Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Система оцінювання навчальних параметрів студента при дистанційному навчанні»

08-54.БКР.032.00.000 ПЗ

Виконав: студент 4 курсу, групи 2КІ-21б
спеціальності 123 — «Комп'ютерна інженерія»
освітня програма — «Комп'ютерна інженерія»
Соловйов Д.Ю. _____

Керівник к.т.н., доц. каф. ОТ
Снігур А.В _____

Рецензент: д.т.н., професор кафедри МБІС
Яремчук Ю.Є. _____

Допущено до захисту
Завідувач кафедри ОТ
д.т.н., проф. Азаров О.Д.
«20» червня 2025 р.

м. Вінниця — ВНТУ 2025

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра обчислювальної техніки

Освітній рівень — бакалавр

Напрям підготовки — 123 Комп'ютерна інженерія

ЗАТВЕРДЖУЮ

Завідувач кафедри

обчислювальної техніки

д.т.н., проф. Азаров О. Д.

«21» березня 2025 р.

З А В Д А Н Н Я

НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Соловйову Дмитру Юрійовичу

1 Тема роботи: «Система оцінювання навчальних параметрів студента при дистанційному навчанні», керівник роботи Снігур Анатолій Васильович , к.т.н., доц. каф. ОТ, затверджено наказом вищого навчального закладу від «20» березня 2025 року №97.

2 Строк подання студентом роботи 13.05.2025

3 Вихідні дані до роботи: методи розробки web-застосунків, навчальні матеріали на тему створення web-застосунків, технічна документація стеку технологій .Net Core та мови програмування C#, системи-аналоги.

4 Зміст розрахунково-пояснювальної записки: вступ; аналітичний огляд комп'ютерних систем та існуючих підходів із оцінювання результатів навчання; моделювання та проєктування підсистеми оцінювання якості навчання; практична реалізація підсистеми оцінювання якості навчання; економічна частина.

5 Ілюстративна частина — алгоритм роботи інтегрованої підсистеми оцінювання якості навчання

6 Консультанти розділів роботи наведені у таблиці 1.

Таблиця 1 — Консультанти роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Спеціальна частина	Снігур А. В. к.т.н., доц. каф. ОТ	21.03.2025	13.05.2025
Нормоконтроль	ас. каф. ОТ Швець С. І.	14.05.2025	30.05.2025

7 Дата видачі завдання 21.03.2025 р.

8 Календарний план розділів роботи наведено у таблиці 2.

Таблиця 2 — Календарний план БКР

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Постановка задачі роботи	21.03.25	Виконано
2	Аналіз аналогів	23.03.25	Виконано
3	Аналіз сучасних підходів до інформаційних систем	27.03.25	Виконано
4	Аналіз поняття інформаційна система	30.03.25	Виконано
5	Аналіз принципів роботи інформаційних систем	05.04.25	Виконано
6	Проєктування інформаційних систем	09.04.25	Виконано
7	Розробка інформаційної системи	14.04.25	Виконано
8	Розробка функціональності фнформаційної системи	17.04.25	Виконано
9	Підготовка матеріалів та опис розробки інформаційної системи	25.04.25	Виконано
10	Аналіз виконання роботи, висновки, додатки	10.05.25	Виконано
11	Перевірка якості виконання бакалаврської роботи та усунення недоліків	13.05.25	Виконано

Студент Д. Ю. Соловйов

Керівник роботи к.т.н., доц. каф. ОТ А. В. Снігур

ЗМІСТ

АНОТАЦІЯ

УДК 004.9:37.018.43

Соловйов Д.Ю. Комп’ютерна система оцінювання параметрів студента при дистанційному навчанні. Бакалаврська кваліфікаційна робота зі спеціальності (123) – «Комп’ютерна інженерія», освітня програма – «Комп’ютерна інженерія». Вінниця: ВНТУ, 2025. 75 с.

На укр. мові. Бібліогр.: 31 назв; рис.: 35; табл. 4.

У бакалаврській кваліфікаційній роботі розроблено веб-систему для дистанційного вивчення слів англійської мови та її підсистеми оцінювання якості навчання.

Основний напрямок дослідження даної частини кваліфікаційної роботи – проєктування та інтеграція підсистеми оцінювання якості навчання користувача. Під час виконання роботи проаналізовано системи-аналоги, виявлено та враховано їх недоліки у власній розробці, вивчено основні способи використання веб-технологій у різних навчальних процесах, у тому числі оцінювання результатів навчання, на основі яких розроблено математичні моделі для перевірки, оцінювання та зберігання результатів навчання.

Ключові слова: веб-розробка, веб-система, дистанційне навчання, оцінювання якості навчання, .NET.

ABSTRACT

Solovyov D.Y. Computer system for assessing student parameters during distance learning. Comprehensive qualification work in the specialty (123) — "Computer Engineering", educational program — "Computer Engineering". Vinnytsia: VNTU, 2025. 75 p.

In Ukrainian. Bibliography: 31 titles; fig.: 35; table. 4.

In the bachelor's qualification work, a web system for distance learning of English words and its subsystem for assessing the quality of learning are developed.

The main research direction of this part of the qualification work is the design and integration of a subsystem for assessing the quality of user training. During the work, similar systems were analyzed, their shortcomings were identified and taken into account in their own development, and the main ways of using web technologies in various educational processes were studied, including the assessment of learning outcomes, on the basis of which mathematical models were developed for checking, evaluating and storing learning outcomes.

Keywords: web development, web system, distance learning, assessment of the quality of learning, .NET.

ВСТУП

Цифрові перетворення призводять до стрімкого зростання попиту на онлайн-вивчення мов. Це економічно вигідно порівняно з традиційним процесом навчання; час можна було ефективно планувати, оскільки освіта була доступна в будь-який момент.

Цільова аудиторія електронного навчання досить обширна і включає студентів, кандидатів, науковців та працюючих громадян, які бажають підвищити свій професійний конкурентоспроможний рівень шляхом отримання додаткової кваліфікації. Це **актуально** у випадку інтеграційних процесів України з Європою.

Сучасний рівень інформаційних технологій відкриває нові горизонти для дистанційного вивчення англійської мови, що значно розширює освітній простір. Використання розвинутих механізмів інформаційного обміну та системної взаємодії забезпечує реалізацію міжпредметних зв’язків та соціокультурних зв’язків. Сфера дистанційної освіти повинна мати необмежений простір щодо взаємодії всіх суб’єктів освітнього процесу, але водночас вона має бути підпорядкована вимогам стандартів освіти під керівництвом викладачів. Інформаційне середовище як платформа для вивчення англійської мови потребує потужного систематизованого ядра, яке координувало б усі процеси та зв’язки.

Одними з провідних вчених з методики дистанційної освіти є: Азаров О.Д., Биков В.Ю., Гриценчук О.О., Залізецький В. В., Конєвщинська О.Є., Крупельницький Л. В., Марков Є.С., Сімонсон М., Черняк О. І. та ін. та інші [1-5].

Поточні математичні моделі для дистанційного навчання мають недоліки: неадекватна оцінка знань користувача, низька точність і дороговизна реального розгортання. Перевірка комп’ютерних систем дистанційної освіти виявила ці недоліки: погані інтерфейси, слабка співпраця між людьми, які використовують їх для навчання, і необхідність платити за використання повнофункціональних версій.

Активний суспільний попит на дистанційну освіту та недостатня розвиненість теоретичної та практичної бази зробили актуальним подальше

дослідження цього напряму, яке буде проводитись у рамках даної інтегрованої магістерської роботи.

Метою роботи є удосконалення системи дистанційного навчання англійської мови шляхом створення СДН.

Для досягнення мети необхіжно виконати такі **задачі**:

— проаналізувати існуючі способи оцінки навчання у СДН.

— вивчити слабкі та сильні сторони сучасних комп'ютерних систем в оцінці якості дистанційної освіти;

— створити макет субсистеми оцінювання якості освіти;

— використовуючи аналіз слабких і сильних сторін існуючих моделей, розробити базові моделі нашої системи: моделі перевірки навчальних результатів та оцінки якості, а також модель обліку навчальних досягнень.

— обґрунтувати вибір засобу реалізації програмного забезпечення

— інтегрувати підсистему оцінки якості освіти в веб-системи дистанційного навчання англійської лексики

— протестувати розроблену систему

Об'єктом дослідження є процес створення веб-систем дистанційного навчального процесу.

Предметом дослідження є методи та засоби створення Web-системи дистанційного вивчення англійської мови.

Методи дослідження. У роботі використані такі методи досліджень: аналіз інформаційної технології автоматизації онлайн-навчання, методи обробки цифрової інформації, методи визначення поточного рівня знань, методи взаємодії між серверами, методи комунікації з базою даних, методи об'єктно-орієнтованого програмування для отримання подальших результатів роботи програми.

Практичне значення отриманих результатів полягає в можливості підвищення рівня володіння англійською мовою за допомогою Web-додатку для оцінювання якості освіти, інтегрованого в Web-систему дистанційного вивчення англійської лексики.

1 ТЕОРЕТИЧНІ ЗАСАДИ ПОБУДОВИ КОМП'ЮТЕРНИХ СИСТЕМ ДИСТАНЦІЙНОГО НАВЧАННЯ

1.1 Порівняльний аналіз комп'ютерних СДН

Технології у СДН мають великий дидактичний потенціал у системі навчання мови на всіх рівнях освіти. Цифровізація та соціально-економічні умови спонукали заклади освіти до оптимізації ресурсного забезпечення, переорієнтації на впровадження інноваційних форм навчання, зокрема дистанційних технологій. Подальше впровадження інформаційно-комунікаційних технологій формує стратегію реформування української системи освіти для завоювання конкурентоспроможності на ринку освітніх послуг.

Контекст веб-технологій ініціює актуальність їх впровадження в навчальний процес для вдосконалення методик і підвищення ефективності навчання, оскільки світ стає все більш цифровим через поширення веб-технологій. Недостатньо уваги приділено ролі нових інформаційно-комунікаційних технологій у викладанні мов, про що свідчить аналіз сучасних методів навчання. Цей факт став основою для нашого вдосконалення вивчення іноземних мов.

Дистанційне навчання — це метод організації навчального процесу на основі інфотехнологій, який використовується як тоді, коли люди, які навчаються, знаходяться далеко один від одного, так і безпосередньо в школі, щоб допомогти створити самостійну роботу в розумінні уроків.

Дистанційне навчання налаштовує роботу на навчальний процес на основі самостійного навчання; інформаційна підтримка та інструменти, які пропонують потрібні дані, спілкування між членами, можливість самостійного навчання незалежно від часу та місця, а також пропонують можливості для перевірки знань. Це оновлена форма заочної форми навчання, основним аспектом якої є використання комп'ютерних технологій та доступ до інтернет.

Робота у СДН розглядається як інтегральний, синтетичний, гуманістичний підхід до освіти з використанням широкого поєднання як звичайних, так і нових технологічних інструментів. Серед ключових особливостей: використання

інформаційно-комунікаційних технологій, вхід учня — його самостійна організація навчального процесу, час, обраний темп та інтенсивність навчальної діяльності, вільний доступ до інформації та можливість асинхронного навчання.

Навчання у СДН вважається пов’язаною, синтетичною, гуманістичною формою освіти, яка передбачає використання як традиційних, так і сучасних інструментів інформаційних технологій. Ключові особливості — використання інформаційно-комунікаційних технологій, активність учня, його самостійність, темп та інтенсивність навчальної діяльності з інтерактивним спілкуванням з викладачем в асинхронному режимі.

У використання СДН є переваги:

— користування курсами і завданнями з будь-якої точки світу;
— індивідуальне навчання;
— консультування з викладачем під час виконання завдань;
— об’єктивна методика оцінювання набутих знань;
— гнучкий підхід до навчання;
— можливості професійного зростання студентів і викладачів;
— швидке створення матеріалів в електронних курсах;
— освітня компетентність через розробку курсу командою експертів;
— економія коштів на навчально-методичне забезпечення

Використання інформаційно-комунікаційних технологій є доцільним і ефективним. Це передбачає постійне оновлення електронних навчально-методичних матеріалів, створення електронних підручників і комплексів для вивчення розмовної та професійної мови.

— розробити підручники, що спрямовані на вдосконалення навичок усного мовлення
— розробити ефективну систему оцінювання знань і вмінь
— розробити методику створення аудіовізуальних матеріалів
— створювати завдання для вдосконалення їх оцінювання

— визначити словниковий запас і граматичні основи для кожного модуля та для курсу в цілому

— розвивати мотивацію вивчення мови учня

До технологічних заходів системи відносяться:

— забезпечення балансу між синхронною (режим «on-line») та асинхронною (режим «off-line») роботою системи;

— підготовка педагогічного колективу до оволодіння інформаційно-комунікаційними технологіями в системі;

— постійний моніторинг та контроль навчального процесу;

— захист інтелектуальної власності та технологічного розвитку.

Економічні та організаційні заходи полягають у створенні експертних робочих груп з розробки інформаційних освітніх систем.

Навчальний процес покликаний вирішувати навчальні завдання з певною дидактичною повнотою. Важливою складовою є моніторинг навчальних результатів.

Діагностика — це багатопланова процедура, яка передбачає вимірювання навчальних досягнень, уточнення умов навчального процесу та отримання додаткової інформації про прогалину в знаннях і причини, пов’язані з результатами або нерезультативністю результатів навчання. Діагностика розглядає освітні результати у зв'язку зі способом їх досягнення, включаючи аналіз, виявлену динаміку.

Діагностична структура включає підсистеми контролю, перевірки результатів, аналізу та прогнозування результатів у дидактиці.

Контроль у дидактиці — це нагляд, спостереження, перевірка й оцінка результатів навчання.

У контексті компетентнісного навчання це проявляється у відповідальності студентів, прагненні до закріплення позитивних результатів, зростанні вимогливості до програмних результатів навчання.

Система дистанційного навчання не є тимчасовою, і питання, пов’язані з її вирішенням, слід вирішувати з огляду на створення центрів підготовки викладачів на міжвузівській базі, комплексну інформатизацію навчальних закладів, створення пунктів оцінювання якості електронних курсів.

Впровадження технологій дистанційного навчання забезпечує високу конкурентоспроможність, потребує належно організованої системи заходів із впровадження.

1.2 Функціональність і архітектура

Багато сайтів і порталів, які ми відвідуємо щодня, є веб-додатками: поштові служби, соціальні мережі, пошукові програми, системи онлайн-банкінгу, магазини, онлайн-редактори слів, ігри.

Основна відмінність веб-додатків від веб-сайтів полягає в тому, що замість статичних документів відображаються динамічні програми. Веб-додатки — це клієнт-серверні рішення, які зберігаються на сервері та працюють як на сервері, так і на стороні користувача.

Серверна сторона веб-програми (Back-end) є основною частиною програмного забезпечення для обчислень на основі запитів користувачів в Інтернеті через веб-браузери. Багато клієнтів можуть отримати доступ до сторони сервера одночасно. До розробки Back-end частин залучені розробники, знайомі з технологіями PHP, Java, Python.

Інтерфейс веб-програми — це програмне забезпечення, яке працює у веб-браузері на пристроях користувача. Він надає інтерфейс користувача та завантажується як динамічні веб-сторінки. Веб-програми працюють на будь-якому пристрої з Інтернет-браузерами. Для розробки його Front-end частин залучаються розробники, які знайомі з React.js, Vue.js, JavaScript, HTML, CSS, Ajax. Розробка веб-додатків на Java є популярною.

База даних веб-програми — це сукупність різноманітних деталей, життєво важливих для торгівлі.

Створення веб-програм дозволяє скинути прив’язку до певної системи — Windows, MacOS, Android або iOS. Веб-програми працюють повсюдно з веб-браузером і посиланням на Інтернет.

1.3 Аналіз підходів до організації процесу контролю й оцінювання результатів навчання

Контроль включає перевірку, оцінювання та облік результатів навчання. Верифікація стосується ідентифікації, вимірювання та порівняння результатів навчання на конкретному етапі процесу навчання.

Оцінювання — визначення та вираження в балах (кількісне оцінювання) або оціночних судженнях учителя (якісне, словесне оцінювання) рівня засвоєння знань, до програмових результатів навчання. Оціночне судження веде учнів до розуміння якості набутих знань; він охоплює методологію та виразністю мовлення. Корисними виявляються рекомендації щодо прогалин у знаннях і конкретна робота з їх усунення. Оціночне судження стосується здатності студентів аналізувати свою роботу; виявляється залежність оцінки від методу роботи. Об’єктивно виставлена оцінка підбиває підсумки виконаної роботи, а слово продумане — на роботу. Бухгалтерський облік — це просто фіксація та зберігання результатів набутих знань у вигляді оцінок та оціночних суджень викладача в журналі, табелі, атестаті тощо. Матеріали обліку результатів навчання стають основою для аналізу навчального процесу та прогнозування, яке, у свою чергу, включає наступне:

Компоненти, які взаємозалежні та взаємодіють у діагностичному процесі:

— прогнозована вчителем активність користувача;

— прогнозування власної діяльності;

— прогнозовані педагогом тенденції навчального процесу.

Отже, завданнями діагностичного процесу є:

— обсяг, рівень і якість матеріалу;

— труднощі та помилки;

— інформація про самостійну роботу

— перевірка ефективності форм організації, методів і засобів навчання;

— перевірка готовності користувачів до вивчення нового матеріалу.

Результати моніторингу навчально-пізнавальної діяльності користувачів відображаються в її оцінці. Термін «оцінка» відноситься до характеристики цінності, рівня або значущості будь-яких об'єктів або процесів. Оцінити означає встановити рівень або якість чогось. З точки зору навчально—пізнавальної діяльності під оцінкою розуміється ступінь відповідності користувачів вимогам поставлених перед ними завдань у процесі навчання, а також їх підготовленість і спроможність, якість набутих знань, сформованих умінь і творчих здібностей.

При індивідуальному контролі враховується змістовність відповіді користувача, логіка його суджень, доказованість положень, можливість застосування знань.

Повсякденна виховна робота включає контроль та діагностику поточного. Це допомагає з'ясувати, як відбувається процес появи нових знань у користувача, з якими труднощами він стикається. Отримані результати дозволяють користувачеві впевнено рухатися далі або вносити певні корективи в навчальну програму.

Остання перевірка проводиться, звісно, після ознайомлення з частиною програми іншої перевірки та передбачає показ знань, умінь, навичок, що дозволить переконатися в справедливості тесту, відповідати потребам індивідуального підходу.

Таблиця 1.1 — Тиип контролю результатів навчального процесу

Контроль	До	Після	Сумарний	Для особи
Дані для входу	Однакові для всіх	Однакові для всіх	Однакові для всіх	Індивідуальні для кожного
Система оцінюання	Стала	Стала	Стала	Гнучкіша
Багаторазовість	+	+	+	-
Вихідні дані	Сталі	Сталі	Сталі	Гнучкіші

Певний вид контролю можна використовувати лише один раз у рамках вивчення одного матеріалу. Наприклад, один матеріал вивчається за певним

критерієм раз і назавжди. Якщо у користувача є певний рівень знань у цій галузі, починати навчання з нуля для нього буде неефективним. Система, в рамках якої здійснюється оцінювання, має однозначно визначати рівень знань користувача, визначений встановленими критеріями, отже, фіксованим, що, у свою чергу, не збільшує обсяг вихідних даних після перевірки результатів навчального процесу. Цей вид контролю може використовуватися знову і знову різними користувачами після одного створення, тому його впровадження забезпечує значний ефект з відносно незначними витратами ресурсів.

Активний контроль застосовується найчастіше з усіх форм контролю, як правило, після завершення вивчення розділу або теми. Його слід застосовувати в будь-якій предметній області для перевірки якості закріплення невеликого обсягу вивченого матеріалу. Результати такого контролю мають бути здатними відразу ж ініціювати зміну навчальних програм із виділенням проблемних моментів для повторного вивчення, не чекаючи закінчення теми чи модуля. Система оцінювання повинна чітко відрізняти засвоєні фрагменти знань від незасвоєних.

Підсумковий контроль визначає рівень знань з певної предметної галузі. Він охоплює значно більшу кількість і різноманітність завдань, ніж поточний контроль, і його слід застосовувати наприкінці вивчення певної теми чи курсу. Результати цього виду контролю дають можливість викладачу оцінити ефективність всього курсу, а користувачеві оцінити кінцевий рівень своїх знань, наприклад, рівень володіння мовою після навчання. Система оцінювання працює на основі добре встановлених критеріїв ідентифікації засвоєння знань і тому є фіксованою. Обсяг вихідних даних фіксований. Цей вид контролю можна застосовувати постійно, і його застосування є необхідністю, оскільки без нього неможливо оцінити ні рівень знань, ні ефективність курсу навчання.

Індивідуальний контроль є найбільш гнучким у порівнянні з тими видами контролю, які практикувалися в минулому. Швидше за все, це буде прийнято як поточна практика. Його унікальність полягає в назві: список завдань користувача та система оцінювання, яка виконується для користувача, що в кінцевому підсумку

призводить до набагато ширшого контролю. Він може давати різні результати залежно від налаштування та мети контролю, наприклад, щодо рівня домінування, освоєного на основі вибірки знань із незв'язаних тем, усунення певних питань або представлення результатів контролю, де відповіді є неповними/неточними, а також використання чи ні довідкової інформації, витраченого часу тощо. Високі витрати ресурсів характерні для реалізації окремого типу контролю, що одночасно дозволяє більше поглиблені результати. Порівняльна характеристика систем-аналогів вказана у таблиці 1.2.

Таблиця 1.2 — Порівняльна характеристика систем-аналогів

Застосунок / Характеристика	Zoom	Google Meet	Microsoft Team	Team Viewer	Розробляємий застосунок
Наявність підтримки LAN	-	-	+	-	+
Інтерактивна взаємодія між клієнтами	+	+	-	+	+
Наявність програми на ПК	+	-	+	+	+
Засоби візуальних підсказок	-	-	-	-	+
Інтеграція з іншими застосунками	-	+	-	+	-
Зручність інтерфейсу	-	+	-	+	+

1.4 Функції діагностики результатів навчання

Для реалізації цілеспрямованого управління навчальним процесом необхідна оперативна комунікація, де відповідає навчальна та перевіряюча функції. Перевірка знань дає інформацію про динаміку пізнавальної діяльності користувачів, про індивідуальні та загальні досягнення та про весь процес перебігу знань, умінь і навичок. Навчально-перевірочна функція включає детектування рівня результатів для кожного користувача, просування навчального програмного матеріалу серед користувачів.

Функція навчання перевіряє кінцеві результати навчання, що допомагає краще зрозуміти матеріал, краще систематизувати знання, а також закріпити їх. Відповідь допомагає користувачеві здійснити критичний аналіз.

До діагностично-просвітницьких функцій відносяться заходи корекції навчального процесу для усунення білих плям у діяльності користувача та викладача. Перевірка також є інструментом виявлення ефективності методів і способів навчання. Учитель запитує себе, у випадку поганої оцінки стосовно своєї правоти. Які висновки слід зробити для майбутньої педагогічної роботи? Тому саме в процесі тестування відбувається справжня корекція. Під корекцією тут розуміється змінення цілей, методології навчання, технології контролю на основі отриманого результату (заповнення прогалин).

Методи контролю сприяють зрілості користувачів і, з цієї причини, він повинен оцінювати стан користувача, який контролюється стимулюючою та мотиваційною функцією. Якщо вимоги занадто великі, то це просто сповільнить процес розвитку. Занижені вимоги шкідливі для користувачів, оскільки не приводять їх у рух. Щоб забезпечити успішний розвиток користувачів від стимуляції під час контролю, він повинен:

— розуміти його вимоги;

— не сумнівався, що знання необхідні.

Тому мотиваційно-формуюча функція пробуджувала і приводила бажання користувачів до кращих результатів, розвивала відповідальність, прищеплювала почуття змагальності та створювала позитивні мотиви.

Контролю потрібно навчити користувачів і робота стане систематичною, а це привносить дисципліну. Все це може сприяти розвитку самостійності. Це допомагає користувачам зрозуміти себе та застосувати методи самоконтролю. Крім того, це розвиває працьовитість і будь-яку пов'язану з нею діяльність разом з особистою творчістю. За своєю природою це заохочує індивіда, частина якого демонструється в догоджанні, пристосуванні, обмані та лицемірстві. Отже, контроль вимагає педагогічного керівництва, великої майстерності й такту.

Розвивальна функція полягає у розвитку пам'яті, уваги та уяви, а також у формуванні вміння працювати самостійно та розвитку способу мислення. Ця функція невіддільна від освітньої та виховної функцій. У процесі виконання завдань його користувачі повинні застосовувати їх у змінених або нових ситуаціях, вчитися виокремлювати головне та відтворювати інформацію різними навчальними завданнями тощо.

Усі перераховані вище функції взаємопов’язані і мають досить складну природу.

Ефективним воно має бути з дотриманням таких педагогічних вимог, у яких вчителі перевіряють підсистему перевірки результатів навчання:

— регулярний контроль результатів навчання;

— зробити його об'єктивним і мотивованим;

— єдність у вимогах до знань, умінь, навичок;

— контроль має бути диференційованим [7].

1.5 Аналіз недоліків та переваг існуючих математичних моделей процесу оцінювання знань користувача

Перевірка знань передбачає завдання двох типів: дихотомічні та політомічні. Простіше кажучи, дихотомічна задача має лише дві відповіді. Тоді як політомне завдання має більше набагато більше.

Одна з головних робіт у цій галузі належить датському математику G. Rasch, який представив модель (рис. 1.1). У ній ймовірність вірної відповіді описується як різниця латентних рис θi і βj – тобто різних рівнів станів користувача та складності завдання [8].

$$P_{ij} = \frac{e^{\theta_i - \beta_j}}{1 + e^{\theta_i - \beta_j}} \tag{1.1}$$

По результатам Раша, отримуються графіки виконання завдань (рис. 1.1).

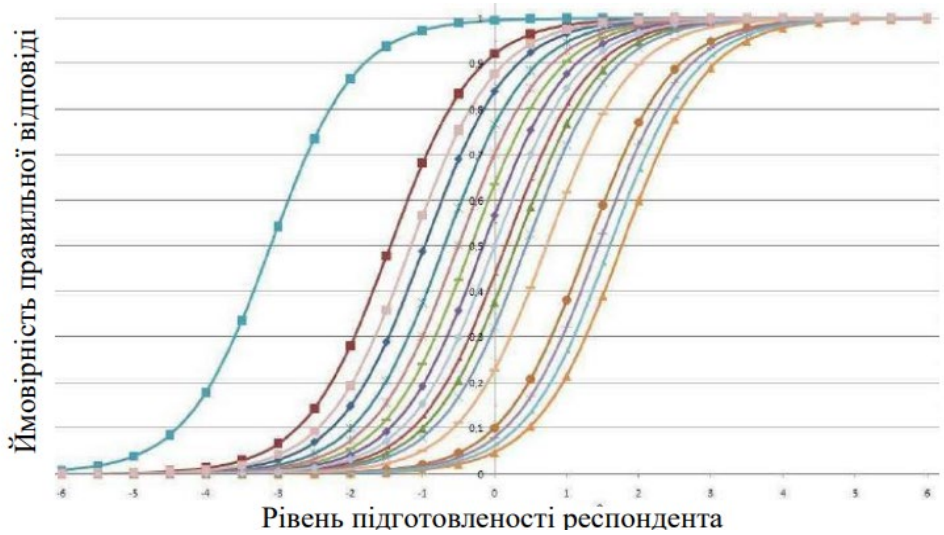

Рисунок 1.1 — Криві, що описують характеристики тестів

Математична модель пов’язує результати емпіричних тестів зі значеннями отриманих параметрів θi і βj.

Описана модель називаєтьмя однопараметричною (1 Parametric Logistic Latent Trait Model — 1PL).

Для цієї запропонованої моделі характеристичні криві завдання мають однакові, тобто значення диференціючої здатності є сталою величиною, яка дорівнює 0,25. А. Бірнбаум ввів додаткове значення α (диференціювальна здатність завдання). Тому ця модель відома як двопараметрична [8]:

$$P_{ij} = \frac{e^{\alpha(\theta_i - \beta_j)}}{1+e^{\alpha(\theta_i - \beta_j)}} \tag{1.2}$$

Параметр α визначає нахил (кривину). Моделі Бірнбаума дають характерні криві, з яких видно, що зі збільшенням значення параметра, тим крива крутіша, а розрізнювальна здатність завдання вище (рис.1.2)

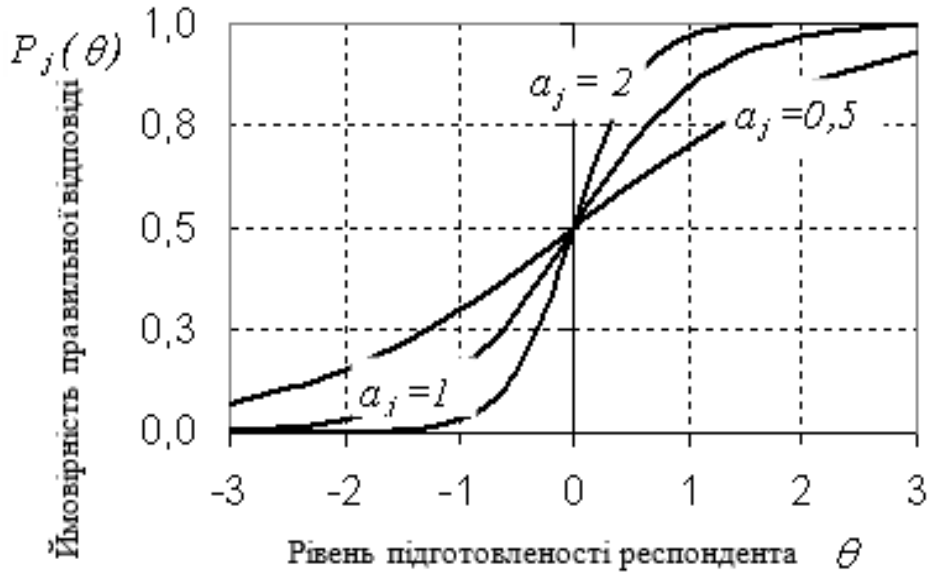

Рисунок 1.2 — Представлення двопараметричної моделі у вигляді кривих

З графіків, що наведені нижче видно, якщо використовувати параметр для вгадувань, то криві зміщуються вгору на значення cj. Звідси випливає досить нетривіальний висновок, що тести із невідомими завданнями гірше розрізняються користувачем [8].

$$P_{ij} = c_j + (1 - c_j) \frac{e^{\alpha(\theta_i - \beta_j)}}{1 + e^{\alpha(\theta_i - \beta_j)}} \tag{1.3}$$

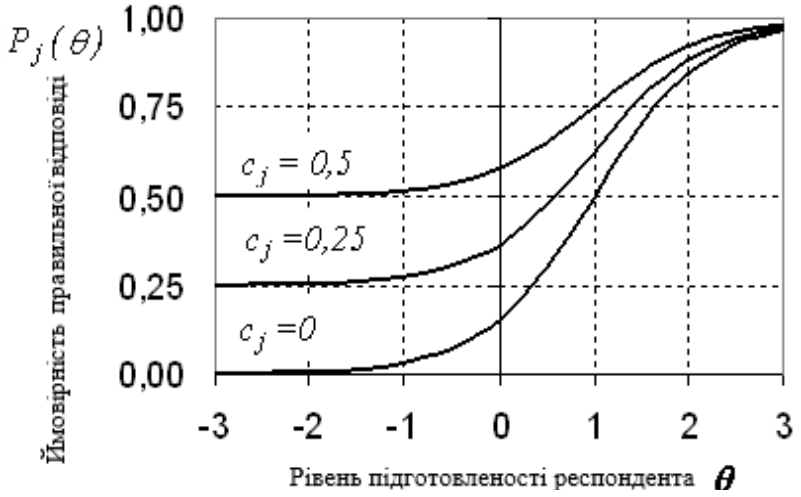

Рисунок 1.3 — Криві, що описують трипараметричну модель

У табл. 1.3 наведено характерні риси всіх згаданих моделей.

Таблиця 1.3 — Особливості моделей перевірки знань

Модель	Особливості моделі
Для 1PL	Стале значення дескримінуючої здатності
Для 2PL	Передбачено враховування десримінуючої здатності
Для 3PL	Передбачено враховування десримінуючої здатності

Модель для двох параметрів представляє абсолютно відмінне уявлення. Очевидно, як видно з рис. 1.2, для параметра складності завдання α=0,5 в діапазоні значень, більших за нуль, є найскладнішим із трьох доступних завдань. Тобто ймовірність, що буде правильна відповідь мінімальна. У діапазоні значень, менших за нуль, це завдання тепер найпростіше. Отже, для слабких користувачів ми можемо сказати, що це найважче завдання

Подібні результати з’являються для моделі з трьома параметрами (рис. 1.3) ілюструє сценарій характеристичних кривих, що не перекриваються, загальні параметри $\beta = 1$ і $\alpha = 1$. Це означає, модель двох параметрів Бірнбаума позбавляється основної поганої частини однопараметричної моделі. А трипараметричну модель можна використовувати лише із запитаннями закритого типу, тому для якісної перевірки знань вона не підходить.

1.6 Комп’ютерні системи та програмні засоби для оцінювання якості навчання

Вплив СДН на трудове навчання є радикальним і різноманітним, і більшість простих правил було перенесено зі старої моделі навчання.

Основні фактори, які вирішують, чому бізнесмени обирають дистанційне навчання [7]:

— пов’язано з вартістю і витратами на проїзд і проживання, оскільки добові використовуються для покриття їжі, яку втрачений працівник мав би вдома.

— пов’язано з масштабованістю, тому що Inernet дозволяють задовольнити потреби в широкому споживанні змісту навчального плану для багатьох презентацій і навчальних матеріалів, щоб зменшити відмінності між користувачами очних навчальних закладів і дистанційного навчання.

— технології дистанційного навчання повинні надавати користувачам доступ до ресурсів, необхідних для негайного застосування знань, що неминуче призведе до підвищення продуктивності;

— методи доступу та зміст мають бути орієнтованими на користувача, виходячи з його потреб і контексту, у якому він застосовуватиме інформацію.

Технічна підтримка СДН включає:

— інструмент розробки змісту освіти;

— підсистеми керування навчанням;

— підсистеми для обмінюввання інформації;

— підсистеми для керування наповненням;

Підсистема для керування навчанням може виконувати:

— доступ студентів;

— керування навчанням;

— синхронний та асинхронний зв'язок;

— аналітику.

Для отримання доступу до навчальних матеріалів студенти звертаються через відповідну базу даних. Комунікації між викладачами та студентами здійснюються за допомогою підсистеми обмінювання інформацією.

Сам по собі контент може бути салим, як—от HTML—сторінки та тексти, та інтерактивним, що включає елементи анімації з голосовою підтримкою.

Сталий контент можна створювати за допомогою Microsoft Word, а інший за допомогою спеціального програмного забезпечення

Модуль СДН для спілкування може мати такі функції:

— асинхронне спілкування, наприклад форуми та електронна пошта;

— синхронне спілкування, таке як голосове із програмним забезпеченням для відеоконференцій або спільного використання, а також віртуальний клас.

Вона дозволяє виконувати завдання різних типів складності. Для цього складаються речення зі слів (рис. 1.4), знаходиться лишнє слово, можна здійснити вірний переклад [21]. Для цього можна вибрати певна кількість слів. Гральний підхід тут не використовується порівняно з іншими програмами. Система також надає загальну статистику пройдених курсів та рівень знань визначається за пройденими курсами. Головною перевагою навчання тут те, що він абсолютно безкоштовний. Крім того, розробники включили в Memrise багато стандартних курсів.

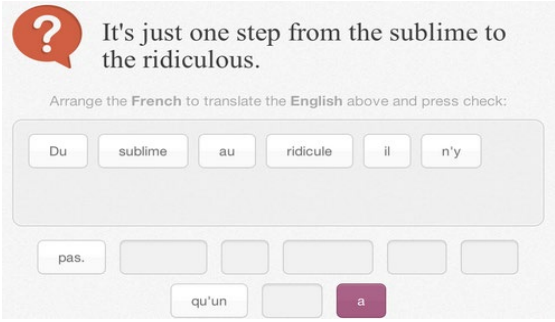

Рисунок 1.4 — У додатку Memrise можна робити перевірку знань

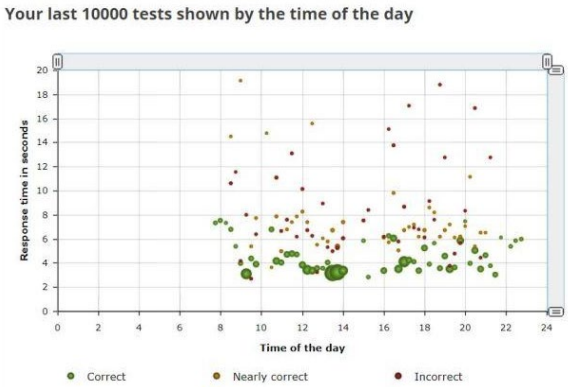

Рисунок 1.5 — У Memrise можна отримати результати навчання

Lingualeo система є досить поширеною. Система дозволяє опрацьовувати відеоматеріали та текстові, що спеціально підбираються для різних користувачів (рис. 1.6). Кожного дня здійснюється написання матеріалів, виконуються аудіо та

відео завдання. Якість вивчення слів аналізується за допомогою кількох статистичних графіків, та аналізується передбачуваний прогрес у письмі, аудіюванні, читанні та граматиці на рисунку 1.7.

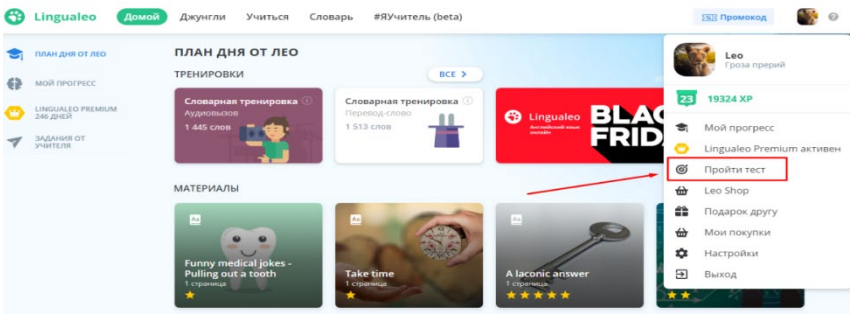

Рисунок 1.6 – Система Lingualeo

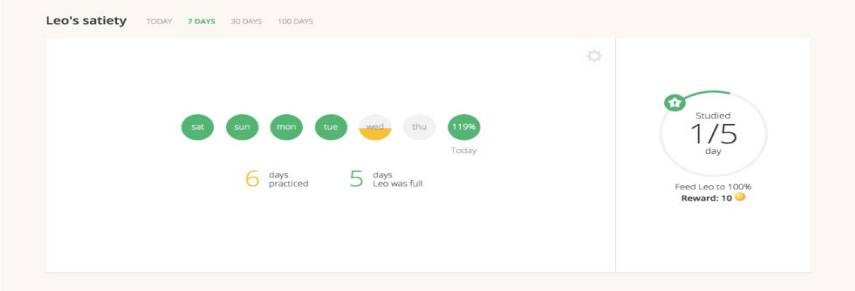

Рисунок 1.7 — Перегляд результатів у Lingualeo

Система Duolingo використовується для представлення навчальних завдань тепер складаються з упорядкування слів і речень або встановлення зв’язків. Нова версія системи має частину для прослуховування діалогів реальних ситуацій, що має дозволити вам вивчати мову. Вправи є одноманітними проте вони корисні для вивчення слів. На вкладці статистики можна подивитись статистику (рис. 1.8).

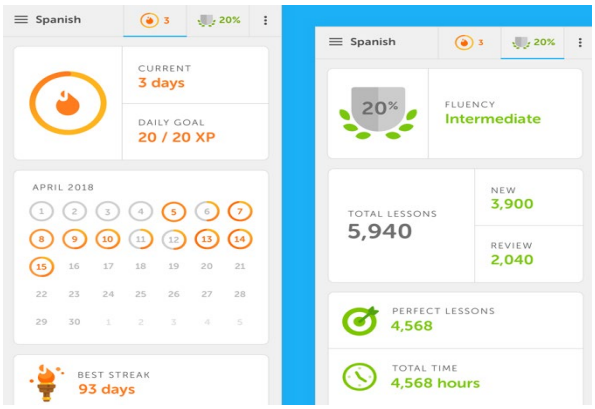

Рисунок 1.8 – Перегляд результатів у Duolingo

Таблиця 1.4 — Системи-аналоги

Контроль	Memrise	Lingualeo	Duolingo	Система, що розробляється
Вступне тестування	-	+	+	+
Поточне тестування	+	+	+	+
Підсумкове тестування	-	+	+	+
Індивідуальне тестування	-	-	-	+
Контроль	+	+	+	+
Розширене оцінювання	-	+	+	+
Присутність людини	-	-	-	+
Перегляд статистики навчання	-	-	-	+
Оплата	+	-	-	+

Усі описані вище системи були спеціально розроблені, щоб допомогти процесу навчання. Курс вибирається і вивчається за спеціальним алгоритмом, за яким дотримуються вказівки системи. Всі існуючі системи вже зазвичай мають стандартні модулі різної сфери та рівня складності.

Однією з основних переваг Memrise є те, що він абсолютно безкоштовний і автономний. Існує мінімальна кількість вправ, яких може бути недостатньо, щоб конкурувати з іншими в списку. Недоліками системи є обмежений вибір перевірок для контролю рівня знань та інформації про результати навчання.

Lingualeo — одна з найскладніших систем на ринку, але назвемо її точно системою для вивчення слів. До переваг системи можна віднести всі види контролю рівня знань, окрім індивідуального, а до недоліків – відсутність роботи з викладачем, а більша частина контенту доступна лише на платній основі.

На момент запуску Duolingo була найпопулярнішою системою для вивчення іноземних мов переважно через її простоту та відсутність реклами та платного контенту. Наразі система дозволяє комплексно контролювати рівень знань та оцінювати результати. Немає роботи з викладачем, а деякі частини його функціоналу доступні лише на платній основі.

Враховуючи вищевикладене, перевагами системи, що розробляється, є всі види контролю рівня знань, різноманітні вправи для перевірки знань, аналіз результатів контролю та статистика користувачів у вигляді інфографіки, а також можливість роботи з викладачем. Не кажучи вже про відсутність реклами та платних обмежень режиму Premium.

2 МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ ПІДСИСТЕМИ ОЦІНЮВАННЯ ЯКОСТІ НАВЧАННЯ

2.1 Розроблення структури підсистеми оцінювання якості навчання

Необхідно правильно оцінити, наскільки добре користувач впорався із завданням, які питання викликали труднощі, скільки часу знадобилося для вирішення завдання, скільки було потрібно спроб тощо. Результати дадуть можливість для подальшого прогнозування, планування та корекції навчального процесу. Глибокий аналіз дозволить зосередитися на найбільш проблемних моментах, не витрачаючи час на повторне вивчення вже засвоєного матеріалу.

У підсистемі розвитку якості навчання були запропоновані такі вимоги:

— визначити обсяг, рівень, якість засвоєння навчального матеріалу тощо;

— виявляти труднощі та помилки тощо;

— перевірити ефективність обраних форм організації, методів і засобів навчання тощо;

— визначити готовність користувачів до вивчення нового матеріалу тощо.

Вищезазначене можна об’єднати з поняттям діагностики та визначити комплексний процес вимірювання результатів навчання, встановлення умов і обставин процесу навчання, отримання інформації про прогалини в уміннях і знаннях, а також про причини, які викликають або перешкоджають досягненню окреслених результатів.

До складу процесу діагностики входять наступні підсистеми управління: облік, аналіз і прогнозування результатів, їх оцінка. У цій роботі ми розробили перші три компоненти, структурна схема яких показана на рис. 2.1 [10].

Зазначені структурні одиниці підсистеми контролю якості освіти, яка розробляється в цій роботі, повинні взаємодіяти, як показано на рис. 2.2:

— ввести нові результати та пройти повний цикл контроль

— вивести попередні результати контролю та оцінювання

— введення нових результатів і проведення повного циклу

— виведення попередніх результатів контролю та оцінювання

Рисунок 2.1 — Структура підсистеми оцінювання якості навчання

Рисунок 2.2 — UML діаграма діяльності системи оцінювання якості навчання

Діаграма класів UML системи оцінювання якості освіти, що включає класи, показані на рисунку 2.3:

— клас контролера системи оцінювання;

— навчання контролю обслуговування класу;

— клас обслуговування оцінки навчання;

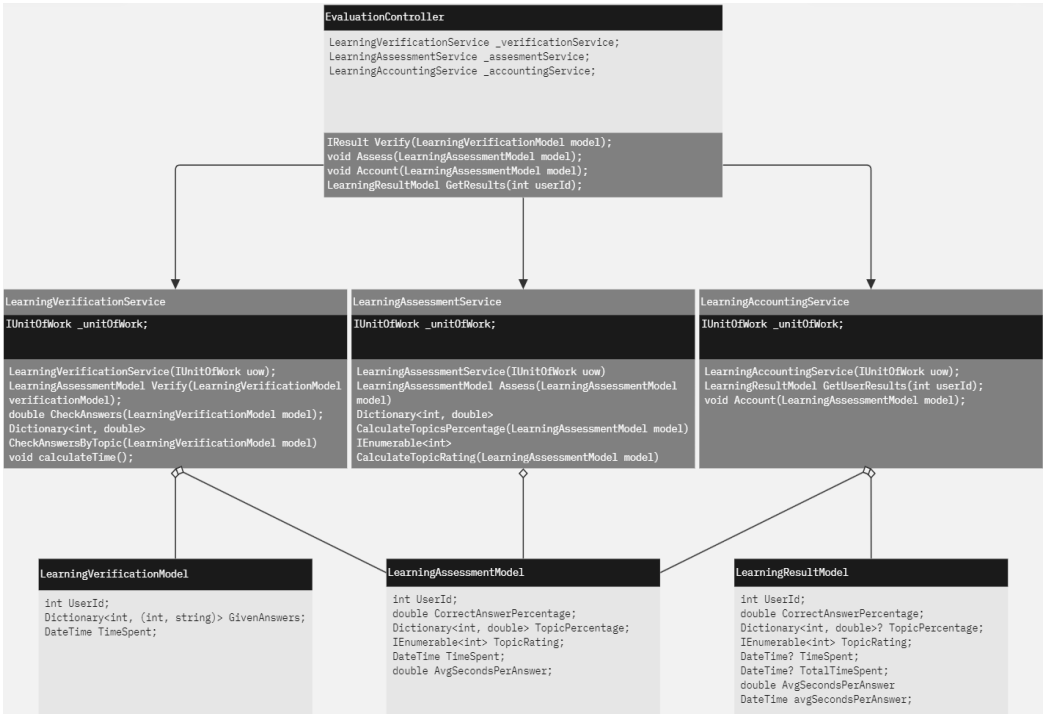

Рисунок 2.3 — UML діаграма класів системи оцінювання якості навчання

— клас обслуговування для підрахунку результатів навчання;

— модельний клас контролюючого навчання;

— модель класу для оцінювання навчання;

— модель класу для підрахунку результатів навчання.

Контролер EvaluationController взаємодіє з користувачами та системою оцінювання навчання. Під час створення отримує три послуги: перевірку, оцінку та облік результатів навчання. Його основною функцією є координація цих процесів перевірки, оцінювання та обліку результатів навчання.

Коли користувач надсилає відповіді, метод Verify отримує модель Learning Verification Model, яка охоплює ідентифікатор користувача, відповіді за темами та загальний час, витрачений на виконання завдань. Цей метод починається з перевірки правильності відповідей, потім обчислюється час для відповіді, а також запускається оцінювання на основі зібраних результатів. Оцінка, сформована за моделлю LearningAssessmentModel, надсилається до методу Assess, де підраховується відсоток загальних правильних відповідей для кожної теми разом із формуванням рейтингу теми. Після цього метод Account записує всі дані в облікову систему.

Служба перевірки навчання повинна втілювати логіку початкової перевірки; він перевірить правильність відповідей. Після виконання перевірки відповідей за темами він повинен потім обчислити час, а також навчитися створювати модель оцінювання навчання на основі отриманих даних. Усе це потрібно буде зробити шляхом зв’язку з базою даних за допомогою шаблону UnitOfWork.

Потім служба LearningAssessmentService наповнює модель оцінювання конкретними даними, обчислюючи загальний відсоток правильних відповідей і обробляючи статистику за темами. Це робиться для встановлення рейтингу тем у порядку зменшення якості засвоєння матеріалу. Це те, що потім може створити цілу картину успіху користувача.

LearningAccountingService зберігає кінцеві результати в базі даних, і ви можете отримати повну модель результатів за допомогою методу GetUserResults або зберегти нові оцінки, викликавши метод Account.

Основні дані в LearningVerificationModel включають відповіді користувачів із темами та загальним часом завершення. Після виконання тесту з’являється відсоток правильних відповідей, щоб створити LearningAssessmentModel разом із детальною інформацією про теми, рейтинги тем і середній час відповіді. Зрештою, ця остаточна модель, LearningResultModel, зберігає ці дані в базі даних і додає користувачеві загальний час навчання.

2.2 Побудова моделі перевірки результатів навчання

Тестування результатів навчання є обов’язковою частиною навчального процесу, яка вимірює результати навчання, дає зворотній зв’язок і є засобом коригування процесу навчання.

Підсистема використовує тестову форму такого процесу та три типи перевірки результатів:

— попередній, який відбувається відразу після реєстрації користувача;
— струм, який відбувається після кожного тесту;
— підсумковий, який охоплює весь пройдений матеріал.

Для компоненту тестування результатів навчання в магістерській роботі запропоновано модель (2.1), яка оцінює відповідь тестованого за двобальною шкалою (правильно чи неправильно).

$$R = \frac{\sum_{i=1}^{k} R_i}{n}, \tag{2.1}$$

де R_i — правильна відповідь для i-го завдання;

k — кількість правильних відповідей із n запропонованих ($k \leq n$).

2.3 Створення моделі оцінювання якості навчання

Рівень засвоєння матеріалу відображає дві складові дидактичного процесу: пізнавальну діяльність студента та діяльність керівництва процесом з боку викладача. Об'єктивні показники засвоєння дадуть змогу кількісно охарактеризувати обидві складові такого процесу. Ми будемо розглядати засвоєння у випадку результату дидактичного процесу та системи, а не вчителя, який обслуговує керівника. Засвоєння є одночасно змістом навчального процесу, певною поетапною діяльністю користувача з оволодіння знаннями [17].

Ця подвійність концепції засвоєння, процесу та результату формує взаємодію, на якій психологія та педагогіка намагаються проводити аналіз в управлінні процесом навчання. Якщо психологічна сторона здебільшого характеризує сам процес, то дидактична – здебільшого його результат.

Факт взаємозалежності процесу засвоєння та його результату привертає увагу до інтегративного психолого—педагогічного підходу до розробки критеріїв оцінювання якості засвоєння знань. Якщо ми намагаємося оцінити факт отримання знань, ми повинні створити відповідні умови, щоб користувач продемонстрував, що він може їх використовувати. Тому для вдосконалення процесу контролю розроблено алгоритм, який базується на класифікації завдань за їх дидактичними характеристиками: величиною (z), складністю (d), специфікою (c). Кількість балів, яку отримує користувач за виконання n завдань, розраховується наступним чином:

$$y = \sum_{i=1}^{n} w_i R_i , \quad (2.3)$$

де R_i — оцінка для i завдання;

n — їх кількість;

w_i — вагові коефіцієнти завдань.

Після всього отримується бал A, для користувача за n завдань:

$$A = \frac{y}{k_n}, \quad (2.4)$$

де y — бали користувача;

k_n — спроби реалізації.

Таким чином уточнений середній бал A`:

$$A' = A \times r + a_1 \frac{k_n}{n} + a_2 \frac{k_c}{n} + a_3 \frac{k_b}{n}, \quad (2.5)$$

де a_1, a_2, a_3 — коефіцієнт ваги (0...1).

Після розрахунків результати за допомогою вектора граничних значень перетворюються в умовну шкалу за Цельсієм. Параметри контролю, значення вагових коефіцієнтів wi, коефіцієнтів аі та значення елементів вектора граничних значень задаються на початковому етапі використання системи або за результатами попередньої перевірки рівня знань користувача.

2.4 Моделювання процесу обліку результатів навчання

Модель для обліку результатів навчання фактично є моделлю даних. Модель даних — це концептуальна схема, яка використовується для опису та представлення бізнес вимог, щоб забезпечити їх передачу та вираження. Він вказує, що це за дані, бізнес правила, що їх контролюють, і спосіб, у який вони зберігатимуться в базі даних [19].

Моделювання даних є одним із найважливіших завдань при розробці програмного додатку. Він закладає основу для зберігання, пошуку та представлення даних. У цій роботі використовується реляційна модель даних.

Реляційна модель — це набір даних, що складається з набору двовимірних таблиць. Більш просунута, ніж ієрархічна та мережева моделі даних, у своїй основі реляційна модель втілює високий рівень абстракції даних. Це табличне представлення даних, яке є простою формою і настільки знайоме, що воно стало фактичним стандартом, якому дотримуються майже всі сучасні комерційні СУБД. Реляційні бази даних реалізовані на основі реляційної моделі даних.

Оскільки в організації табличних даних немає ієрархії елементів, рядки та стовпці можна читати в будь-якому порядку. Ця структура має гнучкість вибору будь-якої підмножини елементів у рядках і стовпцях. Конкретні значення даних стосуються перетину рядків і стовпців; для кожного поля визначається набір його значень. Переваги використання реляційної моделі:

— інформація повинна зберігатися в зручному для читання форматі;

— модель даних має базуватися на строгій математичній нотації, яка дозволяє коротко описати необхідні маніпуляції з даними;

— дані в таблиці повинні бути незалежними від програм, які їх використовують;

— для роботи з моделлю немає необхідності мати повне знання структури бази даних.

Недоліками реляційної моделі є:

— швидкість доступу до даних відносно низька;

— створення бази даних є складним за допомогою реляційної моделі;

— труднощі, пов'язані з реалізацією зв'язків між таблицями

Першим кроком у розробці моделі даних є збір бізнес вимог щодо того, що система матиме наступну здатність зберігати особисту інформацію та слова користувачів, переклади на різні мови та належність до певного модуля.

Подальшим кроком буде формування концептуальної моделі даних. На цьому етапі важливо визначити сутності, які представляють атомарні дані, і налаштувати початкову модель даних. Відповідно до потреб можна розпізнати п’ять окремих сутностей: користувач, модуль, Word, здоров’я та стан здоров’я. Зв’язки між цими сутностями показано на рис. 2.4.

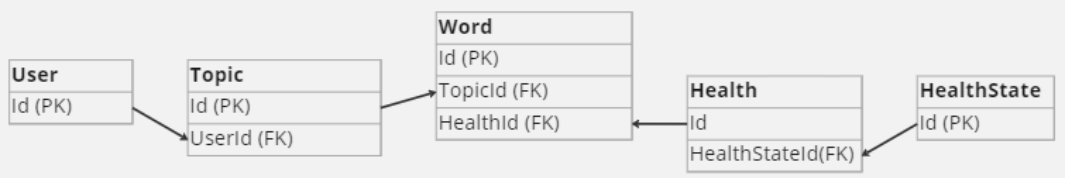

Рисунок 2.4 — Концептуальна модель даних

Після завершуємо опрацьовувати атрибути для сутностей (рис. 2.5).

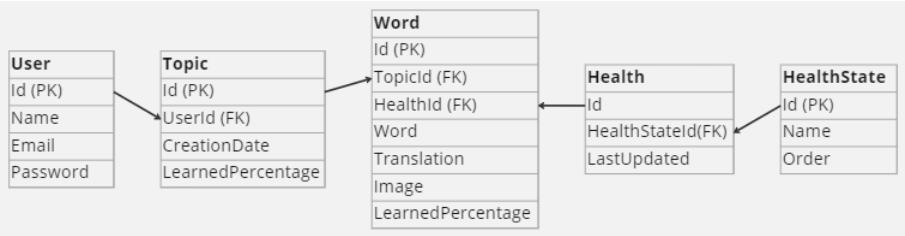

Рисунок 2.5 — Логічна модель даних

Модель описується формально перед реалізацією її за допомогою правильного розроблено та детально описано внутрішню структуру підсистеми оцінювання якості навчання.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ПІДСИСТЕМИ ОЦІНЮВАННЯ ЯКОСТІ НАВЧАННЯ

3.1 Обґрунтування вибору технологій розробки та мов програмування

Веб-розробка включає в себе безліч технологій і продуктів, які є в розпорядженні: мови програмування і мови розмітки в чистому вигляді, бази даних і сховища інформації, програмні продукти, які допомагають спростити розробку (фреймворки і системи управління сайтом), серверне програмне забезпечення. Як показано на рис. 3.1, архітектура сучасного сайту складна і складається з багатьох компонентів.

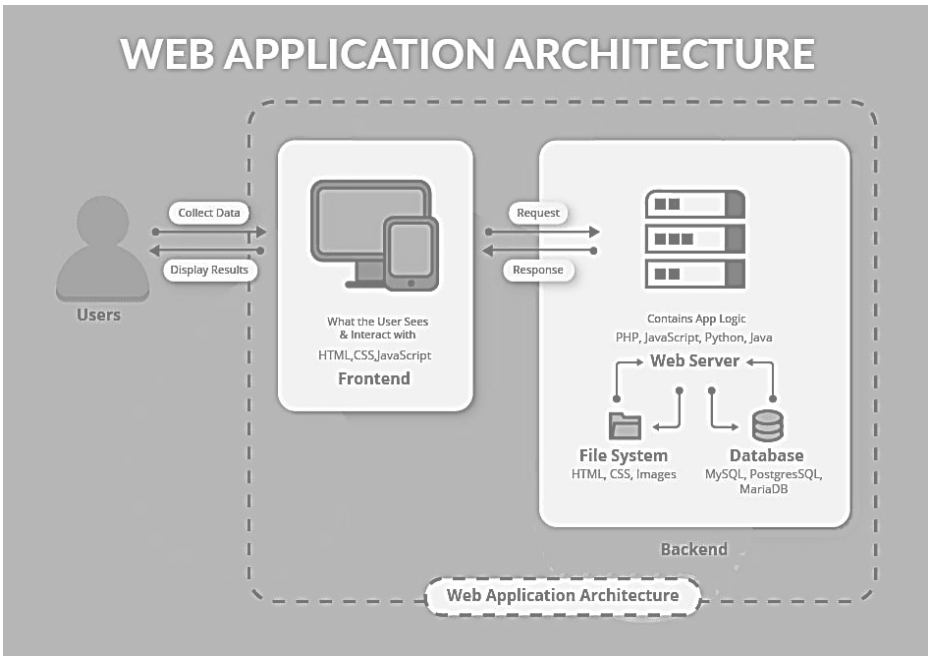

Рисунок 3.1 — Архітектура сучасного веб-додатку

Інтерфейс — це те, що ви бачите, коли відвідуєте сайт, він реалізований за допомогою HTML, CSS і JS.

Програмне забезпечення — це алгоритми, призначені для обробки запитів користувачів. По суті, програмне забезпечення створює або отримує з бази даних інформацію, яку запитує користувач, а також приймає та обробляє дані від користувачів. Він написаний мовами програмування на стороні сервера і зазвичай відповідає платформі розробки.

База даних — це місце, де ми зберігаємо дані, які використовуються на сайті. У ньому зберігається вміст усіх сторінок сайту та посилання на них, особисті файли користувачів системи та інше. Він має знайти та надати правильні записи (наприклад, інформацію, що відображається на сторінці) або записати нові дані (наприклад, змінити пароль для входу).

Хостинг — це сервер, де програмне забезпечення запускає сайт, а також зберігає базу даних. Типи хостингу можуть бути різні, і від його типу багато в чому залежить швидкість і надійність сайту.

Вимоги до компонентів цілком передбачувані:

— інтерфейс повинен бути приємним і легким для сприйняття, коректно відображатися в різних браузерах і на різних пристроях;

— програмна частина повинна швидко і безпомилково відповідати на запити;

— сервер повинен швидко обробляти запити і надійно зберігати інформацію в базі даних;

— сервер повинен працювати без перебоїв, не гальмувати, а також мати можливість витримувати багато одночасних запитів.

Angular — це платформа Google для створення клієнтських програм. В першу чергу він зосереджений на розробці рішень Single Page Application. У цьому сенсі Angular стане наступником іншого фреймворку AngularJS. При цьому це не нова версія AngularJS, а абсолютно новий фреймворк.

Angular пропонує функцію двостороннього зв’язування, яка динамічно змінює дані в одному місці інтерфейсу, коли дані моделі змінюються в іншому, і може реалізовувати шаблони, а також маршрутизацію. Зазвичай схема робочого процесу веб-додатку, заснованого на Angular, зображена на рис. 3.2.

Однією з головних переваг Angular є те, що він використовує мову програмування TypeScript. TypeScript містить багато концепцій, які зазвичай існують в об’єктно-орієнтованих мовах, таких як успадкування, поліморфізм, інкапсуляція та модифікатори доступу, серед інших.

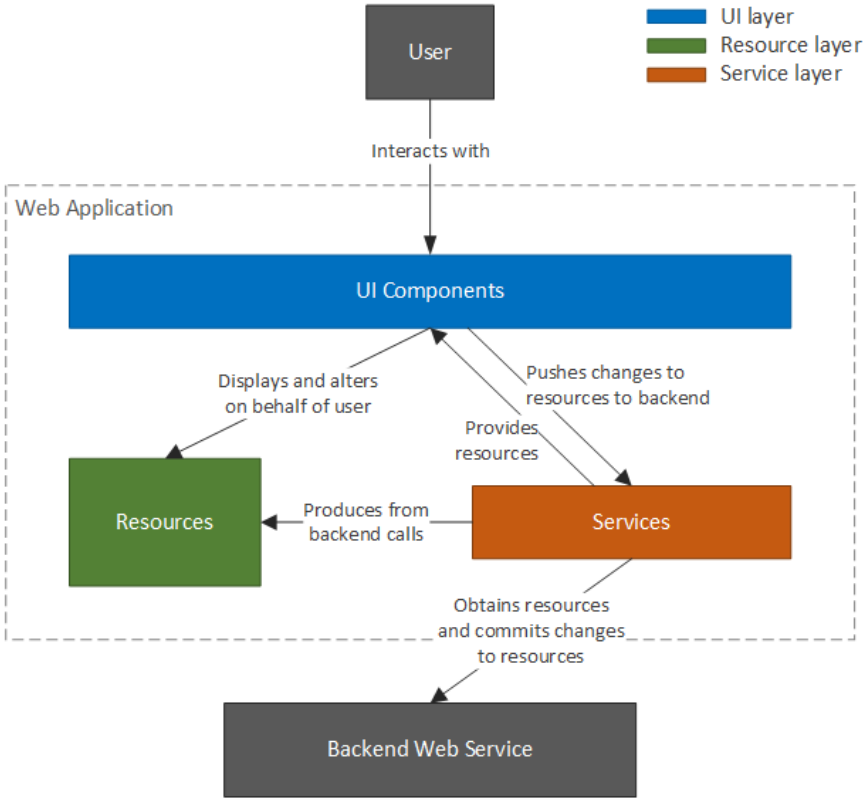

Рисунок 3.2 — Схема веб-додатку на Angular

Передбачається, що TypeScript є строго типізованою мовою, тому її легше зрозуміти розробникам Java, C# та інших строго типізованих мов. Компілятор створює той самий JavaScript, який потім виконує браузер. Однак жорстка типізація зменшує кількість потенційних помилок, які можуть виникнути під час розробки в JavaScript.

Потенціал TypeScript дозволяє швидше та простіше писати великі складні комплексні програми, відповідно їх легше підтримувати, розвивати, масштабувати та тестувати, ніж на стандартному JavaScript.

Крім того, він є кросплатформним, а це означає, що для розробки ми можемо використовувати як Windows, так і MasOS або Linux.

У той же час TypeScript є типізованим надмножиною JavaScript, а це означає, що будь-яка програма JS є програмою на TypeScript. У TS можна використовувати всі ті конструкції, які застосовуються в JS – ті ж оператори, умовні, циклічні конструкції. Більше того, код на TS компілюється в JavaScript. Зрештою, TS – це лише інструмент, покликаний полегшити розробку додатків.

Але ми не обмежені мовою TypeScript. За бажанням можемо писати програми на Angular за допомогою таких мов як Dart або JavaScript. Однак TypeScript є основною мовою для Angular [11].

ASP.NET Core — технологія для створення веб—додатків на платформі .NET, яку розвиває компанія Microsoft. Як мови програмування розробки додатків на ASP.NET Core використовуються C# і F#.

Мова програмування C# на сьогоднішній момент одна з найпотужніших мов, що швидко розвиваються і затребуваних в IT—галузі. Зараз на ньому пишуться різні програми: від невеликих десктопних програм до великих веб—порталів і веб-сервісів, що обслуговують щодня мільйони користувачів.

C# є об'єктно-орієнтованою і в цьому плані багато перейняла у Java та C++. Наприклад, C# підтримує поліморфізм, успадкування, навантаження операторів, статичну типізацію. Об'єктно-орієнтований підхід дозволяє вирішити завдання з побудови великих, але в той же час гнучких, масштабованих і додатків, що розширюються. C# продовжує активно розвиватися і з кожною новою версією з'являється все більше функціональностей [22-25].

Поточну архітектуру платформи ASP.NET Core зображено на рис. 3.3.

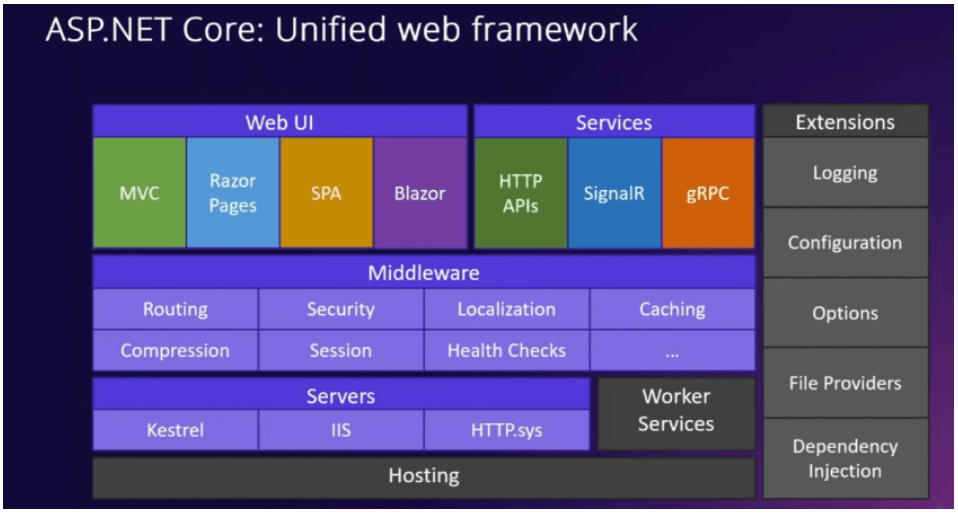

Рисунок 3.3 — Архітектура ASP.NET Core

На найвищому рівні знаходяться різні моделі взаємодії з користувачем. Це технології побудови користувальницького інтерфейсу та керування введенням

даних, такі як MVC, Razor Pages і SPA (Single Page Application — Angular, React, Vue) або Balzor. Крім того, це служби у формі вбудованих HTTP API, бібліотеки SignalR або служб GRPC.

Усі ці технології базуються на чистому ASP.NET Core або взаємодіють із ним, головним чином представленим різноманітними вбудованими компонентами проміжного програмного забезпечення, які використовуються для обробки запиту. Крім того, технології вищого рівня також взаємодіють з деякими розширеннями, які не є безпосередньою частиною ASP.NET Core — не безпосередньо для журналювання, конфігурації тощо.

ASP.NET Core Web API структуровано за шаблоном REST таким чином, що кожен тип http—запиту зіставляється з операцією або методом CRUD (GET для читання, POST для створення, PUT для оновлення, DELETE для видалення). Ресурси налаштовуються за допомогою методів контролера Web API, приклад, який добре працює з цією моделлю додатків, зокрема односторінковими додатками [13].

СУБД SQL Server є однією з найпопулярніших у світі і підходить для проектів будь—якого розміру: від невеликих додатків до великих високонавантажених проектів. Ця СУБД створена компанією Microsoft. Довгий час SQL Server був виключно СУБД для Windows.

Ключові особливості SQL Server:

— продуктивність. SQL працює дуже швидко

— можливість забезпечити безпеку даних за допомогою шифрування

— висока простота використання та адміністрування SQL Server, як і будь-яка інша СУБД, фокусується на базі даних. Дані — це організована сукупність фактів. Часто фізично на жорсткому диску, хоча це не обов'язково. Системи управління базами даних координують і контролюють бази даних.

Реляційна модель використовується MS SQL Server для організації баз даних. Реляційна модель означає, що дані зберігаються в таблицях, що складаються з рядків і стовпців.

Первинний ключ використовується для ідентифікації кожного рядка в таблиці. Один або кілька стовпців можуть виступати в якості первинного ключа. На певний рядок таблиці можна посилатися за допомогою первинного ключа. Два рядки не можуть мати однаковий первинний ключ. Ключі дозволяють зв’язувати одну таблицю з іншою, тобто можна організувати зв’язки між двома таблицями. Саму таблицю можна представити у вигляді відношення.

SQL — це мова, яка використовується для спілкування з базою даних. Клієнт (будь-яка інша програма) надсилає запит за допомогою спеціального API. Потім запит надсилається до СУБД, яка повинна його належним чином зрозуміти та виконати, а потім надіслати результат виконання назад клієнту.

Два типи SQL — це PL-SQL і T-SQL, де СУБД, такі як Oracle і MySQL, використовують PL-SQL. SQL Server використовує T-SQL Transact-SQL.

3.2 Проєктування програмних засобів підсистеми оцінювання якості навчання

Існує багато різноманітних видів і форм архітектури, які успішно використовуються. Одним із найпопулярніших підходів була б проста трирівнева система, яка поділяє програму на три рівні, як показано на рис. 3.4.

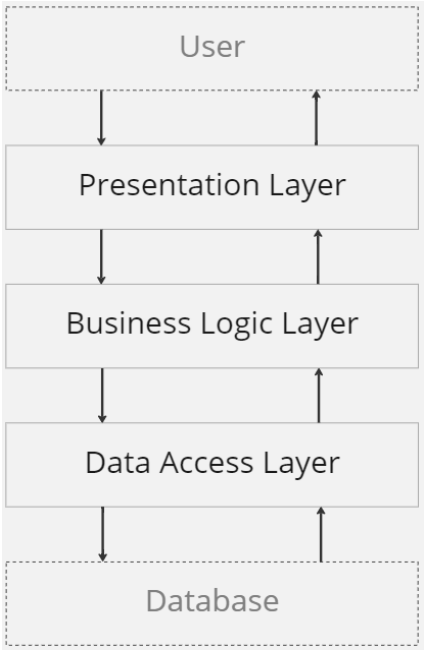

Рисунок 3.4 — Трирівнева архітектура системи

Це клієнт-серверна архітектура, в якій функції представлення, обробки та зберігання розділені. Такий архітектурний шаблон пом'якшує зростаючу складність додатків, спрощує їх більш гнучке масштабування.

Така архітектура складається з різних рівнів, кожен з яких відповідає окремій функції для створення гнучких і багаторазово використовуваних програм. Це значно полегшує весь процес управління системою.

Рівень презентації є першим і найвищим шаром програми; це інтерфейс користувача. Він представляє вміст кінцевому користувачеві через графічний інтерфейс і може бути доступний через будь-який тип клієнтського пристрою.

Сервісний рівень або рівень управління – це рівень бізнес—логіки. Саме на цьому рівні працює бізнес—логіка програми. Бізнес—логіка являє собою набір правил, яким додаток має дотримуватись прийнятих практик роботи. Ці компоненти цього рівня зазвичай працюють на одному або кількох серверах додатків.

Ядро архітектури стосується в основному рівня доступу до даних, що стосується зберігання та видалення даних програми. Наприклад, системні оновлення на цьому рівні не впливають негативно на прикладний рівень цієї архітектури.

Враховуючи вищесказане, система буде розділена на 3 окремі рівні, як показано на рис. 3.5.

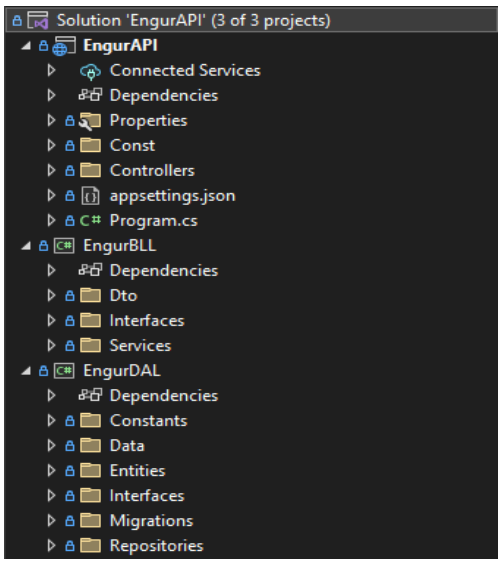

Рисунок 3.5 — Структура файлів системи

Бібліотека EngurDAL dll також містить модель сутності даних. Тут також розміщено конкретні класи та інтерфейси, які працюють із технологіями доступу до бази даних EntityFramework. Шаблон Repository використовується для інкапсуляції логіки роботи з джерелом даних. Він служить абстракцією від конкретних з'єднань джерел даних, які використовуються програмою, і як додатковий зв'язок між класами взаємодії джерел даних та рештою програми. Багато класів репозиторію створено як технічна системна вимога для роботи з кількома сутностями та моделями. Якщо можливо, слід також використовувати шаблон одиниці роботи, оскільки він спрощує роботу з різними репозиторіями та гарантує, що всі репозиторії використовуватимуть той самий контекст даних.

Рівень бізнес-логіки — ще одна бібліотека dll EngurBLL, що містить компоненти, що відповідають за обробку отриманих даних із рівня презентації, виконання програмної логіки та всіх обчислень, роботу з базою даних і передачу результату обробки на рівень презентації. Для кожної сутності створюється окрема служба разом з інтерфейсом і DTO, щоб передача даних між різними рівнями здійснювалася правильно.

Презентаційний рівень реалізовано як додаток Web API EngurAPI — рівень, який торкається користувача через клієнтський додаток. Основним атрибутом цього рівня є контролери. У контролерів зазвичай є методи обробки вхідного HTTP запиту та надсилання відповіді назад клієнту. Деякі методи перевіряються, чи доступні вони, лише якщо користувач певної ролі автентифікований. Крім того, основна конфігурація програми міститься на цьому рівні, де вона включає підключення до бази даних, реєстрацію служб, контролерів, маршрутизацію тощо.

Середовище програмування .NET використовується для проектування бази даних у прямому сенсі за допомогою технології Entity Framework Core. Це взаємодія з СУБД за допомогою сутностей або, скоріше, класів і об'єктів .NET, ніж таблиць бази даних – найвідоміший і функціональний інструмент ORM у C#. ORM має кілька підходів.

Перший підхід — Code First. Для цього підходу дуже важливо визначити класи моделі або сутність, які будуть зберігатися в базі даних, описати їх у класах C# як модель і написати контекстний клас, який працюватиме з використовуваною базою даних. Підхід Code First найчастіше використовується програмістами C# і буде використовуватися в цій роботі.

Другий підхід — це підхід Database First і підходить для тих, хто добре знає SQL; однак у цьому сценарії це не обов'язкова умова для того, щоб добре знати C#. Спочатку створюється база даних, а потім модель бази даних EDMX. Цей XML у файлі .edmx містить відомості про структуру бази даних, модель даних і те, як вони зіставляються один з одним. У Visual Studio є графічний дизайнер, з яким можна працювати .edmx

Model First — це третій підхід ORM, який часто використовують архітектори. При такому підході ви можете не знати синтаксису SQL або C#. Спочатку створюється графічна модель EDMX; у цей час у фоновому режимі створюються класи моделі C#, а потім генерується база даних на основі діаграми EDMX.

Таким чином, у попередній частині був створений каталог для зберігання моделей Entities. Усі таблиці бази даних визначені в Entity Framework як класи моделі або сутності за принципом 1 таблиці. Такі пари називаються угодами, і вони визначаються в класі контексту даних як набори DbSet, і цей підхід працює за замовчуванням.

Створіть новий клас під назвою Word у цій папці, який представлятиме модель слова в системі: публічний клас Word

Модель слова має чотири властивості: унікальний ідентифікатор самого слова, його переклад, і ідентифікатор батьківського модуля.

Інший клас буде створено в папці Entities для представлення модуля — точніше, групи слів, пов'язаних одне з одним: публічний клас Тема

Властивості моделі модуля три: унікальний ідентифікатор, назва модуля та деякий набір слів.

Щоб реалізувати Entity Framework, вам потрібно використовувати менеджер пакетів Visual Studio NuGet [31], щоб інсталювати пакети Microsoft.EntityFrameworkCore, Microsoft.EntityFrameworkCore.SqlServer і Microsoft.EntityFrameworkCore.Tools для проекту (рис. 3.6).

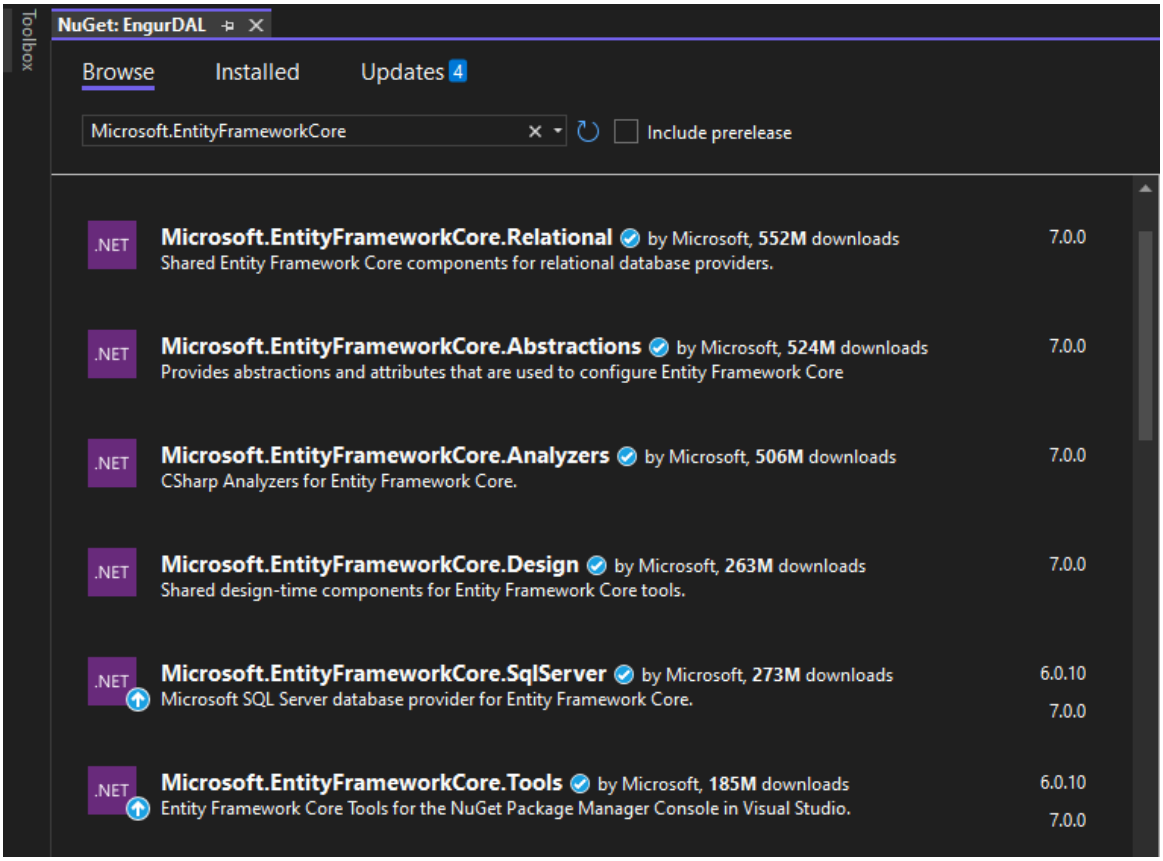

Рисунок 3.6 — Менеджер пакетів NuGet

Для зв’язку з базою даних ми використовуємо контекст даних. Щоб створити контекст, ми повинні успадкувати новий клас не просто від будь—якого класу, а від класу DbContext. Такі властивості, як public DbSet Words { get; комплект; } дозволяють отримати набір даних певного типу з бази даних (наприклад, набір об’єктів Word).

Для підключення до бази даних необхідно встановити параметри підключення. Це робиться шляхом зміни файлу appsettings.json, який за замовчуванням містить лише параметри журналювання; ми змінюємо його, додаючи визначення рядка підключення (рис. 3.7).

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "EngurDbConnection": "Server=(localdb)\\mssqllocaldb;Initial Catalog=Engur"
  }
}
```

Рисунок 3.7 — Файл appsettings.json

У цьому прикладі ми будемо використовувати LocalDB, будучи тонким випуском SQL Server Express, призначеним лише для розробки програм. Як зазначено параметром Server=(localdb)\mssqllocaldb. Добре, сама база даних буде називатися Engur.

Налаштування проекту завершується реєстрацією контексту проекту DbContext у файлі Program.cs. Параметр рядка підключення до бази даних замінено на параметр, указаний у файлі appsettings.json:

```
builder.Services.AddDbContext<EngurDbContext>(opt => opt.UseSqlServer(
builder.Configuration.GetConnectionString("EngurDbConnection")));
```

Створену базу даних можна побачити через MS SQL Server Management Studio (як показано на рис. 3.8).

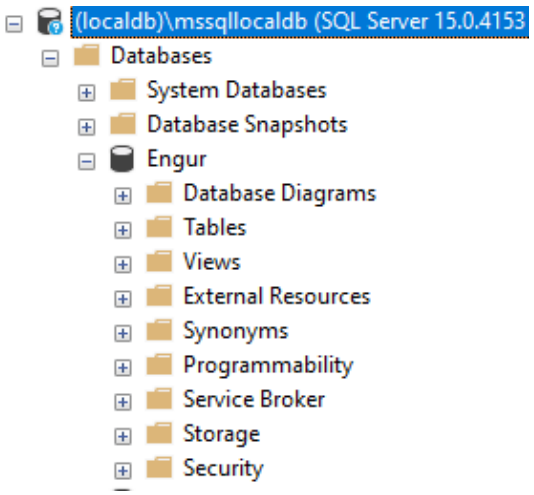

Рисунок 3.8 — База даних Engur

У процесі розробки дуже ймовірно, що зі зміною класу моделі Entity Framework базу даних доведеться видалити, щоб зберегти узгодженість моделі. Однак, коли базу даних буде видалено, усі її дані будуть втрачені.

Щоб зберегти дані у разі зміни моделі, Entity Framework Core має функцію міграції, яка дозволяє застосовувати зміни бази даних узгоджено, щоб узгодити їх із моделлю даних.

Міграції мають операції, які дозволяють видаляти, додавати стовпці та таблиці, зовнішні ключі, змінювати параметри стовпців тощо. Додавати, видаляти, змінювати дані тощо. Під час створення міграції автоматично створюється клас, у якому виконуються операції, необхідні для застосування міграції Up та її методу повернення Down

На цьому налаштування бази даних завершено, і програма має безумовний доступ до бази даних. У разі зміни вимог і необхідності додавання нових моделей буде застосовано механізм міграції.

Відповідно до обраної архітектури система була розділена на 3 логічні рівні — рівень доступу до даних, рівень бізнес-логіки та рівень презентації. Розробка традиційно починається з базового рівня доступу до даних.

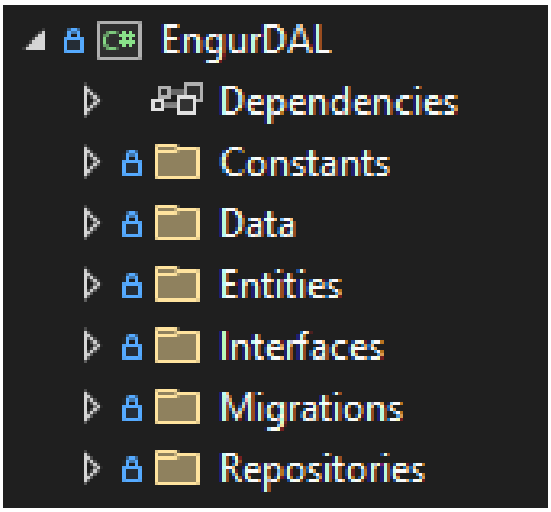

Рисунок 3.9 — Структура рівня DAL

Папка «Константи» має містити дані, які мають залишатися постійними протягом програми, доступними в будь-який час із класу будьякого рівня.

Папка Data повинна містити контекст даних DbContext, детально розглянутий у попередньому розділі, разом із класом UnitOfWork.

Папка Entities має містити лише ключові на цьому рівні моделі даних сутності, розробку яких було описано в попередньому розділі.

Папка Interfaces має містити лише контракти класів.

Міграції мають включати міграції баз даних, пов'язані зі зміною моделей даних і внесенням змін до існуючої бази даних.

Папка Repositories повинна містити класи сховища даних, які детально описують дані.

Патерн Repository використовується для інкапсуляції логіки роботи з базою даних. Додатковий шаблон Unit of Work допомагає спростити роботу з різними сховищами, щоб вони використовували той самий DbContext.

Крім того, використання шаблонів Repository та Unit of Work дозволяє створити правильну структуру для розгортання додатків і впровадження практик DI, що також допомагає при тестуванні проекту [30] (рис. 3.10).

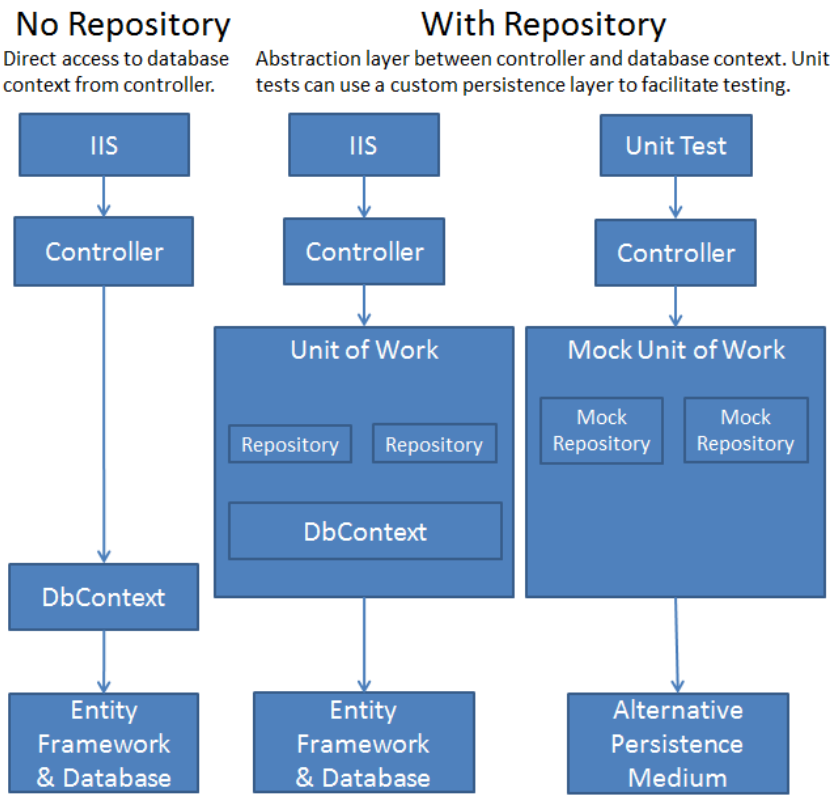

Рисунок 3.10 — Використання патерну Repository та Unit of Work

Приклад сутності Word опише правила, за якими створюються ці дизайни в системі. Будівництво почнеться з наступного інтерфейсу: IWordRepository : IDisposable.

Описаний інтерфейс містить набір методів CRUD, які знадобляться для роботи з базою даних; тобто створити, прочитати, оновити та видалити. Наступним логічним кроком буде реалізація цього інтерфейсу в класі WordRepository : IWordRepository з відповідними елементами. Не потрібно втрачайти жодної інформації під час переписування. Не додавайте, не видаляйте та не повторюйте жодної інформації з оригінального тексту. Не перевищуйте кількість слів оригінального тексту. Не включайте відступи або кліше, якщо вони вже не присутні в оригінальному тексті. Не закінчуйте фрагменти речень знаками питання, якщо знаки питання вже не присутні в оригінальному тексті.

Описаний клас отримує контекст даних через конструктор, має асинхронні методи CRUD з інтерфейсу та включає метод Dispose, який діє як служба, яка допомагає у звільненні некерованих ресурсів збирачем сміття C# [29].

На даний момент створений клас вже можна повністю використовувати в сервісах і контролерах, але з огляду на майбутнє масштабування системи з додаванням нових моделей даних, було б доцільно об’єднати всі репозиторії в одному місці, тому саме для цього введено шаблон Unit Of Work – публічний інтерфейс IUnitOfWork. Інтерфейс повинен мати властивість усіх сховищ, доступних у системі, і сигнатуру асинхронного методу SaveChangesAsync, який буде використовуватися для виконання транзакції. Останнім кроком реалізації буде клас Unit Of Work.

Цей клас має приватні поля для єдиного контексту бази даних і репозиторіїв моделі даних, до яких він отримує доступ через загальнодоступні властивості. Він також реалізує методи служби SaveChangesAsync() і Dispose(), описані раніше.

Рівень бізнес-логіки — це служба, яка діє як посередник між даними та їх представленням. Тут дані отримуються з логічного рівня даних і далі передаються

на логічний рівень презентації, а також, якщо потрібні операції над даними. Він має просту, самоописову структуру (рис. 3.11).

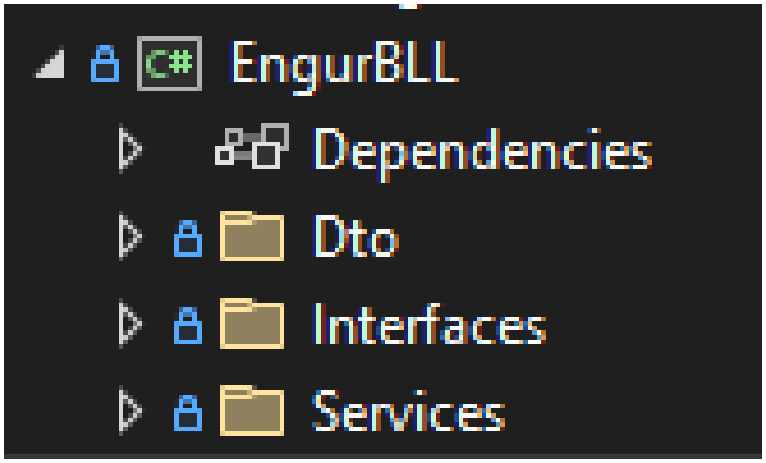

Рисунок 3.11 — Структура рівня BLL

У папці Dto зберігаються моделі об’єктів передавання даних.

Використовуйте папку Interfaces для зберігання контрактів на обслуговування.

Використовуйте папку Services для зберігання класів обслуговування з проміжними операціями з даними.

Модель зберігання даних не зовсім відповідає моделі представлення даних і навпаки. Рішення тут може зберігати всю необхідну для дзвінка інформацію за допомогоюкласу клас WordDto:

Екземпляр сервісу для роботи з сутністю Topic вкаже принципи, за якими сервіси формуються в системі. Розробку слід починати з інтерфейсу: ITopicService.

Цей інтерфейс повторює методи CRUD, описані в репозиторії, і реалізує на їх основі нові складні методи. Наприклад, він поверне не лише сам модуль, але й пов’язані з ним слова для одного ідентифікатора. Принцип реалізації в сервісі подібний до репозиторію, але додатково виконується перетворення даних з рівня зберігання даних на рівень представлення даних.

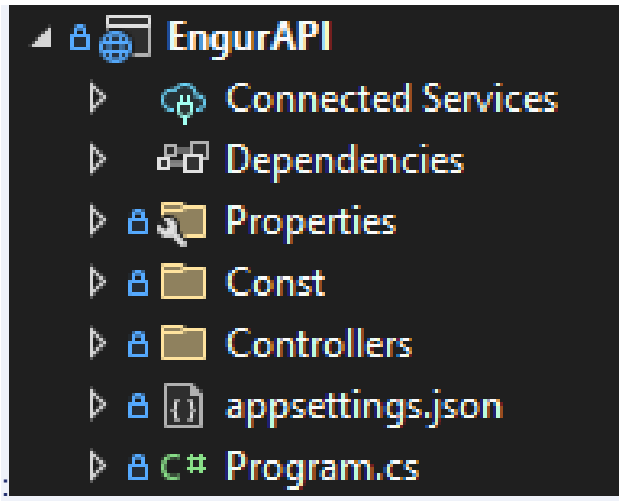

Рисунок 3.12 — Структура рівня PL

Папка Properties має зберігати файли конфігурації, як launchSettings.json. Цей файл містить властивості запуску програми.

Файли, які не змінюватимуться під час роботи програми — і до яких можна отримати доступ у будь-який час, з будь-якого рівня класу — мають міститися в папці Const.

Зверніть увагу на клас Program.cs. Про це згадувалося раніше під час реєстрації контексту бази даних. Як правило, цей клас важливий, оскільки він реєструє та будує всі служби, задіяні в системі, а також метод запуску програми.

Клас походить від ControllerBase і є базовим API, який розширюється методами для обробки запитів, пов’язаних із сутністю Word. Перший рядок тут встановлює загальний маршрут, за яким цей контролер буде доступним. Кожен метод повинен мати унікальний маршрут і може відповідати на певний тип запиту: Get, Put, Post, Delete. Описуючи додавання абсолютно нової сутності, цей контролер візьме модель у форматі Data Transfer Object від клієнта та передасть її до служби BLL, яка потім перетворить об’єкт на модель для зберігання даних і передасть її на рівень DAL для запису в БД.

Таким чином, підсистема розділена на три логічні рівні, які обробляються трьома окремими проектами: Рівень доступу до даних (DAL) і Рівень бізнес-логіки (BLL) є рівнями, які мають форму бібліотек DLL, тоді як Рівень представлення (PL

або API) є проектом Web API, оскільки це основний виконуваний проект і інтерфейс із системою (рис. 3.13).

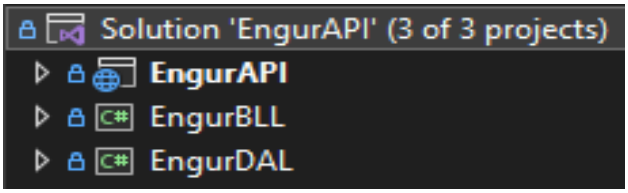

Рисунок 3.13 — Структура проєктів системи

На рівні доступу до даних даними може бути будь-яка база даних або будь-який файл, оскільки цей рівень є єдиним зв’язком між вашою програмою та цими даними.

На рівні бізнес-логіки — перетворення даних з одного формату в інший, а також часто деякі обчислювальні операції.

Рівень презентації — це інтерфейс, через який клієнти спілкуються з ядром вашої системи.

Шари не взаємодіють, оскільки не знають один про одного. Щоб додати проект до сфери іншого проекту, потрібно перейти до контекстного меню проекту та вибрати «Додати», «Посилання на проект».

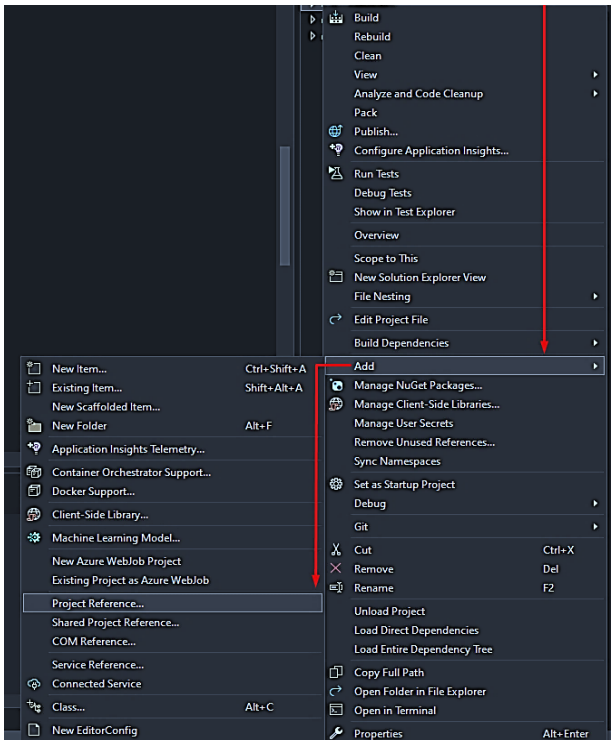

Рисунок 3.14 — Контекстне меню проєкту

Правило цього дизайну полягає в тому, що нижчий рівень не знає, що існує вищий, тому ви можете повторно використовувати код без необхідності писати його ще раз. Відповідно до цього правила рівень DAL не повинен містити посилань на інші проекти. Нарешті, для проекту PL не потрібно додавати посилання на всі рівні, тому що в цьому проекті зареєстровані служби всіх рівнів, такі як контекст даних для рівня DAL або служби рівня BLL (рис. 3.15).

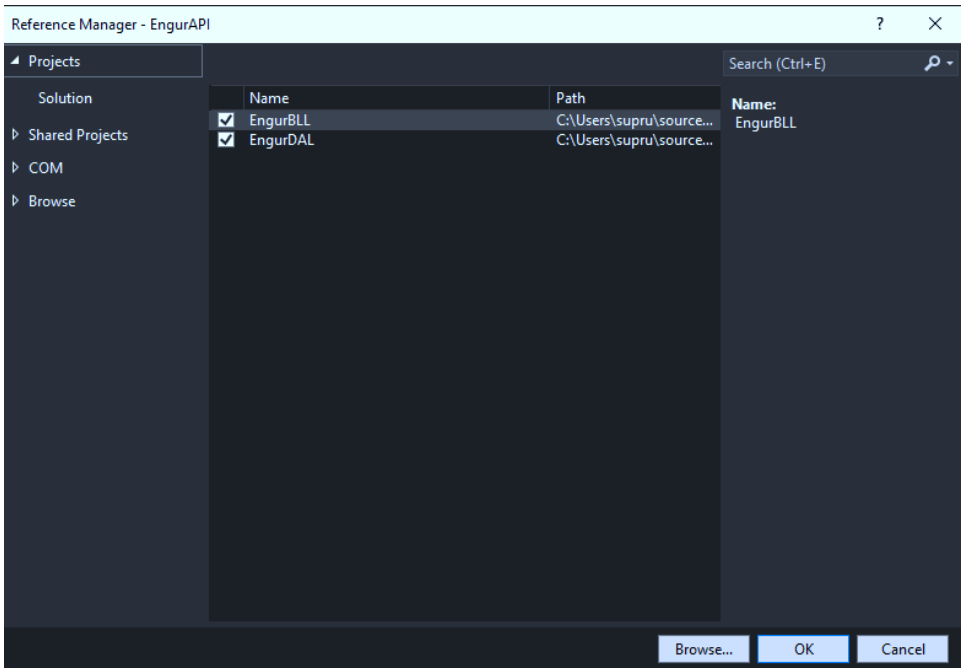

Рисунок 3.15 — Додавання посилань на інші проєкти

3.3 Тестування програмного продукту

Тестування означає перевірку функціонування всієї системи шляхом застосування різних способів і засобів, щоб побачити, чи вона працює правильно. Аналіз, як нових, так і існуючих систем, повинен бути проведений для забезпечення продуктивності ресурсів і підвищення їх ефективності.

Для тестування системи були застосовані наступні види тестування: функціональні та нефункціональні, в ручному та автоматичному режимах.

Функціональне тестування проводиться для перевірки реалізації функціональних системних критеріїв, які є здатністю системи виконувати

поставлені перед нею завдання в заданих умовах. Зазвичай для тестування потрібні такі критерії:

— безпека;

— відповідність вимогам;

— сумісність;

— точність.

Переваги функціонального тестування включають широке охоплення різних функціональних тестів і те, як користувач імітує перевірену систему. Мінуси полягають у тому, що це призводить до надмірності в тестуванні та може викликати логічні помилки в коді.

Нефункціональне тестування — це, в основному, перевірка того, як працює система, де перевіряються наступні вимоги на відповідність:

— юзабіліті – оцінка зручності користувача;

— продуктивність – швидкодія системи при різних навантаженнях;

— безпека (захист даних);

— надійність (як система поводиться, коли щось йде не так і можливість керувати нестандартними діями користувачів).

3 окремих користувача взяли участь у тестуванні системи та допомогли знайти проблеми з налаштуванням функцій програми для правильної роботи.

Система була протестована на основі таких моментів:

— функціональність (працездатність контролерів API);

— коректне відображення контенту;

— нові дані належним чином записані в систему;

— відповідність функціонування підсистеми оцінювання якості освіти;

— кросбраузерність і масштабованість; перевірено в найпопулярніших веб-браузерах;

— зручність використання;

— перевірено, чи зручно користуватися веб-системою простим користувачам.

4 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Аналіз результатів, отриманих на базі запропонованого ПЗ

Проаналізуємо роботу системи. Так наприклад користувач почав вибирати завдання (див. рис. 4.1). Система контролю якості навчання запускається.

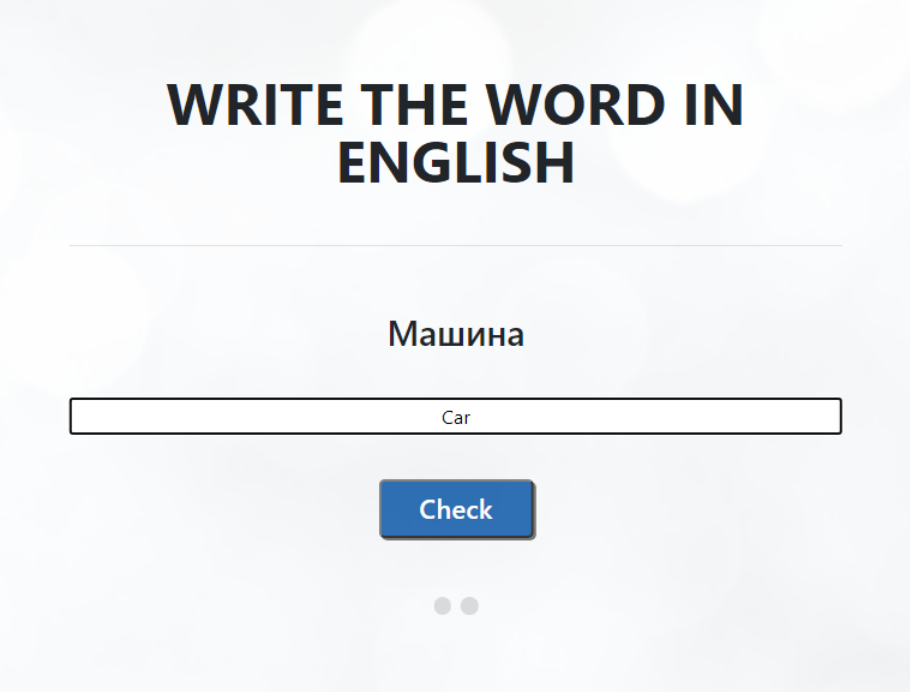

Рисунок 4.1 — Тестове завдання для перевірки знань

Завершивши тест, збираються дані і обчислюються показники з другого розділу дипломної роботи. Спочатку обчислюється показник R.

Перевірка роботи моделі здійснюється на юніт тестах (рис. 4.2).

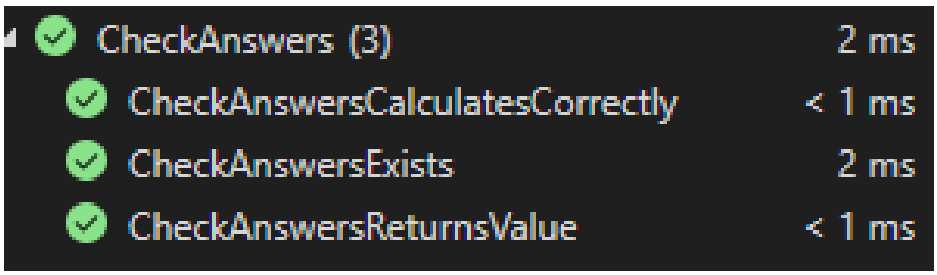

Рисунок 4.2 — Результат виконання юніт тестів

Вікно результатів наведено на (рис. 4.3).

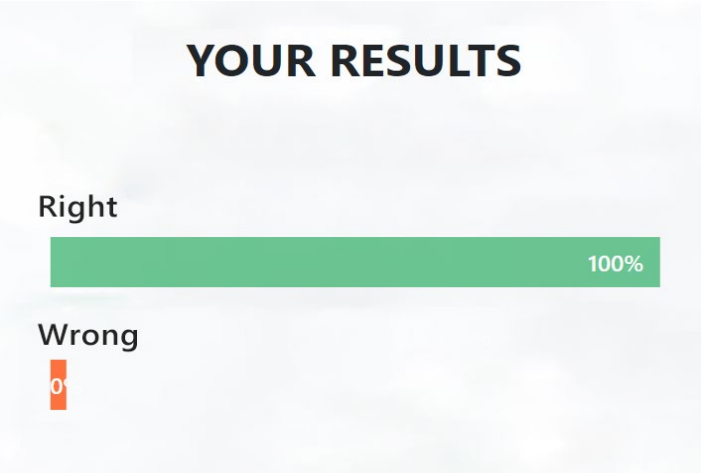

Рисунок 4.3 — Результат виконання тестового завдання

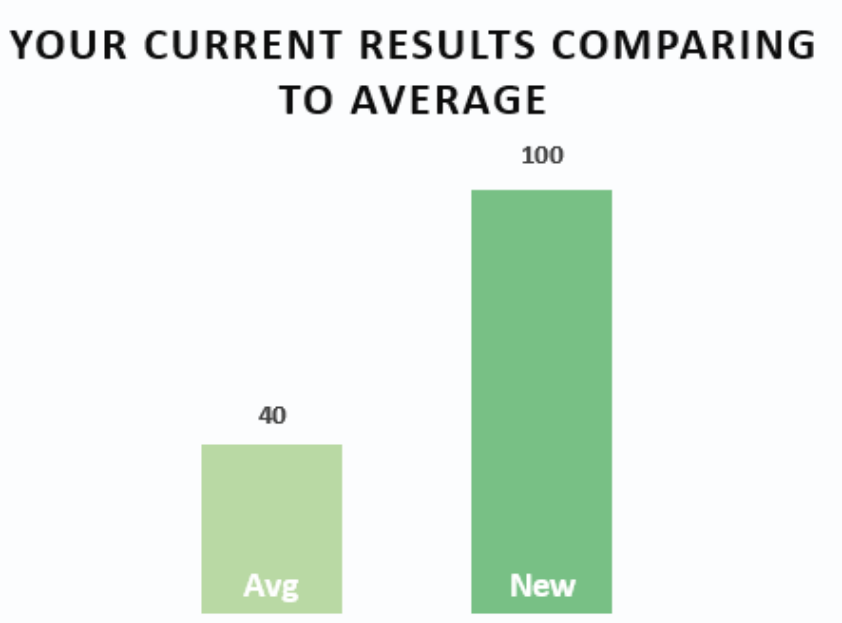

Рисунок 4.4 — Результат виконання перевірки виконання завдання

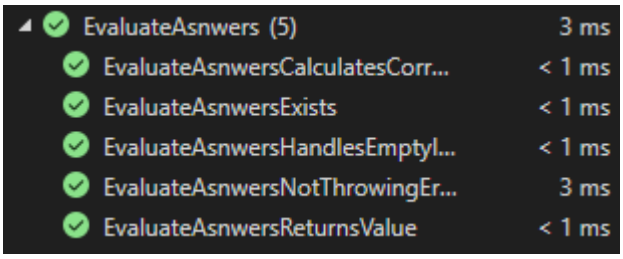

Рисунок 4.4 — Результат виконання юніт тестів

Результат роботи наведений на рис. 4.5.

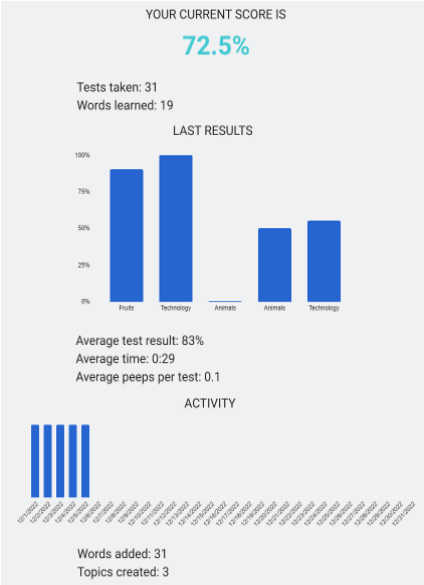

Рисунок 4.5 — Результати навчання у статистиці користувача

4.2 Керівництво оператора

Веб-програми для вивчення англійських слів допомагають не тільки зберегти слова, але й збільшити їх кількість. Додана частина покращує роботу системи. Перевіряючи, що користувач знає, він враховує більше речей, що дозволяє краще перевірити свої мовні навички.

На початку роботи завантажується головна сторінка з формою авторизації. Якщо користувач ще не має облікового запису, він може створити, натиснувши кнопку реєстрації

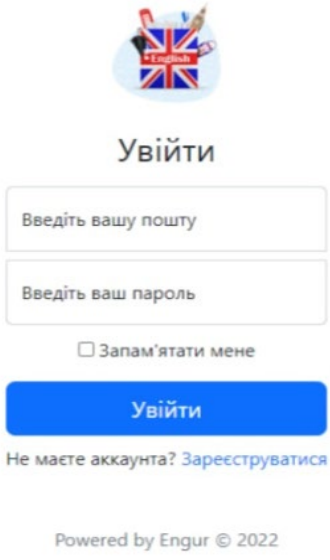

Рисунок 4.6 — Форма введення даних для входу

Натиснувши кнопку «Зареєструватися», користувач потрапляє на сторінку реєстрації (див. рис. 4.7), де необхідно ввести адресу електронної пошти та пароль. Система автоматично перевіряє правильність даних, а також перевіряє, чи є вже зареєстрований обліковий запис з тією самою електронною поштою.

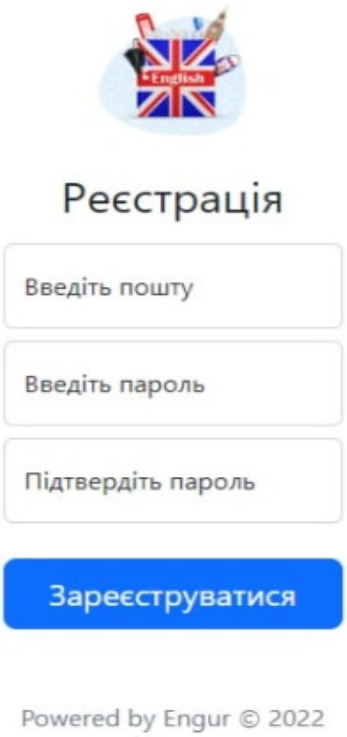

Рисунок 4.7 — Форма введення даних для реєстрації

ВИСНОВКИ

Протягом всієї дипломної роботи робота була зосереджена на розробці веб-додатку для дистанційного навчання англійської мови з інтегрованою системою оцінювання знань

Дослідження виявило наявність певних недоліків в існуючих методах контролю результатів навчання, що потребувало розробки відповідного програмного забезпечення. Детальний аналіз комп'ютерних систем оцінки якості дистанційної освіти дозволив автору сформувати технічні вимоги до цієї платформи.

Архітектура системи складається з трьох взаємопов'язаних компонентів: модуля тестування, блоку оцінювання та системи зберігання результатів, які функціонують послідовно, щоб забезпечити комплексний підхід до контролю знань.

Це двопараметрична модель Бірнбаума, найбільш вдала серед альтернатив. Модель тестування побудована на лінійних принципах; його результати є вхідними даними для алгоритму оцінювання.

Технологічний стек включає Angular для зовнішньої частини, .NET для внутрішньої частини та Microsoft SQL Server для зберігання даних; це повинно забезпечити стабільну роботу з можливістю майбутнього розширення.

БД розроблено в першій нормальній формі (1NF), і нею можна керувати як програмно, так і через SQL Management Studio. Архітектура допускає горизонтальне масштабування шляхом додавання нових модулів.

Основними перевагами платформи є чотири типи перевірки знань, різні форми завдань, детальна аналітика з візуалізацією статистики, взаємодія з викладачем та відсутність комерційних обмежень.

Всі поставлені цілі успішно досягнуті, створений продукт готовий до практичного використання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Система дистанційної колективної самопідготовки / О. Д. Азаров та ін. *Інформаційні технології та комп'ютерна інженерія*. 2016. № 2. Т. 2. С. 15–20.

2. Биков В. Ю. Теоретико—методологічні засади моделювання навчального середовища сучасних педагогічних систем : зб. наук. праць. Київ : Інститут засобів навчання АПН України, 2005. 272 с.

3. Simonson M., Zvacek S. M., Smaldino S. Teaching and Learning at a Distance: Foundations of Distance Education. 7th Ed. Charlotte : Information Age Publishing, 2019. 366 p.

4. Маркова Є. С. Інформаційні технології навчання : навч. посіб. Запоріжжя : Просвіта, 2012. 118 с.

5. Інформаційні та комунікаційні технології навчання в системі загальної середньої освіти зарубіжних країн / Гриценчук О. О. та ін. Київ : Пед. думка, 2012. 176 с.

6. Супрун П. Б. Особливості розробки веб—додатків на ASP.NET Core. *LI науково—технічна конференція ВНТУ* : Електронне наукове видання матеріалів конференції, м. Вінниця, 2022. URL: https://conferences.vntu.edu.ua/index.php/all—fitki/all—fitki—2022/paper/view/14483/12251

7. Ткаченко Л. В., Хмельницька О. С. Особливості впровадження дистанційного навчання в освітній процес закладу вищої освіти. *Педагогіка формування творчої особистості у вищій і загальноосвітній школах*. 2021. № 75. Т. 3. С. 91–96.

8. Організація дистанційного навчання. Створення електронних навчальних курсів та електронних тестів / Вишнівський В. В. та ін. Київ : ДУТ, 2014. 140 с.

9. Дистанційне навчання. Як організувати навчання вдома і не зійти з розуму / Вайзман Р. та ін. Київ : Альпіна паблішер, 2021. 240 с.

10. Буйницька О. Інформаційні технології та технічні засоби навчання : навч. посіб. Київ : Центр навч. літератури, 2019. 240 с.

11. Memrise – онлайн сервіс для вивчення слів іноземних мов. *Memrise* : веб—сайт. URL: https://www.memrise.com/

12. Lingualeo – онлайн сервіс для вивчення англійської та інших іноземних мов. *Lingualeo* : веб—сайт. URL : https://lingualeo.com/

13. Duolingo – самий швидкий шлях вивчити іноземну мову. *Duolingo* : веб—сайт URL : https://www.duolingo.com/

14. Винарчук Т. М. Веб—статистика як інструмент аналізу сайту. *Народна освіта*. 2014. № 23, Т. 6. URL: https://www.narodnaosvita.kiev.ua/?page_id=2424

15. Bliss K. Libertiny J. What is Mathematical Modelling. 2th Ed. COMAP & SIAM, 2019. 236 p.

16. Yodgorov G., Jurakulov T. Mathematical modeling of learning processes based on the theory of control. *Conference Proceedings*. 2021. № 1, Vol. 2365. URL: https://aip.scitation.org/doi/10.1063/5.0057821 .

17. Zhelezniakova E., Silichova T. Mathematical modeling as a means of control and evaluation of the quality of the educational process. *Modestum.* 2020. № 2, Vol. 16. URL: http://repository.hneu.edu.ua/bitstream/123456789/25791/1/Zhelezniakova_Silichova%20%281%29.pdf

18. Sun B. Mathematical Models of Learning Efficiency. *Modestum*. 2017. № 7, Vol. 13. P. 4261–4270.

19. Ivanichkina L., Neporada A. Mathematical Methods and Models of Improving Data Storage Reliability Including Those Based on Finite Field Theory. *Contemporary Engineering Sciences*. 2014. № 28, Vol. 7. P. 1589–1602.

20. An introduction to web development technologies. *Tiller* : website. URL: https://tillerdigital.com/blog/an—introduction—to—web—development—technologies/

21. ng—book: The Complete Guide to Angular 4. Coury F. and oth. 4th Ed. Amazon Digital Services, 2017. 597 p.

22. Richter J. CLR via C# (Developer Reference). Microsoft Press, 2012. 896 p.

23. Mark J. Price. C# 11 and .NET 7 – Modern Cross—Platform Development Fundamentals: Start building websites and services with ASP.NET Core 7, Blazor, and EF Core 7. Packt Publising, 2022. 818 p.

24. Мартін Р. Чиста Архітектура. Київ : Фабула, 2019. 416 р.

25. Прайс М. C# 9 та .NET 5. Розробка та оптимізація. Packt, 2021. 832 с.

26. Hughes A. Microsoft SQL Server. *TeachTarget* : website. URL: https://www.techtarget.com/searchdatamanagement/definition/SQL—Server/

27. Багаторівнева архітектура. *Metanit* : веб—сайт. URL: https://metanit.com/sharp/mvc5/23.5.php/ .

28. Entity Framework Core. *Microsoft* : website. URL: https://learn.microsoft.com/ru—ru/ef/core/

29. Quickstart: Install and use a NuGet package in Visual Studio. *Microsoft* : website. URL: https://learn.microsoft.com/en—us/nuget/quickstart/install—and—use—a—package—in—visual—studio/

30. Фрімен Е., Робсон Е. Характеристики Head First. Патерни проєктування. Київ : Фабула, 2021. 688 с.

31. Скільки шкіл в Україні готові до очного навчання: відповідь міністра освіти. *ТСН* : веб—сайт. URL: https://tsn.ua/ukrayina/skilki—shkil—v—ukrayini—gotovi—do—ochnogo—navchannya—vidpovid—ministra—osviti—2143093.html/

ДОДАТОК А

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
проф., д.т.н. О. Д. Азаров

« » червня 2025 року

ТЕХНІЧНЕ ЗАВДАННЯ
на виконання бакалаврської кваліфікаційної роботи
«Система оцінювання навчальних параметрів
студента при дистанційному навчанні»
08-54.БКР.032.00.000 ТЗ

Науковий керівник к.т.н., доц. каф. ОТ
Снігур А.В.

Студента групи 2КІ-21б
Соловйов Д.Ю.

м. Вінниця — 2025

1 Підставою для виконання бакалаврської кваліфікаційної роботи (БКР)

1.1 Важливим є актуальність дослідження у напрямку бакалаврської роботи, яка обумовлена тим, що в сучасному світі інформаційні технології та Інтернет відіграють вирішальну роль у функціонуванні та розвитку бізнесу. Зокрема, електронна комерція стала одним із основних способів ведення торгівлі що дозволяє компаніям охоплювати глобальні ринки, знижувати витрати та підвищувати ефективність продажів. Інтернет—магазини серверного обладнання, як специфічний сегмент електронної комерції, стають все більш популярними завдяки зростаючому попиту на високотехнологічні рішення для бізнесу та приватних осіб;

2 Наказ про затвердження теми БКР по ВНТУ № 80 від 06.02.2024 р.

Мета БКР та призначення розробки.

Метою бакалаврської кваліфікаційної роботи є аналіз та розробка інформаційної системи для інтернет—магазину серверного обладнання.

Призначення розробки — конкурентоспроможний інтернет магазин з продажу серверного обладнання.

3 Вихідні дані для виконання БКР

3.1 Проведення аналізу існуючих принципів та технологій інформаційних систем та інтернет магазинів;

3.2 Розробка вебсайту та інформаційної системи для інтернет магазину серверного обладнання;

4 Вимоги до виконання БКР

Головна вимога — розробка та впровадження інформаційної системи для інтернет-магазину, яка забезпечить ефективне управління процесами продажу, автоматизацію обробки замовлень та покращення якості обслуговування клієнтів.

5 Етапи БКР та очікувані результати

Етапи роботи та очікувані результати приведено в Таблиці А.1.

№ етапу	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи	Приміт—ка
1	Постановка задачі роботи	21.03.25	вик.
2	Аналіз аналогів	23.03.25	вик.
3	Аналіз сучасних підходів до інформаційних систем	27.03.25	вик.
4	Аналіз поняття інформаційна система	30.03.25	вик.
5	Аналіз принципів роботи інформаційних систем	05.04.25	вик.
6	Проектування інформаційної системи	09.04.25	вик.
7	Розробка інформаційної системи	14.04.25	вик.
8	Розробка функціональності інформаційної системи	17.04.25	вик.
9	Підготовка матеріалів та опис розробки інформаційної системи	25.04.25	вик.
10	Аналіз виконання роботи, висновки, додатки	10.05.25	вик.
11	Перевірка якості виконання бакалаврської роботи та усунення недоліків	13.05.25	вик.

6 Матеріали, що подаються до захисту БКР

До захисту подаються: пояснювальна записка БКР, графічні і ілюстративні матеріали, протокол попереднього захисту БКР на кафедрі, відгук наукового керівника, відгук опонента, протоколи складання державних екзаменів, анотації до БКР українською та іноземною мовами, довідка про відповідність оформлення БКР діючим вимогам.

7 Порядок контролю виконання та захисту БКР

Виконання етапів графічної та розрахункової документації БКР контролюється науковим керівником згідно зі встановленими термінами. Захист

БКР відбувається на засіданні Екзаменаційної комісії, затвердженої наказом ректора

8 Вимоги до оформлення та порядок виконання КБКР

8.1 При оформленні БКР використовуються:

— ДСТУ 3008 : 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302 : 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів спеціальності 123 «Комп'ютерна інженерія» (освітня програма «Комп'ютерна інженерія») — кафедра обчислювальної техніки, ВНТУ 2023.

8.2 Порядок виконання БКР викладено в «Положення про кваліфікаційні роботи на першому (бакалаврському) рівні вищої освіти СУЯ ВНТУ—03.02.02—П.001.01:21»

ДОДАТОК Б

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Система оцінювання навчальних параметрів студента при дистанційному навчанні
Тип роботи: бакалаврська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)
Підрозділ _____кафедра обчислювальної техніки_____
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism __5,5___%

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

✓ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

_____Азаров О.Д., зав. каф. ОТ _____ _____
(прізвище, ініціали, посада) (підпис)
____Крупельницький Л.В., доц. каф ОТ_____ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ ___Захарченко С.М.____
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ ____ к.т.н., доц. кафедри ОТ Снігур А.В.____
(підпис) (прізвище, ініціали, посада)

Здобувач _____ _____Соловйов Д.Ю._____
(підпис) (прізвище, ініціали)

ДОДАТОК В

ІЛЮСТРАТИВНА ЧАСТИНА

СИСТЕМА ОЦІНЮВАННЯ НАВЧАЛЬНИХ ПАРАМЕТРІВ СТУДЕНТА ПРИ ДИСТАНЦІЙНОМУ НАВЧАННІ

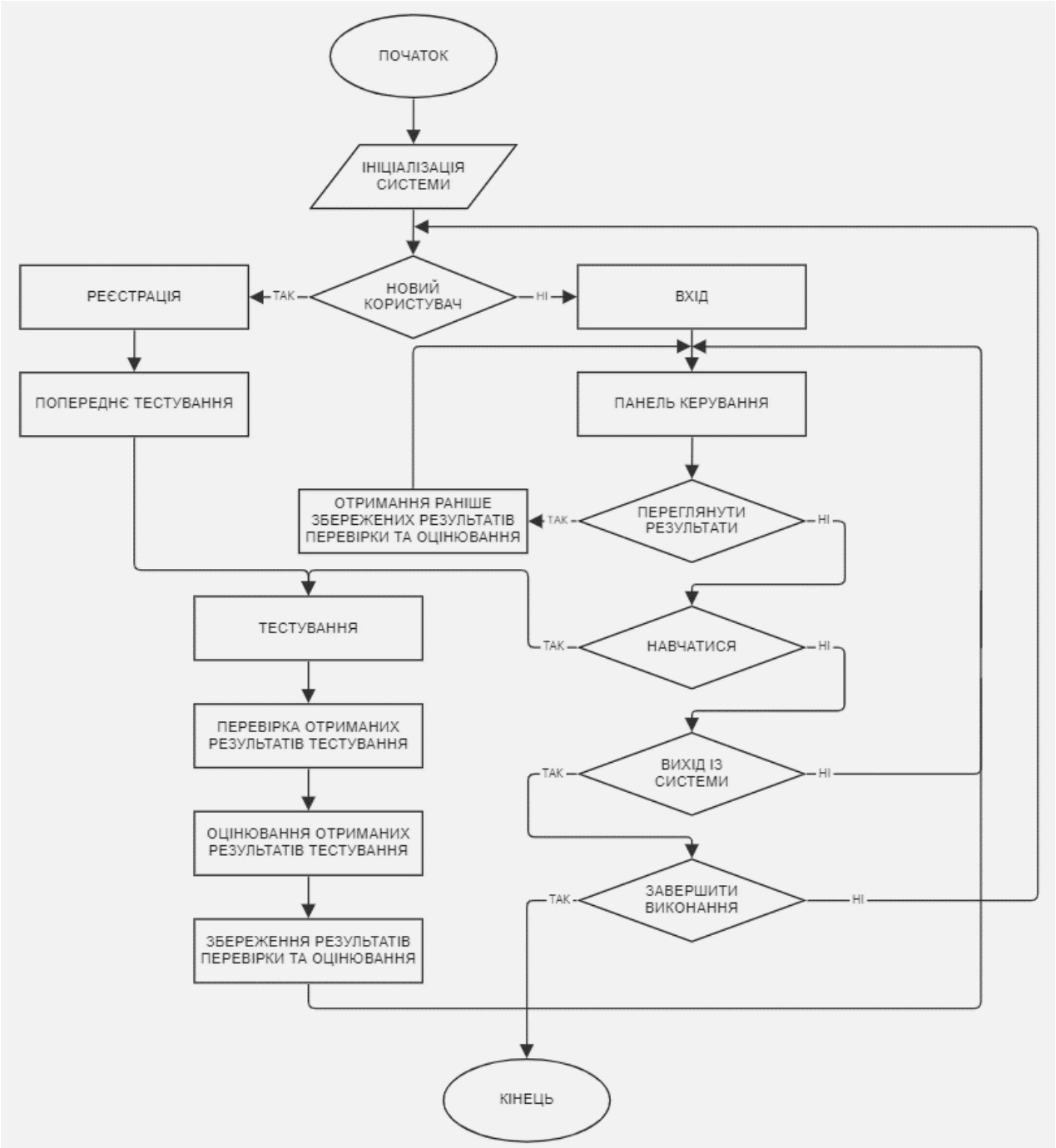

Рисунок В.1 — Алгоритм роботи інтегрованої підсистеми оцінювання якості навчання

ДОДАТОК Г

Лістинг програми

```
public interface IWordRepository : IDisposable
    {
        Task<Word?> GetByIdAsync(int id);
        Task<IEnumerable<Word>> GetAllAsync();
        Task CreateAsync(Word entity);
        Task DeleteByIdAsync(int id);
        void Update(Word entity);
    }
public interface ITopicRepository : IDisposable
    {
        Task<Topic?> GetByIdWithWordsAsync(int id);
        Task<IEnumerable<Topic>> GetAllAsync();
        Task CreateAsync(Topic entity);
        Task DeleteByIdAsync(int id);
        void Update(Topic entity);
    }
public class WordRepository : IWordRepository
    {
        private readonly EngurDbContext _context;
        public WordRepository(EngurDbContext context)
        {
            _context = context;
        }
        public async Task CreateAsync(Word word)
        {
            if (word == null)
            {
                throw new ArgumentNullException(nameof(word));
            }
            await _context.Words.AddAsync(word);
        }
        public async Task DeleteByIdAsync(int id)
        {
            var word = await _context.Words.FirstOrDefaultAsync(w => w.Id == id);
            if (word != null)
            {
                _context.Words.Remove(word);
            }
        }
        public async Task<IEnumerable<Word>> GetAllAsync()
```

```
        {
            return await _context.Words.ToListAsync();
        }
        public async Task<Word?> GetByIdAsync(int id)
        {
            return await _context.Words.FirstOrDefaultAsync(t => t.Id == id);
        }
        public void Update(Word word)
        {
            _context.Words.Update(word);
        }
        public void Dispose()
        {
            _context.Dispose();
        }
    }
public class TopicRepository : ITopicRepository
    {
        private readonly EngurDbContext _context;
        public TopicRepository(EngurDbContext context)
        {
            _context = context;
        }
        public async Task CreateAsync(Topic topic)
        {
            if (topic == null)
            {
                throw new ArgumentNullException(nameof(topic));
            }
            await _context.Topics.AddAsync(topic);
        }
        public async Task DeleteByIdAsync(int id)
        {
            var topic = await _context.Topics.FirstOrDefaultAsync(t => t.Id == id);
            if (topic != null)
            {
                _context.Topics.Remove(topic);
            }
        }
        public async Task<IEnumerable<Topic>> GetAllAsync()
        {
            return await _context.Topics.ToListAsync();
        }
        public async Task<Topic?> GetByIdWithWordsAsync(int id)
```

```
        {
            return await _context.Topics
                .Include(t => t.Words)
                .FirstOrDefaultAsync(t => t.Id == id);
        }
        public void Update(Topic topic)
        {
            _context.Topics.Update(topic);
        }
        public void Dispose()
        {
            _context.Dispose();
        }
    }
public interface IWordService

    {
        Task<WordDto?> GetByIdAsync(int id);
        Task<IEnumerable<WordDto>> GetAllAsync();
        Task CreateAsync(WordDto word);
        Task DeleteByIdAsync(int id);
        Task UpdateAsync(WordDto word);
    }
public interface ITopicService
    {
        Task<TopicDto?> GetByIdWithWordsAsync(int id);
        Task<IEnumerable<TopicDto>> GetAllAsync();
        Task CreateAsync(TopicDto word);
        Task DeleteByIdAsync(int id);
        Task UpdateAsync(TopicDto word);
    }
public class WordService : IWordService
    {
        private readonly IUnitOfWork _unitOfWork;
        public WordService(IUnitOfWork uow)
        {
            _unitOfWork = uow;
        }
        public async Task CreateAsync(WordDto word)
        {
            await
_unitOfWork.WordRepository.CreateAsync(WordDto.MapFromDto(word));
            await _unitOfWork.SaveChangesAsync();
        }
```

```
        public async Task DeleteByIdAsync(int id)
        {
            await _unitOfWork.WordRepository.DeleteByIdAsync(id);
            await _unitOfWork.SaveChangesAsync();
        }
        public async Task<IEnumerable<WordDto>> GetAllAsync()
        {
            var words = await _unitOfWork.WordRepository.GetAllAsync();
            return WordDto.MapToDtos(words);
        }
        public async Task<WordDto?> GetByIdAsync(int id)
        {
            var word = await _unitOfWork.WordRepository.GetByIdAsync(id);
            if (word == null)
            {
                throw new ArgumentException(nameof(word));
            }
            return WordDto.MapToDto(word);
        }
        public async Task UpdateAsync(WordDto word)
        {
            _unitOfWork.WordRepository.Update(WordDto.MapFromDto(word));
            await _unitOfWork.SaveChangesAsync();
        }
    }
public class TopicService : ITopicService
    {
        private readonly IUnitOfWork _unitOfWork;
        public TopicService(IUnitOfWork uow)
        {
            _unitOfWork = uow;
        }
        public async Task CreateAsync(TopicDto topic)
        {
            await
_unitOfWork.TopicRepository.CreateAsync(TopicDto.MapFromDto(topic));
            await _unitOfWork.SaveChangesAsync();
        }
        public async Task DeleteByIdAsync(int id)
        {
            await _unitOfWork.TopicRepository.DeleteByIdAsync(id);
            await _unitOfWork.SaveChangesAsync();
        }
        public async Task<IEnumerable<TopicDto>> GetAllAsync()
```

```
        {
            var topics = await _unitOfWork.TopicRepository.GetAllAsync();
            return TopicDto.MapToDtos(topics);
        }
        public async Task<TopicDto?> GetByIdWithWordsAsync(int id)
        {
            var topic = await _unitOfWork.TopicRepository.GetByIdWithWordsAsync(id);
            if(topic == null)
            {
                throw new ArgumentException(nameof(topic));
            }
            return TopicDto.MapToDto(topic);
        }
        public async Task UpdateAsync(TopicDto topic)
        {
            _unitOfWork.TopicRepository.Update(TopicDto.MapFromDto(topic));
            await _unitOfWork.SaveChangesAsync();
        }
    }
public class WordDto
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public string Translation { get; set; }
        public int TopicId { get; set; }
        public static WordDto MapToDto(Word word)
        {
            return new WordDto()
            {
                Id = word.Id,
                Name = word.Name,
                Translation = word.Translation,
                TopicId = word.TopicId
            };
        }
        public static IEnumerable<WordDto> MapToDtos(IEnumerable<Word> words)
        {
            var wordDtos = new List<WordDto>();
            if(words != null)
            {
                foreach (var word in words)
                {
                    wordDtos.Add(MapToDto(word));
                }
```

```
            }
            return wordDtos;
        }
        public static Word MapFromDto(WordDto word)
        {
            return new Word()
            {
                Id = word.Id,
                Name = word.Name,
                Translation = word.Translation,
                TopicId = word.TopicId
            };
        }
    }
public class TopicDto
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public IEnumerable<WordDto>? Words { get; set; }
        public static TopicDto MapToDto(Topic topic)
        {
            return new TopicDto()
            {
                Id = topic.Id,
                Name = topic.Name,
                Words = WordDto.MapToDtos(topic.Words)
            };
        }
        public static IEnumerable<TopicDto> MapToDtos(IEnumerable<Topic> topics)
        {
            var topicDtos = new List<TopicDto>();
            foreach (var topic in topics)
            {
                topicDtos.Add(MapToDto(topic));
            }
            return topicDtos;
        }
[Route("api/[controller]")]

    [ApiController]
    [Authorize]
    public class WordController : ControllerBase
    {
        private readonly IWordService _wordService;
```

```
        public WordController(IWordService wordService)
        {
            _wordService = wordService;
        }
        [HttpGet]
        public async Task<IEnumerable<WordDto>> GetAllWords()
        {
            return await _wordService.GetAllAsync();
        }
        [HttpGet("{id}")]
        public async Task<IResult> GetWordById(int id)
        {
            var word = await _wordService.GetByIdAsync(id);
            return word == null ? Results.NotFound() : Results.Ok(word);
        }
        [HttpPost]
        public async Task<IResult> CreateWord(WordDto wordDto)
        {
            await _wordService.CreateAsync(wordDto);
            return Results.Ok();
        }
        [HttpPut]
        public async Task<IResult> UpdateWord(WordDto wordDto)
        {
            await _wordService.UpdateAsync(wordDto);
            return Results.Ok();
        }
        [HttpDelete("{id}")]
        public async Task<IResult> DeleteWord(int id)
        {
            await _wordService.DeleteByIdAsync(id);
            return Results.Ok();
        }
    }
[Route("api/[controller]")]
    [ApiController]
    [Authorize]
    public class TopicController : ControllerBase
    {
        private readonly ITopicService _topicService;
        public TopicController(ITopicService topicService)
        {
            _topicService = topicService;
        }
```

```
        [HttpGet]
        public async Task<IEnumerable<TopicDto>> GetAllTopics()
        {
            return await _topicService.GetAllAsync();
        }
        [HttpGet("{id}")]
        public async Task<IResult> GetTopicByIdWithWordsAsync(int id)
        {
            var topic = await _topicService.GetByIdWithWordsAsync(id);
            return topic == null ? Results.NotFound() : Results.Ok(topic);
        }
return Results.Ok();
        }
        [HttpPut]
        public async Task<IResult> UpdateTopic(TopicDto topicDto)
        {
            await _topicService.UpdateAsync(topicDto);
            return Results.Ok();
        }
        [HttpDelete("{id}")]
        public async Task<IResult> DeleteTopic(int id)
        {
            await _topicService.DeleteByIdAsync(id);
            return Results.Ok();
        }
    }
[Route("api/[controller]")]
    [ApiController]
    public class AccountController : ControllerBase
    {
        private readonly UserManager<IdentityUser> _userManager;
        private readonly RoleManager<IdentityRole> _roleManager;
        private readonly IConfiguration _configuration;
        public AccountController(
            UserManager<IdentityUser> userManager,
            RoleManager<IdentityRole> roleManager,
            IConfiguration configuration)
        {
            _userManager = userManager;
            _roleManager = roleManager;
            _configuration = configuration;
        }
        [HttpPost]
        [Route("login")]
```

```
    public async Task<IResult> Login([FromBody] AccountDto dto)
    {
        var user = await _userManager.FindByNameAsync(dto.Username);
        if (user != null && await _userManager.CheckPasswordAsync(user,
dto.Password))
        {
            var userRoles = await _userManager.GetRolesAsync(user);
            var authClaims = new List<Claim>
            {
                new Claim(ClaimTypes.Name, user.UserName),
                new Claim(JwtRegisteredClaimNames.Jti, Guid.NewGuid().ToString()),
            };
            foreach (var userRole in userRoles)
            {
                authClaims.Add(new Claim(ClaimTypes.Role, userRole));
            }
            var token = GetToken(authClaims);
            return Results.Ok(new
            {
                token = new JwtSecurityTokenHandler().WriteToken(token),
                expiration = token.ValidTo
            });
        }
        return Results.Unauthorized();
    }
    [HttpPost]
    [Route("register")]
    public async Task<IResult> Register([FromBody] AccountDto dto)
    {
        var userExists = await _userManager.FindByNameAsync(dto.Username);
        if (userExists != null)
        {
            return Results.BadRequest(new { Status = "Error", Message = "User already
exists!" });
        }
        IdentityUser user = new()                    var result = await
_userManager.CreateAsync(user, dto.Password);
        if (!result.Succeeded)
        {
            return Results.BadRequest(new { Status = "Error", Message = "User creation
failed! Please check the credentials." });
        }
        if(dto.Role == "Teacher")
        {
```

```
            if (!await _roleManager.RoleExistsAsync(UserRole.Admin))
            {
                await _roleManager.CreateAsync(new IdentityRole(UserRole.Admin));
            }
            await _userManager.AddToRoleAsync(user, UserRole.Admin);
        }
        else
        {
            if (!await _roleManager.RoleExistsAsync(UserRole.User))
            {
                await _roleManager.CreateAsync(new IdentityRole(UserRole.User));
            }
            await _userManager.AddToRoleAsync(user, UserRole.User);
        }
        return Results.Ok();
    }
    private JwtSecurityToken GetToken(List<Claim> authClaims)
    {
        var authSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(_configuration["JWT:Secret"]));
        var token = new JwtSecurityToken(
            issuer: _configuration["JWT:ValidIssuer"],
            audience: _configuration["JWT:ValidAudience"],
            expires: DateTime.Now.AddHours(3),
            claims: authClaims,
            signingCredentials: new SigningCredentials(authSigningKey, SecurityAlgorithms.HmacSha256)
            );
        return token;
    }
}
```