]) => ({
                      date,
                      count: value.lessons
                    }))
                  : []
              }
            />
            <div className="mt-4 self-end">
              <Button variant={'secondary'}
onClick={startLastLesson} className="mr-4 text-base">
                Continue learning?
              </Button>
              <Button onClick={() =>
navigate('/profile/achievements')} className="text-base">
                <Trophy />
              </Button>
            </div>
          </div>
        )}
      </div>
      <div className="p-6">
        <h1 className="mb-4 text-xl">Characters statistic</h1>
        <Carousel>
          <CarouselContent className="-ml-4">
            {charactersData &&
              charactersData.map((character) => (
                <CarouselItem key={character.character}
className="basis-1/3 pl-4">
                  <div className="rounded-md border-2 p-4
transition-all">
                    <h2 className="mb-2 text-lg uppercase">
                      {character.character === ' ' ? 'Space' :
character.character}
                    </h2>

                    <div className="grid grid-cols-2 gap-4 text-
base">
                      <div>
                        <p>
                          <span className="">Correct:</span>
{character.correctCount}
                        </p>
```

```
                        <p>
                           <span className="">Errors:</span>
{character.errorCount}
                        </p>
                      </div>

                      <div>
                        <p>
                           <span className="">Accuracy:</span>{'
'}
                          <span
                            className={
                              character.accuracyPercentage > 90
                                ? 'text-correct'
                                : character.accuracyPercentage >
75
                                  ? 'text-chart-3'
                                  : 'text-error'
                            }
                          >
                            {character.accuracyPercentage}%
                          </span>
                        </p>
                        <p>
                           <span>Error Ratio:</span>
{character.errorRatio.toFixed(2)}
                        </p>
                      </div>
                    </div>
                  </div>
                </CarouselItem>
              ))}
          </CarouselContent>
          <CarouselPrevious />
          <CarouselNext />
        </Carousel>
        <div className="flex gap-2"></div>
      </div>
    </div>
  );
};

export default Profile;
```

Лістинг Б.5 – Код сторінки досягнень

```
import { cn } from '@repo/ui/lib/utils';
import { useQuery } from '@tanstack/react-query';
import { BadgeCheck, Lock } from 'lucide-react';

import { trpc } from '@/utils/trpc';
```

```
const Achievements = () => {
  const { data: achievements, isLoading, error } =
useQuery(trpc.userProgress.getAchievements.queryOptions());

  if (isLoading) return <div>Downloading...</div>;
  if (error) return <div>Error loading achievements</div>;
  if (!achievements) return <div>No achievement data
available</div>;

  return (
    <div className="p-4 py-30">
      <h2 className="mb-6 text-3xl">Achievements</h2>
      <div className="grid grid-cols-1 gap-4 sm:grid-cols-2
md:grid-cols-3 lg:grid-cols-4">
        {achievements.map((ach) => (
          <div
            key={ach.id}
            className={cn(
              'rounded-2xl border p-4 shadow-sm transition-all
duration-300',
              ach.unlocked
                ? 'border-green-400 bg-green-100 text-green-900'
                : 'bg-muted text-muted-foreground opacity-50'
            )}
          >
            <div className="mb-2 flex items-center justify-
between">
              <div className="text-xl">{ach.name}</div>
              {ach.unlocked ? <BadgeCheck className="text-green-
500" /> : <Lock className="text-gray-400" />}
            </div>
            <p className="text-sm">{ach.description}</p>
            {ach.unlockedAt && (
              <p className="mt-2 text-xs text-gray-500">Received:
{new Date(ach.unlockedAt).toLocaleDateString()}</p>
            )}
          </div>
        ))}
      </div>
    </div>
  );
};

export default Achievements;
```