Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

РОЗРОБКА ВЕБ-ЗАСТОСУНКУ ДЛЯ ВИВЧЕННЯ СЛІПОГО НАБОРУ ТЕКСТУ НА КЛАВІАТУРІ

DEVELOPMENT OF A WEB APPLICATION FOR STUDYING BLIND TYPING ON A KEYBOARD

спеціальність 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

освітня програма «Інженерія програмного забезпечення»
(назва освітньої програми)

Виконав: здобувач вищої освіти
групи ІПЗ-41
Трощук Вадим Васильович
_____
(підпис)

Керівник:
д.т.н., професор
Андрущак Ігор Євгенович
_____
(підпис)

Кваліфікаційну роботу
допущено до захисту
«8» 06 2023 р.
Гарант освітньої програми:
к.т.н., доцент
Ліщина Наталія Миколаївна
_____
(підпис)

Луцьк – 2023 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *інженерії програмного забезпечення*
Ступінь вищої освіти: *бакалавр*
Галузь знань: *12 Інформаційні технології*
Спеціальність: *121 Інженерія програмного забезпечення*
Освітня програма: *«Інженерія програмного забезпечення»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____
«2» 02 2023р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Трощуку Вадиму Васильовичу

1. Тема кваліфікаційної роботи: *Розробка веб-застосунку для вивчення сліпого набору тексту на клавіатурі / Development of a web application for studying blind typing on a keyboard.*
керівник роботи: *д.т.н., професор Андрущак І.Є.*
затверджені наказом закладу вищої освіти від «03» лютого 2023 р. № 80-05-35
2. Строк подання здобувачем роботи: 08.06.2023 р.
3. Вихідні дані до роботи: *Дослідження та наукові статті щодо сліпого набору тексту: Вимоги до розробки програмного забезпечення. Мова програмування серверної частини – JavaScript з використанням JSON. Технології програмування клієнтської частини – HTML, CSS, JavaScript, React, MaterialUI.*
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): *Аналіз проблеми щодо сліпого набору тексту та її поточного стану:. Аналіз та порівняння навичок використання сліпого набору тексту. Аналіз , визначення вимог до розроблюваного програмного забезпечення та проектування програмного забезпечення. Опис розробки та тестування розблюваного веб-застосунку, опис роботи веб-застосунку.*
5. Перелік графічного матеріалу: *13 рис. 2 діаграми,*

6. Консультанти розділів роботиг

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		Завдання видав	Завдання прийняв
Аналіз предметної області	*Андрущак І.С.*		
Аналіз, визначення вимог до розроблюваного програмного забезпечення та проектування програмного забезпечення	*Андрущак І.С.*		
Розробка веб-застосунка для вивчення сліпого набору тексту на клавіатурі	*Андрущак І.С.*		
Нормоконтроль	*Андрущак І.С.*		
Гарант ОП	*Ліщина Н.М.*		
Показник запозичень тексту	3.37%		
Академічна доброчесність	*Яцук А.А.*		

7. Дата видачі завдання *«2» лютого 2023 р.*

КАЛЕНДАРНИЙ ПЛАН

№ З/п	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1	*Провести огляд літературних джерел по темі кваліфікаційної роботи бакалавра*	*05.02.2023 р.*	Виконано
2	*Провести порівняльну характеристику програмних продуктів*	*27.02.2023 р.*	Виконано
3	*Розробка функціонально-структурної схеми роботи об'єкта проектування*	*10.03.2023 р.*	Виконано
4	*Обґрунтування вибору шляхів, технологій (алгоритмів) і засобів вирішення поставленого завдання*	*17.04.2023 р.*	Виконано
5	*Практична реалізація об'єкта проектування*	*18.05.2023 р.*	Виконано
6	*Тестування та налагодження об'єкта проектування*	*01.06.2023 р.*	Виконано
7	*Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедру*	*08.06.2023 р.*	Виконано

Здобувач вищої освіти _____ Яцук В.В
(підпис) (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ Андрущак І.С
(підпис) (прізвище, ініціали)

Міністерство освіти і науки України
Луцький національний технічний університет

Рецензія
на кваліфікаційну роботу

Здобувач вищої освіти:
Трощук Вадим Васильович

Тема: Розробка веб-застосунку для вивчення сліпого набору тексту на клавіатурі / Development of a web application for studying blind typing on a keyboard.

Коротка характеристика кваліфікаційної роботи:
Робота складається з трьох розділів, протягом яких дипломантом здійснюється огляд поставленої перед ним проблеми, пошук шляхів її вирішення та опис їх реалізації. Для веб-застосунку було використано наступні програмні засоби: текстовий формат передачі даних JSON, мова програмування JavaScript для реалізації основної логіки застосунку, бібліотека для створення інтерфейсу користувача — React.js та Material-UI.
Самостійні розробки і пропозиції автора:
Дипломант пропонує використовувати SPA підхід при створенні веб-застосунку для вивчення сліпого набору тексту на клавіатурі, що включає реалізацію системи тестування швидкості друку та її точності, надання статистки щодо продуктивності друку.
Практичне значення роботи:
Було створено веб-застосунок для вивчення сліпого набору тексту на клавіатурі, що допомагає людям вивчати даний метод користуючись розробленим застосунком.

Недоліки:
У роботі не виявлено новації – вона більш походить на альтернативу існуючим варіантам вирішення проблеми, проте слід помітити що складна структура роботи добре висвітлює вміння дипломанта.
Загальний висновок:
Рівень якості виконаної роботи є задовільним. Усі поставлені завдання успішно й повноцінно виконані. Кваліфікаційна робота виконана на якісному рівні тож рекомендується до захисту та заслуговує на високу оцінку.

Рецензент: старший викладач, к.т.н. Кошелюк В.А.
(прізвище, ім'я, по-батькові, посада)

Рецензент кваліфікаційної роботи _____
(підпис)

«08» червня 2023 р.

АНОТАЦІЯ

Трощук В.В. Розробка веб-застосунку для вивчення сліпого набору тексту на клавіатурі. Рукопис.

Кваліфікаційна робота бакалавра. Кваліфікаційна робота за спеціальністю 121 Інженерія програмного забезпечення. Луцьк 2023. 53 с.

Кваліфікаційна робота бакалавра складається зі вступу, трьох розділів, висновків до кожного розділу, загальних висновків, списку використаних джерел та додатків. У першому розділі досліджено предметну область де висвітлено стан проблеми, переваги, ризики та її перспективи. У другому розділі було здійснено аналіз та визначення вимог до розроблюваного програмного забезпечення та проектування програмного забезпечення . Третій розділ пояснює процес розробки веб-застосунку з посиланнями на лістинги коду у додатках, та здійснення тестування його тестування. У висновках узагальнено інформацію, відображену у розділах. Результат розробки може бути застосований в практичній діяльності, а саме навчання його користувачів для вивчення сліпого набору тексту на клавіатурі.

Ключові слова: веб-застосунок, React, JSON, JavaScript, UML, клавіатура, текст.

ANNOTATION

Troshchuk V.V. Development of a web application for studying blind typing on a keyboard. – Manuscript.

Bachelor's thesis. Bachelor's thesis in the specialty 121 Software engineering. Lutsk 2023. 53p.

The Bachelor's consists of an introduction, three chapters, conclusions for each chapter, a list of used sources, and appendices.

In the first chapter, the subject area was explored, highlighting the current state of the problem, its advantages, risks, and prospects. The second chapter involved an analysis and determination of requirements for the developed software and software design. The third chapter explains the process of developing the web application with code listings provided in the appendices, as well as the testing conducted. The conclusions summarize the information presented in the chapters. The outcome of the development can be applied in practical activities, specifically in the training of users to learn touch typing on a keyboard.

Keywords: web application, React, JSON, JavaScript, UML, keyboard, text.

ЗМІСТ

ВСТУП

Актуальність теми проявляється в тому, що швидкість і точність набору тексту при комп’ютерному наборі відіграють важливу роль у різних сферах діяльності. Знання техніки друку сліпим методом набору, коли людина може друкувати не дивлячись на клавіатуру і використовуючи усі пальці рук, стає все більш потрібним навиком. Це дозволяє збільшити продуктивність, знизити час, витрачений на набір тексту, і зменшити кількість помилок. Проте, багато людей мають труднощі з оволодінням цим навиком, в результаті чого і виникає потреба у розробці веб-застосунку, спрямованого на вивчення сліпого набору тексту на клавіатурі. Тому розробка веб-застосунку для вивчення сліпого набору тексту на клавіатурі має велике значення в сучасному світі, де швидкість та ефективність виконання різноманітних завдань є ключовими факторами успіху.

Метою дослідження є аналіз та розробка основних функціональних можливостей, які потрібні для реалізації веб-застосунка, що допомагає у вивченні сліпого набору тексту на клавіатурі з використанням таких технологій, як JavaScript, React.js, JSON та MaterialUI.

Об’єктом дослідження даної кваліфікаційної роботи є розробка веб-застосунка для вивчення сліпого набору тексту на клавіатурі.

Предметом дослідження – є технології та методи розробки веб-застосунків для вивчення сліпого набору тексту на клавіатурі.

Важливим аспектом розробки веб-застосунку для вивчення сліпого друку є використання технологічних інструментів, таких як React, JSON і JavaScript. Ці інструменти надають широкі можливості для реалізації потрібного функціоналу. Результати цього дослідження значно сприятимуть розумінню процесу розробки веб-застосунків для вивчення сліпого друку. Вони нададуть цінні уявлення та рекомендації для фахівців або організацій, які цікавляться створенням подібних освітніх застосунків

Крім того, дослідження також буде корисним для потенційних користувачів веб-застосунку. Аналізуючи функції та можливості, воно розуміння для студентів, які планують використовувати такі застосунки для вдосконалення навичок сліпого друку. Рекомендації дослідження допоможуть користувачам приймати обґрунтовані рішення при виборі та використанні таких застосунків, забезпечуючи максимальну користь та оптимальний досвід навчання.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАВДАНЬ НА КВАЛІФІКАЦІЙНУ РОБОТУ

1.1 Аналіз сучасного стану проблеми

У сучасному швидкоплинному світі комп'ютерні технології стали невід'ємною складовою нашого щоденного життя. Завдяки ним ми можемо легко спілкуватися, швидко знаходити необхідну нам інформацію та використовувати її для виконання роботи або ж у навчальних цілях. З огляду на неперервний розвиток згаданої сфери, стає все очевидніше, що володіння фундаментальними комп'ютерними навичками є надзвичайно важливим. Це факт, який відбиває зміну суспільства та підкреслює необхідність оволодіти цими навичками для успішної адаптації до сучасного цифрового середовища.

Одним із основних навичок, який набуває все більшої важливості, є вміння сліпого набору тексту на клавіатурі. Сліпий набір передбачає здатність друкувати без необхідності дивитися на клавіші клавіатури, використовуючи м'язову пам'ять та злагоджену роботу рук і очей. У світі, де доступ до інформації є безмежним, вміння швидко та точно набирати текст стає особливо цінним. Вміння швидко та впевнено набирати текст не тільки забезпечує ефективність та точність роботи, але й дозволяє зекономити час. Звільнений час можна використати для розв'язання інших важливих завдань, що сприяє загальній продуктивності та успіху.

Метод сліпого друку – це коли ви набираєте текст, не дивлячись на клавіатуру і працюючи всіма десятьма пальцями [1]. Це розподілення задач має велике значення, оскільки різні ділянки клавіатури призначені для різних рук.. Навички друку складаються з двох компонентів, швидкість і точність друку. Швидкість друку зазвичай вимірюється двома шкалами, а саме CPM (символів за хвилину) і WPM (слів за хвилину). CPM – це кількість правдивих слів, набраних за одну хвилину. WPM – це кількість стандартних слів, які складаються з п'яти літер, що були набрані за одну хвилину. WPM можна

отримати з CPM, що було поділена на п’ять [2]. Точність ж вимірюється у відсотках, де 100 це максимальний показник точності. З меншою кількістю помилок, що виникають під час друку, люди можуть підтримувати високий рівень точності у своїй роботі. Важливою перевагою є миттєве виявлення та виправлення помилок під час друку. У професійних сферах та робочих середовищах, де вимагається абсолютна точність, це стає вирішальним фактором [3].

У дослідженні, проведеному Гордоном Логаном та Джаною Ульріх, вивчалися характеристики непідготовлених людей, які самотужки навчилися друкувати, дослідники оцінили їх продуктивність, моделі погляду та стратегії рухів руками. Дивно, але дослідження показало, що друкарки-самоучки можуть досягти рівня продуктивності, порівнянного з навченими друкарками, незважаючи на використання меншої кількості пальців у своїй техніці друку [4]. Також було виявлено, що друкарі з нетрадиційним способом набору тексту, що не використовували метод сліпого набору тексту, значно залежать від візуальних підказок, і їх швидкість та точність зменшуються, коли їм не дозволяється дивитися на клавіатуру. Однак дослідження також вказує, що різниця в швидкості між тими, хто володіє навичками сліпого друку, та тими, хто не володіє, має мінімальний вплив на завдання, такі як обмін повідомленнями та подібні дії. Ці висновки спонукали дослідників задуматися над необхідністю акцентування уваги навчальних закладів на ранньому навчанні студентів сліпому набору тексту.

Переважна частина літератури про набір тексту будь-яким методом стосується висококваліфікованих професійних машиністів, але сучасні навички друку здебільшого є результатом необмеженої тривалої практики. Тому дослідниками було вирішено провести опитування великої когорти студентів через онлайн платформу та здійснити аналіз їх продуктивності в залежності від того, яким методом набору тексту вони користуються [5]. На рисунку 1.1 продемонстровано звички друкування у відповідних групах опитуваних, де можна зазначити кілька аспектів. Перш за все можна виділити те, скільки

пальців використовується під час друкування, оскільки це впливає на швидкість і точність. Нарешті, важливо враховувати частоту поглядів на клавіатуру під час друкування. На стовпчикових діаграмах можна побачити відсоток найбільш та найменш досвідчених друкарів у кожній категорії цих факторів.

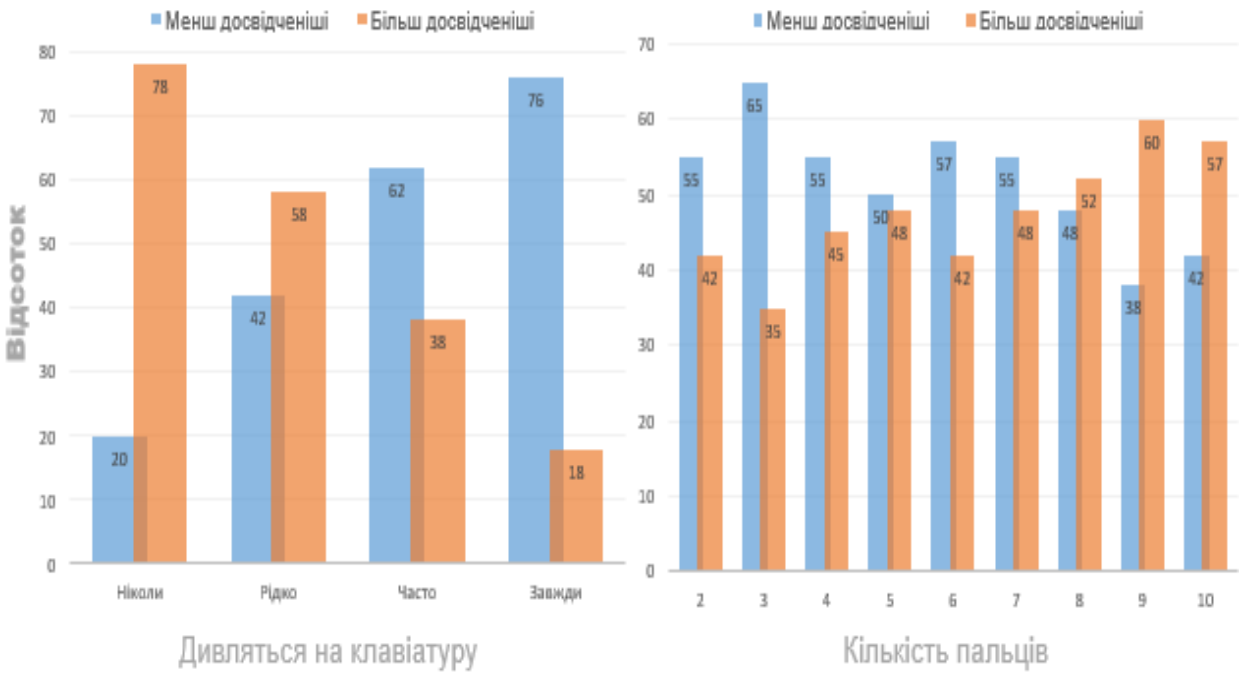

Рисунок 1.1 – Звички друку відповідно до груп опитування

На графічному зображенні (рис. 1.2) представлені результати дослідження, де учасники були розділені на дві групи в залежності від рівня їх досвіду та навичок у друку тексту. Кожна точка на графіку відповідає окремій людині і відображає її показники швидкості та точності набору тексту. Групи продуктивності можна охарактеризувати як більш досвідчених та менш досвідчених друкарів. Кожна група виділена різним кольором та формою точки, що допомагає легко розрізнити їх. Таким чином, ми можемо одразу спостерігати, які учасники відносяться до більш досвідченої групи, а хто - до менш досвідченої. Крім того, у графіку представлені коробкові діаграми, які деталізують розподіл швидкості та точності між цими двома групами. Ці діаграми вказують, яка частка учасників з кожної групи досягає певного рівня

продуктивності та точності. Вони слугують додатковим засобом для порівняння результатів і виявлення відмінностей між групами.

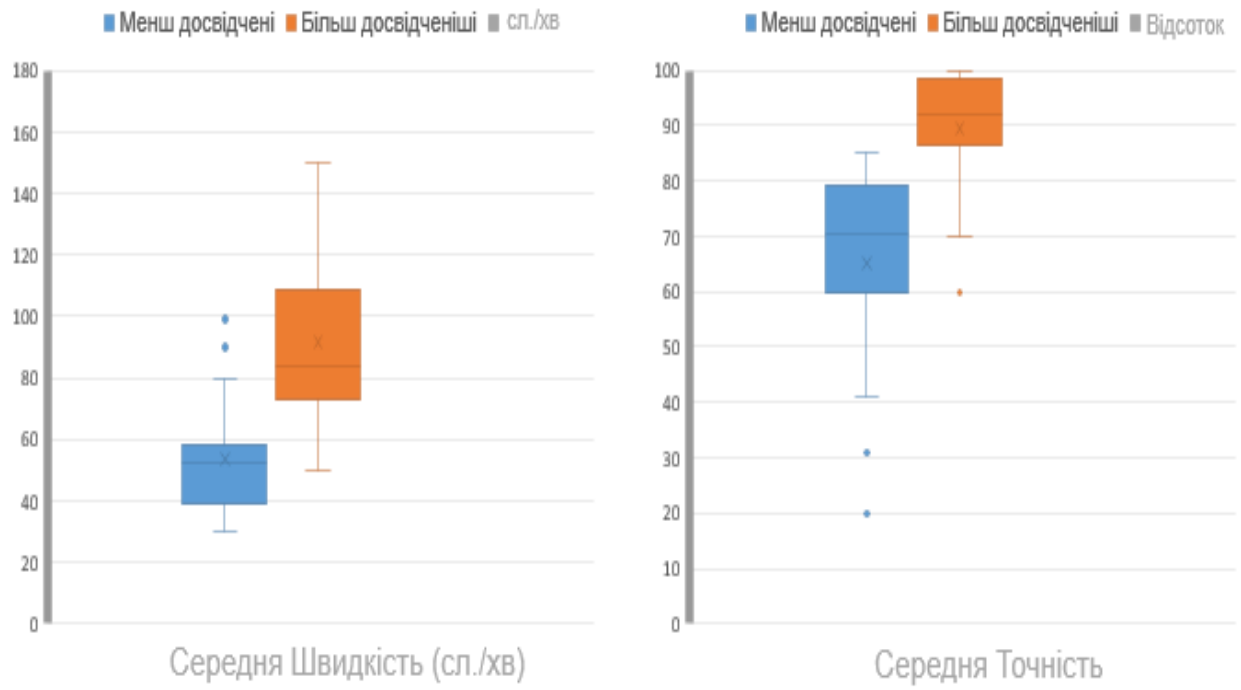

Рисунок 1.2 – Звички друку відповідно до груп опитування

Аналізуючи цю графічну інформацію, ми можемо отримати глибше розуміння професійної ефективності та навичок друкування учасників дослідження. Він допомагає визначити рівень продуктивності та виявити різницю між більш досвідченими та менш досвідченими друкарями. Цей аналітичний підхід дозволяє нам систематично проаналізувати результати, виокремити ключові аспекти та зробити відповідні висновки.

У сфері веб-застосунків для вивчення сліпого набору тексту на клавіатурі важливо мати чітке уявлення про наявні аналоги, які вже розроблені для цієї конкретної галузі. Оглядаючи та оцінюючи ці аналоги, ми можемо отримати цінну інформацію про їх переваги та недоліки. Провівши дослідження, що включало аналіз літератури, щодо сліпого друку, надалі зосередимося на вивченні існуючих аналогів, які доступні на ринку. Цей аналіз дозволить отримати цінні відомості про їх особливості, функціональні можливості та

досвід користувачів існуючих аналогів проектованої системи, що дасть змогу визначити шлях для розробки нашого веб-застосунку.

Коли мова йде про веб-застосунки для вивчення сліпого набору тексту на клавіатурі, Typing.com – один з найперших перших варіантів, що приходить на думку [6]. Ця платформа має декілька помітних переваг, які сприяють його популярності серед тих, хто бажає покращити навички сліпого методу друку.

Інтуїтивний та легкий у користуванні інтерфейс платформи забезпечує зручний досвід для користувачів. Його всебічна навчальна програма охоплює основні техніки вивчення друку і спрямована на учнів різного рівня. Інтерактивні уроки та вправи залучають користувачів та сприяють активному участі, поліпшуючи засвоєння навичок.

Крім того, Typing.com надає потужні функції відстеження прогресу та аналізу результатів, що дозволяє особам, що навчаються відстежувати свій прогрес та виявляти області, що потребують покращення.

Однак варто враховувати кілька недоліків. Одним із таких є обмежені можливості налаштування, що обмежує користувачів у пристосуванні навчального досвіду до своїх конкретних потреб та уподобань. Крім того, хоча і сервіс фокусується на основних навичках сліпим методом друку, в ньому може бракувати функцій або спеціалізованих вправ для досвідчених друкарів, які шукають більш складні техніки.

Продовжуючи огляд існуючих аналогів, наступний сервіс, що вартий своєї уваги це – 10FastFingers [7]. Він виділяється серед платформ для навчання сліпого методу друку завдяки сильному акценту на розвиток навичок швидкого набору тексту. Його спрямованість на швидкість ґрунтується на проведенні тестів, які мають відлік часу, що дозволяє користувачам випробовувати себе і відстежувати прогрес у швидкості набору тексту. Елемент конкуренції додає захоплення, дозволяючи користувачам порівнювати свою швидкість з іншими по всьому світові і підтримувати почуття мотивації та своїх досягнень серед інших користувачів. Ще однією важливою особливістю є його підтримка декількох мов. Цей аспект відрізняє його від багатьох інших аналогічних

платформ, оскільки він дозволяє користувачам тренувати набір тексту на різних мовах. Ця універсальність може бути особливо корисною для осіб, які володіють декількома мовами або бажають покращити свої навички набору тексту не лише українською мовою.

Проте, незважаючи на свої переваги, 10FastFingers має деякі мінуси. Одним з помітних є відсутність всебічної навчальної програми для вивчення набору тексту сліпим методом з нуля. Навіть якщо платформа наголошує на покращенні швидкості, вона може не надати достатньої підтримки або структурованих уроків для початківців, які не знайомі з технікою сліпого методу друку, тому він більше виступає як тренажер, аніж метод для навчання.

Така відсутність системного підходу може завадити розвитку важливих навичок набору тексту та загальному процесу навчання. Крім того, 10FastFingers не відповідає вимогам стеження за прогресом та аналітикою продуктивності. Незважаючи на те, що він надає базові метрики, такі як швидкість письма, у нього відсутні розширені можливості стеження та аналізу. Користувачам може бути складно відстежувати свій прогрес протягом часу або отримати докладні відомості про точність, ефективність письма або області, які потребують вдосконалення. Це обмеження може завадити користувачам встановлювати цілі, відстежувати свою продуктивність та ефективно вдосконалювати свої навички набору тексту.

Ratatype – є відомою веб-платформою яка сприяє вивчення сліпого методу друку [8]. Одна з ключових його переваг полягає в його інтерактивних уроках. Платформа пропонує різноманітні вправи та тренування, що активно залучають користувачів до процесу навчання. Завдяки цим інтерактивним заняттям користувачі можуть розвивати м’язову пам’ять, покращувати розташування пальців та вдосконалювати загальну техніку набору тексту.

Також варто відзначити можливості налаштування, які забезпечує платформа. Сервіс надає користувачам гнучкість у налаштуванні навчального процесу з урахуванням їхніх конкретних потреб та вподобань. Учні можуть обирати конкретні уроки, фокусуватися на певних навичках набору тексту або

навіть налаштовувати рівень складності відповідно до свого рівня. Ця можливість налаштування дає користувачам знаходити свої проблемні зони, після чого покращувати їх, що сприяє більш персоналізованому та ефективному процесу навчання. Також сервіс вміщує функціонал комплексного відстеження результатів та аналітики. Користувачі мають доступ до докладної статистики та звітів про прогрес, що дозволяє відстежувати швидкість набору тексту, точність та загальний прогрес навчання. Цей підхід, що заснований на зборі даних, не лише надає користувачам цінну інформацію про їх розвиток, але й дозволяє встановлювати цілі, відстежувати досягнення та залишатися мотивованими протягом всього навчального процесу.

Незважаючи на свої переваги, Ratatype має деякі прогалини. Одним із них є обсяг його навчальної програми. В той час як платформа надає міцний фундамент для навчання новачкам, людям, які шукають більш складні техніки або спеціалізовані вправи, можуть здатися доступні уроки недостатніми.

Варто зауважити, що він переважно акцентується на покращенні швидкості та точності набору тексту, а не на глибокому розгляді методів для вивчення методів сліпого набору тексту. До того ж, інтерфейс користувача може не бути таким простим у використанні або візуально привабливим, як у деяких інших платформ. Хоча функціональність залишається без змін, більш дотепний та зручний інтерфейс може поліпшити загальний досвід користувачів та зробити платформу більш привабливою та приємною у використанні.

Після детального аналізу сильних та слабких сторін наведених аналогів, стає очевидним, що розробка нового застосунку має пріоритетність у забезпеченні зручного та насамперед інтуїтивно зрозумілого інтерфейсу, комплексної та добре структурованої програми навчання, можливості налаштування, що відповідає індивідуальним потребам користувача, а також засобів відстеження прогресу. Важливо, щоб ці елементи були гармонійно інтегровані. Такий веб-застосунок повинен бути привабливим та простим у використанні, забезпечуючи високу якість викладання та сприяючи успіху в навчанні користувачів.

Дослідження показує, що створення нової системи є цілком доцільним, оскільки в існуючих аналогах були виявлені прогалини. Наприклад, Typing.com хоча і пропонує зручний інтерфейс та комплексну навчальну програму, але має обмежені можливості налаштування та брак функцій для досвідчених користувачів. 10FastFingers акцентує увагу на швидкості, але не має комплексної навчальної програми та можливостей відстеження прогресу. Ratatype пропонує цікаву навчальну програму та відстеження прогресу, але не має елемента конкуренції та підтримки кількох мов для вивчення друку.

Заповнення цих прогалин та впровадження сильних сторін, виявлених у існуючих аналогах, дозволить створити нову покращену систему. Це підтверджує доцільність створення нового веб-застосунку, який надасть користувачам винятковий досвід навчання для поліпшення їх навичок сліпого набору тексту.

1.2 Постановка завдання на кваліфікаційну роботу бакалавра

Основною метою даної кваліфікаційної роботи є реалізація веб-застосунка для вивчення сліпого набору тексту на клавіатурі. Щоб досягти успіху у цьому завданні, необхідно приділити особливу увагу архітектурі та проектуванню програмного забезпечення, керуючись усіма нормами та принципами, а також використовуючи сучасні технології. Провести детальний огляд літератури, що відповідає поставленій задачі. Ретельний архітектурний підхід допоможе створити добре структуроване рішення. Дотримання норм забезпечить надійність та масштабованість, а використання сучасних технологій сприятиме розширенню функціональності та підвищенню безпеки. Такий підхід ляже в основу успішного та високоякісного програмного рішення.

Цілі, що розкривають суть завдання і повинні бути досягнуті в результаті виконання атестаційної роботи:

– провести комплексний огляд наукових джерел з питань сліпого набору тексту та веб-застосунків для його вивчення;

– провести аналіз існуючих аналогів проектованої системи визначивши переваги та недоліки для кожного із них;

– провести аналіз вимог для проектованого веб-застосунку;

– скласти план розробки проектованої системи;

– розробити функціонал веб-застосунку для вивчення сліпого набору тексту на клавіатурі;

– провести тестування веб-застосуноку на відповідність заданим йому вимогам після аналізу ПЗ.

Висновки до розділу 1

У першому розділі було проведено ретельний аналіз наукових джерел, що стосуються предметної області у відповідності до обраної теми кваліфікаційної роботи. Була здійснена комплексна рецензія, в рамках якої були вивчені актуальні дослідження, наукові статті, методики та практики, пов’язані з тематикою вивчення сліпого набору тексту на клавіатурі.

Після цього, існуючі аналоги в цій області були ретельно проаналізовані, звернувши увагу на їх переваги та недоліки. На основі проведеного аналізу, було обґрунтовано доцільність створення веб-застосунку для навчання сліпого набору тексту на клавіатурі.

Були визначені цілі, що розкривають суть завдання кваліфікаційної роботи і які повинні бути досягнуті в результаті її виконання.

РОЗДІЛ 2
СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ

2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проектування програмного забезпечення

У процесі розробки програмного забезпечення вибір відповідної моделі для моделювання системи та її елементів має вирішальне значення. Це закладає основу для всього процесу розробки програмного забезпечення і значно впливає на успіх кінцевого продукту. Обрана модель відіграє важливу роль у виявленні структури, поведінки та взаємодії компонентів програмного забезпечення. Вона надає систематичний підхід до розуміння вимог, проектування архітектури та реалізації функціональності проектованої системи. Вибираючи відповідну модель, ми можемо ефективно представити елементи, поведінку та взаємодію системи, що призводить до добре організованого та зрозумілого користувацького досвіду. Крім того, добре структурована модель сприяє легкому обслуговуванню, масштабованості та розширюваності застосунку.

У контексті даного об’єкту розробки як модель для моделювання проектованої системи було обрано уніфіковану мову моделювання, або ж UML. Вона пропонує стандартизований та широко використовуваний набір нотацій та діаграм, які полегшують ефективну комунікацію та документування програмних систем. Його універсальність дозволяє нам охопити різні аспекти проектованої системи, включаючи його структуру, поведінку та взаємодії користувачів [9].

Обґрунтування вибору UML як мови для моделювання програмної системи ґрунтується на кількох факторах. UML забезпечує візуальне представлення, яке допомагає зрозуміти складність системи та сприяє ідентифікації та аналізу вимог до розроблюваного програмного забезпечення. Застосовуючи діаграми UML, такі як діаграми використання, класів та діяльності, ми можемо візуально зображувати функціональні вимоги, зв’язки

між компонентами системи та його взаємодії між користувачами проектованої системи [10].

Ось декілька ключових аспектів, що лише підтверджують згадане твердження про важливість використання UML:

– надає стандартизовану нотацію та набір символів, що дозволяє забезпечити послідовність та зрозумілість моделювання в різних командах та проектах;

– пропонує різноманітні типи діаграм, такі як діаграми класів, послідовностей та діяльності, що дозволяє розробникам вибрати найбільш зручніший спосіб представлення для різних аспектів системи;

– існує безліч програмних застосунків, які підтримують моделювання UML, надаючи функції, такі як створення діаграм, перевірка та генерація коду, що покращує продуктивність та ефективність розробки;

– допомагає візуалізувати архітектуру системи та взаємозв’язки між її компонентами;

– підтримує виявлення, аналіз та документування вимог до системи, забезпечуючи чітке розуміння необхідної функціональності.

В контексті нашої роботи, що спрямована на розробку веб-застосунку для вивчення сліпого набору тексту на клавіатурі, вже було доведено причину вибору UML як найбільш відповідного нам підходу до моделювання. Після детального обговорення причин за цим рішенням, ми тепер звертаємо увагу на аналіз вимог до програмного забезпечення нашого веб-застосунку.

Аналіз вимог до програмного забезпечення є однією із основ під час проектування майбутньої проектованої системи. Вони дають змогу визначати функціональність, продуктивність та обмеження проектованої системи. Чітко сформульовані вимоги гарантують відповідність кінцевого продукту, що очікують користувачі.

Функціональні вимоги проектованої системи:

1) користувач повинен мати можливість отримати доступ до головної сторінки додатка для навчання друку на клавіатурі;

2) веб-застосунок повинен надавати тестування швидкості друку як у режимі слів, так і в режимі речень;

3) у режимі слів веб-застосунок повинен пропонувати різні джерела слів залежно від рівнів складності, що включають англійську та українську мови;

4) веб-застосунок повинен підтримувати чотири режими тривалості, з можливістю вибору 90 секунд, 60 секунд, 30 секунд та 15 секунд;

5) у режимі речень додаток повинен генерувати випадкові короткі речення як англійською, так і українською мовами;

6) веб-застосунок повинен мати можливість встановити бажану кількість речень за один тест, з можливістю вибору в 5 речень, 10 речень або 15 речень;

7) веб-застосунок повинен надавати статистику щодо продуктивності друку, включаючи:

1) кількість слів за хвилину, або ж WPM показник;

2) кількість натискань клавіш за хвилину або ж KPM показник;

3) показник точності.

8) веб-застосунок повинен включати функцію «Words Card» яка буде випадково показувати слово, що потрібно ввести користувачеві;

9) веб-застосунок повинен вміщувати режим «Coffee Mode» який даватиме можливість вільного набору тексту без обмежень;

10) має бути реалізований тренажер друку на клавіатурі, який включає користувальницький інтерфейс з випадковим відображенням клавіш для тренування друку по кожній клавіші, а також відображенням статистики;

11) повинна бути реалізована можливість зміну дизайну інтерфейсу веб-застосунка в залежності від вподобань користувача;

12) збереження важливих налаштувань, що дозволятиме користувачам зберігати свої вподобання навіть після оновлення сторінки;

13) має бути реалізований «Focus Mode», який буде приховуватиме меню налаштувань і залишатиме лише таймер та статистику;

14) веб-застосунок повинен надавати звукові ефекти при наборі тексту, з можливістю вибору стандартного звуку клавіші або додаткових звуків клавіатури.

Нефункціональні вимоги проектованої системи:

1) застосунок повинен бути адаптивним і здатним пристосовуватися до різних розмірів екранів, забезпечуючи безперешкодний інтерфейс користувача на пристроях, таких як настільні комп’ютери, ноутбуки, планшети та мобільні телефони;

2) користувацький інтерфейс повинен бути простим, зрозумілим та привабливим візуально, сприяючи легкій навігації та зручної взаємодії між елементами інтерфейсу.

Діаграма, що зображена на рисунку 2.1 надає всебічне візуальне представлення веб-застосунку для вивчення сліпого набору тексту на клавіатурі, показуючи різні класи та їх взаємозв’язки, демонструючи моделі елементів у нотації UML. Вона надає загальний огляд того, як різні функції та компоненти в застосунку повинні взаємодіяти між собою.

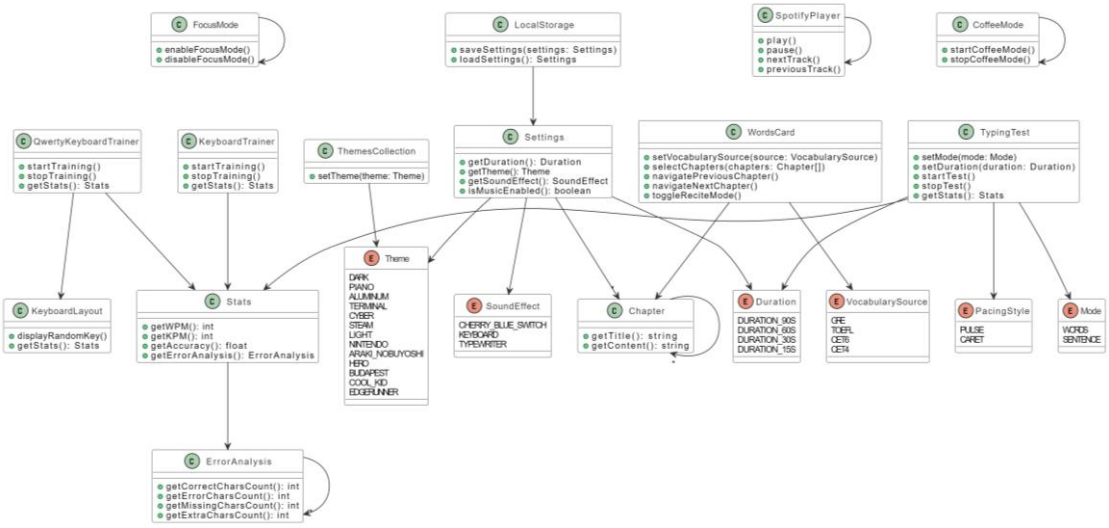

Рисунок 2.1 – Діаграма моделей елементів розроблюваного веб-застосунку

Варто відзначити, що наше початкове структурне моделювання використовує UML діаграму для представлення веб-застосунка, який ми

розробляємо. Однак, важливо розуміти, що ця діаграма є лише початковим кроком і не охоплює всі можливі компоненти або архітектурні рішення. Перед початком роботи над програмним забезпеченням неможливо передбачити всі деталі і вимоги, тому діаграма буде змінюватися протягом розробки. Саме тому, розробка UML діаграми перед початком роботи над ПЗ завжди здійснюється відносно поверхнево. Звідси важливо зрозуміти, що така схема не є фінальною, і може, а скоріше, навіть, буде неодноразово змінюватися в процесі розробки.

Діаграма комунікації (рис. 2.2) демонструє способи взаємодії та обміну інформацією між різними компонентами системи. У випадку розроблюваного веб-застосунку вона демонструє, як користувач взаємодіє з його компонентами.

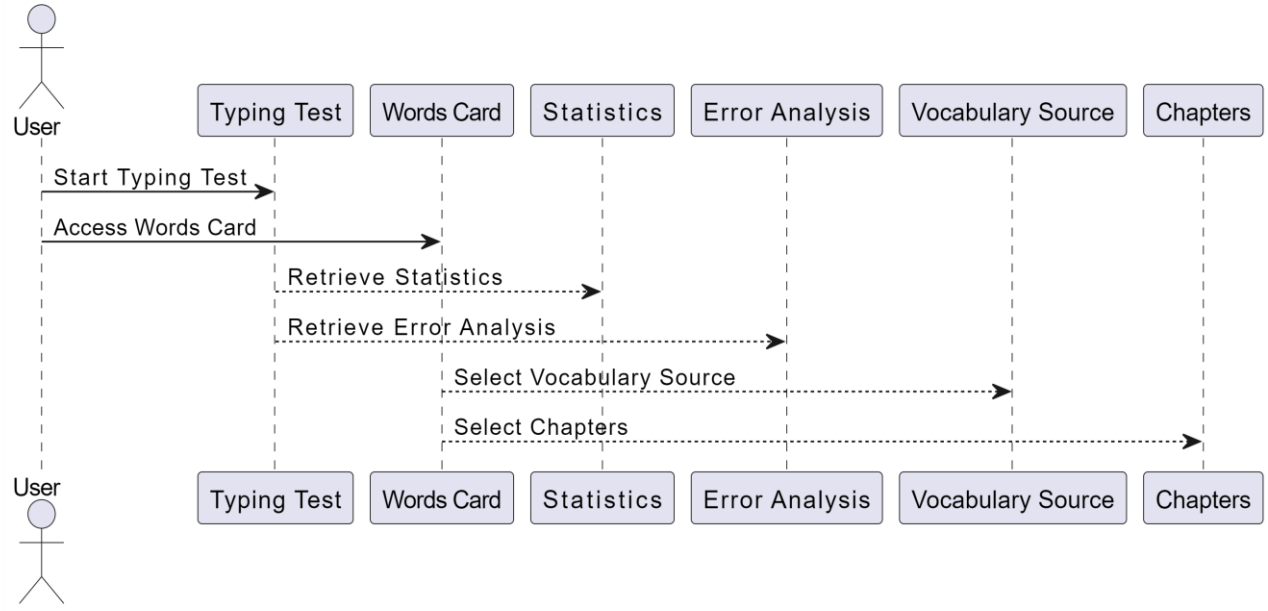

Рисунок 2.2 – Діаграма комунікацій розроблюваного веб-застосунку

Завершивши фазу моделювання нашого веб-додатка, час перейти до наступного важливого етапу, а саме проектування інтерфейсу користувача – UI та дизайну взаємодії з користувачем – UX. На цьому етапі ми зосередимося на створенні візуально привабливого та інтуїтивно зрозумілого дизайну, що буде відповідатиме вимогам користувачів веб-застосунків та сервісів.

Дизайн UI передбачає розробку візуальних елементів веб-застосунка, включаючи розташування, типографіку, різні колірні схеми та інтерактивні компоненти. Крім того, дизайн UX буде спрямований на поліпшення загального досвіду користувача шляхом оптимізації його зручності та доступності.

Спочатку ми зайнялися розробкою макетів для головної сторінки веб-застосунку та створенням загальних компонентів, які використовуватимуться у всьому веб-застосунку.

Ці етапи потребували більше уваги, оскільки вигляд, функціональність і узгодженість загальних компонентів є одними з головних пріоритетів у розробці цього веб-застосунку.

На рисунку 2.3 зображено головний інтерфейс нашого веб-застосунку, який є місцем для доступу до різних функцій та можливостей для користувачів. Макет складається з декількох окремих секцій, кожна з яких відповідає за різні функції.

У верхній частині сторінки буде знаходитися header, де буде розташовуватиметься назва та логотип веб-застосунку. Це дозволять користувачам швидко ідентифікувати та пов’язувати візуальний брендинг зі використовуваним застосунком.

У центрі сторінки буде розташована область контенту, яка динамічно підлаштовуватиметься відповідно до обраного режиму друку. Залежно від вибраного режиму, користувачі зустрічатимуть список тексту, готового до друку. Разом з текстом відображатимуться статистичні дані у реальному часі, такі як швидкість і точність, а також таймер.

Ці елементи надаватимуть користувачам цінну інформацію про їх продуктивність та дозволяють контролювати їхні навички сліпого набору тексту на клавіатурі. Також відображатиметься мова тексту, що можна обрати для друку тексту. На вибір користувачам буде надаватися можливість обирати між англійською та українською мовами.

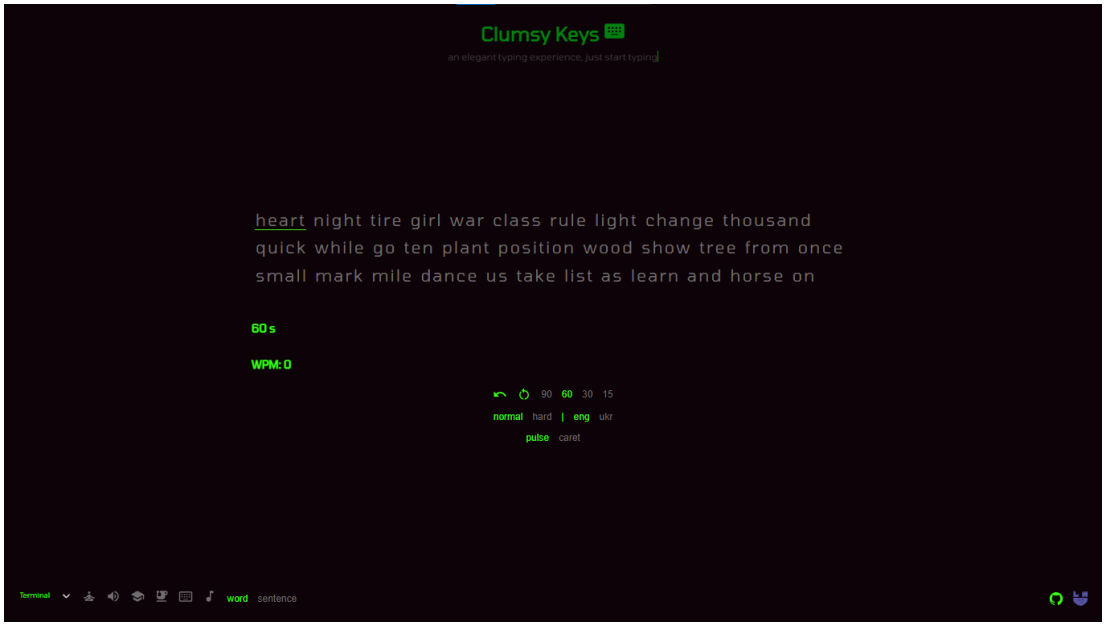

Рисунок 2.3 – Головна сторінка веб-застосунку

У нижній частині сторінки знаходитиметься футер, де буде розміщуватися меню опцій. Тут користувач зможе вимикати звук натисканням клавіш або ж його увімкнути, змінювати інтерфейс веб-застосунку відповідного його уподобанням. Також це меню надаватиме користувачам можливість легко перемикатися між різними режимами застосунку. Завдяки цій гнучкості, користувачі зможуть легко переходити до альтернативних розділів, таких як режими: «Words Card Mode», «Qwerty Keyword», «Focus Mode». Що задовольнятимуть їхні конкретні потреби та вподобання. Варто відзначити, що хедер та футер є незмінними при будь якому обраному активному режимі.

На рисунку 2.4 відображено сторінку з запроектованим дизайном режимом «Qwerty Keyword», що є одним із режимів розроблюваного веб-застосунку, який допомагатиме користувачам покращити свої навички друку на клавіатурі сліпим методом набору.

У ньому використовується інтерфейс з розкладкою клавіатури qwerty, де випадковим чином відображатимуться клавіші, що дозволятиме користувачам практикувати друк сліпим методом набору. У реальному часі надаватимуться статистичні дані, такі як швидкість та точність, щоб відстежувати прогрес користувачів.

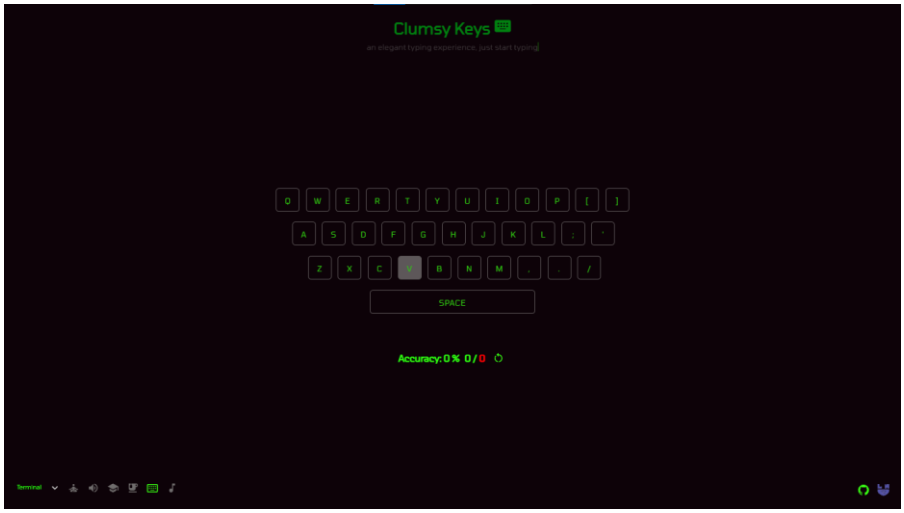

Рисунок 2.4 – Сторінка з активним режимом «Qwerty Keyword»

Режим «Words Card Mode» у нашому веб-застосунку поліпшуватиме словниковий запас і навички набору сліпого набору тексту на клавіатурі (рис. 2.5). Користувачі зможуть вибирати джерела словників, обирати розділи, переміщуватись між різними словами. Його основа мета у тому, що він змушуватиме користувачів згадати слово на основі заданого контексту у форматі аудіо повідомлення, зміцнюючи їх навички сліпого набору тексту на клавіатурі друку розвиваючи м’язову пам’ять.

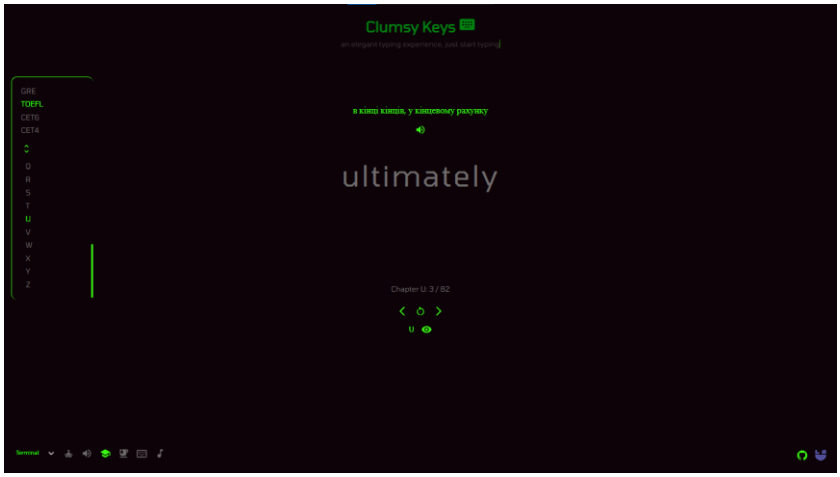

Рисунок 2.5 – Сторінка з активним режимом «Words Card Mode»

Навігаційний інтерфейс цього режиму представляє зручну навігацію, яка дозволятиме користувачам переглядати слова із словників та друкувати їх

2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання

У сучасій веб-розробці існують два основні підходи до створення веб-застосунків це – односторінкові застосунки (SPA) та багатосторінкові застосунки (MPA). Ці підходи відрізняються у способі структурування та представлення контенту застосунків користувачам. Для прийняття обґрунтованих рішень щодо вибору підходу варто розуміти відмінності між SPA та MPA і враховувати вимоги конкретно нашого веб-застосунка.

Односторінковий застосунок (SPA) – це архітектурний шаблон, який ставить на перший план безперервний і десктопний досвід користувача на одній веб-сторінці [11]. У SPA, початковий HTML, CSS та JavaScript завантажуються лише один раз, а подальші оновлення динамічно отримуються з сервера та рендеряться на стороні клієнта без повного перезавантаження сторінки. Цей підхід сприяє високій швидкості роботи застосунку, оскільки користувачі можуть без зусиль переміщатися між різними розділами застосунку, не зазнаючи перезавантаження сторінок. Популярні фреймворки SPA, такі як React, Angular та Vue.js, часто використовуються для розробки SPA. Навпаки, ж багатосторінковий застосунок (MPA) відповідає традиційному підходу до веб-розробки, де кожна сторінка застосунка складається з окремого HTML-документа [11]. У MPA, перехід до різних розділів або функцій зазвичай включає повне перезавантаження сторінки. Хоча MPA може не надавати такого ж рівня швидкості роботи, як SPA, вони мають свої переваги у певних випадках, особливо коли важливо оптимізувати пошукову оптимізацію.

На основі існуючих знань у розробці веб-застосунків та вимогам до проектованої системи було прийнято рішення використовувати підхід SPA при розробці даного застосунку. Для реалізації цього підходу було вибрано бібліотеку React як основний інструмент для розробки додатку, разом із мовою програмування JavaScript. React славиться своєю ефективністю у побудові динамічних та інтерактивних користувацьких інтерфейсів. Додатково, буде

використана популярна бібліотека MaterialUI, яка містить готові компоненти React та набір стилів, що значно полегшує та економить час під час розробки [12].

У розробці веб-додатків важливим аспектом є зберігання даних. Для розроблюваного веб-застосунку було обрано формат зберігання даних JSON (JavaScript Object Notation). JSON має кілька переваг, що робить його відповідним варіантом для зберігання даних у розроблюваному веб-застосунку.

JSON – представляє собою стандартний текстовий формат, призначений для структуруванню даних на основі синтаксису об’єктів JavaScript [13].

Оскільки розроблюваний веб-застосунок буде побудований з використанням JavaScript та бібліотеки React, які нативно підтримують JSON, він ідеально підходить для зберігання даних. За допомогою JSON ми можемо легко представити складні структури даних та ієрархії, що дозволятиме зберігати різні типи інформації, що стосуються розроблюваного веб-застосунку, а саме: статистику, налаштування, та базу даних слів. Загалом, гнучкість JSON дозволятиме ефективно організовувати та керувати даними, забезпечуючи безперешкодне отримання та зміну даних.

В таблиці 2.1 наведено набір інструментів, які використовуватимуться під час розробки веб-застосунку.

Варто зазначити, що це лише частина задіяних інструментів, які будуть задіяні під час розробки.

Таблиця 2.1 – Список інструментів задіяних під час розробки

№ п\п	Інструмент	Застосування	Переваги
1	2	3	4
1	Visual Studio Code	Інтегроване середовище розробки (IDE) для веб-розробки,що використовується при розробці як фронтенду так і бекенду.	Легке у налаштовуванні з широкою підтримкою мов програмування та широкою підтримкою розширень, розширені можливості, такі як автодоповнення коду, налагодження та інтеграція з системами контролю версій.

Продовження таблиці 2.1

1	2	3	4
2	Figma	Інструмент для створення дизайну та веб-інтерфейсів при проектуванню веб-застосунків.	Інтуїтивний інтерфейс та потужні засоби для створення інтерактивних та адаптивних інтерфейсів
3	JavaScript	Мова програмування для веб-розробк	Універсальна та широко підтримувана мова для розробки як на фронтенді, так і на бекенді.
4	JSON	Формат збереження даних	Легкий та читабельний текстовий формат для обміну даними, з легким представлення складних структур даних.
5	React	Бібліотека мови JavaScript, яка використовується для побудови інтерфейсів	ReactJS базується на компонентах, де кожен компонент унікальний у своїх правилах та управлінні, їх можна використовувати повторно, що значно зменшує час витрачений на розробку.
6	MeterialUI	Бібліотека компонентів React для дизайну інтерфейсів	Відрита компонента бібліотека, яка використовує Google Material Design, що включає в собі широкий набір готових компонентів, готових до використання, дозволяючи легко впровадити власне систему дизайну, побудовану на основі цих компонентів.

Висновки до розділу 2

У даному розділі було обґрунтувано вибір конкретного підходу до моделювання програмної системи. Для досягнення цієї мети були застосовані методи виявлення та аналізу вимог, визначення функціональних та нефункціональних вимог. З використанням нотації UML були створені моделі елементів програмного застосунку. Крім того, був проведений UX/UI-дизайн та описані схеми інтерфейсу з функціональними можливостями. Додатково, був проведений огляд різноманітних інструментів для розробки веб-застосунків та надано детальний опис обраних інструментів з їх функціональністю.

РОЗДІЛ 3
РОЗРОБКА ВЕБ-ЗАСТОСУНКА ДЛЯ ВИВЧЕННЯ СЛІПОГО НАБОРУ ТЕКСТУ НА КЛАВІАТУРІ

3.1 Практична реалізація об’єкта проектування

Розробка об’єкта проектування розпочинається з налаштування середовища розробки, а саме Visual Studio Code, нам потрібно налаштувати початкові файли, які будуть використовуватися під час подальшої розробки. У файлі package.json ми прописуємо усі залежності та пакети, що потрібно завантажити для нашого об’єкту розробки, код якого наведений у лістингу 3.1.

Лістинг 3.1 – Код файлу package.json

```
{
  "name": "clumsy-keys",
  "version": "0.3.4",
  "private": true,
  "dependencies": {
    "@emotion/react": "^11.9.0",
    "@emotion/styled": "^11.8.1",
    "@mui/icons-material": "^5.6.1",
    "@mui/material": "^5.6.1",
    "@testing-library/jest-dom": "^5.16.4",
    "@testing-library/react": "^13.0.1",
    "@testing-library/user-event": "^13.5.0",
    "random-words": "^1.1.2",
    "react": "^18.0.0",
    "react-dom": "^18.0.0",
    "react-scripts": "5.0.1",
    "react-select": "^5.3.0",
    "styled-components": "^5.3.5",
    "use-sound": "^4.0.1",
    "web-vitals": "^2.1.4"
  },
```

Кінець лістингу 3.1

Після цього у терміналі ми вводимо команду npm install, в результаті чого менеджер пакетів розпочне встановлювати та завантажувати усі залежності.

Після завершення процесу встановлення залежностей, що зазначені у файлі package.json, вони будуть розміщені в папці під назвою nodemodules у каталозі нашої роботи. Ця папка вміщуватиме усі необхідні бібліотеки, фреймворки та модулі, що необхідні для розробки.

Продовжуючи розробку, ми створемо файл app.js який буде відповідати за основну структуру нашого веб-застосунку, керуватиме різними налаштуваннями та режимами, рендерити відповідні компоненти. Тепер перейдемо до його кодування. На лістингу 3.2 ми імпортуємо встановлені файли залежностей, а також деякі з компонентів розроблюваного веб-застосунку, що дозволить отримати доступ до функціональності, що були встановлені та зв’язати інші файли JavaScript нашого застосунку, давайте їх і оглянемо.

Лістинг 3.2 – Імпортування пакетів та компонентів у файлі app.js

```
{
  import React, { useState, useRef, useEffect } from "react";
  import { ThemeProvider } from "styled-components";
  import { defaultTheme, themesOptions } from "./style/theme";
  import { GlobalStyles } from "./style/global";
  import TypeBox from "./components/features/TypeBox/TypeBox";
  import SentenceBox from "./components/features/SentenceBox/SentenceBox";
  import Logo from "./components/common/Logo";
  import FooterMenu from "./components/common/FooterMenu";
  import FreeTypingBox from "./components/features/FreeTypingBox";
 },
```

Кінець лістингу 3.2

У компоненті FooterMenu.js ми генеруємо футер нашого веб-застосунку, спочатку, імпортуємо певні компоненти з бібліотеки Material-UI, а саме: Grid, Box, Tooltip, Iconbutton та кілька іконок. Далі ми імпортуємо константи, із файлу constant.js, код якого наведено у лістингу А.1, що стосуються нашого футера. Вставляємо в div елемент контейнер Grid компонента, який буде створювати адаптивний макет для кожного контейнера, завдяки вбудованим

CSS класам бібліотеки. Далі у компонент IconButton, що слугує тут кнопкою, вставляється onClick функція обробник подій, яка буде при натисканні на неї визивати функцію toggleFocusedMode. Ця функція, відповідає за активізацію режиму «Focus Mode» у розроблюваному застосунку. Компонент Tooltip відображатиме значення заданої змінної, що буде працювати як підказка і виводити назву цього режиму при наведенні курсору на кнопку. Тепер за аналогічним методом, який був описаний та код якого наведено у лістингу 3.3, ми створюємо інші кнопки, що будуть відповідати за перехід на інші доступні режими у розроблюваному веб-застосунку, таким чином створивши меню налаштувань.

Лістинг 3.3 – Фрагмент коду компонента FooterMenu.js

```
{
  <IconButton onClick={toggleFocusedMode}>
    <Tooltip title={FOCUS_MODE}>
      <span className={getModeButtonClassName(isFocusedMode)}>
                                                    <SelfImprovementIcon
fontSize="medium"></SelfImprovementIcon>
      </span>
    </Tooltip>
  </IconButton>
},
```

Кінець лістингу 3.3

Тепер перейдемо до реалізації основного функціоналу нашого веб-застосунку, реалізацію компонента, який буде відповідати за режим який стоятиме по замовчуванню при запуску веб-застосунку, де буде виводитися речення, що будуть готові до друку. Спочатку створимо компонент SentenceBox.js. Так на лістингу 3.4 реалізовується функціональність для обробки стану натискання клавіш Tab та Enter для перезапуску контейнера, де розташовується текст, який потрібно надрукувати. Стан openRestart використовується для контролю видимості діалогового вікна перезапуска. Він спочатку встановлюється на false. Функція EnterkeyPressReset є обробником

подій, що перевірятиме натискання певних клавіш (Enter, Tab та Space) і виконуватиме відповідні дії, якщо, наприклад, буде натиснута клавіша Enter або Tab вона викликатиме функцію reset для його перезапуску.

Лістинг 3.4 – Фрагмент коду компонента SentenceBox.js

```
{// tab-enter restart dialog
  const [openRestart, setOpenRestart] = useState(false);
  const EnterkeyPressReset = (e) => {
    // press enter/or tab to reset;
    if (e.keyCode === 13 || e.keyCode === 9) {
      e.preventDefault();
      setOpenRestart(false);
      reset(sentencesCountConstant, language, false);
    } else if (e.keyCode === 32) {
      e.preventDefault();
      setOpenRestart(false);
      reset(sentencesCountConstant, language, true);
    } else {
      e.preventDefault();
      setOpenRestart(false);
    }
  };
  };
  const handleTabKeyOpen = () => {
    setOpenRestart(true);
  };
  const getSentencesCountButtonClassName = (buttonSentencesCountConstant)

    if (buttonSentencesCountConstant === sentencesCountConstant) {
      return "active-button";
    }
    return "inactive-button";
  };
}
  const getLanguageButtonClassName = (buttonLanguage) => {
    if (language === buttonLanguage) {
      return "active-button";
    }
    return "inactive-button";
  };
```

Кінець лістингу 3.4

Далі функція буде визначати скільки речень буде в рамках одного тесту, спираючись на те, яка кількість була обрана користувачем в рамках одного тесту.

Тепер реалізуємо функціонал секундоміра в рамках одного тесту. У лістингу 3.5 реалізується функціональність секундоміра, використовуючи хуки useState та useEffect в React.

Лістинг 3.5 – Реалізація секундоміра при кожному тесті

```
{
const [timeRunning, setTimeRunning] = useState(false);
  // stop watch loop
  useEffect(() => {
    let interval;
    if (timeRunning) {
      interval = setInterval(() => {
        setTime((prevTime) => prevTime + 1);
      }, 1000);
    } else if (!timeRunning) {
      clearInterval(interval);
    }
    return () => clearInterval(interval);
  }, [timeRunning]); },
```

Кінець лістингу 3.5

Вони відстежуватимуть пройдений час в секундах та керують станом роботи секундоміра. Для відстеження пройденого часу використовується перемінна time. Функція setTime використовується для оновлення цієї перемінної. Стан timeRunning використовується для визначення, чи працює секундомір чи зупинений. Він спочатку встановлюється на false за допомогою хука useState. Функція setTimeRunning використовується для оновлення цього стану. Основна функціональність секундоміра реалізована у хуку useEffect. Цей хук виконується кожного разу, коли змінюється стан timeRunning. Він налаштовує цикл за допомогою setInterval, коли секундомір працює, і очищає цей інтервал за допомогою clearInterval, коли секундомір зупинено. Усередині хука useEffect, якщо timeRunning дорівнює true, створюється інтервал за

допомогою setInterval. Цей інтервал викликає функцію зворотного виклику кожну секунду, збільшуючи стан часу на 1 кожної секунди за допомогою функції оновлення. Якщо timeRunning дорівнює false, що вказує на зупинку секундоміра, інтервал очищається за допомогою clearInterval з вказаною змінною interval.

Подальшим кроком є реалізація логіки, що буде відповідати за те, які символи вводить користувач при натисканні клавіш, а також обробку цих введених символів для подальшого використання системою, його реалізацію ми можемо побачити у лістингу 3.6.

Лістинг 3.6 – Реалізація секундоміра при кожному тесті

```
{
    if (currInput.length >= sentences[currSentenceIndex].length) {
      checkAndUpdateStats(currSentence, currInput);
      if (currSentenceIndex + 1 === sentencesCountConstant) {
        setStatus("finished");
        setTimeRunning(false);
        return;
      }
      setCurrSentenceIndex(currSentenceIndex + 1);
      setCurrInput("");
      sentenceInputRef.current.value = "";
      return;
    }
    return;
  }
 };
 const getCharClassName = (idx, char) => {
  if (idx < currInput.length) {
    if (currInput[idx] === char) {
      return "correct-sentence-char";
    }
    if (char === " ") {
      return "error-sentence-space-char";
    }
    return "error-sentence-char";
  } return "sentence-char";
 }; },
```

Кінець лістингу 3.6

Функція handleKeyPress визначається для обробки події натискання клавіши, вона перевіряє чи довжина поточного вводу currInput дорівнює або більша за довжину поточного речення. Якщо умова виконується, викликається функція checkAndUpdateStats, щоб порівняти поточне речення з введеним користувачем і оновити статистику відповідно. Далі перевіряється, чи поточне речення є останнім шляхом порівняння, якщо вони рівні, це означає, що користувач завершив всі речення у тесті.

Тепер потрібно реалізувати функціонал генерації речень відповідно того, який рівень складності тесту був вибраний. Функція sentencesGenerator генерує список речень на основі заданих параметрів, як кількість речень у межах одного тесту та мови тексту для вводу. Якщо було обрано англійські речення, функція генерує англомовні речення. Створюється пустий масив зі списком англомовних речень, після цього виконується цикл, який повторюється ту кількість разів, скільки було обрано речень для вводу в рамках одного тесту і випадковим чином вибирається речення з масиву, де розміщений список із реченнями з допомогою функції randomIntFromRange. Для українських речень процес аналогічний. Процес кодування продемонстровано у лістингу 3.7.

Лістинг 3.7 – Генерація випадкових речень

```
{
    if (currInput.length >= sentences[currSentenceIndex].length) {
      checkAndUpdateStats(currSentence, currInput);
      if (currSentenceIndex + 1 === sentencesCountConstant) {
        setStatus("finished");
        setTimeRunning(false);
        return;

      }
      setCurrSentenceIndex(currSentenceIndex + 1);
      setCurrInput("");
      sentenceInputRef.current.value = "";
      return;
}
}
```

Кінець лістингу 3.7

Поки що було описано процес кодування функціоналу режиму, при якому виводяться для друку лише речення. Тепер ми будемо реалізовувати фунціонал для одного із зпроектованих режимів, а точніше «Words Card Mode».

Спочатку створюється компонент WordsCards.js де буде прописана уся логіка стосовно цього режиму. Тут ми отримуємо дані з локального сховища за допомогою window.localStorage.getItem з ключем DictionaryCatalog. Якщо значення існує, воно розпарсюється, із нього вилучається набір словника. Якщо значення не знайдено в локальному сховищі, змінна стану alphabetSet ініціалізується порожнім набором, процес кодування зображено на лістингу 3.8.

Лістинг 3.8 – Фрагмент коду компонента WordsCards.js

```
{
 const [alphabetSet, setAlphabetSet] = useState(() => {
   const stickyAlphabetSet
= window.localStorage.getItem("DictionaryCatalog");
   if (stickyAlphabetSet !== null) {
     const localAlphabetSet = JSON.parse(stickyAlphabetSet);
     const localAlphabetSetList = Object.values(localAlphabetSet)[0];
     return new Set(localAlphabetSetList);
   }
   return new Set();
  });

  const [currInput, setCurrInput] = useState("");

  const [vocabSource, setVocabSource] = useState(() => {
   const stickyAlphabetSe
= window.localStorage.getItem("DictionaryCatalog");
   if (stickyAlphabetSet !== null) {
     const localAlphabetSet = JSON.parse(stickyAlphabetSet);
     const localAlphabetSetKey = Object.keys(localAlphabetSet)[0];
     return localAlphabetSetKey;
   }
}
}
   return "GRE";
  });
```

Кінець лістингу 3.8

Для того, щоб розрахувати такі показники як швидкість набору тексту на клавіатурі та точність введеного тексту, було розроблено алгоритм, кодування якого продемонстровано на лістингу 3.9 де перший рядок обчислює значення WPM, ділячи кількість натиснутих клавіш на час у хвилинах та множачи його на 60 для перетворення на годину на швидкість. Потім результат ділиться на 5, щоб вирахувати середню довжину слова.

Для вирахування відсотка точності, ми ділимо кількість введених правильних символів на загальну кількість введених, що складається з суми правильних, неправильних та додаткових символів.

Потім результат помножається на 100, щоб отримати суму у відсотках, Функція Math.round() використовується для округлення результату до найближчого цілого числа.

Лістинг 3.9 – Розрахунок швидкості та точності набраного тексту

```
{ const wpm = time < 1 ? 0 : ((rawKeyStroke / time) * 60) / 5; WPM: {Math.round(wpm)}
        {status === "finished" && (
          Accuracy:{" "}
          {Math.round(
            (stats.correct /
              (stats.correct + stats.incorrect + stats.extra)) *
              100
}
```

Кінець лістингу 3.9

В коді лістингу 3.10 було реалізовано функціонал, який залежно від обраного слова зі словника в полі для введення тексту передає його ідентифікатор зі словника, щоб згенерувати URL для отримання аудіо з вимовою поточного слова, що є активним і який повинен бути виведений для користувача за допомогою API словника.

Це дозволяє автоматично генерувати URL, щоб отримати файл аудіо який потрібно активувати вимовою для поточного слова, що повинен ввести користувач.

Лістинг 3.10 – Фрагмент коду компонента WordsCards.js

```
{
  const currWord = wordsDict[index].key;    }
  const extra = currInput.slice(currWord.length, currInput.length).split("");  });
if (currInput.length >= currWord.length) {
          e.preventDefault()
        }
const audioSource = ‘https://dict.youdao.com/dictvoice?audio=‘ + currWord +
‘&type=2’;
}
```

Кінець лістингу 3.10

Для того аби отримати базу даних слів ми імпортуємо файли JSON, що містять дані усіх слів, що будуть використовуватися під час виводу тексту, що потрібно буде друкувати. Потім вміст цих файлів призначається відповідним константам, після операторів імпортування ми призначаємо імпортовані дані JSON іншим константам і наприкінці ці константи експортуються за допомогою оператора export, що дозволяє використовувати їх у інших модулях, кодування наведено у лістингу 3.11.

Лістинг 3.11 – Отримання даних з JSON файлу

```
{
import EnglishMostFrequentWords
from ‘../assets/Vocab/EnglishMostFrequentWords.json’;
}
const COMMON_WORDS = EnglishMostFrequentWords;
export {
   COMMON_WORDS
}
```

Кінець лістингу 3.11

База усіх слів та речень, розташовуються у файлі JSON, вони розміщені у форматі колекції пар ключ та значення, де кожне число це ідентифікатор, а значення val, вміщує відповідно текст типу даних string.

Тепер перейдемо до реалізації режиму «Qwerty Keyword». Спочатку ми ініціалізуємо компонент під назвою defaulKeyboard, він буде представляти

інтерфейс клавіатури та обробляти введення користувача. Спочатку компонент ініціалізує деякі змінні стану за допомогою useState. InputChar зберігатиме поточний символ введення, correctCount відстежує кількість правильних введень, відповідно incorrectCount відстежуватиме кількість неправильних введень. Використовуючи useSound зі звукової бібліотеки, ми будемо відтворювати звук на основі перемінної soundType, який в свою чергу визначається в опціях налаштувань, звук ж буде відтворюватися при натисканні клавіш. UseEffect використовується для реалізації фокусування та підсвічення кожного разу коли виводиться клавіша, яку потрібно натискати. Функція handleKeyDown спрацьовує, коли натискається клавіша. Вона відтворює звук, якщо ввімкнено режим звуку soundMode. Функція handleKeyUp спрацьоватиме, коли клавішу відпускатимуть. Вона порівнюватиме відпущену клавішу, що визначається тут як event.key, з поточним станом randomKey, це клавіша, що повинну була бути натиснута. Якщо вони співпадають, оновлюється стан randomKey новим випадковим значенням, збільшується значення правильного вводу correctCount. В іншому випадку збільшується значення incorrectCount. Реалізація верстки даного компоненту продемонстрована у лістингу Б.2. в файлі DefaultKeyboard.js.

Для того, щоб зберігати та отримувати значення, які користувач зробив під час користування веб-застосунком у локальному сховищі браузера, створимо власний хук та надамо йому назву useLocalPersistState. Спочатку імпортуємо useState та useEffect, після цього ініціалізуємо функцію useLocalPersistState, яка буде приймати key та DefaultValue параметри. DefaultValue представлятиме початкове значення для стану, а key є унікальним ідентифікатором, який використовується для зберігання стану в локальному сховищі. Передаючи функцію як початкове значення, ця функція перевіряє, чи є значення, збережене в локальному сховищі, відповідне заданому key, за допомогою window.localStorage.getItem(key). Якщо значення знайдено, воно розпарсюється з його рядкового представлення у форматі JSON за допомогою JSON.parse встановлюється як початкове значення стану. Якщо значення не

знайдено в локальному сховищі, використовується параметр defaultValue як початкове значення. Використовуючи цей власний хук, ми зможемо легко створювати змінні стану, які зберігатимуться навіть при перезавантаженні сторінки або закритті та відкритті браузера. Значення стану автоматично зберігатимуся в локальному сховищі і отримуватимуться за потребою, надаючи зручний спосіб зберігання та отримання даних локально у компонентах React, лістинг даного компоненту продемонстровано в додатку 2.

Для того аби розробити можливість зміни дизайну теми нашого веб-застосунку, ми створимо окремий файл, що буде зберігати усі стилі відповідної теми, та відповідно тому, яку тему поставить користувач в опціях налаштувань. Лістинг даної реалізації наведено в лістингу 3.12. Створивши об’єкти, вони будуть зберігати опцію, що будуть відповідати за кожну із тем. Вони мають значення, що відповідають за різні частини дизайну користувацького інтерфейсу, експортувавши їх, ми зможемо використовувати їх у будь-якому місці нашого об’єкту розробки.

Лістинг 3.12 – Фрагмент коду файлу theme.js

```
{
const lightTheme = {
  label: "Light",
  background: "#F5F5F5",
  text: "#000000",
  gradient: "linear-gradient(315deg, #74ebd5 0%, #ACB6E5 94%)",
};
const darkTheme = {
  label: "Dark",
  background: "#121212",
}; const defaultTheme = darkTheme;
const themesOptions = [
  { value: darkTheme, label: "Dark" },
  { value: lightTheme, label: "Light" }
];
export {
  lightTheme,
  darkTheme}}
```

Кінець лістингу 3.12

3.2 Тестування та налагодження інформаційно-комп’ютерної системи

У даному розділі ми буде проводитися тестування проектованої системи нашого веб-застосунку. Будуть протестовані функціональні елементи веб-застосунку:

– перевірка правильного відгуку інтерфейсу клавіатури на введення користувача;

– перевірка правильності відтворення звукових ефектів при натисканні клавіш;

– перевірка правильності розрахунків вимірюванні швидкості та точності друку.

Спочатку перевіримо відгук інтерфейсу і те система реагує на натиск різних клавіш (рис. 3.1).

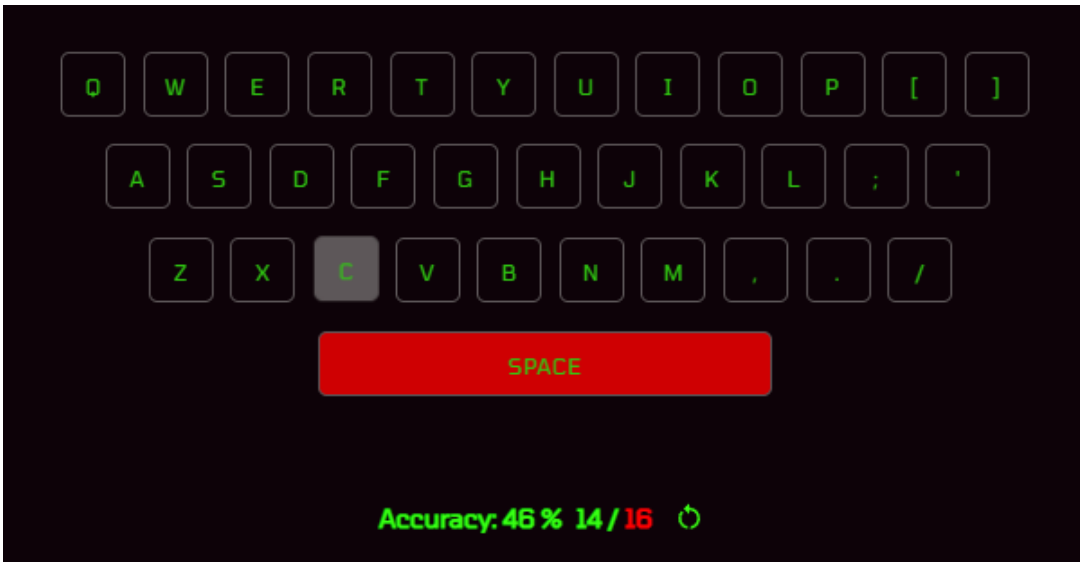

Рисунок 3.1 – Тестування режиму «Qwerty Keyword»

Тут ми можемо побачити, що при натискання не тієї клавіши, яка підсвідчується сірим кольором, клавіша стає червоного кольору, тобто вона коректно реагує на не правильне натискання користувачем клавіш.

Наступним кроком була перевірка відтворення звуку при натисканні клавіш (рис. 3.2).

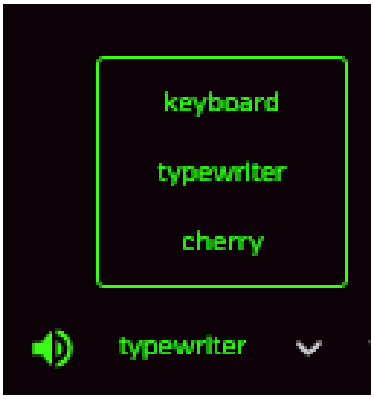

Рисунок 3.2 – Тестування звукових ефектів

Було виявлено, шо при наведенні курсом та кліком на іконку з мікшером звуку відкривається меню вибору типу звуку, що буде відтворюватися під час натиску клавіш, під час тестування було виявлено, що усі 3 режими працюють коректно.

Також було проведено заміри, щодо коректності розрахунків швидкості та точності друку. Порівнявши це з іншими аналогічними сервісами, відхилень у розрахунках та в результаті отриманих даних однієї і тієї ж людини з однаковими навиками друку сліпим методом набору відхилень не було виявлено (рис. 3.3).

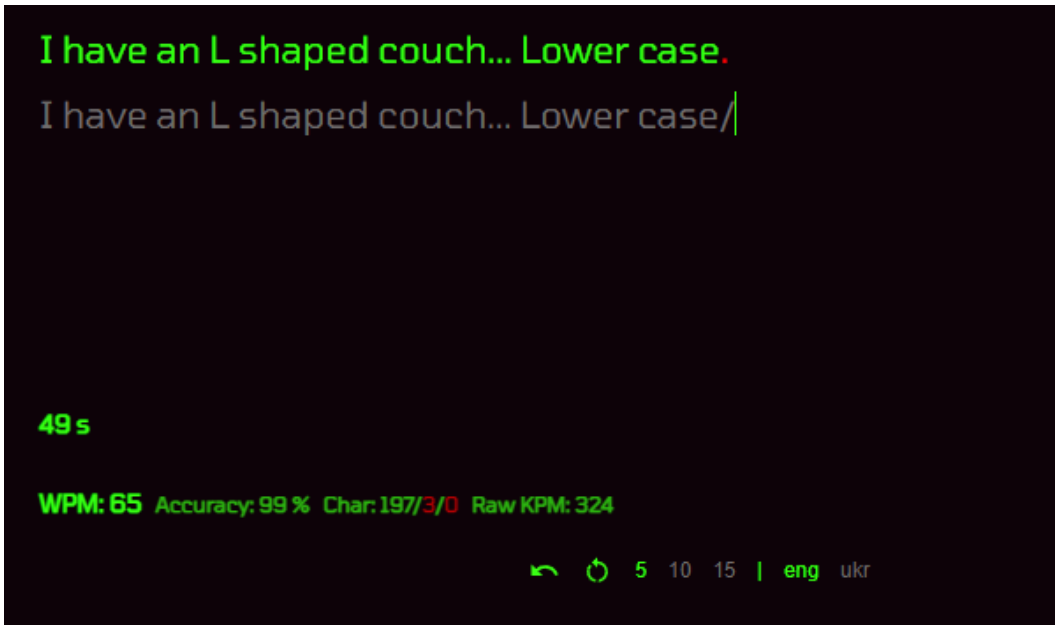

Рисунок 3.3 – Тестування правильності розрахунків

3.3 Розгортання веб-застосунку на хостингову платформу

Створивши власний скрипт, який буде відповідати за процес розгортання веб-застосунку та створення готової збірки об’єкту розробки. Це скрипт, визначений у файлі package.json веб-застосунку, який виконує послідовні кроки збірки та збирає згенеровані файли в одну цільову папку.

Виконавши команду npm run deploy, це запустить скрипт розгортання, визначений у файлі package.json. Першим кроком є збірка React-додатку за допомогою команди npm run build. Ця команда створює готову до використання версію веб-застосунку, об'єднуючи JavaScript, CSS, React та інші файли, що потрібні для правильного функціонування системи, на рисунку 3.4 ми можемо побачити результат роботи, та те, що він готовий до розгортування.

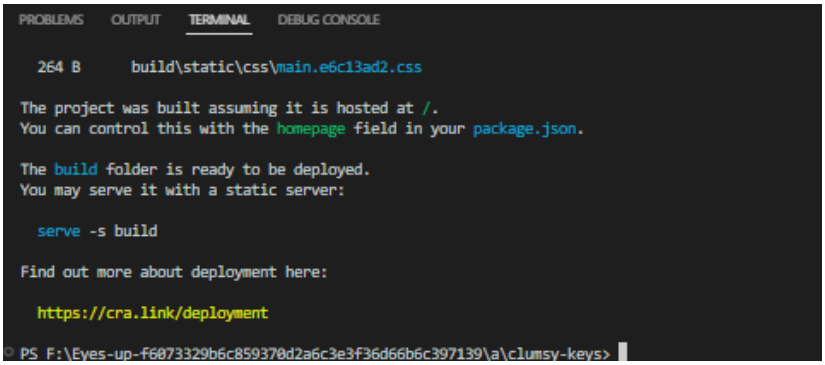

Рисунок 3.4 – Результат роботи запуску скрипту

На рисунку 3.5 ми можемо побачити структуру файлів після збірки.

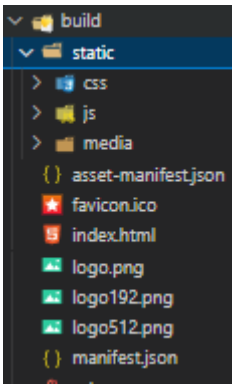

Рисунок 3.5 – Структура файлів веб-застосунку build версії

Сервісом для деплою веб-застосунку було вирішено обрати Netlify. Він є чудовим вибором, оскільки, він забезпечує простоту використання та автоматизацію процесу розгортання. Процес є дуже простим, усе що потрібно це панелі керування Netlify обрати папку, в якому знаходиться наш об’єкт розробки та закинути його на їх сервіс. На рисунку 3.6 ми зможемо побачити успішно розгортку проекту на Netlify.

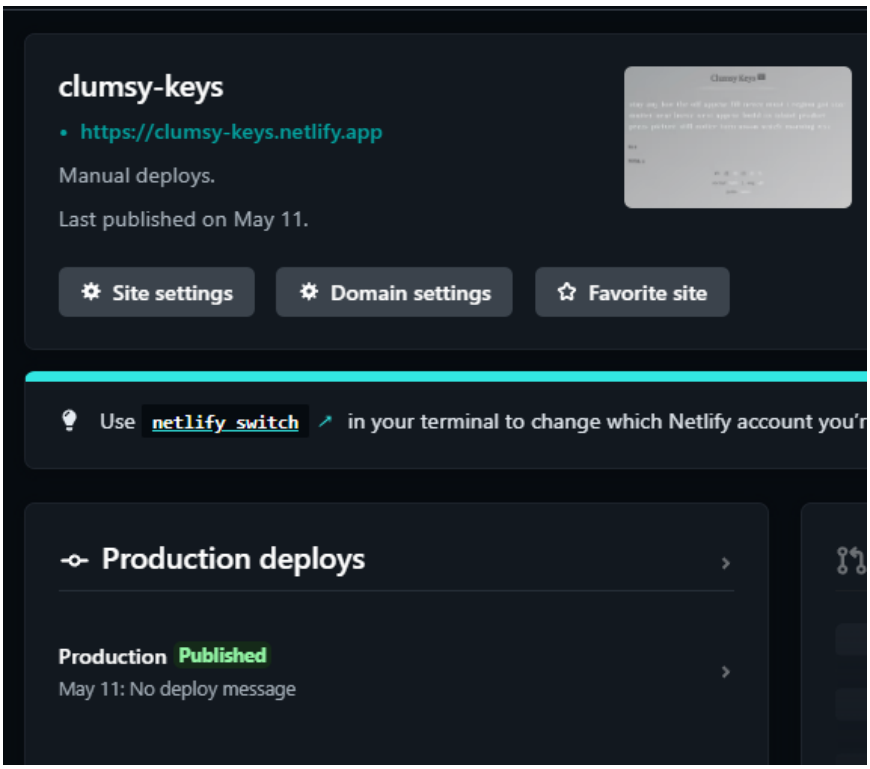

Рисунок 3.6 – Деплой веб-застосунку на Netlify

Висновки до розділу 3

У третьому розділі було здійснено практичну реалізацію об’єкта проектування, було продемонстровано повний процес розробки веб-застосунку для вивчення сліпого набору тексту на клавіатурі. Було здійснено встановлення усіх необхідних компонентів, що необхідні під час розробки веб-застосунків.

Було здійснено тестування та налагодження інформаційно комп’ютерної системи, що включало перевірку коректності роботи функціональних елементів веб-застосунка таких як, відгуку інтерфейсу клавіатури на введення

користувача, відтворення звукових ефектів при натисканні клавіш та коректність розрахунків швидкості та точності під час друку сліпим методом набору на клавіатурі. Після тестування стало зрозуміло, що усі необхідні функції працюють та проблем у роботі веб-застосунка знайдено не було.

Було описано процес розгортання веб-застосунку на хостингову платформу та його підготовку для користування користувачам.

ВИСНОВКИ

В даній роботі було розроблено веб-застосунок для вивчення сліпого набору тексту на клавіатурі. Розроблений веб-застосунок повністю відповідає усім вимогам, що включає в собі аналіз включаючи аналіз проблеми з питань сліпого набору тексту, розробку та проектування користувацького інтерфейсу, а також проведення тестування та налагодження інформаційної системи.

У цій роботі було виконано докладний огляд всіх можливих аналогів, проектованої системи, що могли бути використані при розробці даного програмного забезпечення. Кожен із цих аналогів був ретельно розглянутий, їх переваги та недоліки. Було проведено аналіз вимог для проектованого веб-застосунку, створено план розробки проектованої системи. Результати кваліфікаційної роботи бакалавра повністю підтверджують повноту виконання поставленого завдання.

Аналіз сучасного стану проблеми з питань сліпого набору тексту та розробка веб-застосунку для його вивчення сприяє поліпшенню ефективності друку за допомогою методу сліпого набору тексту.

Для подальшого вдосконалення розроблених досліджень та рішень рекомендується звернути увагу на можливості розширення функціонал веб-застосунку, такі як впровадження системи реєстрації на сайті для користувачів та збереження прогресу при проходженні тестів.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Як підготуватися до повернення в офіс: освоюємо сліпий друк. Work.ua. 2023. URL: https://www.work.ua/articles/jobseeker/2211/ (дата звернення 06.02.2023).

2. How does online typing tool measure a typing speed? Typing Buzz. 2023. URL: https://typingbuzz.com/how-does-online-typing-tool-measure-a-typing-speed/ (дата звернення 10.02.2023).

3. Armiati et al 2018 J. Phys. Conf. Ser. 1114 012105. Journal of Physics. 2018. URL: https://doi.org/10.1088/1742-6596/1114/1/012105 (дата звернення 10.02.2023).

4. Logan, G. D., Ulrich, J. E. D. Different (key)strokes for different folks: How standard and nonstandard typists balance Fitts' law and Hick's law. Journal of Experimental Psychology. 2016. URL: https://doi.org/10.1037/xhp0000272 (дата звернення 10.02.2023).

5. Pinet, S., Zielinski, C., Alario, FX. et al. Typing expertise in a large student population. Cogn. Research. 2022. URL: https://doi.org/10.1186/s41235-022-00424-3 (дата звернення 15.02.2023).

6. Typing. Typing.com. 2023. URL: https://www.typing.com (дата звернення 18.03.2023).

7. 10fastfingers. 10fastfigers.com. 2023. URL: https://10fastfingers.com (дата звернення 19.03.2023).

8. Ratatype. Ratatype.ua. 2023. URL: https://www.ratatype.ua (дата звернення 20.03.2023).

9. What is 'UML'. The Economic Times. 2023. URL: https://economictimes.indiatimes.com/definition/uml (дата звернення 21.03.2023).

10. Why UML Modeling. Visual Paradigm. 2023. URL: https://www.visual-paradigm.com/guide/uml-unified-modeling language/why-uml-modeling/ (дата звернення 23.03.2023).

11. Single page application vs. multi-page application – Which one is the best in 2022. TekRevol Team. 2022. URL: https://www.tekrevol.com/blogs/spa-vs-mpa/ (дата звернення 25.03.2023).

12. Material UI – Overview. Material UI. 2023. URL: https://mui.com/material-ui/getting-started/overview/ (дата звернення 28.03.2023).

13. Working with JSON. Mdn Web Docs. 2023. URL: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON (дата звернення 30.03.2023).

ДОДАТКИ

Додаток А

Лістинг А.1 – Код файлу «Constants.js» мовою програмування JavaScript

```
const DEFAULT_WORDS_COUNT = 200; const DEFAULT_COUNT_DOWN =
COUNT_DOWN_60; const DEFAULT_DIFFICULTY = "normal";

const HARD_DIFFICULTY = "hard";

const RESTART_BUTTON_TOOLTIP_TITLE = "[Tab] + [Enter] to quickly
restart";

const REDO_BUTTON_TOOLTIP_TITLE = "[Tab] + [Space] to quickly redo";

const RESTART_BUTTON_TOOLTIP_TITLE_WORDSCARD = "[Tab] + [Enter]
to quick restart the chapter";

const SELECT_ONE_OR_MORE_CHAPTERS = "Open to select one or more
chapters. Pick the chapters in the typing area."

const RECITE_MODE_TITLE = "hide the word ";

const DEFAULT_DIFFICULTY_TOOLTIP_TITLE =

  "normal mode generates random words from top 1000 most frequently used words
in English dataset.";

const HARD_DIFFICULTY_TOOLTIP_TITLE =

  "hard mode generates random words from blog posts words data, so you
mayencounter longer and less frequently used word.";

const CHAR_TOOLTIP_TITLE =  "correct/incorrect/missing/extra\n extras are
recorded even if deleted.";

const SENTENCE_CHAR_TOOLTIP_TITLE = "correct/incorrect/extra\n";

const ENGLISH_MODE_TOOLTIP_TITLE = "English Mode";

const UKRAINIAN_MODE_TOOLTIP_TITLE = "Ukrainian Mode";

const DEFAULT_DIFFICULTY_TOOLTIP_TITLE_UKRAINIAN =

  "normal mode generates random words from top 5000 most frequently used words
in Ukrainian dataset.";

const HARD_DIFFICULTY_TOOLTIP_TITLE_UKRAINIAN = m  "hard mode
generates random words from top 1500 most used Ukrainian idioms.";

const GITHUB_TOOLTIP_TITLE = "Dear visitors: \n  The working process and the
full feature list of application can be found here \n Thanks! \n";
```

```
const AUTHOR = "author: @VadymTroshchuk\n";

const GITHUB_REPO_LINK = "project: @Github\n";

const LNTU_TOOLTIP_TITLE = "The Blind Typing App is a software application that has been developed as part of a bachelor's degree project. The aim of the app is to assist individuals in improving their typing skills"

const FOCUS_MODE = "Focus mode";

const MUSIC_MODE =  "Spotify player. You will need to login spotify first to use the full feature.";

const FREE_MODE =  "Free typing mode\nType any thing, no pressure, it's coffee time! \n ";

const ENGLISH_MODE = "ENGLISH_MODE";

const UKRAINIAN_MODE = "UKRAINIAN_MODE";

const GAME_MODE = "GAME_MODE";

const GAME_MODE_DEFAULT = "WORD_MODE";

const GAME_MODE_SENTENCE = "SENTENCE_MODE";

const WORD_MODE_LABEL = "word";

const SENTENCE_MODE_LABEL = "sentence";

const TRAINER_MODE = "QWERTY keyboard practice mode";

const DEFAULT_SENTENCES_COUNT = 5;

const TEN_SENTENCES_COUNT = 10;

const FIFTEEN_SENTENCES_COUNT = 15;

const ENGLISH_SENTENCE_MODE_TOOLTIP_TITLE = "English Sentence Mode";

const UKRAINIAN_SENTENCE_MODE_TOOLTIP_TITLE = "Ukrainian Sentence Mode";

const WORDS_CARD_MODE = "Words Card mode, learn something in typing!"

const PACING_CARET = "caret";

const PACING_PULSE = "pulse"
```

Кінець лістингу А.1

Додаток Б

Лістинг Б.1 – Верстка елементу keyword в файлі DefaultKeyboard.js

```
return (
   <div>
    <div className="keyboard">
     <input
      className="hidden-input"
      onBlur={handleInputBlur}
      onKeyDown={handleKeyDown}
      onKeyUp={handleKeyUp}
      ref={keyboardRef}
     ></input>
     <ul className="row row-1">
      <div className={getClassName("q")} id="Q">   Q
      </div>
      <div className={getClassName("w")} id="W">  W
      </div>
      <div className={getClassName("e")} id="E">   E
      </div>
      <div className={getClassName("r")} id="R">  R
      </div>
      <div className={getClassName("t")} id="T">   T
      </div>
      <div className={getClassName("y")} id="Y">    Y
      </div>
      <div className={getClassName("u")} id="U">     U
      <div className={getClassName("i")} id="I">
       I
      </div>
      <div className={getClassName("o")} id="O">  O
      </div>
      <div className={getClassName("p")} id="P">      P
      </div>
     </ul>
     <ul className="row row-2">
      <div className={getClassName("a")} id="A">  A
      </div>
      <div className={getClassName("s")} id="S">  S
      </div>
      <div className={getClassName("d")} id="D">   D
      </div>    </ul>
     <ul className="row row-4">
      <div className={getSpaceKeyClassName()} id="SPACE">
```

```
          SPACE
        </div>
      </ul>{" "}
    </div>
    <div className="keyboard-stats">
      <Box display="flex" flexDirection="row">
        <h3>Accuracy: {accuracy} %</h3>
        <h3>
          <span className="CorrectKeyDowns">{correctCount}</span>
          {"  "} {"/"} {"  "}
          <span className="IncorrectKeyDowns">{incorrectCount}</span>
        </h3>
        <IconButton
          aria-label="restart"
          size="small"
          onClick={() => {
            resetStats();
          }}
        >
          <RestartAltIcon />
        </IconButton>
      </Box>
    </div>

  </div>

 );

};
```

Кінець лістингу Б.1