КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
імені ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій

Кафедра прикладних інформаційних систем

122 «Комп'ютерні науки»

(шифр і назва спеціальності)

«Прикладне програмування»

(назва освітньої програми)

Кваліфікаційна робота бакалавра

на тему: «Програмна система тестування студентів за допомогою сучасних веб-фреймворків»

Виконав _____
(Підпис)
Касміна Ілля Ігорович
(прізвище, ім'я, по батькові)

Керівник _Плескач Валентина Леонідівна
(прізвище, ім'я, по батькові)

_____
(Резолюція «До захисту»)

Попередній захист:

_____
_
(Висновок: "До захисту в екзаменаційній комісії ")
Завідувач кафедри _____ Плескач В.Л.
(Підпис) (Прізвище, ініціали)
(Дата)

Київ – 2021

КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ БАКАЛАВРА

Номер	Назва етапів кваліфікаційної роботи бакалавра	Термін виконання етапів кваліфікаційної роботи бакалавра	Відмітка про виконання
1.	Вибір теми та наукового керівника кваліфікаційної роботи бакалавра	26.10.2020	виконано
2.	Видача завдання кваліфікаційної роботи бакалавра	23.11.2020	виконано
3.	Настановча групова співбесіда з питань кваліфікаційної роботи бакалавра	01.12.2020	виконано
4.	Затвердження плану кваліфікаційної роботи бакалавра	18.02.2021	виконано
5.	Підбір та вивчення літературних та інших джерел з теми дослідження	25.02.2021	виконано
6.	Підготовка і подання науковому керівнику першого варіанту І розділу роботи	05.03.2021	виконано
7.	Підготовка і подання науковому керівнику першого варіанту ІІ розділу роботи	09.04.2021	виконано
8.	Підготовка і подання науковому керівнику першого варіанту ІІІ розділу роботи	07.05.2021	виконано
9.	Подання роботи у першому варіанті	11.05.2021	виконано
10.	Оформлення пояснювальної записки кваліфікаційної роботи бакалавра	12.05.2021	виконано
11.	Подання кваліфікаційної роботи бакалавра на попередній захист	**24.05.2021**	виконано
12.	Врахування зауважень керівника і подання роботи в остаточному варіанті (з відповідним висновком про допуск) на кафедру	28.05.2021	виконано
13.	Затвердження роботи в цілому (підготовка письмового відгуку керівника, письмова рецензія на бакалаврської роботу)	11.06.2021	виконано
14.	Захист кваліфікаційної роботи бакалавра	25.06.2021	виконано

Здобувач вищої освіти _____
(підпис)

Керівник _____
(підпис)

ВІДОМІСТЬ ДИПЛОМНОЇ РОБОТИ

Складові частини дипломної роботи	Обсяг, арк.
Титульний аркуш	1
Завдання до дипломної роботи	1
Календарний план дипломної роботи	1
Відомість дипломної роботи	1
Анотація	1
Анотація (іноземною мовою-англійською)	1
Зміст	1
Перелік скорочень, умовних позначень, термінів	2
Вступ	2
1	9
2	9
3	30
Висновки	1
Перелік використаних джерел	2
Додатки	5

				ДП XXXX 00.000.00		
	ПІБ	Підп.	Дата			
Розробн.	Касміна І.І.			Відомість дипломної роботи	Лист	Листів
Керівн.	Плескач В.Л.					
Н/контр.	Макаренко С.А.					
Зав.каф.	Плескач В.Л.					

АНОТАЦІЯ (РЕФЕРАТ)

Дипломна робота: 67 с., 29 рис., 2 таб., 18 джерел, 1 дод.

Метою дипломної роботи є ефективна організація та проведення тестування студентів за допомогою використання програмної системи.

Для досягнення мети роботи потрібно вирішити такі завдання:

– дослідити теоретичні основи побудови програмних систем тестування студентів;

– проаналізувати програмно-технічні рішення систем тестування студентів на основі сучасних веб-фреймворків;

– спроектувати, розробити та впровадити програмну систему тестування студентів.

Об’єкт дослідження.

Процеси тестування студентів за допомогою використання програмної системи на основі сучасних веб-фреймворків.

Предмет дослідження.

Програмно-технічні, організаційні засади, концептуальні підходи та принципи до побудови програмної системи тестування студентів.

Методи дослідження.

Теорія розподілених систем та порівняльний аналіз ефективності використання сучасних веб-фреймворків для розробки програмних систем тестування студентів, системний аналіз, теорія програмних систем та застосування клієнт-серверної архітектури для досліджуваної системи. Результат виконання дипломної роботи можна використати для розміщення на веб-хостингу, що дасть змогу отримати доступ користувачам до системи за допомогою посилання в мережі Інтернет. Для зберігання даних користувача (профіль, тести, результати тестування) була обрана система управління реляційними базами даних MSSQL. Веб-сервіс був розроблений мовою програмування C# з використанням платформи ASP.NET та сторінок Razor.

Ключові слова: веб-сервіс, тестування студентів, ASP.NET.

ANOTATION (ABSTRACT)

Thesis: 67 p., 29 pictures, 2 tables, 18 sources, 1 annex.

The purpose of the thesis is the effective organization and testing of students using a software system.

To achieve the goal of the work you need to solve the following tasks:

– investigate the theoretical foundations of designing software testing systems;

– analyze software and hardware solutions for student testing systems based on modern web frameworks;

– design, develop and implement a software system for testing students.

Object of study:

Processes of organizing and conducting student testing using a software system based on modern web frameworks.

Subject of study:

Software and technical, organizational principles, conceptual approaches to designing a software system for testing students.

Research methods:

Distributed systems theory and comparative analysis of the effectiveness of modern web frameworks for the development of software testing systems for students, systems analysis, software systems theory and the application of client-server architecture for the studied system. The result of the thesis can be used for hosting on a web host, which will allow users to access the system via a link on the Internet. The MSSQL relational database management system was chosen to store user data (profile, tests, test results). The web service was developed in the C # programming language using the ASP.NET platform and Razor pages.

Keywords: web-service, students testing, ASP.NET.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API (Application Programming Interface) – набір визначень протоколів взаємодії та засобів для розробки програмного забезпечення.

ASP.NET (Active Server Pages для .NET) – платформа від компанії Майкрософт для розробки веб-застосунків, яка містить: веб-сервіси, програмну інфраструктуру, модель програмування.

BEA (The Bureau of Economic Analysis) – бюро економічного аналізу.

CRM (Customer Relationship Management) - управління відносинами з клієнтами.

CORBA (Common Object Request Broker Architecture) – загальна архітектура посередника об'єктних запитів.

CSS (Cascading Style Sheets) – мова стилю, що використовується для опису зовнішнього вигляду веб-сторінок.

DCOM (Distributed Component Object Model) – розширення моделі компонентного об'єкта для підтримки зв'язку між об'єктами на різних комп'ютерах по мережі.

HTML (HyperText Markup Language) – мова тегів, якою розробляються гіпертекстові документи для глобальної мережі Інтернет.

HTTP (Hyper Text Transfer Protocol) – протокол передачі даних, який використовується в комп'ютерних мережах.

IBM (International Business Machines) – онад з найбільших в світі компаній виробників і постачальників апаратного і програмного забезпечення, а також IT-сервісів і консалтингових послуг.

IDE (Integrated development environment) – інтегроване середовище розробки.

JSON (JavaScript Object Notation) – текстовий формат обміну даними між комп'ютерами.

MVC (Model-view-controller) – архітектурний шаблон, який використовується для проектування та розробки програмного забезпечення.

MSSQL (Microsoft structured query language Server) – система управління базами даних, розроблена корпорацією Microsoft.

RMI (Remote Method Invocation) – програмний інтерфейс виклику віддалених методів в мові Java.

RPC (Remote procedure call) – віддалений виклик процедур.

SAML (Security assertion markup language) – мова розмітки декларації безпеки.

SDK (Software Development Kit) – набір засобів розробки, який дозволяє створювати прикладні програми за визначеною технологією або для певної платформи.

SQL (Structured query language) – мова структурованих запитів

UDDI (Universal Description Discovery & Integration) – інструмент для розташування описів веб-сервісів (WSDL) для подальшої їх інтеграції в свої системи іншими організаціями.

URL (Uniform Resource Locator) – уніфікований покажчик ресурсу.

WSDL (Web Services Description Language) – мова опису і доступу до веб-сервісів, заснована на мові XML.

XML (Extensible Markup Language) – розширювана мова розмітки.

YAML (Yet Another Markup Language)–«дружній» формат серіалізації даних, концептуально близький до мов розмітки, але орієнтований на зручність введення-виведення типових структур даних багатьох мов програмування.

БД – база даних.

ПЗ – програмне забезпечення.

МП – мова програмування.

СУБД – система управління базами даних.

ВСТУП

Нині тестові завдання є надзвичайно поширеними, оскільки дозволяють швидко визначити певний рівень знань студентів, прискорити обробку отриманої інформації та охопити великі за обсягом масиви вивченого матеріалу. Сервіси для викладачів та менторів дають можливість створювати завдання й тести, ділитися зі слухачами фото, відео і аудіоматеріалами, графічними зображеннями, здійснювати спільне редагування файлів.

Тестовий зріз знань застосовують під час онлайн навчання, оскільки це дозволяє автоматично збирати та отримувати статистичні дані з успішності студентів за результатами пройденого навчання.

Зважаючи на те, який вид контролю знань планується, варто використовувати різні типи тестових завдань та відповідей. Наприклад, поточний контроль можна провести за допомогою тестів із одним варіантом відповіді або декількома. При підготовці підсумкового чи тематичного контролю, краще поєднати різні види, скориставшись завданням з варіантами відповідей, завданням на співвіднесення понять, а також такі, що потребують розгорнутої відповіді. Це дозволить здійснити контроль знань студентів на різних рівнях.

Тому створення програмної системи ефективної організації тестування студентів є **актуальним науково-прикладним завданням.**

Метою дипломної роботи є ефективна організація та проведення тестування студентів за допомогою використання програмної системи.

Для досягнення мети роботи потрібно вирішити наступні **завдання**:

– дослідити теоретичні основи побудови програмних систем тестування за допомогою сучасних веб-фреймворків;

– проаналізувати програмно-технічні рішення систем тестування студентів на основі сучасних веб-фреймворків;

– спроектувати, розробити та впровадити програмну систему тестування студентів.

Об’єктом дослідження кваліфікаційної роботи є процеси організації та проведення тестування студентів за допомогою використання програмної системи на основі сучасних веб-фреймворків.

Предметом дослідження є програмно-технічні, організаційні засади, концептуальні підходи та принципи до побудови програмної системи тестування студентів.

Методами дослідження є теорія розподілених систем та порівняльний аналіз ефективності використання сучасних веб-фреймворків для розробки програмних систем тестування студентів, системний аналіз, теорія програмних систем та застосування клієнт-серверної архітектури для досліджуваної системи. Результат виконання кваліфікаційної роботи можна використати для розміщення на веб-хостингу, що дасть змогу отримати доступ користувачам до системи за допомогою посилання в мережі Інтернет. Для зберігання даних користувача (профіль, тести, результати тестування) була обрана система управління реляційними базами даних MSSQL. Веб-сервіс був розроблений мовою програмування C# з використанням платформи ASP.NET та сторінок Razor.

1 ТЕОРЕТИЧНІ ОСНОВИ ПОБУДОВИ ПРОГРАМНОЇ СИСТЕМИ ТЕСТУВАННЯ СТУДЕНТІВ

1.1 Поняття та зміст веб-сервісу в глобальній мережі Інтернет

Веб-сервіс – це програмна система, яка ідентифікується унікальною веб-адресою (URL-адресою) зі стандартизованими інтерфейсами, а також HTML-документ сайту, який відображається браузером користувача.

Вперше термін "веб-сервіс" був використаний в 2000 році при презентації нової технології Microsoft.NET. Згодом веб-сервіси перебували на стадії інтенсивного розвитку. Найбільші виробники, такі як Sun, Microsoft, IBM й BEA, об’єдналися для розроблення стандартів, оскільки були зацікавлені у їхньому швидкому прийнятті, розробляють і випускають інструментальне програмне забезпечення (ПЗ), що дозволяє створювати веб-сервіси.

Зазвичай веб-сервісами називають послуги, що надаються в глобальній мережі Інтернет. У цьому вживанні термін вимагає уточнення, чи йде мова про веб-пошту, пошук, зберігання файлів, документів, закладок тощо. Такі веб-сервіси можна використовувати незалежно від комп'ютера, браузера або точки доступу в Інтернет.

Переваги використання веб-сервісів

– веб-сервіси забезпечують незалежну від платформи взаємодію програмних систем. Наприклад, Windows-C#-клієнт може обмінюватися даними з Java-сервером, що працює під Linux;

– веб-сервіси засновані на базі відкритих протоколів і стандартів. Простота розробки і налагодження веб-сервісів досягається завдяки використанню XML;

– використання інтернет-протоколу забезпечується за допомогою HTTP-взаємодії програмних систем через міжмережевий екран. Це визначальна перевага у порівнянні з такими технологіями, як DCOM, CORBA

або Java RMI. З іншого боку, веб-сервіси не прив'язані жорстко до HTTP. Можуть використовуватися і інші протоколи.

Недоліки:

– більший розмір мережевого трафіку та менша продуктивність в порівнянні з технологіями RMI, CORBA, DCOM шляхом використання текстових XML-повідомлень. Однак на деяких веб-серверах є можливість налаштування стиснення мережевого трафіку;

– відповідальні веб-сервіси повинні використовувати кодування або вимагати аутентифікації користувача. Чи достатньо в цьому випадку застосування HTTPS, або кращі такі рішення, як SAML, XML Encryption або XML Signature – має бути вирішено розробником.

Веб-сервіси мають змогу взаємодіяти між собою і зі сторонніми застосунками за допомогою повідомлень, які базуються на певних протоколах (XML-RPC, SOAP) і угодах (REST). Веб-сервіс є одиницею модульності при використанні сервіс-орієнтованої архітектури застосунку.

Архітектура REST дає змогу постачальникам API доставляти дані в різних форматах, наприклад у вигляді простого тексту, HTML, XML, YAML і JSON, що є однією з його найбільших переваг. Завдяки зростаючій популярності REST, легкий і зрозумілий для людини формат JSON також швидко отримав популярність, оскільки він відмінно підходить для швидкого і зрозумілого обміну даними.

JSON (JavaScript Object Notation) – це простий для аналізу і легкий формат призначений для обміну даними. Незважаючи на свою назву, його можна використовувати з будь-якою мовою програмування, а не тільки з JavaScript – JSON повністю незалежний від мови. Його синтаксис є підмножиною 3-го видання стандарту ECMA-262. Файли JSON складаються з наборів пар ім'я/значення і упорядкованих списків значень, які є універсальними структурами даних, що використовуються більшістю мов програмування. Тому JSON може бути легко інтегрований з будь-якою мовою.

Фізично веб-сервіс є фрагментом програмного забезпечення, що називається "агентом". Агент має змогу приймати й передавати повідомлення, а також реалізує функціональність сервісу. Існує різниця між сервісом та агентом – сервіс може бути забезпечений різними агентами.

Процес обміну повідомленнями визначається в описі веб-сервісів (Web Services Description), що є специфікацією інтерфейсу сервісу й охоплює типи даних, формати повідомлень, способи серіалізації, транспортні протоколи, що використовуються при обміні між постачальниками послуг та агентами замовника. Крім того, опис сервісу містить вказівку на одну або декілька точок у мережі, звідки доступний постачальник.

Інструмент Universal Description, Discovery and Integration (UDDI) дозволяє ведення реєстру веб-сервісів. Налаштувавши з’єднання до цього реєстру, споживач має змогу знайти веб-сервіси, які задовольняють його потребам. Механізм UDDI надає можливість пошуку й публікації необхідного сервісу, як людиною, так і програмою-клієнтом.

Концептуальний підхід до розробки веб-сервісів передбачає, що окремі веб-сервіси мають певну обмежену функціональність. Вирішення складних завдань потребує використання функціональності декількох сервісів. В ході розвитку архітектури веб-сервісів це стало причиною виникнення поняття "композиція веб-сервісів" (Web-services composition) і "потік веб-сервісів" (Web-services flow). Нині замість цих понять використовують "оркестровка" (Web Service Orchestration) і "хореографія" (Web Service Choreography) веб-сервісів. Зазначені поняття відображають взаємодію сервісів і послідовність їхнього виконання. Тобто, застосунки, побудовані з використанням веб-сервісів, розглядають як застосунки, засновані на потоках робіт (Workflow-based applications, рис 1.1).

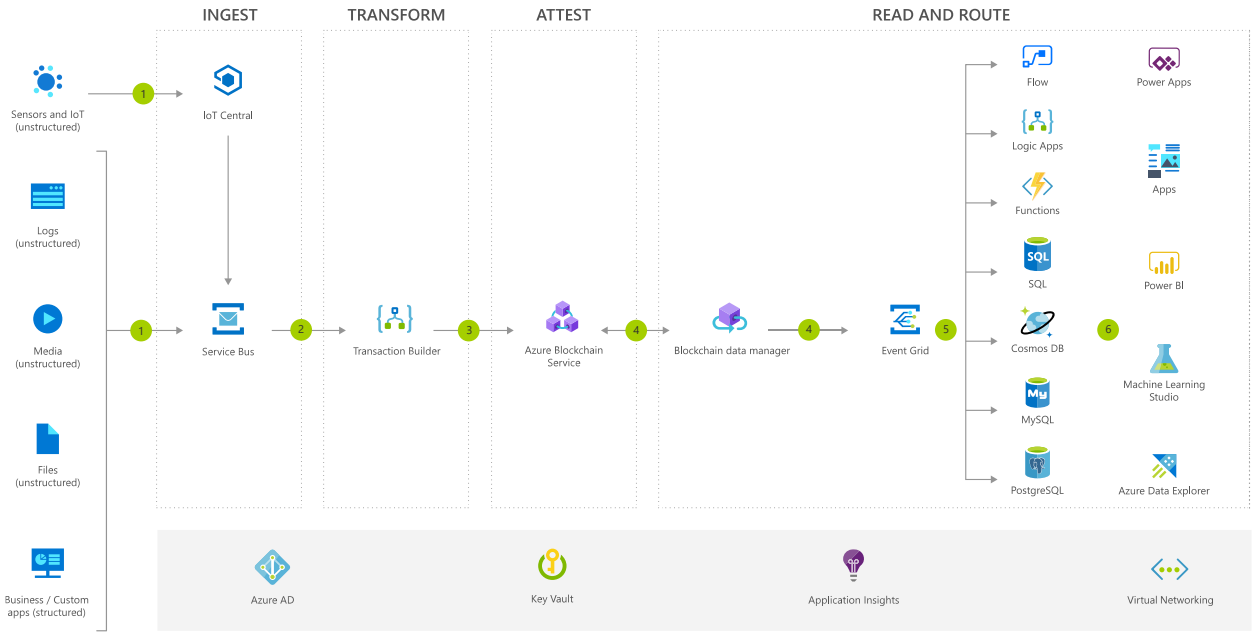

Рисунок 1.1 – Архітектура сервісу на основі потоку даних

1.2 Інформаційні технології побудови веб-сервісів

Веб-сервіси базуються на застосуванні відкритих, затверджуваних консорціумом W3C, протоколах і стандартах, основними з яких є наступні:

– UDDI (Universal Description, Discovery and Integration) – універсальний опис, виявлення та інтеграції веб-сервісів (протокол пошуку сервісів в глобальній мережі Інтернет);

– SOAP (Simple Object Access Protocol) – протокол доступу до простих об'єктів. Інструмент, за допомого якого відбувається передача інформації між інтернет-протоколами та вилученими об'єктами на базі протоколу HTTP;

– WSDL (Web Services Description Language) – мова опису веб-сервісів.

Мета технології веб-сервісів полягає в тому, щоб обійти обмеженість сучасних інформаційних технологій, відмовившись від фіксованих зв'язків між застосунками й компонентами застосунків, замінивши її слабкими більш гнучкими зв'язками. В основі веб-сервісів лежить модель зі слабкими зв'язками, яка не залежить від того, чи використовується вона для з'єднання компонентів в межах організації або для організації комбінованого сервісу, що застосовує функції від багатьох компаній.

Приймаючи рішення щодо вибору основної технології, потрібно враховувати цілий ряд факторів. Підставою використання технології веб-сервісів є вимоги відкритості, масштабованості та можливості доступу ззовні будь-якому клієнтові. Фактором впливу на ефективність розробки також є вибір інструментальних засобів і методології розробки.

Важливу роль у розробці програмних систем має створення архітектури системи як проектування на найвищому рівні. Провідні IT-компаній й аналітики вважають перспективними та важливими напрямками в розвитку інформаційних систем і програмного забезпечення архітектури, орієнтовані на сервіси (Service Oriented Architecture – SOA). До того ж основна увага спрямована на SOA, що орієнтована на глобальну мережу Інтернет та Інтранет, а саме на архітектуру веб-сервісів (Web Services Architecture – WSA).

Орієнтована на сервіси архітектура (SOA), має наступні характерні риси:

– розподілена архітектура. Функціональні модулі (системи) здатні до взаємодії з використанням локальних або глобальних мереж і можуть бути розподілені по безлічі обчислювальних систем;

– можливе підключення і динамічний пошук необхідних функціональних модулів;

– інтерфейс функціональних модулів забезпечує використання модулів незалежно від платформи або технології, у межах якої вони реалізовані;

– загальноприйняті галузеві стандарти є основою архітектури.

Основу орієнтованої на сервіси архітектури, становить взаємодія трьох учасників (рис. 1.2):

– постачальника сервісу;

– споживача сервісу;

– реєстру сервісів.

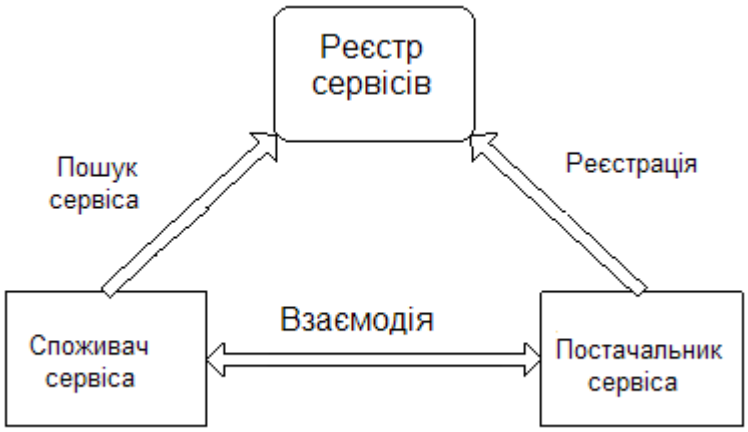

Рисунок 1.2 – Компоненти архітектури SOA

Взаємодії учасників мають наступні базові аспекти:

– пошук сервісу;

– публікація сервісу;

– підключення й використання.

Необхідні три типи угоди для реалізації SOA:

– транспортна угода: про протоколи та формати взаємодії;

– угода про метод виявлення сервісу;

– угода про опис функціональності сервісу, у придатному для автоматичної генерації коду вигляді, що встановлює процес взаємодії між клієнтом і постачальником сервісу.

Архітектура веб-сервісів є однією з реалізацій архітектурного шаблону SOA. Поняття орієнтованої на сервіси архітектури, склалося в ході розвитку концепції веб-сервісів. Водночас існують й інші підходи до реалізації SOA: DCOM (від Microsoft), CORBA (від консорціуму OMG), Java RMI (від Sun Microsystems), DCE (запропонований асоціацією Open Group).

1.3 Роль веб-сервісів у цифровій освіті

В нинішніх умовах дистанційного навчання постає необхідність використання сучасних інформаційних технологій для проведення лекційних занять та контролю знань.

Інформаційні технології вирішують певні завдання, даючи тим самим можливість сконцентруватися на концептуальних ідеях. Використання подібних технологій має широкі можливості. Студенти можуть заздалегідь переглядати навчальні плани, знайомитися зі змістом курсів. Завдяки цьому спілкування з викладачем може стати набагато змістовнішим. З іншого боку, студенти можуть заздалегідь прослухати лекції, вивчити необхідний матеріал, а потім вже підготовленими обмінюватися ідеями, обговорювати різні точки зору тощо. Як показують результати досліджень, такі перетворення в процесі навчання студентів призводять до підвищення якості отриманих знань.

Раціональним рішенням проблем комп'ютеризації освіти є впровадження в навчальний процес веб-сервісів.

В освіті веб-сервіс – це перш за все доступ до навчальних інструментів через браузер. Крім цього, з'являється ряд інших додаткових інструментів, що реалізують принцип веб-сервісів – це використання RSS і підкастингу в навчанні.

Таким чином, концепція веб-сервісів в навчанні передбачає наявність таких функціональних можливостей в освіті: rss-стрічки для підписки на новини, онлайнові лекції у вигляді підкастів, wiki-середовище для створення спільних проектів, застосування блогів викладачів і студентів для формування освітнього контенту при вивченні тих чи інших курсів.

В такому випадку концепція веб-сервісів надає можливість обміну знаннями не тільки від викладача до студентів, а й між студентами і від студентів до викладачів. Ця обставина формує високоякісне освітнє середовище, оскільки з'являється можливість при вивченні курсу задати питання не тільки викладачеві, але іншим експертам у досліджуваній області.

Технології веб-сервісів дозволяють значно підвищити доступність програм навчання, вводити каталогізацію і рейтингову оцінку і багато іншого для однієї простої мети – підвищення доступності та якості навчання. Навчальний курс може створюватися як компіляція навчального матеріалу різних авторів з каталогом курсів з зрозумілим і прозорим рейтингом, щоб у

студента була можливість з вибрати якийсь певний курс і зробити свій внесок в загальний рейтинг.

Значущість використання веб-сервісів в навчальному процесі обґрунтована такими положеннями:

– можливість організації якісного дистанційного навчання за рахунок використання різноманітних інформаційних технологій, що забезпечують інтерактивну взаємодію учнів з навчальним матеріалом;

– можливість використання нових інформаційних технологій, що не входять в перелік вільного програмного забезпечення, використовуваного в освітніх установах;

– організація групової та колективної роботи учасників навчального процесу по розробці проектів;

– поповнення банку загальнодоступних електронних освітніх ресурсів мережі Інтернет різних предметних областей, розроблених педагогічною спільнотою.

Звичайно, використання веб-сервісів на навчальних заняттях вимагає від викладача і студентів постійної роботи над підвищенням рівня своєї ІКТ-компетентності, творчого підходу до проектування занять. Проте подальші результати навчання того варті: грамотне використання веб-сервісів сприяє формуванню інтересу студентів до предмету, розвитку у них креативних здібностей і виховання активної життєвої позиції, підвищенню мотивації до самоосвіти, і в цілому, дозволяє ефективно організовувати навчальний процес.

Висновки до розділу

У цьому розділі було досліджено та проаналізовано сучасні підходи до розроблення і впровадження веб-сервісів та їх роль в цифровій освіті.

Стандартизованість – головна відмінність веб-сервісу від інших способів передачі даних. Прийнявши рішення використовувати веб-сервіси, можна відразу переходити до структури даних і доступним функціям. Існуючі веб-опис сервісів здійснюється в WSDL-документах, які розташовуються або

в спеціальних XML-сховищах, або на сервері застосунків. WSDL-документ може включати посилання на інші WSDL-документи і документи XSD (XML Schema), в яких описані типи даних, що використовуються веб-сервісами. Усередині WSDL-документа знаходиться адреса (URL) веб-сервісу. XML-сховища використовуються для управління WSDL-документами. Веб-сервіси описані і проіндексовані в бізнес-реєстрі, що містить адреси WSDL-документів.

Універсальність веб-сервісів зумовлена роботою на стандартних технологіях, що не залежать від постачальників застосунку і інших ресурсів мережі. Веб-сервіси можуть використовуватися в будь-яких операційних системах, серверах застосунків, мовах програмування. Програмні застосунки, написані на різних платформах і на різних мовах програмування, можуть використовувати веб-сервіси для обміну даними по комп'ютерних мережах, таких як Інтернет, аналогічно взаємодії між процесами на одному комп'ютері. Ця сумісність (наприклад, між застосунками Java і Python або Windows і Linux) обумовлена використанням відкритих стандартів.

Стратегічна цінність веб-сервісів полягає у скороченні часу на реалізацію проекту, збільшенні продуктивності, швидкій інтеграції бізнес-систем та їх застосувань.

2 АНАЛІЗ ПРОГРАМНО-ТЕХНІЧНИХ РІШЕНЬ СИСТЕМ ТЕСТУВАННЯ СТУДЕНТІВ НА ОСНОВІ СУЧАСНИХ ВЕБ-ФРЕЙМВОРКІВ

2.1 Аналіз наявних рішень програмних систем тестування студентів

В умовах змішаної і особливо дистанційної форм навчання постало питання використання онлайнових платформ і сервісів, які допомагають взаємодії викладача зі здобувачами освіти шляхом проходження тестів і вправ.

Для проведення аналізу наявних якісних систем були досліджені наступні сервіси для тестування студентів:

– Online Test Pad.

– Classroom.

– ClassMarker.

– Kahoot.

– Let`s test.

Online Test Pad. Безкоштовний багатофункціональний сервіс для проведення навчального процесу і тестування за допомогою мережі Інтернет. Зручний сайт для створення різноманітних навчальних матеріалів і типів завдань, структурування їх по папках. Інтерфейс представлений декількома мовами, серед яких – український.

Містить вбудований конструктор тестів з налаштуваннями типів питань і результатів, статистичних звітів і стилізації завдань. Формат тестових питань включає різні варіанти: одна або декілька правильних відповідей, відповідь у довільній формі, встановлення послідовності і відповідностей, заповнення пропусків тощо.

Надає можливість опублікувати завдання будь-якого типу для загального доступу на сайті. Розмістити те чи інше завдання на власному сайті викладач може за допомогою спеціального HTML-коду.

До недоліків можна віднести застарілий дизайн тестів; немає повного попереднього перегляду тесту; погана адаптація під різні девайси.

Classroom. Безкоштовна платформа від компанії Google для створення віртуальних класів, розробки та налаштування практичних завдань і тестів, самостійних і контрольних робіт і активного обміну завданнями з викладачем.

Інтерфейс доступний українською мовою. У сервісі можна коментувати роботи студентів і виставляти оцінки, архівувати проведені курси, публікувати оголошення, ділитися файлами з інших застосунків, мати доступ до матеріалів без підключення до глобальної мережі Інтернет. Викладач може спостерігати за процесом виконання завдань в режимі реального часу.

В Google Classroom не передбачена вебінарна кімната. У відкритій версії сервісу Google Classroom немає електронного журналу. Така можливість надається корпоративним користувачам Google Classroom. Для авторів, які мають особисті акаунти, існують обмеження: кількість учасників курсу не більше 250 і приєднатися до курсу протягом дня можуть тільки 100 користувачів.

ClassMarker. Англомовний сервіс для швидкого створення тестових завдань і опитувань з широким форматом відповідей. Викладач може створювати і редагувати свої тести, зберігаючи їх у банк завдань в своєму профайлі. Викладач створює свій клас за допомогою надсилання реєстраційних даних студентів на електронну пошту кожного користувача або відправки реєстраційного коду.

Безкоштовний варіант користування допускає створення не більше 100 тестів. За розширення функціонального пакету необхідно заплатити додаткові кошти.

Даний ресурс також дозволяє продавати свої тести онлайн з оплатою комісії в розмірі 10% від зазначеної суми.

Також серед недоліків англійська мова інтерфейсу, висока ціна, немає можливості вивантажити результати тесту.

Kahoot. Навчальна платформа, яка характеризується інтерактивністю, дозволяє проводити зрізи знань, опитування, тестування, а також виклад нового матеріалу з будь-якої дисципліни в ігровій формі.

Сайт містить багато готових тестів англійською мовою. Викладач може створювати власні завдання за допомогою шаблонів Kahoot! в форматі тесту. До питань можна додавати фото, малюнки, графіку і відео. Платформа дозволяє будь-яке опитування або контрольну організувати у вигляді змагання. Для цього на сайті присутній режим бонусів для студентів за швидкі відповіді.

Платформа включає платний тариф, який пропонує розширене використання функціональних можливостей сервісу.

До недоліків можна віднести обмежені типи питань; обмежені можливості опитування; дуже строгі параметри налаштування; немає підтримки для запитань і відповідей; немає місця для відкритих дискусій; заплутана панель інструментів і інтерфейс; переважна більшість його найкращих функцій приховано за платним доступом, а його заплутані схеми ціноутворення є величезними перешкодами для нових користувачів.

Let`s test. Сервіс спеціалізується виключно на онлайн-тестах. Дозволяє налаштувати доступ до тестування по паролю або попередньої реєстрації та контролювати, хто буде його проходити.

Конструктор містить всього шість типів питань, проте самостійно розібратися в функціоналі і інтерфейсі не так просто. Можна звернутися в службу підтримки і побачити презентацію роботи сервісу на прикладі свого ж тесту. Також немає можливості самостійно налаштувати дизайн, для брендування тесту своїм логотипом потрібно оформлювати заявку.

Сервіс платний. Є пробний період – 5 днів, а далі необхідно вибрати найбільш прийнятний тариф.

Є можливість імпортувати питання з файлу, відправляти сертифікати про проходження.

Сервіс найбільш підходить для організацій, яким важливо отримати власну, ізольовану систему тестування.

Серед недоліків застарілий дизайн конструктора, неінтуїтивний інтерфейс, немає можливості самостійно налаштувати дизайн тесту, немає функції попереднього перегляду.

2.2 Застосування стандартів, які використовують веб-сервіси

«Веб-сервіс» – це веб-застосунок, що надає ресурси в форматі, використовуваному іншими комп'ютерами. Веб-сервіси включають в себе різні типи API, в тому числі REST і SOAP API. Веб-сервіси – це, в основному, запити та відповіді між клієнтами і серверами (комп'ютер запитує ресурс, а веб-сервіс відповідає на запит).

Веб-сервіси можна розглядати як API, що використовують протокол HTTP для передачі запитів і відповідей. У разі веб-сервісів клієнт, який відправляє запит на ресурс, і сервер API, що надає відповідь, можуть використовувати будь-яку мову програмування або платформу. Неважливо яка платформа або мова програмування будуть використовуватися, оскільки запит повідомлення і відповідь надані через загальний веб-протокол HTTP.

REST (Representational State Transfer – передача репрезентативного стану) – це набір архітектурних обмежень, які утворюють надійні, ефективні та масштабовані розподілені системи. Система називається RESTful у випадку, коли дотримується цих обмежень.

Головна концепція REST полягає в тому, що ресурс, наприклад, документ, передається разом зі своїм станом та зв'язками (гіпертекстом) за допомогою чітко визначених, стандартизованих операцій та форматів. Часто API або сервіси визначаються як RESTful, коли прямо змінюють тип документа, як протилежність запуску команд в іншому місці.

Веб-протокол є частиною веб-сервісів: вони незалежні від мови і тому сумісні між різними платформами і системами. При документуванні REST API не має значення, чи розробляють інженери API за допомогою Java, Ruby,

Python або будь-якої іншої мови. Запити виконуються через HTTP, і відповіді повертаються через HTTP (рис 2.1).

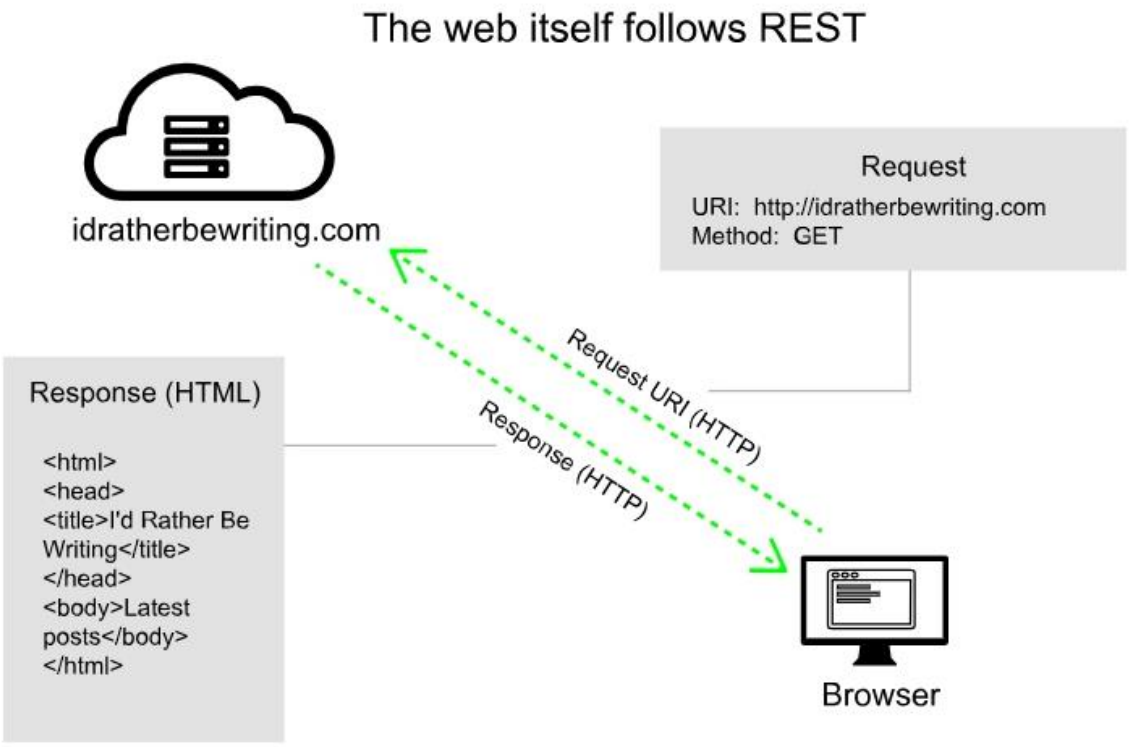

Рисунок 2.1 – Загальна модель REST API

Через те, що HTTP, стандартний протокол Web, також передає документи та гіпертекстові посилання, прості HTTP API іноді називають RESTful API, RESTful сервісами, або просто REST сервісами, хоча вони не обов'язково дотримуються усіх обмежень REST.

Іншою відмінною рисою REST є те, що API-інтерфейси REST фокусуються на ресурсах (тобто речах, а не дії) і способах доступу до ресурсів. Ресурси, як правило, є різними типами інформації. Ви отримуєте доступ до ресурсів через URL, так само як перехід за URL-адресою у вашому браузері дозволяє підключитися до інформаційного ресурсу. URL-адреси супроводжуються методом, який вказує, як ви хочете взаємодіяти з ресурсом.

Загальні методи включають GET (читання), POST (створення), PUT (оновлення) і DELETE (видалення). Кінцева точка зазвичай включає

параметри запиту, які визначають більш детальну інформацію про подання ресурсу, який потрібно побачити (рис. 2.2).

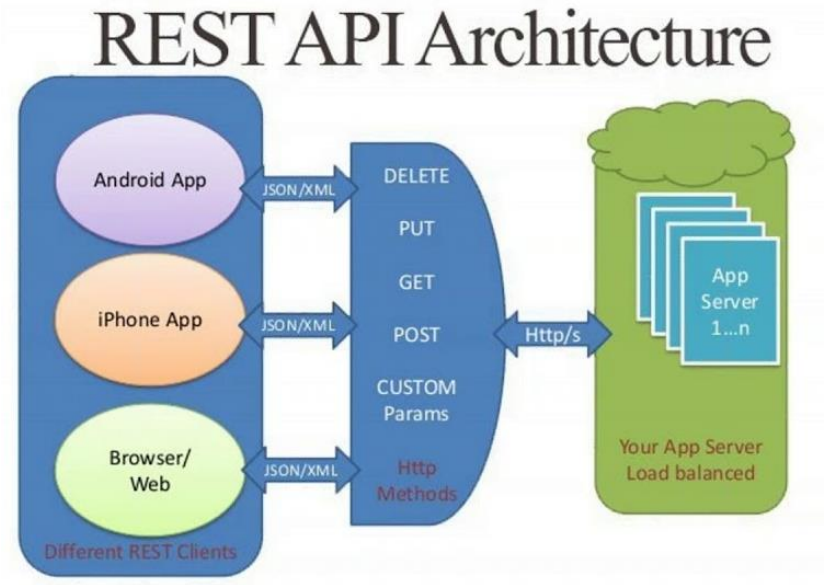

Рисунок 2.2 – REST API архітектура запитів на сервер

REST API не зберігає свої стани і можуть кешуватися. Відсутність стану означає, що кожного разу, коли відбувається звернення до ресурсу через кінцеву точку, API надає одну й ту саму відповідь. Останній запит не запам’ятовується і не враховується при наданні нової відповіді. Іншими словами – немає станів, які б раніше запам'яталися і API враховував при кожному запиті.

З метою підвищення продуктивності, відповіді можуть кешуватися. У випадку, якщо кеш браузера вже містить інформацію, запит до якої був здійснений раніше, браузер може просто повернути інформацію з кешу замість повторного звернення до серверу.

Принцип роботи кешування в REST API аналогічно кешуванню веб-сторінок. Браузер використовує значення часу останньої зміни в заголовках HTTP для того, щоб визначити, чи потрібно йому знову отримувати ресурс. Якщо вміст залишився без змін з моменту останнього вилучення, замість нього можна використовувати кешовану копію. Кешування прискорює швидкість відповіді.

Також до принципів розробки веб-сервісів на основі архітектури REST належить:

– Приведення архітектури до моделі клієнт-сервер. В основі даного обмеження лежить розмежування потреб. Необхідно відокремлювати потреби клієнтського інтерфейсу від потреб сервера, що зберігає дані. Дане обмеження підвищує переносимість клієнтського коду на інші платформи, а спрощення серверної частини покращує масштабованість системи. Саме розмежування на "клієнт" і "сервер" дозволяє їм розвиватися незалежно один від одного.

– Одноманітність інтерфейсу. До фундаментальних вимог REST архітектури відноситься і уніфікований, однаковий інтерфейс. Клієнт повинен завжди розуміти, в якому форматі і на які адреси йому потрібно надсилати запит, а сервер, в свою чергу, також повинен розуміти, в якому форматі йому слід відповідати на запити клієнта. Цей єдиний формат клієнт-серверної взаємодії, який описує, що, куди, в якому вигляді і як відсилати і є уніфікованим інтерфейсом.

– Рівні. Під рівнями мається на увазі ієрархічна структура мереж. Іноді клієнт може спілкуватися безпосередньо з сервером, а іноді – просто з проміжним вузлом. Застосування проміжних серверів здатне підвищити масштабованість за рахунок балансування навантаження і розподіленого кешування. Наведемо приклад. Уявімо собі деякий мобільний застосунок, який користується популярністю у всьому світі. Його невід'ємна частина– завантаження картинок. Так як користувачів – мільйони людей, один сервер не зміг би витримати такого великого навантаження. Розмежування системи на рівні вирішить цю проблему. Клієнт запросить картинку у підвузлів, проміжний вузол запросить картинку у сервера, який найменш завантажений в даний момент, і поверне картинку клієнту. Якщо тут на кожному рівні ієрархії правильно застосувати кешування, то можна домогтися гарної масштабованості системи.

У веб-сервісів, які дотримуються всіх перерахованих вище обмежень, є такі переваги:

– надійність (не потрібно зберігати інформацію про стан клієнта, який може бути втрачений);

– продуктивність (за рахунок використання кеша);

– прозорість системи взаємодії;

– масштабованість;

– простота інтерфейсів;

– легкість внесення змін;

– портативність компонентів;

– здатність еволюціонувати, пристосовуючись до нових вимог.

Висновки до розділу

У цьому розділі було проаналізовані та досліджені наявні рішення програмних систем для тестування систем, підходи та архітектурні рішення до проектування та розробки веб-сервісів. Веб-сервіси все активніше захоплюють області, зайняті об'єктними і компонентними технологіями, представляючи собою альтернативу традиційним підходам до створення корпоративних застосунків.

Поняття веб-сервісу означає зміну принципів роботи звичайних програмних продуктів. Повсюдне використання спеціальних інтерфейсів призводить до того, що звичні застосунки починають працювати через браузер. Це означає, що кожен бажаючий може працювати виключно з веб-застосунками.

Проаналізовано сучасні архітектурні підходи до побудови веб-сервісів. REST визначає правила архітектури для дизайну веб-сервісів, фокусується на систематичних ресурсах, включаючи якого формату стан ресурсів і передається по HTTP, і написаний різними мовами. Якщо порахувати за кількістю веб сервісів, які використовують його, REST став популярним за минулі роки як сервіс моделі дизайну з перевагою. Насправді, REST має

великий вплив і майже замінив SOAP і WSDL так як його набагато простіше і легше використовувати.

Застосування веб-сервісів в якості основної технології для створення корпоративних застосунків призвело до того, що базові протоколи і стандарти, що задовольняли вимогам розробників публічних веб-сервісів, виявилися недостатніми для вирішення завдань, що стоять перед корпоративними програмістами. В результаті з'явилися нові доповнення і розширення базових протоколів, орієнтованих на зростаючі потреби до використання веб-сервісів.

3 ПРОЕКТУВАННЯ, РОЗРОБЛЕННЯ, ВПРОВАДЖЕННЯ ПРОГРАМНОЇ СИСТЕМИ ТЕСТУВАННЯ СТУДЕНТІВ

3.1 Вибір програмного середовища та інженерія вимог до побудови програмної системи тестування

Як інструментарій розробки була використана Microsoft Visual Studio – лінійка продуктів компанії Microsoft, що включає інтегроване середовище розробки ПЗ і ряд інших інструментальних засобів. Інтегроване середовище розробки (IDE) є багатофункціональною програмою, яку можна використовувати для різних напрямків розробки програмного забезпечення. Крім стандартного редактора і відлагоджувача, які пропонують більшість середовищ IDE, Visual Studio включає в себе компілятори, графічні конструктори, засоби автозавершення коду і багато інших функцій для пришвидшення та спрощення процесу розробки.

Visual Studio дозволяє розробляти як консольні застосунки, так і ігри та програми з графічним інтерфейсом та підтримкою технології Windows Forms, а також веб-служби, веб-сайти, веб-застосунки в рідному, а також в керованому кодах для всіх платформ, які підтримують Windows, .NET Framework, Windows Mobile, Windows Phone .NET Compact Framework, Windows CE, Xbox і Silverlight.

Середовище розробки Visual Studio містить редактор вихідного коду з можливістю підтримки технології IntelliSense і найпростішого рефакторингу коду. Вбудований у застосунок відлагоджувач може працювати як на рівні вихідного коду, так і машинного рівня. Решта вбудованих інструментів включає в себе редактор форм для спрощення створення графічного інтерфейсу застосунку, дизайнер класів, веб-редактор і дизайнер схеми бази даних. Середовище розробки надає можливість створювати і підключати сторонні застосунки (плагіни) для розширення базової функціональності майже на кожному рівні, а саме: включення підтримки систем контролю версій вихідного коду, налаштування нових наборів інструментів (наприклад, для

проектування та редагування візуального коду на предметно-орієнтованих МП) або інструментів для інших аспектів процесу розробки та підтримки програмного забезпечення (наприклад, функції Team Explorer для роботи з Team Foundation Server).

Для спільних проектів Visual Studio надає підтримку групової роботи, дозволяючи виконувати спільне редагування і налагодження будь-якої частини коду в реальному часі, а в якості системи управління версіями використовувати Team Foundation або Git.

Visual Studio також передбачає комплекс інструментів для автоматизації тестування застосунків в частині перевірки роботи інтерфейсів, модульного і навантажувального тестування.

Створення і впровадження веб-сервісу для тестування студентів, впровадження повного функціоналу, внесення корегувань у функціонал сервісу та оптимальних рішень, його вдосконалення, а також додавання нових функцій для покращення роботи для користувача − є основним завданням дипломної роботи. Перевагою веб-сервісу є легкий та інтуїтивно зрозумілий інтерфейс, завдяки чому кожен користувач глобальної мережі Інтернет зможе використовувати даний веб-сервіс для швидкого проходження тестів, зручного та надійного зберігання даних, що дозволить легко проводити зріз знань студентів.

Для виконання поставленої мети кваліфікаційної роботи необхідно послідовно вирішити наступні задачі:

– проаналізувати наявні веб-сервіси, які реалізують функціонал проходження тестів для студентів;

– знайти недоліки в існуючих веб-сервісах та вирішити їх у власній програмній системі;

– виявити переваги в наявних веб-сервісах та відтворити їх в майбутньому функціоналі власного сервісу;

– створення дизайну веб-сайту зі зручним та зрозумілим інтерфейсом;

– виконати повну адаптацію для найбільш популярного браузера серед користувачів на всіх пристроях;

– адаптувати дизайн веб-сервісу для планшетів, мобільних телефонів та комп’ютерів;

– додати визначений функціонал в веб-застосунок;

– забезпечити надійну безпеку збереження даних в БД;

– перевірити коректність вхідної інформації та реалізувати підказки при помилковому введенні некоректних даних;

– виконати тестування функціоналу та адаптивності веб-сервісу.

Необхідно детально визначити перелік задач, для розв’язання яких розроблюється програмна система. Для цього, потрібно з’ясувати відомості, необхідні для розробки ефективної та масштабованої структури даних.

Перелік етапів зазначеного процесу:

– визначення цілей створення програмної системи;

– визначення обсягів і типів даних в системі;

– визначення методів використання даних в системі;

На етапі визначення цілей програмної системи потрібно враховувати, що робота програмної системи базується на процесах обміну даними. При реалізації програмної системи будемо використовувати базу даних, яка зберігає та накопичує дані про наявні тестові матеріали, результати проходження тестів та дані про користувачів веб-сервісу.

Отже основними цілями створення програмного продукту є:

– забезпечення можливістю ефективного і надійного зберігання даних;

– забезпечення легкої можливості значного розширення інформації в базі даних;

– створення веб-сервісу тестування студентів.

На етапі створення ефективної структури програмної системи, необхідно визначити обсяги і типи даних, які необхідні та від яких залежить продуктивність роботи з БД.

Внаслідок тенденції постійного зростання обсягів інформації, розмір завантажених даних зростатиме дуже швидко, що буде призводити до суттєвого збільшення навантаження та об’єму сховища даних. Саме тому слід передбачити таку властивість, як розширюваність місткості БД.

На третьому етапі створення ефективної структури програмної системи визначаються рівні доступу до функціоналу системи та даних. Адміністратор має можливість створювати тестові завдання та призначати їх на студентів. Студент має можливість переглядати інформацію про доступні та завершені тести та проходити призначені тести.

Для досягнення високих показників ефективності роботи веб-сервісу, а також для зменшення ймовірності виникнення помилок в системі, необхідним є ефективне налагодження системи безпеки.

Визначимо вимоги до візуального оформлення програмного продукту.

Інтерфейс користувача – це сукупність засобів, які допомагають користувачу взаємодіяти з комп’ютером (системою, веб-сервісом, програмним забезпеченням тощо).

Програмна система повинна мати простий, зручний та інтуїтивно-зрозумілий інтерфейс користувача. При розробці клієнтської частини системи (інтерфейсу користувача), необхідно пам’ятати про важливість деталей інтерфейсу та його простоту в користуванні, оскільки ці фактори є найбільш значущими безпосереднього для користувача системи. Важливо заздалегідь визначити, який тип інтерфейсу користувача можна розробити за допомогою використовуваних програмних технологій та інструментальних засобів.

Інтерфейс веб-застосунку повинен мати такі властивості:

– дружність – інтерфейс повинен бути простим та зрозумілим в користуванні та в повній мірі реалізованим функціонально аби відповідати запитам користувачів;

– адаптованість – інтерфейс повинен відповідати потребами користувачів системи, забезпечувати користувачів вказівками стосовно його функціональних можливостей, забезпечувати запобігання введення

некоректних даних та забезпечувати користувачів різними формами представлення даних залежно від типу запиту;

– гнучкість – це можливість легко адаптувати або додати новий інтерфейс для розв’язання різного роду задач.

Таким чином, користувацький інтерфейс має забезпечити можливості:

– створення облікового запису користувача;

– створення та налаштування тесту;

– створення та налаштування запитань до тесту;

– призначення тесту користувачам;

– перегляд результатів тестування;

– авторизації існуючого користувача;

– додавання, редагування, збереження та видалення даних в БД;

– виконання запитів до серверу і візуальне представлення даних в браузері в простому для розуміння користувачів вигляді;

– перегляд проміжних результатів роботи.

В результаті розробки програмного продукту портальний інтерфейс користувача має містити наступні сторінки:

– сторінка авторизації;

– головна сторінка;

– сторінка профілю користувача;

– сторінка призначених матеріалів;

– сторінка завершених матеріалів;

– сторінка каталогу;

– сторінка перегляду інформації про тест;

– сторінка проходження тесту;

– сторінка результату проходження тесту.

Інтерфейс CRM-частини застосунку має забезпечувати адміністрування процесу створення, налаштування та перегляду результатів тесту і має містити сторінки:

- сторінка створення користувача
- сторінка створення та налаштування тесту;
- сторінка створення та налаштування питань до тесту;
- сторінка призначення тесту користувачам;
- сторінка перегляду результатів тестування студентів.

Для вирішення завдань проектованої системи були проаналізовані можливості сервісів для тестування студентів та виокремлено функціонал, який найкраще підходить для розроблювального веб-сервісу, при цьому не ускладнюючи роботу користувачеві та не зменшуючи продуктивність веб-сервісу в цілому (таблиця 2.1).

Таблиця 2.1 – Основні вимоги до функціональних можливостей системи

Назва функціоналу	Опис функціоналу
Створення користувача	Можливість створення нового користувача в системі за допомогою інтерфейсу CRM-системи.
Створення тесту	Можливість створення та налаштування нового тесту для проходження користувачем.
Створення запитань до тесту	Можливість створення та налаштування питань тесту.
Призначення тесту користувачам	Функція масового призначення тесту користувачам.
Перегляд результатів тестування	Можливість перегляду результатів проходження тесту за допомогою інтерфейсу CRM-системи.

Кінець таблиці 2.1

Авторизація	Для ідентифікація користувача, надання доступу до сторінок та використання функціоналу сервісу необхідно пройти авторизацію. Дані сесії зберігаються для легкого доступу користувачів до системи в майбутньому за допомогою кешу.
Налаштування профілю	Дана функція дозволяє переглядати обліковий запис користувача та вносити додаткові відомості.
Призначення тесту	Функція дозволяє призначити для проходження тест з каталогу.
Відміна призначення	Функція призначена для скасування проходження тесту.
Проходження тесту	Дана функція являє собою алгоритм вибору відповіді з запропонованих варіантів та підтвердження обраної відповіді.
Підтвердження результату	Після проходження тесту користувач підтверджує результат тестування або використовує другу спробу.
Перегляд призначених, завершених та доступних тестів	Функція виведення тестових матеріалів на відповідних сторінках.
Вихід з облікового запису	Функція надає можливість видалити дані про поточну сесію та перенаправляє користувача на сторінку авторизації.

В таблиці наведено перелік основних функціональних вимог до системи, який буде реалізовано в межах дипломного проекту. Дані функції визначають фундаментальний набір необхідних можливостей програмної системи для тестування студентів.

3.2 Проектування програмної системи тестування студентів

Для зрозумілого наочного зображення потоків даних та подання механізмів обробки та передачі інформації у програмній системі тестування студентів використовується діаграма потоків даних, яка ще на етапі системного аналізу проектованого ПЗ дозволяє отримати уявлення про послідовність проходження даних в системі (рис. 3.1).

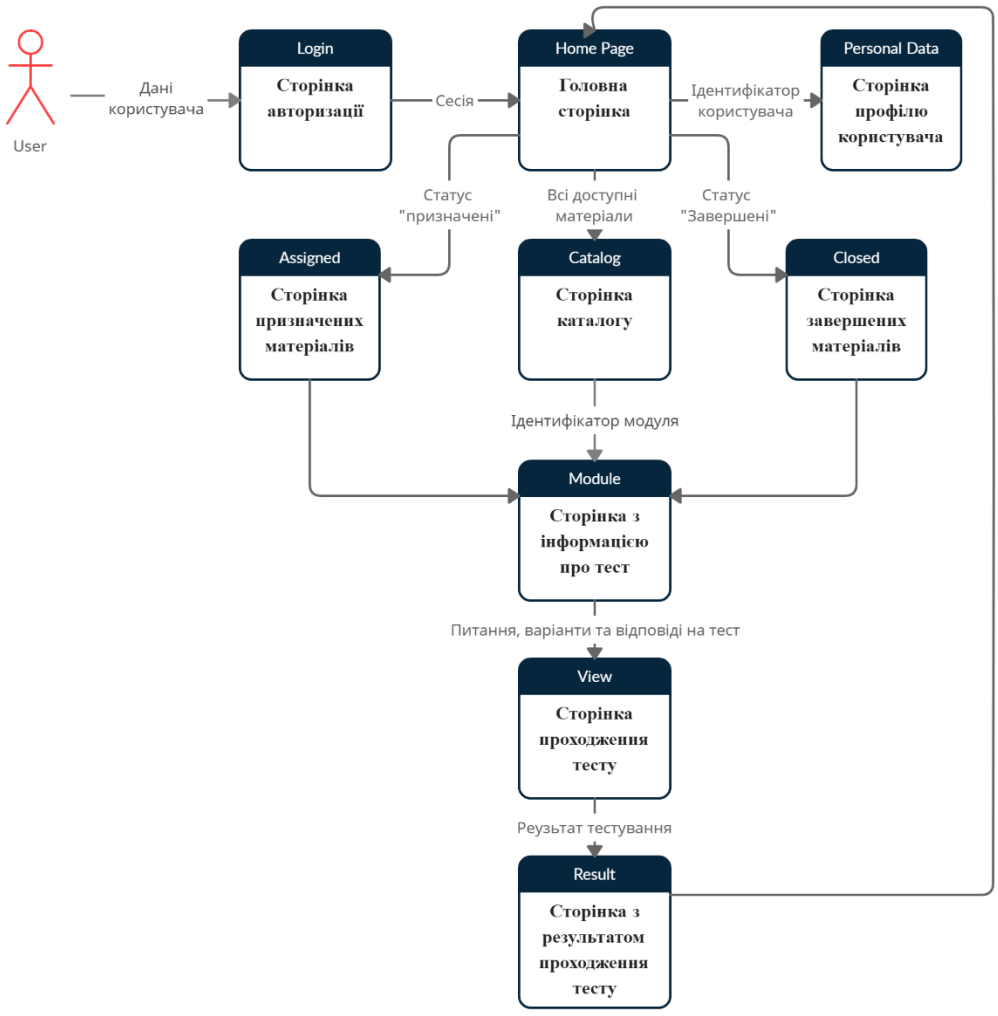

Рисунок 3.1 – Діаграма потоку даних програмної системи тестування студентів

Оскільки веб-сервіс пов’язаний з обробкою та зберіганням інформації, постає питання необхідності створення БД. Мінімальні вимоги до БД – наявність таблиць для зберігання відомостей про користувача і тести. Графічне відображення схеми бази даних надано на рисунку 3.2.

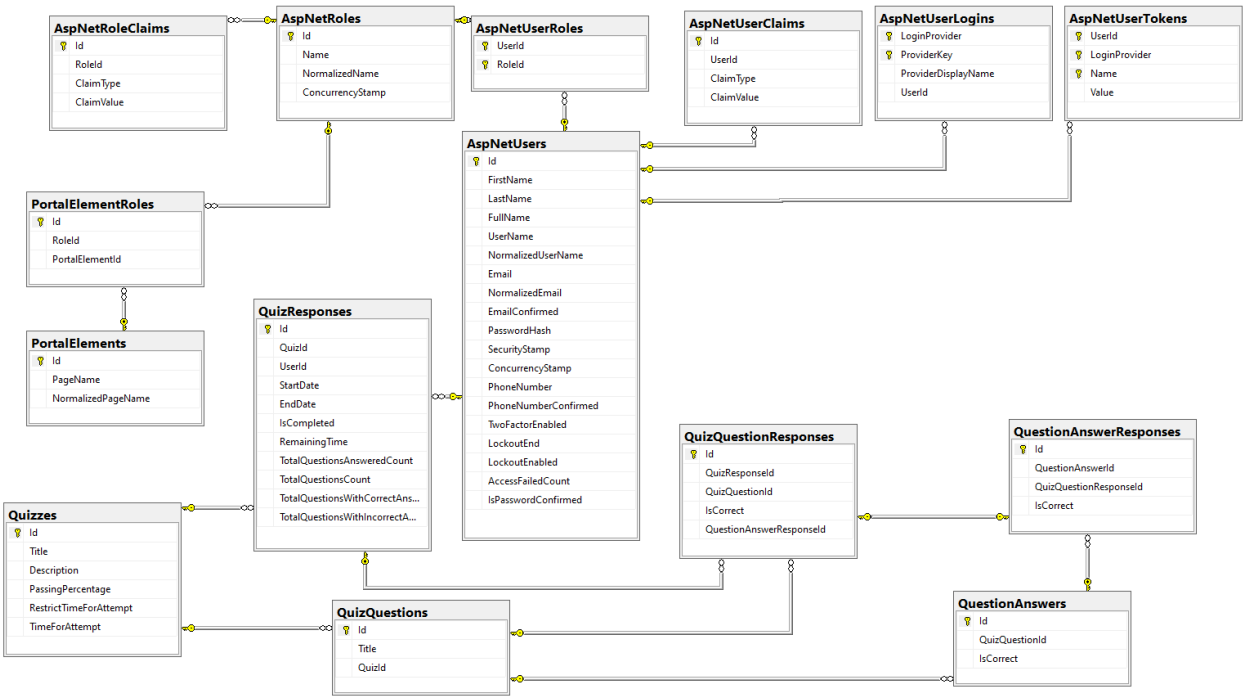

Рисунок 3.2 – Схема бази даних програмної системи тестування студентів

Основними системними елементами в проектній системі є C# компоненти, представлені файлами класів проектованої системи, та Razor компоненти.

Робота MVC застосунку керується головним чином C# контролерами, але безпосередньо користувач має доступ до застосунку у вигляді представлення, що формує зовнішній вигляд програми. Для цього використовуються компоненти Razor, які дозволяють розділяти інтерфейс користувача на незалежні частини або для декількох сторінок веб-сервісу використовувати вбудований елемент. Сторінки Razor зазвичай підкріплюються відповідним файлом класу .cs, який представляє модель для сторінки із властивостями та методами дій. Це дає змогу впроваджувати функціональні можливості C# всередині HTML розмітки для здійснення

певних операцій та динамічного наповнення веб-сторінки. Перелік основних компонентів надано в таблиці 3.1.

Таблиця 3.1 – основні компоненти веб-сервісу

Компонент	Опис
Account	Компонент, завдяки якому реалізовано авторизацію користувача в системі засобами стандартної форми введення даних логіну та пароля.
LoginPage	Компонент сторінки авторизації з можливістю переходу на сторінку відновлення паролю.
Index	Компонент Razor для відображення головної сторінки веб-застосунку, на якій розміщені поточні задачі користувача с можливістю фільтрації по заданим критеріям.
PersonalData	Компонент Razor для представлення сторінки з відображенням персональних даних користувача, з можливістю змінити мову інтерфейсу; додати, змінити або видалити інформація про освіту; переглянути викладачів (менторів), до яких прив’язаний користувач.
Assigned	Компонент Razor для представлення сторінки з модулями зі статусом «Призначені». Надає можливість відсортувати модулі за назвою, датою початку або завершення дії

Кінець таблиці 3.1

	модулю, кількістю днів, що залишилось до кінцевої дати, статусом або відсотком прогресу.
Completed	Компонент Razor для представлення сторінки з модулями зі статусом «Завершені». Надає можливість відсортувати модулі за назвою, статусом або відсотком прогресу.
Catalog	Компонент Razor для представлення сторінки зі всіма доступними модулями користувачу. Надає можливість фільтрації та пошуку модулів.
LearningModule	Компонент Razor для представлення сторінки з інформацією про модуль
View	Компонент Razor для представлення сторінки з виведенням запитань та можливістю відповіді на них.
Result	Компонент Razor для представлення сторінки виведення результату проходження тесту.

Наведені системні компоненти є основними компонентами, з яких складається веб-сервіс тестування студентів. Крім цього, для обробки та здійснення запитів на ресурси використовуються додаткові сервіси, які надають змогу отримувати дані авторизованих користувачів та захищати основні компоненти від несанкціонованого доступу до ресурсів веб-сервісу.

Проект визначає архітектуру, за якої основною частиною є веб-сервер. Він здійснює керування системою, її розгортання для відображення клієнту,

отримує та формує запити між сервісами. Кожна частина структури взаємопов'язані між собою і працюють залежно від стану запиту.

Для зберігання даних сесії та авторизації користувачів використовується повнофункціональний постачальник аутентифікації для створення і обслуговування імен входу Core Identity (рис. 3.3). Додатково дана система дозволяє користувачам створювати облікові записи або використовувати для входу на сайт акаунти зовнішніх провайдерів, таких як Microsoft, Google, Facebook, Twitter та інших.

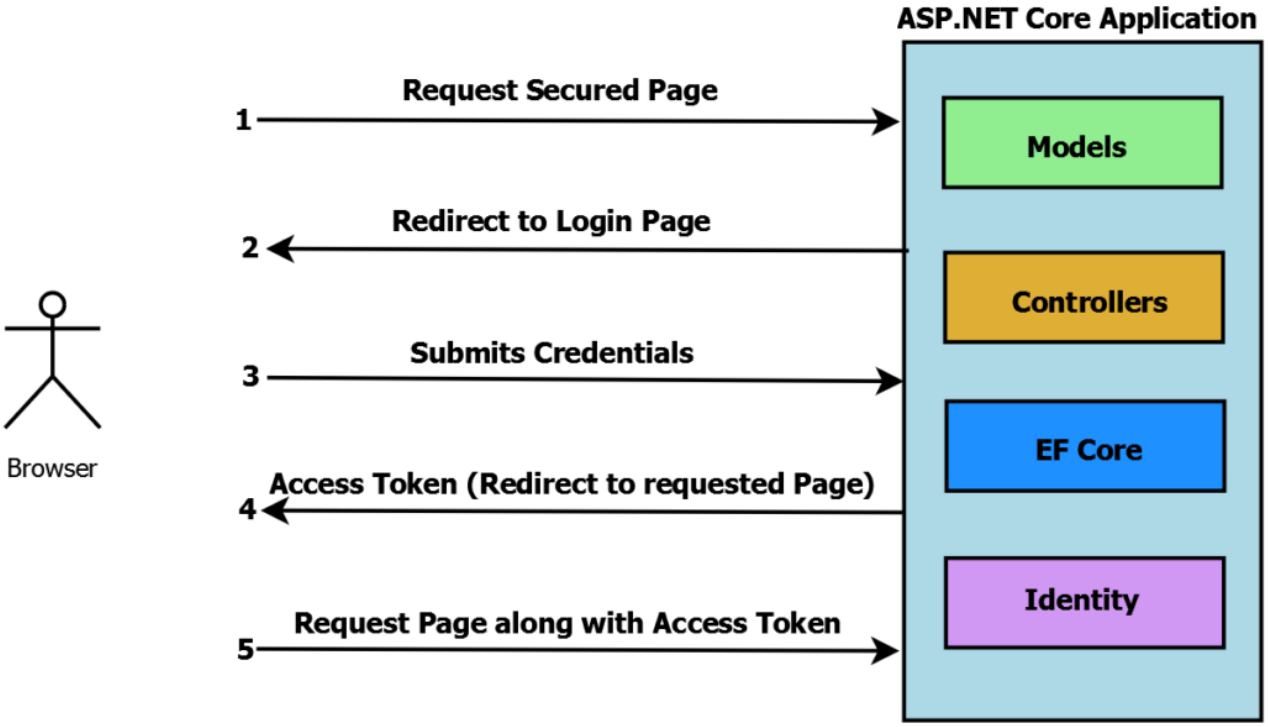

Рисунок 3.3 – Схема перевірки автентичності та авторизації за допомогою ASP.NET Core Identity

Оскільки система, яка розробляється, є веб-застосунком, вона встановлює мінімальні вимоги до програмного забезпечення користувача. Для повноцінного користування веб-сервісом досить мати сучасний мобільний пристрій або персональний комп’ютер з можливістю підключення до мережі Інтернет та браузер, за допомогою якого буде встановлюватися зв'язок клієнт-сервер та буде наданий доступ до веб-сервісу.

3.3 Використане програмне забезпечення веб-сервісу для тестування студентів та інструкція користувача

Для реалізації проектної системи, було використано CRM-систему та технології створення веб-сайтів: ASP.NET Core MVC, Razor Pages, JavaScript, HTML, CSS, Bootstrap та систему управління реляційними базами даних (СУБД) MSSQL. Зазначені технології дозволяють виконати поставленні вимоги технічного завдання. Завдяки детальним відомостям про використані технології буде визначено з якою метою вони були залучені.

Сучасна CRM призначена для вивчення ринку і конкретних потреб клієнтів. На базі отриманих знань розробляються нові товари або послуги завдяки чому компанія досягає поставлених цілей і поліпшує свій фінансовий показник.

Використовуються три CRM-підходи, кожен з яких може бути реалізований окремо від інших:

– співробітницький – взаємодія зі споживачами відбувається без участі персоналу з роботи з клієнтами;

– оперативний – автоматизація споживчих бізнес-процесів, що забезпечує допомогу персоналу в роботі з клієнтами здійснювати свої функції;

– аналітичний – аналіз отриманої інформації про споживачів із різними цілями.

Microsoft Dynamics CRM – це пакет ПЗ для управління взаємовідносинами з клієнтами, який був розроблений компанією Microsoft. CRM орієнтована на організацію маркетингу, продажів та надання послуг (служби підтримки).

Microsoft Dynamics CRM є клієнт-серверним застосунком. Це веб-застосунок на основі набору серверів для декількох служб глобальної мережі Інтернет (IIS), який підтримує множинні інтерфейси веб-сервісів. Отримати доступ до CRM клієнти можуть через браузер, планшети та мобільні пристрої або через клієнтський плагін до Microsoft Outlook. Крім Internet Explorer,

Microsoft Dynamics CRM наразі підтримує браузери Chrome та Firefox. Також працює в браузері Opera. Пакет можна використовувати, як розширену платформу для взаємодії з клієнтами, і може налаштовуватися відповідно до визначених цілей за допомогою програмної платформи .NET.

Багато компаній вважають, що перехід до хмарних технологій стане занадто витратним і небезпечним, але у перенесенні CRM-процесів в хмару більше переваг, ніж недоліків.

До недоліків слід віднести:

– залежність від хостера в питаннях виправлення та оновлення ПЗ;

– непередбачені витрати, що збільшуються разом з підприємством;

– складнощі при переміщенні в хмару даних з розрізнених систем.

Переваги:

– виправлення та оновлення – обов'язки хостера;

– не потрібно купувати дороге обладнання на початку створення системи;

– IT-відділ менше зайнятий рутинним обслуговуванням і ремонтами;

– дані компанії і клієнтів поза корпоративної інфраструктури є менш уразливими при локальних інцидентах;

– інфраструктура масштабується по мірі виходу компанії на нові ринки.

Ціла команда може в режимі реального часу швидко використовувати і переглядати інформацію про клієнтів і замовників, а також ділитися нею за допомогою різних каналів.

В Microsoft Dynamics CRM реалізована модель безпеки, яка контролює цілісність та забезпечує збереження даних, а також доступ до даних в межах командної та спільної роботи. Основними цілями моделі є забезпечення доступу до відповідних рівнів інформації, розподіл категорій типів користувачів по їх ролі, підтримка моделі ліцензування для користувачів, диференціація рівнів доступу та підтримка обміну даними для спільної роботи з ними. Управління забезпечення безпеки здійснюється на основі ролей та об'єктів.

Dynamics CRM є програмним продуктом, який оперує метаданими. Метадані створюють опис структури даних, згідно з яким застосунок використовує та відображає їх. Рівень метаданих підсумовує необхідні деталі з накопичувача даних, такі як схеми та доступи до записів. У поточній версії використовується новий інтерфейс програмування застосунків (API), який дозволяє додавати або оновлювати метадані.

Razor Pages – це інфраструктура, орієнтована на створення сторінок динамічних веб-сайтів, керованих даними, з чітким розділенням задач. На основі поточної версії ASP.NET від Microsoft, Razor Pages підтримує крос-платформну розробку і може бути розгорнута в операційних системах Windows, Unix та Mac.

Платформа Razor Pages легка і дуже гнучка. Вона надає розробнику повний контроль над HTML-розміткою сторінки. Фреймворк вбудований поверх ASP.NET Core MVC і встановлюється за замовчуванням, коли в застосунку .NET Core реалізується шаблон MVC.

Razor Pages – рекомендована основа для генерації HTML на стороні сервера на платформі .NET Core. Для роботи зі сторінками Razor не потрібно мати ніяких знань або розуміння MVC.

Razor Pages використовує популярну мову програмування C# для розробки на сервері, а також простий у вивченні синтаксис шаблону Razor для вбудовування C# в HTML-розмітку для динамічного генерування вмісту для браузерів.

Bootstrap – це набір безкоштовних інструментів з відкритим кодом, який спрощує розробку динамічних веб-сайтів, призначений для стилізації компонентів інтерфейсу веб-сайтів та веб-застосунків (форм, кнопок, навігації та інших), який містить шаблони CSS та HTML, а також додаткові розширення JavaScript. Bootstrap – це клієнтський фреймворк, тобто інтерфейс для користувача, на відміну від коду серверної сторони, який знаходиться на сервері.

Bootstrap має модульну структуру і складається в більшості з наборів таблиць стилів LESS, які реалізують компоненти цього набору інструментів. Розробник може власноруч налаштовувати файли Bootstrap, визначаючи компоненти для свого проекту.

Для виконання власного стилізування та обробки подій на HTML сторінках використовуються JavaScript та каскадні таблиці стилів (CSS) – це технологія опису зовнішнього вигляду документа, написаного мовою гіпертекстової розмітки HTML.

JavaScript – динамічна прототипна мова програмування. Вона не надає низькорівневий доступ до пам'яті або процесору, тому що від початку була створена для браузерів, які цього не потребують.

Можливості JavaScript значно залежать від оточення, в якому він працює. Наприклад, Node.JS підтримує функції виконання мережевих запитів, читання / запису довільних файлів тощо.

У браузері для JavaScript є все, що пов'язано з взаємодією з користувачем і веб-сервером та маніпулюванням веб-сторінками.

Наприклад, в браузері JavaScript може:

– реагувати на дії користувача, переміщення вказівника, клацання миші, натискання клавіш;

– додавати новий HTML-код на сторінку, модифікувати стилі, змінювати існуючий вміст;

– відправляти мережеві запити на віддалені сервера, завантажувати та отримувати файли за допомогою технологій AJAX і COMET;

– отримувати і встановлювати cookies, показувати повідомлення відвідувачеві, задавати питання;

– запам'ятовувати дані на стороні клієнта (local storage).

ECMAScript є стандартом мови JavaScript. Станом на 2012, всі сучасні браузери повністю підтримують ECMAScript 5.1. Більш ранні версії браузерів підтримують принаймні – ECMAScript 3. 2015 року відбувся випуск шостої версії ECMAScript. Ця версія офіційно називається ECMAScript 2015 року, яку

найчастіше називають ECMAScript 2015 або просто ES2015. З недавнього часу стандарти ECMAScript випускаються щорічно.

Для розробки на стороні серверу була використана платформа ASP.NET Core на основі шалону проектування MVC. Платформа представляє технологію, створену компанією Microsoft, призначену для розроблення різного роду веб-застосунків: від простих веб-сайтів до великих веб-порталів і веб-сервісів.

ASP.NET Core може працювати разом з крос-платформним середовищем .NET Core, яка може бути розгорнута на найбільш розповсюджених операційних системах: Windows, Linux, Mac OS.

За допомогою ASP.NET Core є можливість створювати крос-платформні застосунки. Хоча Windows як середовище розробки і розгортання програми досі має перевагу, але тепер вже немає обмеження тільки однією операційною системою.

Для розгортання веб-застосунку можна використовувати стандартний IIS, або Kestrel (крос-платформний веб-сервер).

Завдяки модульності фреймворка всі необхідні компоненти веб-сервісу можуть завантажуватися як окремі модулі через менеджер пакетів Nuget.

ASP.NET Core включає фреймворк MVC, який об'єднує функціональність MVC, Web Pages та Web API. У попередніх версіях платформи ці технології реалізувалися окремо і тому містили велику кількість дублюючої функціональності.

Крім об'єднання вищезазначених технологій в одну модель, в MVC було включено ряд додаткових функцій.

Серед таких функцій є допоміжні теги (tag helper), які дозволяють більш функціонально поєднувати синтаксис HTML з кодом C#.

Характеристикою ASP.NET Core є розширюваність. Фреймворк побудований з набору відносно незалежних компонентів. Також є можливість або використовувати вбудовану реалізацію компонентів, або за необхідності

розширити їх за допомогою механізму спадкування. Іншим шляхом є створення і застосовування свого компоненту з власним функціоналом.

Було спрощено управління посиланнями і конфігурацією проекту. Фреймворк тепер має свій контейнер для впровадження посилань, і більше немає необхідності використання сторонніх контейнерів. Хоча при бажанні їх також можна продовжувати застосовувати.

Підсумовуючи, можна виділити наступні основні відмінності ASP.NET Core від попередніх версій:

– можливість розгортати застосунок як на IIS, так і в межах власного процесу;

– новий легкий і модульний конвеєр HTTP-запитів;

– використання платформи .NET Core і її функціональності;

– інтегрована підтримка для створення та використання пакетів NuGet;

– поширення пакетів платформи через NuGet;

– конфігурація для спрощеного використання в хмарі;

– єдиний стек веб-розробки, що поєднує Web UI і Web API;

– кросплатформеність: можливість розробки і розгортання застосунків ASP.NET на Windows, Mac і Linux;

– вбудована підтримка для впровадження посилань;

– розвиток як Open Source, відкритість до змін.

Entity Framework – це рішення для взаємодії з базами даних, яке використовується в програмуванні на мовах сімейства .NET. Entity Framework дозволяє встановлювати взаємодію з СУБД за допомогою сутностей (entity), а не таблиць.

Безпосередньо працюючи з базами даних, розробник повинен турбуватися про підключення, підготовку, відправлення запитів SQL і параметрів транзакцій. За допомогою Entity Framework все це робиться автоматично – програміст працює безпосередньо з сутностями і тільки вказує EF, які зміни необхідно внести в БД.

Entity Framework є спеціальною об'єктно-орієнтованою технологією на базі .NET Framework для роботи з даними. У випадку з традиційними засобами ADO.NET є змога створювати підключення, команди та інші об'єкти для взаємодії з БД, то Entity Framework є більш високим рівенем абстракції, який дозволяє абстрагуватися від самої БД і працювати з необхідними даними незалежно від типу сховища. У разі якщо на фізичному рівні доводиться оперувати таблицями, первинними і зовнішніми ключами або індексами, то на концептуальному рівні, який пропонує Entity Framework, робота відбувається з об'єктами.

Базовою концепцією Entity Framework є поняття сутності (entity). Сутність передбачає набір даних, які є асоційованими з певним об'єктом. Тому технологія Entity Framework здійснює роботу не з таблицями, а з наборами об'єктів і самими об’єктами.

Будь-яка сутність, як і будь-який об'єкт з навколишнього середовища, має певні властивості. Якщо сутність, наприклад, описує людину, то можна виділити такі властивості, як прізвище, ім'я, вік, вага, зріст. Властивості необов'язково представляють прості дані типу int, але й можуть представляти більш комплексні структури даних. Кожна сутність може містити одну або декілька властивостей, які відрізняють цю сутність від інших і унікально визначають цю сутність. Такі властивості називають ключами.

Сутності можуть бути пов'язані асоціативним зв'язком один-до-одного, один-до-багатьох і багато-до-багатьох, подібно до того, як відбувається зв'язок через зовнішні ключі в звичайній БД.

Особливою рисою Entity Framework є використання запитів LINQ для вибірки даних з БД. LINQ надає можливість не тільки отримувати з БД певні рядки, що зберігають об'єкти, але й діставати об'єкти, пов'язані різними асоціативними зв'язками.

Entity Framework передбачає три основні підходи щодо взаємодії з базою даних:

– Code first: розробник створює клас та властивості моделі даних, які будуть зберігатися в БД, після чого Entity Framework за цією моделлю генерує базу даних і її таблиці.

– Model first: спочатку розробник створює модель бази даних, на основі якої Entity Framework генерує реальну базу даних на сервері;

– Database first: Entity Framework створює набір класів, які є відображенням моделі конкретної бази даних;

Для зберігання даних використана СУБД SQL Server, яка є однією з найбільш популярних систем управління базами даних. MS SQL підходить для різноманітних проектів: від невеликих застосунків до великих високонавантажених проектів.

SQL Server був розроблений компанією Microsoft. Перший зразок СУБД вийшов в 1987 році. А поточною версією є версія 2019, яка вийшла в 2019 році.

SQL Server довгий час був винятково СУБД для Windows, проте починаючи з версії 16 ця система доступна і на платформі Linux.

SQL Server визначає такі особливості:

– надійність і безпека − SQL Server надає шифрування даних;

– продуктивність − SQL Server працює дуже швидко;

– простота − з СУБД відносно легко працювати і вести адміністрування.

Як і в будь-якій СУБД, основним аспектом в MS SQL Server є база даних. База даних представляє сховище даних, організованих певним способом. Часто фізично база даних представляє собою файл на жорсткому диску, хоча така відповідність є необов'язковою. Для зберігання і адміністрування баз даних застосовуються СУБД (DBMS − database management system). І якраз MS SQL Server є однією з таких СУБД.

MS SQL Server використовує реляційну модель для організації баз даних. Ця модель БД була розроблена Едгаром Коддом ще в 1970 році. А нині вона фактично є стандартом для організації БД.

Реляційна модель визначає спосіб зберігання даних у вигляді таблиць, кожна з яких складається з рядків і стовпців. Кожен окремий об'єкт зберігається в окремому рядку, а в стовпчиках розташовані атрибути об'єкта.

Ідентифікації кожного рядка в межах таблиці відбувається за допомогою застосування первинного ключа (primary key). Первинним ключем може виступати один або декілька стовпців. Шляхом використання первинного ключа, є можливість посилатися на певний рядок в таблиці. Таким чином два рядки не можуть мати один і той самий первинний ключ.

Таблиці можуть бути пов’язані між собою через ключі, тобто між двома таблицями можуть бути встановлені асоціативні зв'язки, а сама таблиця може бути представлена у вигляді відносини ("relation").

Мова SQL (Structured Query Language) застосовується для взаємодії з БД. Клієнт (наприклад, зовнішня програма) відправляє запит мовою SQL за допомогою спеціального API. СУБД належним чином інтерпретує і виконує запит, а потім посилає клієнту результат виконання.

Розглянемо інтрукцію використання програмної систмеми тестування студентів. Для заведення нового користувача в системі, адміністратор має змогу за допомогою інтерфейсу CRM додати студента (рис. 3.4).

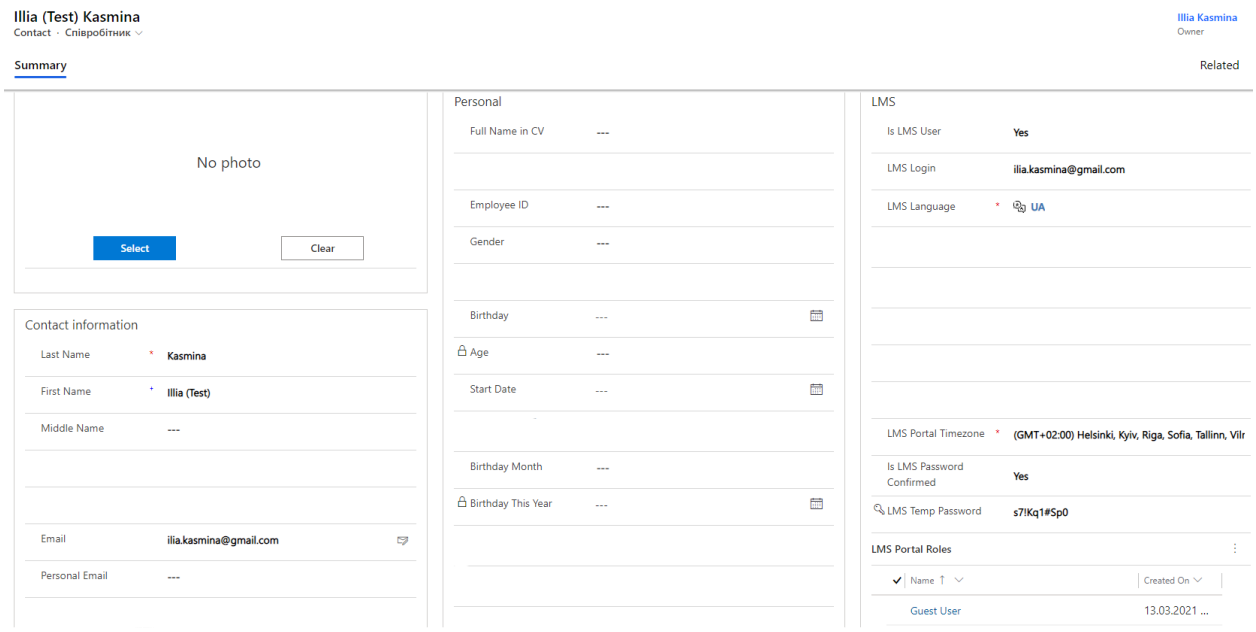

Рисунок 3.4 – Сторінка CRM-системи для створення нового користувача

Далі необхідно створити та налаштувати новий тест для розміщення на веб-сайті, для цього в CRM існує форма створення тесту (рис 3.5).

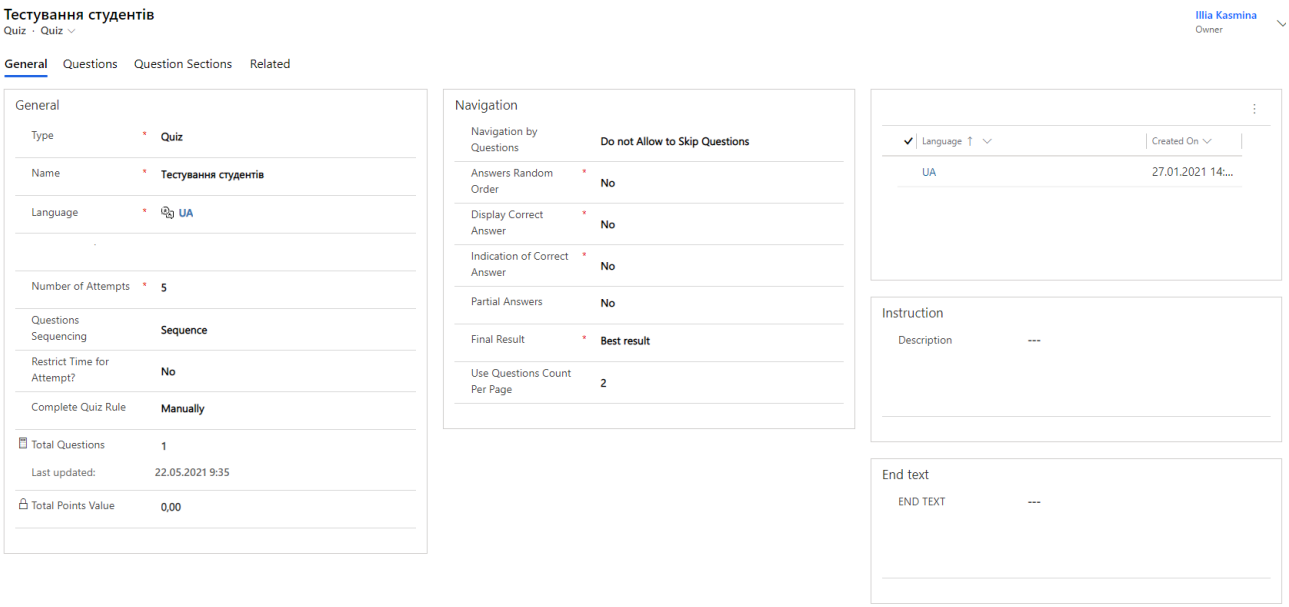

Рисунок 3.5 – Форма створення тесту

Для додавання та налаштування питань тесту використовуємо форму Question в CRM-системі (рис. 3.6) На формі є можливість вказати перелік відповідей на питання та коректність кожної з них.

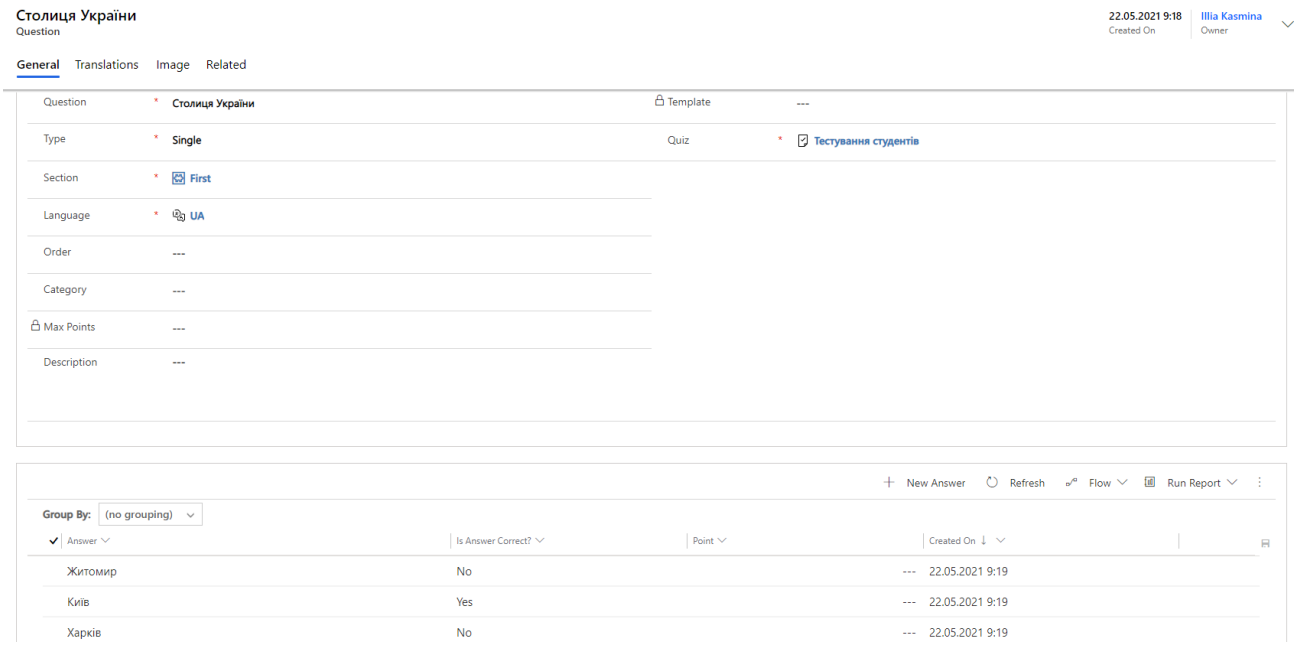

Рисунок 3.6 – Форма створення та налаштування питань тесту

Далі необхідно додати тест до навчального модуля та налаштувати параметри проходження модулю (рис. 3.7).

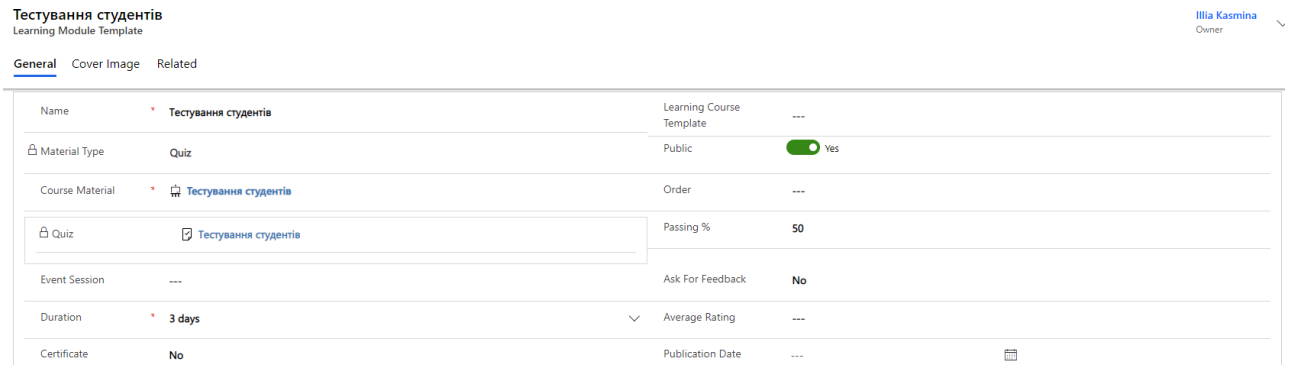

Рисунок 3.7 – Форма створення та налаштування навчального модуля

За необхідності є змога додати навички, які підтвердить користувач в разі успішного проходження тесту. Для відображення тесту на сторінці каталогу, додамо навчальних модуль до ролі доступу (рис. 3.8).

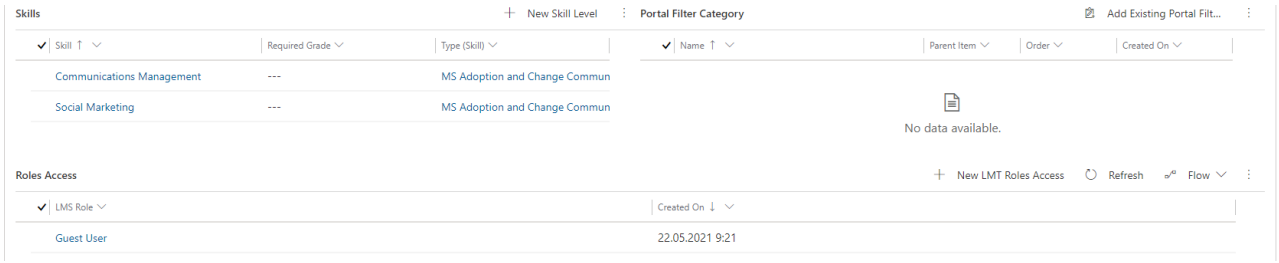

Рисунок 3.8 – Налаштування навичок та ролі доступу до навчального модуля

Для призначення тесту на користувачів скористаємося формою Assignments (рис. 3.9), на якій обираємо відповідний навчальний модуль.

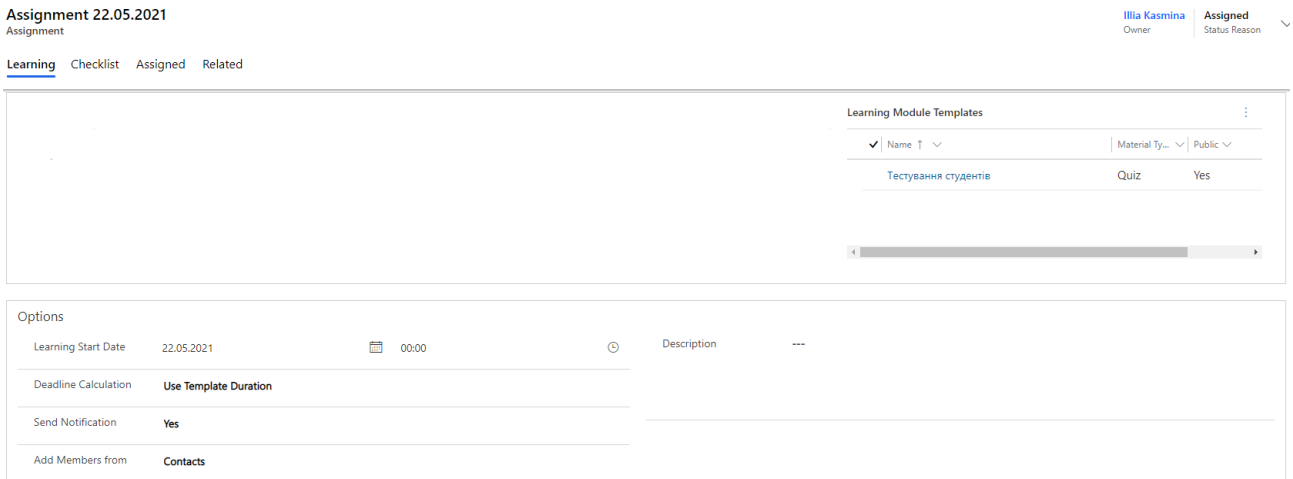

Рисунок 3.9 – Форма призначення тесту користувачам

Додаємо користувачів на формі Assignments, які повинні пройти тестування (рис. 3.10).

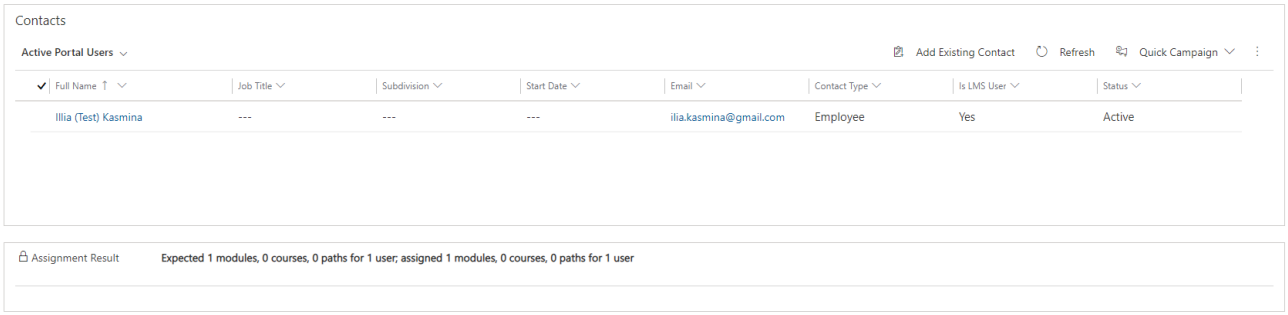

Рисунок 3.10 – Сторінка авторизації користувача

Після потрапляння на сайт, користувач опиняється на сторінці авторизації, на якій йому доступні функції авторизації за логіном та паролем та відновлення паролю (рис. 3.11). При введенні невірних даних облікового запису на формі виводиться повідомлення про помилку.

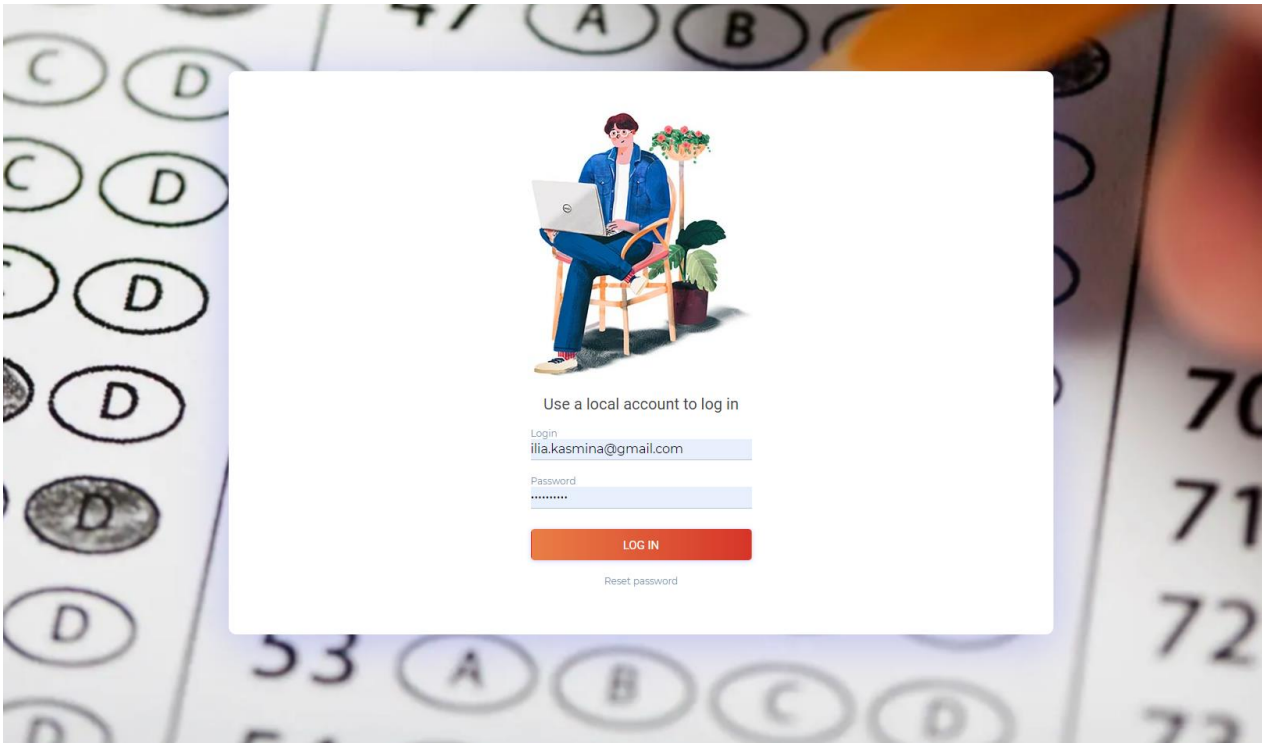

Рисунок 3.11 – Сторінка авторизації користувача

Успішна авторизація дозволяє користувачу потрапити на головну сторінку, на якій відображаються призначенні тести (рис. 3.12). На сторінці є можливість фільтрації матеріалів за певними умовами.

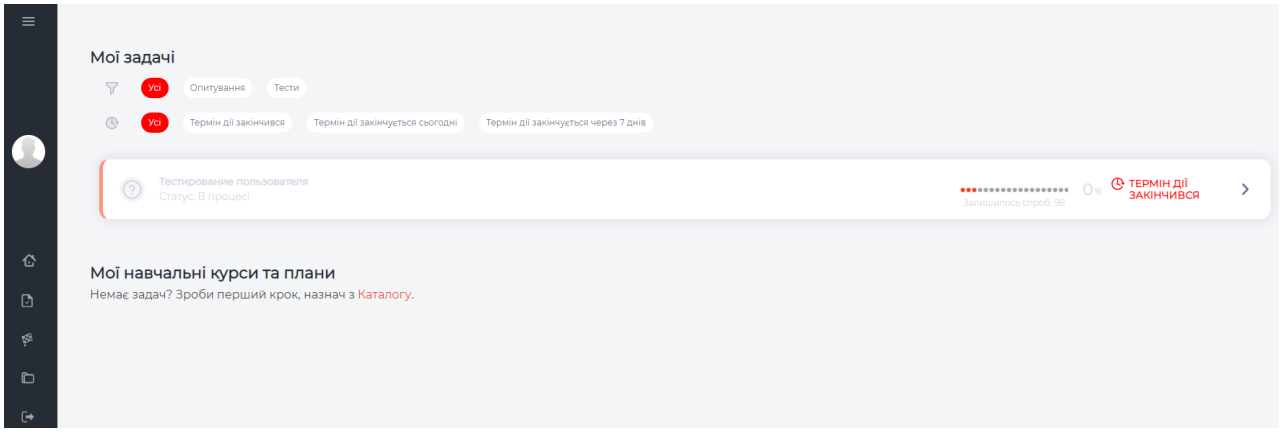

Рисунок 3.12 – Сторінка авторизації користувача

За допомогою панелі навігації є можливість пересування по сторінках сайту (рис. 3.13).

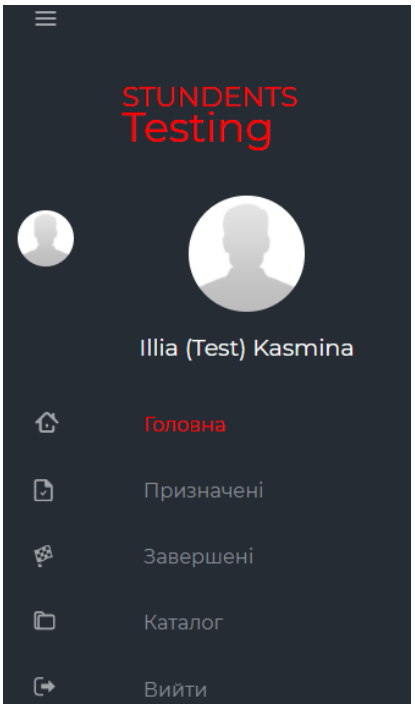

Рисунок 3.13 – Панель навігації

На сторінці профілю відображена інформація про особисті дані користувача, освіту, а також є можливість змінити мову інтерфейсу (рис. 3.14).

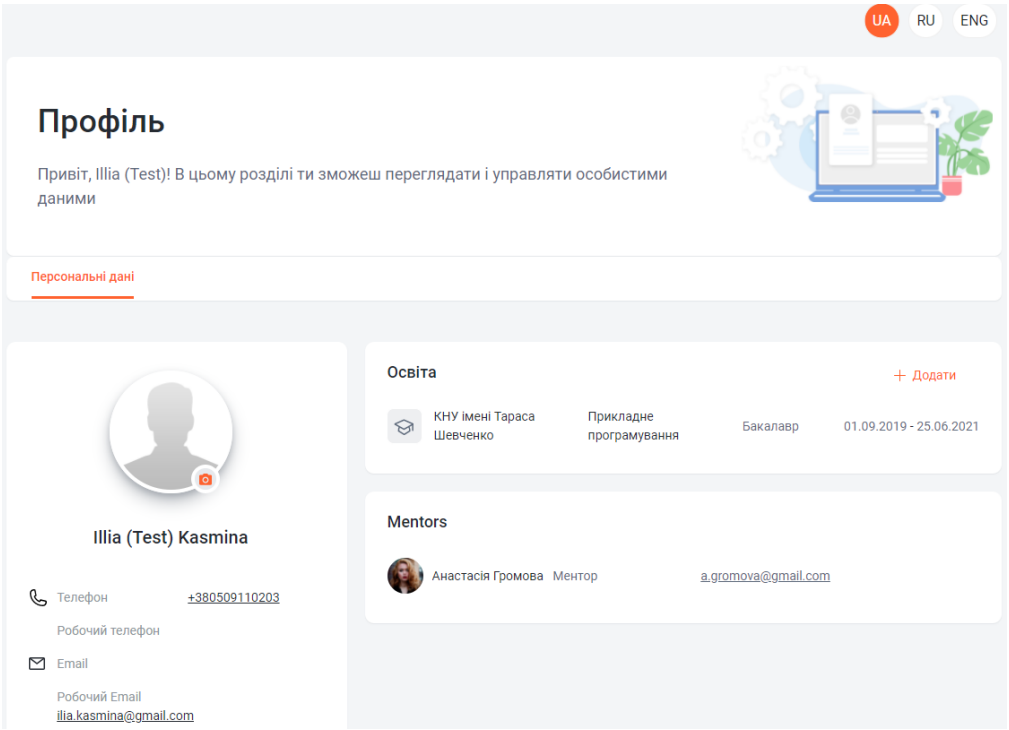

Рисунок 3.14 – Сторінка персональних даних

На сторінці призначених матеріалів відображається таблиця з тестами, призначеними користувачу (рис. 3.15). Є можливість сортування модулів за назвою, датою початку та закінчення, кількістю днів, які залишилися до завершення дії модулю, статусом та прогресом.

Призначені

Тип	Назва	Дата Початку	Дата Закінчення	Залишилось днів	Статус	Прогрес
Навчальний Модуль	Тестирование пользователя	14.03.2021	15.03.2021	0	У процесі	0%

Рисунок 3.15 – Сторінка призначених матеріалів

На сторінці завершених матеріалів відображається таблиця з тестами у статусі «Завершено» (рис. 3.16). Є можливість сортування модулів за назвою, статусом та прогресом.

Завершені

Тип	Назва	Статус	Прогрес
Навчальний Модуль	ppt upload vlboh	Пройдений	100%
Навчальний Модуль	Тест с отображением корректности ответов	Пройдений	75%

Рисунок 3.16 – Сторінка завершених матеріалів

На сторінці каталогу відображені всі доступні користувачеві матеріали (рис. 3.17). Є можливість пошуку за назвою, фільтрації за типом матеріалу та категорією навичок (рис. 3.18).

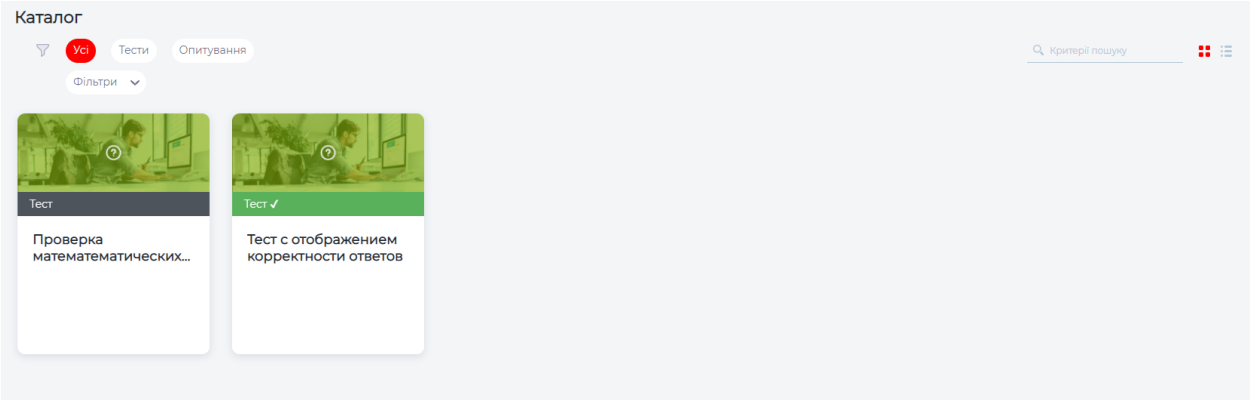

Рисунок 3.17 – Сторінка каталогу

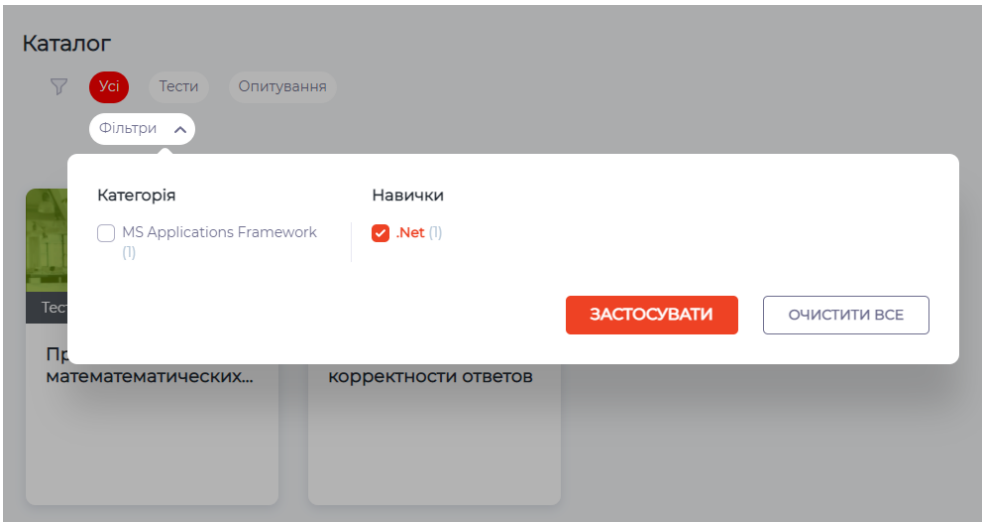

Рисунок 3.18 – Фільтрація матеріалів за категорією та навичками

На сторінці матеріалу є можливість переглянути інформацію про модуль, почати або відмінити призначення модулю (рис. 3.19).

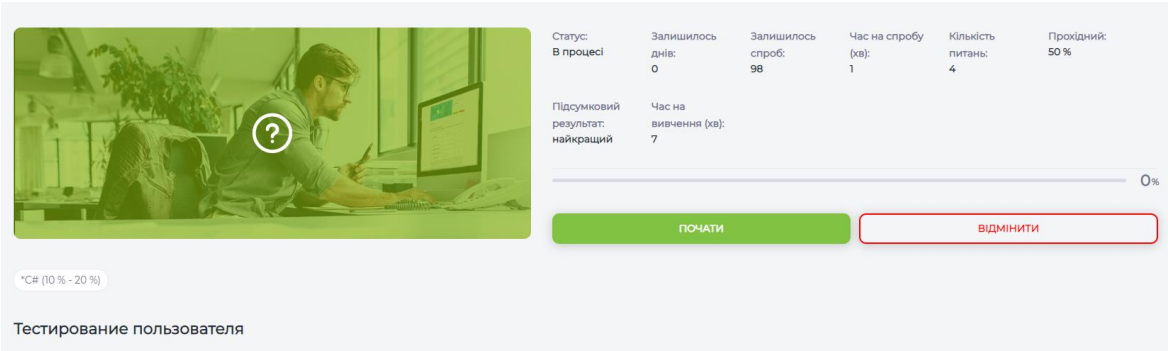

Рисунок 3.19 – Сторінка матеріалу

Після натискання на кнопці «Почати», користувач потрапляє на сторінку початку тестування (рис. 3.20)

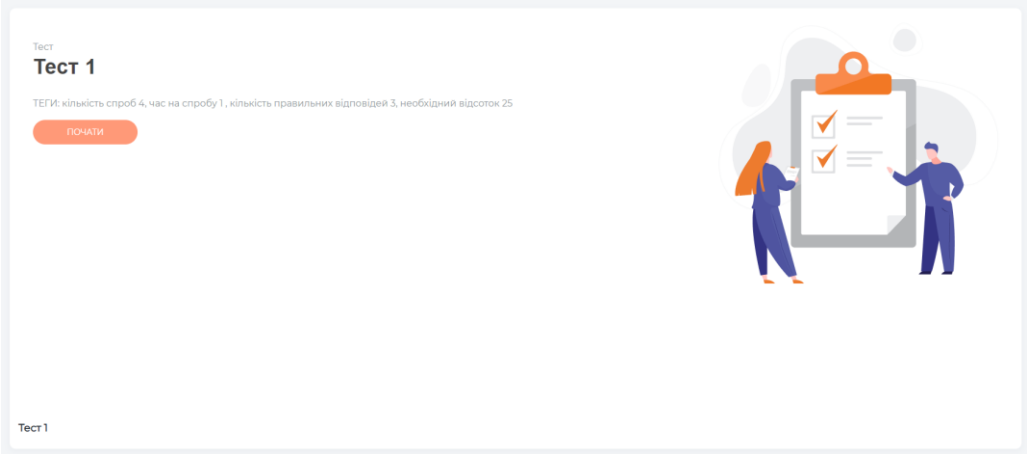

Рисунок 3.20 – Стартова сторінка тесту

Після запуску тесту на сторінці відображається запитання з варіантами відповіді (рис. 3.21). У разі налаштування тесту з обмеженням по часу, у верхній частині відображається зворотній відлік, що вказує на кількість часу, який залишився на проходження тестування. Якщо час вийшов до того, як були надані відповіді на всі запитання, користувача переадресує на сторінку результату.

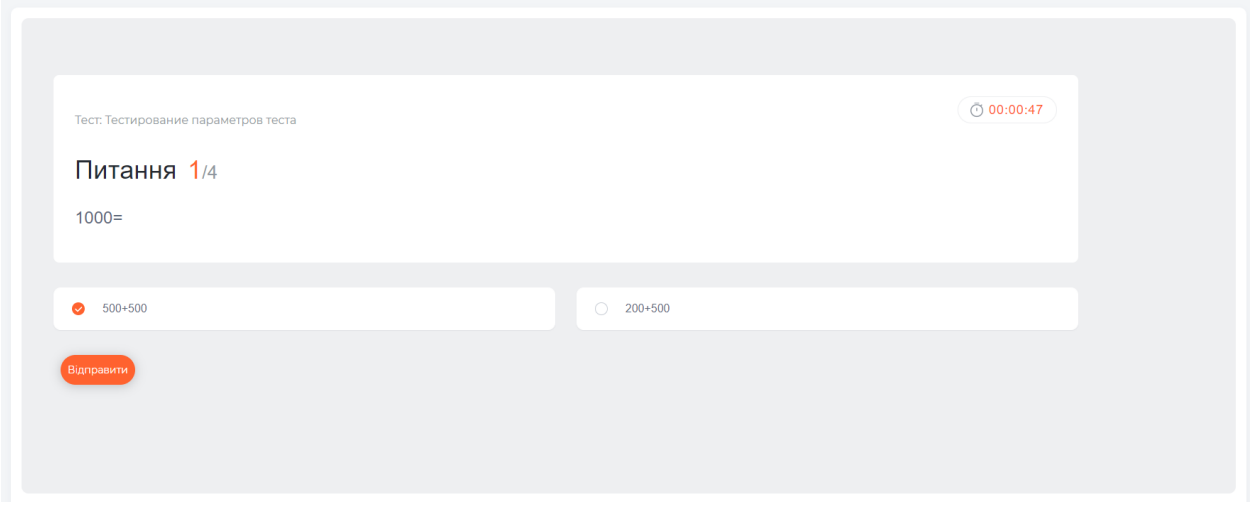

Рисунок 3.21 – Сторінка проходження тесту

Після того, як користувач підтвердив відповідь, відображається коректність наданої відповіді підсвічуванням відповідним кольором: зеленим правильна відповідь, червоним – хибна (рис. 3.22).

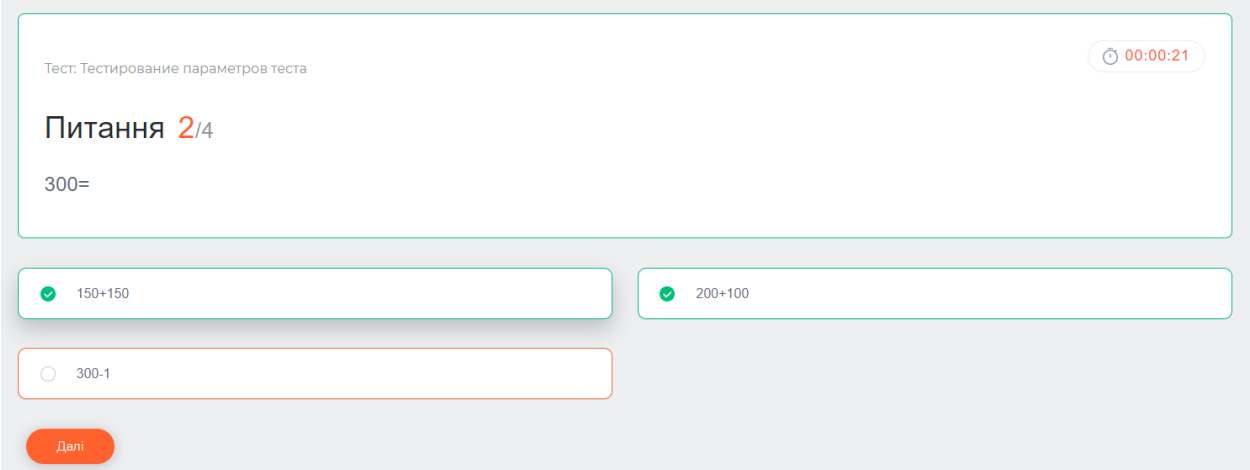

Рисунок 3.22 – Відображення коректності відповіді

Після завершення тесту, користувач потрапляє на сторінку результату, де відображаються поточні показники проходження тестування. Якщо налаштуваннями тесту передбачена можливість використання декількох спроб, можна пройти тест повторно або підтвердити результат (рис. 3.23).

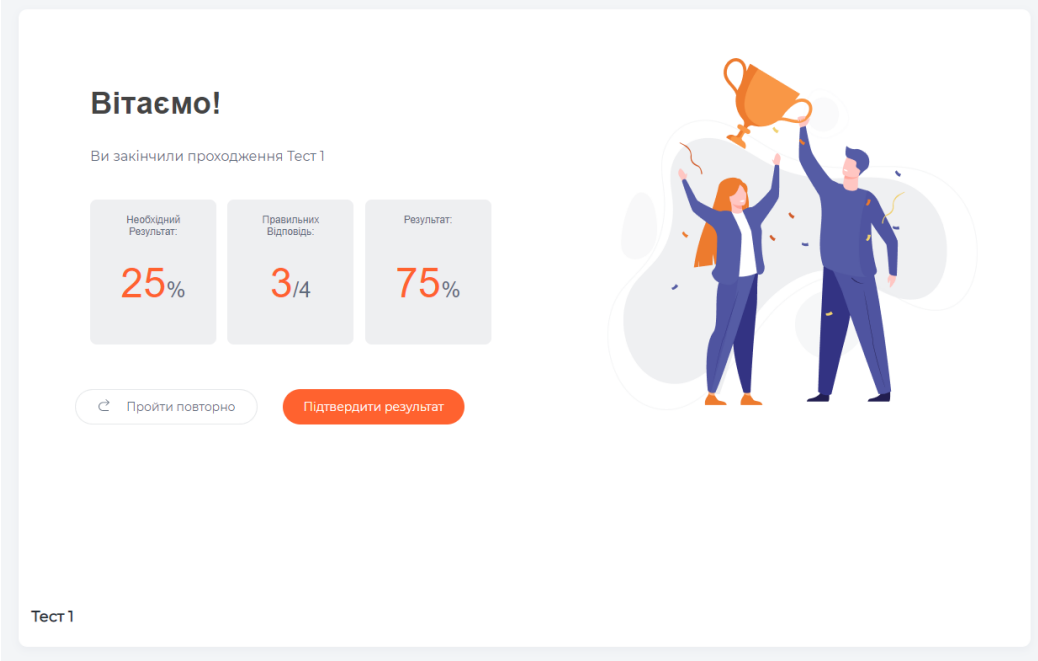

Рисунок 3.23 – Сторінка результату проходження тесту

Після підтвердження результату, користувач потрапляє на головну сторінку, з якої має можливість знову перейти каталог та побачити пройдений тест зі спеціальною відміткою про успішне проходження (рис. 3.24).

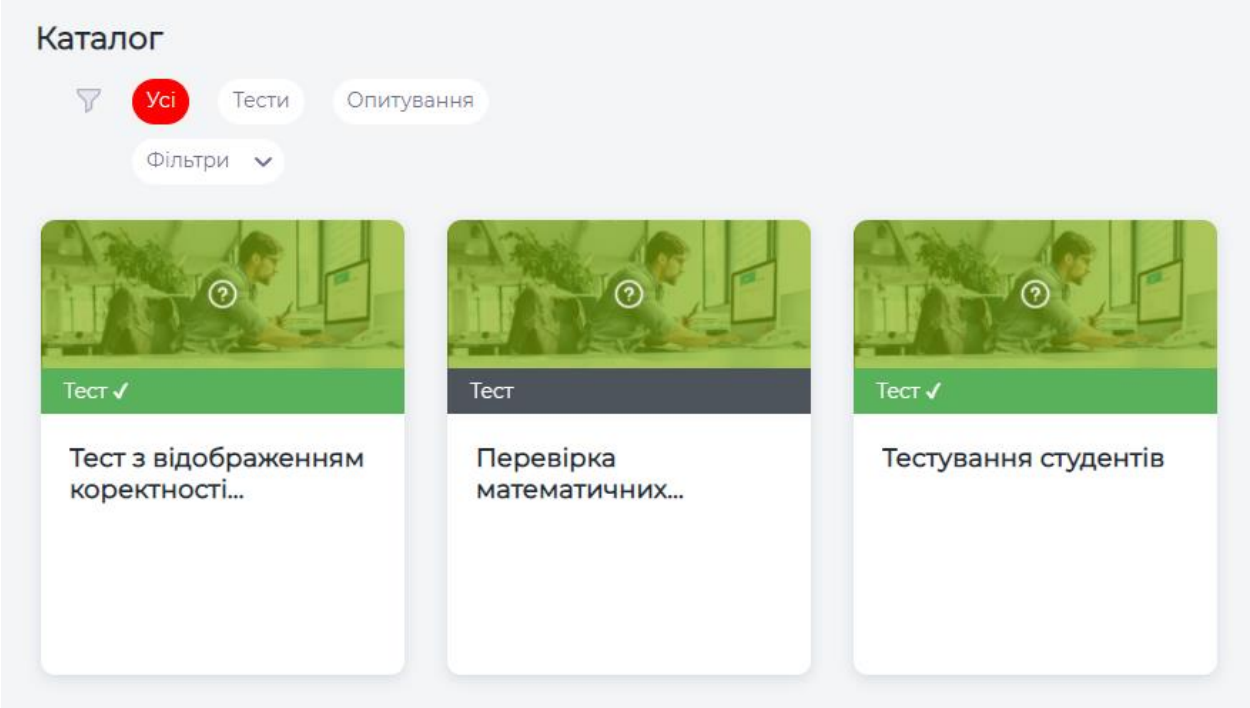

Рисунок 3.24 – Відображення статусу тесту

Для перегляду результатів проходження тестування студентами, переходимо на форму навчального модуля в CRM-системі (рис. 3.25).

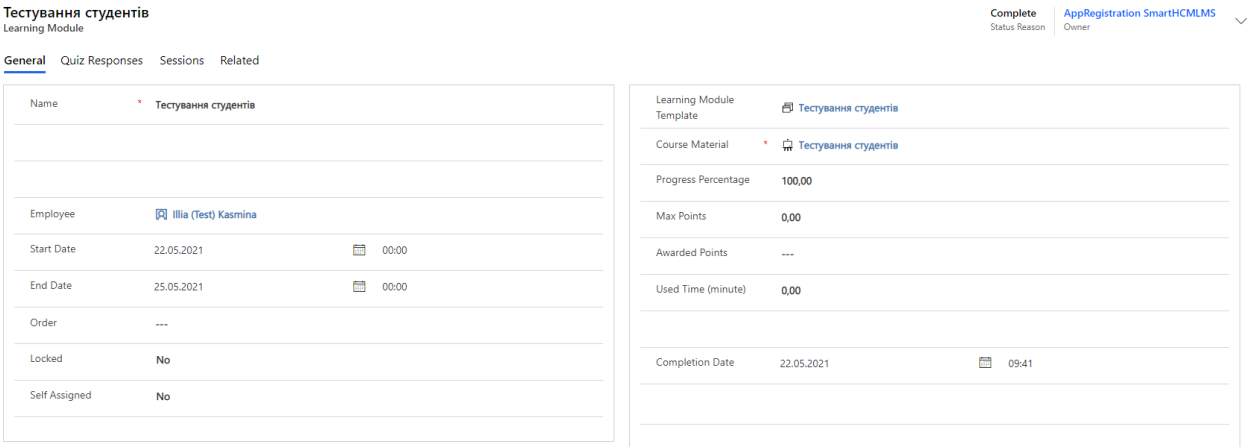

Рисунок 3.25 – Відображення результату проходження тесту

Висновки до розділу

У цьому розділі було проведено детальне дослідження об’єкту програмування, поставлено задачу та критерії для розробки та оформлення програмного продукту, визначено основні вимоги до програмного забезпечення, візуального оформлення та вимоги відповідно поставленому завданню.

Було проведено аналіз потоків даних в програмній системі, визначено програмні інструменти, за допомогою яких можна розробити програмну систему, проаналізована безпосередньо розробка продукту, наведено схему БД, надано опис компонентів веб-застосунку та продемонстровано роботу програмного продукту. Була здійснена практична реалізація веб-сервісу тестування студентів та описана інструкція користувача. Робота була виконана у середовищі Visual Studio 2019 Community за допомогою мови програмування C# на платформі .NET Core 3.1 та суміжних технологій HTML, CSS та JavaScript.

ВИСНОВОК

В результатом виконання кваліфікаційної роботи було створено програмну систему для тестування студентів. Для цього було обрано мову програмування C# на платформі ASP.NET, яка вирізняється зручністю розробки веб-застосунків, наявністю бібліотек для вирішення задач проектування системи, шаблонів та функцій, які необхідні для розробки алгоритмів різної складності.

Під час виконання кваліфікаційної роботи:

– дослідив теоретичні основи проектування та побудови програмної системи тестування студентів;

– проаналізував програмно-технічні рішення і види забезпечень типових веб-сервісів для тестування студентів;

– спроектував і впровадив програмну систему тестування студентів.

Було проаналізовано існуючі рішення програмних систем тестування студентів, виявлено їх переваги та недоліки, досліджена загальна концепція розробки та роботи веб-сервісу, проведено аналіз об’єкту програмування та визначено вимоги до інформаційного, програмного та технічного забезпечення. Була розроблена інструкція користувача щодо використання програмної системи тестування студентів.

Основною перевагою розробленого проекту є зручність у використанні, незалежність від операційної системи, браузеру та пристрою (смартфон або персональний комп’ютер), впровадження основного функціоналу на основі досліджених переваг існуючих рішень.

Програмний продукт має властивість масштабованості, що передбачає можливість впровадження нових функцій та розширення існуючих.

Розроблена програмна система відповідає поставленим вимогам технічного завдання та може бути конкурентною серед існуючих рішень завдяки впровадженим функціональним перевагам й гнучкості у використанні.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Freeman Adam. Pro ASP.NET Core MVC 2: монографія. Apress, 2017. 1017 с.

2. Price Mark. C# 7.1 and .NET Core 2.0: монографія Modern Cross-Platform Development Packt Publishing, 2017. 640 с.

3. Троелсен Е., Джепикс Ф. Мова програмування C# 7 і платформи .NET і .NET Core: монографія. Діалектика-Вільямс, 2019. 1328 с.

4. Richardson L., Amundsen M., Ruby S. RESTful Web APIs: Services for a Changing World: монографія. O'Reilly Media, 2013. 406 с.

5. Фленаган Д. JavaScript. Повне керівництво. 7-е видання: навчальний посібник. Діалектика 2021. 720 с.

6. Дакетт Д. Javascript і jQuery. Інтерактивна веб-розробка: монографія. Ексмо 2018. С. 640.

7. Макфарланд Д. Нова велика книга CSS: навчальний посібник. Питер Пресс, 2020. 720 с.

8. Структура і принципи WEB. WEB-сервери та принципи їх роботи. 2015. URL: https://dl.sumdu.edu.ua/textbooks/86975/413008/index.html.

9. Get started with Razor Pages in ASP.NET Core. 2019. URL: https://docs.microsoft.com/en-us/aspnet/core/tutorials/razor-pages/razor-pages-start?view=aspnetcore-3.1.

10. Entity Framework Core. Управління схемою БД і міграції. 2020. URL: https://metanit.com/sharp/entityframeworkcore/2.15.php.

11. Клімик Н. Сервіси для створення навчальних тестів. 2020. URL: https://buki.com.ua/ru/news/7-servisiv-dlya-stvorennya-navchalnykh-testiv-ta-zavdan-onlayn/.

12. Опис продукту Visual Studio 2019. 2020. URL: https://itpro.ua/product/visual-studio-2019-enterprise/?tab=description.

13. Робота з Visual Studio 2019. 2020. URL: https://professorweb.ru/my/programs/visual-studio/level1/.

14. Демищенко О. Онлайн-тестування в освітній та корпоративної сферах. 2015. URL: https://etutorium.ru/blog/onlajn-testirovanie-v-obrazovatelnoj-i-korporativnoj-sferakh.

15. Храмкін П. Системи онлайн тестування. 2018. URL: https://www.ispring.ru/elearning-insights/sistema-testirovaniya.

16. Земляной О. Веб-сервіс для оренди та продажу нерухомості. 2019. URL: https://ela.kpi.ua/bitstream/123456789/29947/1/Zemlianoi_bakalavr.pdf.

17. Шендрик В., Бондар О., Парфененко Ю. Web-програмування. 2013. URL: https://docplayer.net/113928377-Ministerstvo-osviti-i-nauki-ukrayini-sumskiy-derzhavniy-universitet-v-v-shendrik-o-v-bondar-yu-v-parfenenko-web-programuvannya-konspekt-lekciy.html.

18. Dynamics 365 Marketing. 2020. URL: https://docs.microsoft.com/uk-ua/dynamics365/marketing/help-hub.

ДОДАТКИ

Додаток А

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <link href="~/lib/bootstrap-quiz/dist/css/bootstrap.css" rel="stylesheet" />
    <link href="~/css/quiz.css" rel="stylesheet" />
    <link href="https://fonts.googleapis.com/css?family=Montserrat&display=swap" rel="stylesheet">
    <link rel="stylesheet" href="~/css/buttonStyles.css">
    <link rel="stylesheet" href="~/css/question.css">
    <link rel="stylesheet" href="~/css/fullScreenQuestion.css">
</head>
<body>
    <div id="container" class="col-md-12 col-sm-12 col-xs-12 text-left">
        @if (Model.ActiveQuestions.Count == 1)
        {
            <div id="question-header" isSequence="@isSequence">
                <div class="title-wrapper">
                    <div class="quiz-title">
                        @(quizType == @QuizType.Quiz ? Localizer["Quiz"] : Localizer["Survey"]): @Model.QuizTitle
                    </div>
                    @if (Model.UseTimer)
                    {
                        <div class="timer-wrapper">
                            <svg width="18" height="22" viewBox="0 0 18 22" fill="none" xmlns="http://www.w3.org/2000/svg">
                                <circle cx="9" cy="12.0625" r="8.25" class="v2-sidebar-icon-1-stroke" stroke-width="1.5" stroke-linecap="round" stroke-linejoin="round"/>
                                <path d="M9 9V12" class="v2-sidebar-icon-1-stroke" stroke-width="1.5" stroke-linecap="round" stroke-linejoin="round"/>
                                <line x1="6.125" y1="1.4375" x2="11.875" y2="1.4375" class="v2-sidebar-icon-1-stroke" stroke-linecap="round" stroke-linejoin="round"/>
                                <line x1="14.625" y1="5.37684" x2="15.8143" y2="4.1875" class="v2-sidebar-icon-1-stroke" stroke-width="1.5" stroke-linecap="round" stroke-linejoin="round"/>
                            </svg>
                            <div class="timer" data-seconds-left=@seconds></div>
                        </div>
                    }
                </div>
            </div>
        } else
        {
            @if (Model.UseTimer)
            {
                <div class="timer-wrapper">
                    <svg width="18" height="22" viewBox="0 0 18 22" fill="none" xmlns="http://www.w3.org/2000/svg">
                        <circle cx="9" cy="12.0625" r="8.25" class="v2-sidebar-icon-1-stroke" stroke-width="1.5" stroke-linecap="round" stroke-linejoin="round"/>
                        <path d="M9 9V12" class="v2-sidebar-icon-1-stroke" stroke-width="1.5" stroke-linecap="round" stroke-linejoin="round"/>
                        <line x1="6.125" y1="1.4375" x2="11.875" y2="1.4375" class="v2-sidebar-icon-1-stroke" stroke-linecap="round" stroke-linejoin="round"/>
                        <line x1="14.625" y1="5.37684" x2="15.8143" y2="4.1875" class="v2-sidebar-icon-1-stroke" stroke-width="1.5" stroke-linecap="round" stroke-linejoin="round"/>
                    </svg>
                    <div class="timer" data-seconds-left=@seconds></div>
                </div>
            }
        }
```

Рисунок А1 – Приклад HTML коду сторінки проходження тесту

```
function checkNextButtonRendering() {
    const isAllQuestionAnswered = $("div.question-body").toArray().every(checkIfQuestionAnswered)
    const isAllRequiredAnswered = $(".require-answer").toArray().map(elem => $(elem).closest("div.question-body")).every(checkIfQuestionAnswered);
    let disabled;
    if (window.isSequence) {
        disabled = !isAllQuestionAnswered
    } else {
        disabled = !isAllRequiredAnswered
    }
    $("#next-question-button").prop('disabled', disabled);
    $("#end-test-button").prop('disabled', disabled);

}

function checkIfQuestionAnswered(element) {

    const jElement = $(element);
            const isPredefinedAnswersSelected = jElement.find(".predefined-answer").toArray().some(input => $(input).is(':checked') && !$(input).closest('.list-group-item').hasClass('own-list-group-item'));
                const isCheckboxSelected = jElement.find(".checkbox input").toArray().some(input => $(input).is(':checked') && !$(input).closest('.list-group-item').hasClass('own-list-group-item'));
    const isOwnAnswerSelected = jElement.find(".ownanswerselector").is(":checked");
    const isTextAnswerGiven = jElement.find("textarea.ownanswer, input.own-answer-input").toArray().length > 0 &&
                        jElement.find("textarea.ownanswer, input.own-answer-input").toArray().every(input => $(input).val().replace(/ /g, '').length > 0 || ($(input).attr('defaultValue') && $(input).attr('de
                const isOwnTextAnswerGiven = jElement.find("input.multiple-text-area").toArray().length > 0 &&
                        jElement.find("input.multiple-text-area").toArray().every(input => $(input).val().replace(/ /g, '').length > 0);
    const isScaleAnswerSelected = jElement.find(".scale-answer").toArray().length > 0 && jElement.find(".scale-answer").toArray().every(input => $(input).val() > 0);
return isPredefinedAnswersSelected || isOwnAnswerSelected || isOwnTextAnswerGiven || isTextAnswerGiven || isCheckboxSelected || isScaleAnswerSelected;
}

function onInputSelectionChange(e) {
    checkNextButtonRendering();
}
```

Рисунок А2 – Приклад JavaScript коду сторінки проходження тесту

```
public class QuestionModel : PageModel
{
    private readonly ApplicationDbContext _dbContext;
    private readonly IStringLocalizer<QuestionModel> _localizer;
    private readonly BaseQuestionModelFactory _baseQuestionModelFactory;
    private readonly QuizPassingService _quizPassingService;
    private readonly LocalizationService _localizationService;
    private readonly QuizNavigationBuilder _quizNavigationBuilder;
    private readonly ILoggerService _loggerService;
    private readonly UserAccessor _userAccessor;

    0 references | 0 exceptions
    public QuestionModel(ApplicationDbContext dbContext,
        IStringLocalizer<QuestionModel> localizer,
        BaseQuestionModelFactory baseQuestionModelFactory,
        QuizPassingService quizPassingService,
        QuizNavigationBuilder quizNavigationBuilder,
        LocalizationService localizationService,
        ILoggerService loggerService,
        UserAccessor userAccessor)
    {
        _loggerService = loggerService;
        _dbContext = dbContext;
        _localizer = localizer;
        _baseQuestionModelFactory = baseQuestionModelFactory;
        _quizPassingService = quizPassingService;
        _quizNavigationBuilder = quizNavigationBuilder;
        _localizationService = localizationService;
        _userAccessor = userAccessor;
    }

    3 references | 0 exceptions
    public QuizNavigationModel QuizNavigation { get; set; }
    7 references | 0 exceptions
    public bool UseTimer { get; set; }
    2 references | 0 exceptions
    public DateTime TestEndTime { get; set; }
    17 references | 0 exceptions
    public List<QuizQuestionResponse> QuizQuestionResponses { get; set; }
    49 references | 0 exceptions
    public QuizResponse QuizResponse { get; set; }
    4 references | 0 exceptions
    public int QuestionsPerPage { get; set; }
    2 references | 0 exceptions
    public string QuizTitle { get; set; }
    1 reference | 0 exceptions
    public bool AnyQuestionsAfterSurvey { get; set; }
    6 references | 0 exceptions
    public bool AllQuestionsAreAnswered { get; set; }
    1 reference | 0 exceptions
    public bool IsNextButtonAvailable { get; set; }
```

Рисунок А3 – Приклад підключення даних за допомогою Razor Page на сторінці проходження тесту

```
public async Task<IActionResult> OnGetAsync(Guid quizResponseId, Guid? lastViewedQuestion = null)
{
    QuizResponse = QuizResponse ?? await _quizPassingService.GetQuizResponseByIdAsync(quizResponseId);
    QuizQuestionResponses = QuizQuestionResponses ?? QuizResponse.QuizQuestionResponses
        .ToList();
    var quiz = QuizResponse.LearningModule.Material.Quiz;
    QuestionsPerPage = quiz.QuestionsPerPage ?? 1;
    var quizQuestionResponses = QuizQuestionResponses
        .SkipWhile(qqr => lastViewedQuestion != null && qqr.Id != lastViewedQuestion)
        .Skip(lastViewedQuestion != null ? 1 : 0)
        .Where(qqr => qqr.Answers == null || !qqr.IsCorrect.HasValue)
        .ToList();

    quizQuestionResponses = quizQuestionResponses
           .Paged(QuestionsPerPage)
           .First()
           .ToList();

    QuizTitle = quiz.Title;

    await _localizationService.LocalizeQuestionsAsync(
        QuizQuestionResponses.Select(response => response.Question),
        QuizResponse.Language);

    if (QuizResponse.IsCompleted)
    {
        throw new QuizException(_localizer["Quiz already passed"])
        {
            QuizResponseId = QuizResponse.Id
        };
    }

    UseTimer = quiz.RestrictTimeForAttempt && quiz.TimeForAttempt.HasValue;
    if (UseTimer)
    {
        if (!QuizResponse.EndDate.HasValue)
        {
            var quizResponse = await _dbContext.QuizResponses.FindAsync(QuizResponse.Id);
            quizResponse.EndDate = DateTime.UtcNow + QuizResponse.RemainingTime;
            await _dbContext.SaveChangesAsync();
        }

        if (DateTime.UtcNow > QuizResponse.EndDate)
        {
            await _quizPassingService.EndAttemptAsync(QuizResponse, isQuizCompleted: false);

            //Time expired
            throw new QuizException(_localizer["Unfortunately your time is up"])
            {
                QuizResponseId = QuizResponse.Id
            };
```

Рисунок А4 – Приклад обробки GET запиту за допомогою Razor Page на сторінці проходження тесту

```
public async Task<IActionResult> OnPostNextPageAsync()
{
    IList<QuizQuestionResponse> activeQuizQuestionResponses;

    if (IsShowAnswer)
    {
        activeQuizQuestionResponses = await GetQuizQuestionResponse();
    }
    else
    {
        try
        {
            activeQuizQuestionResponses = await PrepareActiveQuestionChange();
        }
        catch (NotAllQuestionsAnsweredException)
        {
            return await GetQuestionNotAnsweredErrorPageAsync();
        }

        if (QuizResponse.LearningModule.Material.Quiz.Type == QuizType.Branches)
        {
            var currentQuizQuestionResponse = activeQuizQuestionResponses.Last();
            if (currentQuizQuestionResponse.Answers.Count > 0)
            {
                //Already answered
                throw new QuizException(_localizer["You cannot change your answer"])
                {
                    QuizResponseId = QuizResponse.Id
                };
            }

            //End test
            await OnPostEndQuizAsync();
        }

        await _dbContext.SaveChangesAsync();
    }

    var quizResponseId = activeQuizQuestionResponses.First().QuizResponseId;
    var quizQuestionResponseId = QuizResponse.LearningModule.Material.Quiz.Type == QuizType.Survey
        ? new Guid?(activeQuizQuestionResponses.Last().Id)
        : null;

    return RedirectToPage(new { quizResponseId, quizQuestionResponseId });
}
```

Рисунок А5 – Приклад обробки POST запиту для переходу на наступну сторінку тесту за допомогою Razor Page

```
public async Task<IActionResult> OnPostEndQuizAsync()
{
    IList<QuizQuestionResponse> activeQuizQuestionResponses;

    if (IsShowAnswer)
    {
        activeQuizQuestionResponses = await GetQuizQuestionResponse();
    }
    else
    {
        try
        {
            activeQuizQuestionResponses = await PrepareActiveQuestionChange();
        }
        catch (NotAllQuestionsAnsweredException)
        {
            return await GetQuestionNotAnsweredErrorPageAsync();
        }
    }

    if (QuizResponse.LearningModule.Material.Quiz.Type == QuizType.Branches)
    {
        var currentQuizQuestionResponse = activeQuizQuestionResponses.Last();
        if (currentQuizQuestionResponse.Answers.Count > 0)
        {
            //Already answered
            throw new QuizException(_localizer["You cannot change your answer"])
            {
                QuizResponseId = QuizResponse.Id,
            };
        }
    }

    if (!QuizResponse.LearningModule.Material.Quiz.RestrictTimeForAttempt)
    {
        var quizResponseId = QuizResponse.Id;
        var quizResponse = await _dbContext.QuizResponses.FindAsync(quizResponseId);
        quizResponse.EndDate = DateTime.UtcNow;
    }

    await _quizPassingService.EndAttemptAsync(QuizResponse, isQuizCompleted: true);

    if (QuizResponse.LearningModule.Material.Quiz.Type == QuizType.Survey)
    {
        return RedirectToPage("/Quiz/Result",
            new { quizResponseId = QuizResponse.Id, anyNonCalculableAnswersIncluded = true });
    }

    return RedirectToPage("/Quiz/Result", new { quizResponseId = QuizResponse.Id });
}
```

Рисунок А6 – Приклад обробки POST запиту для завершення тесту

```
public class ApplicationDbContext : IdentityDbContext<User, Role, Guid>
{
    private static readonly string defaultSchema = "dbo";

    0 references
    public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options) : base(options)
    {
        var inMemoryDbIsUsed = options.Extensions.Any(extension => extension is InMemoryOptionsExtension);

        if (!inMemoryDbIsUsed)
        {
            Database.SetCommandTimeout(60);
        }
    }

    Identity

    #region Quiz

    1 reference
    public DbSet<Quiz.Quiz> Quizzes { get; set; }
    6 references
    public DbSet<QuizResponse> QuizResponses { get; set; }
    1 reference
    public DbSet<QuizQuestion> QuizQuestions { get; set; }
    7 references
    public DbSet<QuizQuestionResponse> QuizQuestionResponses { get; set; }
    0 references
    public DbSet<QuestionAnswer> QuestionAnswers { get; set; }
    3 references
    public DbSet<QuestionAnswerResponse> QuestionAnswerResponses { get; set; }

    #endregion Quiz

    0 references
    protected override void OnModelCreating(ModelBuilder builder)
    {
        builder.HasDefaultSchema(defaultSchema);

        base.OnModelCreating(builder);

        builder.Entity<QuizQuestionResponse>()
            .HasOne(q => q.QuestionAnswerResponse)
            .WithOne()
            .HasForeignKey<QuizQuestionResponse>(q => q.QuestionAnswerResponseId);

        builder.Entity<QuestionAnswerResponse>()
            .HasOne(q => q.QuizQuestionResponse)
            .WithOne()
            .HasForeignKey<QuestionAnswerResponse>(q => q.QuizQuestionResponseId);
    }
}
```

Рисунок А7 – Приклад класу контексту бази даних

```
public partial class Init : Migration
{
    2 references
    protected override void Up(MigrationBuilder migrationBuilder)
    {
        migrationBuilder.EnsureSchema(
            name: "dbo");

        migrationBuilder.CreateTable(
            name: "AspNetRoles",
            schema: "dbo",
            columns: table => new
            {
                Id = table.Column<Guid>(type: "uniqueidentifier", nullable: false),
                Name = table.Column<string>(type: "nvarchar(256)", maxLength: 256, nullable: true),
                NormalizedName = table.Column<string>(type: "nvarchar(256)", maxLength: 256, nullable: true),
                ConcurrencyStamp = table.Column<string>(type: "nvarchar(max)", nullable: true)
            },
            constraints: table =>
            {
                table.PrimaryKey("PK_AspNetRoles", x => x.Id);
            });

        migrationBuilder.CreateTable(
            name: "AspNetUsers",
            schema: "dbo",
            columns: table => new
            {
                Id = table.Column<Guid>(type: "uniqueidentifier", nullable: false),
                FirstName = table.Column<string>(type: "nvarchar(max)", nullable: true),
                LastName = table.Column<string>(type: "nvarchar(max)", nullable: true),
                FullName = table.Column<string>(type: "nvarchar(max)", nullable: true),
                UserName = table.Column<string>(type: "nvarchar(256)", maxLength: 256, nullable: true),
                NormalizedUserName = table.Column<string>(type: "nvarchar(256)", maxLength: 256, nullable: true),
                Email = table.Column<string>(type: "nvarchar(256)", maxLength: 256, nullable: true),
                NormalizedEmail = table.Column<string>(type: "nvarchar(256)", maxLength: 256, nullable: true),
                EmailConfirmed = table.Column<bool>(type: "bit", nullable: false),
                PasswordHash = table.Column<string>(type: "nvarchar(max)", nullable: true),
                SecurityStamp = table.Column<string>(type: "nvarchar(max)", nullable: true),
                ConcurrencyStamp = table.Column<string>(type: "nvarchar(max)", nullable: true),
                PhoneNumber = table.Column<string>(type: "nvarchar(max)", nullable: true),
                PhoneNumberConfirmed = table.Column<bool>(type: "bit", nullable: false),
                TwoFactorEnabled = table.Column<bool>(type: "bit", nullable: false),
                LockoutEnd = table.Column<DateTimeOffset>(type: "datetimeoffset", nullable: true),
                LockoutEnabled = table.Column<bool>(type: "bit", nullable: false),
                AccessFailedCount = table.Column<int>(type: "int", nullable: false)
            },
            constraints: table =>
            {
                table.PrimaryKey("PK_AspNetUsers", x => x.Id);
            });
```

Рисунок А8 – Приклад міграції до бази даних

```
public class QuizResponse : IEntity
{
    55 references
    public Guid Id { get; set; }
    3 references
    public Guid? QuizId { get; set; }
    13 references
    public Quiz Quiz { get; set; }
    1 reference
    public Guid? UserId { get; set; }
    0 references
    public User User { get; set; }
    15 references
    public ICollection<QuizQuestionResponse> QuizQuestionResponses { get; set; }
    1 reference
    public DateTime? StartDate { get; set; }
    9 references
    public DateTime? EndDate { get; set; }
    5 references
    public bool IsCompleted { get; set; }
    3 references
    public TimeSpan? RemainingTime { get; set; }
    1 reference
    public int? TotalQuestionsAnsweredCount { get; set; }
    1 reference
    public int? TotalQuestionsCount { get; set; }
    2 references
    public int? TotalQuestionsWithCorrectAnswersCount { get; set; }
    1 reference
    public int? TotalQuestionsWithIncorrectAnswersCount { get; set; }
}
```

Рисунок А9 – Приклад моделі даних, яка представлена в базі даних

```
public void ConfigureServices(IServiceCollection services)
{
    services.TryAddSingleton<IHttpContextAccessor, HttpContextAccessor>();

    services.Configure<CookiePolicyOptions>(options =>
    {
        // This lambda determines whether user consent for non-essential cookies is needed for a given request.
        options.CheckConsentNeeded = context => true;
        options.MinimumSameSitePolicy = SameSiteMode.None;
    });

    services.AddSession(options =>
    {
        options.Cookie.Name = "ST.Session";
        options.IdleTimeout = TimeSpan.FromHours(8);
    })
        .AddAntiforgery(options => options.SuppressXFrameOptionsHeader = true)
        .AddAntiforgery(options => options.HeaderName = "XSRF-TOKEN");

    services.AddDataServices(Configuration.GetConnectionString("SqlConnectionString"));

    services.AddAuthentication().AddCookie();

    services.AddMvc(options =>
    {
        var policy = new AuthorizationPolicyBuilder()
            .RequireAuthenticatedUser()
            .Build();
        options.Filters.Add(new AuthorizeFilter(policy));
    });
    services.Configure<IdentityOptions>(options =>
        Configuration.GetSection(nameof(IdentityOptions)).Bind(options));
    services.Configure<CacheOptions>(Configuration.GetSection(nameof(CacheOptions)));

    services.AddTransient<ICacheService, DistributedCacheService>();
    services.AddTransient<UserAccessor>();
    services.AddScoped<UserCachingMiddleware>();
    services.AddScoped<IUserService, UserService>();
    services.AddTransient<ILayoutService, LayoutService>();
    services.AddTransient<QuizPassingService>();

    services.AddDistributedMemoryCache();

    services.AddRazorPages();
}
```

Рисунок А10 – Приклад конфігурації сервісів проекту в файлі Startup.cs