ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ
Навчально-науковий інститут денної освіти
Форма навчання денна
Кафедра комп’ютерних наук та інформаційних технологій

Допускається до захисту
Завідувач кафедри
_____ Олена ОЛЬХОВСЬКА
(підпис)

«___» _____ 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на тему
РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ЗАСТОСУНКУ «КЛАВІАТУРНИЙ ТРЕНАЖЕР»

зі спеціальності 122 Комп’ютерні науки
освітня програма «Комп’ютерні науки»
ступеня магістра

Виконавець роботи Шугай Ігор Олександрович
_____ «____»_____ 2024 р.
(підпис)

Науковий керівник к.пед.н., доцент, Кошова Оксана Петрівна
_____ «____»_____ 2024 р.
(підпис)

Рецензент

ПОЛТАВА 2024 р.

ЗАТВЕРДЖУЮ
Завідувач кафедри _____Олена ОЛЬХОВСЬКА
(підпис)
«___» _____ 2023 р.

ЗАВДАННЯ І КАЛЕНДАРНИЙ ГРАФІК
ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

на тему «Розробка програмного забезпечення застосунку «Клавіатурний тренажер**»**
зі спеціальності 122 Комп'ютерні науки
освітня програма «Комп'ютерні науки»
ступеня магістр
Прізвище, ім'я, по батькові Шугай Ігор Олександрович
Затверджена наказом ректора № 8-Н від «16» січня 2023 р.
Термін подання студентом роботи «___»_____ 2024 р.
Вихідні дані до кваліфікаційної роботи: статті та документації з теми розробки клавіатурних тренажерів.

Зміст пояснювальної записки (перелік питань, які потрібно розробити)

ВСТУП
РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ
РОЗДІЛ 2. ОГЛЯД СИСТЕМ АНАЛОГІЧНОГО ПРИЗНАЧЕННЯ
2.1. Веб застосунки аналогічного призначення
2.2. Мобільні клавіатурні тренажери
2.3. Особливості розробки ПЗ клавіатурних тренажерів
РОЗДІЛ 3. ТЕОРЕТИЧНА ЧАСТИНА
3.1. Опис проектних рішень, інструментів та підходів до розробки ПЗ
3.2. Опис методології створення програмного забезпечення
РОЗДІЛ 4. ПРАКТИЧНА ЧАСТИНА
4.1. Опис та побудова діаграми класів розробленого застосунку
4.2. Опис функціональних можливостей розробленого клавіатурного тренажера
ВИСНОВКИ
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ
ДОДАТОК А

Перелік графічного матеріалу: 18 ілюстрацій.

Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали, посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Постановка задачі	Кошова О.П.		
Інформаційний огляд	Кошова О.П.		
Теоретична частина	Кошова О.П.		
Практична частина	Кошова О.П.		

Календарний графік виконання кваліфікаційної роботи

Зміст роботи	Термін виконання	Фактичне виконання
Вступ		
Вивчення методичних рекомендацій та стандартів та звіт керівнику		
Постановка задачі		
Інформаційний огляд джерел бібліотек та інтернету		
Теоретична частина		
Практична частина		
Закінчення оформлення		
Доповідь студента на кафедрі		
Доробка (за необхідністю), рецензування		

Дата видачі завдання «__» _____ 2023 р.

Здобувач вищої освіти _____ Шугай Ігор Олександрович
(підпис)

Науковий керівник _____ к.пед.н., доц. Кошова О.П.
(підпис) (науковий ступінь, вчене звання, ініціали та прізвище)

Результати захисту кваліфікаційної роботи

Кваліфікаційна робота оцінена на _____
(балів, оцінка за національною шкалою, оцінка за ECTS)

Протокол засідання ЕК № ____ від «____» _____ 2024 р.

Секретар ЕК_____ _____
(підпис) (ініціали та прізвище)

Затверджую
Зав. кафедрою _____
к.ф.-м.н. Олена ОЛЬХОВСЬКА
«____» _____ 2023 р.

Погоджено
Науковий керівник _____
к.пед.н., Оксана КОШОВА
«____» _____ 2023 р.

План

дипломного проекту з фаху
спеціальності 122 Комп'ютерні науки
освітня програма 122 Комп'ютерні науки
на тему «Розробка програмного забезпечення застосунку «Клавіатурний тренажер»
Прізвище, ім'я, по батькові Шугай Ігор Олександрович

ВСТУП
РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ
РОЗДІЛ 2. ОГЛЯД СИСТЕМ АНАЛОГІЧНОГО ПРИЗНАЧЕННЯ
2.1. Веб застосунки аналогічного призначення
2.2. Мобільні клавіатурні тренажери
2.3. Особливості розробки ПЗ клавіатурних тренажерів
РОЗДІЛ 3. ТЕОРЕТИЧНА ЧАСТИНА
3.1. Опис проектних рішень, інструментів та підходів до розробки ПЗ
3.2. Опис методології створення програмного забезпечення
РОЗДІЛ 4. ПРАКТИЧНА ЧАСТИНА
4.1. Опис та побудова діаграми класів розробленого застосунку
4.2. Опис функціональних можливостей розробленого клавіатурного тренажера
ВИСНОВКИ
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ
ДОДАТОК А

Здобувач вищої освіти _____ Шугай І.О.
(підпис)
«____» _____ 2023 р.

РЕФЕРАТ

Записка: 53 сторінки, 18 рисунків, 1 додаток, 13 літературних джерел.

Мета дипломної роботи розробка програмного забезпечення застосунку «Клавіатурний тренажер».

Об’єкт розробки – мобільний застосунок «Клавіатурний тренажер».

Методи дослідження – використання сучасних підходів для написання якісних застосунків для вивчення друку на клавіатурі; мови програмування C#, мови розмітки XAML із використанням фреймворку WPF; середовища розробки Microsoft Visual Studio.

Здійснено аналіз застосунків схожого призначення, опрацьовані особливості розробки ПЗ та технології створення графічних інтерфейсів застосунків типу «Клавіатурний тренажер». Встановлено конкретну методологію для розробки програмного забезпечення. Складено діаграму класів ПЗ розробленого застосунку. Розроблено інтерфейс та програмне забезпечення застосунку та описано функціональні можливості та особливості використання розробленого застосунку.

ЗМІСТ

ВСТУП

Актуальність розробки програмного забезпечення клавіатурного тренажера для студентів із комп'ютерних наук є надзвичайно високою, враховуючи специфіку їхньої академічної та майбутньої професійної діяльності. Студенти цієї спеціальності постійно працюють з великими обсягами тексту, зокрема під час написання коду, створення технічної документації, роботи з різними інструментами розробки та виконання лабораторних завдань. Ефективне оволодіння навичками швидкого та точного набору тексту на клавіатурі значно підвищує їхню продуктивність і сприяє покращенню якості навчання та роботи.

Навички сліпого набору є особливо актуальними для студентів із комп'ютерних наук, оскільки вони дозволяють їм зосереджуватися безпосередньо на задачах програмування чи обробки даних, не відволікаючись на пошук необхідних клавіш. Це сприяє підвищенню концентрації та зменшенню кількості помилок під час написання коду. Крім того, швидкий та точний набір тексту допомагає студентам ефективніше працювати з командним рядком, інтегрованими середовищами розробки (IDE) та іншими інструментами, що є важливою частиною їхньої професійної підготовки.

У сучасній комп'ютерній науці значна увага приділяється практичній роботі, в якій важливу роль відіграє взаємодія з різноманітними системами, зокрема під час роботи з системами контролю версій, розгортання програмних продуктів та їх тестування. Тому вміння ефективно вводити команди, писати скрипти та коментарі до коду є необхідним для успішного виконання цих завдань. Розробка спеціалізованого клавіатурного тренажера, адаптованого під потреби студентів комп'ютерних наук, дозволить їм розвинути навички, які вони активно використовуватимуть як під час навчання, так і в майбутній професійній діяльності.

Також важливим аспектом є інтеграція тренажера з навчальними матеріалами з програмування. Студенти можуть одночасно практикуватися в

наборі тексту та закріплювати знання синтаксису різних мов програмування, таких як Python, Java, C++ та інші. Тренажер може бути налаштований на введення типових фрагментів коду, що дозволить студентам відпрацьовувати використання специфічних символів, скорочень та конструкцій, характерних для цих мов. Це сприятиме покращенню їхньої загальної технічної грамотності та швидкості роботи з кодом.

Водночас розробка програмного забезпечення клавіатурного тренажера для вивчення англійської мови, яка буде реалізована у даній дипломній роботі, є значною з огляду на сучасні вимоги до навичок роботи з інформацією та комунікації в глобалізованому світі. Англійська мова є однією з найпоширеніших у світі, слугує міжнародним стандартом у науковому, бізнесовому та освітньому середовищах. Уміння швидко та правильно набирати текст англійською мовою на клавіатурі стає необхідною навичкою не лише для професійних перекладачів, редакторів чи журналістів, але й для широкого кола спеціалістів, включаючи програмістів, аналітиків, науковців та студентів.

Зростаюча роль дистанційної роботи та навчання також підкреслює необхідність оволодіння навичками ефективного набору тексту англійською мовою. Клавіатурні тренажери можуть сприяти розвитку цих навичок, забезпечуючи практику швидкого та точного введення тексту в умовах, наближених до реальних. Вони дозволяють користувачам вивчати розкладку клавіатури, освоювати сліпий метод набору, що сприяє покращенню моторних навичок та одночасному вивченню правильного написання англійських слів і фраз.

У процесі вивчення англійської мови ключовим фактором є тренування граматики, лексики та правопису. Використання клавіатурних тренажерів у цьому контексті є особливо ефективним, оскільки дозволяє інтегрувати практичні вправи з набору тексту в загальний процес вивчення мови. Більше того, сучасні тренажери можуть бути оснащені функціями перевірки правопису, підсвічуванням помилок та зворотним зв’язком, що підвищує ефективність навчання. Це допомагає користувачам одночасно вдосконалювати

навички друку та закріплювати знання граматичних та лексичних структур англійської мови.

Окрім освітніх переваг, розробка таких програм є актуальною і з точки зору професійного розвитку. Володіння навичками швидкого та точного друку англійською мовою є важливою компетенцією на ринку праці, особливо у сферах, що потребують регулярної взаємодії з іноземними партнерами, обміну інформацією та створення великого обсягу текстового контенту. Роботодавці все частіше висувають вимоги до кандидатів, які включають високий рівень володіння англійською мовою та вміння ефективно працювати з текстами. У зв’язку з цим, розробка програмного забезпечення клавіатурного тренажера з орієнтацією на вивчення англійської мови є обґрунтованою відповіддю на запити сучасного суспільства та ринку праці.

Таким чином, розробка програмного забезпечення для застосунку «Клавіатурний тренажер» є актуальною не тільки для підвищення ефективності навчання студентів спеціальності «Комп’ютерні науки», але й для розв’язання актуальних завдань, пов’язаних їх майбутньою професійною діяльністю.

Мета даної дипломної роботи - розробка програмного забезпечення застосунку «Клавіатурний тренажер».

Об’єктом розробки є мобільний застосунок «Клавіатурний тренажер».

Предметом розробки є програмне забезпечення застосунку «Клавіатурний тренажер» англійською мовою.

Розроблений застосунок для керуванння відеоконтентом реалізований за допомогою мови програмування C#, мови розмітки XAML із використанням фреймворку WPF для створення сучасного, динамічного, графічно-орієнтованого інтерфейсу користувача та середовища розробки Microsoft Visual Studio.

Пояснювальна записка включає чотири основних розділи: формулювання завдань, аналіз існуючих аналогів, теоретичні та практична частини. Структура роботи розрахована на логічне викладення матеріалу та повне розкриття теми дослідження.

РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ

Мета додатку

Метою розробки програмного забезпечення клавіатурного тренажера для студентів із комп’ютерних наук є підвищення їхньої швидкості та точності набору тексту, а також вдосконалення навичок роботи з клавіатурою, особливо англійською мовою, що є ключовими компетенціями в їхній професійній діяльності. Такий тренажер має сприяти ефективному засвоєнню студентами методів сліпого набору та розвитку моторних навичок, необхідних для продуктивного написання коду та взаємодії з різними програмними інструментами.

Окрім основного фокусу на швидкості та точності набору тексту, додаток також має на меті допомогти студентам закріпити знання синтаксису мов програмування та типових конструкцій, що використовуються під час розробки програмного забезпечення. Це може бути досягнуто шляхом інтеграції завдань із введенням коду, включаючи використання специфічних символів та операторів, які є характерними для різних мов програмування. Таким чином, додаток виступає не лише як інструмент для покращення навичок друку, але й як допоміжний засіб для вдосконалення знань у сфері комп’ютерних наук.

Додаткова мета тренажера полягає у сприянні розвитку в студентів здатності до багатозадачності та концентрації уваги, що є необхідним при роботі з великими обсягами коду та під час виконання комплексних завдань з програмування. Регулярне використання тренажера дозволить студентам зменшити кількість помилок, що виникають через відволікання на пошук потрібних клавіш, та допоможе підтримувати постійну концентрацію на логіці програмного коду.

Загалом, мета розробки такого додатку полягає у створенні інтерактивного навчального інструменту, який дозволить студентам комп’ютерних наук підвищити свою професійну підготовку, розвинути важливі практичні навички та покращити ефективність роботи з програмним

забезпеченням. Це сприятиме формуванню у студентів основ для подальшого успішного професійного розвитку у сфері інформаційних технологій.

Основні функції застосунку

Основні функції програмного забезпечення клавіатурного тренажера для студентів комп'ютерних наук спрямовані на забезпечення комплексного та індивідуалізованого підходу до навчання швидкому та точному набору тексту. Реалізація цих функцій дозволяє користувачам ефективно покращувати свої навички, отримувати зворотний зв'язок та стежити за власним прогресом.

Однією з ключових функцій додатку є «авторизація користувачів, яка дозволяє забезпечити персоналізований підхід до тренувань. Система авторизації дає можливість кожному користувачеві створити індивідуальний обліковий запис, у якому зберігається особиста інформація та дані про успіхи та досягнення під час навчання. Це сприяє формуванню індивідуальної траєкторії навчання, дозволяючи адаптувати тренування до рівня підготовки та потреб кожного студента. Крім того, авторизація забезпечує доступ до збережених результатів і прогресу незалежно від пристрою, що полегшує використання додатку в різних умовах.

Іншою важливою функцією є «вибір складності тексту для тренування». Ця можливість дозволяє користувачам самостійно обирати рівень складності відповідно до їхнього поточного рівня навичок та цілей навчання. Додаток може містити різні категорії текстів: від простих вправ з базовим набором символів до складних фрагментів коду з використанням специфічних синтаксичних конструкцій та символів, характерних для різних мов програмування. Така функціональність дозволяє студентам поступово збільшувати рівень складності завдань, що сприяє систематичному розвитку їхніх навичок.

Функція «збереження даних про користувачів» є важливим елементом для відстеження прогресу та надання зворотного зв'язку. Додаток має зберігати

детальну інформацію про виконані тренування, включаючи кількість помилок, швидкість набору, типи символів, які викликають труднощі, та загальний прогрес користувача. Аналіз цих даних дозволяє адаптувати подальші тренування та надавати користувачеві індивідуальні рекомендації щодо покращення навичок. Крім того, така функція забезпечує можливість порівняння власних результатів з попередніми або з досягненнями інших користувачів, що підвищує мотивацію та сприяє змагальному духу в процесі навчання.

Важливою складовою є також «система квестів», яка додає елемент гейміфікації до навчального процесу. Квести можуть бути представлені у вигляді різних завдань, наприклад, досягнення певної швидкості набору, виконання тренування без помилок або введення складних текстів за обмежений час. Виконання квестів стимулює користувачів до активної участі та постійного покращення власних результатів. Нагороди за виконання квестів можуть включати отримання віртуальних значків, підвищення рейтингу або розблокування нових рівнів складності, що додатково мотивує студентів продовжувати тренування.

Отже, поєднання функцій авторизації, вибору складності тексту, збереження даних про користувачів та системи квестів забезпечує комплексний та індивідуалізований підхід до розвитку навичок набору тексту. Це сприяє підвищенню ефективності навчання, дозволяючи студентам комп’ютерних наук систематично покращувати свою майстерність та підготуватися до подальшої професійної діяльності.

РОЗДІЛ 2. ОГЛЯД СИСТЕМ АНАЛОГІЧНОГО ПРИЗНАЧЕННЯ

2.1. Веб застосунки аналогічного призначення

У сучасному інформаційному суспільстві зростає попит на онлайн-інструменти для розвитку навичок ефективного використання клавіатури, що обумовлено необхідністю швидкого та точного введення тексту в різних сферах діяльності. Одними з найбільш поширених та відомих типів таких інструментів є веб-застосунки клавіатурних тренажерів, які пропонують широкий спектр функціональних можливостей для користувачів з різним рівнем підготовки. Ці застосунки доступні різними мовами, включно з українською та англійською, що розширює їхню цільову аудиторію.

Одним із найбільш відомих веб-застосунків є «TypingClub». Цей англомовний тренажер користується популярністю завдяки своїй інтуїтивно зрозумілій структурі та широкому набору функцій, спрямованих на поступовий розвиток навичок набору тексту. TypingClub пропонує інтерактивні уроки, які адаптуються до рівня користувача та містять різноманітні вправи для тренування сліпого методу набору. Застосунок надає детальний зворотний зв’язок, зокрема щодо точності та швидкості введення тексту, що допомагає користувачам відстежувати свій прогрес. Крім того, використання елементів гейміфікації, таких як нагороди та досягнення, сприяє підвищенню мотивації та заохочує до регулярних тренувань.

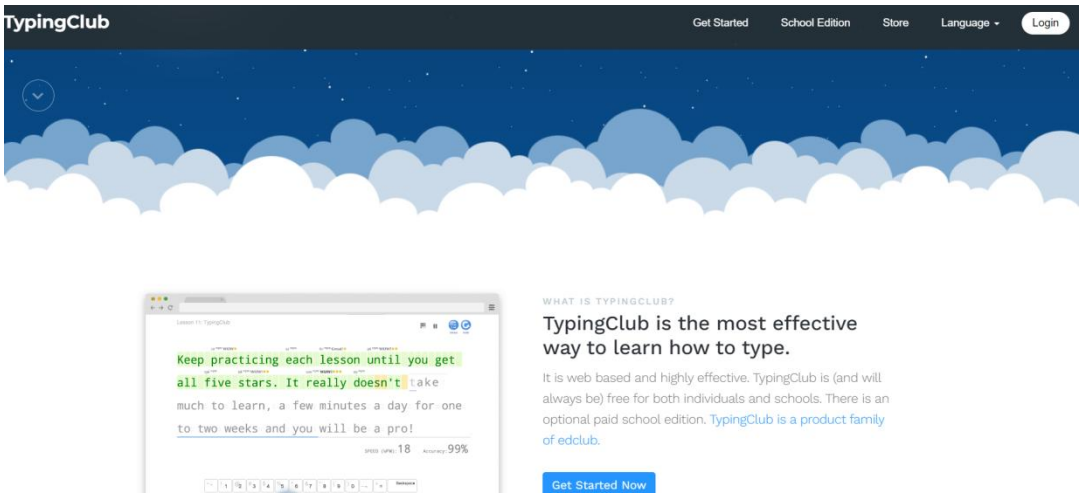

Рисунок 2.1. Веб тренажер TypingClub

Іншим популярним веб-застосунком є «Keybr.com», який також функціонує переважно англійською мовою, але підтримує кілька інших мов, включно з українською. Keybr.com відрізняється своїм підходом до навчання, який базується на генеруванні випадкових текстів, що складаються з найчастіше вживаних літер. Такий підхід дозволяє користувачам поступово освоювати розкладку клавіатури та покращувати швидкість набору. Застосунок використовує алгоритми штучного інтелекту для аналізу помилок та пропонує адаптивні вправи, які спрямовані на відпрацювання слабких сторін користувача. Це забезпечує індивідуальний підхід до навчання та сприяє швидкому вдосконаленню навичок.

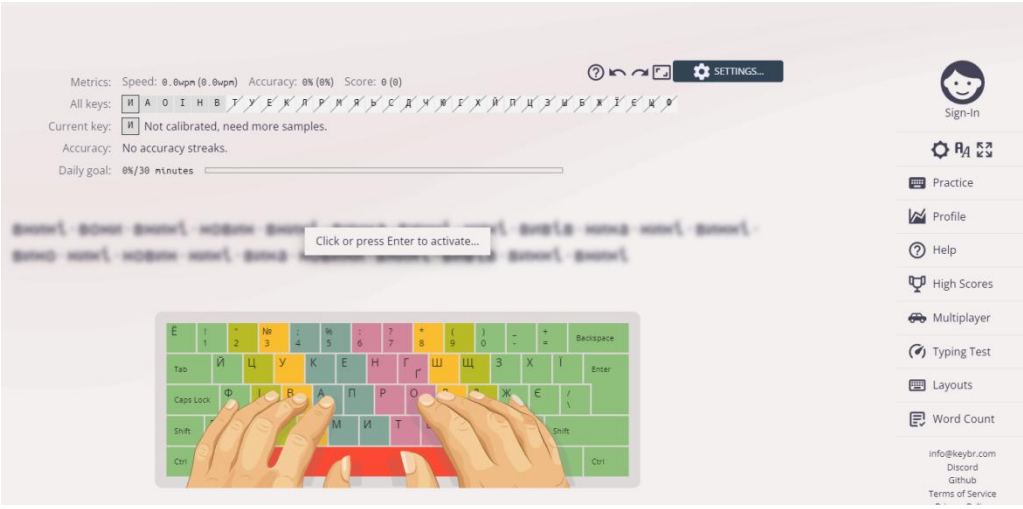

Рисунок 2.2. Веб тренажер Keybr.com

Для користувачів, які віддають перевагу україномовним веб-застосункам, варто відзначити «Ratatype». Цей тренажер пропонує інтерфейс українською мовою та дозволяє тренуватися як на базових рівнях, так і на більш просунутих. Ratatype пропонує велику кількість текстів для тренувань, а також можливість змагатися з іншими користувачами, що сприяє додатковій мотивації. Крім того, застосунок зберігає статистику успіхів кожного користувача та відображає докладну інформацію про швидкість та точність набору, дозволяючи відстежувати прогрес у реальному часі.

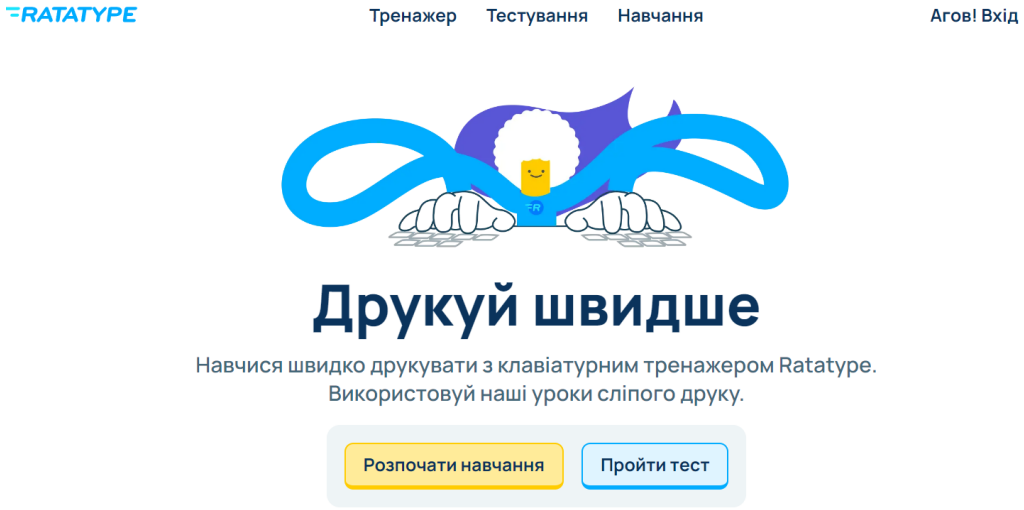

Рисунок 2.3. Тренажер Ratatype

Ще одним відомим англомовним застосунком є «Typing.com», який пропонує широкий спектр навчальних матеріалів для користувачів різного віку та рівня підготовки. Застосунок містить різноманітні уроки та інтерактивні ігри, що робить процес навчання цікавим та захопливим. Typing.com також забезпечує докладний аналіз результатів, включно з помилками та слабкими місцями користувача, що сприяє цілеспрямованому вдосконаленню навичок.

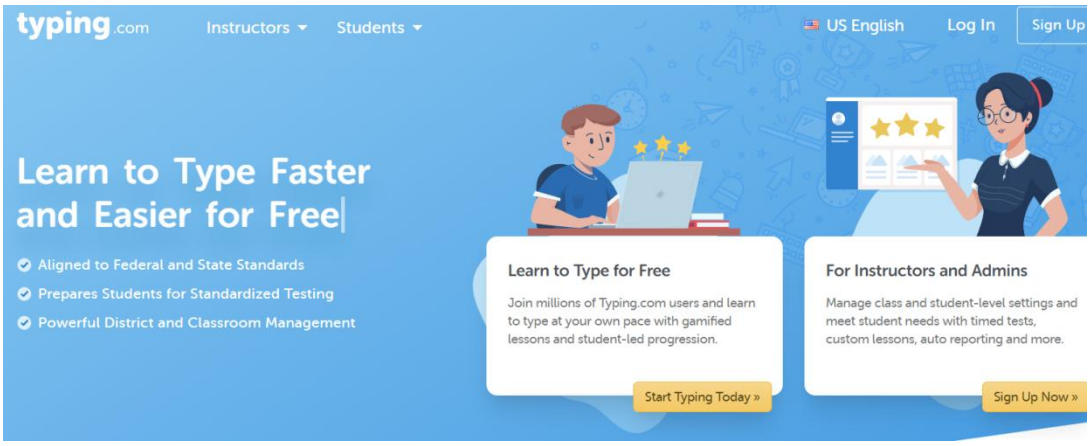

Рисунок 2.4. Клавіатурний тренажер Typing.com

Таким чином, існує багато веб-застосунків клавіатурних тренажерів, які здобули популярність серед користувачів завдяки своїм адаптивним методам навчання, використанню елементів гейміфікації та можливостям відстеження прогресу. Їхня доступність різними мовами, включно з українською та англійською, дозволяє задовольнити потреби широкої аудиторії та сприяє підвищенню загального рівня володіння навичками набору тексту.

2.2. Мобільні клавіатурні тренажери

Серед різноманіття інструментів для вдосконалення навичок набору тексту значне місце посідають клавіатурні тренажери, реалізовані у вигляді мобільних та десктопних застосунків. Їх популярність обумовлена зручністю використання, широким набором функцій та можливістю персоналізації процесу навчання. Розробники таких застосунків надають особливу увагу адаптивності та інтерактивності, що дозволяє користувачам різного віку та рівня підготовки ефективно вдосконалювати свої навички набору тексту.

Одним із найбільш відомих десктопних клавіатурних тренажерів є «TypingMaster». Цей застосунок призначений для використання на персональних комп’ютерах і пропонує комплексну програму навчання сліпому

методу набору. TypingMaster включає інтерактивні уроки, вправи та ігри, які адаптуються до рівня користувача. Особливістю цього тренажера є детальний аналіз помилок та зон слабкості, на основі якого автоматично коригуються наступні вправи для покращення результатів. Крім того, TypingMaster надає користувачам можливість відстежувати свій прогрес за допомогою вбудованих інструментів статистики, що дозволяє побачити динаміку розвитку навичок.

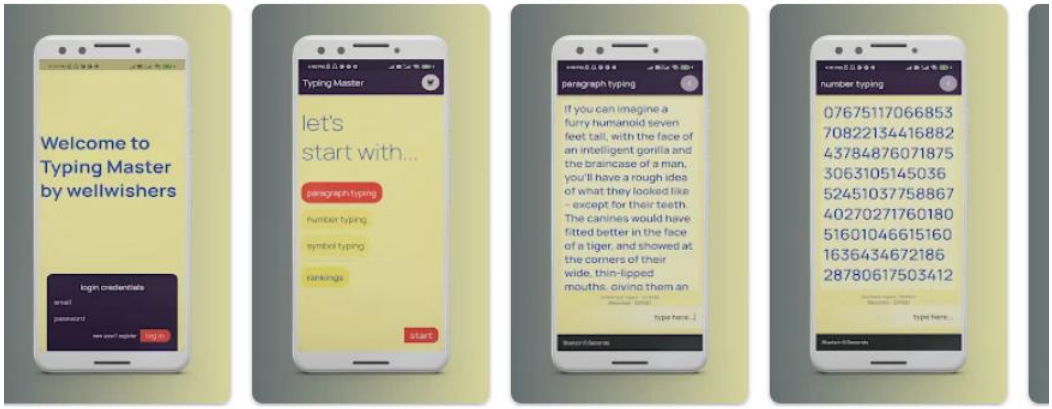

Рисунок 2.5. Android застосунок TypingMaster

Серед мобільних застосунків для тренування набору тексту одним із найпопулярніших є «Keybr – Typing Practice». Цей застосунок доступний для платформ Android та iOS і забезпечує можливість тренування на мобільних пристроях. Keybr – Typing Practice використовує адаптивний алгоритм, який автоматично генерує тексти з урахуванням найчастіше вживаних літер та помилок користувача. Завдяки цьому застосунок пропонує вправи, спрямовані на відпрацювання конкретних слабких місць користувача, що підвищує ефективність навчання. Мобільна версія тренажера також дозволяє користувачам тренуватися у будь-який зручний час, розширюючи можливості для практики.

Ще одним відомим десктопним застосунком є «Klavaro Touch Typing Tutor». Цей тренажер має кросплатформену сумісність та доступний для операційних систем Windows, macOS і Linux. Klavaro Touch Typing Tutor відрізняється простим інтерфейсом та можливістю налаштування різних мов і

розкладок клавіатури, що робить його універсальним для користувачів з різних країн. Застосунок пропонує кілька типів вправ, включаючи введення випадкових текстів, вправи на швидкість та точність, а також модулі для відстеження прогресу. Важливою особливістю є підтримка кастомізації вправ, що дозволяє користувачам самостійно вибирати тексти для тренувань.

Рисунок 2.6. Klavaro Touch Typing Tutor

Серед мобільних тренажерів також варто відзначити застосунок «Gboard Typing Speed Test». Розроблений для пристроїв Android, цей застосунок є частиною клавіатурного програмного забезпечення Gboard від Google. Він пропонує простий спосіб оцінити та вдосконалити швидкість набору тексту на мобільному пристрої. Хоча його функціональність менш розширена порівняно з іншими спеціалізованими тренажерами, можливість використання на смартфонах робить його доступним для широкого кола користувачів та дозволяє проводити тренування в будь-який час.

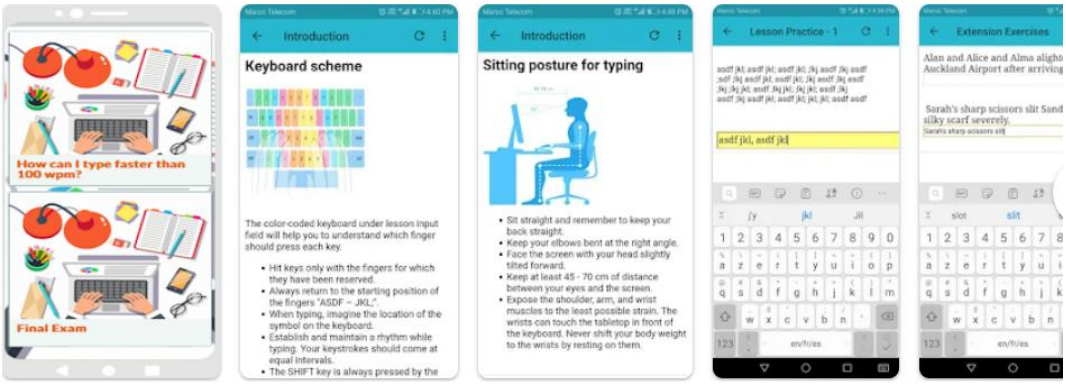

Рисунок 2. 7. Android застосунок Gboard Typing Speed Test

Інший україномовний застосунок, доступний для Android, – «Давай займемось текстом». Він підходить як для тих, хто тільки починає вивчати українську мову, так і для впевнених мовців. Цей застосунок містить навчальні курси, ігри та вправи для розширення словникового запасу та покращення мовних навичок. Програма робить процес навчання веселим та привабливим для користувачів різного віку

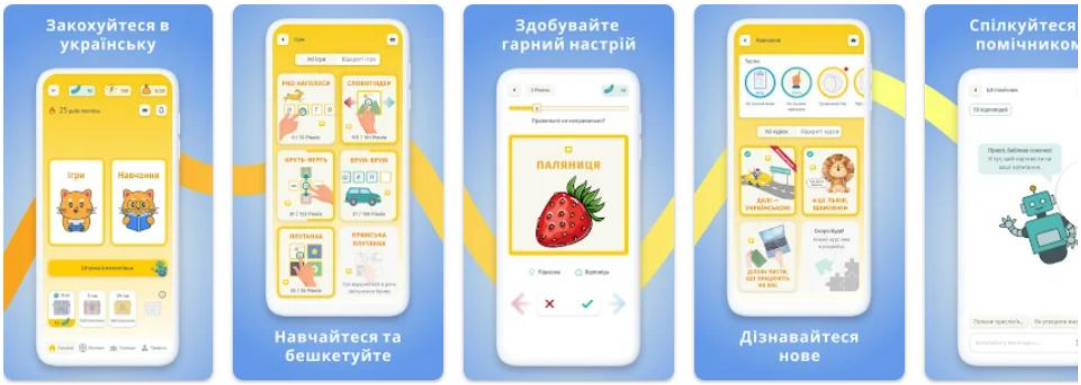

Рисунок 2.8. Тренажер «Давай займемось текстом»

Таким чином, сучасні клавіатурні тренажери, реалізовані як у вигляді мобільних, так і десктопних застосунків, надають користувачам потужний інструментарій для розвитку навичок набору тексту. Вони забезпечують адаптивність навчання, враховують індивідуальні потреби користувача та надають інтерактивний досвід, який сприяє підвищенню мотивації та ефективності навчального процесу. Ці застосунки відзначаються різноманіттям

підходів до навчання, включаючи використання адаптивних алгоритмів, статистики успіхів та кастомізації вправ, що робить їх важливим компонентом сучасної комп’ютерної грамотності.

2.3. Особливості розробки ПЗ клавіатурних тренажерів

Розробка клавіатурних тренажерів вимагає застосування різноманітних технологій, інструментів та мов програмування, які дозволяють створити функціональний, адаптивний та зручний інтерфейс для тренування навичок швидкого та точного набору тексту. У процесі розробки використовуються сучасні засоби для фронтенд- та бекенд-програмування, платформи для розробки інтерфейсу користувача, а також інструменти для забезпечення адаптивності й інтерактивності застосунку.

Для створення «десктопних застосунків» клавіатурних тренажерів найчастіше використовується мова програмування «C#» у поєднанні з технологією «Windows Presentation Foundation (WPF)». C# є сучасною мовою програмування, яка підтримує об’єктно-орієнтовану парадигму та надає широкі можливості для створення графічного інтерфейсу користувача. WPF, у свою чергу, забезпечує векторну графіку, анімацію та гнучку систему прив’язки даних, що дозволяє розробникам створювати адаптивні та інтерактивні інтерфейси для тренажерів. «XAML (eXtensible Application Markup Language)» використовується для декларативного опису елементів інтерфейсу, що дає змогу розділити логіку та візуальну складову застосунку.

Для «мобільних застосунків», зокрема під платформу «Android», застосовують мову програмування «Java» або «Kotlin», а також інструменти, такі як «Android Studio». Java є основною мовою для розробки Android-застосунків та має багатий набір бібліотек і фреймворків для створення інтерактивних інтерфейсів користувача. Kotlin, як сучасніша мова, забезпечує більшу гнучкість і спрощує синтаксис. Крім того, використання технології «XML» дозволяє описувати макети інтерфейсу та компоненти, які можуть бути

інтегровані з бізнес-логікою на Java або Kotlin.

У випадку розробки «веб-застосунків» клавіатурних тренажерів використовуються веб-технології, такі як «HTML, CSS, JavaScript» і фреймворки «React», «Vue.js» або «Angular». HTML та CSS відповідають за структуру та стиль інтерфейсу, тоді як JavaScript забезпечує його динамічність та інтерактивність. Наприклад, фреймворк «React» дозволяє створювати компоненти користувацького інтерфейсу та забезпечує ефективне оновлення лише тих частин сторінки, які потребують змін. Для розробки серверної частини (бекенду) таких застосунків використовуються мови програмування, як-от «Python» з фреймворком «Django», або «JavaScript» з «Node.js», які забезпечують роботу з базами даних, управління користувачами та обробку введених даних.

Для забезпечення збереження даних про користувачів, їхній прогрес та результати тренувань застосовуються «системи управління базами даних (СУБД)», такі як «SQLite», «MySQL» або «MongoDB». Ці СУБД дозволяють організувати структуру даних і забезпечити їхнє швидке зберігання та доступ. У десктопних застосунках, наприклад, може використовуватися локальна база даних SQLite для збереження інформації про результати користувача, тоді як у веб-застосунках доцільно використовувати реляційні (MySQL) або документно-орієнтовані (MongoDB) бази даних на сервері.

Для забезпечення адаптивності та інтерактивності клавіатурних тренажерів також використовуються засоби гейміфікації, що вимагає застосування інструментів для анімації та обробки подій. У мобільних застосунках на Android це можуть бути бібліотеки «Android Animations» для створення візуальних ефектів та інтерактивних елементів, тоді як у десктопних застосунках на WPF застосовується система анімації, інтегрована в XAML.

Розробка сучасного графічного інтерфейсу та візуалізації для клавіатурних тренажерів вимагає використання спеціалізованих інструментів, які забезпечують інтуїтивність, адаптивність та візуальну привабливість застосунку. Ці інструменти дозволяють розробникам створювати високоякісний

інтерфейс користувача, підтримувати інтерактивність елементів та впроваджувати елементи гейміфікації для підвищення ефективності навчання.

Одним із ключових інструментів для розробки графічного інтерфейсу є «Windows Presentation Foundation (WPF)», який використовується для створення десктопних застосунків. WPF забезпечує підтримку векторної графіки, що дозволяє створювати масштабовані та високоякісні графічні елементи незалежно від роздільної здатності екрана. Застосовуючи мову розмітки «XAML (eXtensible Application Markup Language)», розробники можуть декларативно описувати інтерфейс користувача, включаючи компоненти, анімації та стилі. Крім того, WPF підтримує гнучку систему прив’язки даних, що дозволяє інтерактивно оновлювати інтерфейс відповідно до дій користувача, наприклад, підсвічувати правильні та неправильні введення під час тренування.

Для розробки веб-застосунків використовуються фронтенд-фреймворки, такі як «React», «Vue.js» та «Angular». Ці інструменти дозволяють створювати динамічні інтерфейси користувача, забезпечуючи високу реактивність і плавність взаємодії. Наприклад, «React» надає можливість створення компонентів, які можуть повторно використовуватися для відображення різних елементів інтерфейсу, таких як текстові поля для введення, індикатори прогресу чи клавіші клавіатури. Це сприяє зменшенню кількості коду та підвищує гнучкість розробки інтерфейсу. Для візуального оформлення в веб-застосунках часто застосовуються «CSS» та його препроцесори, такі як «Sass» або «LESS», які дозволяють створювати складні стилі та анімації, роблячи інтерфейс більш привабливим та зручним для користувача.

У контексті мобільних застосунків для платформи «Android» використовується «Android Studio» у поєднанні з «XML» для опису макетів інтерфейсу. «ConstraintLayout» та інші компоненти розмітки в Android Studio забезпечують створення адаптивних інтерфейсів, які можуть змінювати свою структуру відповідно до розміру та орієнтації екрана. Крім того, для додавання візуальних ефектів та інтерактивних елементів в Android-застосунках

використовуються бібліотеки, такі як «Android Animations» та «Lottie». «Lottie» особливо корисна для відтворення анімацій, розроблених у спеціалізованих інструментах, як-от «Adobe After Effects», що дозволяє створювати складні та плавні анімації з мінімальним впливом на продуктивність застосунку.

Для створення та редагування графічних елементів інтерфейсу часто використовуються інструменти графічного дизайну, такі як «Adobe Photoshop» та «Adobe Illustrator». Вони дозволяють розробникам та дизайнерам створювати іконки, кнопки, індикатори прогресу та інші графічні елементи, які можуть бути інтегровані в застосунок. Для створення та впровадження анімацій у десктопні та веб-застосунки також використовується «Adobe After Effects», а готові анімації можуть бути експортувані у формат, сумісний з Lottie або іншими бібліотеками анімації.

Для організації та розробки інтерфейсу з урахуванням зручності користувача застосовуються принципи «UX/UI дизайну». Інструменти, такі як «Figma» або «Sketch», використовуються для створення прототипів інтерфейсу, які дозволяють проводити тестування дизайну на ранніх етапах розробки та вносити корективи для покращення користувацького досвіду.

Таким чином, розробка сучасного графічного інтерфейсу та візуалізації для клавіатурних тренажерів вимагає застосування різноманітних технологій та інструментів. Використання фреймворків, таких як WPF, React, Android Studio, у поєднанні з графічними та анімаційними інструментами забезпечує високий рівень адаптивності, інтерактивності та візуальної привабливості інтерфейсу, сприяючи ефективності навчального процесу. Саме тому розробка клавіатурних тренажерів є міждисциплінарною задачею, яка потребує використання різноманітних технологій, інструментів та мов програмування для створення адаптивних, інтуїтивно зрозумілих та ефективних навчальних інструментів.

РОЗДІЛ 3. ТЕОРЕТИЧНА ЧАСТИНА

3.1. Опис проектних рішень, інструментів та підходів до розробки ПЗ

Для розробки програмного забезпечення застосунку «Клавіатурний тренажер» було використано наступні технології:

- мову програмування C#;
- мову розмітки XAML;
- фреймворку Windows Presentation Foundation (WPF)
- середовище розробки Microsoft Visual Studio.

Мова програмування C#

Мова програмування C# є однією з найпопулярніших та найпотужніших мов у сучасній розробці програмного забезпечення, особливо у середовищі Windows [11, 12]. C# – це сучасна мова програмування, розроблена корпорацією Microsoft для платформи .NET, яка відзначається своєю багатофункціональністю та об’єктно-орієнтованим підходом. Вона поєднує в собі простоту, виразність і продуктивність, що робить її ефективним інструментом для широкого спектра програмних завдань, включаючи розробку десктопних додатків. Завдяки своїй тісній інтеграції з платформою .NET, C# надає розробникам доступ до багатого набору бібліотек та інструментів, що сприяє швидкому та надійному створенню високоякісних застосунків.

Одним із прикладів практичного застосування C# є розробка десктопних клавіатурних тренажерів. Такі програми, як правило, мають на меті навчання та вдосконалення навичок друкування, зокрема сліпого набору тексту. Для цього C# пропонує низку можливостей, зокрема потужну обробку подій, розширені можливості роботи з графічним інтерфейсом користувача та засоби для ефективної обробки введення даних з клавіатури.

При розробці клавіатурного тренажера на C# з використанням Windows Presentation Foundation (WPF) розробник може створювати інтуїтивно

зрозумілий та інтерактивний інтерфейс користувача, застосовуючи сучасні графічні можливості. Використання WPF у поєднанні з C# дозволяє реалізувати динамічні візуальні елементи, такі як підсвічування клавіш, анімації та індикатори прогресу, що підвищує ефективність процесу навчання. Крім того, C# забезпечує простоту роботи з різноманітними подіями клавіатури, що є ключовим у створенні тренажера. Наприклад, можна реалізувати обробку натискання клавіш, перевірку коректності введених символів та відображення зворотного зв’язку в реальному часі, що сприяє покращенню навичок користувача.

Завдяки своїй структурованості та багатству інструментів, C# є оптимальним вибором для розробки програм цього типу. Можливість використовувати об’єктно-орієнтований підхід та вбудовані методи обробки даних дозволяє створювати високоякісні, масштабовані та адаптивні клавіатурні тренажери. Це робить мову C# ефективним інструментом не лише для створення простих навчальних програм, але й для розробки складних систем з використанням графіки, анімації та інтерактивного введення.

Фреймворк Windows Presentation Foundation (WPF)

Windows Presentation Foundation (WPF) – це фреймворк для розробки десктопних додатків під операційну систему Windows, впроваджений компанією Microsoft як частина .NET Framework [2, 10]. WPF надає розробникам комплексний набір інструментів для створення сучасних, динамічних та графічно-орієнтованих інтерфейсів користувача. Однією з ключових характеристик WPF є використання XAML (eXtensible Application Markup Language) для визначення елементів інтерфейсу. Це дозволяє відокремити декларативний опис компонентів інтерфейсу від логіки на бекенді, яка зазвичай реалізується за допомогою мови програмування C# або інших мов платформи .NET.

Однією з основних переваг WPF є використання векторної графіки, що

забезпечує масштабованість елементів інтерфейсу без втрати якості. Цей підхід дозволяє створювати гнучкі та адаптивні додатки, незалежно від розміру екрану або роздільної здатності. Крім того, WPF оснащений потужною системою прив’язки даних (Data Binding), що спрощує відображення інформації з різних джерел та дозволяє автоматично оновлювати інтерфейс при зміні даних.

У WPF реалізована система стилів і шаблонів, яка дозволяє застосовувати різноманітні оформлення до елементів інтерфейсу, забезпечуючи створення узгодженого та кастомізованого користувацького досвіду. Крім цього, вбудована підтримка анімації дозволяє додавати візуальні ефекти та забезпечувати динамічну взаємодію користувача з додатком. Важливим аспектом WPF є наявність багатого набору готових компонентів, які можна використовувати для побудови інтерфейсу, зокрема таких, як кнопки, списки та текстові поля.

Загалом, Windows Presentation Foundation є потужним фреймворком для створення складних та кастомізованих інтерфейсів користувача у Windows-застосунках. Його особливості, зокрема підтримка векторної графіки, прив’язка даних та система стилів, роблять його ідеальним вибором для розробки додатків, які потребують відображення багатого контенту, включаючи графіку, медіафайли та інтерактивні елементи [2, 10].

Мова розмітки XAML

XAML (eXtensible Application Markup Language) [3, 8, 13] є мовою розмітки, розробленою корпорацією Microsoft, яка використовується для опису та побудови інтерфейсів користувача у застосунках на платформі .NET, зокрема в середовищах Windows Presentation Foundation (WPF) та Universal Windows Platform (UWP). Відзначаючись своєю декларативною природою, XAML дозволяє розробникам створювати складні та динамічні графічні інтерфейси за допомогою простого та інтуїтивно зрозумілого синтаксису. Завдяки цій мові можна відокремити візуальну складову додатка від його логіки, що забезпечує більш модульний та організований підхід до розробки програмного

забезпечення.

XAML базується на XML і використовує його синтаксис для опису елементів інтерфейсу, що робить цю мову гнучкою та легкою для сприйняття як розробниками, так і дизайнерами. Однією з ключових особливостей XAML є можливість створення та налаштування елементів інтерфейсу користувача, таких як кнопки, текстові поля, панелі та вікна, у декларативному форматі. Це дозволяє розробникам вказувати властивості, стилі та структуру елементів, визначаючи їхній вигляд та поведінку ще на етапі проектування. У процесі виконання додатка XAML розпізнається та обробляється .NET середовищем, перетворюючись у відповідні об’єкти інтерфейсу.

Однією з головних переваг XAML є підтримка концепції прив’язки даних (data binding), що дозволяє зв’язувати елементи інтерфейсу користувача з джерелами даних, такими як властивості об’єктів або колекції. Це сприяє створенню динамічних додатків, де зміни у стані даних автоматично відображаються на інтерфейсі користувача. Крім того, XAML забезпечує можливість використання стилів та шаблонів, які дозволяють уніфікувати вигляд компонентів та полегшують реалізацію кастомізованого дизайну.

Також важливою особливістю XAML є підтримка анімації та візуальних ефектів, що дозволяє створювати інтерактивні та привабливі інтерфейси користувача. Мова XAML надає розробникам засоби для визначення анімаційних ефектів, таких як зміна кольору, розміру або положення елементів у часі, завдяки чому інтерфейс стає більш динамічним та інтерактивним. У поєднанні з можливостями C#, XAML стає потужним інструментом для розробки сучасних, багатофункціональних застосунків з інтуїтивно зрозумілим графічним інтерфейсом [3, 8, 13] .

Середовище розробки Visual Studio

Visual Studio - це інтегроване середовище розробки (IDE), створене компанією Microsoft, яке забезпечує потужні інструменти та сервіси для розробки різноманітного програмного забезпечення, включаючи веб-, мобільні

та настільні додатки, а також серверні системи [4, 5]. Основною метою Visual Studio є спрощення та оптимізація процесу розробки програм за допомогою набору інструментів, які дозволяють ефективно створювати, налагоджувати, тестувати та розгортати програмні продукти (Рисунок 3.1).

Рисунок 3.1. Логотип Microsoft Visual Studio

Visual Studio є інтегрованим середовищем розробки (IDE), розробленим корпорацією Microsoft, що забезпечує розробникам багатий набір інструментів для створення різноманітних типів застосунків, включаючи веб-додатки, десктопні програми, мобільні застосунки та хмарні сервіси. Завдяки своїм розширеним можливостям, Visual Studio відіграє центральну роль у розробці програмного забезпечення на платформі .NET, пропонуючи потужний і гнучкий інтерфейс для розробки, тестування, налагодження та розгортання додатків.

Однією з ключових особливостей Visual Studio є підтримка різноманітних мов програмування, включаючи C#, Visual Basic .NET, C++, JavaScript та Python, що робить його універсальним інструментом для розробників з різним рівнем досвіду та професійними уподобаннями. Вбудований редактор коду забезпечує інтелектуальну допомогу, зокрема підсвічування синтаксису, автодоповнення, навігацію по коду та інтеграцію систем контролю версій, таких як Git. Це спрощує процес написання коду, робить його більш ефективним і дозволяє зосередитися на основних аспектах розробки.

Visual Studio також містить потужний відладчик, який дозволяє виконувати покрокове налагодження додатків, відстежувати значення змінних

та виявляти можливі помилки в коді. Завдяки цьому розробники можуть швидко ідентифікувати та виправляти помилки на ранніх етапах розробки. Інструменти тестування, інтегровані в середовище Visual Studio, надають можливість автоматизувати процес тестування застосунків, використовуючи модульні, інтеграційні та інші типи тестів, що сприяє підвищенню надійності та якості програмного забезпечення.

Для розробки інтерфейсів користувача, зокрема у WPF та UWP, Visual Studio пропонує зручний дизайнер XAML, який дозволяє створювати та редагувати візуальні компоненти у візуальному редакторі. Це значно спрощує процес побудови складних графічних інтерфейсів, даючи можливість одночасно бачити результат змін у коді. У поєднанні з можливістю використання XAML, C# та інших мов програмування, Visual Studio стає універсальним інструментом для розробки сучасних десктопних застосунків з багатофункціональним інтерфейсом.

Ще однією важливою складовою Visual Studio є інтеграція з хмарними сервісами, такими як Microsoft Azure. Це дає розробникам можливість створювати, розгортати та керувати хмарними застосунками та службами безпосередньо з IDE. Завдяки своїм багатогранним можливостям та гнучкій архітектурі, Visual Studio є невід’ємним інструментом для розробників, які прагнуть створювати високоякісні та продуктивні програмні рішення [4, 5].

3.2. Опис методології створення програмного забезпечення

Для розробки клавіатурних тренажерів найкраще застосовувати методологію гнучкої розробки програмного забезпечення (Agile). Такий вибір зумовлений специфікою процесу створення тренажерів, який передбачає забезпечення інтуїтивно зрозумілого інтерфейсу, високої продуктивності та адаптивності до потреб різних категорій користувачів. Гнучкі методології розробки, такі як Agile, сприяють поетапному підходу до розробки, дозволяючи

створювати складний функціонал шляхом ітеративного вдосконалення, з регулярним тестуванням і залученням зворотного зв’язку від користувачів.

Однією з головних переваг використання методології Agile є її ітеративний характер. Оскільки клавіатурні тренажери зазвичай мають низку функціональних вимог, які можуть змінюватися або розширюватися в процесі розробки, Agile дозволяє реалізовувати ці функції поетапно, додаючи нові компоненти, такі як різні режими тренування, статистику результатів або анімаційні ефекти. Такий підхід забезпечує гнучкість та дозволяє тестувати кожен окремий компонент, своєчасно отримуючи зворотний зв’язок, що сприяє покращенню якості кінцевого продукту.

Крім того, гнучкі методології дозволяють залишатися адаптивними до зміни вимог. Враховуючи, що клавіатурні тренажери можуть бути орієнтовані на різні групи користувачів – новачків, досвідчених користувачів або дітей – вимоги до їх функціоналу та дизайну можуть коригуватися у процесі розробки. Agile надає можливість модифікувати вимоги та функціональні характеристики на будь-якому етапі, підтримуючи високий рівень якості та продуктивності.

Значущим аспектом Agile є його фокус на користувачеві. У рамках гнучкої розробки передбачається регулярна взаємодія з кінцевими користувачами та залучення їх до процесу розробки. У випадку створення клавіатурного тренажера це стає важливим, оскільки ефективність навчання залежить від інтуїтивності інтерфейсу та адаптованості до потреб користувача. Agile дозволяє розробникам збирати зворотний зв’язок від цільової аудиторії та на його основі коригувати функціонал і дизайн, що підвищує ефективність та привабливість тренажера.

Також методологія Agile сприяє покращенню якості програмного забезпечення завдяки акценту на постійному тестуванні. Виявлення та виправлення помилок у роботі тренажера, оптимізація його інтерфейсу та логіки відбуваються на кожному етапі розробки, що підвищує надійність продукту та робить його більш ефективним у використанні. Такий підхід є особливо важливим для клавіатурних тренажерів, де користувачі прагнуть

досягти високої швидкості та точності введення.

Модульний підхід, який пропонує Agile, дозволяє розбити розробку тренажера на окремі етапи або спринти, кожен з яких відповідає за реалізацію конкретного функціоналу, такого як введення тексту, підрахунок статистики чи відображення результатів. Це дозволяє команді розробників паралельно працювати над різними частинами системи, сприяючи її швидшому та скоординованому впровадженню.

Таким чином, застосування методології гнучкої розробки є оптимальним підходом для створення клавіатурних тренажерів. Її ітеративність, адаптивність до змін та фокус на користувачі дозволяють ефективно розробляти високоякісні програмні продукти, що відповідають потребам цільової аудиторії [9].

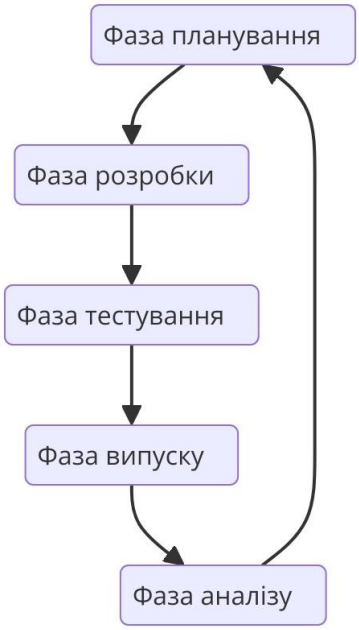

Рисунок. 3.2. Модель розробки застосунку

Розробка клавіатурного тренажера за методологією гнучкої розробки (Agile) передбачає ітеративний процес, який включає декілька основних фаз. Ці фази сприяють поетапному впровадженню функціональних можливостей та вдосконаленню продукту на основі постійного зворотного зв’язку від користувачів і тестування. Основними фазами моделі розробки клавіатурного тренажера за цією методологією є: планування, розробка, тестування, випуск та

аналіз.

Фаза планування є початковою і ключовою фазою в циклі розробки. На цьому етапі команда розробників визначає загальну концепцію клавіатурного тренажера, цільову аудиторію та основні функціональні можливості, які будуть реалізовані. У процесі планування проводиться збір вимог від потенційних користувачів та стейкхолдерів, формується список завдань та пріоритетів для кожної ітерації. Для ефективної організації роботи команда визначає тривалість кожного спринту (зазвичай від одного до чотирьох тижнів) та розподіляє функціональні можливості, які будуть реалізовані протягом кожного з них.

Фаза розробки починається з виконання завдань, визначених у плануванні. Кожен спринт присвячений створенню певного функціоналу тренажера, такого як розробка інтерфейсу користувача, обробка введення з клавіатури, реалізація режимів тренувань або підрахунок статистики. Розробники використовують інструменти, такі як Visual Studio та мову програмування C# з XAML, для побудови інтерфейсу та логіки програми. Фаза розробки також включає створення та інтеграцію модулів, які забезпечують основну функціональність тренажера. В процесі цієї фази команда може здійснювати швидкі коригування дизайну та функціоналу на основі проміжних результатів та зворотного зв’язку.

Фаза тестування проводиться на кожному етапі розробки, а також після завершення кожного спринту. Ця фаза передбачає ретельне перевіряння функціональності клавіатурного тренажера, зокрема його інтерфейсу, точності обробки введення з клавіатури, стабільності роботи та коректності підрахунку статистики. Тестування може включати як модульні тести для перевірки окремих компонентів системи, так і інтеграційні та користувацькі тести для оцінки взаємодії різних модулів та загального досвіду користувача. Важливим аспектом цієї фази є збір зворотного зв’язку від потенційних користувачів, який дозволяє виявляти недоліки та враховувати їх у подальших ітераціях розробки.

Фаза випуску передбачає впровадження завершеного функціоналу тренажера у виробниче середовище. Після успішного тестування розроблений

функціонал готовий до використання кінцевими користувачами. На цьому етапі команда може випустити оновлення, що включають нові можливості або виправлення помилок, виявлених у попередніх фазах. Завдяки гнучкій природі методології Agile випуск функціональних компонентів здійснюється поетапно, дозволяючи користувачам оцінити новий функціонал та надати відгуки щодо його роботи.

Фаза аналізу завершує кожен цикл розробки. На цьому етапі команда оцінює результати спринту, аналізує ефективність впровадженого функціоналу та збирає зворотний зв’язок від користувачів. Аналіз дозволяє визначити, які аспекти тренажера потребують подальшого вдосконалення та які нові вимоги можуть бути додані до наступних ітерацій. Отримана інформація використовується для коригування планів на майбутні спринти, оптимізації процесу розробки та підвищення якості кінцевого продукту.

Таким чином, кожна з описаних фаз сприяє ітеративному вдосконаленню клавіатурного тренажера, забезпечуючи адаптивність до змін вимог та покращення загального користувацького досвіду [9].

РОЗДІЛ 4. ПРАКТИЧНА ЧАСТИНА

4.1. Опис та побудова діаграми класів розробленого застосунку

Розклянемо особливості ПЗ розробленого застосунку у вигляді діаграм класів відповідних процесів, які покажуть ключові можливості та функціональність розробленого програмного забезпечення застосунку «Клавіатурний тренажер».

Зокрема, користувач під час роботи із тренажером виконує різні завдання (квести), які відрізняються рівнем складності. Система зберігання квестів та маніпулювання ними представлена на діаграмі класів (Рисунок 4.1.).

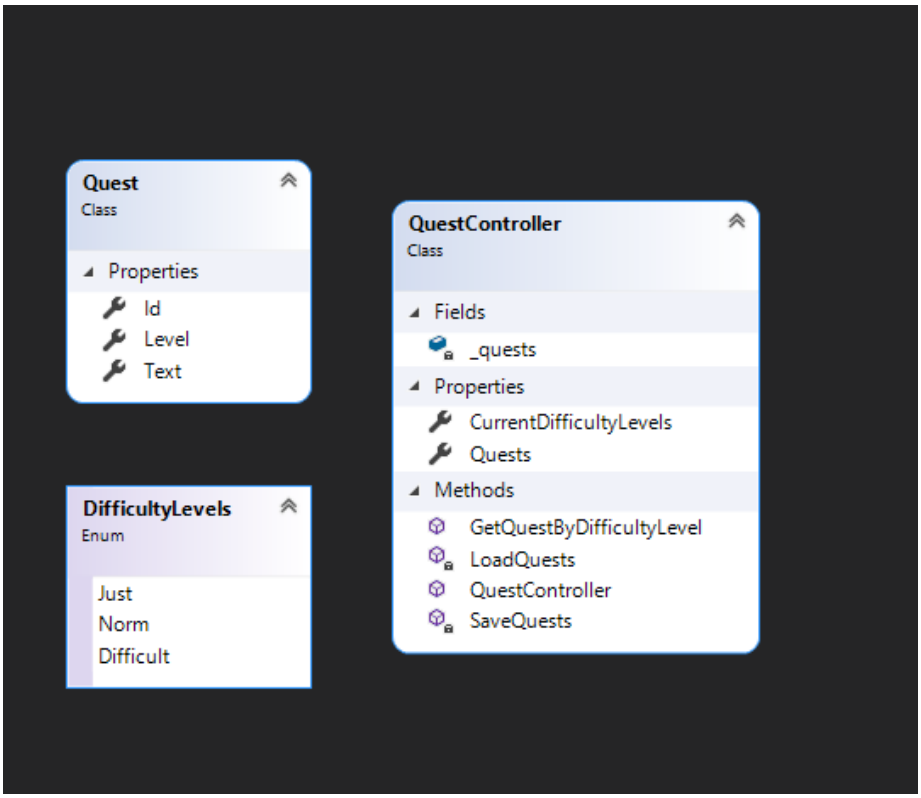

Рисунок 4.1. Діаграма класів зберігання та маніпулювання квестами у застосунку

Квести зберігаються у форматі .xml

Кожен гравець перед грою проходить спрощену авторизацію: просимо

ввести email та пароль.

Система зберігання та маніпулювання користувачами представлена на наступній діаграмі класів:

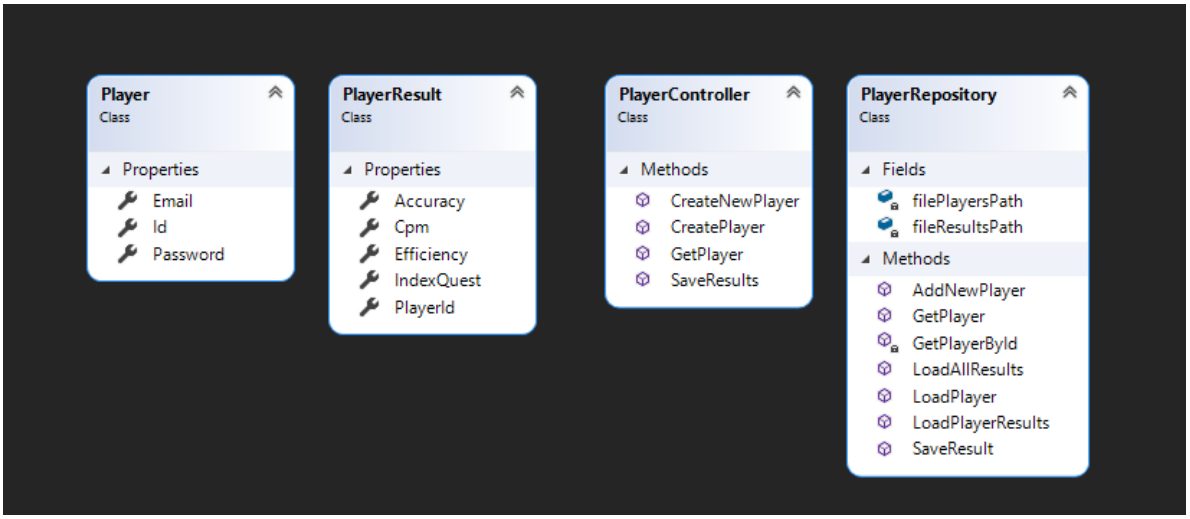

Рисунок 4.2. Діаграма класів для управління користувачами

Інформація про користувачів зберігається у форматі .xml

Проходження кожного квесту відстежується та здійснюється збереження результатів. Що відображається наступним чином:

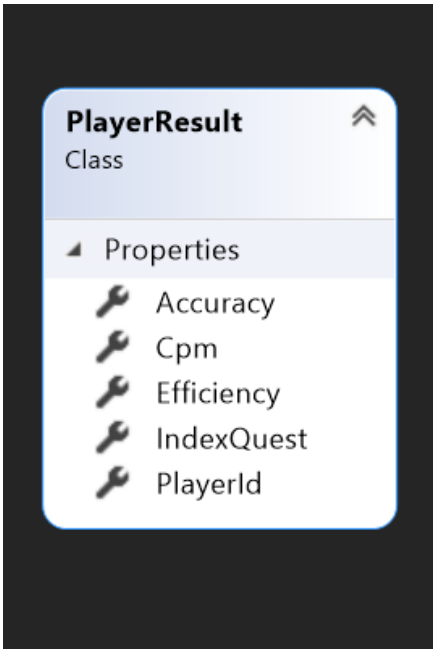

Рисунок 4.3. Результати проходження квестів

Користувацький інтерфейс представлений наступними вікнами:

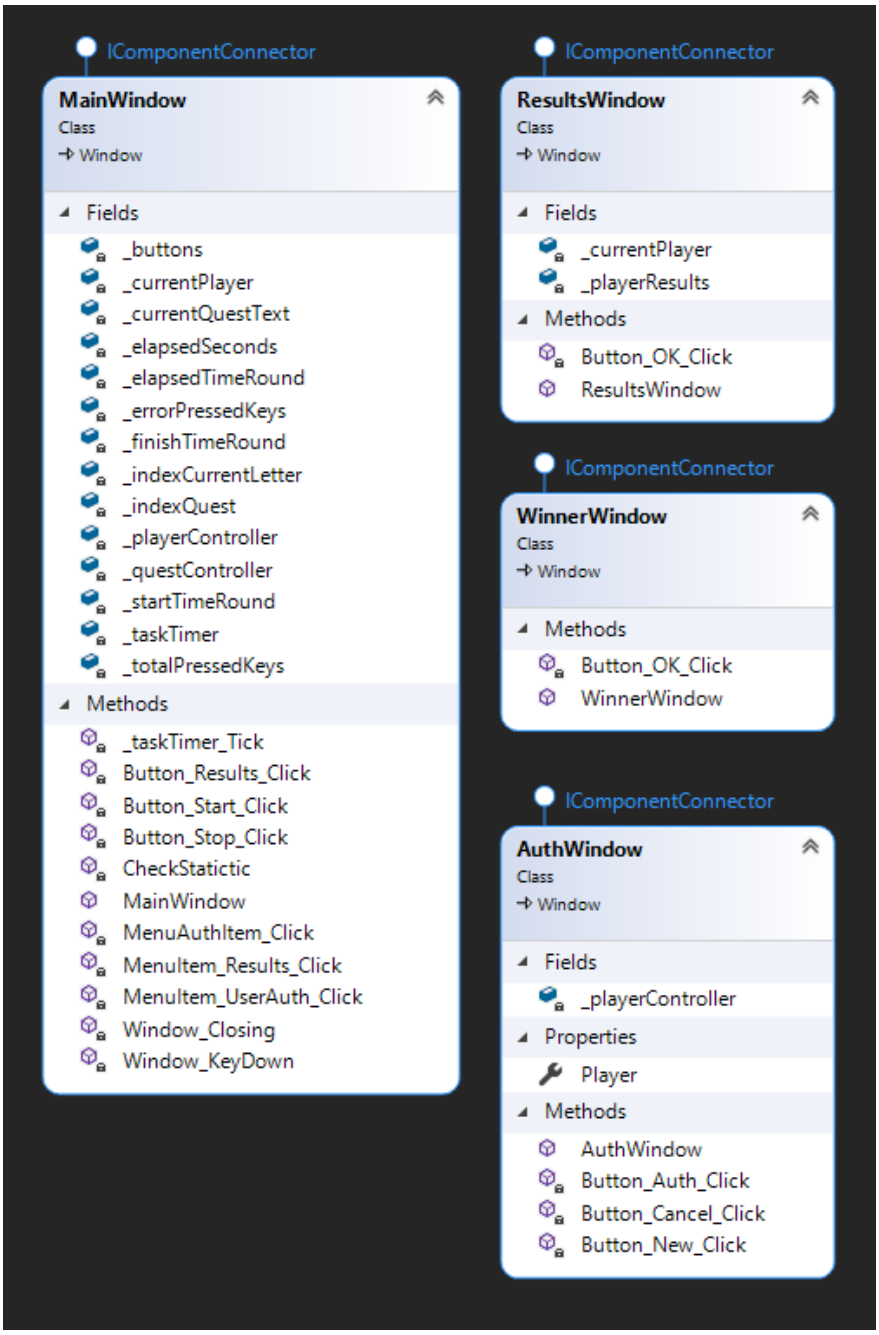

Рисунок 4.4. Користувацький інтерфейс застосунку

Отже, розроблений застосунок «Клавіатурний тренажер» реалізовано за допомогою WPF - платформи інтерфейсу користувача, яка не залежить від дозволу і використовує векторний механізм візуалізації, здатний використовувати всі переваги сучасного графічного обладнання. Водночас WPF надає комплексний набір функцій розробки додатків, які включають мову XAML, елементи управління, прив’язку до даних, макет, двовимірну і тривимірну графіку, анімацію, стилі, шаблони, документи, мультимедіа, текст і

типографічні функції.

4.2. Опис функціональних можливостей розробленого клавіатурного тренажера

На рисунку 4.5 представлено інтерфейс користувача застосунку «Клавіатурний тренажер». У застосунку є декілька ключових особливостей.

Для початку роботи із тренажером користувач повинен авторизуватися – ввести свою пошту та пароль.

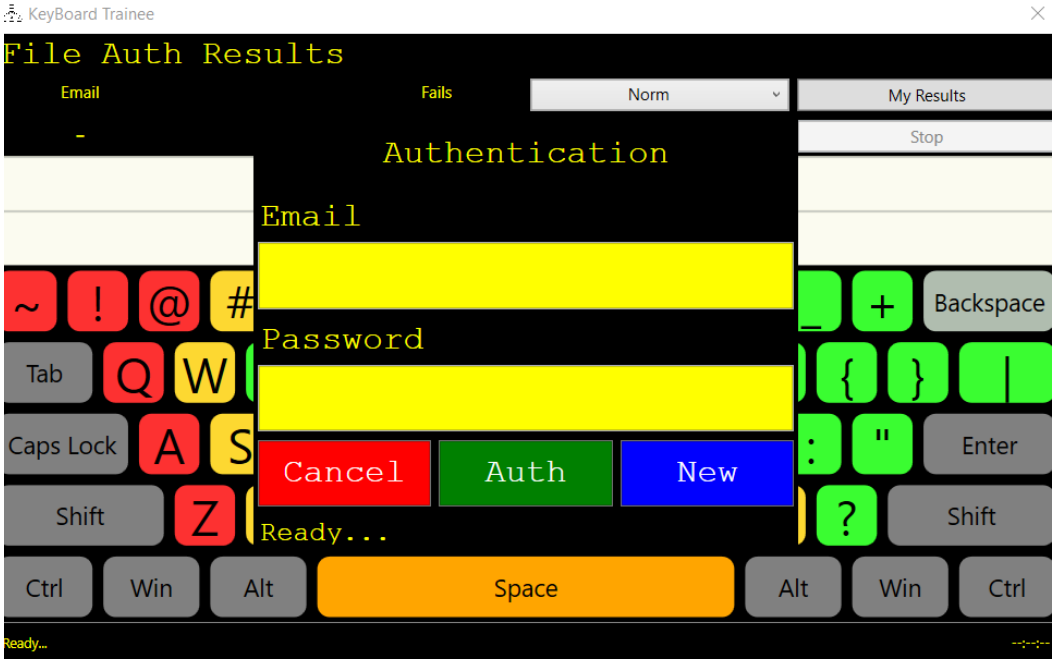

Рисунок 4.5. Авторизація користувачів

Завдання які можна виконувати відрізняються за рівнем складності.

Даний тренажер призначений для тренування швидкості та точності набору тексту на клавіатурі англійською мовою, що дозволить одночасно поглиблювати її знання і правильність написання слів.

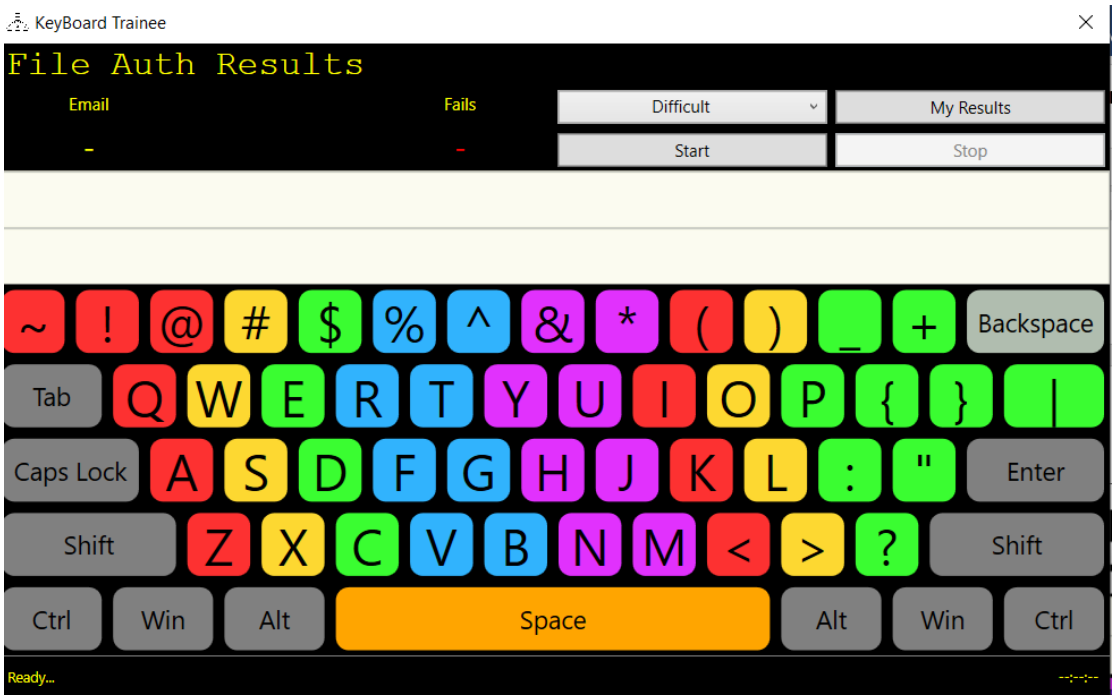

Рисунок 4.6. Інтерфейс користувача застосунку «Клавіатурний тренажер»

У застосунку передбачено три рівні складності. Є основі кнопки управління і клавіатура розділена на зони та «розфарбована» у різні кольори для швидкості орієнтації.

Після виконання завдань інформація про користувачів та результати зберігається у вкладці «Результат».

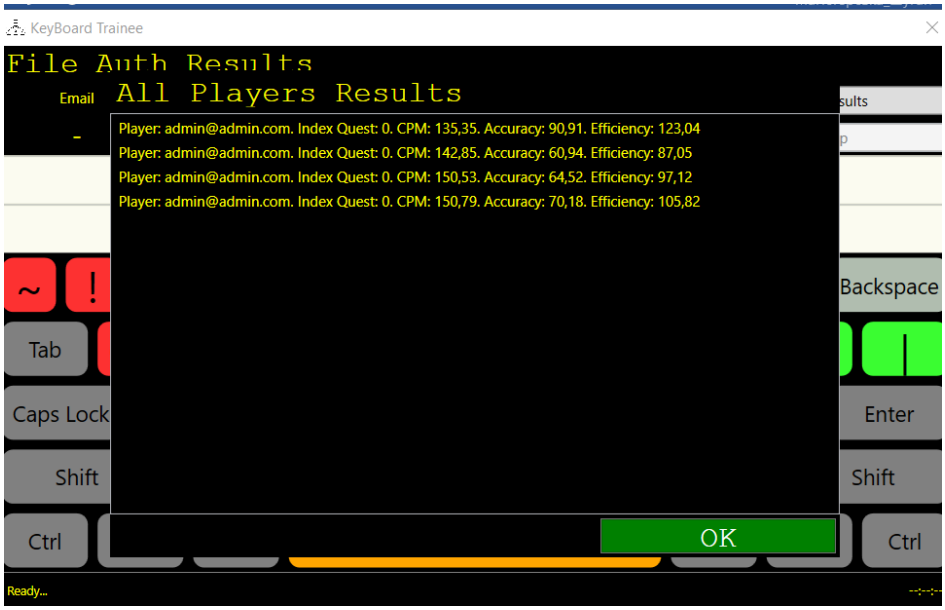

Рисунок 4.7. Результати виконання квестів

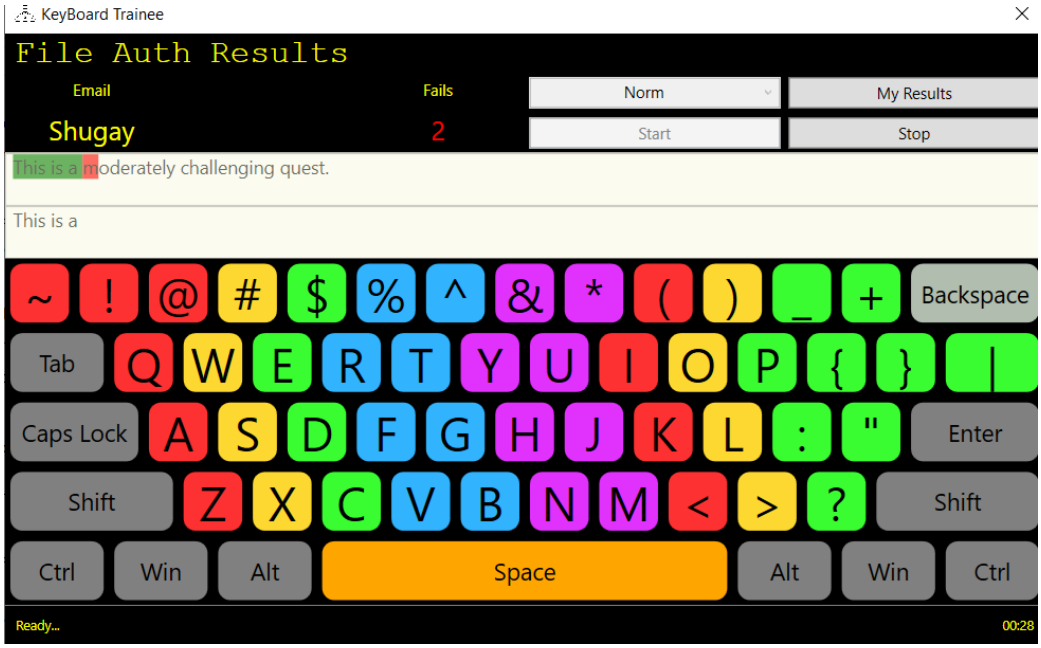

Рисунок 4.8. Процес тренування

Якщо під час проходження квесту користувач набере неправильну літеру чи символ, програма підсвітить помилку червоним кольором та зупинить набір тексту до натискання правильної кнопки.

ВИСНОВКИ

У процесі виконання дипломної роботи за темою «Розробка програмного забезпечення застосунку "Клавіатурний тренажер"» було проаналізовано теоретичні та практичні аспекти створення такого типу застосунків. Розробка програмного забезпечення клавіатурного тренажера є актуальним завданням, зважаючи на потребу вдосконалення навичок швидкого та точного набору тексту англійською мовою, особливо серед студентів комп’ютерних наук. В сучасному світі, де швидкість та ефективність обробки інформації мають вирішальне значення, оволодіння методами сліпого друку стає однією з ключових компетенцій у професійному розвитку фахівців.

У ході роботи було проаналізовано необхідні інструменти та технології для розробки тренажера. Особливу увагу було приділено використанню мови програмування C# у поєднанні з технологією Windows Presentation Foundation (WPF) для створення адаптивного та інтуїтивно зрозумілого інтерфейсу користувача.

Розроблений клавіатурний тренажер має на меті забезпечити користувачам гнучкий підхід до навчання, що включає вибір рівня складності текстів, інтеграцію системи авторизації та збереження даних про результати тренувань. Реалізовано механізми гейміфікації, зокрема систему квестів та нагород, що підвищує мотивацію користувачів до регулярних занять та сприяє більш ефективному засвоєнню навичок.

Отримані результати підтверджують доцільність використання запропонованих технологій та інструментів для створення сучасного клавіатурного тренажера. Виконаний проєкт може бути застосований як у навчальних закладах, так і для індивідуального використання. Розроблений застосунок забезпечує не лише розвиток техніки швидкого друку, але й дозволяє користувачам вдосконалювати навички роботи з комп’ютером, що є важливим фактором у контексті цифрової грамотності та професійної підготовки.

Загалом, проведене дослідження та розробка програмного забезпечення для клавіатурного тренажера демонструють, що застосування сучасних технологій дозволяє створювати ефективні та адаптивні інструменти для навчання. Подальший розвиток застосунку може включати інтеграцію з іншими освітніми платформами, розширення функціональності та використання машинного навчання для покращення адаптивності тренувань.

Отже, в ході виконання дипломного проєкту, було реалізовано всі поставлені завдання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Черненко О. О. Методичні рекомендації щодо виконання кваліфікаційної роботи студентів спеціальності 122 Комп'ютерні науки освітня програма «Комп'ютерні науки» ступеня магістра / С. В. Гаркуша, О. В. Ольховська, О. О. Черненко. – Полтава: ПУЕТ, 2023. – 68 с. Режим доступу: URL: http://elib.puet.edu.ua/action.php?kt_path_info=ktcore.SecViewPlugin.actions.document&fDocumentId=824868 - Назва з екрану.

2. Tutorial: Create a new WPF app with .NET. Режим доступу: URL: https://learn.microsoft.com/en-us/dotnet/desktop/wpf/get-started/create-app-visual-studio?view=netdesktop-8.0 - Назва з екрану.

3. Хохлов Д. та ін. XAML - мова розмітки інтерфейсу Silverlight додатків. Режим доступу: URL: http://www.znannya.org/?view=silverlight-intro1- Назва з екрану.

4. Introduction to Visual Studio. Режим доступу: URL: https://www.geeksforgeeks.org/introduction-to-visual-studio/ - Назва з екрану.

5. Madhu Murali. What is Visual Studio? All You Need to Know. Режим доступу: URL: https://blog.hubspot.com/website/what-is-visual-studio - Назва з екрану.

6. Meet Android Studio. Режим доступу: URL: https://developer.android.com/studio/intro - Назва з екрану.

7. The Benefits of Using Version Control Systems Like Git in Software Development. Режим доступу до ресурсу: URL: https://hojaleaks.com/the-benefits-of-using-version-control-systems-like-git-in-software-development – Назва з екрану.

8. What is a XAML file? Режим доступу: URL: https://docs.fileformat.com/web/xaml/ - Назва з екрану

9. What Is Agile? And When to Use It. Режим доступу: URL: https://www.coursera.org/articles/what-is-agile-a-beginners-guide - Назва з екрану.

10. WPF vs WinForms – Making the Right Decision in 2024. Режим доступу: URL: https://blog.ndepend.com/wpf-vs-winforms-choosing-the-proper-framework-for-your-project/ - Назва з екрану.

11. Євгенія Стенцель. Що таке C#? Кому підходить програмування на Сі Шарп? Режим доступу: URL: https://beetroot.academy/blog/shcho-take-c-chi-pidhodit-meni-cya-mova-programuvannya-chomu-vona-kruta - Назва з екрану.

12. Як вивчити мову програмування C# та стати .NET розробником. Режим доступу: URL: https://edu.cbsystematics.com/ua/blog/learn-csharp-become-dnet - Назва з екрану.

13. WPF Graphics Rendering Overview. Режим доступу: URL: https://learn.microsoft.com/en-us/dotnet/desktop/wpf/graphics-multimedia/wpf-graphics-rendering-overview?view=netframeworkdesktop-4.8- Назва з екрану.

ДОДАТОК А
Код програми

```
<?xml version="1.0" encoding="utf-8"?>
<ClassDiagram MajorVersion="1" MinorVersion="1">
  <Class Name="WpfAppKeyboardTraitee.PlayerEnv.Player">
    <Position X="1.25" Y="1.5" Width="1.5" />
    <TypeIdentifier>

<HashCode>AAACAAAAACAAAAAAAAAAAAAAAAAAAACAAAAAAAAAAAA=</Hash
Code>
      <FileName>PlayerEnv\Player.cs</FileName>
    </TypeIdentifier>
  </Class>
  <Class Name="WpfAppKeyboardTraitee.PlayerEnv.PlayerController">
    <Position X="5" Y="1.5" Width="1.5" />
    <TypeIdentifier>

<HashCode>AAAAAAEAAAAAAAAAEAEAAAAAAAAACAAAAAAAAAAAAAA=</HashC
ode>
      <FileName>PlayerEnv\PlayerController.cs</FileName>
    </TypeIdentifier>
  </Class>
  <Class Name="WpfAppKeyboardTraitee.PlayerEnv.PlayerRepository">
    <Position X="6.75" Y="1.5" Width="1.75" />
    <TypeIdentifier>

<HashCode>AgAAABAgACAAAAAAAAEAAAgAABAAAAAAABAAAAAAQAAA=</HashCo
de>
      <FileName>PlayerEnv\PlayerRepository.cs</FileName>
    </TypeIdentifier>
  </Class>
  <Class Name="WpfAppKeyboardTraitee.PlayerEnv.PlayerResult">
    <Position X="3" Y="1.5" Width="1.5" />
    <TypeIdentifier>

<HashCode>AABAAAAAAAAAAAAAAAAIAAAAAAAgAAAAAAAIAACAAAA=</HashCo
de>
      <FileName>PlayerEnv\PlayerResult.cs</FileName>
    </TypeIdentifier>
  </Class>
  <Class Name="WpfAppKeyboardTraitee.QuestEnv.Quest">
    <Position X="0.75" Y="4.75" Width="1.5" />
    <TypeIdentifier>

<HashCode>AAACAAAAAAAAAAAAAAAAAAAAAAAAAABAAAACAAAAAA=</Hash
Code>
      <FileName>QuestEnv\Quest.cs</FileName>
    </TypeIdentifier>
  </Class>
  <Class Name="WpfAppKeyboardTraitee.QuestEnv.QuestController">
```

```
    <Position X="2.75" Y="5" Width="2.25" />
    <TypeIdentifier>

<HashCode>CAAAAAAAAAAkAAAAAAAAAAgAAAAAgAAAAAAAAAAAABAA=</HashC
ode>
      <FileName>QuestEnv\QuestController.cs</FileName>
    </TypeIdentifier>
  </Class>
  <Class Name="WpfAppKeyboardTraitee.MainWindow">
    <Position X="9.5" Y="1" Width="2.5" />
    <TypeIdentifier>

<HashCode>EBACAAEAAACAAQQMEBAUAggAAAAFAAYQAAAAACAFAAA=</HashCo
de>
      <FileName>MainWindow.xaml.cs</FileName>
    </TypeIdentifier>
    <Lollipop Position="0.2" />
  </Class>
  <Class Name="WpfAppKeyboardTraitee.ResultsWindow">
    <Position X="12.25" Y="1" Width="2" />
    <TypeIdentifier>

<HashCode>AAAAAAAAAAAAAAAAAAAEAAAAQAAAAAAAQAAAAAAAAAA=</Hash
Code>
      <FileName>ResultsWindow.xaml.cs</FileName>
    </TypeIdentifier>
    <Lollipop Position="0.2" />
  </Class>
  <Class Name="WpfAppKeyboardTraitee.WinnerWindow">
    <Position X="12.25" Y="3.5" Width="2" />
    <TypeIdentifier>

<HashCode>AAAAAAAAAAAAAAAAAAAAAAAQAAAAAAAAAAAAAAAAAAA=</Hash
Code>
      <FileName>WinnerWindow.xaml.cs</FileName>
    </TypeIdentifier>
    <Lollipop Position="0.2" />
  </Class>
  <Class Name="WpfAppKeyboardTraitee.PlayerEnv.AuthWindow">
    <Position X="12.25" Y="5.5" Width="2" />
    <TypeIdentifier>

<HashCode>AgAAAAAAAAAAAAAEAAAAAAAAAAAAAAAAACQAAQAAAAA=</HashC
ode>
      <FileName>AuthWindow.xaml.cs</FileName>
    </TypeIdentifier>
    <Lollipop Position="0.2" />
  </Class>
  <Enum Name="WpfAppKeyboardTraitee.QuestEnv.DifficultyLevels">
    <Position X="0.75" Y="6.75" Width="1.5" />
    <TypeIdentifier>
```

```
<HashCode>AAAAAAAAIAAAAAAAAAAAAAAAAAAAAAAAAAAAAAgQAAAAA=</HashC
ode>
      <FileName>QuestEnv\DifficultyLevels.cs</FileName>
    </TypeIdentifier>
  </Enum>
  <Font Name="Segoe UI" Size="9" />
</ClassDiagram>

using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Threading;
using WpfAppKeyboardTraitee.PlayerEnv;
using WpfAppKeyboardTraitee.QuestEnv;

namespace WpfAppKeyboardTraitee
{
    public partial class MainWindow : Window
    {
        private QuestController _questController = null;
        private int _indexQuest = -1;                    // индекс текущей задачи
        private int _indexCurrentLetter = -1;            // индекс текущей задачи
        private string _currentQuestText = String.Empty;    // текущий квест

        private int _totalPressedKeys = 0;
        private int _errorPressedKeys = 0;

        private PlayerEnv.PlayerController _playerController = null;
        private PlayerEnv.Player _currentPlayer = null;

        private DispatcherTimer _taskTimer;
        private DateTime _startTimeRound;
        private DateTime _finishTimeRound;
        private TimeSpan _elapsedTimeRound;
        private int _elapsedSeconds;

        private List<Border> _buttons;

        public MainWindow()
        {
            InitializeComponent();
            _questController = new QuestController();

            _playerController = new PlayerEnv.PlayerController();

            _taskTimer = new DispatcherTimer();
            _taskTimer.Interval = new TimeSpan(0, 0, 1);
```

```
        _taskTimer.Tick += _taskTimer_Tick;
        _buttons = new List<Border>();

        foreach (var oneChild in Grig_Keyboard.Children)
        {
            if (oneChild is Border)
            {
                _buttons.Add((Border)oneChild);
            }
        }
    }

    private void _taskTimer_Tick(object sender, EventArgs e)
    {
        if (_taskTimer.IsEnabled)
        {
            _elapsedSeconds++;

            int minutes = _elapsedSeconds / 60;
            int seconds = _elapsedSeconds % 60;

            Label_Timer.Content = $"{minutes:D2}:{seconds:D2}";
        }
    }

    private void Button_Start_Click(object sender, RoutedEventArgs e)
    {

        if(_currentPlayer == null)
        {
            Label_Info.Content = "You need to authenticate";
            return;
        } else
        {
            Label_Info.Content = "Ready...";
        }
        if (_taskTimer.IsEnabled) return;

        _taskTimer.Start();
        _elapsedSeconds = 0;
        _startTimeRound = DateTime.Now;
        Button_Start.IsEnabled = false;
        Button_Stop.IsEnabled = true;
        ComboBox_DifficultyLevel.IsEnabled = false;
        _indexQuest++;                      // берем новую задачу
        _indexCurrentLetter = 0;            // выполняем задачу с первой буквы
        _currentQuestText = _questController.GetQuestByDifficultyLevel(_indexQuest,
(DifficultyLevels)ComboBox_DifficultyLevel.SelectedIndex).Text;
        RichTextBox_Quest.Document.Blocks.Clear();
        RichTextBox_Answer.Document.Blocks.Clear();


        RichTextBox_Quest.Document.Blocks.Add(new Paragraph(new Run(_currentQuestText)));
```

```
    }

    private void Button_Stop_Click(object sender, RoutedEventArgs e)
    {
        if (!_taskTimer.IsEnabled) return;

        _taskTimer.Stop();
        _finishTimeRound = DateTime.Now;
        _elapsedTimeRound = _finishTimeRound - _startTimeRound;
        //Title = _elapsedTimeRound.TotalSeconds.ToString();
        Button_Start.IsEnabled = true;
        Button_Stop.IsEnabled = false;
        TextBlock_Faills.Text = "-";
        RichTextBox_Quest.Document.Blocks.Clear();
        RichTextBox_Answer.Document.Blocks.Clear();
        ComboBox_DifficultyLevel.IsEnabled = true;
        MessageBox.Show("Test failed.Results will not be saved", "Test Failed",
MessageBoxButton.OK, MessageBoxImage.Exclamation);
    }

    private async void Window_KeyDown(object sender, KeyEventArgs e)
    {
        if (!_taskTimer.IsEnabled) return;

        int keyCode = Convert.ToInt32(e.Key);
        string keySymbol = e.Key.ToString();

        foreach (var oneButton in _buttons)
        {
            if (oneButton.Child is TextBlock)
            {
                if (((TextBlock)oneButton.Child).Text.ToLower().Equals(keySymbol.ToLower()))
                {
                    if (oneButton.Background != Brushes.FloralWhite) {
                        Brush tempB = oneButton.Background;
                        oneButton.Background = Brushes.FloralWhite;
                        await Task.Delay(75);
                        oneButton.Background = tempB;
                    }
                }
            }
        }

        if(keySymbol == "OemPeriod")
        {
            keySymbol = ".";
        }
        if (keySymbol == "OemComma")
        {
            keySymbol = ".";
        }
        if (Keyboard.GetKeyStates(Key.CapsLock) == KeyStates.None)
        {
```

```
            keySymbol = keySymbol.ToLower();
        }

        if (keySymbol.Length == 1 || keyCode == 18)
        {
            var endPosition = RichTextBox_Quest.CaretPosition.GetNextInsertionPosition(LogicalDirection.Forward);
            if(Keyboard.IsKeyDown(Key.LeftShift) || Keyboard.IsKeyDown(Key.RightShift))
            {
                keySymbol = keySymbol.ToUpper();
            }
            if(keyCode == 18)
            {
                keySymbol = " ";
            }
            TextRange textRange = new TextRange(RichTextBox_Quest.CaretPosition, endPosition);
            //если пользователь ввел символ правильно
            if (_currentQuestText[_indexCurrentLetter] == keySymbol[0])
            {
                textRange.ApplyPropertyValue(TextElement.BackgroundProperty, Brushes.Green);
                RichTextBox_Quest.CaretPosition = endPosition;
                RichTextBox_Answer.AppendText(keySymbol);
                _indexCurrentLetter++;
                if(_indexCurrentLetter == _currentQuestText.Length)
                {
                    CheckStatictic();
                }
            }
            else
            {
                //он ввел не правильный символ
                _errorPressedKeys++;
                TextBlock_Faills.Text = _errorPressedKeys.ToString();
                textRange.ApplyPropertyValue(TextElement.BackgroundProperty, Brushes.Red);
            }
        }
        _totalPressedKeys++;
    }

    private void CheckStatictic()
    {
        _taskTimer.Stop();
        _finishTimeRound = DateTime.Now;
        _elapsedTimeRound = _finishTimeRound - _startTimeRound;
        //Title = _elapsedTimeRound.TotalSeconds.ToString();

        double cpm = _totalPressedKeys / (_elapsedTimeRound.TotalSeconds / 60.0); // Обратите внимание на 60.0
        double accuracy = (1.0 - (double)_errorPressedKeys / (double)_totalPressedKeys) * 100; // Приведение к double
        double efficiency = cpm * accuracy / 100;
```

```
            _playerController.SaveResults(_currentPlayer,
(DifficultyLevels)ComboBox_DifficultyLevel.SelectedIndex, _indexQuest, cpm, accuracy,
efficiency);
            WinnerWindow winnerWindow = new WinnerWindow(_currentPlayer, cpm, accuracy,
efficiency);
            winnerWindow.Owner = this;
            winnerWindow.WindowStartupLocation = WindowStartupLocation.CenterOwner;
            winnerWindow.ShowDialog();
            _indexQuest++;
            TextBlock_Faills.Text = "-";
            Button_Start.IsEnabled = true;
            Button_Stop.IsEnabled = false;
            RichTextBox_Quest.Document.Blocks.Clear();
            RichTextBox_Answer.Document.Blocks.Clear();
        }

        private void MenuItem_UserAuth_Click(object sender, RoutedEventArgs e)
        {
            if (_taskTimer.IsEnabled)
            {
                MessageBox.Show("Stop the game process", "Warning", MessageBoxButton.OK,
MessageBoxImage.Warning);
                return;
            }
            this.Close();
        }

        private void Window_Closing(object sender, System.ComponentModel.CancelEventArgs e)
        {
            if (_taskTimer.IsEnabled)
            {
                MessageBox.Show("Stop the game process", "Warning", MessageBoxButton.OK,
MessageBoxImage.Warning);
                return;
            }
        }

        private void MenuAuthItem_Click(object sender, RoutedEventArgs e)
        {
            if(_currentPlayer == null)
            {
                AuthWindow authWindow = new AuthWindow(_playerController);
                authWindow.Owner = this;
                authWindow.ShowDialog();
                if(authWindow.Player != null)
                {
                    _currentPlayer = authWindow.Player;
                    TextBlock_Player_Email.Text = _currentPlayer.Email.Split("@")[0];
                    Label_Info.Content = "Ready to game";
                }
            }
        }
```

```
        private void Button_Results_Click(object sender, RoutedEventArgs e)
        {
            if (_currentPlayer == null)
            {
                Label_Info.Content = "You need to authenticate";
                return;
            }
            else
            {

                ResultsWindow resultsWindow = new ResultsWindow(_currentPlayer);
                resultsWindow.Owner = this;
                resultsWindow.WindowStartupLocation = WindowStartupLocation.CenterOwner;
                resultsWindow.ShowDialog();
            }
        }

        private void MenuItem_Results_Click(object sender, RoutedEventArgs e)
        {
            ResultsWindow resultsWindow = new ResultsWindow(null);
            resultsWindow.Owner = this;
            resultsWindow.WindowStartupLocation = WindowStartupLocation.CenterOwner;
            resultsWindow.ShowDialog();
        }
    }
}

using System;
using System.Collections.Generic;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
using WpfAppKeyboardTraitee.PlayerEnv;

namespace WpfAppKeyboardTraitee
{
    /// <summary>
    /// Interaction logic for ResultsWindow.xaml
    /// </summary>
    public partial class ResultsWindow : Window
    {
        private PlayerEnv.Player _currentPlayer;
        private List<PlayerEnv.PlayerResult> _playerResults;
        public ResultsWindow(PlayerEnv.Player currentPlayer)
        {
            _currentPlayer = currentPlayer;
            InitializeComponent();
```

```
        if (_currentPlayer == null)
        {
            var _playerResults = PlayerRepository.LoadAllResults();
            Label_Results.Content = $"All Players Results";

            // Вывод результатов
            foreach (var (oneResult, email) in _playerResults)
            {
                ListBox_Results.Items.Add($"Player: {email}. Index Quest: {oneResult.IndexQuest}. CPM: {oneResult.Cpm.ToString("F2")}. Accuracy: {oneResult.Accuracy.ToString("F2")}. Efficiency: {oneResult.Efficiency.ToString("F2")}");
            }
        }
        else
        {

            _playerResults = PlayerRepository.LoadPlayerResults(currentPlayer.Id);
            Label_Results.Content = $"Player: {currentPlayer.Email}";
            foreach (var oneResult in _playerResults)
            {
                ListBox_Results.Items.Add($"Index Quest: {oneResult.IndexQuest}. CPM: {oneResult.Cpm.ToString("F2")}. Accuracy: {oneResult.Accuracy.ToString("F2")}. Efficiency: {oneResult.Efficiency.ToString("F2")}");
            }
        }
    }

    private void Button_OK_Click(object sender, RoutedEventArgs e)
    {
        this.Close();
    }
  }
}

using System;
using System.Collections.Generic;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;

namespace WpfAppKeyboardTraitee
{
  /// <summary>
  /// Interaction logic for WinnerWindow.xaml
  /// </summary>
  public partial class WinnerWindow : Window
  {
```

```
        public WinnerWindow(PlayerEnv.Player _currentPlayer, double cpm, double accuracy, double
efficiency)
        {
            InitializeComponent();
            Label_PlayerEmail.Content = _currentPlayer.Email;
            Label_PlayerCPM.Content = "CPM: " +  cpm.ToString("F2");
            Label_PlayerAccuracy.Content = "Accuracy: " + accuracy.ToString("F2");
            Label_PlayerEfficiency.Content = "Efficiency: " +  efficiency.ToString("F2");
        }

        private void Button_OK_Click(object sender, RoutedEventArgs e)
        {
            this.Close();
        }
    }
}
```