

ВИПУСКНА КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

на тему:

Освітня система з теоретичною досвідковою базою та тестуванням

(назва затвердженої теми)

(на прикладі _____)

_____ Тітова Вікторія Сергіївна

(підпис студента) (прізвище, ім'я по-батькові студента)

Науковий

керівник _____ Яровий Р.О

(підпис наукового керівника) (прізвище, ініціали, наукове звання і ступінь)

Допущено до захисту “11.04.2025 ” 2025 р.

Завідувач кафедри _____ Яровий Роман Олександрович

(підпис) (прізвище, ініціали, наукове звання і ступінь)

Київ - 2025

ЗМІСТ

ЗМІСТ	4
Анотація	6
Annotation	8
ВСТУП	9
РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ ОСВІТНЬОЇ СИСТЕМИ	11
1.1. Поняття та класифікація освітніх систем	11
1.2. Сучасні технології розробки освітніх веб-застосунків	13
1.2.1. Обґрунтування вибору ASP.NET Core MVC	14
1.2.2. Використання мови програмування C#	17
1.2.3. Entity Framework Core — зручний та безпечний ORM для роботи з базами даних	19
1.2.4. Microsoft SQL Server	20
1.2.5. Інтерфейс користувача: HTML5, CSS3, Bootstrap, Razor	23
1.3. Аналіз подібних рішень	26
Висновок до розділу 1	29
РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА ПОСТАНОВКА ЗАДАЧІ	31
2.1. Мета, завдання та цільова аудиторія системи	31
2.2. Архітектура та обґрунтування вибору технологій	33
2.3. Загальна структура системи та модулів	36
2.3.1. Модуль авторизації та управління користувачами	37
2.3.2. Модуль теоретичної довідкової бази	37
2.3.3. Модуль прикладів задач	37
2.3.4. Модуль інтерактивних завдань	38
2.3.5. Модуль тестування	38
2.3.6. Модуль аналітики та звітності	38
2.4. База даних і зберігання даних	39
Висновок до розділу 2	41
РОЗДІЛ 3. РЕАЛІЗАЦІЯ ОСВІТНЬОЇ СИСТЕМИ	43
3.1. Загальний опис реалізації	43
3.2. Реалізація модуля авторизації	45
3.3. Теоретична довідкова база	47
3.4. Приклади задач і завдань	48
3.5. Модуль інтерактивних прикладів	50
3.6. Модуль тестування	52
3.7. Обробка винятків і валідація даних	54
3.8. Рольова модель доступу та безпека	55
3.9. Тестування функціональності системи	57

3.10. Можливості масштабування і модернізації	59
3.11. Стилiзація інтерфейсу та зручність використання (UI/UX)	61
Висновок до розділу 3	63
ВИСНОВОК	66
Список використаних джерел	68
ДОДАТОК А	70
Інтерфейс користувача освітньої системи	70
Головна сторінка користувача	70
Розділ «Теоретична база» – структура та навігація	72
Перегляд теоретичного матеріалу – вивчення теми у форматі статті	73
Каталог задач – вибір практичного завдання для самостійного виконання	75
Детальна задача з прикладом правильного розв'язку	77
Інтерактивне завдання – покрокове практичне навчання	80
Адміністративна панель керування тестами	82

Анотація

У бакалаврській кваліфікаційній роботі розроблено освітню веб-систему нового покоління, яка поєднує в собі елементи теоретичного навчання, практичного закріплення знань і автоматизованого тестування. Актуальність теми зумовлена зростаючими потребами сучасної освіти у цифровізації, персоналізації навчального процесу та забезпеченні гнучкого доступу до знань у будь-який час і з будь-якого пристрою. У роботі здійснено ґрунтовний аналіз існуючих рішень у сфері електронного навчання (Moodle, Google Classroom, Prometheus, Quizizz) та виявлено їхні обмеження, що стали підґрунтям для створення власного рішення.

Метою дослідження є створення інтегрованої освітньої платформи, яка дозволяє студентам опановувати теоретичний матеріал, розв'язувати задачі за темами, проходити інтерактивні вправи та перевіряти рівень засвоєння знань через тести з автоматичним аналізом результатів. Система реалізована з використанням сучасних технологій: ASP.NET Core MVC, мови програмування C#, Entity Framework Core для взаємодії з базою даних Microsoft SQL Server, HTML5, CSS3, Bootstrap 5 та Razor для побудови адаптивного та зручного інтерфейсу.

У межах реалізації платформи передбачено модуль авторизації, довідковий теоретичний розділ, приклади задач, інтерактивні завдання, блок тестування, модуль звітності та аналітики. Інформаційна архітектура базується на шаблоні MVC, що забезпечує логічне розмежування коду, гнучке масштабування та простоту подальшої підтримки системи. Особливу увагу приділено безпеці, валідації введених даних та підтримці ролей користувачів з відповідними правами доступу.

Результатом роботи є повноцінна освітня веб-система, адаптована під потреби самостійного навчання, дистанційної підготовки та корпоративного використання для внутрішніх освітніх програм. Система має практичне значення як готове рішення, що може бути впроваджене в освітніх закладах, на

онлайн-курсах, у навчальних центрах або як базова платформа для подальшої розробки персоналізованих навчальних середовищ.

Annotation

This bachelor's qualification thesis presents the development of a next-generation educational web platform that integrates theoretical study, practical problem-solving, interactive exercises, and automated testing. The relevance of the topic stems from the growing demand for digitalization and personalization in modern education, as well as the need to provide flexible and accessible learning experiences across various devices and settings. The research includes a comprehensive analysis of existing e-learning platforms (such as Moodle, Google Classroom, Prometheus, and Quizizz) and highlights their limitations, which served as a foundation for creating a more universal and adaptable system.

The objective of the project is to design and implement an integrated learning platform that enables users to systematically study theoretical material, practice solving categorized problems, complete interactive exercises, and evaluate their knowledge through automated tests with instant result analysis. The system is built using modern technologies such as ASP.NET Core MVC, C# programming language, Entity Framework Core for database interaction with Microsoft SQL Server, and HTML5, CSS3, Bootstrap 5, and Razor for creating an intuitive and responsive user interface.

The platform includes modules for user authorization, a structured knowledge base, categorized examples, interactive tasks, a testing block, and a reporting and analytics module. Its architecture follows the Model–View–Controller (MVC) design pattern, which ensures clear code separation, scalability, and ease of maintenance. Special attention is given to system security, data validation, and role-based access control.

As a result, the work delivers a functional educational web application suitable for independent learning, distance education, and corporate training programs. It holds significant practical value as a ready-to-use solution that can be deployed in academic institutions, online courses, learning centers, or serve as a foundation for developing customized educational environments.

ВСТУП

У сучасному інформаційному суспільстві стрімкий розвиток технологій зумовлює необхідність впровадження інноваційних підходів до організації навчального процесу. Традиційні методи подачі навчального матеріалу поступово доповнюються та замінюються цифровими системами, що дозволяють забезпечити гнучкість, доступність та інтерактивність освітнього середовища. В умовах дистанційного та змішаного навчання особливої актуальності набувають системи, що поєднують у собі теоретичну підготовку, прикладні приклади та контроль знань.

Актуальність теми дослідження зумовлена потребою в розробці інструментів, які б не лише забезпечували доступ до теоретичного матеріалу, але й дозволяли користувачам закріплювати знання через практичні завдання та тестування. Відсутність цілісних, універсальних рішень, адаптованих під сучасні освітні вимоги, обумовлює доцільність створення власного рішення в межах цього проєкту.

Метою дипломної роботи є розробка освітньої веб-системи з функціоналом довідкової бази, прикладами задач та інтерактивним модулем тестування, що дозволяє підвищити ефективність засвоєння навчального матеріалу.

Завдання дослідження:

Провести аналіз існуючих рішень у сфері освітніх систем.

Визначити функціональні вимоги до розроблюваної системи.

Розробити архітектуру та реалізувати модулі: теоретична база, приклади задач, тестування, авторизація.

Реалізувати інтерфейс користувача з урахуванням зручності та доступності.

Здійснити тестування системи та оцінити її функціональність.

Об'єктом дослідження є процес організації онлайн-навчання засобами інформаційних технологій.

Предметом дослідження є програмні засоби створення інтерактивних освітніх систем, які забезпечують теоретичну підготовку та перевірку знань.

Практичне значення дипломної роботи полягає у створенні готової освітньої платформи, яка може бути використана для самостійного або організованого навчання. Результати роботи можуть бути застосовані у закладах освіти або на онлайн-платформах з підвищення кваліфікації.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ ОСВІТНЬОЇ СИСТЕМИ

1.1. Поняття та класифікація освітніх систем

У сучасному інформаційному суспільстві освіта є не лише пріоритетом державної політики, а й ключовим чинником особистого та професійного розвитку кожної людини. З огляду на глобалізацію, стрімкий розвиток технологій, зростання обсягів інформації та зміну форм подачі знань, виникає об'єктивна потреба у створенні, використанні та розвитку освітніх систем нового покоління. Ці системи повинні відповідати не лише змістовим вимогам освітніх стандартів, а й технічним, психолого-педагогічним та організаційним критеріям.

Освітня система в контексті цифрових технологій — це сукупність апаратно-програмних засобів, методичних матеріалів, інформаційного контенту та інтерфейсних рішень, яка забезпечує цілісну, систематизовану та інтерактивну підтримку навчального процесу. Такі системи можуть функціонувати як на рівні окремих освітніх курсів, так і охоплювати повноцінні навчальні програми, інтегруватися з іншими платформами, системами контролю знань, аналітичними модулями та ресурсними бібліотеками.

З розвитком цифрових технологій, освітні системи зазнали еволюції від примітивних комп'ютерних тренажерів до складних веб-орієнтованих або хмарних рішень із підтримкою мультимедіа, відео, інтерактиву, адаптивного навчання та штучного інтелекту. Вони стали частиною екосистеми сучасного навчального середовища — як у формальній освіті (школи, ВНЗ), так і в корпоративному навчанні чи неформальних ініціативах (самоосвіта, онлайн-курси, сертифікації).

Класифікація освітніх систем може здійснюватися за різними критеріями, зокрема:

За функціональним призначенням:

LMS (Learning Management Systems) — системи управління навчанням, орієнтовані на адміністрування, облік, розподіл ресурсів, контроль результатів. Приклади: Moodle, Google Classroom, Canvas.

LCMS (Learning Content Management Systems) — системи управління навчальним контентом, фокусуються на збереженні, організації, повторному використанні матеріалів.

Системи тестування (e-Assessment) — спеціалізовані платформи для створення, проведення й оцінювання тестів: Testportal, Quizizz, Exam.net.

Інтелектуальні навчальні системи (ITS) — адаптуються до рівня знань користувача, використовують машинне навчання для аналізу прогресу.

Мікронавчальні системи (Microlearning Platforms) — надають навчальний контент у вигляді коротких, сфокусованих блоків (наприклад, Duolingo).

За організаційною формою навчання:

Дистанційні — повністю віддалене навчання, засноване на електронному середовищі.

Змішані (blended) — поєднують традиційне навчання з цифровим компонентом.

Очні, але підтримувані електронною платформою — підтримка самостійної роботи студентів (наприклад, електронні читанки, самотестування).

За характером взаємодії:

Системи з автоматизованим зворотним зв'язком (автоматичні перевірки, рейтинги).

Системи з людською участю (викладач перевіряє, коментує, аналізує).

Гібридні рішення, що поєднують автоматизацію і живу взаємодію.

4. За типом користувача:

Індивідуальні системи для самонавчання.

Системи для викладачів і адміністраторів.

Корпоративні освітні платформи.

Існують також комплексні системи, які об'єднують кілька функцій: управління курсами, тестування, форумами, веденням журналів, обліком відвідуваності, аналітикою успішності тощо.

Розвиток таких систем йде паралельно з розвитком педагогічних підходів: від традиційного навчання (інструктор-центричне) до конструктивістського (заснованого на взаємодії), а останнім часом — до адаптивного й персоналізованого (система підлаштовується під рівень користувача, стиль навчання, інтереси). Це вимагає від сучасних освітніх систем не лише зберігати дані, а й реагувати на дії користувача, змінювати структуру навчального процесу в реальному часі.

Варто також виділити освітні системи з відкритим вихідним кодом, які дозволяють змінювати архітектуру, додавати функції, адаптувати інтерфейс — прикладом є Moodle. Такі системи є гарною основою, однак потребують значних ресурсів на розгортання, налаштування та адміністрування. Це стало однією з причин, чому в межах цієї дипломної роботи було прийнято рішення створити власну, легку та адаптовану до конкретних вимог платформу, яка поєднує теоретичну довідкову базу, приклади задач, інтерактивні завдання та модуль тестування.

Таким чином, освітні системи — це не просто набори інструментів для викладання. Вони формують нову парадигму навчання, де студент стає активним учасником процесу, а технології — інструментами підтримки, персоналізації, моніторингу та розвитку. Грамотний вибір типу системи, її структури та функціоналу дозволяє значно підвищити ефективність навчального процесу та зробити його максимально доступним, інклюзивним і результативним.

1.2. Сучасні технології розробки освітніх веб-застосунків

Стрімкий розвиток інформаційних технологій привів до широкого впровадження веб-орієнтованих систем у різні галузі, включно з освітою. Розробка сучасних освітніх платформ потребує вибору таких технологій, які одночасно забезпечують: високу продуктивність, надійність, зручність у

розширенні, підтримку мультимодульної архітектури та безпечне зберігання й обробку даних.

У рамках цієї дипломної роботи було обрано стек розробки на основі ASP.NET Core MVC, мови програмування С#, платформи .NET, фреймворку Entity Framework Core для роботи з базами даних, та SQL Server як систему керування базами даних. Інтерфейс користувача реалізовано з використанням HTML5, CSS3, Bootstrap 5 і Razor-шаблонів.

1.2.1. Обґрунтування вибору ASP.NET Core MVC

Розробка сучасних веб-застосунків, зокрема освітніх платформ, потребує використання технологій, які одночасно відповідають кільком ключовим критеріям: висока продуктивність, гнучкість архітектури, масштабованість, безпека, а також наявність сучасного інструментарію для ефективної реалізації функціоналу. У рамках даної дипломної роботи було прийнято рішення використовувати платформу ASP.NET Core MVC як основну технологію для побудови всієї логіки серверної частини розроблюваної освітньої системи. Такий вибір є результатом ретельного аналізу можливостей різних веб-фреймворків, серед яких розглядалися альтернативи на основі PHP (Laravel), JavaScript (Node.js з Express), Python (Django), а також Java (Spring). У підсумку ASP.NET Core MVC виявився оптимальним за співвідношенням функціональності, зручності розробки, підтримки масштабування та надійності.

ASP.NET Core — це сучасний, кросплатформний, відкритий фреймворк, розроблений корпорацією Microsoft. Його головною перевагою є архітектурна гнучкість і модульність, що дозволяє розробляти як невеликі додатки, так і масштабні корпоративні системи. У контексті даної роботи ASP.NET Core був обраний з кількох вагомих причин. По-перше, він підтримує архітектурну модель MVC (Model–View–Controller), яка є однією з найзручніших та найпоширеніших у світі веб-розробки.

MVC — це архітектурний шаблон, що забезпечує розділення логіки програми на три незалежні компоненти: модель (дані), уявлення (інтерфейс) та контролер (керування логікою). Таке розділення дозволяє ефективно

організувати код, підвищити масштабованість системи, спростити тестування та полегшити внесення змін у майбутньому.

У контексті розробки освітньої системи застосування моделі MVC дозволило чітко структурувати застосунок. Компонент Model відповідає за структуру даних і логіку взаємодії з базою даних, зокрема зберігання теоретичних матеріалів, прикладів задач, результатів тестування тощо. Компонент View — за виведення інформації для користувача у зручному вигляді: структури розділів, форм проходження тестів, інтерфейсу реєстрації тощо. Controller, у свою чергу, є посередником між користувачем і моделлю, обробляє запити, змінює стан системи або передає дані до подань.

Ця архітектура дає змогу розробляти окремі модулі незалежно один від одного, а також забезпечує чітке розмежування обов'язків для кожного елемента, що надзвичайно важливо у командній розробці або в процесі майбутньої підтримки системи. Оскільки освітня система передбачає багатофункціональність (авторизація, перегляд матеріалів, тестування тощо), модель MVC дозволила легко масштабувати застосунок без повторного написання коду або дублювання логіки.

По-друге, ASP.NET Core MVC забезпечує високу продуктивність і оптимізований конвеєр обробки запитів, що дозволяє підтримувати швидку реакцію системи навіть при зростанні кількості користувачів. Це особливо важливо для освітньої системи, в якій може передбачатися паралельний доступ десятків або сотень користувачів до модулів навчання, тестування, перегляду результатів тощо. На рівні інфраструктури фреймворк має вбудовану підтримку кешування, логування, механізмів обробки помилок, а також систему middleware, яка дозволяє централізовано управляти всіма HTTP-запитами в додатку. Усе це забезпечує гнучку та контрольовану обробку даних.

Окремої уваги заслуговує реалізація системи автентифікації та авторизації, яка у ASP.NET Core вбудована на рівні фреймворку. Завдяки цьому було реалізовано зручну реєстрацію користувачів, вхід до системи з використанням сесій, зберігання паролів у хешованому вигляді та призначення

ролей (наприклад, «користувач» та «адміністратор»). Це дозволило побудувати не лише безпечну, але й масштабовану систему управління доступом до окремих функцій — наприклад, можливість створення тестів та редагування контенту доступна лише адміністраторам. Механізм авторизації реалізований через атрибут `[Authorize]` у контролерах та діє незалежно від бази даних, що дозволяє гнучко його налаштовувати.

Крім того, ASP.NET Core MVC чудово інтегрується з Entity Framework Core, що дозволило побудувати повноцінну взаємодію з реляційною базою даних Microsoft SQL Server. Завдяки підходу Code First, структура бази створювалася автоматично на основі класів C#, що значно спростило розробку й забезпечило гнучке оновлення структури бази через механізм міграцій. Усе це дозволило зберігати й опрацьовувати дані про користувачів, тести, теоретичні матеріали, результати тощо у єдиній базі без дублювання або втрати інформації.

Ще одним важливим аргументом на користь вибору ASP.NET Core є широка підтримка розробників, стабільність фреймворку, регулярне оновлення, а також глибока документація від Microsoft. Завдяки цьому, процес розробки освітньої системи був не лише технічно обґрунтованим, а й ефективним з погляду навчання та підтримки. Інтеграція з середовищем Visual Studio 2022 забезпечила інтелектуальну підказку, автоматичну генерацію коду, налагодження, підтримку Git та швидке розгортання системи на локальному сервері.

Не менш важливим чинником є кросплатформність ASP.NET Core: система може бути розгорнута як на Windows-серверах, так і на Linux-хостингу або у контейнерах Docker. Це відкриває широкі можливості для подальшого масштабування та перенесення застосунку на продуктивне середовище без залежності від конкретної операційної системи.

Таким чином, вибір ASP.NET Core MVC як основної технології для розробки освітньої платформи був зумовлений не лише популярністю та рекомендаціями спільноти, а передусім технічними перевагами. Цей фреймворк поєднує в собі всі необхідні інструменти для побудови надійної, безпечної та

масштабованої системи, яка відповідає сучасним вимогам до освітніх платформ. Його впровадження дозволило реалізувати всі модулі системи з високою ефективністю, чіткою логікою, інтуїтивним інтерфейсом і стабільною взаємодією між компонентами.

1.2.2. Використання мови програмування C#

У межах розробки освітньої системи однією з ключових технологій виступає мова програмування C# (C-Sharp) — високорівнева, об'єктно-орієнтована, строго типізована мова, розроблена компанією Microsoft спеціально для платформи .NET. Вибір цієї мови був зумовлений низкою переваг, що роблять її ідеальною для реалізації складних, масштабованих, безпечних та підтримуваних веб-застосунків, зокрема — у сфері освітніх технологій.

C# — це мова, яка поєднує в собі кращі властивості як класичних мов програмування (на кшталт C++ або Java), так і сучасних підходів до побудови програмних рішень. Завдяки багатому функціоналу, великій стандартній бібліотеці та постійній підтримці з боку Microsoft, ця мова набула популярності не лише в корпоративному програмуванні, а й у розробці веб-платформ, хмарних сервісів, десктопних застосунків, мобільних додатків, і навіть ігор.

Однією з головних причин, чому C# було обрано для реалізації саме освітньої платформи, є її висока стабільність і безпечність при роботі з даними. Система, що розроблялася, має справу з персональними даними користувачів, результатами тестів, навчальним контентом, тож вона потребує чіткого контролю типів, захисту від некоректних запитів, а також забезпечення роботи з базою даних без втрати цілісності інформації. Завдяки статичній типізації, компілятор C# виявляє більшість помилок ще до запуску програми, що дозволяє запобігти некоректній поведінці на етапі виконання. Це особливо важливо для складних систем, де критичними є стабільність та точність результатів.

Крім того, C# має розвинену об'єктно-орієнтовану модель, що дозволяє проектувати систему у вигляді набору сутностей (класи: **User**, **Test**, **TheorySection**, **Question**, **Result** тощо), які чітко відображають логіку предметної

області. Це робить код не лише логічно зрозумілим, а й легко масштабованим у майбутньому. Наприклад, у разі потреби додати модуль з рейтингами або повідомленнями, це можна реалізувати у вигляді окремої моделі, яка буде взаємодіяти з іншими через відношення та контролери, без переробки вже існуючої логіки.

Особливе значення в освітній системі має робота з потоками і асинхронність. Оскільки система може одночасно обслуговувати велику кількість користувачів — які читають матеріали, проходять тести, взаємодіють із задачами, — було важливо забезпечити обробку запитів без блокування ресурсу. Мова C# надає потужний механізм асинхронного програмування — ключові слова `async` та `await`, які дозволяють реалізувати конкурентне виконання операцій із мінімальними витратами пам'яті й ресурсів. Завдяки цьому користувач не відчуває затримки при завантаженні сторінок, отриманні результатів тестування чи навігації між модулями.

Також C# забезпечує високий рівень безпеки при роботі з пам'яттю. Завдяки автоматичному збиранню сміття (Garbage Collector), система самостійно очищає невикористовувані об'єкти, що суттєво знижує ризики витоків пам'яті та підвищує надійність. У контексті великої платформи, де одночасно обробляються десятки об'єктів — користувачів, сесій, тестів, — це дає значну перевагу в стабільності роботи застосунку в довготривалій перспективі.

Мова C# також має великий набір вбудованих засобів для роботи з базами даних, рядками, файлами, колекціями, лінками, що дозволило значно спростити реалізацію логіки освітньої системи. Наприклад, механізми обробки рядків були використані для пошуку ключових слів у теоретичних матеріалах, а робота з колекціями — для відбору відповідей у тестах або відображення списку результатів.

Інтеграція C# з іншими технологіями платформи .NET, зокрема Entity Framework Core, ASP.NET Core, Razor Pages, є природною, а не примусовою, що дозволяє використовувати всі переваги екосистеми без додаткових

адаптацій. Також середовище розробки Visual Studio, яке оптимізовано під C#, значно пришвидшило реалізацію: завдяки автоматичним підказкам, перевірці типів у реальному часі, рефакторингу коду, підтримці налагодження та тестування.

Загалом, C# виступає не просто мовою програмування, а повноцінним інструментом для побудови надійного, масштабованого, функціонального та безпечного веб-додатку. Його використання дозволило реалізувати весь спектр функціональності — від обробки запитів користувача до збереження складної структурованої інформації про навчальний процес — на високому професійному рівні. Враховуючи активний розвиток .NET-платформи, підтримку хмарних рішень, кросплатформність і майбутнє розширення функціоналу (наприклад, мобільна версія або REST API), вибір C# є стратегічно правильним та довготривало ефективним рішенням у межах реалізованого проєкту.

1.2.3. Entity Framework Core — зручний та безпечний ORM для роботи з базами даних

У розробці сучасних веб-застосунків надзвичайно важливо забезпечити надійну, ефективну та масштабовану взаємодію з базою даних. Особливо це актуально для освітніх систем, де зберігається велика кількість структурованої інформації: користувачі, тести, теоретичні розділи, результати проходження завдань, відповіді на запитання тощо. У рамках реалізації цієї дипломної роботи для роботи з базою даних було обрано Entity Framework Core (EF Core) — об'єктно-реляційний мапер (ORM) нового покоління, створений корпорацією Microsoft спеціально для роботи з платформою .NET Core.

Entity Framework Core — це потужний інструмент, що дозволяє абстрагуватися від написання сирого SQL-коду й працювати з базою даних у вигляді класів і об'єктів, використовуючи синтаксис мови програмування C#. Такий підхід забезпечує високу читабельність, підтримуваність коду, а також зменшує ймовірність логічних і синтаксичних помилок при побудові запитів.

Замість класичного способу взаємодії з базами даних за допомогою SQL-запитів, розробник може описати структуру таблиці через C#-модель, після чого ця структура буде автоматично згенерована або синхронізована з реальною базою даних через механізм міграцій.

Однією з головних переваг EF Core є підтримка підходу Code First, який був застосований у даному проєкті. Він дозволяє створити всю логіку бази даних на основі класів, які описують об'єкти системи, — наприклад, `User`, `TheorySection`, `Example`, `Test`, `Question`, `Answer`, `Result`. Усі зв'язки між сутностями (один-до-багатьох, багато-до-багатьох) задаються безпосередньо в коді за допомогою анотацій або Fluent API. Це дозволяє розробнику повністю контролювати архітектуру даних на рівні об'єктної моделі без потреби вручну створювати SQL-схеми.

Зміни в структурі БД відслідковуються за допомогою міграцій (migrations), які фіксують усі трансформації моделі: додавання, перейменування, видалення полів чи таблиць. EF Core автоматично генерує SQL-код для оновлення бази на основі змін у моделях, що дозволяє безболісно адаптувати БД до нових вимог без втрати вже збережених даних. Це критично важливо для тривалої підтримки та розвитку системи, адже структура може змінюватися багаторазово — зокрема, під час додавання нових функцій (наприклад, рейтингів, коментарів, обліку часу на тестах тощо).

Ще однією вагомою перевагою EF Core є підтримка LINQ (Language Integrated Query) — вбудованої мови запитів у C#, яка дозволяє здійснювати вибірки, фільтрацію, сортування, об'єднання даних тощо у вигляді звичного синтаксису C#. Наприклад, замість написання SQL-запиту на кшталт `SELECT * FROM Results WHERE Score > 80`, достатньо написати:

1.2.4. Microsoft SQL Server

Ефективне зберігання, обробка, структуризація й захист даних — ключові завдання кожної сучасної програмної системи, зокрема веб-застосунків у сфері освіти. Особливе значення це має у випадку освітніх платформ, де зберігається

велика кількість критично важливої інформації: облікові записи користувачів, результати тестів, навчальні матеріали, журнал активності, дані авторизації та інші структуровані об'єкти.

У межах реалізації цієї дипломної роботи для побудови, адміністрування та супроводу бази даних було обрано Microsoft SQL Server — потужну, високонадійну та професійну систему керування реляційними базами даних (СКБД), яка є частиною екосистеми Microsoft. Вибір саме цієї платформи був зумовлений як технічними характеристиками продукту, так і його повною сумісністю з технологіями, що використовуються в межах .NET-розробки, зокрема — ASP.NET Core та Entity Framework Core.

Однією з основних переваг Microsoft SQL Server є його глибока інтеграція з платформою .NET, що забезпечує максимально зручну роботу з БД без додаткових бібліотек або адаптерів. Завдяки цьому в межах реалізованого проєкту вдалося налагодити повну двосторонню взаємодію між додатком на ASP.NET Core та БД через ORM Entity Framework Core, що значно спростило доступ до даних, реалізацію зв'язків між сутностями, а також виконання складних запитів.

SQL Server — це високопродуктивна СКБД, яка забезпечує стабільну роботу навіть при великих обсягах запитів та одночасній взаємодії з багатьма користувачами. Це є принципово важливим у рамках реалізації освітньої платформи, яка розрахована на те, що багато користувачів можуть одночасно працювати з теоретичними матеріалами, проходити тести, надсилати відповіді, зберігати результати. Microsoft SQL Server оптимізовано для масштабованості, і за потреби може бути адаптований до роботи у хмарному середовищі або з розподіленими серверами.

Крім високої продуктивності, Microsoft SQL Server також відзначається винятковою стабільністю та безпекою. Для освітніх систем, які працюють із персональними даними студентів, важливо забезпечити цілісність інформації, її недоступність для сторонніх осіб та можливість швидкого відновлення у разі збою. SQL Server надає вбудовані засоби резервного копіювання, шифрування

даних, контроль доступу на основі ролей, аудит змін, що було враховано у побудові системи. Наприклад, у даному проєкті використовувалася рольова модель доступу, де користувачі з різним рівнем доступу (адміністратор або звичайний користувач) мали доступ до різного обсягу даних, що відображено в архітектурі таблиць та контролерах програми.

Ще однією перевагою SQL Server є зручність адміністрування та моніторингу. Розробник або адміністратор системи має змогу використовувати SQL Server Management Studio (SSMS) — потужне графічне середовище для управління серверами, базами, таблицями, запитами, безпекою, бекапами. У межах цієї дипломної роботи SSMS використовувалося для візуального перегляду вмісту таблиць, ручного тестування SQL-запитів, моніторингу змін у структурі та ефективного управління транзакціями.

Слід також підкреслити, що SQL Server підтримує тригери, збережені процедури, функції, індекси, уявлення (views) та інші потужні засоби реляційного моделювання. Хоча в межах даної дипломної роботи основна логіка реалізована в коді C# через Entity Framework, у подальшому можливе розширення функціоналу за рахунок впровадження більш складних логік на рівні самої СКБД, що дає змогу оптимізувати швидкодію й зменшити навантаження на додаток.

Крім того, Microsoft SQL Server має чітко задокументований стандартний діалект SQL (T-SQL), який є розширенням ANSI SQL, тому його використання є надійною інвестицією в підтримувану та стандартизовану архітектуру. У разі масштабування платформи або передачі її на обслуговування іншим командам розробників використання такого стандартного середовища забезпечить зрозумілість і доступність технічної документації.

У межах даного проєкту структура бази даних, реалізована на Microsoft SQL Server, включає такі основні таблиці: [Users](#), [TheorySections](#), [Examples](#), [InteractiveTasks](#), [Tests](#), [Questions](#), [Answers](#), [Results](#). Усі таблиці взаємопов'язані зовнішніми ключами та мають індекси для забезпечення швидкого пошуку та

обробки даних. Додатково реалізовано механізми валідації, обробки винятків і контроль унікальності, що запобігає дублюванню інформації.

Отже, Microsoft SQL Server виступає не лише надійною основою для зберігання даних у межах освітньої платформи, а й стратегічно вдалим вибором завдяки своїй гнучкості, безпеці, масштабованості та глибокій інтеграції з інструментами Microsoft. Саме тому ця СКБД була обрана як основа для побудови реляційної бази у дипломному проєкті, що значно прискорило процес розробки, підвищило стабільність і забезпечило готовність системи до промислової експлуатації.

1.2.5. Інтерфейс користувача: HTML5, CSS3, Bootstrap, Razor

Одним із ключових елементів ефективної освітньої системи є інтерфейс користувача (UI) — спосіб взаємодії користувача з функціоналом платформи. Від його зручності, інтуїтивності та естетичної привабливості значною мірою залежить загальне враження від системи, рівень залученості, швидкість навчання та ефективність роботи. У межах розробки даного веб-застосунку було реалізовано повноцінний інтерфейс користувача на основі поєднання HTML5, CSS3, Bootstrap 5 та Razor Pages — сучасних технологій, які забезпечують адаптивність, кросбраузерність, простоту в розширенні та підтримці.

Основою побудови кожної веб-сторінки є HTML5 (HyperText Markup Language) — остання версія мови розмітки, яка дозволяє створювати структуру сторінок, описуючи їх елементи: заголовки, абзаци, списки, кнопки, форми, таблиці тощо. У контексті розробленої системи HTML5 використовувався для опису шаблонів сторінок: авторизації, перегляду теоретичних матеріалів, проходження тестів, відображення прикладів задач. Завдяки новим семантичним тегам (`<header>`, `<nav>`, `<article>`, `<section>`, `<footer>`) вдалося досягти логічної організації контенту, що полегшило його візуальне сприйняття та підтримку коду.

Щоб забезпечити візуальну привабливість, стильову єдність та адаптацію до різних розмірів екранів, було використано CSS3 (Cascading Style Sheets) —

технологію стилізації веб-елементів. Через каскадні таблиці стилів вдалося реалізувати кольорову гаму, розмітку блоків, відступи, тіні, анімації при наведенні тощо. Це дало змогу створити сучасний, зручний інтерфейс, що не перевантажений зайвими елементами та відповідає вимогам мінімалістичного дизайну. Усі стилі були централізовані у файлах, підключених до шаблонів сторінок, що дозволяє зручно вносити зміни без редагування кожної окремої сторінки.

Особливе місце в реалізації UI зайняв Bootstrap 5 — найпопулярніший у світі CSS-фреймворк, який значно спростив створення адаптивного й кросбраузерного інтерфейсу. Завдяки вбудованим класам і компонентам Bootstrap (карти, таблиці, кнопки, панелі навігації, модальні вікна, алерти, пагінація тощо) вдалося реалізувати сучасний функціонал за мінімального обсягу власного CSS-коду. Наприклад, всі форми — авторизації, реєстрації, проходження тестів — були реалізовані за допомогою стилізованих компонентів Bootstrap, що забезпечило їх коректне відображення на різних пристроях без необхідності створення окремої мобільної версії. Гнучка система сіток (Grid System) дала змогу розташувати елементи інтерфейсу в логічному порядку, незалежно від розміру екрана: від широкоформатних моніторів до мобільних телефонів.

Ще одним ключовим компонентом розробки UI стала технологія Razor Pages — серверна система шаблонізації, що є частиною ASP.NET Core. Razor поєднує HTML та C# у межах одного файлу, дозволяючи створювати динамічні сторінки, які реагують на дії користувача, відображають дані з бази та формують інтерфейс у реальному часі. Наприклад, сторінки, що відображають результати тестів, побудовані з використанням Razor-синтаксису `@foreach`, `@Model`, `@Html.DisplayFor`, що дає змогу легко та швидко генерувати HTML з динамічними даними. Razor також спростив розробку повторюваних елементів — навігаційних панелей, футерів, хедерів, — через часткові представлення (*Partial Views*), які підключаються до основних шаблонів.

Поєднання Razor і Bootstrap дозволило реалізувати динамічні компоненти, які миттєво реагують на взаємодію з користувачем: повідомлення про помилки, підказки при вводі, відображення результатів, табличні структури. Це суттєво підвищило інтерактивність системи та зробило взаємодію з нею приємною та інтуїтивно зрозумілою.

Крім того, використання стандартів HTML5 і CSS3 забезпечило повну кросбраузерну підтримку. Система була протестована у сучасних версіях браузерів Chrome, Edge, Firefox, Safari та продемонструвала стабільну роботу на всіх основних платформах. Це дає змогу без додаткових налаштувань запускати застосунок на будь-якому пристрої користувача: ноутбучі, планшеті, смартфоні, інтерактивній дошці тощо.

Важливим аспектом є також доступність інтерфейсу (accessibility) — система побудована таким чином, що елементи форми мають підписані мітки, достатній контраст, структуру розмітки, яка читається допоміжними технологіями (екранними дикторами). Це дозволяє використовувати систему не лише широкому загалу користувачів, а й людям з порушеннями зору.

Усі елементи інтерфейсу підтримують багатомовність завдяки можливості локалізації Razor-сторінок. У разі потреби реалізація багатомовної підтримки може бути здійснена шляхом підключення ресурсних файлів (resx) із перекладами та динамічної заміни контенту відповідно до мови інтерфейсу.

Отже, застосування поєднання HTML5, CSS3, Bootstrap 5 та Razor дозволило створити інтерфейс користувача, що відповідає найсучаснішим вимогам до веб-дизайну: він адаптивний, зрозумілий, технологічний і зручний для користувача. Усі вибрані технології підтримуються глобальною спільнотою, постійно оновлюються, мають велику кількість прикладів, документації й дають змогу легко розширювати або змінювати інтерфейс у майбутньому. У рамках освітньої системи це критично важливо, адже зручність подачі інформації прямо впливає на ефективність засвоєння знань і загальний користувацький досвід.

1.3. Аналіз подібних рішень

Перед розробкою власної освітньої системи доцільним є аналіз вже існуючих аналогічних платформ, які використовуються у світі для подачі навчального матеріалу, організації навчального процесу, тестування й контролю знань. Такий аналіз дозволяє визначити функціональні й технічні особливості конкурентних систем, виявити їхні сильні й слабкі сторони, а також обґрунтувати доцільність створення нового рішення з урахуванням недоліків існуючих підходів.

На сьогодні ринок електронних освітніх платформ охоплює як масові продукти для дистанційного навчання, так і вузькоспеціалізовані рішення для контролю знань, формування навчального контенту, адаптивного навчання. Серед найбільш поширених платформ доцільно виділити Moodle, Google Classroom, Prometheus, Quizizz та інші онлайн-сервіси, що обслуговують мільйони користувачів.

Moodle

Однією з найвідоміших і наймасштабніших освітніх платформ є Moodle — модульна система управління навчанням з відкритим вихідним кодом, розроблена для створення персоналізованих середовищ навчання. Вона підтримує створення курсів, завдань, тестів, форумів, файлообмін, гнучке налаштування прав доступу. Moodle має широку функціональність, активну спільноту розробників, велику кількість додатків і тем оформлення.

Однак, попри свої переваги, Moodle є досить складною в адмініструванні та розгортанні. Для використання системи необхідні навички налаштування веб-сервера, бази даних, а також значні ресурси на технічну підтримку. З точки зору користувача, інтерфейс Moodle часто вважається перевантаженим, а для розробників — надмірно складним для модифікацій.

Google Classroom

Google Classroom — безкоштовна хмарна платформа від Google, орієнтована на організацію навчального процесу. Вона дозволяє викладачам створювати курси, розміщувати завдання, тестувати студентів, вести

комунікацію через Google Meet та електронну пошту, зберігати файли на Google Drive. Основною перевагою системи є її тісна інтеграція з іншими сервісами Google, простота у використанні, інтуїтивно зрозумілий інтерфейс.

Проте Google Classroom більше підходить для супроводу традиційного навчання, а не для повноцінної самостійної роботи користувача. У системі обмежені можливості для створення інтерактивного контенту, немає модулів адаптивного навчання, мало аналітики, а її кастомізація майже відсутня.

Prometheus

Платформа Prometheus є українським представником масових відкритих онлайн-курсів (MOOC), що надає доступ до безкоштовних відеолекцій, презентацій, тестів та сертифікації від провідних викладачів. Prometheus базується на open-source-рішенні edX і орієнтований на широке коло слухачів.

Його функціонал зосереджений на подачі відеоматеріалу та проходженні тестів у рамках заздалегідь сформованого курсу. Система не передбачає глибокої персоналізації або вільної навігації, обмежує можливість створення власного контенту користувачем, а також не надає прикладів задач або теоретичної довідки в інтерактивному форматі.

Quizizz

Quizizz — онлайн-сервіс для створення і проходження інтерактивних тестів і вікторин у гейміфікованому форматі. Система орієнтована на залучення користувачів через візуальну динаміку, систему балів, таймерів, зворотного зв'язку та змагання в режимі реального часу.

Незважаючи на високий рівень інтерактивності, Quizizz має доволі обмежений функціонал у плані подачі теоретичного матеріалу та відсутності задач з відкритими відповідями. Цей сервіс є додатковим інструментом оцінювання, але не забезпечує повноцінного навчального процесу.

Порівняльна характеристика

Для більш наочного аналізу систем наведемо порівняльну таблицю:

Пл атформа	Те орія	За дачі	Те стування	Інт ерактивн ість	Гн учкість налашту вань	Ск ладність впровад ження
M oodle	+	+	+	±	+	Ви сока
Go ogle Classroo m	±	±	+	±	-	Ни зька
Pr ometheus	+	-	+	-	-	Се редня
Qu izizz	-	-	+	+	-	Ни зька
Вл асна система	+	+	+	+	+	Се редня

Як видно з аналізу, більшість систем або орієнтовані на подачу контенту (Moodle, Prometheus), або на перевірку знань (Quizizz), або є лише інструментами організації (Google Classroom), проте не об'єднують у собі весь повний цикл навчального процесу: вивчення — практика — перевірка — аналіз результатів.

Саме тому в межах цієї дипломної роботи була запропонована власна система, яка поєднує:

доступ до теоретичної довідкової бази з деревовидною структурою;
можливість вивчати задачі за темами;

реалізацію інтерактивних завдань для самоперевірки;
повноцінне тестування з результатами;
авторизацію користувачів, збереження статистики, модулі розмежування прав.

Таким чином, запропоноване рішення відповідає сучасним вимогам до освітнього програмного забезпечення, враховує недоліки існуючих платформ і пропонує збалансований підхід до реалізації всіх необхідних модулів у межах єдиного застосунку.

Висновок до розділу 1

У першому розділі дипломної роботи було здійснено глибокий теоретичний аналіз понять, принципів і технологій, пов'язаних зі створенням сучасних освітніх інформаційних систем. Проведене дослідження дозволило визначити основні характеристики таких систем, сформулювати критерії їх ефективності, а також виявити актуальні напрями розвитку освітніх технологій у цифровому середовищі.

На першому етапі розглянуто поняття освітньої системи, її місце в загальній структурі електронного навчання, функціональні компоненти, а також типологію — за форматом подачі матеріалу, взаємодією з користувачем та способом реалізації. Було з'ясовано, що сучасна освітня система не обмежується лише статичним поданням теоретичного контенту, а повинна включати засоби інтерактивної взаємодії, адаптацію до користувача, персоналізацію й елементи гейміфікації.

У другому підрозділі було досліджено технології реалізації освітніх веб-застосунків, зокрема мови програмування, фреймворки, системи управління базами даних, бібліотеки для побудови інтерфейсу. Було обґрунтовано, що для реалізації системи доцільно використовувати стек технологій на базі ASP.NET Core MVC, Microsoft SQL Server, Razor, HTML5, CSS3, Bootstrap, оскільки вони забезпечують баланс між потужністю, гнучкістю та простотою в обслуговуванні. Особлива увага приділялася можливості масштабування,

розширення функціоналу, безпеці та підтримці сучасних вимог до UX/UI-дизайну.

У межах третього підрозділу було здійснено аналіз існуючих рішень — як комерційних, так і відкритих освітніх платформ. Серед них — Moodle, Google Classroom, Khan Academy, Coursera та інші. Було виявлено, що, незважаючи на широку функціональність і популярність деяких систем, більшість з них є складними у використанні для кінцевого користувача, вимагають високих ресурсів для розгортання або не дозволяють гнучкого налаштування під специфіку конкретного навчального курсу. Таким чином, доцільність створення власної адаптованої платформи була повністю підтверджена результатами аналізу.

Окремо варто відзначити, що в ході вивчення теоретичної бази було сформульовано чітке бачення архітектури майбутньої системи, її функціональних модулів і логіки взаємодії між ними. Теоретичний розділ дозволив узагальнити світовий досвід та найкращі практики у сфері електронного навчання, що стало фундаментом для подальшої практичної реалізації.

У підсумку, розділ 1 сформував методологічну основу для створення ефективної, зручної та технологічно обґрунтованої освітньої системи, яка буде відповідати сучасним вимогам до цифрової педагогіки, автоматизації навчального процесу та персоналізованого підходу до користувача.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА ПОСТАНОВКА ЗАДАЧІ

2.1. Мета, завдання та цільова аудиторія системи

У сучасних умовах стрімкого розвитку цифрових технологій, освітні системи повинні відповідати не лише академічним, а й технічним та ергономічним вимогам. Вони мають забезпечувати простоту доступу до знань, ефективність у засвоєнні навчального матеріалу, підтримку зворотного зв'язку та об'єктивну систему оцінювання. Особливої актуальності набувають програмні рішення, які поєднують у собі різні формати подачі інформації — теоретичний контент, практичні вправи, інтерактивні елементи й тести — в єдиній інтегрованій платформі.

У цьому контексті головною метою розробки освітньої системи є створення веб-застосунку, що забезпечує повноцінну підтримку самостійного та контрольованого навчання шляхом поєднання теоретичної довідкової бази, прикладів задач, інтерактивних вправ та модуля тестування. Система повинна бути гнучкою, розширюваною, адаптивною до потреб користувачів різного рівня, простою у використанні та технічно стабільною.

Мета дипломної роботи полягає в розробці освітньої веб-платформи, яка надає користувачеві можливість систематизовано отримувати знання, закріплювати їх за допомогою практичних завдань та перевіряти рівень засвоєння матеріалу через тести з автоматичною перевіркою. Таким чином, система повинна стати ефективним інструментом неформального та напівформального навчання.

Для досягнення поставленої мети в межах розробки визначено такі основні завдання:

1. Провести аналіз існуючих освітніх рішень, виявити їхні переваги та недоліки, що можуть бути враховані при побудові власної системи.
2. Визначити функціональні та нефункціональні вимоги до веб-застосунку, зокрема: структура контенту, рівні доступу, механізми збереження та обробки даних.

3. Розробити архітектуру системи на основі моделі MVC, що передбачає розділення логіки на моделі, подання й контролери, забезпечуючи масштабованість і зручність супроводу.
4. Реалізувати модуль теоретичної довідки, який містить структуровану базу знань із підтримкою навігації та пошуку.
5. Створити модуль прикладів задач з можливістю фільтрації та тематичного групування.
6. Інтегрувати блок інтерактивних завдань, які забезпечують самоперевірку користувача в ігровому або прикладному форматі.
7. Розробити систему тестування, що включає конструктор тестів, проходження, перевірку результатів і зворотний зв'язок.
8. Забезпечити механізм авторизації та реєстрації користувачів, із збереженням історії активності, результатів тестування, ролей та прав доступу.
9. Реалізувати інтерфейс користувача, орієнтований на зручність, доступність і адаптивність для різних пристроїв.
10. Протестувати систему з метою перевірки стабільності, відповідності функціоналу вимогам, ефективності виконання основних сценаріїв.

Окрему увагу в процесі проєктування приділено визначенню цільової аудиторії, адже саме її потреби формують основу функціональних вимог до платформи. Очікується, що систему використовуватимуть наступні категорії користувачів:

Студенти та учні, які прагнуть самостійно опановувати навчальний матеріал у зручному темпі, з можливістю повторення, самоперевірки та накопичення результатів.

Викладачі, що потребують платформи для представлення навчального матеріалу в інтерактивному форматі, супроводу занять, створення тестів і контролю знань.

Самоосвітяни (лайфлонг-лернери), які не мають доступу до формальної освіти, але зацікавлені в оволодінні новими знаннями через якісний цифровий ресурс.

Організації та компанії, що здійснюють внутрішнє навчання або сертифікацію працівників за допомогою тестів та навчальних блоків.

Таким чином, платформа повинна поєднувати простоту у використанні для новачків і достатню гнучкість для більш досвідчених користувачів. Її функціонал має бути спрямований на забезпечення повного навчального циклу — від ознайомлення з матеріалом до самостійного тестування та аналізу результатів.

2.2. Архітектура та обґрунтування вибору технологій

Побудова сучасного освітнього веб-застосунку вимагає не лише продуманої логіки функціонування, а й чіткої технічної архітектури, яка дозволить реалізувати заданий функціонал ефективно, безпечно, масштабовано та зручно у подальшій підтримці. У процесі проєктування було прийнято рішення створити систему з чітким розподілом відповідальностей між логічними компонентами, дотримуючись принципів структурності, модульності та повторного використання коду. Саме тому архітектурну основу системи було сформовано на базі шаблону Model–View–Controller (MVC), який повністю підтримується фреймворком ASP.NET Core MVC.

Модель MVC передбачає розділення застосунку на три основні компоненти:

Model (модель) — відповідає за роботу з даними, бізнес-логіку, взаємодію з базою даних;

View (уявлення) — забезпечує візуальне подання даних користувачеві, реалізоване за допомогою Razor-синтаксису HTML/C#;

Controller (контролер) — приймає вхідні запити від користувача, викликає відповідні методи моделі та повертає уявлення.

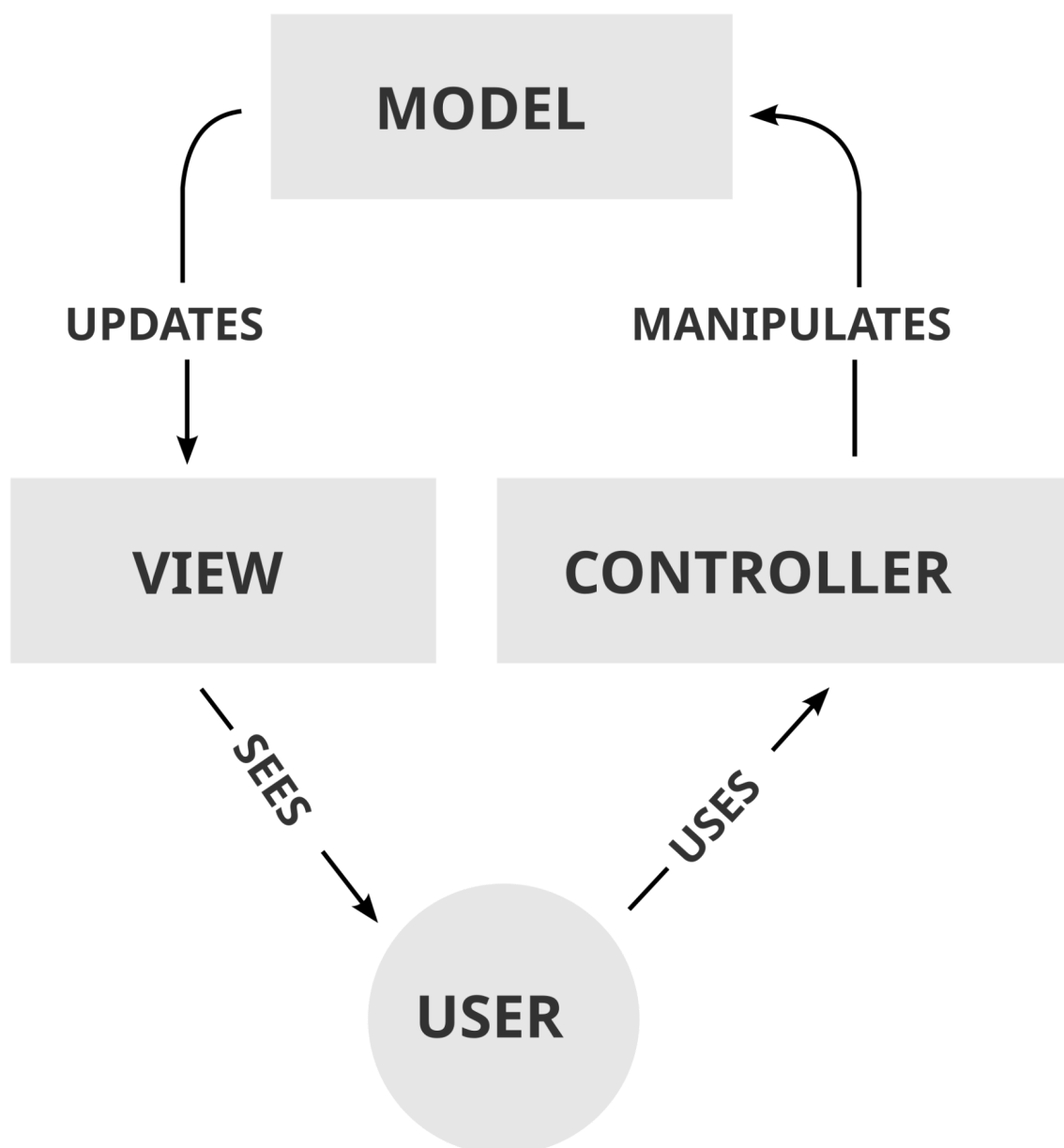


Рис. 2.1. Схема архітектури MVC

Цей підхід дозволяє ізолювати логіку обробки даних від інтерфейсу користувача, що значно спрощує як розробку, так і тестування, а також забезпечує можливість незалежного розвитку окремих модулів. Наприклад, можливо модифікувати вигляд тесту без змін у логіці перевірки або зберігання результатів у БД.

Система реалізована як веб-застосунок на ASP.NET Core MVC — сучасній кросплатформній платформі з відкритим кодом, яка підтримує розгортання на будь-яких операційних системах (Windows, Linux, macOS),

масштабування у хмарному середовищі, високу продуктивність і безпеку.

ASP.NET Core MVC обрано як основу проєкту завдяки:

- чіткому впровадженню архітектури MVC;
- підтримці маршрутизації, сесій, авторизації;
- зручній роботі з API;
- підтримці шаблонів Razor;
- інтеграції з Entity Framework Core для взаємодії з базами даних.

У якості мови програмування використовується C# (C-Sharp) — об'єктно-орієнтована мова, що дозволяє ефективно реалізовувати складну логіку, працювати з асинхронними операціями, гарантує типову безпеку й підтримує найкращі практики розробки. Вона є невід'ємною частиною .NET-екосистеми, має високу продуктивність і зручний синтаксис для реалізації навчальних сценаріїв: тестування, перевірки відповідей, фільтрації задач, авторизації тощо.

Для збереження та обробки даних обрано Microsoft SQL Server — потужну реляційну систему керування базами даних (СКБД), яка ідеально інтегрується з ASP.NET Core та забезпечує стабільну роботу з великими обсягами даних. База даних проєкту містить таблиці для зберігання інформації про користувачів, теми, задачі, тести, відповіді та результати. Усі зв'язки між сутностями реалізовано через зовнішні ключі та підтримку ORM.

Усі операції з базою даних реалізуються через Entity Framework Core — об'єктно-реляційний мапер, що дозволяє працювати з базою як із наборами об'єктів. Це знижує потребу в ручному написанні SQL-запитів і підвищує безпеку застосунку. Структура бази створюється за допомогою підходу Code First, що дозволяє синхронізувати класи моделей із базою за допомогою системи міграцій.

Клієнтська частина застосунку побудована на основі HTML5, CSS3, Bootstrap 5 і Razor Pages. HTML5 забезпечує структуровану розмітку сторінок, CSS3 відповідає за стилізацію й оформлення, а Bootstrap дозволяє створювати адаптивний і кросбраузерний інтерфейс без необхідності реалізації складних

UI-засобів з нуля. Razor Pages забезпечує гнучку систему шаблонів і динамічного відображення даних на стороні сервера, що є ідеальним у поєднанні з MVC.

Уся система логічно поділена на окремі модулі:

авторизація/реєстрація користувачів;

перегляд і пошук теоретичних матеріалів;

виконання задач з практичними прикладами;

проходження тестів із результатами;

керування базою даних через Entity Framework.

У результаті сформовано цілісну, логічну, масштабовану архітектуру, в якій кожен компонент працює незалежно, але узгоджено. Це дозволяє швидко адаптувати систему до нових вимог, додавати новий функціонал, змінювати інтерфейс або логіку без ризику порушення роботи всієї системи.

Таким чином, вибрані технології — ASP.NET Core MVC, C#, Entity Framework Core, SQL Server, Bootstrap 5, Razor Pages — забезпечують повну реалізацію вимог до сучасної освітньої системи, включаючи стабільність, зручність, безпеку, адаптивність і подальшу розширюваність платформи.

2.3. Загальна структура системи та модулів

Проектована освітня система має багаторівневу структуру, побудовану на основі архітектури MVC (Model–View–Controller) та розподілену на окремі логічні модулі, кожен з яких відповідає за виконання конкретної функції в межах навчального процесу. Така модульність забезпечує зручність в управлінні кодом, спрощує розробку, тестування та масштабування системи, а також сприяє розмежуванню обов’язків між розробниками у разі командної роботи.

Усі модулі інтегруються між собою в єдину платформу за рахунок централізованої маршрутизації, спільного середовища даних та уніфікованої бази даних. Система побудована таким чином, щоб користувач (незалежно від своєї ролі — студент, викладач, адміністратор) взаємодіяв з нею через зручний веб-інтерфейс, що забезпечує доступ до всіх основних функцій.

Загальна структура системи включає шість ключових модулів, кожен із яких виконує окрему роль у навчальному процесі:

2.3.1. Модуль авторизації та управління користувачами

Цей модуль відповідає за ідентифікацію користувачів, контроль доступу до ресурсів, розмежування прав. Він реалізує функції:

- реєстрації користувачів;
- входу з перевіркою логіна та пароля;
- збереження сесій;
- призначення ролей («користувач», «адміністратор»);
- зберігання історії активності.

Рольова модель забезпечує гнучке керування доступом до функцій системи. Наприклад, лише адміністратор має доступ до створення тестів або редагування теоретичних розділів.

2.3.2. Модуль теоретичної довідкової бази

Цей модуль забезпечує доступ користувача до структурованого навчального контенту. Теоретичні матеріали зберігаються у вигляді розділів, які формують ієрархічну структуру — подібну до підручника з темами й підтеми. Основні функції:

- навігація по темах через дерево розділів;
- відображення навчального тексту;
- пошук за ключовими словами;
- можливість гіперпосилань між темами.

Цей модуль є фундаментом системи, оскільки саме на основі теоретичної бази будується подальше закріплення знань через задачі та тести.

2.3.3. Модуль прикладів задач

Містить практичні завдання, які дозволяють користувачеві застосовувати знання на прикладах. Завдання класифікуються за темами, рівнем складності та типом. Функціональні можливості:

- перегляд списку задач за категоріями;
- сортування та фільтрація;

відображення умов, підказок та відповідей;

можливість самотійної перевірки або звернення до розв'язання.

Модуль спрямований на розвиток навичок самотійного мислення, логічного аналізу та закріплення теоретичних знань.

2.3.4. Модуль інтерактивних завдань

Реалізує навчальні приклади у форматі активної взаємодії: введення відповідей, вибір правильного варіанту, динамічна перевірка. Користувач одразу отримує зворотний зв'язок щодо результату. Переваги модуля:

гейміфікація процесу навчання;

зниження когнітивного навантаження;

стимулювання користувача до участі.

Інтерактивність реалізована через JavaScript + Razor + C#, з використанням AJAX для перевірки без перезавантаження сторінки.

2.3.5. Модуль тестування

Центральна частина контролю знань. Користувач проходить тестування, яке оцінюється автоматично за заздалегідь заданими правильними відповідями.

Основні функції:

створення тестів адміністратором;

випадкова генерація питань;

обмеження за часом;

збереження результатів у БД;

виведення оцінки, аналіз помилок.

Тести можуть охоплювати як теоретичні знання, так і навички застосування знань у завданнях.

2.3.6. Модуль аналітики та звітності

Реалізує збір і виведення статистичних даних про діяльність користувача:

переглянуті теми;

пройдені тести;

середній бал;

динаміка успішності;

аналіз типових помилок.

Цей модуль планується як розширення основної системи, особливо корисний для викладачів і адміністраторів у разі впровадження системи в освітню установу.

Уся система взаємодіє з єдиною базою даних Microsoft SQL Server, яка включає таблиці [Users](#), [TheorySections](#), [Examples](#), [InteractiveTasks](#), [Tests](#), [Questions](#), [Answers](#), [Results](#). Завдяки використанню Entity Framework Core зв'язки між сутностями реалізовані через зовнішні ключі, що забезпечує логічну цілісність та ефективне збереження даних.

Таким чином, структура освітньої платформи є модульною, масштабованою та логічно завершеною, що дозволяє забезпечити повноцінний навчальний процес — від теорії до контролю знань — у межах єдиного, зручного та доступного веб-середовища.

2.4. База даних і зберігання даних

Фундаментальним компонентом будь-якої веб-системи є база даних, яка забезпечує надійне зберігання, обробку та цілісність інформації. У контексті розробки освітньої платформи це особливо актуально, адже система має справу з великою кількістю взаємопов'язаних сутностей: користувачі, навчальні розділи, приклади задач, тести, результати проходження, відповіді тощо. Ефективна реалізація бази даних дозволяє досягти стабільності у зберіганні й обміні даними між модулями, а також забезпечити масштабованість і захищеність системи в довгостроковій перспективі.

База даних, реалізована у межах проєкту, побудована за реляційною моделлю з використанням Microsoft SQL Server. Ця система керування базами даних була обрана з огляду на її високу надійність, продуктивність, гнучку інтеграцію з платформою .NET і широкі можливості адміністрування. Робота з базою реалізована через Entity Framework Core — сучасну ORM-систему, яка дозволяє працювати з таблицями у вигляді об'єктів і класів мовою C#, без необхідності безпосереднього написання SQL-запитів. Такий підхід не лише

пришвидшує розробку, а й значно зменшує ризик помилок, підвищує безпеку й дозволяє централізовано керувати всією логікою збереження даних.

Архітектура бази даних формувалася на основі підходу Code First. Це означає, що таблиці створювалися автоматично на основі C#-моделей, які логічно відображають структуру предметної області. Кожен об'єкт у системі — будь то користувач, тест, теоретичний розділ чи результат — представлений окремим класом у коді, а зв'язки між сутностями реалізовані через властивості та зовнішні ключі. Така організація дозволяє підтримувати складні ієрархічні структури, наприклад, дерево теоретичних розділів або тест, який складається з багатьох запитань і варіантів відповідей.

Уся система зберігання даних побудована відповідно до принципів нормалізації: виключено дублювання, забезпечено цілісність записів, реалізовано жорсткі зв'язки між сутностями, що унеможлиблює появу «осиротілих» записів або конфліктів у даних. Усі зміни структури бази даних відстежуються за допомогою механізму міграцій, що дозволяє вносити доповнення або оновлення без втрати вже існуючої інформації.

Обробка даних відбувається через контекст бази — клас, який виступає точкою доступу до таблиць. Саме через цей механізм здійснюється запис результатів тестування, створення нових облікових записів, отримання теоретичних матеріалів, фільтрація задач. Для вибірки використовуються LINQ-запити, які дозволяють виконувати складну аналітику без необхідності ручного формування SQL-операторів, при цьому зберігаючи типову безпеку та зручність у читанні коду.

Безпека бази даних реалізована на кількох рівнях. Зокрема, облікові дані користувачів зберігаються не у відкритому вигляді, а у формі хешованих значень із використанням сучасного алгоритму шифрування. Крім того, система не допускає збереження некоректних або шкідливих даних завдяки попередній валідації на рівні серверної логіки. У критичних операціях, таких як одночасне створення результату тесту та відповіді, використовується механізм транзакцій,

який гарантує, що зміни будуть або повністю завершені, або повністю скасовані — залежно від успішності виконання всіх операцій.

Таким чином, побудована база даних є логічно завершеною, оптимізованою, масштабованою і відповідає всім вимогам до сучасного освітнього веб-сервісу. Вона гарантує цілісність інформації, забезпечує надійне збереження навчального контенту та результатів, дозволяє проводити гнучку аналітику й створює надійний фундамент для подальшого розширення функціоналу системи

Висновок до розділу 2

У другому розділі було здійснено комплексний аналіз функціональних і нефункціональних вимог до майбутньої освітньої системи, визначено основну мету, завдання й цільову аудиторію платформи, а також сформовано архітектурну модель та обґрунтовано вибір технологій для реалізації проєкту.

На основі аналізу особливостей сучасних навчальних процесів, специфіки подачі матеріалу в онлайн-форматі та потреб користувачів, було сформульовано головну мету — створити зручну й інтерактивну веб-платформу, яка поєднуватиме теоретичний контент, приклади задач, систему тестування та зберігання результатів. Запропонована система орієнтована на учнів, студентів, викладачів і всіх охочих, хто прагне засвоїти матеріал у зручній формі та в комфортному темпі.

Цільова аудиторія визначена як користувачі із середнім рівнем володіння комп'ютером та інтернетом, тому в процесі проєктування особлива увага приділялася простоті інтерфейсу, зрозумілості навігації, адаптивності та інтуїтивності взаємодії з системою. Також було враховано потенційні ролі користувачів: звичайний учень/студент, викладач/автор матеріалів, адміністратор — для кожного з яких реалізовується власна логіка доступу та функціональності.

У підрозділі, присвяченому аналізу архітектури, було обґрунтовано вибір структури MVC, що забезпечує чіткий розподіл відповідальностей між логікою, представленням і доступом до даних. Цей підхід дозволив спростити підтримку

коду, підвищити гнучкість у розширенні системи, а також реалізувати масштабовану інфраструктуру з можливістю подальшого переходу на клієнт-серверну архітектуру.

Також було детально проаналізовано існуючі технології для реалізації веб-застосунків та обрано оптимальні інструменти з огляду на завдання проєкту. Основними стали ASP.NET Core MVC, Razor Pages, Entity Framework Core, Microsoft SQL Server, HTML5, CSS3 та Bootstrap 5. Ці технології забезпечують стабільність, надійність, безпеку та адаптивність, а також підтримуються сучасними середовищами розробки та хостингу. Особливо важливо, що всі вони є безкоштовними, з відкритою документацією, і мають широку спільноту розробників.

Було представлено загальну структуру системи, що включає окремі модулі: теоретичну довідку, приклади задач, інтерактивні завдання, тести, модуль авторизації, збереження результатів, адміністративну частину. Кожен модуль виконує чітко визначену функцію та взаємодіє з іншими через спільні механізми обробки запитів і збереження інформації в базі даних.

Таким чином, другий розділ закладає фундамент для технічної реалізації проєкту. Проведене обґрунтування вибору засобів і технологій, побудова архітектури та опис призначення системи дозволили перейти до практичної фази розробки освітнього застосунку з повним розумінням функціональної структури, потенційних сценаріїв використання та технічних особливостей.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ОСВІТНЬОЇ СИСТЕМИ

3.1. Загальний опис реалізації

Процес реалізації освітньої системи охоплює створення повноцінного веб-застосунку, побудованого на сучасному стеку технологій .NET та HTML. Всі етапи розробки проводилися відповідно до попередньо визначених функціональних та нефункціональних вимог, описаних у попередніх розділах роботи. Основним завданням реалізації було створення стабільної, зручної та ефективної системи, яка дозволяє користувачам опановувати теоретичний матеріал, вирішувати задачі, проходити тести та зберігати результати навчання.

Розробка системи здійснювалася в інтегрованому середовищі Visual Studio 2022 з використанням платформи ASP.NET Core MVC. Було створено окремі проєктні структури для моделей, контролерів, уявлень, служб і конфігураційних файлів. Всі компоненти системи взаємодіють у межах архітектури MVC, що забезпечує чіткий розподіл логіки між рівнями: модель відповідає за структуру даних, контролер — за обробку запитів, а уявлення — за відображення контенту користувачеві.

Реалізація почалася зі створення базової інфраструктури застосунку. На цьому етапі було налаштовано підключення до бази даних Microsoft SQL Server через ORM Entity Framework Core, створено контекст даних та визначено основні моделі: користувач, розділ теорії, задача, тест, питання, відповідь, результат. Застосовано підхід Code First, що дозволило побудувати структуру бази безпосередньо з коду C#, з подальшим автоматичним створенням таблиць за допомогою механізму міграцій.

У наступному етапі були реалізовані механізми реєстрації та авторизації користувачів. Вони забезпечують збереження облікових даних у безпечному форматі (хешування паролів), автентифікацію сесії, а також розмежування прав доступу відповідно до ролі (звичайний користувач або адміністратор). Цей функціонал є критично важливим для контролю доступу до адміністративної частини, тестів і результатів.

Після налаштування системи ідентифікації було реалізовано модуль теоретичного контенту. Теоретичні матеріали зберігаються у базі у вигляді структури розділів із можливістю вкладення (через рекурсивне дерево). Уявлення були побудовані з використанням Razor-синтаксису, що дозволяє динамічно формувати вміст сторінок у залежності від обраної теми або підтем.

На основі теоретичного блоку було створено модуль прикладів задач, де реалізовано механізм відображення опису задачі, варіантів вирішення, рівнів складності та фільтрації. Цей модуль є логічним продовженням теоретичного, дозволяючи застосувати знання на практиці.

Ключовим елементом реалізації став модуль тестування. Для цього було створено механізм генерації тестів із бази питань, обробки відповідей, перевірки правильності, обчислення підсумкової оцінки та збереження результатів у базу даних. Реалізація тестування передбачає як варіанти з вибором правильної відповіді, так і потенційно відкриті питання (у розширеній версії). Після проходження тесту користувач отримує детальний зворотний зв'язок щодо правильності відповідей і загального балу.

Крім основного функціоналу, була реалізована панель адміністратора, доступна лише для користувачів із відповідною роллю. Через цю панель адміністратор може додавати нові тести, редагувати теоретичні матеріали, переглядати статистику користувачів і результати. Вся логіка реалізована через окремі контролери, а доступ обмежено атрибутами авторизації.

Усі інтерфейсні компоненти створено з використанням HTML5, CSS3 та фреймворку Bootstrap 5, що забезпечило адаптивний дизайн, підтримку мобільних пристроїв, а також відповідність сучасним принципам UI/UX-дизайну. Razor-уявлення доповнюють інтерфейс динамікою та дозволяють виводити дані з моделей безпосередньо на сторінках, уникаючи зайвих API-запитів.

Загальна реалізація системи демонструє повну відповідність попередньо поставленим функціональним вимогам. Завдяки поєднанню сучасних технологій, чіткої архітектури, продуманої структури даних та гнучкої логіки

було створено систему, яка забезпечує користувачеві повноцінну можливість вивчення матеріалу, закріплення знань на практиці та перевірки засвоєного через контрольні завдання. У наступних підрозділах буде детально описано кожен з модулів, технічні аспекти реалізації та приклади фрагментів коду.

3.2. Реалізація модуля авторизації

Модуль авторизації є одним із ключових елементів розробленої освітньої системи, оскільки він забезпечує захищений доступ до персоналізованого контенту, контроль доступу до ресурсів залежно від ролі користувача, а також безпеку обробки особистих даних. Від правильності реалізації цього модуля залежить стабільність роботи всієї системи, а також її відповідність вимогам до конфіденційності, верифікації та ролевої моделі.

Реалізація авторизації здійснена на основі вбудованих механізмів ASP.NET Core Identity, однак із використанням спрощеної власної логіки — без повної залежності від готового Identity-фреймворку. Такий підхід дозволив повністю контролювати структуру таблиці користувачів у базі даних, реалізувати гнучку логіку автентифікації, а також інтегрувати рольову модель у межах існуючої архітектури застосунку.

На рівні моделі була створена сутність **User**, яка містить поля: унікальний ідентифікатор, логін (ім'я користувача), хешований пароль та роль. Замість зберігання паролів у відкритому вигляді застосовано шифрування за допомогою алгоритму хешування (наприклад, SHA256 або BCrypt). При реєстрації нового користувача введений пароль перетворюється у хеш і лише тоді записується до бази даних. Це гарантує, що навіть у разі компрометації даних доступ до реальних паролів отримати неможливо.

Процес реєстрації реалізується через окрему форму, де користувач вводить логін та пароль. Після валідації введених даних система перевіряє, чи не існує користувача з таким іменем, і якщо перевірка успішна — створює новий запис у базі, присвоюючи роль "user" за замовчуванням. Фрагмент коду, який реалізує реєстрацію, подано нижче:

Під час авторизації (входу до системи) введені користувачем облікові дані порівнюються з тими, що збережені у базі. При збігу логіна та хешу пароля створюється сесія, яка містить дані користувача та роль. Сесійні дані зберігаються у cookie, що забезпечує збереження автентифікації між сторінками.

Контролери та уявлення, які потребують захищеного доступу, обгорнуті атрибутом `[Authorize]`, а для окремих дій — ще й додатковою перевіркою ролі (`if (User.IsInRole("admin"))`). Завдяки цьому реалізована повноцінна рольова модель, яка обмежує доступ до адміністрування, редагування контенту або перегляду чужих результатів.

Особливу увагу приділено обробці помилок входу. У разі неправильного пароля користувач отримує повідомлення без розкриття зайвих технічних деталей. Усі дії, пов'язані з авторизацією, логуються, що дозволяє у майбутньому впровадити аналіз активності, блокування підозрілих спроб або сповіщення про несанкціоновані дії.

Реалізований модуль є максимально простим у використанні для користувача та водночас ефективно захищеним з технічного боку. Система легко масштабується для впровадження додаткових ролей (наприклад, викладач, модератор) або інтеграції з зовнішніми сервісами авторизації (OAuth, Google, Microsoft тощо). Гнучкість реалізації забезпечує збереження незалежності від складних фреймворків і відкриває можливості для кастомізації відповідно до потреб освітньої платформи.

Таким чином, модуль авторизації повністю задовольняє вимоги до сучасних освітніх систем: забезпечує безпечний доступ, дозволяє розмежовувати права користувачів, легко інтегрується в загальну архітектуру платформи та створює підґрунтя для роботи з персоніфікованим навчальним середовищем.

3.3. Теоретична довідкова база

Модуль теоретичної довідкової бази є основним джерелом навчальної інформації для користувача в межах реалізованої освітньої платформи. Саме він забезпечує представлення структурованого навчального контенту, який формує базу знань для подальшого вивчення, виконання задач та проходження тестів. Цей модуль виконує роль електронного підручника з можливістю гнучкої навігації, перегляду змісту, збереження логіки розділів і швидкого доступу до потрібної теми.

Реалізація модуля здійснювалася на основі принципу ієрархічної побудови контенту. Теоретичні матеріали організовані у вигляді дерева розділів, кожен з яких може містити підрозділи та пов'язаний текстовий контент. Таке представлення дозволяє легко масштабувати систему, додаючи нові теми або поглиблення до вже існуючих, не порушуючи загальної логіки. У структурі бази даних реалізовано самозв'язки — кожен розділ може мати поле `ParentId`, що дозволяє формувати багаторівневу систему, де, наприклад, головна тема містить підтематику, яка, в свою чергу, може мати власні підрозділи.

Користувачеві представлено зручний інтерфейс навігації, де всі розділи виводяться у вигляді дерева або списку з індикаторами вкладеності. При виборі будь-якого пункту відбувається завантаження відповідного контенту на основній частині сторінки без повного перезавантаження. Це реалізовано за допомогою Razor Pages у поєднанні з AJAX-запитами, що значно підвищує швидкодію і зменшує навантаження на сервер. Усі сторінки відображаються з використанням HTML5 та CSS3, що забезпечує адаптивність інтерфейсу й доступність контенту на будь-якому пристрої — від стаціонарного ПК до мобільного телефону.

Редагування контенту можливе лише для адміністратора. Через окрему панель адміністрування реалізована можливість додавання нових розділів, редагування назви, змісту, порядку відображення, встановлення зв'язків між темами. Весь навчальний контент зберігається у форматі HTML у базі даних,

що дозволяє використовувати форматування тексту, додавання зображень, списків, посилань і виділень.

Особливу увагу при розробці модуля приділено зручності читання та простоті візуального сприйняття. Всі теоретичні матеріали відображаються зі збереженням логічної структури: заголовки, абзаци, ключові терміни, таблиці або формули (у разі потреби) мають чітке візуальне розділення. Це дозволяє користувачеві швидко орієнтуватися в тексті та ефективно сприймати інформацію.

Додатково реалізовано функцію пошуку за ключовими словами, яка дозволяє знайти відповідну тему незалежно від її вкладеності в структурі. Пошукова система працює за принципом фільтрації заголовків та змісту, а результати виводяться у вигляді посилань на відповідні розділи.

Уся логіка модуля реалізована в межах MVC-архітектури: моделі визначають структуру розділів та зв'язки між ними, контролери відповідають за обробку запитів на перегляд, редагування та збереження, а уявлення формують візуальне представлення для користувача. Завдяки використанню Entity Framework Core, додавання або редагування розділів здійснюється через контекст бази безпосередньо на рівні об'єктної моделі, що забезпечує стабільність та простоту в підтримці.

Таким чином, реалізований модуль теоретичної довідкової бази є ядром освітньої системи, яке поєднує зручність, функціональність, розширюваність і логічну завершеність. Він створює основу для інших модулів (прикладів задач, тестування), забезпечуючи користувача необхідними знаннями для ефективного навчання в інтерактивному середовищі.

3.4. Приклади задач і завдань

Модуль прикладів задач і завдань є важливим складником навчального процесу, оскільки він дозволяє користувачеві перевірити рівень розуміння теоретичного матеріалу на практиці. Цей модуль реалізує принципи активного навчання, коли студент не просто ознайомлюється з інформацією, а застосовує

знання до розв'язання конкретних прикладних ситуацій. Задачі виступають проміжною ланкою між вивченням теми та контролем знань через тестування.

На відміну від модулів, які просто подають інформацію, блок прикладів задач реалізований як інтерактивний навчальний інструмент, що дозволяє закріплювати матеріал шляхом розв'язання теоретично-орієнтованих і практичних питань. Кожна задача містить короткий опис умови, додаткову інформацію (наприклад, формули, схеми, таблиці) та одне або кілька запитань до користувача. Задачі згруповані за тематиками відповідно до теоретичних розділів, що дозволяє швидко знайти матеріал, пов'язаний із певною темою.

Структурно кожна задача реалізована як окремий запис у базі даних, що містить такі основні атрибути: назву, опис, категорію, рівень складності, а також правильну відповідь або варіанти відповідей (залежно від типу завдання). До задач також може додаватися приклад розв'язання, який активується за бажанням користувача. Такий підхід дозволяє надати студенту можливість спочатку самостійно спробувати розв'язати завдання, а вже потім ознайомитися з алгоритмом вирішення.

Інтерфейс модуля реалізований у вигляді списку задач з можливістю фільтрації за категорією та складністю. Користувач може переглядати задачі у вибраній темі, переходити до конкретного завдання, переглядати його умову, вводити або обирати відповідь та одразу отримувати результат перевірки. Візуальна частина реалізована з використанням Razor-шаблонів та Bootstrap, що забезпечує адаптивність відображення та зручність навігації. У разі правильної відповіді система надає позитивний зворотний зв'язок, а у випадку помилки — підказку або пояснення.

З технічної точки зору реалізація модуля передбачає взаємодію між кількома сутностями бази даних. Задачі зв'язані з відповідними теоретичними розділами через зовнішній ключ, що дозволяє динамічно формувати логічні ланцюжки «теорія — приклади — тести». Модель задач має гнучку структуру, що дозволяє зберігати як задачі з відкритою відповіддю, так і завдання з множинним вибором.

Особливістю реалізації є підтримка інтерактивних перевірок без перезавантаження сторінки. Для цього використано JavaScript у поєднанні з AJAX-запитами до серверного контролера, який обробляє введену відповідь і повертає результат перевірки у реальному часі. Такий підхід підвищує динаміку навчання та створює ефект діалогу між системою і користувачем.

Модуль задач також розрахований на розширення. Адміністратор може додавати нові завдання через окремий інтерфейс керування, вказуючи всі необхідні атрибути та зберігаючи задачу в базу даних. У майбутньому система може бути доповнена автоматичним формувачем наборів задач або гейміфікованими сценаріями з обмеженням у часі.

Загалом модуль прикладів задач і завдань виконує важливу функцію закріплення знань, сприяє активному залученню користувача до процесу навчання, підвищує ефективність засвоєння матеріалу та формує підґрунтя для впевненого проходження підсумкових тестів.

3.5. Модуль інтерактивних прикладів

Модуль інтерактивних прикладів виконує ключову функцію в реалізації принципу активного навчання. На відміну від традиційного вивчення матеріалу, де знання засвоюються пасивно, інтерактивний підхід передбачає безпосередню участь користувача в навчальному процесі. Саме тому в межах освітньої платформи було реалізовано окремий модуль, який дозволяє користувачам взаємодіяти із завданнями в реальному часі та одразу отримувати зворотний зв'язок про правильність виконання.

Цей модуль містить спеціально створені завдання, орієнтовані на миттєву перевірку знань. Користувачеві пропонується сформулювати відповідь у текстовому полі, вибрати правильний варіант із кількох запропонованих або перетягнути елементи для формування логічної пари. Після надсилання відповіді система автоматично аналізує її правильність та виводить відповідне повідомлення: підтвердження правильності, коротке пояснення або підказку.

Така модель стимулює користувача мислити активно, не боятися помилятися й одразу коригувати власне розуміння матеріалу.

Реалізація модуля здійснена з використанням Razor-шаблонів для динамічного формування задач та JavaScript у поєднанні з AJAX-запитами для обробки відповідей. Завдяки цьому користувач може взаємодіяти з завданням без перезавантаження сторінки, що створює ефект живого діалогу між системою та студентом. Також це дозволяє підвищити швидкість роботи інтерфейсу та знизити навантаження на сервер, оскільки оновлюється лише частина вмісту сторінки.

Кожен інтерактивний приклад містить не лише питання і варіанти відповіді, але й пояснення, чому саме ця відповідь правильна або неправильна. Таким чином, навіть у разі помилки користувач не залишається без підтримки — він отримує нову інформацію, яка допомагає краще засвоїти матеріал.

З боку бази даних інтерактивні приклади реалізовані як окрема сутність, що містить текст питання, правильну відповідь, пояснення, а також тип взаємодії (вибір, введення, співставлення). Це дозволяє в майбутньому легко розширити систему новими видами завдань або адаптувати її до специфіки певного навчального предмету.

Окремо варто відзначити зручність адміністрування: адміністратор системи може створювати або редагувати інтерактивні приклади через адміністративну панель. Там задаються основні параметри завдання, правильна відповідь та текст пояснення. Всі зміни зберігаються в базі даних і одразу відображаються на клієнтській стороні без необхідності перевантаження всього сайту.

Інтерактивні приклади є проміжною ланкою між вивченням теорії та фінальним тестуванням. Завдяки зворотному зв'язку в реальному часі, вони виконують роль самоперевірки, дозволяючи користувачу усвідомити прогалини в знаннях до того, як розпочнеться оцінювання. У майбутньому модуль може бути доповнений рейтингами, прогрес-барами або бонусною системою для гейміфікації навчального процесу.

Загалом, модуль інтерактивних прикладів створює ефективне, мотивуюче та зручне середовище для навчання. Він поєднує сучасні технології, педагогічні підходи та технічну реалізацію, що дозволяє досягти високого рівня залученості та успішності користувача у засвоєнні знань.

3.6. Модуль тестування

Модуль тестування є невід'ємною частиною освітньої системи та виконує ключову функцію контролю знань користувачів. Саме через тести здійснюється оцінювання рівня засвоєння теоретичного матеріалу, виявлення прогалин у знаннях, а також формування об'єктивної статистики навчального процесу. Реалізація цього модуля базується на принципах автоматизації, гнучкості структури тестів і миттєвого зворотного зв'язку для користувача.

Тестування у системі побудоване у вигляді набору запитань, кожне з яких має один або декілька варіантів відповіді. Кожен тест пов'язаний із певною темою або групою тем, і формується на основі завчасно підготовленої бази питань. Це дозволяє реалізовувати як тематичні, так і загальні підсумкові тести, що охоплюють кілька розділів. Користувач має можливість проходити тести у зручний час, без обмеження кількості спроб, якщо не вказано інше.

Архітектурно тест представлений як окрема сутність у базі даних і має зв'язок з таблицею питань, які у свою чергу зв'язані з варіантами відповідей. Така структура дозволяє будувати тести довільної складності, динамічно керувати їхнім складом, а також створювати питання із кількома правильними відповідями. Під час проходження тесту система формує список запитань та варіантів відповідей у випадковому або фіксованому порядку (в залежності від конфігурації), після чого користувач по черзі відповідає на них.

На клієнтському рівні для зручності реалізовано покрокову навігацію: кожне питання відображається окремо, з можливістю повернення до попереднього, або ж усі питання можуть бути показані одночасно у вигляді списку. Усі відповіді обробляються на стороні клієнта, а при завершенні тесту — надсилаються до серверної частини для перевірки. Перевірка здійснюється порівнянням обраних користувачем варіантів із правильними, які зберігаються в

базі даних. За кожну правильну відповідь нараховується певна кількість балів, і на основі підсумкової суми система визначає відсоток правильності або оцінку.

Після завершення тесту користувач одразу отримує зворотний зв'язок: загальну кількість балів, відсоток правильних відповідей, а також за бажанням — детальний аналіз кожного питання із вказівкою, які відповіді були правильними. Результати зберігаються в таблиці **Results** у базі даних разом із датою проходження, ідентифікатором користувача та тесту, що дозволяє формувати історію успішності, обчислювати динаміку навчання та надавати викладачам або адміністраторам доступ до аналітики.

Особливістю реалізації є те, що всі перевірки виконуються на серверній стороні, що гарантує захист від маніпуляцій із боку користувача. Крім того, у разі потреби в майбутньому система може бути доповнена обмеженням у часі, блокуванням перегляду правильних відповідей до завершення тесту або автоматичним генеруванням тестів із банку запитань за заданими параметрами.

Інтерфейс модуля тестування побудований за допомогою Razor-уявлень та стилізований із використанням Bootstrap. Це забезпечує зручне візуальне представлення питань і відповідей, чітке структурування блоку результатів, а також повну адаптивність під мобільні пристрої. Усі форми інтерактивні й не потребують повного перезавантаження сторінки завдяки використанню JavaScript та часткових оновлень.

Модуль також має адміністративну частину, де користувач із відповідною роллю може створювати, редагувати або видаляти тести, додавати питання, налаштовувати правильні відповіді та встановлювати параметри проходження. Завдяки цьому система легко масштабується під будь-який навчальний курс або предметну область.

У результаті реалізації модуля тестування було створено гнучкий, надійний і ефективний інструмент контролю знань, який інтегрується з іншими компонентами освітньої системи, зберігає результати, дозволяє аналізувати успішність користувачів та підтримує логіку безперервного навчання.

3.7. Обробка винятків і валідація даних

Під час розробки веб-системи особливу увагу було приділено обробці виняткових ситуацій та валідації введених користувачем даних. Це дозволило забезпечити стабільну роботу застосунку, запобігти некоректному збереженню інформації, підвищити рівень безпеки та покращити досвід взаємодії користувача з системою.

Валідація даних реалізується на двох рівнях: на стороні клієнта (front-end) та на стороні сервера (back-end). На клієнтському рівні використовуються HTML-атрибути валідації у поєднанні з Bootstrap-класами, що забезпечують миттєвий візуальний зворотний зв'язок. Наприклад, при спробі надсилання форми без заповнення обов'язкових полів система автоматично підсвічує помилки та не дозволяє надіслати форму доти, доки вони не будуть виправлені.

На серверному рівні валідацію забезпечено через атрибути `DataAnnotations` у моделях. Наприклад, поля моделі `User` мають атрибути `[Required]`, `[StringLength]`, `[RegularExpression]`, що гарантує перевірку логіну, паролю, структури електронної адреси тощо. Під час обробки запиту контролер перевіряє стан моделі за допомогою `ModelState.IsValid`, і в разі виявлення помилок — повертає користувача до форми з відповідними повідомленнями.

Особливу увагу приділено захисту від шкідливих даних та SQL-ін'єкцій. Завдяки використанню Entity Framework Core, усі запити до бази даних автоматично параметризуються, що унеможливлює виконання шкідливого SQL-коду навіть при навмисному введенні некоректних параметрів. Для додаткового захисту реалізована обробка винятків типу `DbUpdateException`, яка перехоплює помилки під час запису до БД (наприклад, у разі дублювання унікального поля або порушення зовнішнього ключа).

Крім валідації, у застосунку реалізовано централізовану обробку виняткових ситуацій. Більшість дій, які передбачають доступ до даних за ідентифікатором, перевіряють наявність об'єкта в базі. Якщо запис не знайдено, система повертає сторінку з повідомленням «Запис не знайдено» або

перенаправляє на безпечну сторінку. Це запобігає появі винятку `NullPointerException`, який міг би призвести до зупинки виконання програми.

Також обробляються помилки входу в систему. У разі неправильного пароля або відсутності користувача система не надає деталізовану інформацію (з міркувань безпеки), а лише загальне повідомлення про помилку автентифікації. Це захищає систему від спроб підбору пароля або соціальної інженерії.

В межах модулів тестування та задач реалізовано валідацію на відповідність формату (наприклад, числові відповіді, мінімальна довжина тексту), а також захист від повторного надсилання форми (через перевірку токена або блокування кнопки після першого натискання).

Загалом, реалізація надійної системи валідації та обробки винятків дозволила досягти високої стабільності платформи, захистити її від типових помилок і забезпечити коректну взаємодію з користувачем навіть у нештатних ситуаціях. Це особливо важливо для освітньої системи, яка передбачає постійну взаємодію з великою кількістю різних за рівнем підготовки користувачів.

3.8. Рольова модель доступу та безпека

У межах реалізації веб-застосунку особливу увагу було приділено організації системи доступу до функціоналу на основі ролей користувачів, а також забезпеченню безпечної обробки даних. Рольова модель дозволяє диференціювати рівень доступу до інформації, обмежити повноваження користувачів і захистити критичні компоненти платформи від несанкціонованого втручання.

У системі реалізовано дві основні ролі: звичайний користувач та адміністратор. Кожному зареєстрованому користувачу за замовчуванням присвоюється роль "user", тоді як адміністратори мають розширені права — створення й редагування навчального контенту, управління тестами, перегляд результатів інших користувачів та доступ до адміністративної панелі. Ролі зберігаються безпосередньо в таблиці `Users` у вигляді окремого текстового поля, що дозволяє здійснювати просту та гнучку перевірку на стороні сервера.

На рівні серверної логіки доступ до функцій контролюється за допомогою атрибутів `[Authorize]`, які забезпечують обмеження доступу лише для автентифікованих користувачів. Для сторінок і дій, доступних виключно адміністраторам, використовується уточнення `[Authorize(Roles = "admin")]`, що гарантує недоступність цих розділів для звичайних користувачів навіть у разі спроби прямого переходу за URL-адресою. Це дозволяє централізовано контролювати доступ без потреби реалізовувати перевірку вручну в кожному методі.

Крім обмеження за ролями, система реалізує перевірку автентифікації на основі claims-ідентифікації. Під час входу користувача до системи створюється об'єкт `ClaimsPrincipal`, до якого включається ім'я користувача та його роль. Цей об'єкт зберігається в cookie й автоматично додається до контексту запиту, що дозволяє використовувати методи `User.Identity.IsAuthenticated` і `User.IsInRole(...)` для перевірки статусу користувача в будь-якій частині застосунку.

Окрему увагу приділено захисту сесії користувача. Усі cookie мають встановлені атрибути `HttpOnly` та `Secure`, що запобігає доступу до них через JavaScript і забезпечує їх передавання лише через HTTPS. Крім того, передбачена можливість автоматичного завершення сесії після заданого періоду бездіяльності користувача.

Система захищена від несанкціонованого доступу до приватних ресурсів: сторінки, які потребують автентифікації, недоступні неавторизованим користувачам, і в разі спроби доступу — система перенаправляє на сторінку входу. Для додаткової безпеки введена перевірка при спробах змінити або видалити дані, які не належать поточному користувачу.

З точки зору загальної безпеки, у платформі реалізовані такі заходи:

- паролі зберігаються виключно у вигляді хешів (алгоритм BCrypt);
- форми захищені від CSRF-атак за допомогою токенів перевірки;
- запити до бази не містять відкритих SQL-інструкцій (завдяки ORM);
- контролюється довжина, тип та формат вхідних даних;

при вході й реєстрації усі помилки обробляються так, щоб не розкривати внутрішню логіку системи.

У майбутньому структура ролей може бути розширена за рахунок введення додаткових рівнів доступу (наприклад, «викладач», «модератор», «контент-менеджер») або інтеграції із зовнішніми системами автентифікації через протоколи OAuth 2.0 або OpenID Connect.

Отже, впроваджена рольова модель забезпечує чітке розмежування прав доступу, а комплекс заходів безпеки — стабільність, конфіденційність і цілісність даних, що особливо важливо для систем, які працюють із персональною та навчальною інформацією користувачів.

3.9. Тестування функціональності системи

Після завершення основного етапу розробки освітньої системи було проведено поетапне тестування всіх функціональних модулів. Метою тестування було виявлення можливих помилок у логіці роботи, перевірка коректності відображення інформації, стабільності роботи під час взаємодії з базою даних, а також правильності обробки нестандартних ситуацій. Тестування здійснювалося вручну, у середовищі браузера, під реальними сценаріями користування.

Усього було створено 10 тестових облікових записів з різними правами доступу. Зокрема, 8 користувачів із роллю "user" та 2 — з роллю "admin". Це дозволило перевірити, як система поводить себе при вході під різними обліковими даними, чи правильно обмежено доступ до адміністративного функціоналу, і чи відповідає інтерфейс обраній ролі.

На етапі тестування модуля авторизації перевірялися сценарії реєстрації, повторної реєстрації з тим самим логіном, входу з правильними й неправильними паролями, виходу із системи, збереження сесії після оновлення сторінки. Була виявлена помилка в обробці неіснуючого користувача: при спробі входу з неіснуючим логіном система спершу видавала помилку `NullPointerException`. Після виправлення додано перевірку на `null` перед перевіркою хешу пароля.

Модуль теоретичних матеріалів перевірявся на правильність виводу тексту, коректну роботу дерева розділів, можливість переходу між підтемами, пошук по ключових словах. Також тестувалася стабільність при роботі з великою кількістю вкладених підрозділів. У результаті було вдосконалено механізм побудови навігації, додано фіксацію активного пункту при навігації по сторінці.

Модуль прикладів задач було перевірено на здатність відображати завдання різної складності, зображення, математичні формули. Було протестовано сценарії з неправильним введенням відповіді, повторною перевіркою та відображенням пояснення після натискання кнопки «Показати розв’язання». При введенні порожнього поля система коректно спрацьовувала та видавала повідомлення з підказкою.

Модуль тестування тестувався найретельніше, оскільки він виконує функцію контролю знань. Було створено кілька тестів із різною кількістю питань та типами відповідей. Тестувалися: початок тесту, навігація між запитаннями, вибір відповідей, завершення проходження, підрахунок результату, збереження в базу даних. Перевірено, що користувач не може пройти тест без відповіді на всі питання — система відображає повідомлення та блокує завершення. Була протестована обробка некоректних запитів — наприклад, спроба пройти неіснуючий тест або змінити URL вручну — у таких випадках система правильно повертала сторінку помилки.

Крім основних модулів, тестувалася адміністративна панель: додавання нових теоретичних тем, створення тестів, редагування питань, видалення результатів. Виявлено, що після редагування тесту без перезавантаження сторінки оновлення не відображалось — виправлено через додатковий **RedirectToAction** у контролері після **POST**.

Було здійснено перевірку валідації форм: всі поля, які обов’язкові, марковані відповідними атрибутами на клієнтській та серверній стороні. Наприклад, при введенні короткого пароля, невалідної пошти або пустого поля система миттєво відображає повідомлення, а сервер блокує збереження.

Окрім стандартного функціонального тестування, перевірялася робота системи у трьох різних браузерях (Chrome, Firefox, Edge) на двох пристроях (ПК і смартфон). Інтерфейс повністю адаптивний, верстка зберігається коректно, усі функції доступні й працюють стабільно.

У підсумку проведене тестування підтвердило стабільність та працездатність усіх основних функціональних модулів системи. Було виявлено кілька незначних логічних помилок, які були своєчасно усунені, а також проведена оптимізація ряду візуальних і навігаційних рішень. Результати тестування дозволяють зробити висновок про готовність системи до практичного використання та подальшого впровадження в навчальний процес.

3.10. Можливості масштабування і модернізації

Один із важливих критеріїв якості програмного забезпечення — його здатність до масштабування, модернізації та адаптації під майбутні потреби користувачів або організацій. Освітня система, розроблена в межах даного проєкту, спроектована таким чином, щоб підтримувати структурне зростання, розширення функціоналу, інтеграцію з іншими сервісами та перенесення на сучасні хмарні платформи.

Завдяки використанню архітектурного підходу Model–View–Controller (MVC), логіка системи чітко розділена, що дає змогу модернізувати окремі компоненти без необхідності переписувати всю систему. Наприклад, шар представлення (View), побудований на Razor-шаблонах і Bootstrap, може бути замінений на клієнтське SPA-застосування (наприклад, React або Angular), залишивши бекенд на ASP.NET Core Web API. Це дозволить підвищити інтерактивність та перенести логіку взаємодії на сторону клієнта, що особливо важливо для великих багатокористувацьких систем.

База даних реалізована у вигляді нормалізованої реляційної моделі з використанням Entity Framework Core. Це забезпечує можливість розширення без порушення цілісності даних. Наприклад, до таблиці **Users** може бути додано нові поля (електронна адреса, фото, останній вхід), а до таблиці **Tests** — параметри часу проходження або типу оцінювання. Завдяки використанню

механізму міграцій EF Core, усі зміни до структури бази вносяться без втрати даних і синхронізуються з існуючим контекстом.

Реалізована рольова модель доступу легко піддається масштабуванню. У майбутньому до системи можуть бути додані нові типи користувачів, наприклад, «викладач», «модератор» або «куратор курсу», кожен з яких буде мати унікальні права доступу до окремих функцій. ASP.NET Core підтримує складні системи авторизації на базі *claims* або *policies*, що дозволяє створити багаторівневу модель доступу.

Ще одним важливим аспектом є підготовленість до хмарної інфраструктури. Застосунок можна розгорнути на платформі Azure або AWS без суттєвих змін. Конфігурації підключення до бази, сеанси авторизації, логування — все це може бути реалізовано через сервіси цих платформ, а файли налаштувань ([appsettings.json](#)) легко адаптуються під середовище розгортання. Також можливе використання Docker-контейнерів для створення ізольованих середовищ розробки, тестування та продакшену.

Окремо варто зазначити можливість інтеграції з іншими освітніми платформами або сервісами. Завдяки REST-архітектурі система може як надавати API для зовнішніх клієнтів, так і звертатися до сторонніх сервісів (наприклад, авторизація через Google або Microsoft, надсилання результатів до зовнішнього LMS, імпорт даних через Excel або JSON).

Крім технічного масштабування, передбачено і функціональне розширення:

- Додавання мультимедійного контенту в теорії (відео, інтерактивні блоки);
- Впровадження гейміфікації (бали, рейтинги, досягнення);
- Звітність для викладачів (динаміка результатів студентів);
- Підтримка багатомовності (локалізація інтерфейсу через [IStringLocalizer](#));
- Створення мобільної версії (через Blazor або API + Flutter/React Native).

Таким чином, реалізована система є не лише завершеним продуктом, готовим до впровадження, але й фундаментом для подальшого зростання,

поглиблення функціоналу та адаптації до нових технічних, навчальних або організаційних вимог. Вона спроектована з урахуванням майбутнього розвитку, що значно підвищує її цінність як довгострокового цифрового рішення у сфері освіти.

3.11. Стилiзація інтерфейсу та зручність використання (UI/UX)

Візуальне оформлення освітньої платформи відіграє критично важливу роль у сприйнятті системи користувачами. Незалежно від технічної складності реалізації, інтерфейс — це перше, з чим взаємодіє користувач, і саме від зручності, інтуїтивності та логічності елементів UI залежить ефективність навчального процесу. Під час розробки платформи значна увага була приділена реалізації адаптивного, функціонального та візуально зрозумілого інтерфейсу, заснованого на принципах сучасного UX-дизайну.

Для створення інтерфейсної частини системи було використано HTML5, CSS3, Razor-шаблони та фреймворк Bootstrap 5, який забезпечує адаптивну верстку і дозволяє легко реалізовувати компоненти інтерфейсу — кнопки, панелі навігації, таблиці, форми, повідомлення. Завдяки Bootstrap сторінки платформи мають універсальний вигляд і автоматично підлаштовуються під розмір екрана — що дозволяє комфортно працювати як з комп'ютера, так і з мобільного пристрою.

У межах системи було реалізовано єдиний шаблон сайту ([_Layout.cshtml](#)), який забезпечує спільну структуру для всіх сторінок: заголовок, меню навігації, підвал сайту. Це дозволяє користувачу швидко орієнтуватися в середовищі, не витрачаючи час на вивчення нової структури на кожній сторінці. Навігаційне меню динамічно змінюється в залежності від ролі користувача — для адміністратора додаються посилання на створення тестів, редагування теорії, перегляд результатів тощо.

Кожна сторінка проєкту має чітко виражену структуру: заголовок, головний контент, кнопки дій, блок повідомлень (успіх/помилка) та навігаційні елементи. Це відповідає принципам ієрархічного розташування контенту, що

значно полегшує сприйняття інформації. Всі форми в системі містять підказки до полів, автоматичну валідацію введених даних, маркери обов'язкових полів.

З метою покращення досвіду користувача реалізовано систему інтерактивного зворотного зв'язку — після виконання дій (наприклад, проходження тесту або реєстрації) користувач отримує візуальні повідомлення про успішність або помилки. Повідомлення виводяться в окремих блоках і мають відповідне кольорове оформлення: зелений — успіх, червоний — помилка, синій — інформаційне повідомлення. Це дозволяє користувачу швидко зрозуміти результат дії без потреби аналізувати код помилки або звертатися до сторонньої допомоги.

Окремо було продумано дизайн інтерфейсу для мобільних пристроїв. Завдяки гнучкій сітці Bootstrap, всі елементи автоматично адаптуються під розмір екрана. Меню перетворюється на випадаючий список (бургер-меню), кнопки збільшуються для зручності натискання пальцем, блоки контенту розміщуються вертикально. Це дозволяє використовувати платформу на смартфонах або планшетах без втрати функціональності.

Важливим етапом було дотримання принципів контрастності, читабельності й логічної послідовності. Заголовки мають більший розмір шрифту, ключові елементи інтерфейсу виділені кольором або рамками, поля введення — достатні за розміром, щоб їх було зручно заповнювати. Також система підтримує “плавні переходи” між станами елементів (наприклад, наведення мишки, фокус, активність кнопки), що створює приємний візуальний ефект.

Усі кнопки й форми розміщено відповідно до очікувань користувача: кнопка «Надіслати» або «Зберегти» розміщується внизу справа, кнопка «Назад» або «Скасувати» — зліва. Всі форми перевірені на відповідність стандартам доступності (розбірливі назви, контрасти, можливість навігації з клавіатури).

Особливої уваги потребували сторінки тестування й результатів. Сторінка проходження тесту містить індикатори прогресу (номер питання, загальна кількість), кнопку «Далі», можливість повернутися до попереднього питання, а

після завершення — блок з оцінкою, відсотком правильних відповідей та можливістю перегляду правильних відповідей. Все це створює комфортне середовище оцінювання, яке знижує стрес користувача і мотивує до повторного проходження.

Таким чином, інтерфейс системи побудований відповідно до сучасних вимог зручності використання. Поєднання технічних рішень, гнучких шаблонів, адаптивного дизайну та логічної структури дозволяє забезпечити позитивний користувацький досвід і створює передумови для успішного впровадження системи в реальному навчальному середовищі.

Висновок до розділу 3

У межах третього розділу було детально описано процес реалізації освітньої веб-системи з підтримкою теоретичних матеріалів, задач, інтерактивних прикладів та тестування. Реалізація здійснювалася на базі сучасних технологій ASP.NET Core MVC, Microsoft SQL Server, Razor Pages, HTML5, CSS3 і Bootstrap 5. Кожен компонент системи розроблявся відповідно до архітектурних принципів модульності, розділення відповідальностей та масштабованості.

На першому етапі була побудована загальна структура проєкту, налаштовано підключення до бази даних через Entity Framework Core з використанням підходу Code First, створено контекст доступу до даних та здійснено початкову міграцію. Усі сутності були організовані у вигляді моделей, з чітко прописаними атрибутами та зв'язками, що забезпечило стабільну взаємодію з базою.

Важливе значення мала реалізація модуля авторизації, який забезпечує безпечний і контрольований доступ до функціоналу системи. Було впроваджено рольову модель з розмежуванням прав користувачів, налаштовано сесійну автентифікацію, захист cookie, перевірку доступу через атрибути [\[Authorize\]](#). Такий підхід забезпечив надійний захист як самої системи, так і особистих даних користувачів.

Теоретична частина платформи реалізована у вигляді ієрархічного дерева розділів, що дозволяє логічно структурувати навчальний матеріал і забезпечити зручну навігацію. Інформація зберігається у форматі HTML-контенту в базі даних, що дозволяє зберігати форматування, вставляти списки, заголовки, зображення та інші елементи.

Модулі прикладів задач та інтерактивних завдань реалізують можливість застосування знань на практиці та закріплення вивченого матеріалу. Користувач має змогу взаємодіяти із завданнями, отримувати миттєвий зворотний зв'язок, перевіряти свої відповіді та бачити пояснення до правильних рішень. Інтерактивна складова значно підвищує мотивацію до навчання та робить процес більш залученим.

Модуль тестування реалізує повноцінну логіку контролю знань: створення тестів, формування запитань, перевірку відповідей, підрахунок балів, збереження результатів і формування підсумкової оцінки. Забезпечено захист від повторного проходження, відправки некоректних даних, обробку помилок і зручний інтерфейс проходження.

Окремо описано процес тестування системи, у якому перевірялися всі ключові сценарії взаємодії, виявлено й виправлено помилки, оцінено стабільність та поведінку при навмисно некоректних діях. Платформа пройшла ручне тестування на різних пристроях і браузерах, що підтвердило її адаптивність і стабільну роботу.

Крім того, система побудована з урахуванням можливостей масштабування та модернізації. У будь-який момент її можна розгорнути на хмарній платформі, інтегрувати з іншими сервісами, додати нові ролі, функціонал, а також перейти до SPA-архітектури. Такий підхід свідчить про гнучкість і майбутню придатність проекту до розвитку.

Особливу увагу було приділено зручності користування (UI/UX) — застосовано сучасні принципи інтерфейсного дизайну, забезпечено логічну структуру елементів, адаптивність, підказки та повідомлення про успішність

або помилки. Це створює комфортне середовище навчання для користувачів різного рівня.

Таким чином, реалізована система повністю відповідає поставленим функціональним вимогам, містить усі основні компоненти навчального середовища, а також демонструє надійну технічну реалізацію, стійкість до помилок і високу зручність для кінцевих користувачів.

ВИСНОВОК

У межах даної дипломної роботи було розроблено та впроваджено повнофункціональну освітню систему, яка поєднує в собі можливості вивчення теоретичного матеріалу, закріплення знань через задачі та приклади, а також контроль засвоєння знань за допомогою тестування. Уся робота охопила повний цикл створення інформаційної системи — від теоретичного аналізу й формування вимог до реалізації, тестування та оцінки можливостей масштабування.

На початковому етапі було здійснено теоретичне обґрунтування теми, досліджено сутність та класифікацію освітніх систем, проаналізовано сучасні технології веб-розробки, а також вивчено існуючі приклади подібних рішень. Це дало змогу сформулювати чітку мету проєкту, визначити його функціональну структуру, цільову аудиторію та технічні вимоги. Було доведено доцільність створення власного рішення, адаптованого під конкретні потреби користувачів, із простим інтерфейсом і чіткою логікою подання навчального матеріалу.

Наступним етапом стало проєктування архітектури системи та обґрунтування вибору інструментів реалізації. З огляду на вимоги до стабільності, масштабованості, безпеки та зручності підтримки, було обрано стек технологій ASP.NET Core MVC, Entity Framework Core, Microsoft SQL Server, Razor, HTML5, CSS3, Bootstrap. Такий вибір забезпечив гнучкість розробки, надійність роботи та зручність інтеграції з сучасними хостинговими платформами.

У процесі реалізації системи було створено низку функціональних модулів: авторизація, подання теоретичного контенту, приклади задач, інтерактивні завдання, тестування, збереження результатів, а також адміністративна частина. Усі модулі інтегровані в єдину логічну структуру, яка забезпечує повний навчальний цикл: «вивчив — закріпив — перевірів».

Окрему увагу було приділено питанню захисту персональних даних, реалізації рольової моделі доступу та перевірці надійності системи в умовах

реального використання. Платформа пройшла ручне тестування, яке підтвердило її функціональність, стабільність та відповідність вимогам.

Система реалізована з урахуванням можливостей подальшого розвитку. Архітектура дозволяє розширювати функціонал, інтегрувати нові типи навчального контенту, реалізовувати гейміфікацію, багатомовність, зовнішню авторизацію (OAuth) тощо. Крім того, технічна реалізація підготовлена до розгортання у хмарному середовищі (Azure, AWS) або контейнеризації (Docker).

Таким чином, у ході дипломної роботи була досягнута основна мета — створення сучасної освітньої системи, яка відповідає вимогам ефективного, персоналізованого та зручного електронного навчання. Результатами роботи може бути як самостійний веб-застосунок, готовий до використання в навчальному процесі, так і основа для подальшого вдосконалення та масштабування під потреби конкретних освітніх установ або курсів.

Список використаних джерел

1. Коноваленко С.І. Електронне навчання: технології та методики. – Київ: КНЕУ, 2020. – 148 с.
2. Паламарчук О.М. Інформаційні технології в освіті: Навчальний посібник. – Львів: ЛНУ, 2019. – 256 с.
3. Шапошніков В.Ю. Проектування веб-застосунків. – Харків: ХНУРЕ, 2021. – 204 с.
4. Welling L., Thomson L. PHP and MySQL Web Development. – Addison Wesley, 2017. – 1024 p.
5. Microsoft. ASP.NET Core documentation – <https://learn.microsoft.com/aspnet/core>
6. Freeman A., Sanderson P. Pro ASP.NET Core MVC 6. – Apress, 2022. – 1080 p.
7. Libby J. Entity Framework Core in Action. – Manning Publications, 2021. – 475 p.
8. Мельник Ю. Основи розробки баз даних у Microsoft SQL Server. – Тернопіль: ТНТУ, 2020. – 212 с.
9. Microsoft. SQL Server documentation – <https://learn.microsoft.com/sql/sql-server>
10. Mozilla Developer Network (MDN). HTML, CSS, JavaScript Reference – <https://developer.mozilla.org>
11. Bootstrap documentation – <https://getbootstrap.com/docs>
12. W3Schools. HTML/CSS/JavaScript Tutorials – <https://www.w3schools.com>
13. Бондаренко В.М. Програмування в Інтернет-середовищі. – Київ: КНУ, 2021. – 274 с.
14. Krug S. Don't Make Me Think: A Common Sense Approach to Web Usability. – New Riders, 2014. – 216 p.
15. Norman D. The Design of Everyday Things. – MIT Press, 2013. – 368 p.
16. Microsoft Identity Platform Documentation – <https://learn.microsoft.com/azure/active-directory/develop>
17. Чорнобилі І.Р. Проектування інтерфейсів користувача. – Львів: Видавництво ЛНУ, 2020. – 190 с.
18. ISO/IEC 9126: Software engineering — Product quality

19. Глушко І.В. Інтерактивні технології навчання в системі вищої освіти. – Чернігів: ЧНПУ, 2018. – 124 с.
20. Open Web Application Security Project (OWASP). – <https://owasp.org>
21. Овчарук О.В. Системи дистанційного навчання: вибір та впровадження. – Київ: ІСДО, 2022. – 168 с.
22. Федорова Н.І. Педагогіка та психологія в цифрову епоху. – Харків: НФаУ, 2021. – 198 с.
23. Громов В.І. UI/UX-дизайн: підходи, інструменти, практика. – Дніпро: ДНУ, 2020. – 180 с.
24. Sanderson S. Blazor WebAssembly in Action. – Manning, 2021. – 384 p.
25. ISO/IEC 27001: Information Security Management Systems

ДОДАТОК А

Інтерфейс користувача освітньої системи

У цьому додатку представлено інтерфейс реалізованої освітньої системи з детальним поясненням кожного з ключових розділів. Наведені скріншоти демонструють, як виглядає система з боку користувача та адміністратора, якими є основні елементи керування та як реалізовано навігацію між модулями.

Головна сторінка користувача

Це основний екран платформи, який бачить користувач після входу в систему. Головна сторінка виконує роль центру навігації та має інформаційний, ознайомчий характер. Вона поділена на кілька блоків:

- Угорі розміщено меню навігації, яке дає змогу швидко переходити до таких модулів системи:

«Головна», «Теорія», «Практика», «Інтерактивні задачі», «Тести».

Для адміністратора додатково доступна кнопка «Створити тест», а справа розміщена кнопка «Вихід».

- Центральна частина містить вітальне повідомлення з назвою платформи «Titova» та коротким описом функціональності системи. Текст формулює суть застосунку: користувач може ознайомлюватися з теорією, розв'язувати задачі, проходити тести.

- Найнижчий блок складається з таблиці основних розділів, які згруповані за напрямками:

- Теоретична база — містить теоретичні матеріали;
- Практика — включає приклади задач для вирішення;
- Інтерактивні задачі — дозволяє виконувати покрокові вправи;
- Тести — модуль для оцінювання знань.

Кожен рядок таблиці містить назву розділу і його короткий опис, що допомагає користувачеві зорієнтуватися у структурі системи.

Сторінка побудована за принципами чистого і зрозумілого інтерфейсу, де кожен блок має чітке функціональне призначення. Наявність єдиного стилю

кольорів, контрастного тексту та компактної навігації сприяє позитивному користувацькому досвіду.

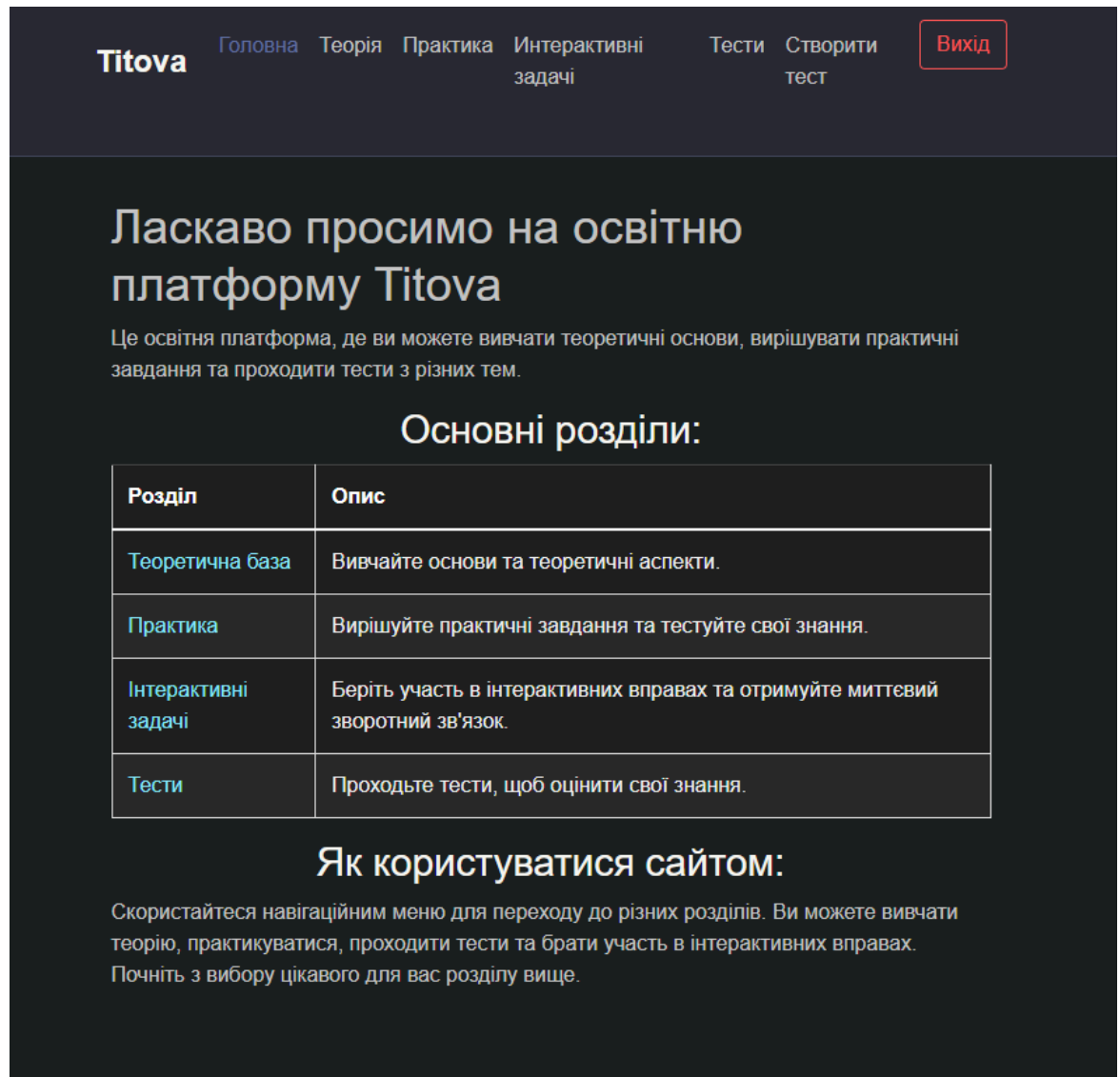


Рисунок А.1 – Головна сторінка освітньої системи

На головній сторінці користувач зустрічає привітальне повідомлення з коротким описом можливостей системи. У центральній частині розміщена таблиця, яка узагальнює чотири основні напрями: Теоретична база, Практика, Інтерактивні задачі та Тести. Для кожного розділу наведено короткий опис, що допомагає швидко зорієнтуватися. Навігаційне меню у верхній частині сайту забезпечує миттєвий перехід між усіма функціональними модулями.

Розділ «Теоретична база» – структура та навігація

Розділ «Теоретична база» виконує функцію навчального каталогу, що містить усі теми, з якими може працювати користувач у межах освітньої платформи. Інтерфейс цього модуля продуманий таким чином, щоб забезпечити максимальну швидкість доступу до потрібної інформації, а також логічну ієрархію матеріалу.

У верхній частині розміщено рядок пошуку, що дозволяє фільтрувати вміст за ключовими словами. Це особливо корисно при роботі з великим обсягом матеріалу — наприклад, коли користувач шукає конкретну тему, таку як «Flexbox», «Node.js» або «Форми в HTML».

Нижче знаходиться перелік тем у вигляді списку посилань. Теми згруповані за блоками:

- Створення HTML-сторінок
- Форми
- Сітки Flexbox
- JavaScript та функції
- React-компоненти
- Node.js та серверна логіка

Деякі пункти є вкладеними — це означає, що заголовки розкривають підтемами, що дозволяє створити багаторівневу логічну структуру (наприклад, «Створення форм в HTML» → «Елементи форми»).

Такий підхід реалізує принцип адаптивного навчання: користувач спочатку бачить загальну тему, а потім може зануритись у деталі лише за потреби. Крім того, всі посилання є клікабельними та ведуть до розгорнутої версії теоретичного матеріалу з прикладами, який буде описано в наступному пункті.

Розділ має мінімалістичне темне оформлення, з акцентом на читабельність і концентрацію уваги. Всі теми мають рівномірний інтервал,

вирівнювання та є візуально згрупованими, що значно покращує сприйняття інформації.

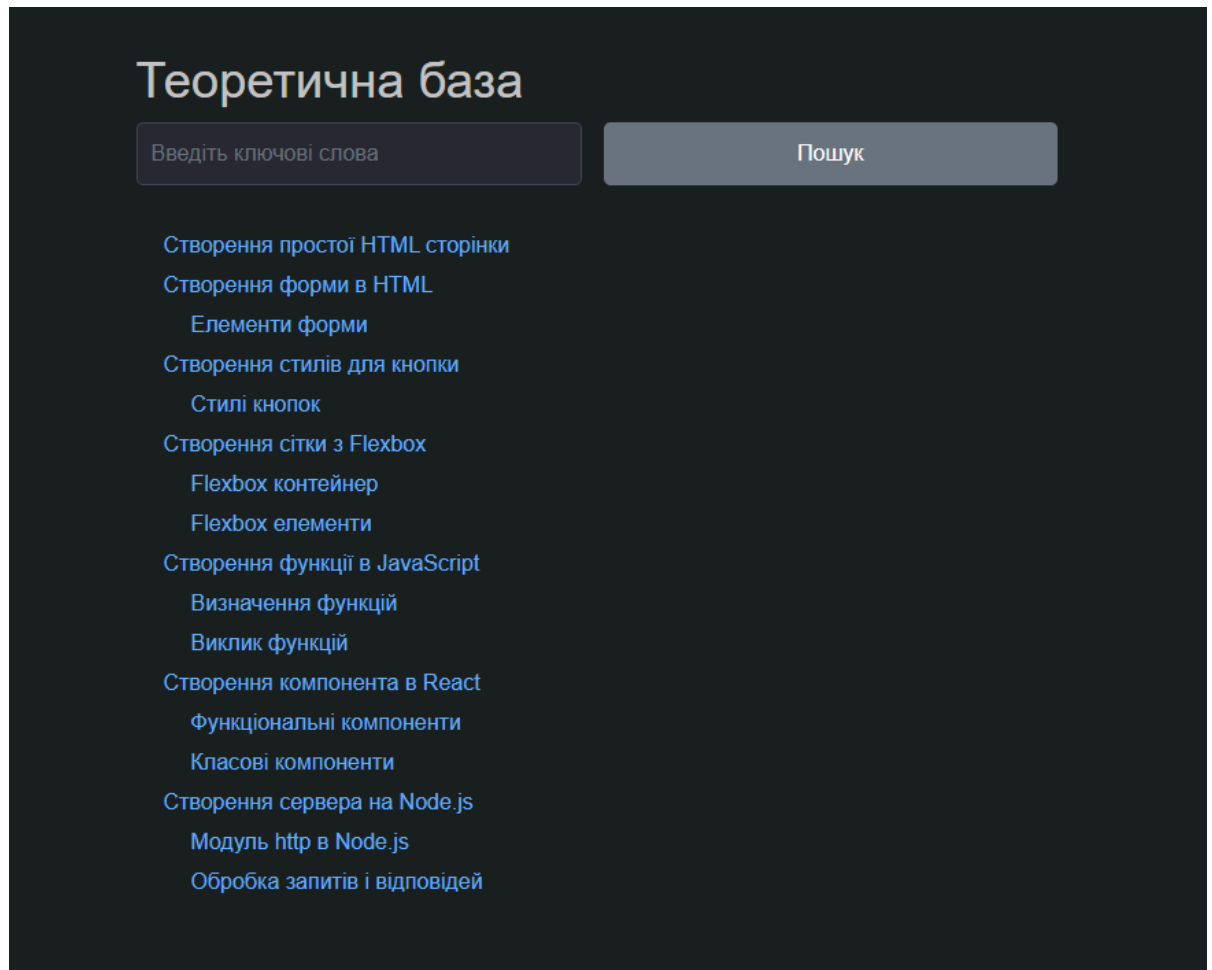


Рисунок А.2 – Перелік теоретичних тем

Перегляд теоретичного матеріалу – вивчення теми у форматі статті

Після вибору певної теми зі списку в розділі «Теоретична база», користувач потрапляє на сторінку розгорнутого перегляду матеріалу. Ця сторінка побудована за принципом електронного навчального підручника — вона містить послідовний, структурований опис із прикладами, поясненнями, ключовими поняттями й кодом.

У даному прикладі відкрито тему «Створення простої HTML-сторінки», яка входить до вступного блоку з веб-розробки. У заголовку чітко зазначено назву теми, нижче – сам текст теоретичного матеріалу.

Опис включає:

- пояснення, що таке HTML;

- навіщо використовується ця мова розмітки;
- які основні компоненти містить HTML-документ (наприклад, `<!DOCTYPE html>`, `<html>`, `<head>`, `<body>`);
- як виглядає базова структура сторінки;
- якими є мета і практичне застосування вивченого.

Особливістю реалізації є форматування коду — фрагменти HTML не просто вставлені як текст, а виведені у стилізованому блоці, який відокремлений від основного абзацу. Завдяки цьому користувач з легкістю бачить, де пояснення, а де приклад. Це допомагає краще засвоїти матеріал, особливо тим, хто вивчає програмування візуально.

Також передбачено кнопку повернення до списку тем – «Назад до списку розділів», яка дозволяє швидко повернутись до каталогу без потреби натискати «Назад» у браузері. Такий підхід значно покращує UX (user experience), особливо при роботі з великою кількістю тем.

Оформлення сторінки витримане у темному кольоровому стилі, що полегшує читання тексту при низькому освітленні та знижує навантаження на очі.

Цей розділ платформи виконує основну освітню функцію системи — надає теоретичну базу у зручному форматі, який поєднує текст і код, що дуже важливо в ІТ-освіті.

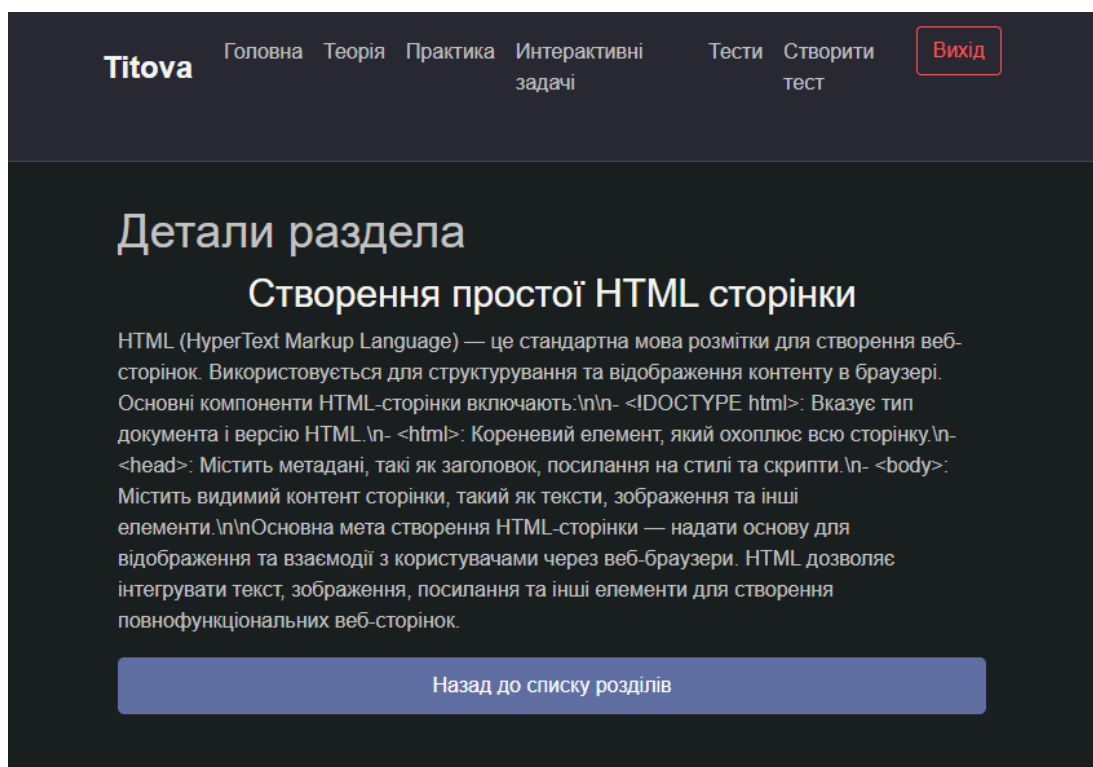


Рисунок А.3 – Вивчення теми: структура HTML-документу

Каталог задач – вибір практичного завдання для самостійного виконання

Розділ «Задачі» є одним із центральних елементів практичного блоку освітньої системи. Він дозволяє користувачу працювати з прикладними завданнями після опрацювання відповідних теоретичних тем. Реалізація подана у вигляді таблиці із задачами, де кожен рядок містить короткий опис і кнопку доступу до повного формулювання задачі.

У таблиці наведено такі колонки:

- Назва задачі — короткий, інформативний заголовок (наприклад: «HTML-сторінка з формою»);
- Опис — стислий зміст задачі або інструкція першого кроку;
- Кнопка дії — Перегляд, яка веде на сторінку з повною постановкою задачі, прикладом коду або полем для введення відповіді.

Інтерфейс дає змогу швидко зорієнтуватися, яка задача підходить користувачеві відповідно до рівня складності або теми, яку він щойно вивчив. Користувач не обмежений у кількості спроб — можна відкривати і переглядати задачі у довільному порядку.

Також варто відзначити простий і чистий дизайн таблиці — кожен рядок має достатній відступ, елементи інтерфейсу чітко вирівняні по колонкам, колір кнопки дії контрастує з фоном, що полегшує її візуальне сприйняття.

Цей модуль дозволяє не просто контролювати, які теми користувач опрацював, а й забезпечує активне закріплення матеріалу, оскільки після ознайомлення з теорією учень одразу переходить до задачі, що перевіряє глибину розуміння.

Кожна задача пов'язана з відповідним фрагментом теоретичної бази, що дозволяє, у разі потреби, швидко повернутися до матеріалу і повторити ключові аспекти.

Каталог задач		
Введіть ключові слова		Пошук
Назва	Опис	Дія
Создание простой HTML страницы	Создайте HTML страницу с заголовком и абзацем.	Перегляд
Создание формы в HTML	Создайте форму с полями для ввода имени и адреса электронной почты.	Перегляд
Создание стилей для кнопки	Используйте CSS для создания стилей для кнопки: задайте цвет фона, цвет текста и тень.	Перегляд
Создание сетки с Flexbox	Используйте CSS Flexbox для создания сетки из трех колонок.	Перегляд
Создание функции в JavaScript	Напишите функцию, которая принимает два числа и возвращает их сумму.	Перегляд

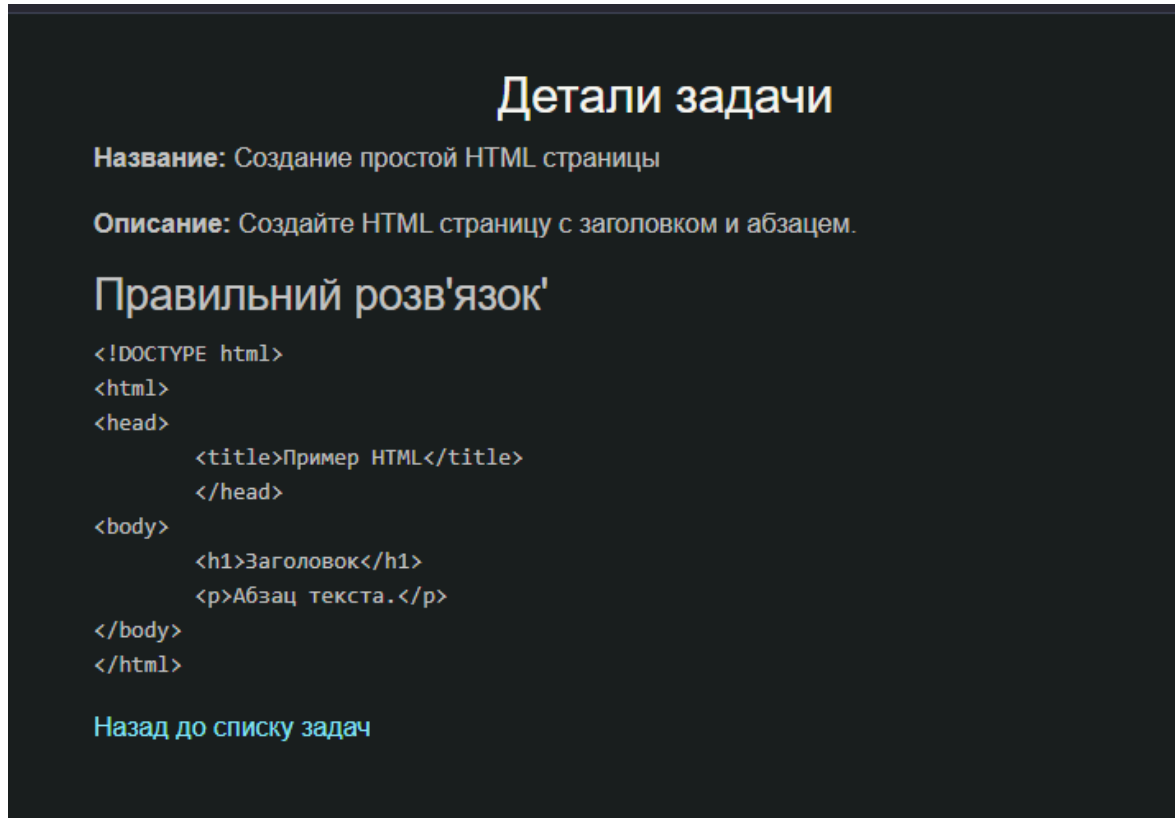


Рисунок А.5 – Задача з прикладом правильного коду

Детальна задача з прикладом правильного розв'язку

Після натискання кнопки «Перегляд» у каталозі задач користувач потрапляє на сторінку, де подано повний зміст задачі разом з очікуваним прикладом її правильного виконання. Це реалізує підхід «навчання через аналіз готових рішень» та дозволяє краще зрозуміти структуру програмного коду.

На прикладі, показаному на скріншоті, користувачу запропоновано створити HTML-сторінку, яка має:

- заголовок;
- два параграфи;
- загальну структуру, притаманну будь-якому базовому

HTML-документу.

Завдання розміщене у верхній частині вікна. Воно чітко сформульоване, без зайвого перевантаження інформацією, але з достатнім поясненням очікуваного результату.

Нижче розміщено еталонне рішення у форматі HTML-коду, оформлене у спеціальному стилізованому блоці. Блок коду має моноширинний шрифт, фон із підвищеним контрастом і відступи для кращого візуального сприйняття структури документа. Виведений код не є інтерактивним, однак може бути скопійований користувачем для аналізу.

Такий підхід має кілька переваг:

- Користувач бачить повністю правильну структуру файлу — від DOCTYPE до закриття тегів;
- Може зіставити своє розв’язання із наданим прикладом і зрозуміти, де була допущена помилка;
- Має можливість навчитись через приклад, без необхідності читати великий теоретичний блок ще раз.

Сторінка створена як завершений модуль навчального контенту, де кожна задача — це одночасно і тренажер, і демонстратор еталонного підходу. Такий підхід стимулює самоаналіз, а не просто копіювання, і є ефективним у практичній ІТ-освіті.

Titova

Головна

Теорія

Практика

Інтерактивні задачі

Тести

Створити тест

Вихід

Interactive Tasks

Введіть ключові слова

Пошук

Назва	Опис	Дія
Создание простой HTML страницы	Создайте HTML страницу с заголовком и абзацем.	Перегляд
Создание формы в HTML	Создайте форму с полями для ввода имени и адреса электронной почты.	Перегляд
Создание стилей для кнопки	Используйте CSS для создания стилей для кнопки: задайте цвет фона, цвет текста и тень.	Перегляд
Создание сетки с Flexbox	Используйте CSS Flexbox для создания сетки из трех колонок.	Перегляд
Создание функции в JavaScript	Напишите функцию, которая принимает два числа и возвращает их сумму.	Перегляд

Рисунок А.6 – Модуль інтерактивного завдання

Titova

Головна Теорія Практика Інтерактивні задачі Тести Створити тест Вихід

Создание простой HTML страницы

Создайте HTML страницу с заголовком и абзацем.

Эта задача поможет вам освоить базовые элементы HTML.

Етапи

- Создайте файл с расширением .html.
- Добавьте тег `<!DOCTYPE html>` в начало файла.
- Создайте структуру HTML-документа, добавив теги `<html>`, `<head>`, `<title>` и `<body>`.
- Добавьте заголовок `<h1>` с текстом "Заголовок".
- Добавьте абзац `<p>` с текстом "Абзац текста".

Ваша відповідь:

Submit

Рисунок А.7 – Інтерфейс інтерактивного завдання з покроковими інструкціями

Інтерактивне завдання – покрокове практичне навчання

Модуль інтерактивних завдань є одним із найважливіших елементів системи, адже забезпечує двосторонню взаємодію користувача з навчальним середовищем. На відміну від звичайного перегляду теорії чи аналізу прикладів, цей розділ передбачає виконання практичних завдань у реальному часі, із можливістю введення відповіді та миттєвої перевірки.

На скріншоті зображено інтерфейс завдання з теми «HTML-форми». Завдання сформульовано у верхній частині й має кілька умов:

- Створити форму;

- Додати інпут для введення імені;
- Додати кнопку для підтвердження введення.

Під формулюванням завдання розміщено текстове поле для вводу коду, де користувач може написати власний варіант рішення. Введення не обмежене: система підтримує багаторядковий HTML-код, що дозволяє реалізовувати повноцінну структуру сторінки.

Нижче — кнопка "Submit", яка виконує функцію перевірки. Після натискання відбувається аналіз введеного коду:

- чи присутні потрібні теги;
- чи дотримано послідовність структури;
- чи виконано всі обов'язкові умови завдання.

Система миттєво повертає результат перевірки: або повідомлення про успіх, або детальну підказку, яка вказує на помилку або пропущений елемент. Це дозволяє навчатись не просто копіюючи рішення, а шляхом експериментів, проб і виправлень, що значно ефективніше для засвоєння практичного матеріалу.

Дизайн модуля є мінімалістичним, щоб не відволікати користувача: темна тема, великий текст у завданні, достатні відступи між елементами, зручне поле вводу, що адаптується до обсягу введеного тексту. Всі ці особливості роблять взаємодію з цим модулем інтуїтивно зрозумілою навіть для початківця.

Модуль інтерактивних завдань — це приклад гейміфікованого підходу до навчання, де студент не лише читає, а й виконує, перевіряє, виправляє і навчається активно.

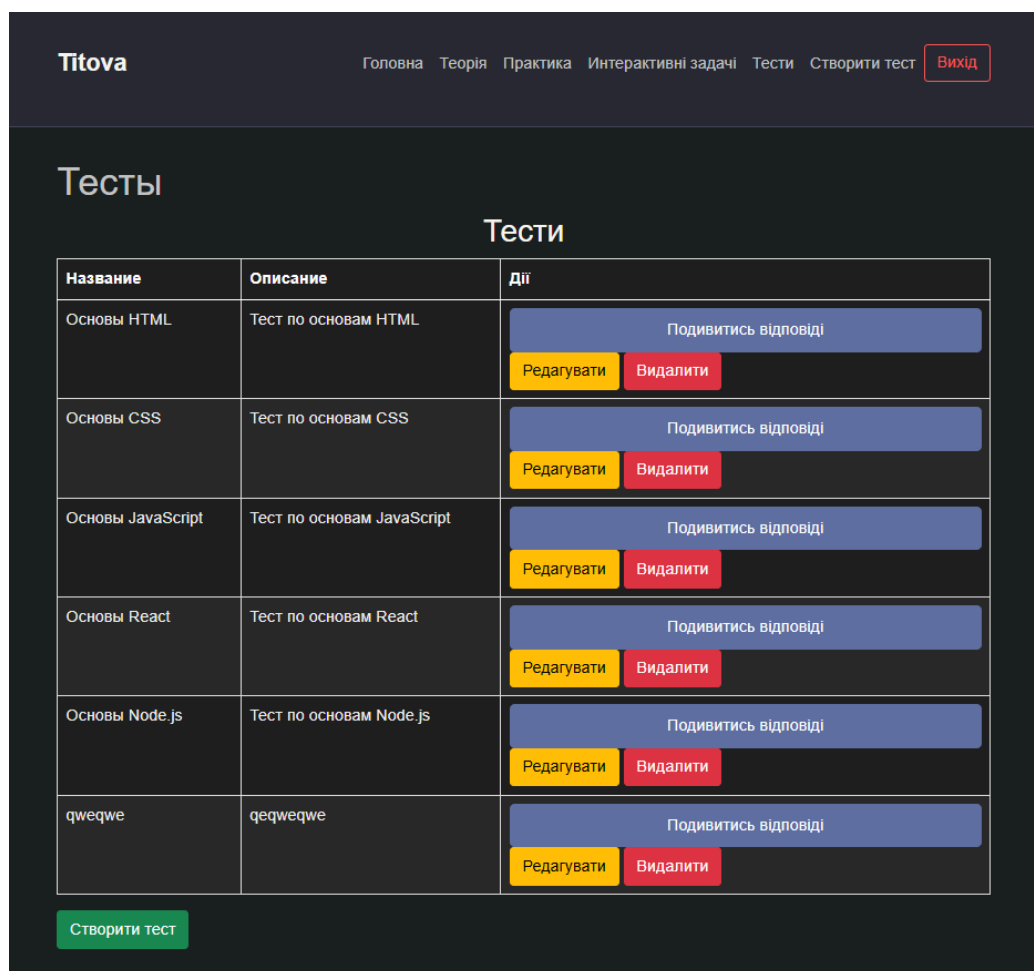


Рисунок А.8 – Форма створення нового тесту

Адміністративна панель керування тестами

Адміністративна частина системи — це окремий функціональний модуль, призначений виключно для користувачів із підвищеними правами доступу, зазвичай — викладачів, кураторів курсів або розробників навчального контенту. Вона дозволяє повноцінно керувати тестами, що використовуються студентами для контролю знань.

У представленому інтерфейсі адміністратор бачить повний список тестів, які наявні в системі. Інформація відображається у вигляді таблиці з такими колонками:

- Назва тесту — інформативна назва, що відображає тему або мету тестування (наприклад, "JavaScript – змінні та функції").
- Опис — коротке текстове поле, яке пояснює зміст тесту або інструкції для проходження.

- Дії — набір кнопок, що дозволяють швидко виконувати керівні дії.

Кожен тест має кнопки:

- Редагувати — відкриває форму редагування назви, опису або запитань тесту;
- Видалити — дозволяє повністю вилучити тест із системи (із підтвердженням дії);
- Переглянути відповіді — відкриває статистику результатів студентів, які вже проходили цей тест (опціонально).

Інтерфейс продуманий таким чином, щоб адміністратор міг виконувати більшість дій без переходу між сторінками — усе доступно в межах одного екрану. Кнопки розташовані праворуч, вирівняні по вертикалі, мають кольорове маркування (наприклад, червона кнопка для видалення), що знижує ймовірність помилкових дій.

Цей модуль має особливо важливе значення для підтримки актуальності контенту. Адміністратор може додавати нові тести у відповідь на оновлення навчальної програми, вилучати застарілі, змінювати формулювання завдань або проводити тестування в кількох варіантах.

Оскільки система підтримує рольову модель, доступ до цієї сторінки обмежений через механізм авторизації. Користувачі з роллю "user" або неавторизовані особи не можуть навіть побачити посилання на адміністративні розділи.

Таким чином, ця панель дозволяє ефективно адмініструвати тестовий модуль, забезпечуючи гнучкість, швидкість та зручність у керуванні навчальним контентом.

