fragmentation of existing educational tools and improves the efficiency of communication and data management within educational institutions.

The proposed solution demonstrates the effectiveness of integrating various educational processes into a single digital platform, allowing users to manage academic activities in a structured and convenient way. This approach significantly simplifies interaction between students, teachers, and administrators while improving transparency and organization of learning processes.

The main outcomes of the work include:

- *analysis of existing educational platforms and identification of their limitations;*
- *formulation of functional requirements for a unified system;*
- *design and implementation of a full-stack web application using modern technologies;*
- *integration of authentication, role management, and educational workflows;*
- *testing and validation of system functionality and usability.*

The results of the study confirm that the developed system improves the efficiency of educational process management and can be further extended with additional features such as analytics, mobile applications, and external integrations in the future.

Keywords: *web application, information system, client-server architecture, database, PostgreSQL, Prisma, Node.js, TypeScript, NestJS, REST API, JWT, authentication, authorization, React, SPA, Redux, Vite, MUI, Figma, Docker, MinIO, MailHog, deployment, web development.*

Одержано редакцією 20.09.2024 р.
Прийнято до публікації 30.10.2024 р.

УДК 004.415.2:004.75

DOI 10.31651/2076-5886-2024-1-90-102

PACS 07.05.Tp, 89.20.Ff

ТКАЧЕНКО Олексій Олексійович
студент спеціальності «Інформаційні системи та технології» Черкаського національного університету імені Богдана Хмельницького

ДІДКОВСЬКИЙ Руслан Михайлович,
доктор технічних наук, доцент, доцент кафедри прикладної математики та інформатики Черкаського національного університету імені Богдана Хмельницького
e-mail: didkovskyirm@vu.cdu.edu.ua
ORCID 0000-0002-5166-7564

ПІСКУН Олександр Варфоломійович
кандидат технічних наук, доцент, завідувач кафедри прикладної математики та інформатики, Черкаський національний університет ім. Б. Хмельницького
e-mail: piskun@ukr.net
ORCID 0000-0001-5334-6337

РОЗРОБКА КЛІЄНТСЬКОЇ ЧАСТИНИ МУЛЬТИПЛЕЄРНОГО КЛАВІАТУРНОГО ТРЕНАЖЕРА З ВИКОРИСТАННЯМ REACT ТА WEBSOCKET-ТЕХНОЛОГІЙ

У роботі розглянуто підходи до проектування та розробки клієнтської частини мультиплеєрного веб-застосунку для тренування швидкості та точності клавіатурного

набору тексту. Проведено порівняльний аналіз наявних рішень у даній предметній галузі – TypingClub, Nitro Type та 10FastFingers – на підставі якого виявлено обмеження існуючих підходів і сформульовано вимоги до власного застосунку. Обґрунтовано вибір технологічного стеку: React.js із хуками та контекстом для побудови компонентної SPA-архітектури, TypeScript для статичної типізації, TailwindCSS для стилізації й підтримки тем оформлення, ActionCable на основі протоколу WebSocket (RFC 6455) для забезпечення низькозатримкового двостороннього зв'язку між учасниками гри в режимі реального часу, а також механізми авторизації на основі JWT (RFC 7519) та Google OAuth2. Описано архітектуру клієнтської частини застосунку: компонентну структуру, механізми маршрутизації (React Router), управління глобальним станом через React Context та обробку подій WebSocket-каналу. Детально розглянуто реалізацію центрального компонента ігрової кімнати, що підтримує два режими – онлайн-змагання та офлайн-практику, – таблиці лідерів із сортуванням і пошуком, а також мультиплеєрної сторінки з управлінням ігровими кімнатами. Порівняно застосовані технічні підходи з альтернативами. Встановлено, що реалізований застосунок поєднує можливості, відсутні в будь-якому з аналізованих аналогів.

Ключові слова: *клавіатурний тренажер, React.js, TypeScript, TailwindCSS, WebSocket, ActionCable, JWT, OAuth2, мультиплеєрний веб-застосунок, SPA, однобічний потік даних.*

Вступ

Зростання обсягів цифрової комунікації, поширення дистанційних форм роботи і навчання, а також збільшення частки текстоорієнтованих засобів взаємодії підвищують значущість навичок клавіатурного введення тексту. Ця компетенція є важливою не лише для ІТ-фахівців, а й для широкого кола користувачів: студентів, офісних працівників, журналістів, науковців та інших категорій. Дослідження в галузі ергономіки та методики навчання показують, що цілеспрямована практика із зворотним зв'язком є ефективнішою, ніж неструктуроване введення тексту.

Клавіатурні тренажери є перевіреним інструментом формування даної навички: вони надають зворотний зв'язок щодо швидкості та точності введення, дозволяють визначати слабкі місця і відслідковувати динаміку прогресу. Водночас більшість наявних рішень або орієнтовані виключно на індивідуальні тренування без соціальної складової, або пропонують мультиплеєрний режим без повноцінного аналітичного інструментарію і персоналізованого інтерфейсу.

З технічної точки зору мультиплеєрний тренажер реального часу є складнішою задачею, ніж індивідуальний. Потрібно забезпечити синхронну передачу стану між учасниками з мінімальною затримкою, коректно обробляти підключення та відключення гравців, підтримувати конкурентний доступ до ресурсів сервера і при цьому надавати відзивчивий інтерфейс. Такі вимоги формують специфічний технічний контекст, де вибір технологій для клієнтської частини безпосередньо впливає на якість кінцевого продукту.

Зазначені чинники визначають актуальність дослідження та практичну значущість розробки мультиплеєрного клавіатурного тренажера з повноцінною сучасною клієнтською частиною.

На сьогодні ринок освітніх веб-застосунків є одним із найактивніше зростаючих сегментів EdTech. Зростання попиту на дистанційне навчання та самовдосконалення в зручному темпі стимулює розробку нових інструментів, що поєднують навчальну ефективність із соціальною залученістю. Гейміфікація освітніх платформ – включення механік нагород, змагань, рейтингів і прогресивних досягнень – є доведеним інструментом підвищення мотивації та утримання користувачів. Саме тому поєднання гейміфікованого мультиплеєрного режиму з аналітичним інструментарієм у рамках єдиного сучасного веб-застосунку є актуальним науково-практичним завданням.

Обраний технологічний стек відповідає сучасному стану галузі фронтенд-розробки: React залишається однією з найширше застосовуваних бібліотек для

побудови SPA, TypeScript набув статусу стандарту де-факто для великих TypeScript-проектів, TailwindCSS стрімко набирає популярність як альтернатива CSS-in-JS рішенням, а WebSocket є промисловим стандартом для застосунків реального часу. Вибір зрілих, широко підтримуваних технологій з відкритим кодом забезпечує перспективу тривалої підтримки і розвитку системи.

Мета статті – розробити клієнтську частину мультиплеєрного клавіатурного тренажера з використанням технологічного стеку React.js, TypeScript, TailwindCSS та ActionCable/WebSocket, що забезпечує змагання в режимі реального часу, автентифікацію користувачів та відображення аналітики результатів.

Виклад основного матеріалу

1. Огляд існуючих рішень та аналіз літератури

Для визначення вимог до власного застосунку проведено аналіз трьох популярних веб-рішень у сфері тренування клавіатурного введення.

TypingClub [1] є одним із найбільш функціонально насичених тренажерів: він пропонує структуровану програму навчання з поетапним підвищенням складності, інтерактивні анімації та звуковий супровід. Перевага TypingClub – методологічна структурованість навчального контенту та зручний інтерфейс для початківців. Однак відсутність мультиплеєрного режиму є суттєвим обмеженням для користувачів, яким важлива змагальна складова, а також відсутня детальна аналітика за окремими ігровими сесіями.

Nitro Type [2] позиціонує себе як гейміфікований тренажер у форматі автомобільних перегонів із мультиплеєрним режимом. Змагальна механіка та ігрові елементи підвищують залученість і мотивацію. Разом із тим надмірний акцент на швидкості без рівного акценту на точності може негативно впливати на якість формованих навичок; детальна аналітика сесії не передбачена.

10FastFingers [3] – мінімалістичний, легкодоступний інструмент зі спрощеним інтерфейсом і підтримкою введення різними мовами; наявний режим змагань. Недоліком є відсутність структурованої навчальної програми, детальної аналітики прогресу та персоналізованого інтерфейсу.

Порівняльна характеристика аналізованих рішень подана у таблиці 1.

Аналіз показав: жодне з розглянутих рішень не поєднує повноцінний мультиплеєрний режим, детальну аналітику ігрових сесій у вигляді графіків, персоналізований інтерфейс зі зміною теми та сучасні механізми авторизації. Це підтверджує доцільність розробки власного рішення та формує чіткий орієнтир для проектування.

Таблиця 1

Порівняльна характеристика наявних клавіатурних тренажерів

Критерій	TypingClub	Nitro Type	10FastFingers
Мультиплеєр у реальному часі	Ні	Так	Частково
Структурована програма навчання	Так	Ні	Ні
Детальна аналітика сесії (графіки)	Ні	Ні	Ні
Персоналізація теми інтерфейсу	Ні	Частково	Ні
Таблиця лідерів із сортуванням	Ні	Так	Так
Авторизація через Google	Ні	Ні	Ні
Метрика «символи за секунду»	Ні	Ні	Так

З технічної точки зору, проблематика клієнтських застосунків реального часу широко розглядається у контексті вибору між технологіями: HTTP-опитування

(polling), Server-Sent Events та повнодуплексний WebSocket (RFC 6455 [14]). WebSocket забезпечує найнижчу затримку при двосторонньому обміні за рахунок єдиного постійного TCP-з'єднання. Питанням проектування React-застосунків, управління станом і обробки WebSocket-потоків приділяється значна увага у технічній документації та публікаціях, що підтверджує зрілість обраних технологій.

2. Постановка задачі та архітектура застосунку

На підставі виявлених недоліків аналогів сформульовано функціональні вимоги до розроблюваного застосунку:

1. Підтримка мультиплеєрного режиму з відображенням прогресу суперника в режимі реального часу.
2. Авторизація через власний обліковий запис (JWT) та через Google OAuth2.
3. Збереження та відображення аналітики ігрових сесій: швидкість у символах за секунду (sps) та точність (%).
4. Таблиця лідерів із можливістю пошуку за нікнеймом та двонапрямним сортуванням за ключовими метриками.
5. Перемикання між кількома кольоровими темами інтерфейсу.
6. Режим офлайн-практики без взаємодії з сервером (не зберігається у базі даних).

Клієнтська частина реалізована як односторінковий застосунок (SPA – Single Page Application) з компонентною архітектурою на базі React.js [4]. Підхід SPA дозволяє уникати повних перезавантажень сторінки при навігації, що критично для застосунків реального часу: підписки WebSocket і стан контексту зберігаються протягом усієї сесії без розривів.

Маршрутизація між сторінками здійснюється бібліотекою React Router [12]. Глобальний стан (дані авторизованого користувача, поточна тема) передається через React Context API [10], що усуває необхідність «прокидання» пропсів через численні рівні ієрархії.

Ключові маршрути застосунку:

– / – головна сторінка (Home): відображення останніх ігор, кнопки переходу до режимів;
– /gameRoom і /gameRoom/:room_id – ігрова кімната (офлайн і онлайн режими);
– /multiplayerRoom – перегляд і створення ігрових кімнат;
– /leaderboard – таблиця лідерів;
– /account – профіль користувача.

3. Технологічний стек клієнтської частини

3.1 React.js: компонентна архітектура та Virtual DOM

React.js [4] – бібліотека JavaScript для побудови користувацьких інтерфейсів, розроблена компанією Meta у 2013 році. Основна концепція – декомпозиція інтерфейсу на незалежні, повторно використовувані компоненти. Кожен компонент інкапсулює власну логіку стану та відображення, що сприяє модульності й полегшує тестування.

Ключовою особливістю React є механізм Virtual DOM – легковагової копії реального DOM-дерева. При зміні стану компонента React оновлює не реальний DOM безпосередньо, а спочатку вносить зміни у Virtual DOM, після чого алгоритм узгодження (reconciliation) визначає мінімальний набір операцій над реальним DOM. Це суттєво знижує навантаження при частих оновленнях інтерфейсу – властивість,

критична для ігрового тренажера, де стан (позиція каретки, прогрес гравців) змінюється при кожному натисканні клавіші.

Архітектура застосунку побудована виключно на функціональних компонентах із хуками:

- useState – управління локальним станом компонента (поточна позиція каретки, стан символів, швидкість тощо);
- useEffect – обробка побічних ефектів: HTTP-запити до сервера при завантаженні, підписки на WebSocket-канали, запуск/зупинка таймерів;
- useContext – доступ до глобального стану (UserContext, ThemeContext);
- useRef – зберігання посилань на DOM-елементи без ініціювання повторного рендерингу (наприклад, для фокусування поля введення).

Однобічний потік даних (one-way data flow) [5] забезпечує передбачуваність стану: дані передаються від батьківських компонентів до дочірніх виключно через пропси, а зворотний зв'язок здійснюється через callback-функції. Такий підхід спрощує відлагодження та унеможливлює несанкціоновану мутацію стану.

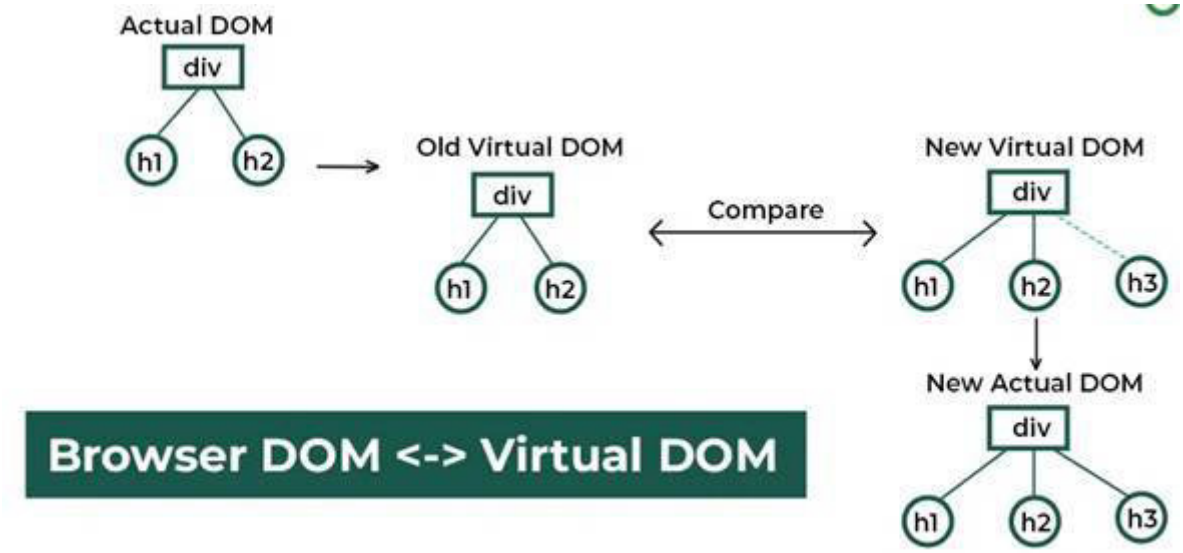

Рис. 1. Механізм Virtual DOM

3.2 TypeScript: статична типізація у динамічному середовищі

TypeScript [8] – надмножина JavaScript, розроблена компанією Microsoft, що додає систему статичних типів із виведенням типів і підтримку сучасних стандартів ECMAScript. Код TypeScript компілюється у стандартний JavaScript, сумісний з будь-яким сучасним браузером чи середовищем Node.js.

Статична типізація TypeScript дозволяє виявляти категорію помилок безпосередньо на етапі компіляції, до виконання програми. У контексті мультиплеєрного тренажера це особливо важливо в двох аспектах:

1. *Структури WebSocket-повідомлень*: визначення TypeScript-інтерфейсу для очікуваної структури повідомлення каналу RaceChannel гарантує, що звернення до поля зі швидкістю суперника не призведе до undefined за умови зміни формату серверної відповіді.

2. *Передача пропсів між компонентами*: визначення типів пропсів для кожного компонента, наприклад опису ігрової сесії для компонента SessionLineChart, усуває цілий клас помилок часу виконання, пов'язаних із неправильним форматом переданих даних.

Додаткові переваги у проекті: самодокументований код завдяки іменованим типам та інтерфейсам, розширена підтримка IDE (автодоповнення, рефакторинг, навігація за типами), підтримка всіх можливостей ECMAScript 6+ (класи, деструктуризація, генератори).

3.3 TailwindCSS: утилітарна стилізація та теми оформлення

TailwindCSS [9] – утилітарний CSS-фреймворк, що надає набір атомарних класів для стилізації безпосередньо у JSX-розмітці. На відміну від компонентних фреймворків на зразок Bootstrap, TailwindCSS не нав'язує готових компонентів – він надає примітиви (flex, grid, text-lg, bg-green-500 тощо), з яких розробник будує будь-який дизайн.

У розробленому застосунку TailwindCSS виконує дві ключові ролі:

1. *Базова стилізація компонентів***:** написання класів безпосередньо у JSX усуває необхідність підтримки окремих CSS-файлів і дозволяє бачити стилі поруч із розміткою.
2. *Підтримка п'яти кольорових тем* (зелена, червона, жовта, синя, чорна): завдяки механізму ThemeContext, що зберігає відповідний набір класів TailwindCSS, перемикання теми реалізовано динамічною підстановкою CSS-класів без написання окремих таблиць стилів для кожної теми та без перезавантаження сторінки.

Переваги підходу з TailwindCSS: висока швидкість розробки завдяки відмові від перемикання між файлами, адаптивний дизайн через префікси (sm:, md:, lg:), простота рефакторингу стилів.

3.4 WebSocket та ActionCable: двостороння комунікація в реальному часі

WebSocket [6] – протокол, стандартизований у RFC 6455 [14], що забезпечує повнодуплексний зв'язок між клієнтом і сервером через єдине постійне TCP-з'єднання. Після первинного HTTP-рукостискання (handshake) з'єднання переходить у стан WebSocket, після чого обидві сторони можуть надсилати фрейми даних у будь-який момент без нових запитів.

Порівняння доступних підходів до реалізації взаємодії в реальному часі подано у таблиці 2.

WebSocket обрано як оптимальний варіант: він забезпечує мінімальну затримку при двосторонньому обміні, що є критичним при відображенні прогресу суперника в режимі реального часу, і має широку підтримку у всіх сучасних браузерах.

ActionCable [7] – бібліотека на основі WebSocket, вбудована у фреймворк Ruby on Rails, що надає абстракцію каналів для організації підписок. Клієнтська частина підписується на канал RaceChannel; отримані повідомлення з полем швидкості суперника оновлюють відповідний стан компонента, що призводить до перерендерингу позиції «автомобіля» суперника на ігровому полі.

4. Реалізація механізмів авторизації

4.1 Управління глобальним станом через React Context

Механізм React Context [10] забезпечує передачу даних через дерево компонентів без «прокидання» пропсів через кожен проміжний рівень ієрархії. Фактично, це

глобальне сховище на рівні React-застосунку.

Таблиця 2

Порівняння технологій реального часу для клієнтських за стосунків

Технологія	**Механізм**	**Затримка**	**Напрямок**	**Накладні витрати**
HTTP short polling	Periodical GET	Висока	Клієнт → сервер	Великі (новий HTTP-запит кожного разу)
HTTP long polling	Утримуваний GET	Середня	Сервер → клієнт	Середні
Server-Sent Events	HTTP streaming	Середня	Лише сервер → клієнт	Низькі
WebSocket (RFC 6455)	TCP full-duplex	Низька	Двосторонній	Мінімальні після handshake

Визначено два контексти:

- UserContext – зберігає userId, userData (дані профілю) та стан авторизації. Після входу в систему всі компоненти, що споживають цей контекст (навігаційна панель, GameRoom, LeaderBoard, Account), отримують оновлені дані без додаткових запитів.
- ThemeContext – зберігає поточне значення теми та відповідний набір CSS-класів TailwindCSS. Перемикання теми через ThemeButton мгновенно оновлює стилі по всьому дереву компонентів.

4.2 JWT-авторизація

JSON Web Token (JWT) [11, 15] – стандарт (RFC 7519) компактного та самодостатнього способу передачі верифікованої інформації між сторонами у вигляді JSON-об'єкта з цифровим підписом. Токен складається з трьох частин: заголовка (алгоритм підпису), корисного навантаження (claims: userId, термін дії) та підпису.

Потік авторизації через JWT у застосунку:

1. Користувач вводить логін та пароль у компоненті LoginModal.
2. Клієнт надсилає POST-запит на сервер із обліковими даними.
3. Сервер перевіряє дані та повертає підписаний JWT.
4. Клієнт зберігає токен у localStorage і записує userId до UserContext.
5. При кожному HTTP-запиті або при встановленні WebSocket-з'єднання JWT включається у заголовок Authorization: Bearer <token>.
6. Сервер перевіряє підпис токена та термін його дії, після чого повертає захищений ресурс.

Перевага JWT перед традиційними серверними сесіями полягає у відсутності необхідності зберігати стан сесії на сервері: вся необхідна інформація знаходиться в самому токені, а сервер лише перевіряє цифровий підпис. Це спрощує горизонтальне масштабування серверної частини та дозволяє використовувати один токен для звернення до кількох мікросервісів або API-ендпоінтів.

Необхідно зауважити, що зберігання JWT у localStorage є прийнятним підходом для клієнтських SPA-застосунків при умові захисту від XSS-атак. У розробленому застосунку вхідні дані валідуються на стороні сервера, а зі сторони клієнта небезпечний HTML-вміст не вставляється у DOM без санітізації.

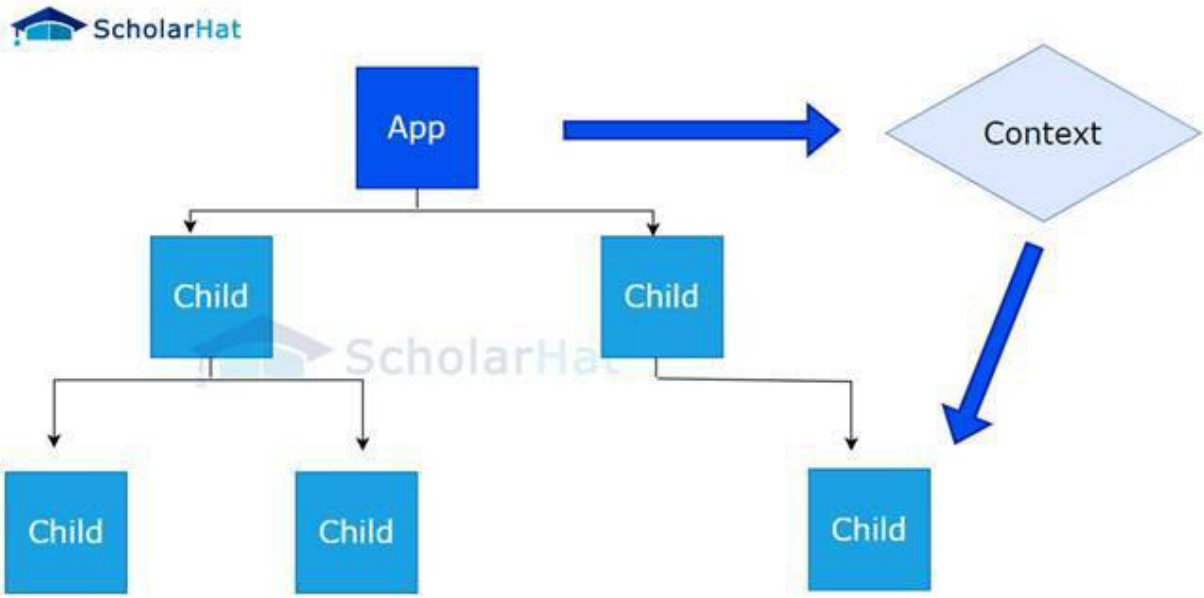

Рис. 4. Концепція контексту React (взято з сайту https://www.scholarhat.com/)

4.3 Авторизація через Google OAuth2

Альтернативний шлях авторизації реалізовано через Google OAuth2. Переваги: користувач не створює окремий пароль для застосунку, а сервер не зберігає паролі. Потік: клієнт перенаправляє користувача на сторінку Google → користувач підтверджує доступ → Google надає токен доступу → клієнт передає токен на сервер → сервер верифікує його через Google API, реєструє або авторизує користувача і повертає власний JWT.

5. Реалізація ігрової кімнати

Центральний компонент застосунку – GameRoom – відповідає за повний ігровий цикл. Компонент обслуговує два режими залежно від наявності параметра room_id у маршруті:

1. *Офлайн-режим* (/gameRoom): гравець практикується самостійно, результати відображаються після сесії, але не записуються у базу даних; WebSocket-підключення не використовується, що знижує навантаження на сервер.
2. *Онлайн-режим* (/gameRoom/:room_id): двосторонній WebSocket-зв'язок із сервером через RaceChannel; прогрес обох гравців синхронізується в реальному часі.

При завантаженні компонента в онлайн-режимі виконуються HTTP-запити:

1. /nick_names – отримання нікнеймів обох гравців кімнати;
2. /get_text – отримання тексту для поточної сесії;
3. /role – визначення ролі поточного користувача (хост або гість), що впливає на наявність кнопки «Start game».

Стан клавіатурного введення відслідковується посимвольно: для кожного символу підтримується статус correct / incorrect / pending. На основі цього обчислюється точність у реальному часі. Швидкість у символах за секунду (sps) обчислюється як відношення кількості правильно введених символів до часу від

початку гри та передається на сервер через sendMessage по WebSocket-каналу при кожній зміні.

Сервер трансльює отримане значення швидкості суперникові; клієнт відповідного гравця отримує повідомлення через підписку на RaceChannel і оновлює позицію «автомобіля» суперника. Ігровий процес таким чином синхронізується між двома клієнтами через сервер без прямого P2P-зв'язку.

Обробка помилок та граничних станів. При завантаженні онлайн-кімнати застосунок перевіряє приналежність user_id до поточної кімнати та відображає відповідне повідомлення: «Waiting for host to start…» якщо гравець не є хостом, «Start game» для хоста, «You are not allowed in this room» у разі спроби несанкціонованого доступу.

Анімація та візуалізація. Рух «автомобілів» відображається у форматі ASCII-art і оновлюється відповідно до поточної sps кожного гравця. Для плавних анімацій переходів між станами використовується Framer Motion [13]. Після завершення ігрової сесії компонент SessionLineChart відображає графік зміни швидкості введення впродовж сесії.

6. Реалізація таблиці лідерів та мультиплеєрної сторінки

6.1 Таблиця лідерів

Компонент LeaderBoard завантажує дані від ендпоінта /leaderboard методом GET і надає такі можливості взаємодії:

1. *Пошук* за нікнеймом: у хуку useMemo обчислюється масив filteredData – підмножина data, де поле nickname включає рядок із searchData. Результат оновлюється реактивно при кожній зміні введення.
2. *Сортування за точністю* (accuracy): двонапрямне сортування масиву за значенням відповідного поля; стан sortByAccuracy перемикається при кожному натисканні.
3. *Сортування за швидкістю* (sps): аналогічний механізм; при активації скидає стан sortByAccuracy.

Таблиця 3 порівнює метрики та можливості взаємодії в існуючих і розробленому рішеннях.

Таблиця 3

Метрики результативності та можливості таблиці лідерів

Метрика / функція	TypingClub	Nitro Type	10FastFingers	Розроблений застосунок
Кількість символів за секунду (sps)	Ні	Ні	Так	Так
Точність введення (%)	Так	Ні	Так	Так
Графік швидкості за сесію	Ні	Ні	Ні	Так
Таблиця лідерів	Ні	Так	Так	Так
Сортування в таблиці лідерів	Ні	Ні	Ні	Так
Пошук у таблиці лідерів	Ні	Ні	Ні	Так
Перегляд останніх ігор	Ні	Ні	Ні	Так

6.2 Мультиплеєрна сторінка

Компонент MultiplayerRoom відображає список активних ігрових кімнат, отриманих із сервера через ендпоінт /get_info_rooms. Підтримується сортування за

трьома критеріями:

1. *Наявність пароля* (password_status): кімнати з паролем на початку або в кінці списку.
2. *Кількість гравців* (players_count): заповнені або порожні кімнати – першими.
3. *Статус блокування* (game_lock_status): активні або заблоковані кімнати – першими.

Сортувальні стани є взаємовиключними: активація одного скидає інші. Пошук кімнат реалізовано за аналогією з LeaderBoard.

Для *створення нової кімнати* відображається модальне вікно, де користувач встановлює необов'язковий пароль. Після натискання «Create» виконується POST-запит на сервер; у відповідь повертається room_id, і клієнт перенаправляється до GameRoom з роллю хоста.

7. Аналіз результатів та наукова новизна

Розроблений застосунок реалізує декілька технічних підходів, сукупність яких відсутня в будь-якому з проаналізованих аналогів.

Поєднання однобічного потоку даних і двосторонньої WebSocket-комунікації. Стан ігрового поля є похідним від двох незалежних потоків: локального (клавіатурне введення) та зовнішнього (WebSocket-повідомлення від сервера). Розрізнення цих потоків у межах React-компонента вирішено через поділ стану: локальний стан управляється через useState, а ефекти підписки на WebSocket – через useEffect. Це забезпечує декларативне управління обома потоками в межах парадигми React без порушення принципів однобічного потоку даних.

Статична типізація для WebSocket-даних. Оголошення TypeScript-інтерфейсів для структур повідомлень каналу RaceChannel забезпечує перевірку кореляції між очікуваною і фактичною формою даних ще на етапі збірки. Це знижує ризик помилок виконання при зміні серверного API.

Динамічна тематизація через ThemeContext і TailwindCSS. Замість написання окремих CSS-файлів для кожної теми або використання CSS-змінних, підхід базується на динамічній підстановці наборів утилітарних класів TailwindCSS через контекст. Перемикання теми відбувається без перезавантаження сторінки та без втрати стану WebSocket-підписок.

Уніфікований компонент для двох режимів гри. Єдиний компонент GameRoom обслуговує офлайн-практику та онлайн-змагання, визначаючи режим через наявність параметра room_id у маршруті. Це зменшує дублювання коду і забезпечує консистентність ігрового інтерфейсу в обох режимах.

Порівняно з наявними рішеннями (TypingClub, Nitro Type, 10FastFingers) розроблений застосунок пропонує якісно відмінний набір можливостей: поєднання змагальної механіки, детальної аналітики сесій та персоналізованого інтерфейсу, заснованого на відкритому технологічному стеку зі стандартизованими протоколами (RFC 6455, RFC 7519).

Обмеження та напрями вдосконалення. Поточна реалізація підтримує лише двох гравців в одній кімнаті; розширення до груп потребуватиме перегляду архітектури каналу та структури даних. Зберігання JWT у localStorage є допустимим для навчального проекту, але у виробничому середовищі варто розглянути використання HttpOnly Cookies для підвищення стійкості до XSS. Адаптивний підбір текстів залежно від рівня підготовки користувача (наприклад, на основі частотності помилок по позиціях клавіатури) є перспективним напрямом, що зробить тренажер ефективнішим для цілеспрямованого вдосконалення навичок.

Загалом, результати роботи підтверджують, що сучасний стек React + TypeScript + TailwindCSS + WebSocket є достатнім і продуктивним рішенням для розробки інтерактивних освітніх застосунків реального часу в рамках кваліфікаційного проекту.

Висновки

У роботі розроблено клієнтську частину мультиплеєрного клавіатурного тренажера з використанням технологічного стеку React.js, TypeScript, TailwindCSS та ActionCable/WebSocket.

Проведений огляд наявних рішень (TypingClub, Nitro Type, 10FastFingers) виявив відсутність застосунку, що одночасно поєднує мультиплеєрний режим реального часу, детальну аналітику ігрових сесій, персоналізацію теми інтерфейсу та сучасні механізми авторизації.

Реалізовано такі ключові компоненти:

1) авторизація на основі JWT та Google OAuth2 із глобальним станом через React Context;
2) ігрова кімната (GameRoom) з підтримкою онлайн/офлайн режимів, посимвольним відстеженням точності та WebSocket-синхронізацією;
3) таблиця лідерів із пошуком і двонапрямним сортуванням;
4) мультиплеєрна сторінка з управлінням ігровими кімнатами;
5) механізм динамічної зміни теми через ThemeContext і TailwindCSS.

Обрані технологічні рішення забезпечують: мінімальну затримку відображення стану суперника (WebSocket, RFC 6455); надійність коду на етапі компіляції (TypeScript); модульність та повторне використання компонентів (React із хуками); гнучкість стилізації та підтримки тем (TailwindCSS).

Практичне значення розробки полягає у можливості використання застосунку як інструменту тренування навичок клавіатурного введення у навчальних закладах та для самостійного вдосконалення.

Перспективами подальшого розвитку є: впровадження алгоритмів адаптивного підбору тексту відповідно до поточного рівня підготовки користувача; розширення мультиплеєрного режиму до груп більше двох учасників; реалізація серверного рендерингу (SSR) для поліпшення продуктивності початкового завантаження; додавання інтернаціоналізації (i18n) для підтримки тренування введення різними мовами; впровадження Progressive Web App (PWA) для можливості офлайн-використання та встановлення на пристрій.

Розроблений застосунок демонструє доцільність застосування сучасного компонентного стеку (React + TypeScript + TailwindCSS) у поєднанні зі стандартизованими протоколами реального часу (WebSocket, RFC 6455) для побудови інтерактивних освітніх застосунків. Результати роботи можуть слугувати практичним орієнтиром для розробників аналогічних систем у сфері EdTech.

Список використаної літератури

1. TypingClub [Електронний ресурс]. – Режим доступу: https://www.typingclub.com (дата звернення: 20.05.2024).
2. Nitro Type [Електронний ресурс]. – Режим доступу: https://www.nitrotype.com (дата звернення: 20.05.2024).
3. 10FastFingers [Електронний ресурс]. – Режим доступу: https://10fastfingers.com (дата звернення: 20.05.2024).
4. React – A JavaScript library for building user interfaces [Електронний ресурс]. – Режим доступу: https://reactjs.org (дата звернення: 29.04.2024).

5. React State and Lifecycle. One-way data flow [Електронний ресурс]. – Режим доступу: https://reactjs.org/docs/state-and-lifecycle.html (дата звернення: 29.04.2024).
6. WebSocket API [Електронний ресурс] / MDN Web Docs. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/API/WebSocket (дата звернення: 19.05.2024).
7. How to use Action Cable with React and Rails [Електронний ресурс]. – Режим доступу: https://medium.com/swlh/how-to-use-action-cable-with-react-and-rails-3129a554c7d (дата звернення: 19.05.2024).
8. TypeScript – JavaScript with syntax for types [Електронний ресурс]. – Режим доступу: https://www.typescriptlang.org (дата звернення: 29.04.2024).
9. Tailwind CSS – A utility-first CSS framework [Електронний ресурс]. – Режим доступу: https://tailwindcss.com (дата звернення: 29.04.2024).
10. React Context API [Електронний ресурс]. – Режим доступу: https://reactjs.org/docs/context.html (дата звернення: 29.04.2024).
11. Introduction to JSON Web Tokens [Електронний ресурс]. – Режим доступу: https://jwt.io/introduction (дата звернення: 19.05.2024).
12. React Router [Електронний ресурс]. – Режим доступу: https://reactrouter.com (дата звернення: 29.04.2024).
13. Framer Motion – A production-ready motion library for React [Електронний ресурс]. – Режим доступу: https://www.framer.com/motion (дата звернення: 29.04.2024).
14. Fette I., Melnikov A. The WebSocket Protocol. RFC 6455 [Електронний ресурс] / IETF. – 2011. – Режим доступу: https://datatracker.ietf.org/doc/html/rfc6455 (дата звернення: 19.05.2024).
15. Jones M. et al. JSON Web Token (JWT). RFC 7519 [Електронний ресурс] / IETF. – 2015. – Режим доступу: https://datatracker.ietf.org/doc/html/rfc7519 (дата звернення: 19.05.2024).

References

1. TypingClub [Electronic resource]. Access: https://www.typingclub.com (accessed: 20.05.2024).
2. Nitro Type [Electronic resource]. Access: https://www.nitrotype.com (accessed: 20.05.2024).
3. 10FastFingers [Electronic resource]. Access: https://10fastfingers.com (accessed: 20.05.2024).
4. React – A JavaScript library for building user interfaces [Electronic resource]. Access: https://reactjs.org (accessed: 29.04.2024).
5. React State and Lifecycle [Electronic resource]. Access: https://reactjs.org/docs/state-and-lifecycle.html (accessed: 29.04.2024).
6. WebSocket API [Electronic resource] / MDN Web Docs. Access: https://developer.mozilla.org/en-US/docs/Web/API/WebSocket (accessed: 19.05.2024).
7. How to use Action Cable with React and Rails [Electronic resource]. Access: https://medium.com/swlh/how-to-use-action-cable-with-react-and-rails-3129a554c7d (accessed: 19.05.2024).
8. TypeScript [Electronic resource]. Access: https://www.typescriptlang.org (accessed: 29.04.2024).
9. Tailwind CSS [Electronic resource]. Access: https://tailwindcss.com (accessed: 29.04.2024).
10. React Context API [Electronic resource]. Access: https://reactjs.org/docs/context.html (accessed: 29.04.2024).
11. Introduction to JSON Web Tokens [Electronic resource]. Access: https://jwt.io/introduction (accessed: 19.05.2024).
12. React Router [Electronic resource]. Access: https://reactrouter.com (accessed: 29.04.2024).
13. Framer Motion [Electronic resource]. Access: https://www.framer.com/motion (accessed: 29.04.2024).
14. Fette I., Melnikov A. (2011) The WebSocket Protocol. RFC 6455. IETF. Access: https://datatracker.ietf.org/doc/html/rfc6455.
15. Jones M. et al. (2015) JSON Web Token (JWT). RFC 7519. IETF. Access: https://datatracker.ietf.org/doc/html/rfc7519.

TKACHENKO Oleksii,
Student, Department of Applied Mathematics and Informatics, The Bohdan Khmelnytsky National University of Cherkasy, Ukraine
DIDKOWSKY Ruslan,
Doctor of Technical Sciences, Associate Professor, Department of Informatics and Applied Mathematics, The Bohdan Khmelnytsky National University of Cherkasy, Ukraine
PISKUN Oleksandr,
Candidate of Technical Sciences, Associate Professor, Head of Department of Applied Mathematics and Informatics, Bohdan Khmelnytsky National University of Cherkasy

DEVELOPMENT OF THE CLIENT-SIDE PART OF A MULTIPLAYER KEYBOARD TRAINER USING REACT AND WEBSOCKET TECHNOLOGIES

Summary. Introduction. *The increasing volume of digital communication and the spread of remote work and learning enhance the importance of keyboard typing skills. This competence is essential not only for IT professionals but for a wide range of users: students, office workers, journalists, and researchers. Specialized typing trainers with feedback on speed and accuracy are a proven tool for developing this skill. However, most available solutions either focus exclusively on individual practice without a social component or provide multiplayer mode without comprehensive session analytics and personalized interface. From a technical perspective, a real-time multiplayer trainer presents challenges not present in individual trainers: synchronous state transmission with minimal latency, correct handling of player connections and disconnections, concurrent server resource access, and a responsive interface.*

Purpose. *The aim of this work is to develop the client-side part of a multiplayer keyboard trainer using the technology stack of React.js, TypeScript, TailwindCSS, and ActionCable/WebSocket, enabling real-time competitions, user authentication, and session result analytics.*

Results. *A comparative analysis of three popular keyboard trainer web applications – TypingClub, Nitro Type, and 10FastFingers – was conducted. The analysis revealed that none of these solutions combines real-time multiplayer, detailed session analytics in graphical form, personalized interface with theme switching, and modern authentication mechanisms. Based on this, the requirements for the developed application were defined.*

The client-side is implemented as a Single Page Application (SPA) using React.js with hooks and the Context API for state management. TypeScript provides compile-time type safety, which is especially important for WebSocket message structures. TailwindCSS enables dynamic theme switching across five color schemes without separate CSS files. ActionCable over the WebSocket protocol (RFC 6455) ensures low-latency bidirectional communication between game participants. Two authentication flows are implemented: JWT-based (RFC 7519) and Google OAuth2. The GameRoom component supports both offline practice and online multiplayer modes, determined by the presence of the room_id route parameter; character-by-character accuracy tracking and speed synchronization via WebSocket are implemented. The LeaderBoard component provides search and bidirectional sorting by accuracy and speed. The MultiplayerRoom component enables browsing and creating game rooms with sorting by password status, player count, and lock status.

Conclusion. *The developed application combines real-time multiplayer functionality, detailed session analytics, flexible interface personalization, and standard-based authentication mechanisms – a combination absent in any of the analyzed existing solutions. The technology stack ensures code reliability at compile time (TypeScript), minimal latency in multiplayer synchronization (WebSocket, RFC 6455), component reusability and predictable state management (React with one-way data flow), and styling flexibility (TailwindCSS). Directions for further development include adaptive text selection algorithms based on user skill level, support for group competitions with more than two participants, and server-side rendering for improved initial load performance.*

Keywords: *keyboard trainer, React.js, TypeScript, TailwindCSS, WebSocket, ActionCable, JWT, OAuth2, multiplayer web application, SPA, one-way data flow.*

Одержано редакцією 05.11.2024 р.
Прийнято до публікації 11.12.2024 р.