МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
УМАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет економіки і підприємництва

Кафедра інформаційних технологій

Допущено до захисту
"_____" _____ 2025 р.
Зав. кафедри, к.т.н., доцент
_____ **Ліщук Р.І.**

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

РОЗРОБКА ДОДАТКУ ДЛЯ ТЕСТУВАННЯ ЗНАНЬ З ІНОЗЕМНОЇ МОВИ

Виконав – студент 2 курсу, групи 22 к-кн
освітньої програми "Комп’ютерні науки"
спеціальності 122 Комп’ютерні науки
Бублієнко Денис Сергійович

Науковий керівник – к.е.н., доцент
Концеба Сергій Михайлович

Умань – 2025

ЗМІСТ

ВСТУП

РОЗДІЛ 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

1.2. Порівняльний аналіз аналогів

1.3. Постановка задач для розробки системи тестування знань іноземної мови

РОЗДІЛ 2 ПРОЕКТУВАННЯ МЕТОДУ, МОДЕЛІ ТА АЛГОРИТМІВ ПРОГРАМНОГО ПРОДУКТУ

2.1. Аналіз вхідних даних системи

2.2. Проектування моделі системи

2.3. Проектування методу формування тестів та проходження тестування

2.4. Проектування загального алгоритму роботи системи

2.5. Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного забезпечення

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

3.1. Розробка засобу керування обліковими записами

3.2. Розробка засобу отримання даних з хмарного середовища

3.3. Розробка графічного інтерфейсу

3.4. Тестування системи

ВИСНОВКИ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ДОДАТКИ

ВСТУП

Актуальність дослідження. У сучасному світі володіння іноземною мовою є важливою складовою професійного та особистого розвитку людини. Стрімке зростання обсягів інформації, міжнародна інтеграція, розширення міжнародних зв'язків у різних сферах діяльності обумовлюють необхідність знання принаймні однієї іноземної мови. Особливої актуальності набувають засоби автоматизації контролю та оцінювання знань, які дозволяють підвищити ефективність освітнього процесу, забезпечити об'єктивність перевірки та зручність для користувача.

Тестування є одним із найпоширеніших способів перевірки рівня знань, а створення програмних продуктів для автоматизованого тестування дозволяє оптимізувати цей процес як для викладачів, так і для учнів. Розробка сучасного програмного забезпечення для тестування знань з іноземної мови з використанням новітніх технологій, таких як Windows Presentation Foundation (WPF) та хмарних баз даних Amazon Relational Database Service (AWS RDS), є актуальним завданням сьогодення.

Метою роботи є розробка програмного додатку для тестування знань з іноземної мови, що забезпечує зручний та ефективний інтерфейс для користувача, а також можливість централізованого збереження результатів тестування у хмарній базі даних AWS RDS.

Для досягнення поставленої мети у кваліфікаційній роботі вирішуються такі завдання:

- провести аналіз існуючих засобів тестування знань з іноземної мови;
- розробити архітектуру програмного забезпечення із використанням шаблону проєктування Model-View-ViewModel (MVVM);
- створити користувацький інтерфейс із застосуванням технології WPF;
- спроєктувати структуру бази даних для збереження інформації про користувачів, тести та результати проходження тестів;
- реалізувати інтеграцію додатку з хмарною базою даних AWS RDS;

- забезпечити можливість реєстрації, авторизації користувачів та проходження тестів;
- провести тестування розробленого програмного продукту та оцінити його ефективність.

Об’єктом дослідження є процес контролю та оцінювання рівня знань користувачів з іноземної мови.

Предметом дослідження є методи, засоби та технології розробки програмних додатків для тестування знань, а також програмна реалізація інтерфейсу, логіки взаємодії та системи збереження даних користувачів.

У роботі застосовувалися такі **методи дослідження**:

- аналіз літературних джерел та програмних продуктів у сфері тестування знань;
- системний аналіз для визначення вимог до розроблюваного програмного забезпечення;
- методи об’єктно-орієнтованого проектування для створення архітектури системи;
- програмування мовою C# із застосуванням патерну MVVM для реалізації функціоналу додатку;
- тестування програмного забезпечення для перевірки його працездатності та відповідності вимогам.

РОЗДІЛ 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

На сучасному етапі розвитку освіти особливої актуальності набуває питання якісної оцінки рівня знань здобувачів освіти з іноземної мови. Це зумовлено як підвищенням вимог до комунікативної компетентності студентів, так і потребою в об’єктивному та достовірному контролі рівня володіння мовними навичками. Традиційні методи контролю поступово поступаються місцем сучасним інформаційно-комунікаційним технологіям, які дозволяють зробити процес тестування більш ефективним, доступним, а також зменшити вплив людського фактора.

На сьогодні існує велика кількість засобів та систем, що застосовуються для тестування знань з іноземної мови. Умовно їх можна поділити на дві основні групи: традиційні (паперові) методи та комп’ютеризовані (електронні) системи тестування.

До традиційних засобів тестування належать письмові контрольні роботи, диктанти, усні опитування, тестові завдання на паперових носіях. Основними перевагами таких методів є простота застосування, доступність для будь-якого закладу освіти, можливість гнучкої адаптації завдань під рівень учнів. Однак традиційні методи мають і суттєві недоліки: перевірка результатів часто є трудомісткою і суб’єктивною, неможливо оперативно проаналізувати типові помилки великої кількості учасників тестування, обмежене охоплення різних аспектів мовної компетенції (наприклад, складно перевірити навички аудіювання чи говоріння).

Значного поширення останнім часом набули програмні засоби комп’ютерного тестування, які дозволяють здійснювати автоматизовану перевірку знань студентів. До таких засобів належать як спеціалізовані

програмні продукти, що встановлюються локально на комп’ютерах, так і онлайн-системи, доступні через інтернет.

Однією з найпопулярніших систем є Moodle — відкрита платформа для організації дистанційного навчання та тестування. Вона забезпечує можливість створення різноманітних типів завдань: закритих, відкритих, завдань на відповідність, доповнення речень тощо. Перевагами цієї системи є гнучкість, можливість масштабування, автоматична перевірка результатів, ведення статистики успішності. Проте для ефективного використання Moodle необхідні певні технічні навички як викладача, так і студентів.

Широко використовуються також безкоштовні сервіси Google Forms, які дозволяють створювати онлайн-тести з миттєвою перевіркою закритих завдань. До переваг цього засобу належать простота у використанні, доступність, можливість оперативного збору результатів у хмарному сховищі. Водночас Google Forms має обмежену функціональність для створення складних тестів, зокрема таких, що передбачають перевірку навичок говоріння або письма.

Серед інноваційних рішень слід відзначити платформи типу Quizlet і Kahoot!, які дозволяють організувати інтерактивне, ігрове тестування. Вони спрямовані переважно на розвиток лексичних навичок та мотивацію студентів до вивчення іноземної мови. Однак такі інструменти більше підходять для повторення матеріалу, ніж для серйозного підсумкового контролю знань.

Особливу групу засобів тестування становлять міжнародні стандартизовані тести — TOEFL, IELTS, Cambridge Exams, які оцінюють знання мови за міжнародною шкалою CEFR. Їх характерною рисою є всебічне тестування всіх видів мовної діяльності: читання, аудіювання, письма та говоріння. Дані системи забезпечують високий рівень об'єктивності та достовірності результатів, однак вони потребують значних фінансових і часових витрат, а також спеціальної підготовки як з боку викладача, так і студента.

Аналізуючи сучасний стан розвитку систем тестування знань з іноземної мови, можна зробити висновок, що вони активно розвиваються у напрямку автоматизації процесів перевірки, розширення функціоналу, інтеграції

мультимедійних засобів. Водночас існують певні проблеми, які потребують вирішення: забезпечення перевірки продуктивних видів мовленнєвої діяльності (усного мовлення та письма), розробка адаптивних тестів, що враховують індивідуальні особливості учня, гарантування безпеки даних користувачів.

Отже, сучасні засоби тестування знань з іноземної мови є різноманітними за функціоналом, призначенням та рівнем складності. Вибір конкретного інструменту має базуватися на цілях тестування, технічних можливостях закладу освіти, рівні підготовки викладачів і студентів, а також на потребі в автоматизації процесу та об'єктивності результатів.

1.2 Порівняльний аналіз аналогів

У зв’язку з активним впровадженням цифрових технологій в освітній процес значно зросла популярність програмних засобів, що дозволяють здійснювати тестування та оцінювання рівня знань іноземних мов, зокрема англійської. Сучасні програмні продукти відрізняються функціональним наповненням, дидактичними можливостями, інтерактивністю та спрямованістю на розвиток різних мовленнєвих компетентностей.

До найбільш відомих рішень відносять:

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

Busuu (рис. 1.1) вирізняється досить широкими можливостями для розвитку всіх видів мовленнєвої діяльності, зокрема письма та говоріння завдяки функції надсилання відповідей для перевірки носіям мови. Також Busuu дозволяє проходити тематичні тести після кожного модуля, що є корисним для формувального оцінювання. Однак платформа більше орієнтована на початковий та середній рівні (A1–B2) і не забезпечує повноцінного іспитового тестування для рівнів C1–C2. [7].

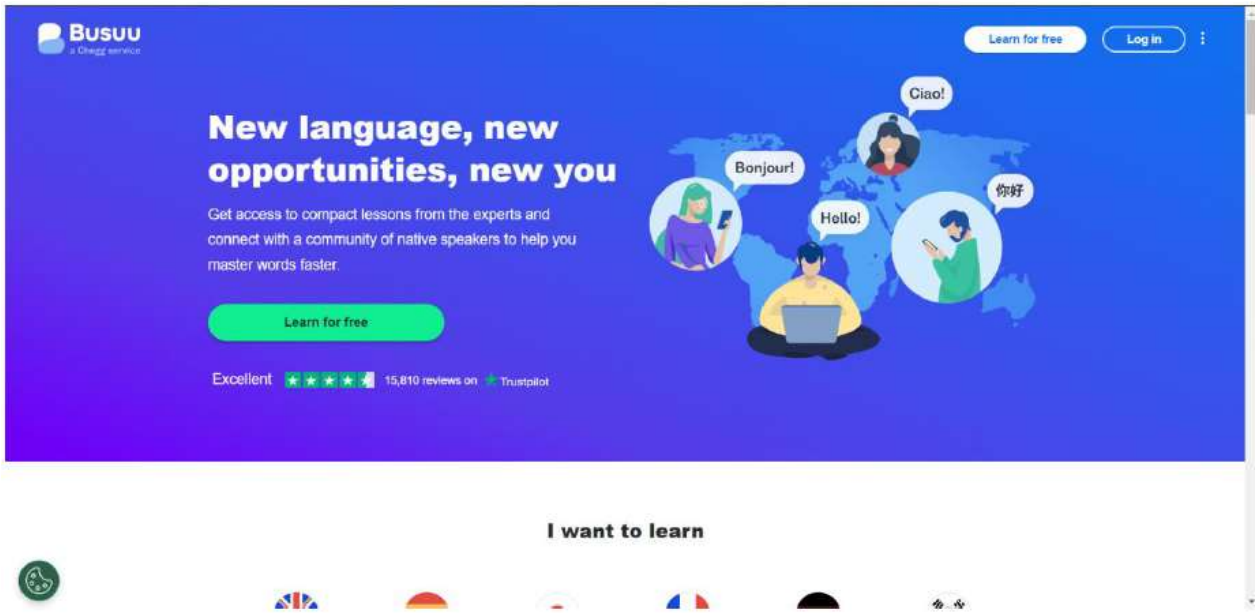


Рис. 1.1. Головна сторінка сайту платформи Busuu

Drops (рис. 1.2) демонструє високу ефективність у запам’ятовуванні нових слів, завдяки коротким інтерактивним ігровим сесіям, що розвивають візуальну пам’ять. Проте цей додаток повністю ігнорує такі критично важливі навички, як граматика, побудова речень, сприйняття на слух та говоріння. Відсутність тестування робить його лише допоміжним засобом для розширення лексики, а не повноцінним інструментом для оцінювання мовних знань [9].

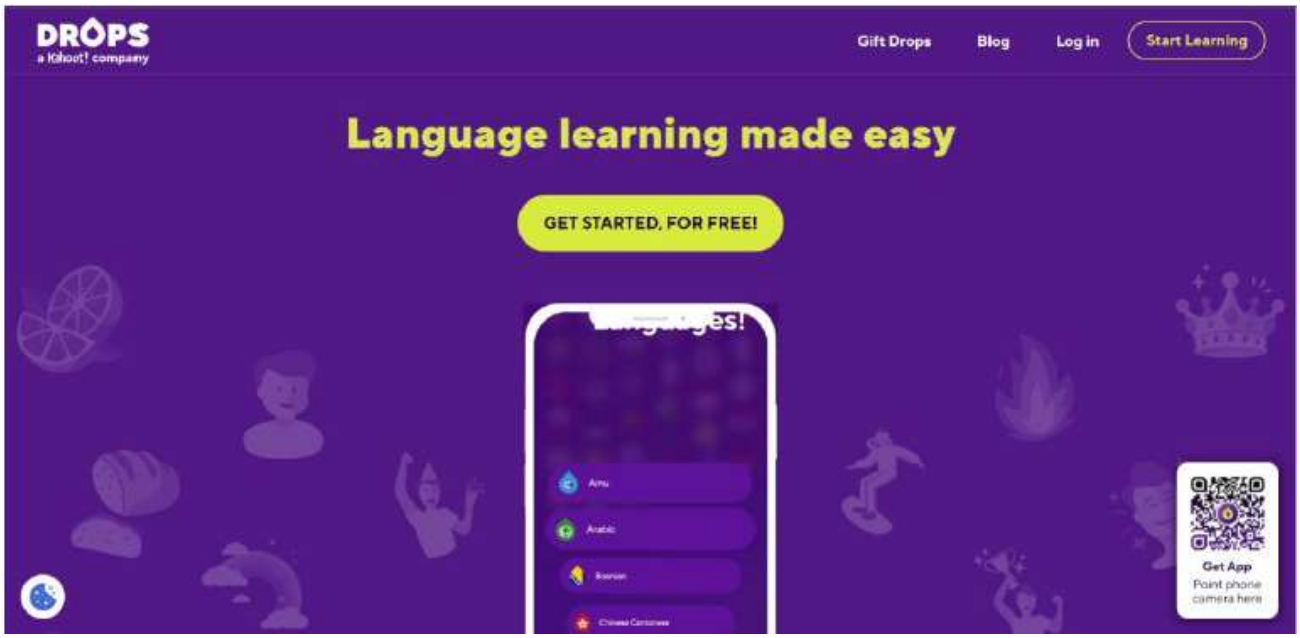


Рис. 1.2. Головна сторінка сайту платформи Drops

LingQ (рис. 1.3) має перевагу для студентів із середнім та високим рівнем володіння англійською мовою, адже платформа дозволяє працювати з реальними автентичними текстами та аудіо, формуючи "живий" словниковий запас і навички розуміння контексту. Разом із тим, система не має стандартних тестових завдань або контрольних модулів, що ускладнює використання платформи саме як інструменту для оцінювання знань. [10].

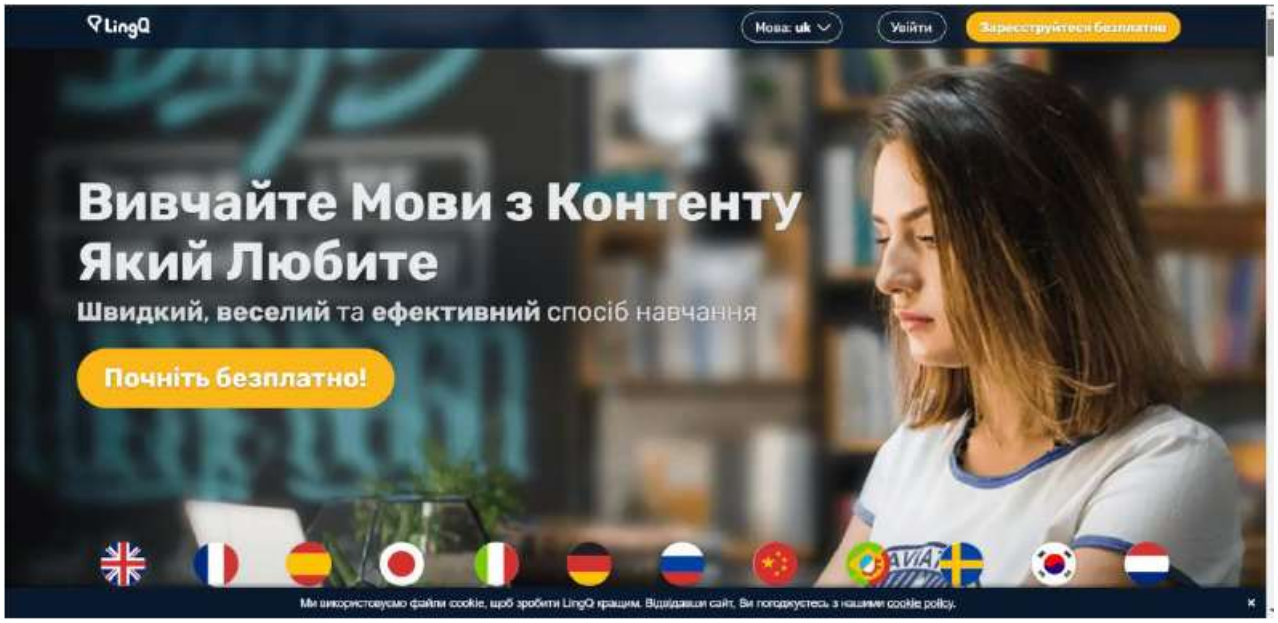


Рис.1.3. Головна сторінка сайту платформи LingQ

Preply (рис. 1.4) істотно відрізняється від інших платформ тим, що базується на індивідуальному спілкуванні з викладачами. Це забезпечує можливість персоналізованого тестування, включаючи перевірку письма, аудіювання, говоріння, граматики залежно від потреб конкретного студента. Крім того, саме ця платформа може бути корисною для підготовки до офіційних мовних іспитів (TOEFL, IELTS) завдяки професійним викладачам. Недоліком Preply є висока вартість повноцінних курсів у порівнянні з іншими додатками. [11].

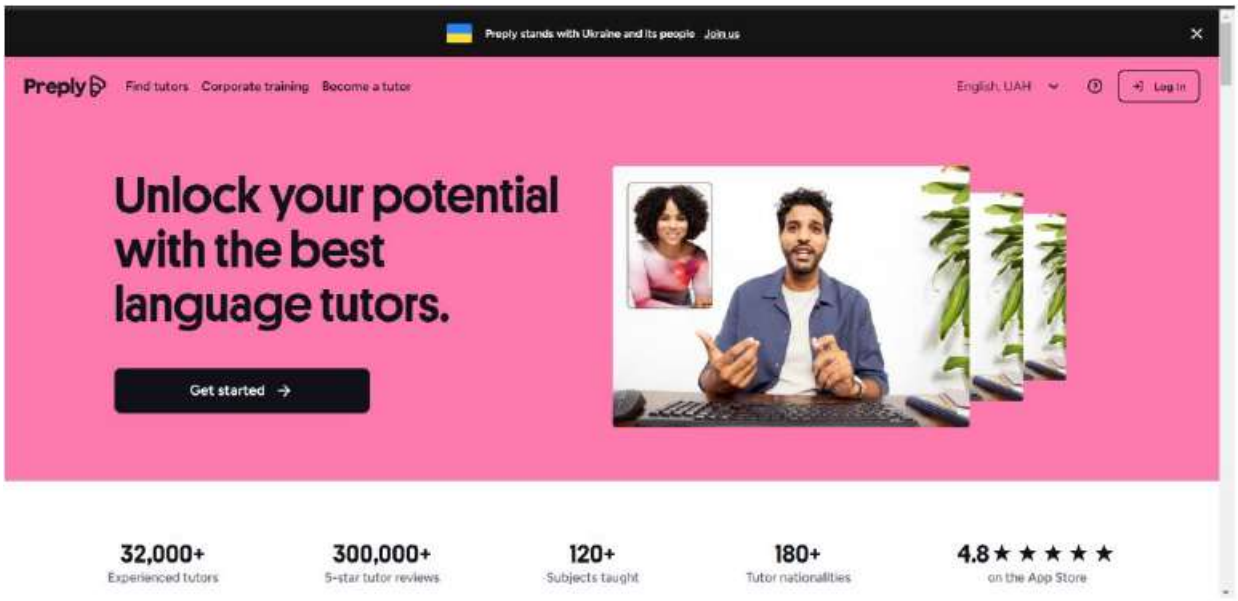


Рис. 1.4. Головна сторінка сайту платформи Preply

Duolingo демонструє збалансовану стратегію навчання на рівнях A1–B2, використовуючи гейміфікацію для підвищення мотивації користувача. Додаток дозволяє відпрацьовувати лексику, граматику, читання та аудіювання. Хоча функція говоріння присутня, її ефективність обмежена через автоматичну перевірку системи. Недоліком є також недостатній акцент на розвиток навичок письма, а результати тестів не можуть бути офіційним підтвердженням рівня володіння мовою (за винятком окремого продукту — Duolingo English Test). [13].

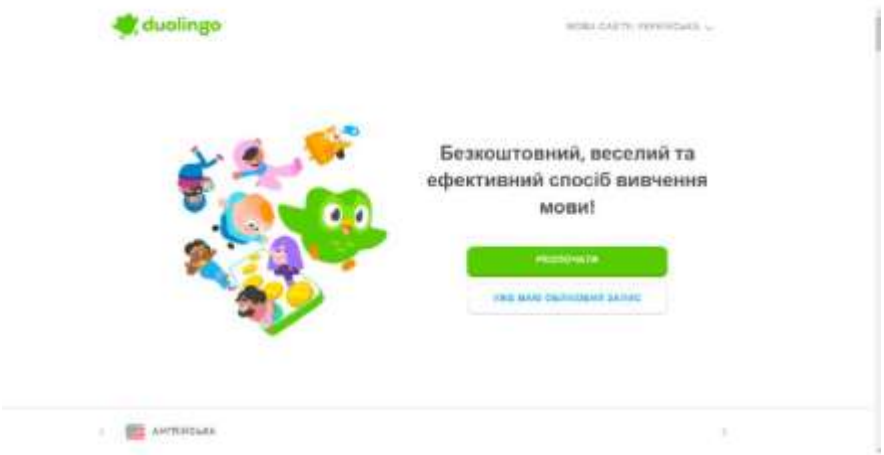


Рис. 1.5. Головна сторінка сайту платформи Duolingo

Для узагальнення проведеного аналізу представлено порівняльну таблицю основних характеристик розглянутих систем (табл. 1). Результати порівняння аналогів зведено в табл. 1.1.

1.1. Порівняльні характеристика онлайн платформ для вивчення іноземних мов

Критерій	Busuu	Drops	LingQ	Preply	Duolingo	Критерій
Комплексність покриття мовних навичок	Висока	Низька	Середня	Висока	Середня	Комплексність покриття мовних навичок
Оцінка аудіювання	Так	Ні	Так	Так	Так	Оцінка аудіювання
Оцінка письма	Так	Ні	Частково	Так	Обмежено	Оцінка письма
Оцінка говоріння	Так	Ні	Ні	Так	Частково	Оцінка говоріння
Оцінка читання	Так	Ні	Так	Так	Так	Оцінка читання
Наявність тестів/контролю знань	Так	Ні	Ні	Так	Так	Наявність тестів/контролю знань
Можливість отримання сертифікату	Так	Ні	Ні	Можливо	Ні	Можливість отримання сертифікату
Персоналізація навчання	Середня	Низька	Висока	Висока	Середня	Персоналізація навчання
Вартість	Безкоштовно + преміум	Безкоштовно + преміум	Безкоштовно + преміум	Платно	Безкоштовно + преміум	Вартість

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.

Таким чином, жоден із розглянутих сервісів самостійно не забезпечує повного циклу якісного тестування всіх мовних компетентностей відповідно до

міжнародних стандартів (CEFR). Для повноцінної оцінки знань користувачу доцільно комбінувати кілька платформ:

- Busuu та Duolingo — для регулярного тренування лексико-граматичних знань;
- LingQ — для покращення навичок читання та аудіювання на автентичних матеріалах;
- Preply — для розвитку продуктивних навичок (говоріння, письма) та отримання професійної оцінки викладача;
- Drops — як допоміжний інструмент для активного запам’ятовування нових слів.

Перспективним напрямом розвитку програмних засобів тестування є інтеграція штучного інтелекту для повноцінної автоматизованої перевірки усного та письмового мовлення, адаптивні системи формувального тестування, а також створення універсальних платформ, що поєднують можливості всіх зазначених сервісів в єдиній системі.

1.3 Постановка задач для розробки системи тестування знань іноземної мови

В умовах сучасного освітнього процесу особливу актуальність набувають інформаційні системи, що забезпечують автоматизовану перевірку знань іноземної мови. Використання таких систем дозволяє значно підвищити ефективність навчання, здійснювати об’єктивний контроль рівня засвоєння навчального матеріалу, а також забезпечити адаптивність і персоналізацію навчальних курсів відповідно до потреб кожного студента.

Основною метою розробки системи тестування знань іноземної мови є створення програмного засобу, який дозволяє здійснювати комплексну діагностику мовних компетентностей користувача за такими видами мовленнєвої діяльності, як читання, аудіювання, письмо та говоріння.

Відповідно до поставленої мети необхідно вирішити наступні задачі:

1. Аналіз існуючих засобів тестування знань іноземної мови (Busuu, Drops, LingQ, Preply, Duolingo) для виявлення їх переваг та недоліків з метою врахування під час проєктування нової системи.

2. Визначення функціональних вимог до системи, а саме:

- можливість створення тестів різного типу (вибір однієї/декількох відповідей, встановлення відповідності, заповнення пропусків, введення власного варіанту відповіді);
- автоматизована перевірка результатів тестування та надання користувачеві зворотного зв’язку;
- підтримка різних рівнів складності тестів відповідно до рівнів CEFR (A1–C2);
- збереження результатів тестування у базі даних для подальшого аналізу та моніторингу прогресу;
- можливість формування індивідуальних завдань залежно від рівня підготовки студента.

3. Розробка загальної архітектури системи, яка передбачає:

- клієнтську частину (інтерфейс користувача для проходження тестів);
- серверну частину (обробка тестів, збереження даних, формування звітів);
- базу даних для зберігання тестів, відповідей та результатів користувачів.

4. Створення зручного та інтуїтивно зрозумілого інтерфейсу користувача, що забезпечує легкий доступ до тестів, можливість реєстрації/авторизації, перегляду власних результатів, рекомендацій для подальшого навчання.

5. Забезпечення можливості багатомовної підтримки системи, включаючи англійську мову як основний об’єкт тестування.

6. Тестування працездатності програмного продукту за допомогою реальних користувачів для оцінки його ефективності, зручності використання та відповідності поставленим цілям.

Таким чином, результатом виконання даного проєкту має стати розробка веборієнтованої системи тестування знань з англійської мови, яка забезпечує інтерактивний контроль знань, зберігає результати проходження тестів та надає користувачеві рекомендації для подальшого вдосконалення мовних навичок.

РОЗДІЛ 2 ПРОЕКТУВАННЯ МЕТОДУ, МОДЕЛІ ТА АЛГОРИТМІВ ПРОГРАМНОГО ПРОДУКТУ

2.1 Аналіз вхідних даних системи

У сучасній розробці програмного забезпечення важливим є не лише вибір методологій, а й обґрунтований добір технологічних інструментів, які забезпечують надійність, масштабованість, гнучкість і продуктивність створюваного програмного продукту. У межах даного дослідження для розробки системи тестування знань з іноземної мови були обрані мова програмування C#, засоби побудови інтерфейсу користувача WPF, хмарне сховище AWS RDS для зберігання даних та інтегроване середовище розробки Microsoft Visual Studio. Розглянемо кожен із цих компонентів детальніше.

Основною мовою програмування для реалізації функціональних можливостей системи є C#. Це сучасна об’єктно-орієнтована мова, яка входить до складу екосистеми .NET від компанії Microsoft. Її перевагою є висока читабельність коду, розвинуті можливості для побудови як клієнтських (настільних), так і серверних додатків, підтримка асинхронного програмування, а також велика кількість вбудованих бібліотек та фреймворків, що дозволяють вирішувати різноманітні завдання розробки.

C# дозволяє ефективно створювати бізнес-логіку програми: обробку відповідей користувачів, проведення перевірки коректності введених даних, роботу з базою даних, формування статистичних звітів тощо. Завдяки широким можливостям роботи з мережею C# також забезпечує реалізацію функцій взаємодії з хмарним сервером та виконання запитів до віддаленої бази даних.

Однією з сильних сторін цієї мови є використання платформи .NET Core / .NET 6+, що дає змогу розробляти кросплатформні рішення з підтримкою Windows, Linux та macOS, хоча для створюваної системи передбачається реалізація саме для операційної системи Windows [14].

Графічний інтерфейс користувача у системі тестування реалізується за допомогою Windows Presentation Foundation (WPF) — сучасної платформи для створення настільних програм під операційні системи родини Windows.

Важливою перевагою WPF є використання мови розмітки XAML (Extensible Application Markup Language), що дозволяє розробнику відокремити візуальну складову програми від її логіки. Завдяки цьому розділенню можлива реалізація шаблонів MVVM (Model-View-ViewModel), які значно спрощують структуру коду, підвищують його гнучкість та зручність підтримки.

WPF забезпечує створення інтуїтивно зрозумілих, адаптивних та візуально привабливих інтерфейсів із застосуванням стилів, анімацій, графічних ефектів, які можна використовувати для створення елементів тестових завдань, повідомлень про результати тестування, підказок або додаткових інструкцій. До того ж платформа підтримує інтеграцію мультимедійних об’єктів, що є особливо актуальним для перевірки навичок аудіювання або роботи з відеоматеріалами.

Крім того, WPF має розвинуті можливості роботи з об’єктами управління (кнопками, полями введення, списками), що дозволяє створювати динамічні, змінні за контентом форми — необхідну складову для адаптивних тестових завдань [15].

Для зберігання навчальних даних, інформації про користувачів, результатів тестів та історії проходження тестів передбачається використання хмарного сховища AWS RDS (Amazon Web Services Relational Database Service). Це керована служба реляційних баз даних, що дозволяє розробникам створювати, налаштовувати та масштабувати хмарні бази даних без необхідності керування фізичним серверним обладнанням.

Переваги використання AWS RDS у контексті даної системи тестування полягають у забезпеченні високої доступності даних, автоматизованого резервного копіювання, стійкості до збоїв та можливості горизонтального масштабування в разі зростання кількості користувачів системи. Крім того, AWS RDS підтримує популярні системи управління базами даних, такі як MySQL, PostgreSQL, SQL Server та інші, що дає можливість обрати оптимальну технологію для конкретного завдання.

Застосування хмарного сховища робить систему доступною з будь-якої точки світу за умови наявності доступу до мережі Інтернет, що особливо важливо для дистанційного тестування студентів або співробітників компаній[16].

Усі етапи розробки програмного забезпечення виконуються у середовищі Microsoft Visual Studio — потужному інтегрованому середовищі розробки (IDE), яке забезпечує зручний і функціональний інтерфейс для написання, налагодження, тестування та розгортання програм. Visual Studio підтримує роботу з мовою C#, має розвинуті інструменти для створення WPF-застосунків, а також дозволяє легко підключатися до хмарних сервісів AWS.

Середовище має такі переваги, як інтеграція з системами контролю версій (наприклад, Git), автоматичне виявлення синтаксичних і логічних помилок, можливість використання розширень та сторонніх бібліотек, зручну систему побудови графічного інтерфейсу, профілювання продуктивності програми та багато інших корисних функцій.

Завдяки використанню Visual Studio розробник має змогу значно скоротити час створення програмного продукту, знизити ймовірність помилок, забезпечити високу якість і підтримуваність програмного коду [17].

Отже, поєднання мови програмування C#, конструктора інтерфейсів WPF, хмарного сховища AWS RDS та середовища розробки Microsoft Visual Studio дозволяє створити сучасну, масштабовану, зручну в користуванні та ефективну систему тестування знань з іноземної мови. Вибір зазначених технологій забезпечує як високу якість реалізації програмного продукту, так і його відповідність сучасним вимогам до програмних систем у сфері освіти та тестування.

2.2 Проектування 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).

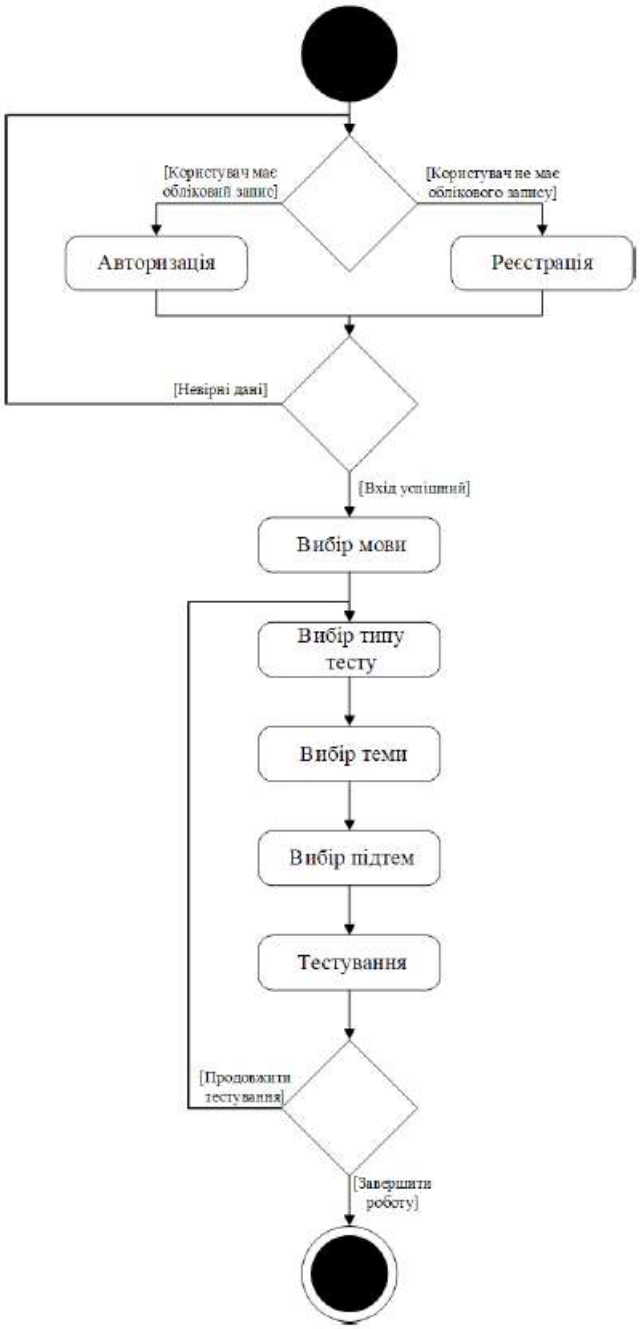


Рис.2.1. Модель системи у вигляді діаграми діяльності

На рисунку наведено UML-діаграму діяльності, яка відображає послідовність дій користувача під час роботи із системою тестування знань з іноземної мови. Дана діаграма моделює типовий сценарій використання програмного забезпечення та демонструє логічний порядок виконання основних функцій системи.

Робота користувача починається з початкового стану, який позначений заповненим чорним колом. Після цього система перевіряє, чи має користувач

обліковий запис у системі. Якщо обліковий запис уже існує, користувач переходить до етапу авторизації. У випадку відсутності облікового запису система пропонує пройти реєстрацію для створення нового профілю.

Після заповнення реєстраційної або авторизаційної форми система виконує перевірку достовірності введених даних. У разі виявлення помилок користувач отримує повідомлення про некоректні дані та повертається до етапу авторизації або реєстрації. Якщо ж перевірка пройшла успішно, користувач допускається до основного функціоналу системи.

Першим кроком після входу є вибір мови тестування. Наприклад, користувач може обрати англійську мову як основний предмет для перевірки знань. Далі пропонується визначити тип тесту, що дозволяє налаштувати формат завдань (наприклад, тести на граматику, лексику, аудіювання тощо). Наступним етапом є вибір загальної теми тесту, наприклад: "Подорожі", "Бізнес", "Освіта". Після цього користувач уточнює підтему для конкретизації напряму тестування (наприклад, "Подорож літаком", "Співбесіда англійською" тощо).

Після завершення налаштування параметрів розпочинається основний етап — тестування. У цей період користувач послідовно виконує запропоновані завдання відповідно до обраних тем та типу тесту.

Після завершення тестування система пропонує користувачу два варіанти подальших дій: або продовжити тестування, повернувшись до етапу вибору параметрів нового тесту, або завершити роботу в системі. У випадку завершення роботи процес переходить до кінцевого стану, позначеного заповненим подвійним чорним колом.

Таким чином, представлена діаграма діяльності повністю описує життєвий цикл взаємодії користувача із системою тестування знань з іноземної мови — від входу в систему до виходу з неї. Діаграма дозволяє наочно продемонструвати логіку роботи програмного продукту, забезпечуючи структуроване розуміння процесів, які відбуваються під час використання системи.

2.3 Проектування методу формування тестів та проходження тестування

У системі тестування знань з іноземної мови важливу роль відіграє процес формування тестів та їх проходження користувачем. Для наочного представлення цього процесу розроблено блок-схему, що зображає послідовність дій системи та користувача під час вибору параметрів тесту, виконання завдань та отримання результатів (Рис. 2.2).

Робота алгоритму розпочинається з початкового етапу, де активується процедура вибору основних параметрів майбутнього тесту. Першим кроком користувач має визначити мову, з якої буде здійснюватися тестування. Наприклад, для перевірки знань англійської мови обирається відповідний пункт зі списку доступних мов.

Наступним етапом є вибір типу тестування. Система передбачає різноманітні варіанти тестів, серед яких можуть бути завдання на перевірку знань граматики, лексики, аудіювання, читання або загальних мовних навичок. Після вибору типу тесту користувач переходить до визначення загальної теми, що дозволяє зосередитися на певній сфері знань. Наприклад, можливо обрати такі теми як "Подорожі", "Бізнес-англійська", "Щоденне спілкування" тощо.

Для більш точного налаштування тестування користувачеві пропонується вибрати підтему, яка деталізує зміст завдань. Так, для теми "Подорожі" підтема може стосуватися "Поїздки літаком", "Заселення в готель" або "Заповнення митної декларації".

Після налаштування усіх параметрів розпочинається основний етап роботи — тестування. У цей момент система формує колекцію завдань відповідно до обраних критеріїв. Далі реалізується цикл проходження тестів: перевіряється наявність ще не опрацьованих тестових завдань у колекції. Якщо завдання залишились, наступне питання відображається на екрані.

Користувач переглядає запитання та обирає відповідь із запропонованих варіантів. Після вибору система автоматично здійснює перевірку правильності

відповіді, порівнюючи її із заздалегідь визначеним еталоном. Результати кожного питання фіксуються для подальшого підрахунку.

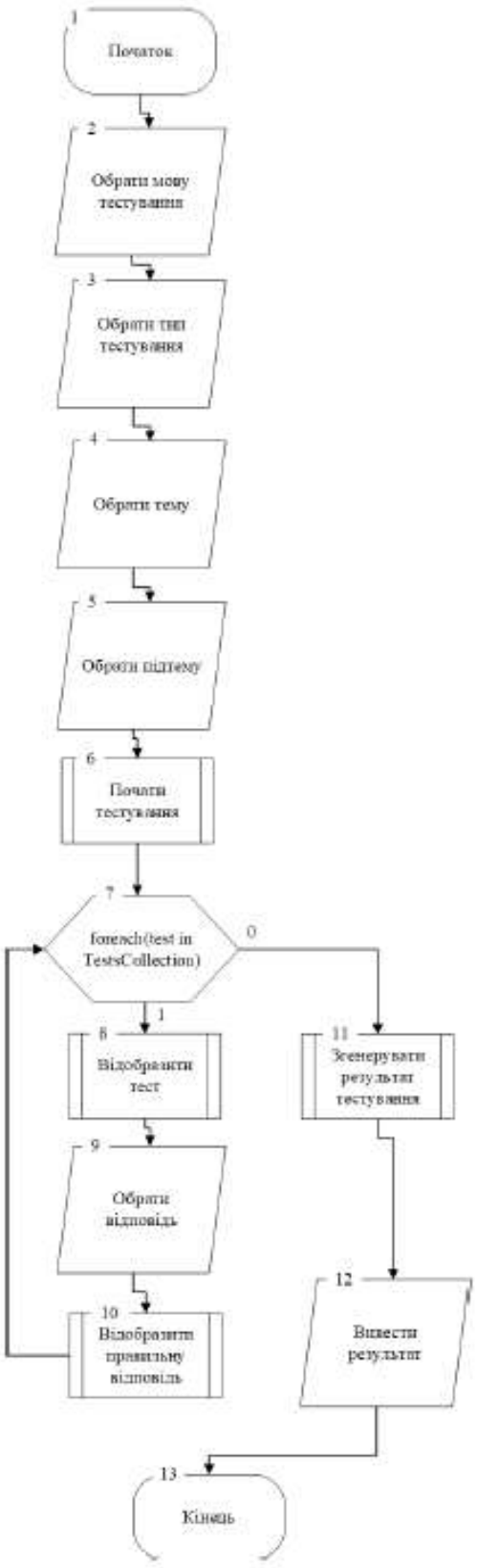


Рис. 2.2. Блок-схема алгоритму роботи методу формування тестів та проходження тестування

Процес повторюється до тих пір, поки всі завдання не буде пройдено. Після завершення тестування система переходить до етапу обробки отриманих результатів. Здійснюється обчислення загальної кількості правильних відповідей, розраховується підсумковий бал, відсоток правильності виконання тесту.

На завершальному етапі результати тестування виводяться на екран користувача у зручному вигляді: у вигляді повідомлення про загальний бал, графічної шкали успішності або короткої аналітичної довідки. Після цього алгоритм завершує свою роботу, пропонуючи користувачу завершити сесію або розпочати нове тестування з іншими параметрами.

Запропонована схема демонструє послідовний і структурований підхід до організації процесу тестування, що дозволяє забезпечити зручність роботи для користувача, автоматизовану перевірку результатів та гнучке налаштування параметрів тесту.

2.4 Проектування 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). Загальний алгоритм роботи системи тестування знань з іноземної мови складається з послідовності етапів, які забезпечують взаємодію користувача із системою — від входу до виведення результатів тестування. Опис алгоритму наведено нижче.

Робота системи розпочинається з ініціалізації (позначено як «Початок»), після чого користувачу пропонується ввести логін та пароль для авторизації в системі. На цьому етапі здійснюється перевірка введених даних на коректність. Якщо введені логін або пароль неправильні, система повертається до кроку авторизації та просить користувача повторити введення.

У разі успішної авторизації система переходить до етапу налаштування параметрів майбутнього тесту. Першим налаштуванням є вибір мови тестування — наприклад, англійської, німецької, французької тощо. Після цього користувач

повинен обрати тип тесту — це може бути тест на знання лексики, граматики, аудіювання або загальних мовних навичок.

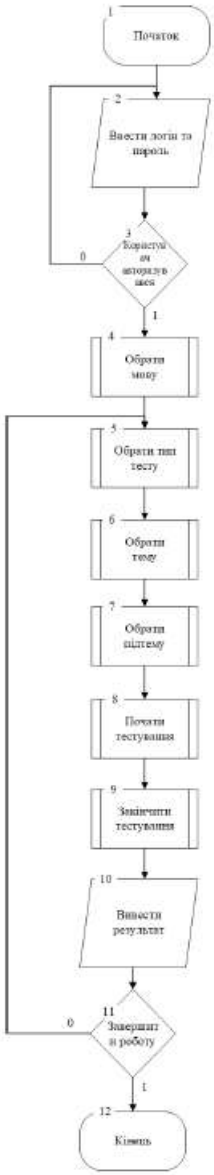


Рис. 2.3. Загальний алгоритм роботи системи

Наступним кроком є вибір загальної теми тесту — наприклад, "Подорожі", "Бізнес-англійська", "Щоденне спілкування" тощо. Для деталізації змісту тесту користувачу пропонується додатково обрати підтему. Це дозволяє звузити коло питань до певної конкретної ситуації — наприклад, "Аеропорт", "Готель", "Ресторан" та інше.

Після завершення налаштувань починається основний етап — тестування. Користувачу послідовно виводяться завдання, які потрібно виконати відповідно

до раніше обраних параметрів. Після завершення виконання всіх завдань тестування завершується.

Далі система переходить до етапу виведення результатів. Підсумкові дані обробляються: обраховується кількість правильних відповідей, розраховується відсоток успішності, визначається рівень знань користувача. Отриманий результат виводиться на екран у зрозумілому та зручному вигляді.

Після ознайомлення з результатами користувачу пропонується або завершити роботу, або розпочати новий цикл тестування з іншими параметрами. У випадку вибору повторного тестування система повертається до етапу вибору параметрів тесту. У разі завершення роботи виконується вихід із системи (позначено як «Кінець»).

Таким чином, наведений алгоритм забезпечує повний цикл роботи системи тестування — від ідентифікації користувача до отримання результатів тестування, що робить роботу системи логічною, зрозумілою та зручною для користувача.

2.5 Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного забезпечення

У процесі розробки системи тестування знань з іноземної мови було здійснено обґрунтований вибір відповідних засобів програмної реалізації, а саме: мови програмування C#, середовища побудови графічного інтерфейсу Windows Presentation Foundation (WPF), хмарного сховища даних Amazon Relational Database Service (AWS RDS) та архітектурного шаблону Model-View-ViewModel (MVVM). Кожне з цих рішень має конкретні переваги, що роблять їх доцільними для створення ефективного, зручного та масштабованого програмного продукту.

Для створення прикладної частини системи обрано мову програмування C#, яка є сучасною об’єктно-орієнтованою мовою та входить до складу платформи .NET. Серед її основних переваг варто зазначити:

- повну інтеграцію з технологіями .NET, що дозволяє використовувати готові бібліотеки для роботи з базами даних, обробки подій, створення графічного інтерфейсу;
- високу продуктивність виконання коду;
- вбудовані засоби автоматичного управління пам’яттю (garbage collection);
- підтримку паралельних обчислень, що дозволяє реалізувати ефективну багатопоточну обробку запитів користувача;
- активну спільноту розробників, доступність великої кількості довідкових матеріалів та навчальних курсів.

Крім того, мова C# дозволяє створювати кросплатформні рішення, проте у даному проєкті основним середовищем використання є операційна система Windows.

Для створення графічного інтерфейсу користувача застосовано технологію Windows Presentation Foundation (WPF), яка є частиною платформи .NET та дозволяє реалізувати сучасний, гнучкий та привабливий інтерфейс. До основних переваг WPF можна віднести:

- реалізацію патерна проєктування MVVM, що забезпечує розділення логіки додатку та графічного інтерфейсу;
- розвинуті можливості для налаштування дизайну елементів управління;
- підтримку роботи з мультимедійним контентом, анімаціями, 2D та 3D графікою;
- інтеграцію з середовищем розробки Visual Studio та повну сумісність із C#.

Завдяки використанню WPF інтерфейс системи є інтуїтивно зрозумілим для користувача, зручним у використанні та достатньо гнучким для можливого розширення функціоналу у майбутньому [18].

У якості сховища даних для системи обрано хмарний сервіс Amazon Relational Database Service (AWS RDS), який є одним із найпопулярніших рішень для зберігання структурованих даних у вигляді реляційних баз даних. Основними причинами вибору AWS RDS стали:

- надійність та безпека: автоматичне резервне копіювання, шифрування даних, підтримка багатозонного розгортання;
- масштабованість: можливість динамічно змінювати обчислювальні ресурси залежно від навантаження;
- підтримка популярних СУБД: таких як MySQL, PostgreSQL, SQL Server, що дозволяє розробнику обрати відповідний варіант залежно від потреб проєкту;
- інтеграція з іншими хмарними сервісами AWS, що полегшує побудову комплексної системи;
- спрощене адміністрування: AWS RDS автоматично виконує оновлення, патчі та моніторинг стану бази даних, знижуючи витрати часу на обслуговування.

Використання AWS RDS дозволяє організувати централізоване та безпечне збереження інформації про тести, результати тестування користувачів та інші необхідні дані, забезпечуючи їх доступність з будь-якої точки світу [16].

Таким чином, використання у розробці системи тестування знань з іноземної мови таких технологій як C#, WPF та AWS RDS дозволяє досягти високої якості програмного продукту, що проявляється у зручності користування, надійності роботи, масштабованості та простоті супроводу і розвитку програмного забезпечення.

У процесі розробки програмного забезпечення для системи тестування знань з іноземної мови було обрано архітектурний шаблон Model-View-ViewModel (MVVM) як основний підхід до побудови структури застосунку. MVVM є стандартом для побудови застосунків у середовищі WPF завдяки своїй здатності ефективно розділяти логіку представлення даних, бізнес-логіку та самі дані, що робить застосунок модульним, масштабованим і зручним для тестування.

Основні компоненти шаблону MVVM:

1. Model (Модель) Відповідає за доступ до даних та їхнє збереження. У межах розроблюваної системи до моделі належать:
   - сутності бази даних: користувачі, тести, питання, результати;

- сервіси роботи з базою даних (наприклад, класи для підключення до AWS RDS);
- допоміжні класи, які реалізують бізнес-логіку (перевірка правильності відповідей, обрахунок результатів тощо).

2. View (Представлення) Містить лише XAML-опис інтерфейсу користувача. View відповідає за візуальне відображення даних користувачу та не містить жодної бізнес-логіки. У системі реалізовано такі представлення:

- Вікно авторизації та реєстрації користувача;
- Вікно вибору параметрів тестування (мова, тема, тип);
- Вікно проходження тестування;
- Вікно відображення результатів тестування.

3. ViewModel (Модель Представлення) Є посередником між Model та View, містить:

- властивості для прив'язки до елементів View;
- команди (ICommand), які реагують на дії користувача (натискання кнопок, вибір із списків тощо);
- логіку підготовки та обробки даних перед їх відображенням у View.

У розробці застосунку створено такі ViewModel:

- LoginViewModel — обробка авторизації та реєстрації;
- TestSelectionViewModel — обробка вибору параметрів тестування;
- TestingViewModel — управління поточним тестом та відповідями;
- ResultViewModel — формування та відображення результатів тестування;
- MainViewModel — забезпечує навігацію між іншими ViewModel.

Переваги використання MVVM у проєкті:

- Забезпечує розділення відповідальностей, що підвищує читаність і підтримуваність коду;
- Полегшує модульне тестування бізнес-логіки без залучення графічного інтерфейсу;
- Підтримує Data Binding, що дозволяє автоматично оновлювати відображення даних у View при їх зміні у ViewModel;

• Створює можливість для легкої модифікації або розширення функціоналу без зміни основних частин програми;

• Підвищує масштабованість програми у разі додавання нових функцій (наприклад, нові типи тестів, додаткові мови).

Особливості реалізації MVVM у застосунку:

• Кожна ViewModel реалізує інтерфейс INotifyPropertyChanged для підтримки двосторонньої прив'язки даних;

• Для обробки дій користувача використовуються команди, реалізовані через класи, що успадковують інтерфейс ICommand;

• Для зручної навігації між сторінками використовується MainViewModel, яка зберігає посилання на поточний ViewModel;

• Логіка роботи з базою даних (AWS RDS) ізольована у шарі Model для уникнення прямого доступу до даних із ViewModel.

Недоліки, що були враховані:

• Підвищена складність початкового проєктування структури програми;

• Необхідність створення додаткового коду для реалізації механізмів зв’язку між шарами (наприклад, реалізація INotifyPropertyChanged) [19].

Загальна структура застосування MVVM у даному проєкті представлена на рисунку 2.4.

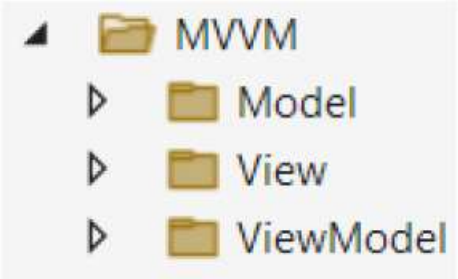


Рис. 2.4. Папки для класифікації файлів шаблону MVVM

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. 2.5).

Цей клас містить поля, представляють сутності бази даних для збереження інформації про користувачів, тести та результати тестування.

Опис основних класів:

- UserEntity — зберігає інформацію про користувача (ID, логін, пароль, роль);
- TestEntity — містить дані про тест (ID, мову, тип, тему, підтематику);
- QuestionEntity — питання тесту (ID, текст питання, варіанти відповідей, правильна відповідь);
- ResultEntity — результати тестування користувача (ID, UserID, TestID, результат, дата проходження).

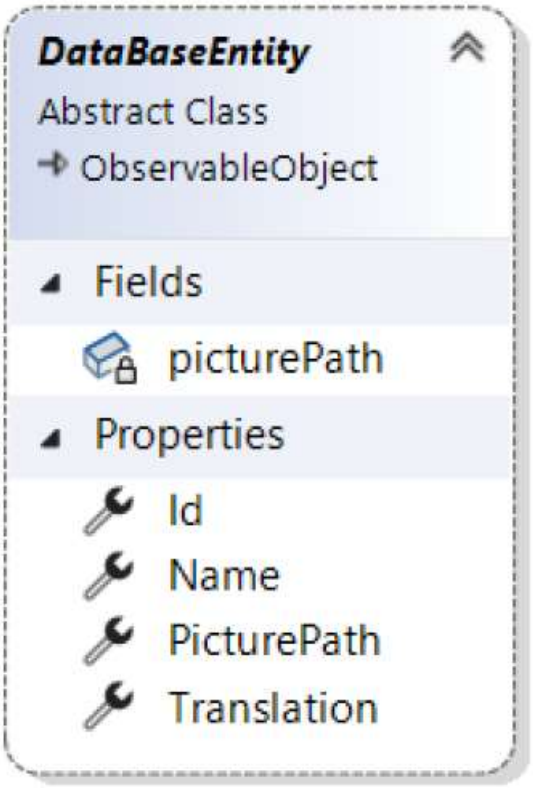


Рис. 2.5. Діаграма класу DataBaseEntity

Класи типу viewmodel наслідуються від класу ViewModel (рис. 2.6).

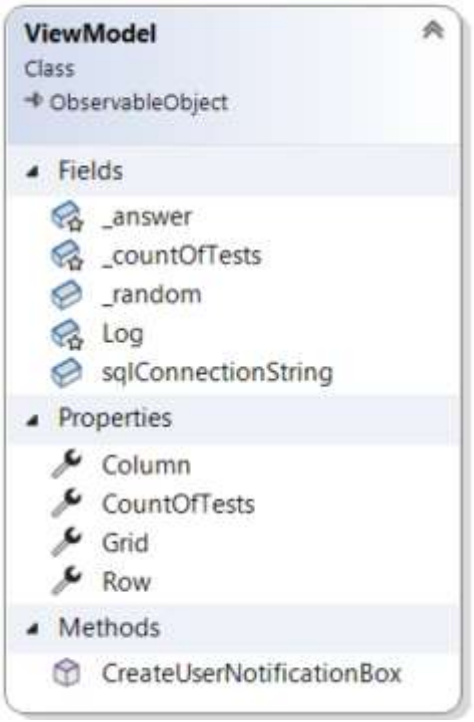


Рис. 2.6. Діаграма класу ViewModel

Цей клас реалізовує прив'язку даних між View та Model.

Опис основних класів:

- MainViewModel – управляє загальною навігацією та станом програми;
- LoginViewModel – обробляє авторизацію та реєстрацію користувача;
- TestSelectionViewModel – відповідає за вибір мови, типу тесту, теми та підтематики;
- TestingViewModel – обробляє процес проходження тестів, зберігає відповіді, підраховує результати;
- ResultViewModel – забезпечує відображення результатів користувачу.

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

3.1 Розробка засобу керування обліковими записами

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

Діаграма послідовності роботи засобу реєстрації зображена на рисунку 3.1.

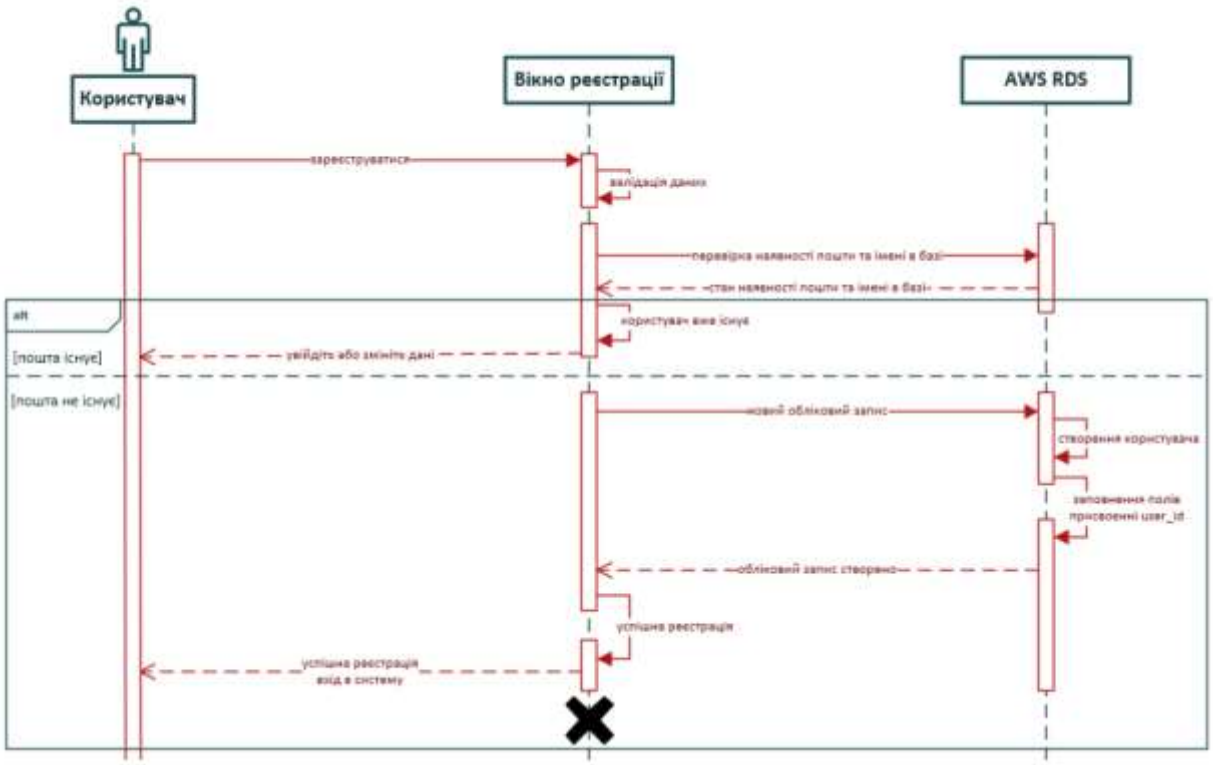


Рис. 3.1. Діаграма послідовності роботи засобу реєстрації

На діаграмі послідовності, зображеній вище, показано детальний процес реєстрації нового користувача в системі тестування знань з іноземної мови. У цьому процесі беруть участь три основні актори: Користувач, Вікно реєстрації (інтерфейсна частина додатку) та AWS RDS (віддалене хмарне сховище даних).

1. Користувач ініціює процес реєстрації шляхом натискання відповідної кнопки у вікні програми. Він заповнює обов'язкові поля форми реєстрації (електронна пошта, ім'я, пароль тощо) та надсилає дані для обробки у вигляді запиту до модуля інтерфейсу – Вікно реєстрації.

2. Вікно реєстрації здійснює локальну валідацію введених даних на відповідність вимогам системи:

- Перевіряється заповненість усіх обов'язкових полів;
- Перевірка формату електронної пошти;
- Мінімальні вимоги до паролю;
- Відсутність заборонених символів тощо.

3. У разі успішної валідації формується запит до хмарної бази даних AWS RDS для перевірки наявності користувача з вказаними поштою або ім’ям. База даних виконує пошук записів за полями "email" та "username".

4. Далі система реагує залежно від результату перевірки:

- Якщо пошта існує:
  - AWS RDS повертає повідомлення про наявність запису.
  - Користувачу відображається повідомлення: «Обліковий запис з такою поштою вже існує. Увійдіть або змініть дані».
  - На цьому реєстраційний процес завершується до моменту внесення змін користувачем.
- Якщо пошта не існує:
  - Вікно реєстрації ініціює створення нового облікового запису.
  - Формується запит до AWS RDS на створення нового запису користувача.
  - Система заповнює поля нового запису, включаючи автоматично згенерований унікальний ідентифікатор (user_id), що гарантує унікальність облікового запису.

5. Після успішного створення нового облікового запису база даних підтверджує завершення операції. Інтерфейс отримує відповідь про успішну реєстрацію.

6. Користувачу виводиться повідомлення про успішне створення облікового запису з пропозицією увійти до системи.

Особливості та переваги процесу:

- Перевірка унікальності пошти у базі даних гарантує, що не створюється дублюючих облікових записів.
- Використання AWS RDS як хмарного сховища забезпечує високу надійність збереження та доступності даних, їх резервне копіювання та масштабованість.
- Усі перевірки здійснюються як на стороні клієнта (валідація), так і на стороні сервера (перевірка у базі), що дозволяє уникнути помилок та забезпечити безпеку.
- Логіка виконання умов реалізована у вигляді альтернативного фрагменту (alt) на діаграмі, що чітко демонструє залежність подальших дій від наявності користувача у базі.
- Створення облікового запису супроводжується присвоєнням унікального user_id, що дозволяє однозначно ідентифікувати кожного користувача в системі.

Дана діаграма послідовності повністю відповідає вимогам до процесу реєстрації у сучасних програмних продуктах та забезпечує:

- цілісність даних,
- зручність для користувача,
- відсутність дублювання записів,
- безпечну роботу з обліковими записами.

Крім того, використання хмарної бази AWS RDS дозволяє реалізувати централізоване управління обліковими даними користувачів із можливістю масштабування в майбутньому, що важливо для розширення функціональності системи.

Вхід до системи тестування знань реалізовано за допомогою механізму авторизації, що забезпечує перевірку правильності введених користувачем облікових даних (логіну та паролю). Діаграма послідовності на рисунку 3.6 демонструє алгоритм виконання даної операції та взаємодію основних компонентів системи: користувача, вікна авторизації та бази даних AWS RDS.

Діаграма послідовності роботи засобу авторизації зображена на рисунку 3.2.

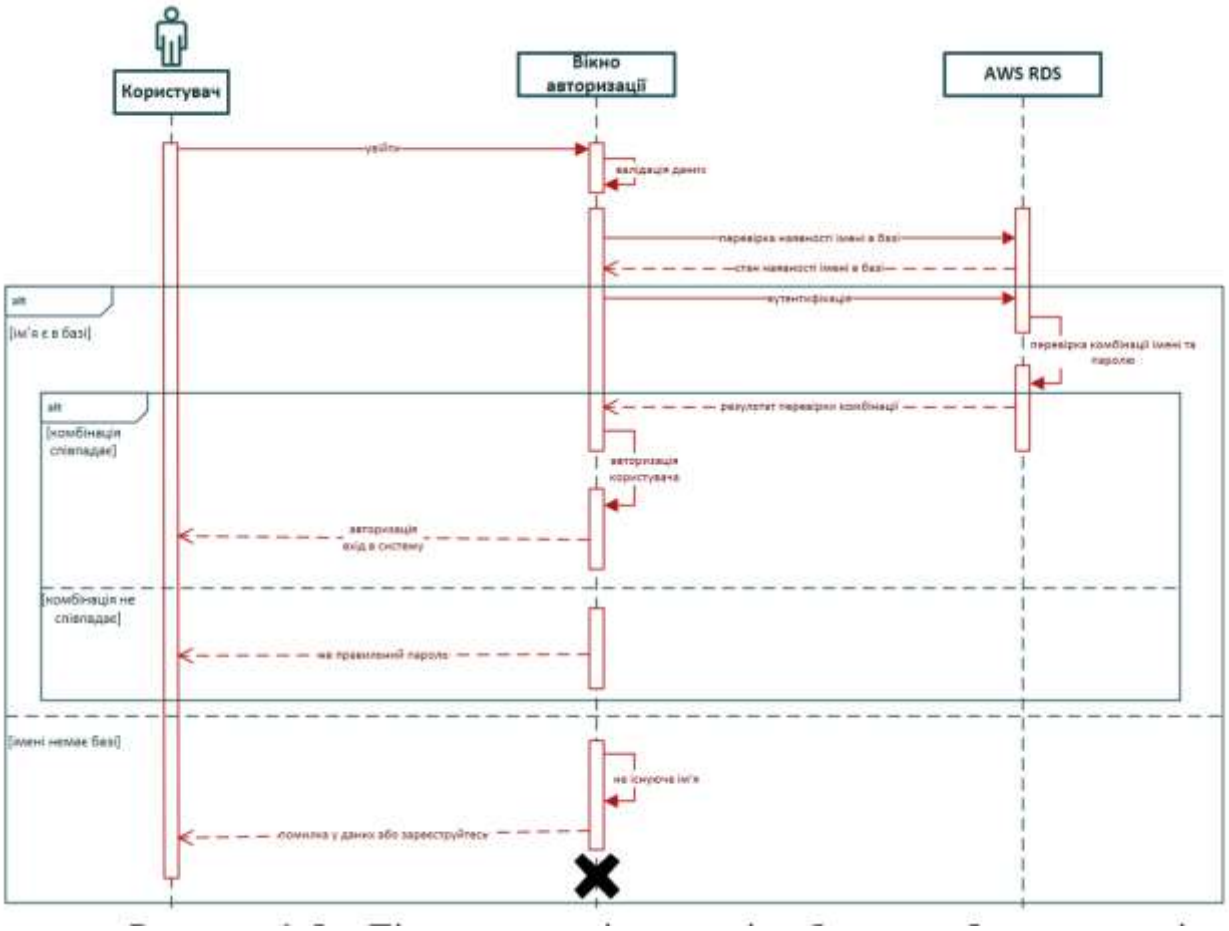


Рис. 3.2. Діаграма послідовності роботи засобу авторизації

Основні етапи процесу авторизації:

1. Ініціація авторизації. Користувач вводить ім’я (логін) та пароль у відповідних полях форми авторизації та натискає кнопку «Увійти». Запит передається до вікна авторизації.
2. Валідація даних. На клієнтській частині здійснюється перевірка заповненості обов’язкових полів:
   - Введення логіну та паролю;
   - Відповідність формату допустимим значенням.
3. Перевірка наявності імені користувача у базі. Після успішної локальної валідації вікно авторизації надсилає запит до AWS RDS для пошуку користувача за введеним ім’ям:
   - Якщо ім’я знайдене: відбувається перехід до наступного етапу.
   - Якщо ім’я відсутнє: повертається повідомлення «Не існує ім’я» з пропозицією перевірити введені дані або пройти реєстрацію.

4. Аутентифікація. У випадку виявлення імені виконується перевірка правильності комбінації логіну та паролю. Результат цієї перевірки може бути двох типів:
    - Комбінація співпадає: користувачу надається дозвіл на вхід у систему, формується сеанс роботи.
    - Комбінація не співпадає: відображається повідомлення «Неправильний пароль» із пропозицією повторити спробу.
5. Результат авторизації.
    - У разі успішної перевірки користувач отримує доступ до основного функціоналу програми.
    - У разі помилки система надає детальне пояснення: некоректні дані або відсутність користувача у базі.

Запропонований алгоритм авторизації забезпечує надійний контроль доступу до системи тестування знань, знижує ймовірність несанкціонованого входу та підвищує зручність для кінцевого користувача завдяки деталізованим підказкам у разі помилок.

3.2 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. 3.3).

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

Рис. 3.3. 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. 3.4) програма отримує дані, які користувач ввів, під’єднується до хмарного середовища та перевіряє чи існує такий користувач за такими даним.

Даний алгоритм реалізує перевірку правильності введених облікових даних (логіну та паролю), а також здійснює відповідні дії залежно від результатів перевірки.

1. Початок.
   - Ініціація процесу авторизації.
2. Перевірка на заповненість поля логіну (userName):
   - Якщо поле логіну порожнє (string.IsNullOrEmpty(userName) повертає true), викликається функція CreateUserNotificationBox(), яка формує відповідне повідомлення для користувача (наприклад, "Поле логіну не заповнене"). Після цього алгоритм завершує роботу.
3. Перевірка на заповненість поля паролю (userPassword):

- Якщо поле паролю порожнє (string.IsNullOrEmpty(userPassword) повертає true), також викликається функція CreateUserNotificationBox(), яка інформує користувача про необхідність заповнення цього поля. Подальше виконання переривається.

4. Підготовка до виконання SQL-запиту:

- Створюється об'єкт типу User для зберігання введених даних (logInUser).
- Формується SQL-запит для отримання користувача з бази даних (sqlGetUser).
- Створюється та ініціалізується підключення до бази даних (sqlConnection).

5. Відкриття з’єднання з базою даних:

- Викликається метод sqlConnection.Open(), що встановлює з’єднання з базою AWS RDS.

6. Формування та виконання SQL-команди:

- Створюється SQL-команда (sqlCommand) для пошуку користувача.
- Отримується результат виконання у вигляді об’єкта sqlDataReader.

7. Перевірка наявності рядків у результаті запиту:

- Якщо записів у результаті немає (sqlDataReader.HasRows == false), викликається функція CreateUserNotificationBox(), що повідомляє користувача про помилку авторизації (наприклад, "Користувача не знайдено"). З’єднання із базою даних закривається, авторизація завершується.

8. Читання отриманих даних:

- Якщо записи знайдені (sqlDataReader.HasRows == true), виконується зчитування даних (sqlDataReader.Read()), після чого витягується пароль користувача (gotUser = sqlDataReader.GetString()).

9. Перевірка правильності паролю:

- Якщо пароль з бази (gotUser.Pass) співпадає з введеним паролем (logInUser.Pass), викликається функція SetAccountData(gotUser), а також відкривається головне вікно системи (OpenMainWindow()).

- У випадку невідповідності паролів викликається CreateUserNotificationBox(), яка інформує користувача про невірний пароль.

10. Закриття з’єднання з базою даних:

- Викликається метод sqlConnection.Close() для завершення роботи з базою.

11. Завершення процесу.

- Завершення виконання алгоритму.

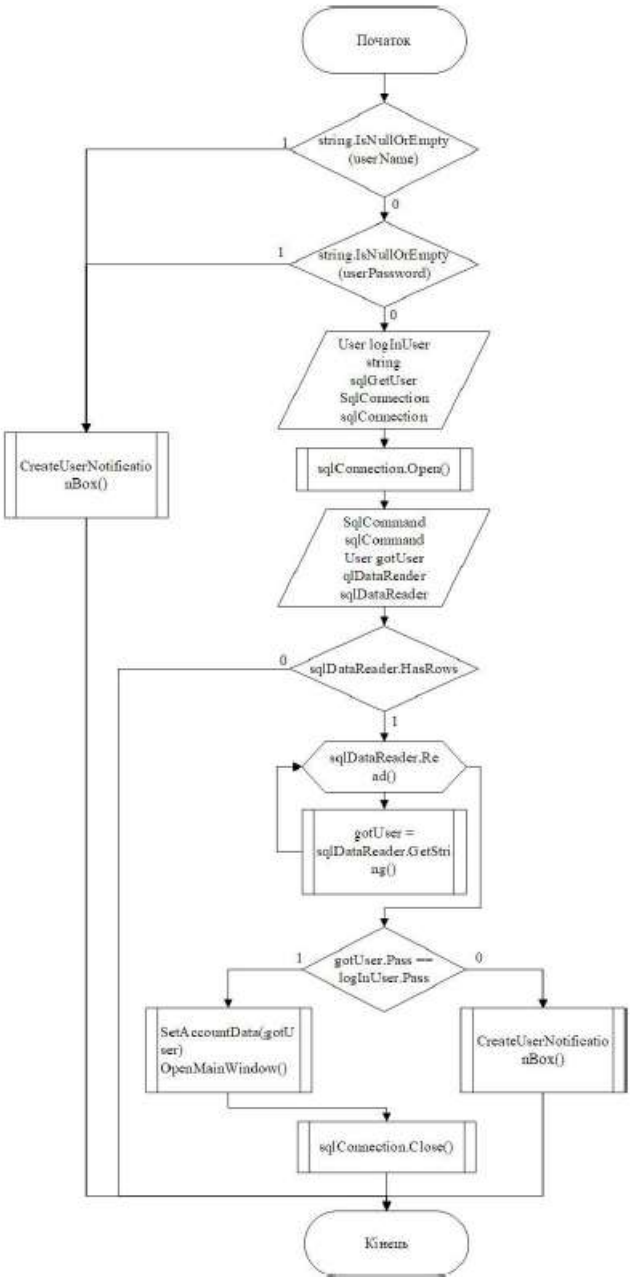


Рис.3.4. Блок-схема алгоритму авторизації користувача

Представлений алгоритм забезпечує надійну перевірку облікових даних користувача та мінімізує ризики несанкціонованого доступу до системи. Усі виняткові ситуації супроводжуються інформативними повідомленнями, що підвищує зручність взаємодії користувача із системою.

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. 3.5).

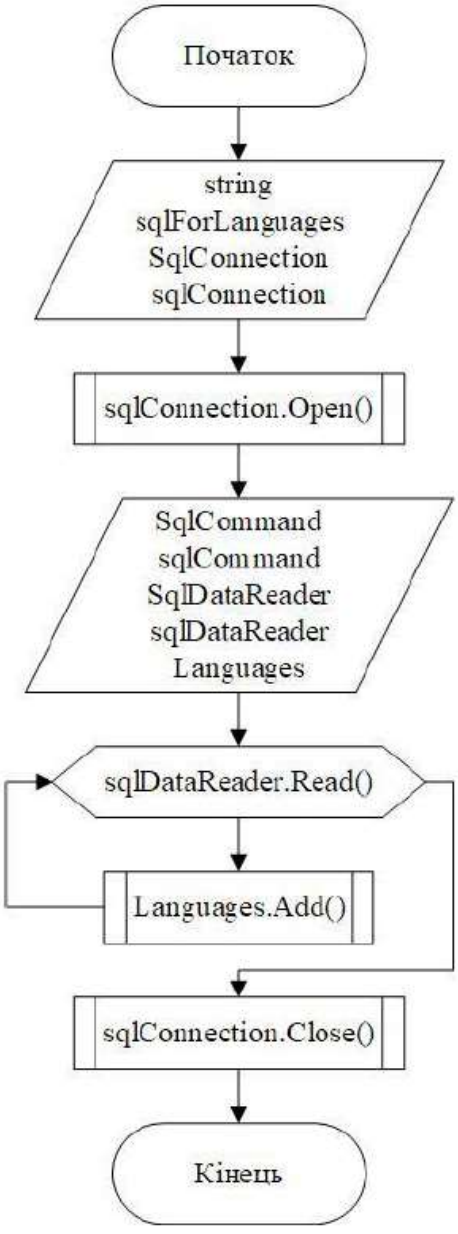


Рис. 3.5. Блок-схема алгоритму отримання списку мов для тестування

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

Отримання даних про тестування (рис. 3.6) являється найскладнішим, адже відбувається створення одразу набору тестувань певного типу для всіх слів з обраних підтем.

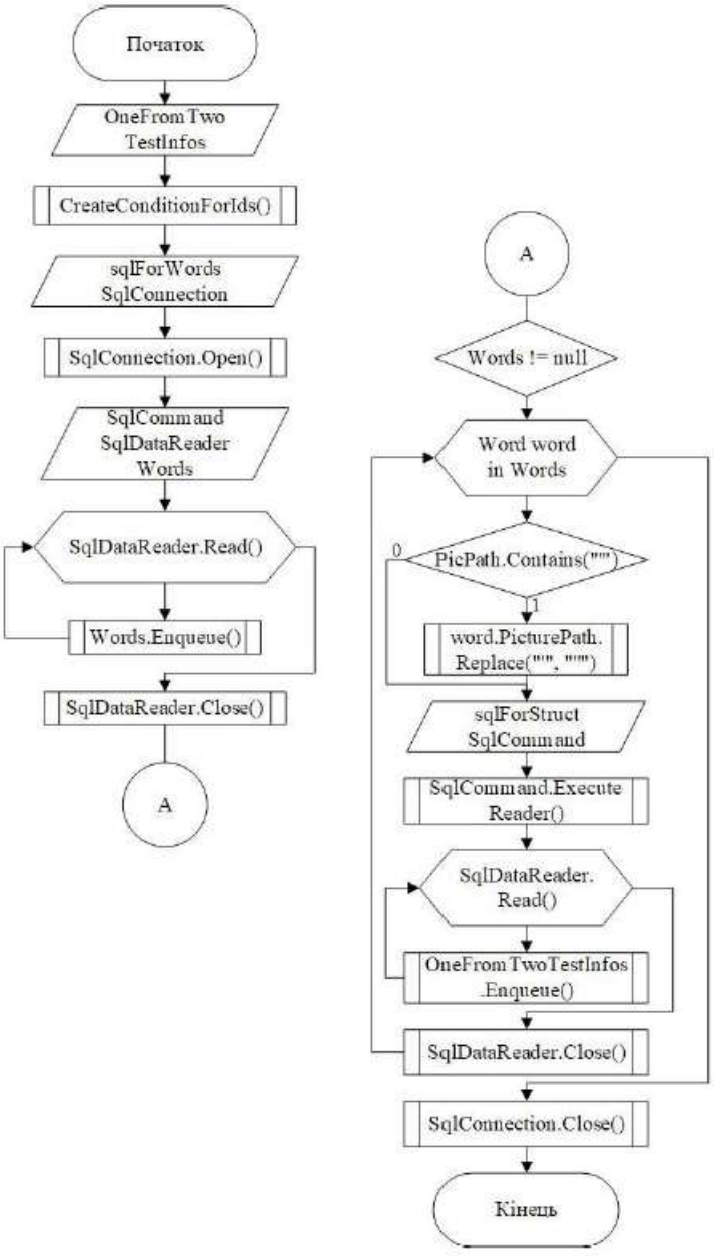


Рис. 3.6. Блок-схема алгоритму отримання даних про тестування

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 3.7.

```
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); }
    }
}
```

Рис. 3.7. Програмна реалізація класу DataBaseEntity

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

3.3 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. 3.8). 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.

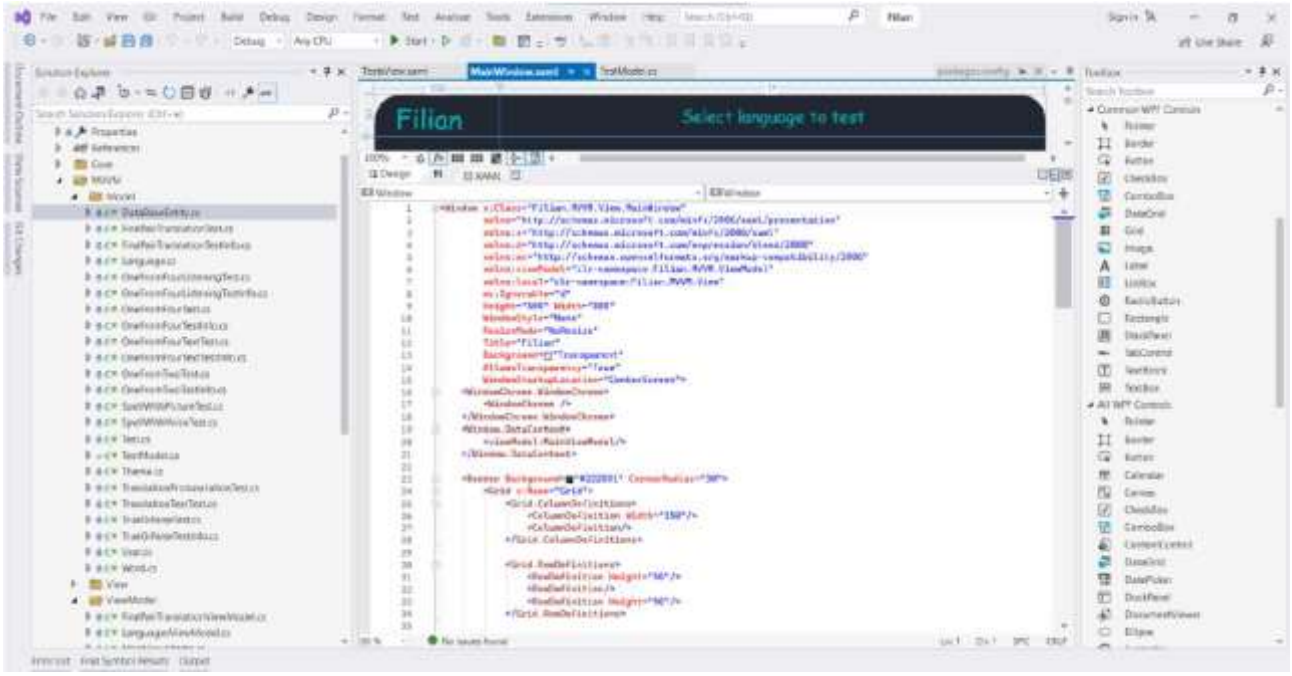


Рис. 3.8. Розробка графічного інтерфейсу

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. 3.9), за допомогою якого користувач зможе зайти в систему.



Рис. 3.9. 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.3.10).

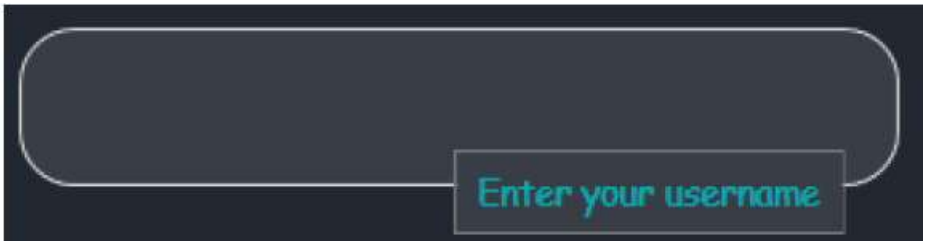


Рис.3.10. Підказка для введення даних в поле

Розробка вікна авторизації зображена на рисунку 3.11.

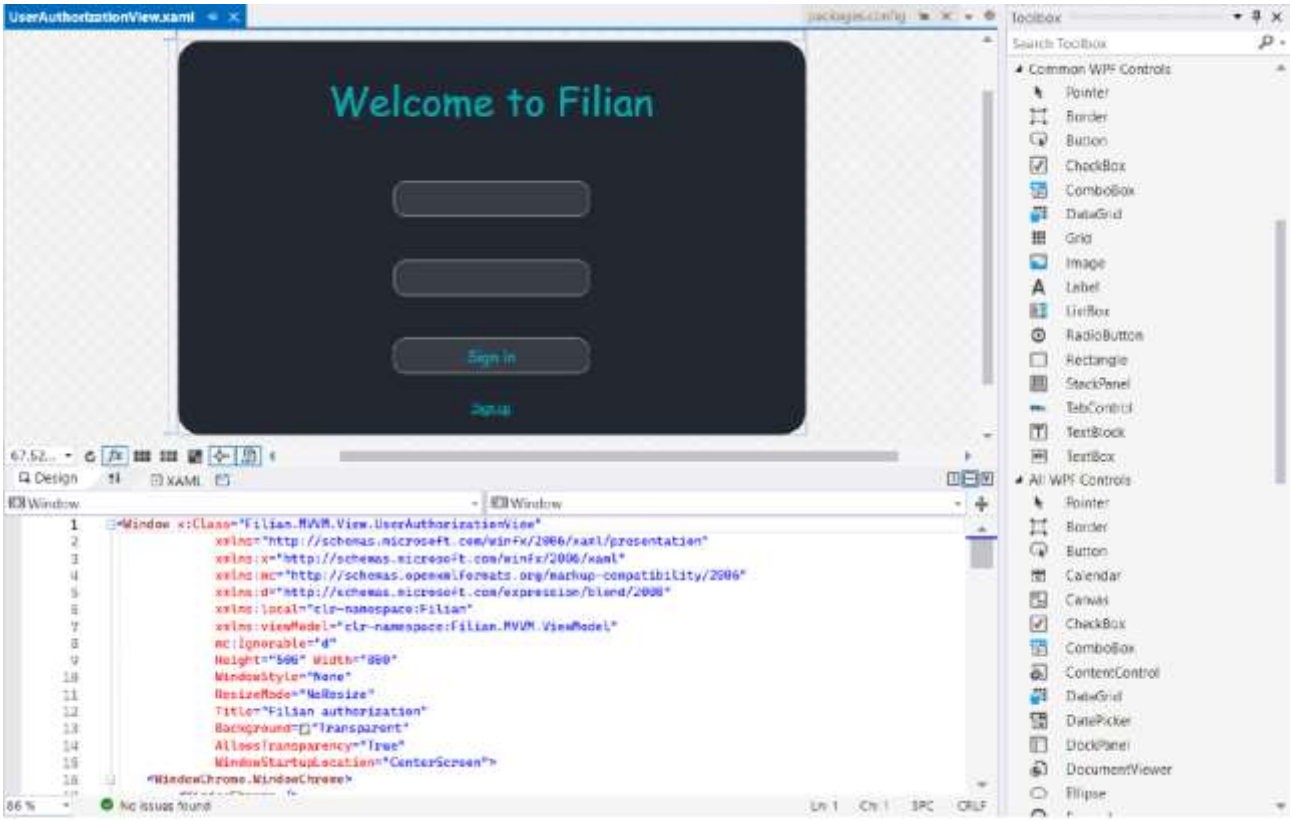


Рис. 3.11. Розробка вікна авторизації

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. 3.12). 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.

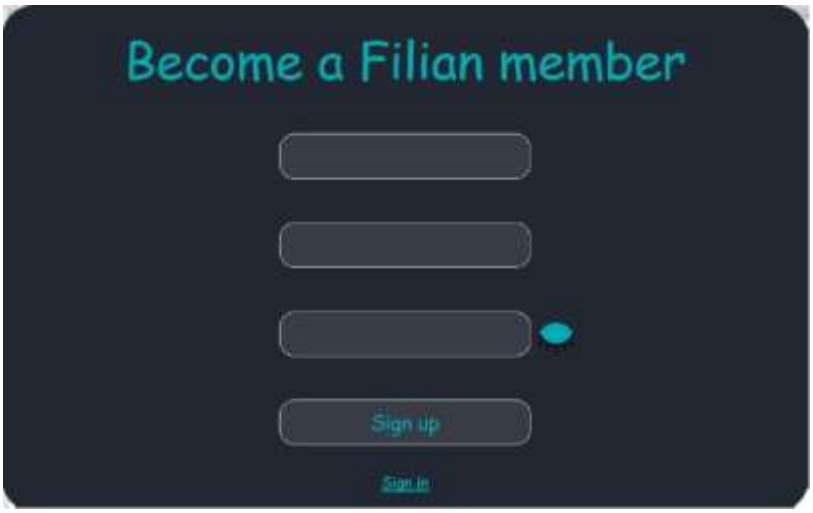


Рис. 3.12. Вікно реєстрації користувача

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

Вікно авторизації та вікно реєстрації (рис.3.13) являються майже ідентичними. Їх відрізняє лише текст та додаткове поле у вікні реєстрації для введення електронної адреси для створення користувача.

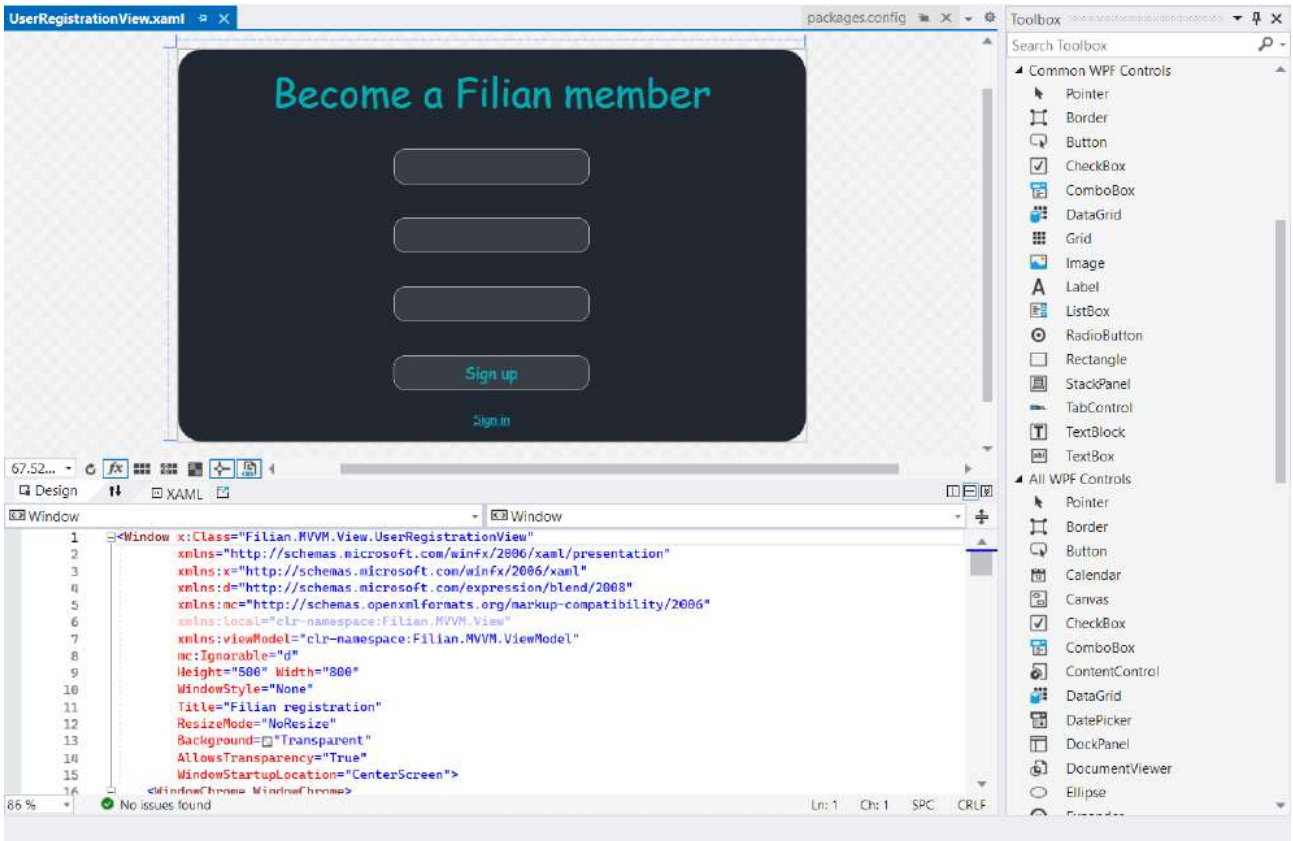


Рис. 3.13. Розробка вікна реєстрації

У вікнах авторизації та реєстрації поле для пароля приховує сам пароль спеціальними символами (рис. 3.14), і для того щоб перевірити вірність введення, можна скористуватись функцією «показати пароль», натиснувши на зображення закритого ока (рис. 3.15).



Рис. 3.14. Приховування паролю



Рис. 3.15. Функція «показати пароль»

Після того, як користувач авторизувався, він побачить вікно програми, яка вже пропонує йому обрати мову для тестування (рис. 3.16). Наразі наявно 7 мов, а саме: українська, англійська, іспанська, португальська, китайська, німецька та польська.



Рис. 3.16. 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. 3.17) є основним засобом взаємодії користувача з програмою.

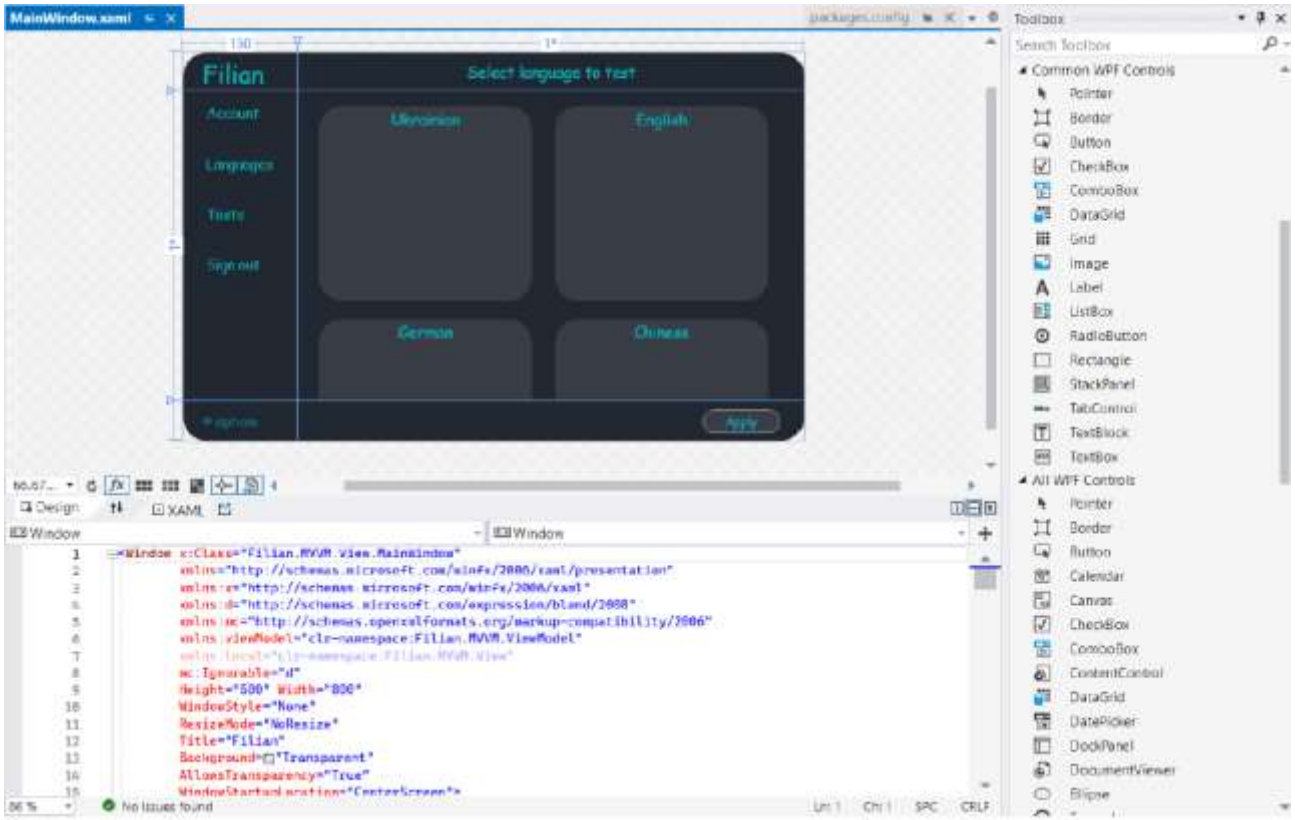


Рис. 3.17. Розробка головного вікна

Сторінка облікового запису (рис. 3.18) містить інформацію про поточного користувача в системі.



Рис. 3.18. 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 3.19.



Рис. 3.19. 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. 3.20), користувач може побачити перелік тестів різного типу та обрати один з десяти доступних.

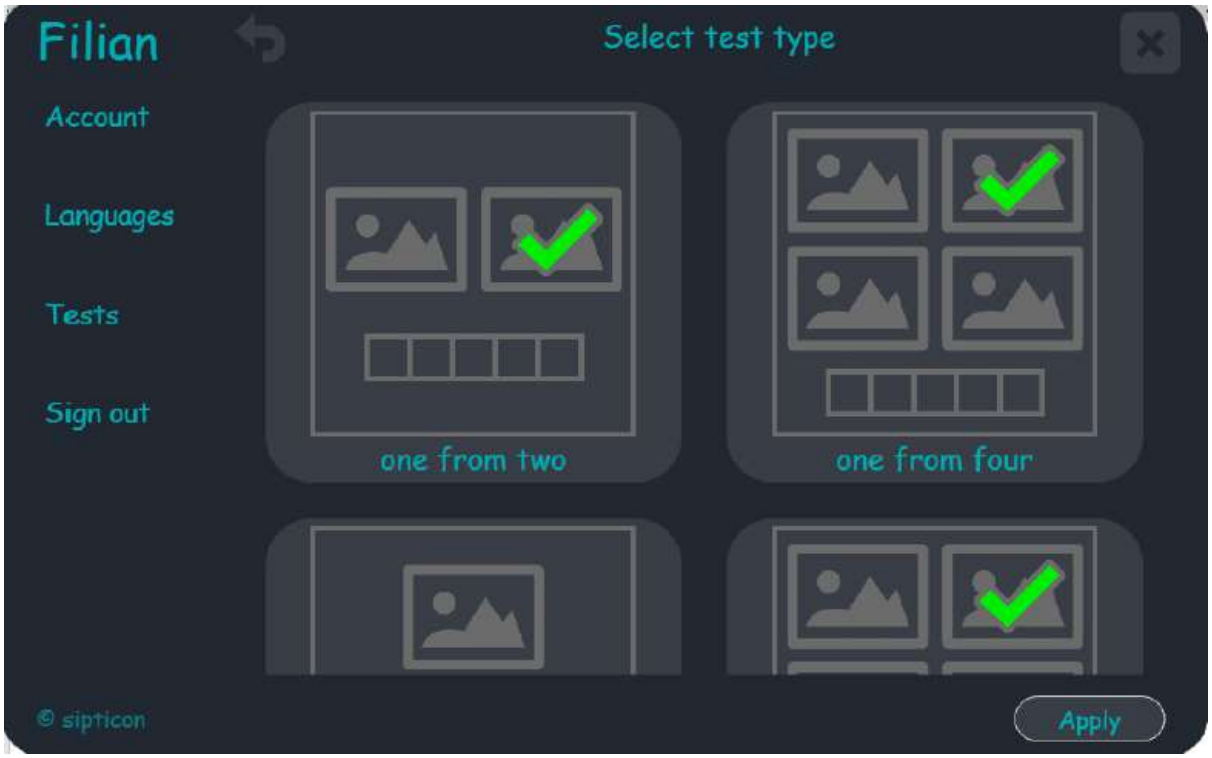


Рис. 3.20. Сторінка вибору типу тесту

Після того, як користувач обрав тип тестування, він зможе вибрати тему, по якій проводитиметься тестування (рис. 3.21). 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.

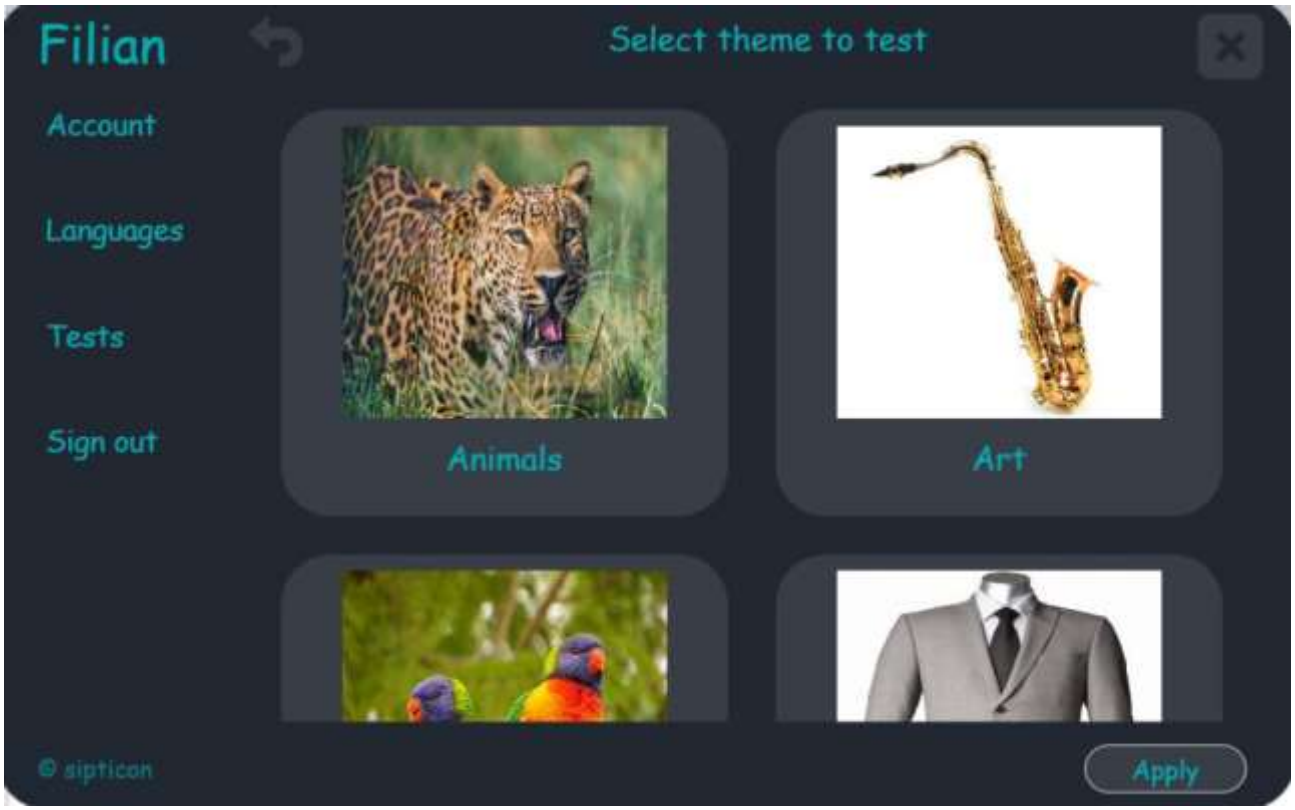


Рис. 3.21. Сторінка вибору теми

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.

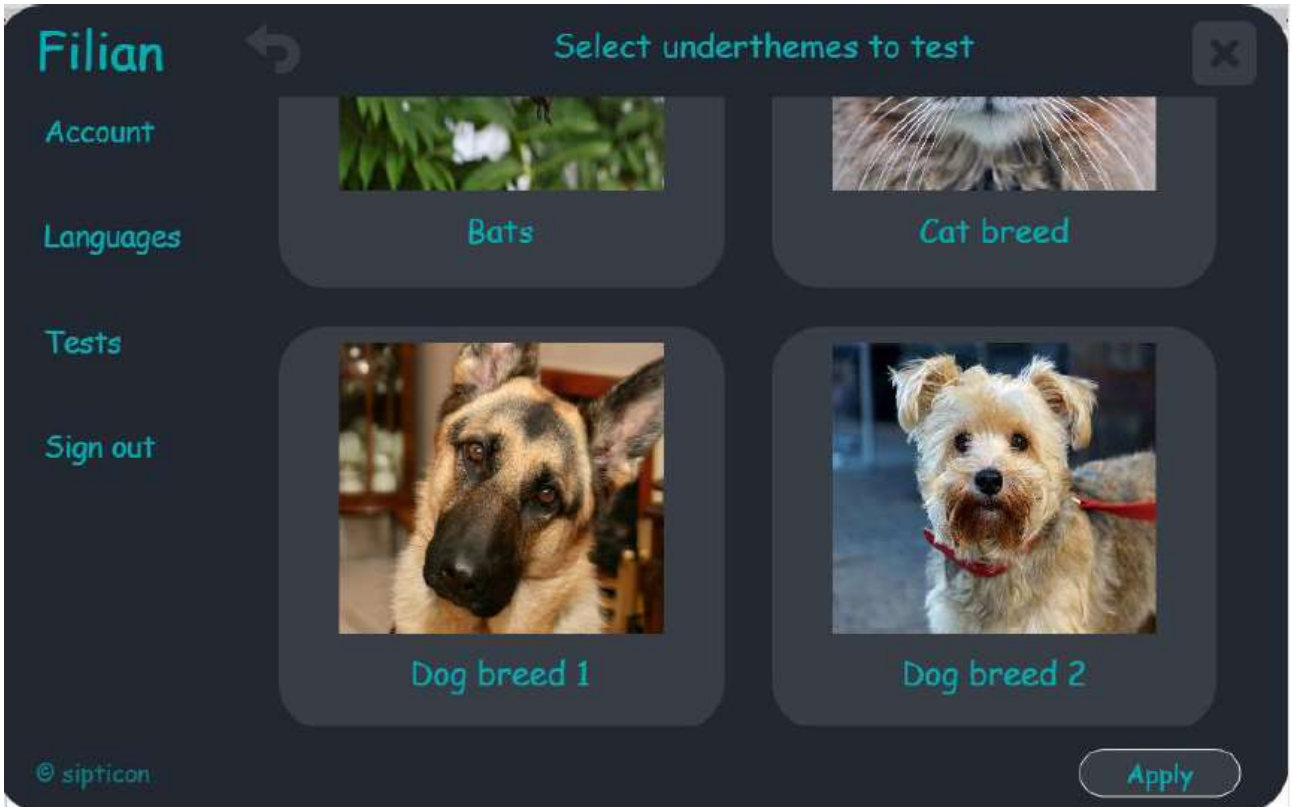


Рис. 3.22. Сторінка вибору підтем для теми «Тварини»

Після вибору підтем – розпочнеться тестування (рис. 3.23). Вгорі вікна можна побачити кількість тестів яка залишилась до закінчення тестування.

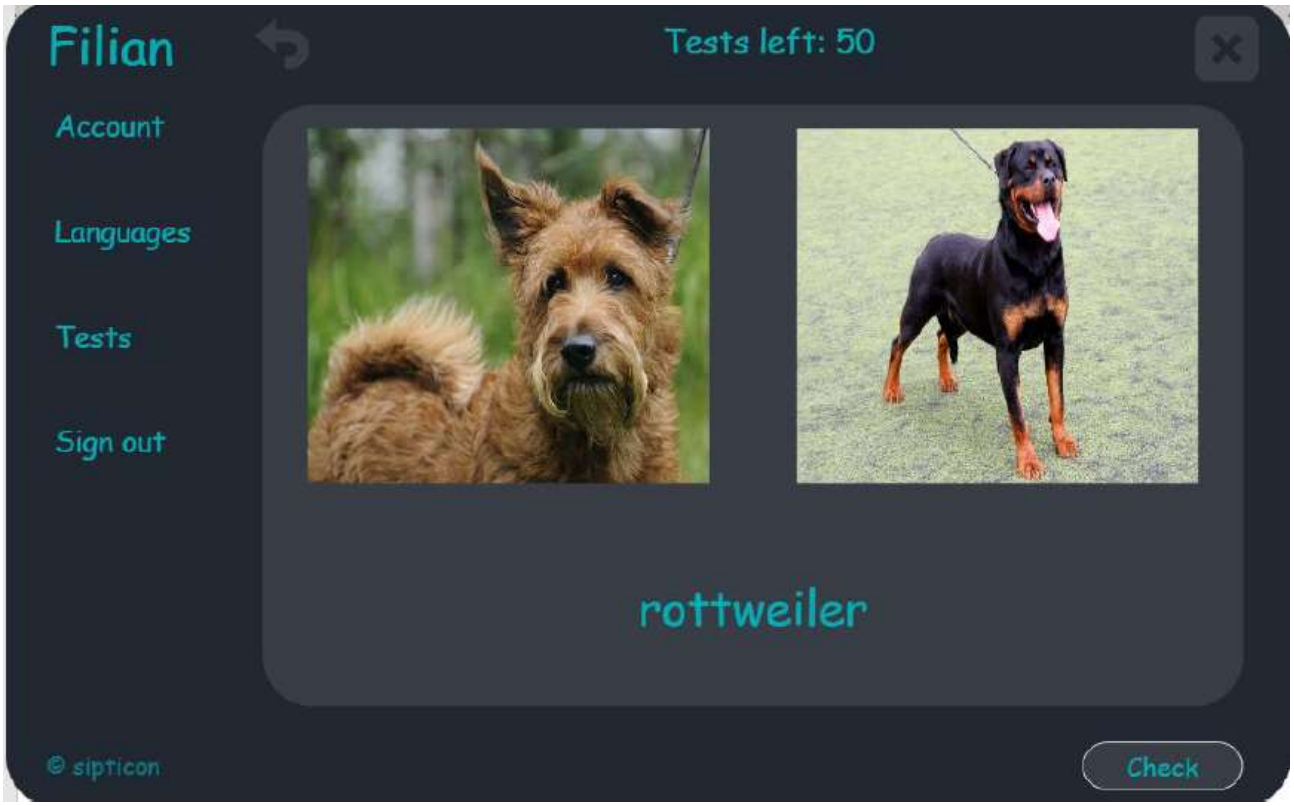


Рис. 3.23. Сторінка обраного тесту

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. 3.24) 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.

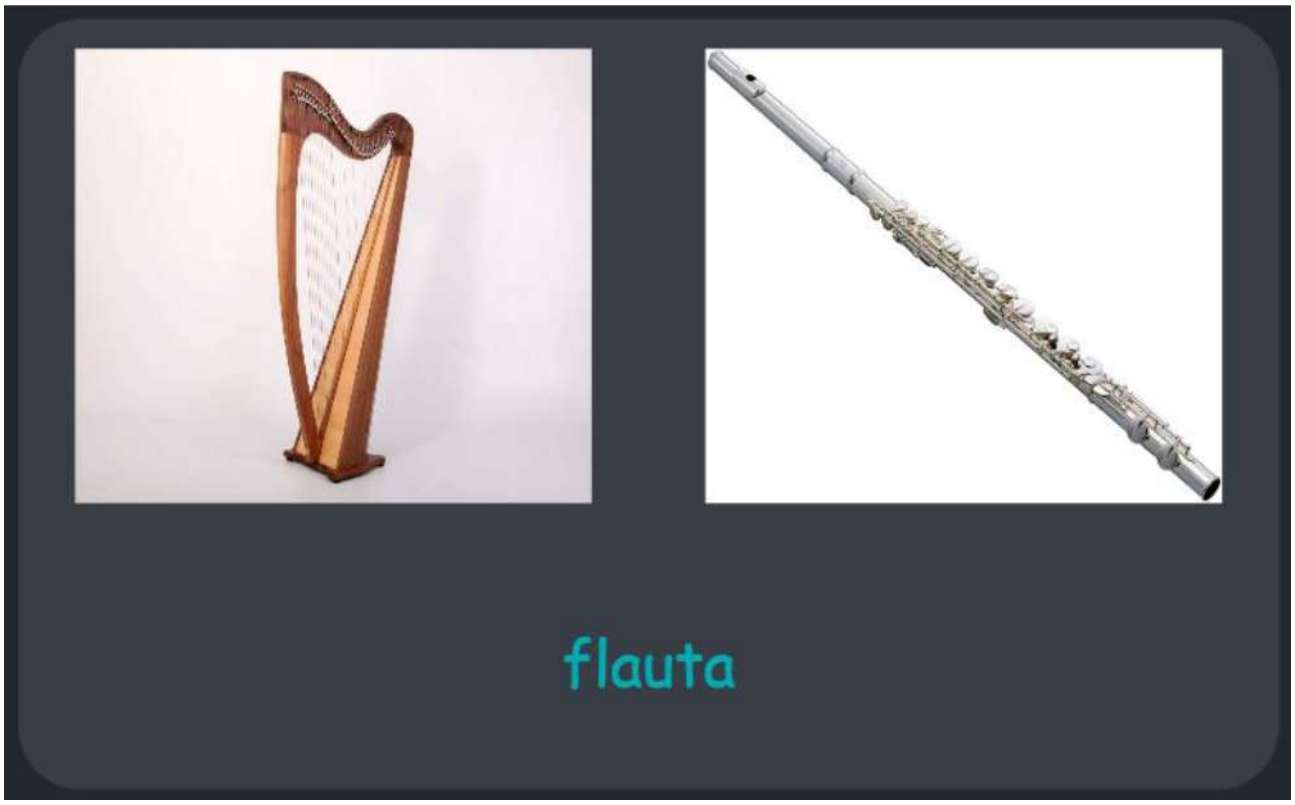


Рис. 3.24. Форма першого тестування

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

Вікно сповіщення (рис. 3.25) 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.

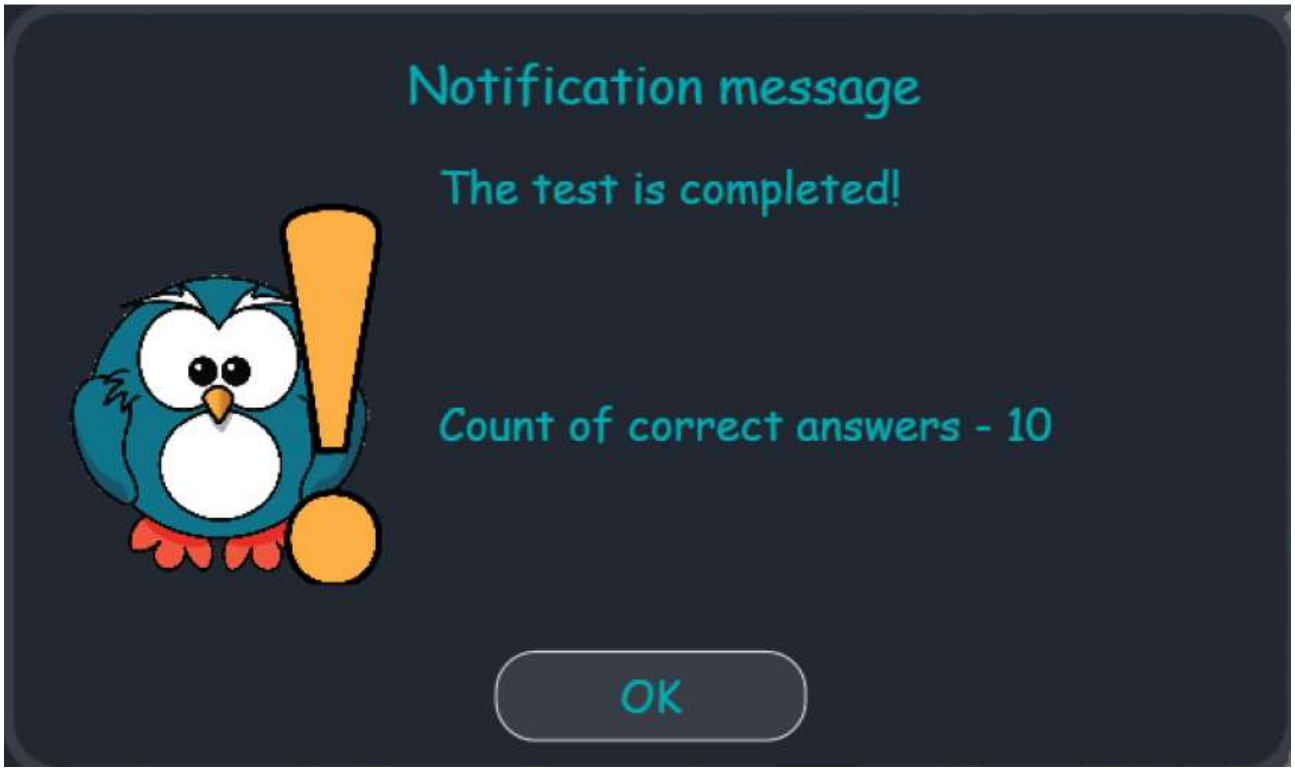


Рис. 3.25. Вікно сповіщення

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.

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. 3.26) користувач, за допомогою «» може обрати тип тестування за яким хоче переглянути статистику правильних відповідей.



Рис.3.26. 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 3.27.

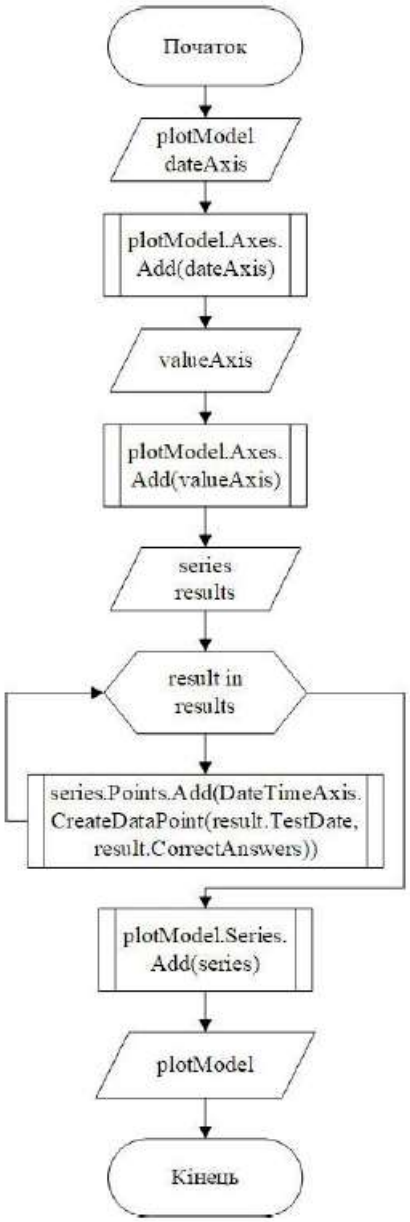


Рис. 3.27. Блок-схема алгоритму створення графіку звітності проходження тестувань

3.4 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. 3.28). 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.

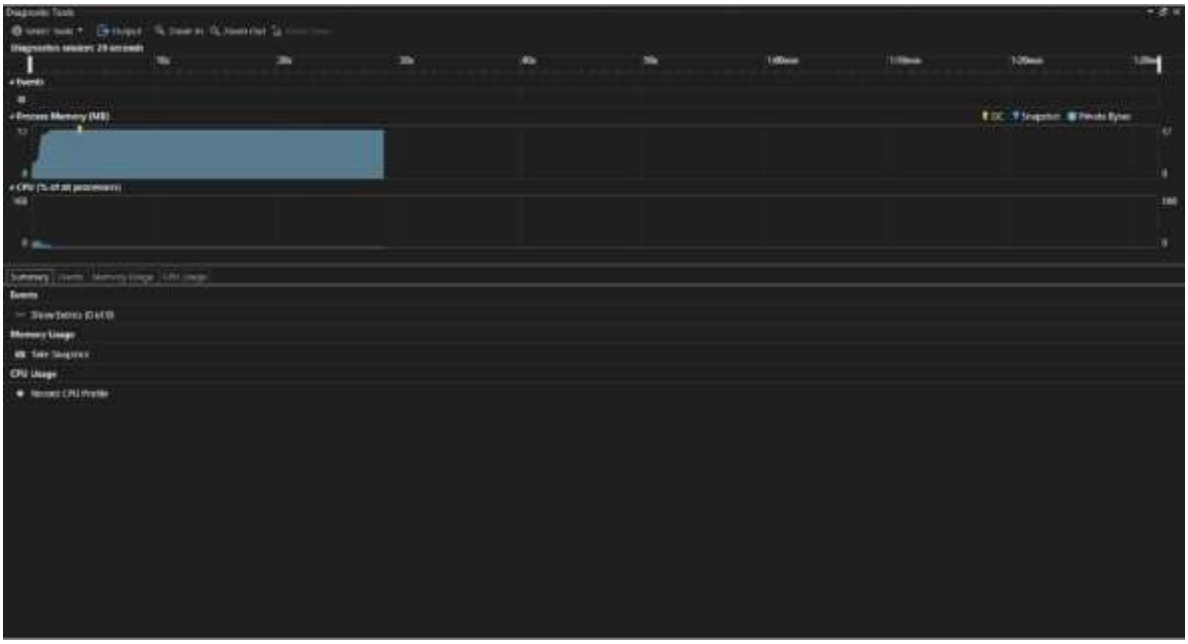


Рис. 3.28. Вікно тестування

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. 3.29), що є правильною поведінкою системи, оскільки інформація про ім'я є обов'язковою для подальшої роботи користувача.

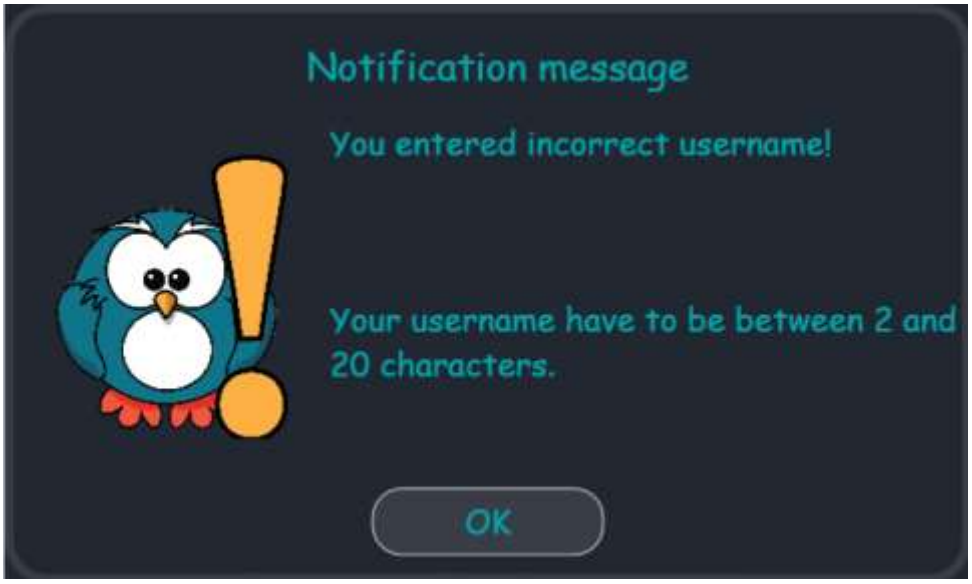


Рис. 3.29. Авторизація користувача без імені

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. 3.30).

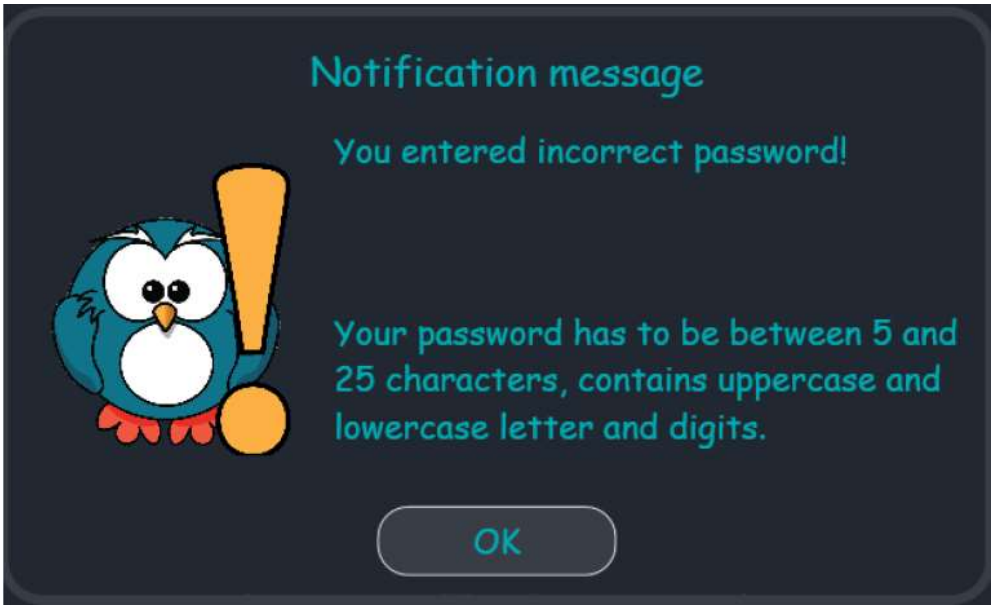


Рисунок 3.30. Авторизація користувача без пароля

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. 3.31).

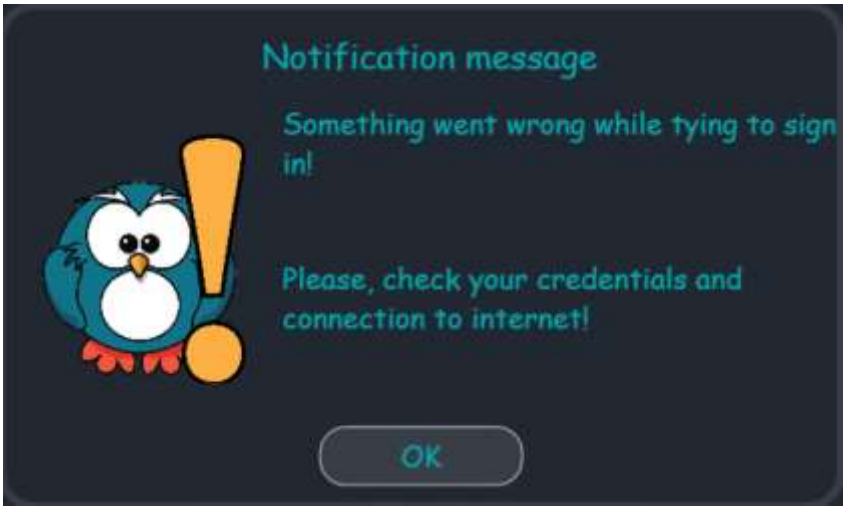


Рис. 3.31. Відсутність інтернет з’єднання

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. 3.32).

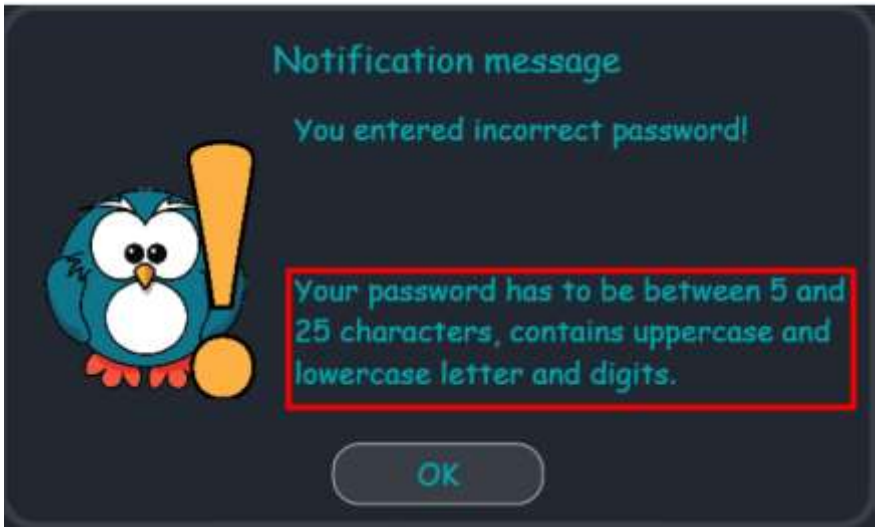


Рис. 3.32. Валідація поля паролю

ВИСНОВКИ

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

Поставлена мета роботи була досягнута шляхом реалізації всіх запланованих завдань:

1. Проведено аналіз існуючих програмних рішень для тестування знань з іноземної мови. В результаті було визначено основні переваги та недоліки існуючих систем, що дало змогу обґрунтувати необхідність створення власного програмного продукту з урахуванням актуальних вимог користувачів. Проведений аналіз засвідчив, що для комплексного контролю знань з англійської мови найбільш ефективними є платформи Busuu, Preply та Duolingo, які дозволяють здійснювати тестування основних видів мовленнєвої діяльності — читання, аудіювання, письма та говоріння. Сервіси Drops і LingQ доцільно використовувати як додаткові інструменти для розвитку лексичної компетентності та навичок розуміння автентичних текстів, проте вони не забезпечують повноцінного тестування граматичних та продуктивних навичок.
2. Сформульовано вимоги до функціоналу та інтерфейсу розроблюваного додатку. Основна увага приділялася зручності користування, простоті навігації, адаптивності до різних типів тестів та можливості роботи з базою даних у хмарному середовищі.
3. Розроблено архітектуру програмного забезпечення з використанням шаблону проєктування Model-View-ViewModel (MVVM), що дозволило забезпечити чіткий розподіл відповідальності між логікою програми, даними та інтерфейсом користувача.
4. Спроєктовано та реалізовано базу даних на платформі Amazon Relational Database Service (AWS RDS) для збереження інформації про користувачів, тестові завдання та результати проходження тестів. Використання хмарної

бази даних забезпечує надійність збереження даних та їх доступність з будь-якої точки світу.

5. Розроблено WPF-додаток мовою C#, що реалізує всі функціональні можливості: реєстрацію та авторизацію користувачів, вибір мови та типу тесту, проходження тестування, отримання результатів та їх збереження у базі даних.
6. Проведено тестування програмного забезпечення, що підтвердило його працездатність, відповідність заданим функціональним та нефункціональним вимогам, а також зручність використання кінцевими користувачами.

Таким чином, у процесі виконання дипломної роботи було розв’язано всі поставлені завдання, а розроблена система має перспективи для подальшого вдосконалення та розширення функціоналу, зокрема шляхом інтеграції з мобільними платформами, додаванням можливості адаптивного тестування та розширенням бази тестових завдань.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Важливість вивчення мов URL: https://clubinternational.com.ua/?p=233.
2. Де самостійно вивчати іноземну мову URL: https://happymonday.ua/13-populyarnyh-mobilnyh-dodatkiv.
3. Монастирський О.М., Войтко В.В. Розробка програми тестування знань з іноземної мови. Матеріали LIII Науково-технічної конференції підрозділів Вінницького національного технічного університету, Вінниця, 2024. URL: 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. Науковий часопис НПУ імені М. П. Драгоманова, Київ, 2019. URL: http://nc-s5.npu.edu.ua/article/view/161611.
5. Топ-10 застосунків для вивчення іноземних мов URL: https://speka.media/top-10-zastosunkiv-dlya-vivcennya-inozemnix-mov-pkl0d9.
6. Busuu a Cheg Servise URL: https://www.busuu.com/.
7. Busuu Wikipedia URL: https://uk.wikipedia.org/wiki/Busuu.
8. LanguageDrops URL: https://languagedrops.com/.
9. Drops Wikipedia URL: https://en.wikipedia.org/wiki/Drops_(app).
10. LingQ URL: https://www.lingq.com/uk/.
11. Preply URL: https://preply.com/.
12. Duolingo URL: https://uk.duolingo.com/.
13. Duolingo Wikipedia URL: https://uk.wikipedia.org/wiki/Duolingo.
14. C Sharp URL: 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 URL: https://uk.wikipedia.org/wiki/Microsoft_Visual_Studio.
18. XAML URL: 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 URL: https://uk.wikipedia.org/wiki/Реляційна_база_даних.
21. NoSQL URL: https://uk.wikipedia.org/wiki/NoSQL.
22. Microsoft SQL server URL: https://uk.wikipedia.org/wiki/Microsoft_SQL_Server.
23. Oxyplot URL: https://oxyplot.github.io/.
24. Richards J.C., Rodgers T.S. Approaches and Methods in Language Teaching. Cambridge : Cambridge University Press, 2014. 400 p.
25. Brown H.D. Teaching by Principles: An Interactive Approach to Language Pedagogy. New York : Pearson Education, 2015. 576 p.
26. Weir C.J. Language Testing and Validation: An Evidence-Based Approach. Palgrave Macmillan, 2005. — 328 p.
27. Савченко С.О. Теорія і практика тестування знань. Київ : Центр учбової літератури, 2019. 248 с.
28. Прессман Р.С. Розробка програмного забезпечення: практичний підхід. К. : Либідь, 2020. 944 с.
29. Мартін Р.К. Чистий код : створення, аналіз і рефакторинг. К. : Наш Формат, 2021. 464 с.
30. Freeman E., Robson E. Head First Design Patterns. O'Reilly, 2020. 694 p.
31. Welling L., Thomson L. PHP and MySQL Web Development. Addison-Wesley, 2017. 1008 p.
32. Flanagan D. JavaScript: The Definitive Guide. O'Reilly Media, 2020. 706 p.
33. Nixon R. Learning PHP, MySQL & JavaScript. O'Reilly Media, 2018. 812 p.

34. Lutz M. Learning Python. O'Reilly Media, 2013. 1648 p.
35. Кухаренко В.М. Дистанційне навчання: теорія і практика. Київ : Центр учбової літератури, 2019. 256 с
36. Іванова І.В. Інформаційно-комунікаційні технології у навчанні іноземних мов. Київ : КНЕУ, 2020. — 184 с.
37. Сорокоумова Т.А. Електронне навчання: нові можливості освіти. Київ : Освіта України, 2018. 204 с.
38. White C. Language Learning in Distance Education. Cambridge : Cambridge University Press, 2003. 232 p.
39. Hubbard P. Computer Assisted Language Learning: Critical Concepts in Linguistics. Routledge, 2009. 320 p.
40. Levy M. Computer-Assisted Language Learning: Context and Conceptualization. Oxford University Press, 1997. 280 p.
41. Blake R.J. Brave New Digital Classroom. Georgetown University Press, 2013. 232 p.
42. Литвиненко О.М. Особливості комп'ютерного тестування у процесі вивчення іноземних мов. Педагогічна освіта: теорія і практика. 2019. Вип. 26. — С. 45–50.
43. Гусєва О.В. Розробка тестових систем для контролю знань з іноземних мов // Інформаційні технології і засоби навчання. 2020. №2 (76). С. 35–41.
44. Шевченко О.В. Онлайн-платформи для контролю знань з іноземної мови: переваги та недоліки // Освіта в умовах цифровізації. 2021. Т. 1. С. 88–94.
45. Chapelle C.A., Voss E. Computer-Assisted Language Learning: A Research Agenda. John Benjamins Publishing, 2016. 192 p.