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

КОМПЛЕКСНА МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

Web-система для дистанційного вивчення слів англійської мови. Частина 2. «Підсистема коригування процесу навчання»

ПОЯСНЮВАЛЬНА ЗАПИСКА

Виконав: студент 2-го курсу, групи 1КІ-21м
спеціальності 123 – Комп'ютерна інженерія
_____ Бугай. О. С.

Керівник: к.т.н., проф. каф. ОТ
_____ Азарова А. О.

«21» 12 2022 р.

Опонент: к.т.н., доцент каф. МБІС
_____ Карпінець В. В.

«22» 12 2022 р.

Допущено до захисту
Завідувач кафедри ОТ
_____ д.т.н., проф. Азаров О. Д.

«23» 12 2022 р.

ВНТУ 2022

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Освітній рівень — магістр
Напрям підготовки — 123 Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри
обчислювальної техніки
_____ проф., д.т.н. О. Д. Азаров
«15» 09 2022 р.

З А В Д А Н Н Я

НА КОМПЛЕКСНУ МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ

респонденту **Бугаю Олексію Сергійовичу**

1 Тема роботи: Web-система для дистанційного вивчення слів англійської мови. Частина 2. «Підсистема коригування процесу навчання».

Керівник роботи: к.т.н., проф. каф. ОТ Азарова А. О.

Затверджені наказом вищого навчального закладу від 15.09.2022 р. № 205-А.

2 Строк подання студентом роботи 22.12.2022р.

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

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):

— вступ;

— аналітичний огляд комп'ютерних систем та існуючих підходів до контролю та коригування процесу навчання;

— моделювання та проєктування підсистеми коригування проце навчання;

— практична реалізація підсистеми коригування процесу навчання;;

— економічна частина.

5 Структура підсистеми коригування плану навчання: система контрол коригування та покращення результатів навчання.

6 Консультантів розділів роботи наведено у табл. 1.

Таблиця 1 — Консультанти розділів КМКР

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
2	к.т.н., доц. каф. ОТ Снігур А. В.		
5	к.е.н., проф. каф. ЕПВМ Небава М. І.	Небава	Небава

7 Дата видачі завдання _____

8 Календарний план розділів роботи наведено у таблиці 2.

Таблиця 2 — Календарний план КМКР

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів роботи	Примітка
1	Постановка задачі роботи	09.09.2022	вик.
2	Дослідження методів дистанційного вивчення іноземної мови	19.09.2022	вик.
3	Математичне моделювання процесу коригування навчання	3.10.2022	вик.
4	Аналіз та вибір технологій проєктування та розробки web-додатків	17.10.2022	вик.
5	Програмна реалізація системи	31.10.2022	вик.
6	Підготовка матеріалів для опису проєктування підсистеми оцінювання якості навчання	7.11.2022	вик.
7	Оформлення пояснювальної записки	21.11.2022	вик.
8	Перевірка якості виконання магістерської роботи	01.12.2022	вик.

Студент _____ Бугай Олексій Сергійович

Керівник _____ к.т.н., проф. каф. ОТ Азарова Анжеліка Олексіївна

АНОТАЦІЯ

УДК 004.62

Бугай О. С. Web-система для дистанційного вивчення слів англійської мови. Частина 2. «Підсистема коригування процесу навчання». Комплексна магістерська кваліфікаційна робота зі спеціальності 123 — комп'ютерна інженерія, освітня програма — комп'ютерна інженерія. Вінниця: ВНТУ, 2022. 148 с.

На укр. мові. Бібліогр.: 40 назв; рис.: 50; табл.: 11.

Комплексна магістерська кваліфікаційна робота присвячена розробленню веб-системи для дистанційного вивчення слів англійської мови та її підсистеми коригування процесу навчання.

Основний напрямок дослідження даної частини магістерської роботи — проєктування та інтеграція підсистеми коригування процесу навчання користувача. Під час виконання магістерської кваліфікаційної роботи проаналізовано системи-аналоги, виявлено та враховано їхні недоліки у власній розробці, розглянуто технології для розроблення адаптивний навчальних планів, вивчено основні способи використання веб-технологій у різних навчальних процесах, у тому числі для коригування процесу засвоєння слів, на базі яких із застосуванням статистичних даних розроблено математичну модель процесу засвоєння матеріалу респондентом.

Розроблено та реалізовано масштабовану архітектуру системи з використанням патернів проєктування. Проведено функціональне та мануальне тестування роботи системи. Систему розроблено в програмному середовищі .NET і Angular та призначена для використання у веб-браузерах.

Ключові слова: веб-розробка, веб-система, дистанційне навчання, коригування процесу навчання, Angular, .NET.

ABSTRACT

Buhai O. S. Web-system for remote learning of English words. Part 1. «Subsystem for quality evaluation of the education». Comprehensive master's qualification work on specialty 123 — computer engineering, educational program — computer engineering. Vinnytsia: VNTU, 2022. 148 p.

In Ukrainian language. Bibliography: 40 titles; fig.: 50; tab.: 11.

The complex master's qualification work is devoted to the development of a web system for remote learning of English words and its subsystem of adjusting the education process.

The main direction of research of this part of the master's thesis is the design and integration of the user training adjustment subsystem. During the master's qualification work analogue systems were analyzed, their disadvantages were identified in their own development and taken into account, technologies for the development of adaptive curricula were considered. The main ways of using web-technologies in various educational processes were studied, including for adjusting the process of learning words on the basis of which with the use of statistical data, mathematical models of the process of memorizing words and assimilation of material by the respondent were developed.

A scalable architectural system was developed and implemented using design patterns. The web system is developed in the .NET and Angular programming environments and is intended for use in web browsers.

Keywords: web development, web system, distance learning, adjustment of the education process, Angular, .NET.

ЗМІСТ

ВСТУП

У сучасному світі людині доводиться запам'ятовувати величезну кількість інформації за досить короткі терміни. У зв'язку з тим, кількість даних, що має опрацьовувати людина збільшується, це спричинює ускладнене сприйняття постійно змінюваного навколишнього середовища. Це особливо стосується вивчення іноземних мов, тому що володіння ними є одним із визначальних факторів у побудові успішної кар'єри.

Кількість слів у різних мовах з часом збільшується, тому є необхідним постійно актуалізувати свої знання. Дане завдання не є легким для людини, особливо враховуючи необхідність володіння мінімальним словниковим запасом у 5000-10000 слів для того, щоб людина могла відносно вільно володіти іноземною мовою.

Серед провідних науковців, які займалися розвитком методології коригування процесу навчання за згаданими критеріями необхідно зазначити таких дослідників, а саме: Азарова О. Д., Бикова В. Ю., Гриценчука О. О., Залізецького В. В., Коневщинську О. Е., Крупельницького Л. В., Маркову Є. С., Сімонсона М., Черняка О. І., Міллер Д. А., Еббінзауз Г., Джонс М., Бьюзен Т. та ін. [1–10].

Зауважимо, що існуючі математичні моделі та методи дистанційного навчання не є досконалими, оскільки спричиняють навантаження користувачів такого ресурсу зайвим навчальним матеріалом та рекламним контентом, є дорогими і недостатньо точно визначають результат.

У зв'язку з розвитком інформаційних технологій наявною є різні засоби дистанційного навчання, що дозволяють підтримувати свій рівень мови в активному стані, а також покращувати поточний рівень володіння нею. Характерною рисою таких засобів є відсутність математичних моделей, що дозволяють адаптувати систему дистанційного навчання до можливостей користувача, слідкувати за його навантаженням та підтримувати зацікавленість у подальшому навчанні. Крім того, наявні системи дистанційного навчання не

мають розвинених модулів для оцінювання повторюваного контенту вивчених слів користувачем. Разом із тим, такі аспекти мають бути обов'язковими складовими якісного та повнофункціонального використання ресурсів, що надає веб-додаток.

Отже, аналіз наявних систем дистанційного навчання дозволив виявити низку їх суттєвих обмежень, зокрема, надлишковість інформації, непродуманий інтерфейс, платний доступ до повного функціоналу системи та ін.

Враховуючи недостатню розвиненість наявних комп'ютерних систем дистанційного навчання та високий попит громадськості на цей тип освіти, необхідними є подальші дослідження для усунення недоліків, зазначених вище. Саме така мета переслідується у комплексній магістерській роботі, що і зумовлює її актуальність.

Метою роботи є удосконалення дистанційної системи вивчення англійської мови шляхом створення власної підсистеми коригування процесу навчання та її інтеграції до веб-системи дистанційного навчання.

Для досягнення такої мети було поставлено та вирішено такі **завдання**:

— проаналізовано існуючі системи дистанційного вивчення слів англійської мови;

— виявлено недоліки та переваги наявних систем коригування процесу навчання;

— розроблено математичну модель засвоєння матеріалу (слів англійської мови) респондентом;

— обґрунтовано засоби програмної реалізації системи та її інтерфейс;

— програмно реалізовано підсистему коригування процесу вивчення слів англійської мови;

— здійснено тестування розробленої підсистеми та доведено її працездатність, наведено її переваги порівняно з існуючими аналогами;

— обґрунтовано економічну доцільність інвестиційних вкладень у підсистему коригування процесу навчання.

Об’єкт дослідження — процес розроблення веб-системи для дистанційного навчання.

Предмет дослідження — розроблення підсистеми коригування процесу вивчення слів англійської мови у межах Web-системи дистанційного навчання.

Наукова новизна отриманих результатів магістерської роботи полягає в удосконаленні системи дистанційного вивчення слів англійської мови, що, на відміну від існуючих підходів, шляхом впровадження підсистеми коригування процесу навчання засобами математичного моделювання дозволяє враховувати індивідуальні особливості користувача.

Практичне значення отриманих результатів полягає в можливості підвищення рівня володіння англійською мовою засобами створеної підсистеми коригування процесу навчання, що вбудовується до Web-системи для дистанційного вивчення слів англійської мови.

Апробацію результатів магістерської роботи було здійснено на регіональній конференції ВНТУ (м. Вінниця, 2022).

Публікації. За результатами роботи підсистеми коригування процесу навчання було опубліковано тези доповідей [11], подано заяву на реєстрацію авторського права на програму.

1 АНАЛІЗ КОМП’ЮТЕРНИХ СИСТЕМ ТА ІСНУЮЧИХ ПІДХОДІВ ДО КОНТРОЛЮ ТА КОРИГУВАННЯ НАВЧАЛЬНОГО ПРОЦЕСУ

1.1 Значення систем дистанційного навчання у загальному навчальному процесі

На даному етапі людського розвитку поява та вдосконалення нових технологій відбувається з небувалою швидкістю, наукові розробки стають головним двигуном суспільного прогресу. З'явилася нова проблема, яка полягає в тому, що кількість вимог навіть до середнього працівника у тій чи іншій сфері діяльності зростає. Щоб бути конкурентоспроможним у сучасному світі, що швидко змінюється, людині необхідно запам'ятовувати велику кількість інформації і вміти оперувати нею. Однак здібності людей не безмежні, і можливості людського мозку у тому числі. Людська пам'ять ділиться на довготривалу і короткочасну пам'ять, і якщо довгострокова пам'ять може зберігати величезну кількість даних, то короткочасна немає, ймовірно, її ємність становить лише 7±2 об'єкта [10]. І для того, щоб інформація перейшла з коротко-строкової у довгострокову пам'ять, потрібне її правильне запам'ятовування та повторення. Вивчення англійської мови не є винятком. Для її вивчення потрібно прикласти неабияких зусиль, тому люди часто прибігають до використання веб-додатків чи мобільних застосунків аби полегшити цей процес.

Методи онлайн-навчання добре зарекомендували себе як серед викладачів, так і серед тих, хто вивчає мову. Завдяки своїй гнучкості та зручності він ідеально підходить для тих, хто не може дотримуватися розкладу в класі, але може працювати в групах та індивідуально. Ознайомившись із різними методами вивчення англійської мови, користувач може обирати той, який найбільше подобається.

Все більше можливостей розробники вкладають у веб-системи, не винятком стало і вивчення англійської мови. На даний момент є безліч подібних застосунків, але не всі можуть задовольнити вимогам користувачів.

Основним недоліком багатьох веб-систем дистанційного вивчення англійської мови є неможливість адаптації під індивідуальні потреби користувача. Це означає, що немає чіткого плану вивчення слів, є можливість повторення одного і того самого матеріалу або взагалі пропущення важливого наборів слів, до якого давно не поверталися, але які могли б бути корисними [12].

Сьогодні люди віддають перевагу дистанційній освіті, і для того, щоб забезпечити можливість здобуття якісної дистанційної освіти.

Г. Вебер [13] пропонує доповнити систему дистанційного навчання (СДН) адаптивною СДН, що підлаштовується під особливості індивідуального навчання. Ця орієнтація має велике значення, оскільки підвищує загальний рівень якості освіти.

Поширення дистанційної освіти як форми організації навчального процесу та сучасні інформаційно-комунікаційні технології ставлять завдання побудови такої адаптаційної системи, яка характеризується такими особливостями [13]:

— брати до уваги попередні знання та досвід користувачів;

— можливість повторного використання, розвитку та накопичення освітнього контенту, що дає змогу більш ефективно керувати ним;

— автоматичне створення навчальних курсів;

— інтелектуалізація соціальних освітніх мереж;

— інтелектуалізація управління знаннями.

Під впливом освітніх тенденцій було створено низку адаптивних гіпермедійних систем, які зберігають у своїх моделях інформацію про характеристики та знання користувача та використовують їх для адаптації до різних візуальних аспектів.

На абстрактному рівні адаптивні гіпермедійні системи, зокрема, дистанційного навчання, складаються з трьох основних взаємодіючих компонентів.

Модель домену, що описує інформаційний зміст документа на структурному рівні, вказуючи як на існування стосунків між самими поняттями, так і на зв'язок між поняттями та інформаційними фрагментами або сторінками.

Моделі користувачів представляють характеристики, що відображають знання, цілі та історію навігації, і, крім того, відстежують динаміку зміни відповідних характеристик у процесі навчання.

Педагогічні моделі, також звані адаптивними, дають змогу реалізувати в системі необхідні механізми адаптації, наприклад, за допомогою так званих педагогічних або адаптивних правил.

Застосовуючи принципи адаптації, мультимедійні компоненти та різні види зворотного зв'язку, система може сформувати модель цілей, вподобань і знань конкретного користувача, яка потім може бути використана в процесі взаємодії для її адаптації до його потреб [14].

1.2 Вивчення недоліків та переваг методів побудови навчальних автоматизованих систем

Адаптивні методи включають широкий спектр програмних рішень, які дозволяють налаштовувати методи передавання та представлення різних типів інформації для задоволення потреб користувачів в автоматичному режимі.

Із широким впровадженням нових комп’ютерних технологій у навчальний процес сьогодні існують різноманітні класифікації. В останні роки такий поділ не розглядався, хоча слід припустити, що всі методи адаптації, які використо-вуються в адаптивних системах навчання, походять із області інтелектуальних систем навчання або області адаптивної гіпермедіа. Детальний аналіз типів

технологій в інтелектуальних освітніх системах надає П. Брусиловський у [15]. Розглянемо типи технологій, які нам надає автор.

Побудова послідовності курсу навчання, характеризується забезпеченням респондента найбільш прийнятною, індивідуально спланованою послідовністю інформаційних блоків і послідовністю навчальних завдань.

Інтелектуальний аналіз відповідей респондента, на основі відповідей респондента інтелектуальний аналізатор забезпечує зворотній зв'язок і оновлює його модель.

Інтерактивна підтримка розв’язування завдань, надання інтелектуальної допомоги респонденту на кожному кроці розв’язання задачі.

Допомога у розв’язанні прикладних завдань, допомагаючи респондентам розв’язувати нові прикладні завдання, а також аналогічні завдання, які були успішно розв’язані раніше.

Порівняно з технічною класифікацією, запропонованою П. Брусиловським [15], було розширено класифікацію відповідно до обраного методу адаптації (рис. 1.1).

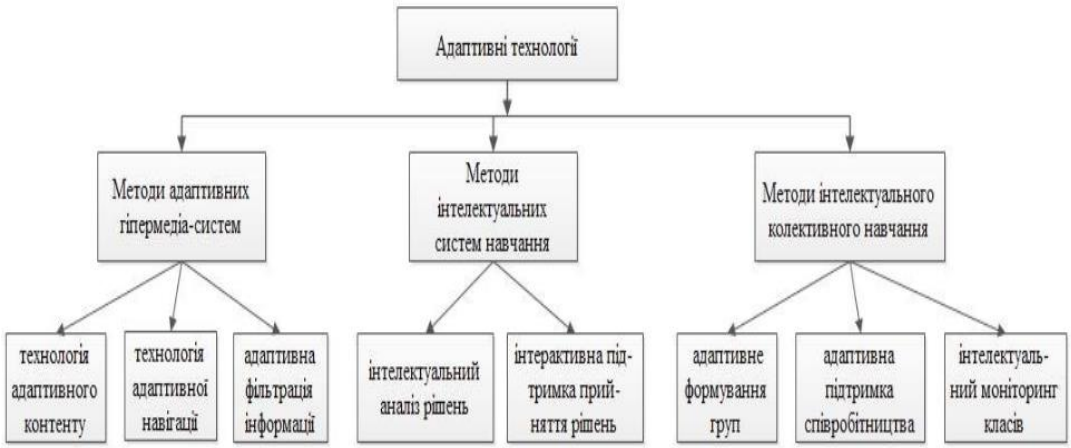


Рисунок 1.1 — Розширена класифікація технологій

Д. А. Зайцева в [16] проаналізувала методи контролю та моделі оцінювання знань. На думку автора магістерської роботи, методи контролю знань необхідно поділяти на три категорії (рис. 1.2):

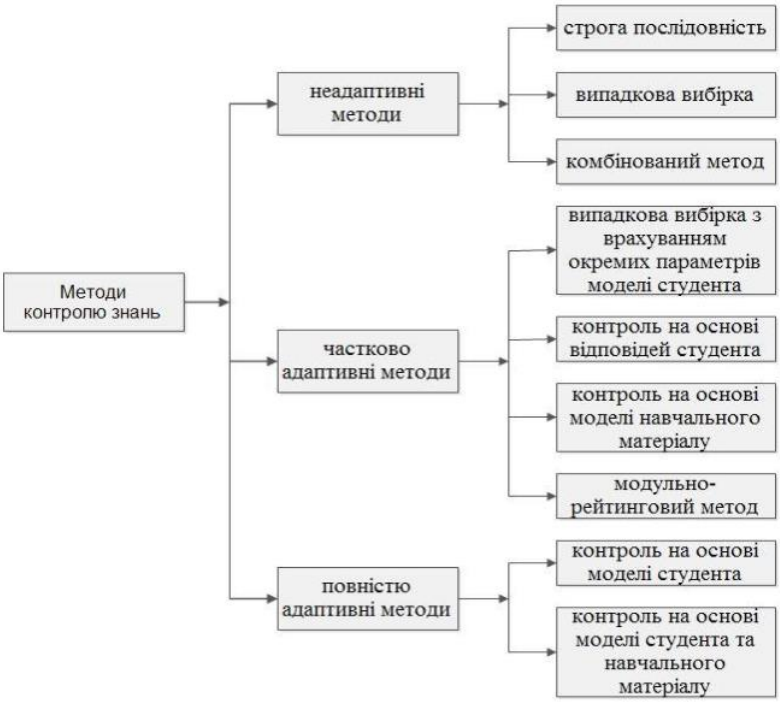


Рисунок 1.2 — Методи контролю знань

— неадаптивні;

— частково адаптивні;

— повністю адаптивні методи.

Характерною рисою всіх неадаптивних методів є те, що всі респонденти виконують однакову, заздалегідь визначену послідовність завдань. У частково адаптивних методах послідовність та кількість контрольних завдань різні для сильних, середніх та слабких рівнів та формуються в залежності від попередніх відповідей користувача або на основі певного передбаченого сценарію.

Адаптивні методи максимально використовують інформацію з моделі респондента і / або моделі навчального матеріалу та дозволяють організувати контроль індивідуально для кожного респондента, підтримуючи, наприклад, найбільш прийнятний для нього рівень контрольних завдань або формуючи індивідуальні стратегії контролю за окремою темою.

У сучасних адаптивних системах навчання, в залежності від мети їх застосування, застосовується унікальний підхід до технологічних розробок, особливо до загальновідомих [16].

Технологія, заснована на ітеративному навчанні (ІН) [17]. ІН характеризується тим, що система навчання багаторазово виконує дії, спроби тощо для досягнення мети за певних зовнішніх умов.

Перевагою цієї технології є те, що ІН можуть бути описані кількісно за допомогою кривих навчання, які дозволяють виразити залежність критеріїв рівня навчання від часу та повторення.

Разом із тим, недоліком такого методу є заздалегідь заготований матеріал для навчання, у якого відсутній механізм адаптації до конкретного респондента.

Багатомовна адаптивна технологія навчання (ML-технологія). Запропоновано удосконалену модель та алгоритм навчання, що використовує параметр асоціації та відображає ступінь зв'язку між словом рідної мови та його значенням в іншій іноземній мові. Алгоритм навчання будується шляхом формування асоціативних полів для понять, що підлягають запам'ятовуванню. Це дозволяє більш інтенсивно та одночасно поповнювати активний словниковий запас за фахом кількома мовами.

Адаптивна технологія одиниць розподіленого контенту [17]. Дана технологія має такі характеристики:

— формування електронних підручників (ЕП), адаптованих до конкретних особливостей респондентів, на основі попередньо розробленої онтології відповідних дисциплін;

— реалізація в електронному підручнику як міжмодульних та міжпонятійних зв'язків, так і змішаних зв'язків;

— розробка навчальних матеріалів та методик за участю двох типів авторів: реальні версії ЕП створюються викладачами, які безпосередньо працюють з певними респондентами;

— напівавтоматичне перетворення текстів у гіпертекст;

— розвиток пошукових функцій на основі онтології.

Недоліками такої системи є обмежений вибір тем для вивчення і тестування, оскільки наповнення контенту потребує значних ресурсів.

До переваг належать можна віднести моніторинг процесу навчання, що дозволяє відслідковувати, наскільки успішно проходить навчання.

Технологія мультимножин. Мультимножини — це множини з повтореннями, які є корисними як модель для представлення та вивчення об'єктів, що характеризуються множинністю та повторенням даних. Відомо багато практичних рішень з використанням мультимножин для розв'язання теоретичних і прикладних задач [18].

Оскільки матеріал, що подається респондентам, можна розглядати як сукупність неподільної інформації, в якій можна знайти однакові елементи, то бажано використовувати теорію множин як математичний апарат для проєктування і наповнення бази знань освітньої системи. Таким чином, технології навчання враховують зміни параметрів і структури моделі об'єкта, виходячи з поточної інформації, і визначають способи реалізації адаптивного підходу під час навчального процесу.

Використання мультимножинної технології, перевагою якої є визначення глибини засвоєння окремих фрагментів інформації з метою побудови моделі предметної області магістерської дипломної роботи.

Недоліками такої технології є необхідність побудови усієї множини ефективних альтернатив.

1.3 Порівняльний аналіз програмного забезпечення систем-аналогів

Стрімкий розвиток інформаційних технологій у 21 столітті актуалізує питання модернізації системи освіти. В Україні сутність такої модернізації найкраще відображає концепція дистанційної освіти, яка завдяки глобальному феномену Інтернету охоплює широкий сегмент суспільства і є найважливішим чинником його розвитку.

Розглянемо підсистеми коригування процесу навчання англійської лексики аналогів та проаналізуємо її переваги та недоліки.

Для порівняння оберемо такі системи-аналоги, як uTalk, Lingualeo та Duolingo.

uTalk призначений для тих, хто хоче вивчити часто використовувані слова і сталі вирази іноземною мовою. Програма ідеально підходить для початківців, а також для тих, хто хоче заповнити нестачу словникового запасу або поліпшити вимову. Додаток охоплює величезну кількість загальновживаних повсякденних слів і фраз, розбитих на категорії за темами; кожну тему можна вивчити приблизно за дві-три години, але ви можете занурюватися в неї скільки завгодно разів.

Основними перевагами, що відрізняють підсистему коригування процесу навчання uTalk, що відрізняють даний веб застосунок від інших, є:

— застосунок пропонує понад 100 мов навчання;

— кожна мова містить 60 заздалегідь заготованих тем;

— кожна тема складається з слів, згрупованих загальною тематикою;

— використання інтерактивного режиму навчання.

Також слід зазначити недоліки даного додатку [19]:

— неможливість створювати власні теми;

— неможливість редагувати заготовані теми;

— преміум режим, що обмежує функціонал системи;

— застарілий інтерфейс;

— відсутність адаптації до можливостей користувача;

— відсутність механізму генерації навчального плану користувача;

— наявність реклами, що заважає процесу навчання.

Сторінка з списком тем веб-системи для вивчення іноземних слів uTalk зображено на (рис. 1.3).

Lingualeo — одна з найкращих програм для вивчення іноземної мови. Автори визначили сильні та слабкі сторони граматики, щоб мати можливість

забезпечити систему навчання. Користувачеві необхідно лише дотримуватися наданих рекомендацій. Це ідеальний проєкт для дітей, адже він стимулює їх піклуватися про лева.

Рисунок 1.3 — Сторінка з списком тем веб-системи uTalk

До переваг підсистеми коригування плану навчання веб3-системи Lingualeo можем віднести:

— наявність карти навчання;

— система нагородження за успіхи в навчанні;

— використання інтерактивного режиму навчання;

— сучасний інтерфейс;

— можливість створення власних наборів слів;

— можливість обирати набір слів в режимах інтерактивного навчаннях.

До недоліків можемо віднести [20]:

— орієнтовано на дітей;

— відсутність можливості вплинути на матеріали карти навчання;

— карта навчання заздалегідь заготовлена для кожного з рівнів англійської мови;

— преміум режим, що обмежує функціонал системи;

Сторінка з картою навчання веб-системи для вивчення англійської мови Lingualeo зображено на (рис. 1.4).

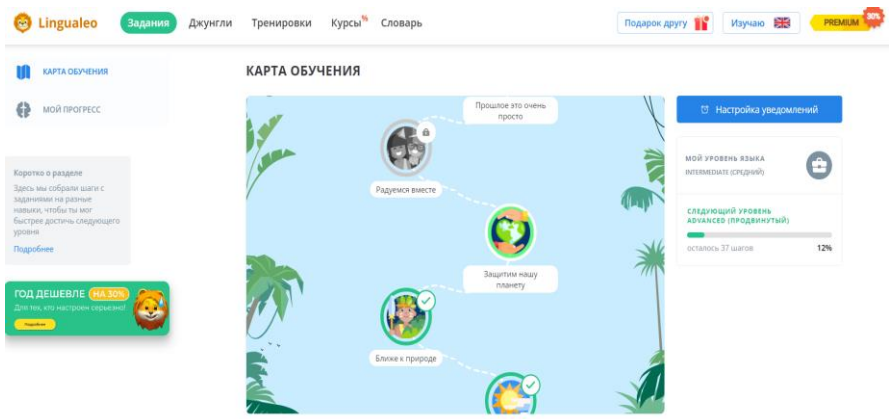


Рисунок 1.4 — Сторінка з картою навчання Lingualeo

Також одним із лідерів на ринку являється Duolingo, де ви можете вивчити не тільки англійську, але й німецьку, французьку, іспанську, італійську та португальську мови.

Це відома онлайн-система, яка фокусується в основному на вивченні слів іноземної мови. Підсистема для вивчення слів включає в себе можливість вивчення слів шляхом перегляду графа в якості механізму корегування процесу навчання.

Перевагами підсистеми коригування процесу навчання веб-системи Duolingo є:

— наявний план для навчання користувачів у виді графу;
— використання інтерактивного режиму;
— сучасний інтерфейс;
— система нагородження за успіхи в навчанні;
— наявність механізму зацікавленості користувачів;

До недоліків можемо віднести [21]:

— орієнтовано на користувачів з нульовим або початковим рівнем знань;
— повна залежність від матеріалів, що надає система;
— відсутність можливості вплинути на матеріали графу навчання;

— преміум режим, що обмежує функціонал системи;

— відсутність механізму створення власних наборів слів;

— наявність реклами, що заважає процесу навчання.

Сторінка з графом навчання веб-системи для вивчення іноземних слів Duolingo зображено на рис. 1.5.

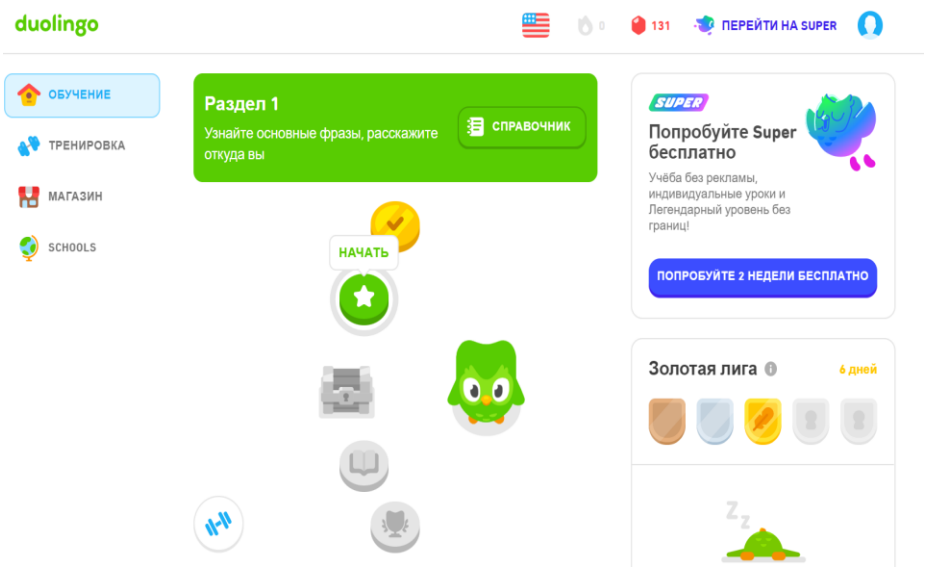


Рисунок 1.5 — Сторінка з графом навчання Duolingo

Порівняльний аналіз підсистем забезпечення навчання uTalk, Lingualeo та Duolingo зображено в табл. 1.1.

Таблиця 1.1 — Порівняльний аналіз підсистем коригування процесу навчання

Характеристика	uTalk	Lingualeo	Duolingo
Наявність механізму навчального плану	-	+	+
Можливість модифікації навчального плану	-	-	-
Можливість створювати користувацькі набори слів	-	+	-
Наявність вибору режиму тренування слів	+	+	-
Відсутність обмеження функціоналу Premium режимом	-	-	-
Відсутність реклами, що заважає процесу навчання	-	-	-

Висновки до розділу 1

У даному розділі було проведено аналітичний огляд комп’ютерних систем та існуючих підходів до контролю та коригування навчального процесу. Розглянуто особливості автоматизованого адаптивного навчального плану.

Проаналізовано методи до побудови навчальних автоматизованих систем. Розглянуто функції діагностики, на основі яких запропоновано вимоги до підсистеми коригування процесу навчання, що розробляється в магістерській роботі.

Проведено порівняння існуючих на ринку веб-систем та доведено доцільність розробки власної системи.

Вимогами до розроблюваної системи є побудова підсистеми з можливістю користувача впливати на створення індивідуального автоматизованого навчального плану на основі статистичних даних, з механізмом адаптування до можливостей користувача, а також відсутність реклами та обмежень преміум режимом.

Отже, аналіз переваг та недоліків існуючих веб-систем зумовлює вимоги до підсистеми, що розробляється в магістерській роботі, а саме:

— можливість використовувати веб-систему цілодобово;
— простий та інтуїтивно зрозумілий дизайн інтерфейсу;
— адаптація до потреб користувача;
— покроковий процес навчання;
— можливість редагувати та додавати набори слів;
— захист даних користувача;
— план навчання на основі можливостей респондента.

2 МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ ПІДСИСТЕМИ КОРИГУВАННЯ ПРОЦЕСУ НАВЧАННЯ

2.1 Побудова організаційної структури підсистеми коригування процесу навчання

Основою для побудови розроблюваної адаптивної системи дистанційного навчання є класичний метод автоматичного навчання. Респондентам презентують матеріал для навчання, представлений в інтерактивних режимах. Наступна частина теоретичних знань формується на основі попередніх результатів.

Отже, загальну модель підсистеми коригування процесу навчання автор магістерської роботи пропонує такою, як подано на рис. 2.1.

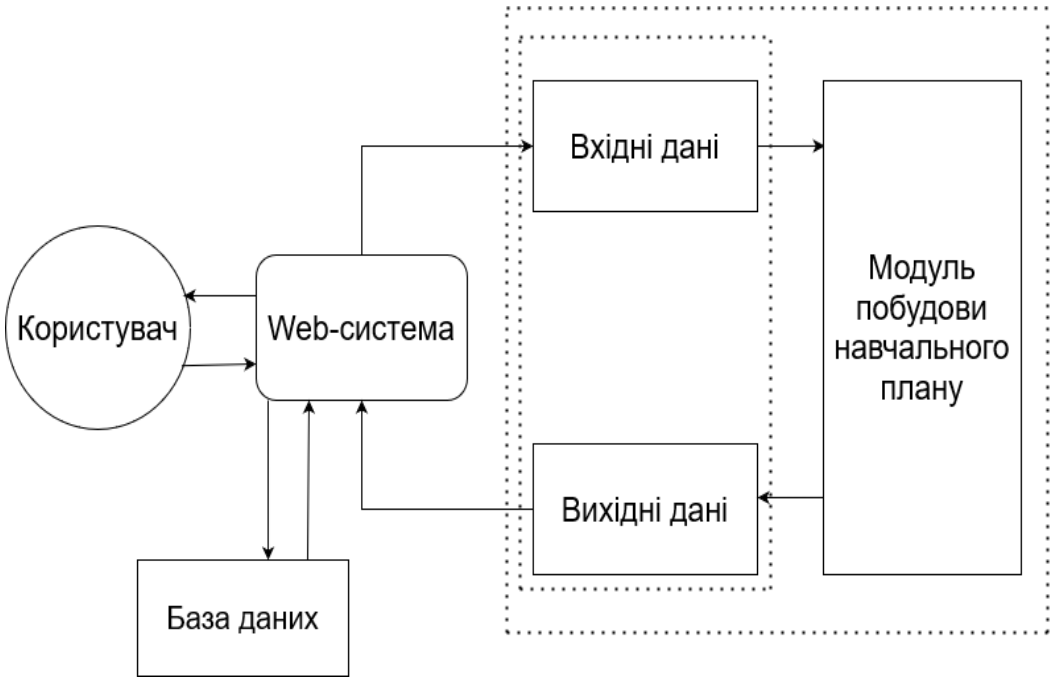


Рисунок 2.1 — Модель підсистеми коригування процесу навчання

Модуль побудови навчального плану виконує функцію управління процесом навчання користувача на основі алгоритму, що описує процес засвоєння інформації. На будь-якому етапі процесу навчання вхідні дані модуля адаптації є результатом навчання респондента, тобто набутими знаннями.

Навчальний підрозділ спрямовує процес навчання на основі кількісних і якісних характеристик, що відображають поточний рівень набутих знань, таких як один із режимів (перенавчання, додаткове навчання або тренінг). У цьому разі використовуються правила предикатів, кожен з яких за допомогою моделювання попередньої ситуації навчання.

Процедура призначення навчання виконує такі функції, обходячи модель семантичної мережі цільової області навчання, і забезпечує відображення результатів навчання та їх зберігання в системі.

Блок прийняття рішень як засіб адаптивного управління використовує накладну модель респондента, представлену набором параметрів. Кожен параметр відображає поточний рівень набутих навичок і компетенцій, або кількісно, або якісно. Рішення, що стосуються вибору сценарію (режиму) навчання, якому піддається система, такі на основі аналізу векторів і матриць, що представляють набір попередників всіх можливих траєкторій для продовження процесу навчання. Ця система розв'язує складну багатокритеріальну проблему вибору альтернатив для отримання освітньої вигоди від вивчення курсу [22].

Структура блоку прийняття рішень є циклічною, запит, що надходить може кілька раз бути опрацьованим. Основна мета такого підходу — отримати оптимальне рішення, що є підґрунтям для видачі рекомендації.

Отже, уточнена структурна модель процесу коригування дистаційного навчання набуває вигляду, зображеного на рис. 2.2.

Програмна реалізація описаних кроків модуля адаптації в освітній системі здійснюється шляхом формалізації теоретичної основи:

— графічне представлення процесу адаптації;

— теорія мультимножин, заснована на квантовому розбитті навчального змісту для визначення глибини засвоєння інформації;

— методи кластерного аналізу і математико-логічний апарат для визначення режиму навчання.

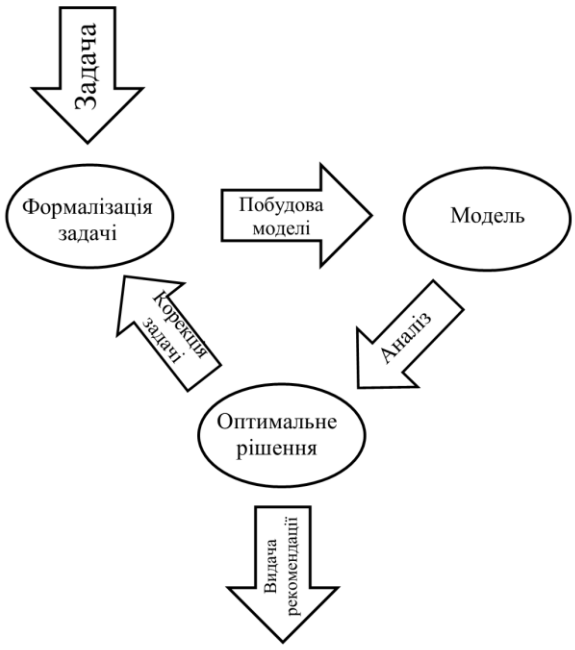


Рисунок 2.2 — Уточнена модель підсистеми коригування процесу навчання

Це дає змогу автоматизованій системі надавати індивідуальні адаптивні траєкторії для вивчення навчальних курсів.

2.2 Математичне моделювання процесу запам’ятовування слів

У процесі навчання дуже важливу роль відіграють процеси пам'яті. Пам'ять являє собою один із найважливіших психічних процесів для засвоєння знань. Протягом століть було розроблено багато теорій (психології, фізіології, хімії тощо) щодо природи та методів пам’яті. Однак загальноприйнятої теорії пам’яті та її визначення не існує.

Пам'ять означає збереження, подальше розпізнавання та відтворення слідів минулого досвіду. Тому, щоб щось запам’ятати, необхідно зберігати інформацію (зберігання або кодування), яка передається в мозок протягом певного періоду часу (зберігання), а потім витягувати (відтворювати) у разі потреби [23]. У

напрямі психології це перша спроба науково пояснити феномен пам'яті з наукової точки зору.

Питання ефективної пам'яті та уповільнення процесу забування залишається важливим, оскільки навчання базується на цих процесах. Г. Ебінгауз у [24], був першим у розробленні кількісного методу дослідження пам’яті та забування та побудував залежну від обсягу пам’яті криву (тобто криву забування) на основі часу, що минув (рис. 2.2) [24]. Ця крива називається кривою забування. Крива описується формулою (2.1) [24]:

$$b = \frac{100k}{(\log t)^c + k}, \tag{2.1}$$

де b — відсоток матеріалу, що утримується в пам'яті під

час експерименту (або контролю), або обсяг пам'яті у «відсотках зберігання»;

t — час (у годинах) з моменту повного вивчення матеріалу;

c і k — константи, що було отримано методом найменших квадратів із вхідних експериментальних даних.

Американський психолог М. Джонс провів аналогічний експеримент щодо забування важливих речей і отримав криву, близьку до кривої Еббінгауза. Експеримент полягав у такому: перед останньою лекцією з психології Джонс попередив респондентів, що в кінці вони отримають картку із запитаннями щодо змісту лекції та мають дати письмові відповіді [25].

Лекції чіткі та легкі для читання зі швидкістю 75 слів на хвилину. Письмові опитування проводилися п'ять разів у різні проміжки часу. Результати експерименту наведено в табл. 2.1.

Таблиця 2.1 — Результати експерименту

f, %	65	45,3	34,6	30,6	24,1
t, год.	2	96	168	336	1344

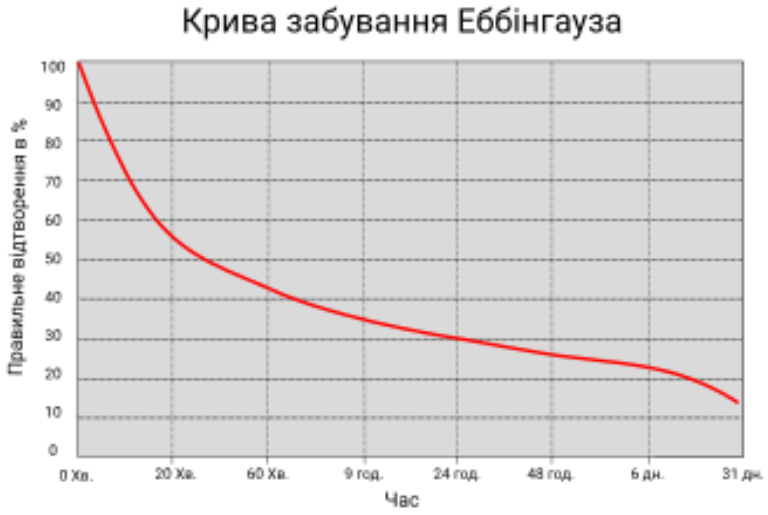


Рисунок 2.2 — Крива забування Еббінгауза

У поєднанні з експериментом М. Джонса та його експериментальними даними було побудовано криву забування та визначено закон забування важливих речовин. Закон забування важливих речей представлений приблизною логарифмічною функцією виду (2.2) [25]

$$f(t) = a \cdot ln(t) + b, \tag{2.2}$$

де t — час, що минув із моменту повного засвоєння матеріалу;

a, b — параметри, що відображають характеристики пам’яті респондентів, визначені за методом індивідуальних статистичних найменших квадратів 3-4 тестів.

На основі отриманих даних із табл. 2.1 маємо: $f(t) = 6{,}458 \cdot ln(t) + 70{,}133$.

Криву забування Джонса відображено на рис. 2.3.

Сьогодні відомими є чинники, що впливають на швидкість забування, а саме: недостатнє розуміння матеріалу, великий обсяг матеріалу, труднощі в навчанні, діяльність після навчання, розумова або фізична втома, роль зовнішніх подразників, інтерес до матеріалу.

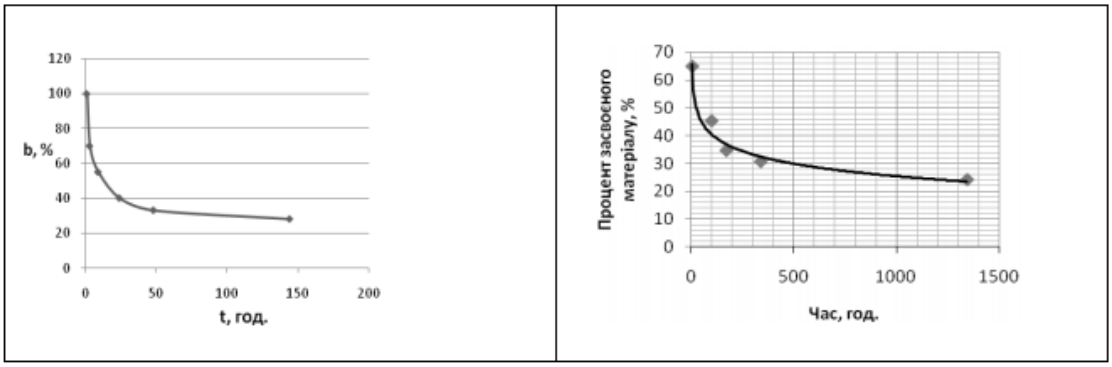


Рисунок 2.3 — Результати експерименту Джонсона

Погано структуровані матеріали, як відомо, важко запам’ятати, тоді як добре організовані матеріали легко запам’ятати з невеликим повторенням. Коли матеріали не мають чіткої структури, люди схильні розділяти або поєднувати їх на основі ритму, симетрії тощо. Людина може намагатися реконструювати матеріал, щоб краще його запам'ятати [25].

Якщо ми поглянемо на методи, які використовуються для запам'ятовування, то можна зробити висновок, що вони не є ефективними за такими критеріями як навчання, лінійна анотація та повторення.

У міру того, як запам’ятовується більше інформації, починають з’являтися нові методи пам’яті. На стику психології та інформатики виник ще один метод пам'яті, так звана карта розуму [26], як показано на рис. 2.4 .

Рисунок 2.4 — Карта розуму

Це абсолютно новий підхід до аналітичного представлення інформації на основі графічних зображень, асоціацій або логічних зв'язків.

Тоні Бьюзен, автор книги Mind Mapping Techniques, рекомендує певні дії, для покращення процесу запам’ятовування, про які слід згадати [27].

Записати інформацію в радіальному форматі, з темою в центрі та гілками з ключовими словами з цього. Наступними кроками слід нагадувати про те, що ключове слово чи фраза є на гілці, яка відхиляється від основної теми чи ідеї. Тоді кожне ключове поняття буде у фокусі, і ключові слова будуть відокремлені від нього таким же чином, а ключові слова розміщені на кольорових гілках. Гілки мають бути пов’язані, а не ієрархічні. Асоціації можуть підтримуватися символьними графами [27].

Розглянемо адаптивний модуль дистанційної системи адаптивного навчання, який враховує особистий час забування профілю кожного респондента, наведено в табл. 2.2.

Таблиця 2.2 — Карта забуття респондента

Час, дні	Відсоток засвоєного матеріалу, %		
	Респондент 1	Респондент 2	Респондент 3
14	50	75	81
28	30	53	68
84	23	37	52
224	10	21	39
Рівняння кривої забування			
$f(t)$	$-16{,}84 \cdot ln(t) + 95{,}85$	$-14{,}87 \cdot ln(t) + 104{,}24$	$-11{,}4 \cdot ln(t) + 104{,}06$

На рис. 2.5 зображено побудовані криві забування.

Якщо респонденти запам’ятали більше 50% вивченого матеріалу, то це відповідає зоні засвоєння матеріалу.

Розглянемо криву забування зони асиміляції. Для визначення медіанного значення, тобто кожен респондент запам’ятовує 50% часу на вивчення матеріалу, тобто при f(t) = 50 розв’язуємо рівняння (2-3) для параметра t, як наведено в табл. 2.3.

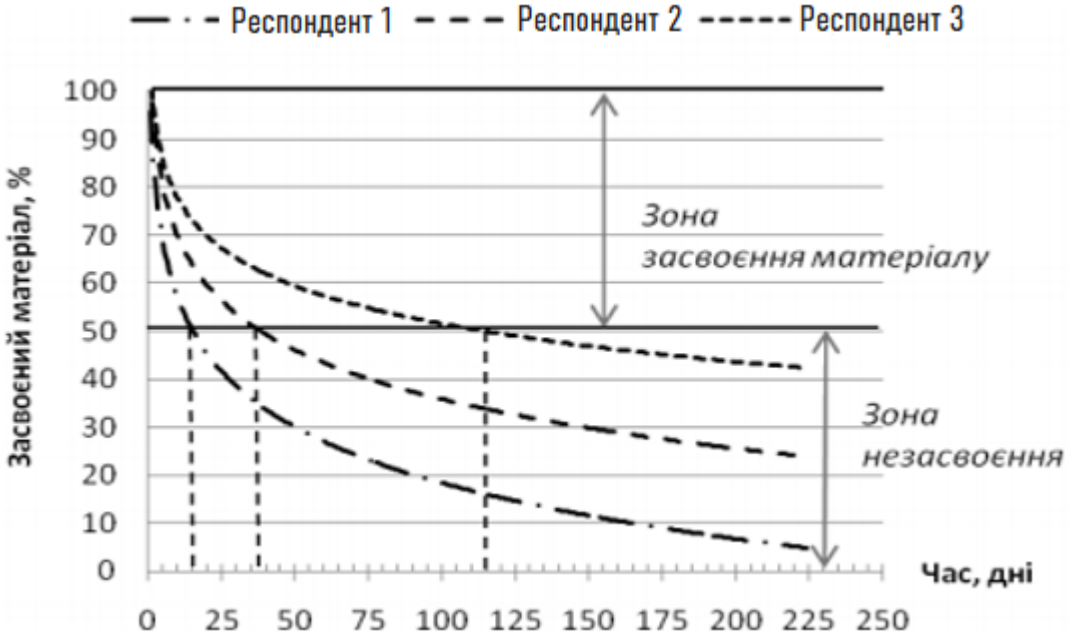


Рисунок 2.5 — Криві забування

Таблиця 2.3 — Визначення проміжного значення

Респондент 1	$f(t) = -16{,}84 \cdot ln(t) + 95{,}85$	(2)	$t = 15{,}2$
Респондент 2	$f(t) = -14{,}87 \cdot ln(t) + 104{,}24$	(3)	$t = 38{,}4$
Респондент 3	$f(t) = -11{,}4 \cdot ln(t) + 104{,}06$	(4)	$t = 114{,}7$

Логарифмічне та експоненціальне наближення на основі даних табл. 2.2 і табл. 2.3. З огляду на надійність значення апроксимації, значення в зоні асиміляції більше підходить для експоненціальної залежності поглиненої інформації від часу, як зображено на рис. 2.6).

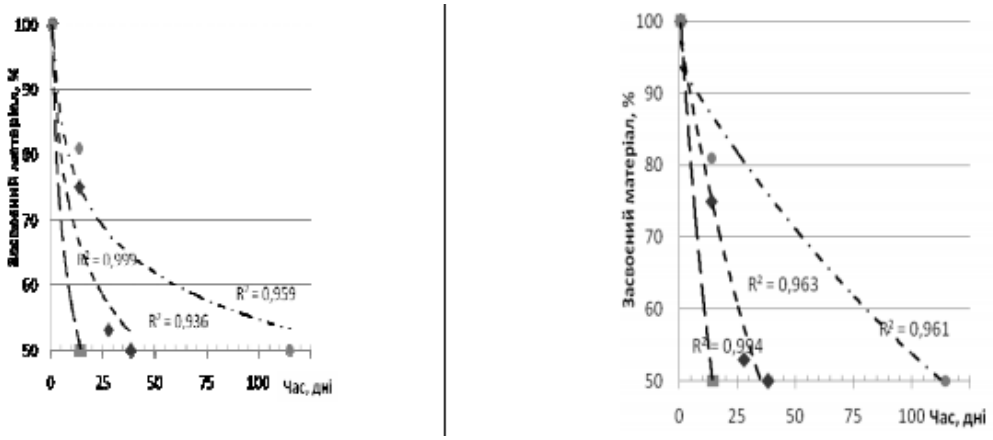


Рисунок 2.6 — Логарифмічна апроксимація експериментальних даних

На основі кривої забування адаптивна система формує календар повторюваних тем і визначає, коли кожному респонденту потрібно повторити навчальний матеріал. У визначений день респондентам буде надано нагадування та посилання на теми для перегляду [27].

Таким чином, графічне представлення ключових слів, ідей і пунктів дозволяє респондентам переглянути тему після її вивчення. При необхідності він може переглядати і редагувати блоки, які забув.

Використовуючи інтелектуальну карту для представлення кожної навчальної теми, можна охопити всю ситуацію і запам’ятати багато інформації, знайти зв’язки між різними елементами, запам’ятати інформацію та мати можливість відтворювати її протягом тривалого часу. Розумні графіки — інструменти для дослідження та повторення пройденого матеріалу. Після перевірки інтелектуальної карти кожної навчальної теми можна побудувати карту прогалин у знаннях.

Карта прогалин у знаннях — це розумова карта певної теми чи модуля, яка показує ключові поняття засвоєних і незасвоєних знань. Аналізуючи карту прогалин у знаннях, викладачі можуть отримати детальну інформацію про навчальні концепції, або теми [27].

2.3 Побудова математичної моделі засвоєння матеріалу респондентом

У розвиненій системі освіти модель респондента будується з урахуванням певних процесів:

— базується на поточних знаннях і навичках респондента шляхом аналізу його навчальної поведінки, динамічної (поведінкової або поточної) моделі, яка змінюється разом зі зміною особи, яку навчають;

— складається нормативна модель респондентів, яка включає вимоги до

кінцевого статусу респондента, професійних навичок та інших навчальних предметних навичок, що задаються викладачами (розробниками курсу).

— результатом навчання є такий стан, коли динамічна модель респондента збігається з його нормативною моделлю.

Будучи фундаментальним компонентом адаптивної системи освіти, модель респондента дозволяє комп'ютеризованим системам швидко адаптуватися до поточних потреб користувача.

Модель респондента містить різноманітну інформацію про його результати поточної роботи (час виконання завдань, кількість звернень до теми модуля), особисті психологічні характеристики (тип і спрямованість особистості, репрезентативна система, здатність до навчання, ступінь хвилювання-тривоги, пам'ять, загальний рівень підготовки) [28].

У магістерській роботі запропоновано структуру моделі респондента, яка враховує як індивідуальні особливості респондента, так і його початковий рівень знань за модулем, а також поточну успішність засвоєння нових знань (рис. 2.7).

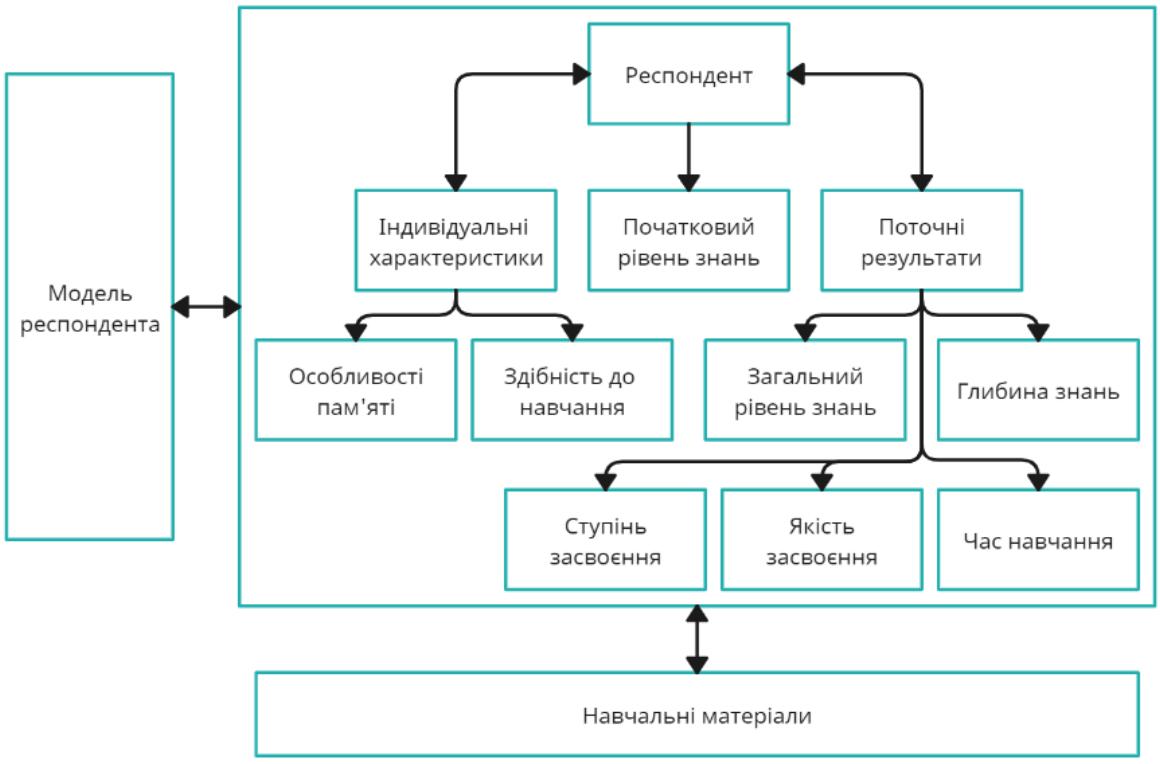


Рисунок 2.7 — Структурна модель респондента

У структурі системи адаптивної освіти модель респондента є обов’язковою фундаментальною складовою. Завдяки використанню моделі респондента більш об’єктивно формується порядок роботи респондентів із системою та навчальним матеріалом. Він визначає початкові налаштування системи і є основою для подальшої співпраці з ними [28].

На сьогоднішній день найбільш поширеними є оверлейні моделі охоплення знань (вектори та мережі, так звані графи знань) [28].

В освітніх системах, створених для адаптивного управління навчальною поведінкою респондентів, використовується оверлейна модель, яка дозволяє кожному поняттю бази знань зберігати значення, яке є оцінкою рівня знань респондента про це поняття або чи є індикатор чи ні було навчено.

Загальну математичну модель знань користувача (model of user) автор магістерської роботи пропонує у вигляді (2.3)

$$Mu = \{Tu_1, Tu_1, \ldots, Tu_n\}, \quad (2.3)$$

де Tu_n — модель вивчення n-ої теми.

Модель вивчення n-ої теми опишемо так:

$$Tu_n = \{MSu, s^n_l, k^n\}, \quad (2.4)$$

де MSu — узагальнена модель теми, що вивчається;

s^n_l — слова, які належать до заданої теми;

l — кількість слів у n-ій темі;

k^n_l — коефіцієнт кількості повторень даних слів.

Узагальнена математична модель визначається формулою (2.5)

$$MSu = \{P_1, P_2, \ldots, P^n\}, \quad (2.5)$$

де P^n — окремий параметр, що описує довільну характеристику респондента.

Набір функціональних залежностей між різними характеристиками респондентів моделює матрицю індивідуальних навчальних модулів. Він використовується освітніми системами для побудови траєкторій подальшого навчання.

Дана математична модель слугує ядром інтелектуальної частини системи та організує гнучке адаптивне управління навчальною діяльністю з максимальною для респондента швидкістю, а отримання знань найбільш повно відповідає індивідуальним особливостям.

Нехай n-ий модуль (1, 2, ..., n) складається m (1, 2, ..., m) навчальних тем, кожна з яких може відповідати P-ій кількості слів, p=1,…,P. Кожному слову передбачено коефіцієнт кількості повторень даного слова k_p (чим більше k_p, тим краще респондент засвоїв дане слово).

Кожна тема вивчається в певний період часу і фіксується в системі як $t_{ост.повт.}$ – час, коли в останнє користувач вивчав дану тему. Для визначення Δt — кількості минулого часу від початку проходження теми скористаємося залежністю

$$\Delta t = t_{повт.сьог.} - t_{ост.повт.}, \quad (2.6)$$

де $t_{повт.сьог}$ — час на сьогоднішній день;

$t_{ост.повт}$ — час останнього повторення.

На основі вище викладених виразів, математична модель респондента набуває вигляду системи рівнянь (2.7), що оцінює рівень засвоєння матеріалу респондентом:

$$M_u = \begin{cases} T_1(s_1^1, \dots, s_a^1) = \frac{(k_1^1 \cdot s_1^1) + (k_2^1 \cdot s_2^1) + \dots + (k_a^1 \cdot s_a^1)}{\Delta t} \cdot 100\% \\ T_2(s_1^1, \dots, s_b^1) = \frac{(k_1^2 \cdot s_1^2) + (k_2^2 \cdot s_2^2) + \dots + (k_b^2 \cdot s_b^2)}{\Delta t} \cdot 100\% \\ \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \\ T_n(s_1^n, \dots, s_l^n) = \frac{(k_1^n \cdot s_1^n) + (k_2^n \cdot s_2^n) + \dots + (k_l^n \cdot s_l^n)}{\Delta t} \cdot 100\%. \end{cases}, \quad (2.7)$$

де $T_n(s_1^n, \dots, s_l^n)$— визначення кількості слів в *n*-ій темі;

$k_l^n \cdot s_l^n$— оцінка засвоєння слова відповідної теми;

Δt — кількість часу, що минув із моменту останнього повторення.

Результати отримуються у відсотковому значенні. Відповідно до цих результатів підсистемою, що розробляється, визначається, чи є необхідність респонденту додавати тему до плана навчання.

Є шкала успішності засвоєння слів, за якою оцінюється доцільність повторення теми у навчальному плані:

— від 90% до 100% означає, що тема вивчена і не потребує повторного вивчення – зелений статус;

— від 60% до 89% означає, тема потребує повторного вивчення, але не в першу чергу – статус жовтий;

— від 0% до 59% означає, тема потребує обов'язкового повторення, в першу чергу на повторне вивчення – статус червоний.

Отже, визначивши теми, які попадають у відсоток відповідної градації підсистема будує навчальний план для респондента.

2.4 Створення алгоритму роботи підсистеми

Процес навчання респондента розпочинається з реєстрації в системі. Це є обов'язковим і невід'ємним етапом, оскільки відбувається визначення

важливого параметру, а саме часу — t_u, год., який користувач готовий приділяти на вивчення матеріалу в день.

Система може визначити яку кількість слів (T_{count}) в середньому людина спроможна вивчити певний період часу. Щоб розрахувати дану кількість, застосовується статистика (S_{avg}), яка зберігається в базі системи:

$$S_{avg} = \frac{s_1 t_{u1} + s_2 t_{u2} + \cdots + s_n t_{un}}{n}, \quad (2.8)$$

де $s_n t_{un}$ — кількість можливих слів для вивчення у вказаний час користувачем;

n — кількість користувачів.

Отже, отримуємо для визначення кількості матеріалу, яку людина спроможна в середньому вивчити за певний час описується формулою (2.9)

$$T_{count} = S_{avg} \cdot t_u. \quad (2.9)$$

Визначившись з кількістю матеріалу, підсистема формує навчальний план.

Формування плану відбувається не з початку навчання користувача, так як він сам обирає собі теми для проходження.

З урахуванням всіх параметрів і застосовуючи систему рівнянь 2.7 для кожної теми, веб-система створює план навчання.

Матеріал у плані навчання респондента подається у 5 режимах (рис. 2.8):

Study Modes
Cards
Word-translation
Translation-word
1of4images
1of4words

Рисунок 2.8 — Режими відображення навчання

Ці режими система випадковим чином визначає для відповідного матеріалу. Розглянемо їх детальніше:

— cards – режим, де слово подається англійською і під час натискання на картку, вона перегортається і відображається переклад і картинка;

— word-translation – режим, де відображено слово на англійській мові, картинка і поле для введення його перекладу українською;

— translation-word – режим, у якому відображено слово на українській мові, картинка і поле для введення його перекладу англійською;

— 1of4images – режим, де надається зображення і для нього перелік з 4 слів, одне з яких є правильним варіантом відповіді для заданої картинки;

— 1of4words – режим, в якому надається слово з картинкою, до якого надано перелік варіантів з 4 слів, одне з яких є правильним варіантом перекладу.

Кожен респондент після реєстрації проходить опитування, щоб система оцінила його приблизні початкові характеристики. На основі його індивідуальних навчальних матеріалів, та результатів опитування, розрахуємо кількість слів, з яких побудуємо навчальний план на поточний день.

Використовуючи запропоновану вище математичну модель засвоєння матеріалу респондентом, отримаємо список слів, кількість якої обмежена лімітом поточного дня, та на його основі згенеруємо індивідуальний навчальний план. Час на виконання обмежений поточним днем.

Таким чином, через певний період часу система знову формує навчальний план, на основі отриманих даних респондента. Алгоритм роботи зображено на рис. 2.9.

Висновки до розділу 2

У даному розділі магістерської кваліфікаційної роботи здійснено моделювання та проєктування підсистеми коригування процесу навчання.

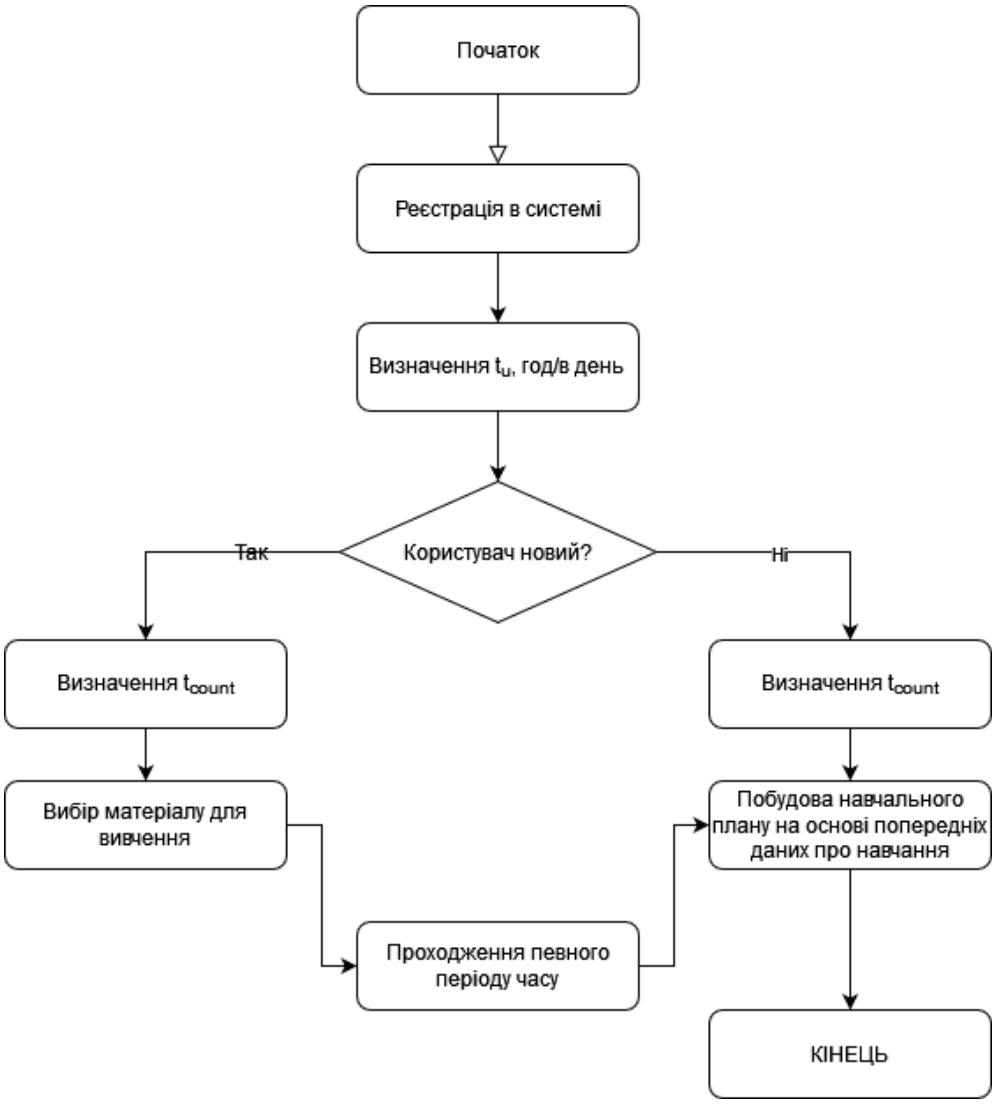


Рисунок 2.9 — Алгоритм роботи підсистеми коригування процесу навчання

Запропоновано підхід до побудови навчального плану.

Застосування математичної моделі Еббінгауза дозволяє розраховати оптимальне навантаження на респондента та час, необхідний до наступного повторення пройденого матеріалу і його кращого засвоєння.

Досліджено механізм роботи пам’яті та на його основі розроблено математичну модель засвоєння матеріалу респондентом у підсистемі коригування процесу навчання.

Математична модель респондента відслідковує загальний показник його успішності у системі на основі статистичних даних по кожній із пройдених тем, з урахуванням успішності всіх елементів у ній. Результатом роботи такої моделі

є формування пріоритетних задач для побудови індивідуального навчального плану.

Було запропоновано алгоритм взаємодії вище описаних математичних моделей, що дозволяє розробити програмний код для підсистеми коригування процесу навчання у межах системи дистанційного вивчення слів англійської мови.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ПІДСИСТЕМИ КОРИГУВАННЯ ПРОЦЕСУ НАВЧАННЯ

3.1 Обґрунтування методу розроблення веб-системи

Програмування веб-систем вимагає інтеграції трьох ключових елементів у закодовану програму: HTML, CSS, JavaScript та певними додатками (бібліотеками, фреймворками). Це спрощує створення сайту та підвищує його функціональність [29].

Для створення веб-системи необхідно виконати таке:

— визначення концепції;

— визначення проблеми, яку система повинна вирішити;

— планування робочого процесу

— створення алгоритму, за яким буде видно детальні кроки розробки;

— створення прототипу;

— визначення технологій для побудови;

— тестування.

Ще одним важливим етапом є розгортання веб-системи.

Підготовку можна розділити на два кроки: перший потребує окремої збірки або компіляції, а другий — безпосередній запуск на серверу, який ще називають хостингом, для цілодобової можливості користувачам мати доступ до системи.

Одним із способів налагодити спілкування між клієнтом та сервером використовують динамічні HTML-сторінки. Це такі сторінки, що мають здатність змінювати своє представлення в режимі реального часу в залежності від їх стану [29].

Для створення динамічних HTML-сторінок зазвичай використовуються спеціальні розширення веб-серверів, які називаються скриптами. Типовим завданням для сценарію є отримання інформації з якогось зовнішнього джерела

та представляють їх у форматі HTML на сервері перед передачею клієнту. Крім того, скрипти можна використовувати в інтерактивному режимі, де вони взаємодіють з даними, що пересилаються між клієнтом і сервером.

Створення статичної веб-сторінки є першим кроком у створенні веб-документа. Це робиться за допомогою створення спеціальних інструментів для розробки HTML.

Друге завдання передбачає розроблення інструментів, що розширюють можливості веб-сервера, і полягає у реалізації можливості динамічної зміни вмісту сторінок у поєднанні з можливістю інтерактивної роботи.

Третя проблема полягає в створенні розширень на стороні сервера з використанням різноманітних інструментів розробки. У процесі створення веб-сторінки прийнято розрізняти дві складові:

— веб-дизайн;

— веб-програмування.

Між ними немає чіткої межі. Найчастіше під веб-дизайном розуміють розробку статичної частини веб-сторінки в HTML. Включення фрагментів JS в HTML-документ зазвичай є доменом веб-програмування. Розроблення розширень для веб-сервера належить виключно до області веб-програмування.

Веб-програми використовують протокол HTTP для спілкування. Для передачі бінарних файлів по протоколу HTTP використовується специфікація Multipurpose Internet Mail Extension (MIME) [29].

MIME — це розширення оригінального протоколу електронної пошти Simple Mail Transport Protocol (SMTP). Це дозволяє користувачам обмінюватися різними типами файлів даних, включаючи аудіо, відео, зображення та прикладні програми, через електронну пошту. На відміну від SMTP, MIME підтримує надсилання як тексту ASCII, так і даних, що не є ASCII, електронною поштою. Для тексту в наборах символів, відмінних від ASCII, потрібен протокол MIME. Базову схему роботи веб-додатків зображено на (рис. 3.1).

Рисунок 3.1 — Базова схема роботи веб-додатку

3.2 Вибір технологій створення розробки та мов програмування

На даний момент існує безліч технологій, що реалізують логіку веб-додатків на кожному з етапів. Для виконання програмної реалізації підсистеми для коригування процесу навчання дамо перевагу таким технологіям.

ASP.NET Core — вільне та відкрите програмне забезпечення каркас веб застосунків, з продуктивністю вищою ніж у ASP.NET, розроблена корпорацією Microsoft і співтовариством. Це модульна структура, яка працює як на повній платформі .NET Framework, так і на платформі .NET Core.

Фреймворк являє собою повний перепис, який об'єднує раніше окремі ASP.NET MVC та ASP.NET Web API у єдину програмувальну модель.

Не зважаючи на те, що це є новим фреймворком, побудованим на новому веб стеку, ASP.NET Core має високий ступінь сумісності концепцій з ASP.NET MVC, який об'єднує функціональність MVC, Web API та Web Pages.

В попередніх версіях платформи дані технології реалізовані окремо і тому містять багато дублювальної функціональності. Тепер це об'єднано в одну програмну модель ASP.NET Core MVC.

Вебформи повністю вийшли в минуле. Програми ASP.NET Core підтримують програмні версії, в якій різні програми, що працюють на одному комп'ютері, можуть орієнтуватися на різні версії ASP.NET Core. Це не можливо з попередніми версіями ASP.NET Core [30].

ASP.NET Core Web API представляє реалізацію патерну REST, у якому кожному за типу http-запиту (GET, POST, PUT, DELETE) призначений окремий ресурс. Такі ресурси визначаються у вигляді методів контролера Web API. Ця модель особливо підходить для односторінкових програм [30]. Архітектуру платформи ASP.NET Core зображено на рис 3.2.

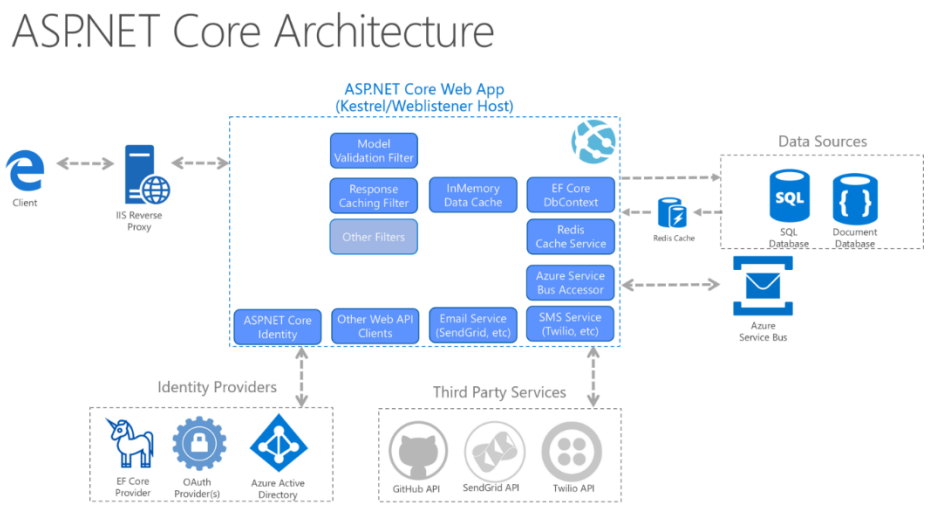


Рисунок 3.2 — Архітектура ASP.NET Core

Angular (версія 2 і вище) — відкрита і вільна платформа для розробки веб-застосунків, написана мовою TypeScript, що розробляється командою з Google, а також спільнотою розробників з різних компаній. Angular повністю переписаний фреймворк від тієї ж команди, яка написала AngularJS [31].

Angular являє собою платформу та фреймворк для створення односторінкових клієнтських програм за допомогою HTML і TypeScript. Angular написаний на TypeScript. Він реалізує основні та додаткові функції як набір бібліотек TypeScript, які ви імпортуєте у свої програми.

Архітектура програми Angular спирається на певні фундаментальні концепції. Основними будівельними блоками фреймворку Angular є компоненти Angular, які організовані в NgModules. NgModules збирають пов’язаний код у функціональні набори; додаток Angular визначається набором NgModules. Додаток завжди має принаймні кореневий модуль, який уможливлює початкове

завантаження, і зазвичай має набагато більше модулів функцій. Архітектуру програми Angular зображено на рис. 3.3.

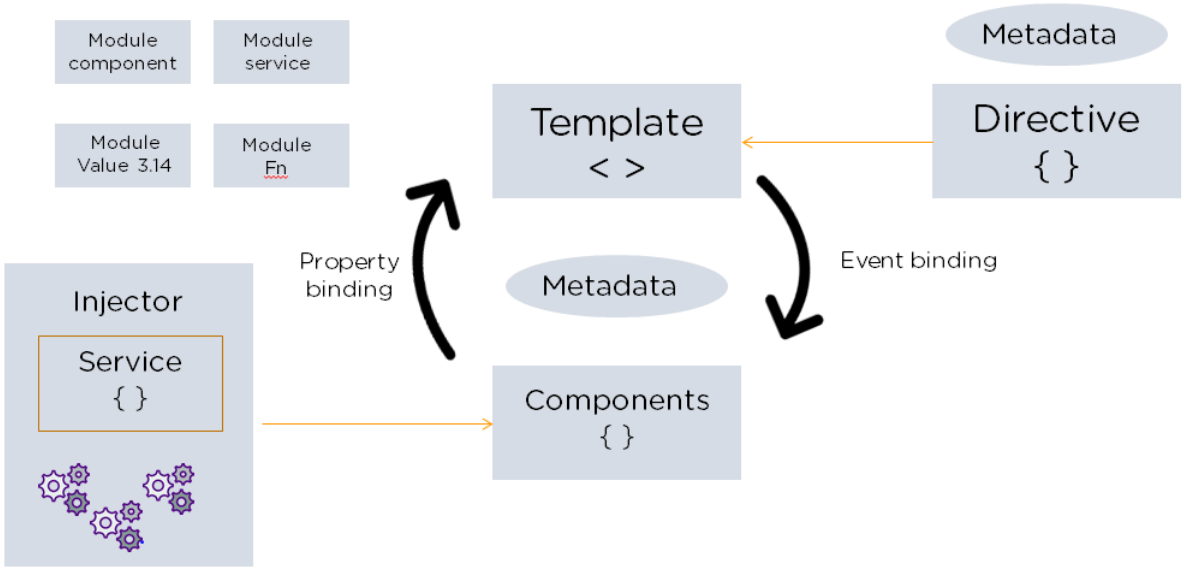


Рисунок 3.3 — Архітектура програми Angular

Компоненти визначають представлення, які є наборами елементів екрану, які Angular може вибирати та змінювати відповідно до логіки та даних вашої програми.

Компоненти використовують служби, які надають певні функції, не пов’язані безпосередньо з представленнями. Постачальники послуг можна вставляти в компоненти як залежності, роблячи ваш код модульним, придатним для повторного використання та ефективним.

Модулі, компоненти та служби — це класи, які використовують декоратори. Ці декоратори позначають свій тип і надають метадані, які вказують Angular, як їх використовувати.

Метадані для класу компонента пов’язують його з шаблоном, який визначає представлення. Шаблон поєднує звичайний HTML з директивами Angular і розміткою прив’язки, що дозволяє Angular змінювати HTML перед його візуалізацією для відображення.

Метадані для класу обслуговування надають інформацію, необхідну Angular, щоб зробити її доступною для компонентів через впровадження залежностей (DI) [32].

Компоненти програми зазвичай визначають багато представлень, упорядкованих ієрархічно. Angular надає службу Router, щоб допомогти вам визначити шляхи навігації між видами. Маршрутизатор надає складні можливості навігації в браузері.

SQL(Structured Query Language) Server — одна з найпопулярніших систем управління базами даних у світі. Система управління базами даних використовується для розробки великої кількості проєктів, включаючи прості додатки та великі системи.

Характерні особливості SQL Server:

— висока продуктивність;

— надійність;

— простота використання.

Основним об'єктом MS SQL Server та будь-якої іншої СУБД є база даних. База даних — це сховище даних, організоване певним чином. Фізично кажучи, база даних — це зазвичай файл, що зберігається на жорсткому диску. Для управління базою даних потрібна система управління.

MS SQL Server використовує реляційну модель для організації бази даних. Сьогодні він фактично став стандартом організації баз даних. Реляційна модель передбачає зберігання даних у вигляді таблиць, і кожна таблиця складається з рядків і стовпців. Кожен рядок зберігає окремий об'єкт, а стовпці містять атрибути цього об'єкта. Первинний ключ використовується для ідентифікації рядків у таблиці. Первинний ключ може містити один або кілька стовпців. Використовуючи первинний ключ, ви можете посилатися на конкретні рядки таблиці. Тому кожен рядок має свій первинний ключ. Також потрібен ключ для прив'язки однієї таблиці до іншої, тобто для встановлення зв’язків у базі даних.

SQL Server використовує для роботи з базами даних. Клієнт (можливо, зовнішня програма) використовує необхідний API для створення та надсилання запитів у SQL. База даних обробляє і перетворює запит, і відправляє назад результат виконання запиту [33].

3.3 Проєктування програмних засобів підсистеми коригування процесу навчання

У попередньому розділі було обгрунтовано технології, які використовуватимуться для побудови веб-системи. Даний підрозділ присвячено безпосередньо реалізації підсистеми.

Розберемо детально кожний етап роботи підсистеми на прикладі побудови навчального плану респондента від клієнтської частини до бази даних.

Angular складається з окремих модулів. Як правило, програми складаються з декількох модулів. Кожен модуль являє собою окрему, незалежну одиницю застосунку, яку ми можемо імпортувати та перевикористовувати де завгодно. Кожен додаток Angular має як мінімум один кореневий модуль (root module).

Згенеруємо новий модуль на дамо йому назву personal-plan. Лістинг файлу модуля personal-plan зображено на рис. 3.4.

```
@NgModule({
  declarations: [
    PersonalPlanComponent
  ],
  imports: [
    CommonModule,
    PersonalPlanRoutingModule,
    MatProgressBarModule,
    MatIconModule,
    MatButtonModule,
    MatCardModule
  ]
})
export class PersonalPlanModule { }
```

Рисунок 3.4 — Лістинг файлу модуля personal-plan

NgModule представляє функцію-декоратора, яка приймає об'єкт, властивості якого описують метадані модуля. Найважливіші властивості [34]:

— declarations класи уявлень (view classes), що належать модулю. Angular має три типи класів уявлень: компоненти (components), директиви (directives), канали (pipes);

— exports набір класів уявлень, які мають використовуватися у шаблонах компонентів з інших модулів;

— imports інші модулі, класи яких необхідні для шаблонів компонентів із поточного модуля;

— providers класи, що створюють послуги, що використовуються модулем;

— bootstrap кореневий компонент, який викликається за замовчуванням під час завантаження програми.

Єдиним класом представлення модуля персонального плану є компонент PersonalPlanComponent, тому він вказується на властивості declarations. І, оскільки його дія залежить від списку інших модулів, то дані модулі вказуються для властивості imports.

Одним із ключових елементів програми є компоненти. Компонент керує відображенням подання на екрані. Компонент складається з html, css, js файлів. Оскільки в нашому прикладі ми використовуємо бутстрап для стилізації шаблонів, тому файли css в нас пусті і не будуть використовуватись в якості прикладів Розглянемо інші файли компонента детальніше. Лістинг html файлу компонента PersonalPlanComponent зображено на рис. 3.5.

Html файл компонента являє собою невеликий фрагмент коду, який ми можемо імпортувати в інші компоненти через визначений селектор. Головною перевагою такого підходу являється повторне перевикористання програмного коду. Замість сотень рядків коду, достатньо написати лише один, і решта буде імпортована до іншого компоненту, замість вказаного селектора.

Він працює в парі з js файлом компонента, для представлення даних, що ми отримуємо з серверної частини. Лістинг js файлу компонента PersonalPlanComponent зображено на рис. 3.6.

```
<div class="container">
    <div class="text-center mb-4">
        <div class="fs-3">Особистий план</div>
        <div class="fs-6">Виконуйте завдання, збирайте медалі, ви зможете обміняти їх на призи</div>
        <div class="fs-6">Ви займаєтесь <strong>7 раз на тиждень</strong></div>
    </div>
    <div class="d-flex justify-content-between align-items-center">
        <div>
            <div><strong>40% плану виконано</strong></div>
            <mat-progress-bar mode="determinate" value="40"></mat-progress-bar>
        </div>
        <div class="d-flex align-items-center">
            <div>
                <button mat-icon-button aria-label="<">
                    <mat-icon>arrow_back_ios</mat-icon>
                </button>
            </div>
            <div class="ms-4 me-4 fs-5">
                <strong>16 грудня</strong>
            </div>
            <div>
                <button mat-icon-button aria-label=">">
                    <mat-icon>arrow_forward_ios</mat-icon>
                </button>
            </div>
        </div>
        <div class="d-flex">
            <div>Ви вже заробили:</div>
            <mat-icon class="ms-2 me-1">workspace_premium</mat-icon>
            <div><strong>X</strong>64</div>
        </div>
    </div>

    <div class="mt-4">
        <mat-card class="mt-2" *ngFor="let item of items">
            <mat-card-content class="p-2 d-flex align-items-center justify-content-between">
                <div>
                    <img height="60" src="../../../assets/img/logo.png" alt="" class="me-4">
                    <a href="#" class="text-decoration-none fs-5"></a>
                </div>
                <div>
                    battary-place
                </div>
                <div>25 хвилин</div>
                <div class="d-flex align-items-center">
                    <mat-icon class="me-1">workspace_premium</mat-icon>
                    <div><strong>X</strong>5</div>
                </div>
            </mat-card-content>
```

Рисунок 3.5 — Лістинг html файлу компонента PersonalPlanComponent

Щоб клас міг використовуватися в інших модулях, він визначається ключовим словом export . У самому ж класі визначено лише одну змінну, яка як значення зберігає певний рядок.

Для створення компонента необхідно імпортувати функцію декоратора @Component з бібліотеки @angular/core. Декоратор @Component дає змогу ідентифікувати клас як компонент.

```
import { Component, OnInit } from '@angular/core';
import { ThemePalette } from '@angular/material/core';
import { ProgressBarMode } from '@angular/material/progress-bar';

@Component({
  selector: 'app-personal-plan',
  templateUrl: './personal-plan.component.html',
  styleUrls: ['./personal-plan.component.scss']
})
export class PersonalPlanComponent implements OnInit {
  color: ThemePalette = 'primary';
  mode: ProgressBarMode = 'determinate';

  constructor() { }

  ngOnInit(): void {
  }
}
```

Рисунок 3.6 — Лістинг js файлу компонента PersonalPlanComponent

Якби не було застосовано декоратор @Componentдо класу PersonalPlanComponent, то клас PersonalPlanComponent компонентом не вважався б.

Фреймворк визначає, як працювати з компонентом, передаючи його об’єкт конфігурації через параметр декоратора @Component.

Шаблон відображає попередньо визначений фрагмент HTML-коду, який є основою для проєкту Angular. Шаблони складаються як з HTML-коду, так і з коду Angular, вони забезпечують основу для взаємодії користувачів із програмою [34].

Визначення шаблону безпосередньо через властивість template не є обов’язковим для кожного шаблону компонента. Замість цього можна використовувати зовнішній HTML для розмітки ваших шаблонів і підключення їх через властивість templateUrl. Багаторядковий шаблон пропонує косі лапки, які не слід плутати з однорядковими лапками.

Також у прикладі вище встановлюється властивість selector , яке визначає селектор CSS. В елемент з цим селектором Angular додаватиме уявлення компонента. Наприклад, у прикладі вище селектор має значення app-personal-

plan. У самому ж класі визначено лише одну змінну, яка як значення зберігає певний рядок.

Маршрутизація дозволяє зіставляти запити до програми з певними ресурсами всередині програми.

Ключовим для роботи маршрутизації є модуль RouterModule , який знаходиться в пакеті @angular/router . Тому при роботі з маршрутизацією цей пакет має бути вказаний у списку залежностей у файлі package.json. Лістинг файлу package.json зображено на рис. 3.7.

Рисунок 3.7 — Лістинг файлу package.json

Далі необхідно створити модуль маршрутизації. Назвемо його PersonalPlanRoutingModule. Як базова адреса буде розглядатися корінь програми. Кожен маршрут порівнюється з певним компонентом. Для кожного із цих компонентів ми можемо визначити свій маршрут [35]. Для цього змінимо код модуля. Лістинг файлу PersonalPlanRoutingModule зображено на рис. 3.8.

Імпортуємо модуль маршрутизації RouterModule та клас Routes, що представляє колекцію маршрутів. Тут визначено три маршрути, кожен із яких оброблятиметься окремим компонентом. Для вказівки маршруту використовується параметр path. Наприклад, шлях із порожнім значенням представлятиме

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { PersonalPlanComponent } from './personal-plan/personal-plan.component';

const routes: Routes = [
  {
    path: '',
    component: PersonalPlanComponent
  }
];

@NgModule({
  imports: [RouterModule.forChild(routes)],
  exports: [RouterModule]
})
export class PersonalPlanRoutingModule { }
```

Рисунок 3.8 — Лістинг файлу PersonalPlanRoutingModule

запит типу http://localhost:3000/ і оброблятиметься класом PersonalPlanComponent.

Крім локалізованого файлу маршрутизації модуля побудови навчального плану, реалізуємо глобальну маршрутизацію підсистеми, в яку імпортуємо нашу вище описану локалізовану маршрутизацію. Лістинг файлу глобальною маршрутизації підсистеми зображено на рис. 3.9.

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { LayoutComponent } from './layout/layout/layout.component';

const routes: Routes = [
  {
    path: '',
    component: LayoutComponent,
    children: [
      { path: '', loadChildren: () => import('./personal-plan/personal-plan.module').then(m => m.PersonalPlanModule)},
      { path: 'topics', loadChildren: () => import('./topic/topic.module').then(m => m.TopicModule)},
    ]
  },
  {
    path: 'auth', loadChildren: () => import('./auth/auth.module').then(m => m.AuthModule)
  }
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Рисунок 3.9 — Лістинг файлу глобальної маршрутизації

У даному прикладі обраний компонент сам повинен приймати як внутрішній вміст якийсь інший компонент залежно від запиту. У цьому випадку нам треба використовувати дочірні маршрути Child Route.

Якщо необхідно організувати маршрутизацію таким чином, щоб спочатку підвантажувався LayoutComponent, а в середині нього PersonalPlanComponent, як результат кожен із дочірніх маршрутів зіставляється лише з частиною адреси. Далі, щоб застосувати такі маршрути, у маршруту для компонента LayoutComponent застосовується властивість loadChildren, що імпортує наші локалізовані файли маршрутизації окремих модулів і підставляє в файл глобальної маршрутизації.

Компоненти можуть напряму реалізовувати методи для надсилання запитів на серверну сторону додатку, але такий підхід порушує правила гарного тону та деякі принципи програмування, тому для правильної структуризації даного процесу створимо та налаштуємо два сервіси.

Сервіси в Angular представляють досить широкий спектр класів, які виконують деякі специфічні завдання, наприклад логування, роботу з даними.

На відміну від компонентів і директив, сервіси не працюють з уявленнями, тобто з розміткою html, не надають на неї прямого впливу. Вони виконують строго певне і досить вузьке завдання.

Стандартні завдання сервісів:

— надання даних додатку;

— представлення каналу взаємодії між окремими компонентами програми;

— інкапсулювання бізнес-логіки, різноманітні обчислювальні завдання, завдання з логування, які краще виносити з компонентів.

Сервіс може сам зберігати дані в пам'яті або для отримання даних може звертатися до якогось джерела даних, наприклад, до сервера. Таким чином, код компонентів буде зосереджений безпосередньо на роботі з поданням. Крім того,

тим самим можна вирішити проблему повторення коду, якщо потрібно виконати те саме завдання в різних компонентах і класах.

Створимо перший сервіс для інкапсулювання бізнес-логіки модуля побудови навчального плану. Лістинг файлу PersonalPlanService зображено на рис. 3.10.

```
@Injectable({
  providedIn: 'root'
})
export class PersonalPlanService {
  constructor(private accessor: PersonalPlanAccessorService) { }

  async getPersonalPlan(studentId: number): Promise<PersonalPlan | undefined> {
    return await this.accessor.getPersonalPlan(studentId);
  }
}
```

Рисунок 3.10 — Лістинг файлу PersonalPlanService

Для отримання даних, необхідно отримати доступ до функціоналу іншого сервісу, а саме PersonalPlanAccessorService.

Для реалізації даної задачі необхідно використати впровадження залежностей використати впровадження залежності. Впровадження залежності обробляється ядром Angular автоматично, для реалізації нам необхідно в конструктор даного сервісу передати сервіс, функціонал якого планується до використання в межах сервісу, що реалізується.

Щоб вказати, що сервіс може використовуватися в інших сервісах, клас сервісу застосовується декоратор Injectable. Даний декоратор гарантує, що вбудований механізм впровадження залежностей зможе створити об'єкт цього класу і передати його в залежності від іншого об'єкта (в інший сервіс або компонент).

Існує загальна рекомендація від розробників Angular застосовувати Injectable до будь-якого класу сервісу, адже у практичному плані даний декоратор необхідний, якщо сервіс, що впроваджується, сам має деякі залежності [36].

Далі для отримання навчального плану необхідно створити асинхронний метод сервісу getPersonalPlan, який як аргумент приймає ідентифікатор респондента, та завдяки сервісу аксесора, залежність якого впроваджено на минулому етапі, поверне як результат виконання метода – персональний план респондента.

Наступним кроком буде реалізація безпосередньо сервісу аксесора. Лістинг файлу PersonalPlanAccessorService зображено на рис. 3.11.

```
@Injectable({
  providedIn: 'root'
})
export class PersonalPlanAccessorService {
  private baseUrl: string = 'http://localhost:3000/';

  constructor(private http: HttpClient) { }

  getPersonalPlan(studentId: number): Promise<PersonalPlan | undefined> {
    return this.http.get<PersonalPlan>(this.baseUrl + `personal-plan/${studentId}`).toPromise();
  }
}
```

Рисунок 3.11 — Лістинг файлу PersonalPlanAccessorService

Аналогічно до попереднього етапу створимо сервіс PersonalPlanAccessorService.

Для реалізації задачі даного сервісу необхідно впровадити залежність HttpClient сервісу, який, у свою чергу, являє собою один із багатьох сервісів різноманітних модулів безпосередньо Angular, в даному випадку HttpClientModule, та вміє надсилати запити за критеріями, які надаємо його методам.

Створимо змінну, що зберігатиме в собі доменне посилання, на якому хоститься серверна частина додатку, а саме http://localhost:3000/.

Реалізуємо метод сервісу, який надсилає http запит типу get на серверну частину підсистеми, на PersonalPlanController з параметром ідентифікатора респондента, обгорнемо запит у проміс, для подальшої реалізації асинхронності запитів.

Контролери в Web API приймають запит у вигляді об'єкта HttpRequestMessage, обробляють його за допомогою одного з методів і посилають у відповідь результат оброблення у вигляді об'єкта HttpResponseMessage.

HttpRequestMessage за допомогою своїх властивостей передає низку даних про запит:

— content, що повертає об'єкт HttpContent, який представляє тіло запиту;

— headers, що повертає набір заголовків запиту як об'єктів HttpRequestHeader;

— method, що містить інформацію про тип запиту (Get/Post/Put/Delete);

— properties являє собою словник, що містить об'єкти, що надаються хостинговим середовищем;

— requestUri являє собою запитаний ресурс URL;

— version, що містить інформацію про версія протоколу HTTP.

Використовуючи властивість контролера ControllerContext, що представляє об'єкт HttpControllerContext, містить дані запиту, як і властивість RequestContext, крім цього і властивість ControllerContext також містить деякі дані про контролер.

HttpControllerContext визначає такі властивості:

— configuration, що представляє об'єкт HttpConfiguration, який містить інформацію про конфігурацію програми;

— controller, що повертає контролер, що обробляє поточний запит;

— controllerDescriptor, що містить об'єкт HttpControllerDescriptor, який зберігає інформацію про контролера;

— request, зберігає об'єкт Http.HttpRequestMessage, що надає інформацію про запит;

— requestContext, що представляє об'єкт HttpRequestContext, що містить інформацію про поточний запит, специфічну для Web API;

— routeData, що зберігає дані маршруту.

Використовуючи контекст даних, можна отримати всю необхідну інформацію на запит [36].

Лістинг файлу PersonalPlanController зображено на рис. 3.12.

```
using Microsoft.AspNetCore.Mvc;

namespace EngurAPI.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    1 reference | 0 changes | 0 authors, 0 changes
    public class PersonalPlanController : ControllerBase
    {
        private readonly IPersonPlanService _personPlanService;

        0 references | 0 changes | 0 authors, 0 changes
        public PersonalPlanController(IPersonPlanService personPlanService)
        {
            _personPlanService = personPlanService;
        }

        [HttpGet("{studentId}")]
        0 references | 0 changes | 0 authors, 0 changes
        public async Task<IResult> GetPersonalPlanAsync(int studentId)
        {
            var plan = await _personPlanService.GetPersonalPlanAsync(studentId);

            return plan == null ? Results.NotFound() : Results.Ok(plan);
        }
    }
}
```

Рисунок 3.12 — Лістинг файлу PersonalPlanController

Web Api контролер повинен наслідувати логіку ControllerBase, мати в назві ключове слово Controller, та мати деяку спецефічну анотацію даних як до самого контролера, так і до його методів.

Створимо метод GetPersonalPlan, що буде реалізовувати http get запит, та повертати нам результат. Для гарної продуктивності серверної частини підсистеми методи контролера повинні бути асинхронні.

Для отримання безпосередньо даних з бази даних нам необхідно аналогічно до впровадження залежностей клієнтської частини реалізувати подібний механізм на базі платформи .Net для відокремленого сервісу. Однак на відміну від ядра Angular, впровадження залежностей на базі платформи .Net потрібно вказувати явно. Тобто необхідно в файлі конфігурації вписати, який

саме екземпляр класу, що реалізовує інтерфейс, повинен бути підставлений середовищем на його місце. Лістинг файлу конфігурації впровадження залежностей зображено на рис. 3.13.

```
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

builder.Services.AddDbContext<EngurDbContext>(opt => opt.UseSqlServer(
    builder.Configuration.GetConnectionString("EngurDbConnection")));
builder.Services.AddScoped<IUnitOfWork, UnitOfWork>();
builder.Services.AddScoped<IWordRepository, WordRepository>();
builder.Services.AddScoped<IPersonPlanRepository, PersonPlanRepository>();
builder.Services.AddScoped<IWordService, WordService>();
builder.Services.AddScoped<IPersonPlanService, PersonPlanService>();

var app = builder.Build();
```

Рисунок 3.13 — Лістинг файлу конфігурації впровадження залежностей

Використовуючи різні способи впровадження залежностей, можна керувати життєвим циклом створюваних сервісів.

Сервіси, що створюються механізмом впровадження залежностей, можуть представляти один із трьох типів, в залежності від призначення сервісу.

Протягом одного запиту може бути кілька звернень до сервісу, відповідно при кожному зверненні створюватиметься новий об'єкт. Подібна модель життєвого циклу найбільше підходить для легковажних сервісів, які не зберігають даних про стан, для даного виду використовується transient.

Якщо протягом одного запиту є кілька звернень до одного сервісу, і при всіх цих зверненнях використовуватиметься той самий об'єкт сервісу, для даного виду використовується scoped.

Якщо об'єкт сервісу створюється при першому зверненні до нього, всі наступні запити використовують той самий раніше створений об'єкт сервісу, тоді було використано singleton.

Для створення кожного типу сервісу призначений відповідний метод AddTransient(), AddScoped() та AddSingleton().

Одним із найчастіше використовуваних патернів під час роботи з даними є патерн репозиторій.

Репозиторій дозволяє абстрагуватися від конкретних підключень до джерел даних, з якими працює програма, і є проміжною ланкою між класами, що безпосередньо взаємодіють з даними, та іншою програмою, іншими словами інкапсулює роботу з джерелом даних.

Зазвичай патерн репозиторій реалізовують разом з патерном Unit of Work.

Патерн Unit of Work дозволяє створити централізовану точку доступу до репозиторіїв, можна провести аналогію з папкою з файлами: одна папка – багато файлів, всі файли в одній папці.

Припустимо, є одне підключення до бази даних MS SQL Server. Однак, якщо в якийсь момент часу необхідно змінити підключення з MS SQL на інше, наприклад, до БД MySQL або MongoDB, то завдяки централізації репозиторіїв при використанні Unit of Work заміна одного вендора бази даних на інший не створить зайвих труднощів.

При стандартному підході, навіть у невеликому додатку, який здійснює вибірку, додавання, зміну та видалення даних, довелося б зробити велику кількість змін, або у процесі роботи програми, залежно від різних умов, можна використовувати два різні підключення.

Таким чином, репозиторій додає програмі гнучкість під час роботи з різними типами підключень.

Лістинг файлу репозиторія навчального плану зображено на рис. 3.14.

Клас репозиторія навчального плану містить в собі метод для отримання плану з контексту бази даних, який ми за допомогою механізму впровадження залежностей передаємо до класу. Метод є асинхронним, приймає як параметр ідентифікатор респондента і повертає об’єкт навчального плану як результат виконаної роботи [37].

```
using EngurDAL.Data;
using EngurDAL.Entities;
using EngurDAL.Interfaces;
using Microsoft.EntityFrameworkCore;

namespace EngurDAL.Repositories
{
    3 references | 0 changes | 0 authors, 0 changes
    public class PersonPlanRepository : IPersonPlanRepository
    {
        private readonly EngurDbContext _context;

        1 reference | 0 changes | 0 authors, 0 changes
        public PersonPlanRepository(EngurDbContext context)
        {
            _context = context;
        }

        0 references | 0 changes | 0 authors, 0 changes
        public async Task<PersonPlan?> GetPersonPlanAsync(int studentId)
        {
            return await _context.PersonPlans
                .FirstOrDefaultAsync(student => student.Id == studentId);
        }

        0 references | 0 changes | 0 authors, 0 changes
        public void Dispose()
        {
            _context.Dispose();
        }
```

Рисунок 3.14 — Лістинг файлу репозиторія навчального плану

Entity Framework — це технологія в ADO.NET для доступу до даних та надає можливість взаємодії з об’єктами бази даних засобами мови інтегрованих запитів (Language Integrated Query).

Технологія Entity Framework може використовуватися спільно з концепцією сутностей і дозволяє використовувати для оброблення об'єкти замість таблиць, а також використовується для оброблення даних.

Ядро Entity Framework — це обгортка, яка з'єднує базу даних з об'єктно-орієнтованою парадигмою для створення бази даних віртуальних об'єктів. Він був створений Microsoft на платформі Net.Core для розробників, щоб дозволити їм зберігати дані в додатках у реляційних базах даних.

Основним завданням Entity Framework є зберігання об’єктів у базі даних, доступ до них та забезпечення взаємодії між базою даних та програмами ASP.NET Core.

Entity Framework використовує реляційну базу даних, де дані представлені у вигляді таблиць з рядками. Запити до реляційних баз даних записуються в SQL. Entity Framework залежить від версії сервера, яка реалізує запити до бази даних. Для того, щоб отримати об'єкт, ядро .Net Entity Framework реалізує команду SQL, яка запитує сервер баз даних для отримання даних як об'єкта, а поля об'єкта заповнюють поля об'єкта .Net [38].

Для оброблення колекцій Entity Framework використовує мову LINQ, вбудовану в платформу .Net, яка може маніпулювати колекціями об'єктів .Net.

Для реалізації Entity Framework в проєкті використаємо підхід Code First.

Підхід Code First спочатку створює суть класу, використовуючи атрибути, які їх визначають. Entity Framework створить бази даних і таблиці на основі визначених класів сутності. Тому база даних генерується з коду. Після компіляції коду база даних буде створена.

Переваги підходу Code First:

— можливість створення баз даних і таблиць з бізнес-об'єктів;

— можна вказати доступні колекції, що не потрібно завантажувати та серіалізувати;

— контроль над версією бази даних;

— не має необхідності працювати на пряму з БД;

— підходить для невеликих додатків.

Недоліки підходу Code First:

— всю базу даних потрібно описати у форматі програмного коду;

— якщо вам потрібно змінити якийсь вміст у таблиці бази даних, вам потрібно змінити код, а потім запустити оновлення через консоль диспетчера пакетів, так звану міграцію;

— не підходить для додатків, які постійно змінюють свою структуру.

Принцип роботи підходу Code First зображено на рис. 3.15.

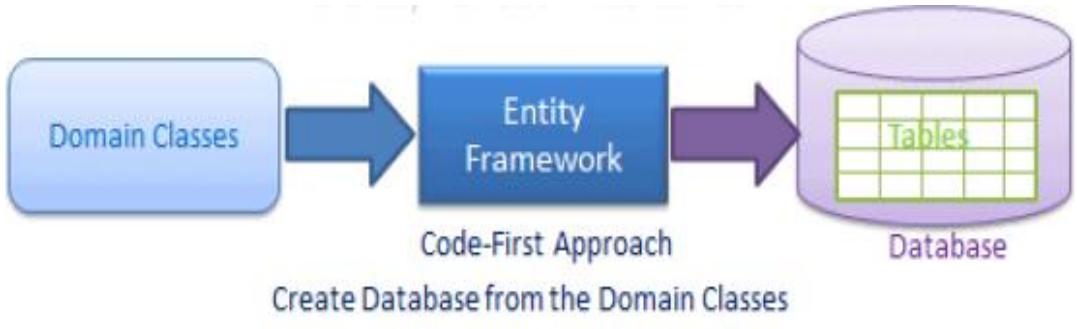


Рисунок 3.15 — Принцип роботи підходу Code-First

Для зв’язку об’єктів бази даних Entity Framework використовує так звані навігаційні властивості, що використовують одну із трьох стратегій завантаження, таких як eager loading (жадібне завантаження), explicit loading (явне завантаження), lazy loading (ледаче завантаження). На практиці частіше за все використовуються тільки дві з перерахованих стратегій, а саме жадібне та ледаче завантаження.

Жадібне завантаження допомагає завантажувати всі необхідні об’єкти одночасно, тобто всі ваші дочірні сутності будуть завантажені під час одного виклику бази даних. Цього можна досягти, використовуючи метод Include, який повертає відповідні сутності як частину запиту, і відразу завантажується великий обсяг даних.

Ледаче завантаження використовується за замовчуванням у Entity Framework, де дочірня сутність завантажується лише тоді, коли до неї звертаються вперше. Це просто затримує завантаження відповідних даних, поки не буде на це запиту.

Явне завантаження в Entity Framework — це стратегія, що досить схожа на ледаче завантаження, але, на відміну від нього, під час звернення до дочірніх сутностей, вони не будуть завантажені, проте все ще можливо завантажувати пов’язані сутності, явно викликаючи метод Load [38].

Оберемо стратегію ледачого завантаження для реалізації в проєкті.

Для початку роботи з Entity Framework та стратегією ледачого завантаження необхідно у середовищі за допомогою диспетчера керування пакетами NuGet додати до проєкту такий перелік пакетів:

— Microsoft.AspNetCore.Diagnostics.EntityFrameworkCore;

— Microsoft.EntityFrameworkCore.SqlServer;

— Microsoft.EntityFrameworkCore.Proxies.

Відповідно до підходу Code First необхідно створити моделі об’єктів, що ми хочемо зберігати у базі даних, та встановити зв’язки між ними за допомогою навігаційних властивостей. Лістинг моделі класу навчального плану зображено на рис. 3.16.

```
0 references | 0 changes | 0 authors, 0 changes
public class PersonalPlan
{
    0 references | 0 changes | 0 authors, 0 changes
    public int Id { get; set; }
    0 references | 0 changes | 0 authors, 0 changes
    public int StudentId { get; set; }
    0 references | 0 changes | 0 authors, 0 changes
    public int ComplitedPercentage { get; set; }
    0 references | 0 changes | 0 authors, 0 changes
    public DateTime CreationDate { get; set; }
    0 references | 0 changes | 0 authors, 0 changes
    public DateTime ExpirationDate { get; set; }
    0 references | 0 changes | 0 authors, 0 changes
    public Dictionary<Topic, Game>? Plans { get; set; }
}
```

Рисунок 3.16 — Лістинг моделі класу навчального плану

Клас навчального плану містить унікальний ідентифікатор та має власні властивості, такі як унікальний ідентифікатор плану і респондента, відсоток його виконання, дату створення і закінчення терміну дійсності та навігаційні властивості, що вказують на об’єкти топіку і гри, що належать даному навчальному плану.

Перейдемо до створення класу контексту бази даних, у нашому випадку ми унаслідуємось від контексту Identity, щоб отримати доступ до контексту даних респондентів та розширити можливості батьківського класу.

Перевизначимо метод OnConfiguring для ініціалізації параметрів конфігурації контексту. А саме нам необхідно вказати контексту що ми хочемо використовувати стратегію лінивого завантаження та вказати SQL Server як СУБД для контексту, в якості параметра передаємо стрічку під'єднання до бази даних. Лістинг класу контексту бази даних зображено на рис. 3.17.

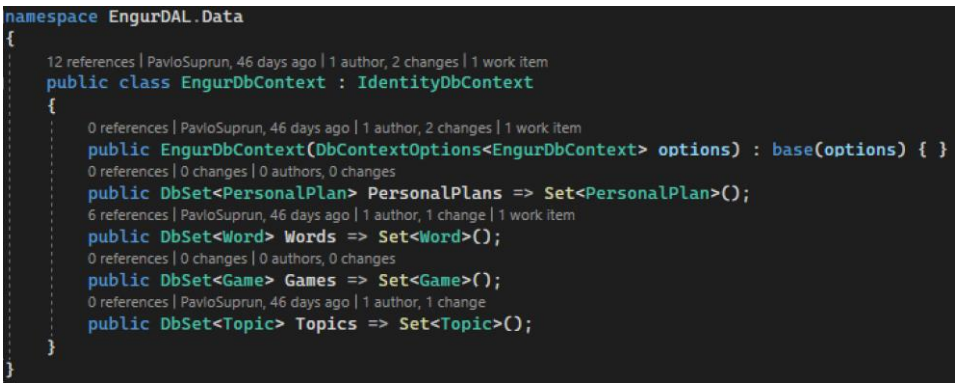


Рисунок 3.17 — Лістинг класу контексту бази даних

При першій компіляції Entity Framework на основі моделей, що було створено та додано до наборів даних у контексті створить базу даних. Структуру створеної бази даних з урахуванням ASP.NET Core Identity зображено на рис. 3.18.

Отже, підсистема поділена на три функціональні одиниці:

— клієнтська;

— серверна;

— база даних.

Клієнтська частина реалізована з використанням фреймворку Angular та складається з модулів, компонентів, сервісів, аксесорів, моделей та маршрутизацією. Структуру клієнтської частини зображено на рис. 3.19.

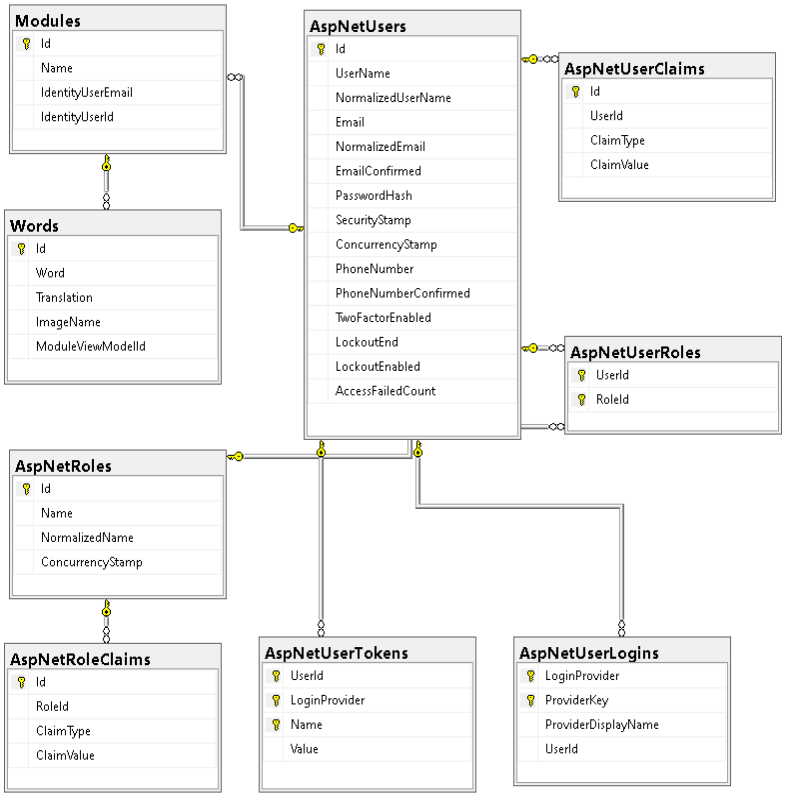


Рисунок 3.18 — Структура бази даних з урахування Identity

Серверна частина реалізована з використанням ASP .NET Core та Web API та поділена на три логічних рівні, представлених окремими проєктами: рівні Data Access Layer (DAL) та Business Logic Layer (BLL) являють собою DLL-бібліотеки, а рівень Presentation Layer (PL або API) являє собою проєкт Web API, оскільки він є основним виконуваним проєктом і інтерфейсом для взаємодії з системою. Структуру серверної частини зображено на рис. 3.20.

База даних реалізована з використанням інтегрованої системи для роботи з базами даних Entity Framework Core, Microsoft Identity, MSSQL Server та Code First Approach [38]. Що позволило нам не працювати з базою даних на пряму, а завдяки міграціям і використаним підходам працювати з базою через код, і на базі зв’язків між моделями даних в додатку створити структуру бази даних на стороні SQL серверу.

Структуру бази даних зображено на рис. 3.21.

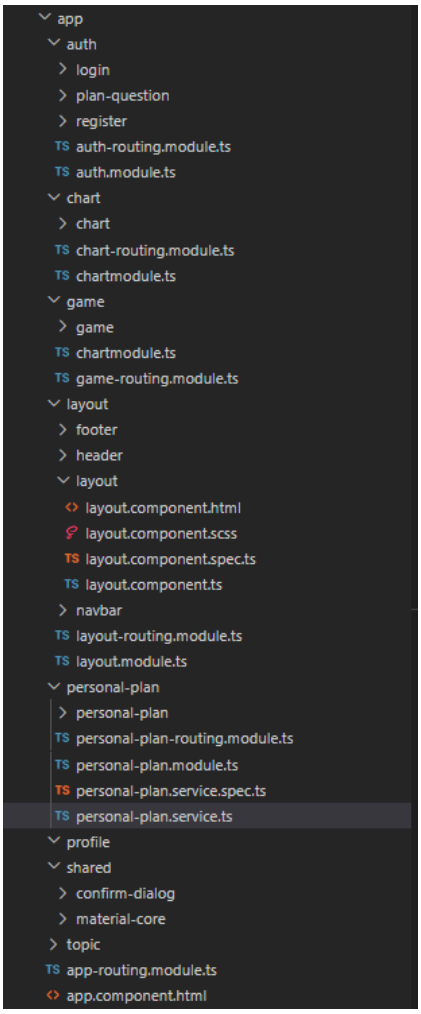


Рисунок 3.19 — Структура клієнтської частини

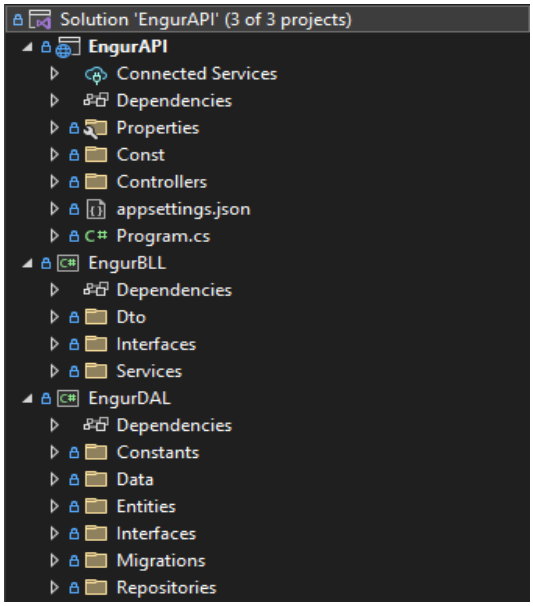


Рисунок 3.20 — Структура серверної частини

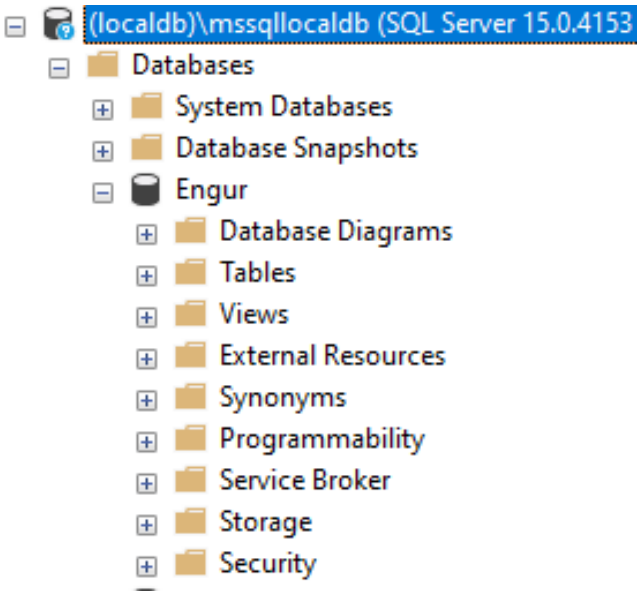


Рисунок 3.21 — Структура бази даних

Висновки до розділу 3

У даному розділі комплексної магістерської кваліфікаційної роботи описано програмну реалізацію підсистеми коригування процесу навчання у межах системи дистанційного вивчення слів англійської мови.

Насамперед, було обґрунтовано технології та мову програмування для розроблення дистанційної системи коригування процесу навчання.

Для клієнтської частини було вирішено розробляти Angular-застосунок із візуальним інтерфейсом користувача.

Для серверної частини ключовою технологією виступатиме .NET та стек його основних фреймворків. Як сервер бази даних обрано Microsoft SQL Server.

Описано проєктування програмних засобів підсистеми: обґрунтовано вибір архітектури програмного комплексу, спроєктовано базу даних, розроблено основну частину додатку та виконано взаємодію елементів усього комплексу.

У результаті роботи було розроблено застосунок, що складається з клієнтської, серверної частин та бази даних.

Клієнтська частина використовує підходи розбиття структури на модулі компонентів та їх маршрутизації.

Серверна частина базується на трирівневій архітектурі та надає інтерфейс керування програмою клієнтській частині, а також виконує основні маніпуляції з даними на їхньому шляху від клієнта до бази даних і навпаки.

4 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Функціональне тестування програмного забезпечення

Тестування — це постійний процес, який зазвичай відбувається під час і після фази створення системи. Причому можна автоматизувати тестування або зробити це вручну.

Тестування відіграє життєво важливу роль у процесі розроблення і створення якісного програмного забезпечення. Необхідно серйозно ставитися до аналізу і проєктування структурованого процесу, який забезпечить своєчасний і успішний випуск проєкту.

На етапі тестування потрібно спробувати охопити тестування функціональності, зручності використання, сумісності, безпеки та продуктивності.

Важливо пам’ятати, що довіру респондентів дуже просто втратити, а виправити допущені помилки може коштувати дорожче, ніж спочатку провести повну підготовку та тестування.

Для тестування готової системи використаємо Unit-тестування та ручне тестування.

Unit-тестування — це вид тестування програмного забезпечення, ціль якого протестувати кожну одиницю програмного коду, щоб перевірити, чи виконується програма так, як очікується [39].

Процес Unit-тестування розподілено на дві частини, для кожної частини використано окремі фреймворки тестування:

— серверне тестування (NUnit);

— клієнтське тестування (Jasmine).

Для написання тестів використано модель тестів Arrange-Act-Assert, що представляє цілу парадигму тестування, яка використовується багатьма фреймворками для тестування:

— arrange, встановлює початкові умови для виконання тесту;

— act, виконує тест (зазвичай представляє один рядок коду);

— assert, перевіряє результат тесту.

Приклад написання Unit-тесту з використанням фреймворку NUnit зображено на рис. 4.1, із використанням фреймворку Jasmine зображено на рис. 4.2.

```
[Test]
0 references | 0 changes | 0 authors, 0 changes
public async Task GetPersonalPlan_ShouldCallServiceGivenParams()
{
    // Arrange
    var service = new Mock<IPersonalPlanService>();
    var controller = new PersonalPlanController(service.Object);
    var studentId = 12345;

    // Act
    await controller.GetPersonalPlan(studentId);

    // Assert
    service.Verify(x => x.GetPersonalPlan(It.Is<int>);
}
```

Рисунок 4.1 — Приклад Unit-тесту з використанням фреймворку NUnit

```
describe('getPersonalPlan', () => {
  it('should call httpClient GET with ./personal-plan/${studentId}', () => {
    // Arrange
    const getSpy = spyOn<any>(service, 'sendGet').and.callThrough();
    const studentId = 12345;

    // Act
    service.getPersonalPlan(studentId);

    // Assert
    expect(getSpy).toHaveBeenCalledTimes(1);
    expect(getSpy.calls.mostRecent().args[0]).toEqual(`/personal-plan/${studentId}`);
  });
});
```

Рисунок 4.2 — Приклад Unit-тесту з використанням фреймворку Jasmine

Запустимо на виконання Unit-тести на клієнтській та серверній частинах. Вікно з відображенням працездатності системи в результаті запуску тестів відображено на рис. 4.3 та рис. 4.4.

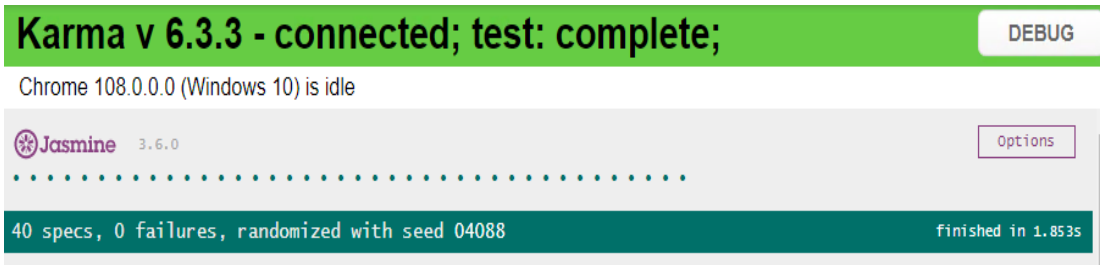


Рисунок 4.3 — Вікно відображення результату тестів фреймворку Jasmine

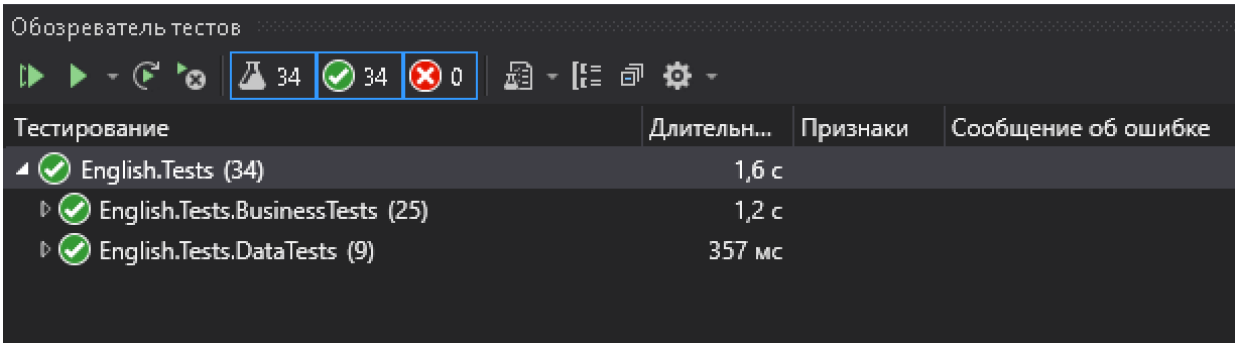


Рисунок 4.4 — Вікно відображення результату тестів фреймворку NUnit

4.2 Мануальне тестування програмного забезпечення

Мануальне тестування — це вид тестування програмного забезпечення, головною ознакою якого являється те, що тестування відбувається завдяки ручного тестування людиною без використання будь-якої автоматизації у програмному коді [39].

Для мануального тестування було залучено 3 користувачів, що під час тестування виявляли проблеми та надавали поради для поліпшення функціоналу програми.

На відміну від Unit-тестування, мануальне тестування, набагато довше, для виконання якого потрібно залучати додаткових осіб, щоб протестувати систему, але подібний вид тестування дозволяє знайти помилки в роботі системи, на які під час розробки не звернули увагу, шляхом виконання стратегій, що не передбачені, як звичайна поведінка респондента.

Для мануального тестування необхідно відтворити сценарій роботи підсистеми коригування процесу навчання крок за кроком, щоб переконатися, що підсистема працює відповідно до вимог.

Робота підсистеми починається з моменту реєстрації респондента в системі, вікно реєстрації користувача зображено на рис. 4.5.

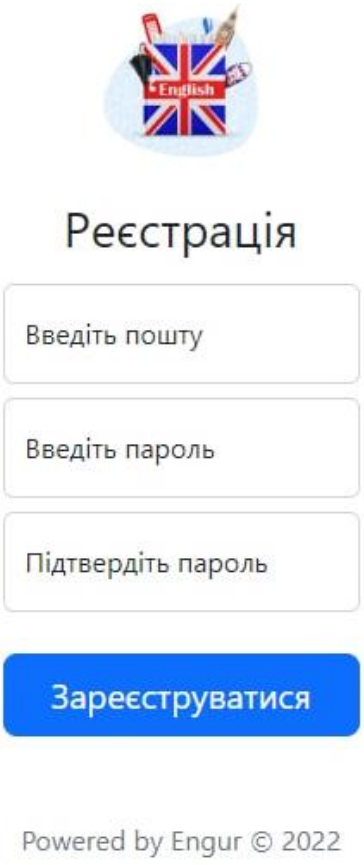


Рисунок 4.5 — Вікно реєстрації респондента

Після успішної реєстрації респонденту пропонується виконати швидке опитування, яке складається з трьох запитань, щоб підсистема зрозуміла рівень респондента, зображено на рис. 4.6, його цілі зображено на рис. 4.7, а кількість часу, що він готовий витрачати в день на навчання – на рис. 4.8.

Наступним кроком після проходження опитування респонденту пропонується створити свій перший модуль слів для подальшого навчання та заповнити його словами, які він планує навчати.

Модуль респондента, заповнений словами, зображено на рис. 4.9.

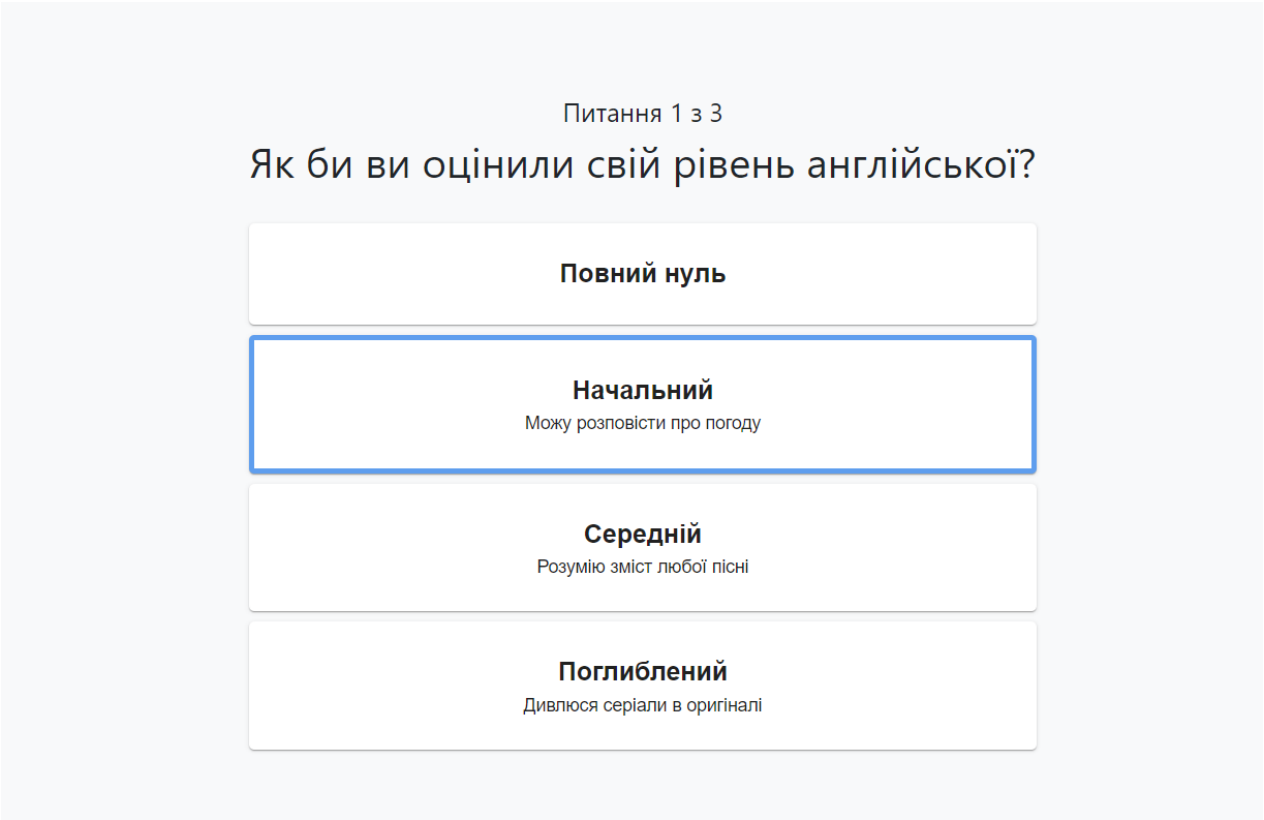


Рисунок 4.6 — Вікно опитування рівня респондента

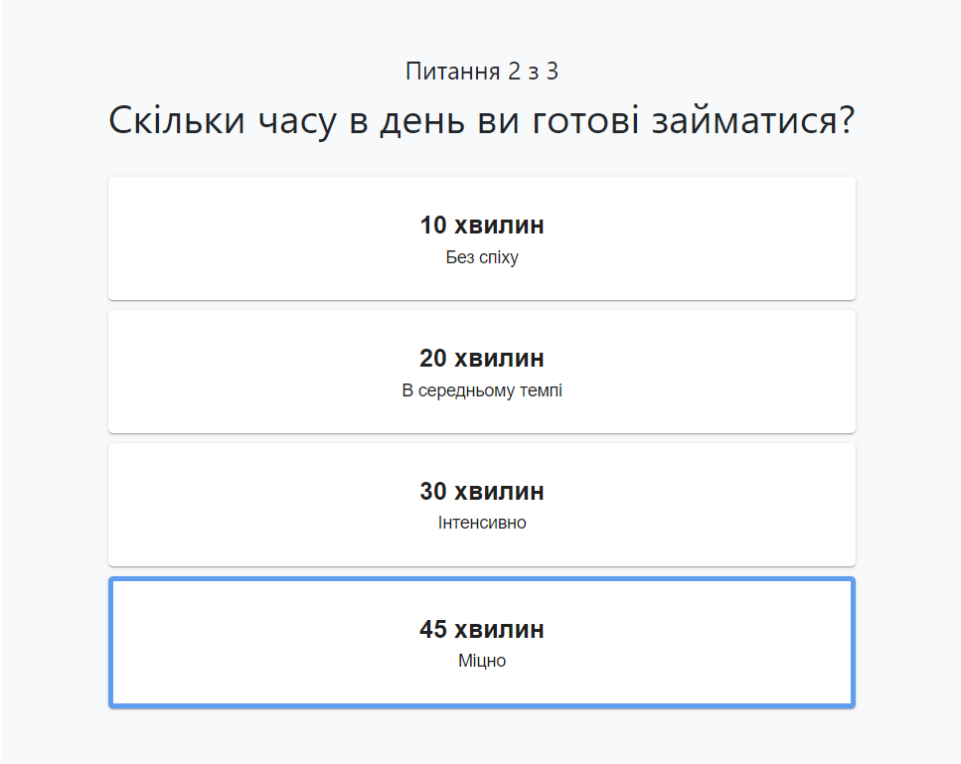


Рисунок 4.7 — Вікно опитування інтенсивності навчання респондента

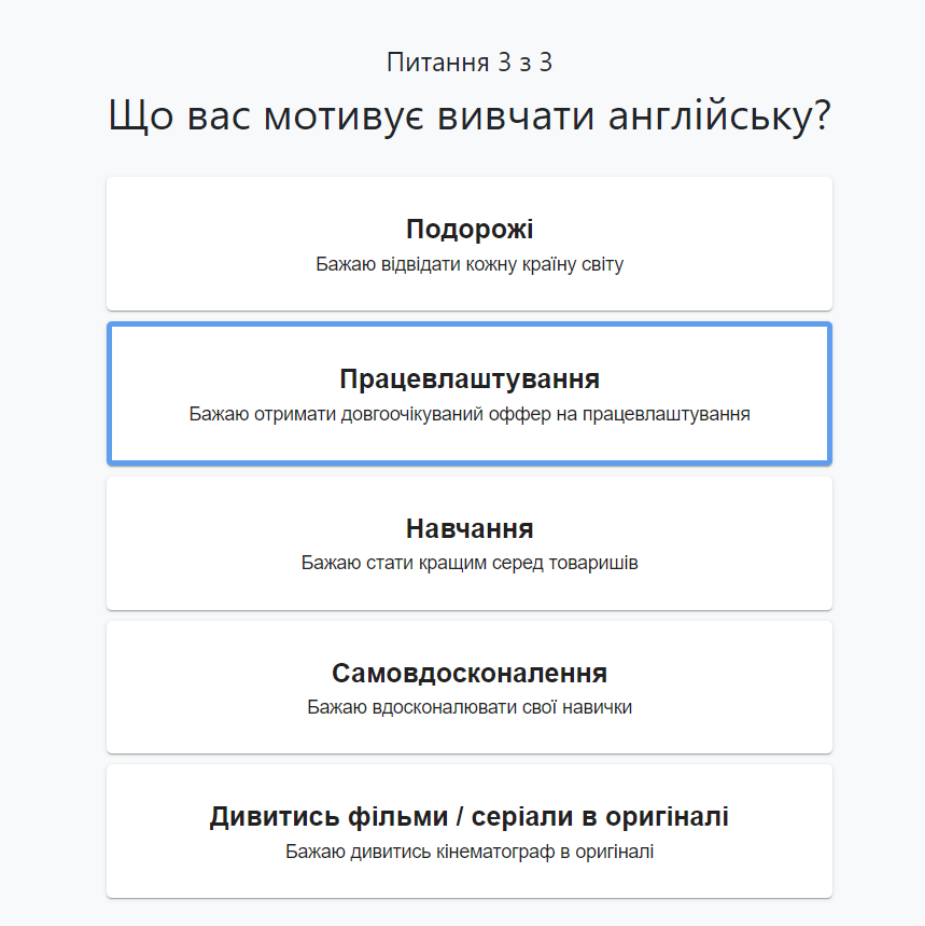


Рисунок 4.8 — Вікно опитування цілі респондента

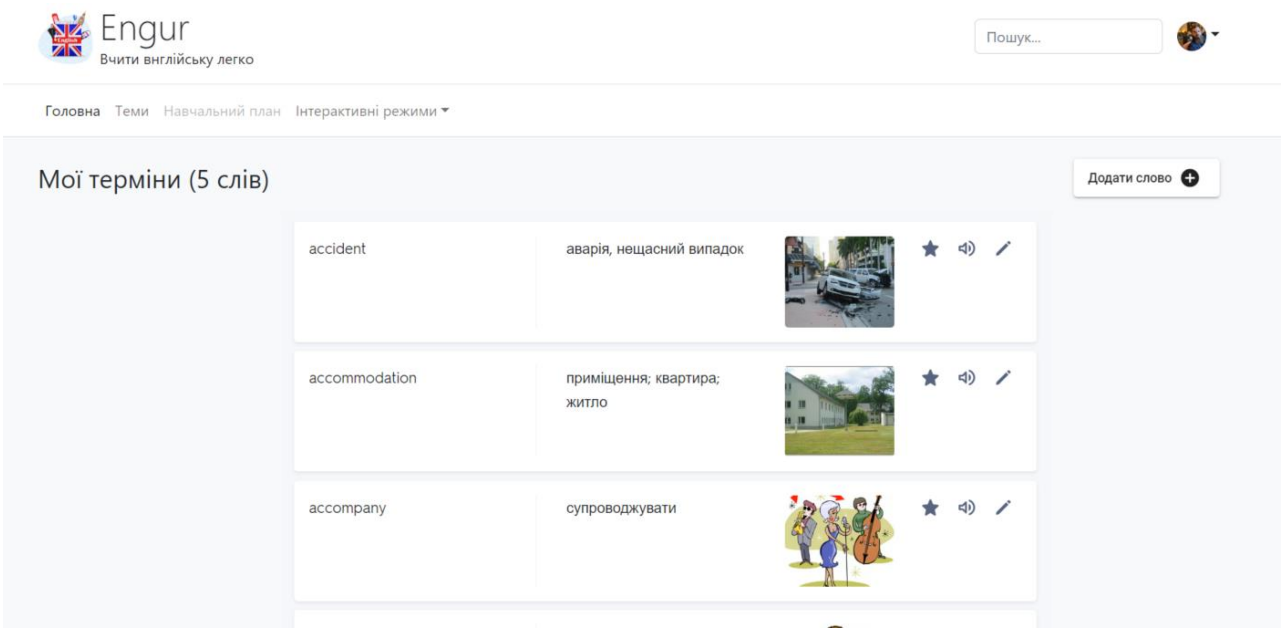


Рисунок 4.9 — Модуль респондента заповнений словами

Як тільки у респондента з’являється хоча б один модуль, механізм побудови персонального навчального плану стає йому доступний, і при його першому відкритті підсистема генерує навчальний план на поточний день, який складається з елементів навчального плану, кожен з яких дає інформацію респонденту щодо інтерактивного режиму навчання, рівень стадії повторення та нагороду, яку отримає респондент, у разі виконання елемента навчального плану.

Також користувач отримує інформацію щодо його прогресу на поточний день, загальну кількість отриманих нагород та можливість повернутися на один з попередніх днів, щоб виконати завдання навчального плану, яке він не встиг виконати раніше, проте, нагорода за виконання такого елемента є меншою, порівняно з нагородою, що він міг отримати в день, коли план було згенеровано вперше. Вікно з відображенням навчального плану зображено на рис. 4.10.

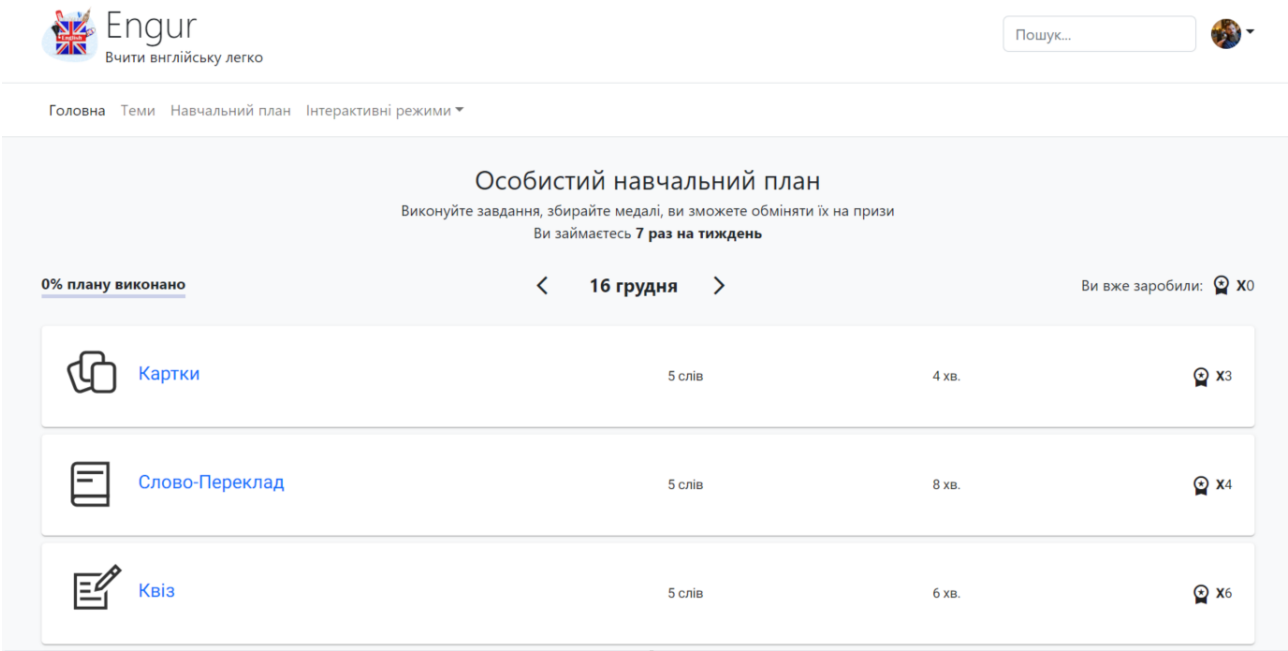


Рисунок 4.10 — Вікно відображення навчального плану користувача

При натисканні на елемент навчального плану респондента перемістить на сторінку з інтерактивним режимом навчання, що зображено на рис. 4.11, за проходження якого він отримає винагороду та індикацію виконаного завдання навчального плану, як зображено на рис. 4.12.

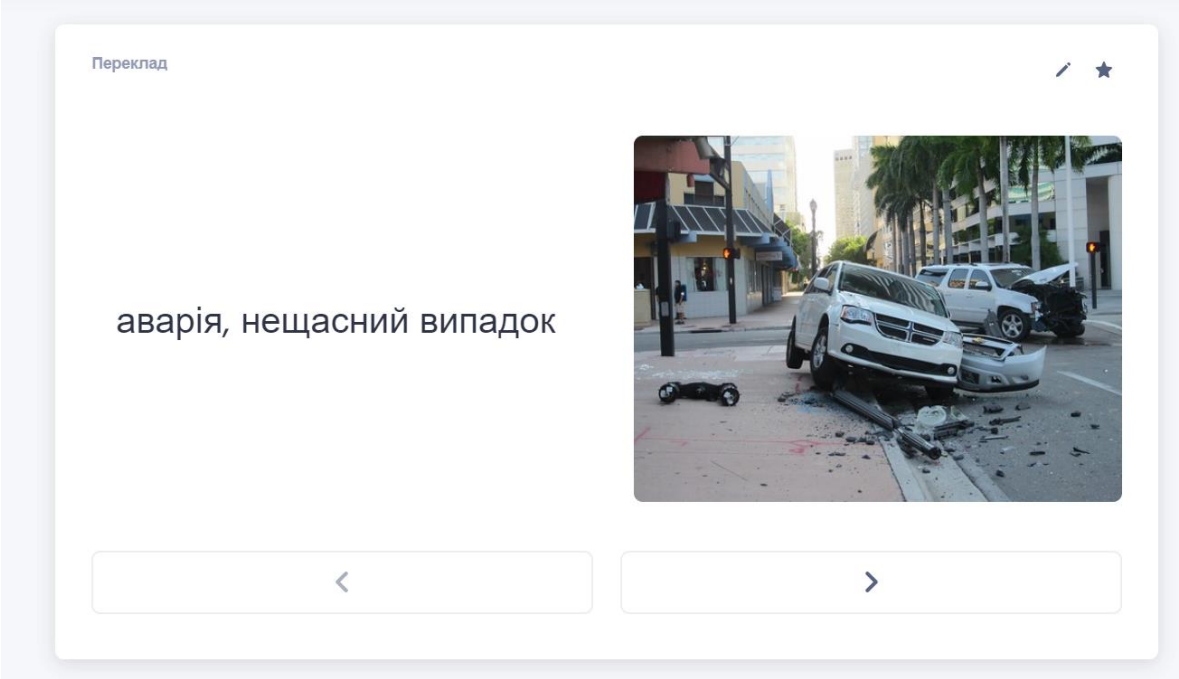


Рисунок 4.11 — Вікно відображення інтерактивного режиму навчання

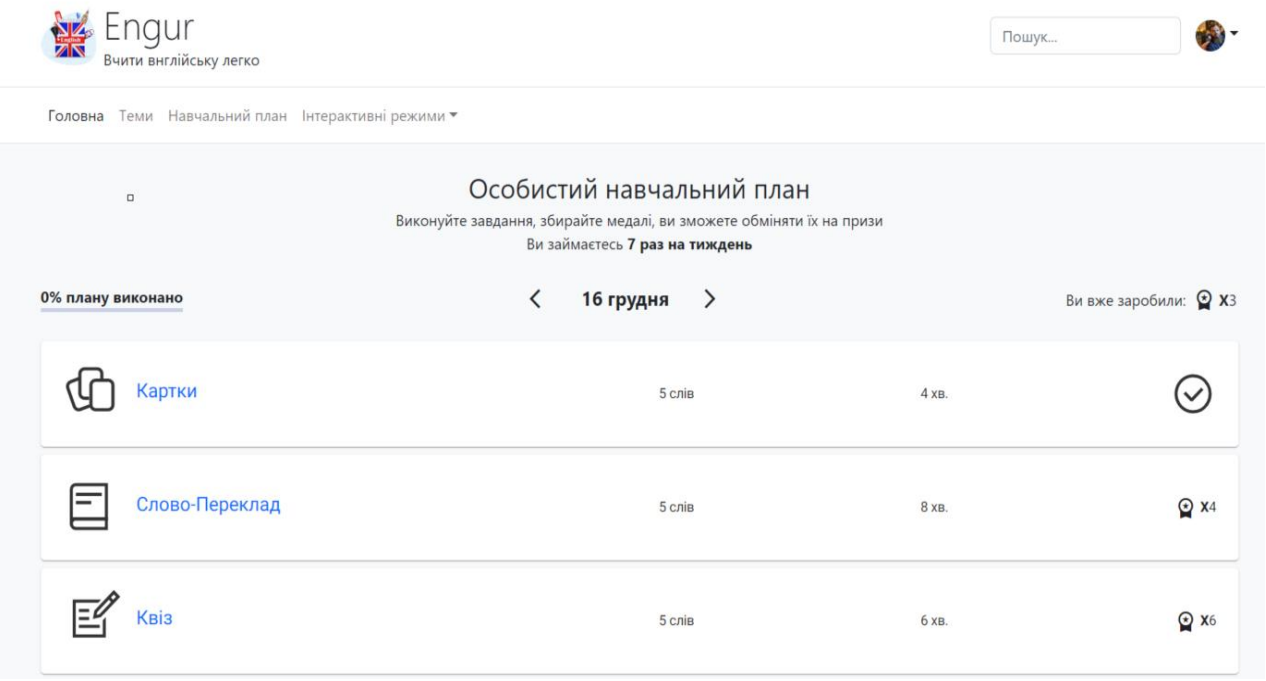


Рисунок 4.12 — Вікно індикації виконаного респондентом елемента плану

Висновки до розділу 4

У даному розділі магістерської роботи проведено тестування та аналіз роботи розробленого програмного забезпечення. Реалізовано автоматизоване тестування за допомогою Unit-тестів та виконано мануальне тестування підсистеми коригування процесу навчання.

Продемонстровано роботу ключових функцій та дано детальні пояснення дій користувача та програмних модулів, які за них відповідають.

Порівняння роботи запропонованої системи з аналогами дозволило виявити низку переваг першої, а саме: власна система є більш гнучкою, наявним є побудова адаптивного індивідуального плану навчання користувача, на основі його власних модулів слів, що дозволяє вивчати користувачеві тільки те, що він вважає за потрібне, у зручний для себе час та з контрольованим навантаженням.

Підсистема не має рекламних акцій та додаткової оплати.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Комерційний та технологічний аудит науково-технічної розробки

Метою даного розділу є проведення технологічного аудиту, в даному випадку нового програмного продукту Web-система для дистанційного вивчення слів англійської мови. Частина 2. «Підсистема коригування процесу навчання». Розробляється повноцінна система (бек + фронт) по типу Duolingo, Memrise, Lingualeo, яка дозволяє вивчати слова іноземної мови (в даному випадку англ.) з використанням методик навчання та тестування.

Робота комплексна, в даній частині акцент зроблений на розробці та програмній реалізації методики коригування процесу навчання завдяки формуванню індивідуального плану навчання для кожного користувача. Розраховуватися будуть витрати лише на дану частину роботи.

Особливістю програми є розширення можливостей системи дистанційного вивчення слів англійської мови, шляхом впровадження підсистеми коригування процесу навчання, що працює за власними математичними моделями, враховуючи індивідуальні фактори та прогрес респондента.

Ціни аналогів: Lingualeo — 3840 грн/рік (доступ до більшої кількості навчальних матеріалів, більш цікавих та ефективніших способів вивчення, повторення помилок, персональний навчальний план). Duolingo — 3840 грн/рік (немає реклами, необмежена кількість спроб, повторення помилок, навчальний план — узагальнений для всіх учнів, ніяк не коригуєтсья до індивідуальних показників конкретного респондента, на відміну від підсистеми, що розробляється)

uTalk — 14400 грн/рік (доступ до більшої кількості навчальних матеріалів, способів вивчення, але немає в наявності системи навчального плану).

Для проведення комерційного та технологічного аудиту залучають не менше 3-х незалежних експертів. Оцінювання науково-технічного рівня

розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням п’ятибальної системи оцінювання за 12-ма критеріями. Результати оцінювання комерційного потенціалу занесено в таблиці 5.1.

Таблиця 5.1 — Результати оцінювання комерційного потенціалу розробки

Критерії оцінювання	ПІБ експертів		
	Експерт 1	Експерт 2	Експерт 3
	Бали		
Технічна здійсненність концепції	4	4	4
Наявність аналогів на ринку	3	3	3
Цінова політика	3	4	4
Технічні та споживчі властивості виробу	4	3	4
Експлуатаційні витрати	3	4	3
Ринок збуту	4	3	4
Конкурентоспроможність	3	4	3
Фахівці з технічної і комерційної реалізації	4	3	4
Фінансування	3	4	4
Матеріально-технічна база	3	3	3
Термін реалізації ідеї	4	4	4
Супровідна документація	4	4	4
Сума	42	43	44
Середньоарифметична сума балів	(42+43+44) / 3 = 43		

За даними табл. 5.2 можна зробити висновок щодо рівня комерційного потенціалу даної розробки. Для цього доцільно скористатись рекомендаціями, наведеними в табл. 5.3.

Таблиця 5.3 — Рівні комерційного потенціалу розробки

Середньоарифметична сума балів СБ , розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0-10	Низький
11-20	Нижче середнього
21-30	Середній
31-40	Вище середнього
41-48	Високий

Як видно з табл. 5.3, рівень комерційного потенціалу розроблюваного нового програмного продукту є високим, що досягається за рахунок того, що

програмний продукт відрізняється від існуючих розширенням можливостей системи дистанційного вивчення слів англійської мови, шляхом впровадження підсистеми коригування процесу навчання, що працює за власними математичними моделями, враховуючи індивідуальні фактори та прогрес респондента.

5.2 Прогнозування витрат на виконання науково-дослідної роботи

Розрахунок основної заробітної плати розробників розраховується за формулою (5.1)

$$З_о = \frac{M}{T_р} \cdot t, \quad (5.1)$$

де M — місячний посадовий оклад конкретного розробника (дослідника), грн.;

$T_р$ — число робочих днів в місяці, 22 днів;

t — число днів роботи розробника (дослідника).

Результати розрахунків зведемо до табл. 5.4.

Таблиця 5.4 — Основна заробітна плата розробників

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник проєкту	25000	1136,36	30	34090,909
Програміст	20000	909,09	30	27272,727
Всього				61363,64

Так як розробляється програмний продукт, розробник виступає одночасно і програмістом, і тестувальником розроблюваного програмного продукту.

Додаткова заробітна плата дорівнює 12% від основної, формули (5.2) і (5.3)

$$З_д = З_о \cdot 12\ \% / 100\ \%\ , \tag{5.2}$$

$$З_д = (61363{,}64 \cdot 12\ \% / 100\ \%\ ) = 7363{,}64\ \ (грн.). \tag{5.3}$$

Згідно діючого законодавства нарахування на заробітну плату складають 22 % від суми основної та додаткової заробітної плати, формули (5.4) і (5.5)

$$Н_з = (З_о + З_д) \cdot 22\ \% / 100\%\ , \tag{5.4}$$

$$Н_з = (61363{,}64 + 7363{,}64) \cdot 22\ \% / 100\ \% = 15120{,}00\ (грн.). \tag{5.5}$$

Розрахуємо, амортизаційні витрати на комп'ютер балансова вартість якого становить 48655 грн., фактичного використання — 1,36 міс.

Амортизація обладнання, що використовувалось для розробки в спрощеному вигляді амортизація обладнання, що використовувалась для розробки розраховується за формулою (4.6) і (4.7)

$$А = \frac{Ц}{Т_в} \cdot \frac{t_{вик}}{12}\ [грн.], \tag{5.6}$$

де $Ц$ — балансова вартість обладнання, грн.;

$Т$ — термін корисного використання обладнання, років;

$t_{вик}$ — термін використання під час розробки, місяців.

$$А_{обл} = (48655 / 2) \cdot (1{,}36 / 12) = 2764{,}489\ (грн.). \tag{5.7}$$

Аналогічно визначаємо амортизаційні витрати на інше обладнання та приміщення. Розрахунки заносимо до табл. 5.2.

Таблиця 5.2 — Амортизаційні відрахування матеріальних і нематеріальних ресурсів для розробників

Найменування обладнання	Балансова вартість, грн.	Строк корисного використання, років	Термін використання обладнання, місяців	Амортиза-ційні відрахуван-ня, грн.
Комп'ютер та комп'ютерна периферія	48655	2	1,36	2764,489
Офісне обладнання (меблі)	20000	4	1,36	568,182
Приміщення	850000	20	1,36	4829,545
Всього				8162,22

Оскільки вартість ліцензійної ОС та спеціалізованих ліцензійних нематеріальних ресурсів менше 20000 грн (Microsoft Windows 10 Pro — 7 899,00 грн., Microsoft SQL Server — 7 686,00 грн), його вартість включається у вартість розробки повністю, $B_{нем.ак.}$ = 15585 грн.

Також потрібно включити у вартість нематеріальних ресурсів і вартість підписки на такий нематеріальний актив як Visual Studio Business IDE — 1 655,00 грн/міс. Актив використовувався 1,36 місяця, тому вартість склала 2250,8 грн.

Тарифи на електроенергію для непобутових споживачів відрізняються від тарифів електроенергії для населення. При цьому тарифи на розподіл електроенергії у різних постачальників, будуть різними.

Тарифи на розподіл електроенергії для всіх компаній встановлює Національна комісія з регулювання енергетики і комунальних послуг. Витрати на силову електроенергію розраховуються за формулами (5.8) і (5.9):

$$Ве = В \cdot П \cdot Ф \cdot К, \tag{5.8}$$

де $В$ — вартість 1 кВт-години електроенергії для 1 класу, $В$ = 6,2 грн./кВт;

$П$ — встановлена потужність обладнання, кВт. $П$ = 0,45 кВт;

$Ф$ — фактична кількість годин роботи обладнання, годин;

$К_п$ — коефіцієнт використання потужності, $К_п$ = 0,9.

$$В_е = 0,9 \cdot 0,45 \cdot 8 \cdot 30 \cdot 6,2 = 602,64 \text{ (грн.)}. \quad (5.9)$$

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені на собівартість досліджень. Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників, формули (5.10) і (5.11):

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (5.10)$$

де $Н_{ів}$ — норма нарахування за статтею «Інші витрати».

$$I_в = 61363,64 \cdot 70\% / 100\% = 42954,55 \text{ (грн.)}. \quad (5.11)$$

До статті «Накладні витрати» належать: витрати, пов’язані з управлінням організацією; «Накладні витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників, формули (5.12) і (5.13):

$$Н_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (5.12)$$

де $Н_{нзв}$ — норма нарахування за статтею «Накладні витрати».

$$Н_{нзв} = 61363,64 \cdot 100\ \% / 100\ \% = 61364 \text{ (грн.)}. \quad (5.13)$$

Сума всіх попередніх статей витрат дає загальні витрати на проведення науково-дослідної роботи:

$$В_{заг} = 61363,64+7363,64+15120,00+195+15585+2250,8+8162,22+$$
$$+602,64+ 42954,55+61364 = 214961,11 \text{ (грн)}. \quad (5.14)$$

Загальні витрати на завершення науково-дослідної роботи та оформлення її результатів розраховуються *ЗВ*, визначається за формулою:

$$ЗВ = \frac{В_{заг}}{\eta} \ (грн), \quad (5.15)$$

де η — коефіцієнт, який характеризує етап виконання науково-дослідної роботи.

Так, якщо науково-технічна розробка знаходиться на стадії: науково-дослідних робіт, то η=0,1; технічного проєктування, то η=0,2; розробки конструкторської документації, то η=0,3; розробки технологій, то η=0,4; розробки дослідного зразка, то η=0,5; розробки промислового зразка, то η=0,7; впровадження, то η=0,9. Оберемо η = 0,5, так як розробка, на даний момент, знаходиться на стадії дослідного зразка, і розрахунок за формулою:

$$ЗВ = 214961,11 / 0,5 = 429922 \text{ (грн)}. \quad (5.16)$$

5.3 Розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором

Щоб розрахувати можливе зростання чистого прибутку у потенційного інвестора від можливого впровадження науково-технічної розробки необхідно:

— вказати, з якого часу можуть бути впроваджені результати розробки;

— зазначити, протягом скількох років після впровадження цієї науково-технічної розробки очікуються основні позитивні результати для інвестора;

— кількісно оцінити величину існуючого та майбутнього попиту на цю науково-технічну розробку та назвати основних суб’єктів цього попиту;

— визначити ціну реалізації на ринку науково-технічних розробок з аналогічними чи подібними функціями.

Майбутній економічний ефект від модернізації буде формуватися на основі таких даних:

$$\Delta П_i = (\pm \Delta Ц_0 \cdot N + Ц_0 \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (5.17)$$

де $\pm\Delta Ц_o$ — зміна вартості програмного продукту від впровадження результатів науково-технічної розробки в аналізовані періоди часу;

N — кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки;

$Ц_o$ — основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

$Цб$ — вартість програмного продукту у році до результатів розробки;

ΔN — збільшення кількості споживачів продукту, в аналізовані періоди часу, від покращення його певних характеристик;

λ — коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість дорівнює 20%, а коефіцієнт $\lambda = 0{,}8333$;

p — коефіцієнт, який враховує рентабельність продукту;

ϑ — ставка податку на прибуток, у 2022 році ϑ =18%.

Припустимо, що при прогнозованій ціні 2000 грн. за одиницю виробу, термін збільшення прибутку складе 3 роки. Після завершення розробки і її вдосконалення, можна буде підняти її ціну на 300 грн. Кількість одиниць реалізованої продукції також збільшиться: протягом першого року — на 3000 шт., протягом другого року — на 2000 шт., протягом третього року на 1000 шт. В Україні на 25.08.2022 р. є понад 23,5 тис. закладів освіти [40]. Розрахуємо майбутній економічний ефект від модернізації за 3 роки:

$$\Delta П_1 = (0 \cdot 300 + (2000 + 300) \cdot 3000) \cdot 0{,}8333 \cdot 0{,}45) \cdot \quad (5.18)$$

$$\cdot (1 - 0{,}18) = 1844999{,}926 \text{ (грн)},$$

$$\Delta П_2 = (0 \cdot 300 + (2000 + 300) \cdot (3000+2000) \cdot 0{,}8333 * 0{,}45) \cdot \cdot (1 - 0{,}18) = 3536249{,}859 \text{ (грн)}, \quad (5.19)$$

$$\Delta П_3 = (0 \cdot 300 + (2000 + 300) \cdot (3000+2000+1000) \cdot \cdot 0{,}8333 \cdot 0{,}45) \cdot (1 - 0{,}18) = 4243499{,}83 \text{(грн)}. \quad (5.20)$$

Отже, комерційний ефект від реалізації результатів розробки за три роки складе 9624749,62 грн.

Розраховуємо приведену вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_1^T \frac{\Delta П_i}{(1+\tau)^t}, \quad (5.21)$$

де $\Delta П_i$ — збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої науково-дослідної (науково-технічної) роботи, грн;

T — період часу, протягом якою виявляються результати впровадженої науково-дослідної (науково-технічної) роботи, роки;

τ — ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, $\tau = 0{,}05 \ldots 0{,}15$;

t — період часу (в роках).

Збільшення прибутку отримаємо з першого року:

$$ПП = (1844999{,}926/(1+0{,}1)^1) + (3536249{,}859/(1+0{,}1)^2) + \quad (5.22)$$
$$+ (4243499{,}830/ /(1+0{,}1)^3) = 1677272{,}66 + 2922520{,}544 +$$
$$3188204{,}23 = 7787997{,}435 \text{ (грн.)}.$$

Далі розраховують величину початкових інвестицій *PV*, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ, \tag{5.23}$$

де $k_{інв}$ — коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай $k_{інв}$=2...5, але може бути і більшим;

ЗВ — загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.:

$$PV = 2 \cdot 429922 = 859844{,}44 \text{ (грн.)}. \tag{5.24}$$

Тоді абсолютний економічний ефект $E_{абс}$ або чистий приведений дохід *(NPV, Net Present Value)* для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV, \tag{5.25}$$

$$E_{абс} = 7787997{,}435 - 859844{,}44 = 6928152{,}99 \text{ (грн.)}.$$

Оскільки $E_{абс} > 0$, то вкладання коштів на виконання та впровадження результатів даної науково-дослідної роботи може бути доцільним.

Для остаточного прийняття рішення з цього питання необхідно розрахувати внутрішню економічну дохідність або показник внутрішньої норми дохідності *(IRR, Internal Rate of Return)* вкладених інвестицій та порівняти її з так званою бар’єрною ставкою дисконтування, яка визначає ту мінімальну

внутрішню економічну дохідність, нижче якої інвестиції в будь-яку науково-технічну розробку вкладати буде економічно недоцільно.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \quad (5.26)$$

де $T_ж$ — життєвий цикл наукової розробки, роки.

$$\sqrt{E_в = 3\ (1 + 6928152{,}99/859844{,}44} - 1 = 1{,}085. \quad (5.27)$$

Визначимо мінімальну ставку дисконтування, яка у загальному вигляді визначається так:

$$\tau = d + f, \quad (5.28)$$

$$T_{ок} = 1 / 1{,}061 = 0{,}94\ (р), \quad (5.29)$$

де d — середньозважена ставка за депозитними операціями в комерційних банках; в 2022 році в Україні d = (0,09…0,14);

f — показник, що характеризує ризикованість вкладень; зазвичай.

Так як $E_в > \tau_{min}$, то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Розрахуємо термін окупності вкладених у реалізацію наукового проєкту інвестицій:

$$T_{ок} = \frac{1}{E_в}, \quad (5.30)$$

$$T_{ок} = 1 / 1{,}085 = 0{,}92\ р.$$

Оскільки $T_{ок}$ < 3-х років, а саме термін окупності рівний 0,92 роки, то фінансування даної наукової розробки є економічно доцільним.

Висновки до розділу 5

Економічна частина даної роботи містить розрахунок витрат на розробку нового програмного продукту, сума яких складає 429922 гривень. Було спрогнозовано орієнтовану величину витрат по кожній з статей витрат. Також розраховано чистий прибуток, який може отримати виробник від реалізації нового технічного рішення, розраховано період окупності витрат для інвестора та економічний ефект при використанні даної розробки.

У результаті аналізу розрахунків можна зробити висновок, що розроблений програмний продукт за ціною дешевший за аналог і є конкурентоспроможним, а також економічно доцільним, оскільки період окупності складе близько 0,92 роки.

ВИСНОВКИ

У магістерській кваліфікаційній роботі було спроєктовано та розроблено веб-систему для дистанційного вивчення слів англійської мови в межах підсистеми коригування процесу навчання.

Проведено аналітичний огляд комп’ютерних систем та існуючих підходів до контролю та коригування навчального процесу. Розглянуто особливості автоматизованого адаптивного навчального плану.

Вивчено недоліки та переваги наявних веб-систем коригування процесу навчання, що дозволило обґрунтувати вимоги до розроблення власної підсистеми.

Розроблено структуру підсистеми коригування процесу навчання та наведено інтерфейс для її інтеграції до загальної веб-системи вивчення слів англійської мови. Запропонована підсистема складається з модулів: побудови навчального плану та коригування процесу навчання.

На базі аналізу існуючих математичних моделей було запропоновано підхід до реалізації побудови навчального плану, досліджено механізм роботи пам’яті та на його основі розроблено математичну модель процесу модель засвоєння матеріалу (слів англійської мови) респондентом в підсистемі коригування процесу навчання.

Обґрунтовано вибір інструментів для програмної реалізації розробки для клієнтської та серверної частин окремо, а також для бази даних. Для клієнтської частини та відображення інтерфейсу користувача було вирішено розробляти Angular-застосунок. Для серверної частини ключовою технологією обрано .NET та стек його суміжних фреймворків. Сервер бази даних обрано Microsoft SQL Server.

У результаті роботи було розроблено веб-застосунок та базу даних.

Клієнтська частина використовує підходи розбиття структури на модулі компонентів та їх маршрутизації. Серверна частина базується на трирівневій

архітектурі та виконує основну логіку та обчислення над даними, а також надає інтерфейс керування програмою клієнтській частині.

Розроблено нормалізовану базу даних, що керується беспосередньо кодом додатку, але також має можливість ручного керування за допомогою Microsoft SQL Management Studio. Під час розроблення програмного комплексу було враховано можливість подальшого горизонтального масштабування системи, шляхом додавання до вже існуючої підсистеми оцінювання якості навчання інших підсистем, наприклад, підсистеми оцінки якості навчання.

Здійснено тестування та аналіз роботи розробленї системи. Спочатку було проведено автоматизоване тестування за допомогою Unit-тестів, де детальну увагу приділено послідовності та коректності виконання модулів системи. Наступним етапом було проведено мануальне тестування, що продемонструвало роботу ключових функцій та надано детальні пояснення дій програмних модулів, які за них відповідають.

Ключовою перевагою розроблюваної системи являється механізм побудови адаптивного індивідуального плану навчання, який підлаштовується під потреби та можливості користувача, на основі його власних модулів та статистичних даних.

Крім того, до інших переваг підсистеми, що розробляється належать:

— відсутність обмеження преміум режимом та реклами;

— повний доступ до модифікування власних матеріалів навчання;

— можливість впливати на генерування особистого навчального плану;

— взаємодія з іншими підсистемам, наприклад підсистемою оцінювання якості навчання;

— можливість створювати власні набори слів;

— наявність режимів інтерактивного вивчення слів.

Наведено дослідження рівня комерційного потенціалу розробки, який становить 43 бали, що є високим показником. Подібний рівень досягається за рахунок того, що розроблений програмний продукт відрізняється від вже

існуючих аналогів розширенням можливостей системи дистанційного вивчення слів англійської мови шляхом впровадження підсистеми коригування процесу навчання.

У результаті аналізу розрахунків можна зробити висновок, що розроблений програмний продукт дешевший за системи-аналоги і є конкурентоспроможним. Термін окупності інвестиційних вкладень в нього складає 0,92 роки, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційних інвесторів профінансувати її впровадження та вивести на ринок.

Отже, задачі, поставлені у магістерській кваліфікаційній роботі, було виконано в повному обсязі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

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. 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.

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

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

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

10. George A. Miller. The Magical Number Seven, Plus or Minus Two. *The Psychological Review*. 1956. Vol. 63. pp. 81—97.

11. Бугай О. С. Підсистема забезпечення навчання web – системи для дистанційного вивчення слів англійської мови *L науково-технічна конференція ВНТУ* : Електронне наукове видання матеріалів конференції, м. Вінниця, 2022. URL: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2021/paper/view/12004 (дата звернення 19.09.2022).

12. Винарчук Т. М. Веб-статистика як інструмент аналізу сайту. *Народна освіта*. 2014. № 23. Т. 6. URL: https://www.narodnaosvita.kiev.ua/?page_id=2424 (дата звернення 12.10.2022).

13. Gerhard W., Marcus S. User Modeling and Adaptive Navigation Support Tutoring Systems. Vienna : 2016. 210 c.

14. 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 (accessed: 12.10.2022).

15. Брусиловський П. Л. Адаптивні та інтелектуальні технології в мережевому навчанні. *Новини штучного інтелекту*. – 2002. – № 5. – С. 25–31

16. Зайцева Л. В. Моделі та методи адаптації до учнів у системах комп'ютерного навчання. *Edication Technology & Society*. - 2013. - №6. – С. 204–212.

17. 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.

18. An introduction to web development technologies. *Tiller* : website. URL: https://tillerdigital.com/blog/an-introduction-to-web-development-technologies/ (accessed: 12.10.2022)

19. uTalk – онлайн сервіс для вивчення слів іноземних мов. *uTalk* : веб-сайт. URL: https://www.utalk.com/ (дата звернення 24.09.2022).

20. Lingualeo – онлайн сервіс для вивчення англійської та інших іноземних мов. *Lingualeo* : веб-сайт. URL : https://lingualeo.com/ (дата звернення 24.09.2022).

21. Duolingo – самий швидкий шлях вивчити іноземну мову. *Duolingo* : веб-сайт URL : https://www.duolingo.com/ (дата звернення 24.09.2022).

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

23. Пам’ять. *Wikipedia* : website. URL: https://uk.wikipedia.org/wiki/%D0%9F%D0%B0%D0%BC%27%D1%8F%D1%82%D1%8C/ (дата звернення: 15.10.2022).

24. П. А. М’ясоїд. Загальна психологія. Навч. посіб. Київ: Вища школа, 1998, 479 с.

25. Buzan T. Use Both Sides of Your Brain. *Plume; Revised edition*. 1983. Vol. 63. pp. 51—62.

26. Палій А. А. Методи діагностики психічного розвитку. *Прикарпатський національний університет імені Василя Стефаника*., 2013. 430 с.

27. Buzan T., Buzan B. The Mind Map Book. *Plume; Reprint edition*. 1996. Vol. 96. pp. 141—155.

28. 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 (accessed: 12.10.2022).

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

30. Лок Е. ASP.NET Core in Action : 2018. 84с.

31. Wilken J. Angular in Action : Shelter Island. New York : 2018. 320 с.

32. Clow M. Angular 5 Projects: Learn to Build Single Page Web Applications Using 70+ Projects : Apress : 2018. 465 с.

33. Hughes A. Microsoft SQL Server. TeachTarget : website. URL: https://www.techtarget.com/searchdatamanagement/definition/SQL-Server/ (accessed: 12.10.2022).

34. Clow M. Angular 5 Projects: Learn to Build Single Page Web Applications Using 70+ Projects : Apress : 2018. 465 с.

35. Маршрутизація. *Metanit* : веб-сайт. URL: https://metanit.com/web/angular2/7.1.php/ (accessed: 16.10.2022).

36. Фрімен А. Pro ASP.NET Core MVC 2 : 2017. 41с.

37. Фагерберг Д. ASP.NET Core 2.0 MVC & Razor Pages for Beginners: How to Build a Website : 2017. 21с.

38. Entity Framework Core. *Microsoft* : website. URL: https://learn.microsoft.com/ru-ru/ef/core/ (accessed: 16.10.2022).

39. Тестування веб-проєктів: основні етапи та поради. *QaLight*: website. URL: https://qalight.ua/baza-znaniy/testuvannya-veb-pro%D1%94ktiv-osnovni-etapi-ta-poradi/ (дата звернення: 16.10.2022).

40. 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/ (дата звернення 20.11.2022).

ДОДАТОК А

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ

Завідувач кафедри

обчислювальної техніки

_____проф., д.т.н. О. Д. Азаров

«___» _____ 2022 року

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання комплексної магістерської кваліфікаційної роботи

Web-система для дистанційного вивчення слів анлійської мови.

Частина 1 «Підсистема оцінювання якості навчання»

08-23.КМКР.033.00.000 ТЗ

Науковий керівник к.т.н., проф. каф. ОТ

_____ Азарова А. О.

Студента групи 1КІ-21м

_____ Бугая О. С.

ВНТУ 2022

1 Підставою для виконання комплексної магістерської кваліфікаційної роботи є наказ про затвердження теми дипломної роботи, а також актуальність розробки системи для вивчення слів іноземної, в даному випадку англійської мови, яка би забезпечувала ефективний процес навчання іноземної мови, коригуючи процес навчання.

2 Мета і призначення КМКР:

— метою є розробка веб-системи для вивчення слів іноземної мови та підсистеми коригування процесу навчання;

— призначенням розробки є виконання комплексної магістерської кваліфікаційної роботи з можливістю подальшого впровадження та масштабування.

3 Вихідні дані для виконання КМКР:

— розробити веб-систему для дистанційного вивчення слів англійської мови, а також підсистему для коригування процесу навчання;

— основна технологія Angular;

— середовище програмування Microsoft Visual Studio 2022;

— мова програмування C#, SQL, TypeScript.

4 Технічні вимоги до виконання КМКР:

— виведення математичної моделі коригування процесу навчання.

— впровадження підсистеми для коригування процесу навчання.

5 Етапи КМКР та очікувані результати (див. табл. А.1).

Таблиця А.1 — Етапи роботи

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Дослідження методів дистанційного вивчення іноземної мови.	01.09.2022	09.09.2022	Розділ 1
2	Математичне моделювання процесу оцінювання якості навчання	12.09.2022	26.09.2022	Розділ 2

Продовження таблиці А.1

3	Аналіз та вибір технологій проектування та розробки веб-додатків	3.10.2022	10.10.2022	Розділ 1, 3 частково
4	Програмна реалізація системи	17.10.2022	31.10.2022	Розділ 3 повністю
5	Підготовка економічної частини	7.11.2022	14.11.2022	Розділ 4
6	Оформлення матеріалів до захисту КМКР	21.11.2022	1.12.2022	Пояснювальна записка, графічний матеріал, лістинг, презентація

6 Матеріали, що подаються до захисту КМКР:

— пояснювальна записка КМКР;

— графічні і ілюстративні матеріали;

— протокол попереднього захисту КМКР на кафедрі;

— відгук наукового керівника;

— рецензія на виконану роботу;

— анотації до КМКР українською та іноземною мовами;

— нормоконтроль про відповідність оформлення КМКР діючим вимогам.

7 Порядок контролю виконання та захисту КМКР:

— виконання етапів графічної та розрахункової документації КМКР контролюється науковим керівником згідно зі встановленими термінами;

— захист КМКР відбувається на засіданні Державної екзаменаційної комісії, затвердженою наказом ректора.

8 Вимоги до оформлення КМКР викладені в методичних вказівках до виконання магістерських кваліфікаційних робіт зі спеціальності 123 — «Комп’ютерна інженерія, ДСТУ 3008 : 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання», ДСТУ 8302 : 2015 «Бібліографічні посилання. Загальні положення та правила складання».

ДОДАТОК Б

Структура комп’ютерної системи дистанційного навчання

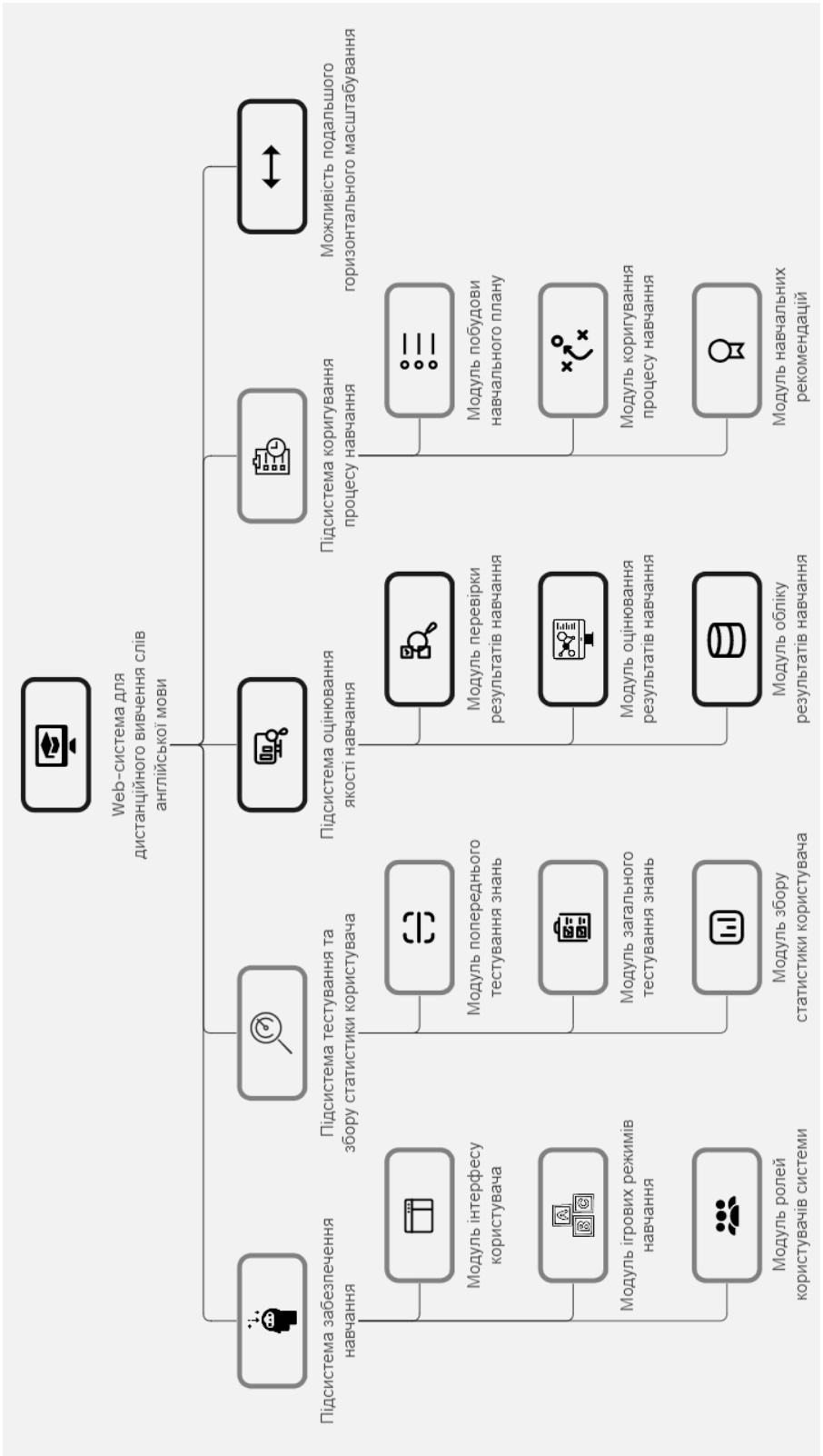


Рисунок Б.1 — Структурна схема комп’ютерної системи дистанційнго навчання

ДОДАТОК В

Лістинг представлення персонального плану

```
<div class="container">
  <div class="text-center mb-4">
    <div class="fs-3">Особистий план</div>
    <div class="fs-6">Виконуйте завдання, збирайте медалі, ви зможете обміняти їх на призи</div>
    <div class="fs-6">Ви займаєтесь <strong> </strong></div>
  </div>
  <div class="d-flex justify-content-between align-items-center">
    <div>
      <div><strong>40% плану виконано</strong></div>
      <mat-progress-bar mode="determinate" value="40"></mat-progress-bar>
    </div>
    <div class="d-flex align-items-center">
      <div>
        <button mat-icon-button aria-label="<">
          <mat-icon>arrow_back_ios</mat-icon>
        </button>
      </div>
      <div class="ms-4 me-4 fs-5">
        <strong>16 грудня</strong>
      </div>
      <div>
        <button mat-icon-button aria-label=">">
          <mat-icon>arrow_forward_ios</mat-icon>
        </button>
```

```
            </div>
        </div>
        <div class="d-flex">
            <div>Ви вже заробили:</div>
            <mat-icon class="ms-2 me-1">workspace_premium</mat-icon>
            <div><strong>X</strong>64</div>
        </div>
    </div>

    <div class="mt-4">
        <mat-card class="mt-2" *ngFor="let item of items">
            <mat-card-content class="p-2 d-flex align-items-center justify-content-
between">
                <div>
                    <img height="60" src="../../../assets/img/logo.png" alt=""
class="me-4">
                    <a href="#" class="text-decoration-none fs-5"> </a>
                </div>
                <div>
                    battary-place
                </div>
                <div>25 хвилин</div>
                <div class="d-flex align-items-center">
                    <mat-icon class="me-1">workspace_premium</mat-icon>
                    <div><strong>X</strong>5</div>
                </div>
            </mat-card-content>
        </mat-card>
    </div></div>
```

ДОДАТОК Г

Лістинг репозиторіїв

```
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();
    }
}
```

ДОДАТОК Д

Лістинг сервісів

```
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;
}

public static Topic MapFromDto(TopicDto topic)
{
    return new Topic()
    {
        Id = topic.Id,
        Name = topic.Name           };
```

ДОДАТОК Е

Лістинг скрипту інтерактивних режимів

```
function buildCrosswordBlocks(crosswordblocks) {
	var graphs = [];
	var lastacross = false;

	for (var word in crosswordblocks) {
		if (!crosswordblocks.hasOwnProperty(word) || word == '(unmatched)')
continue;

		var subwords = crosswordblocks[word];
		var longestwordlength = getLongestWordLength(subwords);

		var across = true;

		if(areWeRandomizingAcrossDownChoices()) {
			across = randomTrueFalse();
		}

		var matrix = [];
		var matrixpositions = [];

		if(across) {
			matrix[longestwordlength - 1] = word;
			matrixpositions[word] = [longestwordlength - 1, 0];

			for(var i = 0; i < subwords.length; i++) {
				var subwordentry = subwords[i];
```

```
                    var subword = subwordentry[0];
                    var subletter = subwordentry[1];

                    var matchingposition =
findMatchingLetterMatrixPosition(matrix, word, subletter, longestwordlength - 2);
                    var matchingoffset = findMatchingOffset(subword,
subletter);
                    matrixpositions[subword] = [longestwordlength -
matchingoffset - 1, matchingposition];
                    matrix = setLetterMatrixVertically(matrix, subword,
longestwordlength - matchingoffset - 1, matchingposition);
                }
            } else {
                matrix = fillLetterMatrixVertically(matrix, word,
longestwordlength + 1, 0);
                matrixpositions[word] = [0, longestwordlength];

                for(var i = 0; i < subwords.length; i++) {
                    var subwordentry = subwords[i];

                    var subword = subwordentry[0];
                    var subletter = subwordentry[1];
                    var matchingposition =
findMatchingLetterMatrixPositionVertical(matrix, word, subletter,
longestwordlength - 1);
                    var matchingoffset = findMatchingOffset(subword,
subletter);
```

```
                        matrixpositions[subword] = [matchingposition,
longestwordlength - matchingoffset];
                        matrix = setLetterMatrixHorizontally(matrix, subword,
matchingposition, longestwordlength - matchingoffset);
                }
        }
        var graph = {
                'matrix':matrix,
                'matrixpositions':matrixpositions,
                'across':across,
                'word':word,
        };

        graphs.push(graph);
}

if(crosswordblocks['(unmatched)']) {
        var graph =
buildUnassignedCrosswordBlock(crosswordblocks['(unmatched)']);
        graphs.push(graph);
}

return graphs;
}

/* buildUnassignedCrosswordBlock(unmatchedcrosswords)
```

At the end of making our groups of words with spine words, we have smoe that could not be matched at all. Group these together as a block.

```
	*/

function buildUnassignedCrosswordBlock(unmatchedcrosswords) {
	var across = true;

	if(areWeRandomizingAcrossDownChoices()) {
		across = randomTrueFalse();
	}

	var longestwordlength = getLongestWordLength(unmatchedcrosswords);

	var matrix = [];
	var matrixpositions = [];

	if(across) {
		for(var i = 0; i < unmatchedcrosswords.length; i++) {
			var unmatchedcrossword = unmatchedcrosswords[i];
			matrix[i] = unmatchedcrossword;
			matrixpositions[unmatchedcrossword] = [0,i];
		}
	} else {
		for(var i = 0; i < unmatchedcrosswords.length; i++) {
			var unmatchedcrossword = unmatchedcrosswords[i];
			matrix = setLetterMatrixVertically(matrix, unmatchedcrossword,
0, i);
			matrixpositions[unmatchedcrossword] = [i,0];
		}
	}
```

```
	var graph = {
		'matrix':matrix,
		'matrixpositions':matrixpositions,
		'across':!across,
		'word':'(unmatched)',
	};

	return graph;
}

	/* insertLetterAtStringPosition(letter, string, position)

		Arrays are immutable within JavaScript. So, this method allows us to edit strings by inserting letters at positions.

	*/

function insertLetterAtStringPosition(letter, string, position) {
	if(!letter) {
		letter = ' ';
	}
	return string.substr(0, position) + letter + string.substr(position + 1);
}

	/* setLetterMatrixHorizontally(matrix, word, y, x)

		Build a graph from a group of words horizontally.
```

```
        */

function setLetterMatrixHorizontally(matrix, word, y, x) {
        for(var i = 0; i < word.length; i++) {
                var position = i + x;
                if(!matrix[y]) {
                        matrix[y] = '';
                }
                letters = matrix[y];

                if(letters.length < position) {
                        while(letters.length < position) {
                                letters += ' ';
                        }
                        letters += word[i];
                } else {
                        letters = insertLetterAtStringPosition(word[i], letters, position);
                }

                matrix[y] = letters;
        }
        return matrix;
}
function setLetterMatrixVertically(matrix, word, y, x) {
        for(var i = 0; i < word.length; i++) {
                var position = i + y;
                if(!matrix[position]) {
                        matrix[position] = '';
                }
```

```
            letters = matrix[position];

            if(letters.length < x) {
                while(letters.length < x) {
                    letters += ' ';
                }

                letters += word[i];
            } else {
                letters = insertLetterAtStringPosition(word[i], letters, x);
            }

            matrix[position] = letters;
        }
        return matrix;
}
function findMatchingOffset(word, letter) {
        for(var i = 0; i < word.length; i++) {
            if(word[i] == letter) {
                return i;
            }
        }
        return false;
}
function findMatchingLetterMatrixPositionVertical(matrix, word, subletter, index) {
        for(var i = 0; i < word.length; i++) {
            var letter = word[i];
            if(!matrix[i]) {
                matrix[i] = '';
```

```
            }

            if(subletter == letter && (!matrix[i][index] || matrix[i][index] == ' ') &&
(!matrix[i][index + 2] || matrix[i][index + 2] == ' ')) {
                return i;
            }
        }
        return false;
}
function findMatchingLetterMatrixPosition(matrix, word, subletter, index) {
        for(var i = 0; i < word.length; i++) {
            var letter = word[i];
            if(!matrix[index]) {
                matrix[index] = '';
            }
            if(subletter == letter && (!matrix[index][i] || matrix[index][i] == ' ') &&
(!matrix[index + 2] || !matrix[index + 2][i] || matrix[index + 2][i] == ' ')) {
                return i;
            }
        }
        return false;
}
function fillLetterMatrixVertically(matrix, word, index) {
        var spacing = Array(index).join(" ");
        for(var i = 0; i < word.length; i++) {
            matrix[i] = spacing + word[i];
        }
        return matrix;
}
```

```
	/* buildUnmatchedBlock(unmatchedblock)

		Compose the unmatched block. Since nothing matches, there's nothing to build here.

	*/

function buildUnmatchedBlock(unmatchedblock) {
	return unmatchedblock;
}
function getLongestWordLength(words) {
	var length = 0;

	for(var i = 0; i < words.length; i++) {
		var word = words[i];
		var wordlength = word[0].length;
		if(wordlength > length) {
			length = wordlength;
		}
	}

	return length;
}
function compactCrosswordBlockSources(graphs) {
	for(var i = 0; i < graphs.length; i++) {
		var graph = graphs[i];

		var matrix = graph['matrix'];
```

```
            graph = compactCrosswordBlockSource(graph);

            graphs[i] = graph;
        }
        return graphs;
}
function compactCrosswordBlockSource(graph) {
        graph = compactCrosswordBlockBottom(graph);
        graph = compactCrosswordBlockTop(graph);
        graph = compactCrosswordBlockLeft(graph);
        graph = compactCrosswordBlockRight(graph);
        return graph;
}

        /* compactCrosswordBlockTop(graph)

            Compact the crossword block from the top.

        */

function compactCrosswordBlockTop(graph) {
        var crosswordblock = graph['matrix'];
        var crosswordblocksolutions = graph['matrixpositions'];
        var crosswordblockacross = graph['across'];

        var crosswordblocklength = crosswordblock.length;

        for(var i = 0; i < crosswordblocklength; i++) {
```

```
                var row = crosswordblock[i];
                var trimmedrow = $.trim(row);
                if(!row || !trimmedrow.length) {
                        crosswordblock.splice(i, 1);
                        crosswordblocksolutions =
incrementCrossWordBlockHeights(crosswordblocksolutions);
                        i--;
                        crosswordblocklength--;
                } else {
                        i = crosswordblocklength;
                }
        }

        graph['matrix'] = crosswordblock;
        graph['matrixpositions'] = crosswordblocksolutions;

        return graph;
}
function incrementCrossWordBlockHeights(crosswordblocksolutions) {
        if(!crosswordblocksolutions) {
                return crosswordblocksolutions;
        }

        crosswordblockwords = Object.keys(crosswordblocksolutions);
        for(var i = 0; i < crosswordblockwords.length; i++) {
                var crosswordblockword = crosswordblockwords[i];

                crosswordblocksolutions[crosswordblockword][0]--;
        }
```

```
        return crosswordblocksolutions;
}

        /* incrementCrossWordBlockLengths(crosswordblocksolutions)

                Increase the horizontal position of the words in a crossword block by
one.

        */

function incrementCrossWordBlockLengths(crosswordblocksolutions) {
        if(!crosswordblocksolutions) {
                return crosswordblocksolutions;
        }

        crosswordblockwords = Object.keys(crosswordblocksolutions);
        for(var i = 0; i < crosswordblockwords.length; i++) {
                var crosswordblockword = crosswordblockwords[i];

                crosswordblocksolutions[crosswordblockword][1]--;
        }
        return crosswordblocksolutions;
}

        /* compactCrosswordBlockBottom(graph)

                Compact a crossword block on the bottom.

        */
```

```
function compactCrosswordBlockBottom(graph) {
	var crosswordblock = graph['matrix'];
	var crosswordblocksolutions = graph['matrixpositions'];
	var crosswordblockacross = graph['across'];

	var crosswordblocklength = crosswordblock.length;
	for(var i = crosswordblocklength - 1; i >= 0; i--) {
		var row = crosswordblock[i];
		var trimmedrow = $.trim(row);
		if(!trimmedrow.length) {
			crosswordblock.splice(i, 1);
		} else {
			i = -1;
		}
	}

	graph['matrix'] = crosswordblock;
	graph['matrixpositions'] = crosswordblocksolutions;

	return graph;
}

	/* compactCrosswordBlockLeft(graph)

		Compact a crossword block on the left.

	*/
```

```
function compactCrosswordBlockLeft(graph) {
	var crosswordblock = graph['matrix'];
	var crosswordblocksolutions = graph['matrixpositions'];
	var crosswordblockacross = graph['across'];

	var crosswordblocklength = crosswordblock.length;

	var shorten = true;

	while(shorten) {
		if(crosswordblocklength) {
			for(var i = 0; i < crosswordblocklength; i++) {
				if(crosswordblock[i]) {
					var crosswordrow = crosswordblock[i];
					if(crosswordrow && crosswordrow[0] &&
crosswordrow[0] != ' ') {
						shorten = false;
						i = crosswordblocklength;
					}
				}
			}
		} else {
			shorten = false;
		}

		if(shorten) {
			for(var i = 0; i < crosswordblocklength; i++) {
				var crosswordrow = crosswordblock[i];
```

```
                        crosswordblock[i] = crosswordrow.substr(1,
crosswordrow.length);
                    }

                    crosswordblocksolutions =
incrementCrossWordBlockLengths(crosswordblocksolutions);
                }
        }

        graph['matrix'] = crosswordblock;
        graph['matrixpositions'] = crosswordblocksolutions;

        return graph;
}

        /* compactCrosswordBlockRight(graph)

                Compact a crossword block on the right.

        */

function compactCrosswordBlockRight(graph) {
        var crosswordblock = graph['matrix'];
        var crosswordblocksolutions = graph['matrixpositions'];
        var crosswordblockacross = graph['across'];

        var longestpiece = getWidestLine(crosswordblock) - 1;
        var crosswordblocklength = crosswordblock.length;
```

```
	var shorten = true;

	while(shorten) {
		if(crosswordblocklength) {
			for(var i = 0; i < crosswordblocklength; i++) {
				if(crosswordblock[i]) {
					var crosswordrow = crosswordblock[i];
					if(crosswordrow[longestpiece] &&
crosswordrow[longestpiece] != ' ') {
						shorten = false;
						i = crosswordblocklength;
					}
				}
			}
		} else {
			shorten = false;
		}
		if(shorten) {
			longestpiece--;
			for(var i = 0; i < crosswordblocklength; i++) {
				var crosswordrow = crosswordblock[i];
				crosswordblock[i] = crosswordrow.substr(0,
crosswordrow.length - 1);
			}
		}
	}

	graph['matrix'] = crosswordblock;
	graph['matrixpositions'] = crosswordblocksolutions;
```

```
	return graph;
}

	/* generateCrosswordBlockSources(shuffledwords)

		Make the crossword block sources, which are the sub-graphs or mini-graphs. These will be put together to make the full crossword puzzle.

	*/

function generateCrosswordBlockSources(shuffledwords) {
	var crosswordblocks = [];
	var checkedcrosswords = [];
	var clues = [];
	for(var i = 0; i < shuffledwords.length; i++) {
		var shuffledword = shuffledwords[i];
		var word = shuffledword[0].toLowerCase();
		var clue = shuffledword[1];
		clues[word] = clue;

		crosswordclues[word] = clue;

		var checkedcrosswordkey = word + '-' + clue;

		var unmatchedwords = [];

		if(!checkedcrosswords[checkedcrosswordkey]) {
			var wordletters = getLettersHashCountForWord(word);
```

```
                var crosswordblock = [];

                for(var j = i + 1; j < shuffledwords.length; j++) {
                        var nextshuffledword = shuffledwords[j];

                        var nextword = nextshuffledword[0].toLowerCase();
                        var nextclue = nextshuffledword[1];
                        var nextcrosswordkey = nextword + '-' + nextclue;

                        if(!checkedcrosswords[nextcrosswordkey]) {
                                var matchingletter = getMatchingLetter(wordletters,
nextword);
                                if(matchingletter && matchingletter.length) {
                                        wordletters[matchingletter]--;
                                        checkedcrosswords[nextcrosswordkey] = true;
                                        crosswordblock.push([nextword,
matchingletter]);
                                }
                        }
                }

                if(crosswordblock.length) {
                        crosswordblocks[word] = crosswordblock;
                } else {
                        unmatchedwords.push(word);
                }
                checkedcrosswords[checkedcrosswordkey] = true;
        }
```

```
            if(unmatchedwords.length) {
                crosswordblocks['(unmatched)'] = unmatchedwords;
            }
        }

        return {
            'blocks':crosswordblocks,
            'clues':clues,
        };
}

        /* getLettersHashPositionsForWord(word)

            Get a hash of the letters and the positions of the letters of a word.

        */

function getLettersHashPositionsForWord(word) {
        var lettershash = [];

        for(var i = 0; i < word.length; i++) {
            var letter = word[i];
            if(lettershash[letter]) {
                lettershash[letter].push(i);
            } else {
                lettershash[letter] = [i];
            }
        }
```

```
        return lettershash;
}

        /* getLettersHashCountForWord(word)

                Get a hash of the counts for the letters of a word.

        */

function getLettersHashCountForWord(word) {
        var lettershash = [];

        for(var i = 0; i < word.length; i++) {
                var letter = word[i];
                if(lettershash[letter]) {
                        lettershash[letter]++;
                } else {
                        lettershash[letter] = 1;
                }
        }

        return lettershash;
}
function getMatchingLetter(letters, nextword) {
        var matchingletter = '';

        for(var i = 0; i < nextword.length; i++) {
                var letter = nextword[i];
                if(letters[letter]) {
```

```
                return letter;
            }
        }

        return matchingletter;
}

        /* shuffle(array)

            Randomize array.

        */

function shuffle(array) {
        var currentIndex = array.length, temporaryValue, randomIndex;

        while (0 !== currentIndex) {
            randomIndex = Math.floor(Math.random() * currentIndex);
            currentIndex -= 1;

            temporaryValue = array[currentIndex];
            array[currentIndex] = array[randomIndex];
            array[randomIndex] = temporaryValue;
        }

        return array;
}
```

ДОДАТОК Ж

Лістинг контролерів

```
[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);
    }

    [HttpPost]
    public async Task<IResult> CreateTopic(TopicDto topicDto)
    {
        await _topicService.CreateAsync(topicDto);

        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()
        {
            Email = dto.Email,
            SecurityStamp = Guid.NewGuid().ToString(),
            UserName = dto.Username
        };
```

```
        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;
    }
}
```

ДОДАТОК И

Ілюстративний матеріал до захисту

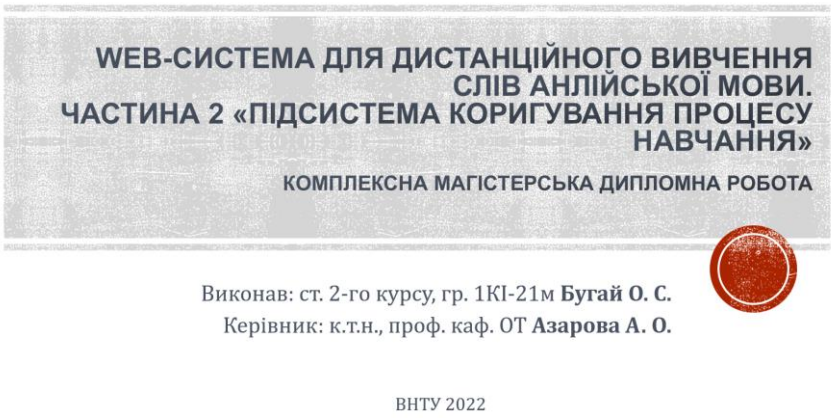


Рисунок И.1 — Слайд презентації 1

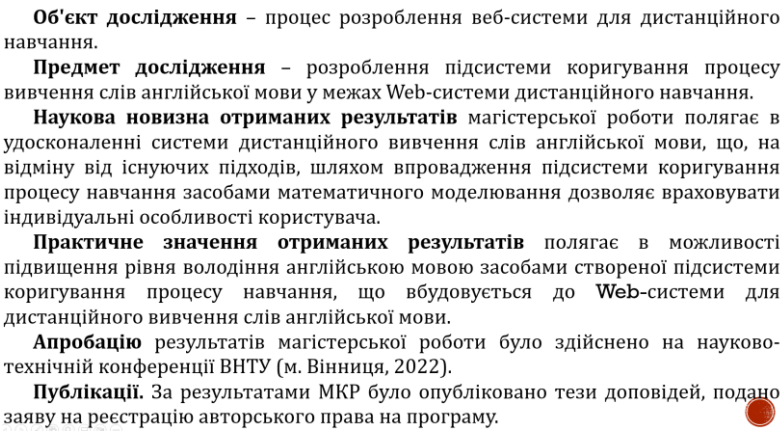


Рисунок И.2 — Слайд презентації 2

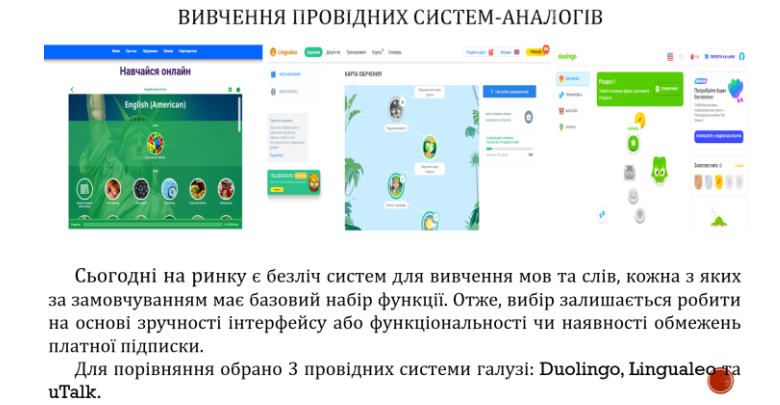
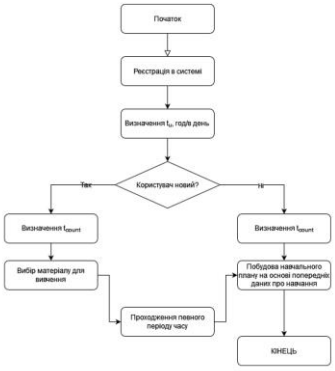


Рисунок И.3 — Слайд презентації 3

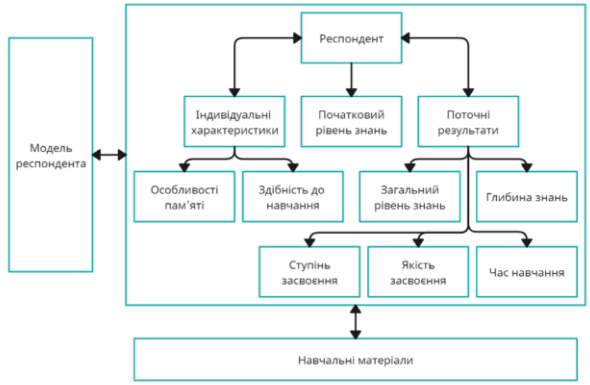
ПІДСИСТЕМА КОРИГУВАННЯ ПРОЦЕСУ НАВЧАННЯ



Алгоритм роботи підсистеми коригування процесу навчання

Рисунок И.4 — Слайд презентації 4

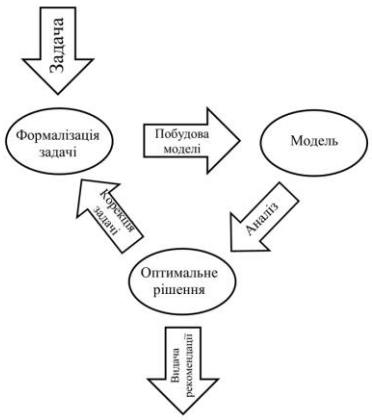
ПІДСИСТЕМА КОРИГУВАННЯ ПРОЦЕСУ НАВЧАННЯ



Структура моделі засвоєння матеріалу респондентом

Рисунок И.5 — Слайд презентації 5

ПІДСИСТЕМА КОРИГУВАННЯ ПРОЦЕСУ НАВЧАННЯ



Алгоритм видачі навчальних рекомендації

Рисунок И.6 — Слайд презентації 6

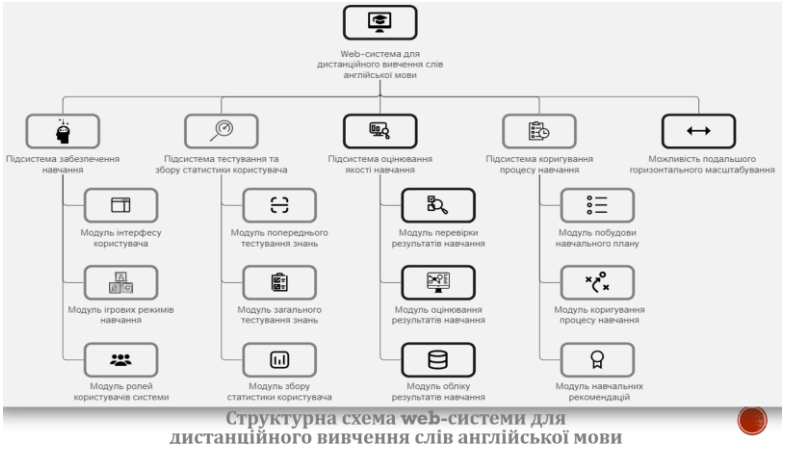


Рисунок И.7 — Слайд презентації 7

Для розташування сайту в Інтернет, необхідно замовити послуги хостинг-провайдера і зареєструвати доменне ім'я.

- Основними характеристиками при виборі серверу хостинг-провайдера є:
  - тип диску носія пам'яті (HDD, SSD, NVMe SSD);
  - центральний процесор (CPU);
  - кількість сайтів, розміщених одночасно;
  - кількість оперативної пам'яті;
  - місцезнаходження сервера;
  - ціна оренди;
- Для якісної роботи веб-системи віддамо перевагу таким характеристикам сервера: **NVMe SSD 512Gb, CPU: Intel XEON Ten Core 2.4Ghz, 16Gb RAM, UA located.**
- Для подальшого маштабування системи слід покращити апаратну складову існуючого серверу, та подальше розширення шляхом оренди нових серверів.

Рисунок И.8 — Слайд презентації 8

МАТЕМАТИЧНА МОДЕЛЬ ЗАСВОЄННЯ МАТЕРІАЛУ РЕСПОНДЕНТОМ

Загальну математичну модель знань користувача визначемо наступною залежністю:

$$Mu = \{Tu_1, Tu_1, ..., Tu_n\},$$

де Tu_n — модель вивчення n-ої теми.

Модель вивчення n-ої теми опишемо так:

$$Tu_n = \{MSu, s^n_l, k^n\},$$

де MSu — узагальнена модель теми, що вивчається;
s^n_l — слова, які належать до заданої теми;
l — кількість слів у n-ій темі;
k^n_l — коефіцієнт кількості повторень даних слів.

Рисунок И.9 — Слайд презентації 9

МАТЕМАТИЧНА МОДЕЛЬ ЗАСВОЄННЯ МАТЕРІАЛУ РЕСПОНДЕНТОМ

$$MSu = \{P_1, P_2, \ldots, P^n\},$$

де P^n — окремий параметр, що описує довільну характеристику респондента.

$$\Delta t = t_{повт.сьог.} - t_{ост.повт.},$$

де $t_{повт.сьог.}$ — час на сьогоднішній день;

$t_{ост.повт}$ — час останнього повторення.

$$S_{avg} = \frac{s_1 t_{u1} + s_2 t_{u2} + \cdots + s_n t_{un}}{n},$$

де $s_n t_{un}$ — кількість можливих слів для вивчення у вказаний час користувачем;

n — кількість користувачів.

Рисунок И.10 — Слайд презентації 10

МАТЕМАТИЧНА МОДЕЛЬ ЗАСВОЄННЯ МАТЕРІАЛУ РЕСПОНДЕНТОМ

$$M_u = \begin{cases} T_1(s_1^1, \ldots, s_a^1) = \frac{(k_1^1 \cdot s_1^1) + (k_2^1 \cdot s_2^1) + \cdots + (k_a^1 \cdot s_a^1)}{\Delta t} \cdot 100\% \\ T_2(s_1^1, \ldots, s_b^1) = \frac{(k_1^2 \cdot s_1^2) + (k_2^2 \cdot s_2^2) + \cdots + (k_b^2 \cdot s_b^2)}{\Delta t} \cdot 100\% \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ T_n(s_1^n, \ldots, s_l^n) = \frac{(k_1^n \cdot s_1^n) + (k_2^n \cdot s_2^n) + \cdots + (k_l^n \cdot s_l^n)}{\Delta t} \cdot 100\%. \end{cases},$$

де $T_n(s_1^n, \ldots, s_l^n)$— визначення кількості слів в n-ій темі;

$k_l^n \cdot s_l^n$— оцінка засвоєння слова відповідної теми;

Δt — кількість часу, що минув із моменту останнього повторення.

Рисунок И.11 — Слайд презентації 11

РЕЗУЛЬТАТ РОБОТИ РОЗРОБЛЕНОГО ПЗ

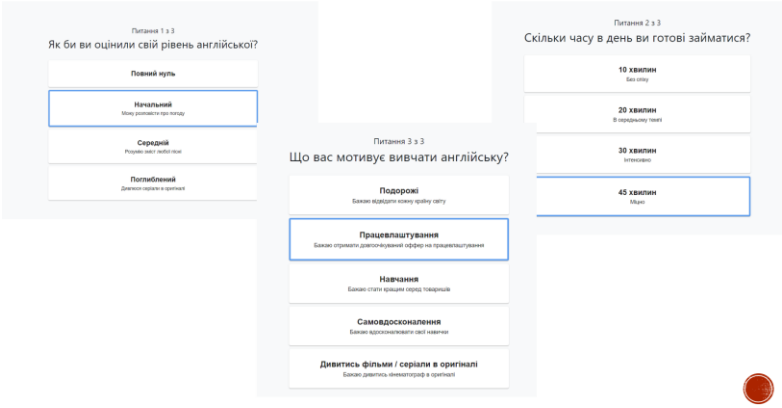


Рисунок И.12 — Слайд презентації 12

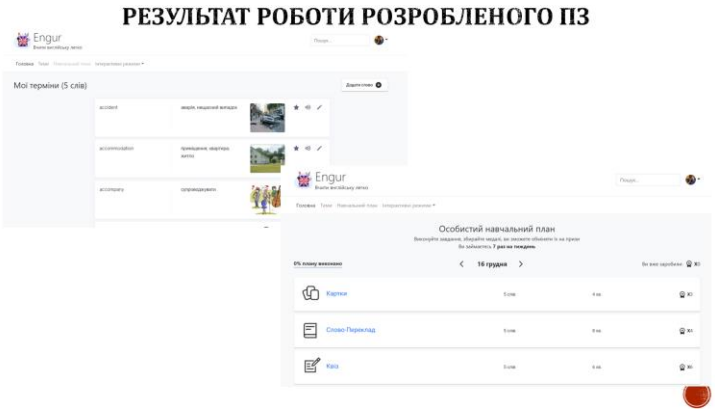


Рисунок И.13 — Слайд презентації 13

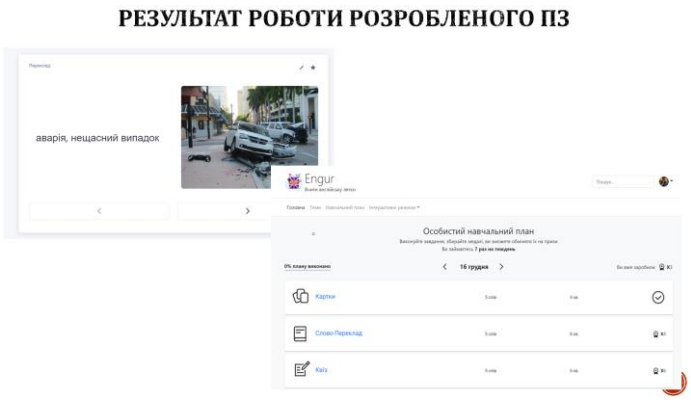


Рисунок И.14 — Слайд презентації 14

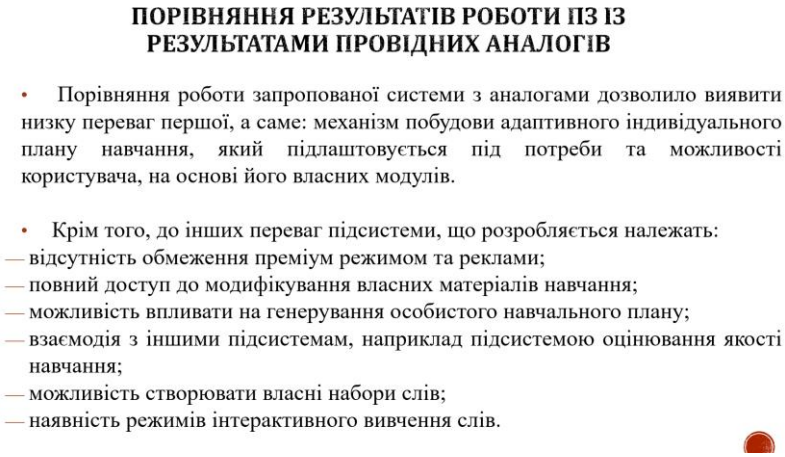


Рисунок И.15 — Слайд презентації 15

ВИСНОВОК

У магістерській кваліфікаційній роботі було спроєктовано та розроблено **web**-систему для дистанційного вивчення слів англійської мови в межах підсистеми коригування процесу навчання.

Проведено аналітичний огляд комп'ютерних систем та існуючих підходів до контролю та коригування навчального процесу. Розглянуто особливості автоматизованого адаптивного навчального плану.

На базі аналізу існуючих математичних моделей було запропоновано підхід до реалізації побудови навчального плану, досліджено механізм роботи пам'яті та на його основі розроблено математичну модель процесу модель засвоєння матеріалу (слів англійської мови) респондентом в підсистемі коригування процесу навчання.

Обґрунтовано вибір інструментів для програмної реалізації розробки для клієнтської та серверної частин окремо, а також для бази даних. Для клієнтської частини та відображення інтерфейсу користувача було вирішено розробляти **Angular**-застосунок.

Рисунок И.16 — Слайд презентації 16

ВИСНОВОК

Для серверної частини ключовою технологією обрано **.NET** та стек його суміжних фреймворків. Сервер бази даних обрано **Microsoft SQL Server**.

У результаті роботи було розроблено веб-застосунок та базу даних.

Клієнтська частина використовує підходи розбиття структури на модулі компонентів та їх маршрутизації. Серверна частина базується на трирівневій архітектурі та виконує основну логіку та обчислення над даними, а також надає інтерфейс керування програмою клієнтській частині.

Здійснено тестування та аналіз роботи розробленої системи.

Ключовою перевагою розроблюваної системи являється механізм побудови адаптивного індивідуального плану навчання, який підлаштовується під потреби та можливості користувача, на основі його власних модулів.

Отже, задачі, поставлені у магістерській кваліфікаційній роботі, було виконано в повному обсязі.

Рисунок И.17 — Слайд презентації 17

ДОДАТОК К

Протокол перевірки кваліфікаційної роботи

Назва роботи: Web-система для дистанційного вивчення слів англійської мови. Частина 2. «Підсистема коригування процесу навчання»

Тип роботи: магістерська кваліфікаційна робота
(БДР, МКР)

Підрозділ кафедра обчислювальної техніки
(кафедра, факультет)

Показники звіту подібності
Unicheck

Оригінальність 80% Схожість 20%

Аналіз звіту подібності (відмітити потрібне):

- ✓ Запозичення, виявлення у роботі, оформлені коректно і не містять ознак плагіату.
- □ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.
- □ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

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

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Бугай О. С.
(підпис) (прізвище, ініціали)

Керівник роботи _____ Азарова А. О.
(підпис) (прізвище, ініціали)