Звіт подібності

Метадані

Назва організації	підрозділ
National University of Water and Environmental Engineering	**National University of Water and Environmental Engineering**

Заголовок

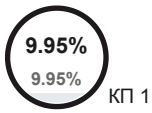
2025_Leshchuk_126_master.docx.docx

Автор	Науковий керівник / Експерт
Лещук Владислав Петрович	**Лещук Владислав Петрович**

Кількість слів	Кількість символів	Дата звіту	Дата редагування	ІД документу
11817	**91953**	**12/18/2025**	**---**	**332901720**

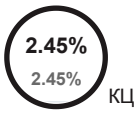
Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.



КП 1

11817
Кількість слів



КЦ

91953
Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв	ß	0
Інтервали	A→	0
Мікропробіли	·	0
Білі знаки	ß	0
Парафрази (SmartMarks)	a	39

Джерела

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://studfile.net/preview/21632686/page:3/	120 1.02 %
2	https://webtune.com.ua/statti/web-rozrobka/frontend-i-backend-rozrobka/	118 1.00 %
3	https://webtune.com.ua/statti/web-rozrobka/frontend-i-backend-rozrobka/	117 0.99 %
4	https://redstone.agency/blog/yak-vybraty-pravylnyi-tekhnolohichnyi-stek-dlia-vashoho-proektu/	73 0.62 %
5	https://ep3.nuwm.edu.ua/30214/1/04-05-84%D0%9C.pdf	68 0.58 %

6	https://otherreferats.allbest.ru/programming/01338787_0.html	67 0.57 %
7	https://webtune.com.ua/statti/web-rozrobka/frontend-i-backend-rozrobka/	52 0.44 %
8	https://studfile.net/preview/21632686/page:3/	42 0.36 %
9	https://foxminded.ua/express-js/	41 0.35 %
10	https://foxminded.ua/express-js/	33 0.28 %

з бази даних RefBooks (0.09 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
джерело: Paperity		
1	Technological and Socio-communication Aspects of Establishment and Development of Mobile Services in Ukraine Victoria Bondarenko;	11 (1) 0.09 %

з домашньої бази даних (0.11 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
2	Бакалаврська робота Корж ІСТ-41.docx [redacted] 6/13/2025 National University of Water and Environmental Engineering (National University of Water and Environmental Engineering)	13 (2) 0.11 %

з програми обміну базами даних (0.20 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
3	ФКПІ_2023_125_РуденкоАМ [redacted] 7/11/2024 Ukrainian national aviation university (Ukrainian national aviation university)	14 (2) 0.12 %
4	SUMDU/out2019/Sokol_mag_rob.pdf [redacted] 7/22/2019 Sumy State University (SUMDU)	10 (1) 0.08 %

з Інтернету (9.55 %)

ПОРЯДКОВИЙ НОМЕР	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
5	**https://webtune.com.ua/statti/web-rozrobka/frontend-i-backend-rozrobka/**	**287 (3) 2.43 %**
6	**https://studfile.net/preview/21632686/page:3/**	**214 (5) 1.81 %**
7	**https://redstone.agency/blog/yak-vybraty-pravylnyi-tekhnolohichnyi-stek-dlia-vashoho-proektu/**	**98 (2) 0.83 %**
8	**https://foxminded.ua/express-js/**	**97 (3) 0.82 %**
9	**https://ep3.nuwm.edu.ua/26949/1/04-05-71%D0%9C.pdf**	**79 (3) 0.67 %**
10	**https://ep3.nuwm.edu.ua/30214/1/04-05-84%D0%9C.pdf**	**78 (2) 0.66 %**
11	**https://otherreferats.allbest.ru/programming/01338787_0.html**	**67 (1) 0.57 %**
12	**https://uk.wikipedia.org/wiki/Node.js**	**47 (2) 0.40 %**

13	https://fashionpark.kiev.ua/index/vvedenna/uk/bd-vvedenna-v-subd-castina-4.html	36 (3) 0.30 %
14	http://document.kdu.edu.ua/metod/2021_3169.pdf	32 (3) 0.27 %
15	https://www.bartleby.com/questions-and-answers/ould-i-fix-this-issue-const-express-requireexpress-const-exphbs-requireexpress-handlebars-const-body/a28bdd44-2a54-4cc3-b1bc-63ebf9d61bbf	26 (4) 0.22 %
16	https://www.c-sharpcorner.com/article/hashing-password-with-bcrypt-in-node/	19 (3) 0.16 %
17	https://devdotcode.com/how-to-implement-many-to-many-association-in-mysql-node-js-api-using-async-await/	18 (1) 0.15 %
18	https://files.eric.ed.gov/fulltext/EJ1311305.pdf	15 (1) 0.13 %
19	https://dglib.nubip.edu.ua/server/api/core/bitstreams/51891758-348e-4713-ab6a-5a4da65cca52/content	15 (2) 0.13 %

Список прийнятих фрагментів

ПОРЯДКОВИЙ НОМЕР	ЗМІСТ	КІЛЬКІСТЬ ОДНАКОВИХ СЛІВ (ФРАГМЕНТІВ)

2

10 **МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
Національний університет водного господарства та природокористування
Навчально-науковий інститут кібернетики, інформаційних технологій та інженерії
Кафедра комп'ютерних технологій та економічної кібернетики

Допущено до захисту:
Завідувач кафедри
комп'ютерних технологій та
економічної кібернетики
д. е. н., проф. П. М. Грицюк
_____
«____» _____ 20__ р.

КВАЛІФІКАЦІЙНА РОБОТА
на здобуття ступеня «магістр»
за освітньо-професійною програмою
«Інформаційні технології в бізнесі»
спеціальності 126 «Інформаційні системи та технології»
на тему: «Інноваційні технології для формування цифрових навичок. Реалізація інформаційної системи»

10 **Виконав:**
здобувач вищої освіти 2 курсу,
групи ІТБ-61м
Лещук Владислав Петрович
Керівник:
к. т. н, доцент Джоші О.І.
2 **Рецензент: к. е. н., доцент Волошин В.С.**

2 **Рівне - 2025**

РЕФЕРАТ
Кваліфікаційна робота магістра: 74 с., 22 рис., 17 табл., 25 літературних джерел.
Актуальність теми зумовлена потребою бізнесу у фахівцях з цифровими компетенціями. Існуючі LMS (Learning Management System, система управління навчанням) не враховують індивідуальний рівень навичок користувача. Дана робота вирішує це шляхом розробки адаптивної інформаційної системи. Система проводить вхідний асесмент (тестування), діагностує прогалини у знаннях та автоматично формує персоналізовану навчальну траєкторію, рекомендуючи курси відповідно до найслабших навичок користувача. Використання стеку Node.js забезпечує гнучкість платформи.
Об'єкт дослідження - інноваційні технології для формування цифрових навичок.
Предметом дослідження реалізація інформаційної системи з використанням технологій Node.js, Express, HTML та Handlebars.
Метою магістерської роботи є проектування та розробка інформаційної системи для проведення асесменту цифрових навичок та надання адаптивного навчального контенту.
У магістерській роботі було: проаналізовано існуючі системи управління навчанням; обґрунтовано вибір технологічного стеку; спроектовано

концептуальну та логічну модель MySQL бази даних для зберігання даних про користувачів, курси, модулі, завдання та прогрес; розроблено алгоритм вхідного тестування; програмно реалізовано веб-систему, що включає функціонал автентифікації, тестування, відображення каталогу курсів, відстеження прогресу та повний CRUD-цикл для адміністрування контенту.

КЛЮЧОВІ СЛОВА: ІНФОРМАЦІЙНА СИСТЕМА, АДАПТИВНЕ НАВЧАННЯ, ЦИФРОВІ КОМПЕТЕНЦІЇ, АСЕСМЕНТ, WEB-ЗАСТОСУНОК, NODE.JS, EXPRESS, MYSQL, HTML, HANDLEBARS, LMS, CRUD.

ЗМІСТ

ВСТУП

В умовах стрімкої цифрової трансформації суспільства та бізнесу, рівень володіння цифровими компетенціями (digital skills) стає визначальним фактором конкурентоспроможності фахівців та ефективності цілих організацій. Просте накопичення навчальних годин чи сертифікатів вже є недостатнім; ключовим викликом для сучасних інформаційних систем в освіті стає забезпечення цільового, вимірюваного та ефективного формування саме тих навичок, які є критичними для конкретного фахівця чи бізнес-процесу.

Більшість існуючих LMS (такі як Moodle, Canvas LMS, Blackboard Learn, та інші)пропонують уніфікований "one-size-fits-all" підхід. Вони надають загальний каталог курсів, але не враховують індивідуальний стартовий рівень користувача та наявні в нього "прогалини у компетенціях". Як результат, користувачі витрачають час на вивчення або вже відомого їм матеріалу, або, навпаки, беруться за надто складні теми, не маючи базових знань.

Таким чином виникає гостра необхідність у створенні адаптивних інформаційних систем навчання. Мета такої системи - не просто надати доступ до контенту, а спершу провести валідований вхідний асесмент для діагностики рівня володіння ключовими цифровими навичками.

Метою магістерської роботи є вирішення науково-прикладної задачі неефективного формування цифрових компетенцій шляхом проектування та розробки адаптивної інформаційної системи. На відміну від традиційних LMS, що пропонують уніфікований каталог, дана система реалізує повний цикл персоналізованого навчання: від вхідного тестування для діагностики прогалин у знаннях за п'ятьма ключовими вимірами, до автоматичного формування рекомендацій та надання цільового навчального контенту (курсів, модулів, завдань).

Об'єктом дослідження є інноваційні технології для формування цифрових навичок.

Предметом дослідження реалізація інформаційної системи з використанням технологій Node.js, Express, HTML та Handlebars.

Методи дослідження включають: методи аналізу джерел інформації та існуючих LMS-платформ, системний аналіз для обґрунтування вибору стеку технологій, методи моделювання для проектування реляційної бази даних, а також програмування та тестування програмної реалізації і ефективності адаптивної системи.

Для досягнення поставленої мети у роботі необхідно вирішити наступні основні завдання дослідження:

1. Дослідити теоретичні основи формування цифрових навичок, проаналізувавши існуючі LMS.
2. Обґрунтувати вибір стеку інноваційних технологій (Node.js, Express, MySQL, Handlebars).
3. Спроектувати модель даних (концептуальну і логічну) та розробити алгоритмічну модель вхідного тестування.
4. Програмно реалізувати веб-систему, що включає функціонал автентифікації користувачів, проходження тестування, відображення адаптивного каталогу курсів та відстеження навчального прогресу.
5. Розробити адміністративний модуль з повним CRUD-функціоналом.
6. Провести тестування реалізованої інформаційної системи та оцінити її ефективність у вирішенні задачі формування цифрових навичок.

Практичне значення одержаних результатів полягає у тому, що розроблена інформаційна система є готовим програмним продуктом для вирішення конкретної бізнес-задачі - цільового підвищення цифрових компетенцій. На відміну від суто теоретичних моделей, дана робота представляє реалізовану веб-платформу, здатну проводити автоматизований асесмент навичок користувачів та негайно надавати персоналізовані навчальні рекомендації.

Логіка дослідження зумовила структуру **кваліфікаційної роботи на здобуття ступеня** «магістр»: **вступ, три** розділи, висновки, **список використаних джерел та додатки.**

У першому розділі аналізуються теоретичні основи формування цифрових навичок, досліджуються існуючі інформаційні системи (LMS) та їхні особливості. На основі аналізу обґрунтовується вибір сучасного стеку технологій (Node.js, Express, MySQL), необхідного для реалізації адаптивної веб-платформи.

У другому розділі розглядається процес проектування системи, що включає розробку концептуальної та логічної моделі реляційної бази даних (MySQL), доведення її відповідності нормальним формам, а також розробку алгоритмічної моделі вхідного тестування для визначення рівня

компетентності користувача.
У третьому розділі описується програмна реалізація інформаційної системи. Детально розглядається архітектура backend на Node.js та реалізація frontend з використанням HTML та шаблонізатора Handlebars. Особливу увагу приділено реалізації логіки тестування, відстеження прогресу, CRUD-функціоналу для адміністрування навчального контенту, а також автоматизованій генерації сертифікатів про успішне завершення курсів. Робота завершується висновками, що містять результати досліджень та перелік виконаних завдань, та списком використаних джерел.
Програмні продукти та технології, що використовувались під час написання кваліфікаційної роботи: середовище розробки Visual Studio Code, система управління базами даних MySQL, серверна платформа Node.js, фреймворк Express, шаблонізатор Handlebars, HTML5 та CSS3.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ЗАСАДИ

1.1 Аналіз предметної області, та що таке цифрові навички

Розвиток **цифрового середовища та глобальної мережі охоплює практично всі сфери** економіки і, без сумніву, впливає не лише на розвиток ринку праці, співвідношення на ньому вікових груп, сфер діяльності, а й вимагає від працівників нових знань і компетенцій.
Ми живемо в період, коли досягнення в області штучного інтелекту й автоматизації робочих процесів швидко прогресують, руйнуючи усталені ролі та професії. На основі вивчення потреб ринку праці були визначені найбільш затребувані фахівці та сформований список десятьох професій, необхідних сучасній економіці. А це - розробник програмного забезпечення, торговий представник, project manager, IT-адміністратор, менеджер по роботі з клієнтами, digital маркетолог, співробітник служби підтримки, аналітик даних, фінансовий аналітик і графічний дизайнер. [2]
На базовому рівні цифрові скіли визначаються як навички, необхідні для використання комп'ютерів, цифрового зв'язку, онлайн-додатків та інших цифрових пристроїв. Ці компетенції життєво необхідні на багатьох сучасних посадах.
Різні робочі місця вже протягом багатьох років постійно вимагають від співробітників володіння цифровими навичками. Електронні комунікації, комп'ютери та цифрові пристрої - усе це спонукає працівників відточувати ці компетенції.
Потреба в цифрових вміннях зростає вже давно, а з 2020 року почала зростати ще стрімкіше. Дослідження AWS Global Digital Skills Study показало, що 87% роботодавців стверджують, що пандемія COVID-19 прискорила впровадження цифрових технологій у їхніх компаніях. І це якнайкраще можна відстежити на прикладі віддаленої роботи, яка з'явилася, коли проблеми зі здоров'ям змусили багатьох співробітників залишити офіс і працювати вдома.
Під час дистанційної роботи спілкування з колегами, менеджерами та клієнтами переважно здійснюється через цифрові канали зв'язку та часто вимагає прогресивних технологічних компетенцій. Віддаленим співробітникам також потрібні навички віртуальної співпраці, такі як використання інструментів управління проєктами, а також комфорт у спілкуванні через програмне забезпечення для обміну миттєвими повідомленнями. [1]
Онлайн-освіта, штучний інтелект, автоматизація - те, що ще кілька років тому здавалося майбутнім, сьогодні визначає конкурентоспроможність фахівців і бізнесу. Зараз успіх у професії визначає не лише досвід, а й здатність працювати з цифровими платформами та інноваційними технологіями. Цифрові навички відкривають українцям доступ до сучасних можливостей - від розвитку власного бізнесу до роботи на європейському ринку.
Цифрові компетенції вже давно перестали бути «додатковою перевагою» - сьогодні це одна з базових умов працевлаштування. За даними Європейської комісії, майже 90% вакансій у ЄС вимагають базових цифрових навичок, тоді як 4 із 10 дорослих їх не мають. Статистика Eurostat свідчить: у 2023 році лише 56% європейців віком 16-74 років володіли мінімальними цифровими компетенціями, хоча ціль ЄС до 2030 року - досягти 80% (рис. 1.1).

Рис. 1.1 Статистика людей, які мають принаймні базові загальні цифрові навички у 2023 році [14]

Світова економіка швидко переходить у цифрову площину, і попит на фахівців, які розуміються на технологіях, лише зростає. Аналітика даних, генеративний контент, автоматизація процесів - ці вміння стають необхідними у більшості галузей. Освоєння таких інструментів відкриває нові кар'єрні можливості, допомагає швидко адаптуватися до технологічних змін і ефективніше реалізовувати власні ідеї. [3]

1.2 LMS, якими вони бувають

Системи управління навчанням, або LMS (Learning Management Systems) є необхідним інструментом для ефективної організації освітнього процесу. Вони дозволяють привнести технологічний елемент в навчальний процес і забезпечити зручність у доступі до матеріалів для учнів та викладачів. Існує багато різних LMS, але лише кілька з них можна вважати найкращими у своїй сфері.
На сьогоднішній день, до ТОП 10 LMS входять такі системи, як Moodle, Google Classroom, Canvas, Blackboard, Schoology, Edmodo, TalentLMS, Brightspace, Docebo та iSpring Learn.
Moodle відзначається своєю безкоштовністю і широкими можливостями налаштування, що робить її популярною серед навчальних закладів по всьому світу. Google Classroom, своєю чергою, інтегрується з іншими продуктами Google, пропонуючи простий у використанні інтерфейс, особливо для шкіл, що вже використовують сервіси Google. Canvas стане в пригоді університетам та коледжам завдяки своїм можливостям для організації навчального контенту та видової інтерактивності. Blackboard є класичним вибором для багатьох установ завдяки своїм потужним функціям і широкому спектру підтримки.
Інші системи, як Schoology та Edmodo, більше підходять для шкіл, оскільки надають інструменти, подібні до соціальних мереж, для взаємодії учнів і вчителів. TalentLMS, Brightspace, Docebo та iSpring Learn більше орієнтовані на корпоративне навчання, але також здатні забезпечити високу якість освіти в академічному середовищі. Усі ці LMS мають свої унікальні особливості, які роблять їх незамінними в певних умовах, показуючи, що вибір правильної системи залежить від конкретних потреб навчального закладу або організації.
У сучасному освітньому та корпоративному середовищі вибір правильної системи управління навчанням (LMS) може стати ключовим фактором успіху. LMS-системи допомагають створювати, доставляти й відстежувати навчальний контент, автоматизувати адміністрування, підтримувати дистанційне і гібридне навчання, а також підвищувати залученість учасників. Нижче наведено огляд десяти провідних LMS, які користуються популярністю в 2024-2025 роках. [5]
Порівняльний аналіз LMS наведено в табл. 1.1.
Таблиця 1.1
Порівняння LMS: Moodle, Canvas, Blackboard Learn

Показник	Learning Management System		
	Moodle	Canvas LMS	Blackboard Learn

Ліцензія та вартість Відкритий код. Безкоштовна (самостійний хостинг) / платний хостинг та підтримка від Партнерів. Платна підписка. Вартість збільшується з часом. Доступна безкоштовна версія для викладачів (з обмеженнями). Преміум-класу (Enterprise-level). Висока вартість, підходить для великих установ. Ціна індивідуальна.
Гнучкість та кастомізація Максимальна гнучкість. Високий рівень налаштування через понад 2000 плагінів. Ідеально для створення власної адаптивної логіки. Обмежена кастомізація порівняно з Moodle. Оновлення виходять щомісяця. Складна і важко керована. Сфокусована на потужних корпоративних функціях.
Користувацький інтерфейс (UI) Сучасний вигляд. Крута крива навчання для новачків через велику кількість функцій. Сучасний та інтуїтивно зрозумілий. Вважається легким у використанні. Чистий, досить інтуїтивно зрозумілий (Blackboard Ultra), але часто сприймається як застарілий.
Мобільність Підтримує мобільне навчання. Бездоганна мобільна доступність та зручність. Підтримує навчання на смартфонах.
Аналітика та звітність Базова вбудована аналітика. Глибший аналіз можливий через інтеграцію сторонніх плагінів. Надійні інструменти аналітики та звітності в режимі реального часу (Canvas Data). Потужна аналітика та інструменти для відстеження прогресу.
Впровадження Самостійний хостинг або керований (MoodleCloud). SaaS (Програмне забезпечення як послуга). SaaS (Cloud), не вимагає інсталяції. Усі оновлення автоматичні.
Спільнота та підтримка Активна глобальна спільнота, що постійно робить свій внесок (плагіни, ресурси). Активна та підтримуюча спільнота користувачів. Підтримка здебільшого через вендора (Blackboard Inc.).
Найкраще підходить для Шкіл, ЗВО, гнучких установок, установ, що потребують повного контролю та кастомізації. Установ, які надають пріоритет сучасному дизайну та простоті використання. Дуже великих університетів та установ корпоративного рівня.

Незважаючи на низку переваг розглянутих вище LMS, потрібно зауважити, що на сьогоднішній день ними не передбачено обов'язкового та системного самоаналізу поточного рівня Digital Skills або інших компетентностей користувача перед початком навчання. Як видно з порівняльної таблиці, основна функціональність цих систем сфокусована на адмініструванні навчального процесу (управління курсами, користувачами, оцінками) та наданні інструментів для комунікації та інтерактиву. Вони пропонують обширний та структурований каталог курсів, але підхід до вибору контенту залишається значною мірою пасивним, покладаючись на самостійну оцінку власних потреб і знань здобувачем освіти.
Саме тут виявляється ключова проблематика, що обґрунтовує необхідність розробки інформаційної системи. Типові LMS, не маючи вбудованого механізму початкової діагностики, не враховують індивідуальний стартовий рівень користувача та наявні в нього «прогалини у компетенціях» (skill gaps) у тій чи іншій сфері. Як результат, користувачі витрачають свій час на вивчення вже відомого їм матеріалу, що призводить до демотивації та зниження ефективності процесу. Або, навпаки, вони беруться за надто складні теми, не маючи необхідних базових знань, що створює фрустрацію та ризик не завершити курс.
Отже, існує критична потреба у створенні системи, яка б інтегрувала механізм початкового тестування та автоматичного формування індивідуальної освітньої траєкторії. Така система, на відміну від існуючих LMS, повинна не лише надавати доступ до навчального контенту, а й активно використовувати результати діагностики для персоналізованої рекомендації курсів. Це дозволить оптимізувати час навчання, сфокусувавши увагу користувача виключно на тих вимірах цифрової компетентності, де виявлено найнижчий рівень знань, забезпечуючи таким чином справді ефективний та цілеспрямований розвиток Digital Skills.

1.3 ALS та їхні переваги
Незважаючи на значний розвиток LMS, їхня базова функціональність переважно обмежується доставкою контенту та управлінням процесами. Сучасні системи часто ігнорують гетерогенність знань та потреб студентів, пропонуючи лінійний навчальний план, який є неефективним для користувачів з різним рівнем підготовки.
Це призводить до двох критичних проблем, які підривають ефективність онлайн-навчання. Перша - перенавчання (over-study), коли користувачі витрачають час на вже відомий їм матеріал, що знижує їхню мотивацію. Друга - когнітивне перевантаження (cognitive overload), коли, навпаки, вони беруться за надто складні теми, не маючи необхідних базових знань, що створює бар’єр для подальшого засвоєння. Таким чином, індивідуальний навчальний шлях залишається неоптимізованим.
З огляду на це, виникає гостра необхідність у створенні адаптивних інформаційних систем навчання (Adaptive Learning Systems - ALS), мета яких - забезпечити персоналізований навчальний досвід. Ключовим етапом у такій системі є не просто надання доступу до контенту, а спершу проведення валідованого вхідного асесменту (initial assessment) для точної діагностики рівня володіння ключовими цифровими навичками. Результати цього асесменту мають слугувати підґрунтям для динамічного формування індивідуального навчального плану.
Дослідження показують, що адаптивні системи електронного навчання, засновані на стилях навчання, є більш ефективними, оскільки вони скорочують час на навчання, підвищують рівень завершення предмета та покращують академічну успішність [15]. Були розроблені нові методи для забезпечення більш надійного та продуктивного персоналізованого навчального процесу для учнів [16].
Нижче представлені найбільш значущі розроблені адаптивні системи:
1. AHS (Adaptive Hypermedia Systems). Ця система може надавати користувачеві персоналізований досвід. Це досягається за допомогою гіпертекстів та гіпермедіа [17].
2. AEHS (Adaptive Educational Hypermedia Systems). Ці системи були створені для допомоги учням в освіті, зосереджуючись на їхніх потребах та адаптуючи навчальний матеріал залежно від попередніх знань та навчальних цілей, які обирає викладач. Таким чином, система може запропонувати навчальний досвід, який найкраще відповідає потребам учня [18].
3. AES (Adaptive Educational Systems). Відмінність від вищезгаданих систем полягає в тому, що додається фактор стилю навчання (Learning Style). Стилі навчання ґрунтуються на освітніх теоріях, і таким чином навігація та матеріал, що надається учням, адаптується не лише до їхніх потреб, але й до їхніх навчальних переваг [19]. Навчальний досвід стає приємнішим та інтерактивнішим, що призводить до досягнення навчальних цілей та активної участі студентів. Як наслідок, задоволеність навчальним процесом зростає.
4. ITS (Intelligent Tutoring System). Ця система розроблена для використання даних, які користувач генерує під час використання системи, і на основі цих даних та отриманих результатів спрямовує користувача у правильному напрямку. Система записує дані, що стосуються руху миші, необхідного часу, отриманих оцінок та типу помилок, зроблених користувачем. Аналіз таких даних призводить до створення профілю, який буде використовуватися для надання користувачеві відповідного контенту. Таким чином, користувачеві надається адаптивне навчальне середовище [20]. [9]
Розробка адаптивних до користувача систем набуває все більшого значення для промислових застосувань. Моделювання користувачів виникло з необхідності представляти в системі знання про користувача, щоб дозволити приймати обґрунтовані рішення щодо адаптації до потреб користувача. [10]

Рис. 1.2 Класична та фундаментальна модель для пояснення архітектури та принципів роботи будь-якої Адаптивної системи навчання (Adaptive Learning System, ALS) [11]

Дана схема ілюструє трьохкомпонентний процес, який лежить в основі функціонування адаптивної системи. Вона показує, як система перетворює сирі дані про користувача на персоналізований навчальний досвід.
1. Збір даних про користувача (Data about user)
Це початковий і критично важливий етап. Система (System) активно збирає (Collects) та обробляє (Processes) інформацію про користувача. На цьому етапі збираються результати валідованого вхідного асесменту (тестування).
2. Моделювання користувача (User Modeling)
Процес Моделювання користувача (User Modeling) перетворює зібрані сирі дані на структуровану, придатну для прийняття рішень, інформацію. Модель фіксує, чи є рівень користувача достатнім (наприклад, усі бали ≥ певної межі), чи є слабкі місця (знайдено мінімальний бал).
3. Адаптація та Ефект адаптації (Adaptation and Adaptation effect)
Система використовує (Processes) Модель користувача для застосування механізмів Адаптації (Adaptation).
Користувач отримує індивідуальний навчальний план, який є максимально релевантним його поточному рівню знань, що підвищує ефективність та мотивацію, а також скорочує час навчання.
Отже, адаптивні системи навчання (ALS), такі як AEHS та ITS, представляють собою еволюційний стрибок від статичних LMS, оскільки вони, використовуючи принципи Моделювання Користувача, забезпечують персоналізований навчальний досвід, що доведено підвищує академічну успішність, скорочує час навчання та збільшує задоволеність учнів, адаптуючи контент відповідно до попередніх знань та, у випадку AES, стилів навчання.

1.4 Аналіз та вибір технологічного стеку системи

Вибір технологічного стека є одним із найважливіших рішень при розробці будь-якого програмного продукту. Від цього рішення залежить продуктивність, масштабованість, безпека та загальна успішність проекту.

Технологічний стек - це набір технологій, які використовуються разом для розробки та підтримки програмного забезпечення. Зазвичай він складається з таких компонентів:

1. **Фронтенд (клієнтська частина): Технології, що використовуються для створення інтерфейсу** користувача.
2. **Бекенд (серверна частина): Технології для роботи з бізнес-логікою, базами даних та серверними** процесами.
3. **База даних: Системи для зберігання та управління** даними.
4. **Система управління версіями: Інструменти для відстеження змін у** коді.
5. Інструменти **для розгортання та підтримки: Технології для автоматизації розгортання та підтримки проекту.**

Перш за все, визначимо, які вимоги до технологій висуває адаптивна система:
1. Швидкість та гнучкість розробки (Agile): Вибір технологій, що дозволяють швидко реалізовувати та тестувати адаптивну логіку.
2. Ефективна робота з моделюванням користувача (User Modeling): Необхідність у стабільній та швидкій СУБД для зберігання та оперативного доступу до Моделі Користувача (результатів тестування та прогресу).
3. Акцент на Backend: Оскільки ядром системи є логіка адаптації (рекомендація курсів), необхідна потужна серверна частина.
4. Сумісність з Open-Source: Бажано, щоб стек був сумісний з відкритими рішеннями (наприклад, Moodle, якщо ви його оберете).
Згідно цього проаналізуємо найпопулярніші мови (наприклад, Node.js/JavaScript, Python, PHP) за критеріями, релевантними до дослідження (табл. 1.2).

Таблиця 1.2
Вибір мови програмування

Критерій	Мови програмування		
	Node.js (JavaScript)	Python	PHP
Синхронність/Асинхронність	Асинхронний, ідеально для I/O-операцій (робота з БД).	Переважно синхронний.	Синхронний.
Екосистема	Велика екосистема NPM, що містить багато бібліотек для веб-розробки.	Широко використовується для Data Science та Machine Learning.	Ідеально для розробки CMS та простих веб-додатків.
Крива навчання	Помірна, якщо вже знайомий з JavaScript.	Відносно легка.	Помірна.
Висновок для проєкту	Оптимально. Швидкий, асинхронний та єдиний стек (Full-Stack JS).	Добре, але надмірний для простого CRUD та адаптивної логіки.	Добре для традиційних веб-систем, але менш гнучкий.

Для реалізації серверної логіки адаптивної системи обрано Node.js (JavaScript). Це забезпечує асинхронну роботу, що є критично важливим для швидкого виконання запитів до бази даних, які формують Модель Користувача, а також дозволяє використовувати єдиний стек технологій для Frontend і Backend, прискорюючи розробку.
Перейдемо до порівняння СУБД реляційних та нереляційних баз даних (наприклад, MySQL, PostgreSQL, MongoDB) з точки зору зберігання структурованих даних (результати тестів, курси, прогрес):

Таблиця 1.3
Вибір СУБД

Критерій	СУБД		
	MySQL	PostgreSQL	MongoDB (NoSQL)
Тип даних	Реляційна (таблиці).	Реляційна (таблиці).	Нереляційна (документи JSON).
Схема даних	Фіксована та строга.	Фіксована, але з більшою гнучкістю.	Динамічна та гнучка.
Складність запитів	Стандартний SQL. Ідеально для складних зв'язків (JOIN).	Найкраща підтримка складних запитів та цілісності даних.	Обмежено. Складні JOIN вимагають додаткової логіки.
Висновок для проєкту	Оптимально. Ідеально підходить для зберігання структурованих даних (тести, курси, зв'язок користувачів з прогресом) та забезпечення цілісності. Сумісний з Moodle..	Чудово, але може бути надмірним для цього проєкту.	Не рекомендовано, оскільки дані

(результати тестів та прогрес) мають чітку реляційну структуру.

Тому, для забезпечення цілісності та структурованості даних, необхідних для коректного функціонування Моделі Користувача (зв'язки "Користувач - Тест - Навичка - Курс - Прогрес"), обрано MySQL. MySQL є однією з найпоширеніших реляційних СУБД, забезпечує високу надійність транзакцій та легко інтегрується з Node.js та інфраструктурою LMS.
Отже, для розробки адаптивної системи було обрано технологічний стек Node.js (Backend) та MySQL (СУБД). Ця комбінація гарантує швидке та асинхронне виконання логіки адаптації, надійне зберігання структурованих даних Моделі Користувача, а також створює єдину ефективну платформу для подальшої реалізації всіх функціональних вимог.

1.5 Архітектура та обґрунтування компонентів
Реалізація інформаційної системи «Digital Skills» ґрунтується на трирівневій архітектурі (клієнт, сервер застосунку та сервер бази даних) з використанням підходу Multi-Page Application (MPA)
Розробник може вибирати з багатьох комбінацій технологій та мов програмування для створення застосунку. Рішення приймаються на основі того, яка база даних, веб-сервер та фронтенд-фреймворк найкраще відповідають вимогам застосунку та можливостям команди. Обрана комбінація стає технологічним «стеком», на якому будується застосунок.
Двома найпопулярнішими технологічними стеками сьогодні є стек MEAN та стек MERN. Обидва є надійним вибором для створення динамічно оновлюваних веб- та мобільних додатків, і, як можна зрозуміти з їхніх назв, вони мають багато спільного. Але є ключові відмінності, які можуть спонукати розробника обрати один з них.
Стек MEAN - це комбінація технологій з відкритим кодом, які зазвичай використовуються разом для створення динамічних веб- або мобільних додатків. Ці технології - MongoDB, Express, Angular та Node.js. Усі вони базуються на JavaScript або, у випадку MongoDB, безперешкодно працюють з технологіями JavaScript .
Як і стек MEAN, стек MERN - це група технологій з відкритим кодом, що використовуються для створення веб- та мобільних додатків. Він також використовує JavaScript як основну мову програмування та складається з MongoDB, Express, React та Node.js. Цей стек зріс у популярності в останні роки. [22]
В основі нашої архітектури лежить подібний стек технологій, але з модифікацією:
1. MySQL - база даних.
2. Express.js - веб-фреймворк.
3. Handlebars - шаблонізатор.
4. Node.js - серверна платформа.
Така конфігурація забезпечує необхідну гнучкість і високу швидкість розробки, що є критично важливим для науково-дослідного проекту.
Серверна частина системи реалізована на платформі Node.js та мінімалістичному веб-фреймворку Express.js.

Рис. 1.3 Лого Node.js + Express.js

12 **Node.js - платформа з відкритим кодом для виконання високопродуктивних мережевих застосунків, написаних мовою JavaScript.**
12 **Платформа Node.js перетворила JavaScript на мову загального використання з великою спільнотою розробників. Node.js має наступні** властивості:
1. **асинхронна** однониткова модель **виконання** запитів;
2. **неблокуючий** ввід/вивід;
3. **система модулів** CommonJS;
4. рушій **JavaScript Google V8.** [23]

Вибір Node.js обумовлений його асинхронною, подієво-орієнтованою моделлю (Event-Driven Architecture). Це дозволяє системі ефективно обробляти велику кількість одночасних запитів (наприклад, реєстрація користувачів, проходження тестів, завантаження курсів) без блокування основного потоку виконання.
Оскільки більшість операцій інформаційної системи (робота з базою даних MySQL) є операціями вводу/виводу (I/O), Node.js є оптимальним вибором для забезпечення високої реактивності системи.
8 **Express.js - це фреймворк для Node.js, що надає набір функцій для створення веб- і мобільних додатків. Express.js заснований на стандартних API Node.js, що робить його простим у вивченні та використанні.**

8 **Сьогодні Express.js є невід'ємною частиною екосистеми Node.js і продовжує залишатися одним із найпопулярніших виборів для розроблення серверної частини веб-додатків.**
8 Express.js **надає безліч функцій для розробки веб-додатків, включно з** ними:
1. **Маршрутизація.**
2. **Підтримка шаблонів: інтеграція з різними шаблонізаторами (Jade, Handlebars і** EJS).
3. **Підтримка баз даних: фреймворк легко інтегрується з різними базами даних (MongoDB, PostgreSQL і** MySQL).
4. **Підтримка тестування.** [24]

У проєкті, Express.js виступає як основний каркас для побудови API та веб-застосунку. Його ключовими функціональними блоками є:
1. Маршрутизація (Routing): Визначення кінцевих точок, які відповідають на GET, POST запити. Всі маршрути системи згруповані за функціоналом: основні маршрути (/login, /register) знаходяться в index.js, логіка курсів та прогресу - у courses.js, а логіка адаптивного тестування - у test-routes.js.
2. Проміжне ПЗ (Middleware): Це функції, які виконуються послідовно між отриманням запиту та відправкою відповіді.
У системі Middleware використовується для:
1. Автентифікації та Авторизації: Функції isLoggedIn та isAdmin (courses.js, test-routes.js) гарантують, що доступ до захищених маршрутів (наприклад, /test, /courses/manage) мають лише авторизовані користувачі або адміністратори.
2. Управління Сесіями: Використання бібліотеки express-session (index.js) для зберігання стану користувача (req.session.username, req.session.id_users) між запитами, що є основою для персоналізації даних.

Таким чином, комбінація Node.js і Express.js створює масштабовану та гнучку платформу, яка ідеально підходить для реалізації асинхронних I/O-операцій та складної логіки адаптивного моделювання користувача, що є центральним елементом нашої роботи.
Для формування інтерфейсу користувача (Frontend) та забезпечення цілісності архітектури було обрано традиційний підхід Multi-Page Application (MPA), реалізований за допомогою техніки Server-Side Rendering (SSR). Цей вибір дозволив зосередити обчислювальну потужність на реалізації адаптивної логіки на стороні сервера.
Вибір між багатосторінковими та односторінковими додатками має великий вплив на взаємодію з користувачем веб-сайту, пошукову оптимізацію, витрати на розробку та загальний успіх.
Переваги MPA:
1. Пропонують покращену SEO-оптимізацію, швидше завантаження та простіше обслуговування.
2. Складаються з окремих, взаємопов'язаних сторінок.
3. Можуть мати повільніші переходи порівняно з ООПТ.
4. Варто Враховувати цілі, аудиторію, складність та ресурси, вибираючи MPA.
5. Серед успішних MPA є Amazon, eBay, WordPress та Medium.
Багатосторінкові додатки - це вебсайти, які мають багато пов’язаних сторінок. Кожна сторінка - це окремий файл, і коли користувачі відвідують ці вебсайти, їм доводиться завантажувати нові сторінки із сервера.
З іншого боку, односторінкові додатки (SPA) завантажують все одразу. Замість оновлення всієї сторінки, вони оновлюють лише певні розділи веб-сторінки, що робить взаємодію з нею більш плавною та інтерактивною для користувачів. [25]
Server-Side Rendering (SSR) - підхід який використовується для генерації HTML сторінки на стороні сервера, при якому сервер застосунку (Express.js) повністю генерує HTML-код сторінки на основі запиту, даних з бази даних та обраного шаблону, і відправляє його клієнту як готовий, цілісний документ.

Рис. 1.4 Як працює рендеринг на стороні сервера [7]

Переваги SSR:
1. Швидкий Початковий Рендеринг:
Оскільки HTML генерується на сервері, користувач швидше бачить сторінку, що може поліпшити загальне сприйняття продуктивності.
2. SEO Оптимізація:
Пошукові системи краще індексують сторінки, згенеровані на сервері, оскільки вони можуть легко отримати весь контент сторінки.
3. Швидке створення додатку, так як ввесь додаток створюється в одному місці.
4. Оптимізація для Вмісту (MPA):
Оскільки інформаційна система містить значну кількість статичного та динамічного контенту (описи курсів, модулів, завдань), архітектура MPA забезпечує кращу індексованість (SEO), оскільки пошукові системи отримують повністю сформований HTML-код для кожної унікальної сторінки.
Недоліки SSR:
5. Завантаження Сервера:
Генерація HTML на сервері може збільшувати навантаження, особливо при високому трафіку.
6. Обмежена Інтерактивність:
Динамічні функції, які вимагають частих змін контенту, можуть бути менш ефективними при SSR.
7. Маштабованість:
Важкі системи важко масштабувати, так як часто бібліотеки для генерації веб-сторінок обмежені, та не підтримують модульність. Такий підхід важче підтримувати та масштабувати. [8]
У SSR, як правило, пошукові системи можуть індексувати сторінки до їх завантаження у браузері. Це може допомогти зі швидкістю сканування та рейтингом сайту в цілому. Це також полегшує пошуковим системам перегляд контенту, доступного на кожній сторінці, що може допомогти з виявленням контенту та показниками залученості, такими як показник відмов або час перебування на сайті.
Крім того, SSR може бути корисним для сайтів з динамічним контентом (наприклад, блогів) або для сайтів, які хочуть приховати певні елементи, доки їх не буде прокручено вниз під згин статті (наприклад, галерея зображень). [27]
У даній архітектурі Handlebars (HBS) функціонує як основний інструмент SSR.
Handlebars - це проста мова шаблонів . Він використовує шаблон та об'єкт введення для генерації HTML або інших текстових форматів. Шаблони виглядають як звичайний текст із вбудованими виразами Handlebars.
Застосування шаблонних двигунів у JavaScript-розробці має кілька переваг:
8. Розділення логіки та представлення: Використання шаблонних двигунів дозволяє розділити логіку додатку від представлення. Розробники можуть працювати над логікою програми безпосередньо в JavaScript-коді, а дизайнери можуть працювати над виглядом і розміткою в HTML-шаблонах.
9. Перевикористання коду: Шаблонні двигуни дозволяють перевикористовувати код шаблонів. Розробники можуть створювати загальні шаблони і використовувати їх у різних частинах додатку. Це забезпечує більшу модульність і зручність у розробці.
10. Покращена швидкодія: Деякі шаблонні двигуни, такі як Handlebars.js, використовують механізм кешування, що дозволяє зберігати скомпільовані шаблони і використовувати їх повторно. Це покращує швидкодію додатків, особливо у випадках, коли шаблони рендеряться багаторазово.
Механізм роботи Handlebars:
1. Після отримання запиту, Express.js виконує маршрут (app.get(...)), звертається до бази даних (MySQL) і збирає необхідні динамічні дані (наприклад, список курсів, ім'я користувача, оцінки).
2. Ці дані передаються шаблонізатору через метод res.render("template_name.hbs", { dynamic_data }).
3. Handlebars об'єднує статичну структуру шаблону (.hbs файли) із змінними, створюючи фінальний HTML-відповідь.
Важливим технічним рішенням, що підвищує гнучкість системи, є використання об'єкта res.locals. Цей механізм дозволяє зберігати глобальні змінні (наприклад, статус автентифікації користувача isLoggedIn та його права isAdmin), роблячи їх доступними для усіх шаблонів Handlebars без необхідності їх явної передачі в кожному res.render(). Це спрощує навігацію та умовне відображення елементів (наприклад, посилання "Адміністрування" видно лише для користувачів з правами isAdmin).
Таким чином, використання SSR/MPA на базі Node.js/Express та Handlebars забезпечує швидке та надійне відображення контенту, дозволяючи

розробці сфокусуватися на логіці адаптивного навчання на серверній стороні.
Для узагальнення взаємодії обраних технологій та візуалізації структурних зв'язків інформаційної системи було розроблено компонентну схему архітектури (рис. 1.4). Вона демонструє розподіл зон відповідальності між серверною частиною (Backend), яка реалізує бізнес-логіку та маршрутизацію, системою зберігання даних (MySQL) та інтерфейсом користувача, побудованим на шаблонізаторі Handlebars.

Рис. 1.5 Компонентна схема архітектури інформаційної системи «Digital Skills» (розроблено з допомогою Mermaid Live Editor)
Окремим рівнем виділено бізнес-сервіси (Business Services), які відповідають за унікальний функціонал системи: алгоритми аналізу прогалин у навичках (Skill Gap Analysis) та генерацію підтверджуючих документів. Така архітектура забезпечує чітке розмежування логіки, даних та представлення, що відповідає сучасним стандартам розробки веб-застосунків.

РОЗДІЛ 2.АНАЛІЗ
2.1 Концептуальна Модель Даних (ER-модель)
Моделювання даних - процес, що використовується для визначення й аналізу вимог до даних, необхідних для підтримки бізнес-процесів у межах відповідних інформаційних систем в організаціях.
Існують три різних типи моделей даних, що виробляються протягом переходу від вимог до дійсної бази даних для використання в інформаційній систем.
Вимоги до даних спочатку записуються як концептуальна модель даних, яка по суті є набором технологічно залежних специфікацій про дані та використовується для обговорення початкових вимог із зацікавленими сторонами бізнесу. Концептуальна модель потім перетворюється на логічну модель даних, яка документує структури даних, що можуть реалізовуватися в базах даних. Реалізація однієї концептуальної моделі даних може вимагати багатьох логічних моделей даних. Останнім кроком у моделюванні даних є перетворення логічної моделі даних на фізичну модель даних, яка організує дані в таблиці та пояснює деталі доступу, продуктивності та зберігання. Моделювання даних визначає не тільки елементи даних, а й також їх структури та відношення між ними.
Модель «сутність-зв'язок» (ER-модель) модель даних, яка дозволяє описувати концептуальні схеми за допомогою узагальнених конструкцій блоків. ER-модель - це мета-модель даних, тобто засіб опису моделей даних. Модель сутність-зв'язок є результатом систематичного процесу, який описує та визначає деяку предметну область. Вона не визначає сам процес, а лише візуалізує його. Дані представлені у вигляді компонентів (сутностей), які пов'язані між собою певними зв'язками, які виражають залежності і вимоги між ними. [12]
Моделювання предметної області в моделі «сутність - зв'язок» (ER - entity - relationship) базується на використанні графічних діаграм, що включають невелике число різнорідних компонентів. Основними поняттями ER-моделі є сутність, зв'язок і атрибут. [13]

2.2 Визначення ключових сутностей
Сутністю називається реальний або представлений об'єкт, **інформація про який повинна зберігатися в базі даних. У діаграмах ER-моделі сутність представляється у вигляді прямокутника, що містить її ім'я. Кожна сутність має своє унікальне ім'я. При цьому ім'я сутності - це ім'я типу об'єкта, а не деякого конкретного примірника цього типу.** [13]
На основі функціональних вимог системи, що охоплює користувачів, навчальний контент, запис та прогрес, визначено 10 ключових сутностей (табл. 2.1).
Таблиця 2.1
Ключові сутності

No	Назва Таблиці (Сутність)	Призначення	Основні Ключі
1	users	Облік користувачів (студентів/адмінів) та їхніх облікових даних.	id_users (PK)
2	dimensions_ds	Довідник 5-ти основних вимірів цифрових навичок (DigComp).	id_dimensions (PK)
3	courses	Облік навчальних курсів, включаючи їхню складність та тривалість.	id_courses (PK)
4	modules	Структурні блоки навчальних курсів.	id_modules (PK), id_courses (FK)
5	tasks	Конкретні завдання (лекції, вправи) для відстеження прогресу.	id_tasks (PK), id_modules (FK)
6	enrollments	Реєстрація користувача на курс	id_enrollments (PK), id_users (FK), id_courses (FK)
7	users_progress	Фіксація факту виконання завдання	id_progress (PK), id_users (FK), id_tasks (FK)
8	tests	Результати первинного тестування Digital Skills.	test_id (PK), id_users (FK)
9	digital_skills	Довідник детальних навичок, що можуть бути пов'язані з курсами.	id_ds (PK)
10	courses_skills	Проміжна таблиця для зв'язку курсів із деталізованими навичками (M:M).	id_courses (FK), id_ds (FK)

Таким чином, розроблена логічна схема бази даних (рис. 2.1), забезпечує повний облік користувачів (users), навчального контенту (courses, modules, tasks), а також відстеження неформальної освіти: запис на курси (enrollments), прогрес у навчанні (users_progress) та результати первинного оцінювання цифрових навичок (tests).

Рис. 2.1 Концептуальна модель (ER-модель)

Наведена концептуальна модель бази даних відображає архітектуру інформаційної системи "Digital Skills", яка поєднує функції традиційної LMS з інструментами персоналізованої оцінки знань. У центрі моделі лежить взаємозв'язок між ключовими сутностями, що дозволяє системі не просто реєструвати прогрес навчання, а й динамічно формувати індивідуальні освітні траєкторії, базуючись на результатах початкової діагностики, забезпечуючи таким чином цільовий та ефективний розвиток цифрових навичок.

2.3 Детальний Опис Атрибутів Таблиць
Детальний опис атрибутів (полів) є наступним кроком у розробці логічної моделі, який визначає тип, розмір та обмеження для кожного елемента даних у таблицях. Наведений нижче аналіз атрибутів ґрунтується на вимогах до зберігання даних та підтверджується програмною логікою, реалізованою на Node.js/Express, зокрема, операціями CRUD (Create, Read, Update, Delete) та логікою фільтрації, яка вимагає специфічних типів даних (наприклад, INT для courses_difficulty або DATE для users_registration_date).
Перейдемо до опису кожної сутності.

Таблиця dimensions_ds (Виміри Digital Skills) містить ієрархічні категорії або виміри, за якими оцінюються та класифікуються курси (табл. 2.2)
Таблиця 2.2
Атрибути сутності dimensions_ds

Назва Атрибуту	Тип Даних	Розмір / Обмеження	Опис
id_dimensions	INT	11, unsigned, PRIMARY KEY	Унікальний ідентифікатор виміру компетенції (наприклад, 'Information and data literacy').
dimensions_name	VARCHAR	100	Повна назва виміру (наприклад, "Information and data literacy").
dimensions_abb	VARCHAR	10	Скорочена назва виміру (наприклад, "IDL").

Таблиця digital_skills (табл. 2.3) деталізує конкретні навички (наприклад, "Пошук інформації"), які належать до загальних Вимірів (dimensions_ds) (наприклад, "Information and data literacy").
Таблиця 2.3
Атрибути сутності digital_skills

Назва Атрибуту	Тип Даних	Розмір / Обмеження	Опис
id_ds	INT	11, unsigned, PRIMARY KEY	Унікальний ідентифікатор конкретної цифрової навички.
ds_name	VARCHAR	100	Назва детальної навички.
ds_description	TEXT		Детальний опис навички.
ds_abb	VARCHAR	10	Скорочена назва навички.
id_dimesions	INT	11, unsigned, FOREIGN KEY	Ідентифікатор виміру, до якого належить ця навичка.

Таблиця courses (Курси) містить загальну інформацію про всі доступні в системі навчальні курси (табл. 2.4)
Таблиця 2.4
Атрибути сутності courses

Назва Атрибуту	Тип Даних	Розмір / Обмеження	Опис
id_courses	INT	11, unsigned, PRIMARY KEY	Унікальний ідентифікатор курсу.
courses_title	VARCHAR	100	Повна назва курсу.
courses_description	TEXT		Детальний опис курсу.
courses_difficulty	ENUM	1-8	Рівень складності курсу.
courses_duration	INT	11	Орієнтовна тривалість курсу в днях.
id_demensions	INT	11, unsigned, FOREIGN KEY	ID виміру компетенції, до якого належить курс.
courses_img	VARCHAR	255	Шлях до зображення-обкладинки курсу.

Таблиця courses_skills є проміжною і призначена для реалізації зв'язку "Багато-до-Багатьох" (M:M) між сутностями courses та digital_skills, містячи лише два поля, які спільно утворюють складений первинний ключ (Composite Primary Key), що гарантує унікальність кожної пари "Курс-Навичка" (табл. 2.5).
Таблиця 2.5
Атрибути сутності courses_skills

Назва Атрибуту	Тип Даних	Розмір / Обмеження	Опис
id_courses	INT	11, unsigned, FOREIGN KEY	Ідентифікатор курсу.
id_ds	INT	11, unsigned, FOREIGN KEY	Ідентифікатор детальної навички.

Таблиця users (Користувачі) містить основну інформацію про всіх користувачів системи, необхідну для автентифікації та персоналізації (табл. 2.6).
Таблиця 2.6
Атрибути сутності users

Назва Атрибуту	Тип Даних	Розмір / Обмеження	Опис
id_users	INT	11, unsigned, PRIMARY KEY	Унікальний ідентифікатор користувача.
users_name	VARCHAR	100	Повне ім'я користувача.
users_mail	VARCHAR	50	Електронна адреса (використовується як логін).
users_password	VARCHAR	100	Пароль користувача.
users_type	ENUM	admin', 'student'	Рівень прав користувача.
users_registration_date	DATE		Дата реєстрації користувача.

Таблиця user_progress (Прогрес Користувача) тісно пов’язана з таблицею users, та фіксує виконання окремих навчальних елементів (завдань/тасків) користувачами, реалізуючи принцип детального моніторингу прогресу (табл. 2.7).
Таблиця 2.7
Атрибути сутності user_progress

Назва Атрибуту	Тип Даних	Розмір / Обмеження	Опис
id_progress	INT	11, unsigned, PRIMARY KEY	Унікальний ідентифікатор запису про прогрес.
id_users	INT	11, unsigned, FOREIGN KEY	Ідентифікатор користувача.
id_task	INT	11, unsigned, FOREIGN KEY	Ідентифікатор завдання, яке було виконано.
progress_status	ENUM	Not started, In progress, Completed	Поточний статус виконання завдання.
completion_date	DATE	nullable	Дата завершення завдання.

Таблиця tests (Результати тестування) зберігає результати первинного тестування користувачів за п'ятьма вимірами Digital Skills (табл. 2.8).
Таблиця 2.8
Атрибути сутності tests

Назва Атрибуту	Тип Даних	Розмір / Обмеження	Опис

test_id INT 11, unsigned, PRIMARY KEY Унікальний ідентифікатор результату тесту.
id_users INT 11, unsigned, FOREIGN KEY Ідентифікатор користувача, який пройшов тест.
dim_1_score - dim_5_score INT 11 Бали (оцінка) користувача за відповідний вимір компетенції (від 1 до 8).
test_date DATE Дата проходження тесту.

Таблиця enrollments (Записи на Курси) реалізує зв'язок "Багато-до-Багатьох" (M:M) між users та courses і слугує точкою відліку для початку навчання користувача на певному курсі (табл. 2.9).

Таблиця 2.9
Атрибути сутності enrollments
Назва Атрибуту Тип Даних Розмір / Обмеження Опис
id_enrollment INT 11, unsigned, PRIMARY KEY Унікальний ідентифікатор запису на курс.
id_users INT 11, unsigned, FOREIGN KEY Ідентифікатор користувача.
id_courses INT 11, unsigned, FOREIGN KEY Ідентифікатор курсу.
enrollment_date DATE Дата, коли користувач записався на курс.
enrollment_status ENUM Not started, In progress, Complete Статус проходження курсу.

Таблиця modules (Модулі Курсу) описує структуру курсу, розділяючи його на логічні навчальні блоки (табл. 2.10)
Таблиця 2.10
Атрибути сутності modules
Назва Атрибуту Тип Даних Розмір / Обмеження Опис
id_module INT 11, unsigned, PRIMARY KEY Унікальний ідентифікатор модуля.
id_courses INT 11, unsigned, FOREIGN KEY Ідентифікатор курсу, до якого належить модуль.
module_title VARCHAR 100 Назва модуля.
module_order INT 11 Порядковий номер модуля в курсі.

Таблиця tasks (Завдання / Навчальні Елементи) містить деталізований контент кожного навчального елемента, які входять до складу модулів. Це найменша одиниця навчання, прогрес за якою відстежується у таблиці user_progress. (табл. 2.11)
Таблиця 2.11
Атрибути сутності tasks
Назва Атрибуту Тип Даних Розмір / Обмеження Опис
id_task INT 11, unsigned, PRIMARY KEY Унікальний ідентифікатор завдання.
id_module INT 11, unsigned, FOREIGN KEY Ідентифікатор модуля, до якого належить завдання.
task_title VARCHAR 150 Назва або тема завдання.
task_content TEXT Безпосередній навчальний матеріал (текст, посилання на відео/ресурси, інструкції).
task_type ENUM Lesson', 'Quiz', 'Video' Тип навчального елемента.
task_order INT 11 Порядковий номер завдання в межах модуля.

Таким чином, детальний опис атрибутів усіх 10 сутностей демонструє цілісну, гнучку та функціональну структуру бази даних, яка повністю відповідає вимогам розроблюваної інформаційної системи. Встановлені зв'язки «Один-до-Багатьох» (між courses та modules) та «Багато-до-Багатьох» (реалізовані через enrollments і courses_skills) забезпечують цілісність даних та гнучкість системи.
Найважливіше, що ця модель інтегрує два ключові функціональні блоки: традиційне адміністрування навчання (через courses, modules і tasks) та систему персоналізації (через tests, digital_skills та user_progress), дозволяючи автоматично діагностувати прогалини в компетенціях користувача та створювати для нього індивідуальну навчальну траєкторію, що є основною інноваційною метою даної магістерської роботи.
Також, у процесі розробки веб-платформи «Digital Skills» було спроектовано логічну модель бази даних, яка відображає ієрархічну структуру навчального контенту та механізми взаємодії користувачів із системою (рис. 2.2). Центральними сутностями моделі є users (користувачі) та courses (курси), зв'язок між якими реалізовано через проміжну таблицю enrollments. Такий підхід дозволяє реалізувати відношення «багато-до-багатьох» та зберігати специфічні атрибути запису на курс, такі як статус проходження (enrollments_status) та дата реєстрації. Структура навчального матеріалу декомпозована на рівні: курс (courses) містить модулі (modules), які, у свою чергу, складаються з окремих завдань (tasks). Для відстеження успішності реалізовано таблицю user_progress, яка фіксує факт виконання конкретного завдання конкретним студентом.

Рис. 2.2 Логічна модель ІС
Важливим етапом проектування стало забезпечення цілісності даних шляхом нормалізації схеми. Розроблена **модель відповідає вимогам третьої нормальної форми (3** НФ).
Це досягнуто завдяки тому, що **всі неключові** атрибути таблиць залежать **виключно від первинного ключа,** а транзитивні залежності усунуто. Наприклад, інформація про навички (dimensions_ds) та їх зв'язок з курсами винесена в окремі сутності, що унеможливлює дублювання даних.
Використання такої нормалізованої структури забезпечує відсутність надлишковості інформації, підвищує швидкість обробки запитів та гарантує консистентність даних при оновленні або видаленні записів.

РОЗДІЛ 3. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ
Фронтенд - розробка інтерфейсу користувача і функцій, які працюють на стороні клієнта певного ресурсу. Іншими словами - це все, що браузер може читати і виводити перед користувачем на екран або запускати. Він являє **собою публічну сторону додатку, з якою людина може взаємодіяти, встановлюючи контакт напряму. Front end - це процес по створенню даної частини програми. Всього, що бачить користувач, коли заходить на ту чи іншу сторінку. Фронтенд вміщує в себе кілька компонентів:**
1. **HTML (мова розмітки документа, за допомогою якого створюються заголовки, абзаци, списки і** т.д.);
2. **CSS (мова, яка використовується для опису і стилізації документа, завдяки якому задаються потрібні шрифти, кольори,**

розміщуються певним чином блоки та ін.);

3. **JavaScript (мова, що реагує на дії відвідувачів сайту: кліки мишкою, пересування курсору, натискання клавіш). Розробка фронтенда передбачає кропітку роботу, в результаті якої кожна ікона, кнопка або текст стоять на своєму місці, виглядають цілісно, не заважають і не перекривають один одного (так звана** веб-верстка, рис. 3.1). **І наочно побачити код сторінки може будь-хто в своєму браузері. При цьому значення має не тільки зовнішній дизайн ресурсу, але, щоб всі його елементи виконували своє пряме призначення, тобто з їх допомогою можна було здійснити необхідні дії.**

Рис. 3.1 Сторінка входу в інформаційну систему (Додаток Б)

5 **Тому для випуску якісного продукту фронтенд-розробнику доведеться налагодити взаємодію з іншими фахівцями: програмістами, маркетологами, дизайнерами та** іншими.

Бекенд **- це все, що працює на сервері. Виходячи з цього, бекенд розробка - це робота над програмними засобами, спрямованими на реалізацію логіки ресурсу. Ця частина прихована від очей користувача, оскільки відбувається за межами його браузера або конкретно взятого комп'ютера. Розробник сайтів в даному випадку використовує ті ресурси, які є на сервері. При цьому його обов'язки можуть значно варіюватися, залежно від того про який продукт йде мова. Так, фахівець може займатися створенням, інтеграцією баз даних, забезпечувати безпеку ресурсу, налаштовувати технології резервного копіювання або ж відновлення інформації. Бекенд являє собою процес об'єднання користувача з сервером, який неможливо відстежити неозброєним поглядом.**

Тобто, 5 **основні відмінності даних понять полягають в тому, що одне з них (фронтенд) являє собою все, що бачить користувач при роботі з сайтом, а інше (бекенд), навпаки, перебуває поза полем зору людини. Це дві частини одного і того ж проекту, одного цілого, і є кілька варіантів, як вони будуть взаємодіяти один з одним.** [21]

3.1. Програмна реалізація модуля адаптивного асесменту

Основою серверної частини розробленої системи є файл index.js, який виступає головною точкою входу веб-застосунку. У цьому модулі реалізовано ініціалізацію сервера на базі платформи Node.js та фреймворку Express, здійснено налаштування підключення до бази даних MySQL із використанням пулу з'єднань для оптимізації продуктивності, а також сконфігуровано механізми управління сесіями користувачів та обробки HTTP-запитів.

Окрім базових налаштувань, файл відповідає за ініціалізацію шаблонізатора Handlebars із реєстрацією кастомних допоміжних функцій (helpers) для динамічного рендерингу інтерфейсу та підключення окремих модулів маршрутизації, що відповідають за логіку курсів і тестування. Повний лістинг програмного коду файлу index.js наведено у Додатку А.

Критично важливим етапом у реалізації адаптивної інформаційної системи є збір первинних даних про рівень володіння користувачем ключовими цифровими навичками. Цей етап реалізується через маршрут вхідного асесменту (тестування), що дозволяє системі створити початкову Модель Користувача.

Перед початком тестування система виконує перевірку, щоб уникнути повторного проходження, оскільки результати тесту є статичними і формують базову модель.

Маршрут app.get("/test", isLoggedIn, ...) (файл test-routes.js) виконує запит до бази даних: SELECT * FROM tests WHERE id_users = ?

Якщо запис у таблиці tests знайдено (data.length > 0), користувач автоматично перенаправляється на сторінку результатів (/test/result).

Якщо запис відсутній, користувачу відображається шаблон test.hbs із формою для проходження тесту (рис. 3.2).

Рис. 3.2 Сторінка тестування

Вхідний асесмент охоплює п'ять ключових вимірів цифрових навичок відповідно до концептуальної моделі:

18 **1. Information and data literacy; 2. Communication and collaboration; 3. Digital content creation;** 4. Safety;

5. Problem solving.

На сторінці тесту користувач надає одну оцінку (бал від 1 до 8) для кожного виміру.

Форма тестування відправляє дані на сервер через POST-запит, які обробляються маршрутом app.post("/test", ...) у файлі test-routes.js.

У цьому маршруті відбувається збір та первинна валідація відповідей:

1. Дані з тіла запиту (req.body) парсяться (функція, яка перетворює рядок на ціле число, використовуючи задану систему числення) та перетворюються на цілі числа (parseInt(..., 10)).
2. Сервер формує об'єкт testData, який відображає поля таблиці tests:

```
const testData = {
    id_users: req.session.id_users,
    dim_1_score: parseInt(req.body.dim_1_score, 10),
    // ... до dim_5_score
    test_date: getFormattedDate() // Дата проходження тесту
};
```

Таким чином проводиться мінімальна валідація, щоб переконатися, що користувач надав відповіді на всі 5 питань.

Фінальний об'єкт testData записується в таблицю tests бази даних MySQL.

Успішний запис у таблицю tests фіксує початкову модель користувача і слугує точкою відліку для подальшої адаптації. Після цього користувач перенаправляється на сторінку результатів (/test/result) для аналізу та отримання рекомендацій.

Важливий етап адаптації та діагностики прогалин реалізує логіку Моделювання Користувача, перетворюючи сирі бали з таблиці tests на конкретні рекомендації щодо навчання. Логіка повністю зосереджена в маршруті app.get("/test/result", ...) (файл test-routes.js).

Після отримання результатів опитування (dim_1_score до dim_5_score), система ініціює алгоритм діагностики:

1. Пошук Мінімального Балу: Першим кроком є знаходження найнижчого балу серед усіх п'яти вимірів за допомогою функції Math.min(...).
2. Прийняття Рішення: На основі minScore система використовує імпліцитну модель знань (навчення, що відбувається без усвідомлення того, що саме є його предметом [6]):

- Достатній Рівень: Якщо minScore >= 7 (обраний поріг, що вказує на високий рівень у всіх сферах, ≈ 0,9 × 8 ≈ 7), система виводить повідомлення про достатність рівня Digital Skills.

- Виявлення Прогалин: Якщо minScore < 7, система ідентифікує всі виміри, чий бал дорівнює цьому мінімальному значенню. Ці виміри позначаються як найслабші компетенції (weakestDimensionIds).
Виявлені ідентифікатори слабких вимірів використовуються для генерації персоналізованого контенту, який і є ефектом адаптації.
У результаті на сторінці test-result.hbs відображається два варіанти:
1. Позитивний Сценарій: Повідомлення про високий рівень, але з пропозицією ознайомитися з темами, що цікаві (рис. 3.3).
2. Адаптивний Сценарій: Виведення списку конкретних навичок (weakestSkills), які отримали найнижчий бал. Це прямо відповідає меті магістерської роботи: автоматичне формування персоналізованої навчальної траєкторії (рис. 3.4).

Рис. 3.3 Позитивний сценарій тестування

Рис. 3.4 Адаптивний сценарій тесту

Таким чином програмний код реалізує теоретичні засади адаптивної системи навчання (ALS), використовуючи результати тестування для діагностики прогалин і надання цільових рекомендацій.

3.2. Реалізація освітнього середовища та каталогу курсів
Файл courses.js є основним модулем для роботи з навчальним контентом і містить дві групи функцій (рис. 3.5):
1. Функції Доступу та Контролю: Middleware (isLoggedIn, isAdmin) для перевірки прав та функція getFormattedDate для уніфікації формату даних.
2. Центральна Функція Обробки Даних: fetchCoursesAndRender - універсальна функція, яка використовується для відображення як загального каталогу курсів (для студентів), так і сторінки управління (для адміністраторів).

Рис. 3.5 Сторінка з курсами

Також наявні маршрути управління та прогресу: набір маршрутів для керування курсами, модулями, завданнями та маршрути для запису на курс (/enroll), перегляду деталей (/courses/view/:id) та відстеження прогресу (/task/complete/:id_task).
Функція fetchCoursesAndRender виконує складний запит до бази даних, який є основою для відображення адаптивного каталогу. Адаптивність каталогу тут полягає у відображенні статусу запису (enrollments_status) для кожного курсу, що дозволяє користувачеві швидко орієнтуватися, що він вже почав, завершив, або може розпочати.
Запит, який використовується в цій функції, об'єднує дані трьох таблиць для кожного користувача:

```
SELECT
    t1.*,
    t2.dimensions_name AS dimension_name,
    t3.enrollments_status
FROM
    courses t1
LEFT JOIN
    dimensions_ds t2 ON t1.id_demensions = t2.id_dimensions
LEFT JOIN
    enrollments t3 ON t1.id_courses = t3.id_courses AND t3.id_users = ?
```

3. courses t1 та dimensions_ds t2: Використовується LEFT JOIN для отримання основних даних курсу та його категорії (виміру компетенції).
4. enrollments t3: Другий, і ключовий, LEFT JOIN виконується для таблиці enrollments.

Зв'язок відбувається не просто за ID курсу, а за складеною умовою: t1.id_courses = t3.id_courses AND t3.id_users = ?.
Завдяки LEFT JOIN, запит повертає ВСІ курси. Якщо користувач записаний на курс, поле t3.enrollments_status міститиме його статус (Not started, In progress, Complete). Якщо користувач НЕ записаний, це поле буде NULL, що і використовується для відображення кнопки "Записатись" (рис. 3.6).

Рис. 3.6 Статуси курсів
Таким чином, одним ефективним запитом сервер отримує всі дані, необхідні для персоналізованого відображення, мінімізуючи навантаження на базу даних.
Функція fetchCoursesAndRender також динамічно формує частину WHERE SQL-запиту на основі вхідних параметрів, отриманих із форми фільтрації (POST /courses).
Реалізація Фільтрів (рис. 3.7):
1. Фільтр за Вимірами (filterDimensions) та Складністю (filterDifficulties):
Використовується оператор WHERE IN для множинного вибору. Наприклад, якщо користувач обрав ID вимірів 1 та 3, умова буде виглядати як t1.id_demensions IN (1, 3). Це дозволяє швидко відфільтрувати курси за категорією або за рівнем складності.
2. Фільтр за Тривалістю (filterDuration):
Реалізовано за допомогою оператора BETWEEN. Наприклад, для діапазону "8-14 днів" система використовує умову t1.courses_duration BETWEEN 8 AND 14. Це забезпечує точний пошук курсів, що відповідають потребам користувача щодо часових обмежень.
3. Пошук за Назвою:
Для часткового співпадіння використовується оператор LIKE з універсальними символами (%). Умова t1.courses_title LIKE ? (де параметр має вигляд %search_term%) дозволяє знаходити курси, які містять вказаний текст у назві.

Рис. 3.7 Фільтрація контенту
Отже, динамічне формування SQL-запиту гарантує, що каталог адаптується не лише до статусу запису користувача, але й до його поточних

потреб у фільтрації контенту.
Маршрут для відображення деталей курсу виконує важливу логіку авторизації контенту. Він вирішує, чи має користувач право бачити навчальні матеріали (модулі та завдання).
Ключова логіка визначена на основі двох змінних (табл 3.1).

Таблиця 3.1
Дві основні змінні відображення деталей курсу
Змінна Умова Значення Джерело
isEnrolled courseData.id_enrollments != null Користувач записаний на курс (має запис у таблиці enrollments). Запит sqlCourse
isAdmin req.session.prava === 'admin' Користувач є адміністратором. Сесія

На основі цих змінних приймається рішення щодо відображення елементів:
4. Чи показувати Модулі? (showModuleList). Модулі та завдання курсу будуть відображені, якщо користувач записаний АБО він є адміністратором.
5. Чи показувати Кнопку "Записатись"? Кнопка "Записатись" відображається лише в тому випадку, якщо користувач НЕ записаний І він НЕ є адміністратором (адміністратору не потрібно записуватися).

Рис. 3.8 Вікно курсу для ще не записаного користувача

Рис. 3.9 Вікно курсу для користувача, що записався на курс

Якщо користувач не записаний і не є адміністратором, система повертає сторінку курсу, показуючи лише опис та кнопку "Записатись" (рис. 3.8), приховуючи список модулів. Якщо ж доступ надано, система виконує додаткові запити для завантаження структури курсу (модулі, завдання) та прогресу користувача у ньому (рис. 3.9).

3.3. Механізми відстеження прогресу та сертифікації
Модуль courses.js реалізує функціонал відстеження прогресу, який є невід'ємною частиною моделювання користувача після тестування.
1. Обробка завершення завдання (POST /task/complete/:id_task)
Коли користувач натискає "Позначити як виконане" на сторінці завдання:
1. Фіксація прогресу: Створюється запис у таблиці users_progress:
2. Оновлення статусу курсу: Система автоматично перевіряє та оновлює статус курсу в таблиці enrollments:
1. Перехід у In progress: Якщо статус курсу був 'Not started', він примусово оновлюється до 'In progress' (рис. 3.10).
2. Перехід у Completed: Це найскладніший етап, який виконується спеціальною допоміжною функцією (рис. 3.11).

Рис. 3.10 Статус “In progress”

Рис. 3.11 Статус “Completed”

2. Допоміжна функція checkIfCourseCompleted
Ця функція виконує перевірку завершення курсу шляхом порівняння двох показників:
1. Загальна Кількість Завдань: Рахується загальна кількість завдань у курсі.
2. Кількість Виконаних Завдань: Рахується кількість виконаних завдань цим користувачем у цьому курсі:
Якщо totalTasks > 0 та totalTasks === completedTasks, функція повертає true, і статус курсу в таблиці enrollments оновлюється на 'Completed'.
Цей механізм гарантує, що прогрес користувача відстежується точно, а статус курсу (від 'Not started' до 'Completed') завжди актуальний, що підтверджує функціонал системи як повноцінної LMS з елементами ALS.
Для користувачів, які досягли статусу 'Completed', було реалізовано маршрут завантаження сертифікату, що підтверджує успішне проходження курсу.
GET /download/certificate/:id_enrollment:
1. Валідація: Перевіряє, чи є запис про проходження курсу, і чи має він статус 'Completed'.
2. Збір даних: Отримує ім'я користувача (users_name), назву курсу (courses_title) та дату завершення.
3. Генерація DOCX: Використовує бібліотеки docxtemplater та PizZip для завантаження шаблону (certificate_template.docx) та заміни плейсхолдерів (наприклад, {{users_name}}, {{course_title}}) фактичними даними.
4. Завантаження файлу: Встановлює спеціальний HTTP-заголовок Content-Disposition з кодуванням UTF-8 (RFC 5987) для коректної підтримки кириличних символів в імені файлу, після чого відправляє згенерований outputBuffer назад клієнту.
Впровадження механізмів відстеження прогресу та автоматичної сертифікації дозволило перетворити статичний каталог курсів на інтерактивне навчальне середовище із чітким зворотним зв'язком. Реалізована програмна логіка забезпечує динамічну зміну статусів навчання («Not started», «In progress», «Completed») у режимі реального часу, базуючись на фіксації виконаних завдань у реляційній базі даних. Застосування розробленого алгоритму перевірки повноти проходження курсу (checkIfCourseCompleted) гарантує точність визначення фінального результату, а інтеграція бібліотеки docxtemplater для генерації іменних сертифікатів надає користувачам документальне підтвердження здобутих цифрових компетенцій, завершуючи повний цикл освітнього процесу.

3.4 Розробка підсистеми адміністрування (CRUD)
Робота з інформацією зводиться до чотирьох основних операцій: створення, читання, оновлення та видалення. Ці ж операції можна виконати з даними в базі даних або іншій системі управління інформацією.
CRUD - це скорочення від англійських слів, що позначають ці чотири операції:
1. Create (створювати);
2. Read (читати);

3. Update (оновлювати);
4. Delete (видаляти).
Операції CRUD становлять основу для багатьох застосунків та систем управління інформацією, даючи змогу користувачам взаємодіяти з даними та змінювати їх відповідно до потреб. [4]
Адміністративний модуль був розроблений для забезпечення повного контролю над контентом платформи: курсами, модулями та завданнями. Це реалізовано за принципом CRUD? що дозволяє адміністратору (користувач з prava === 'admin') створювати, переглядати, змінювати та видаляти відповідні сутності.

Усі адміністративні маршрути захищені за допомогою спеціалізованого middleware (isAdmin), що гарантує доступ лише користувачам із правами 'admin' (рис. 3.12).

Рис. 3.12 Кнопки для управління контентом для користувача з правом доступу Admin
"Middleware" - це функція, яка працює з кожним запитом перед його обробкою будь-якою конкретною операцією шляху (path operation), а також з кожною відповіддю перед її поверненням.
Він:
1. отримує кожен запит, що надходить до Вашого застосунку.
2. може виконати певні дії із цим запитом або запустити необхідний код.
3. далі передає запит для обробки основним застосунком (операцією шляху).
4. отримує відповідь, сформовану застосунком (операцією шляху).
5. може змінити цю відповідь або виконати додатковий код.
6. повертає відповідь клієнту.
Функція isAdmin (у courses.js):

```
const isAdmin = (req, res, next) =&gt; {
    if (req.session.prava === 'admin') {
        next();
    } else {
        res.redirect('/courses');
    }
};
```

Це є ключовим елементом безпеки, який запобігає несанкціонованим змінам даних.

Для управління курсами було створено спеціальну сторінку /courses/manage (рис. 3.13), яка повторно використовує функціонал фільтрації та пошуку, реалізований для звичайного каталогу.

Рис. 3.13 Сторінка для керуванням курсами

- Створення нового курсу включає збір даних про назву, опис, складність, тривалість, вимір (Dimension) та посилання на зображення (рис. 3.14).

Рис. 3.14 Сторінка створення нового курсу
GET /courses/create: Відображає форму, попередньо завантажуючи список доступних вимірів (dimensions_ds) для вибору.
POST /courses/create: Обробляє форму.
Ключова логіка: Оскільки у структурі БД поле id_courses не є AUTO_INCREMENT, перед вставкою нового запису система визначає максимальний існуючий id_courses і присвоює новому курсу значення MAX(id) + 1.
- Редагування Курсу (Update) (рис. 3.15)
Рис. 3.15 Сторінка редагування курсу
GET /courses/edit/:id_courses: Завантажує дані конкретного курсу та список усіх вимірів, щоб користувач міг бачити поточні значення та вносити зміни.
POST /courses/edit: Виконує запит UPDATE до таблиці courses на основі отриманого ID курсу.

- Видалення Курсу (Delete)
POST /courses/delete/:id_courses: Виконує запит DELETE.
Важливий аспект: Завдяки конфігурації бази даних з використанням обмежень ON DELETE CASCADE на зовнішніх ключах, видалення курсу автоматично видаляє всі пов'язані модулі, завдання та записи користувачів (enrollments, users_progress), забезпечуючи цілісність даних.

Управління модулями та завданнями (рис. 3.16)

Рис. 3.16 Доступ користувача з правами Admin до редагування модулями та завданнями

Керування структурою курсу (модулями та завданнями) відбувається через детальний перегляд курсу (/courses/view/:id), до якого адміністратор має повний доступ (табл. 3.2, 3.3).
Таблиця 3.2
Управління модулями

CRUD Операція	Маршрут	Опис
Створити	POST /modules/create	Додає новий модуль до вказаного курсу з назвою та порядковим номером (modules_order).
Редагувати	POST /modules/edit	Оновлює назву модуля.
Видалити	POST /modules/delete/:id_module	Видаляє модуль. Завдяки ON DELETE CASCADE видаляються також і всі пов'язані завдання

(tasks).

Таблиця 3.3
Управління завданнями
CRUD Операція Маршрут Опис
Створити POST /tasks/create Додає нове завдання до вказаного модуля з назвою, описом та типом (tasks_type).
Редагувати POST /tasks/edit Оновлює деталі завдання (назву, опис, тип).
Видалити POST /tasks/delete/:id_task Видаляє завдання. Записи про прогрес користувачів (users_progress) також видаляються автоматично через ON DELETE CASCADE.

Реалізація адміністративного модуля, побудованого на принципах CRUD-функціоналу, забезпечила повну керованість контентом системи "Digital Skills". Завдяки розробці спеціалізованих маршрутів та застосуванню захисного middleware (isAdmin), було гарантовано, що лише авторизовані адміністратори можуть здійснювати операції створення, редагування та видалення курсів, модулів і завдань. Це є критично важливим для підтримки актуальності навчального контенту та безпеки даних платформи.
Крім того, інтеграція адміністративних функцій з механізмом відстеження прогресу та реалізація допоміжних функцій, як-от checkIfCourseCompleted та генерація сертифікатів з використанням docxtemplater, завершує формування повноцінного освітнього циклу.
Таким чином, розроблений адміністративний модуль створює надійну основу для масштабування та динамічного розвитку навчальної платформи, перетворюючи її на повноцінну, керовану систему електронного навчання.

ВИСНОВОК
У магістерській кваліфікаційній роботі вирішено актуальну науково-прикладну задачу підвищення ефективності формування цифрових компетенцій шляхом проектування та програмної реалізації адаптивної інформаційної системи «Digital Skills».
В умовах стрімкої цифрової трансформації бізнесу та суспільства, традиційні підходи до електронного навчання, реалізовані в більшості класичних LMS (Learning Management Systems), демонструють недостатню гнучкість. Вони пропонують уніфікований контент без урахування індивідуальних прогалин у знаннях користувачів, що призводить до неефективного витрачання часу та зниження мотивації. Розроблена система вирішує цю проблему, інтегруючи механізми діагностичного асесменту та автоматичної побудови персоналізованої навчальної траєкторії.
За результатами проведеного дослідження та розробки можна зробити наступні висновки:
1. Проаналізовано предметну область та обмеження існуючих рішень. У ході дослідження було визначено, що цифрові навички є критичним фактором конкурентоспроможності на сучасному ринку праці. Аналіз провідних систем управління навчанням (Moodle, Canvas, Blackboard Learn) показав, що попри їхній потужний адміністративний функціонал, вони здебільшого реалізують пасивну модель навчання. Відсутність вбудованих інструментів для вхідного діагностування прогалин у знаннях (skill gaps) змушує користувачів витрачати час на вивчення вже відомого матеріалу або зіштовхуватися з надмірною складністю. На основі аналізу теоретичних засад адаптивних систем (ALS) було обґрунтовано необхідність створення системи, яка базується на трьохкомпонентній моделі: збір даних про користувача (асесмент), моделювання його профілю та адаптація контенту.
2. Обґрунтовано та використано сучасний технологічний стек. Для реалізації системи було обрано архітектуру на базі платформи Node.js та фреймворку Express.js. Вибір Node.js обумовлений його асинхронною, подієво-орієнтованою моделлю, що забезпечує високу продуктивність при обробці I/O-операцій, необхідних для роботи веб-платформи. Використання реляційної системи управління базами даних MySQL дозволило забезпечити сувору структурованість та цілісність даних, що є критичним для збереження складних зв'язків між користувачами, результатами тестування, курсами та прогресом навчання. Реалізація інтерфейсу за принципом Server-Side Rendering (SSR) з використанням шаблонізатора Handlebars забезпечила швидке відображення контенту, розділення логіки та представлення, а також оптимізацію для пошукових систем.
3. Спроектовано комплексну архітектуру даних. Розроблена концептуальна та логічна структура бази даних, що складається з 10 взаємопов'язаних сутностей, стала фундаментом для адаптивної логіки системи. Ключовою особливістю моделі є реалізація зв'язків між результатами тестування (tests), п'ятьма вимірами цифрових навичок (dimensions_ds) та навчальним контентом (courses, modules, tasks). Це дозволило програмно реалізувати механізм, де результати асесменту безпосередньо впливають на рекомендації курсів. Забезпечено нормалізацію бази даних та налаштовано каскадне видалення (ON DELETE CASCADE), що гарантує відсутність надлишковості та цілісність даних при адміністративних операціях видалення курсів чи користувачів.
4. Програмно реалізовано модуль адаптивного асесменту та моделювання користувача. Одним із головних науково-практичних результатів роботи є розробка алгоритмічної моделі вхідного тестування. Система оцінює користувача за п'ятьма ключовими вимірами цифрової грамотності (Information and data literacy, Communication, Content creation, Safety, Problem solving) . Реалізовано програмну логіку (файл test-routes.js), яка автоматично аналізує результати тесту, визначає мінімальний показник (найслабшу ланку) серед 5 вимірів та формує індивідуальну модель користувача. На основі цієї моделі система динамічно генерує рекомендації, пропонуючи користувачеві саме ті курси, які спрямовані на усунення виявлених прогалин. Це підтверджує адаптивний характер розробки та відрізняє її від лінійних навчальних курсів.
5. Розроблено функціонал адаптивного каталогу та інтерактивного навчання. Реалізовано динамічний каталог курсів, який адаптується до поточних фільтрів. Використання складних SQL-запитів із LEFT JOIN у модулі courses.js дозволило в реальному часі відображати статус кожного курсу («Not started», «In progress», «Completed») та керувати доступом до навчальних матеріалів. Впроваджено систему динамічної фільтрації контенту за кількома параметрами (категорія навички, складність, тривалість), що значно покращує навігацію. Розроблено логіку автоматичного відстеження прогресу: система фіксує виконання окремих завдань (лекцій, тестів) та, за допомогою спеціально розроблених алгоритмів перевірки повноти курсу (checkIfCourseCompleted), автоматично змінює статус навчання, забезпечуючи інтерактивний зворотний зв'язок.
6. Створено повноцінну підсистему адміністрування та сертифікації. Для забезпечення повного життєвого циклу системи розроблено адміністративний модуль із захищеним доступом (реалізовано через Middleware isAdmin). Реалізовано повний набір CRUD-операцій (Create, Read, Update, Delete) для управління ієрархічною структурою контенту: курсами, модулями та завданнями. Це дозволяє адміністраторам гнучко налаштовувати та масштабувати навчальну програму без втручання в програмний код. В якості фінального етапу навчання інтегровано модуль генерації документів: система автоматично формує та видає персоналізовані сертифікати у форматі .docx (з використанням бібліотеки docxtemplater) для користувачів, які успішно завершили навчання, що додає елемент гейміфікації та офіційного підтвердження здобутих навичок.
Практичне значення роботи полягає у створенні готового до впровадження програмного продукту - веб-орієнтованої інформаційної системи «Digital Skills». Система є повністю функціональною, має інтуїтивно зрозумілий інтерфейс та вирішує конкретну бізнес-задачу: оптимізацію часу навчання персоналу шляхом фокусування на реальних потребах у розвитку навичок, діагностованих під час вхідного асесменту.

Перспективи подальшого розвитку системи вбачаються у впровадженні більш складних алгоритмів адаптації на основі машинного навчання (Machine Learning) для аналізу поведінки користувачів під час проходження завдань, розширенні типів тестових завдань та інтеграції з зовнішніми освітніми API для збагачення контенту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Що таке цифрові навички і які з них потрібно освоїти вже зараз? URL: https://budni.robota.ua/career/shho-take-tsifrovi-navichki-i-yaki-z-nih-potribno-osvoyiti-vzhe-zaraz (дата звернення: 10.11.2025).
2. Цифрові навички: як і для чого їх розвивати? URL: http://lviv.dcz.gov.ua/novyna/cyfrovi-navychky-yak-i-dlya-chogo-yih-rozvyvaty (дата звернення: 10.11.2025).
3. Цифрові навички відкривають нові можливості для українців. URL: https://business.diia.gov.ua/history-of-success/tsyfrovi-navychky-vidkryvaiut-novi-mozhlyvosti-dlia-ukraintsiv (дата звернення: 10.11.2025).
4. Що таке CRUD простими словами: функції, переваги та приклади. URL: https://highload.tech/uk/shho-take-crud-prostimi-slovami-funktsiyi-perevagi-ta-prikladi/ (дата звернення: 10.11.2025).
5. ТОП 10 LMS: огляд найкращих систем управління навчанням. URL: https://7sky.ua/blog/top-10-lms-an-overview-of-the-best-learning-management-systems/ (дата звернення: 10.11.2025).
6. Імпліцитне навчання. URL: https://uk.wikipedia.org/wiki/%D0%86%D0%BC%D0%BF%D0%BB%D1%96%D1%86%D0%B8%D1%82%D0%BD%D0%B5_%D0%BD%D0%B0%D0%B2%D1%87%D0%B5%D0%BD%D0%BD%D1%8F (дата звернення: 10.11.2025).
7. What is Server-Side Rendering: Definition, Benefits and Risks. URL: https://solutionshub.epam.com/blog/post/what-is-server-side-rendering (дата звернення: 10.11.2025).
8. SSR(Server side Rendering)-патерн та як це працює на Spring Boot. URL: https://abitap.com/3-5-ssrserver-side-rendering-patern-ta-yak-cze-praczyuye-na-spring-boot/ (дата звернення: 10.11.2025).
9. Adaptive e-learning systems through learning styles: A review of the literature. URL: https://www.researchgate.net/publication/355317942_Adaptive_e-learning_systems_through_learning_styles_A_review_of_the_literature (дата звернення: 10.11.2025).
10. 1 Introduction Adaptive Educational Hypermedia : From Ideas to Real Systems. URL: https://www.semanticscholar.org/paper/1-Introduction-Adaptive-Educational-Hypermedia-%3A-to-Beaumont-Brusilovsky/71bddb94187569230039bcd36549569bdc90c440#citing-papers (дата звернення: 10.11.2025).
11. Adaptive Hypermedia for Education and Training. URL: https://www.researchgate.net/publication/233898436_Adaptive_Hypermedia_for_Education_and_Training (дата звернення: 10.11.2025).
12. Моделювання даних: створення моделі даних для інформаційної системи; концептуальна, логічна, фізична моделі даних; er-модель; нотації er-моделей. URL: https://studfile.net/preview/21632686/page:3/ (дата звернення: 10.11.2025).
13. Проектування баз даних. Поняття сутності, атрибута, ключа, зв'язку. Модель «сутність-зв'язок» предметної області. URL: https://vseosvita.ua/lesson/proektuvannia-baz-danykh-poniattia-sutnosti-atrybuta-kliucha-zviazku-model-sutnist-zviazok-predmetnoi-oblasti-586464.html (дата звернення: 10.11.2025).
14. 56% of EU people have basic digital skills. URL: https://ec.europa.eu/eurostat/web/products-eurostat-news/-/ddn-20231215-3 (дата звернення: 10.11.2025).
15. Sabine Graf, Tzu-Chien Liu , Kinshuk , Nian-Shing Chen , Stephen J.H. Yang Computers in Human Behavior 25 (2009) 1280-1289. C-1. URL: https://sgraf.athabascau.ca/publications/graf_etal_CiHB09.pdf (дата звернення: 12.11.2025).
16. Fasihuddin, H., Skinner, G., & Athauda, R. (2016). Using learning styles as a basis for creating adaptive open learning environments: An evaluation. International Journal of Learning Technology, 11(3), 198-217. URL: https://www.inderscienceonline.com/doi/abs/10.1504/IJLT.2016.079034 (дата звернення: 12.11.2025).
17. P. Brusilovsky & Peylo, Adaptive and Intelligent Web-based Educational Systems, 2003. URL: https://www.researchgate.net/publication/32229453_Adaptive_and_Intelligent_Web-based_Educational_Systems (дата звернення: 12.11.2025).
18. Al-Azawei & Badii, State of The Art of Learning Styles-Based Adaptive Educational Hypermedia Systems (Ls-Baehss), 2014. URL: https://www.researchgate.net/publication/274175079_State_of_The_Art_of_Learning_Styles-Based_Adaptive_Educational_Hypermedia_Systems_Ls-Baehss (дата звернення: 12.11.2025).
19. Mulwa, 2010; Papadimitriou, 2012; Popescu, WELSA: An Intelligent and Adaptive Web-Based Educational System, 2009. URL: https://www.researchgate.net/publication/225681390_WELSA_An_Intelligent_and_Adaptive_Web-Based_Educational_System (дата звернення: 12.11.2025).
20. Abueloun & Naser, 2017; Akkila & Naser, 2017; Aleven et al., 2016 Teaching the right letter pronunciation in reciting the holy Quran using intelligent tutoring system. URL: https://www.researchgate.net/publication/314229916_Teaching_the_right_letter_pronunciation_in_reciting_the_holy_Quran_using_intelligent_tutoring_system (дата звернення: 12.11.2025).
21. Frontend і Backend розробка. URL: https://webtune.com.ua/statti/web-rozrobka/frontend-i-backend-rozrobka/ (дата звернення: 15.11.2025).
22. MEAN vs. MERN Stack Explained. URL: https://www.oracle.com/ua/database/mean-mern/ (дата звернення: 15.11.2025).
23. Node.js. URL: https://uk.wikipedia.org/wiki/Node.js (дата звернення: 15.11.2025).
24. Основи роботи з фреймворком Express.js. URL: https://foxminded.ua/express-js/ (дата звернення: 15.11.2025).
25. What is a Multi Page Application (MPA) Exactly? URL: https://medium.com/@julianneagu/multi-page-application-mpa-a-good-business-fit-36029c7be9f0 (дата звернення: 15.11.2025).

ДОДАТКИ

Додаток А

Код index.js

```
const mysql = require("mysql2");
const express = require("express");
const bodyParser = require("body-parser"); const hbs = require("hbs");
const expressHbs = require("express-handlebars");
const path = require("path");
```

```
const app = express(); const urlencodedParser = bodyParser.urlencoded({extended: false});
app.use( express.static('img'))
app.use(express.static('css'));
const zagolovok = "Digital Skills"

function createStore() {
var pool = mysql.createPool({
    connectionLimit: 5,
    host: "localhost",
    user: "root",
    database: "digitalskills",
    password: ""
});
global.pool = pool
}

var cookieParser = require("cookie-parser") var session = require( "express-session")

app.use(bodyParser.json())

var store = createStore()

app.use(cookieParser())
app.use( session({
    store:store,
    resave: false,
    saveUninitialized: true,
    secret: 'supersecret'
}))

// Цей "посередник" (middleware) виконується для КОЖНОГО запиту
// Він робить змінні з 'res.locals' доступними у ВСІХ шаблонах .hbs
app.use(function(req, res, next) {
    res.locals.users_name = req.session.users_name; // Передаємо ім'я
    res.locals.isAdmin = req.session.prava === 'admin'; // Можемо навіть isAdmin сюди перенести!
    next();
});

global.zagolovok = zagolovok;
global.pool = pool;

app.engine("hbs", expressHbs.engine({
    layoutsDir: "views",
    defaultLayout: "index",
    extname: "hbs",
    partialsDir: "views",

    helpers: {
        contains: function (arr, item, options) {
            // Універсальний Helper для вбудованого та блокового використання
            const arrToUse = Array.isArray(arr) ? arr : [arr];
            const itemString = item &amp;amp;&amp;amp; item.toString();

            const result = arrToUse.some(el =&gt; el.toString() === itemString);

            // Якщо використовується як блоковий Helper (наприклад, {{#if (contains ...)}})
            if (options.fn &amp;amp;&amp;amp; options.inverse) {
                return result ? options.fn(this) : options.inverse(this);
            }
            // Якщо використовується як простий Helper (наприклад, {{contains ...}})
            return result;
        },

        eq: function (a, b, options) {
            // Визначаємо, чи Helper викликаний як блоковий (має options.fn)
            if (options.fn) {
                if (a == b) {
```

```
                return options.fn(this);
            } else {
                return options.inverse(this);
            }
        }
        // Якщо викликаний як вбудований (inline)
        return a == b;
    },

    // УНІВЕРСАЛЬНИЙ HELPER ДЛЯ ПОРІВНЯННЯ (lt)
    lt: function (a, b, options) {
        if (options.fn) {
            if (a &lt; b) {
                return options.fn(this);
            } else {
                return options.inverse(this);
            }
        }
        return a &lt; b;
    },

    // УНІВЕРСАЛЬНИЙ HELPER ДЛЯ ПОРІВНЯННЯ (gte)
    gte: function (a, b, options) {
        if (options.fn) {
            if (a &gt;= b) {
                return options.fn(this);
            } else {
                return options.inverse(this);
            }
        }
        return a &gt;= b;
    }
  }
}));
app.set("view engine", "hbs");

const coursesRouter = require('./courses'); // Підключаємо наш новий модуль
coursesRouter(app); // Викликаємо його, передаючи app

const testRouter = require('./test-routes'); // Підключаємо модуль тестування
testRouter(app); // Викликаємо його

app.get("/", function(req, res) {
    res.render("start.hbs", { // start.hbs імпортується в index.hbs в параметр {{{body}}}
        zagolovok: zagolovok,
        isHomePage: true
    });
});

const PORT = process.env.PORT || 3000;  // якщо порт є то його, інакше ставимо 3000
app.listen(PORT, function(){
    console.log("Server");
});

app.post("/login", function(req, res)  {
    var foundUser
    // Отримуємо дані з клієнтського запиту
    var users_mail = req.body.username // Поле "username" з AJAX-запиту - це e-mail
    var users_password = req.body.password // Поле "password" з AJAX-запиту - це пароль

    pool.query("SELECT " +
        "id_users,\n" +
        "users_mail,\n" +
        "users_password,\n" +
        "users_type,\n" +
        "users_name\n" + // &lt;-- Ось ця зміна
        "FROM\n" +
```

```
        "users\n" +
        "WHERE\n" +
        "users_mail = ? AND\n" +
        "users_password = ?",
        [users_mail, users_password], function(err, data) { // Використовуємо нові змінні

        if(err) return console.log(err);

        if (data.length &gt; 0) { // Якщо користувач існує
            foundUser = data[0].users_mail // &lt;--- Змінено
            req.session.prava = data[0].users_type // Зберігаємо права в сесії
            req.session.users_name = data[0].users_name; // &lt;-- Ось ця зміна
            req.session.id_users = data[0].id_users;
        }
        console.log("Права користувача:", req.session.prava); // Читаємо з сесії
        console.log("Ім'я користувача:", req.session.users_name); // Додамо лог для перевірки

        if (foundUser != undefined) {
            req.session.username = foundUser
            console.log("Успішний вхід:", req.session.username);
            res.send(req.session.username)
            // return res.redirect("/konferentsiyi") // Можете змінити URL-адресу перенаправлення тут
        } else {
            console.log('Помилка входу: користувача не знайдено');
            res.status(401).send("Невірний логін або пароль"); // Надсилаємо статус помилки
        }
    });
});

app.get("/logout", function(req,res){
    req.session.username = '';
    req.session.prava = ''; // Добра практика
    req.session.users_name = ''; //  ОЧИЩУЄМО ІМ'Я
    req.session.id_users = ''; // ОЧИЩУЄМО ID
    res.redirect("/");
});

// A. Маршрут для відображення сторінки реєстрації
app.get("/register", function(req, res) {
    res.render("register.hbs", {
        zagolovok: zagolovok,
        isHomePage: true
    });
});

// B. Маршрут для обробки даних з форми реєстрації
// Обов'язково 'urlencodedParser'
app.post("/register", urlencodedParser, function (req, res) {

    const { users_name, users_mail, users_password, users_password_repeat } = req.body;

    if (users_password !== users_password_repeat) {
        return res.render("register.hbs", {
            zagolovok: zagolovok,
            error: "Паролі не співпадають!"
        });
    }

    pool.query("SELECT * FROM users WHERE users_mail = ?", [users_mail], function(err, data) {
        if (err) {
            console.error(err);
            return res.render("register.hbs", { zagolovok: zagolovok, error: "Помилка бази даних." });
        }

        if (data.length &gt; 0) {
            return res.render("register.hbs", {
                zagolovok: zagolovok,
```

```
            error: "Користувач з таким e-mail вже існує."
        });
    }

    // 1. Створюємо сьогоднішню дату
    const today = new Date();

    // 2. Форматуємо її у 'YYYY-MM-DD'
    const yyyy = today.getFullYear();
    const mm = String(today.getMonth() + 1).padStart(2, '0'); // Місяці 0-11, тому +1
    const dd = String(today.getDate()).padStart(2, '0');
    const formattedDate = `${yyyy}-${mm}-${dd}`;

    // 3. Створюємо об'єкт користувача з ВІДФОРМАТОВАНОЮ датою
    const newUser = {
        users_name: users_name,
        users_mail: users_mail,
        users_password: users_password,
        users_type: 'student',
        users_registration_date: formattedDate // &lt;-- Використовуємо рядок, а не new Date()
    };

    pool.query("INSERT INTO users SET ?", newUser, function(err, result) {
        if (err) {
            console.error("Помилка при реєстрації:", err);
            return res.render("register.hbs", { zagolovok: zagolovok, error: "Помилка при створенні акаунту." });
        }

        res.redirect("/");
    });
  });
});
Додаток Б
Код start.hbs
&lt;div class="container-fluid" style="min-height: 50vh"&gt;
&lt;script&gt;
  window.onload = function () {
    var btn = document.getElementById('btn')
    var inputs = document.getElementsByTagName('input')

    btn.addEventListener('click', function () {
      var xhr = new XMLHttpRequest()

      xhr.open('POST', '/login') //переходимо по маршруту /login
      var userData = {
        username: inputs[0].value, //передаємо ім'я користувача
        password: inputs[1].value
      }

      xhr.onload = function () {
        location.href = '/courses' //після успішної авторизації переходимо на конференції
      }

      xhr.setRequestHeader('Content-Type', 'application/json')
      xhr.send(JSON.stringify( userData))
    })
  }
&lt;/script&gt;
  &lt;div class="row"&gt;
    &lt;div class="col-xs-12 col-sm-12 col-md-6 col-lg-4 text-center mx-auto auth-card"&gt;
      &lt; h4&gt; Log in&lt;/h4&gt;
      &lt;form name="login"&gt;
        &lt;div class="form-group"&gt;
          &lt;label for="name_user"&gt; Mail:&lt;/label&gt;
          &lt; input type= "email" class="form-control" name="name_user" placeholder="Write yours login"&gt;
        &lt;/div&gt;
        &lt; div class="form-group"&gt;
```

```
                &lt;label for="parol"&gt; Password:&lt;/ label&gt; &lt;input type= "password" class="form-control" name="parol" placeholder="Write your password"&gt;
            &lt;/ div&gt; &lt;input type= "button" class="btn btn-primary mx-2 my-2" value="Увійти" id="btn"/&gt;

            &lt;p class="mt-3"&gt;
                Немає акаунту? &lt;a href="/register"&gt;Зареєструватись&lt;/a&gt;
            &lt;/p&gt;
        &lt;/form&gt;
    &lt;/div&gt;
  &lt;/div&gt;
&lt;/div&gt;
```