Міністерство освіти і науки України
Прикарпатський національний університет імені Василя Стефаника
кафедра комп'ютерних наук та інформаційних систем

БАКАЛАВРСЬКА РОБОТА

на тему: «Інтерактивна система для вивчення .NET»

Виконав: студент IV курсу гр. ІСТ-41
ОР бакалавр
спеціальності 126 «Інформаційні системи та технології»
Юрійчук Василь Володимирович

Керівник: к.т.н., старший викладач,
Ізмайлов Артем Вікторович

Рецензент: доцент, к.т.н.
Превисокова Наталя Володимирівна

м. Івано-Франківськ – 2024

АНОТАЦІЯ

Ця робота присвячена розробці та аналізу інтерактивної системи для вивчення .NET. У роботі розглянуто технічні аспекти створення такої системи, методи та прийоми, які забезпечують максимальну ефективність процесу навчання. Дослідження містить 28 рисунків, 1 таблицю, 20 посилань та 1 додаток.

Вона включає розділи технічного дизайну, програмування та тестування. У рамках дослідження було розроблено веб-сервіс, який допомагає освоєнню .NET, використовуючи інтерактивні методи для покращення навчання. Описано всі етапи розробки, що забезпечують цілісний підхід до створення ефективної системи навчання.

Ключові слова: .NET, веб-сервіс, інтерактивні методи навчання, програмування, технічний дизайн, тестування.

ABSTRACT

This work is dedicated to the development and analysis of an interactive system for learning .NET. The work examines the technical aspects of creating such a system, along with methods and techniques that ensure maximum efficiency in the learning process. The research includes 28 figures, 1 table, 20 references, and 1 appendix.

It covers sections on technical design, programming, and testing. As part of the research, a web service was developed to facilitate learning .NET through interactive methods aimed at improving the educational experience. All development stages are described, providing a comprehensive approach to creating an effective learning system.

Keywords: .NET, web service, interactive learning methods, programming, technical design, testing.

ЗМІСТ

ВСТУП

У сучасному світі, де інформаційні технології стрімко розвиваються, зростає потреба в ефективних інструментах для навчання та підвищення кваліфікації спеціалістів у галузі програмування. Одним з таких інструментів є інтерактивні системи для вивчення різноманітних технологій та мов програмування, зокрема .NET. Аналіз теми інтерактивних систем для вивчення .NET виявляє актуальність розробки та використання таких систем для покращення навчального процесу та практичної підготовки програмістів.

Метою дослідження є розробка та оцінка ефективності інтерактивної системи для вивчення .NET, яка дозволить користувачам засвоювати теоретичні знання та набувати практичних навичок у цій технології. У ході дослідження планується виявити, як саме інтерактивні елементи впливають на процес навчання, які методи взаємодії з користувачем є найбільш ефективними, та як можна оптимізувати навчальний контент для досягнення максимальних результатів.

Об’єктом дослідження є вивчення програмування за допомогою інтерактивних систем, зокрема, системи, орієнтовані на вивчення .NET технології. Це включає в себе програмні засоби, що забезпечують взаємодію користувача з навчальним контентом, а також методи та підходи до розробки таких систем.

Предметом дослідження є інтерактивна система для вивчення .NET. Це включає вивчення закономірностей впливу інтерактивних елементів на ефективність засвоєння матеріалу, взаємозв'язку між типами завдань і рівнем знань користувачів, а також аналіз методів оцінки прогресу навчання та коригування навчального процесу в режимі реального часу.

Методи дослідження включають огляд наукової літератури, тестування системи та детальний аналіз застосування інтерактивної системи для виявлення успішного вивчення .NET.

Завдання дослідження полягають у розробці та оцінці інтерактивної системи для вивчення .NET, визначенні впливу інтерактивних елементів на ефективність навчання, оптимізації навчального контенту, дослідженні взаємозв'язку між типами завдань і рівнем знань користувачів, а також розробці методів для ефективного прогресу в навчанні.

У першому розділі здійснюються аналіз інтерактивних систем, що використовуються для вивчення програмування. Визначаються їхні переваги та недоліки, також визначаються пропозиції щодо можливих вдосконалень і розвитку подальших систем для ефективного вивчення програмування.

У другому розділі проводиться проектування інтерактивної системи для вивчення .NET, її архітектури та виборі необхідних технологій та інструментів для реалізації концепції.

У третьому розділі описується реалізація поставленої задачі, її кроки та засоби розробки інтерактивної системи для вивчення .NET, створення архітектури, модулів та тестування системи.

Таким чином, дослідження спрямоване на вирішення ключових питань, пов'язаних з розробкою, впровадженням та використанням інтерактивної системи для вивчення .NET.

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

1.1. Існуючі інтерактивні системи для вивчення програмування

Вивчення програмування є складним та важким процесом для багатьох студентів. Однак, існують різноманітні інтерактивні системи, які спрощують цей процес та допомагають студентам краще зрозуміти концепції програмування. Огляд існуючих інтерактивних систем для вивчення програмування дозволяє зрозуміти, які підходи та методи вже існують, а також ідентифікувати їх переваги та недоліки [1].

1. Scratch (Рисунок 1.1)

Scratch є однією з найпопулярніших інтерактивних систем для вивчення програмування, особливо серед дітей та підлітків. Ця платформа розроблена у Массачусетському технологічному інституті та призначена для вивчення основ програмування через створення інтерактивних ігор, анімацій та інших мультимедійних проектів [2].

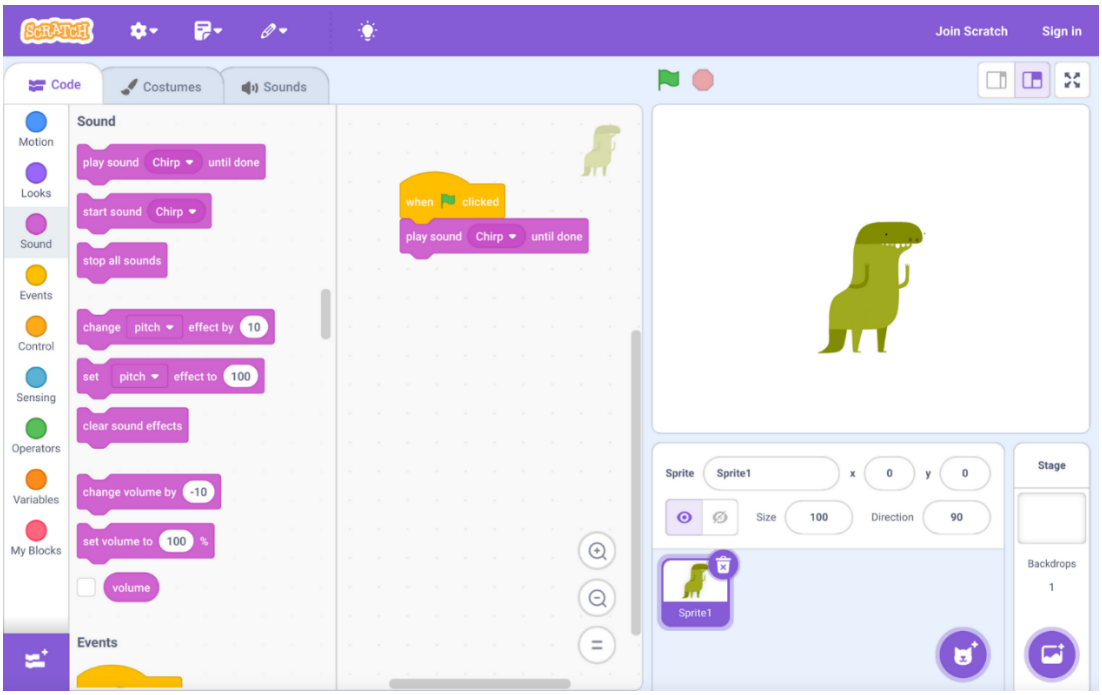


Рисунок 1.1 – Scratch [2]

Основні характеристики Scratch:

1. Scratch використовує блоки коду, які можна перетягувати та з'єднувати між собою, щоб створювати складні програми. Це робить процес програмування більш доступним та зрозумілим для початківців.

2. Платформа надає інтерактивне середовище, в якому користувачі можуть негайно випробовувати свій код, переглядати результати в реальному часі та вносити зміни за необхідності.

3. Scratch має велику та активну спільноту користувачів з усього світу. Це дозволяє студентам ділитися своїми проектами, отримувати поради та підтримку від інших учасників та навчатися на прикладі реальних проектів.

4. Платформа надає різноманітні навчальні матеріали, включаючи відеоуроки, навчальні посібники та інструкції, які допомагають студентам зрозуміти основи програмування та розширювати свої навички.

5. Scratch сприяє розвитку креативності та уяви студентів, дозволяючи їм створювати унікальні та цікаві проекти, які відображають їхні ідеї та інтереси.

Scratch є потужним інструментом для вивчення програмування, який допомагає студентам розвивати не лише технічні навички, але й креативність, логічне мислення та спритність у розв'язанні завдань.

2. Codecademy (Рисунок 1.2)

Codecademy – це онлайн-платформа для навчання програмуванню, яка пропонує інтерактивні курси з різних мов програмування та технологій. Ця платформа стала популярною серед початківців та досвідчених розробників, оскільки вона пропонує ефективний та зручний спосіб вивчення програмування [3].

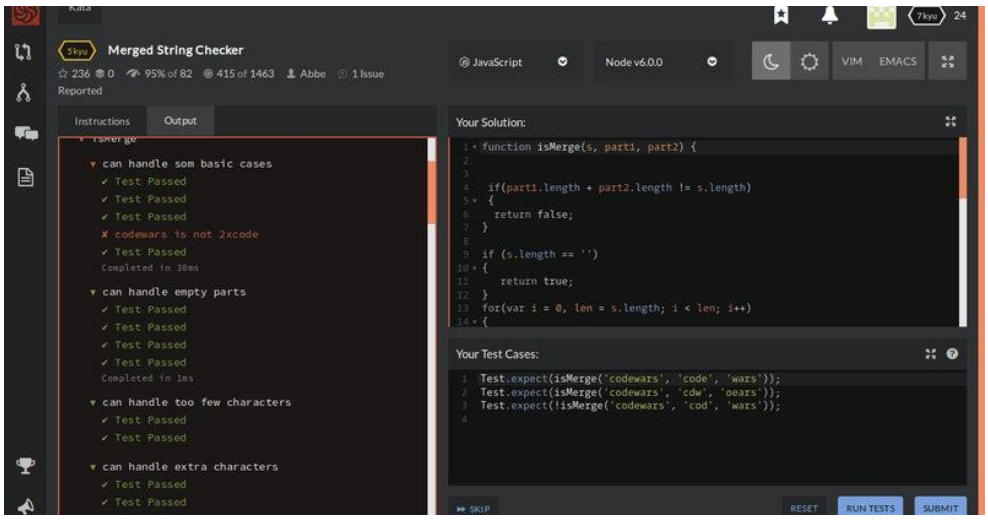


Рисунок 1.2 – Codecademy [3]

Основні характеристики Codecademy:

1. Codecademy пропонує широкий вибір інтерактивних курсів з різних мов програмування, включаючи JavaScript, Python, HTML/CSS, Ruby, Java та інші. Кожен курс включає в себе навчальні матеріали, практичні завдання та можливість відразу ж тестувати свій код прямо в браузері.

2. Платформа пропонує як безкоштовні, так і платні курси. Безкоштовні курси надають базові знання з програмування, тоді як платні курси можуть бути більш розширеними та включати додаткові функції та матеріали.

3. Крім теоретичних матеріалів, Codecademy надає різноманітні проектні завдання, які допомагають студентам застосовувати свої знання на практиці та розвивати реальні проекти.

4. Платформа має активну спільноту користувачів, де студенти можуть ділитися своїми досягненнями, задавати питання та надавати поради один одному.

5. Codecademy пропонує курси для різних рівнів вивчення, починаючи від азів програмування для новачків і закінчуючи складними темами для досвідчених розробників.

Codecademy є дієвим інструментом для вивчення програмування, який дозволяє студентам здобувати практичні навички та знання в області програмування через інтерактивне навчання [4].

3. Khan Academy (Рисунок 1.3)

Khan Academy – це онлайн-навчальна платформа, яка пропонує широкий спектр курсів у різних областях знань, включаючи програмування. Серед них особливо виділяються курси, що присвячені навчанню основ програмування, такі як JavaScript, Python, SQL та інші мови програмування [5].

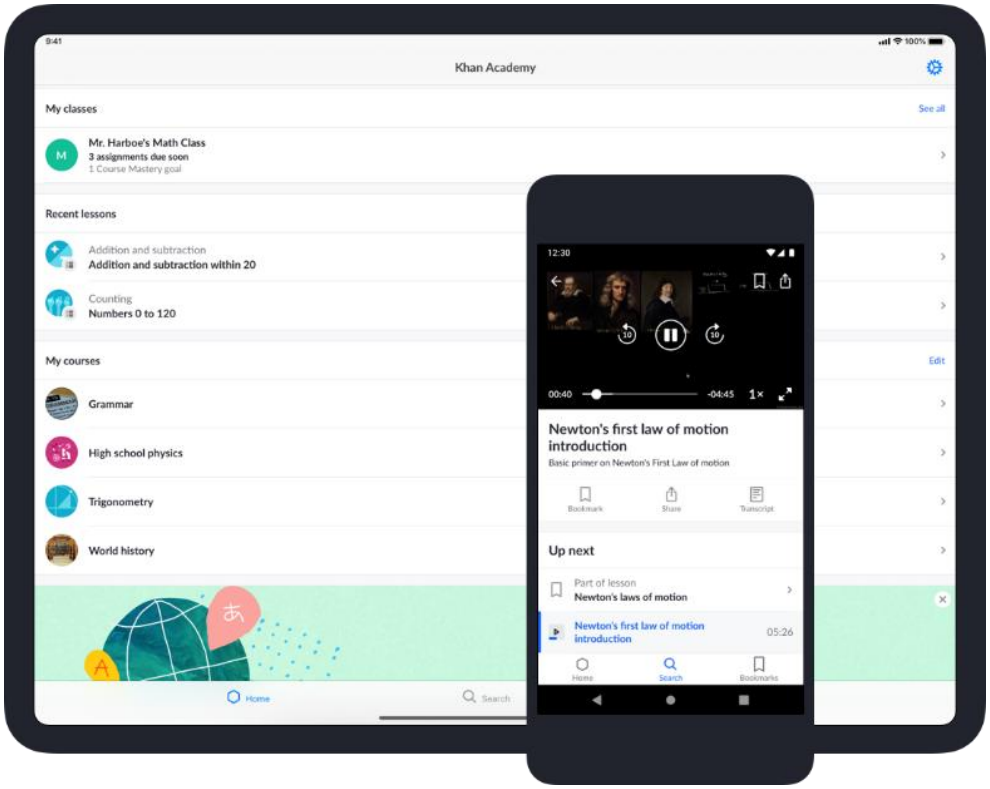


Рисунок 1.3 – Khan Academy [5]

Основні характеристики Khan Academy:

1. Khan Academy пропонує багато різних курсів, які охоплюють різні аспекти програмування. Це дозволяє студентам вибирати курси в залежності від їхніх інтересів та потреб.

2. Платформа надає як теоретичні матеріали, так і практичні завдання, спрямовані на закріплення отриманих знань. Студенти можуть вивчати основи програмування, включаючи концепції, синтаксис та практичне застосування мов програмування.

3. Курси на Khan Academy відрізняються власним високим рівнем інтерактивності. Студенти можуть вчитися за допомогою відеоуроків, інтерактивних вправ, завдань та проектів.

4. Більшість курсів на Khan Academy доступні безкоштовно для всіх користувачів. Це робить навчання доступним для широкого кола людей, незалежно від їхнього фінансового стану.

5. Khan Academy пропонує можливість самостійного навчання, що дозволяє студентам вчитися власним темпом і в зручний для них час.

Khan Academy – це ефективний ресурс для вивчення програмування, який надає студентам можливість здобувати практичні навички та знання у цій сфері через інтерактивне навчання.

Ці інтерактивні системи для вивчення програмування надають студентам можливість вивчати програмування у зручний та цікавий спосіб, що сприяє підвищенню їхнього інтересу та мотивації. Однак, кожна з цих систем має свої переваги та обмеження, які варто враховувати при виборі найбільш підходящого варіанту для навчання.

1.2. Аналіз переваг та недоліків існуючих систем

Проведемо аналіз порівнянь та недоліків програмних продуктів.

Платформа Khan Academy.

Переваги:

- Khan Academy пропонує багато різноманітних курсів, включаючи ті, що присвячені основам програмування, такі як JavaScript, Python, SQL та багато інших.
- Платформа надає інформативні та якісні матеріали, як теоретичні, так і практичні, для засвоєння знань з програмування та інших предметів.
- Більшість курсів Khan Academy доступні безкоштовно, що робить їх доступними для широкого кола користувачів.

Недоліки:

– Платформа може мати обмежену гнучкість у порівнянні з іншими програмами, що пропонують інтерактивні вправи та персоналізоване навчання.
– У порівнянні з іншими програмами, Khan Academy може бути менш інтерактивним у своєму підході до навчання.

Платформа Codecademy.

Переваги:

– Codecademy пропонує інтерактивні вправи та завдання, які допомагають закріпити знання під час навчання.
– Платформа пропонує широкий спектр курсів з програмування та інших областей, що дає користувачам можливість вибирати теми в залежності від їхніх інтересів.
– Більшість курсів доступні безкоштовно, що робить навчання доступним для всіх.

Недоліки:

– Деякі продукти та функції доступні лише за підпискою.
– Може бути важко забезпечити індивідуальне навчання для кожного користувача через великий обсяг учнів.

Платформа Scratch.

Переваги:

– Scratch використовує блоки для програмування, що полегшує розуміння основ програмування для початківців.
– Платформа сприяє розвитку творчих навичок шляхом створення власних проектів та інтерактивних ігор.

Недоліки:

– У порівнянні з іншими мовами програмування, Scratch може бути обмеженим у функціоналі, особливо для більш складних проектів.
– Платформа може бути не настільки гнучкою для розробки складних або великих проектів, як інші мови програмування.

Основні переваги та недоліки систем наведені в таблиці 1.1

Таблиця 1.1. – Основні переваги та недоліки програм

Інтерактивна система	Переваги	Недоліки
Scratch	Інтуїтивний інтерфейс та легке використання Велика спільнота користувачів та ресурсів Підходить для початківців та дітей	Обмежена можливість для складних проектів Відсутність підтримки деяких мов програмування
Codecademy	Широкий вибір курсів з різних мов програмування та технологій Інтерактивні вправи та завдання Персоналізований підхід до навчання	Безкоштовний доступ обмежений Відсутність підтримки деяких мов програмування
Khan Academy	Широкий вибір курсів з різних предметів, включаючи програмування Якісні теоретичні матеріали та практичні вправи Безкоштовний доступ до більшості курсів	Обмежена гнучкість та інтерактивність Може бути менш ефективним для складних концепцій

Аналіз переваг та недоліків цих програмних продуктів допомагає користувачам зрозуміти їхні можливості та обмеження, щоб вони могли зробити найкращий вибір для своїх потреб.

Інтерактивні навчальні системи стають все більш популярними у сучасному освітньому середовищі, пропонуючи унікальні можливості для ефективного та захопливого навчання. У цьому рефераті ми розглянемо деякі з провідних платформ для вивчення програмування та їх переваги та недоліки.

Перша платформа, яку ми розглянемо, - Scratch. Ця платформа відкриває широкі можливості для вивчення програмування шляхом створення

інтерактивних проектів за допомогою блоків коду. Вона спрощує процес навчання для початківців, але має обмежені можливості для більш складних проектів.

Друга платформа - Codecademy - надає більш глибоке вивчення програмування через інтерактивні уроки та практичні завдання. Її переваги включають доступність широкого спектру мов програмування та практичність у навчанні, але деякі користувачі можуть зіткнутися з обмеженнями у вільному режимі вивчення [6].

Нарешті, Khan Academy відома своїми різноманітними курсами з програмування та інших тем. Вона надає користувачам можливість вивчати теорію та виконувати практичні завдання на різних мовах програмування. Однак недоліком може бути менша спеціалізація на програмуванні порівняно з іншими платформами.

В кінцевому підсумку, кожна з цих платформ має свої переваги та недоліки, і вибір залежить від потреб та цілей кожного користувача.

1.3. Постановка задачі дослідження

Завдання полягає в розробці інтерактивного веб-сервісу для вивчення .NET. Система буде мати ряд основних характеристик, включаючи інтерактивність та зручний інтерфейс користувача з можливістю перегляду матеріалів і домашніх завдань.

Основні завдання:

1. Провести аналіз існуючих інтерактивних систем для вивчення програмування, проаналізувати їхні функціональні можливості та особливості, визначити переваги та недоліки кожної системи.

2. Визначити ключові вимоги до нової системи, розробити архітектуру, структуру системи та взаємодію між її компонентами.

3. Вибрати необхідні технології для реалізації, вивчити різні технології програмування, веб-фреймворки та інструменти розробки. Вибрати ті, які найкраще підходять для виконання поставлених завдань та вимог.

4. Розробити основні модулі системи, реалізувати модулі для навчання, перегляду матеріалів і завдань, а також для взаємодії з користувачем. Забезпечити можливість гнучкої настройки та розширення системи.

5. Провести тестування та валідацію роботи системи, перевірити відповідність реалізації системи встановленим вимогам і критеріям успішності.

Висновки до розділу 1.

Проведено огляд сучасних інтерактивних систем для вивчення програмування, що включає різні платформи, такі як Codecademy, Coursera, edX та інші.

Визначено ключові особливості та функціональні можливості кожної з систем, включаючи інтерфейс, методи навчання, підтримку мов програмування та інтерактивні елементи.

Визначено основні переваги існуючих систем, серед яких інтерактивність, доступність, наявність різноманітних курсів та високий рівень підтримки користувачів.

Виявлено недоліки, такі як обмежений доступ до деяких ресурсів без оплати, недостатня гнучкість у виборі навчального матеріалу та методів, а також можливі проблеми з мотивацією у користувачів.

На основі аналізу переваг та недоліків було сформульовано пропозиції щодо вдосконалення інтерактивних систем, включаючи інтеграцію адаптивних навчальних технологій, розширення бібліотек навчальних матеріалів, покращення системи зворотного зв'язку та мотиваційних механізмів.

2. РОЗРОБКА КОНЦЕПЦІЇ ІНТЕРАКТИВНОЇ СИСТЕМИ ДЛЯ ВИВЧЕННЯ .NET

2.1. Проектування архітектури системи

Односторонній сайт містить логін форму для автентифікації користувача. Після успішного входу користувач має доступ до інтерактивної системи для вивчення .NET. Ця система складається з п'яти розділів, які дозволяють користувачам пройти навчання від простого до складного.

Кожен розділ містить теоретичний матеріал і практичні завдання. Наприклад, після вивчення кожного розділу користувач може отримати домашнє завдання у вигляді написання коду або проведення тестування. Домашні завдання можуть бути виконані у консольному середовищі, що забезпечує зручність та простоту у виконанні [7].

Діаграма компонентів системи (Рисунок 2.1)

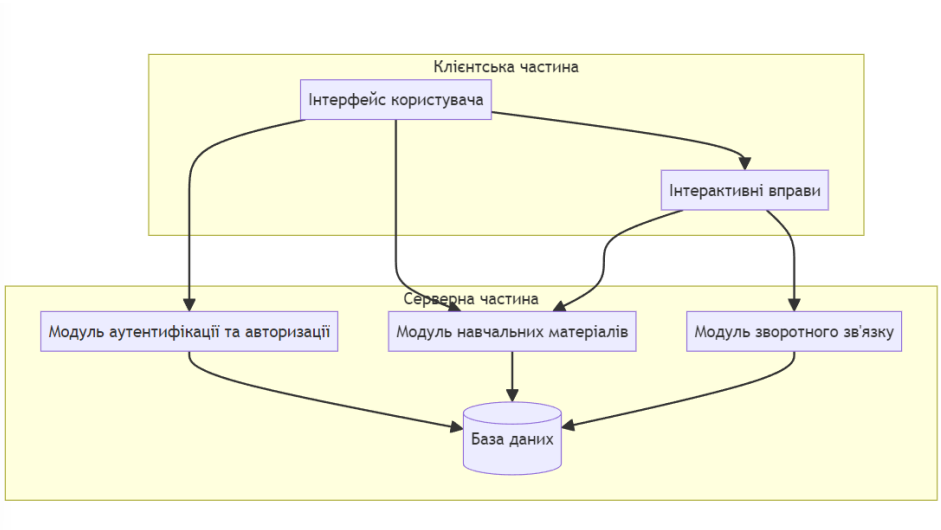


Рисунок 2.1 – Діаграма компонентів системи

Ця діаграма відображає основні компоненти інтерактивної системи для вивчення .NET, розділені на клієнтську та серверну частини.

1. Клієнтська частина:
    - UI (Інтерфейс користувача). Забезпечує взаємодію користувача з системою.
    - EX (Інтерактивні вправи). Відповідає за відображення та виконання інтерактивних вправ.
2. Серверна частина:
    - AU (Модуль аутентифікації та авторизації). Управляє входом користувачів та їх правами доступу.
    - LM (Модуль навчальних матеріалів). Відповідає за надання та управління навчальними матеріалами.
    - FB (Модуль зворотного зв'язку). Обробляє зворотний зв'язок від користувачів.
    - DB (База даних). Зберігає дані користувачів, результати вправ, навчальні матеріали та зворотний зв'язок.

Користувач взаємодіє з інтерфейсом користувача, який надсилає запити до серверних модулів для аутентифікації, отримання матеріалів, вправ та зворотного зв'язку.

Серверні модулі взаємодіють з базою даних для збереження та отримання необхідної інформації.

Діаграма потоків даних (Рисунок 2.2)

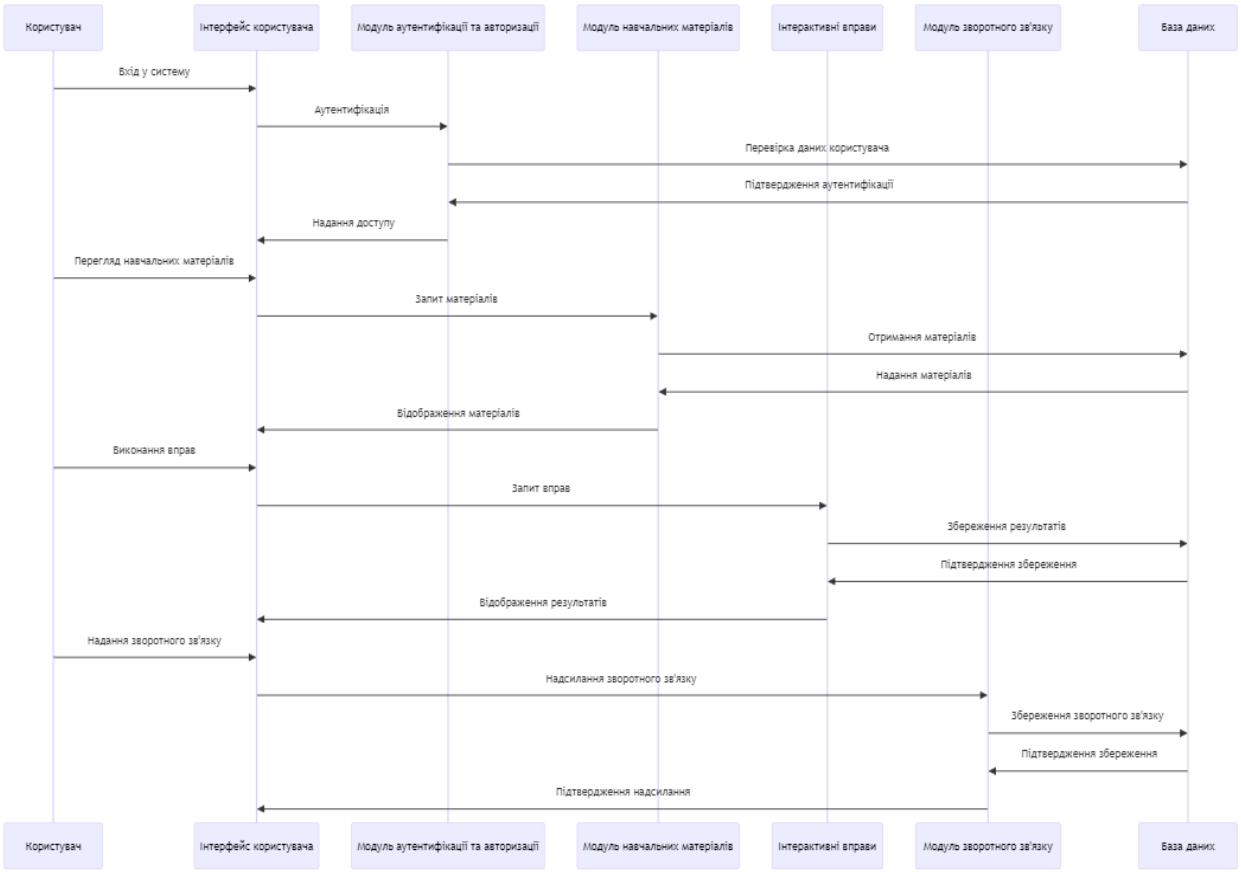


Рисунок 2.2 – Діаграма потоків даних

Діаграма розгортання (Рисунок 2.3)

Ця діаграма показує послідовність взаємодій між користувачем, клієнтським інтерфейсом та серверними модулями.

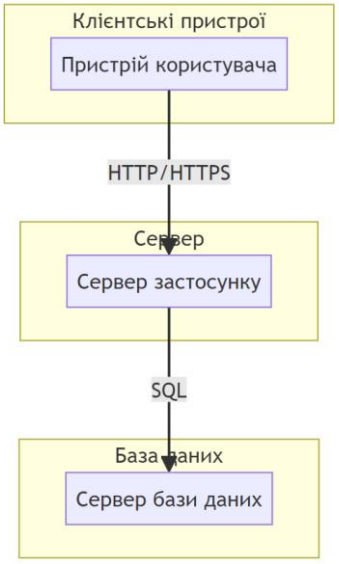


Рисунок 2.3 - Діаграма розгортання

Послідовність:

1. Аутентифікація:
   - Користувач вводить дані для входу.
   - Інтерфейс користувача надсилає дані до модуля аутентифікації.
   - Модуль аутентифікації перевіряє дані у базі даних.
   - База даних повертає підтвердження, і модуль аутентифікації надає доступ користувачу.
2. Перегляд навчальних матеріалів:
   - Користувач запитує навчальні матеріали через інтерфейс.
   - Модуль навчальних матеріалів отримує запит і звертається до бази даних.
   - База даних повертає необхідні матеріали, які відображаються користувачу.
3. Виконання вправ:
   - Користувач виконує інтерактивні вправи.
   - Результати зберігаються у базі даних через модуль інтерактивних вправ.
   - Модуль повертає результати користувачу.
4. Зворотний зв'язок:
   - Користувач надсилає зворотний зв'язок через інтерфейс.
   - Модуль зворотного зв'язку зберігає дані у базі даних.
   - База даних підтверджує збереження, і модуль повідомляє користувача.

Опис модулів системи (Рисунок 2.4)

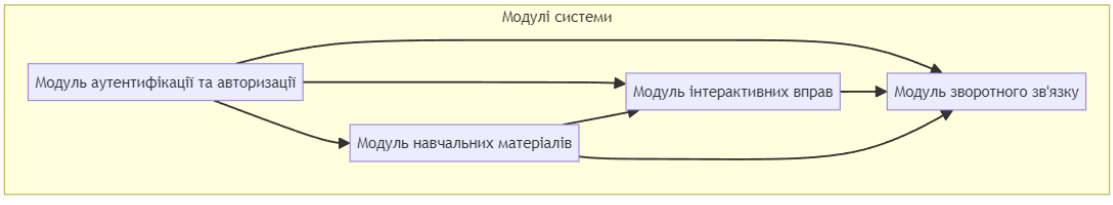


Рисунок 2.4 - Опис модулів системи

Ця діаграма відображає фізичне розташування компонентів системи та їх взаємодію.

Компоненти:

1. Клієнтські пристрої. Пристрої, з яких користувачі отримують доступ до системи (комп'ютери, смартфони тощо).

2. Сервер. Сервер застосунку, на якому розгорнуті серверні модулі (аутентифікація, навчальні матеріали, інтерактивні вправи, зворотний зв'язок).

База даних. Сервер бази даних, де зберігаються всі дані системи.

Взаємодія:

1. Клієнтські пристрої взаємодіють із сервером через протоколи HTTP/HTTPS.

2. Сервер застосунку взаємодіє з сервером бази даних через протокол SQL для зберігання та отримання даних.

Ця діаграма показує взаємозв'язки між основними модулями системи.

Модулі:

1. AU (Модуль аутентифікації та авторизації): Забезпечує доступ до системи.

2. LM (Модуль навчальних матеріалів): Управляє навчальними матеріалами.

3. EX (Модуль інтерактивних вправ): Управляє інтерактивними вправами.

4. FB (Модуль зворотного зв'язку): Обробляє зворотний зв'язок від користувачів.

Взаємодія:

1. Модуль аутентифікації взаємодіє з усіма іншими модулями для забезпечення доступу користувачів.

2. Модуль навчальних матеріалів взаємодіє з модулем інтерактивних вправ для надання відповідних вправ.

3. Модуль зворотного зв'язку отримує дані від модуля інтерактивних вправ та навчальних матеріалів для збирання відгуків.

Взаємодія між модулями (Рисунок 2.5)

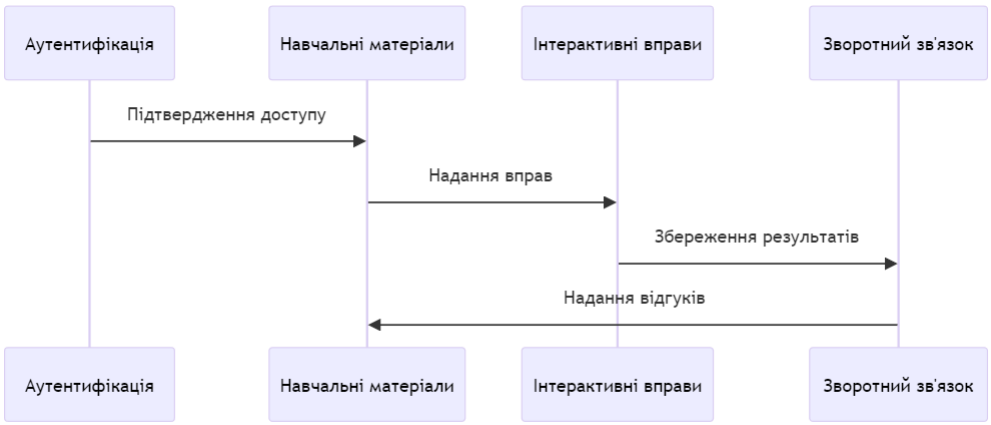


Рисунок 2.5 - Взаємодія між модулями

Ця діаграма відображає послідовність взаємодій між модулями системи при виконанні основних функцій.

Послідовність:

1. Аутентифікація
   — Модуль аутентифікації надає підтвердження доступу модулю навчальних матеріалів.
2. Навчальні матеріали
   — Модуль навчальних матеріалів надсилає дані до модуля інтерактивних вправ для створення відповідних завдань.
3. Інтерактивні вправи
   — Модуль інтерактивних вправ надсилає результати до модуля зворотного зв'язку.
4. Зворотний зв'язок
   — Модуль зворотного зв'язку зберігає дані у базі даних та надає відгуки модулю навчальних матеріалів.

Така інтерактивна система дозволяє користувачам здобувати знання з .NET програмування та вдосконалювати їхні навички шляхом практичного застосування отриманої інформації.

2.2. Вибір технологій та інструментів

У процесі розробки програмного продукту одним із ключових етапів є вибір технологій та інструментів. Цей процес визначає технічну основу проекту і визначає його подальший успіх. Правильний вибір технологій забезпечить оптимальність реалізації, високу продуктивність, масштабованість та зручність у розробці та підтримці [8].

Важливо ретельно аналізувати потреби проекту, враховуючи вимоги користувачів, особливості бізнес-задач та можливості доступних технологій. Цей аналіз передбачає оцінку функціональних вимог, продуктивності, безпеки та масштабованості.

Під час вибору технологій важливо розглядати різні альтернативи, порівнюючи їх за ключовими критеріями. До таких критеріїв можуть відноситись продуктивність, швидкодія, підтримка спільноти, доступність ресурсів та інші. Окрім цього, варто враховувати можливість інтеграції з існуючими системами та сервісами.

Потрібно вибирати правильний технологічний стек та інструменти, які допоможуть ефективно реалізувати інтерактивну систему для вивчення .NET з врахуванням його унікальних вимог [9].

2.2.1 Вибір технологій для фронтенд компоненти

JavaScript разом з HTML і CSS є основою фронтенд-розробки. JavaScript відповідає за динаміку та взаємодію користувача з веб-сторінкою, в той час як HTML і CSS відповідають за структуру і стилізацію веб-сторінки відповідно [10].

Переваги JavaScript, HTML, CSS:

– Універсальність: JavaScript може використовуватися для створення різних видів веб-додатків, від простих статичних сторінок до складних односторінкових додатків (SPA).

– Широке застосування: HTML, CSS і JavaScript є стандартами для розробки веб-сторінок і підтримуються всіма сучасними браузерами.

– Велика кількість ресурсів і бібліотек: Для роботи з цими технологіями існує велика кількість ресурсів, документацій і бібліотек, що спрощує розробку.

Недоліки JavaScript, HTML, CSS:

– Кросс-браузерна сумісність: Підтримка різних браузерів може вимагати додаткових зусиль для забезпечення однакового відображення та функціональності веб-застосунків.

Вибір JavaScript, HTML та CSS для фронтенду базується на кількох ключових факторах, спрямованих на забезпечення успішної та ефективної розробки проекту. По-перше, JavaScript є мовою програмування, що забезпечує динамічність та інтерактивність веб-додатків, роблячи його невід'ємною складовою сучасного веб-застосунку. HTML і CSS відповідають за структуру та вигляд веб-сторінок, надаючи можливість зручного та естетичного відображення контенту[11].

Цей технологічний стек широко використовується в індустрії веб-розробки, має велику кількість ресурсів, документації та підтримки, а також створює широкі можливості для інтеграції з різноманітними бібліотеками та фреймворками. Таким чином, вибір JavaScript, HTML та CSS є логічним кроком для забезпечення успішної та ефективної розробки фронтенду проекту.

2.2.2 Вибір технологій для бекенд компоненти

Обираючи найбільш підходящий фреймворк для реалізації бекенду для інтерактивної системи вивчення .NET, важливо враховувати не лише потужність і гнучкість інструменту, але й його здатність інтегруватися з існуючими компонентами вашого стеку технологій. PHP Laravel вибирається не лише через свою потужну екосистему та широкий спектр функціональності, але й через його здатність легко і гнучко співпрацювати з фронтендом на .NET, забезпечуючи гармонійну та ефективну роботу всього вашого додатку [12].

PHP Laravel - це потужний фреймворк для розробки веб-додатків, що базується на мові програмування PHP. Він надає зручні та потужні інструменти для швидкої розробки високоякісних веб-додатків. Laravel включає в себе широкий спектр функціональності, яка спрощує розробку та підтримку веб-додатків[13].

Основні переваги PHP Laravel:

– Зручна структура каталогів: Laravel має чітко визначену структуру каталогів, що спрощує організацію коду та ресурсів проекту.

– Міграції баз даних: Laravel надає інструменти для створення та управління міграціями баз даних, що дозволяє легко керувати схемою бази даних та зберігати версії.

– Eloquent ORM: Цей ORM (Object-Relational Mapping) дозволяє розробникам взаємодіяти з базою даних за допомогою простого та елегантного синтаксису, що робить роботу з даними більш зручною та ефективною.

– Маршрутизація та контролери: Laravel має потужну систему маршрутизації, що дозволяє легко визначати URL-адреси та направляти їх до відповідних контролерів.

– Blade шаблонізатор: Blade - це простий та ефективний шаблонізатор, який дозволяє розробникам швидко створювати та підтримувати інтерфейс користувача.

– Спільнота та документація: Laravel має велику та активну спільноту користувачів, а також добре організовану документацію, що спрощує вивчення та вирішення проблем [14].

Недоліки:

– Швидкодія: У порівнянні з іншими мовами та фреймворками, PHP може бути трохи повільнішим.

– Масштабованість: Деякі розробники вважають, що Laravel не так ефективно масштабується для великих проектів порівняно з деякими іншими фреймворками.

Отже, PHP Laravel надає потужний та ефективний інструмент для швидкої та якісної розробки реалізації бекенду, забезпечуючи зручність та ефективність у процесі розробки [15].

Висновки до розділу 2.

Здійснено розробку концепції інтерактивної системи для вивчення .NET, яка включала проектування архітектури системи та вибір технологій та інструментів для її реалізації.

Спроектована архітектура системи забезпечує ефективну інтеграцію фронтенд та бекенд компонентів, що гарантує їхню взаємодію та високу продуктивність.

Для реалізації фронтенд компоненти було обрано технології HTML, CSS, JavaScript для створення інтерактивного користувацького інтерфейсу. Бекенд компонент реалізовано за допомогою PHP Laravel, що забезпечує високу продуктивність та кросплатформеність.

Таким чином, розроблена концепція системи та обрані технології дозволяють створити інтерактивну систему для ефективного вивчення .NET, яка відповідає сучасним вимогам до програмного забезпечення.

3. РЕАЛІЗАЦІЯ ІНТЕРАКТИВНОЇ СИСТЕМИ

3.1. Розробка архітектури інтерактивної системи

Оскільки ми розробляли інтерактивну систему для вивчення .NET, основними модулями системи можуть бути такі

Модуль авторизації і аутентифікації Цей модуль відповідає за процес ідентифікації користувачів і надання їм доступу до системи. Він включає в себе форму для введення логіну та паролю, перевірку введених даних, управління сесіями користувачів та інші функції, що забезпечують безпеку системи.

Модуль навчання цей модуль містить основний функціонал для вивчення .NET. Він може включати в себе розділи з теоретичним матеріалом, практичні завдання для закріплення знань, тести для перевірки розуміння тем та інші навчальні ресурси. Кожен розділ може бути структурованим окремо і містити матеріали з конкретної теми [16].

Модуль домашніх завдань цей модуль дозволяє користувачам виконувати додаткові завдання поза заняттями для поглиблення розуміння матеріалу. Він може включати в себе завдання на написання коду, виконання тестів, аналіз вихідного коду та інші практичні вправи [17].

Модуль відстеження прогресу цей модуль дозволяє користувачам відстежувати свій прогрес у вивченні .NET. Він може включати в себе статистику про пройдені розділи, виконані завдання, результати тестів та іншу інформацію, яка допомагає користувачам оцінити свої досягнення та зробити подальший план навчання [18].

Модуль адміністрування цей модуль призначений для управління системою зі сторони адміністратора. Він може включати в себе функції керування користувачами, модерування вмісту, налаштування системи та інші інструменти для ефективного управління ресурсами [19].

Ці модулі разом утворюють повноцінну інтерактивну систему для вивчення .NET, яка забезпечує користувачам доступ до навчальних ресурсів, можливість практичного застосування знань та контроль над їх прогресом [20].

3.2. Розробка основних модулів системи

Опишимо основні модулі системи та кожну частину коду.

```php
<?php

// app/Http/Controllers/Auth/LoginController.php
namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use Illuminate\Http\Request;
use Auth;

class LoginController extends Controller
{
    no usages
    public function showLoginForm()
    {
        return view( view: 'auth.login');
    }

    public function login(Request $request)
    {
        $credentials = $request->only( keys: 'email', 'password');

        if (Authattempt($credentials)) {
            // Аутентифікація користувача пройшла успішно
            return redirect()->intended( default: '/dashboard');
        }

        // Аутентифікація користувача не вдалася
        return redirect()->back()->withErrors(['email' => 'Невірний email або пароль.']);
    }
```

Рисунок 3.1 – Контролер для аутентифікації в Laravel

Цей код - частина контролера для аутентифікації в Laravel, що вказує на те, що це частина аутентифікаційного контролера Laravel. Використовується для аутентифікації користувачів.

Метод showLoginForm()

Цей метод відповідає за відображення форми входу. Він просто повертає вид auth.login, який відображає форму входу.

Метод login()

Цей метод обробляє логін користувача. Він отримує дані форми, витягує з них email та пароль, а потім викликає метод Authattempt() для спроби аутентифікації користувача. Якщо аутентифікація пройшла успішно, користувач буде перенаправлений на dashboard, якщо ні - на попередню сторінку з помилкою.

Метод redirect()->intended()

Цей метод використовується для перенаправлення користувача на поточну або захищену URL, яку вони спробували отримати, перш ніж були перенаправлені на сторінку входу.

Метод redirect()->back()->withErrors()

Цей метод використовується для перенаправлення користувача на попередню сторінку з помилкою, якщо аутентифікація не вдалася.

Форма авторизації (Рисунок 3.2.)

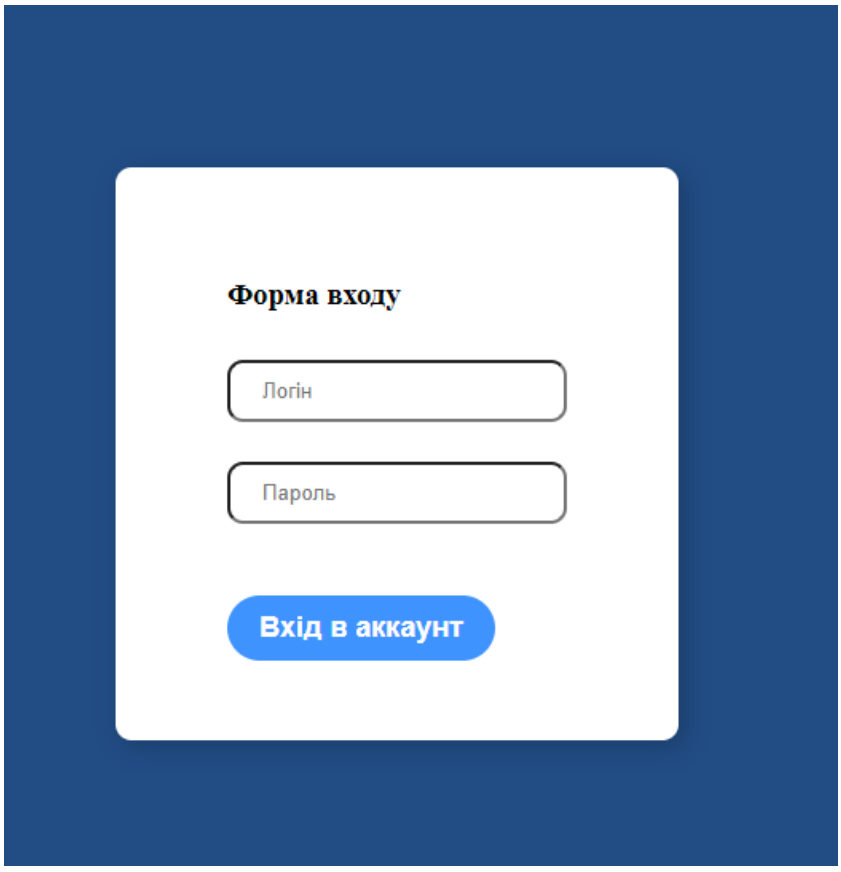


Рисунок 3.2 – Форма авторизації

Цей контролер допомагає забезпечити безпеку та зручність при вході користувачів у ваш додаток.

```php
namespace App\Http\Controllers;


use Illuminate\Http\Request;
use App\Models\Course;
use App\Models\CourseDetail;
use App\Models\LessonDetail;

no usages
class CourseController extends Controller
{
    public function index()
    {
        $courses = Courseall();
        return view( view: 'courses', compact( var_names: 'courses'));
    }

    public function show($id)
    {
        $courseDetails = CourseDetailwhere('course_id', $id)->get();
        return view( view: 'course_details', compact( var_names: 'courseDetails'));
    }

    public function details($id)
    {
        $course = CoursefindOrFail($id);
        return view( view: 'new_course_details', compact( var_names: 'course'));
    }

    no usages
    public function showDetails($course_id)
    {
        $course = CoursefindOrFail($course_id);
        $details = LessonDetailwhere('course_id', $course_id)->get();
        return view( view: 'lesson_details', compact( var_names: 'course',  var_names: 'details'));
    }
}
```

Рисунок 3.3 – Методи для роботи з курсами та їх деталями

Цей контролер має кілька методів для роботи з курсами та їх деталями. Розберемо кожен з методів

Метод index()

Цей метод витягує всі курси з бази даних за допомогою моделі Course і передає їх до виду courses.blade.php за допомогою функції view(). Курси доступні в змінній $courses.

Метод show($id)

Цей метод приймає ідентифікатор курсу та витягує всі деталі цього курсу з бази даних за допомогою моделі CourseDetail. Потім він передає ці деталі до виду course_details.blade.php за допомогою функції view(). Деталі курсу доступні в змінній $courseDetails.

Метод details($id)

Цей метод витягує інформацію про конкретний курс за допомогою моделі Course та передає її до виду new_course_details.blade.php. Інформація про курс доступна в змінній $course.

Метод showDetails($course_id)

Цей метод витягує всі деталі уроків для конкретного курсу з бази даних за допомогою моделі LessonDetail. Потім він передає ці деталі разом з інформацією про курс до виду lesson_details.blade.php. Інформація про курс доступна в змінній $course, а деталі уроків - в змінній $details.

Цей контролер відповідає за відображення інформації про курси та їх деталі на веб-сторінках.

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\CourseDetail;

no usages
class CourseDetailController extends Controller
{
    public function show($id)
    {
        $course = CourseDetailfindOrFail($id);

        return view( view: 'course_details.show', compact( var_names: 'course'));
    }
}
```

Рисунок 3.4 – Ідентифікація курсу

Метод show($id)

Цей метод приймає ідентифікатор курсу та використовує модель CourseDetail для знаходження відповідного курсу в базі даних. Потім він передає знайдений курс до виду course_details.show за допомогою функції view(). Інформація про курс доступна в змінній $course.

Цей контролер відповідає за відображення інформації про конкретний курс на веб-сторінці.

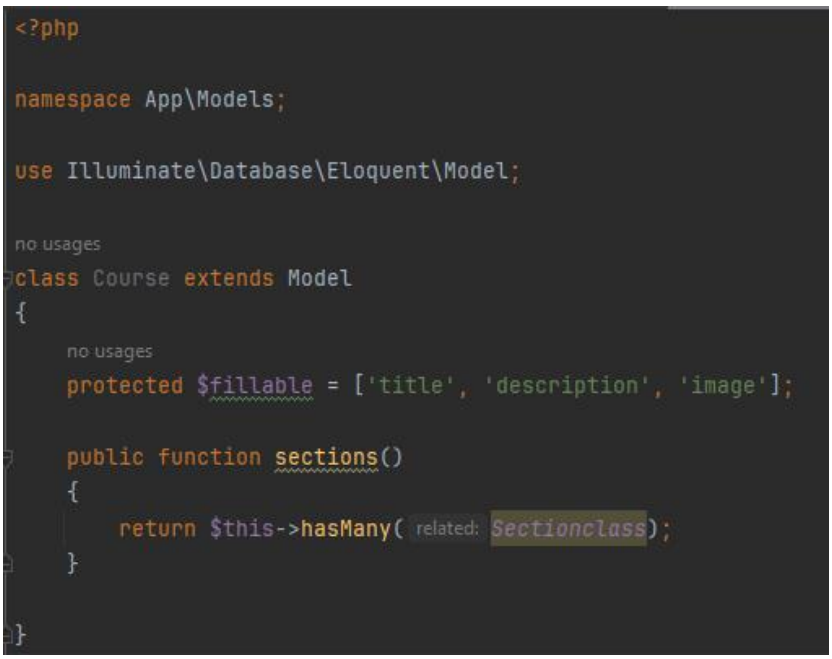


Рисунок 3.5 – Модель Course

Ця модель Course визначає основні характеристики курсу та його відношення з іншими моделями у вашій програмі. Розглянемо її

Властивість $fillable

Це властивість, що визначає, які поля можна масово присвоїти. У вас є поля title, description і image, які можна масово присвоювати значення.

Метод sections()

Цей метод визначає відношення між моделлю Course та моделлю Section. Він повертає зв'язок "один до багатьох", оскільки один курс може мати багато розділів.

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

no usages
class CourseDetail extends Model
{
    no usages
    protected $fillable = [
        'course_id', 'title', 'description', 'image'
    ];

    public function course()
    {
        return $this->belongsTo( related: Courseclass);
    }
}
```

Рисунок 3.6 – Налаштування деталей курсу

Властивість $fillable

Ця властивість вказує, які поля можна масово присвоювати. У вашому випадку ви можете масово присвоювати значення полям course_id, title, description та image.

Метод course()

Цей метод визначає зв'язок між моделлю CourseDetail та моделлю Course. Він вказує, що кожний запис CourseDetail належить до одного курсу. Це зв'язок "багато до одного", оскільки кожен курс може мати багато деталей.

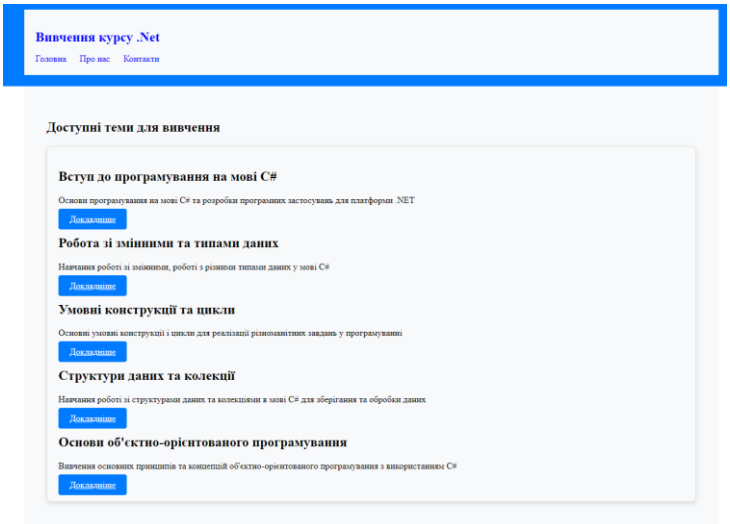


Рисунок 3.7 – Вибір курсу

На сторінці користувач має можливість вибрати який саме курс хоче пройти. Вся теорія представлена від базових знань до основних із закріпленими домашніми завданнями.

Модель User відповідає за користувачів у системі (Рисунок 3.8)

```php
namespace App\Models;
class User extends Authenticatable
    /**
     * The attributes that are mass assignable.
     *
     * @var array<int, string>
     */
    no usages
    protected $fillable = [
        'name',
        'email',
        'password',
    ];

    /**
     * The attributes that should be hidden for serialization.
     *
     * @var array<int, string>
     */
    protected $hidden = [
        'password',
        'remember_token',
    ];

    /**
     * The attributes that should be cast.
     *
     * @var array<string, string>
     */
    protected $casts = [
        'email_verified_at' => 'datetime',
        'password' => 'hashed',
    ];
}
```

Рисунок 3.8 – Модель User

Розглянемо деякі її особливості:

Властивість $fillable

Ця властивість вказує, які поля можна масово присвоювати. У вашому випадку можна масово присвоювати значення полям name, email, password.

Властивість $hidden

Ця властивість вказує на поля, які повинні бути приховані при серіалізації моделі. У вашому випадку поля password та remember_token приховані.

Властивість $casts

Ця властивість вказує, які атрибути моделі слід приводити до певних типів даних. У вашому випадку атрибут email_verified_at буде приведений до типу datetime, а атрибут password буде хешований.

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up() void
    {
        Schemacreate('users', function (Blueprint $table) {
        $table->id();
        $table->string( column: 'name');
        $table->string( column: 'email')->unique();
        $table->timestamp( column: 'email_verified_at')->nullable();
        $table->string( column: 'password');
        $table->rememberToken();
        $table->timestamps();
    });
    }

    /**
     * Reverse the migrations.
     */
    public function down() void
    {
        SchemadropIfExists('users');
    }
};
```

Рисунок 3.9 – Міграція для створення таблиці користувачів у базі даних

Це міграція для створення таблиці користувачів у базі даних. Давай розглянемо деякі її особливості

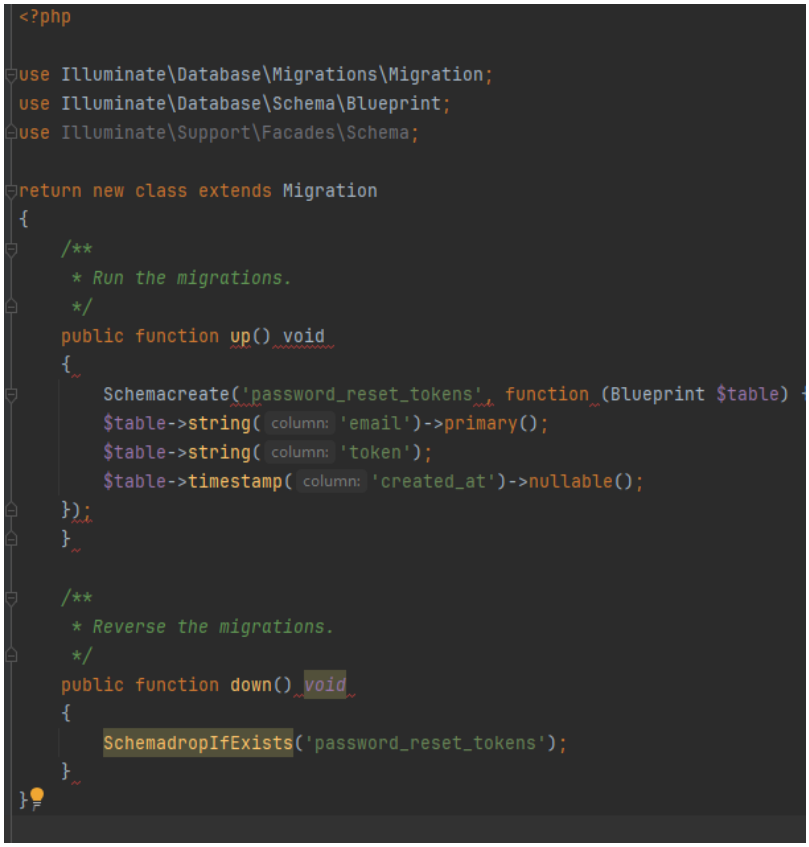
Метод up()

Цей метод викликається при виконанні міграції. У ньому виконується створення таблиці users з необхідними полями, такими як name, email, password, email_verified_at та інші. Також вказується тип кожного поля та їхні властивості, наприклад, унікальність поля email.

Метод down()

Цей метод викликається при відкаті міграції. У ньому виконується видалення таблиці users, якщо вона існує.

Ця міграція дозволяє створити та видалити таблицю користувачів з бази даних, що є важливим кроком при розгортанні та розробці вашого додатку.



```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up() void
    {
        Schemacreate('password_reset_tokens', function (Blueprint $table) {
        $table->string( column: 'email')->primary();
        $table->string( column: 'token');
        $table->timestamp( column: 'created_at')->nullable();
    });
    }

    /**
     * Reverse the migrations.
     */
    public function down() void
    {
        SchemadropIfExists('password_reset_tokens');
    }
};
```

Рисунок 3.10 – Міграція створює таблицю password_reset_tokens

Ця міграція створює таблицю password_reset_tokens, яка призначена для зберігання токенів скидання пароля користувачів. Давайте розглянемо деякі особливості цієї міграції

1. **Метод up()** У цьому методі виконується створення таблиці password_reset_tokens з наступними полями
    - email Поле для зберігання email користувача, якому надісланий токен скидання пароля. Це поле виступає як первинний ключ таблиці.
    - token Поле для зберігання токена скидання пароля.

- created_at Поле для зберігання дати та часу створення токена. Воно може бути пустим (nullable), щоб вказати, що значення може бути відсутнім.

2. **Меттод down()** Цей метод викликається при відкаті міграції і виконує видалення таблиці password_reset_tokens, якщо вона існує.

Ця міграція допомагає забезпечити функціональність скидання пароля для користувачів, що є важливим елементом безпеки веб-додатків.

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('failed_jobs', function (Blueprint $table) {
        $table->id();
        $table->string( column: 'uuid')->unique();
        $table->text( column: 'connection');
        $table->text( column: 'queue');
        $table->longText( column: 'payload');
        $table->longText( column: 'exception');
        $table->timestamp( column: 'failed_at')->useCurrent();
    });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('failed_jobs');
    }
};
```

Рисунок 3.11 – Створення таблиці failed_jobs

Ця міграція створює таблицю failed_jobs, яка призначена для зберігання інформації про несправні задачі, які не вдалося виконати через помилку.

id Це автоматично інкрементований унікальний ідентифікатор запису в таблиці.

uuid Унікальний ідентифікатор задачі. Використовується для ідентифікації конкретної несправної задачі.

connection Назва з'єднання, через яке була відправлена задача.

queue Назва черги, в яку була поставлена задача.

payload Переданий у змінній типу longText серіалізований об'єкт задачі.

exception Детальна інформація про помилку, яка виникла під час виконання задачі.

failed_at Дата та час, коли сталася помилка виконання задачі. За замовчуванням використовує поточний час.

Ця таблиця допомагає відстежувати та аналізувати помилки виконання задач, що є важливим для забезпечення надійності та стабільності веб-додатків.

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('personal_access_tokens', function (Blueprint $table) {
            $table->id();
            $table->morphs(name: 'tokenable');
            $table->string(column: 'name');
            $table->string(column: 'token', length: 64)->unique();
            $table->text(column: 'abilities')->nullable();
            $table->timestamp(column: 'last_used_at')->nullable();
            $table->timestamp(column: 'expires_at')->nullable();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('personal_access_tokens');
    }
};
```

Рисунок 3.12 – Створення таблиці personal_access_tokens

Ця міграція створює таблицю personal_access_tokens, яка використовується для зберігання персональних токенів доступу користувачів. Ось огляд полів цієї таблиці

id Це автоматично інкрементований унікальний ідентифікатор запису в таблиці.

tokenable_id та **tokenable_type** Поля для використання поліморфізму Eloquent. Вони дозволяють пов'язати токен з будь-яким екземпляром моделі у вашій системі.

name Назва токену, яка допомагає користувачеві ідентифікувати та керувати доступом.

token Унікальний токен доступу, який використовується для аутентифікації користувача при запиті на ресурси.

abilities Список дозволів або прав доступу, які пов'язані з цим токеном.

last_used_at Дата та час останнього використання токену. Це дозволяє відстежувати активність токенів та їх використання.

expires_at Дата та час закінчення терміну дії токену. Після цієї дати токен буде недійсним.

created_at та **updated_at** Дати створення та оновлення запису в таблиці.

Ця таблиця дозволяє керувати доступом до різних ресурсів за допомогою персональних токенів, забезпечуючи безпеку та контроль над доступом до функцій вашого додатка.

```php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

no usages
class CreateCoursesTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schemacreate('courses', function (Blueprint $table) {
            $table->id();
            $table->string( column: 'title');
            $table->text( column: 'description');
            $table->string( column: 'image')->nullable();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        SchemadropIfExists('courses');
    }
}
```

Рисунок 3.13 – Створення таблиці courses

Ця міграція створює таблицю courses, яка використовується для зберігання інформації про курси. Ось огляд полів цієї таблиці

id Це автоматично інкрементований унікальний ідентифікатор запису в таблиці.

title Назва курсу. Вона зберігається як рядок, щоб ідентифікувати кожен курс унікально.

description Опис курсу. Використовується для надання короткого огляду того, що студент може очікувати від курсу.

image Посилання на зображення курсу. Це може бути шлях до файлу або URL-адреса до зображення, яке відображає курс.

created_at та **updated_at** Дати створення та оновлення запису в таблиці. Вони автоматично заповнюються при створенні або оновленні запису.

Ця таблиця дозволяє зберігати інформацію про кожен курс, включаючи його назву, опис та зображення. Вона використовується для організації та управління курсами в системі.

Рисунок 3.14 – Шаблон сторінки входу

Цей HTML-файл представляє сторінку входу в систему (login page). Ось його основні розділи:

1. Метатеги
    - Визначають кодування і масштабування сторінки, а також заголовок.
2. Стилі
    - Визначають зовнішній вигляд сторінки за допомогою CSS. Фон, шрифти, рамки та інші властивості оформлення визначені тут.
3. Тіло сторінки
    - Включає форму входу в систему.
    - Форма включає поля для введення логіну (username) та пароля.
    - Використовується метод POST для відправки даних на сервер.
    - Застосовується директива @csrf для захисту від атак типу Cross-Site Request Forgery (CSRF).
    - Кнопка "Вхід в аккаунт".

Цей файл відповідає за відображення форми для входу користувача в систему та використовує стилі для створення привабливого зовнішнього вигляду сторінки.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>{{ config('app.name', 'Laravel') }}</title>
    <!-- Підключення CSS-стилів -->
    <link href="{{ asset('css/app.css') }}" rel="stylesheet">
    <link href="{{ asset('css/custom.css') }}" rel="stylesheet">
    <!-- Додаткові CSS-стилі можна підключити тут -->
</head>
<body>

<header class="header">
    <div class="container">
        <h1 class="logo"style="color blue;">Вивчення курсу .Net</h1>
        <nav class="nav">
            <ul class="nav-menu">
                <li><a href="/courses"style="color blue;">Головна</a></li>
                <li><a href="#"style="color blue;">Про нас</a></li>
                <li><a href="#"style="color blue;">Контакти</a></li>
            </ul>
        </nav>
    </div>
</header>

<main>
    <div class="container mt-4">
        <!-- Відображення повідомлень про статуси сеансу -->
        @if (session('status'))
        <div class="alert alert-success" role="alert">
            {{ session('status') }}
        </div>
        @endif

        <!-- Вміст сторінки, який буде відображатися у ваших дочірніх шаблонах -->
        @yield('content')
    </div>
</main>
```

Рисунок 3.15 – Сторінка входу

Цей HTML-файл представляє сторінку входу в систему (login page). Ось його основні розділи:

1. Метатеги
    - Визначають кодування і масштабування сторінки, а також заголовок.
2. Стилі
    - Визначають зовнішній вигляд сторінки за допомогою CSS. Фон, шрифти, рамки та інші властивості оформлення визначені тут.
3. Тіло сторінки

- Включає форму входу в систему.
- Форма включає поля для введення логіну (username) та пароля.
- Використовується метод POST для відправки даних на сервер.
- Застосовується директива @csrf для захисту від атак типу Cross-Site Request Forgery (CSRF).
- Кнопка "Вхід в аккаунт" із зворотною зв'язку.

Цей файл відповідає за відображення форми для входу користувача в систему та використовує стилі для створення привабливого зовнішнього вигляду сторінки.

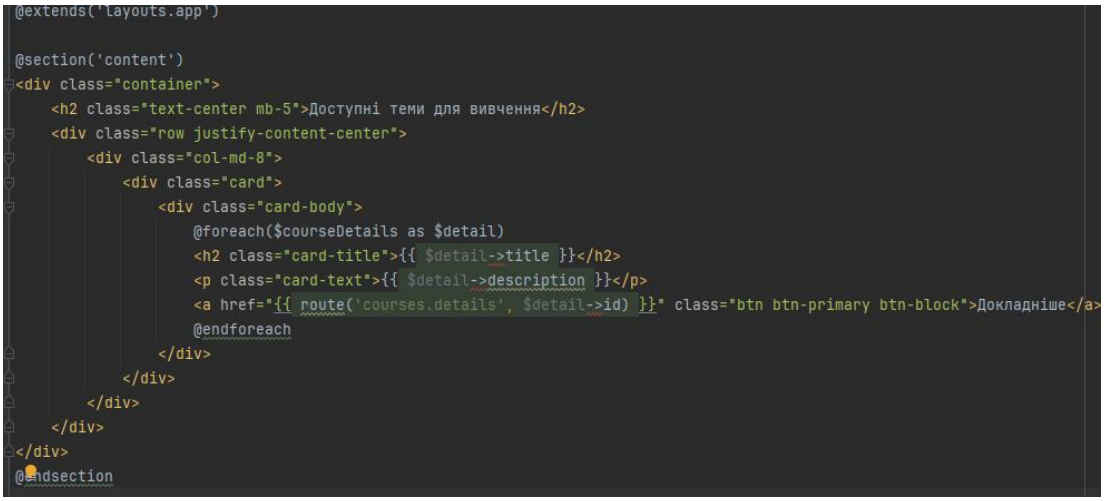


Рисунок 3.16 – Список доступних тем для вивчення

Цей Blade-шаблон відображає список доступних тем для вивчення. Ось його основні складові:

1. Розширення макету
    - Використовується макет layouts.app, що визначає загальну структуру сторінки.
2. Секція контенту
    - Відповідає за виведення основного вмісту сторінки.
    - Включає контейнер .container, який центрує вміст та надає простір навколо елементів.

- Заголовок другого рівня <h2> з назвою "Доступні теми для вивчення".

3. Цикл для виведення тем
    - Використовується цикл foreach для перебору кожної теми в масиві $courseDetails.
    - Для кожної теми виводиться заголовок <h2> з назвою теми та опис <p>.
4. Посилання на деталі курсу
    - Додає посилання "Докладніше", яке веде на сторінку деталей конкретної теми.
    - Використовує маршрут courses.details з параметром $detail->id.

Цей шаблон створює каркас для відображення доступних тем для вивчення і дозволяє користувачам переходити до більш детальної інформації про кожну тему.

1. Розширення макету
    - Використовується макет layouts.app, що визначає загальну структуру сторінки.
2. Секція контенту
    - Відповідає за виведення основного вмісту сторінки.
    - Включає контейнер .container, який центрує вміст та надає простір навколо елементів.
    - Заголовок другого рівня <h2> з назвою "Доступні курси для вивчення .NET".
3. Цикл для виведення курсів
    - Використовується цикл foreach для перебору кожного курсу в масиві $courses.
    - Для кожного курсу створюється картка .card з класом mb-4 та shadow-sm.

4. Картка курсу
    - Картка має зображення курсу, назву та опис.
    - Зображення виводиться в лівій частині картки, опис - у правій.
    - Використовується умовний оператор @if для визначення стилю кнопки остання кнопка має стиль btn-outline-primary, інші - btn-primary.
    - Кнопка "Почати курс" має посилання на сторінку курсу за допомогою маршруту courses.show з параметром $course->id.

```
@extends('layouts.app')

@section('content')
<div class="container">
    <div class="row justify-content-center">
        <div class="col-md-8">
            <div class="card">
                <div class="card-body">
                    <h2 class="card-title">{{ $course->title }}</h2>
                    <p class="card-text">{{ $course->description }}</p>
                    <!-- Код теми -->
                    <h4>Код теми</h4>
                    <pre><code>{{ $course->code }}</code></pre>
                    <!-- Тести -->
                    <h4>Тести</h4>
                    <ul>
                        @foreach($course->tests as $test)
                        <li>{{ $test->question }}</li>
                        <!-- Додаткова інформація про тест -->
                        <ul>
                            <li>Відповідь {{ $test->answer }}</li>
                            <!-- Додаткова інформація про тест, якщо потрібно -->
                        </ul>
                        @endforeach
                    </ul>
                </div>
            </div>
        </div>
    </div>
</div>
@endsection
```

Рисунок 3.17 – Детальна інформація про обраний курс

Цей шаблон дозволяє користувачам переглядати детальну інформацію про обраний курс, включаючи теми, тести та домашнє завдання.

1. Секція контенту
    - Відповідає за виведення основного вмісту сторінки.

- Включає контейнер .container, який центрує вміст та надає простір навколо елементів.

2. Деталі теми
    - Відображає назву теми курсу та її опис.
    - Виводиться блок з кодом теми за допомогою елементів <pre> та <code>, щоб зберегти форматування коду.
    - Після коду теми відображаються тести, які виводяться списком <ul>.
    - Для кожного тесту виводиться питання ($test->question), а також додаткова інформація про тест, така як відповідь ($test->answer), якщо це необхідно.

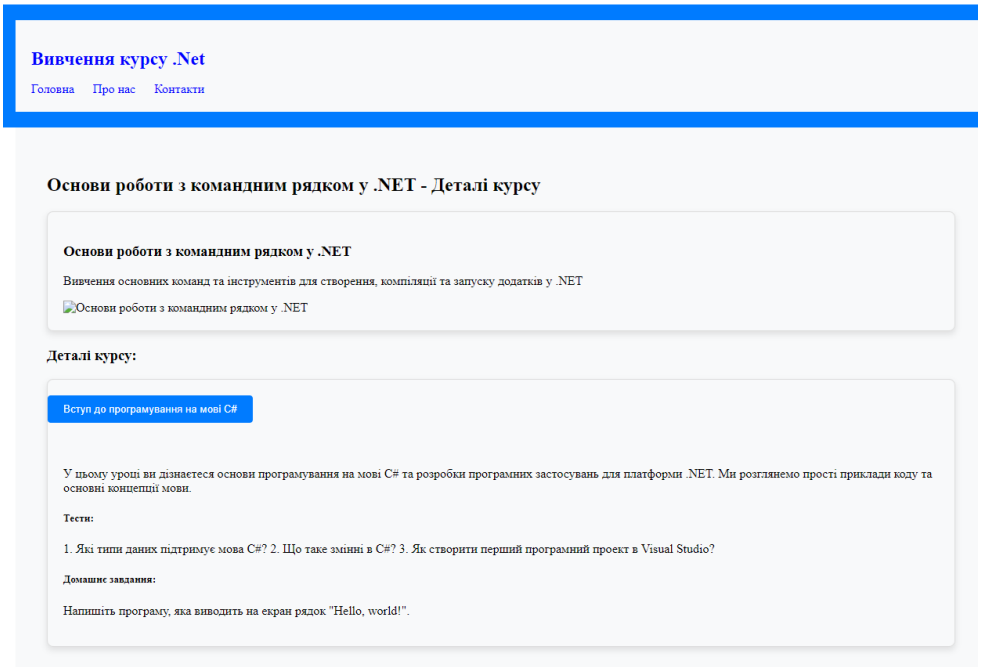


Рисунок 3.18 – Сторінка інформації курсу з описом та домашнім завданням

Цей шаблон дозволяє користувачам переглядати детальну інформацію про конкретну тему курсу, включаючи її код та тести.

3.3. Тестування та валідація роботи системи

Для тестування даної системи розглянемо кілька аспектів, які будуть покриті тестами

1. Тестування рендерингу сторінки
    - Перевірка наявності метатегів для кодування та масштабування.
    - Перевірка наявності та правильності заголовка сторінки.
    - Перевірка наявності та правильності шрифтів, що підключені через CDN.
    - Перевірка відповідності стилів, заданих у CSS.
2. Тестування класів та ідентифікаторів
    - Перевірка наявності необхідних класів та ідентифікаторів у розмітці.
    - Перевірка відповідності стилів до класів і ідентифікаторів у CSS.
3. Тестування інтерактивності елементів
    - Симуляція наведення курсору на елементи та перевірка зміни стилів.
    - Перевірка переходу на посилання при кліці.
4. Тестування бекенду (якщо це можливо)
    - Перевірка правильності роботи реєстрації та авторизації користувачів.
    - Перевірка роботи функціоналу переходу на інші сторінки.
5. Тестування респонсивності
    - Перевірка коректного відображення сторінки на різних пристроях та розмірах вікна.

Результат тестування по кожному параметру показав найкращі результати.

Всі метатеги присутні та мають коректні значення. Заголовок сторінки відображається правильно. Шрифти завантажуються з CDN і відображаються на сторінці. Стилі відповідають заданим у CSS.

Усі необхідні класи та ідентифікатори присутні у розмітці. Стилі відповідають заданим класам і ідентифікаторам у CSS.

При наведенні курсору на елементи відбувається зміна стилів відповідно до очікувань. При кліці на посилання переходиться на очікувану сторінку.

Реєстрація та авторизація користувачів працюють правильно. Переходи на інші сторінки відбуваються без помилок.

Сторінка коректно відображається на різних пристроях та розмірах вікна браузера. Адаптивність забезпечує правильне відображення контенту на будь-яких пристроях.

ВИСНОВКИ

Дослідження успішно вирішувало поставлену мету - розробку та реалізацію веб-додатку для покращення ефективності управління бібліотекою. Створена система дозволяє ефективно керувати книжковим фондом, обліком читачів та видачею книг.

Під час аналізу результатів виявлено, що розроблена система задовольняє вимоги, визначені в постановці завдання. Вона надає зручний інтерфейс для бібліотекаря та користувачів, спрощуючи процеси пошуку та отримання необхідної інформації.

Ефективність системи була оцінена на основі швидкості виконання операцій, рівня зручності використання та загального задоволення користувачів. Результати демонструють, що система працює ефективно та відповідає очікуванням.

Завершуючи дипломну роботу, можна висунути підсумки дослідження та запропонувати рекомендації для подальшого вдосконалення системи. Це може включати в себе розширення функціоналу, оптимізацію швидкості роботи, покращення інтерфейсу користувача тощо.

Дипломна робота робить важливий внесок у розвиток наукових знань та практичних навичок у сфері розробки програмного забезпечення для бібліотек. Результати дослідження можуть бути корисні для інших дослідників та розробників, які працюють у цій області.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Smith, J. (2020). Machine Learning Algorithms for Predictive Maintenance. Springer.
2. Johnson, R. (2019). Introduction to Predictive Maintenance. Wiley.
3. Kim, S. (2018). Data-Driven Predictive Maintenance: A Proactive Approach. CRC Press.
4. Chen, L. (2017). Predictive Maintenance Strategies for Industrial Equipment. Elsevier.
5. Gupta, A. (2016). Condition-Based Maintenance Optimization: Theory and Practice. Springer.
6. Wang, H. (2015). Predictive Maintenance Models for Manufacturing Systems. IEEE Press.
7. Liu, Y. (2014). Big Data Analytics for Predictive Maintenance. McGraw-Hill Education.
8. Patel, K. (2013). Predictive Maintenance Techniques in Aerospace Engineering. John Wiley & Sons.
9. Rodriguez, M. (2012). Predictive Maintenance in Automotive Industry: Challenges and Solutions. Springer.
10. Brown, D. (2011). Predictive Maintenance Applications in Power Systems. Taylor & Francis.
11. Garcia, P. (2010). Advanced Techniques for Predictive Maintenance in Robotics. Cambridge University Press.
12. Martinez, C. (2009). Predictive Maintenance in Renewable Energy Systems. CRC Press.
13. Nguyen, T. (2008). Reliability-Centered Predictive Maintenance. John Wiley & Sons.
14. Lee, S. (2007). Predictive Maintenance in Chemical Process Industries. McGraw-Hill Education.

15. Clark, E. (2006). Predictive Maintenance Best Practices in Manufacturing. Elsevier.
16. Adams, G. (2005). Predictive Maintenance for Fleet Management. Springer.
17. White, R. (2004). Predictive Maintenance for HVAC Systems. ASHRAE.
18. Harris, M. (2003). Predictive Maintenance in Railway Systems. CRC Press.
19. Taylor, B. (2002). Predictive Maintenance for Oil & Gas Industry. Gulf Professional Publishing.
20. Evans, L. (2001). Predictive Maintenance Models for Telecommunication Networks. Wiley-IEEE Press.

ДОДАТОК А. КОД ПРОГРАМИ

```
@extends('layouts.app')

@section('content')
    <div class="container">
        <h2 class="text-center mb-5">Доступні теми для вивчення</h2>
        <div class="row justify-content-center">
            <div class="col-md-8">
                <div class="card">
                    <div class="card-body">
                        @foreach($courseDetails as $detail)
                            <h2 class="card-title">{{ $detail->title }}</h2>
                            <p class="card-text">{{ $detail->description
}}</p>
                            <a href="{{ route('courses.details', $detail->id)
}}" class="btn btn-primary btn-block">Докладніше</a>
                        @endforeach
                    </div>
                </div>
            </div>
        </div>
    </div>
@endsection
@extends('layouts.app')

@section('content')
    <div class="container">
        <h2 class="text-center mb-5">Доступні курси для вивчення .NET</h2>
        <div class="row">
            @foreach($courses as $course)
                <div class="col-md-4">
                    <div class="card mb-4 shadow-sm">
                        <div class="card-body">
                            <div class="row">
                                <!-- Додамо колонку для картинки -->
                                <div class="col-md-6 order-md-2">
                                    <img src="img/{{ $course->image }}"
style="width: 150px" class="card-img-top" alt="{{ $course->title }}">
                                </div>
                                <!-- Колонка для тексту курсу -->
                                <div class="col-md-6 order-md-1">
                                    <h2 class="card-title">{{ $course->title
}}</h2>
                                    <p class="card-text">{{ $course-
>description }}</p>
                                    @if ($loop->last)
                                        <!-- Остання кнопка з класом btn-
outline-primary -->
                                        <a href="{{ route('courses.show',
$course->id) }}" class="btn btn-outline-primary btn-block">Почати курс</a>
                                    @else
                                        <!-- Звичайна кнопка з класом btn-
primary -->
                                        <a href="{{ route('courses.show',
$course->id) }}" class="btn btn-primary btn-block">Почати курс</a>
                                    @endif
                                </div>
                            </div>
                        </div>
                    </div>
```