Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська дипломна робота

на тему: «Розробка методу та програмних засобів системи тестування знань з іноземної мови»

Виконав: студент IV курсу
групи 4ПІ-20б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Монастирський О. М.
(прізвище та ініціали)
Керівник: к.т.н., доц. каф. ПЗ Войтко В. В.
(прізвище та ініціали)
Рецензент:, к.т.н., доц. каф. ОТ Снігур А. В.
(прізвище та ініціали)

Допущено до захисту

Зав. кафедри ПЗ _____
« 10 » червня 2024 р.

ВНТУ – 2024

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший (бакалаврський)
Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітньо-професійна програма «Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
О. Н. Романюк
« 12 » березня 2024 р.

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Монастирському Олександру Миколайовичу

1. Тема роботи – розробка методу та програмних засобів системи тестування знань з іноземної мови.

Керівник роботи: Войтко Вікторія Володимирівна, к. т. н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від 11 березня 2024 р. № 80.

2. Строк подання студентом роботи 3 червня 2024.

3. Вихідні дані до роботи: засоби та методи тестування знань іноземної мови, середовище розробки – Microsoft Visual Studio 2022, мова розробки – C#, операційна система – Windows 10, засіб створення та адміністрування бази даних – Microsoft SQL Server Management Studio 2019, хмарне сховище для зберігання бази даних – AWS RDS.

4. Зміст текстової частини: вступ; аналіз стану розвитку засобів системи тестування знань з іноземної мови та постановка задач дослідження; розробка методу, моделі та алгоритмів програмного продукту; розробка програмних засобів реалізації системи з використанням засобів тестування знань з іноземної мови; тестування програми; висновки; список використаних джерел; додатки; ілюстративна частина.

5. Перелік ілюстративного матеріалу: титульний слайд; актуальність розробки; мета, предмет та об'єкт дослідження; постановка задачі; новизна; практична цінність отриманих результатів; існуючі аналоги; аналіз аналогів; метод формування тестів та проходження тестування (ч.1); метод формування тестів та

проходження тестування (ч.2); блок-схема алгоритму методу формування тест
проходження тестування; модель системи на прикладі діаграми діяльності; бл
схема загального алгоритму роботи системи тестування знань з іноземної мо
розробка графічного інтерфейсу; використані технології; функціонал систе
демонстрація роботи; апробація та публікація результатів роботи; висно
фінальний слайд.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1-4	Войтко В. В., к.т.н., доцент кафедри ПЗ	13.03.24	03.06.24

7. Дата видачі завдання 13 березня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

Ч.ч.	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз стану розвитку засобів системи тестування знань з іноземної мови та постановка задач дослідження	14.03.24 – 24.03.24	Виконано
2	Розробка методу, моделі та алгоритмів програмного продукту	25.03.24 – 15.04.24	Виконано
3	Розробка програмних засобів реалізації системи з використанням засобів тестування знань з іноземної мови	15.04.24 – 30.04.24	Виконано
4	Тестування програми	01.05.24 – 10.05.24	Виконано
5	Оформлення матеріалів до захисту БДР	11.05.24 – 30.05.24	Виконано

Студент Sipticon **О. М. Монастирс**
(підпис) (ініціали та пр

Керівник бакалаврської дипломної роботи **В. В. Во**
(підпис) (ініціали та пр

АНОТАЦІЯ

Бакалаврська дипломна робота складається з 64 сторінок формату А4, на яких є 52 рисунка, 3 таблиці, список використаних джерел містить 24 найменування.

У бакалаврській дипломній роботі проведено детальний аналіз методів і засобів тестування знань з іноземної мови та існуючих аналогів. Описано об’єкт, предмет, завдання та методи дослідження. Сформульовано мету досліджень – підвищення якості тестування знань з іноземних мов, зокрема, з англійської, української, німецької, польської, іспанської, китайської та португальської мов, за рахунок розробки та використання спеціалізованої системи тестування знань, яка використовує різні типи тестів, що дозволить підвищити об’єктивність оцінювання знань за результатами тестування.

Розроблено програмні засоби реалізації системи з використанням новітніх засобів розробки програмного забезпечення. Розроблений застосунок призначений для комп’ютерної системи, яка здатна надавати користувачам можливість протестувати свої знання з іноземної мови.

Додаток розроблено з використанням мови C# та середовища Microsoft Visual Studio 2022. Для відображення застосунку використовувалась технологія WPF, база даних розташована у хмарному сховищі AWS RDS.

У результаті виконання бакалаврської дипломної роботи отримано навчальну комп’ютерну систему, що підвищить ефективність процесу вивчення іноземної мови та сприятиме зацікавленості учнів вивченням іноземної мови.

Отримані в бакалаврській дипломній роботі результати можна використати для тестування знань іноземної мови та впровадження в заклади освіти з метою покращення якості тестування.

Ключові слова: тестування, іноземна мова, C#, WPF, AWS RDS.

ABSTRACT

The bachelor's thesis consists of 64 A4 pages, which have 52 figures, 3 tables, the list of sources used contains 24 items.

A detailed analysis of methods and means of testing knowledge of a foreign language and existing analogues was carried out in the bachelor thesis. The object, subject, tasks and research methods are described. The goal of the research is formulated – improving the quality of testing knowledge of foreign languages, in particular, of English, Ukrainian, German, Polish, Spanish, Chinese and Portuguese languages, due to the development and use of a specialized knowledge testing system that uses different types of tests, which will increase the objectivity of knowledge assessment according to the test results.

Software tools for system implementation have been developed using the latest software development tools. The developed application is intended for a computer system capable of providing users with the opportunity to test their knowledge of a foreign language.

The application was developed using the C# language and the Microsoft Visual Studio 2022 environment. The WPF technology was used to display the application, the database is in the AWS RDS cloud storage.

As a result of completing the bachelor's thesis, an educational computer system was obtained, which increases the effectiveness of the foreign language learning process and the favorable interest of students in studying foreign language.

The results obtained in the bachelor thesis can be used for testing knowledge of a foreign language and implementation in educational institutions in order to improve the quality of testing.

Keywords: testing, foreign language, C#, WPF, AWS RDS.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Знання іноземних мов – це ключ до успіху в сучасному світі, де спілкування іноземними мовами та обробка величезних обсягів інформації набуває все більшого значення. Інтерес до вивчення мов традиційно великий, бо перефразовуючи відомий вислів, можна сміливо сказати, що той, хто володіє мовами, володіє світом.

Загалом людина, яка володіє мовами, – різнобічно розвинута особистість, володіє кращими здібностями до вивчення нового, вільніша та більш впевнена у спілкуванні з людьми [1]. Зі зростанням попиту на інноваційні технології, розробка систем, що допомагають вивчати та тестувати іноземні мови, стала популярним трендом. Однак розробка таких систем вимагає використання спеціалізованих програмних засобів, здатних впоратися зі складними завданнями, пов'язаними з обробкою даних.

Розробка програмних засобів для реалізації засобів системи тестування знань з іноземної мови є складним завданням, яке включає в себе кілька етапів розробки. Засоби повинні вміти ефективно управляти даними, обробляти вхідні дані і представляти їх користувачеві в цікавій і безперешкодній формі. Крім того, вони повинні бути сумісними з конкретними вимогами системи та пристрою користувача.

Вивчення іноземних мов за допомогою різних додатків уже давно стало мейнстримом. І не дивно, адже це зручний та дієвий спосіб прокачати знання безкоштовно або за невелику плату, займаючись коли і скільки вам комфортно [2]. Це зумовило потребу в програмних засобах, які можуть бути використані для реалізації таких систем. Ці програмні засоби можуть допомогти розробникам створювати засоби системи тестування знань з іноземної мови, надаючи користувачам захоплюючий і захоплюючий досвід.

Існує безліч аналогічних систем та засобів тестування знань з іноземної мови, проте вони є або лише в якості мобільного додатку, або мають складний інтерфейс.

До того ж, зазвичай потужні засоби є платними, тому актуальною є розробка власних засобів системи тестування знань з іноземної мови.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Метою бакалаврської дипломної роботи є підвищення якості тестування знань з іноземних мов, зокрема, з англійської, української, німецької, польської, іспанської, китайської та португальської мов, за рахунок розробки та використання спеціалізованої системи тестування знань, яка використовує різні типи тестів, що дозволить підвищити об'єктивність оцінювання знань за результатами тестування.

Робота передбачає розробку різних засобів, які будуть інтегровані в систему. Ці засоби включають управління даними, роботу з базою даних та графічним інтерфейсом. Крім того, буде розроблено засіб для відбору та відображення релевантної інформації щодо протестованих знань користувача.

Задачі дослідження:

– провести аналіз існуючих методів і засобів тестування знань з іноземної мови для визначення напрямків підвищення їх продуктивності;

– запропонувати новий метод формування тестів і проведення тестування знань іноземної мови;

– розробити модель системи для тестування знань з іноземної мови, яка включатиме основний функціонал застосунку;

– програмно реалізувати функціонал системи для тестування знань з іноземної мови;

– розробити програмні компоненти та систему візуалізації на основі запропонованого методу;

– провести експериментальні дослідження розроблених засобів тестування знань з іноземної мови.

Об’єкт дослідження – процес розробки системи тестування знань іноземної мови.

Предмет дослідження – методи та засоби реалізації системи тестування знань з іноземної мови.

Методи дослідження. У процесі досліджень використовувались:

– методи і засоби реалізації десктоп-застосунків мовою C# з використанням технології WPF для розробки логіки системи та графічного інтерфейсу користувача;

– методи побудови блок-схем алгоритмів, методи оптимізації коду для програмної реалізації застосунку;

– методи створення бази даних для проєктування та розробки сховища даних системи;

– методи захисту даних у AWS RDS для забезпечення захисту системи та даних користувачів;

– методи тестування для перевірки працездатності роботи програми.

Новизна отриманих результатів:

1. Подальшого розвитку набув метод формування тестів і проведення тестування знань з іноземної мови, який, на відміну від існуючих, реалізує персоналізований підхід до процесу тестування, дозволяє параметричний вибір тестів та режимів тестування, що надає користувачеві можливість самостійно створити необхідне тестування на основі власних потреб.

2. Подальшого розвитку набула модель тестування знань іноземної мови, яка, на відміну від існуючих, орієнтована на спрощення інтерфейсу до зрозумілого користувачеві, додавання великої кількості тем та підтем, а також впровадження тестування сімома різними мовами на вибір, що дозволяє покрити потреби користувача.

Практична цінність отриманих результатів. Практична цінність одержаних результатів полягає в тому, що розроблена система здатна надавати користувачам можливість тестувати власні знання з іноземної мови. Програмні

засоби, розроблені в роботі, можуть бути використані розробниками для створення комп’ютерних систем та дозволять впровадити їх у сферу освіти та інші сфери діяльності, які потребують тестування знань з іноземної мови.

Особистий внесок здобувача. Усі наукові результати, викладені у бакалаврській дипломній роботі, отримані особисто автором. У друкованій праці [3], опублікованій у співавторстві, автору належать такі результати: розробка методу та програмних засобів системи тестування знань з іноземної мови.

Апробація матеріалів бакалаврської дипломної роботи. Результати бакалаврської дипломної роботи доповідалися на LIII Науково-технічній конференції факультету інформаційних технологій та комп'ютерної інженерії (2024).

Публікації. За тематикою дослідження опубліковано 1 наукову працю у збірнику матеріалів LIII Науково-технічної конференції факультету інформаційних технологій та комп'ютерної інженерії (2024) [3].

Структура та обсяг роботи. Бакалаврська дипломна робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел, що містить 24 найменування, 4 додатків. Робота містить 52 ілюстрації та 3 таблиці.

1 АНАЛІЗ РОЗВИТКУ ЗАСОБІВ СИСТЕМИ ТЕСТУВАННЯ ЗНАНЬ З ІНОЗЕМНОЇ МОВИ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Аналіз стану засобів системи тестування знань з іноземної мови

Тестування залишається ефективним засобом організації контролю формування іншомовної комунікативної компетенції студентів закладів вищої освіти. Цікавість до тестування пояснюється тим, що воно значно підвищує ефективність навчального процесу, оптимально сприяє самостійності роботи кожного студента, а також є одним із засобів індивідуалізації в навчальному процесі. Тестування – це один із найсучасніших та найпоширеніших засобів перевірки знань іноземної мови в процесі навчання, а також водночас є навчальною справою і засобом контролю. Такий метод оцінювання знань студентів ефективний під час викладання іноземної мови [4].

Наразі існує безліч засобів тестування знань з іноземної мови, включаючи веб-сайти, мобільні застосунки та навчальні матеріали, такі як книги, журнали, статі тощо. Який вид обрати – залежить від потреб користувачів. Проте, в сучасному світі, більшість людей звертається до цифрових засобів, якими вони можуть користуватись у будь-який час. Наразі не існує єдиної унікальної системи тестування знань іноземної мови, яка підійшла б будь-якому користувачеві.

Ключовою проблемою є те, що майже всі програмні засоби, в основному, спрямовані на вивчення іноземних мов, а не тестування. Через цей недолік користувачі, кінцевою ціллю яких є саме тестування знань з іноземної мови, змушені використовувати інші джерела замість цифрових, що певною мірою ускладнює цей процес.

Щоб вирішити цю проблему, потрібно задовільнити потребу користувачів у системі, спрямованої саме на тестування знань іноземної мови. Попит на даний продукт у сучасних реаліях є чималим, адже наразі все більша кількість людей починає розширювати свої мовні кордони за допомогою вивчення нових іноземних мов.

Отже, розробка програмних засобів системи тестування знань з іноземної мови є критично важливим завданням у сучасному цифровому ландшафті. Використовуючи досвід спеціалізованих компаній з розробки програмного забезпечення, компанії можуть створювати інноваційні рішення, які покращують користувацький досвід та надають конкурентні переваги. Зважаючи на постійну жагу користувачів до вивчення та тестування знань з іноземної мови, важливо бути в курсі останніх розробок і тенденцій у цій галузі, щоб залишатися конкурентоспроможними та актуальними.

1.2 Порівняльний аналіз аналогів

У сучасному світі не обійтися без знання мов, адже колеги можуть перебувати у різних точках планети – від Японії до Бразилії чи Нової Зеландії. Володіння англійською – базова навичка для тих, хто хоче досягти успіху. А ще краще, коли людина може спілкуватися двома або навіть трьома іноземними мовами. Це складне завдання? Насправді це реально, якщо постійно здобувати нові знання [5]. Розробка програмних засобів системи тестування знань з іноземної мови може допомогти створити унікальний та цікавий користувацький досвід, підвищити залученість та утримання користувачів, а також запропонувати нові можливості для бізнесу для взаємодії зі своїми клієнтами. До найбільш відомих рішень відносять:

- Busuu;
- Drops;
- LingQ;
- Preply;
- Duolingo.

Одним із прикладів системи вивчення та тестування знань іноземної мови є популярна платформа Bussu [6] (рис. 1.1). Busuu – це платформа для вивчення мов в Інтернеті та на iOS і Android, яка дає користувачам змогу взаємодіяти з носіями мови.

Курси доступні 12 мовами: англійською, іспанською, французькою, німецькою, італійською, португальською, російською, польською, турецькою, арабською, японською та китайською. Учні вибирають одну або кілька з цих мов і виконують уроки самостійно, вивчаючи лексику й граматику. Після цього вони можуть практикувати свої знання в усних або письмових розмовах із носіями мови в спільноті Busuu [7].

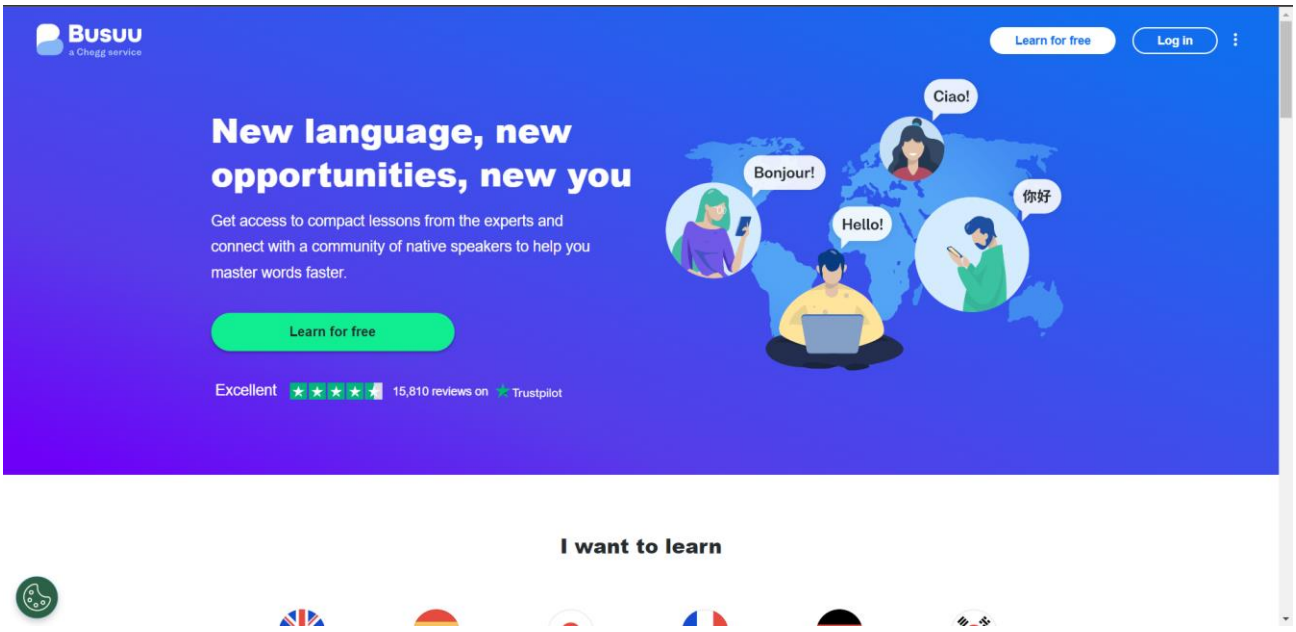

Рисунок 1.1 – Головна сторінка сайту платформи Busuu

Інший приклад – додаток Drops [8] (рис. 1.2). Drops – це програма для вивчення мови, створена в Естонії Даніелем Фаркасом і Марком Шульовським у 2015 році. Це другий продукт компанії після того, як у їх першому додатку LearnInvisible виникли проблеми зі збереженням активності користувача протягом необхідного періоду часу. Серед доступних мов – рідна гавайська та маорі, і вона була класифікована як одна з п’ятдесяти «Найінноваційніших компаній» 2019 року за версією Fast Company [9].

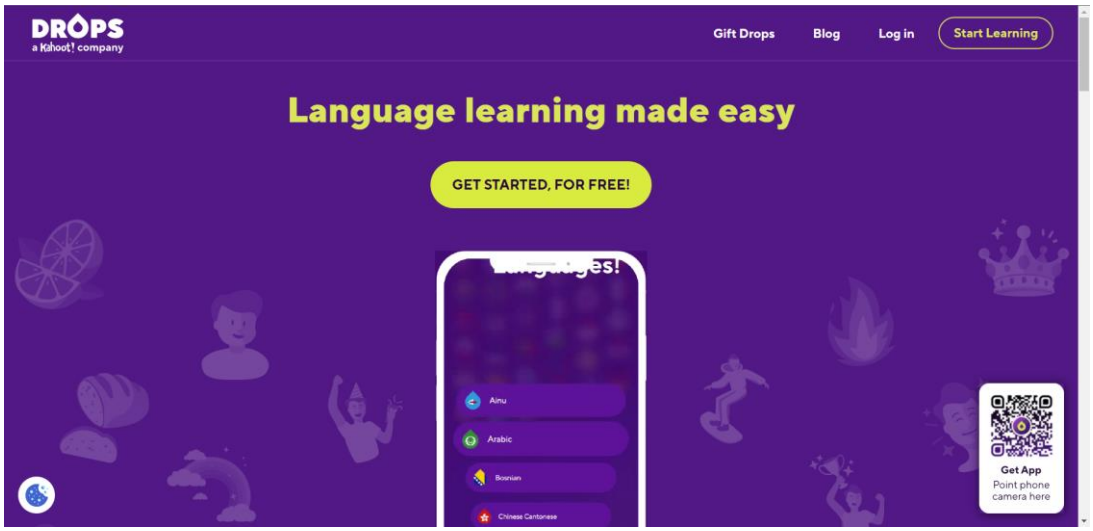

Рисунок 1.2 – Головна сторінка сайту платформи Drops

Наступним прикладом можна вважати систему LingQ [10] (рис. 1.3). Це онлайн-ресурс із великою бібліотекою аудіо- та текстових матеріалів, що пропонує користувачам інструменти для вивчення 24 іноземних мов. Долучитися до спільноти можна на сайті, а також завантаживши застосунок iOS або Android.

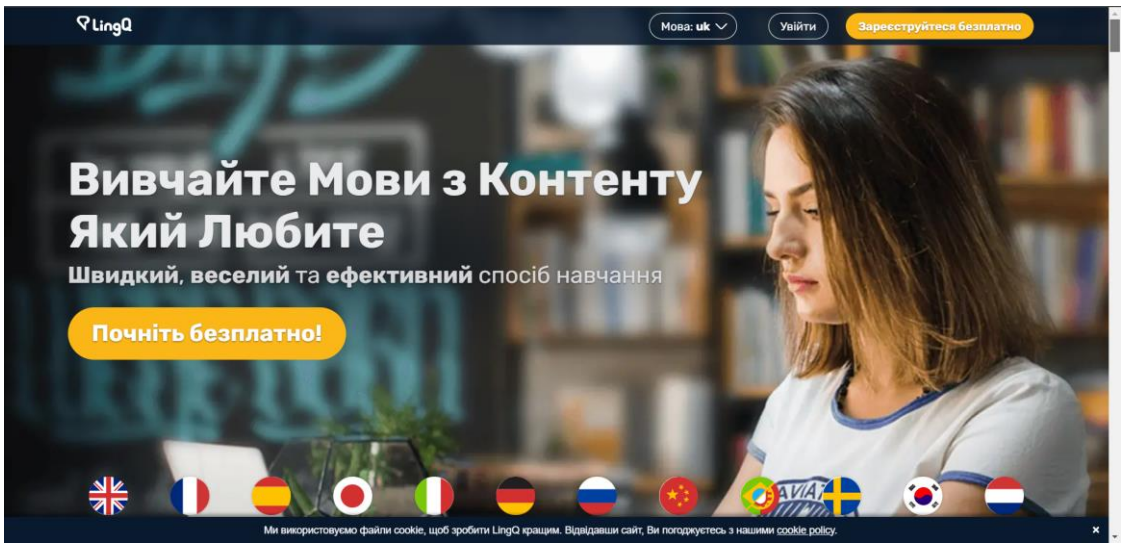

Рисунок 1.3 – Головна сторінка сайту платформи LingQ

Preply (рис. 1.4) — це освітня платформа, яка допомагає учням і викладачам зі всього світу знайти один одного на сайті [11]. Preply — це онлайн-платформа для вивчення мов. Вона дозволяє з'єднувати викладачів із учнями за допомогою

алгоритму машинного навчання. Платформа працює в 180 країнах світу, має близько 35 000 репетиторів і підтримує 50 мов.

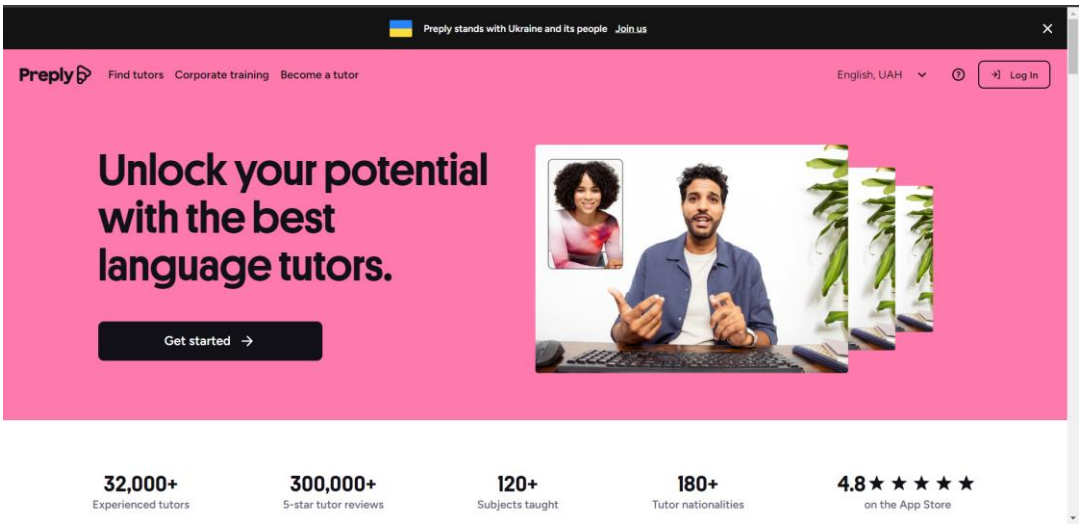

Рисунок 1.4 – Головна сторінка сайту платформи Preply

Один з найвідоміших сервісів, пов’язаних з вивченням та тестуванням іноземних мов, є Duolingo [12] (рис. 1.5). Це електронна платформа вивчення мови і краудсорсингового перекладу тексту, що має платний та безкоштовний види акаунтів. Сервіс розроблений так, що під час уроків користувачі допомагають перекладати веб-сайти, статті та інші документи [13].

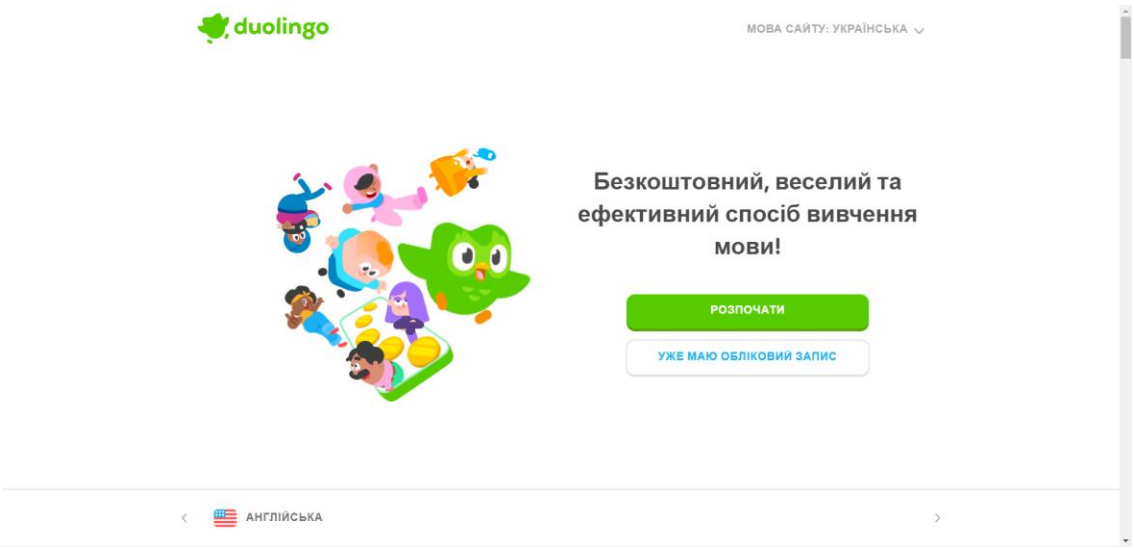

Рисунок 1.5 – Головна сторінка сайту платформи Duolingo

Розробка програмних засобів системи тестування знань з іноземної мови вимагає досвіду в галузі розробки програмного забезпечення та дизайну користувацького досвіду. Це передбачає створення внутрішньої системи, яка може розпізнавати та обробляти інтерактивні дії користувача з системою для конструювання подальших дій у роботі. Інтерфейсна система передбачає розробку користувацьких інтерфейсів, які використовують сучасні технологій відображення контенту для створення цікавого та інтуїтивно зрозумілого досвіду.

Результати порівняння аналогів зведено в табл. 1.1.

Таблиця 1.1 – Порівняльні характеристика онлайн платформ для вивчення іноземних мов

Критерії	Busuu	Drops	LingQ	Preply	Duolingo	Власна розробка – застосунок «Filian»
Функції тестування	-	-	-	-	-	+
Різноманітність тестів	+	+	+	+	+	+
Різноманітність мов	-	-	+	+	+	+
Простий інтерфейс	+	-	+	-	+	+
Наявність мобільного додатку	+	+	+	+	+	-
Загальна оцінка	60%	40%	80%	60%	80%	80%

Відповідно до таблиці порівняльних характеристик аналогів (табл.1.1) розробка власних програмних засобів системи тестування знань з іноземної мови є доцільною. Розроблений застосунок «Filian» зможе покрити недоліки існуючих рішень та забезпечити новий досвід та можливості тестування знань іноземної мови.

Отже, розробка програмних засобів системи тестування знань з іноземної мови відкриває перед бізнесом широкі можливості для взаємодії зі своїми клієнтами, створення унікального користувацького досвіду, а також підвищення рівня залученості та утримання користувачів. Розуміючи потенційні можливості застосування технології тестування знань іноземної мови та вимоги до розробки програмного забезпечення, компанії можуть створювати інноваційні та привабливі системи, які використовують технологію тестування знань іноземної мови для зростання та успіху.

1.3 Аналіз методів розв’язання задачі

Розробка програмних засобів системи тестування знань з іноземної мови вимагає ретельного розгляду найкращих підходів до вирішення проблеми. Існують різні методи, кожен з яких має свої переваги та недоліки. У цьому розділі проаналізовано найбільш ефективні методи розробки таких програмних модулів.

Одним з варіантів вирішення питання розробки програмних засобів системи тестування знань з іноземної мови є створення окремої бібліотеки, яка дозволяє інтегрувати модуль в існуючі рішення. Однак такий підхід має певні недоліки, хоча це швидкий спосіб розробки модуля, перебудовування архітектури системи та відв’язку даних від інтерфейс. Це може ускладнити роботу на системою. Тому такий підхід не варто застосовувати.

Інший підхід полягає в створенні окремого сервісу для інтеграції модуля у вже існуючі системи. Цей метод дозволяє розробникам сконцентруватися на розробці самого модуля, не витрачаючи часу на створення повноцінної онлайн-платформи. Користувачі можуть легко інтегрувати готове рішення у свої системи,

що сприяє зменшенню ресурсів, необхідних для інтеграції, порівняно з попереднім підходом. Цей підхід також включає розробку внутрішньої системи авторизації для захисту даних онлайн-платформ, які прагнуть інтегрувати цей модуль. Однак для ефективної роботи цей сервіс може вимагати значних апаратних ресурсів, особливо при інтеграції з багатьма сервісами. Таким чином, цей підхід вважається ефективним лише у випадку, якщо його можна швидко і стабільно масштабувати.

Найпоширенішим рішенням є розробка власної онлайн-платформи з вбудованими засобами системи тестування знань з іноземної мови. У цьому випадку немає необхідності адаптувати систему під різні потреби сторонніх платформ. Більше того, в результаті такого методу розробки виходить унікальна платформа, яка містить унікальний функціонал. Однак такий підхід має і свої недоліки, оскільки вимагає повноцінної розробки платформи.

Розглянувши переваги та недоліки кожного методу, було вирішено використати метод розробки власної платформи з вбудованими засобами системи тестування знань з іноземної мови. Такий підхід дозволить отримати повноцінний програмний продукт без необхідності використання потужного апаратного забезпечення. Більше того, такий метод розробки дозволить створити унікальну платформу з унікальним функціоналом.

1.4 Постановка задач для розробки системи тестування знань іноземної мови

Проаналізувавши переваги та недоліки існуючих програмних засобів системи тестування знань з іноземної мови, було визначено завдання, які необхідно виконати для розробки системи:

– провести аналіз існуючих методів і засобів тестування знань з іноземної мови для визначення напрямків підвищення їх продуктивності;

– запропонувати новий метод формування тестів і проведення тестування знань іноземної мови;

– розробити модель системи для тестування знань з іноземної мови, яка включатиме основний функціонал застосунку;

– програмно реалізувати функціонал системи для тестування знань з іноземної мови;

– розробити програмні компоненти та систему візуалізації на основі запропонованого методу;

– провести експериментальні дослідження розроблених засобів тестування знань з іноземної мови.

1.5 Висновки

У першому розділі було розглянуто стан засобів системи тестування знань з іноземної мови. Було проведено порівняння відомих платформ тестування знань з іноземної мови, таких як: Busuu, Drops, LingQ, Preply, Duolingo.

У результаті порівняння було виявлено, що досліджувані платформи набули значної популярності серед користувачів завдяки своїй зручності, доступності та спрямованості саме на вивчення іноземних мов. Проаналізувавши переваги та недоліки відомих платформ, було виявлено, що вони не надають можливості створення тесту за необхідними параметрами. Таким чином було доведено доцільність розробки власного програмного рішення.

Здійснено постановку задач та обрано метод розробки власної платформи з вбудованими засобами системи тестування знань з іноземної мови.

2 РОЗРОБКА МЕТОДУ, МОДЕЛІ ТА АЛГОРИТМІВ ПРОГРАМНОГО ПРОДУКТУ

2.1 Аналіз вхідних даних системи

Для реалізації програмних засобів системи тестування знань з іноземної мови будуть використані мова програмування C#, конструктор вікон WPF, хмарне сховище AWS RDS та середовище розробки Microsoft Visual Studio.

C# – мова для розробки програм з ООП, безпечною системою введення тексту від платформи .NET, створена Андерсом Гейлсбергом, Скоттом Вілтамутом і Пітером Голдом у компанії Microsoft. Особливість написання коду на C# подібний до C++ та Java. Мова має статичну типізацію та має поліморфізм, перевантаження операторів, покажчики на функції-члени класу, атрибути, події, властивості, винятки та коментарі у стилі XML [14].

Windows Presentation Foundation (WPF)– графічна (презентаційна) підсистема (аналог WinForms), яка починаючи з .NET Framework 3.0 в складі цієї платформи. Це перше реальне оновлення технологічного середовища призначеного для користувача інтерфейсу з часу випуску Windows 95. Воно включає нове ядро для заміни GDI і GDI+, використовувані в Windows Forms. WPF є високорівневим об'єктно-орієнтованим функціональним шаром, що дозволяє створювати двовимірні та тривимірні інтерфейси [15].

Amazon Relational Database Service (або Amazon RDS) – це розподілена реляційна база даних (РБД), яка надається як сервіс від Amazon Web Services (AWS). Є вебсервісом, який виконується у «хмарі». Спеціально розроблений для спрощення установки, простоти обслуговування та легкого масштабування РБД в застосунках [16].

Microsoft Visual Studio – серія продуктів фірми Майкрософт, які містять інтегроване середовище розробки програмного забезпечення та низку інших інструментальних засобів. Ці продукти дають змогу розробляти як консольні програми, так і програми з графічним інтерфейсом, включно з підтримкою

технології Windows Forms, а також вебсайти, вебзастосунки, вебслужби як у рідному, так і в керованому кодах для всіх платформ, що підтримуються Microsoft Windows, Windows Mobile, Windows Phone, Windows CE, .NET Framework, .NET Compact Framework та Microsoft Silverlight [17]. Поєднання цих сучасних засобів розробки програмного забезпечення дозволять створити програмний засіб, які повністю виправдає очікування користувача.

Розробка програмних засобів системи тестування знань з іноземної мови передбачає кілька етапів. По-перше, потрібно створити засіб, який може відображати усі дані, отримані з хмарного сховища. Цей засіб буде відповідати за звернення до бази даних у хмарному сховищі та отриманні саме необхідної користувачу інформації. Далі буде розроблений засіб для відображення контенту. Він використовуватиме технологію WPF, для простої та зручної взаємодії користувача з системою.

Для збереження усіх даних користувача, а саме результату тестувань, буде створено модуль облікового запису користувача, з інформацією про нього та пройдене ним тестування. Також необхідні засоби для реєстрації та авторизації користувача у систему, для забезпечення безпеки даних.

Розробка цих програмних засобів вимагає глибокого розуміння реляційних баз даних, графічного програмування та програмування на C#. Для реалізації цих засобів будуть використані різноманітні інструменти та бібліотеки програмування, зокрема Microsoft Visual Studio, WPF та AWS RDS. Програмні засоби будуть протестовані та доопрацьовані, щоб забезпечити безперебійну роботу в тестуванні знань іноземної мови для користувачів.

Отже, розробка програмних засобів системи тестування знань з іноземної мови на C#, WPF, AWS RDS та Microsoft Visual Studio є комплексним завданням, яке включає кілька етапів. Засоби повинні бути ретельно спроектовані, реалізовані та протестовані, щоб забезпечити якісне проходження тестування. Однак, маючи правильні інструменти та досвід, можна створити систему, яка може революціонізувати спосіб тестування знань іноземної мови.

2.2 Розробка моделі системи

Для кращого розуміння роботи програмних засобів системи тестування знань з іноземної мови було вирішено розробити модель роботи системи на прикладі UML діаграми діяльності (рис. 2.1). Згідно вказаної діаграми першим ділом користувач повинен ввійти в обліковий запис або зареєструватися. Наступним кроком користувач повинен обрати мову для тестування. Перед початком тестування потрібно обрати тип тестування. Далі користувач обирає тему для тестування, підтеми та розпочинає безпосередньо саме тестування за обраними параметрами.

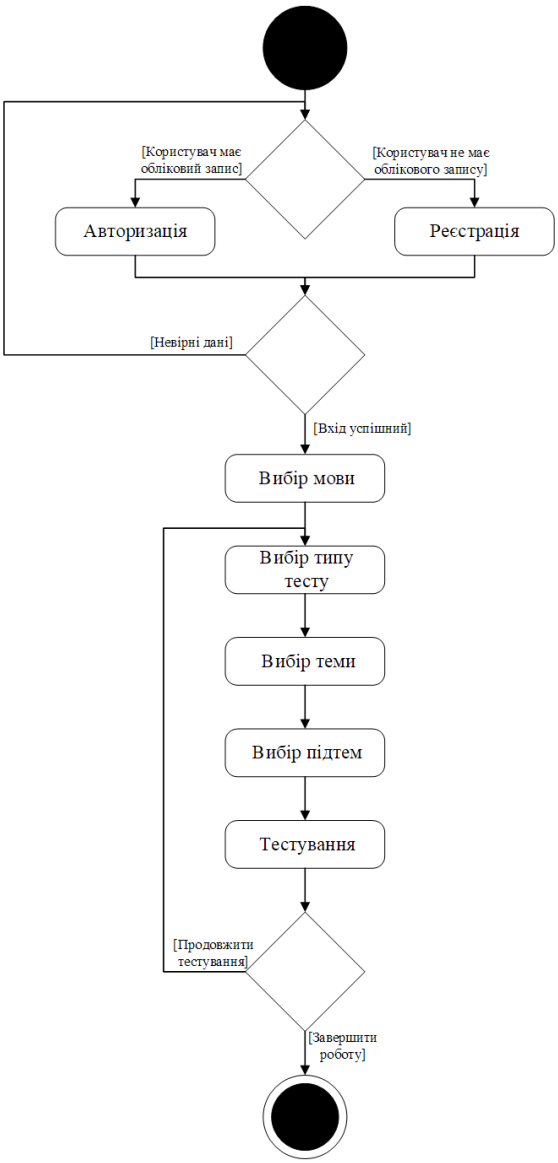

Рисунок 2.1 – Модель системи у вигляді діаграми діяльності

Діаграма діяльності системи показує загальну роботу програми на високому рівні абстракції для легшого розуміння робочих процесів.

2.3 Розробка методу формування тестів та проходження тестування

Ключовою особливістю розроблюваної системи є надання користувачеві можливості створення тестування за власними потребами, що являється перевагою над проаналізованими аналогами.

Метод формування та проходження тестування дозволяє обрати мову, тип та тему тесту, створюючи унікальні набори та комбінації ґрунтуючись на потребах та вподобаннях користувачів.

Метод включає такий функціонал:

1. Після авторизації користувача, програма одразу запропонує користувачеві обрати одну з семи наявних мов для тестування. Для цього клас LanguagesViewModel, попередньо приєднавшись до бази даних за допомогою класу SqlConnection, створює запит для отримання наявних мов та їх параметрів. Отримані за допомогою класу SqlDataReader записи додаються в колекцію ObservableCollection<Language> у вигляді класу Language. Потім за допомогою форми розмітки, створеної за допомогою технології WPF, створюється колекція класів LanguagesView для відображення отриманих мов з бази даних користувачеві.

2. Користувач обирає необхідну мову для тестування та натискає кнопку «Apply».

3. За попереднім принципом клас TestsViewModel робить запит на отримання наявних типів тестів та, після їх отримання, записує екземпляри класів Test у колекцію ObservableCollection<Test>. За допомогою форми розмітки TestsView створює набір отриманих типів тестів та відображає їх користувачеві.

4. Далі користувачеві необхідно обрати бажаний тип тестування та продовжити.

5. Клас ThemesViewModel також формує запит у базу для отримання інформації про існуючі теми, записує їх в колекцію ObservableCollection<Theme> у вигляді набору класів Theme. Після цього користувач отримує відображення всіх наявних тем, які створені за допомогою класу ThemesView.

6. Тепер перед користувачем стоїть вибір теми для тестування з двадцяти наявних. Після вибору рухаємося далі.

7. Для формування колекції підтем клас UnderThemesViewModel надсилає запит у базу даних з параметром у вигляді id обраної користувачем теми, адже в кожної теми підтеми різні та у різній кількості. Клас UnderThemesView відображає кожну підтему окремо, складається в колекцію UnderThemes та відображається в графічному інтерфейсі.

8. На даному етапі користувач має можливість обрати одразу декілька підтем та розпочати тестування.

9. Вибір параметрів завершено, початок тестування.

10. Ґрунтуючись на обраних властивостях, головний клас програми MainViewModel створює чергу Queue<CurrentTestTypeInfo> з тестами, обираючи відповідну графічну форму, та відображає її користувачеві.

11. Користувач обирає відповідь та натискає кнопку «Check».

12. Програма сповіщає користувача про правильність відповіді.

13. Користувач натискає кнопку «Next».

14. Відображається наступний тест з черги.

15. Користувач проходить тестування поки черга тестів не стане пустою.

16. Кількість тестів в черзі = 0.

17. Програма формує звіт про проходження тестування за допомогою графічного відображення UserNotificationView та виводить його користувачеві.

18. Користувач переглядає повідомлення та натискає кнопку «ОК».

19. Закінчення тестування.

Описаний метод включає кроки приєднання до бази даних, формування та надсилання запитів, створення відповідних колекцій та графічних відображень,

вибір користувача, отримання створених тестів з черги, відображення результату користувачеві. Алгоритм роботи цього методу наведено на рисунку 2.3.

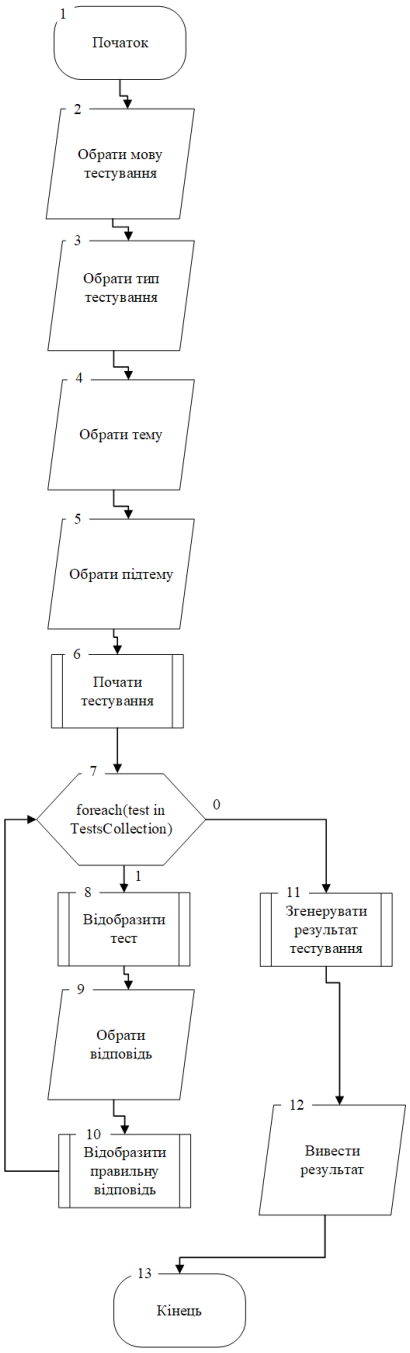

Рисунок 2.3 – Блок-схема алгоритму роботи методу формування тестів та проходження тестування

Розглянувши запропонований алгоритм, можна зрозуміти, що розроблений метод дає змогу користувачам самим керувати та обирати дані для тестування. Алгоритм дозволяє повністю автоматизувати систему тестувань, що робить

розроблений додаток найкращим вибором для тих, хто хоче протестувати свої знання іноземної мови.

2.4 Розробка загального алгоритму роботи системи

Для розробки засобів системи тестування знань з іноземної мови, необхідно розробити загальний алгоритм, згідно якого буде працювати додаток (рис. 2.3).

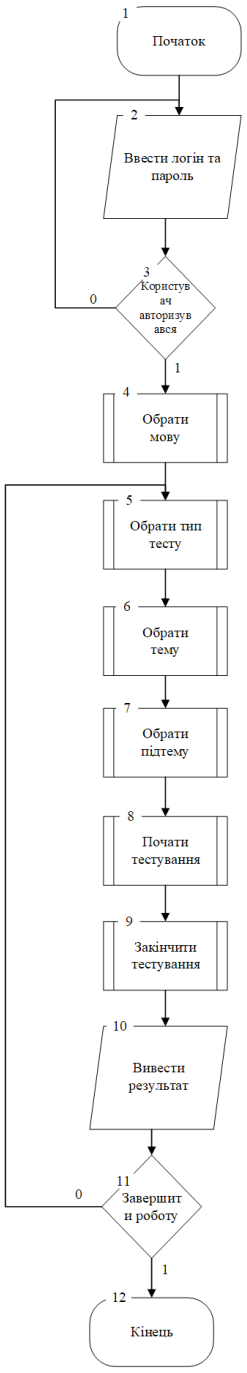

Рисунок 2.3 – Загальний алгоритм роботи системи

Згідно даного алгоритму спочатку користувач має авторизуватися в системі. Тут відбувається перевірка на коректність введених даних. Далі користувачеві надається можливість обрати тип тестування. Після цього він може обрати тему та підтеми. Далі розпочинається тестування. Після проходження тестування виводиться результат та перевіряється вибір користувача, вийти з системи чи продовжити тестування далі. Отже, після входу в систему, запускається засіб тестування знань з іноземної мови, заснований на виборі та діях користувача.

Запропонований алгоритм спрямований на ілюстрацію засобів тестування знань іноземної мови на високому, тобто абстрактному рівні. З його допомогою будь-хто зможе окреслити межі проєкту, та зрозуміти, яким чином працює дана система, без допоміжних знань у проєктуванні архітектури систем або програмуванні такого типу засобів.

2.5 Висновки

У другому розділі було проведено аналіз вхідних та вихідних даних програмного продукту.

На прикладі UML діаграми діяльності розроблено модель роботи системи, яка описує основний функціонал системи. Також розроблено метод формування тестів та проходження тестування.

Наведено блок-схеми алгоритму роботи розробленого методу та загального алгоритму роботи системи.

3. РОЗРОБКА ПРОГРАМНИХ ЗАСОБІВ РЕАЛІЗАЦІЇ СИСТЕМИ З ВИКОРИСТАННЯМ ЗАСОБІВ ТЕСТУВАННЯ ЗНАНЬ З ІНОЗЕМНОЇ МОВИ

3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного забезпечення

Розробляючи програмні засоби для системи, яка використовує інструменти тестування знань іноземної мови, важливо ретельно обирати технології, що використовуються.

C# – популярна мова програмування для розробки додатків під різні платформи від компанії Microsoft. Завдяки можливостям, які надає дана мова, програміст може створити будь-що, починаючи від ігор і закінчуючи веб-сайтами.

WPF – сучасна технологія від компанії Microsoft, яка надає можливості розробникам легко створювати графічний інтерфейс для десктопних застосунків використовуючи мову розмітки XAML. Мова XAML безпосередньо представляє створення екземплярів об'єктів в конкретному наборі резервних типів, визначених у збірках. У цьому полягає її відмінність від більшості інших мов розмітки, які, як правило, є інтерпретованими мовами без прямого зв'язку з системою резервних типів [18].

Windows Presentation Foundation (WPF) – графічна (презентаційна) підсистема (аналог WinForms), яка починається з .NET Framework 3.0 в складі цієї платформи. Це перше реальне оновлення технологічного середовища, призначеного для користувача інтерфейсу з часу випуску Windows 95. Середовище включає нове ядро для заміни GDI і GDI+, використовувані в Windows Forms. WPF є високорівневим об'єктно-орієнтованим функціональним шаром, що дозволяє створювати двовимірні та тривимірні інтерфейси [15]. Така технологія забезпечує повний контроль над створенням графічного інтерфейсу користувача починаючи з створення каркасної демоверсії та закінчуючи повноцінним графічним інтерфейсом з розміткою та стилями.

Amazon Relational Database Service (або Amazon RDS) – це розподілена реляційна база даних (РБД), яка надається як сервіс від Amazon Web Services (AWS). Є вебсервісом, який виконується у «хмарі». Спеціально розроблений для спрощення установки, простоти обслуговування та легкого масштабування РБД в застосунках [16]. Завдяки цьому сервісу, який пропонує компанія Amazon, користувачі можуть бути певними, що їх дані у безпечному сховищі та надійно захищені, а розробники – що система, яка використовує даний сервіс, не матиме збоїв при читанні та записі даних.

Поєднання цих технологій дозволяє розробити систему, яка забезпечує унікальний і захоплюючий досвід тестування знань іноземної мови. C# та WPF забезпечують чисту та ефективну кодову базу, в той час як AWS RDS забезпечує надійне зберігання даних у хмарному сховищі та безпеку цих даних.

Отже, вибір технологій для розробки програмних модулів для системи тестування знань іноземної мови в C#, WPF, AWS RDS має вирішальне значення для створення надійного і захоплюючого досвіду. Поєднання цих технологій надає необхідні інструменти та фреймворки для розробки ефективної, надійної та простої в обслуговуванні системи.

3.2 Розробка архітектури та логіки системи

Проте, в будь-якому програмному забезпеченні головна роль дістається саме логіці, тобто бек-енду програми. Без цього графічний інтерфейс буде лише картинкою, яка не виконує жодної функції, а отримані дані з хмарного середовища зберігатимуться в системі без змоги бути використаними. Розроблювана система тестування знань з іноземної мови не є винятком цього правила, а навпаки – яскравим підтвердженням цього факту.

Безумовно, головним компонентом розроблюваної системи являється саме створення та проходження тестування користувачем. У даному випадку, інтерфейс та дані з хмарного середовища являються лише допоміжними компонентами,

якими керує логіка програми. Проте, разом ці компоненти утворюють повноцінну систему.

Програмний продукт розроблявся за шаблоном проєктування Model-View-ViewModel (MVVM) [19]. Даний шаблон дозволяє розділити програму на три окремо розроблюваних частини, які можна змінювати без необхідності додавати зміни в іншу частину. Model – є представленням даних, які отримуються з хмарного середовища. View – графічний інтерфейс. ViewModel – логіка системи, компонент який слугує містком між даними та графічним інтерфейсом, та діє так, що поєднуванні компоненти не знаю про існування один одного. За допомогою даного принципу, архітектура системи стає зрозумілою та легко підтримуваною, що однозначно є перевагою.

Для зручності розробки, було створено папки в проєкті (рис. 3.1), які будуть класифікувати файли на належність до одного з компонентів патерну.

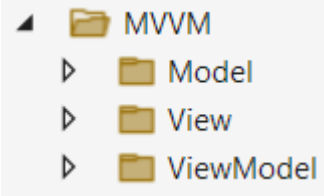

Рисунок 3.1 – Папки для класифікації файлів шаблону MVVM

Даний підхід надає зручність користування програмним середовищем під час розробки програмного продукту, адже не потрібно шукати якийсь компонент серед безлічі за іменами.

Для класів, які належать до model та viewmodel, було створено батьківські класи, які містять спільні поля та методи, а також дозволяються створювати подібні файли з невеликою різницею.

Таким чином, для model як для даних з хмарного середовища батьківським класом є DataBaseEntity (рис. 3.2).

Цей клас містить поля, які можуть зберігати шлях до зображення, ідентифікаційний номер, слово та переклад. Усі існуючі класи в компоненті model

засновані на цих полях, адже, в залежності від вибору користувача, міститимуть різні значення, які згодом будуть відображенні у графічному інтерфейсі.

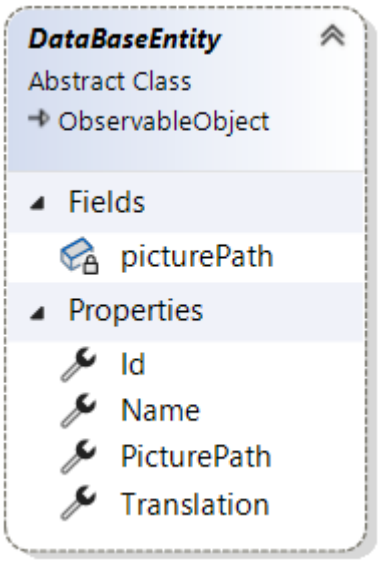

Рисунок 3.2 – Діаграма класу DataBaseEntity

Класи типу viewmodel наслідуються від класу ViewModel (рис. 3.3).

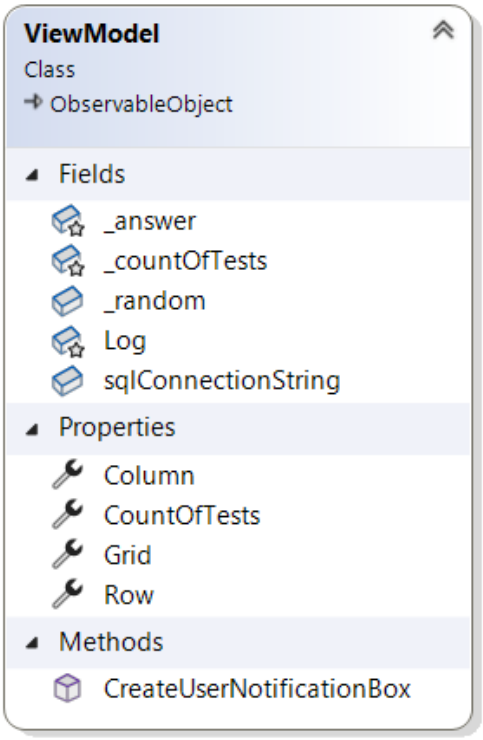

Рисунок 3.3 – Діаграма класу ViewModel

Цей клас реалізовує можливості запису відповіді користувача, зберігання поточної кількості тестів, розміщення зображення правильної відповіді, створення вікна сповіщення, а також надає усі ці можливості дочірнім класам за допомогою принципу наслідування.

Для кожного вікна та форми існує загалом по три класи: дані, графічний інтерфейс та логіка, яка поєднує попередні компоненти. Разом вони створюють повноцінну систему тестування знань іноземної мови.

3.3 Розробка засобу керування обліковими записами

Засіб, що реалізовуватиме процес керування обліковими записами у системі складається з вікна реєстрації та авторизації користувача.

Вікно реєстрації необхідне для того, щоб користувач міг створити обліковий запис та зберігати прогрес проходження тестування. Оскільки застосунок є десктопним, то не авторизований користувач не зможе користуватися системою тестування знань з іноземної мови. А для авторизації потрібно мати створений обліковий запис.

Вікно авторизації потрібно для того, щоб зареєстровані користувачі могли увійти в систему та отримати змогу користуватися системою та тестувати знання з іноземної мови.

Основні цілі вікна реєстрації включають:

1. Створення облікового запису: вікно реєстрації засобу керування обліковими записами дозволяє користувачам ввести дані, такі як ім’я, електронна пошта та пароль. Потім ці дані будуть використовуватись для ідентифікації та авторизації користувача в системі.

2. Перевірка та валідація даних: надана користувачем інформація буде перевірятись на наступні умови: наявність електронної пошти у базі даних (користувач вже існує), правильність формату пошти, довжину пароля та імені, наявність заповнення обов’язкових полів.

3. Зберігання даних: після перевірки та валідації даних, засіб реєстрації зберігає інформацію користувача у базі даних для подальшого її використання.

4. Безпека: вікно реєстрації повинен включати заходи безпеки, такі як шифрування паролів, захист від атак переповнення, обмеження спроб авторизації та інші механізми, щоб забезпечити безпеку облікових записів користувачів.

Діаграма послідовності роботи засобу реєстрації зображена на рисунку 3.4.

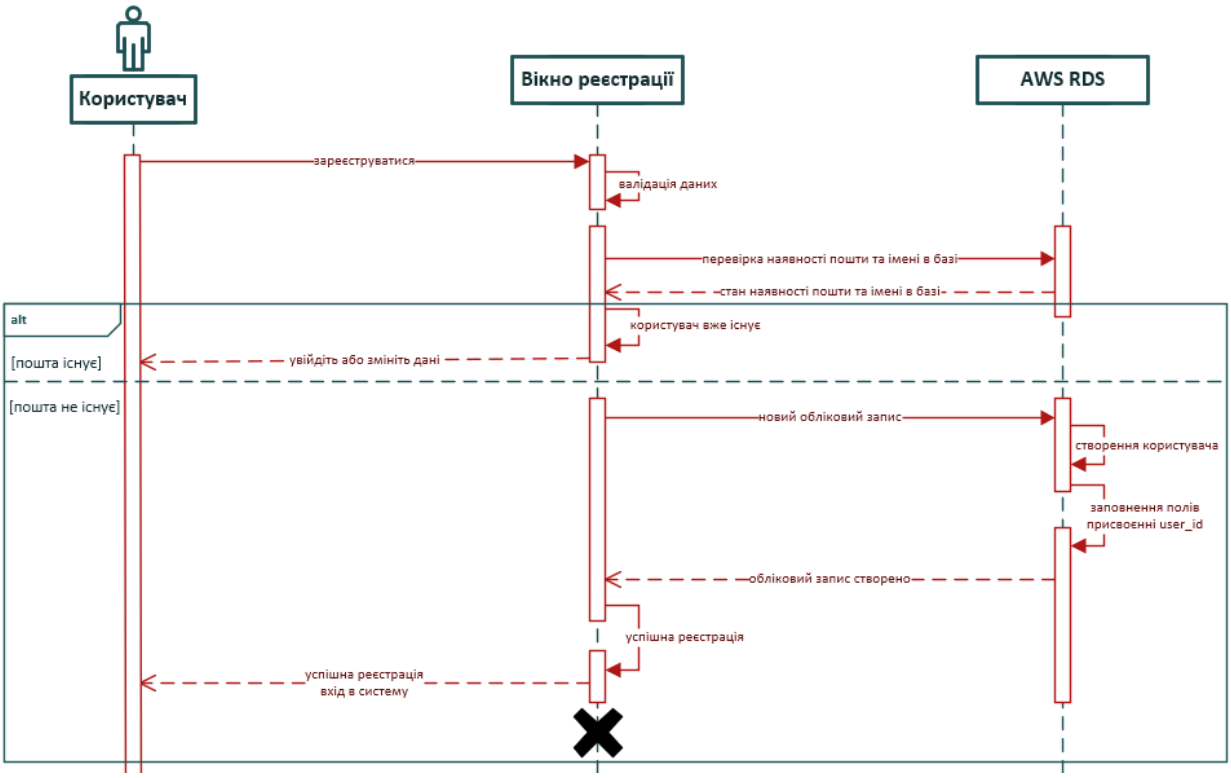

Рисунок 3.4 – Діаграма послідовності роботи засобу реєстрації

Основні цілі вікна авторизації включають:

1. Введеня та валідація облікових даних: у вікні авторизації користувач заповнює форму ім’я та пароль, які було використано при реєстрації. Після чого користувач натискає кнопку “Log in” і система, валідуючи ввведені дані, формує запит до бази даних

2. Перевірка даних у базі: якщо дані проходять валідацію на стороні користувача, то система надсилає запит на сервер AWS RDS, де облікові дані користувача перевіряються на належність до вже зареєстрованих облікових

записів, а також на відповідність комбінації паролю та імені існуючого облікового запису.

3. Аутентифікація: якщо облікові дані користувача виявляються правильними, сервер AWS RDS підтверджує валідність користувача та повертає відповідні дані для формування особистого кабінету системою.

4. Авторизація та управління сесією: після успішної авторизації, користувач може взаємодія з системою тестування знань з іноземної мови. Після проходження тестування дані про результат будуть записані локально в обліковий запис користувача. Та при бажанні вийти з облікового запису користувач може обрати функцію «Exit» або закрити вікно, що призведе до закінчення сесії авторизації. При цьому, всі збереженні дані система надішле в базу даних перед завершенням роботи.

Діаграма послідовності роботи засобу авторизації зображена на рисунку 3.5.

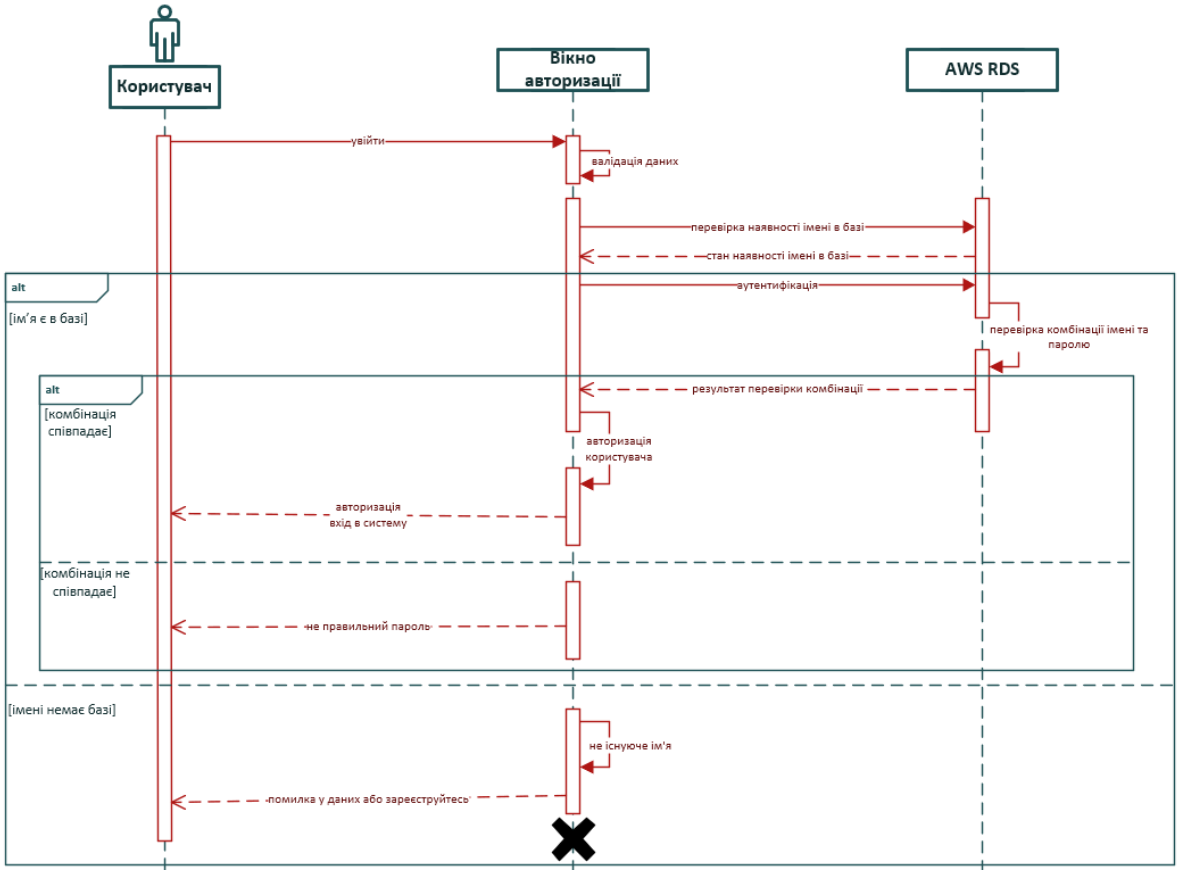

Рисунок 3.5 – Діаграма послідовності роботи засобу авторизації

3.4 Розробка засобу отримання даних з хмарного середовища

У будь-якому програмному забезпеченні важливу роль відіграють дані та місце, де вони зберігаються. Від місця залежить безпека та коректність збереження, а також коректність отриманих даних. Зазвичай для зберігання даних використовують бази даних (реляційна або не реляційні). Реляційні бази даних забезпечують чітку та жорстку систему структури зберігання даних, які запобігають певним помилками за допомогою нормалізації [20]. Нереляційні бази даних, або ж NoSQL, не орієнтовані на жорстко структуровану систему збереження даних, проте забезпечує механізм зберігання та видобування даних відмінний від підходу таблиць-відношень в реляційних базах даних [21].

Для тестової системи була обрана реляційна база даних, а саме Microsoft SQL Server. Сервер даних виконує головну функцію по збереженню та наданню даних у відповідь на запити інших застосунків, які можуть виконуватися як на тому ж самому сервері, так і у мережі [22].

Проте така система має певні недоліки, якщо задачею є розміщення даних у хмарному середовищі для доступу з будь-якої системи. Тому було вирішено розмістити створену базу даних у хмарному середовищі Amazon Web Services Relational Database Service, яка дасть змогу користувачам користуватись розроблюваною системою з будь-якого девайсу, підключеного до інтернету.

Дане хмарне середовище створено на основі Microsoft SQL Server та потребує доступу до нього. Підключення до бази даних відбувається за допомогою рядку підключення (рис. 3.6).

```
protected static readonly string SqlConnectionString = $"Data Source=filian-database.c5ce82k0wnfb.us-
  west-2.rds.amazonaws.com,1433;Initial Catalog=filian_db;{CredentialsDecryptor.DecryptCredentials()}";
```

Рисунок 3.6 – Рядок підключення до хмарного середовища

Після успішного підключення система готова до використання, а саме до входу користувача в систему або реєстрацію нового облікового запису, також до отримання даних про тестування.

Для авторизації (рис. 3.7) програма отримує дані, які користувач ввів, під’єднується до хмарного середовища та перевіряє чи існує такий користувач за такими даним. Якщо є – відкривається головне вікно, а якщо ні – сповіщає користувача що обліковий запис не існує.

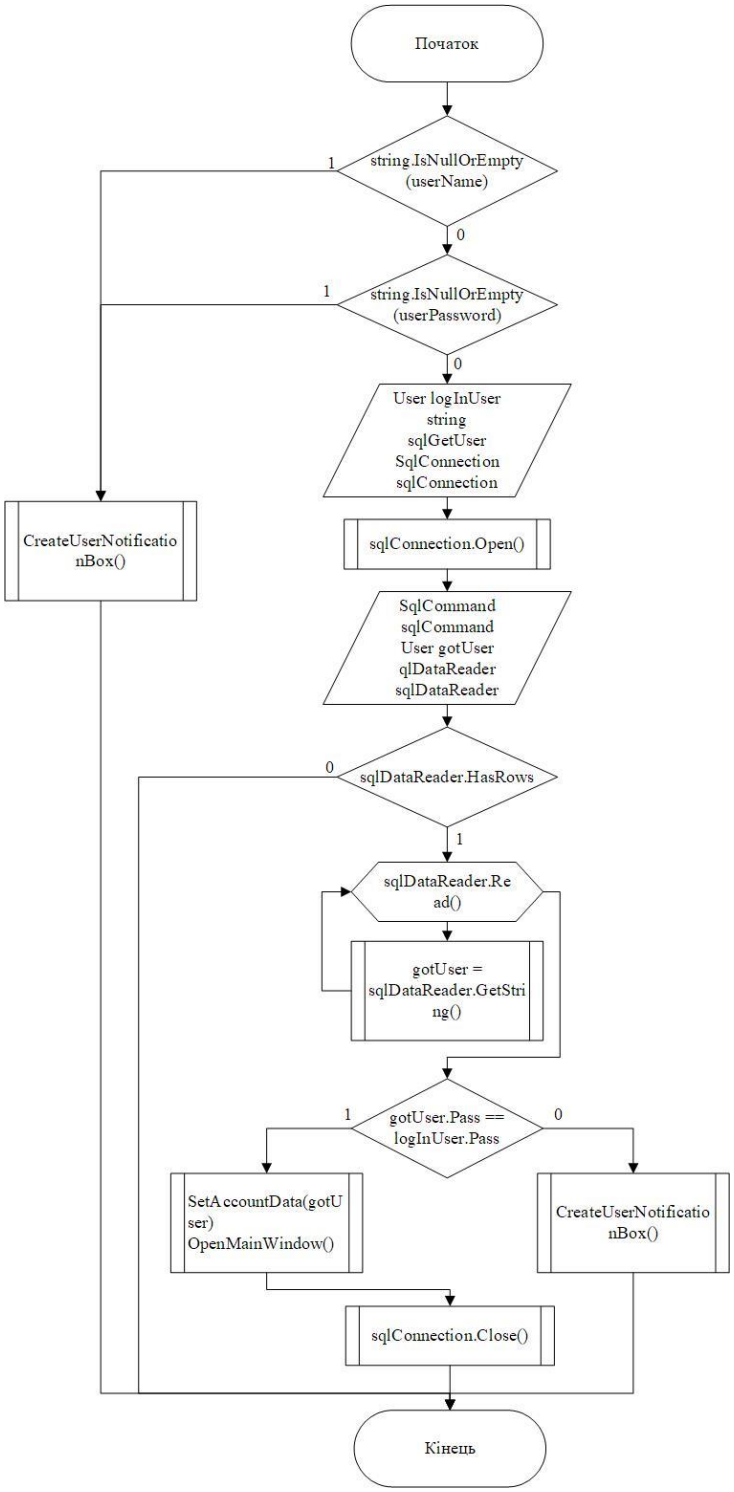

Рисунок 3.7 – Блок-схема алгоритму авторизації користувача

Після того, як користувач увійшов в систему, головне вікно одразу пропонує обрати мову для тестування, адже вже отримав список доступних мов для тестування з бази даних (рис. 3.8).

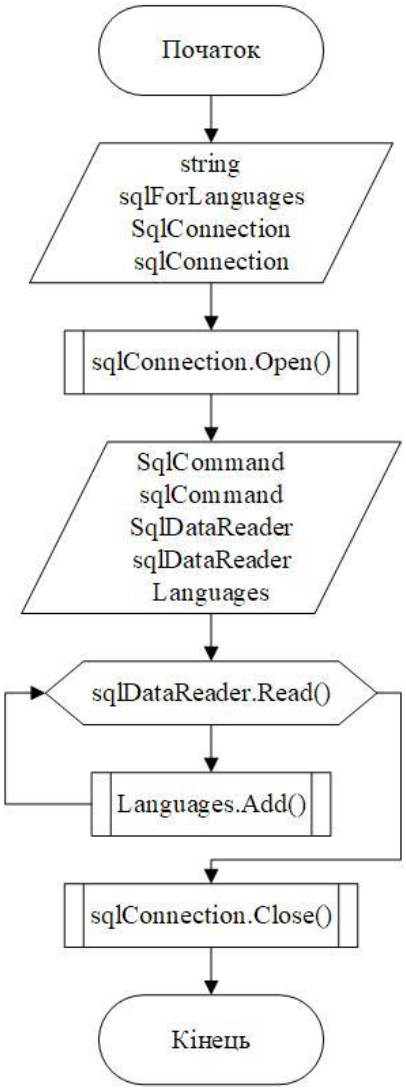

Рисунок 3.8 – Блок-схема алгоритму отримання списку мов для тестування

Отримання даних про типи тестування, теми та підтеми відбувається за таким же принципом.

Отримання даних про тестування (рис. 3.9) являється найскладнішим, адже відбувається створення одразу набору тестувань певного типу для всіх слів з обраних підтем.

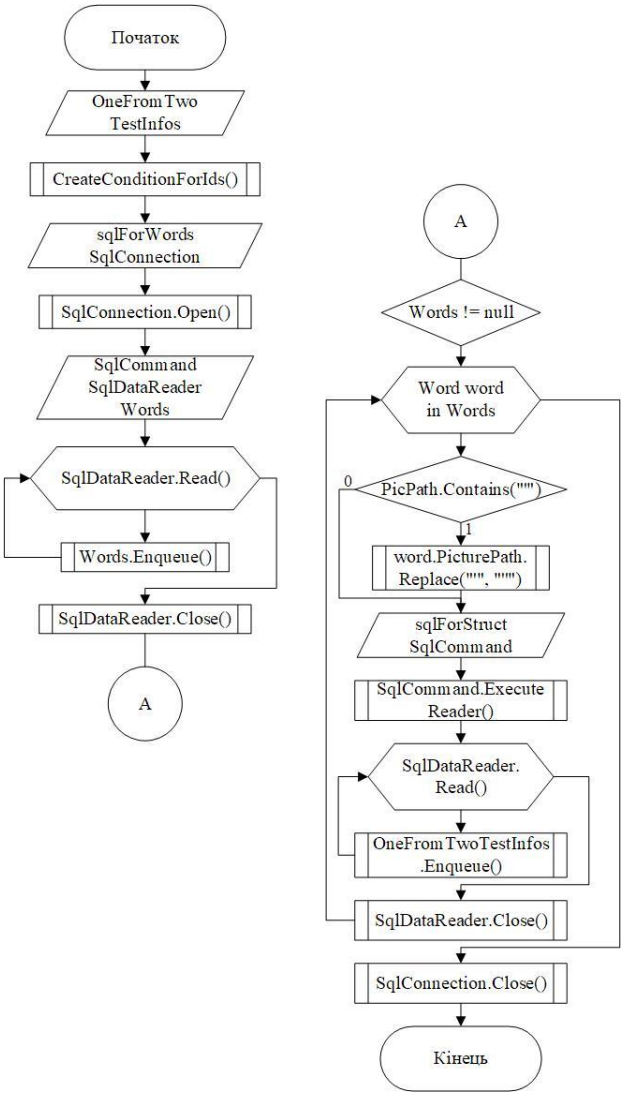

Рисунок 3.9 – Блок-схема алгоритму отримання даних про тестування

Отримання даних для інших типів тестування відбувається за таким же принципом. Єдина відмінність – набір параметрів запиту. Для одних в запиті можуть бути вказані лише id підтеми та шлях до картинки. Для інших – шлях до звуку. Усі дані, отримані з бази зберігаються в класах з такою ж назвою але з приставкою -Info, які в свою чергу є наслідниками універсального класу DataBaseEntity, який являє собою сутність отриманих даних з бази. Він має такі поля як: picturePath – для зберігання шляху до картинки, Id – унікальний ідентифікатор сутності з бази, Name – поле для слова в оригіналі, Translation – поле для перекладу слова обраною користувачем мовою, та властивість для поля picturePath – для корегування шляху до картинки на комп’ютері користувача.

Реалізацію даного класу зображено на рисунку 3.10.

```
public abstract class DataBaseEntity : ObservableObject
{
    private string picturePath;
    18 references
    public int Id { get; set; }
    24 references
    public string Name { get; set; }
    22 references
    public string Translation { get; set; }

    29 references
    public string PicturePath
    {
        get { return picturePath;}
        set { picturePath = Path.GetFullPath(value); }
    }
}
```

Рисунок 3.10 – Програмна реалізація класу DataBaseEntity

За допомогою описаних методів відбувається надійний обмін даними між користувачем та хмарним середовищем.

3.5 Розробка графічного інтерфейсу

Графічний інтерфейс користувача відіграє найважливішу роль в розробці будь-якого програмного застосунку. В залежності від взаємодії користувача з графічним інтерфейсом будується наступний план дій відображення об’єктів та робота компонентів на стороні бек-енду.

Розробляючи інтерфейс застосунку «Filian», вирішено зробити його максимально простим та дружнім для використання користувачем.

Основним кольором інтерфейсу обрано темно-синій, оскільки даний колір є нейтральним. Допоміжними кольорами обрано аква та білий, оскільки вони влучно контрастують один одному.

Для розробки графічного інтерфейсу може відбуватись за допомогою написання XAML коду, який використовує систему тегів для створення елементів, або за допомогою вікна інструментів, яке надає середовище розробки та

дизайнерського вікна, в якому розробник може взаємодіяти з елементами та міняти їх розташування. При виборі якогось конкретного елементу – з'являється вікно властивостей, за допомогою якого можна змінити зовнішній вигляд, поведінку та взаємодію з кодом цього елементу. На даному етапі середовище розробки відіграє важливу роль, адже саме за допомогою допоміжних елементів залежить зручність розробки графічного інтерфейсу користувача.

Написання XAML коду відбувається паралельно з переглядом результату (рис. 3.11). За допомогою цього, розробник одразу бачить результат своє роботи, та може паралельно працювати у двох вікнах, рухаючи об'єкти за допомогою коду та за допомогою взаємодії з відображуваними об'єктами.

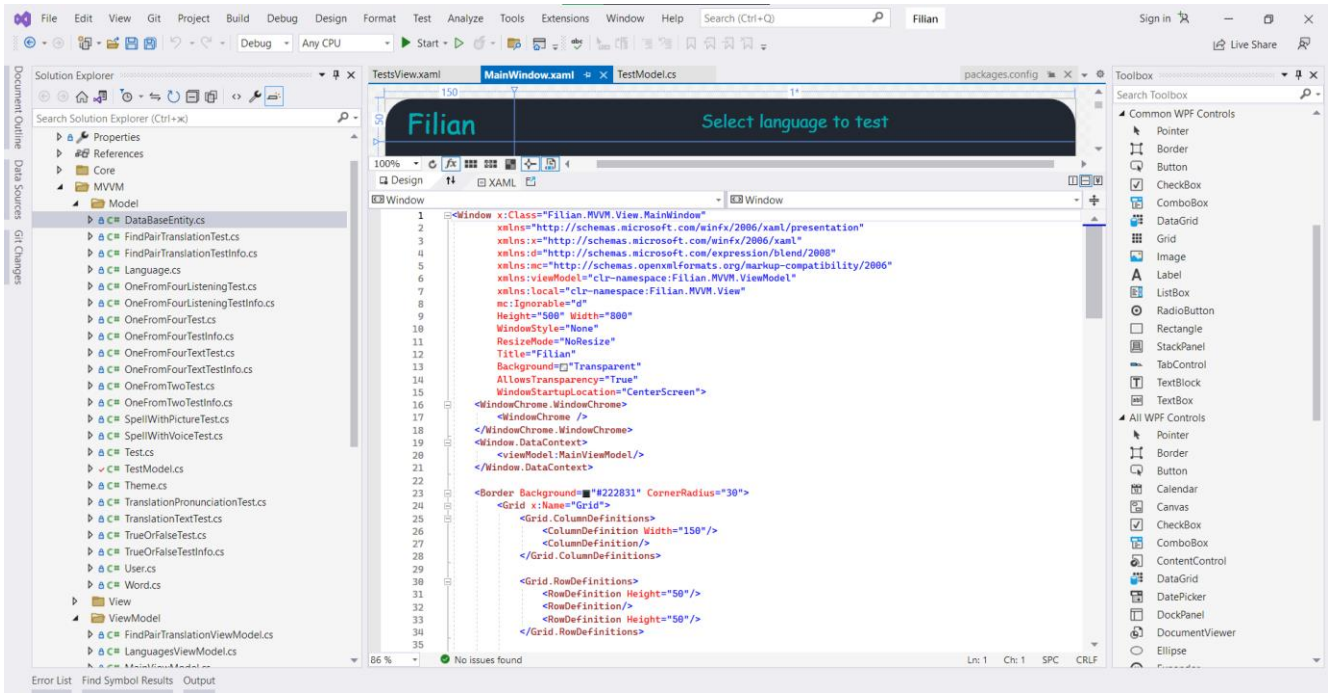

Рисунок 3.11 – Розробка графічного інтерфейсу

У програмі використовується кілька вікон для відображення усієї необхідної інформації: вікно реєстрації, вікно авторизації, головне вікно та вікно сповіщення. Також є форми для відображення тестування та особистого кабінету користувача у головному вікні.

При відкритті додатку «Filian», користувача зустрічає екран авторизації з полями для логіна та пароля облікового запису (рис. 3.12), за допомогою якого користувач зможе зайти в систему.

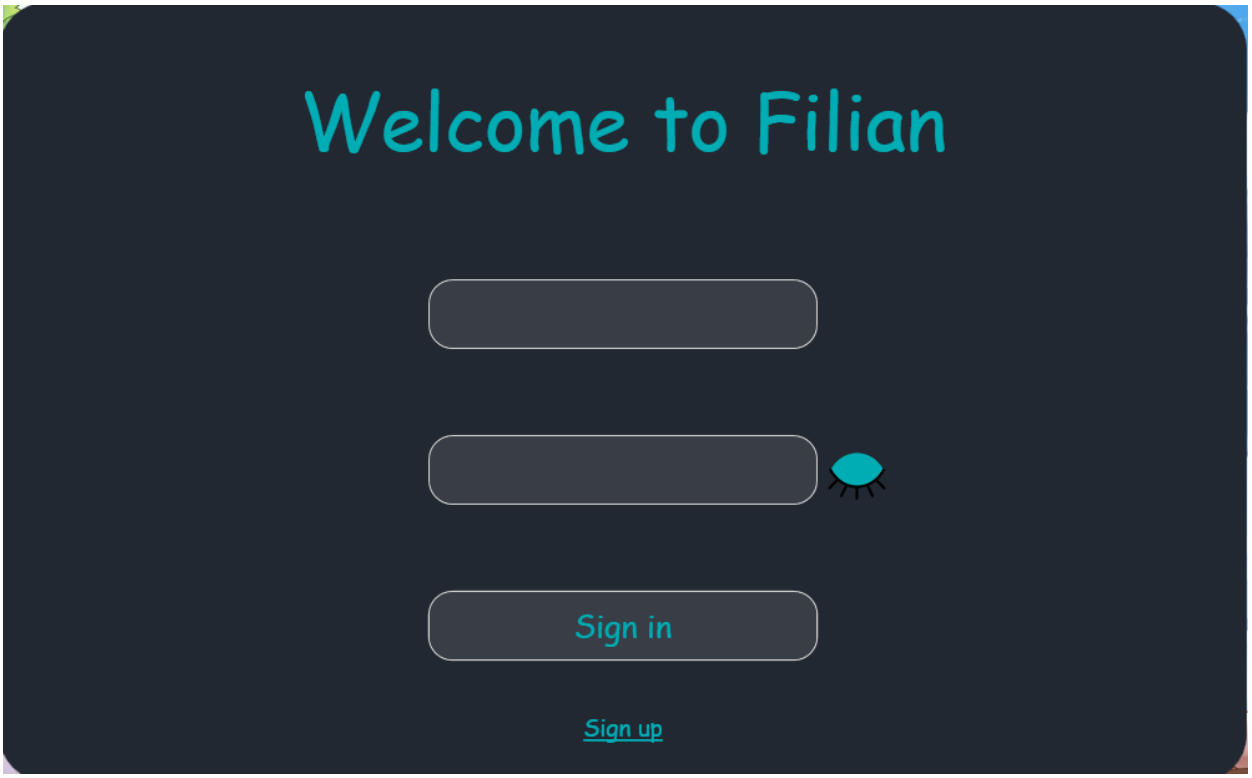

Рисунок 3.12 – Вікно авторизації

Для розробки вікна авторизації було використано два поля для імені користувача та паролю до облікового запису. Також присутня кнопка «Sign in» та посилання на вікно реєстрації, якщо користувач все ще не має облікового запису в «Filian». Для того, щоб користувач дізнався, які дані в які поля потрібно вводити, йому потрібно навести курсор миші на відповідне поле, тоді з’явиться підказка, що потрібно вводити (рис.3.13).

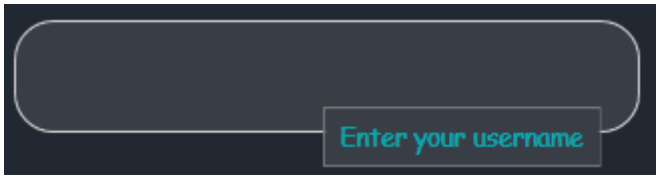

Рисунок 3.13 – Підказка для введення даних в поле

Розробка вікна авторизації зображена на рисунку 3.14.

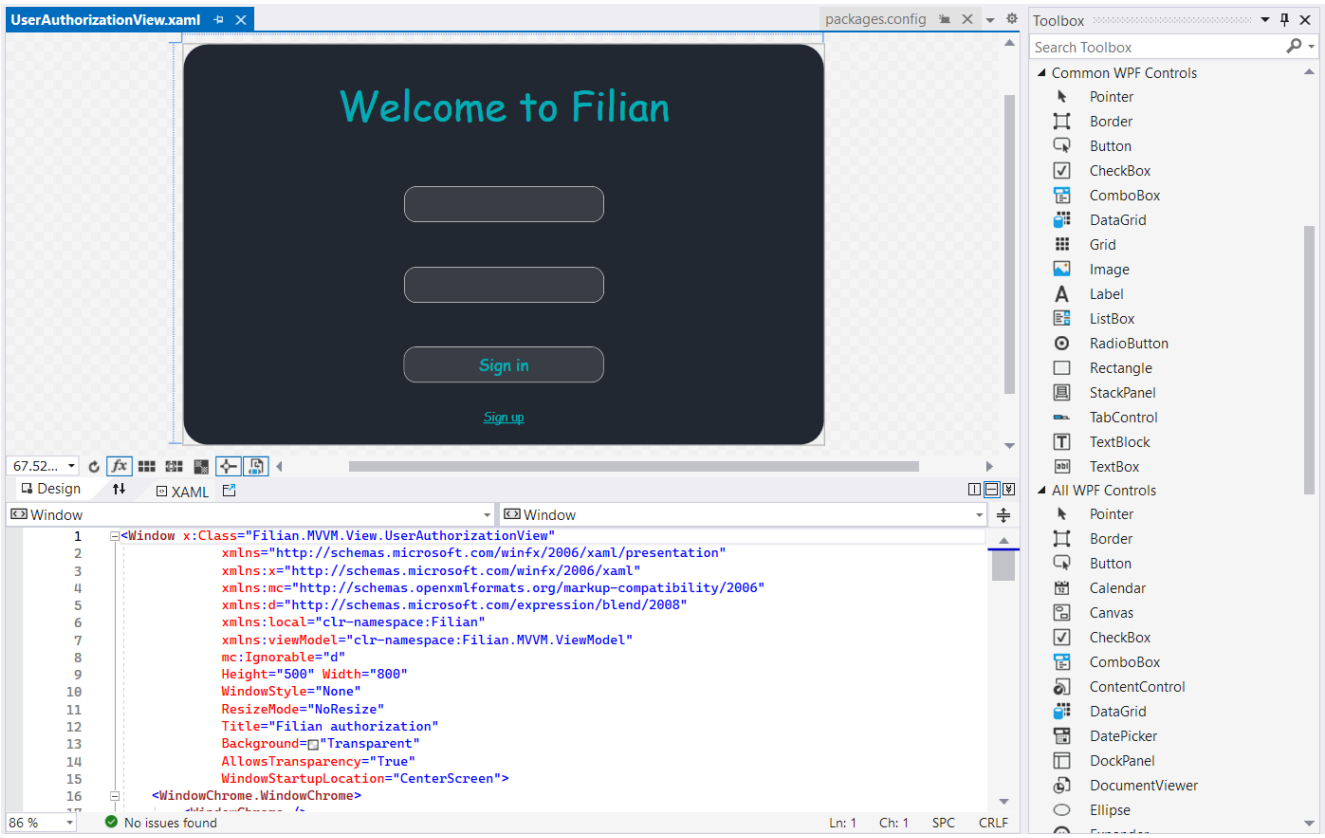

Рисунок 3.14 – Розробка вікна авторизації

За допомогою цього вікна користувач може авторизуватися в свій обліковий запис та почати тестування знань іноземної мови.

Оскільки онлайн система «Filian» передбачає реєстрацію окремих користувачів, спроєктовано та додано екран із реєстрацією користувача (рис. 3.15). На цьому екрані користувач системи повинен ввести персональні дані та пароль. Як і в вікні авторизації, при наведенні курсора на поле – висвічується підказка, за що саме це поле відповідає. При реєстрації існує валідація полів, тобто перевірка на правильність введення. Ім’я користувача може бути від 3 до 20 символів, електронна адреса має відповідати стандартам, пароль повинен містити великі та малі літери, а також цифри.

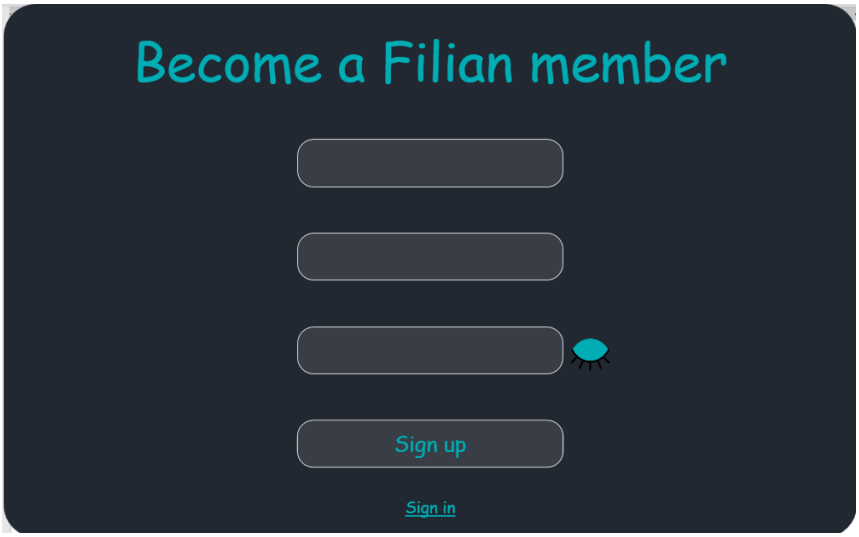

Рисунок 3.15 – Вікно реєстрації користувача

У цьому вікні користувач може ввести свої дані та зареєструвати обліковий запис в застосунку.

Вікно авторизації та вікно реєстрації (рис.3.16) являються майже ідентичними. Їх відрізняє лише текст та додаткове поле у вікні реєстрації для введення електронної адреси для створення користувача.

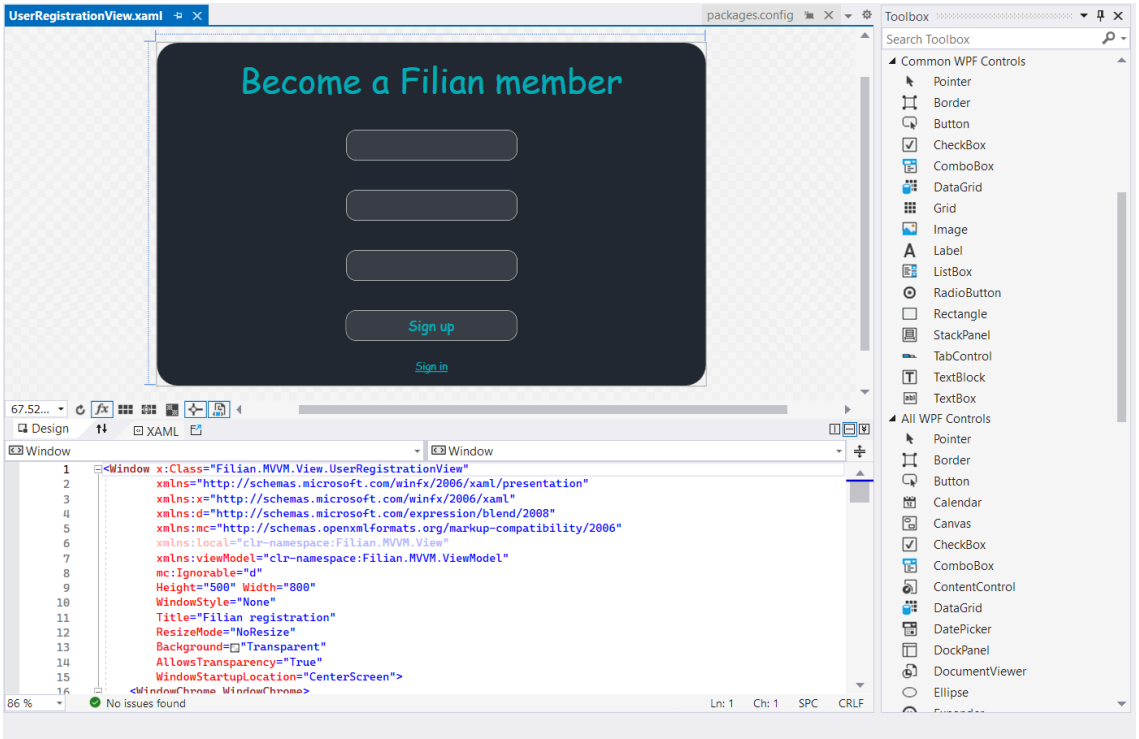

Рисунок 3.16 – Розробка вікна реєстрації

У вікнах авторизації та реєстрації поле для пароля приховує сам пароль спеціальними символами (рис. 3.17), і для того щоб перевірити вірність введення, можна скористуватись функцією «показати пароль», натиснувши на зображення закритого ока (рис. 3.18).

Рисунок 3.17 – Приховування паролю

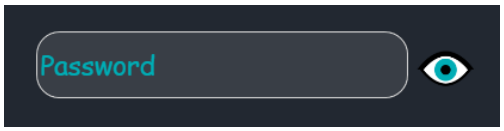

Рисунок 3.18 – Функція «показати пароль»

Після того, як користувач авторизувався, він побачить вікно програми, яка вже пропонує йому обрати мову для тестування (рис. 3.19). Наразі наявно 7 мов, а саме: українська, англійська, іспанська, португальська, китайська, німецька та польська.

Рисунок 3.19 – Інтерфейс програми та вікно вибору мови

У головному вікні користувачеві доступні всі можливості застосунку тестування знань іноземної мови: вибір мови для тестування, вибір типу тестування, вибір теми та підтеми тестування. Також користувач може переглянути дані свого облікового запису у вкладці «Account» та вийти з системи. Відображення всіх цих форм вбудовано в головне вікно та доступно завдяки компоненту «ContentControl».

На цьому етапі користувач може обрати мову, перейти на сторінку власного облікового запису або перейти до тестів (в такому випадку обереться мова тестування – англійська).

Головне вікно (рис. 3.20) є основним засобом взаємодії користувача з програмою.

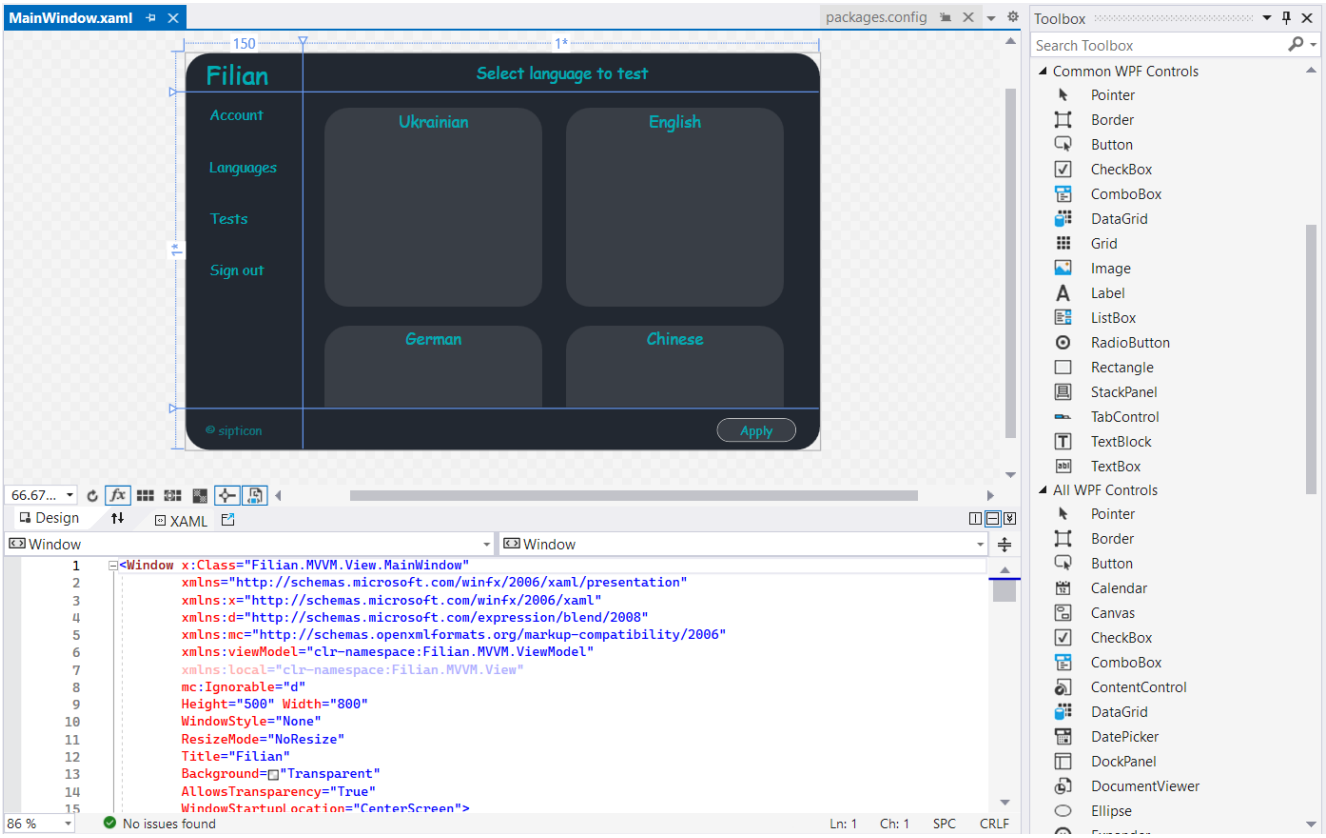

Рисунок 3.20 – Розробка головного вікна

Сторінка облікового запису (рис. 3.21) містить інформацію про поточного користувача в системі.

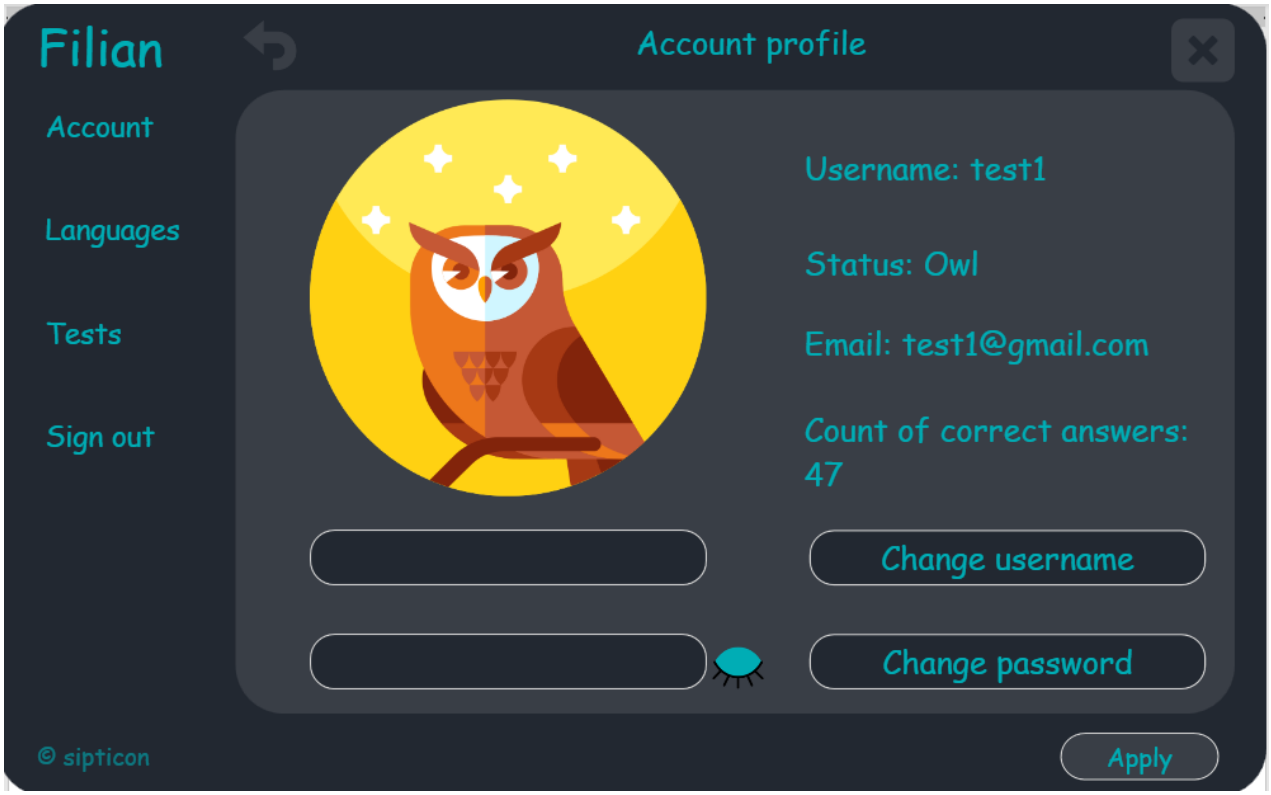

Рисунок 3.21– Сторінка аккаунту користувача

На цьому екрані користувач може побачити ім’я користувача, введене при реєстрації, статус облікового запису (надається за кількість правильних відповідей), електрону пошту та загальну кількість правильних відповідей. Також є можливості змінити своє ім’я та пароль. Після внесення змін дані одразу ж оновляться в базі даних.

Форма вибору мови створена у вигляді списку з двох колонок, який користувач може гортати. Форми типу тестування, теми та підтем створено за одним шаблоном, адже загальний принцип вибору – однаковий, а відрізняються лише дані відображення.

Форма вибору мови зображена на рисунку 3.22.

Рисунок 3.22 – Форма вибору мови для тестування

У таких формах користувачу пропонується обрати те, що йому потрібно, та натиснути кнопку «Apply». Після цього обробляються дані та створюється форма тестування на основі обраних параметрів.

На сторінці з тестами (рис. 3.23), користувач може побачити перелік тестів різного типу та обрати один з десяти доступних.

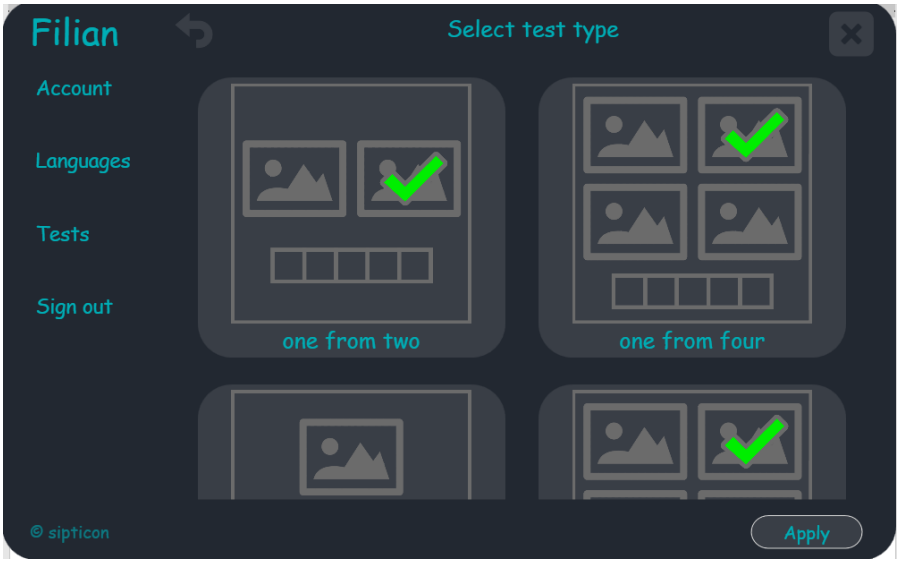

Рисунок 3.23– Сторінка вибору типу тесту

Після того, як користувач обрав тип тестування, він зможе вибрати тему, по якій проводитиметься тестування (рис. 3.24). На вибір доступно 20 різних тем, які цілком можуть допомогти в тестуванні знань з іноземної мови. До вибору доступні наступні теми: тварини, мистецтво, птахи, одяг та аксесуари, динозаври, риби та молюски, їжа, географія та архітектура, дім та офіс, людина, комахи, дитячі іграшки, природа та навколишнє середовище, об’єкти, рослини, рептилії, спорт, транспорт, дієслова та «котрий». Кожна з цим тем містить безліч слів, якими люди користуються у повсякденні. Саме тому важливо їх знати. Для того щоб перевірити, які слова доступні користувачеві – надана можливість тестування усіх цих слів.

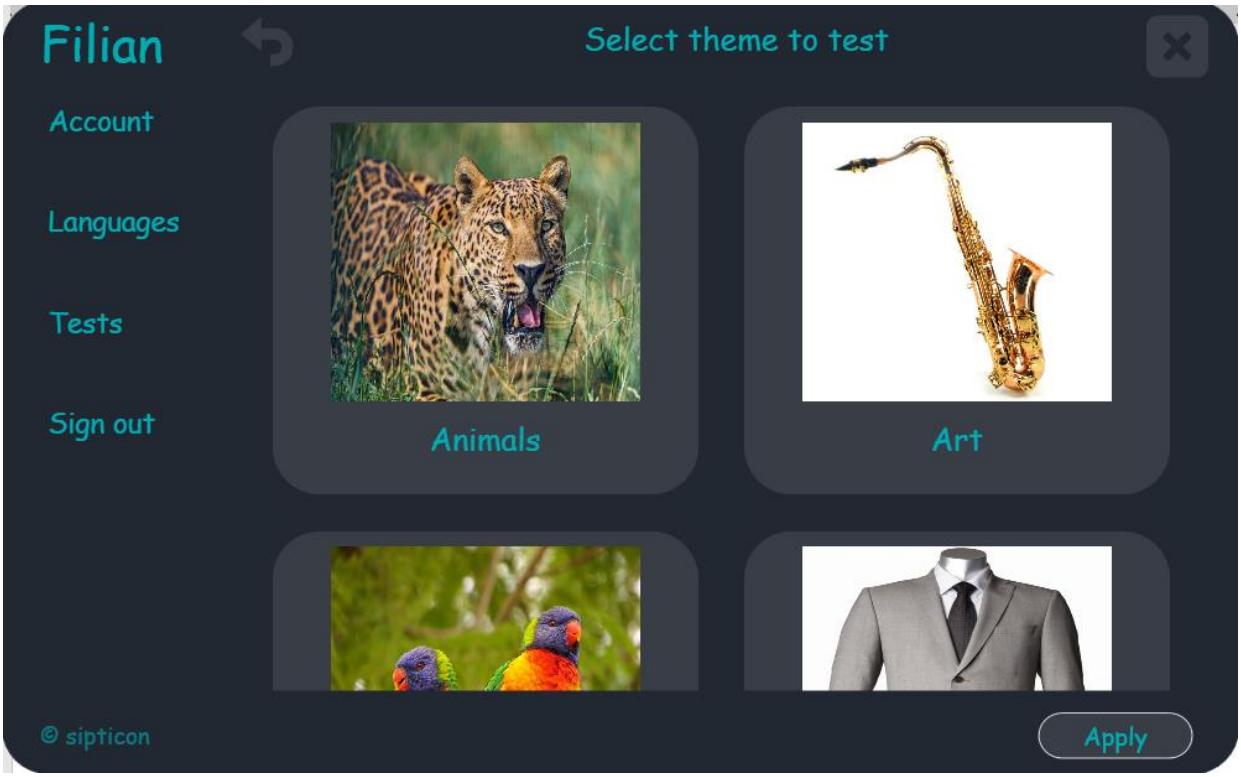

Рисунок 3.24– Сторінка вибору теми

Вибравши тему, система відобразить форму зі списком підтем обраної теми та запропонує користувачеві обрати одну або декілька (рис. 3.25) для початку проведення тестування. Для кожної теми – різна кількість підтем. Користувач може обрати будь-яку кількість підтем (мінімум 1). Обрані підтеми підсвічуватимуться синім відтінком для позначення того що з них буде формуватися тестування.

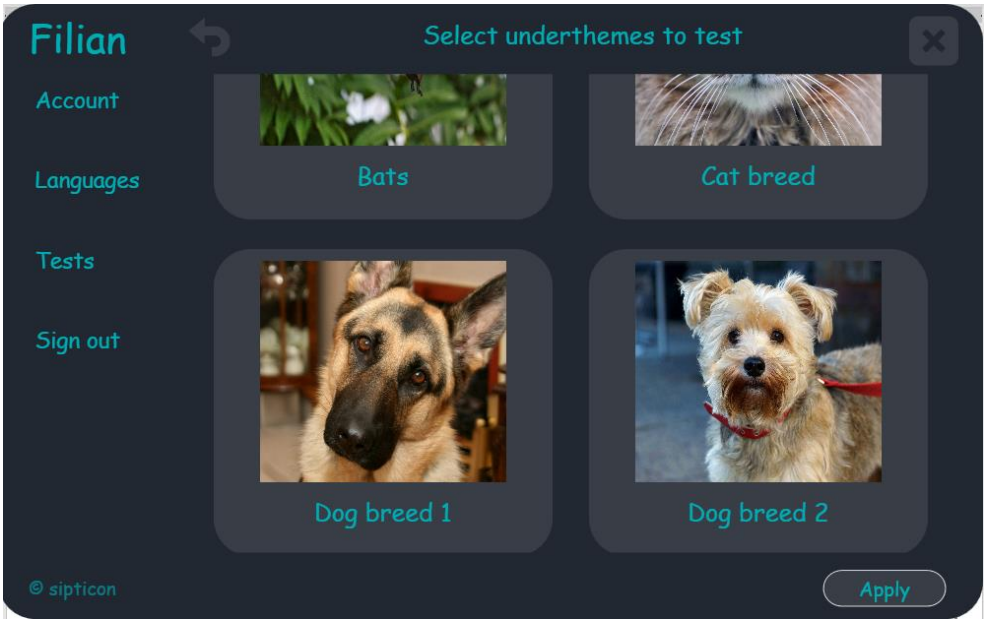

Рисунок 3.25– Сторінка вибору підтем для теми «Тварини»

Після вибору підтем – розпочнеться тестування (рис. 3.26). Вгорі вікна можна побачити кількість тестів яка залишилась до закінчення тестування.

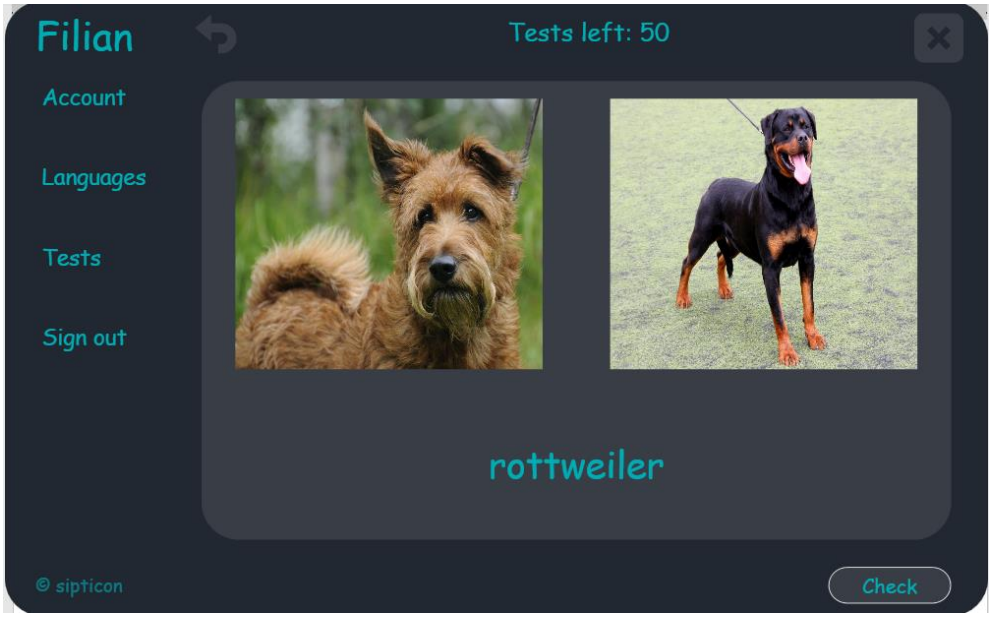

Рисунок 3.26– Сторінка обраного тесту

Користувач не може повернутись назад під час тесту або перейти на інші сторінки. Також забезпечено можливість виходу з облікового запису, зміна мови та

типу тестування. Після завершення поточного тестування – програма запропонує користувачеві обрати наступний тип тестувань.

У застосунку доступно десять різних типів тестувань, тому форма відображення у кожного різна та розроблялась для кожного окремо. Тому буде розглянуто лише створення першого типу тестування.

Тестування типу «вибери одне з двох» (рис. 3.27) пропонує на вибір 2 картинки та слово, написане обраною мовою користувачем. Задача полягає у виборі картинки, зображення якої відповідає слову. Після вибору – користувачеві необхідно натиснути кнопку «Check» і програма відобразить чи була правильно обрана відповідь та перейде до наступного тестування.

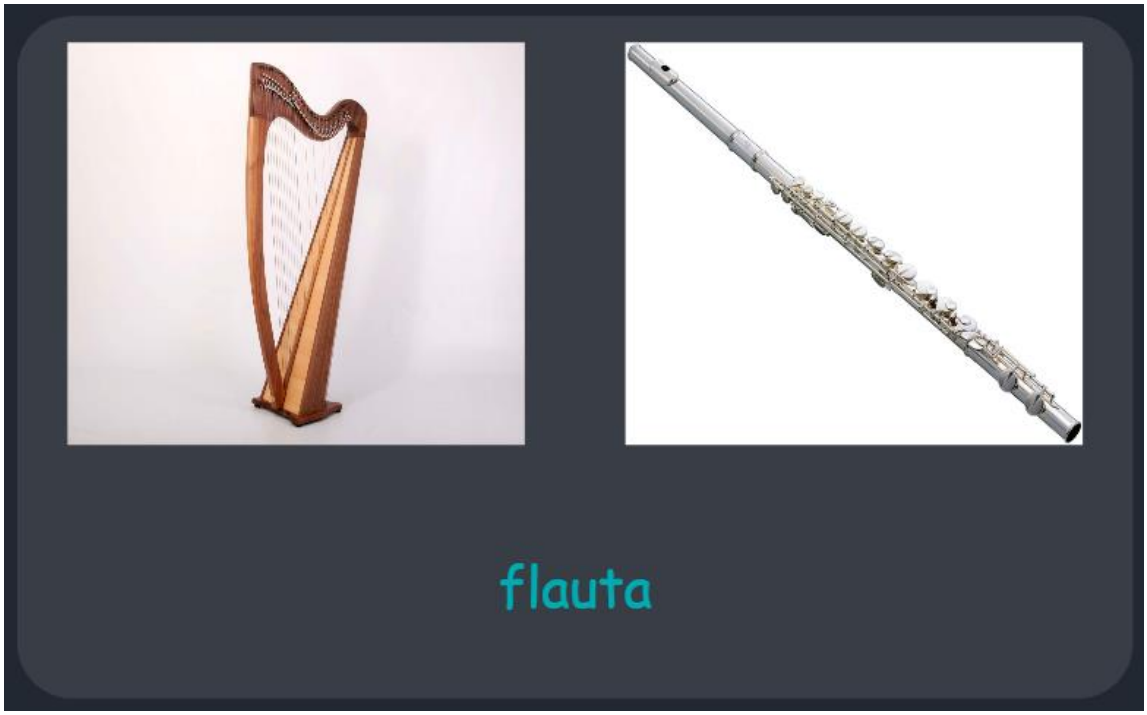

Рисунок 3.27 – Форма першого тестування

Після закінчення тестування на екрані користувача з’явиться вікно сповіщень, яке проінформує про кількість правильних відповідей.

Вікно сповіщення (рис. 3.28) є важливим інструментом для сповіщень користувачів про різні події, як от не правильні параметри для реєстрації або входу

в обліковий запис, помилка з’єднання з базою даних, та результат проходження тестування.

Рисунок 3.28 – Вікно сповіщення

Дане вікно сповіщень інформує користувача, що введене ім’я є некоректним.

Розробка графічного інтерфейсу платформи була проведена у середовищі розробки Microsoft Visual Studio з використанням технології WPF та являється однією з найважливіших етапів розробки програмного застосунку.

3.6 Розробка засобу відображення графіку проходження тестувань

Кожен користувач розроблюваної системи має на меті покращити свої знання з іноземної мови, і щоб спостерігати прогрес було реалізовано засіб формування і відображення прогресу навчання у вигляді графіку, де користувач зможе отримати дані щодо того, в який день і яку кількість правильних відповідей він отримав.

Дана функція доступна до перегляду на головному вікні, зліва в панелі навігації. Користувачеві необхідно натиснути кнопку «Report» і він отримає звітність щодо проходження тестувань.

У вікні звіту (рис. 3.29) користувач, за допомогою «» може обрати тип тестування за яким хоче переглянути статистику правильних відповідей.

Рисунок 3.29 – Вікно графіку звітності щодо проходження тестувань

Даний засіб реалізовано за допомогою бібліотеки OxyPlot [23], яка надає методи та засоби необхідні для створення графіків у WPF.

За створення графіка відповідає метод CreatePlotModel() який отримує параметр testType, тобто тип тесту, по якому потрібно сформувати графік, та повертає клас типу PlotModel, який передається у view, що відповідає за графічне відображення. Даний метод покроково створює кожен компонент графіку, як от: plotModel – модель всього графіку, в який і додаються компоненти, dateAxis – вісь для відображення дати, valueAxis – вісь для відображення шкали кількості правильних відповідей, series – точки перетину (яка кількість правильних відповідей була в на поточний день), results – дані щодо конкретного типу тестування.

Блок-схема алгоритму роботи даного засобу зображена на рисунку 3.30.

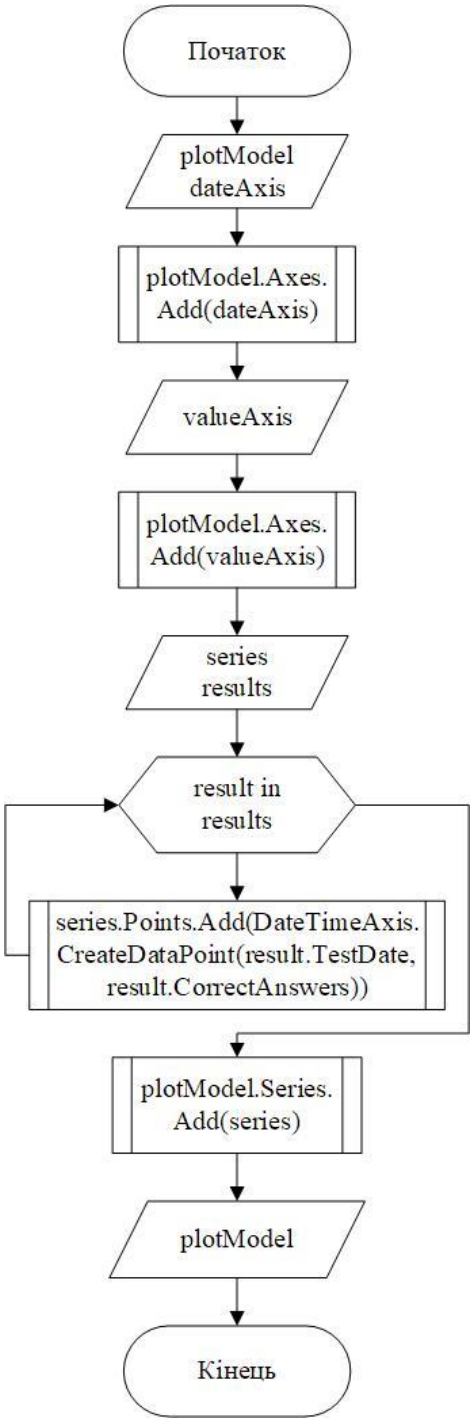

Рисунок 3.30 – Блок-схема алгоритму створення графіку звітності проходження тестувань

3.7 Висновки

У третьому розділі було обґрунтовано вибір мови програмування та технологій для розробки системи. Проаналізувавши потреби програмного продукту було обрано мову програмування C#. Для зручності розробки графічного

інтерфейсу було обрано WPF через можливість написання графічного інтерфейсу з використанням мови XAML. В якості хмарного середовища було обрано AWS RDS створеного на основі Microsoft SQL Server.

Було здійснено розробку основних програмних засобів, а саме:

– загальну архітектуру та логіку системи, що забезпечує легку підтримку коду, впровадження нових модулів та продовження роботи на системою;

– засіб керування обліковими записами, для реєстрації, авторизації та використання облікового запису користувачем;

– засіб отримання даних з хмарного середовища, для отримання даних системи та збереження даних користувача;

– графічний інтерфейс програмного додатку, який є простим та забезпечує зручну взаємодію користувача з системою;

– засіб відображення графіку правильних відповідей за поточну дату, для відстеження прогресу навчання користувача.

Було описано створення одного з важливих компонентів системи, а саме графічного інтерфейсу користувача. Також описано процеси отримання та запису даних у хмарному середовищі та загальну архітектуру системи, яка була створена на основі шаблону проєктування Model-View-ViewModel включно з головним компонентом системи – логікою, тобто бек-ендом.

4 ТЕСТУВАННЯ ПРОГРАМИ

4.1 Тестування системи

Тестування розробленої системи являється процесом перевірки функціональності, безпеки та коректності роботи програми на десктопній платформі. Одним з найважливіших критеріїв успішного тестування є перевірка сумісності програми з різними версіями операційної системи.

Для тестування необхідно встановити тестову версію програми на пристрій з операційною системою Windows, яка буде використовуватись під час тестування. Далі проводяться функціональні тести для перевірки правильності роботи програми, які включають тести на введення та виведення даних, тести на реакцію програми на негативні вхідні дані, а також тести на відповідність бізнес логіки.

Після цього проводяться тести на безпеку програми, зокрема перевірка на проникнення, тести на збереження конфіденційної інформації та тести на злам програми. Для цього можуть використовуватись спеціальні програмні засоби, які дозволяють моделювати атаки на програму та виявляти її вразливості.

Після успішного проходження всіх тестів проводиться оцінка продуктивності програми, зокрема тести на швидкість роботи програми, тести на використання ресурсів пристрою та тести на життєвий цикл програми.

Усі результати тестування записуються у відповідну документацію та використовуються для подальшого вдосконалення програми та забезпечення якості її роботи на платформі Wndows.

Середовище розробки, таке як Microsoft Visual Studio дозволяє протестувати програму на ресурсозатратність (рис. 4.1). Цей засіб є ефективним при тестуванні розроблюваного застосунку, адже показує, які саме елементи являються ресурсозатратними. Також цей засіб є ефективним, оскільки не потребує сторонніх програм. Тобто розробник може тестувати програму без встановлення додаткових засобів тестування.

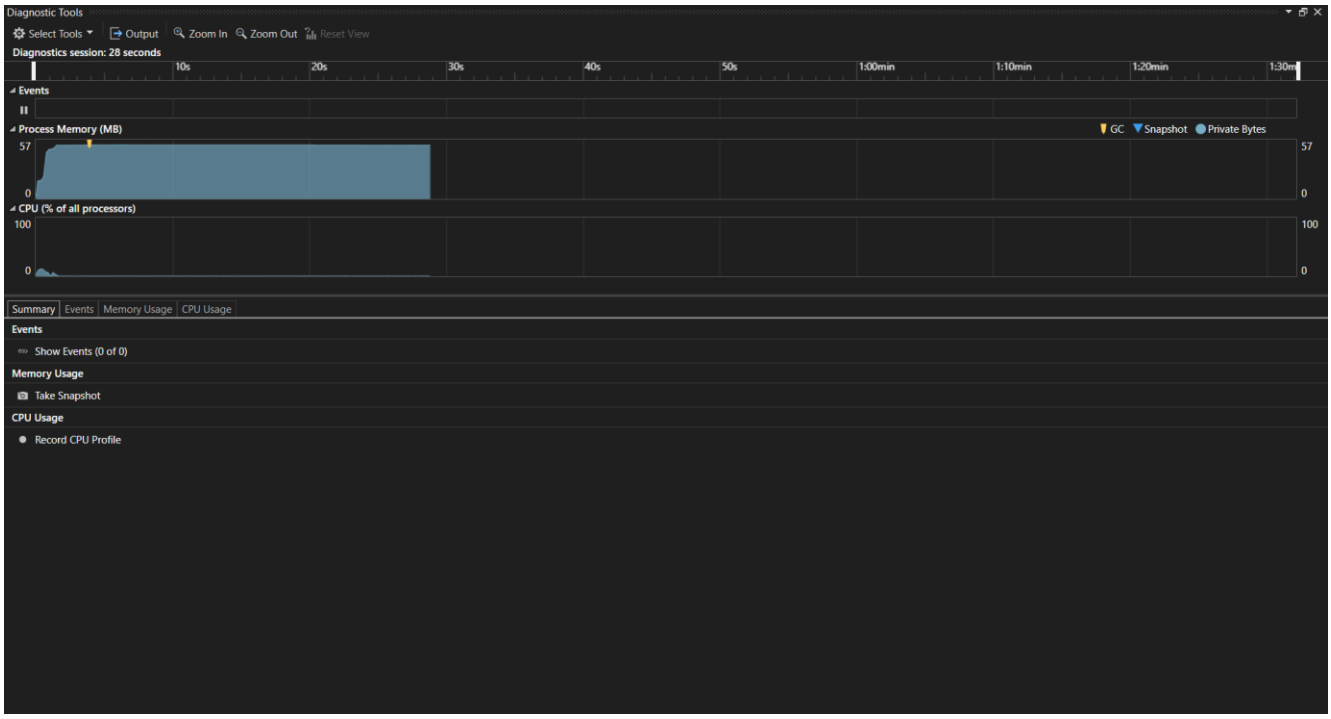

Рисунок 4.1 – Вікно тестування

Наступним кроком буде перевірка вікна авторизації користувача користувача шляхом спроби входу без введення імені. У разі спроби входу без обов'язкової інформації про ім'я, система повідомить про помилку та запропонує ввести відповідні дані (рис. 4.2), що є правильною поведінкою системи, оскільки інформація про ім'я є обов'язковою для подальшої роботи користувача.

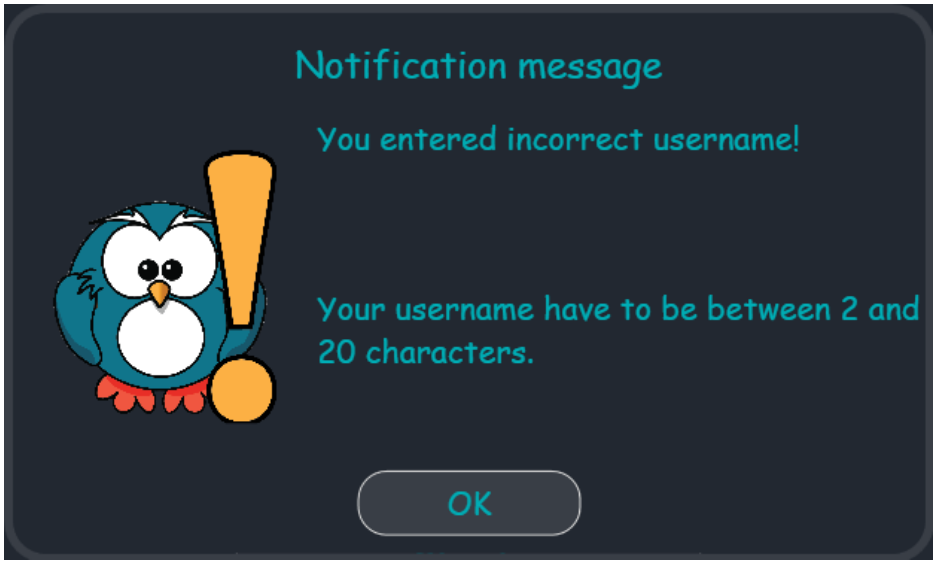

Рисунок 4.2 – Авторизація користувача без імені

Переконавшись, що система не дозволяє вхід користувача без імені, спробуємо ввести його, проте без пароля. У результаті тестування отримаємо сповіщення про те, що потрібно ввести пароль (рис. 4.3).

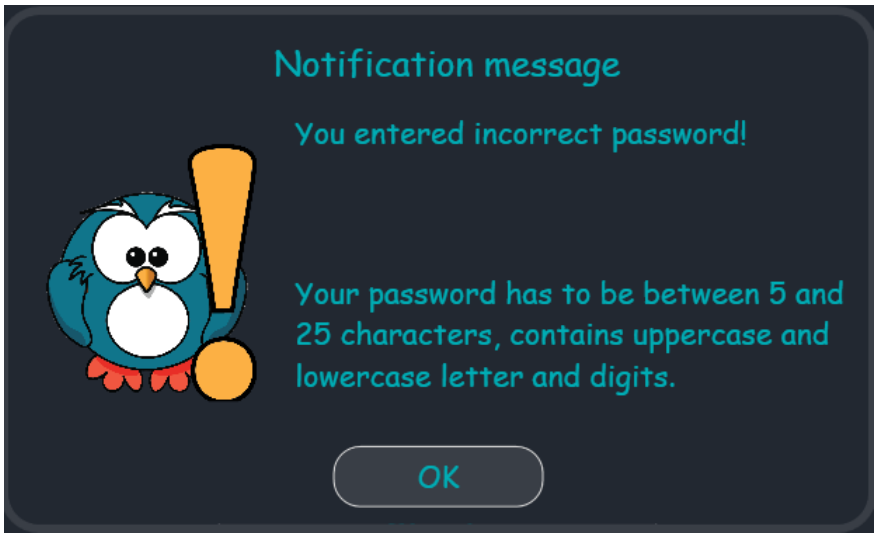

Рисунок 4.3 – Авторизація користувача без пароля

Перевіривши введення не дійсних даних, спробуємо заповнити всі поля вірно і перевірити можливий вхід в систему при неможливості з’єднання за хмарним середовище. У результаті перевірки система сповістить користувача, що щось пішло не так і потрібно перевірити коректність введених даних або з’єднання з інтернетом (рис. 4.4).

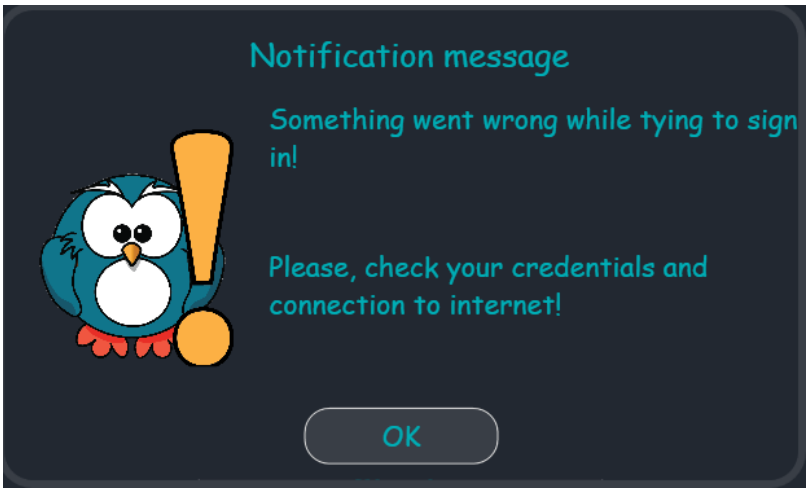

Рисунок 4.4 – Відсутність інтернет з’єднання

Якщо ж всі поля заповнено коректно та є з’єднання з інтернетом – система дозволить вхід та відкриє головне вікно.

Важливим аспектом платформи є валідація вхідних даних при реєстрації. Вони мають відповідати умовам та бути відповідними для збереження в базі даних. Тому, при невідповідності полів, система сповістить користувача, які вони мають бути (рис. 4.5).

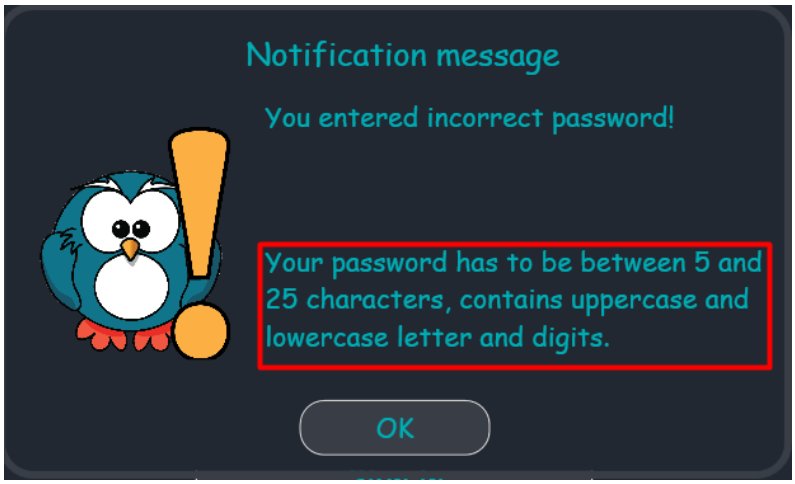

Рисунок 4.5 – Валідація поля паролю

4.2 Розробка інструкції користувача

Інструкція користувача передбачає визначення технічних вимог для запуску програмного продукту. Деталі щодо мінімальної та рекомендованої конфігурації серверного комп’ютера можна знайти в таблицях 4.1 та 4.2.

Таблиця 4.1 – Мінімальна конфігурація:

Тип процесора	32-розрядний (x86) або 64-розрядний (x64) процесор з тактовою частотою 1 ГГц
Об’єм оперативної пам’яті	2 ГБ
Місце на жорсткому диску	64 ГБ
Операційна система	Windows 7/8/10/11

Таблиця 4.2 – Рекомендована конфігурація:

Тип процесора	64-розрядний (x64) процесор з тактовою частотою 2 ГГц
Об’єм оперативної пам’яті	4 ГБ
Місце на жорсткому диску	128 ГБ
Операційна система	Windows 7/8/10/11

Для початку користувач повинен авторизуватися в системі (рис. 4.6).

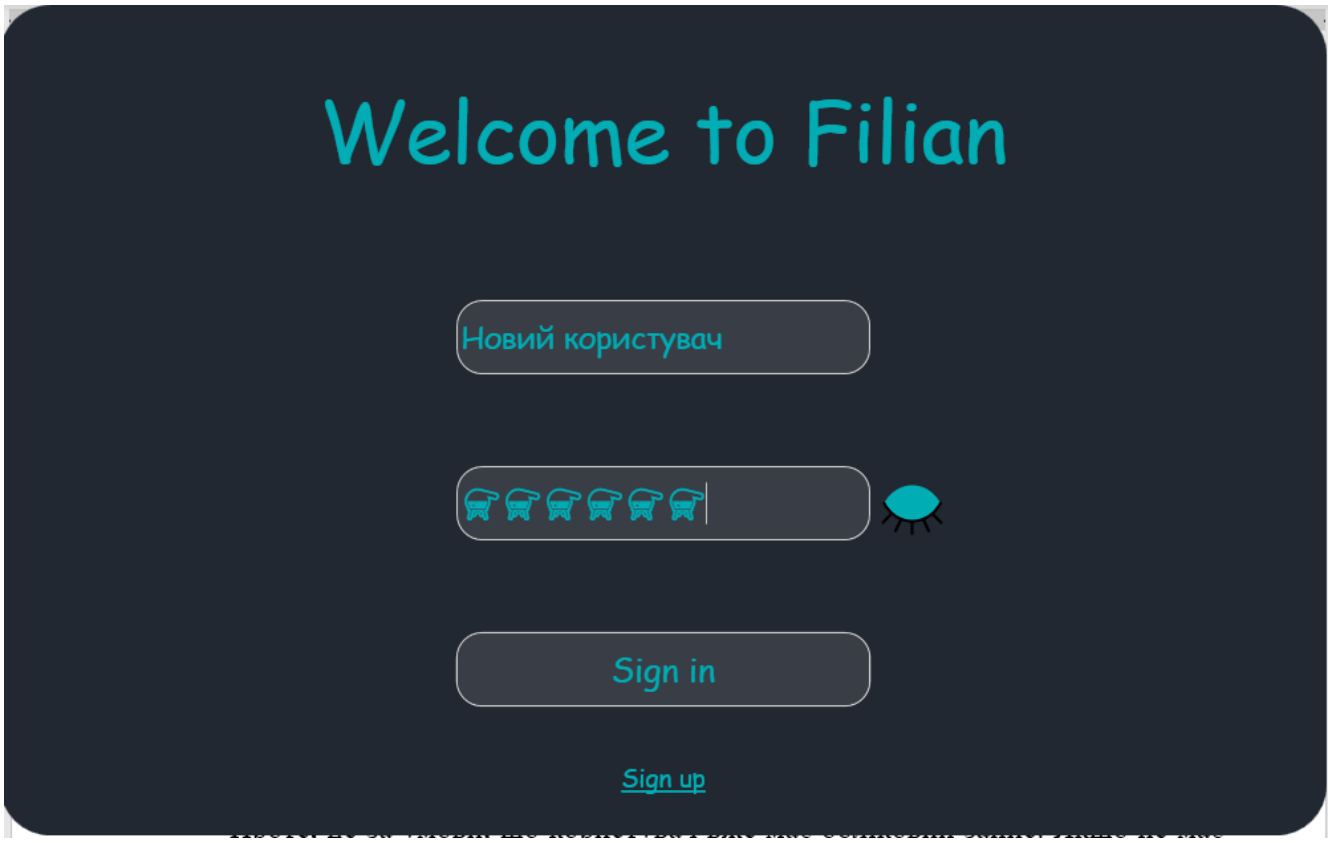

Рисунок 4.6 – Вікно авторизації

Проте, лише за наявності облікового запису користувач зможе авторизуватись. Якщо облікового запису не має – користувач може перейти в вікно реєстрації (рис. 4.7) за допомогою кнопки «Sing up» внизу вікна та створити обліковий запис вписавши відповідні дані.

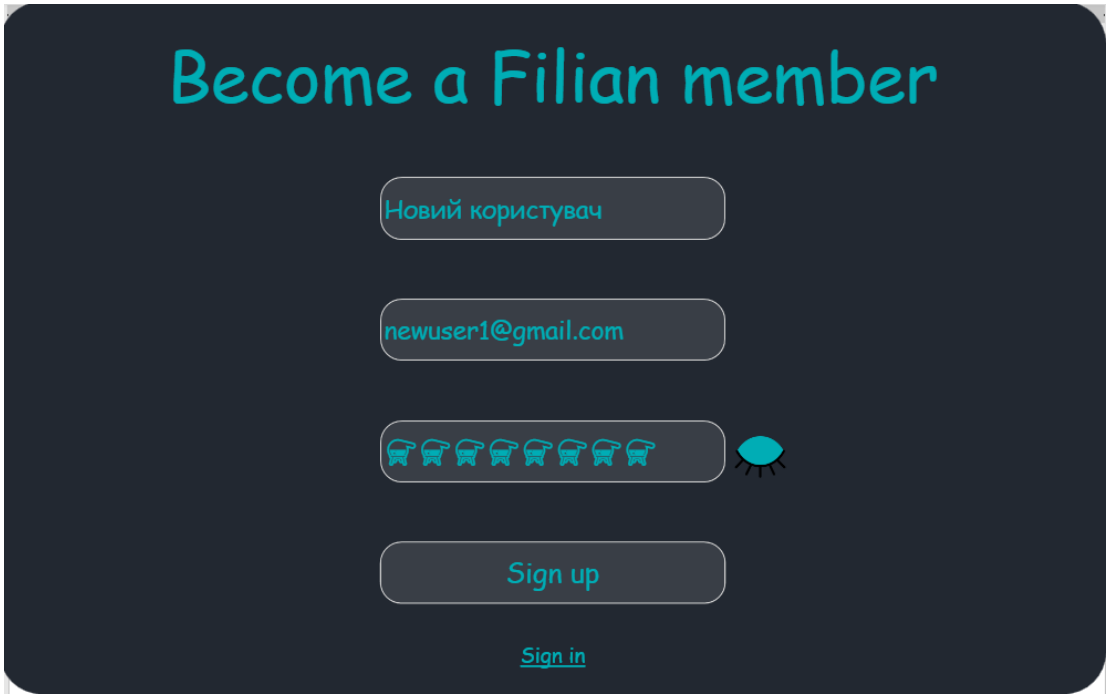

Рисунок 4.7 – Вікно реєстрації

Після успішного входу система автоматично відкриє головне вікно. В головному вікні програма одразу запропонує користувачеві обрати мову для тестування (рис. 4.8).

Рисунок 4.8 – Головне вікно з формою вибору мови

Після того, як користувач вибере мову, йому стане доступним на вибір десять різних типів тестувань(один з двох, один з чотирьох, один з чотирьох: текст, один з чотирьох: слухати, вірно/не вірно, написання з картинкою, написання зі звуком, переклад – текст, переклад – вимова, знайди пару – переклад) , за допомогою яких він може закрити свої потреби тестування знань іноземної мови (рис. 4.9).

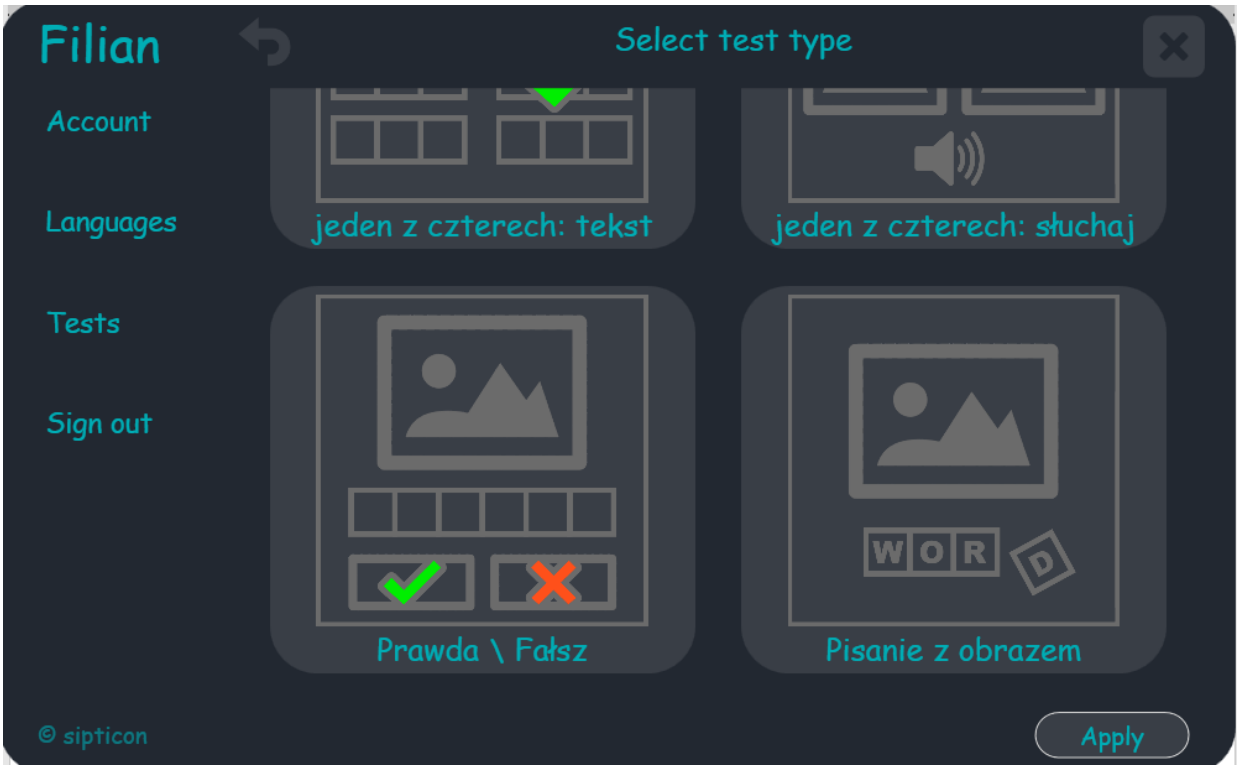

Рисунок 4.9 – Головне вікно з формою вибору типу тесту польською мовою

Після вибору типу тестування система відкриє форму з переліком доступних тем (рис. 4.10). Тому наступним кроком необхідно обрати одну з двадцяти доступних тем. На вибір користувачеві наразі доступний такий перелік тем: тварини, мистецтво, птахи, одяг та аксесуари, динозаври, риби та молюски, їжа, географія та архітектура, дім там офіс, людина, комахи, для малюків, природа та навколишнє середовище, предмети, рослини, рептилії, спорт, транспорт, дієслова, який. Кожна з цих тем є унікальною та має великий набір слів для тестування знань з іноземної мови, що дозволить системі в певній мірі покрити потреби користувачів будь-якого віку та роду діяльності.

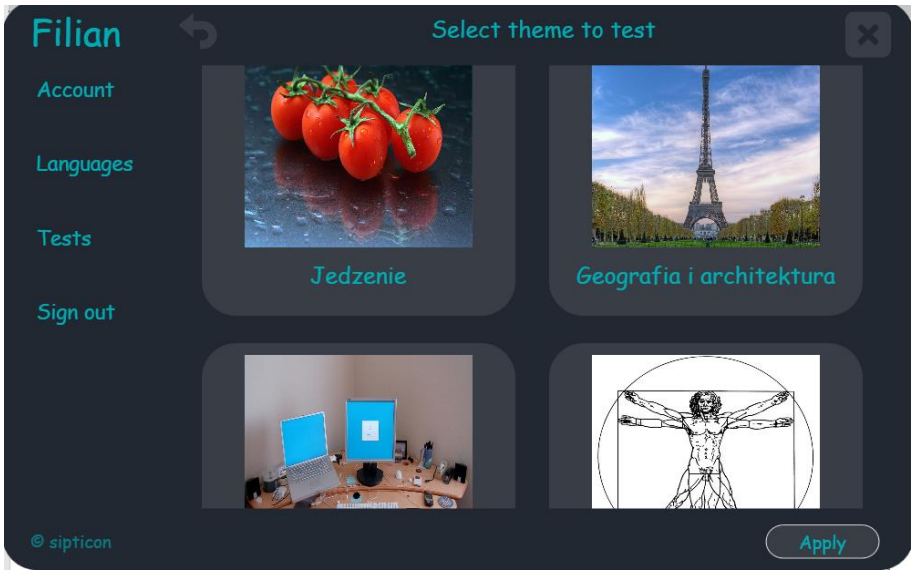

Рисунок 4.10 – Головне вікно з формою вибору теми польською мовою

Різноманітність тем є важливою, адже дозволяє протестувати знання різних користувачів. Для кожної теми доступна різна кількість підтем, наприклад: для теми «Тварини» користувач може обрати будь-які існуючі підтеми, як от «Кажани», «Породи кішок», «Породи собак», «Домашні тварини», «Парнокопитні», «Комахоїди», «Сумчасті», «Непарнокопитні», «Хижаки», «Примати», «Гризуни», «Морські ссавці», «Дикі кішки».

Після вибору необхідних підтем розпочнеться тестування (рис. 4.11).

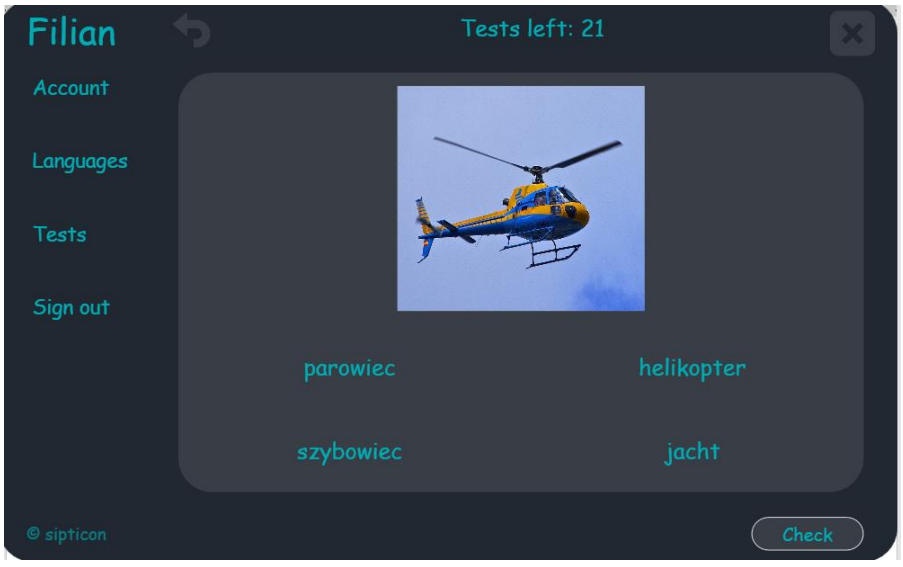

Рисунок 4.11 – Тестування за обраними параметрами польською мовою

Вгорі вікна користувач бачитиме кількість тестів, яка залишилась до кінця тестування (рис. 4.12). Підтвердження вибору всього відбувається за допомогою кнопки «Apply», перевірка правильності вибору під час тестування – «Check», перехід до наступного тесту – «Next».

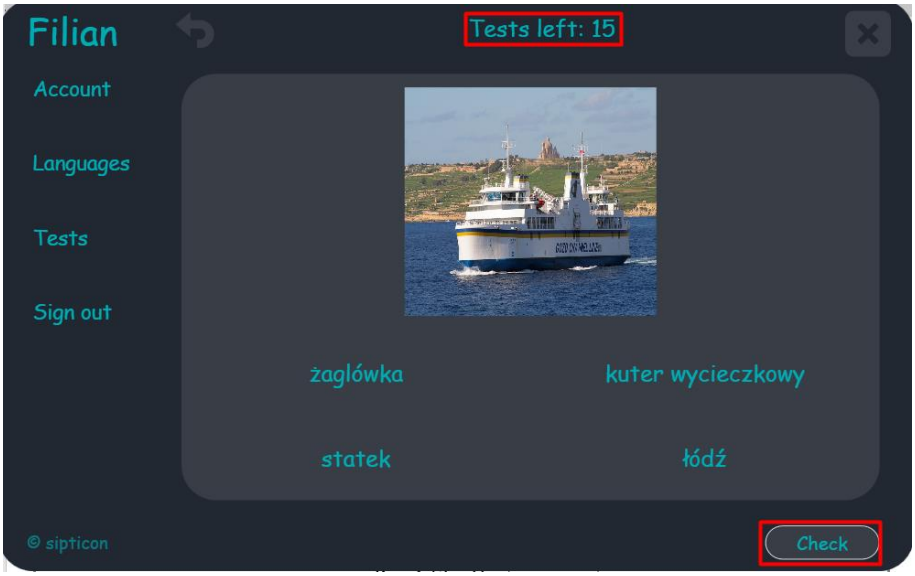

Рисунок 4.12 – Розміщення інформації про кількість тестів та кнопки у вікні тесту польською мовою

Після закінчення тестування, користувач отримає інформацію про проходження тесту та кількість коректних відповідей (рис. 4.13).

Рисунок 4.13 – Результат проходження тестування

Після цього система знову запропонує обрати тему. Також користувач, в головному вікні, може перейти в особистий кабінет з інформацією про обліковий запис (рис. 4.14), змінити мову та тип тестування за потреби, а також вийти з системи. Після виходу всі дані зберігаються в хмарному середовищі.

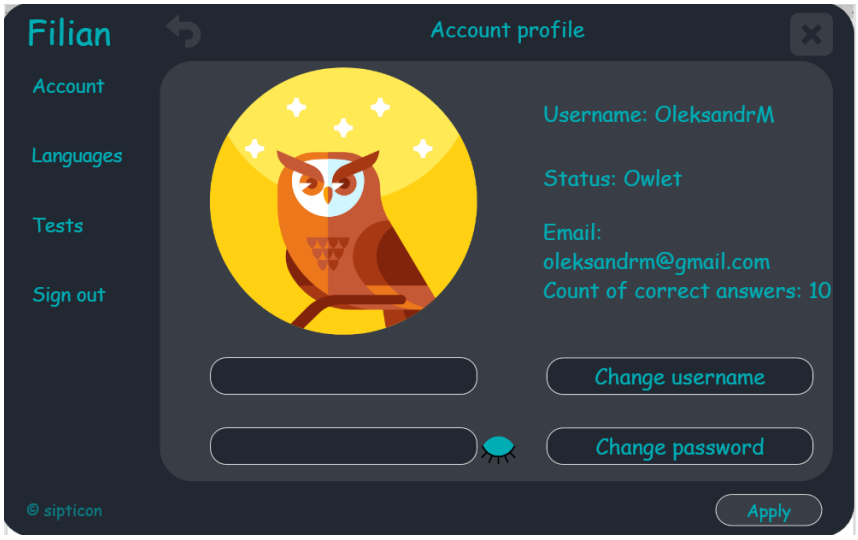

Рисунок 4.14 – Особистий кабінет користувача

Таким чином відбувається тестування знань іноземної мови.

4.3 Висновки

У четвертому розділі було проведено тестування програмних засобів системи тестування знань з іноземної мови. Було перевірено ресурсозатратність програми а також валідацію полів при реєстрації та авторизації користувача.

Перевірено відображення елементів інтерфейсу та отриманих даних з бази. Також було розроблено інструкцію користувача по встановленню та початковій експлуатації програмного продукту.

У результаті тестування розроблена системи показала цілковиту працездатність програмного продукту та відповідність поставленому технічному завданню.

ВИСНОВКИ

У бакалаврській дипломній роботі було розроблено систему тестування знань з іноземної мови «Filian» для покращення проведення тестування та надання користувачеві можливості самостійно формувати тестування за власними потребами.

Проведено аналіз стану систем тестування знань з іноземної мови, розглянуто існуючі аналоги, такі як: Busuu, Drops, LingQ, Preply та Duolingo. Визначено їхні переваги та недоліки в порівнянні з розроблюваної системою та встановлено, що розроблюваний програмний засіб переважає у функціональності розглянуті аналоги.

Базуючись на результатах аналізу та порівняння, було поставлено основні задачі бакалаврської дипломної роботи та обрано методи розробки системи. Проведено варіантний аналіз та обґрунтування вибору засобів реалізації програмного продукту. В якості мови програмування було обрано мову програмування C#. В якості середовища програмної розробки – Microsoft Visual Studio 2022. Для створення графічного інтерфейсу обрано технологію WPF. База даних розроблена за допомогою Microsoft SQL Server та розміщена на AWS RDS.

Відповідно до поставлених задач було:

– на прикладі UML-діаграм діяльності розроблено модель системи для тестування знань з іноземної мови, яка включає основний функціонал застосунку;

– програмно реалізовано функціонал системи для тестування знань з іноземної мови;

– розроблено програмні компоненти та систему візуалізації на основі запропонованого методу;

– проведено тестування програми, яке показало цілковиту працездатність програмного продукту та відповідність поставленому технічному завданню.

Крім того, було розроблено метод формування тестів та проходження тестування, який дозволяє користувачеві самостійно формувати тестування за власними потребами.

Також було розроблено інструкцію користувача.

Бакалаврську дипломну роботу було оформлено відповідно до методичних вказівок [24].

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Важливість вивчення мов [Електронний ресурс] – Режим доступу до ресурсу: https://clubinternational.com.ua/?p=233.

2. Де самостійно вивчати іноземну мову [Електронний ресурс] – Режим доступу до ресурсу: https://happymonday.ua/13-populyarnyh-mobilnyh-dodatkiv.

3. Монастирський О. М. Розробка програми тестування знань з іноземної мови / О. М. Монастирський, В. В. Войтко // Матеріали LIII Науково-технічної конференції підрозділів Вінницького національного технічного університету, Вінниця, 2024. [Електронний ресурс] – Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-024/paper/view/20400/16962

4. Кожем’яко Н. В. Тестування як форма перевірки знань студентів під час вивчення іноземної мови у вищих навчальних закладах / Н. В. Кожем’яко // Науковий часопис НПУ імені М. П. Драгоманова, Київ, 2019. [Електронний ресурс] – Режим доступу до ресурсу: http://nc-s5.npu.edu.ua/article/view/161611.

5. Топ-10 застосунків для вивчення іноземних мов [Електронний ресурс] – Режим доступу до ресурсу: https://speka.media/top-10-zastosunkiv-dlya-vivcennya-inozemnix-mov-pkl0d9.

6. Busuu a Cheg Servise [Електронний ресурс] – Режим доступу до ресурсу: https://www.busuu.com/.

7. Busuu Wikipedia [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Busuu.

8. LanguageDrops [Електронний ресурс] – Режим доступу до ресурсу: https://languagedrops.com/.

9. Drops Wikipedia [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Drops_(app).

10. LingQ [Електронний ресурс] – Режим доступу до ресурсу: https://www.lingq.com/uk/.

11. Preply [Електронний ресурс] – Режим доступу до ресурсу: https://preply.com/.

12. Duolingo [Електронний ресурс] – Режим доступу до ресурсу: https://uk.duolingo.com/.

13. Duolingo Wikipedia [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Duolingo.

14. C Sharp [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/C_Sharp.

15. Del Sole A. Hidden WPF. Secrets for Creating Great Applications in Windows Presentation Foundation» – 2013. – 1173 с.

16. Alogni R. AWS. The Most Complete Guide to Amazon Web Services from Beginner to Advanced Level – 2019. – 224 с.

17. Microsoft Visual Studio [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Microsoft_Visual_Studio.

18. XAML [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/XAML.

19. Chelliah P. R., Subramanian H., Architectural Patterns. Uncover Essential Patterns in the Most Indispensable Realm of Enterprise Architecture – 2017. – 468 с.

20. Реляційна база даних [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Реляційна_база_даних.

21. NoSQL [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/NoSQL.

22. Microsoft SQL server [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Microsoft_SQL_Server.

23. Oxyplot [Електронний ресурс] – Режим доступу до ресурсу: https://oxyplot.github.io/.

24. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів студентів спеціальності 121 "Інженерія програмного забезпечення" / Уклад. О. Н. Романюк, Г. О. Черноволик – Вінниця: ВНТУ, 2022. – 52с.

ДОДАТКИ

66

Додаток А – Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
«12» березня 2024 р.

Технічне завдання
на бакалаврську дипломну роботу
«Розробка методу та програмних засобів системи тестування знань з іноземної мови»
за спеціальністю 121 – Інженерія програмного забезпечення

Керівник бакалаврської дипломної роботи:
к.т.н., доц. Войтко В. В.
«12» березня 2024 р.

Виконав:
студент гр. 4ПІ-20б Монастирський О. М.
«12» березня 2024 р.

Вінниця – 2024

1. Найменування та галузь застосування

Бакалаврська дипломна робота: «Розробка методу та програмних засобів системи тестування знань з іноземної мови».

Галузь застосування – навчальні інтернет ресурси.

2. Підстава для розробки.

Завдання на роботу, яке затверджене на засіданні кафедри програмного забезпечення – протокол №17 від 20 березня 2024 р.

3. Мета та призначення розробки.

Метою бакалаврської дипломної роботи є підвищення якості тестування знань з іноземних мов за рахунок розробки та використання спеціалізованої системи тестування знань, яка використовує різні типи тестів, що дозволить підвищити об’єктивність оцінювання знань за результатами тестування.

Призначення роботи – розробка програмного продукту для тестування знань з іноземної мови.

4. Вихідні дані для проведення НДР

1. Alogni R. AWS. The Most Complete Guide to Amazon Web Services from Beginner to Advanced Level – 2019. – 224 с.
2. Del Sole A. Hidden WPF. Secrets for Creating Great Applications in Windows Presentation Foundation – 2013. – 1173 с.
3. Chelliah P. R., Subramanian H., Murali A., Dr. Kayarvizhy N. Architectural Patterns. Uncover Essential Patterns in the Most Indispensable Realm of Enterprise Architecture – 2017. – 468.

5. Технічні вимоги

Комп’ютер з 64-розрядним процесором та тактовою частотою 2 ГГц. Об’єм оперативної пам’яті 4 ГБ, розмір жорсткого диску 128 ГБ. Операційна система

Windows 7/8/10/11.

6. Конструктивні вимоги.

Додаток має відповідати ергономічним та технічним вимогам. Текстова та графічна документація повинна відповідати стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

– пояснювальна записка до бакалаврської дипломної роботи;

– технічне завдання;

– лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи
1	Аналіз стану розвитку засобів системи тестування знань з іноземної мови та постановка задач дослідження	14.03.24 – 24.03.24
2	Розробка методу, моделі та алгоритмів програмного продукту	25.03.24 –15.04.24
3	Розробка програмних засобів реалізації системи з використанням засобів тестування знань з іноземної мови	15.04.24 – 30.04.24
4	Тестування програми	01.05.24 – 10.05.24
5	Оформлення матеріалів до захисту БДР	11.05.24 – 30.05.24

10 Порядок контролю та прийняття.

Всі етапи бакалаврської роботи контролюються науковим керівником згідно плану по виконанню роботи. Прийняття бакалаврської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком захисту. Дозволяється корегування бакалаврської дипломної роботи.

69

Додаток Б – ПРОТОКОЛ ПЕРЕВІРКИ БАКАЛАВРСЬКОЇ ДИПЛОМНОЇ РОБОТИ НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: «Розробка методу та програмних засобів системи тестування знань з іноземної мови»

Тип роботи: БДР

Підрозділ : кафедра програмного забезпечення, ФІТКІ

Науковий керівник: Войтко В.В.

Оригінальність	97.8%
Схожість	2.2%

Аналіз звіту подібності

■ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку: _____ Черноволик Г. О.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи: Siptcon Монастирський О.М

Керівник роботи: _____ Войтко В.В

Додаток В – Лістинг програми

Лістинг модуля MainViewModel.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using Filian.Core;
using System.Windows;
using System.Windows.Controls;
using Filian.MVVM.Model;
using Filian.MVVM.View;
using System.Windows.Media.Imaging;
using Image = System.Windows.Controls.Image;
using System.Linq;
using Application = System.Windows.Application;
using System.Data.SqlClient;
using System.IO;

namespace Filian.MVVM.ViewModel
{
   public class MainViewModel :  ViewModel, IDisposable
   {
      public RelayCommand OpenLanguagesViewCommand { get; set; }
      public RelayCommand OpenTestsViewCommand { get; set; }
      public RelayCommand OpenUserAccountViewCommand { get; set; }
      public RelayCommand ApplyCommand { get; set; }
      public RelayCommand ExitCommand { get; set; }
      public RelayCommand BackCommand { get; set; }
      public RelayCommand SignOutCommand { get; set; }
      public LanguagesViewModel LanguagesVm { get; set; }
      public TestsViewModel TestsVm { get; set; }
      public ThemesViewModel ThemesVm { get; set; }
```

```
        public UnderThemesViewModel UnderThemesVm { get; set; }
        public OneFromTwoViewModel OneFromTwoVm { get; set; }
        public OneFromFourViewModel OneFromFourVm { get; set; }
        public OneFromFourTextViewModel OneFromFourTextVm { get; set; }
        public OneFromFourListeningViewModel OneFromFourListeningVm { get; set; }
        public TrueOrFalseViewModel TrueOrFalseVm { get; set; }
        public SpellWithPictureViewModel SpellWithPictureVm { get; set; }
        public SpellWithVoiceViewModel SpellWithVoiceVm { get; set; }
        public TranslationTextViewModel TranslationTextVm { get; set; }
        public TranslationPronunciationViewModel TranslationPronunciationVm { get; set; }
        public FindPairTranslationViewModel FindPairTranslationVm { get; set; }
        public UserAccountViewModel UserAccountVm { get; set; }
        public UserAuthorizationView UserAuthorizationView { get; set; }
        private object _currentView;
        private static int _themeId;
        private int _testId;
        private new int _countOfTests = 0;
        private int _countOfCorrectAnswers = 0;

        private bool _backButtonActive = true;
        private bool _navigatePanelButtonsActive = true;

        private bool _isAnswerShown = false;

        private string _pageHeader = "Select language to test";
        private string _textOnMainButton = "Apply";

        private static List<int> _underThemeIds = new List<int>();
        private readonly Stack _previousViews;

        public static readonly string _correctAnswerImagesource =
Path.GetFullPath(@"FilianFiles\Test_Icons\grey\Checkbox_Yes.png");
        public static readonly string _wrongAnswerImagesource =
Path.GetFullPath(@"FilianFiles\Test_Icons\grey\Checkbox_No.png");
```

```
private string backButtonImagePath = Path.GetFullPath(@"Images\back.png");
private string closeButtonImagePath = Path.GetFullPath(@"Images\close.png");

public string BackButtonImagePath
{
   get => backButtonImagePath;
   set
   {
      backButtonImagePath = Path.GetFullPath(value);
      OnPropertyChanged("BackButtonImagePath");
   }
}

public string CloseButtonImagePath
{
   get => closeButtonImagePath;
   set
   {
      closeButtonImagePath = Path.GetFullPath(value);
      OnPropertyChanged("CloseButtonImagePath");
   }
}

public static string userName;
public static string userEmail;
public static string userStatus;
public static string countOfCorrectAnswers;

private int _userCountOfCorrectAnswers;

public object CurrentView
{
   get => _currentView;
```

```
    set
    {
        _currentView = value;
        OnPropertyChanged();
    }
}
public int ThemeId
{
    get => _themeId;
    set => _themeId = value;
}
public static int LanguageId { get; set; } = 2;

public string PageHeader
{
    get => _pageHeader;
    set
    {
        _pageHeader = value;
        OnPropertyChanged("PageHeader");
    }
}

public string TextOnMainButton
{
    get => _textOnMainButton;
    set
    {
        _textOnMainButton = value;
        OnPropertyChanged("TextOnMainButton");
    }
}

public List<int> UnderThemeIds
```

```
        {
            get => _underThemeIds;
            set => _underThemeIds = value;
        }

        public MainViewModel()
        {
            Log.Info("Application started.");

            LanguagesVm = new LanguagesViewModel();
            TestsVm = new TestsViewModel();

            _userCountOfCorrectAnswers = Convert.ToInt32(countOfCorrectAnswers);

            _previousViews = new Stack();

            ChangeView(LanguagesVm);

            OpenLanguagesViewCommand = new RelayCommand(o => { NavigateToLanguagesView(); });
            OpenTestsViewCommand = new RelayCommand(o => { NavigateToTestsView(); });
            OpenUserAccountViewCommand = new RelayCommand(o => { NavigateToUserAccountView(); });
            ApplyCommand = new RelayCommand(o => { Apply_Click(); });
            ExitCommand = new RelayCommand(o => { Exit_Click(); });
            BackCommand = new RelayCommand(o => { Back_Click(); });
            SignOutCommand = new RelayCommand(o => { SignOut(); });
        }

        private void Apply_Click()
        {
            if (CurrentView == LanguagesVm)
            {
                var languageId = LanguagesVm.SelectedLanguage;
```

```
    if (languageId != null)
    {
        LanguageId = languageId.Id;
        TestsVm = new TestsViewModel();
        ChangeView(TestsVm);
        PageHeader = "Select test type";
    }
}
else if (CurrentView == TestsVm)
{
    var test = TestsVm.SelectedTest;
    if (test != null)
    {
        _testId = test.Id;
        ThemesVm = new ThemesViewModel();
        ChangeView(ThemesVm);
        PageHeader = "Select theme to test";
    }
}
else if (CurrentView == ThemesVm)
{
    var theme = ThemesVm.SelectedTheme;
    if (theme != null)
    {
        ThemeId = theme.Id;
        UnderThemesVm = new UnderThemesViewModel();
        ChangeView(UnderThemesVm);
        PageHeader = "Select underthemes to test";
    }
}
else if (CurrentView == UnderThemesVm)
{
    var underThemes = UnderThemesVm.SelectedItems;
    if (underThemes != null)
```

```
            {
                foreach (Theme underTheme in underThemes)
                {
                    UnderThemeIds.Add(underTheme.Id);
                }

                switch (_testId)
                {
                    case 1 :
                        OneFromTwoTest oneFromTwoTest = new
OneFromTwoTest(UnderThemeIds);
                        OneFromTwoVm = new OneFromTwoViewModel();
                        _countOfTests = OneFromTwoVm.CountOfTests - 1;
                        ChangeView(OneFromTwoVm);
                        break;
                    case 2:
                        OneFromFourTest oneFromFourTest = new
OneFromFourTest(UnderThemeIds);
                        OneFromFourVm = new OneFromFourViewModel();
                        _countOfTests = OneFromFourVm.CountOfTests - 1;
                        ChangeView(OneFromFourVm);
                        break;
                    case 3:
                        OneFromFourTextTest oneFromFourTextTest = new
OneFromFourTextTest(UnderThemeIds);
                        OneFromFourTextVm = new OneFromFourTextViewModel();
                        _countOfTests = OneFromFourTextVm.CountOfTests - 1;
                        ChangeView(OneFromFourTextVm);
                        break;
                    case 4:
                        OneFromFourListeningTest oneFromFourListeningTest = new
OneFromFourListeningTest(UnderThemeIds);
                        OneFromFourListeningVm = new OneFromFourListeningViewModel();
                        _countOfTests = OneFromFourListeningVm.CountOfTests - 1;
```

```
                ChangeView(OneFromFourListeningVm);
                break;
            case 5:
                TrueOrFalseTest trueOrFalseTest = new TrueOrFalseTest(UnderThemeIds);
                TrueOrFalseVm = new TrueOrFalseViewModel();
                _countOfTests = TrueOrFalseVm.CountOfTests - 1;
                ChangeView(TrueOrFalseVm);
                break;
            case 6:
                SpellWithPictureTest spellWithPictureTest = new
SpellWithPictureTest(UnderThemeIds);
                SpellWithPictureVm = new SpellWithPictureViewModel();
                _countOfTests = SpellWithPictureVm.CountOfTests - 1;
                ChangeView(SpellWithPictureVm);
                break;
            case 7:
                SpellWithVoiceTest spellWithVoiceTest = new
SpellWithVoiceTest(UnderThemeIds);
                SpellWithVoiceVm = new SpellWithVoiceViewModel();
                _countOfTests = SpellWithVoiceVm.CountOfTests - 1;
                ChangeView(SpellWithVoiceVm);
                break;
            case 8:
                TranslationTextTest translationTextTest = new
TranslationTextTest(UnderThemeIds);
                TranslationTextVm = new TranslationTextViewModel();
                _countOfTests = TranslationTextVm.CountOfTests - 1;
                ChangeView(TranslationTextVm);
                break;
            case 9:
                TranslationPronunciationTest translationPronunciationTest = new
TranslationPronunciationTest(UnderThemeIds);
                TranslationPronunciationVm = new TranslationPronunciationViewModel();
                _countOfTests = TranslationPronunciationVm.CountOfTests - 1;
```

```
                        ChangeView(TranslationPronunciationVm);
                        break;
                    case 10:
                        FindPairTranslationTest findPairTranslationTest = new
FindPairTranslationTest(UnderThemeIds);
                        FindPairTranslationVm = new FindPairTranslationViewModel();
                        _countOfTests = FindPairTranslationVm.CountOfTests - 1;
                        ChangeView(FindPairTranslationVm);
                        break;
                }
                PageHeader = $"Tests left: {_countOfTests}";
                _backButtonActive = false;
                _navigatePanelButtonsActive = false;
                TextOnMainButton = "Check";
            }
        }
        else
        {
            if (_testId == 1)
            {
                MoveToTheNextStep(OneFromTwoVm.Word,
OneFromTwoViewModel.SelectedImage);
            }
            else if (_testId == 2)
            {
                MoveToTheNextStep(OneFromFourVm.Word,
OneFromFourViewModel.SelectedImage);
            }
            else if (_testId == 3)
            {
                MoveToTheNextStep(OneFromFourTextViewModel.CorrectTranslation,
OneFromFourTextViewModel.SelectedWord);
            }
            else if (_testId == 4)
```

```
                {
                    MoveToTheNextStep(OneFromFourListeningVm.Word,
OneFromFourListeningViewModel.SelectedImage);
                }
                else if (_testId == 5)
                {
                    MoveToTheNextStep((TrueOrFalseVm.ShownTranslation                    ==
TrueOrFalseVm.CorrectTranslation).ToString(),
TrueOrFalseViewModel.IsCurrentWordRight.ToString());
                }
                else if (_testId == 6)
                {
                    MoveToTheNextStep(SpellWithPictureVm.Translation.ToLower(),
SpellWithPictureVm.Answer.ToLower());
                }
                else if (_testId == 7)
                {
                    MoveToTheNextStep(SpellWithVoiceVm.Translation.ToLower(),
SpellWithVoiceVm.Answer.ToLower());
                }
                else if (_testId == 8)
                {
                    MoveToTheNextStep(TranslationTextVm.Word.ToLower(),
TranslationTextVm.Answer.ToLower());
                }
                else if (_testId == 9)
                {
                    MoveToTheNextStep(TranslationPronunciationVm.Word.ToLower(),
TranslationPronunciationVm.Answer.ToLower());
                }
                else if (_testId == 10)
                {
                    MoveToTheNextStep(FindPairTranslationVm.CorrectWord,
FindPairTranslationViewModel.SelectedWord);
```

```
        }
    }
}

private void Exit_Click()
{
  Dispose();
  Application.Current.Shutdown();
}

private void Back_Click()
{
    try
    {
        if (_backButtonActive && _previousViews.Count > 1)
        {
            CurrentView = _previousViews.Pop();
            Log.Info("Successfully returning to the previous page.");
        }
        else
        {
            CreateUserNotificationBox("Can't go back!", "You can't navigate to back!");
        }
    }
    catch (Exception ex)
    {
        Log.Error("Failed while return to the previous page: ",ex);
    }
}

private void SignOut()
{
    if (_navigatePanelButtonsActive)
    {
```

```
                UpdateUserInfo();
                UserAuthorizationView = new UserAuthorizationView();
                UserAuthorizationView.Show();
                Application.Current.Windows.OfType<MainWindow>().First().Close();
            }
            else
                CreateUserNotificationBox("You are in the test!", "You can't sign out while test is
running!");
        }

        private void NavigateToLanguagesView()
        {
            if (_navigatePanelButtonsActive)
            {
                ChangeView(LanguagesVm);
                PageHeader = "Select language to test";
            }
            else
                CreateUserNotificationBox("You are in the test!", "You can't navigate to other tabs
while test is running!");
        }

        private void NavigateToTestsView()
        {
            if (_navigatePanelButtonsActive)
            {
                ChangeView(TestsVm);
                PageHeader = "Select test type";
            }
            else
                CreateUserNotificationBox("You are in the test!", "You can't navigate to other tabs
while test is running!");
        }
```

```
        private void NavigateToUserAccountView()
        {
            if (_navigatePanelButtonsActive)
            {
                UserAccountVm = new UserAccountViewModel()
                {
                    UserName = userName,
                    UserEmail = userEmail,
                    UserStatus = userStatus,
                    CountOfCorrectAnswers = _userCountOfCorrectAnswers.ToString()
                };
                ChangeView(UserAccountVm);
                PageHeader = "Account profile";
            }
            else
                CreateUserNotificationBox("You are in the test!", "You can't navigate to other tabs while test is running!");
        }

        private void ChangeView(object newView)
        {
            try
            {
                if(_backButtonActive)
                    _previousViews.Push(CurrentView);
                CurrentView = newView;
                Log.Info($"View successfully changed to {newView}");
            }
            catch (Exception ex)
            {
                Log.Error("Exception while changing view: ",ex);
            }
        }
```

```
        private void CheckResultOfChoice(string correctAnswer, string selectedAnswer)
        {
            if (selectedAnswer.Contains(correctAnswer + ".") || selectedAnswer == correctAnswer)
            {
                _countOfCorrectAnswers++;
                Grid.Children.Add(CreateImage(_correctAnswerImagesource));
            }
            else
                Grid.Children.Add(CreateImage(_wrongAnswerImagesource));
        }

        private void MoveToTheNextStep(string correctAnswer, string selectedAnswer)
        {
            if (String.IsNullOrEmpty(selectedAnswer))
                CreateUserNotificationBox("You didn't select the answer!", "Please, select answer!");
            else if (_countOfTests == 1)
            {
                CheckResultOfChoice(correctAnswer, selectedAnswer);
                CreateUserNotificationBox("The test is completed!", $"Count of correct answers -
{_countOfCorrectAnswers}");
                EndOfTest();
            }
            else
            {
                if (!_isAnswerShown)
                {
                    CheckResultOfChoice(correctAnswer, selectedAnswer);
                    _isAnswerShown = true;
                    TextOnMainButton = "Next";
                }
                else
                    MoveToTheNextTest();
            }
```

```
}
private void MoveToTheNextTest()
{
    object newView = null;
    switch (_testId)
    {
        case 1:
            newView = new OneFromTwoView();
            OneFromTwoViewModel.SelectedImage = "";
            break;
        case 2:
            newView = new OneFromFourView();
            OneFromFourViewModel.SelectedImage = "";
            break;
        case 3:
            newView = new OneFromFourTextView();
            OneFromFourTextViewModel.SelectedWord = "";
            break;
        case 4:
            newView = new OneFromFourListeningView();
            OneFromFourListeningViewModel.SelectedImage = "";
            break;
        case 5:
            newView = new TrueOrFalseView();
            TrueOrFalseViewModel.IsCurrentWordRight = null;
            break;
        case 6:
            newView = new SpellWithPictureView();
            SpellWithPictureVm.Answer = "";
            break;
        case 7:
            newView = new SpellWithVoiceView();
            SpellWithVoiceVm.Answer = "";
            break;
```

```
            case 8:
                newView = new TranslationTextView();
                TranslationTextVm.Answer = "";
                break;
            case 9:
                newView = new TranslationPronunciationView();
                TranslationPronunciationVm.Answer = "";
                break;
            case 10:
                newView = new FindPairTranslationView();
                FindPairTranslationViewModel.SelectedWord = "";
                break;
        }
        _countOfTests--;
        PageHeader = $"Tests left: {_countOfTests}";
        ChangeView(newView);
        _isAnswerShown = false;
        TextOnMainButton = "Check";
    }

    private Image CreateImage(string pathToImage)
    {
        Image myImage = new Image();
        BitmapImage myBitmapImage = new BitmapImage();
        myBitmapImage.BeginInit();
        myBitmapImage.UriSource = new Uri(Path.GetFullPath(pathToImage));
        myBitmapImage.EndInit();
        myImage.Source = myBitmapImage;
        myImage.Width = 70;
        myImage.Height = 70;
        Grid.SetColumn(myImage,Column);
        Grid.SetRow(myImage, Row);
        return myImage;
    }
```

```
        private void EndOfTest()
        {
            Log.Info($"Test {CurrentView} successfully finished.");
            ChangeView(TestsVm);
            _previousViews.Clear();
            _userCountOfCorrectAnswers += _countOfCorrectAnswers;
            _countOfCorrectAnswers = 0;
            _countOfTests = 0;
            _backButtonActive = true;
            _navigatePanelButtonsActive = true;
            UnderThemeIds = new List<int>();
            TextOnMainButton = "Apply";
            PageHeader = "Select test type";
        }

        public void UpdateUserInfo()
        {
            string newUserStatus = "";
            string sqlChangeUserInfo = "";
            if (_userCountOfCorrectAnswers >= 200 && _countOfCorrectAnswers < 500)
                newUserStatus = "Owl";
            else if (_userCountOfCorrectAnswers >= 500 && _countOfCorrectAnswers < 1000)
                newUserStatus = "Old owl";
            else if (_userCountOfCorrectAnswers >= 1000)
                newUserStatus = "Legend owl";
            else
                newUserStatus = "";

            sqlChangeUserInfo = !string.IsNullOrEmpty(newUserStatus)
                ? $"UPDATE users SET correct_answers='{_userCountOfCorrectAnswers}',
user_status='{newUserStatus}' WHERE username = '{userName}'"
                : $"UPDATE users SET correct_answers='{_userCountOfCorrectAnswers}' WHERE
username = '{userName}'";
```

```
            SqlConnection sqlConnection = new SqlConnection(sqlConnectionString);
            sqlConnection.Open();

            SqlCommand sqlCommand = new SqlCommand(sqlChangeUserInfo, sqlConnection);
            try
            {
                sqlCommand.ExecuteNonQuery();
            }
            catch (Exception ex)
            {
                Log.Error("Error while trying to change username: " + ex);
            }
            finally
            {
                sqlConnection.Close();
            }
        }

        public void Dispose()
        {
            try
            {
                UpdateUserInfo();
                Log.Info("Application successfully closed.");
            }
            catch (Exception ex)
            {
                Log.Error("Failed while closing application", ex);
            }
        }
    }
}
```

Додаток Г – Ілюстративний матеріал

ІЛЮСТРАТИВНА ЧАСТИНА

РОЗРОБКА МЕТОДУ ТА ПРОГРАМНИХ ЗАСОБІВ СИСТЕМИ ТЕСТУВАННЯ ЗНАНЬ З ІНОЗЕМНОЇ МОВИ

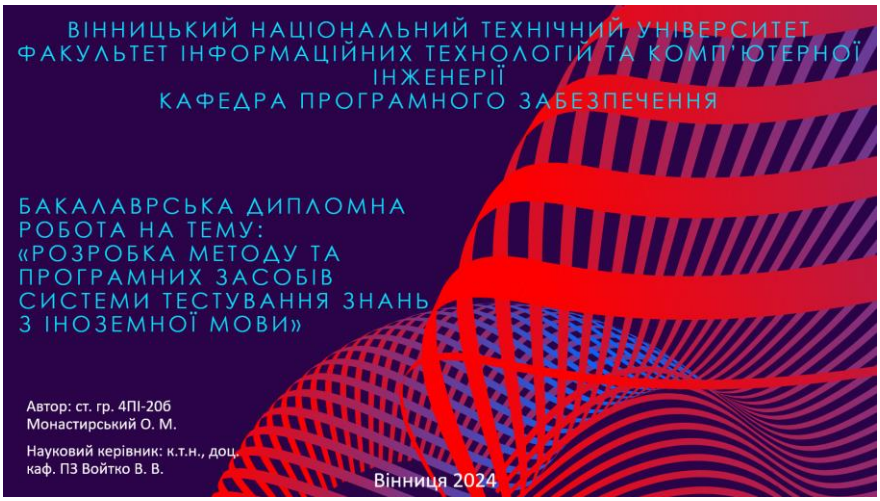

Рисунок Г.1 – Титульний слайд

Рисунок Г.2 – Слайд «Актуальність розробки»

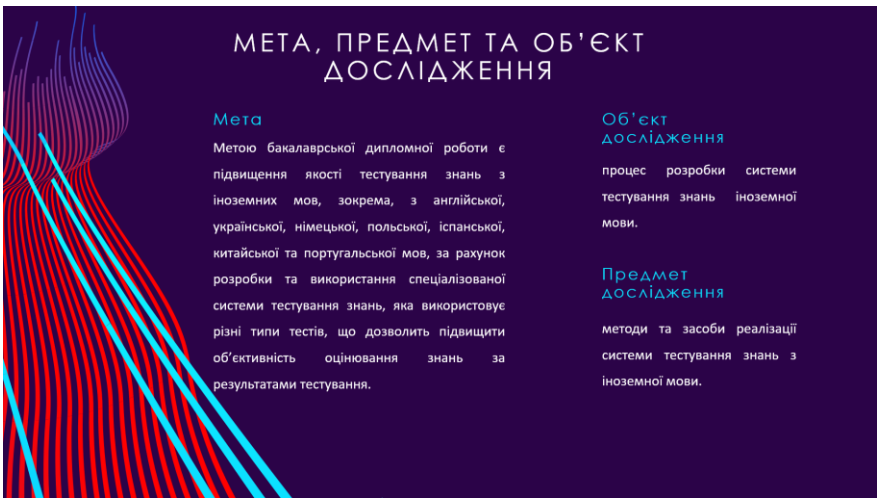

Рисунок Г.3 – Слайд «Мета, предмет та об’єкт дослідження»

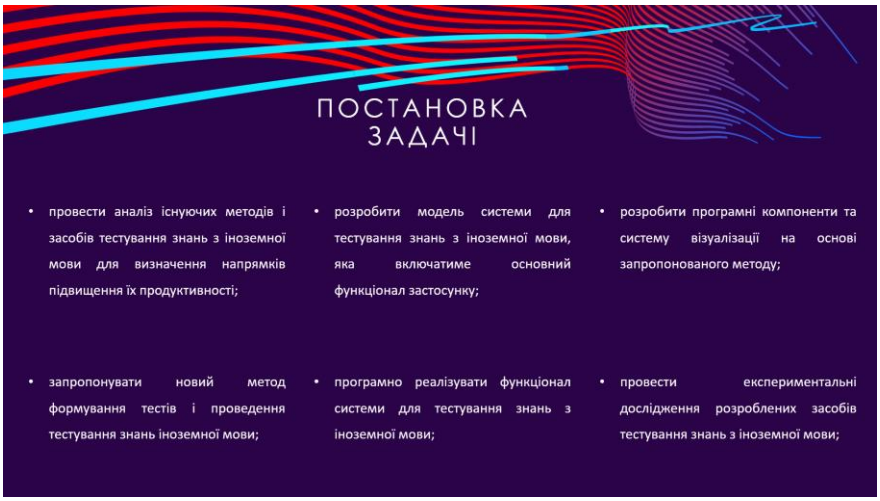

Рисунок Г.4 – Слайд «Постановка задачі»

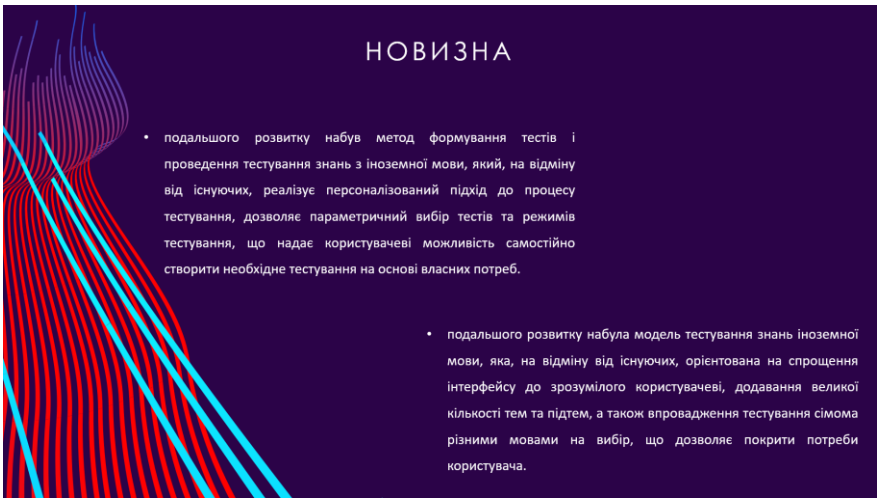

Рисунок Г.5 – Слайд «Новизна»

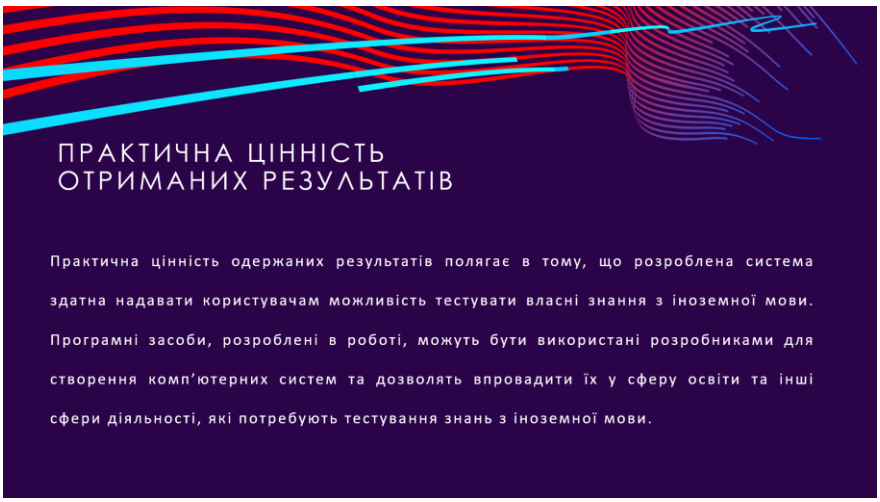

Рисунок Г.6 – Слайд «Практична цінність отриманих результатів»

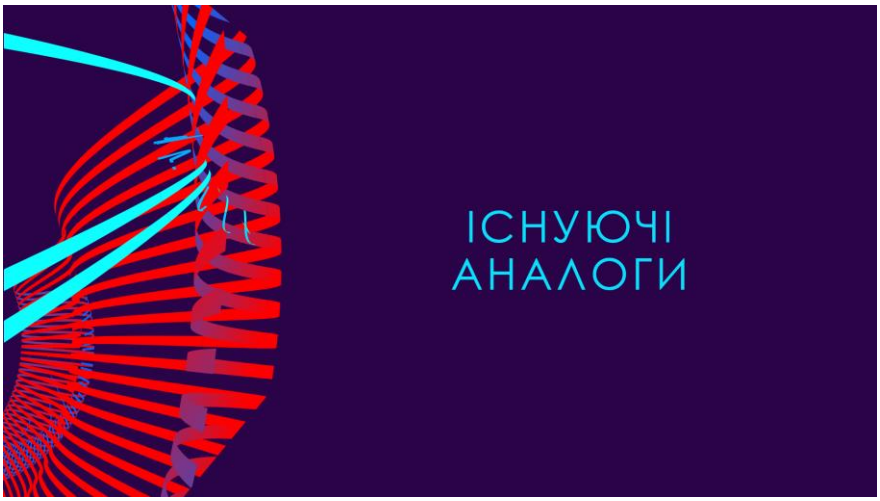

Рисунок Г.7 – Слайд «Існуючі аналоги»

Рисунок Г.8 – Слайд «Аналіз аналогів»

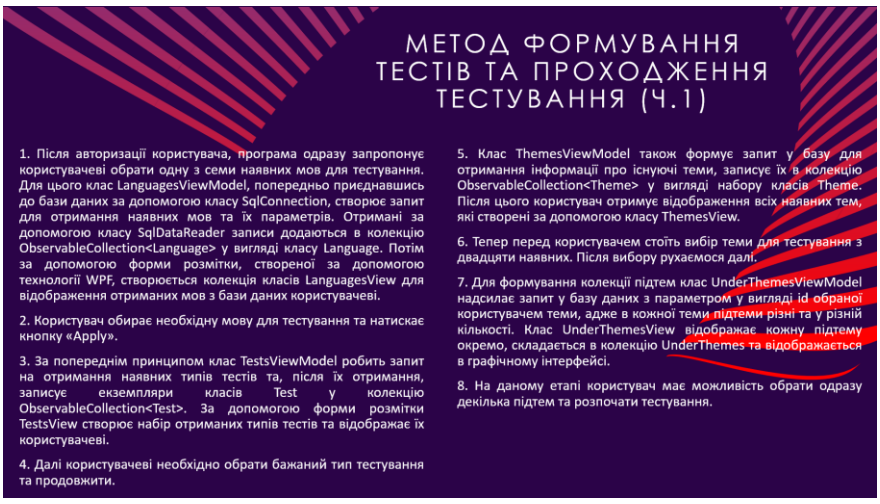

Рисунок Г.9 – Слайд «Метод формування тестів та проходження тестування (ч.1)»

Рисунок Г.10 – Слайд «Метод формування тестів та проходження тестування (ч.2)»

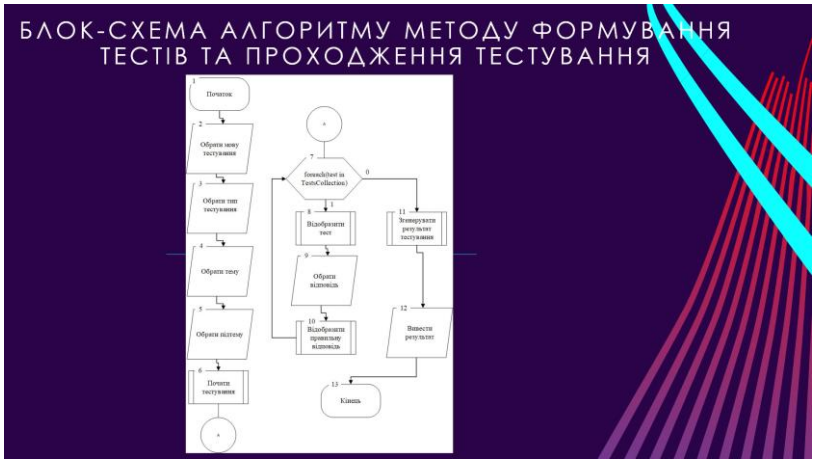

Рисунок Г.11 – Слайд «Блок-схема алгоритму Методу формування тестів та проходження тестування»

Рисунок Г.12 – Слайд «Модель системи на прикладі діаграми діяльності»

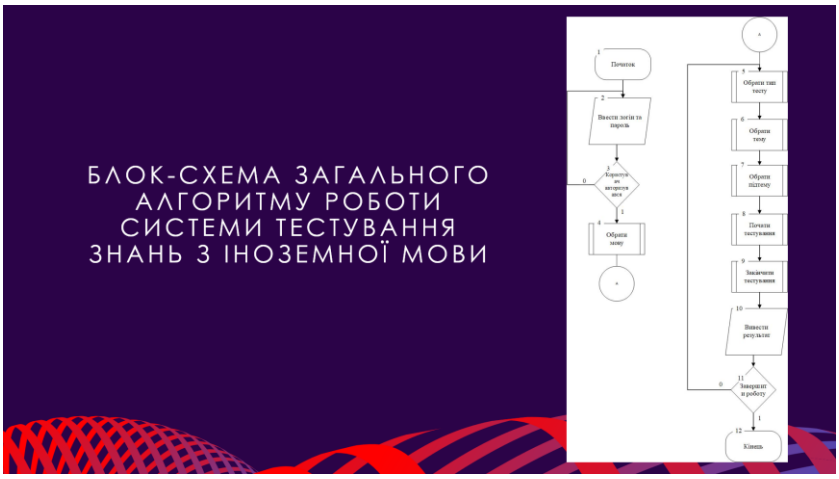

Рисунок Г.13 – Слайд «Блок-схема загального алгоритму роботи системи тестування знань з іноземної мови»

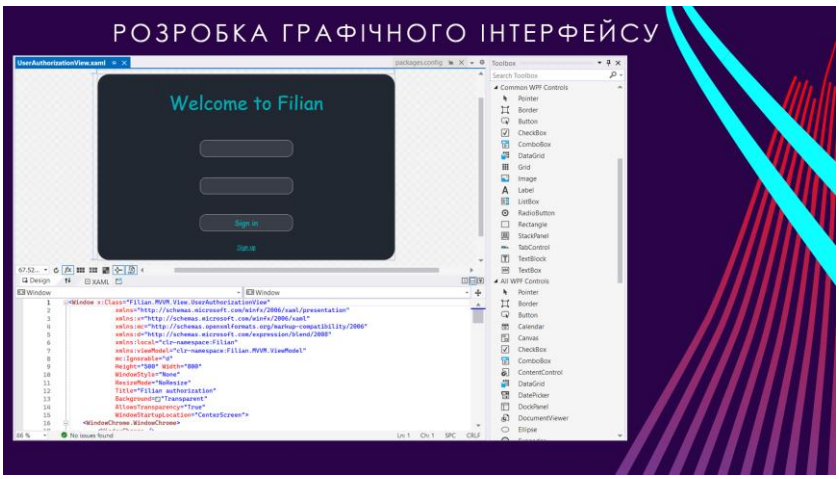

Рисунок Г.14 – Слайд «Розробка графічного інтерфейсу»

Рисунок Г.15 – Слайд «Використані технології»

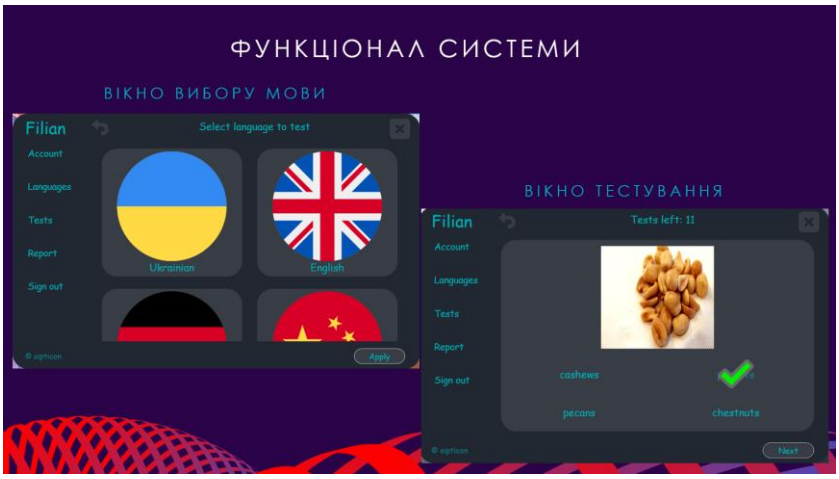

Рисунок Г.16 – Слайд «Функціонал системи»

Рисунок Г.17 – Слайд «Демонстрація роботи»

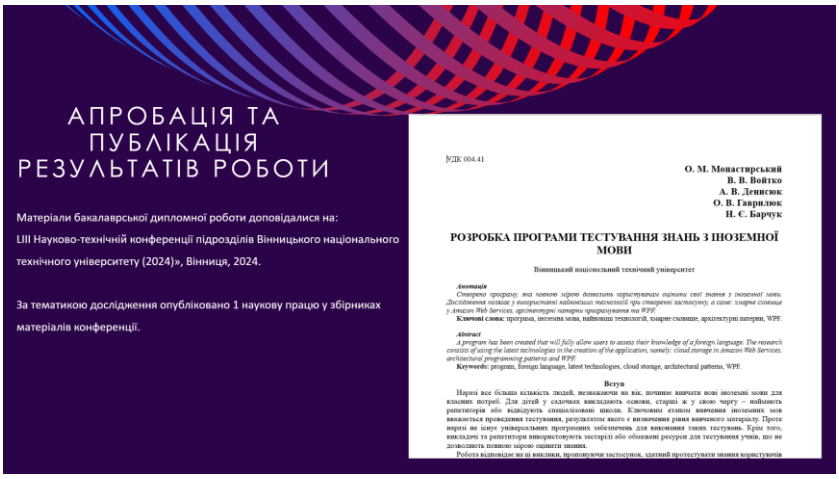

Рисунок Г.18 – Слайд «Апробація та публікація результатів роботи»

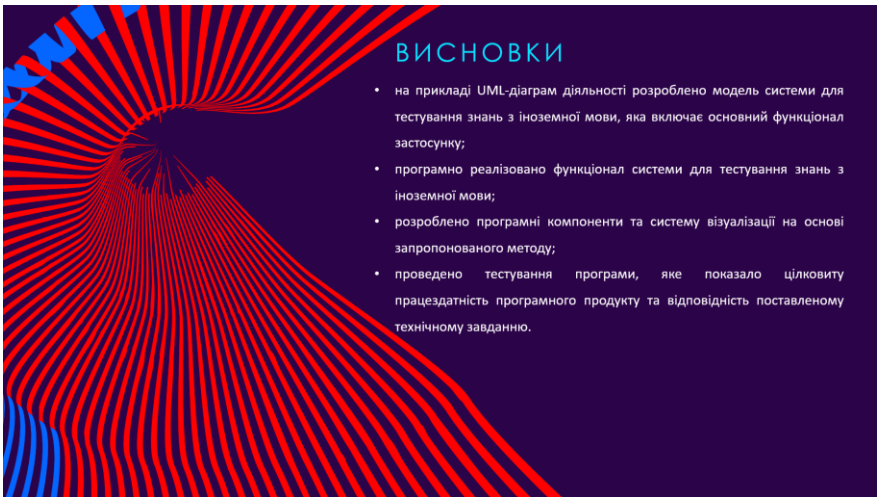

Рисунок Г.19 – Слайд «Висновки»

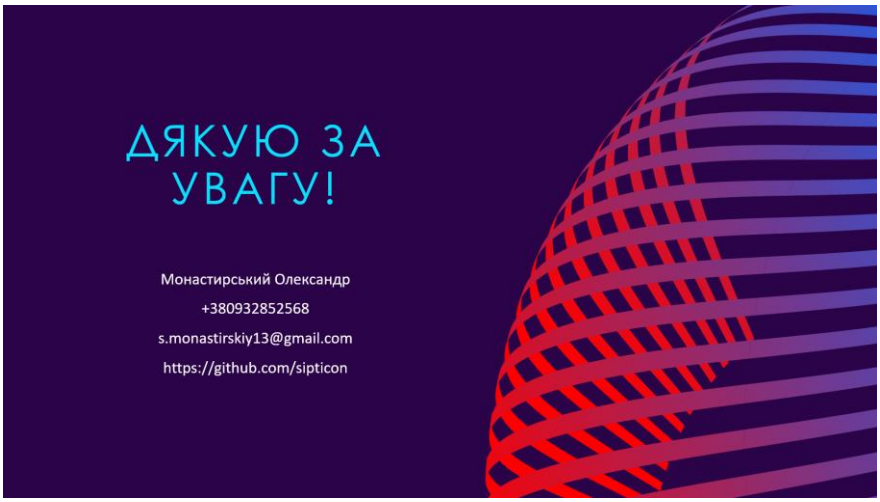

Рисунок Г.20 – Фінальний слайд