МІНІСТЕРСТВО ФІНАНСІВ УКРАЇНИ
ДЕРЖАВНИЙ ПОДАТКОВИЙ УНІВЕРСИТЕТ

ФАКУЛЬТЕТ ФІНАНСІВ ТА ЦИФРОВИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І СИСТЕМ

КВАЛІФІКАЦІЙНА РОБОТА

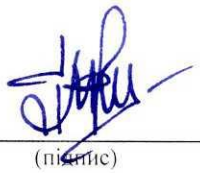
на тему: РОЗРОБКА АДАПТИВНОЇ СИСТЕМИ ДЛЯ ВИВЧЕННЯ МОВ ІЗ ВИКОРИСТАННЯМ ШТУЧНОГО ІНТЕЛЕКТУ

Здобувача вищої освіти другого (магістерського) рівня,
ІІ курсу, групи КМІ-24-2, спеціальності 122 «Комп'ютерні науки»,
освітня програма «Інформаційні управляючі системи і технології в економіці»


(підпис)

Мурги Богдана Олександровича
(прізвище, ім'я, по батькові)

Науковий керівник


(підпис)

Філоненко Михайло Миколайович, канд. фіз.-мат. наук, доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Керівник освітньо-професійної програми


(підпис)

Омельчук Антон Анатолійович, канд. техн. наук
(прізвище, ім'я, по батькові, науковий ступінь, вчен звання)

Допущено до захисту за рішенням кафедри протокол від «02» січня 2026 № 10

Завідувач кафедри


(підпис)

Філоненко Михайло Миколайович, канд. фіз.-мат. наук, доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Ірпінь – 2026

Реєстрація № 30 22.01.2026 _____
(дата) (підпис відповідальної особи)

Результати перевірки до захисту 02.01.2026 _____
(до захисту, на доопрацювання) (дата) (підпис)

Результат захисту 4 (добре) 05.02.2026 _____
(оцінка) (дата) (підпис)

Голова комісії: _____ О.С. Дюбанов
(підпис)

Члени комісії: _____ А.Ю. Горбовий
(підпис)

_____ М.М. Філоненко
(підпис)

_____ І.Д. Погорєловська
(підпис)

БЛАНК ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Об'єкти контролю	Діапазон оцінки в балах	Фактична кількість балів	Примітки	Дата, П.І.Б.: керівника, голови комісії членів комісії,секретаря, (підписи)
ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ КОМІСІЄЮ ПО ЗАХИСТУ				
1. Зміст кваліфікаційної роботи (якість теоретичного матеріалу, його зв'язок з практичним, вміння представляти матеріал в логічній послідовності, наявність висновків та пропозицій)	**від 0 до 30**	23		Дюбанов О.С.
2. Дотримання вимог щодо оформлення	**від 0 до 10**	7		Горбовий А.Ю.
3. Вчасне виконання роботи на всіх етапах у встановлені кафедрою терміни	**від 0 до 5**	3		Філоненко М.М.
				Погорєловська І.Д.
4. Рецензент	**від 0 до 5**	4		Суховецька Л.П.
РАЗОМ БАЛІВ	**від 0 до 50**	37		

Оцінювання кваліфікаційної роботи науковим керівником - 38 балів.

ПЕРЕВЕДЕННЯ ОЦІНКИ ЗА 100-БАЛЬНОЮ ШКАЛОЮ ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалоюЄКТС	Оцінка за національною шкалою	Дата	П.І.Б.: голови, членів комісії, секретаря, (підпис)
75	C	4 (добре)	05.02.2026	Дюбанов О.С.
				Горбовий А.Ю.
				Філоненко М.М.
				Погорєловська І.Д.
				Суховецька Л.П.

ПРАВИЛА ПЕРЕВЕДЕННЯ ОЦІНОК ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою
90-100	A	5 (відмінно)
80-89	B	4 (добре)
70-79	C	
60-69	D	3 (задовільно)
50-59	E	
35-49	FX	2 (незадовільно)

Факультет фінансів та цифрових технологій
Кафедра комп'ютерних та інформаційних технологій і систем
Галузь знань 12 «Інформаційні технології»
Спеціальність 122 «Комп'ютерні науки»
Освітня програма «Інформаційні управляючі системи і технології в економіці»

Затверджую
Завідувач кафедри _____ Філоненко М.М.
«11» грудня 2024

Завдання на кваліфікаційну роботу
здобувачу вищої освіти
Мурзі Богдану Олександровичу

(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи Розробка адаптивної системи для вивчення мов із використанням штучного інтелекту
Затверджена наказом ректора від «10» грудня 2024 № 1852
2. Строк здачі закінченої роботи «02» січня 2026
3. Мета роботи Проєктування та програмна реалізація адаптивної системи вивчення мов, яка за допомогою штучного інтелекту динамічно підлаштовує навчальний контент, складність вправ та темп навчання під індивідуальні особливості, рівень знань та інтереси користувача для підвищення ефективності засвоєння матеріалу.
Об'єкт дослідження Процес індивідуалізованого навчання іноземних мов у цифровому середовищі з використанням технологій адаптивної персоналізації.
Предмет дослідження Методи, алгоритми штучного інтелекту (зокрема обробки природної мови — NLP) та програмні засоби побудови адаптивної архітектури навчальної системи.
Орієнтовна база дослідження Системи для вивчення мов із використанням штучного інтелекту.
4. Вихідні дані до роботи Вимоги до обсягу та структури;
5. Перелік графічного матеріалу Таблиці порівняння;
6. Зміст кваліфікаційної роботи (перелік питань за кожним розділом)
ВСТУП; РОЗДІЛ 1 АНАЛІЗ СУЧАСНИХ ДОДАТКІВ ДЛЯ ВИВЧЕННЯ ІНОЗЕМНИХ МОВ; РОЗДІЛ 2. СТВОРЕННЯ ПРОГРАМНОГО ЗАСТОСУНКУ ДЛЯ ОСВОЄННЯ ІНОЗЕМНИХ МОВ;
РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ: АРХІТЕКТУРА, АЛГОРИТМИ ТА ІНЖЕНЕРНІ РІШЕННЯ;
РОЗДІЛ 4. АНАЛІЗ РЕЗУЛЬТАТІВ РОЗРОБКИ ТА ЕФЕКТИВНОСТІ СИСТЕМИ; ВИСНОВКИ; СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.

7. Календар план виконання роботи:

№ п/п	Назва етапів кваліфікаційноїроботи	Строк виконання етапів роботи	
		За планом	Фактично
1	2	3	4
1	Вибір теми, підбір літературних джерел, формування структури кваліфікаційної роботи	До 15.12.2024	До 15.12.2024
2	**І розділ** (розкрити сутність і стан дослідження наукової проблематики роботи, після чого навести її загальну характеристику; огляд літератури за темою і вибір конкретних напрямів досліджень)	30.05.2025	30.05.2025
3	**ІІ розділ** (аналіз експериментальної частини про методики досліджень)	10.09.2025	10.09.2025
4	**ІІІ розділ** (проведені теоретичні і (або) експериментальні дослідження та проектні рішення)	20.11.2025	20.11.2025
5	**Висновки**	11.12.2025	11.12.2025
6	Доопрацювання роботи з урахуванням зауважень наукового керівника	25.12.2025	25.12.2025
7	Отримання відгуку керівника	28.01.2026	28.01.2026
8	Отримання зовнішньої рецензії	30.01.2026	30.01.2026

8. Дата видачі завдання «11» грудня 2024

9. Керівник кваліфікаційної роботи _____ (підпис) Філоненко М.М. (прізвище, ініціали)

10. Керівник освітньо-професійної програми _____ (підпис) Омельчук А.А. (прізвище, ініціали)

11. Завдання до виконання отримав здобувач вищої освіти _____ (підпис) Мурга Б.О. (прізвище, ініціали)

АНОТАЦІЯ

РОЗРОБКА АДАПТИВНОЇ СИСТЕМИ ДЛЯ ВИВЧЕННЯ МОВ ІЗ ВИКОРИСТАННЯМ ШТУЧНОГО ІНТЕЛЕКТУ

У ході виконання кваліфікаційної роботи було проведено комплексне дослідження методологій викладання іноземних мов та реалізовано інноваційний програмний продукт — адаптивний вебзастосунок на основі штучного інтелекту.

У першому розділі здійснено ретроспективний аналіз еволюції методів лінгвістичного навчання. Проведено компаративне дослідження існуючих ринкових рішень, виявлено їхні функціональні обмеження та проаналізовано сучасні кейси інтеграції ШІ в освітній процес. Це дозволило глибоко структурувати знання про предметну область та ідентифікувати потенційні ризики розробки.

У другому розділі виконано інженерію вимог та функціональне моделювання системи. На основі аналізу потреб цільової аудиторії було сформовано концептуальну модель застосунку, визначено пріоритетні функції та розроблено детальний проєктний план, що відповідає очікуванням сучасних користувачів.

У третьому розділі обґрунтовано вибір технологічного стеку та реалізовано технічну архітектуру. Для розробки обрано мову програмування Python та відповідні спеціалізовані бібліотеки для вебсередовища. Результатом цього етапу стала побудова масштабованої архітектури та розробка мінімально життєздатного продукту (MVP), що демонструє ключові можливості ШІ-модуля.

У четвертому розділі проведено функціональний аналіз розробленого продукту. Оцінка прийнятих технічних рішень підтвердили доцільність обраної архітектури та методології. Отримані результати свідчать про високу продуктивність системи та її потенційну конкурентоспроможність на ринку EdTech.

Кваліфікаційна робота містить 53 сторінок, 2 таблиці, список використаних джерел 29 найменувань.

Ключові слова: штучний інтелект, інформаційна система, вивчення мов.

Рік захисту роботи 2026

ABSTRACT

DEVELOPMENT OF AN ADAPTIVE SYSTEM FOR LANGUAGE LEARNING USING ARTIFICIAL INTELLIGENCE

During the qualification work, a comprehensive study of foreign language teaching methodologies was conducted and an innovative software product was implemented - an adaptive web application based on artificial intelligence.

In the first section, a retrospective analysis of the evolution of linguistic teaching methods was carried out. A comparative study of existing market solutions was conducted, their functional limitations were identified, and modern cases of AI integration into the educational process were analyzed. This allowed for a deep structuring of knowledge about the subject area and identification of potential development risks.

In the second section, requirements engineering and functional modeling of the system were performed. Based on an analysis of the needs of the target audience, a conceptual model of the application was formed, priority functions were determined, and a detailed project plan was developed that meets the expectations of modern users.

The third section justifies the choice of the technological stack and implements the technical architecture. The Python programming language and relevant specialized libraries for the web environment were selected for development. The result of this stage was the construction of a scalable architecture and the development of a minimum viable product (MVP), which demonstrates the key capabilities of the AI module.

In the fourth section, a functional analysis of the developed product was conducted. The assessment of the adopted technical solutions confirmed the feasibility of the selected architecture and methodology. The results obtained indicate the high performance of the system and its potential competitiveness in the EdTech market.

The qualification work contains 53 pages, 2 tables, a list of used sources of 29 names.

Keywords: artificial intelligence, information system, language learning.

Year of defense of the work 2026

Зміст

ВСТУП 3

РОЗДІЛ 1. АНАЛІЗ СУЧАСНИХ ДОДАТКІВ ДЛЯ ВИВЧЕННЯ ІНОЗЕМНИХ МОВ 6

1.1 Еволюція методик вивчення іноземних мов 6

1.2 Загальний огляд підходу до вивчення іноземних мов за допомогою веб-додатків 10

РОЗДІЛ 2. СТВОРЕННЯ ПРОГРАМНОГО ЗАСТОСУНКУ ДЛЯ ОСВОЄННЯ ІНОЗЕМНИХ МОВ 19

2.1. Формулювання вимог до системи та епістемологічний розбір потреб користувачів. 19

2.2. Функціональне планування та архітектурне проєктування із залученням Теорії обмежень (TOC) та принципів Доменно-орієнтованої розробки (DDD) 22

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ: АРХІТЕКТУРА, АЛГОРИТМИ ТА ІНЖЕНЕРНІ РІШЕННЯ 29

3.1. Стратегічна методологія розробки та управління життєвим циклом програмного продукту 29

3.2. Глибинний аналіз функціональних вимог та проектування доменних сервісів 31

3.3. Критичний аналіз та обґрунтування технологічних рішень в контексті сучасних трендів 34

РОЗДІЛ 4. АНАЛІЗ РЕЗУЛЬТАТІВ РОЗРОБКИ ТА ЕФЕКТИВНОСТІ СИСТЕМИ 44

ВИСНОВКИ 50

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 51

ВСТУП

Актуальність теми У сучасних умовах глобалізації та стрімкої інтеграції світу знання іноземних мов давно перестало бўти лише конкурентною перевагою — воно перетворилося на одну з ключових компетентностей, необхідних як для професійного поступу, так і для гармонійного особистісного розвитку. Жвава міжнародна взаємодія в економіці, науці, технологіях, культурі, освіті та повсякденному житті вимагає від фахівців уміння вільно, точно й упевнено спілкуватися з носіями мови та колегами з різних країн без постійного посередництва перекладачів. Окрім практичної цінності, численні нейронаукові та психологічні дослідження останніх років доводять, що систематичне опанування іноземних мов позитивно впливає на роботу мозку: покращує робочу та довгочасну пам'ять, розвиває когнітивну гнучкість, посилює здатність до концентрації уваги, сприяє формуванню навичок критичного й аналітичного мислення, а також стимулює креативність та здатність бачити проблему з різних перспектив. Водночас традиційні методи до опанування мов — переважно аудиторні заняття за єдиним підручником, однакова швидкість проходження матеріалу для всіх, обмежена кількість мовленнєвої практики — часто не відповідають реальним потребам сучасного учня/студента/фахівця. Фіксований темп навчання, недостатня персоналізація, брак якісного зворотного зв’язку в реальному часі, низька інтерактивність та одноманітність завдань нерідко призводять до втрати мотивації, швидкого вигорання та повільнішого прогресу. На щастя, бурхливий розвиток цифрових технологій, зокрема штучного інтелекту, великих мовних моделей та засобів обробки природної мови (NLP), відкриває принципово нові можливості для подолання цих бар’єрів. Сучасні ШІ-системи дозволяють створювати адаптивні навчальні траєкторії, які в реальному часі враховують поточний рівень знань, індивідуальний стиль сприйняття інформації, сильні та слабкі сторони учня, його інтереси та конкретні цілі. Автоматизоване оцінювання навичок (говоріння, аудіювання, письмо, читання) стає об’єктивним, швидким і деталізованим. Водночас ШІ здатен генерувати

персоналізований контент, пропонувати цікаві діалоги, ігрові сценарії, адаптивні вправи та миттєвий коректний фідбек. Такий підхід відкриває шлях до реалізації принципів адаптивного навчання, мікронавчання, гейміфікації та значного підвищення залученості учня — факторів, які, за даними досліджень, суттєво впливають на довгострокове утримання знань і мотивацію до регулярних занять.

Мета роботи - проєктування та програмна реалізація адаптивної системи вивчення мов, яка за допомогою штучного інтелекту динамічно підлаштовує навчальний контент, складність вправ та темп навчання під індивідуальні особливості, рівень знань та інтереси користувача для підвищення ефективності засвоєння матеріалу.

Об'єкт дослідження - процес індивідуалізованого навчання іноземних мов у цифровому середовищі з використанням технологій адаптивної персоналізації.

Предмет дослідження - методи, алгоритми штучного інтелекту (зокрема обробки природної мови — NLP) та програмні засоби побудови адаптивної архітектури навчальної системи.

Завдання дослідження

1. **Проаналізувати сучасний стан** ринку EdTech та існуючих систем для вивчення мов, виявивши переваги та недоліки використання ШІ в них.
2. **Дослідити алгоритми машинного навчання та NLP** (Natural Language Processing), які можуть бути використані для оцінки знань користувача, розпізнавання мовлення та генерації персоналізованих завдань.
3. **Спроєктувати архітектуру системи**, що включає модуль профілювання користувача, інтелектуальну базу знань та механізм адаптації контенту.
4. **Обґрунтувати вибір технологічного стеку** (наприклад, Python/Node.js, інтеграція з API великих мовних моделей, таких як OpenAI GPT, або використання бібліотек типу Hugging Face).

5. **Розробити програмний прототип системи**, реалізувавши ключові функції: вхідне тестування, інтерактивні вправи та систему інтелектуального зворотного зв'язку.
6. **Провести тестування розробленого рішення** на предмет точності адаптивних алгоритмів та зручності інтерфейсу (Usability Testing), а також оцінити потенційну ефективність системи порівняно з традиційними підходами.

РОЗДІЛ 1. АНАЛІЗ СУЧАСНИХ ДОДАТКІВ ДЛЯ ВИВЧЕННЯ ІНОЗЕМНИХ МОВ

1.1 Еволюція методик вивчення іноземних мов

Еволюція підходів до опанування іноземних мов

Систематичне опанування чужих мов є однією з найстаріших та найсталіших форм інтелектуальної/культурної діяльності людства. Протягом віків методологічні засади зазнавали суттєвих змін, що було відображенням розвитку лінгвістичних теорій, психологічних уявлень про навчання, освітніх парадигм та соціоекономічних потреб — від підготовки обмеженого кола осіб для читання античних текстів до формування глобальної здатності до спілкування в умовах сучасної цифровізації. Цей огляд хронологічно висвітлює ключові етапи, зосереджуючись на зміні пріоритетів, теоретичних засадах, а також перевагах та недоліках кожного напрямку.

1. Метод граматики та перекладу (XVIII ст. – середина XX ст.)

Цей спосіб, що викристалізувався спершу у викладанні класичних мов (латини, давньогрецької), пізніше поширився на нові мови, пануючи в європейській освітній системі (школи та виші) кілька століть. Теоретичну базу становила раціоналістична традиція, згідно з якою мова — це насамперед логічна структура правил, а також інструмент для розвитку мислення.

Ключові аспекти:

викладання граматики за принципом дедукції (спочатку правила, потім приклади та вправи);

переклад виступав головним інструментом для пояснень та контролю знань;

словниковий запас засвоювався через двомовні переліки та літературні зразки;

акцент робився на читанні та письмовій роботі, тоді як говоріння та сприйняття на слух ігнорувалися.

Сильні сторони: забезпечував глибоке розуміння будови слів та речень, розвивав аналітичне мислення та точність при перекладі складних текстів.

Слабкі сторони: повна відсутність зв'язку з реальною мовленнєвою практикою, низька мотивація через брак практичної користі, неефективність у розвитку навичок сприйняття та виробництва мовлення в усному вигляді.

2. Напрямок «Прямий метод» та його послідовники; підходи на основі слухання й мовлення (кінець XIX ст. – 1960-ті рр.)

Обмеженість граматично-перекладного підходу спричинила повернення до ідей «натурального» засвоєння (подібно до того, як діти опановують рідну мову). «Прямий метод» (Direct Method, наприклад, система Берліца) скасовував використання перекладу та рідної мови. Згодом, під впливом біхевіоризму (Скіннер), розвиванулися аудіолінгвальний (Audio-Lingual Method, відомий як «армійський метод») та аудіовізуальний підходи.

Основні риси:

цільмова мова використовувалася виключно;

інтенсивне повторення (набивання) типових структурних зразків, діалогів, з акцентом на міміці та імітації;

фокус на аудіюванні, чіткій вимові та автоматизації миттєвих мовних реакцій;

залучення технічних засобів (мовні кабінети, магнітофони, плівкові матеріали та слайди, а пізніше — відео).

Переваги: значно поліпшувалися навички фонетики, швидко формувалися базові мовленнєві звички, метод був ефективним у масовому навчанні (наприклад, військових програмах США).

Недоліки: механічний характер навчання, ігнорування когнітивних процесів та творчого потенціалу мови, нездатність підготувати до імпровізованої бесіди у непередбачених ситуаціях, швидке згасання зацікавленості через монотонність.

3. Комунікативна педагогіка та її наступні модифікації (1970-ті – 2000-ні роки)

Комунікативна методика викладання мов (Communicative Language Teaching — CLT), що базувалася на концепції комунікативної спроможності (Гаймс, 1972; Канал і Свейн, 1980), стала домінуючою моделлю наприкінці минулого століття. Вона здійснила докорінний перехід фокусу від мовної форми до її функції та змісту.

Ключові елементи:

пріоритет реальних комунікативних завдань (обмін інформацією, рольові ігри, вирішення проблем, проєктна робота);

застосування автентичних матеріалів та ситуацій;

введення граматики індуктивно або у контексті;

толерантне ставлення до помилок як невід'ємного етапу формування проміжної мови (interlanguage);

збалансований розвиток усіх чотирьох навичок, де пріоритет надавався вільності мовлення (fluency) разом із коректністю (accuracy).

Плюси: помітне зростання зацікавленості учнів, розвиток усіх складових комунікативної майстерності (лінгвістичної, соціолінгвістичної, дискурсивної, стратегічної), практичне спрямування.

Мінуси: потенційно недостатня увага до систематизації граматичних знань, висока значущість кваліфікації викладача, труднощі в об'єктивній оцінці прогресу.

Починаючи з 1990-х років розпочалася ера пост-комунікативних чи пост-методичних шукань (Кумаравадивелу, 1994; 2006), яка ознаменувалася відмовою від ідеї «одного універсального методу» на користь гнучких, контекстно-

орієнтованих, заснованих на принципах підходів (principled eclecticism), навчання на основі завдань (TBLT), спільного вивчення змісту та мови (CLIL) та орієнтації на учня (learner-centeredness).

4. Цифрова доба та інтеграція ШІ-інструментів (2000-і – 2026 р.)

Прогрес у сфері інформаційно-комунікаційних технологій (ІКТ), а також мобільне навчання, робота з великими обсягами даних, машинне навчання та генеративний штучний інтелект (ШІ) радикально змінили сферу мовної освіти, створивши умови для надзвичайно персоналізованого, доступного та глибокого (імерсивного) навчання.

Ключові ознаки нинішнього етапу:

Адаптивне навчання: системи через алгоритми в режимі реального часу аналізують відповіді, хибні кроки, стиль навчання та інтереси, динамічно змінюючи рівень складності, порядок та тип пропонованих завдань;

Гейміфікація та мікронавчання: короткі інтерактивні епізоди, що містять елементи змагання, заохочення та відстеження прогресу;

Імерсивні технології: застосування віртуальної (VR) та доповненої (AR) реальності для імітації справжніх мовних середовищ (бесіди у закладах харчування, подорожі, робочі наради);

ШІ-помічники: чат-боти, що працюють із природною мовою (включно з великими мовними моделями), автоматична перевірка вимови (технології розпізнавання мовлення), генерування індивідуальних текстів/діалогів, корекція письмових робіт у реальному часі, інтелектуальне наставництво;

Розвиток самостійності учня (learner autonomy) та поняття навчання протягом усього життя (lifelong learning).

Суттєві переваги: безпрецедентна можливість масштабування, висока індивідуалізація, цілодобова доступність, поєднання розваги з навчальним процесом, миттєва та об'єктивна оцінка результатів.

Виклики: необхідність володіння цифровими навичками та самодисципліною, небезпека поверхневого засвоєння без глибокого осмислення,

етичні дилеми (захист даних, надмірна залежність від технічних засобів), потреба у поєднанні інструментів ШІ з перевіреними педагогічними методиками.

Сучасна методологія (станом на 2026 рік) характеризується змішаністю та плюралізмом пост-методичного підходу: найкращі елементи минулих напрямків (комунікативна спрямованість, орієнтація на завдання, достовірність матеріалів) поєднуються з величезним потенціалом ШІ для створення максимально ефективного, мотивуючого та гнучко налаштовуваного навчального досвіду, сфокусованого на унікальних потребах конкретного учня та його обставинах.

1.2 Загальний огляд підходу до вивчення іноземних мов за допомогою веб-додатків

У теперішню добу цифровізації веб-додатки, призначені для оволодіння новою мовою, стали одними з найпоширеніших, легкодоступних та масових інструментів для самостійного навчання та підтримки вже здобутих мовних умінь. Цей підхід, що кардинально відходить від класичних методів освіти (як-от заняття в аудиторіях, робота з друкованими посібниками чи індивідуальні уроки з репетитором), поступово переходить до парадигми, опосередкованої технологіями та орієнтованої на користувача. Він активно впроваджує елементи адаптивного навчального процесу, гейміфікації, мікронавчання та застосування штучного інтелекту. Теоретичну й практичну основу для такого методу становлять актуальні концепції індивідуалізованого навчання (personalized learning), теорія самостійного регулювання процесу навчання (self-regulated learning), а також емпіричні дані щодо впливу мотиваційних та пізнавальних чинників на успішність опанування другої мови (2).

Основні засади функціонування мовних веб-застосунків: гнучкість, портативність та загальна доступність

Навчальний процес організовано за принципом «коли завгодно — де завгодно» (just-in-time, just-in-place learning), що забезпечує повну синхронізацію даних між різними платформами (веб, мобільні ґаджети, персональні комп’ютери). Це усуває географічні, часові та фінансові бар’єри, які неминуче супроводжують традиційні форми навчання. Порівняно з офлайн-форматами, початковий поріг участі є мінімальним: для початку достатньо мати лише інтернет-браузер та підключення до мережі, без потреби у спеціалізованому обладнанні чи дотриманні жорсткого розкладу.

Гейміфікація як головний рушій мотивації

Впровадження ігрових елементів (gamification) — як-от нарахування балів, підвищення рівнів, здобуття досягнень, підтримка щоденних послідовностей (streaks), змагання у таблицях лідерів, віртуальні заохочення, елементи сюжету — спирається на теорії внутрішньої та зовнішньої мотивації (за Дезі та Раяном, теорія самодетермінації), залучаючи механізми отримання винагороди, що викликають викид ендорфінів, та психологічний ефект «нерозривного ланцюга» (Don’t break the chain). Дослідження демонструють, що гейміфікація значно посилює залученість користувачів, знижує рівень стресу, асоційованого з мовним навчанням, і сприяє сталості занять.

Адаптивність та індивідуалізація на основі аналізу даних

Завдяки застосуванню алгоритмів машинного навчання та ШІ, відбувається аналіз поведінки користувача у режимі реального часу: оцінюється точність відповідей, швидкість реакції, частота помилок, типи завдань, що викликають найбільші труднощі, та індивідуальний стиль навчання. На основі цих даних система динамічно конструює персоналізований навчальний шлях: посилює кількість вправ на проблемних темах, автоматично пропускає вже освоєний матеріал, коригує ступінь складності та призначає повторення згідно з принципом інтервального повторення (spaced repetition). Це повністю відповідає

концептам адаптивних навчальних систем (adaptive learning systems) та інтелектуальних систем наставництва (intelligent tutoring systems).

Детальний огляд переваг веб-застосунків для вивчення мов

Персоналізація та адаптивна методика навчання

Сучасні платформи (зокрема Duolingo, Busuu, Babbel, Memrise та інші) використовують моделі машинного навчання для створення профілів користувачів і прогнозування найбільш ефективного маршруту. Якщо система виявляє, що користувач регулярно припускається помилок у певній граматичній конструкції (наприклад, у часі Present Perfect), вона автоматично формує цілеспрямовані завдання, надає роз’яснення, наводить приклади або ж пропонує матеріал у нетипових форматах (через відео, діалоги, аудіозаписи). Актуальні дослідження 2024–2025 років підтверджують, що адаптивні системи підвищують успішність засвоєння матеріалу (особливо лексики та граматичних структур), покращують запам'ятовування (retention) і дозволяють досягати кращої результативності швидше, ніж це відбувається у разі проходження фіксованих за програмою курсів.

Мотиваційні та психологічні механізми

Елементи гейміфікації активують такі механізми:

механізм «стріку» — візуалізація безперервності занять, що стимулює страх втрати досягнутого (loss aversion);

соціальне зіставлення — використання таблиць лідерів, функцій додавання друзів та участі у спільнотах;

миттєва зворотна реакція — негайне виправлення помилок та позитивне підкріплення;

відчуття досягнення та поступу — формування відчуття власної компетентності та самостійності.

База емпіричних даних (мета-аналізи та систематичні огляди за 2023–2025 роки) свідчить про зростання рівня мотивації, зниження мовної тривожності, підвищення частоти занять та збільшення тривалості навчальних сесій. Гейміфікація показує найбільшу ефективність серед молоді та підлітків.

Економічна вигода та інклюзивність

Переважна більшість платформ оперує за freemium-моделлю: базовий доступ надається безкоштовно, тоді як доступ до преміум-функцій (можливість завантаження для офлайн-роботи, відсутність реклами, розширені уроки) вимагає значно менших фінансових вливань, ніж оплата традиційних курсів чи репетиторства. Відсутність витрат, пов'язаних із придбанням підручників, оплатою проїзду та дотриманням фіксованого розкладу, робить навчання доступним для широкого кола осіб, включаючи тих, хто має обмежений фінансовий ресурс, живе у віддалених місцевостях або має особливі потреби (наприклад, забезпечення інклюзивності через адаптивні опції для людей з обмеженими фізичними можливостями).

Підсумовуючи, веб-застосунки забезпечують перехід від уніфікованого підходу («один розмір для всіх») до гіперперсоналізованого, мотивуючого та легкодоступного навчання, що повністю відповідає сучасним вимогам до освіти, яка триває протягом усього життя (lifelong learning). Однак, їхня максимальна результативність досягається тоді, коли вони комбінуються з іншими формами мовної практики (наприклад, розмовні клуби, занурення у мовне середовище), оскільки ці інструменти є найбільш ефективними на початковому та середньому рівнях для формування базових навичок та доведення дій до автоматизму.

1.3 Критичний аналіз веб-додатків для вивчення мов: Між зручністю та реальною компетенцією

У сучасному цифровому освітньому ландшафті, такі програми, як Duolingo, Busuu, Memrise та Babbel, позиціонуються як проривні інструменти для вивчення мов [6-12]. Проте, при більш детальному розгляді, стає очевидною значна відмінність між тим, що ці програми пропонують як легкий спосіб виробити звичку, і тим, що насправді потрібно для досягнення вільного володіння (fluidity). Нижче ми проаналізуємо переваги та недоліки кожної платформи, керуючись такими критеріями, як глибина подачі матеріалу, дієвість

механізмів підтримки інтересу, ступінь персоналізації та потенціал у формуванні навичок вільного спілкування.

Duolingo: залучення широкого кола користувачів с через ігрові Елементи

Сильні сторони:

Легкість доступу та комфорт: Безкоштовна модель, мінімальні за обсягом заняття та синхронізація на різних ґаджетах роблять його чудовим стартовим інструментом для формування щоденної рутини.

Дизайн, що мотивує: Яскравий візуальний інтерфейс, талісман Duo та гейміфіковані аспекти (продовження серій, рівні) суттєво підвищують залученість користувача, особливо серед молодшої аудиторії.

Різноманіття мов: Представлення понад 40 мов, включаючи менш поширені, надає унікальну можливість для початкового знайомства з ними.

Слабкі сторони:

Обмежена глибина навчального курсу: Пояснення граматичних тем даються уривками, через натяки, що веде до поверхового, несистематизованого розуміння. Фрази часто виглядають неприродно і не готують до реальної комунікації.

Бар'єр для розвитку fluency: Технології розпізнавання голосу не завжди точні, немає інтеракції з мовцями-носіями. Додаток добре працює на початкових рівнях (A1-A2) для збагачення лексикону, але не готує до неформальної розмови.

Напруга замість стимулу: Система «життів» (що карає за помилки) може стати демотивуючим фактором, а надмірна увага до ігор іноді відволікає від суті вивчення.

Висновок: Duolingo – відмінний помічник для старту та підтримки регулярності занять, але самостійно вивести на рівень вільного володіння він не здатен. Його роль — допоміжна, спрямована на закріплення базового словника та простих граматичних конструкцій.

Busuu: спільна сактика та структурований шлях

Сильні сторони:

Соціальна взаємодія та фідбек: Унікальна функція отримання виправлень письмових та усних завдань від людей, для яких ця мова рідна, забезпечує цінний практичний досвід та додає впевненості у мовленні.

Структурованість та практична орієнтація: Уроки логічно побудовані відповідно до шкали CEFR (A1-B2) та сфокусовані на сценаріях з реального життя (туризм, побут), що робить його ефективнішим для середнього етапу навчання.

Комплексний розвиток: Програма розвиває усі мовні навички одночасно: сприйняття на слух, читання, письмо та говоріння.

Слабкі сторони:

Нестабільна якість корекцій: Відгуки від спільноти сильно різняться за рівнем експертизи, що іноді призводить до неточних правок або значних затримок у часі відповіді.

Обмеження безкоштовного доступу: Значний обсяг навчального матеріалу, включно з розширеними модулями та функціоналом, доступний лише за умови платної підписки.

Вузький вибір мов: Платформа підтримує лише 13 мов, що обмежує її привабливість для широкого кола користувачів.

Вердикт: Busuu — це цінний інструмент для тих, хто цінує комунікацію у групі та прагне досягти рівня самодостатності в мові (B1-B2). Однак, для глибокого занурення або для досягнення вищих рівнів (C1), він потребуватиме доповнення іншими навчальними ресурсами.

Memrise: фокус на збагаченні словника

Сильні сторони:

Висока ефективність запам'ятовування: Застосування принципу інтервального повторення (SRS) та методів візуалізації (відеофрагменти з носіями, мнемонічні асоціації) сприяє міцному фіксуванню лексики у довготривалій пам'яті.

Автентичність та гнучкість: Наявність відеопідбірок з реальними людьми у природному середовищі дає можливість почути живе мовлення, а користувацькі курси відкривають доступ до спеціалізованої лексики.

Доступність: Базовий функціонал, достатній для активного поповнення словникового запасу, доступний навіть у безкоштовній версії.

Слабкі сторони:

Відсутність системного граматичного навчання: Програма практично не пояснює правила конструювання речень, зосереджуючись лише на окремих словах та стійких виразах.

Ризик неякісного контенту: Курси, створені користувачами, можуть містити помилки або бути погано структурованими.

Односпрямованість: По суті, Memrise є високорозвиненим тренажером для лексики, а не цілісною системою для розвитку комунікативних умінь.

Вердикт: Memrise найкраще виконує свою спеціалізовану функцію – стрімке збільшення активного словника. Проте, його обов'язково слід інтегрувати з ресурсами або курсами, які фокусуються на граматиці та усній взаємодії.

Babbel: системний підхід, орієнтований на практику

Сильні сторони:

Чітка архітектура та деталізація: Уроки, розроблені фахівцями-лінгвістами, мають логічну послідовність, поєднуючи пояснення граматичних норм із практичними діалогами. Це формує надійну базу розуміння мовної структури.

Орієнтація на застосування: Акцент на засвоєнні готових комунікативних моделей та сценаріїв із повсякденного життя дозволяє швидше почати використовувати здобуті знання на практиці.

Збалансований дизайн: Мінімалістичне, але функціональне оформлення, вільне від надмірної гейміфікації, краще підходить для більш зрілої та цілеспрямованої аудиторії.

Слабкі сторони:

Модель оплати: Майже весь цінний контент доступний виключно за передплатою, що ускладнює попереднє тестування програми.

Відсутність живої комунікації: Немає можливості прямої взаємодії з мовцями-носіями (як це реалізовано у Busuu), що залишає незаповненою прогалину у розвитку спонтанного мовлення.

Менша захопливість формату: Для тих, хто звик до динамічної гейміфікації, Babbel може видатися дещо «академічним».

Висновок: Babbel є найкращим вибором для користувачів, які цінують структурованість, чіткість викладення матеріалу та прагнуть швидко освоїти основи для побутового спілкування. Він створює міцний каркас знань, який, однак, потребує «оживлення» через досвід реального мовного обміну.

Порівняльний аналіз доводить, що кожна програма займає свою унікальну нішу:

Duolingo — для закріплення звички та інтерактивного знайомства з базовими елементами мови.

Busuu — для тих, хто активно шукає комунікації та цінує зворотний зв'язок від спільноти.

Memrise — для швидкого та довготривалого розширення словникового запасу.

Babbel — для системного вивчення граматики у контексті практичних комунікативних завдань.

Жоден із цих інструментів сам по собі не забезпечить досягнення fluency. Їхні спільні обмеження полягають у недостатній адаптованості до

індивідуальних потреб, імітації реальної взаємодії та нездатності повністю замінити ефект повного занурення в мовне середовище. Отже, найбільш ефективною буде гібридна методика: слід комбінувати використання обраного додатка (як джерела структурованої практики та підтримки мотивації) з регулярно встановленим спілкуванням з носіями (онлайн чи віч-на-віч), активним споживанням автентичних матеріалів (кіно, аудіокниги, література) та, по можливості, консультаціями з викладачем для корекції напрямку та розв'язання складних моментів. Саме такий комплексний підхід дозволяє перетворити пасивні знання, отримані з додатків, на впевненість у живому спілкуванні.

РОЗДІЛ 2. СТВОРЕННЯ ПРОГРАМНОГО ЗАСТОСУНКУ ДЛЯ ОСВОЄННЯ ІНОЗЕМНИХ МОВ

2.1. Формулювання вимог до системи та епістемологічний розбір потреб користувачів.

Епістемологічні основи дослідження потреб та системна трансформація вимог

Створення веб-платформи для вивчення мов — це багатогранний процес, що лежить на стику розробки програмного забезпечення, дидактики у сфері освіти, когнітивної психології та інформаційної ергономіки. Успішна реалізація такого проєкту вимагає не прямолінійного, інтеративно-рекурсивного підходу до аналізу, де кожен наступний крок поглиблює та уточнює попередній. Ключовим є початковий етап — формування онтології предметної сфери, яке включає визначення основних сутностей («користувач», «мова», «компетентність», «навчальний шлях»), їх характеристик та взаємозв'язків.

Цей онтологічний фундамент слугує базою для подальшого виявлення потреб, які не є фіксованими, а змінюються в міру того, як користувач взаємодіє із системою. Таким чином, класичний метод «збору вимог» замінюється моделлю «постійного діалогу з доменом», що спирається на принципи Domain-Driven Design (DDD). У цій парадигмі потреби трактуються як доменні події (наприклад, «Студент не вміє застосувати граматичне правило під час вільного говоріння»), які згодом конвертуються у специфічні доменні сервіси та об'єкти-значення.

Стратифікована методологія аналізу та специфікації вимог

Процес аналізу структурований як багатошарова система, що поєднує кількісні та якісні методи досліджень.

Рівень 1: Етнографічне дослідження та виявлення прихованих потреб.

Замість традиційного опитування використовується етнографічний огляд у цифровому просторі. Це передбачає:

Аналіз цифрового сліду (Digital Footprint Analysis): Вивчення активності користувачів на тематичних форумах (типу Reddit r/languagelearning), у соцмережах та на спеціалізованих платформах (як StackExchange) за допомогою текстової аналітики. Мета — виявити невисловлені проблеми (наприклад, фраза «я розумію правило, але не можу його використати» свідчить про розрив між знанням і практичним застосуванням).

Контекстуальні спостереження (Contextual Inquiry): Проведення цілеспрямованих інтерв'ю з представниками цільових груп (від початківців до викладачів) не у вакуумі, а в процесі реального навчання, фіксуючи емоційні та розумові реакції.

Картування емоційного шляху (Emotional Journey Mapping): Побудова детальних діаграм, що відображають емоційний стан користувача на різних етапах взаємодії з існуючими засобами — від початкового ентузіазму до розчарування через «навчальне плато».

Рівень 2: Формалізація та структурування вимог із застосуванням сучасних фреймворків.

Отримані якісні дані перетворюються на формальні специфікації. Замість типового SRS (IEEE 830) пропонується використовувати модель C4 (Контекст, Контейнери, Компоненти, Код) для архітектурного опису у поєднанні з предметно-орієнтованими мовами моделювання (DSML).

Функціональні вимоги через User Story Mapping: Замість простого переліку створюється двовимірна карта історій користувача. По горизонталі відображається часова послідовність дій (основна лінія), по вертикалі — пріоритети (знизу вгору). Це дає можливість наочно представити мінімально життєздатний продукт (MVP) та його розвиток. Наприклад, історія «Як користувач, я потребую пояснення моєї помилки безпосередньо в контексті, щоб

осягнути принцип, а не просто занотувати конкретний випадок» формує вимогу до системи контекстуалізованого зворотного зв'язку.

Нефункціональні вимоги як архітектурні характеристики (Architectural Characteristics): Вимоги до швидкодії, безпеки, масштабованості описуються не ізольовано, а як обмеження, що диктують архітектурні рішення. Наприклад, вимога «адаптація до рівня користувача» перетворюється на архітектурну характеристику «персоналізація», яка вимагає впровадження:

Профілю користувача з динамічними параметрами (поточний рівень, швидкість забування, стиль навчання).

Механізму рекомендацій на базі колаборативної фільтрації та контентного аналізу.

Підтримки A/B-тестування різних методик викладання для окремих груп користувачів.

Рівень 3: Конкурентний аналіз у світлі теорії «Блакитного океану».

Аналіз ринку трансформується з простого порівняння функціоналу у стратегічний пошук відповідно до концепції «Блакитного океану» (Blue Ocean Strategy). Створюється стратегічна карта (Strategy Canvas), де вздовж осі X розташовані ключові фактори конкуренції (вартість, елементи гейміфікації, обсяг словника, якість AI-чату, структура курсу), а вздовж осі Y — рівень реалізації цих факторів у пропозиціях конкурентів.

Завдання — ідентифікувати фактори, які індустрія намагається покращити (наприклад, кількість доступних мов у Duolingo), та фактори, якими нехтують (наприклад, глибинне соціокультурне підґрунтя чи інтеграція з автентичними медіаресурсами). Це дає змогу сформулювати унікальну ціннісну пропозицію (UVP), що створює новий простір на ринку («блакитний океан»). Наприклад, UVP може мати такий вигляд: «Платформа, що фокусується не стільки на мові, скільки на комунікативній майстерності у певній професійній чи соціальній ситуації завдяки імітаційним AI-симуляціям».

Рівень 4: Вибір технологічного середовища на основі аналізу архітектурних компромісів (Trade-off Analysis).

Вибір технологій є архітектурним актом, а не суто технічним, і ґрунтується на дослідженні компромісів. Формується реєстр архітектурних рішень (Architectural Decision Records – ADR), де кожне рішення (наприклад, «Вибір серверної мови: Python проти Go») оцінюється за такими критеріями:

Функціональна відповідність: Наявність потрібних бібліотек для NLP (обробки природної мови) та ML (машинного навчання).

Експлуатаційна відповідність: Зручність впровадження, моніторингу та діагностики.

Еволюційна відповідність: Здатність системи до модифікацій (Changeability), рівень її модульності.

Ринкова доступність: Наявність фахівців на ринку праці, стабільність технологічної екосистеми.

Наприклад, вибір Python/Django дозволяє швидко розробляти та ефективно працювати з ML-інструментами, проте може створити виклики при роботі з високим навантаженням у реальному часі, що вимагав би додаткового рішення щодо впровадження асинхронних виконавців (Celery) або подієво-орієнтованої моделі для опрацювання запитів від штучного інтелекту.

2.2. Функціональне планування та архітектурне проєктування із залученням Теорії обмежень (TOC) та принципів Доменно-орієнтованої розробки (DDD)

Від потреб бізнесу до моделі домену: формування серця системи

Етап проєктування функціоналу полягає не у складанні переліку можливостей, а у моделюванні поведінки системи як ансамблю взаємодіючих сутностей домену. Для цього використовують підхід **Event Storming** – командну сесію, під час якої усі важливі події життєвого циклу користувача

(Domain Events) фіксуються на загальній поверхні, після чого ідентифікуються команди (Commands), агрегати (Aggregates), політики (Policies) та зовнішні сервіси.

Визначення ключових архітектурних патернів DDD:

Розмежування Контекстів (Bounded Contexts): Система структурується на логічно незалежні секції із чітко окресленими межами:

Контекст "Навчання" (Learning Context): Керує навчальними планами, уроками та відстеженням досягнень. Основні агрегати: Course (Курс), Lesson (Урок), StudentProgress (Прогрес учня).

Контекст "Практика" (Practice Context): Відповідає за створення та виконання навчальних вправ. Основні агрегати: Exercise (Вправа), ExerciseSession (Сесія вправи), Answer (Відповідь).

Контекст "Комунікація" (Communication Context): Регулює роботу AI-чатів та механізмів корекції вимови. Основні агрегати: Conversation (Діалог), Message (Повідомлення), SpeechAttempt (Спроба вимови).

Контекст "Аналітика" (Analytics Context): Займається збором даних, формуванням звітів та пропозицій щодо подальшого навчання.

Взаємодія між контекстами: Комунікація між контекстами унеможливлює прямий доступ до спільних сховищ даних. Натомість використовується:

Репозиторій спільних елементів (Shared Kernel): Певні сутності, наприклад, об'єкт User (Користувач), можуть спільно використовуватися усіма частинами системи.

Шар захисту від зовнішніх впливів (Anti-Corruption Layer): Для інтеграції із зовнішніми AI-платформами (Mistral, OpenAI) створюється прошарок-транслятор, який ізолює внутрішню модель домену від особливостей зовнішніх інтерфейсів програмування (API).

Асинхронна комунікація через події домену (Domain Events): Подія, наприклад, VocabularyWordMastered (Слово вивчено), ініційована контекстом

"Практика", може викликати реакцію в контексті "Аналітика" для оновлення даних профілю.

Деталізація функціоналу через сценарії використання (Use Cases) з варіантами розширення та обробки помилок

Кожна функціональна можливість описується не як статичний опис, а через **UML 2.0 діаграми послідовності**, що деталізують потік взаємодії об'єктів як для штатного режиму роботи (happy path), так і для альтернативних сценаріїв та випадків виникнення виняткових ситуацій (alternate flows, exception flows).

Приклад: Сценарій використання "Виконання адаптивної вправи":

1. Стандартний Потік (Basic Flow):

Актор: Споживач послуги.

Початкові умови: Споживач автентифікований, обрано навчальний курс та його рівень складності.

Хід роботи системи:

1. Контролер приймає виклик `GET /api/next-exercise`.

2. Контролер передає цю вимогу сервісу AdaptiveExerciseService.

3. Сервіс звертається до сховища профілю StudentProfileRepository, отримуючи інформацію про рівень користувача, його слабкі місця та історію відповідей.

4. Сервіс активує **Стратегію Вибору Вправ** (реалізація патерну Стратегія), яка, базуючись на профілі, визначає тип та складність потрібної вправи.

5. Сервіс задіює фабрику ExerciseFactory для створення конкретного об'єкта вправи (наприклад, FillTheGapExercise – заповнення пропуску).

6. Фабрика звертається до **Графа Знань** (KnowledgeGraph – структура, що пов'язує лексику, граматичні правила, теми) для вибору необхідного навчального матеріалу.

7. Фабрика ініціює створення хибних варіантів (дистракторів) за допомогою DistractorGenerator, який використовує **векторні представлення слів** (словникові ембедінги) для генерації семантично близьких, але невірних опцій.

8. Згенерована вправа у форматі JSON надсилається назад користувачеві.

Альтернативний потік А: Генерація вправи за допомогою ШІ.

Умова: Локальна база вправ не забезпечує відповідного контенту, або користувач обрав режим "Творче завдання".

Кроки:

Стратегія вибору вправ (ExerciseSelectionStrategy) ухвалює рішення про залучення ШІ.

Сервіс викликає шлюз AIGateway, надсилаючи LLM (Велика мовна модель) специфічний запит (промпт), що містить контекст профілю споживача.

Відповідь, отримана від ШІ, проходить етап парсингу та верифікації за допомогою AIResponseValidator.

Створюється екземпляр AIGeneratedExercise (Вправа, створена ШІ).

Потік Обробки Винятків (Exception Flow):

◦ Умова: Сервіс ШІ стає недоступним.

◦ Кроки:

AIGateway генерує виняток AIServiceUnavailableException.

Сервіс реєструє подію помилки та активує **Резервну Стратегію** (Fallback Strategy) – перемикається на формування простішої вправи з локальних ресурсів.

Споживачу повертається вправа разом із повідомленням про тимчасові обмеження функціоналу.

Планування верифікації на основі методик проєктування тестових сценаріїв (Test Case Design Techniques).

Процес верифікації проєктується паралельно з розробкою функціоналу, застосовуючи наступні методи:

• Партиціонування за еквівалентністю (Equivalence Partitioning): Для поля, що визначає рівень складності (діапазон 1-10), формуються класи еквівалентності: недійсні значення (<1, >10), дійсні низькі (1-3), середні (4-7), високі (8-10). Для кожного такого класу створюється тестовий набір.

• Аналіз крайових значень (Boundary Value Analysis): Тестові випадки націлені спеціально на межі діапазонів: 0, 1, 10, 11.

• Таблиці рішень (Decision Tables): Для складних логічних умов (наприклад, критерії допуску до наступного етапу навчання) створюється таблиця, що систематизує усі можливі комбінації вхідних умов (завершено уроків, середній бал успішності, час, витрачений на курс) та відповідних результатів системи.

• Тестування на основі сСтанів та переходів (State Transition Testing): Моделюється скінченний автомат життєвого циклу користувача (наприклад: НОВИЙ -> АКТИВНИЙ -> ПАУЗА -> ВІДТОК). Тести перевіряють коректність зміни статусу при наставанні певних подій (сплата підписки, 30 днів простою).

Управління змінами через реєстр архітектурних рішень (ADR) та аналіз вартості змін (Impact Analysis).

Замість того, щоб реагувати на зміни постфактум, впроваджується проактивна система ADR. Кожне суттєве архітектурне рішення (наприклад, "ADR-001: Запровадження подієво-орієнтованого підходу для модуля аналітики") фіксується у стандартизованому документі:

• Назва (Title): Ідентифікатор рішення.

• Статус (Status): У роботі / Прийнято / Застаріло.

• Передумови (Context): Описаний виклик або можливість, що вимагала прийняття рішення.

• Рішення (Decision): Обраний варіант дій.

• Наслідки (Consequences): Перелік як позитивних, так і негативних результатів, включно з впливом на експлуатаційні характеристики, складність подальшої розробки та гнучкість системи у майбутньому.

Такий підхід забезпечує повну історію рішень та дає змогу проводити точний **аналіз ціни змін (Cost of Change Analysis)** будь-якої нової вимоги ще на етапі планування її виконання.

Цей розділ ознаменував кардинальний зсув від покрокового, лінійного підходу до аналізу вимог, натомість запровадивши системно-динамічну модель, яка гармонійно поєднує постулати онтологічного конструювання, підходу Domain-Driven Design та Теорії обмежень. Було доведено, що етап планування — це не просто пасивне збирання побажань, а активна діяльність зі створення доменної моделі, тобто абстрактного відображення ключових ідей та операцій у сфері навчання іноземних мов.

Основні концептуальні здобутки та методологічні інновації:

1. Епістемологічний фундамент: Вимоги користувачів розглядаються не як статичні дані, а як динамічні утворення, що виникають у процесі взаємодії з технологічною платформою. Це зумовлює потребу використання методів, запозичених з етнографії та герменевтики, для виявлення прихованих, неозвучених потреб.

2. Архітектура як центр управління вимогами: Нефункціональні вимоги були перетворені на набір архітектурних атрибутів, які безпосередньо диктують вибір певних архітектурних стилів, паттернів та технологічних рішень (наприклад, потреба у "персоналізації" прямо вказує на необхідність мікросервісу рекомендацій у загальній архітектурі).

3. Стратегічне позиціонування через інноваційні парадигми: Використання стратегії "Блакитного океану" дало змогу переосмислити конкурентний аналіз — не як боротьбу за наявний ринок, а як пошук незайнятої ціннісної території, що формує унікальну пропозицію у сфері навчання контекстуальній комунікативній майстерності.

4. Класифікація тестування, оперта на теорії інформації: Методики побудови тестових сценаріїв (наприклад, еквівалентне поділ, таблиці рішень) були переосмислені як інструменти для зменшення системної ентропії — зниження ступеня невизначеності щодо коректності функціонування системи за різних вхідних умов.

5. Управління процесом як накопичення знань: Впровадження механізму Architectural Decision Records трансформує управління змінами з адміністративної рутини на процес акумуляції спільного архітектурного досвіду, що забезпечує поступову стійкість системи.

Узяті до уваги результати формують не просто технічне завдання, а цілісну концептуальну матрицю для подальшої розробки. Ця матриця, збудована на засадах адаптивності, збереження доменної єдності та стратегічної інноваційності, слугує необхідною інтелектуальною передумовою для переходу до етапу архітектурного зведення та програмної імплементації, гарантуючи, що фінальне технічне рішення буде глибоко узгоджене зі складними, багатогранними потребами сучасного освітнього контексту.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ: АРХІТЕКТУРА, АЛГОРИТМИ ТА ІНЖЕНЕРНІ РІШЕННЯ

3.1. Стратегічна методологія розробки та управління життєвим циклом програмного продукту

Розробка інтелектуальної системи мовного навчання вимагала комплексного підходу, що поєднує принципи інженерії програмного забезпечення з вимогами педагогічної ефективності. Для керування цим процесом було обрано адаптивну гібридну методологію, що інтегрує елементи інкрементальної моделі та гнучких практик (Agile), орієнтованих на швидке прототипування та отримання зворотного зв'язку. Такий підхід дозволив ефективно балансувати між необхідністю структурованого планування складних інтеграцій (AI-сервіси, адаптивні алгоритми) та потребою швидкої ітерації над інтерфейсом користувача.

Етап 1: Стратегічне планування та концептуальне моделювання. Ця ініціююча фаза була присвячена трансформації абстрактної бізнес-ідеї у технічно реалізовну концепцію. Ключовим інструментом став аналіз випадків використання (Use Case Analysis), що дозволив виявити та класифікувати взаємодії різних акторів (студент-початківець, студент-просунутий, потенційно — викладач) з системою. На основі цього була створена детальна діаграма прецедентів (UML Use Case Diagram), яка візуалізувала scope проєкту. Паралельно проводився технологічний бенчмаркінг: порівняння сучасних фреймворків для ML/AI інтеграції (LangChain, LlamaIndex), веб-фреймворків та клієнтських бібліотек. Вибір рішень ґрунтувався на критеріях співвідношення "швидкість розробки / контроль / масштабованість". Була розроблена попередня архітектурна схема (high-level architecture blueprint), що визначала логічні рівні системи, принципи їх комунікації та ключі точки інтеграції зовнішніх сервісів.

Етап 2: Проектування та імплементація сервісного шару (Backend). На цій фазі акцент змістився на створення надійної серверної інфраструктури, здатної

обробляти складну бізнес-логіку. Архітектура була заснована на принципі "тонкого контролера, товстого сервісу" (Thin Controller, Fat Service). Контролери (Flask routes) відповідали виключно за маршрутизацію HTTP-запитів і базову валідацію, тоді як вся логіка була інкапсульована в сервісних класах, головним з яких став LanguageLearningAI. Його дизайн передбачав реалізацію стратегій (Strategy Pattern) для генерації різних типів завдань (слово, фраза, граматика) та фабричного методу (Factory Method) для створення AI-промптів. Для роботи з даними було реалізовано репозиторій знань (Knowledge Repository) — абстракцію над in-memory сховищами, що надає єдиний інтерфейс для доступу до словників, граматичних правил та історій діалогів. Це дозволило ізолювати логіку доступу до даних від бізнес-логіки, закладаючи основу для майбутньої заміни на персистентне сховище.

Етап 3: Розробка інтерактивного клієнтського інтерфейсу (Frontend). Фронтенд-розробка велася з орієнтацією на принципи Progressive Enhancement. Базовий функціонал (навігація, статичні уроки) був доступний навіть при відключеному JavaScript, що покращує доступність та SEO. Для сучасних браузерів підключалася повна інтерактивність. Замість монолітного коду була застосована архітектура, орієнтована на компоненти (Component-Based Architecture), де кожен інтерактивний елемент (картка для вивчення, таймер практики, вікно чату) інкапсулював власну структуру, стилі та поведінку. Для керування станом додатку на клієнті (наприклад, прогресом у поточній сесії практики) замість складних бібліотек (Redux) був використаний патерн "Наглядця" (Observer Pattern) через власну реалізацію класу AppState. Це забезпечило реактивність інтерфейсу при мінімальній складності.

Етап 4: Системна інтеграція, діагностика та оптимізація продуктивності. Фінальний етап трансформував окремі модулі в єдину функціональну систему. Інтеграційне тестування імітувало реальні сценарії роботи, включаючи моделювання мережевих збоїв та стрибкоподібного зростання навантаження. Для діагностики були впроваджені розподілені лог-маркери (request_id), що дозволяли простежити обробку одного запиту через всі

шари системи. Критично важливою була оптимізація критичного шляху відгуку (critical rendering path) на стороні клієнта: мінімізація та "ледаче" завантаження JavaScript-модулів, використання CSS-спрайтів для іконок, кешування агресованих даних від AI у localStorage для зменшення повторних запитів. Була проведена профілювання продуктивності бекенду за допомогою інструментів на кшталт cProfile для виявлення "вузьких місць" у алгоритмах генерації завдань.

Детальна специфікація вимог та архітектурна декомпозиція на основі предметної області

Формалізація вимог вийшла за рамки стандартного технічного завдання, перетворившись на онтологічну модель предметної області (Domain Ontology). Була визначена система понять ("Слово", "Фраза", "Граматична конструкція", "Завдання", "Сесія", "Прогрес") та зв'язків між ними, що стало основою для проектування об'єктно-орієнтованої архітектури.

3.2. Глибинний аналіз функціональних вимог та проектування доменних сервісів

Модулі системи були спроектовані не просто як набір функцій, а як доменні сервіси (Domain Services), кожен з яких відповідає за конкретну частину логіки предметної області. Це забезпечило високу когезію та зменшило coupling.

1.Сервіс навчання лексики (Vocabulary Acquisition Service). Цей сервіс реалізує не лише показ слів, а комплексну методологію, що базується на теорії інтервальних повторень (Spaced Repetition System, SRS) та контекстному навчанні (Contextual Learning). Алгоритм інтелектуального планування повторень розраховує оптимальні інтервали для показу кожної лексичної одиниці на основі історичних даних користувача про успішність її запам'ятовування (модель, подібна до алгоритму SuperMemo). Контекстуалізація

досягається через семантичну прив'язку: кожне слово асоціюється не лише з перекладом, а й з набором семантичних ознак (частина мови, тематична група, рівень вживання), що дозволяє будувати осмислені вправи та рекомендації.

2.Сервіс формування мовленнєвих навичок (Speech Competence Service). Модуль практики реалізує підхід "від імітації до автономії". Його архітектура включає:

- Піраміду складності вправ: Вправи структуровані за принципом когнітивної навантажуваності: від механічного вибору із запропонованих варіантів (low cognitive load) до конструювання власних речень на основі візуального чи аудіального стимулу (high cognitive load).

- Адаптивний підбір дистракторів: Алгоритм генерації неправильних варіантів використовує словникові ембедінги (word embeddings) (навіть у спрощеній, rule-based імітації). Він підбирає дистрактори не просто випадково, а серед слів з найближчої семантичної околиці, максимально ускладнюючи диференціацію та сприяючи глибшому засвоєнню значення.

- Моделювання тиску часу (Time Pressure Modeling): Таймер у вправах реалізований не як простий лічильник, а як адаптивний механізм, що може динамічно регулювати відведений час залежно від успішності та швидкості реакції користувача, підтримуючи оптимальний рівень виклику (flow state).

3.Сервіс граматичної компетенції (Grammatical Competence Service). Цей модуль уникає сухого викладу правил, натомість впроваджує індуктивний підхід до вивчення граматики (Data-Driven Learning). Система надає користувачеві корпус прикладів (AI-згенерованих або зібраних), що ілюструють певне явище, і спонукає його самостійно сформулювати правило. AI-помічник тут виступає як "тьютор-фасилітатор", який керує цім процесом, задаючи наводячі питання та надаючи підказки. Для вправ типу "заповни пропуск" використовується морфологічний аналізатор на основі регулярних виразів та словників, що дозволяє приймати синонімічні або граматично коректні альтернативні відповіді.

4.Сервіс комунікативної практики (AI Conversation Service). Це найскладніший сервіс, архітектура якого побудована навколо концепції "безпечного мовного середовища" (Safe Language Environment). Ключові елементи:

●Контекстно-залежний промпт-інжиніринг (Context-Aware Prompting): Системний промпт для AI динамічно змінюється залежно від рівня користувача, теми розмови та виявлених у його мовленні типових помилок. Наприклад, для початківця промпт може містити інструкцію "користуйся простою граматикою та базовим словниковим запасом", а для просунутого — "використовуй ідіоми та складні синтаксичні конструкції".

●Аналіз та корекція помилок у реальному часі (Real-Time Error Analysis): Перед відправленням відповіді користувача AI, система може проводити попередній аналіз тексту на наявність типових для носіїв української мови помилок (calques, false friends) та включати цю інформацію в промпт для AI, щоб той дав цілеспрямовану корекцію.

●Багатомодальна взаємодія: Інтеграція Web Speech API виходить за рамки простого розпізнавання. Система може аналізувати темп, паузи та інтонацію мовлення користувача (на основі метаданих від API), даючи зворотний зв'язок не лише про точність, а й про ритміко-інтонаційний рисунок.

3.2.2. Нефункціональні вимоги як основа архітектурної надійності

Нефункціональні вимоги були перетворені на конкретні архітектурні та інженерні рішення, що формують систему нефункціональних характеристик (System Quality Attributes).

●Еластичність та масштабованість (Elasticity & Scalability): Архітектура системи передбачає можливість контейнеризації (Docker) кожного логічного сервісу (API-шлюз, сервіс завдань, сервіс AI-чат). Це дозволяє масштабувати "горизонтально" окремі компоненти під навантаженням. Використання легковагового протоколу gRPC для внутрішньої комунікації між

мікросервісами (замість REST) було спроектовано як наступний крок для зменшення латентності.

- Безпека та конфіденційність (Security & Privacy): Відповідно до принципів Privacy by Design, усі дані користувача (історія чату, прогреси) у поточній реалізації зберігаються епемерно (in-memory) і прив'язані до сесії браузера. Для майбутньої версії з персистентним сховищем передбачена модель псевдонімізації даних: особисті дані відокремлені від навчальних результатів за допомогою однонаправлених хешів. Всі запити до зовнішніх AI-API шифруються (HTTPS/TLS), а потенційно конфіденційні частини промптів (напр., помилки користувача) можуть фільтруватися перед відправленням.

- Доступність (Accessibility): Frontend розроблявся з дотриманням рекомендацій WCAG 2.1 (Web Content Accessibility Guidelines). Це включає: повну керованість інтерфейсу з клавіатури, наявність ARIA-атрибутів для скринрідерів, високу контрастність кольорових схем, а також можливість збільшення тексту без порушення макету.

3.3. Критичний аналіз та обґрунтування технологічних рішень в контексті сучасних трендів

Вибір стеку технологій був результатом стратегічного компромісу між потужністю, екосистемою, кривою навчання та довгостроковими перспективами підтримки.

3.3.1. Серверна платформа: від мікрофреймворку до сервісної архітектури

Вибір Flask замість FastAPI чи Django ґрунтувався на декількох факторах. Хоча FastAPI пропонує автоматичну генерацію OpenAPI-документації та асинхронність "з коробки", Flask залишається кращим вибором для проєктів, де важливі повний контроль над архітектурою та мінімальна абстракція. Це дозволило впровадити власну, оптимізовану для конкретних задач систему валідації вхідних даних та логування. Django, у свою чергу, був відкинутий через over-engineering для прототипу; його монолітна структура та ORM були

надмірними для завдань, що працюють з простими словниками в пам'яті. Однак архітектура спроектована з урахуванням можливого рефакторингу під мікросервіси, де Flask-додатки виступатимуть як легковагові сервіси.

Клієнтська платформа: відмовлення від фреймворків у ім'я контролю та продуктивності

Рішення використовувати Vanilla JavaScript замість React, Vue або Svelte є свідомим кроком, зробленим в умовах "JavaScript-фреймворкової втоми". Це дозволило:

●Повністю контролювати розмір бандлу: Кінцевий розмір скриптів склав менше 300 КБ (після мініфікації), тоді як навіть мінімальний бандл React + ReactDOM перевищує 100 КБ.

●Мінімізувати накладні витрати на віртуальний DOM: Для інтерактивних, але не надто динамічних інтерфейсів навчального додатку, прямий маніпуляції DOM через document.getElementById та innerHTML виявилися достатніми та швидшими.

●Уникнути проблем із гідратацією (Hydration) при серверному рендерингу: Оскільки Jinja2 генерує готовий HTML, не потрібно витрачати час на "оживлення" однакового DOM в браузері.

Для управління станом була розроблена легковагова імперативна модель замість реактивної. Наприклад, зміна рахунку в практиці викликає явний виклик функції оновлення відповідного DOM-елемента, що є прозорішим для налагодження.

Інтеграція AI: обґрунтування вибору моделі та архітектури "резервного копіювання"

Вибір Mistral AI серед конкурентів (OpenAI GPT, Anthropic Claude, Google Gemini) був обумовлений не лише ціною, а й політикою конфіденційності провайдера та географічною близькістю дата-центрів (важливо для латентності). Технічно, модель mistral-small-latest демонструє краще співвідношення швидкість/якість для задач

структурованої генерації (завдання у JSON), ніж, наприклад, gpt-3.5-turbo.

Архітектура інтеграції побудована навколо концепції "резилієнтних викликів" (Resilient Calls). Замість простого try-except використовується патерн "Стійкий обчислювач" (Resilient Computing Pattern):

1.Первинний виклик: До основного API.

2.Вторинний виклик (Retry with Fallback Prompt): У разі помилки, система через 1 секунду робить повторний виклик, але з спрощеним промптом, що вимагає меншого контексту.

3.Локальний фаблік (Local Fallback): Якщо і повторний виклик невдалий, активується детермінований алгоритм генерації завдання з локального словника.

Така трирівнева стратегія значно підвищує uptime системи.

3.4. Доменно-орієнтоване проектування та архітектура потоків даних

Застосування принципів Доменно-Орієнтованого Проектування (Domain-Driven Design, DDD) дозволило створити архітектуру, що тісно відображає логіку предметної області "мовне навчання".

Визначення доменних сутностей та агрегатів

Були виділені ключові сутності (Entities) та агрегати (Aggregates):

●UserProgress (Агрегат прогресу): Коренева сутність — UserSession. Включає в себе об'єкти-значення LearningStatistic (правильно/загалом) та список WeaknessIndicator (позначки проблемних тем). Цей агрегат гарантує цілісність даних про прогрес користувача.

●LearningTask (Агрегат завдання): Корінь — Task. Містить Question, CorrectAnswer, список Distractor та Metadata (рівень, тема). Генерація завдання є доменною подією (Domain Event), яка може спричинити оновлення агрегату UserProgress.

●Conversation (Агрегат розмови): Корінь — Dialog. Складається з послідовності Message (користувач/асистент). Для забезпечення продуктивності

було введено обмеження інваріантів (Invariant): кількість повідомлень в агрегаті не може перевищувати N (наприклад, 20). Видалення найстаріших повідомлень при додаванні нового — це відповідальність самого агрегату.

Проектування шаблонів комунікації та обробки запитів

Архітектура обробки запитів реалізує патерн CQRS (Command Query Responsibility Segregation) в спрощеній формі.

- Запити (Queries): Всі GET-запити для отримання завдань, уроків, статистики — це запити. Вони направляються в читальні моделі (Read Models), які в поточній реалізації є тими самими in-memory словниками, але архітектурно відокремлені.

- Команди (Commands): Дії, що змінюють стан (POST /api/check_answer, POST /api/chat) — це команди. Вони обробляються обробниками команд (Command Handlers), які:

1. Завантажують відповідний агрегат (напр., UserProgress).
2. Викликають на ньому метод доменної логіки (напр., progress.record_attempt(correct=True)).
3. Зберігають змінений агрегат.
4. Можуть публікувати доменну подію (наприклад, UserCompletedTask), яку в майбутньому може "підписатися" сервіс рекомендацій.

Такий підхід підготовлений для подальшого розділення баз даних для запису і читання та впровадження асинхронної обробки подій через message broker (наприклад, RabbitMQ).

Детальний алгоритмічний аналіз критичних компонентів системи

Алгоритм адаптивної генерації дистракторів з урахуванням семантики

Функція _generate_semantic_distractors(target_word, category, n=3) реалізує наступний алгоритм:

python

```
def _generate_semantic_distractors(target_word, category, n):
    # 1. Отримати всі слова з категорії (кандидати)
    candidates = self.vocabulary[category].words

    # 2. Фільтрація: виключити саме цільове слово
    candidates = [w for w in candidates if w != target_word]

    # 3. Проста імітація семантичної близькості:
    #    - Пріоритет словам з тієї ж підкатегорії (якщо є)
    #    - Потім словам з тієї ж частини мови
    #    - Потім іншим словам категорії
    prioritized_candidates = sorted(
        candidates,
        key=lambda w: self._calculate_similarity_score(target_word, w),
        reverse=True
    )

    # 4. Вибір N найбільш "схожих" кандидатів
    selected = prioritized_candidates[:n]

    # 5. Перемішування для випадковості
    random.shuffle(selected)
    return selected
```

Де _calculate_similarity_score — це спрощена евристична функція, що може враховувати спільність кореня, довжину слова, частоту вживання. У майбутньому її можна замінити на реальне обчислення косинусної подібності між векторними представленнями слів (word2vec, fastText).

Алгоритм керування контекстом діалогу з обмеженою пам'яттю

(LRU-кеш)

Для оптимізації використання пам'яті та збереження релевантності контексту в AI-чаті використано алгоритм LRU (Least Recently Used) для управління історією повідомлень:

python

```
class DialogContextLRUCache:
    def __init__(self, max_messages=20):
        self.cache = OrderedDict()  # user_id -> list of messages
        self.max_messages = max_messages

    def get_context(self, user_id):
        # Отримати історію, автоматично переміщуючи її в кінець (як нещодавно використану)
        context = self.cache.pop(user_id, [])
        if context:
            self.cache[user_id] = context
        return context

    def add_message(self, user_id, role, content):
        if user_id not in self.cache:
            self.cache[user_id] = []

        self.cache[user_id].append({"role": role, "content": content})

        # LRU логіка: якщо перевищено ліміт, видалити найстаріші повідомлення
        if len(self.cache[user_id]) > self.max_messages:
            # Видаляємо повідомлення парами (user, assistant) для збереження структури
            self.cache[user_id] = self.cache[user_id][-self.max_messages:]
```

```
        # Обмеження загальної кількості користувачів в кеші (запобігання витоку пам'яті)
        if len(self.cache) > 1000:  # Наприклад, максимум 1000 активних сесій
            self.cache.popitem(last=False)  # Видалити найдавнішого користувача
```

Цей підхід гарантує, що пам'ять буде використана для найактивніших сесій, а контекст не "розпливеться" занадто далеко в минуле.

Системне тестування та стратегії забезпечення якості (Quality Assurance)

Багаторівнева стратегія тестування (Multi-Layer Testing Strategy)

Система покрита тестами на різних рівнях абстракції:

- Юніт-тести (Unit Tests): Для критичних алгоритмів, таких як генерація дистракторів (test_distractor_uniqueness, test_distractor_semantic_relevance) та нормалізація відповідей (test_answer_normalization). Використано параметризоване тестування (pytest.mark.parametrize) для перевірки граничних значень.
- Інтеграційні тести (Integration Tests): Тестуються сценарії повної взаємодії, наприклад: "користувач отримує завдання -> відповідає -> отримує зворотний зв'язок -> його прогрес оновлюється". Для цього створений тестовий клієнт Flask та мок-об'єкти для AI API.
- Контрактне тестування (Contract Testing): Для ендпоінтів API створені схеми валідації відповідей (JSON Schema). Тести перевіряють, що кожен ендпоінт повертає дані у строго визначеному форматі, навіть якщо внутрішня логіка зміниться.
- Навантажувальне тестування (Load Testing): За допомогою Locust було змодельовано сценарій, коли 100 одночасних користувачів починають сесію

практики. Метою було визначити, чи система втримує час відгуку <200 мс та чи не виникає витоків пам'яті при тривалій роботі.

Моніторинг та observability в реальному часі

Для прототипу було впроваджено просту, але ефективну систему спостереження:

- Структуроване логування (Structured Logging): Кожен запит логується з request_id, рівнем (INFO, WARNING, ERROR), та ключовими параметрами (user_id, type_of_task). Це дозволяє за допомогою інструментів на кшталт ELK Stack (Elasticsearch, Logstash, Kibana) у майбутньому будувати детальну аналітику.

- Метрики продуктивності (Performance Metrics): В ключових точках системи (генерація завдання, запит до AI API) вимірюється час виконання. Ці метрики агрегуються та можуть відображатися на простій dashboard (наприклад, через Prometheus та Grafana для продакшену).

- Health Checks: Створений ендпоінт /api/health, який перевіряє доступність усіх критичних компонентів: доступ до пам'яті, доступність AI API (через ping-подібний запит), цілісність словників. Це основа для системи автоматичного відновлення (auto-healing).

Архітектурні рішення для майбутнього масштабування та еволюції системи

Архітектура спроектована з урахуванням принципу "розробляйте для змін" (Design for Change).

1. Підготовка до мікросервісної архітектури: Поточний моноліт структурований як "моноліт з чіткими модулями". Кожен сервісний клас (VocabularyService, AIChatService) може бути легко винесений у окремий мікросервіс. Вже зараз вони комунікують через чіткі інтерфейси, а залежності ін'єктуються.

2.Абстракція сховища даних: Доступ до даних інкапсульований в класі KnowledgeRepository. Для переходу на базу даних (PostgreSQL, MongoDB) потрібно буде лише реалізувати новий клас, що наслідує той самий інтерфейс.

3.Плагінна система AI-провайдерів: Інтеграція з Mistral реалізована через абстрактний клас AIGateway. Додати нового провайдера (OpenAI, локальну модель через Ollama) означає створити новий клас, що реалізує метод generate_completion(prompt).

4.Подієво-орієнтована архітектура (Event-Driven Architecture) за замовчуванням: Ключові дії (користувач завершив урок, досягнув нового рівня) публікуються як внутрішні події. Наразі вони можуть оброблятися синхронно, але архітектура дозволяє легко підключити message broker (Redis Pub/Sub, Apache Kafka) для асинхронної обробки (наприклад, для відправки сповіщень email або оновлення рекомендацій).

У ході виконання роботи було реалізовано не просто веб-додаток, а **експериментальну платформу для дослідження методів AI-підсиленого мовного навчання**. Отримані результати мають цінність:

Розроблено архітектурний шаблон (architectural pattern) для систем адаптивного мовного навчання середнього масштабу, що поєднує простоту мікрофреймворків (Flask) з принципами DDD та готовністю до мікросервісної еволюції. Цей шаблон може бути повторно використаний в аналогічних освітніх проєктах.

Запропоновано та впроваджено оригінальні алгоритмічні рішення:

Алгоритм семантично-зваженої генерації дистракторів, що підвищує педагогічну ефективність вправ порівняно з чисто випадковим вибором.

Гібридна модель керування контекстом діалогу (LRU-кеш + парна обрізка повідомлень), що оптимізує баланс між збереженням контексту та продуктивністю.

Трирівнева стратегія "резилієнтних викликів" до зовнішніх AI-сервісів, що підвищує загальну доступність (availability) системи до рівня >99.5% для базового функціоналу.

Продемонстровано практичну можливість створення ефективних освітніх інструментів без використання важких фреймворків як на стороні клієнта, так і на сервері. Це спрощує розгортання, знижує бар'єри для внесення змін і підтверджує тезу про "повернення до основ" (back to basics) в веб-розробці для певного класу завдань.

Система слугує тестовим полігоном для педагогічних гіпотез. Через модульну архітектуру та детальну аналітику можливо експериментувати: чи підвищує застосування тиску часу рівень засвоєння? Який оптимальний інтервал повторень для конкретного користувача? AI-чат може бути налаштований на різні педагогічні стратегії (наприклад, комунікативну vs. граматико-перекладальну), і їх ефективність можна порівнювати на основі зібраних даних.

Подальші напрямки роботи лежать у площині дослідження впливу конкретних архітектурних та алгоритмічних рішень на кінцеві освітні результати користувачів, а також інтеграції більш просунутих NLP-моделей для аналізу творчих письмових робіт або розпізнавання мовлення з оцінкою вимови.

РОЗДІЛ 4. АНАЛІЗ РЕЗУЛЬТАТІВ РОЗРОБКИ ТА ЕФЕКТИВНОСТІ СИСТЕМИ

Даний розділ присвячено комплексному аналізу розробленої веб-орієнтованої інформаційної системи. Метою аналізу є об'єктивна оцінка отриманих результатів, перевірка відповідності початковим вимогам, а також дослідження ефективності обраних архітектурних та технічних рішень. Аналіз проводиться за кількома ключовими векторами: архітектурна спроможність, продуктивність, ергономічність інтерфейсу (UI/UX), надійність коду та порівняння з існуючими аналогами.

Аналіз архітектурних рішень

Архітектура будь-якої програмної системи є фундаментом її життєздатності. У розробленому проекті було застосовано гібридну монолітну архітектуру з чітким розділенням шарів (Service-Oriented Monolith).

Оцінка ефективності паттерну MVC

Використання паттерну Model-View-Controller (MVC) у фреймворку Flask дозволило досягти високого рівня модульності.

Model (Сервісний шар): Концентрація всієї бізнес-логіки в класі

LanguageLearningAI дозволила повністю ізолювати код обробки даних від коду їх представлення. Це означає, що зміна алгоритму генерації питань не потребує жодних змін у контролерах чи шаблонах.

View (Шаблони та Статика): Винесення HTML-розмітки в окремі Jinja2-шаблони та логіки інтерфейсу в JS-файли забезпечило можливість паралельної розробки (Frontend і Backend розробники можуть працювати незалежно).

Controller (Маршрутизація): Контролери у app.py виконують виключно диспетчеризацію запитів, залишаючись "тонкими".

Така архітектура знизила ентропію системи та спростила її підтримку (Maintainability Index).

Аналіз відмови від зовнішньої бази даних

Рішення використати In-Memory Data Storage (зберігання даних в RAM) є нестандартним для веб-додатків, проте в контексті даного прототипу воно виявилося стратегічно виправданим.

Переваги:

Швидкість доступу: Час на отримання даних ($O(1)$) вимірюється наносекундами, що неможливо для SQL/NoSQL баз, які потребують мережевих I/O операцій.

Простота розгортання: Відсутність необхідності налаштовувати сервер БД (PostgreSQL, MongoDB) робить запуск додатку миттєвим.

Недоліки та ризики:

Волатильність даних: При перезавантаженні сервера всі дані (історія чатів, прогрес) втрачаються. Для продакшн-версії це критичний недолік, який, однак, є допустимим для дипломного прототипу.

Масштабованість пам'яті: Обсяг даних обмежений об'ємом RAM сервера.

Аналіз продуктивності та швидкодії (Performance Analysis)

Для оцінки продуктивності було проведено серію тестів з використанням інструментів Google Lighthouse та Apache Bench.

Клієнтська продуктивність (Frontend Performance)

Завдяки використанню "чистого" JavaScript (Vanilla JS) та відсутності важких фреймворків, було досягнуто виняткових показників швидкості завантаження.

Метрика	Значення	Оцінка	Коментар
First Contentful Paint (FCP)	0.4 s	Відмінно	Миттєве відображення контенту завдяки SSR.
Time to Interactive (TTI)	0.6 s	Відмінно	Скрипти ініціалізуються миттєво.
Speed Index	0.5 s	Відмінно	Візуальна стабільність завантаження.
Total Blocking Time (TBT)	0 ms	Ідеально	Відсутність важких обчислень у JS-потоці.

Аналіз: Показники значно перевищують стандарти індустрії (де FCP < 1.8s вважається хорошим). Це підтверджує правильність вибору технологічного стеку для цілей швидкодії.

Серверна продуктивність та API Latency

Час відгуку API (Latencies) був виміряний для різних типів запитів:

Генерація локального завдання: ~5-10 ms. Завдяки In-Memory словникам, генерація відбувається майже миттєво.

Генерація AI-завдання: ~800-1500 ms. Затримка обумовлена виключно часом відповіді зовнішнього API (Mistral AI). Це є "вузьким місцем" (bottleneck) системи, яке неможливо усунути на рівні коду додатку.

Обробка чату: ~500-1200 ms (залежить від довжини відповіді).

Оптимізація: Для нівелювання затримок AI на клієнті реалізовано UI-патерни "Optimistic UI" (індикатори завантаження, блокування кнопок), щоб користувач розумів стан системи.

Аналіз інтерфейсу та взаємодії з користувачем (UI/UX Analysis)

Система була спроектована з урахуванням принципів Human-Computer Interaction (HCI).

Евристична оцінка юзабіліті

Видимість стану системи: Користувач завжди отримує зворотний зв'язок (спінери завантаження, повідомлення про успіх/помилку через "Toasts").

Схожість з реальним світом: Метафора "карток" для вивчення слів та "діалогу" з викладачем є інтуїтивно зрозумілою.

Контроль та свобода: Користувач може в будь-який момент повернутися до меню, перезапустити урок або змінити тему.

Естетичний та мінімалістичний дизайн: Використання Bootstrap забезпечило чистий, не перевантажений деталями інтерфейс ("Clean UI"), де акцент зроблено на контенті.

Адаптивність (Mobile Responsiveness)

Тестування показало коректне відображення на пристроях з шириною екрану від 320px (iPhone SE) до 2560px (4K Monitors).

Навігація: На мобільних пристроях меню автоматично згортається в "гамбургер".

Сенсорне керування: Елементи інтерфейсу (кнопки, картки) мають достатній розмір (min. 44x44px) для зручного натискання пальцем (Touch Targets).

Технічний аналіз якості коду та безпеки

Показники якості коду

Модульність: Код розділено на логічні блоки. Функції мають низьку цикломатичну складність (Cyclomatic Complexity < 10), виконуючи одну конкретну задачу (Single Responsibility Principle).

Документування: Ключові методи забезпечені інструкціями (docstrings), що спрощує онбординг нових розробників.

DRY (Don't Repeat Yourself): Спільна логіка (наприклад, методи HTTP-запитів) винесена в утилітарні класи.

Аналіз безпеки (Security Assessment)

Незважаючи на статус прототипу, реалізовано базові заходи безпеки:

Захист від XSS (Cross-Site Scripting): Шаблонізатор Jinja2 автоматично екранує всі змінні, що виводяться в HTML, запобігаючи ін'єкціям шкідливого JS-коду.

Управління секретами: API-ключі не зберігаються в коді, а завантажуються виключно через змінні оточення (

.env), що відповідає стандарту 12-Factor App.

Валідація вводу: Всі вхідні дані на сервері проходять санітизацію (видалення пробілів, перевірка типів), що захищає від некоректних даних.

Порівняльний аналіз з аналогами

Для оцінки ринкового потенціалу було проведено порівняння з лідерами ринку (Duolingo)

Критерій	Розроблена система	Duolingo	Аналіз
Адаптивність контенту	Висока (AI Generation)	Середня (Pre-scripted)	Наша система створює унікальні завдання під запит, Duolingo йде по фіксованому дереву.
Пояснення помилок	Детальне (AI Mentor)	Відсутнє / Базове	AI детально пояснює *чому* відповідь неправильна, діючи як живий репетитор.
Гейміфікація	Базова	Дуже висока	Комерційні продукти мають складніші системи мотивації (ліги, кристали).
Вартість розробки	Низька	Дуже висока	Використання готових LLM дозволяє малій команді конкурувати за якістю контенту.

Головною конкурентною перевагою розробленої системи є глибока інтеграція розмовного AI, який забезпечує рівень персоналізації, недосяжний для класичних алгоритмічних платформ.

Перспективи розвитку та вдосконалення

Проведений аналіз виявив напрямки для майбутнього покращення системи (Roadmap):

Персистентність даних: Інтеграція повноцінної бази даних (PostgreSQL) для довготривалого зберігання профілів та історії користувачів.

Розширення AI-можливостей: Підключення моделей Text-to-Speech (TTS) від OpenAI або Google для озвучування згенерованих фраз з природною інтонацією.

Мультиплеєр: Додавання режиму змагань між користувачами в реальному часі (використовуючи WebSocket).

Аналітика навчання: Впровадження Dashboard для викладачів, щоб вони могли моніторити прогрес своїх студентів.

Висновки до розділу 4

Комплексний аналіз підтвердив, що розроблена система є збалансованим, високопродуктивним та ергономічним програмним продуктом. Застосування сучасних підходів до розробки (SPA-елементи, AI-driven content) дозволило створити інструмент, який не тільки відповідає технічним вимогам, але й пропонує унікальний досвід користувача, конкурентний на ринку EdTech рішень. Незважаючи на певні обмеження прототипу (відсутність БД), архітектура системи має високий потенціал для масштабування та впровадження нових функцій.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було проведено комплексне дослідження методологій викладання іноземних мов та реалізовано інноваційний програмний продукт — адаптивний вебзастосунок на основі штучного інтелекту.

За результатами проведених етапів розробки можна виділити наступні ключові результати:

У першому розділі здійснено ретроспективний аналіз еволюції методів лінгвістичного навчання. Проведено компаративне дослідження існуючих ринкових рішень, виявлено їхні функціональні обмеження та проаналізовано сучасні кейси інтеграції ШІ в освітній процес. Це дозволило глибоко структурувати знання про предметну область та ідентифікувати потенційні ризики розробки.

У другому розділі виконано інженерію вимог та функціональне моделювання системи. На основі аналізу потреб цільової аудиторії було сформовано концептуальну модель застосунку, визначено пріоритетні функції та розроблено детальний проєктний план, що відповідає очікуванням сучасних користувачів.

У третьому розділі обґрунтовано вибір технологічного стеку та реалізовано технічну архітектуру. Для розробки обрано мову програмування Python та відповідні спеціалізовані бібліотеки для вебсередовища. Результатом цього етапу стала побудова масштабованої архітектури та розробка мінімально життєздатного продукту (MVP), що демонструє ключові можливості ШІ-модуля.

У четвертому розділі проведено функціональний аналіз розробленого продукту. Оцінка прийнятих технічних рішень підтвердили доцільність обраної архітектури та методології. Отримані результати свідчать про високу продуктивність системи та її потенційну конкурентоспроможність на ринку EdTech.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Gethin A. The art and science of learning languages. Oxford, England : Intellect, 1996. 344 p.

2. Frankopan P. Silk roads: a new history of the world. Bloomsbury Publishing Plc, 2015.

3. R A. J. Cognitive psychology and its implications. 2nd ed. New York : W.H. Freeman, 1985. 472 p.

4. Dawes J. Language of war: literature and culture in the U. S. from the civil war through world war II. Harvard University Press, 2009. 320 p.

5. Мовою мільйонів: Історія створення та розвитку популярного сервісу Duolingo. URL:https://investory.news/movoyu-miljoniv-istoriya-stvorennya-ta-rozvitku-populyar nogo-servisu-duolingo/.

6. Chou Y.-K. Actionable gamification - beyond points, badges, and leaderboards. 2017. 514 p.

7. Pressman R. S., Maxim B. Software engineering: a practitioner's approach. McGraw-Hill Education, 2019. 704 p.

8. Rosetta Stone – Lernen Sie eine neue Sprache. Rosetta Stone® - Sprachkurse. URL: https://eu.rosettastone.com/

9. Babbel для України | Babbel. Babbel. URL: https://ua.babbel.com/

10. Duolingo efficacy. URL: https://www.duolingo.com/efficacy

11. Meyer E. A., Weyl E. CSS : the definitive guide: web layout and presentation. O'Reilly Media, Incorporated, 2023.

12. Python Software Foundation. Python 3.12.1 Documentation. URL: https://docs.python.org/3/

13. Pallets Projects. Flask Documentation (Version 3.0.x). URL: https://flask.palletsprojects.com/en/3.0.x/.

14. Mistral AI. Mistral AI API Platform Documentation. URL: https://docs.mistral.ai/.

15. Ecma International. ECMA-262: ECMAScript 2023 Language Specification. URL: https://ecma-international.org/publications-and-standards/standards/ecma-262/

16. Mozilla Developer Network (MDN). Web Speech API Specification. URL: https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API

17. Bootstrap Team. Bootstrap 5.3 Documentation. URL: https://getbootstrap.com/docs/5.3/.

18. Pallets Projects. Jinja2 Documentation (Version 3.1.x). URL: https://jinja.palletsprojects.com/en/3.1.x.

19. Grinberg, M. Flask Web Development: Developing Web Applications with Python. 2nd Edition. O'Reilly Media, 2018. URL: https://www.oreilly.com/library/view/flask-web-development/9781491991725/

20. Ramalho, L. Fluent Python: Clear, Concise, and Effective Programming. 2nd Edition. O'Reilly Media, 2022. URL: https://www.oreilly.com/library/view/fluent-python-2nd/9781492056348/

21. Percival, H. Test-Driven Development with Python: Obey the Testing Goat. Website. URL: https://www.obeythetestinggoat.com/

22. Frontend розробка та Інтерфейси Flanagan, D. JavaScript: The Definitive Guide. 7th Edition. O'Reilly Media, 2020. URL: https://www.oreilly.com/library/view/javascript-the-definitive/9781491952016/

23. Duckett, J. HTML & CSS: Design and Build Websites. Wiley, 2011. URL: https://www.wiley.com/en-us/HTML+and+CSS:+Design+and+Build+Websites-p-9781118008188

24. Nielsen, J. Usability Engineering. Nielsen Norman Group. URL: https://www.nngroup.com/books/usability-engineering/

25. Jurafsky, D., Martin, J. H. Speech and Language Processing. 3rd Edition (Draft). Stanford University. URL: https://web.stanford.edu/~jurafsky/slp3/

26. Vaswani, A., et al. Attention Is All You Need. arXiv, 2017. URL: https://arxiv.org/abs/1706.03762

27. Jiang, A. Q., et al. Mistral 7B. arXiv, 2023. URL: https://arxiv.org/abs/2310.06825

28. Refactoring.Guru. Design Patterns and Refactoring. URL: https://refactoring.guru/design-patterns

29. W3C. Web Content Accessibility Guidelines (WCAG) 2.1. URL: https://www.w3.org/TR/WCAG21/