Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Програмна система для онлайн-навчання кулінарному мистецтву
(тема)

Виконав:
здобувач 4 року навчання
групи ПЗПІ-21-1
Тарас НЕХ
(Власне ім'я, ПРІЗВИЩЕ)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Програмна інженерія
(повна назва освітньої програми)

Керівник доц. Ірина АФАНАСЬЄВА
(посада, Власне ім'я, ПРІЗВИЩЕ)

Допускається до захисту
Зав. кафедри _____ Кирило СМЕЛЯКОВ
(підпис) (Власне ім'я, ПРІЗВИЩЕ)

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерних наук
Кафедра програмної інженерії
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми Освітньо-професійна
Освітня програма Програмна Інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Неху Тарасу Віталійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Програмна система для онлайн навчання кулінарному мистецтву
Затверджена наказом по університету від 19.05. 2025р. № 397 Ст
2. Термін подання студентом роботи до екзаменаційної комісії 16.06.2025
3. Вихідні дані до роботи Розробити програмну систему для онлайн навчання кулінарному мистецтву, котра дозволить користувачам як навчатись так і навчати інших, мовами програмування C# використовуючи фреймворк .NET.
4. Перелік питань, що потрібно опрацювати в роботі
Вступ, аналіз предметної галузі, формування вимог до програмної системи, архітектура та проєктування програмного забезпечення, опис прийнятих програмних рішень, тестування розробленого програмного забезпечення, висновки, додатки.

КАЛЕНДАРНИЙ ПЛАН

	Назва етапів роботи	Термін виконання етапів роботи	Примітка
	Аналіз предметної галузі	01.05.2025 - 02.05.2025	*виконано*
	Створення специфікації ПЗ	03.05.2025 - 05.05.2025	*виконано*
	Проєктування ПЗ	06.05.2025 - 07.05.2025	*виконано*
	Розробка ПЗ	08.05.2025 - 18.05.2025	*виконано*
	Тестування ПЗ	19.05.2025 - 20.05.2025	*виконано*
	Оформлення пояснювальної записки	21.05.2025 - 30.05.2025	*виконано*
	Створення заяви щодо самостійного виконання кваліфікаційної роботи	31.05.2025 - 01.06.2025	*виконано*
	Підготовка презентації та доповіді	02.06.2025 - 05.06.2025	*виконано*
	Створення електронного архіву з матеріалами	08.06.2025	*виконано*
	Перевірка пояснювальної записки керівником	05.06.2025	*виконано*
	Підготовка роботи до перевірки на плагіат	06.06.2025	*виконано*
	Підготовка роботи для нормоконтроля, проходження нормоконтроля	06.06.2025 - 10.06.2025	*виконано*
	Оцінка роботи рецензентом	11.06.2025	
	Попередній захист	13.06.2025	*виконано*
	Нормоконтроль, рецензування	11.06.2025	*виконано*
	Здача роботи у електронний архів	13.06.2025	*виконано*
	Допуск до захисту у зав. кафедри	15.06.2025	*виконано*
	Захист кваліфікаційної роботи	16.06.2025	*виконано*

Дата видачі завдання « 6 » « квітня » 2025р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Ірина АФАНАСЬЄВА
(підпис) (посада, Власне ім’я, ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи бакалавра, 78 стор., 14 рис., 17 джер.

ОНЛАЙН-НАВЧАННЯ, ОСВІТНІЙ ВЕБ-СЕРВІС, КУЛІНАРНА ОСВІТА, ASP.NET CORE, RAZOR PAGES, C#, SQL SERVER, MVC АРХІТЕКТУРА, КЛІЄНТ-СЕРВЕРНА АРХІТЕКТУРА.

Об’єкт розробки – програмна система для онлайн-навчання кулінарному мистецтву.

Мета розробки – створення веб-застосунку, що надає можливість користувачам проходити кулінарні курси, виконувати практичні завдання, переглядати навчальні матеріали, проходити тести, отримувати оцінки та взаємодіяти з викладачами.

Метод розв’язання – використання ASP.NET Core з Razor Pages, мови програмування C#, бази даних SQL Server, інструментів проєктування інтерфейсу (наприклад, Figma), архітектури REST API та багаторівневої структури (Presentation, BLL, DAL).

У результаті розробки створено веб-застосунок, що забезпечує реєстрацію користувачів, доступ до навчального контенту, тестування, оцінювання, перегляд відео й аналіз особистого прогресу.

ABSTRACT

ONLINE LEARNING, EDUCATIONAL WEB SERVICE, CULINARY EDUCATION, ASP.NET CORE, RAZOR PAGES, C#, SQL SERVER, MVC ARCHITECTURE, CLIENT-SERVER ARCHITECTURE.

Development Object – a software system for online culinary education.

Development Goal – to create a web application that enables users to take culinary courses, complete practical tasks, view educational materials, take tests, receive grades, and interact with instructors.

Solution Method – implementation using ASP.NET Core with Razor Pages, C# programming language, SQL Server database, interface design tools (such as Figma), REST API architecture, and a multi-tiered structure (Presentation, BLL, DAL).

Development Result – a web application was developed that provides user registration, access to educational content, testing, grading, video viewing, and personal progress analysis.

ЗМІСТ

ВСТУП

У сучасних умовах онлайн-освіта набула широкого поширення в різних сферах, включаючи медицину, ІТ, інженерію, мистецтво та, зокрема, кулінарне мистецтво. Дистанційні технології відкрили нові можливості для здобуття знань, дозволяючи навчатись у зручний час, у зручному місці та у власному темпі. Це особливо актуально для людей, які не мають змоги відвідувати традиційні навчальні заклади, зокрема через щільний графік, віддалене місце проживання або інші обставини. Актуальність розробки освітніх веб-платформ обумовлена загальними тенденціями цифровізації суспільства, які були відзначені на міжнародному рівні як перспективні напрями розвитку освіти [1]. Онлайн-курси з кулінарії дають змогу опанувати як базові, так і професійні навички приготування страв, працювати з різними інгредієнтами, техніками та кухнями світу. Завдяки інтерактивному формату навчання студенти мають можливість переглядати відеоуроки, виконувати практичні завдання, брати участь у тематичних обговореннях, отримувати зворотний зв’язок від викладачів, а також оцінки та рекомендації щодо покращення результатів. Такий підхід сприяє не лише кращому засвоєнню матеріалу, а й формуванню реальних навичок, необхідних для роботи в кулінарній галузі.

Метою даної роботи є розробка програмної системи для онлайн-навчання кулінарному мистецтву, яка забезпечить зручну, інтуїтивно зрозумілу та функціональну платформу для взаємодії між студентами й викладачами. Ця система дозволить користувачам проходити авторизований доступ до персонального кабінету, обирати курси відповідно до інтересів і рівня підготовки, переглядати відеоматеріали, ознайомлюватися з текстовими статтями та додатковими ресурсами, виконувати практичні завдання, проходити тести для перевірки знань і отримувати оцінки від викладачів. Також користувачі матимуть змогу залишати коментарі, брати участь в обговореннях і спілкуватися в чаті, що значно підвищує рівень взаємодії та створює ефект присутності. У процесі розробки системи особливу увагу приділено дотриманню принципів архітектури програмного забезпечення, зокрема впровадженню багаторівневої структури, яка

включає шар представлення (інтерфейс користувача), шар бізнес-логіки (обробка запитів, валідація, правила роботи з даними) та шар доступу до даних (взаємодія з базою даних). Такий підхід забезпечує гнучкість, масштабованість і зручність підтримки системи в майбутньому.

У процесі виконання роботи було проведено аналіз специфіки онлайн-освіти в сфері кулінарії, досліджено сучасні тенденції та підходи до організації дистанційного навчання, вивчено потреби цільової аудиторії, яка включає як новачків, так і досвідчених кулінарів, що прагнуть удосконалити свої навички. На основі зібраних вимог було визначено функціональні та нефункціональні характеристики майбутньої системи, обрано сучасні інструменти розробки — мову програмування, фреймворки, базу даних, шаблони архітектури — і здійснено проєктування загальної структури веб-застосунку. Результатом даної роботи стане повнофункціональна програмна система, що відповідає сучасним стандартам якості, безпеки та зручності користування, а також дозволяє ефективно організувати процес онлайн-навчання кулінарному мистецтву з урахуванням реальних потреб користувачів і вимог ринку.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

Сфера кулінарної освіти останніми роками активно адаптується до цифрового середовища, що обумовлено глобальними змінами в освітніх підходах та стрімким розвитком інформаційних технологій. Традиційні методи навчання у кулінарних школах, що передбачають очну присутність студентів на заняттях, дедалі частіше доповнюються інноваційними цифровими форматами. У багатьох випадках ці інновації не лише доповнюють класичні курси, а й повністю їх замінюють. Онлайн-курси з кулінарії дозволяють учням опановувати як базові навички приготування їжі (таких як нарізання, термічна обробка, подача страв), так і спеціалізовані компетенції, зокрема приготування страв національної кухні, роботу з вегетаріанськими інгредієнтами, безглютеновими продуктами, дієтичними рецептами тощо.

Завдяки онлайн-навчанню користувачі мають змогу проходити теоретичний матеріал у зручний для них час, переглядати якісні відеоуроки з детальним поясненням технік, брати участь у вебінарах і майстер-класах із досвідченими шеф-кухарями, виконувати практичні завдання та отримувати зворотний зв’язок від викладачів. Все це можливо без необхідності фізичної присутності в аудиторії, що особливо актуально для людей, які живуть у віддалених регіонах або мають обмежений час для навчання. Онлайн-курси також дозволяють суттєво економити час і кошти на дорогу, проживання, навчальні матеріали, що робить такий формат особливо привабливим для широкої аудиторії користувачів, включаючи як початківців, так і професіоналів [2].

Кулінарія як навчальна дисципліна поєднує в собі потребу у візуальному, теоретичному й практичному підході. Саме тому сучасні цифрові платформи, які надають можливість навчатися кулінарії онлайн, повинні бути не просто сховищем відео або текстів, а комплексними інтерактивними системами. Вони мають забезпечити інтуїтивно зрозумілий інтерфейс, який дозволить користувачеві легко орієнтуватися у великому обсязі контенту. Важливо також, щоб навчальні матеріали були подані у форматі, зручному для різних типів користувачів –

наприклад, у вигляді покрокових інструкцій, інтерактивних елементів, практичних вправ.

Особливу роль відіграє система перевірки знань – тести, автоматизовані перевірки, а також можливість отримання фідбеку від викладачів. Без цього процес засвоєння матеріалу буде одностороннім і менш ефективним. Платформа має забезпечити й підтримку двосторонньої комунікації, зокрема через чати, форуми або систему повідомлень. Ще одним важливим чинником є зручна організація контенту – логічна структура курсів, поділ на модулі, прогрес-бар, відстеження успішності студента. Все це створює позитивний користувацький досвід.

Крім того, цифрове середовище має підтримувати персоналізоване навчання – адаптацію курсу під індивідуальні потреби студента, можливість обирати темп навчання, отримувати рекомендації на основі результатів тестування або вже пройденого матеріалу. Важливо також передбачити ефективний механізм оцінювання практичних навичок. У випадку кулінарії це може бути завантаження фото або відео виконаної страви, оцінювання її викладачем за чіткими критеріями, а також організація відеозустрічей, під час яких учень може продемонструвати свої навички в реальному часі [3]. Усе це робить онлайн-освіту з кулінарії не просто можливою, а й ефективною та результативною.

1.2 Цільова аудиторія

Програмна система розрахована на широку аудиторію. До неї входять як початківці, що хочуть навчитися готувати для себе або родини, так і досвідчені аматори, які прагнуть розширити свої знання та спробувати нові кулінарні техніки. Крім того, система буде корисною для професіоналів, які бажають пройти підвищення кваліфікації або ознайомитися з новими напрямками в гастрономії. Завдяки зручному та інтуїтивному інтерфейсу, навчатися зможуть люди різного віку, досвіду та рівня технічної підготовки. Таким чином, програмна система сприяє популяризації кулінарного мистецтва та створює доступне освітнє середовище, яке враховує індивідуальні потреби кожного користувача незалежно від його початкових знань.

1.3 Аналіз існуючих аналогів

Щоб визначити сильні й слабкі сторони сучасних онлайн-рішень для навчання кулінарії, було проаналізовано три відомі платформи, які пропонують кулінарні курси або матеріали: BBC Good Food, MasterClass і ChefSteps. Кожна з них має свій підхід до подачі інформації, аудиторію та набір функцій. Розглянемо їх детальніше.

BBC Good Food (рис. 1.1) — це один із найпопулярніших кулінарних сайтів у світі. Він надає величезну базу рецептів, порад, технік приготування страв та натхнення для щоденного готування. Контент публікується професійними кухарями та редакторами BBC. Платформа орієнтована переважно на самостійне освоєння навичок без явної структури навчального курсу, тому більше підходить для повсякденного використання.

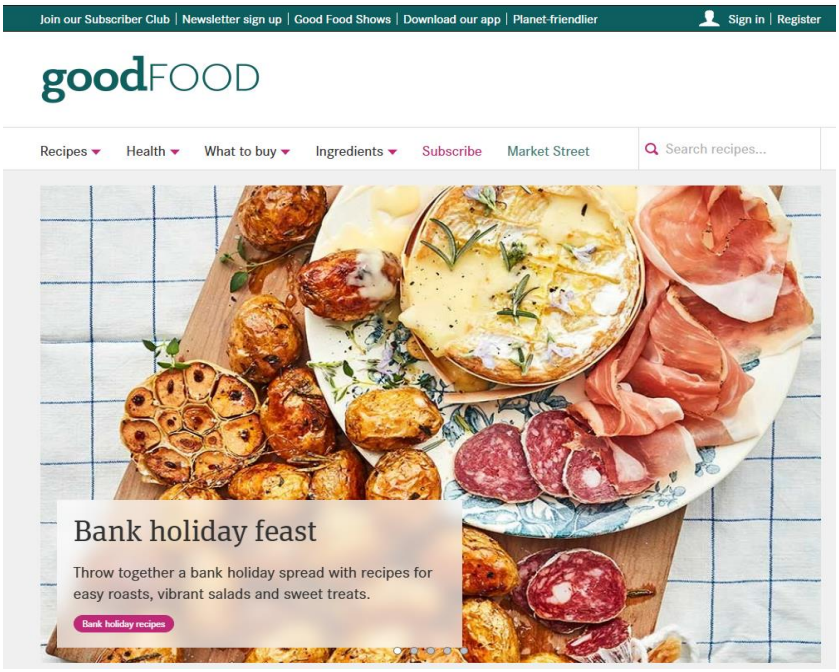

Рисунок 1.1 – Кулінарний сайт BBC Good Food (за даними [4])

Переваги:

- велика база безкоштовних рецйептів та порад;
- наявність фотографій, відео та інструкцій для кожної страви;
- регулярне оновлення контенту;
- проста та інтуітивнозрозуміла навігація.

Недоліки:

– відсутність структурованих курсів;
– немає перевірки знань користувачів;
– відсутність взаємодії з викладачами або іншими користувачами;
– платформа доступна лише англійською мовою.

MasterClass (рис. 1.2) — це преміум-платформа, що пропонує відеокурси від відомих експертів у своїх галузях. У сфері кулінарії серед викладачів є відомі шефи, такі як Gordon Ramsay, Thomas Keller, Alice Waters. Курси мають високу якість зйомки, цікаву подачу та поділені на модулі. Це гарний варіант для користувачів, які прагнуть надихнутися та дізнатися секрети від справжніх зірок гастрономії, але не завжди підходить новачкам.

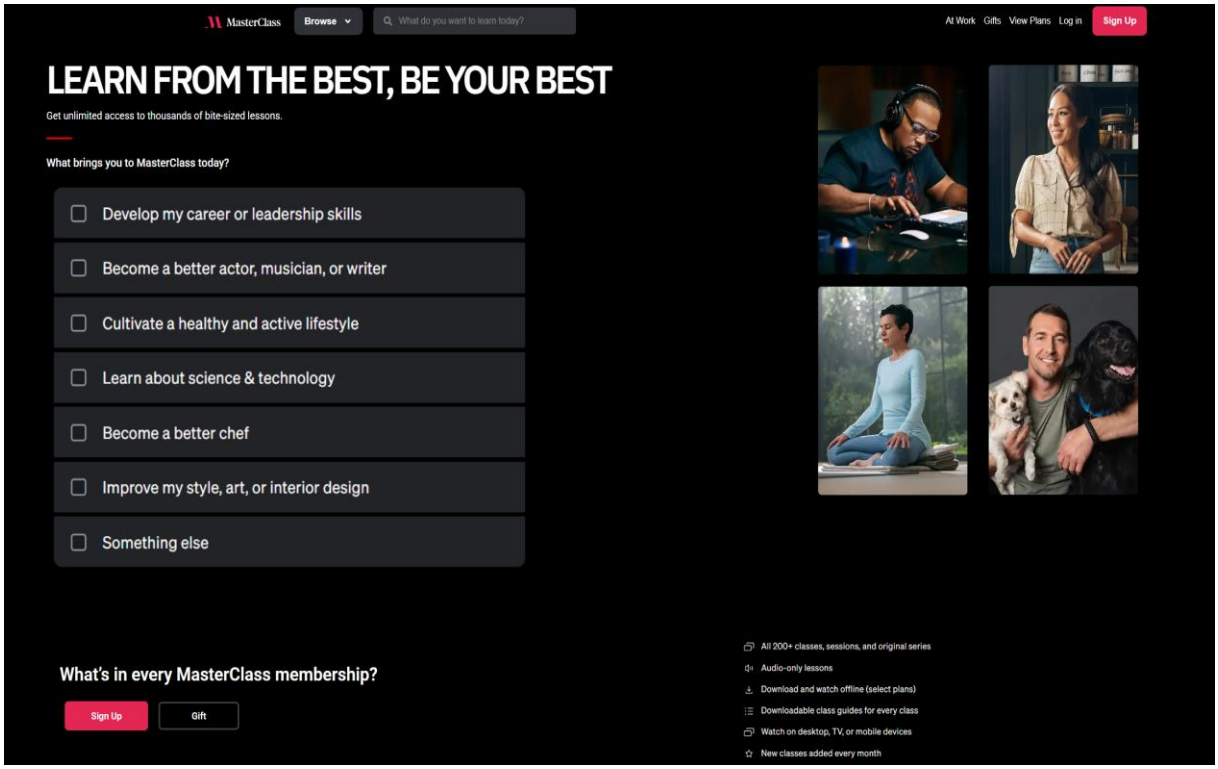

Рисунок 1.2 – платформа для навчання кулінарним навичкам MasterClass (за даними [5])

Переваги:

– високоякісний відеоконтент зі знаменитими шефами;
– чітка структура курсів із логічною подачею матеріалу;
– додаткові робочі зошити для кожного курсу.

Недоліки:

– платна підписка з досить високою вартістю;
– відсутність перевірки практичних знань;
– немає зворотнього зв’язку від викладача;
– платформа доступна лише англійською мовою.

ChefSteps (рис. 1.3) — це платформа, що спеціалізується на сучасних техніках приготування, зокрема молекулярній гастрономії. Вона орієнтована на користувачів із досвідом, які хочуть поглибити свої знання та експериментувати. Курси та рецепти містять детальні пояснення, демонстрації та наукові підходи. ChefSteps також інтегрує свої продукти, наприклад прилад Joule для су-від готування, що трохи звужує цільову аудиторію.

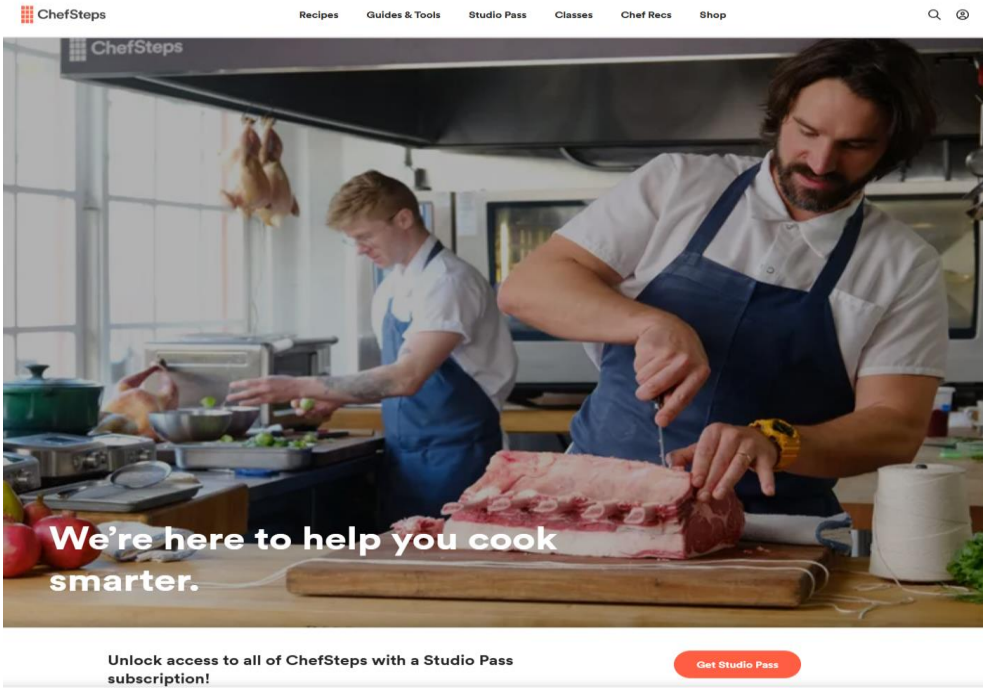

Рисунок 1.3 – платформа для навчання кулінарним навичкам ChefSteps (за даними [6])

Переваги:

– глибокі технічні знання та нестандартні підходи;
– висока якість відео та деталізація процесів;
– професійга орієнтація на просунутих користувачів;
– сучасний, зручний інтерфейс.

Недоліки:

- більшість матеріалів доступні лише за підпискою;
- складність для новачків у кулінарії;
- платформа доступна лише англійською мовою.

1.4 Виявлення проблем

Аналіз існуючих рішень на ринку онлайн-навчання кулінарії дозволив окреслити низку суттєвих проблем, які залишаються нерозв’язаними або не повністю врахованими сучасними платформами. Ці проблеми обумовлюють потребу у створенні нового підходу до навчання кулінарному мистецтву з орієнтацією на локального користувача.

Основні проблеми, виявлені під час аналізу:

- відсутність україномовного контенту. Переважна більшість сервісів пропонують навчальні матеріали англійською мовою, що ускладнює їх сприйняття широким загалом українських користувачів, особливо для початківців;
- відсутність україномовного контенту. Переважна більшість сервісів пропонують навчальні матеріали англійською мовою, що ускладнює їх сприйняття широким загалом українських користувачів, особливо для початківців;
- нестача інтерактивності та адаптивності навчального процесу. Курси часто представлені у вигляді відеолекцій без можливості гнучко реагувати на потреби конкретного учня або коригувати зміст залежно від рівня знань;
- неврахування локального кулінарного контексту. Існуючі курси не адаптовані під особливості української кухні, місцевих інгредієнтів чи кулінарних традицій, що обмежує їх практичну цінність;
- висока вартість та комерційна орієнтованість. Деякі платформи пропонують якісні послуги, але за надто високою ціною, що унеможливлює доступ до навчання широкому колу користувачів;

- відсутність спільноти або підтримки між користувачами. У більшості сервісів відсутній механізм для обміну досвідом, рецептами, обговорення результатів або отримання допомоги від інших учасників;

Сучасні рішення або надто спеціалізовані, або занадто загальні, не враховують культурного та мовного контексту, не підтримують взаємодії та індивідуалізованого підходу до навчання. Це створює значну нішу для розробки програмної системи, яка буде враховувати:

- локалізацію (україномовний інтерфейс та контент);
- ападптивність до рівня користувача;
- наявність практичних занять з оцінюванням;
- можливість зворотнього зв’язку.

1.5 Постановка задачі

Відповідно до попереднього аналізу предметної області та виявлених проблем, метою кваліфікаційної роботи є розробка програмної системи, що забезпечує ефективне онлайн-навчання кулінарному мистецтву для широкої цільової аудиторії. Для досягнення поставленої мети має бути реалізований алгоритм розробки, який забезпечить послідовне та логічно обґрунтоване проходження усіх необхідних етапів створення програмної системи. Такий підхід дозволяє систематизувати роботу, мінімізувати помилки та досягти високої якості кінцевого продукту. Алгоритм розробки включає як теоретичні, так і практичні складові, починаючи з аналізу проблемної області та завершуючи тестуванням реалізованої системи. Нижче наведено основні етапи роботи над проєктом:

- провести аналіз предметної області онлайн-навчання кулінарії, а також вивчити конкурентні рішення;
- виконати концептуальне, інфологічне та логічне моделювання системи (буде створено UML-діаграми, діаграми станів, ER-діаграму бази даних тощо);
- реалізувати програмну систему за допомогою сучасних вебтехнологій, зокрема Razor Pages[7] для фронтенду та ASP.NET Core (C#)[8] для

серверної логіки;

- розробити реляційну базу даних у Microsoft SQL Server[9] з використанням ORM-технології Entity Framework Core[10];
- виконати тестування функціональних можливостей системи, перевірити зручність інтерфейсу, стабільність роботи та відповідність поставленим вимогам.

Основними функціональними можливостями програмної системи є:

- створення та редагування кулінарних курсів і рецептів викладачами;
- організація навчання у вигляді покрокових інструкцій (відео, фото, текстові блоки);
- можливість реєстрації та авторизації користувачів;
- проходження тестування після завершення курсів;
- формування особистого кабінету користувача з історією проходження курсів та результатами;
- коментування курсів та оцінювання рецептів;
- адаптивність інтерфейсу під різні пристрої;
- автоматичне формування електронного сертифікату після проходження навчального модуля;
- ведення обліку досягнень користувача
- ведення статистики переглядів курсів, успішності користувачів та збереження результатів у базі даних;
- реалізація зворотного зв’язку з користувачем, можливість залишати відгуки або звертатися до технічної підтримки.

Передбачається, що викладачі зможуть додавати нові рецепти, вказувати інгредієнти, час приготування, рівень складності тощо. Для користувачів система має бути інтуїтивно зрозумілою, з мінімальною кількістю кліків для доступу до потрібної інформації.

База даних проєкту повинна зберігати інформацію про:

- навчальні модулі (тематика, викладач, контент, дата публікації);

- користувачів (ПІБ, e-mail, тип облікового запису, дата реєстрації, результати курсів);
- тестування та відповіді;
- коментарі та рейтинги;
- список досягнень учнів;
- статистику переглядів, проходження курсів та успішності.

Проєктна технологія:

- frontend: Razor Pages;
- backend: ASP.NET Core (C#);
- база даних: Microsoft SQL Server;
- ORM: Entity Framework Core.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

2.1 Функціональні вимоги

Програмна система повинна надавати користувачам можливість реєстрації та авторизації на платформі, забезпечуючи розмежування прав доступу залежно від ролі користувача. Ролі користувачів включають звичайних користувачів (студентів) та викладачів. Викладачі мають доступ до створення та редагування курсів і рецептур, а студенти – до перегляду матеріалів, курсів та участі в тестах і отриманні результатів.

Система повинна дозволяти викладачам створювати, редагувати, переглядати та видаляти кулінарні курси. Курси можуть містити не лише текстові матеріали, а й відеоуроки, зображення, інструкції з приготування страв, а також інтерактивні компоненти для практичного навчання. Для кожного курсу повинна бути надана можливість вказати рівень складності, тривалість проходження, тематику та можливість фільтрувати курси за цими параметрами.

Користувачі мають мати можливість сортувати доступні курси за різними критеріями: тематикою, рівнем складності, тривалістю навчання та популярністю. Система повинна забезпечувати потужний механізм пошуку, що дозволяє користувачам швидко знаходити необхідні курси та рецепти за ключовими словами, що можуть бути присутніми у заголовках або описах курсів.

Після реєстрації на курсі студент отримує доступ до навчальних матеріалів та завдань. Кожен курс повинен містити інтерактивні тестування для перевірки знань та оцінки рівня засвоєння матеріалу. Результати тестів автоматично зберігаються в особистому кабінеті студента та можуть бути використані для формування статистичних даних про успішність учня, на основі яких система зможе пропонувати рекомендації щодо подальшого навчання.

Після завершення курсу студент має отримати підтвердження про успішне проходження у вигляді сертифікату або досягнення, яке буде відображене в його особистому кабінеті. Система повинна автоматично генерувати досягнення на основі результатів тестів, практичних завдань та часу, витраченого на проходження курсу.

2.2 Нефункціональні вимоги

Інтерфейс користувача повинен бути адаптивним, що гарантує комфортне використання системи як на мобільних пристроях, так і на персональних комп'ютерах. Це забезпечить максимальну зручність для користувачів, оскільки вони зможуть навчатися у будь-якому зручному для них форматі, незалежно від типу пристрою чи розміру екрану. Інтерфейс повинен автоматично підлаштовуватися під розмір екрану, а також бути інтуїтивно зрозумілим і легким у використанні, навіть для людей без спеціальних технічних навичок.

Система повинна демонструвати високу продуктивність та швидкий доступ до навчальних матеріалів. Навігація по курсам, завданням та відеоматеріалам повинна бути без затримок. Особливо важливо це для роботи з відео та інтерактивними тестами, де кожен етап завантаження або перевірки повинен бути виконаний за кілька секунд.

Архітектура програмного забезпечення повинна бути масштабованою. Це означає, що з розвитком проєкту система повинна мати можливість без суттєвих змін у коді чи базі даних легко додавати нові курси, категорії, а також розширювати базу користувачів. Така масштабованість важлива для забезпечення стабільної роботи системи при зростанні кількості користувачів або матеріалів.

Безпека є важливим аспектом при розробці системи. Персональні дані користувачів повинні бути надійно захищені від несанкціонованого доступу за допомогою сучасних механізмів шифрування. Для гарантування безпеки необхідно реалізувати чітке розмежування прав доступу для кожної ролі. Викладачі повинні мати обмежений доступ до навчальних матеріалів і статистики, а студенти – лише до своїх результатів та матеріалів курсів, на які вони записалися.

Система повинна забезпечити цілісність даних, тобто інформація про курси, профілі користувачів, відповіді на тести та результати навчання повинна зберігатися коректно і без втрат навіть у випадку збоїв у роботі системи. Це досягається шляхом використання надійних методів резервного копіювання та відновлення даних.

2.3 Вимоги до платформи

Онлайн школа кулінарного мистецтва реалізується у вигляді вебзастосунку, доступного через основні веб-браузери, такі як Google Chrome, Mozilla Firefox, Microsoft Edge та Safari. Застосунок повинен підтримувати сучасні технології веб-розробки, такі як HTML5, CSS3 та JavaScript, щоб забезпечити ефективну роботу з мультимедійним контентом та інтерактивними елементами. Для забезпечення зручності користувачів вебзастосунок має бути адаптивним, що дозволить здійснювати навчання не лише через персональні комп'ютери, а й за допомогою планшетів і смартфонів.

У майбутньому може бути розроблена мобільна версія додатку для операційних систем Android та iOS. Мобільна версія повинна підтримувати мінімальну версію API 21 для Android (Android 5.0 Lollipop) і iOS 10 для сумісності з більшістю сучасних пристроїв. Це дозволить забезпечити доступ до навчальних матеріалів і курсів для ще ширшої аудиторії користувачів, що надасть їм змогу навчатися в будь-якому місці й у будь-який час.

2.4 Технологічні вимоги

Серверна частина програмної системи онлайн-навчання кулінарному мистецтву реалізується з використанням мови програмування C# у поєднанні з фреймворком ASP.NET Core. Це сучасна, потужна та гнучка технологія, яка забезпечує високу продуктивність, безпеку, масштабованість і стабільність застосунку. Застосування ASP.NET Core дозволяє реалізовувати складну бізнес-логіку, налаштовувати різні рівні доступу, керувати сесіями користувачів, а також забезпечувати гнучку структуру архітектури проєкту. Крім того, ASP.NET Core легко інтегрується з різноманітними зовнішніми сервісами, API або мікросервісами, що дає можливість розширення функціональності системи в майбутньому.

Клієнтська (frontend) частина буде створена з використанням Razor Pages — підходу, який поєднує переваги серверного рендерингу з можливістю часткового оновлення вмісту без повного перезавантаження сторінки. Razor Pages дозволяють

більш гнучко взаємодіяти з сервером, відображати актуальні дані в реальному часі та створювати зручний для користувача інтерфейс. Цей підхід спрощує розробку, покращує продуктивність і зменшує обсяг зайвого коду, що особливо важливо для підтримки масштабованого та зрозумілого проєкту.

У якості основи для зберігання даних використовується SQL Server 2019 — перевірена часом потужна реляційна система управління базами даних (СУБД), яка забезпечує високу надійність, масштабованість та продуктивність при обробці великих обсягів інформації. SQL Server підтримує складні запити, транзакції, механізми безпеки, індексацію та інші інструменти для роботи з даними, що особливо важливо для навчального застосунку з великою кількістю користувачів та контенту.

Для доступу до бази даних застосовується Entity Framework Core (EF Core) — ORM (Object-Relational Mapping) бібліотека, яка дозволяє розробникам працювати з базою даних через об’єктно-орієнтовану модель. EF Core підтримує LINQ-запити, міграції структури бази, lazy loading, зв’язки між таблицями та інші інструменти, які значно прискорюють розробку та зменшують кількість помилок. Завдяки цьому інструменту можна ефективно поєднати логіку бізнес-рівня з даними, забезпечуючи гнучкість та простоту підтримки коду.

Таким чином, обрані технології — ASP.NET Core, Razor Pages, SQL Server 2019 та EF Core — дозволяють побудувати продуктивну, надійну, адаптивну та розширювану програмну систему, яка буде відповідати сучасним стандартам та забезпечить комфортне використання як для викладачів, так і для учнів.

3 АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ

3.1 UML проєктування ПЗ

Для візуалізації функціональності програмної системи онлайн-навчання кулінарному мистецтву було використано мову моделювання UML (Unified Modeling Language). Цей стандарт дозволяє ефективно описати як структуру системи, так і сценарії її використання з боку кінцевих користувачів. Використання UML є важливим етапом розробки, оскільки надає можливість чітко і наочно передати функціональність системи. Моделювання дає змогу на ранніх етапах розробки формалізувати вимоги до програмного забезпечення та уникнути непорозумінь на більш пізніх етапах проєкту.

У проєкті було створено діаграму прецедентів, яка демонструє основні взаємодії між користувачами та системою. Вона відображає не тільки доступні функції для кожного типу користувачів, але й взаємодію між ними через програмне забезпечення. Діаграма прецедентів є одним із основних інструментів для аналізу і визначення функціональних вимог до системи. Вона дозволяє відобразити, хто і які операції може виконувати, що допомагає краще зрозуміти ролі користувачів та їхню взаємодію з системою.

Основними користувачами цієї системи є учень і викладач. Учень має можливість пройти кулінарні курси, переглянути навчальні матеріали, проходити тести, отримувати сертифікати, переглядати рейтинг курсів, залишати коментарі, а також звертатися до викладача. Викладач, у свою чергу, відповідає за створення та редагування курсів, наповнення їх мультимедійними матеріалами, перевірку тестів, а також за взаємодію з учнями через внутрішню систему повідомлень.

Діаграма прецедентів (див. рис. 3.1) дає змогу зрозуміти, які функціональні можливості доступні кожному користувачу і які саме дії вони можуть виконувати. Вона також є основою для створення більш детальних вимог і може бути використана як основний інструмент при розробці системи, щоб гарантовано реалізувати всі необхідні можливості для користувачів на етапах проєктування та розробки.

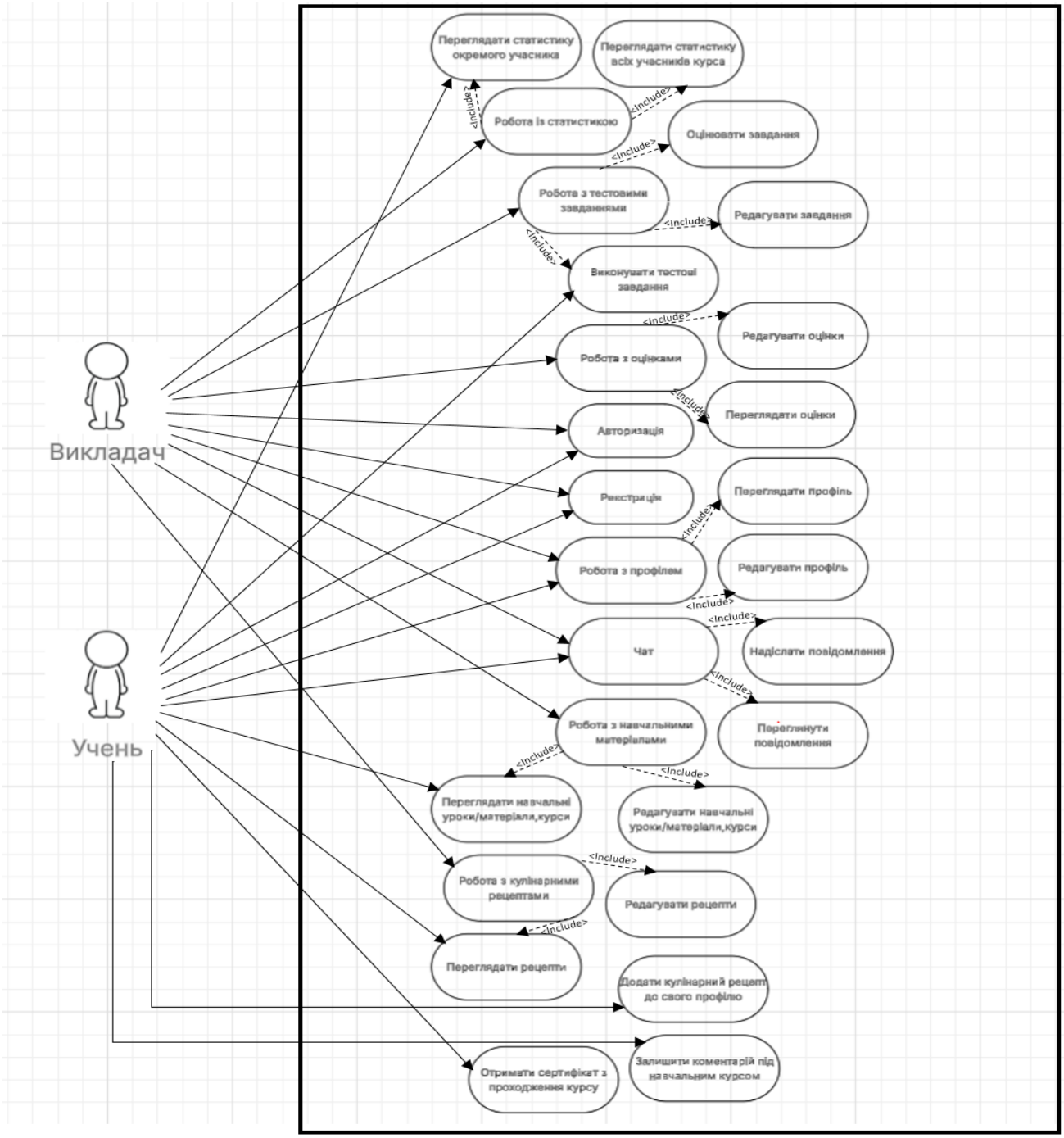

Рисунок 3.1 – UML діаграма прецедентів (рисунок виконано самостійно)

Окрім діаграми прецедентів, для кращого розуміння фізичного розміщення компонентів системи була побудована діаграма розгортання (див. рис. 3.2). Вона ілюструє, як клієнтська частина взаємодіє з серверною частиною і базою даних у хмарному середовищі. Ця діаграма дозволяє побачити, як саме інформація буде передаватися між компонентами системи і як буде забезпечено доступ до даних. Діаграма розгортання є корисним інструментом для розуміння, як функціонує система в цілому і як її компоненти взаємодіють у фізичному середовищі.

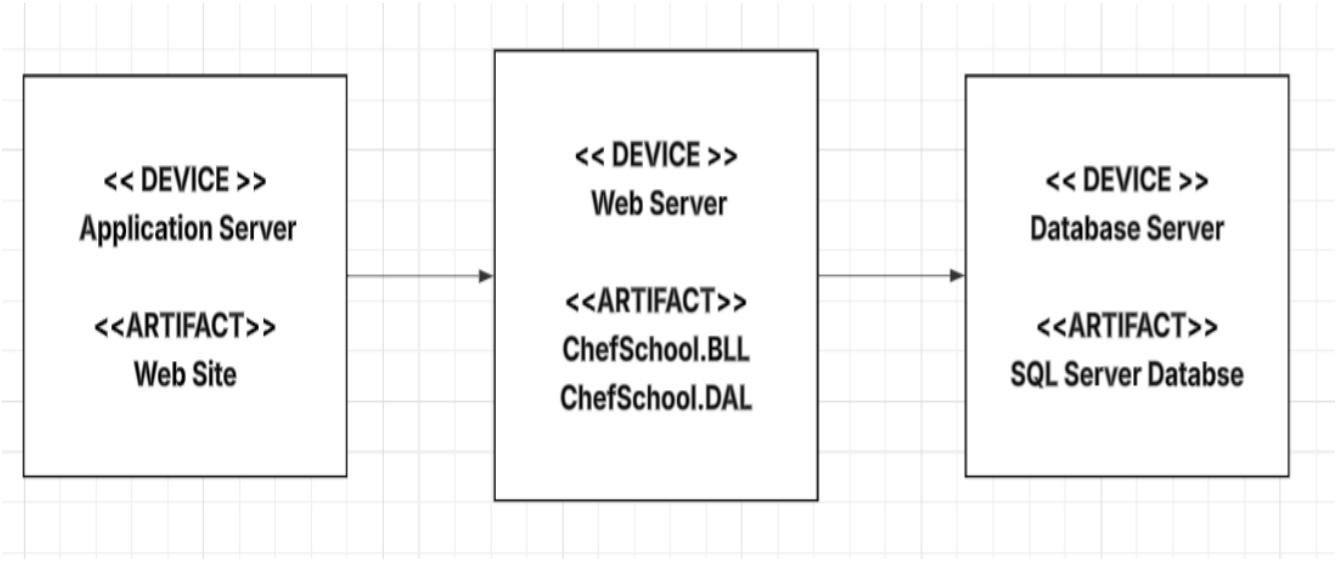

Рисунок 3.2 – UML діаграма розгортання (рисунок виконано самостійно)

Діаграма розгортання показує взаємодію між сервером, клієнтським додатком і базою даних, а також допомагає визначити архітектуру системи. Вона демонструє, як вебдодаток працюватиме на різних типах пристроїв, таких як персональні комп'ютери, планшети і мобільні телефони. Такий підхід дозволяє забезпечити адаптивність верстки, що робить систему доступною для користувачів на різних пристроях.

Ця діаграма допомагає побачити, як компоненти програми з’єднані між собою і як забезпечується взаємодія з користувачами через вебдодаток. Вона також дозволяє виявити можливі проблеми в інфраструктурі на етапі проєктування, що допомагає уникнути технічних проблем на пізніших етапах. Це особливо важливо для масштабування системи та забезпечення її надійної роботи в умовах зростання навантаження на сервери та бази даних.

3.2 Проєктування архітектури ПЗ

При проєктуванні архітектури програмної системи було враховано підходи, що використовуються у подібних розробках, які мають схожі вимоги до збереження даних, навантаження та обслуговування користувачів [11]. Для реалізації програмної системи онлайн-навчання кулінарному мистецтву було

обрано багаторівневу архітектуру на основі принципів Model–View–Controller (MVC[12]) у поєднанні з REST API [13]. Такий підхід дозволяє досягти високої гнучкості та розширюваності системи, а також забезпечує зручну та ефективну взаємодію між клієнтською та серверною частинами. Використання MVC дозволяє чітко розділити відповідальності між різними компонентами системи, що робить код більш зрозумілим, підтримуваним і масштабованим.

Клієнтська частина реалізована з використанням технології Razor Pages, що дає можливість створювати динамічні веб-сторінки з інтерактивним інтерфейсом, адаптованим під різні типи пристроїв — від десктопних до мобільних. Це важливий аспект, оскільки багато користувачів можуть отримувати доступ до онлайн-курсу з різних пристроїв, тому інтерфейс повинен бути максимально зручним і пристосованим під розміри екранів різних типів. Razor Pages дозволяє розробляти сучасні веб-додатки з мінімальною кількістю коду, завдяки чому забезпечується висока продуктивність і швидкість завантаження сторінок. Клієнтська частина відповідає за відображення всіх основних функцій системи, таких як перегляд курсів, взаємодія з мультимедійним контентом (відео, зображення, інструкції), проходження тестів, а також за роботу з особистим кабінетом користувача. Такий підхід дозволяє забезпечити ефективну та інтуїтивно зрозумілу взаємодію користувачів із системою.

Серверна частина побудована за допомогою ASP.NET Core, потужного фреймворку, який забезпечує високу продуктивність, безпеку та масштабованість. Кожен запит користувача обробляється відповідним контролером, що дозволяє чітко відокремити логіку обробки запитів від бізнес-логіки додатка. Контролери викликають сервіси бізнес-логіки, які реалізують основні функції системи, такі як управління курсами, обробка тестових завдань і взаємодія з користувачами. Для доступу до бази даних використовуються репозиторії, які абстрагують деталі роботи з базою даних, що дає змогу підтримувати високий рівень гнучкості та зручності при роботі з даними.

Для збереження даних використовується SQL Server, що є надійною і високопродуктивною базою даних, здатною забезпечити збереження та обробку

великих обсягів інформації. Для роботи з базою даних використовуються дві основні технології: Entity Framework Core і Dapper[14]. Entity Framework Core є об’єктно-реляційним маппером (ORM), який дозволяє працювати з базою даних на високому рівні абстракції, що значно спрощує розробку. Dapper, у свою чергу, є мікро-ORM, який дає можливість виконувати складні SQL-запити з високою продуктивністю, що особливо корисно для виконання складних або оптимізованих запитів до бази даних. Комбінація цих двох технологій дозволяє досягти оптимального балансу між зручністю роботи з базою даних і високою швидкістю виконання запитів.

Дані зберігаються в централізованій базі даних, до якої мають доступ тільки авторизовані компоненти серверної частини. Це дозволяє забезпечити високу безпеку системи, адже тільки авторизовані викладачі можливість змінювати контент. Запити до API обробляються асинхронно, що дозволяє забезпечити високу швидкість обробки запитів і відповіді на них навіть у разі великої кількості одночасних користувачів, що є важливою вимогою для систем, які повинні працювати з великою кількістю користувачів одночасно. Асинхронність запитів дозволяє ефективно використовувати ресурси серверів і запобігає затримкам у взаємодії з користувачами.

Загальна архітектура цієї програмної системи дозволяє легко масштабувати її в майбутньому. Якщо з часом виникне потреба в додаванні нових функціональностей, таких як підтримка відеоконференцій, інтеграція з іншими платформами чи зміни в інтерфейсі, то завдяки правильно побудованій багаторівневій архітектурі система дозволяє здійснювати ці зміни з мінімальними витратами часу та ресурсів. Крім того, архітектура дозволяє ефективно розмежовувати ролі доступу для різних користувачів, таких як учні та викладачі, що гарантує контроль доступу до важливих функцій і даних. Вся система була спроєктована з урахуванням можливостей майбутнього розвитку, що дозволить розширювати її функціональність і підвищувати ефективність роботи в разі збільшення кількості користувачів або обсягу навчальних матеріалів.

3.3 Проєктування бази даних

База даних програмної системи онлайн-навчання кулінарному мистецтву є важливим компонентом, оскільки вона забезпечує збереження облікових записів користувачів, навчального контенту, комунікації між користувачами, оцінювання результатів та відстеження прогресу учнів. Структура бази даних складається з різних сутностей, які взаємодіють між собою, утворюючи логічну та взаємопов'язану модель. Всього в системі реалізовано сімнадцять сутностей, які включають: User, Profile, Course, Category, Skill, VideoMaterial, BookMaterial, ArticleMaterial, Comment, ChatMessage, Test, Question, Answer, TestResult, Recipe, RoleEnum, а також абстрактну сутність BaseMaterial, яка використовується для уніфікації різних типів навчальних матеріалів.

Між сутностями встановлені різні типи зв’язків. Наприклад, кожен User (користувач) має один Profile (профіль), але профіль може бути підписаний на багато різних Course (курсів), що забезпечує гнучкість у моделюванні взаємодії між користувачами та курсами. Крім того, кожен Course може містити різноманітні типи матеріалів, такі як VideoMaterial, BookMaterial та ArticleMaterial, які є основою навчального контенту. Кожен курс також може бути пов’язаний з Recipe (рецептами), Test (тестами) та Comment (коментарями), що дозволяє організувати навчальний процес за допомогою різних форм контенту. Зв'язки між матеріалами курсу та результатами тестів організовані через сутності TestResult та Answer, що дозволяє відстежувати успішність користувачів і зберігати дані про результати їхнього навчання.

Для забезпечення структурованості та логічної цілісності даних база даних була спроєктована за підходом Database First. Спочатку було створено ER-діаграму (див. рис. 3.3), що чітко відображає зв’язки між усіма сутностями та їхні атрибути, що дозволило оптимально спланувати структуру зберігання даних. Після цього, на основі діаграми, були згенеровані моделі в коді, що дозволило спростити подальший процес розробки і зменшити ймовірність помилок у реалізації. Такий підхід дозволив спершу визначити всі можливі зв’язки між сутностями, врахувати всі аспекти нормалізації та уникнути надмірності в базі даних. Це також дозволяє

забезпечити логічну цілісність даних, що є важливим аспектом для підтримки коректної роботи програми та уникнення проблем, пов'язаних із пошкодженням або втратами даних.

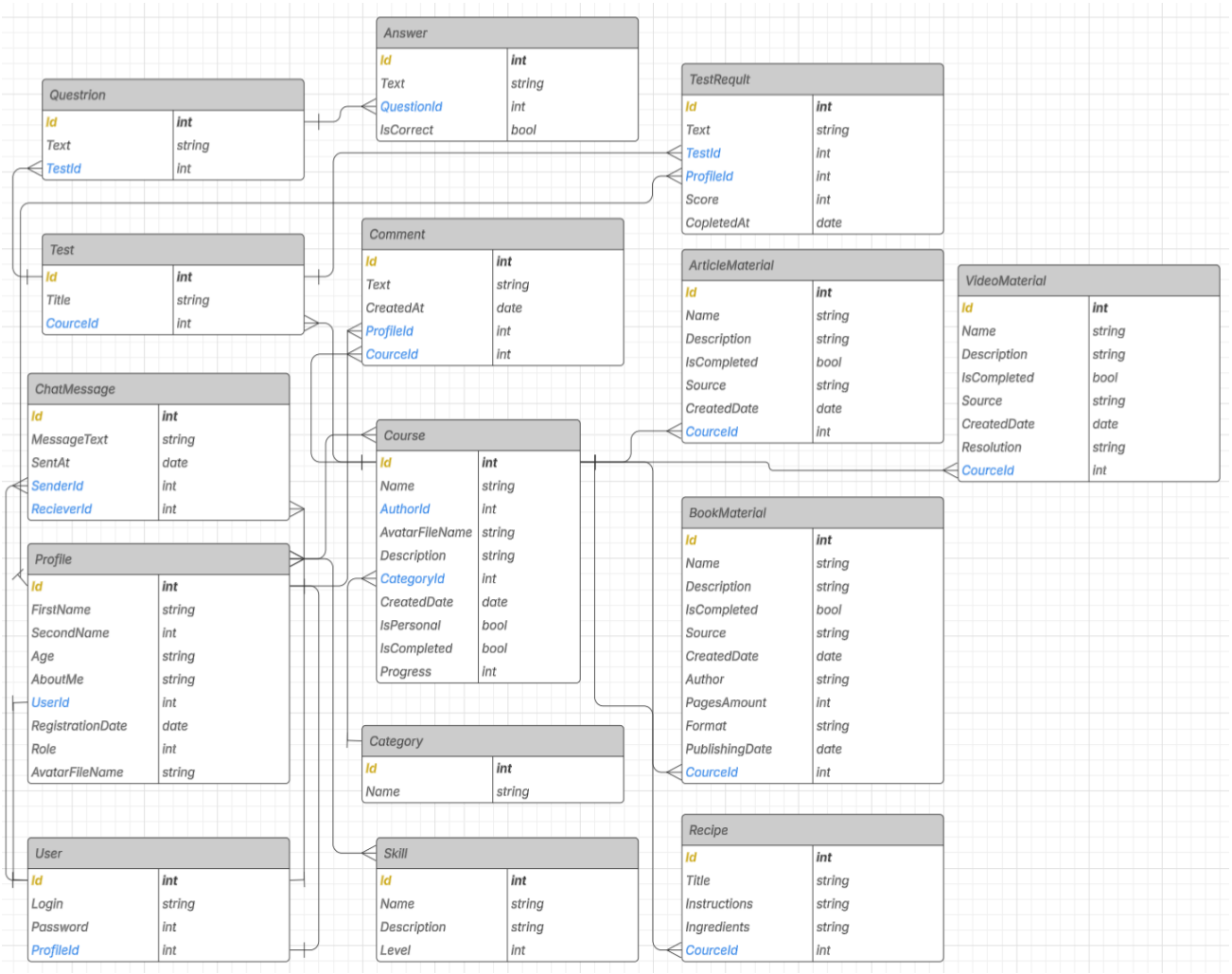

Рисунок 3.3 –ER діаграма сутностей (рисунок виконано самостійно)

Структура бази даних спроєктована з урахуванням принципів нормалізації, що дозволяє зберігати дані без зайвих дублювань і забезпечує ефективне зберігання та обробку запитів. Нормалізація дозволяє досягти не лише зниження надмірності, але й полегшує підтримку та розширення системи в майбутньому. Така архітектура забезпечує високий рівень ефективності при виконанні запитів, адже всі сутності логічно організовані, а зв’язки між ними забезпечують зручний доступ до необхідної інформації.

Можливість масштабування бази даних та її розширення є важливою перевагою такої моделі. У разі потреби можна без проблем додавати нові сутності

або змінювати існуючі, не порушуючи основну структуру. Це дозволяє в подальшому додавати нові курси, додаткові матеріали, нові типи тестів або навіть нові функціональні можливості для користувачів, зберігаючи при цьому високу продуктивність і надійність системи. Розширення функціональності також може включати додавання нових зв'язків між сутностями, що дозволяє інтегрувати додаткові послуги чи інтерактивні елементи, наприклад, відеоконференції, нові форми комунікації або додаткові методи оцінювання успішності учнів.

Загалом, така модель бази даних не лише відповідає вимогам до нормалізації та ефективності зберігання даних, але й надає потужну основу для подальшого розвитку програмної системи, її масштабування та адаптації до нових вимог або змін в умовах онлайн-освіти.

Сутність User відповідає за зберігання основної інформації про зареєстрованого користувача. Кожен користувач має унікальний ідентифікатор (Id), унікальне ім'я користувача (Username), адресу електронної пошти (Email) і хеш пароля (PasswordHash), що гарантує безпечне зберігання облікових даних. Додатково, поле Role дозволяє визначити роль користувача у системі — студент або викладач. Поле IsActive відповідає за активність облікового запису, а RegisteredAt зберігає дату реєстрації. Зв'язки з іншими сутностями включають один-до-одного зв'язок із Profile, а також один-до-багатьох із Course, Comment, TestResult і ChatMessage. Таким чином, один користувач може мати лише один профіль, але водночас бути автором декількох курсів, залишати численні коментарі, проходити тести та брати участь у чатах.

Сутність Profile створена для розширення інформації про користувача. Вона містить ідентифікатор (Id), зовнішній ключ UserId, який вказує на відповідного користувача, повне ім'я (FullName), шлях до аватару (AvatarUrl), коротку біографію або опис (Bio), рівень досвіду (ExperienceLevel) та список навичок. Навички зберігаються через окрему проміжну таблицю зв'язку з сутністю Skill. Таким чином, Profile виступає в ролі детального представлення особистості користувача, особливо важливого для викладачів, оскільки учні можуть обирати курси на основі досвіду та навичок викладача.

Ключовою сутністю, що визначає зміст навчання, є Course. Вона містить заголовок (Title), опис (Description), посилання на викладача (InstructorId, зовнішній ключ до User), а також посилання на категорію курсу (CategoryId). Крім того, вказано дату створення (CreatedAt) і статус публікації (IsPublished). Курс пов’язаний із низкою інших сутностей: VideoMaterial, BookMaterial, ArticleMaterial, Recipe, Test та Comment. Це дозволяє організувати курс як багатокомпонентну навчальну одиницю, яка включає теоретичний і практичний контент, перевірку знань та взаємодію з іншими користувачами.

Серед навчальних матеріалів окрему роль відіграє VideoMaterial, що включає відеоуроки, пов’язані з курсом. Ця сутність містить заголовок, опис, посилання на відео (VideoUrl), тривалість (Duration) і порядок відображення у курсі (OrderIndex). Всі відеоматеріали мають зовнішній ключ до курсу, до якого належать.

BookMaterial відображає навчальні книги, рекомендовані в рамках курсу. Вона зберігає назву книги, автора, видавництво, рік видання та посилання на завантаження. Це джерело додаткових знань, які доповнюють основний навчальний процес.

ArticleMaterial виконує функцію представлення окремих статей із корисною інформацією. Вона включає заголовок, текст вмісту, дату публікації та автора. Як і інші матеріали, вона прив’язана до курсу через зовнішній ключ.

Для тематичного структурування навчального контенту використовується сутність Category, яка класифікує курси за напрямами. Вона містить назву та опис, а кожна категорія може охоплювати багато курсів.

Сутність Skill визначає конкретні кулінарні навички. Вона має назву і опис, і пов’язується з Profile через таблицю багато-до-багатьох. Це дозволяє кожному користувачу мати декілька навичок і навпаки — кожна навичка може бути притаманна багатьом користувачам.

Практичні завдання у вигляді приготування страв реалізуються через сутність Recipe. Вона включає заголовок, опис, список інгредієнтів, покрокову інструкцію (Steps), приблизний час приготування (EstimatedTime) і зображення. Рецепт завжди належить до певного курсу.

Перевірка знань здійснюється за допомогою сутності Test, яка містить заголовок тесту та загальну кількість балів. Вона прив’язана до курсу й має зв’язок із сутністю Question, яка представляє окреме запитання у тесті. Кожне запитання містить текст, тип (наприклад, множинний вибір), а також порядок розміщення.

Кожне питання має відповідні варіанти відповідей, які описані в сутності Answer. Тут вказується текст відповіді та ознака правильності (IsCorrect). Це дозволяє будувати повноцінні тести з перевіркою знань.

Коли користувач проходить тест, його результат зберігається в сутності TestResult. Тут фіксується, хто проходив тест (UserId), який саме тест (TestId), отриманий бал (Score), дата проходження (PassedAt) і позначка, чи був тест складений успішно (IsPassed).

Для взаємодії користувачів у контексті курсу чи матеріалів використовується сутність Comment, що дозволяє залишати коментарі. Коментар пов’язаний із користувачем і курсом, містить текст повідомлення та дату його публікації.

Нарешті, комунікація в режимі реального часу між користувачами реалізується через сутність ChatMessage. Повідомлення має відправника (SenderId) і отримувача (ReceiverId), текст повідомлення, дату надсилання (SentAt) і стан прочитання (IsRead). Така структура дозволяє будувати повноцінну систему повідомлень між студентами та викладачами.

Загалом, модель бази даних є логічно структурованою, підтримує гнучке додавання нового контенту, зберігає відносини між учасниками навчального процесу і надає всі необхідні можливості для реалізації навчання, тестування та взаємодії в рамках кулінарної онлайн-платформи.

3.4 Архітектура серверної частини

Серверна частина системи онлайн-навчання кулінарному мистецтву ChefSchool побудована відповідно до трирівневої архітектури, яка забезпечує логічне розділення функціональності між окремими шарами та сприяє масштабованості й зручності в обслуговуванні. Уся логіка системи розділена на

рівень доступу до даних, рівень бізнес-логіки та презентаційний рівень, що дозволяє ефективно управляти як простими, так і складними операціями.

Рівень доступу до даних реалізований із використанням ORM-бібліотеки Dapper, яка дозволяє працювати з базою даних на основі SQL-запитів, зберігаючи при цьому продуктивність і контроль над запитами. Усі операції з даними інкапсулюються в окремих репозиторіях, що забезпечує гнучкість, повторне використання коду та легкість у тестуванні.

На рівні бізнес-логіки реалізується основна функціональність системи: обробка запитів, перевірка правил, реалізація сценаріїв взаємодії між користувачами, курсами, тестами, коментарями, оцінками та рецептами. Цей рівень пов’язаний з DAL через сервіси, що дозволяє централізовано керувати логікою й уникати дублювання коду.

Презентаційний рівень побудовано на основі ASP.NET Web API. Саме тут обробляються HTTP-запити, які надходять із клієнтської частини системи. Запити передаються на обробку в бізнес-рівень, після чого користувач отримує відповідь. Для забезпечення уніфікованого підходу до обробки запитів та помилок використовується базовий клас контролера, що дозволяє централізовано керувати відповідями API.

Окрему увагу в архітектурі приділено безпеці та масштабованості. Для забезпечення авторизації й автентифікації користувачів використовується токен-орієнтований підхід із JWT (JSON Web Token), що дозволяє гнучко керувати правами доступу до ресурсів системи. Завдяки модульності архітектури, у майбутньому можливе легке розширення функціональності, наприклад, шляхом інтеграції нових сервісів, додавання сторонніх API або впровадження механізмів аналітики користувацької активності. Такий підхід сприяє не лише підвищенню якості обслуговування, а й оптимізації процесу розробки та супроводу програмного продукту.

Для візуального уявлення логічної структури та взаємодії компонентів було побудовано UML-діаграму пакетів, яка демонструє основні залежності між трьома рівнями архітектури (див.рис 3.4).

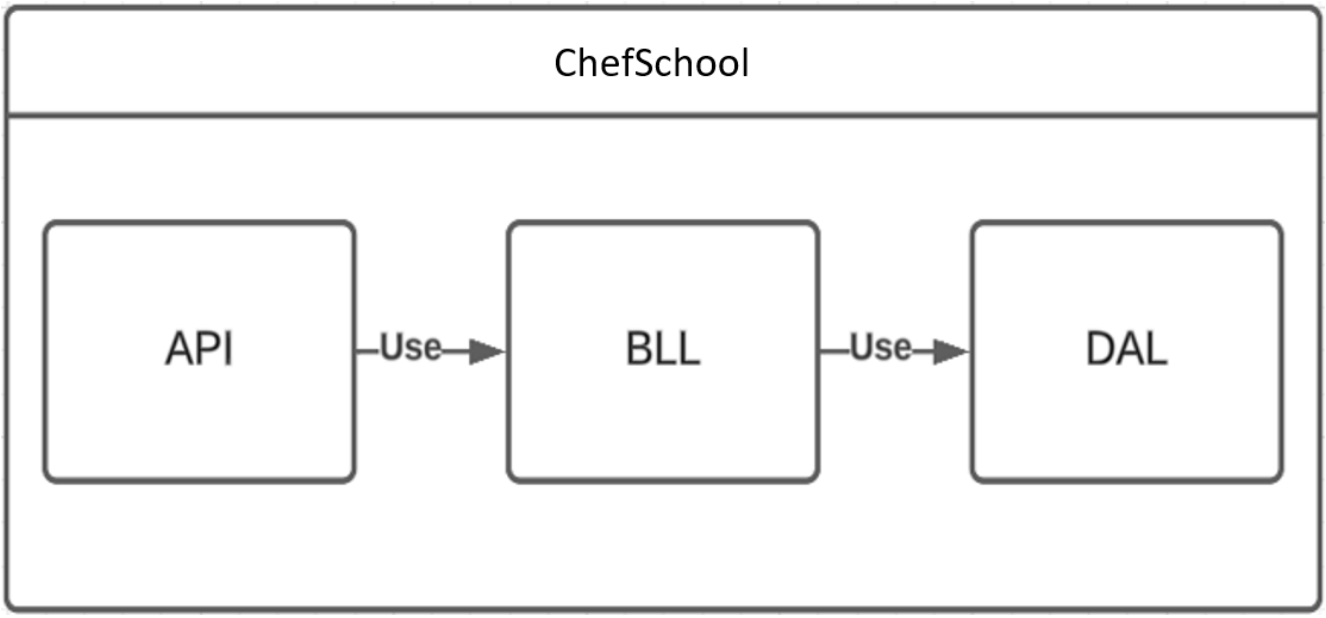

Рисунок 3.4 – UML-діаграма пакетів серверного частини проєкту (рисунок виконано самостійно)

Ця діаграма відображає чітке розмежування відповідальності між рівнями доступу до даних, бізнес-логіки та презентаційним рівнем.

3.5 Приклади найцікавіших алгоритмів та методів

У процесі розробки програмної системи було реалізовано декілька ключових алгоритмів, які забезпечують інтелектуальну обробку даних і підвищують зручність використання системи. Одним із прикладів є механізм хешування паролів користувачів, що гарантує базовий рівень захисту персональної інформації [15]. Для цього використовується алгоритм SHA-256, який перетворює пароль у незворотний хеш-код:

```
public static class HashPasswordHelper
{
    public static string HashPassword(string password)
    {
        using (var sha256 = SHA256.Create())
        {
            var hashedBytes = sha256.
            ComputeHash(Encoding.UTF8.GetBytes(password));
            var hash = BitConverter.
            ToString(hashedBytes).Replace("-", "").ToLower();
            return hash;
        }
```

```
    }
}
```

Іншою важливою частиною логіки є автоматичний розрахунок прогресу проходження курсу. Система визначає відсоткове співвідношення завершених матеріалів до загальної кількості, що дозволяє динамічно відображати користувачу стан виконання:

```
public static class CourseProgressCalculator
{
    public static int CalculateProgress(Course course)
    {
      int totalMaterials = course.Videos.Count + course.Articles.Count
      + course.Books.Count;
        if (totalMaterials == 0) return 0;

        int completedMaterials = course.Videos.Count(v =>
        v.IsCompleted) + course.Articles.Count(a => a.IsCompleted) +
        course.Books.Count(b => b.IsCompleted);

        return (int)((double)completedMaterials / totalMaterials *
        100);
    }
}
```

Ще однією цікавою реалізацією стала система перевірки результатів тестування. Вона дозволяє обробляти список відповідей користувача, порівнювати їх із правильними варіантами і формувати підсумкову оцінку:

```
public class TestChecker
{
    public static int CheckTest(Test test, List<int> userAnswers)
    {
        int correctCount = 0;
        for (int i = 0; i < test.Questions.Count; i++)
        {
            if (test.Questions[i].CorrectAnswerId == userAnswers[i])
                correctCount++;
        }
        return correctCount;
    }
}
```

Після цього можна порівняти загальну кількість питань з правильними відповідями і порахувати оцінку у відцотках.

3.6 Створення UI/UX

У процесі проєктування програмної системи особливу увагу було приділено створенню зручного та зрозумілого користувацького інтерфейсу. На основі вимог цільової аудиторії було спроєктовано основні елементи UI/UX, які орієнтовані на простоту навігації, естетичну привабливість та доступність функціоналу. Було створено макети кількох ключових сторінок: головної сторінки, сторінки курсу та сторінки профілю користувача.

Головна сторінка (див. рис. 3.5) спроєктована як стартова точка взаємодії користувача з платформою. У верхній частині розміщується навігаційна панель, що містить посилання на розділи “Курси”, “Користувачі”, “Контакти” та “Профіль”. Основна частина макета містить привітальний блок із коротким описом призначення системи та кнопкою переходу до навчання. Нижче передбачено секцію з рекомендованими категоріями, які представлені у вигляді інтерактивних карток з посиланням на відповідну категорію.

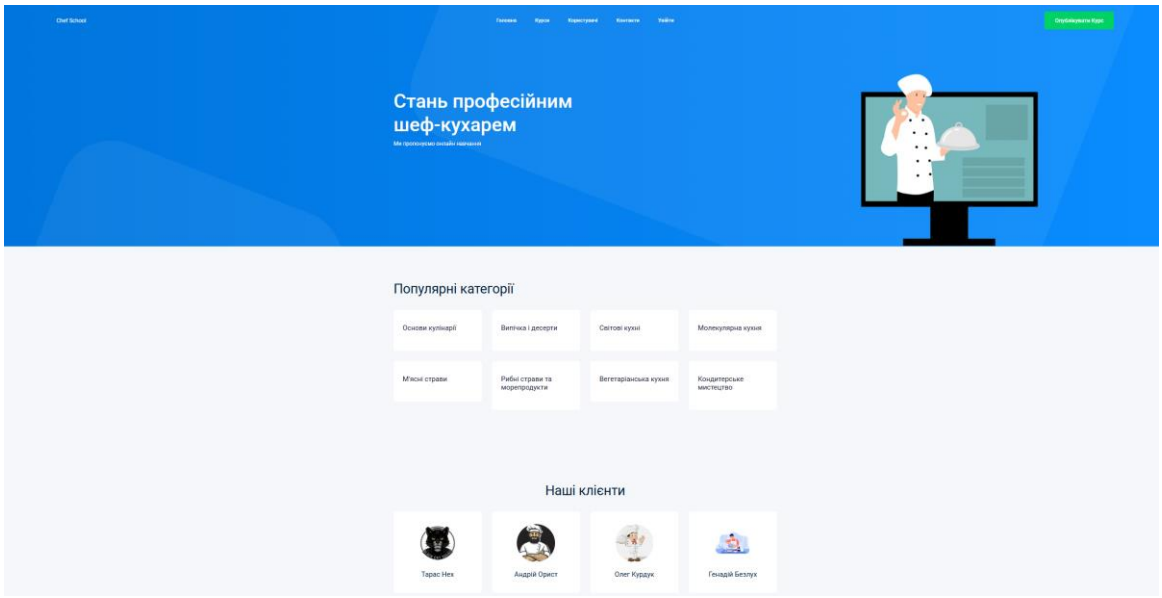

Рисунок 3.5 – Дизайн головної сторінки (скріншот виконано самостійно)

Сторінка курсу (див. рис. 3.6) охоплює всю необхідну інформацію про вибраний курс. Візуальна структура макета включає назву курсу, опис, автора, дату створення, категорію та опис доступних матеріалів (відео, статті, книги).

Нижче розміщено кнопки для запису на курс та редагування (доступні лише автору курсу). Для підвищення динаміки інтерфейсу передбачено мікроанімації: при наведенні кнопки змінюють колір і масштабуються, а повідомлення про успішну дію (наприклад, запис на курс) з'являється з анімацією плавного появлення знизу екрана. Такі ефекти створюють відчуття живої взаємодії та зменшують когнітивне навантаження під час навігації. Також застосовується анімація при завантаженні самої сторінки: основні блоки з'являються поступово за допомогою fade-in ефекту, що робить інтерфейс візуально приємним та сучасним.

Особливу увагу приділено плавності переходів між різними станами елементів — це забезпечує візуальну цілісність і зменшує відчуття різких змін на екрані. Крім того, інтерактивні елементи реагують на дії користувача з мінімальними затримками, що сприяє позитивному досвіду взаємодії. У перспективі передбачається використання бібліотек анімацій (на кшталт Framer Motion або CSS-анімацій) для забезпечення кросбраузерної сумісності, високої продуктивності, а також більш складних сценаріїв анімації, зокрема появи діалогових вікон, спливаючих підказок і контекстних меню.

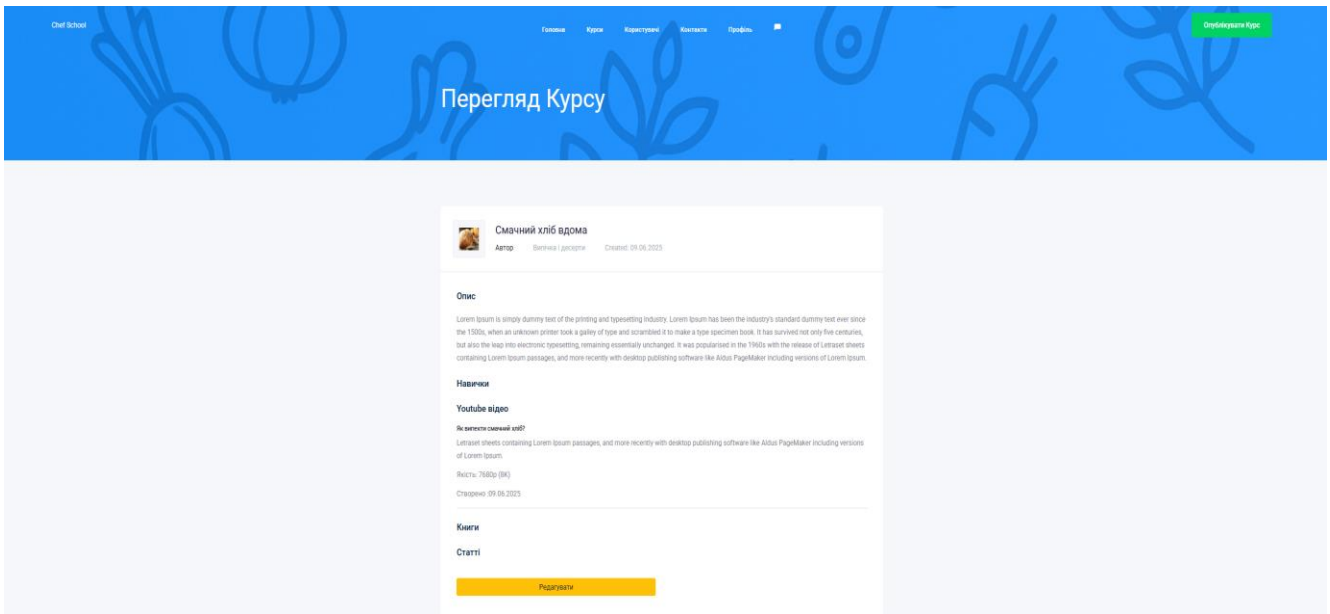

Рисунок 3.6 – Дизайн сторінки курсу (скріншот виконано самостійно)

Сторінка профілю (див. рис. 3.7) користувача спроєктована для зручного перегляду та редагування персональної інформації. У макеті відображаються ім'я,

прізвище, вік, коротка біографія, аватар та роль користувача. Нижче передбачено секцію з курсами, у яких бере участь користувач, із візуальним відображенням прогресу у вигляді шкал. Крім того, відображаються набуті навички та активність у системі.

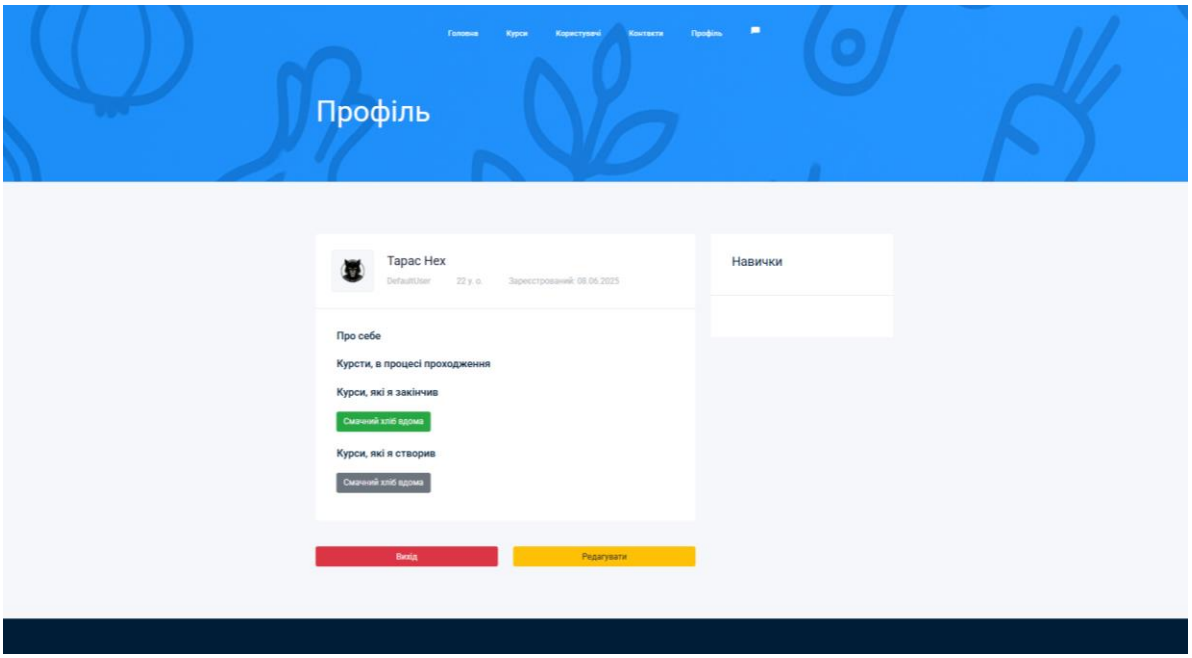

Рисунок 3.7 – Дизайн сторінки профілю (скріншот виконано самостійно)

На даному етапі користувацький інтерфейс програмної системи перебуває у вигляді детально опрацьованих прототипів, створених за допомогою сучасного інструменту проєктування інтерфейсів — Figma. Прототипи охоплюють основні аспекти взаємодії користувачів із системою, включаючи структуру головної сторінки, сторінок курсів, перегляду матеріалів, виконання практичних завдань, проходження тестів, перегляду оцінок, редагування профілю користувача, взаємодії з викладачем тощо. Враховано принципи юзабіліті, логічної організації інформації, доступності інтерфейсу для людей різного віку, зокрема й тих, хто має мінімальний досвід користування сучасними веб-застосунками.

У межах наступного етапу розробки планується реалізувати візуальну частину застосунку безпосередньо у середовищі ASP.NET Core, використовуючи Razor Pages як основну технологію побудови користувацького інтерфейсу. Razor Pages є потужною частиною ASP.NET Core, що дозволяє органічно поєднувати HTML-розмітку з C#-логікою без необхідності використання сторонніх

фреймворків. Це значно полегшує процес підтримки, відлагодження та розширення системи в майбутньому. Кожна сторінка інтерфейсу буде реалізована у вигляді окремої Razor-сторінки з відповідними обробниками подій на сервері, що забезпечить високий рівень безпеки, ефективну роботу з базою даних і плавну інтеграцію із серверною логікою.

Для реалізації дизайну системи буде використано цілу низку сучасних веб-технологій, що забезпечать високу якість, адаптивність та зручність інтерфейсу. Зокрема, для розмітки структури сторінок буде застосовано HTML5, що надає всі необхідні можливості для створення семантичної і зручної для пошукових систем структури веб-сторінок. Для реалізації візуального оформлення та адаптивності на сторінках використовуватимуться сучасні можливості CSS3, зокрема модулі Flexbox і Grid, що дозволяють ефективно розташовувати елементи на сторінці і забезпечувати правильне відображення на різних пристроях, а також CSS-змінні, що спрощують управління стилями на всіх рівнях. Ці технології дають змогу створити не лише красивий, але й зручний для користувача інтерфейс, що адаптується під різні розміри екранів та типи пристроїв.

Для додання інтерактивності інтерфейсу буде використано JavaScript у поєднанні з Razor. Це дозволить забезпечити динамічне перемикання вкладок, реалізацію підказок, валідацію форм і інших елементів користувацького інтерфейсу, що значно підвищить зручність взаємодії з системою. Завдяки цьому користувачі зможуть взаємодіяти з платформою на різних етапах навчання, отримуючи миттєвий зворотний зв'язок і мати можливість без затримок коригувати введені дані.

У випадку, якщо у процесі експлуатації системи виникне потреба в розширенні функціоналу клієнтської частини, в майбутньому можна буде частково інтегрувати компоненти Blazor для більш гнучкої побудови інтерактивних елементів або використовувати популярні JavaScript-фреймворки, такі як React, для розробки окремих модулів. Це дозволить додавати нові функції, зберігаючи при цьому високу продуктивність і зручність користування системою. Зокрема, модулі для статистики або чату можна буде побудувати з використанням цих технологій,

що дозволить інтегрувати їх безпосередньо в існуючу архітектуру, значно покращивши користувацький досвід і можливості платформи.Весь інтерфейс розроблятиметься з дотриманням принципів адаптивного дизайну, що дає змогу забезпечити коректне відображення на різноманітних пристроях — від мобільних телефонів до настільних комп'ютерів. Це дозволить залучити ширшу аудиторію користувачів, які зможуть комфортно навчатися як вдома, так і в дорозі. Особлива увага приділятиметься доступності (accessibility), щоб забезпечити відповідність сучасним стандартам (зокрема WCAG 2.1) і зробити систему придатною для використання людьми з порушеннями зору або іншими обмеженнями.

Проєктуванню інтерфейсу передувала детальна аналітична робота, що включала вивчення потреб кінцевих користувачів. Це дозволило сформувати чіткий перелік основних функціональних блоків, які повинні бути реалізовані в системі. До них відносяться перегляд курсів, можливість фільтрації за категоріями, відстеження особистого прогресу учнів, перегляд відеоуроків, завантаження виконаних завдань у форматі зображень чи відео, система коментарів і повідомлень між учнями та викладачами, а також система оцінювання та тестування. Всі ці функції будуть реалізовані у зручному та інтуїтивно зрозумілому інтерфейсі відповідно до розроблених макетів, що стали частиною UI/UX-дизайну.

Важливим аспектом є забезпечення консистентності інтерфейсу, що дозволить користувачам швидко звикнути до навігації по сайту. Всі візуальні елементи, такі як кнопки, заголовки, індикатори прогресу, блоки контенту, будуть витримані в єдиному стилі, що відповідає візуальній концепції бренду. Це створить відчуття єдності та зручності для користувачів. Крім того, система буде підтримувати дві основні теми оформлення: темну та світлу, що дозволить задовольнити різноманітні вподобання користувачів і підвищить комфортність використання для різних категорій людей.

Таким чином, завдяки використанню Razor Pages, доповненого сучасними фронтенд-технологіями, буде створено функціональний, зручний та адаптивний інтерфейс, що забезпечить відмінний користувацький досвід.

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

4.1 Реалізація моделі курсів із мультимедійним наповненням

Однією з головних складових платформи є можливість створення викладачами навчальних курсів, що включають різноманітний мультимедійний контент: відео, книги, статті, рецепти. Для реалізації цієї логіки було створено базовий клас BaseMaterial, від якого успадковуються такі типи матеріалів, як VideoMaterial, BookMaterial та ArticleMaterial.

Фрагмент класу BaseMaterial:

```
public abstract class BaseMaterial
{
    public int Id { get; set; }
    public string Title { get; set; }
    public DateTime CreatedAt { get; set; }
    public int CourseId { get; set; }
    public Course Course { get; set; }
}
```

Наслідування типів матеріалів:

```
public class VideoMaterial : BaseMaterial
{
    public string VideoUrl { get; set; }
    public TimeSpan Duration { get; set; }
}
public class BookMaterial : BaseMaterial
{
    public string FilePath { get; set; }
    public string Description { get; set; }
}
public class ArticleMaterial : BaseMaterial
{
    public string Content { get; set; }
}
```

Відображення відеоматеріалів за допомогою влаштованого You-Tube плеєра:

```
<div class="single_wrap">
    <h4>Youtube Videos</h4>

    @{
        foreach (var video in @Model.Videos)
        {
            <h6>@video.Name</h6>
```

```
                    <p>@video.Description</p>
                    if (!string.IsNullOrEmpty(@video.Resolution))
                    {
                        <p>Resolution: @video.Resolution</p>
                    }
                    <p>Created :@video.CreatedDate.ToShortDateString()</p>
                    if (Model.IsPersonal)
                    {
                        <iframe class="w-100" height="500"
src="https://www.youtube.com/embed/@video.Source" title="YouTube video
player" frameborder="0" allow="accelerometer; autoplay; clipboard-write;
encrypted-media; gyroscope; picture-in-picture" allowfullscreen></iframe>
                    }
                    if (Model.IsPersonal && !video.IsCompleted)
                    {
                        <a style="margin-top: 15px" asp-
action="CompleteMaterial" asp-route-courseId="@Model.Id" asp-route-
materialId="@video.Id" asp-route-materialTypeId="1" class="btn btn-success
w-30">Mark as Completed</a>
                    }
                    <hr />
                }
            }
        </div>
```

Таким чином, курс динамічно наповнюється різними типами контенту. Кожен тип матеріалу може мати свою візуалізацію на Razor-сторінках. Рисунок 4.1 демонструє приклад сторінки з відеоматеріалом у межах курсу.

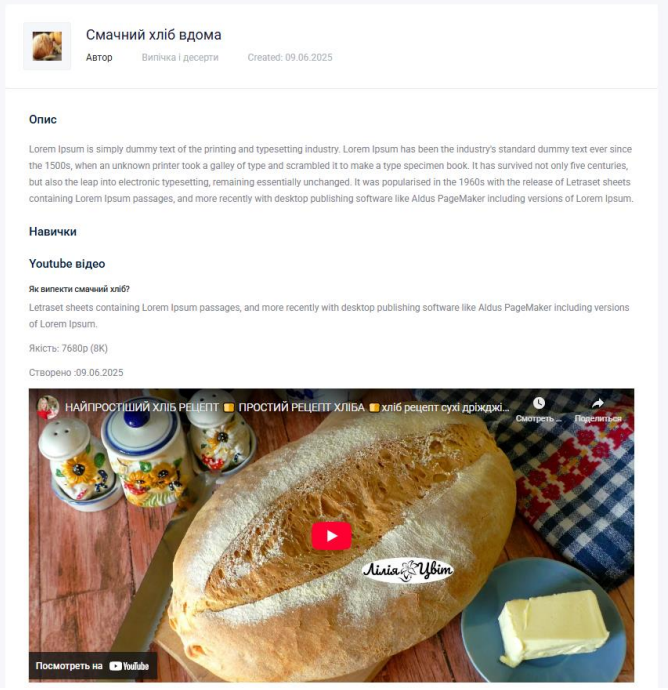

Рисунок 4.1 – Сторінка курсу з відеоматеріалом (скріншот виконано самостійно)

Окрім відео, в якості активності може бути представлений PDF документ.

4.2 Механізм проходження тестування з підрахунком результатів

Для перевірки знань учнів у кожному курсі реалізовано механізм проходження тестів. Кожен тест складається з набору запитань, кожне з яких має кілька варіантів відповідей. Користувач обирає правильні відповіді, після чого система підраховує загальний результат і зберігає його до бази даних.

Фрагмент моделі Test::

```
public class Test
{
    public int Id { get; set; }
    public string Title { get; set; }
    public int CourseId { get; set; }
    public List<Question> Questions { get; set; }
}
```

Підрахунок результатів (сервісна логіка):

```
public int CalculateScore(Test test, Dictionary<int, int>
userAnswers)
{
    int score = 0;
    foreach (var question in test.Questions)
    {
        if (userAnswers.TryGetValue(question.Id, out int answerId))
        {
            if (question.Answers.Any(a => a.Id == answerId &&
a.IsCorrect))
                score++;
        }
    }
    return score;
}
```

Крім того, передбачено можливість повторного проходження тесту, що стимулює користувачів до вдосконалення своїх знань. В історії результатів фіксується дата проходження та набраний бал, що дозволяє аналізувати динаміку успішності. У майбутньому планується розширення функціональності: зокрема, додавання таймера на виконання тесту, обмеження кількості спроб та виведення пояснень до правильних відповідей для підвищення ефективності навчання.

Рисунок 4.2 демонструє інтерфейс з варіантами відповідей та кнопкою надсилання результату.

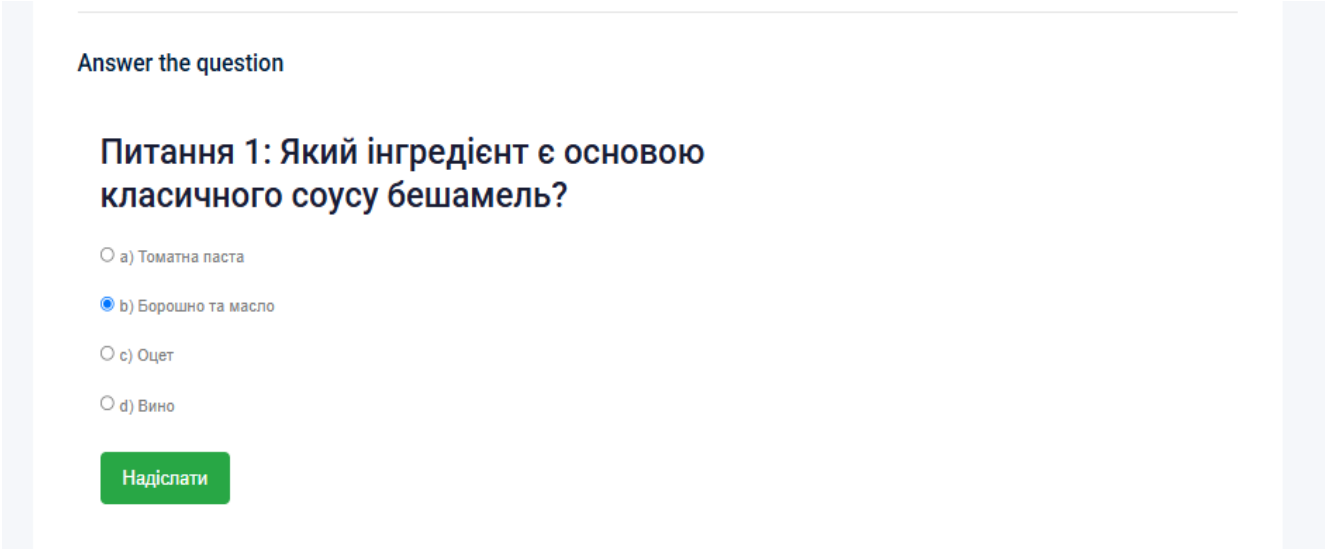

Рисунок 4.2 – Сторінка з варіантами відповіді та кнопкою надсилання результату (скріншот виконано самостійно)

Після проходження тесту результат зберігається у таблицю TestResult, що дозволяє користувачеві переглядати свій прогрес.

4.3 Реалізація внутрішнього чату між учнями та викладачами

Комунікація є важливою частиною навчального процесу, тому в системі реалізовано внутрішній чат, який дає змогу користувачам надсилати один одному повідомлення. Структура таблиці ChatMessage дозволяє зберігати текст повідомлення, дату надсилання, а також інформацію про відправника й отримувача. Нижче наведений код класу повідомлення в чаті ChatMessage та метод надсилання повідомлення користувачеві з подальшим збереженням його до бази даних.

Модель ChatMessage:

```
public class ChatMessage
{
    public int Id { get; set; }
    public int SenderId { get; set; }
    public int ReceiverId { get; set; }
    public string MessageText { get; set; }
    public DateTime SentAt { get; set; }
}
```

Метод надсилання повідомлення:

```
    public async Task SendMessageAsync(int senderId, int receiverId, string
message)
    {
        var chatMessage = new ChatMessage
        {
            SenderId = senderId,
            ReceiverId = receiverId,
            MessageText = message,
            SentAt = DateTime.UtcNow
        };
        _context.ChatMessages.Add(chatMessage);
        await _context.SaveChangesAsync();
    }
```

На клієнтській стороні чат реалізовано через AJAX-виклики, що дозволяє оновлювати повідомлення без перезавантаження сторінки. Надсилаючи повідомлення, користувач бачить його одразу в інтерфейсі.

Рисунок 4.3 демонструє вигляд чату з історією листування між учнем і викладачем.

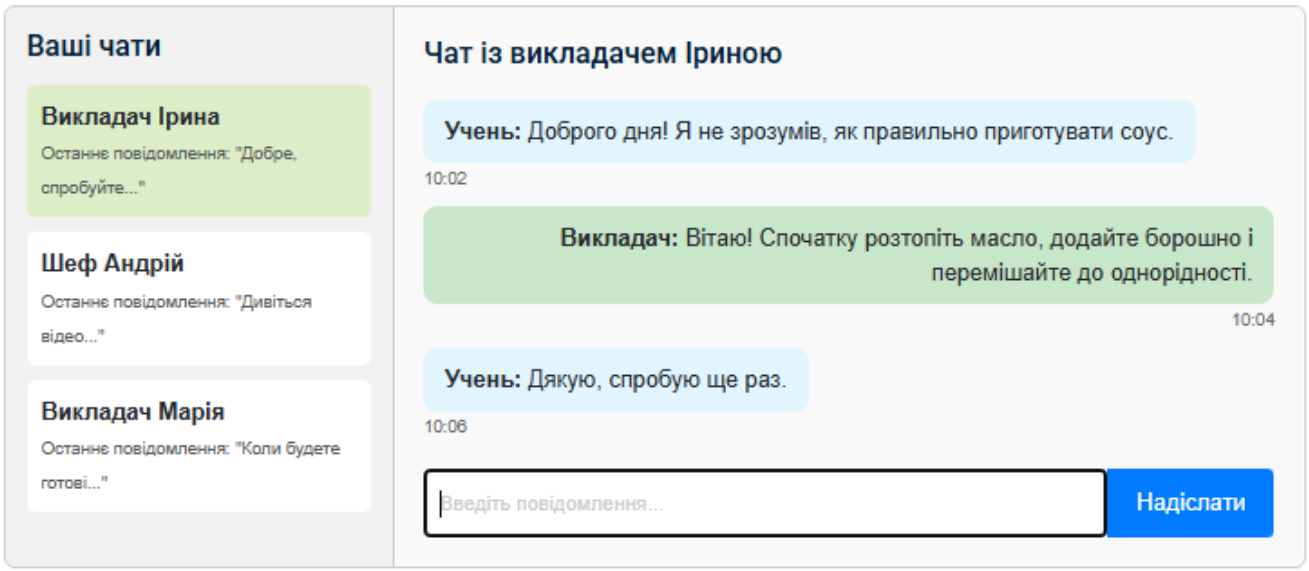

Рисунок 4.3 – Сторінка з варіантами відповіді та кнопкою надсилання результату (скріншот виконано самостійно)

Такий доволі простий та лаконічний дизайн дозволяє зберігати увагу користувача на основних елементах управління, що значно прискорює роботу з

програмною системою навіть для нових або недосвідчених користувачів у різних сценаріях її використання.

4.4 Використання Dependency Injection

У проєкті застосовано принцип впровадження залежностей (Dependency Injection), який є стандартною практикою в ASP.NET Core. Це дозволяє зменшити зв’язність між компонентами системи та забезпечити високу гнучкість і тестованість. Усі сервіси, репозиторії та контексти бази даних реєструються у вбудованому контейнері служб під час запуску застосунку в методі ConfigureServices. Замість створення екземплярів об'єктів вручну в контролерах або інших класах, залежності автоматично передаються через конструктор. Такий підхід дозволяє легко змінювати реалізацію сервісу (наприклад, для підключення до іншого API або бази даних) без модифікації залежних компонентів. Також метод ConfigureServices встановлює строку підключення до бази даних, яку зчитує з JSON файлу конфігурації.

Нижче навдений код методу ConfigureServices:

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddControllersWithViews();

    string connectionString =
Configuration.GetConnectionString("DefaultConnection");

    services.AddDbContext<AppDbContext>(x => x.UseSqlServer(
        connectionString
    ));


services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationSchem
e).AddCookie(options => {
            options.LoginPath = new
Microsoft.AspNetCore.Http.PathString("/Account/Login");
            options.AccessDeniedPath = new
Microsoft.AspNetCore.Http.PathString("/Account/Login");
    });

    services.AddScoped<IBaseRepository<Course>, CourseRepository>();
    services.AddScoped<IBaseRepository<User>, UserRepository>();
    services.AddScoped<IService<Course>, CourseService>();
    services.AddScoped<IService<Course>, CourseService>();
    services.AddScoped<IService<User>, UserService>();
```

```
    services.AddScoped<IService<User>, UserService>();
    services.AddScoped<IAccountService, AccountService>();
    services.AddScoped<IAccountService, AccountService>();
}
```

Цей механізм використано для організації доступу до бази даних, бізнес-логіки та логування. Наприклад, інтерфейс ICourseService має свою реалізацію CourseService, яка впроваджується у відповідний Razor Page чи контролер через DI.

Таким чином досягається інверсія контролю, що є фундаментом для побудови масштабованих і гнучких програмних рішень.

4.5 Трирівнева архітектура

Під час розробки веб-застосунку було реалізовано трирівневу архітектуру, яка є однією з найпоширеніших та ефективних моделей побудови сучасних програмних систем. Такий підхід дозволяє досягти високого ступеня структурованості, розділення обов’язків і, як наслідок, значно спрощує супровід, модифікацію та масштабування проєкту. Завдяки чіткому розмежуванню між рівнями досягається ізоляція змін: зміна реалізації одного рівня не вимагає обов’язкової зміни решти частин системи, що позитивно впливає на стабільність і продуктивність розробки.

Архітектура поділяється на три основні рівні: представницький (Presentation Layer), рівень бізнес-логіки (Business Logic Layer) та рівень доступу до даних (Data Access Layer). Кожен із них виконує чітко визначені функції та взаємодіє з іншими рівнями через добре сформовані інтерфейси. На рівні представлення використовується технологія Razor Pages, яка дає змогу формувати інтерфейс користувача з підтримкою динамічного оновлення даних без повного перезавантаження сторінки. Цей рівень безпосередньо працює з користувацькими діями, передає інформацію на обробку у бізнес-рівень та відображає отримані результати.

Бізнес-рівень містить усю функціональну логіку застосунку. Тут відбувається обробка даних, застосування бізнес-правил, перевірка умов, трансформація об’єктів тощо. Він не містить жодного коду, пов’язаного з

виведенням інформації чи взаємодією з базою даних, завдяки чому залишається універсальним та легко тестованим.

Рівень доступу до даних реалізується за допомогою Entity Framework Core, що дозволяє працювати з базою даних у вигляді об'єктів за допомогою LINQ-запитів. Усі запити до SQL Server 2019, як і збереження змін, реалізовані через відповідні репозиторії, що інкапсулюють логіку доступу до даних. Це дозволяє зменшити дублювання коду та централізувати обробку інформації з бази, забезпечуючи при цьому високу продуктивність і контроль над транзакціями.

Загальна структура проєкту відповідає вищезазначеній архітектурі, що підтверджується структурою рішення, показаною на скріншоті (див. рис. 4.4).

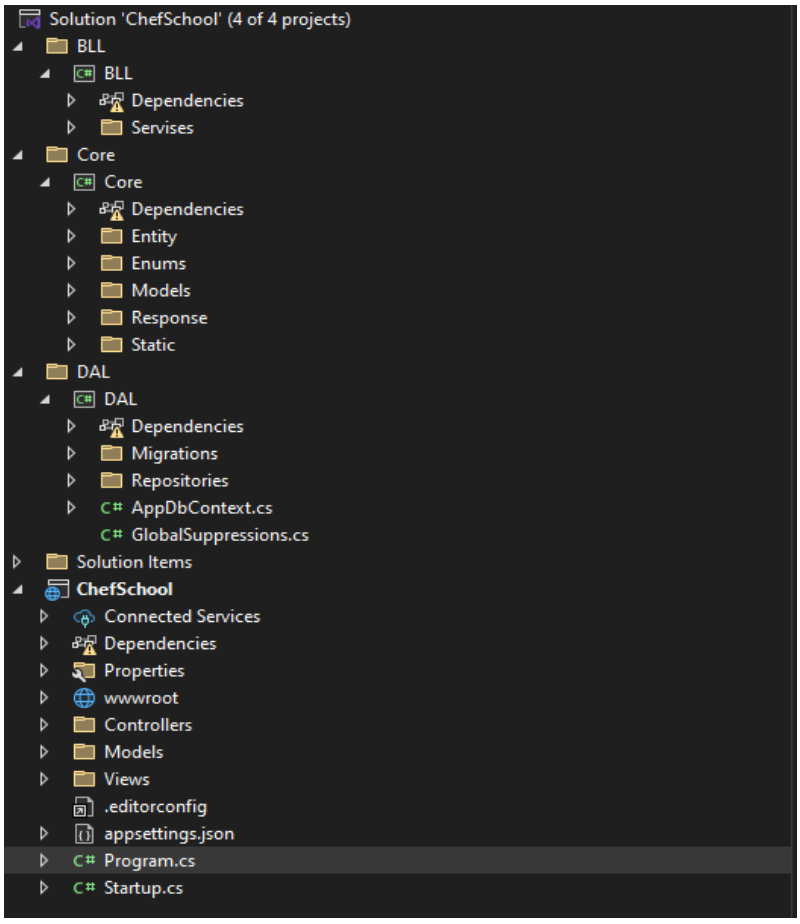

Рисунок 4.4 – Структура проєкту (скріншот виконано самостійно)

Застосування трирівневої архітектури у цьому програмному продукті дозволило досягти високої якості коду, спростити процес налагодження та полегшити реалізацію нових функцій без потреби у масштабних змінах у вже реалізованих частинах системи.

5 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Тестування веб-застосунку

У процесі розробки веб-застосунку для онлайн-навчання кулінарному мистецтву ключовим етапом стало його тестування, що дало змогу впевнитися у працездатності основних функцій, відповідності реалізованого функціоналу вимогам користувачів, а також зручності використання інтерфейсу. Саме тестування відіграє важливу роль у гарантуванні якості програмного забезпечення, оскільки дозволяє виявити й усунути помилки на ранніх етапах життєвого циклу продукту. Тестування проводилось як вручну, так і за допомогою юніт-тестування, використовуючи принципи білого та чорного ящиків[16]. Головний акцент був на функціональні сценарії, що охоплюють типову поведінку користувача під час роботи з системою, включаючи всі ключові модулі веб-застосунку. Це дозволило виявити потенційні недоліки ще до етапу розгортання застосунку на продакшн-сервері, що значно підвищило загальну якість і стабільність продукту.

Основна увага була зосереджена на перевірці таких важливих елементів, як процес реєстрації та авторизації користувача, доступ до навчального контенту, динамічне завантаження матеріалів, проходження інтерактивних тестів із подальшою відправкою відповідей, збереження результатів користувача, а також функціонування вбудованого чату між учнем і викладачем для забезпечення зворотного зв’язку. Крім цього, додатково перевірялися такі аспекти, як обробка помилок, валідація вхідних даних, збереження сесій користувачів, робота навігаційного меню та наявність зручної логіки переходів між сторінками. Застосунок перевірявся на предмет відповідності очікуваним результатам, правильності обробки введених даних, наявності всіх необхідних перевірок і обмежень у формах, а також стійкості до неправильних або неочікуваних дій з боку користувача, зокрема введення некоректних даних, навмисного пропуску полів або багаторазового надсилання форм.

Для кожного окремого тестового сценарію було розроблено відповідний тест-кейс, що містив докладний опис послідовності дій користувача, передумови для виконання тесту, вхідні дані, очікувану поведінку системи на кожному кроці, а

також фактичний результат виконання з відповідною фіксацією статусу проходження. Тест-кейси охоплювали щонайменше сім логічних кроків, що дозволяло більш повно перевірити не лише основний функціонал, але й логіку взаємодії з інтерфейсом, з урахуванням можливих відхилень від стандартного сценарію. Це сприяло виявленню дрібних недоліків і підвищенню зручності взаємодії користувача із застосунком. Під час виконання тестування жодних критичних помилок виявлено не було. Застосунок стабільно працював у підтримуваних браузерах (зокрема Chrome, Firefox, Edge), коректно обробляв усі введені дані, не допускав некоректних операцій та виконував передбачені сценарії згідно з функціональними вимогами та очікуваннями кінцевого користувача.

Тест-план, структура проведення перевірок, а також усі створені тест-кейси із зазначенням кроків, результатів тестування та сформульованих висновків оформлено у вигляді таблиць (див. Додаток А), що забезпечує наочність та простоту аналізу результатів тестування при подальшій роботі над удосконаленням застосунку.

ВИСНОВКИ

В результаті роботи над кваліфікаційною роботою бакалавра було здійснено повноцінну розробку програмної системи для онлайн-навчання кулінарному мистецтву, що включала всі етапи – від аналізу вимог до реалізації та тестування.

Нижче наведено основні етапи роботи над проєктом:

- виконано аналіз предметної області онлайн-навчання кулінарії, а також досліджено наявні конкурентні рішення;
- здійснено концептуальне, інфологічне та логічне моделювання системи, зокрема створено UML-діаграми варіантів використання, класів, активностей, а також ER-діаграму бази даних;
- реалізовано програмну систему з використанням сучасних вебтехнологій: Razor Pages для клієнтської частини та ASP.NET Core (C#) для серверної логіки;
- розроблено реляційну базу даних у Microsoft SQL Server з використанням ORM-технології Entity Framework Core;
- виконано ручне функціональне тестування системи, перевірено відповідність реалізованих функцій вимогам, зручність користування, стабільність та безпечність роботи програмного забезпечення.

У результаті виконання аналізу предметної області було визначено основні функціональні вимоги до програмної системи для онлайн-навчання кулінарному мистецтву. Проведене дослідження сучасного стану ринку освітніх веб-рішень дало змогу виокремити актуальні проблеми, зокрема нестачу україномовного контенту, обмежені можливості взаємодії між учасниками навчального процесу та недостатній рівень персоналізації навчання. Це дозволило сформулювати чіткі цілі проєкту — створення повноцінної освітньої платформи, яка не лише надає доступ до навчального контенту, а й забезпечує зворотний зв’язок, оцінювання, аналіз прогресу та адаптацію до індивідуальних потреб користувачів.

Було окреслено основних користувачів системи — учнів і викладачів, сформовано типові сценарії взаємодії кожного типу користувача з платформою.

Особливу увагу було приділено UX-аспектам, що вплинуло на вибір архітектури, дизайну інтерфейсу та функціонального наповнення. Завдяки цьому вдалося визначити оптимальний набір функцій для кожної ролі, забезпечивши зручність і логічність використання системи навіть для користувачів без спеціальної технічної підготовки.

На етапі архітектурного проєктування створено фундаментальні компоненти системи. Була розроблена трирівнева архітектура (Presentation Layer, Business Logic Layer, Data Access Layer), яка дозволяє досягти модульності, масштабованості та підтримуваності коду. В рамках моделювання системи створено UML-діаграми класів, варіантів використання та активностей, а також ER-діаграма бази даних, яка наочно демонструє взаємозв’язки між сутностями, такими як курси, матеріали, користувачі, оцінки тощо.

На основі сформованої логіки реалізовано прототип користувацького інтерфейсу, який включає основні сторінки: головну (з переліком курсів і фільтрами), сторінку конкретного курсу (з детальним описом, матеріалами, кнопками запису або редагування) та сторінку профілю (з особистими даними, історією навчання, статистикою). Інтерфейс створено з урахуванням принципів зручності та доступності, із застосуванням інструментів UI/UX-дизайну (Figma), а також додано анімації для покращення взаємодії з користувачем.

У процесі реалізації окремих модулів платформи було впроваджено низку важливих алгоритмів і допоміжних методів. Зокрема, реалізовано функціональність хешування паролів для забезпечення безпеки даних користувачів (на основі SHA256), перевірку унікальності при реєстрації нового користувача, обчислення відсотка проходження курсу та системи оцінювання. Ці технічні рішення забезпечують надійність, безпеку та гнучкість у подальшому масштабуванні системи.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Афанасьєва І. В., Лозиченко А. В. The 7 th International scientific and practical conference “Science, society, education: topical issues and development prospects” (June 7-9, 2020) SPC “Sci-conf.com.ua”, Kharkiv, Ukraine. 2020. p. 358-361.

2. The best 21 cooking websites [Електронний ресурс] – URL: https://www.chefspencil.com/the-best-20-cooking-websites/ (дата звернення: 10.05.2025)

3. Leading National Culinary Arts Schools [Електронний ресурс] – URL: https://www.culinaryschools.org/top-culinary-schools/ (дата звернення: 10.05.2025)

4. BBC Good Food [Електронний ресурс] – URL: https://www.bbcgoodfood.com/ (дата звернення: 10.05.2025)

5. MasterClass [Електронний ресурс] – URL: https://www.masterclass.com/ (дата звернення: 10.05.2025)

6. ChefSteps [Електронний ресурс] – URL: https://www.chefsteps.com/ (дата звернення: 10.05.2025)

7. Introduction to Razor Pages in ASP.NET Core [Електронний ресурс] – URL: https://learn.microsoft.com/en-us/aspnet/core/razor-pages/?view=aspnetcore-9.0&tabs=visual-studio (дата звернення: 10.05.2025)

8. Overview of ASP.NET Core [Електронний ресурс] – URL: https://learn.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-9.0 (дата звернення: 10.05.2025)

9. Microsoft SQL Server [Електронний ресурс] – URL: https://www.microsoft.com/en-us/sql-server (дата звернення: 10.05.2025)

10. Entity Framework Core [Електронний ресурс] – URL: https://learn.microsoft.com/en-us/ef/core/ (дата звернення: 10.05.2025)

11. Афанасьєва І. В., Горішня К., Онищенко К. Г., Голян Н. В., Голян В. В. ПРОЕКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ ДЛЯ БРОНЮВАННЯ КВИТКІВ.

12. Model–view–controller [Електронний ресурс] – URL: https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller (дата звернення: 10.05.2025)

13. REST [Електронний ресурс] – URL: https://en.wikipedia.org/wiki/REST (дата звернення: 10.05.2025)

14. Dapper ORM [Електронний ресурс] – URL: https://dappertutorial.net/ (дата звернення: 10.05.2025)

15. Password Storage Cheat Sheet [Електронний ресурс] – URL: https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html (дата звернення: 10.05.2025)

16. Афанасьєва І. В., Голян Н. В., Голян В. В. BLACK AND WHITE-BOX UNIT TESTING FOR WEB APPLICATIONS. 2022. p. 79-83.

17. Посилання на GitHub репозиторій з вихідним кодом, відео та специфікацією [Електронний ресурс] - URL: https://github.com/NureNekhTaras/DIPLOM (дата звернення: 08.06.2025)