НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

УДК 004.9:811.111

«ПОГОДЖЕНО»
Декан факультету
інформаційних технологій
Глазунова О.Г., д.п.н., професор_
_____
____ _____ 202_ р.

«ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ»
Завідувач кафедри комп'ютерних наук
Голуб Б.Л., к.т.н., старший викладач
_____
____ _____ 202_ р.

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему __Програмне забезпечення системи навчання англійської мови
_____

Спеціальність _____121 - Інженерія програмного забезпечення_____
(код і назва)

Освітня програма ____Програмне забезпечення інформаційних систем__
(назва)

Орієнтація освітньої програми _____освітньо-професійна_____
(освітньо-професійна або освітньо-наукова)

Гарант освітньої програми
___к.фіз.-мат.н., доцент_____ _____ ___Кириченко В.В.____
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник магістерської кваліфікаційної роботи
_____ст. викладач_____ _____ ____Бородкін Г.О.___
(науковий ступінь та вчене звання) (підпис) (ПІБ)
_____к.т.н., доцент_____ _____ ____Бородкіна І.Л.___
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ _____Кононенко Р.П._____
(підпис) (ПІБ студента)

КИЇВ-2023

НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет (ННІ) _____інформаційних технологій_____

ЗАТВЕРДЖУЮ
Завідувач кафедри_____
___комп'ютерних наук_____
___к.т.н., доцент_____ _____ Голуб Б. Л._
(науковий ступінь, вчене звання) (підпис) (ПІБ)
"____" _____року

З А В Д А Н Н Я
ДО ВИКОНАННЯ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ СТУДЕНТУ

_____Кононенку Ростиславу Петровичу_____
(прізвище, ім'я, по батькові)

Спеціальність _____121 - Інженерія програмного забезпечення_____
(код і назва)

Освітня програма ____Програмне забезпечення інформаційних систем_____
(назва)

Орієнтація освітньої програми_____освітньо-професійна_____
(освітньо-професійна або освітньо-наукова)

Тема магістерської кваліфікаційної роботи Програмне забезпечення системи навчання англійської мови
затверджена наказом ректора НУБіП України від "29" листопада 2021р. №_1170

Термін подання завершеної роботи на кафедру "08" листопада 2023р.
(рік, місяць, число)

Вихідні дані до магістерської кваліфікаційної роботи _____
1. Дані із мобільного додатку_____
Перелік питань, що підлягають дослідженню:
1. Використання OLAP та Data Mining технологій для аналізу_____
2. Моделювання та створення мобільного додатку_____
Перелік графічного матеріалу (за потреби) _____
_____

Дата видачі завдання "_____"_____ 20____ р.

Керівник магістерської кваліфікаційної роботи _____ __Бородкін Г.О._

Консультант _____ __Бородкіна І.Л._
(підпис) (прізвище та ініціали)

Завдання прийняв до виконання _____ _____Кононенко Р.П._____
(підпис) (прізвище та ініціали студента)

Зміст

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – інтерфейс програмування застосунків

DM – Data Mining

ERD – Essence Relation Diagramming

ID – ідентифікатор для бази даних

JS – JavaScript

JWT – JSON Web Token

OLAP – Online analytical processing

SSAS – SQL Server Analysis Services

UML – Unified Modeling Language

БД – база даних

ІТ – інформаційні технології

СУБД – система управління базою даних

ШІ – штучний інтелект

ВСТУП

У сучасному взаємопов'язаному та глобалізованому світі вміння ефективно спілкуватися англійською мовою стало безцінною навичкою. Англійська мова стала лінгва франка міжнародного бізнесу, дипломатії, освіти та культури, долаючи кордони та об'єднуючи людей з різним мовним походженням. Як наслідок, різко зріс попит на ефективні та доступні ресурси для вивчення англійської мови, що породило динамічну галузь програмного забезпечення для вивчення англійської мови, яка постійно розвивається. У цій дисертації досліджується багатогранна сфера програмного забезпечення для вивчення англійської мови, заглиблюючись у його походження, розвиток та вплив на засвоєння мови. Дослідження має на меті пролити світло на революційні трансформації у сфері вивчення мови, вивчаючи роль, яку відіграли технології у формуванні способу взаємодії людей з англійською мовою.

Інтеграція технологій у вивчення мови відкрила нову еру педагогічних можливостей, кидаючи виклик традиційним методам і відкриваючи двері для інноваційних підходів, які задовольняють широкий спектр учнів. Ця дисертація має на меті розплутати складний гобелен програмного забезпечення для вивчення англійської мови, проаналізувавши безліч інструментів, платформ і додатків, які з'явилися у відповідь на глобальний попит на знання англійської мови. Таким чином, ми надаємо всебічну картину сучасного ландшафту програмного забезпечення для вивчення англійської мови, дослідимо теорії та методології, що лежать в основі його розробки, та оцінимо його ефективність у сприянні вивченню мови.

Крім того, дослідження не лише заглиблюється в технічні аспекти розробки програмного забезпечення, але й проаналізує когнітивні та педагогічні аспекти, беручи до уваги психологічні та освітні теорії, що лежать в основі розробки та впровадження програмного забезпечення. Завдяки такому комплексному аналізу дисертація має на меті запропонувати погляд на майбутнє вивчення англійської мови, де програмне забезпечення продовжує адаптуватися,

розвиватися та розширювати свої можливості для задоволення різноманітних потреб учнів у всьому світі.

В епоху цифрових технологій та швидкого розвитку програмного забезпечення для вивчення англійської мови стає необхідним дослідити його вплив на мовну освіту, окремих учнів та суспільство в цілому. Ця стаття розпочинає подорож динамічним і трансформаційним світом програмного забезпечення для вивчення англійської мови, намагаючись розгадати його минуле, зрозуміти його сьогодення і передбачити його майбутнє в невпинному прагненні зробити вивчення англійської мови доступнішим, ефективнішим і приємнішим для учнів по всьому світу.

Об’єктом дослідження виступає програмне забезпечення системи навчання англійської мови.

Предмет – оцінка, аналіз та розробка програмного забезпечення для покращення ефективності процесу навчання англійської мови.

Метою цього дослідження є вивчення та розробка інноваційних компонентів для програмного забезпечення для вивчення англійської мови, які мають потенціал для покращення досвіду вивчення мови для користувачів. Конкретні цілі цього дослідження такі:

1. Розробка системи стеження:
    a. Створити комплексну систему відстеження в програмному забезпеченні для вивчення англійської мови, щоб систематично контролювати та записувати прогрес користувачів.
    b. Впроваджувати механізми зворотного зв’язку, які дозволяють користувачам оцінювати свій навчальний шлях і отримувати цінну інформацію від програмного забезпечення.
2. Розробка адаптивного алгоритму:
    a. Розробити та інтегрувати адаптивний первинний алгоритм для створення персоналізованих планів навчання, адаптованих до окремих користувачів.

   b. Використовувати підходи, керовані даними, для коригування темпу навчання, вмісту та вправ, забезпечуючи оптимальне та індивідуальне навчання для кожного користувача.
3. Впровадження машинного навчання та штучного інтелекту:
   a. Використати потужність технологій машинного навчання та штучного інтелекту для підвищення загальної ефективності та інноваційної спроможності освітніх систем.
   b. Використати ці технології для аналізу даних користувача, оптимізації доставки контенту та надання інтелектуальних рекомендацій, тим самим покращуючи процес навчання.
4. Розробка ефективних алгоритмів розпізнавання та перекладу:
   a. Створюйте удосконалені алгоритми розпізнавання та перекладу, щоб полегшити створення симуляторів та навчальних ігор для вивчення англійської мови.
   b. Ці алгоритми підвищать точність та інформативність мовних дій, дозволяючи користувачам краще розуміти англомовний контент.
5. Створення віртуального вчителя для персоналізованого навчання:
   a. Розробити віртуального вчителя в програмному забезпеченні, щоб надавати індивідуальне керівництво, підтримку та зворотний зв’язок окремим учням.
   b. Ця функція надасть користувачам доступ до індивідуального досвіду навчання, який відповідає їхнім унікальним потребам і здібностям, таким чином сприяючи прискореному засвоєнню мови.

Досягнувши цих цілей дослідження, дослідження прагне зробити свій внесок у постійну еволюцію програмного забезпечення для вивчення англійської мови, гарантуючи, що воно стане більш ефективним, адаптивним і привабливим інструментом для тих, хто вивчає мову. Зрештою, мета полягає в тому, щоб дати користувачам змогу досягти суттєвого прогресу у своєму володінні англійською мовою, пропонуючи більш персоналізований та ефективний досвід навчання.

Пояснювальний документ складається з чотирьох основних сегментів, заключних зауважень і бібліографії використаних посилань. Початковий сегмент охоплює комплексне дослідження системи, окреслюючи особливості предметної області та цілі програмного забезпечення. Наступний розділ ілюструє побудову об'єктно-орієнтованих моделей. Третій розділ пояснює розвиток системи у дві ключові фази, а саме, розробку інформації та програмного забезпечення. Четвертий розділ охоплює результати, що стосуються інтелектуальних конструкцій, і оцінку їхньої ефективності на практиці.

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Предметна область, що розглядається, обертається навколо багатогранної сфери програмного забезпечення для вивчення англійської мови. Це охоплює різноманітні цифрові інструменти, додатки та платформи, призначені для сприяння засвоєнню англійської мови та оволодінню нею. Предметна область заглиблюється в тонкощі цих програмних систем, з'ясовуючи їхні особливості, функції та цілі. У ньому розглядаються методи, теорії та технології, що лежать в основі розробки такого програмного забезпечення, з метою забезпечення повного розуміння того, як воно задовольняє різноманітні потреби тих, хто вивчає мову. Предметна область далі досліджує еволюцію цих програмних рішень, описуючи їх історичний розвиток і оцінюючи їх сучасне значення в мовній освіті. Загалом, він охоплює багатий ландшафт технологічних ресурсів для вивчення мови, які відіграють ключову роль у підвищенні рівня володіння англійською мовою.

На додаток до вищезгаданого, предметна область також глибоко вкорінена на перетині педагогіки, когнітивної науки та інформаційних технологій. Він розглядає не лише цифрові архітектури, на яких побудовані ці інструменти, але й педагогічну філософію, яка інформує про їхній дизайн та функціональність.

Розуміння теми вимагає глибокого занурення в те, як ці програмні інструменти відтворюють, покращують або навіть переосмислюють традиційний досвід у класі. Він розглядає, як структуровані уроки, як надається зворотний зв'язок і як учні взаємодіють у цьому віртуальному просторі. Такі ключові поняття, як гейміфікація, адаптивне навчання та персоналізовані шляхи, стають помітними тенденціями в цій галузі, оскільки розробники прагнуть використовувати можливості технологій, щоб зробити вивчення мови більш привабливим, адаптованим та ефективним.

Крім того, предметна область оцінює проблеми та обмеження програмного забезпечення для вивчення англійської мови. Незважаючи на величезні можливості, які пропонують ці інструменти, вони не позбавлені підводних каменів. Питання, що стосуються доступності, потенціалу деперсоналізації досвіду навчання та ефективності програмного навчання порівняно з традиційними методами, є гарячими темами в цій галузі. Ця область також досліджує, як культурні нюанси та відмінності можуть бути вбудовані або не помічені в такому програмному забезпеченні, а також наслідки цих рішень.

Крім того, у міру того, як світ стає все більш взаємопов'язаним, предметна область відображає наслідки глобальної аудиторії. Він розмірковує над тим, як програмне забезпечення може бути розроблено таким, щоб бути інклюзивним, обслуговувати учнів із різним мовним походженням, і що означає вивчати англійську мову в цифрову глобалізованому епоху.

У сукупності предметна область надає всебічний огляд вивчення англійської мови в епоху цифрових технологій, яка характеризується поєднанням технологічних досягнень, педагогічних інновацій і потреб, що постійно розвиваються у всьому світі.

1.2 Аналіз існуючих рішень

Цифрова ера стала свідком безлічі рішень для вивчення англійської мови з програмними платформами, які обслуговують різні аспекти вивчення мови. Аналіз цих існуючих рішень дає змогу зрозуміти їхні сильні сторони, проблеми та унікальні функції, допомагаючи потенційним користувачам і розробникам визначити сфери інновацій та вдосконалення.

1. Платформи для вивчення мов:
    a. Приклади: Duolingo [1], Rosetta Stone [2], Babbel [3], Memrise [4].
    b. Сильні сторони: елементи гейміфікації, структуровані плани уроків і зручні інтерфейси роблять ці платформи привабливими для учнів. Деякі пропонують реальні контекстуальні уроки, які допомагають у

практичному використанні мови.

c. Проблеми: часто їм не вистачає глибини в певних лінгвістичних областях, вони можуть не повністю враховувати культурні нюанси або можуть бути надто загальними, не допускаючи персоналізованих шляхів навчання.

2. Інтерактивні електронні книги та онлайн-курси:
    a. Приклади: Oxford Online English [5].
    b. Сильні сторони: комплексна навчальна програма, професійне озвучення та мультимедійні елементи забезпечують цілісний досвід навчання. Гнучкість темпу дозволяє учням просуватися з власною швидкістю.
    c. Проблеми: обмежена інтерактивність у деяких курсах, залежність від самомотивації, а іноді й відсутність зворотного зв’язку чи можливостей оцінювання.
3. Віртуальні класи та платформи викладачів:
    a. Приклади: VIPKid [6], iTalki [7].
    b. Сильні сторони: пропонують реальну людську взаємодію, надаючи персоналізований зворотний зв’язок. Вони задовольняють конкретні потреби, такі як зменшення акценту або ділова англійська.
    c. Проблеми: вартість може бути непомірно високою для деяких користувачів. Ефективність багато в чому залежить від якості окремих репетиторів.
4. Спільноти мовного обміну:
    a. Приклади: Tandem [8].
    b. Сильні сторони: Сприяти автентичному мовному обміну, покращуючи навички спілкування. Бонусом є культурний обмін.
    c. Проблеми: якість взаємодії може бути непостійною. Деякі користувачі можуть зіткнутися з труднощами під час пошуку відданих партнерів.
5. Інструменти граматики та письма:

a. Приклади: Grammarly [9].
b. Сильні сторони: надайте миттєвий відгук про письмову англійську, допомагаючи користувачам удосконалити свої навички письма.
c. Проблеми: може не запропонувати вичерпних пояснень певних помилок, а надмірна довіра може перешкоджати природному процесу навчання.

6. Рішення доповненої реальності (AR) і віртуальної реальності (VR):
   a. Приклади: MondlyAR [10].
   b. Сильні сторони: ефект занурення робить навчання контекстним і таким, що запам’ятовується. Пропонуйте реалістичне моделювання реальних ситуацій.
   c. Проблеми: висока вартість обладнання VR може бути перешкодою. У порівнянні з більш традиційними платформами доступний обмежений вміст.

1.3 Постановка завдань дослідження

Під час розробки розширеного програмного забезпечення для вивчення англійської мови чіткі цілі дослідження є обов’язковими. Ці цілі визначають напрямок проекту, гарантуючи, що програмне забезпечення відповідає очікуванням передбачуваних користувачів і забезпечує збагачення досвіду. У цій главі буде розглянуто специфіку цих цілей, класифікованих за програмним забезпеченням, надійністю системи, технічними засобами та вимогами до інтерфейсу користувача.

1.3.1 Вимоги до програмного забезпечення. Програма повинна забезпечувати можливість виконання наступних функцій:

1. Реєстрація облікових записів:
   a. Розробити безперебійний і безпечний процес реєстрації, що дозволить користувачам створювати індивідуальні облікові записи.

   b. Включити кілька варіантів реєстрації, наприклад електронну адресу, номер телефону або інтеграцію в соціальні мережі.
2. Визначення рівня англійської мови користувачів:
   a. Впровадити адаптивну систему рівня, яка оцінює кваліфікацію користувача та переводить його на відповідний етап навчання.
   b. Переконатися, що в міру прогресу користувачів програмне забезпечення динамічно коригує складність і зміст уроків.
3. Перегляд прогресу користувачів:
   a. Створити повну інформаційну панель, яка візуалізує прогрес користувача, включно з пройденими уроками, сильними сторонами та областями, які потребують вдосконалення.
   b. Інтегрувати механізми зворотного зв’язку, які пропонують користувачам практичну інформацію та рекомендації на основі їх ефективності.
4. Вибір завдань для вивчення англійської мови:
   a. Запропонувати користувачам різноманітний набір завдань і вправ, класифікованих за типом навичок (наприклад, читання, вивчення слів, виконання вправ).
   b. Дозволити користувачам власноруч обирати конкретні завдання або сфери уваги на основі особистих навчальних цілей.

2 МОДЕЛЮВАННЯ СИСТЕМИ

2.1 Об’єктно-орієнтоване моделювання

Unified Modeling Language (UML) [11] — це стандартизована мова візуального моделювання, яка використовується в розробці програмного забезпечення та проектуванні систем. Створений у середині 1990-х років UML став важливим інструментом для розробників програмного забезпечення, системних архітекторів і бізнес-аналітиків. Він забезпечує загальний і виразний спосіб представлення структури, поведінки та взаємозв’язків складних систем. UML служить схемою для проектування, документування та передачі різних аспектів розробки програмного забезпечення та системи.

Ключові особливості та аспекти UML включають:

1. Візуальне представлення: UML використовує діаграми та символи для представлення різних елементів і понять у системі. Ці діаграми зазвичай малюються на білих дошках або за допомогою спеціалізованих інструментів моделювання UML, які допомагають зробити складні системи більш зрозумілими.
2. Стандартизація: UML управляється та підтримується Object Management Group (OMG), некомерційним консорціумом професіоналів галузі. Ця стандартизація гарантує, що UML є послідовним і широко прийнятим у спільноті розробників програмного забезпечення.
3. Типи діаграм: UML пропонує кілька типів діаграм, кожна з яких служить певній меті. Деякі з найбільш часто використовуваних діаграм UML включають:
    a. Діаграми класів: вони ілюструють статичну структуру системи, показуючи класи, атрибути, методи та їхні зв’язки.
    b. Діаграми варіантів використання: вони зображують, як система взаємодіє із зовнішніми акторами, і визначають

функціональні вимоги системи.

c. Діаграми послідовності: вони показують взаємодію між об’єктами в часі, зокрема у відповідь на випадки використання або системні операції.
d. Діаграми кінцевого автомата: вони описують стани та переходи об’єкта, допомагаючи моделювати динамічну поведінку.
e. Діаграми діяльності: вони використовуються для представлення робочих процесів і процесів у системі.
f. Діаграми компонентів: вони ілюструють фізичні компоненти системи та їхні залежності.
g. Діаграми розгортання: вони показують апаратні та програмні компоненти системи та їхні взаємозв’язки.
h. Абстракція моделювання: UML забезпечує різні рівні абстракції, дозволяючи розробникам програмного забезпечення створювати моделі високого рівня для розуміння загальної структури системи або детальні моделі для опису конкретних аспектів системи.

4. Спілкування та документація: UML-діаграми служать ефективним засобом спілкування між членами команди та зацікавленими сторонами. Їх можна використовувати для аналізу вимог, обговорення дизайну та документації. Забезпечуючи спільну мову та візуальне представлення, UML допомагає гарантувати, що всі, хто бере участь у проекті, мають спільне розуміння системи.
5. Розширюваність: UML можна розширити для створення предметно-орієнтованих мов моделювання (DSML) або для адаптації до конкретних галузевих потреб, що робить його універсальним інструментом для різних програм.
6. Підтримка інструментів: доступні численні інструменти моделювання UML, як з відкритим кодом, так і комерційні, які

полегшують створення, редагування та керування діаграмами UML. Ці інструменти часто надають такі функції, як створення коду, зворотне проектування та інтеграція контролю версій.

7. Міждисциплінарне використання: UML не обмежується розробкою програмного забезпечення; його можна застосовувати в різних сферах, включаючи моделювання бізнес-процесів, проектування баз даних і апаратного забезпечення. Це робить його цінним інструментом для вирішення широкого кола складних проблем.

2.1.1 Діаграма прецедентів. Діаграма прецедентів [12], окреслює ряд дій, ініційованих суб’єктом, будь то окрема особа чи система. Ці дії передбачають взаємодію або з інформаційною системою, або з іншими акторами, що призводить до обміну повідомленнями. Основна мета цієї діаграми — зобразити сценарії взаємодії між акторами і прецедентами, проливаючи світло на функціональні аспекти системи.

Основна мета цієї діаграми випадків полягає в тому, щоб з’ясувати функціональність і поведінку відповідної системи, сприяючи спільним обговоренням між замовником, користувачем і розробником. Центральні компоненти, зображені на рисунку 2.1, охоплюють:

1. Актори, які представляють активних осіб, таких як користувачі, аналітики та модератори.
2. Прецеденти, що позначають випадки використання або дії, такі як реєстрація, авторизація та отримання даних із бази даних.
3. Межі системи, що охоплюють усі варіанти використання в системі.

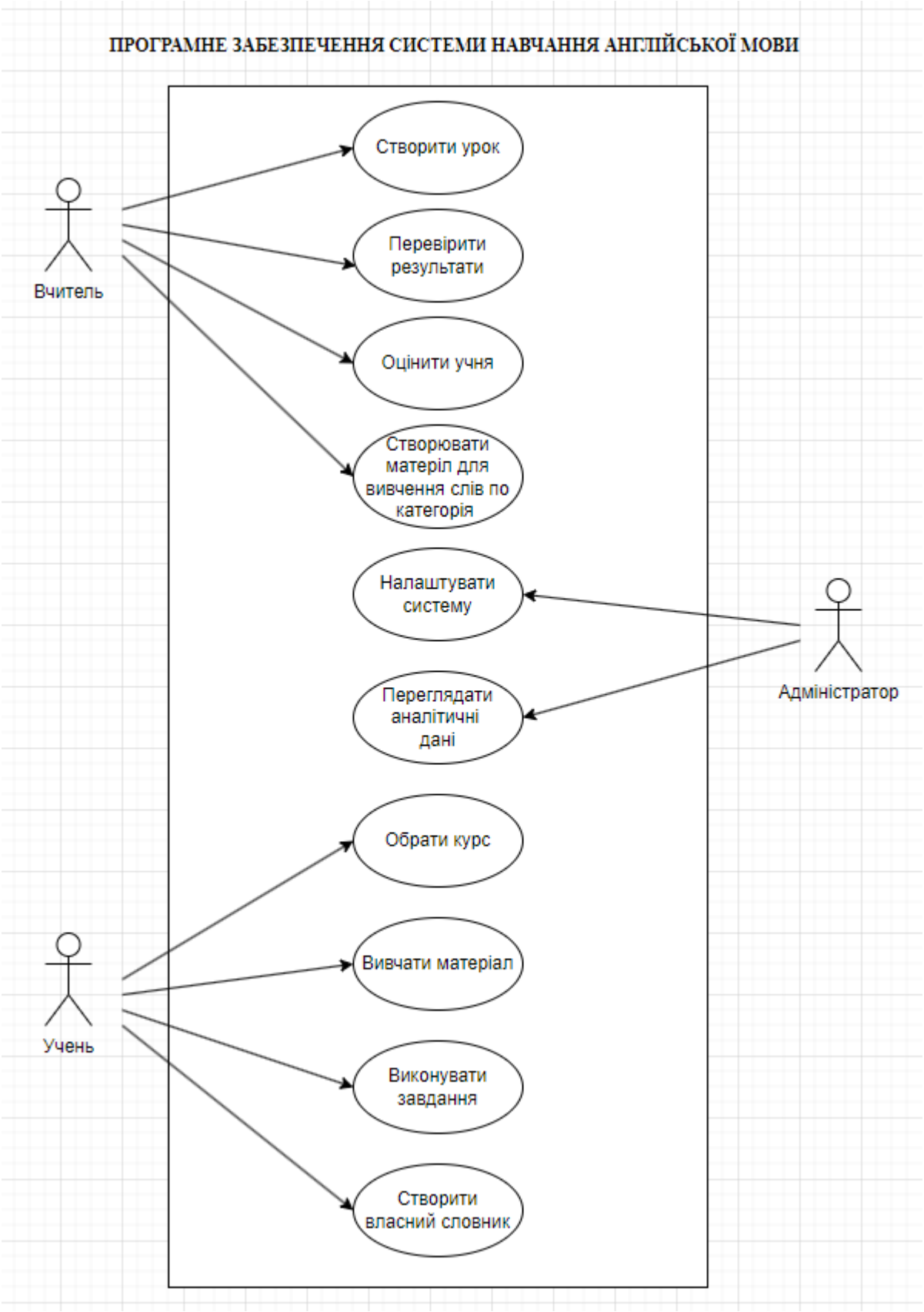

Рис. 2.1 Діаграма прецедентів

2.1.2 Діаграма станів. Діаграма стану [13] — це візуальне представлення, яке відстежує еволюцію станів об’єкта протягом його життєвого циклу, діючи як основний інструмент моделювання поведінки відповідно до UML.

Ключові елементи діаграми стану включають::

- Початковий стан: позначається простим кружечком, він означає початкову точку еволюції стану об’єкта.
- Кінцевий стан: це представлено колом, охопленим меншим колом всередині. Він позначає кінцеву точку або кінцевий стан, якщо такий існує.
- Стан: зазвичай зображується у вигляді закругленого прямокутника, він характеризує окремий стан об’єкта. Назва штату розташована у верхній частині прямокутника. Якщо можливо, у прямокутнику можна намалювати горизонтальну лінію, під якою ви побачите дії, пов’язані з цим конкретним станом.
- Перехід: це візуально представлено стрілкою, яка відображає перехід від одного стану до іншого. Якщо для цього переходу є подія, що запускає, її ім’я буде позначено над або під стрілкою. Для умовних переходів захисний вираз, оточений квадратними дужками, може бути розміщений перед "/". Це означає, що перехід залежить від значення істинності виразу. Крім того, якщо під час цього переходу ініціюється певна діяльність, це задокументовано після назви події за допомогою символу «/».
- Точки розгалуження або з’єднання: ці елементи, позначені чіткою горизонтальною лінією, представляють конвергенцію або розбіжність переходів відповідно. Вони важливі для ілюстрації паралельних станів або одночасних шляхів переходу.

Уточнюючи далі, діаграми станів служать багатьом цілям:

- Чіткість: вони пропонують чітку візуалізацію того, як поводиться

об’єкт або система, розмежовує її різні стани та події, які ініціюють переходи між цими станами.

- Аналіз системи: фіксуючи динамізм об’єкта, ці діаграми допомагають виявити аномалії чи неефективність у системі, відкриваючи шлях для оптимізації.
- Документація: як стандартизований інструмент моделювання, діаграми станів забезпечують послідовну точку відліку для розробників, інженерів та інших зацікавлених сторін протягом життєвого циклу проекту.
- Комунікація: вони діють як спільна мова між різними учасниками проекту, гарантуючи, що кожен має єдине розуміння поведінки системи.

На рисунку 2.2 представлена діаграма стану, що описує процедуру авторизації. Ця діаграма ілюструє основні стани, через які проходить система, коли користувач авторизує доступ через мобільний додаток системи.

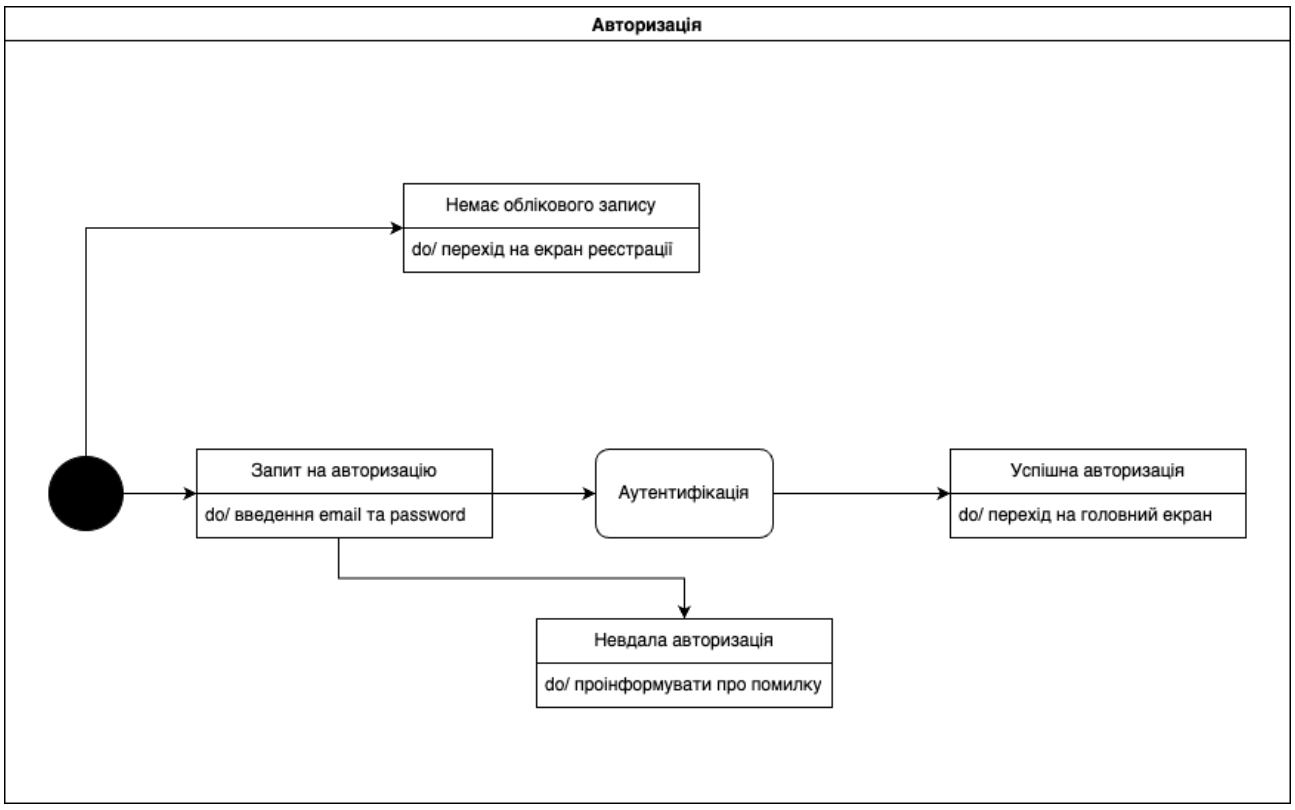

Рис. 2.2 Діаграма станів авторизації

2.2 Опис операцій технологічного процесу обробки даних

Технологічний процес обробки даних у контексті «Програмного забезпечення системи навчання англійської мови» є критичним аспектом функціональності програмного забезпечення. Він включає серію операцій, які разом дозволяють програмному забезпеченню забезпечувати ефективне та результативне вивчення мови. Ці операції включають, але не обмежуються:

1. Реєстрація користувача та створення профілю:
    a. Користувачі повинні мати можливість створювати індивідуальні облікові записи, надаючи основну інформацію та параметри.
    b. Система повинна безпечно зберігати ці дані та дозволяти користувачам отримувати доступ, змінювати або видаляти свої профілі.
2. Автентифікація та авторизація користувача:
    a. Захищені механізми входу, такі як ім’я користувача/пароль або біометрична автентифікація, повинні бути реалізовані для перевірки особи користувача.
    b. Необхідно визначити ролі та дозволи користувачів, гарантуючи, що користувачі мають доступ лише до відповідного вмісту та функцій.
3. Оцінювання та вирівнювання:
    a. Програмне забезпечення має оцінювати початковий рівень мовного рівня користувача за допомогою оцінок, анкет або тестових завдань.
    b. На основі результатів оцінювання система призначає користувачу відповідний рівень навчання або навчальну програму.
4. Подача контенту та навчальні завдання:
    a. Користувачі повинні мати доступ до різноманітних навчальних матеріалів, вправи, тести та мультимедійний вміст.
    b. Система повинна забезпечувати структуровану навчальну програму або шлях, що направляє користувачів через прогресивні складні завдання.

5. Відстеження прогресу та відгуки:
    a. Програмне забезпечення має відстежувати та записувати прогрес користувача, включаючи виконані завдання, результати тестів і час, витрачений на кожен урок.
    b. Користувачі повинні негайно отримувати відгуки про їх продуктивність із пропозиціями щодо покращення та наступними кроками.
6. Адаптивне навчання:
    a. Система повинна динамічно адаптувати процес навчання на основі продуктивності та переваг користувача.
    b. Він може регулювати складність завдань, рекомендувати конкретні уроки або запроваджувати новий вміст для задоволення індивідуальних навчальних потреб.
7. Аналіз даних і персоналізація:
    a. Аналіз даних користувача, таких як шаблони взаємодії та результати тестів, повинен інформувати систему про рекомендації.
    b. Алгоритми персоналізації повинні пристосовувати зміст і темп навчання до сильних і слабких сторін кожного користувача.
8. Соціальні та спільні функції:
    a. Впроваджувати функції, які дозволяють користувачам взаємодіяти з іншими учнями, наприклад відгуки однолітків щодо завдань.
9. Розпізнавання мови та переклад:
    a. Інтегрувати інструменти розпізнавання мови, які забезпечують точний зворотний зв’язок щодо вимови.
    b. Застосувати можливості перекладу, щоб користувачі розуміли та практикували в контексті реального життя.
10. Підтримка користувачів і довідкові ресурси:
    a. Надати канали підтримки користувачів, наприклад відповіді на поширені запитання.

b. Включити довідкові ресурси, підказки та пояснення складних мовних понять.

11.Гейміфікація та мотивація:

a. Включити елементи гейміфікації, як-от бали, значки та таблиці лідерів, щоб мотивувати користувачів.
b. Винагороджувати досягнення, щоб підтримувати зацікавленість користувачів.

12. Оптимізація продуктивності та сумісність:

a. Переконатись що програмне забезпечення оптимізовано для безперебійної роботи на різних пристроях, включаючи мобільні телефони та планшети

13.Оновлення та обслуговування вмісту:

a. Регулярно оновлювати та розширюйте бібліотеку вмісту, щоб вона залишалася актуальною та привабливою.
b. Виконувати технічне обслуговування та оновлення системи для забезпечення надійності та безпеки.

3 РОЗРОБКА СИСТЕМИ

3.1 Інформаційне забезпечення системи

3.1.1 Логічна модель даних. Логічна модель даних [14] - це фундаментальна концепція в галузі проектування баз даних та управління інформацією. Вона слугує абстрактним представленням елементів даних, їхніх взаємозв'язків і правил, що регулюють роботу з даними в організації. Основна мета логічної моделі даних - надати чітке і структуроване уявлення про дані, яке не залежить від будь-якої конкретної системи управління базами даних або фізичної реалізації.

На рисунку 3.1 представлено логічну схему даних для системи, яка пропонує схематичне зображення зв'язків між різними об'єктами та конкретної інформації, яку вони зберігають:

- User – сутність представляє інформацію про користувачів системи. Вона містить дані, пов'язані з обліковими записами користувачів та їхніми ролями.
- Course – сутність представляє інформацію про курси, що пропонуються у системі. Вона пов'язана з сутністю "user", щоб встановити зв'язок між викладачами та курсами.
- Course Review – сутність призначена для збору відгуків та оцінок, наданих студентами для певних курсів. Вона встановлює зв'язки з сутністями "cource" і "user", щоб пов'язати відгуки з курсами і студентами.
- Lesson – сутність представляє інформацію про окремі уроки в межах курсу. Вона пов'язана з сутністю "course", щоб встановити зв'язок між уроками та курсами.
- Lesson Review – сутність призначений для збору відгуків та оцінок,

наданих учнями для конкретних уроків. Вона встановлює зв'язки з сутністями "lesson" і "user", щоб пов'язати відгуки з уроками та учнями.

- Material – сутність представляє інформацію про матеріали, пов'язані з уроками. Вона пов'язана з сутністю "lesson", щоб вказати урок, до якого належить матеріал.
- Material Comment – сутність використовується для збору коментарів або обговорень, пов'язаних з конкретними матеріалами. Вона встановлює зв'язки з таблицями "material" і "user", щоб пов'язати коментарі з матеріалами і користувачами.
- Quiz – сутність використовується для визначення тестів, пов'язаних з певними матеріалами.
- Quiz Card – сутність зберігає окремі питання та відповіді для вікторин, пов'язаних з матеріалами. Вона пов'язана з сутністю "quiz", щоб вказати вікторину, до якої належать запитання і відповідь.
- Quiz Card Result – сутність використовується для зберігання відповідей студентів на окремі питання. У ньому зберігається інформація про результати учня в конкретному питанні.
- Sentence – сутність зберігає речення, що використовуються в матеріалах для вивчення мови.
- Sentence Result – сутність використовується для зберігання відповіді учнів на вправи з реченнями. Він пов'язує відповіді учнів з реченнями та учнями.
- Paragraph – сутність представляє інформацію про абзаци, що використовуються в для представлення інформації, замітки до уроку.
- Dictionary Category та Dictionary – сутності використовуються для впорядкування та керування словниками у вашій системі. Ці таблиці

забезпечують структурований спосіб категоризації та зберігання словників.

- Dictionary Review – сутність використовується для оцінювання словників користувачами. Вона пов'язує оцінки користувачів з конкретними словниками.
- Dictionary Card – сутність використовується для зберігання слів для вивчення. Вона пов'язує слова зі словниками, дозволяючи користувачам отримати доступ до визначень слів.
- Dictionary Quiz та Dictionary Quiz Card – сутності використовуються для тестів, створених на основі словників. Вони дозволяють користувачам практикувати і перевіряти свої знання змісту словника.

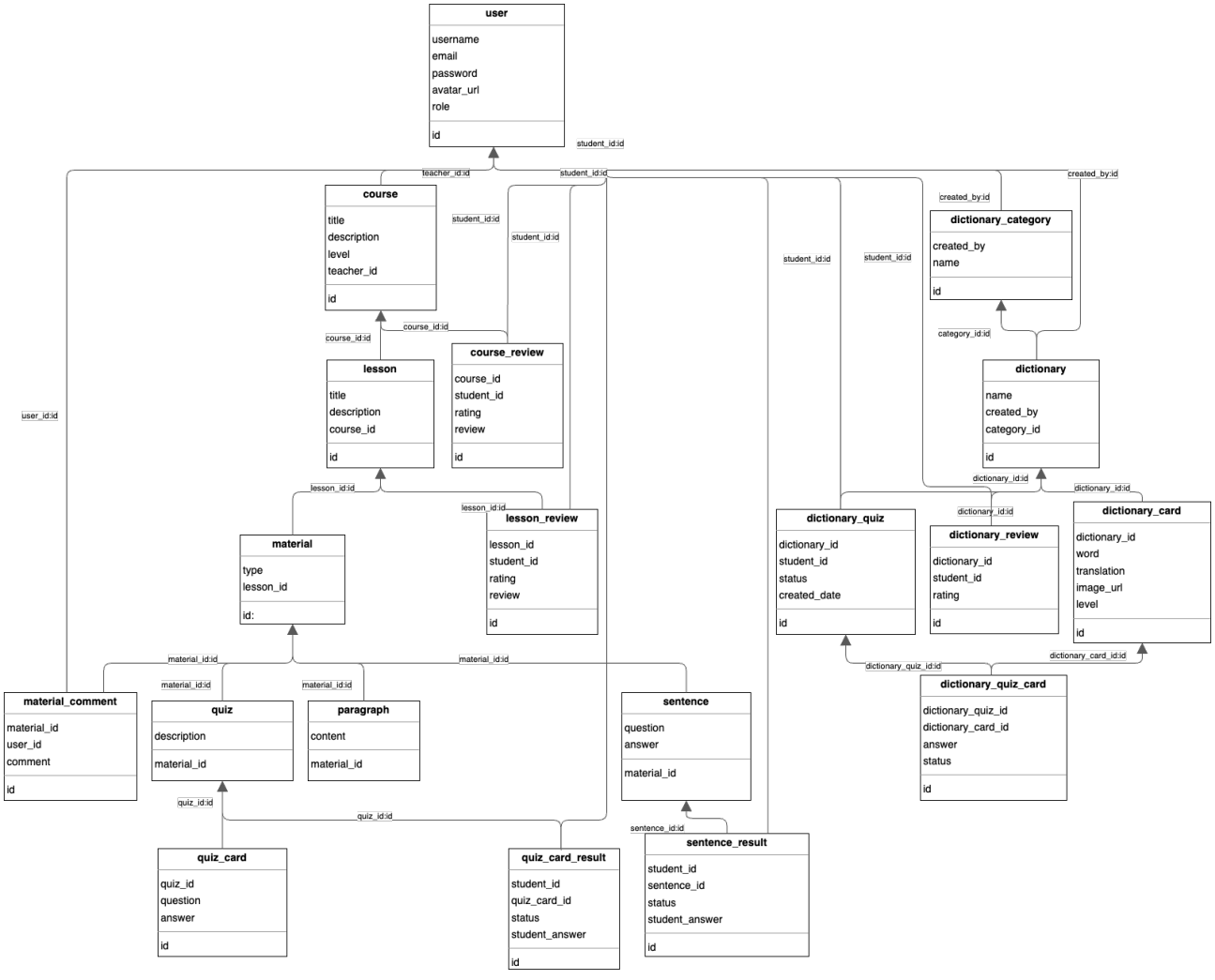

Рис. 3.1 Логічна модель системи

3.1.2 Вибір середовища проектування БД. Для системи було вирішено використовувати Microsoft SQL Server [15]. У величезній сфері систем керування реляційними базами даних (RDBMS) Microsoft SQL Server, який часто позначають як MSSQL, виділяється як надійне та універсальне рішення. Заснований технологічним гігантом Microsoft.

За своєю суттю MSSQL призначений для зберігання, отримання та керування реляційними даними. Однак глибина його можливостей виходить за межі простого зберігання даних. Головною особливістю MSSQL є різноманітність версій. Обслуговуючи широкий спектр користувачів, від транснаціональних корпорацій до незалежних розробників, ці випуски пропонують гнучкість і масштабованість. Випуск Enterprise, наприклад, створений для великих організацій, яким потрібні розширені функції та максимальна масштабованість. На іншому кінці спектра лежить версія Express, безкоштовна, але обмежена версія, ідеальна для невеликих програм або навчальних цілей. Крім того, випуск для розробників заслуговує на особливу згадку, пропонуючи всі можливості версії Enterprise, але спеціально ліцензований для розробки та тестування, що робить його улюбленим серед спільноти розробників.

Однією з видатних характеристик MSSQL є його повна інтеграція з іншими продуктами Microsoft. Чи то Windows Server, який забезпечує оптимізоване середовище для розгортання MSSQL, чи хмарна пропозиція Microsoft, Azure, MSSQL забезпечує синхронізовану та ефективну роботу. Ця інтеграція не тільки оптимізує роботу, але й надає підприємствам, які глибоко інвестували в інфраструктуру Microsoft, єдину екосистему.

Обговорення MSSQL було б неповним без згадки про SQL Server Management Studio (SSMS). Як наріжний інструмент для адміністраторів, розробників і аналітиків даних, SSMS забезпечує комплексне інтегроване середовище. Завдяки графічному інтерфейсу користувача користувачі можуть без особливих зусиль виконувати запити, керувати структурами бази даних і

навіть виконувати такі важливі завдання, як резервне копіювання та відновлення даних. Саме це поєднання потужності та зручності робить SSMS незамінним.

Підсумовуючи, Microsoft SQL Server — це не просто RDBMS; це свідчить про прагнення Microsoft надавати компаніям і розробникам інструменти керування даними найвищого рівня. Завдяки різноманітним випускам, тісній інтеграції з іншими продуктами Microsoft і безцінній SSMS, MSSQL продовжує залишатися ключовою силою у світі керування та аналізу даних.

3.1.3 Реалізація фізичної моделі даних. Функціональна база даних PostgreSQL була створена після аналізу логічної моделі даних та налаштувань дизайну бази даних. На рисунку 3.2 показано схему бази даних.

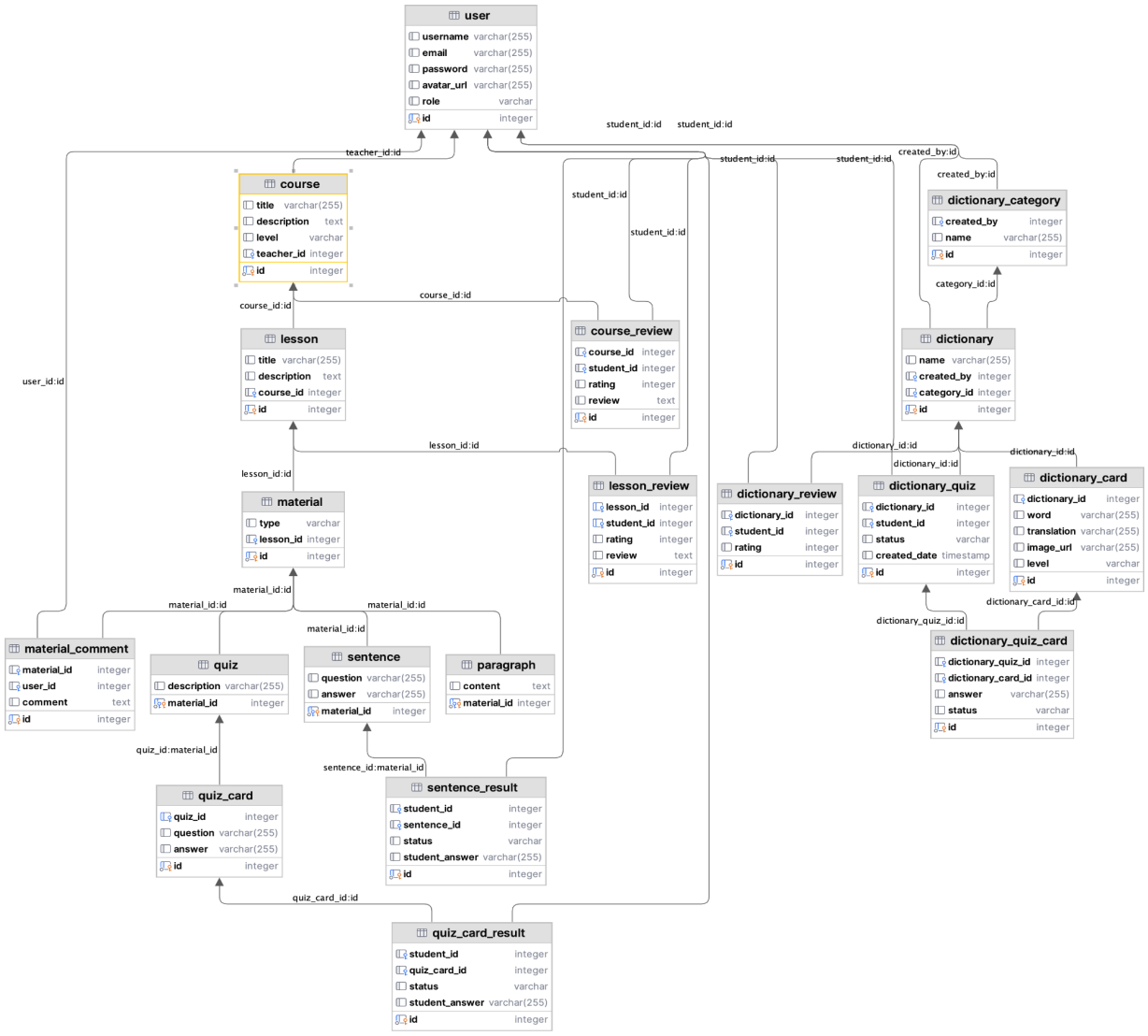

Рис. 3.2 Інфологічна модель бази даних

База даних складається з дев'ятнадцять таблиць

- User – таблиця представляє користувачів системи. Стовпці включають:
  - id – унікальний ідентифікатор для кожного користувача. Являється первинним ключем
  - username – ім'я, вибране користувачем для свого профілю;
  - email – адреса електронної пошти користувача;
  - password – зашифрований пароль (сподіваємось, хешований і доданий) для користувача;
  - avatar_url – URL-адреса зображення профілю користувача;
  - role – роль користувача, імовірно вказуючи, чи є він учнем, викладачем, адміністратором тощо.
- Course – таблиця представляє різні курси на платформі. Стовпці включають:
  - id – унікальний ідентифікатор для кожного курсу. Являється первинним ключем;
  - title – назва курсу;
  - description – текстовий опис курсу;
  - level – рівень або складність курсу;
  - teacher_id – ідентифікатор користувача, який створив курс. Це посилання на стовпець id у таблиці користувача.
- Course Review – таблиця представляє відгуки, залишені студентами на курси. Стовпці включають:
  - id – унікальний ідентифікатор для кожного огляду. Являється первинним ключем;
  - course_id – ID курсу, який було перевірено;
  - student_id – ID студента, який залишив відгук;
  - rating – наданий чисельний рейтинг;
  - review – Текст рецензії, наданий студентом.

- Lesson – таблиця являє собою індивідуальні заняття в рамках курсу. Стовпці включають:
  - id – унікальний ідентифікатор для кожного уроку. Являється первинним ключем;
  - title – назва уроку;
  - description – текстовий опис уроку;
  - course_id – ID курсу, до якого належить цей урок.
- Lesson Review – таблиця представляє відгуки, залишені учнями до індивідуальних занять. Стовпці включають:
  - id – унікальний ідентифікатор для кожного відгуку;
  - lesson_id – зовнішній ключ, що посилається на конкретний урок;
  - student_id – зовнішній ключ, що посилається на учня, який залишив відгук;
  - rating – числовий рейтинг уроку;
  - review – текстовий відгук про урок.
- Material – таблиця представляє матеріали, пов'язані з уроком, такі як параграфи, вікторини. Стовпці включають:
  - id: унікальний ідентифікатор для кожного матеріалу;
  - type: тип матеріалу (наприклад, параграф, вікторина);
  - lesson_id: зовнішній ключ що є ідентифікатором уроку, до якого належить цей матеріал.
- Material Comment – таблиця відображає коментарі, залишені користувачами до матеріалів. Стовпці включають:
  - id – унікальний ідентифікатор для кожного коментаря;
  - material_id – зовнішній ключ, який посилається на певний матеріал;
  - user_id: зовнішній ключ, що посилається на користувача, який залишив коментар;
  - comment - вміст коментаря.

- Quiz – таблиця являє собою вікторини, які є різновидом матеріалу. Стовпці включають:
  - material_id – ID матеріалу, який є тестом. Являється первинним ключем та зовнішнім ключем;
  - description - опис вікторини.
- Dictionary – таблиця представляє різні словники зі словами для вивчення. Стовпці включають:
  - id – унікальний ідентифікатор словника;
  - name – ім'я або назва словника;
  - created_by – зовнішній ключ, що посилається на користувача, який створив словник;
  - category_id – зовнішній ключ, що посилається на категорію, до якої належить словник.
- Dictionary Review – таблиця представляє рецензії, залишені учнями до конкретних словників. Стовпці включають:
  - id – унікальний ідентифікатор для кожного відгуку;
  - dictionary_id – зовнішній ключ, що посилається на конкретний словник;
  - student_id – зовнішній ключ, що посилається на учня, який залишив відгук;
  - rating – числовий рейтинг словника.
- Dictionary Card – таблиця представляє окремі слова в словнику. Стовпці включають:
  - id – унікальний ідентифікатор для кожної картки словника;
  - dictionary_id – зовнішній ключ, що посилається на конкретний словник;
  - word – слово або термін;
  - translation – переклад або пояснення слова;
  - image_url – URL-адреса пов'язаного зображення;

- level – складність.

- Dictionary Quiz – таблиця являє собою вікторини за словниковими картками. Стовпці включають:
  - id – унікальний ідентифікатор для вікторини;
  - dictionary_id – зовнішній ключ, що посилається на певний словник;
  - student_id – зовнішній ключ, що посилається на студента, який пройшов або пройде тест;
  - status – статус тесту, наприклад, "in_progress", "created";
  - created_date – позначка часу, коли тест було створено або пройдено.

- Dictionary Quiz Card – таблиця являє собою відповіді студентів на питання в межах словникової вікторини. Стовпці включають:
  - id – унікальний ідентифікатор для кожної картки вікторини;
  - dictionary_quiz_id – зовнішній ключ, що посилається на конкретний словниковий тест;
  - dictionary_card_id – зовнішній ключ, що посилається на конкретну картку словника, на якій базується питання;
  - answer: відповідь учня;
  - status – статус відповіді, наприклад, "incorrect".

- Sentence – таблиця представляє окремі речення, які є типом матеріалу, для вправ з заповненням пропущених слів, букв або переклад. Стовпці включають:
  - material_id – зовнішній ключ-посилання на таблицю матеріалів. Виконує роль первинного ключа;
  - question - речення з пропущеними словами, буквами або оригінальний текст;
  - answer – правильна форма речення або переклад.

- Sentence Result – таблиця представляє результати спроб студентів

відповісти або перекласти речення. Стовпці включають:

- id - унікальний ідентифікатор для кожного результату;
- student_id - посилання на користувача, який намагався виконати завдання;
- sentence_id – посилання на речення, яке намагаються виконати;
- status – результат, наприклад, "correct", "incorrect";
- student_answer – відповідь або переклад речення, наданий учнем.

● Paragraph – таблиця представляє абзаци, які є типом матеріалу для вправ на розуміння прочитаного або перекладу. Стовпці включають:

- material_id – посилання на таблицю матеріалів і діє як первинний ключ;
- content – текстовий вміст абзацу.

● Dictionary Category – таблиця представляє категорії для словників, що дозволяють організувати різні типи словників, наприклад, "Медичні терміни", "Фрази для подорожей". Стовпці включають:

- id – унікальний ідентифікатор для кожної категорії;
- created_by – посилання на користувача, який створив категорію;
- name – назва категорії.

Зв’язки між таблицями:

- Material і Sentence - відношення один до одного. Кожне речення є типом матеріалу;
- User і Sentence Result – користувач може мати кілька результатів речень, але кожен результат відповідає одному користувачеві і одному реченню;
- Material і Paragraph: Відношення один до одного. Кожен параграф є типом матеріалу;
- User і Dictionary Category – користувач може створити кілька категорій словників, але кожна категорія створюється одним користувачем;
- Dictionary Category і Dictionary – категорія може мати кілька словників, але

кожен словник належить до однієї категорії;

- User і Dictionary – користувач може створити кілька словників, але кожен словник створюється одним користувачем;
- User і Dictionary Review – користувач може залишити кілька відгуків на словники, але кожен відгук відповідає одному користувачеві і одному словнику;
- Dictionary і Dictionary Card – словник може мати кілька карток, але кожна картка належить до одного словника;
- User і Dictionary Quiz – користувач може брати участь у кількох тестах на знання словника, але кожен тест відповідає одному користувачеві та одному словнику;
- Dictionary Quiz і Dictionary Quiz Card – вікторина може мати кілька карток, але кожна картка відповідає одній вікторині;
- User і Course – користувач може викладати декілька курсів, але кожен курс викладається одним користувачем;
- User і Course Review – користувач може залишити кілька відгуків на курс, але кожен відгук прив'язаний до одного користувача і одного курсу;
- User і Lesson Review – користувач може залишити кілька відгуків до уроку, але кожен відгук прив'язаний до одного користувача і одного уроку;
- Course і Lesson – курс може мати кілька уроків, але кожен урок прив'язаний до одного курсу;
- Lesson і Material – урок може мати кілька матеріалів, але кожен матеріал прив'язаний до одного уроку;
- Material і Quiz - зв'язок один до одного. Кожна вікторина також є матеріалом, але не всі матеріали є вікторинами.
- Quiz і Quiz Card – вікторина може мати кілька карток (запитань), але кожна картка прив'язана до однієї вікторини;
- User and Material Comment – користувач може залишати кілька коментарів до матеріалів, але кожен коментар прив'язаний до одного користувача і одного матеріалу;

- User і Quiz Result – користувач може мати результати з декількох вікторин, але кожен результат прив'язаний до одного користувача і однієї вікторини.

3.1.4 Проектування сховища даних. Сховище даних [16]– спеціалізована база даних, що має такі особливості:

- предметність – сховище містить інформацію, що стосується конкретного об’єкту бізнес процесу;
- узагальнення – сховище отримує дані з різних джерел інформації;
- чистота – вхідні дані проходять фільтрацію, що дозволяє прибрати дублі, доповнити чи узагальнити записи;
- незмінність – наповнення сховища даних відбувається досить рідко, після накопичення вагомого об’єму даних та їх обробки;
- залежність від часу – визначною характеристикою сховища даних є часовий вимір, що дозволяє відслідковувати динаміку розвитку об’єкта а також прогнозувати її.

Технологічно сховища даних тісно пов'язані із засобами оперативної аналітичної обробки даних (OLAP-технологіями), що дозволяють аналітикам, керівникам і керівникам вищої ланки вивчати великі обсяги взаємопов'язаних даних за допомогою швидкого інтерактивного відображення інформації на різних рівнях деталізації.

3.2 Програмне забезпечення системи

3.2.1 Архітектура програмного забезпечення. У сфері цифрових рішень проектування та розробка системних архітектур є ключовими. На основі ретельного аналізу предметної області та складно розроблених БД і СД було прийнято рішення на користь моделі клієнт-сервер, системної архітектури, яка стала синонімом мобільних додатків.

Архітектура [17] містить кілька явних переваг. По-перше, це втілення простоти. Користувачі не обтяжені завантаженням великого програмного

забезпечення чи програм. Натомість компактний мобільний додаток служить шлюзом до безлічі функцій. По-друге, виділяється фактор доступності. Передумови для доступу до цієї архітектури мінімальні – достатньо мати мобільний пристрій із програмою та стабільне підключення до Інтернету. Нарешті, архітектура блищить з точки зору ефективності. Оскільки більша частка операцій виконується на стороні сервера, вимоги до обчислювальних ресурсів мобільного пристрою помітно зменшуються, що забезпечує безперебійну роботу користувача.

Роз’яснюючи далі, рисунок 3.4, який є невід’ємною частиною нашого обговорення, представляє топологію нашої передбачуваної системи. Він чітко розмежовує два основні компоненти:

- Мобільний пристрій: представляє клієнтську сторону, охоплює такі пристрої, як смартфони або планшети, які служать основним інтерфейсом користувача з системою.
- Вузли з даними: діючи як магістраль, це сторона сервера, що містить усі необхідні компоненти, гарантуючи, що система працює.

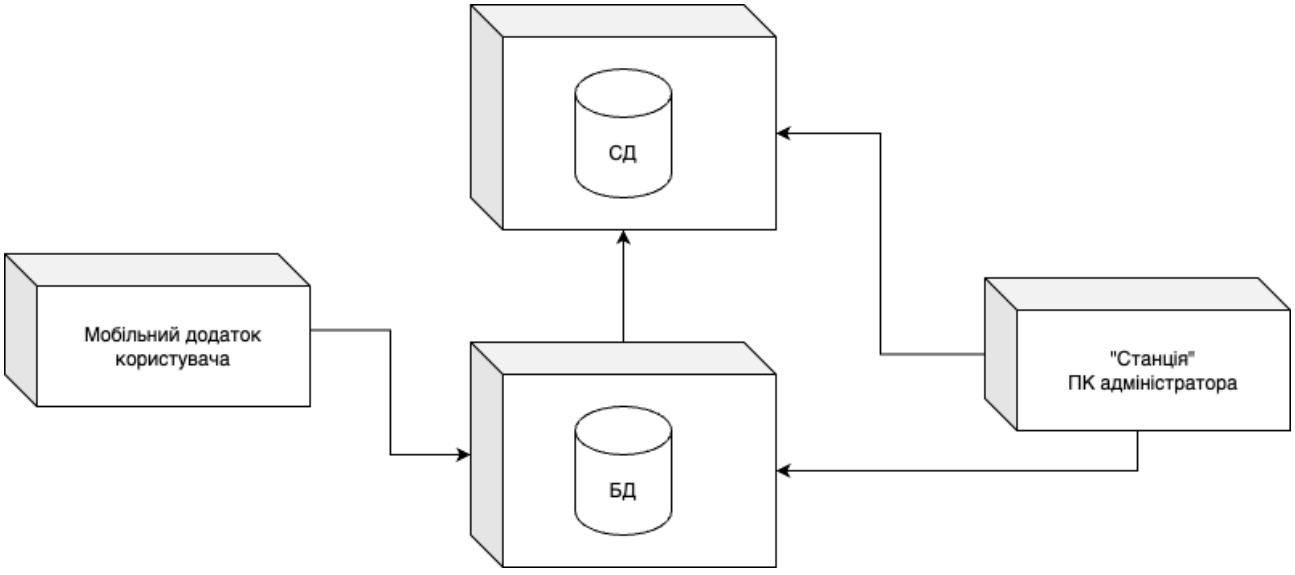

Рис. 3.4 Топологія системи

Підсумовуючи, клієнт-серверна архітектура, адаптована для мобільних додатків, пропонує гармонійне поєднання простоти, доступності та

ефективності. Він не лише задовольняє потреби користувачів, що постійно змінюються, але й забезпечує надійну та масштабовану структуру для розробників і компаній.

3.2.2 Обрані інструменти та програмні засоби для реалізації програмного забезпечення системи. Для реалізації системи були використані такі технології:

- React Native [18] – фреймворк для створення мобільних додатків
- Java [19] – кросплатформена мова програмування для реалізації серверної частини
- Spring Framework [20] – фреймворк для Java, який допомагає з реалізацією серверної частини
- OpenAI API [21] – API для використання штучного інтелекту

3.2.2.1 React Native. React Native дозволяє розробникам створювати мобільні програми як для платформ iOS, так і для Android, використовуючи єдину кодову базу. Він використовує той самий дизайн, що й React [22], що дає розробникам можливість створювати багатий мобільний інтерфейс за допомогою декларативних компонентів.

Ключові особливості:

- Кросплатформна розробка.
- Рідні компоненти: React Native використовує нативні компоненти, що означає, що компоненти інтерфейсу користувача, які використовуються, відображатимуться за допомогою своїх рідних аналогів платформи.
- Продуктивність: хоча React Native використовує JavaScript, він виконує код JS в окремому потоці від основного потоку інтерфейсу користувача, тому інтерфейс користувача програми залишається плавним.
- Швидке перезавантаження під час розробки: дозволяє розробникам миттєво побачити результат останньої зміни.
- Спільнота: React Native має та активну спільноту. Це означає, що доступна велика кількість сторонніх бібліотек, плагінів і розширень.

3.2.2.2 Java. Java — це об’єктно-орієнтована мова програмування високого рівня, вперше випущена компанією Sun Microsystems у 1995 році. Вона відома своєю портативністю, масштабованістю та надійністю. Слоган мови «Write Once, Run Anywhere» (WORA) підкреслює її кросплатформні можливості, оскільки скомпільований код Java може працювати на будь-якому пристрої, обладнаному віртуальною машиною Java (JVM).

Ключові особливості:

- Незалежність від платформи: код Java компілюється в байт-код, який потім інтерпретується JVM. Це означає, що той самий скомпільований байт-код може працювати на будь-якому пристрої з JVM, незалежно від його базової архітектури.
- Об’єктно-орієнтованість: Java переважно об’єктно-орієнтована, що означає, що вона використовує об’єкти та класи для організації коду. Це забезпечує модульність, повторне використання коду та полегшує обслуговування.
- Надійність: Java робить сильний акцент на ранній перевірці помилок, перевірці під час виконання та збиранні сміття, що робить її надійною мовою, менш схильною до збоїв.
- Безпека: такі функції, як перевірка байт-коду та перевірки безпеки під час виконання, роблять Java стійкою до багатьох вразливостей.
- Багатопоточність: Java підтримує багатопотокове програмування, що дозволяє розділяти завдання на менші, паралельні частини, щоб потенційно швидше виконуватися.
- Багата стандартна бібліотека: Java надає велику стандартну бібліотеку, відому як Java Standard Edition API, яка охоплює введення/виведення, мережу, структури даних, графіку та багато іншого.

3.2.2.3 Spring. Spring Framework, яку часто називають просто «Spring», створив Род Джонсон і вперше випустив у 2003 році. Її основний принцип полягає в забезпеченні комплексної підтримки інфраструктури для розробки

додатків Java, дозволяючи розробникам зосереджуватися виключно на бізнес-логіці рівня програми без зайвих зв’язків. до певних середовищ розгортання.

Ключові особливості:

- Інверсія контролю (IoC) [23]: Spring представляє концепцію IoC через впровадження залежностей (DI) [24]. Об’єктам надаються їхні залежності замість того, щоб їх створювати, що робить код більш роз’єднаним і легшим для тестування.
- Аспектно-орієнтоване програмування (AOP) [25]: Spring підтримує AOP, дозволяючи розробникам визначати наскрізні проблеми (наприклад, ведення журналів або облік транзакцій) окремо від основної бізнес-логіки.
- Доступ до даних: спрощує взаємодію з базою даних, забезпечуючи підтримку традиційних фреймворків JDBC [26] і ORM [27], таких як Hibernate [28], JPA [29] та JDO [30].
- Управління транзакціями: забезпечує програмне та декларативне керування транзакціями як для програмних, так і для декларативних операцій читання-запису бази даних.
- MVC Framework [31]: Spring MVC — це веб-модуль, який забезпечує багату структуру для створення веб-додатків. Він бездоганно інтегрується з різними технологіями перегляду, такими як JSP [32], Thymeleaf [33] тощо.
- Безпека: Spring Security [34] пропонує комплексні функції безпеки, такі як автентифікація, авторизація та захист від поширених вразливостей.
- Модульність: Spring має модульну структуру, що дозволяє розробникам вибирати, які модулі потрібні для їхніх програм.

3.2.2.3 OpenAI API. OpenAI API є шлюзом до однієї з найдосконаліших у світі моделей машинного навчання. Забезпечуючи простий інтерфейс для цих моделей, OpenAI дозволяє користувачам використовувати їхні можливості без необхідності тренувати чи налаштовувати самі моделі, роблячи передовий ШІ доступним для ширшої аудиторії.

OpenAI API демократизує доступ до найсучасніших моделей машинного навчання. Завдяки простому у користуванні інтерфейсу та широким можливостям він має потенціал революціонізувати те, як підприємства, розробники та дослідники використовують потужність штучного інтелекту. Чи для створення нових програм, чи для пошуку відповідей на складні проблеми, OpenAI API є свідченням прогресу та потенціалу сучасного ШІ.

3.2.3 Технології OLAP та Data mining. У сучасному середовищі, де першочергово панують дані, такі інструменти, як OLAP [35] і інтелектуальний аналіз даних, є критично важливою основою для аналізу даних і бізнес-аналітики. Ці технології можуть мати певну спільну мову, але в основному вони задовольняють різні потреби та заповнюють різні ніші в широкій сфері науки про дані.

OLAP — це абревіатура від Online Analytical Processing, її призначення полягає у забезпеченні швидких і складних аналітичних запитів. В основному він працює з багатовимірними структурами даних, які зазвичай уявляють собі як куби, що дозволяє користувачам детально досліджувати та досліджувати свої дані. Користувачі можуть змінювати точки зору та деталізацію за допомогою таких функцій, як нарізка, розрізання, деталізація або згортання, сприяючи динамічному підходу до аналізу даних. OLAP вимагає попередньої стадії підготовки даних, яка включає очищення, перетворення та підсумовування, таким чином надаючи аналітикам раціональну та ефективну колекцію даних. Провідні програми в цій області, такі як Microsoft Analysis Services і Oracle OLAP, дозволили компаніям використовувати потенціал даних завдяки зручним інтерфейсам і надійній обробці на стороні сервера.

3.2.4 Розгортання OLAP-куба. Конструкція, відома як Cube-OLAP, представляє складну архітектуру даних, оптимізовану для швидкого й ефективного маніпулювання великими наборами даних, які можуть базуватися на основі сховища даних. Завдяки можливостям служби SSAS ця спеціалізована структура даних дозволяє відносно легко обробляти великі обсяги інформації.

Служби аналізу SQL Server або SSAS легко інтегруються в середовище Visual Studio, надаючи надійний набір інструментів для роботи з технологіями OLAP і інтелектуального аналізу даних. Суть функціональних можливостей SSAS полягає в його вмілому обробленні кубів OLAP, розроблених на основі сховищ даних, що полегшує розширену аналітику даних.

Ініціація куба OLAP у структурі SSAS — це процедура, яка починається зі встановлення підключення до проекту сховища даних. Цей модуль діє як навігаційний інструмент, який направляє користувачів через складний процес налаштування та розгортання куба OLAP, гарантуючи, що складні дані, що містяться в сховищі даних, можуть бути перетворені в структуровану форму, налаштовану на детальний аналіз і генерацію розуміння. Процес розгортання, хоч і має технічний характер, оптимізується за допомогою майстра джерела даних, що робить його доступним для користувачів кроком для переходу від необроблених даних до аналітичної системи, готової для дослідження та відкриття.

Після початкового підключення до сховища даних у проекті на основі SSAS наступним важливим кроком є створення представлення джерела даних. Це є обов’язковим, оскільки з’єднання з джерелом даних, хоча воно і забезпечує фундаментальний зв’язок із реляційною базою даних, недостатньо для розширених аналітичних функцій, необхідних у OLAP. Перегляд джерела даних розширює цей зв’язок, об’єднуючи зв’язки, дозволяючи створювати обчислювані поля та визначаючи логічні первинні та зовнішні ключі. По суті, представлення джерела даних служить візуальним представленням того, як SSAS інтерпретуватиме та організовуватиме сховище даних, зіставляючи його з реляційною схемою. Це являє собою додатковий рівень абстракції над реляційною базою даних, спрощуючи складні структури даних у більш зручну для аналізу форму.

Спираючись на цю основу, наступним етапом у розробці куба OLAP є побудова його розмірів – вони визначаються як упорядковані набори ієрархій, які структурують дані в кубі, забезпечуючи осі, за якими проводиться аналіз. Як

правило, куб OLAP структурований щонайменше за чотирма основними вимірами, як показано на рисунку 3.5. Ці параметри є не просто групуванням, а невід'ємною частиною структури для нарізки та розділення даних, що дозволяє користувачам заглиблюватись у різні рівні деталізації та таким чином полегшує багатогранний аналіз. Кожен вимір діє як шлюз до глибшого розуміння даних, допомагаючи розкрити тонкощі та виявити закономірності, які можуть бути непомітними на поверхневому рівні.

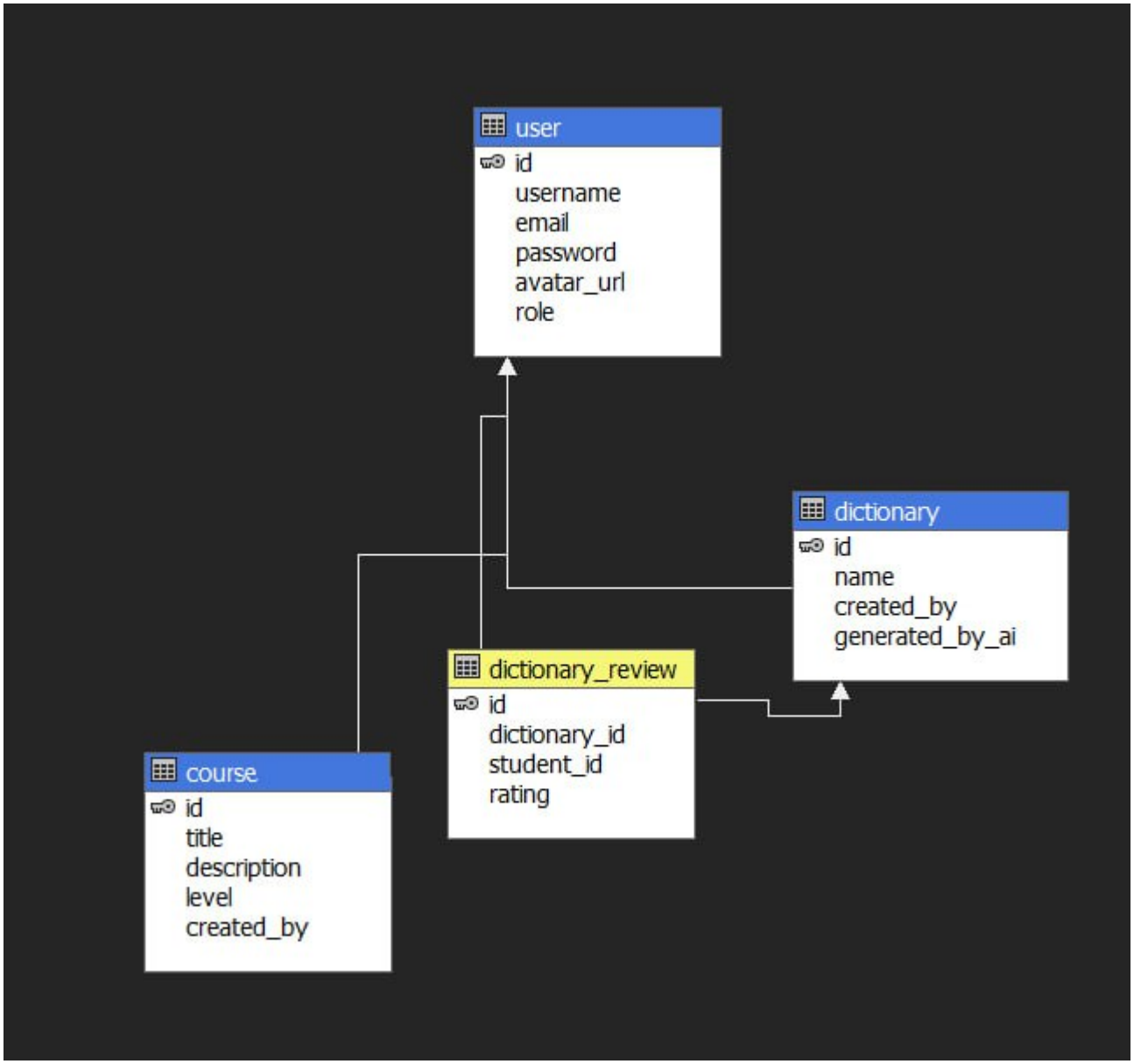

Рис. 3.5 Куб

3.2.5 Розробка аналітичного модуля системи.

3.2.5.1 Naive Bayes Algorithm. Алгоритм Microsoft Naive Bayes — це набір класифікаторів, підкріплених теоремою Байєса, примітний не тим, що є єдиним підходом, а скоріше набором алгоритмів, пов'язаних єдиною концепцією: припущенням про взаємну незалежність між кожною парою ознак, що класифікуються. Прізвище «наївний» походить від цього припущення, оскільки воно спрощує байєсівський підхід, і не звертає увагу на потенційні залежності між функціями.

Незважаючи на свою простоту, алгоритм Naive Bayes виділяється своєю обчислювальною ефективністю порівняно з іншими більш складними алгоритмами, які пропонує Microsoft. Ця особливість робить його особливо вигідним для швидкої розробки моделей майнінгу, спрямованих на виявлення взаємодії між наборами вхідних даних і їх відповідними виходами.

3.2.5.2 Association Algorithm. Алгоритм правил асоціації Microsoft є потужною технікою інтелектуального аналізу даних, яка використовується для виявлення інтригуючих асоціацій між різними елементами набору даних. Основна мета алгоритму полягає в тому, щоб отримати та сформувати правила асоціації, які зв'язують різні точки даних.

Щоб ефективно розгорнути цей алгоритм, потрібно пройти кілька ключових кроків:

1. Основна робота передбачає структурування даних у форматі, зручному для алгоритму, як правило, у табличній формі, де рядки позначають окремі транзакції, а стовпці відповідають предметам, які можна придбати. У цій матриці двійкові індикатори (1 для наявності, 0 для відсутності) позначають появу кожного елемента в транзакціях.
2. Побудова наборів правил асоціації: алгоритм шукає комбінації продуктів, які часто купують разом, обчислюючи певні показники, такі як підтримка та впевненість, важливі терміни в цьому контексті:
3. Підтримка: відображає, як часто набір продуктів з'являється в наборі даних, причому висока підтримка вказує на значну поширеність асоціації.

4. Впевненість: вимірює ймовірність того, що клієнти купуватимуть певну комбінацію продуктів, якщо купуватимуть інший визначений набір товарів.
5. Вибір правил: після встановлення рівнів підтримки та надійності встановлюються порогові значення для цих показників. Правила, які відповідають цим пороговим значенням, позначаються як важливі та відповідні для подальшого застосування.
6. Інтерпретація та застосування: результуючі правила асоціації можна інтерпретувати та використовувати в різних сферах, таких як мерчандайзинг, оптимізація розміщення продукту та індивідуальні рекомендації.

Такі відомі алгоритми, як «Apriori» та «FP-growth», уособлюють методи, які використовуються для ефективного аналізу правил у величезних наборах даних. Такі алгоритми відіграють ключову роль у виявленні прихованих зв'язків між елементами даних, таким чином надаючи інформацію та покращуючи процеси прийняття рішень.

Відмінний аспект алгоритму асоціативних правил полягає в природі даних, які він обробляє, які особливо структуровані для цієї форми аналітичного дослідження.

3.2.5.3 Clustering Algorithm. Алгоритм кластеризації Microsoft — це складний інструмент, розроблений для сегментації даних, який систематично обробляє точки даних для організації їх у окремі кластери. Ці кластери являють собою колекції випадків даних зі спільними атрибутами, і вони виявляються дуже корисними для різноманітних аналітичних цілей, включаючи перевірку даних, виявлення аномалій і прогнозну аналітику.

Цей конкретний алгоритм чудово розкриває закономірності та зв'язки в наборі даних, які можуть бути не відразу очевидними під час випадкового спостереження, забезпечуючи таким чином глибше розуміння, кероване даними.

Щоб виконати кластерний аналіз за допомогою цього алгоритму, ми будемо використовувати інструменти, надані SQL Server Analysis Services

(SSAS). Процес передбачає використання аналітичного майстра, який взаємодіє з SSAS, який допомагає створити структуровану аналітичну модель на основі попередньо сформульованого куба OLAP. Застосувавши алгоритм кластеризації до цієї структури, ми можемо розпізнати природні групування в даних, пропонуючи цінні перспективи, які допомагають зрозуміти складні виміри набору даних, що вивчається.

4 РЕЗУЛЬТАТИ

4.1 Інтерфейс системи

Перша взаємодія користувача з системою відбувається через інтерфейс, створений для оптимізації та забезпечення безперебійного зв’язку з внутрішнім сервером. Важливо постійно оновлювати цей інтерфейс, зберігаючи знайомі елементи та додаючи нові функції. Давайте глибше заглибимося в архітектуру та можливості мобільного додатку.

Після встановлення мобільного додатка користувачам одразу відкривається сторінка входу, і вони мають можливість перейти до сторінки реєстрації, як показано на малюнку 4.1. Додаток використовує метод авторизації маркерів JWT для входу. Цей метод, який зазвичай використовується на веб-сайтах і в програмах, покладається на веб-токен JSON (JWT) для обміну та перевірки даних користувача. JWT представляє стандартизований спосіб обміну інформацією між сторонами за допомогою компактних об’єктів JSON.

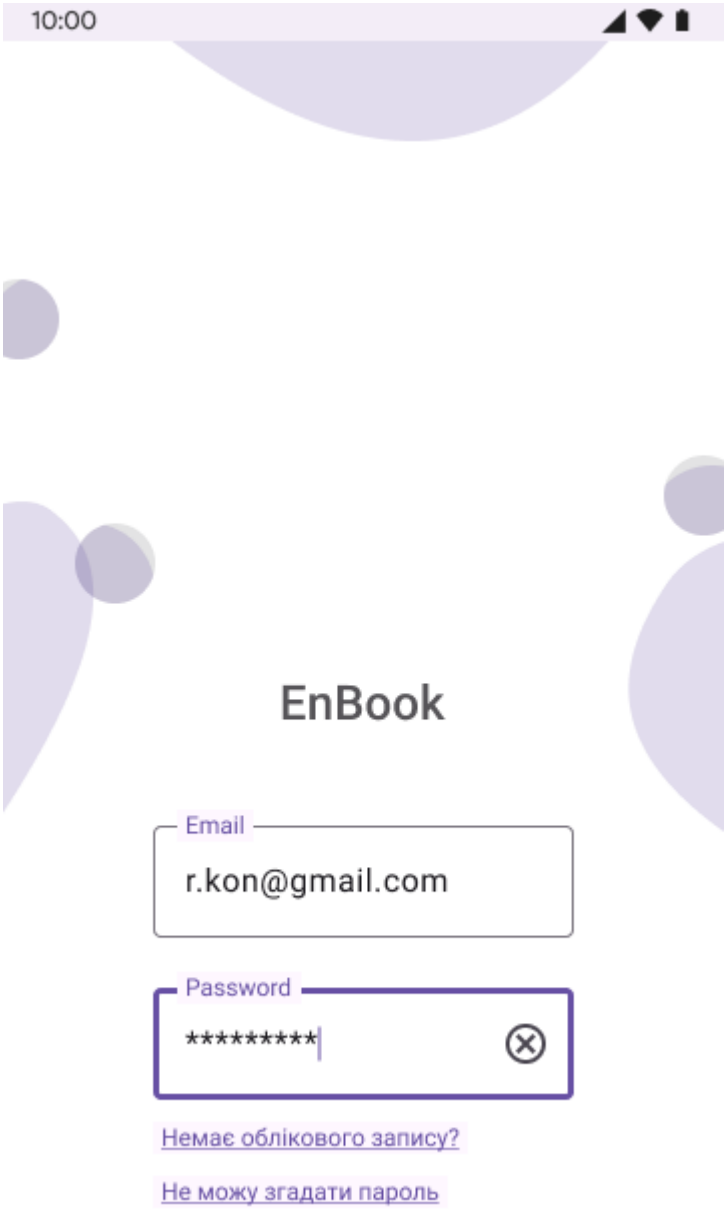

Рис. 4.1 екран авторизації користувача

За допомогою перевірки на основі маркерів JWT, коли користувач підтверджує свою особу за допомогою електронної пошти та пароля, маркер JWT формулюється та надсилається користувачеві. Потім клієнт зберігає цей маркер і приєднує його до кожного наступного запиту до сервера. Отримавши це, сервер оцінює автентичність маркера. У разі перевірки користувачеві надається доступ до потрібних ресурсів.

Токен JWT складається з трьох окремих компонентів: заголовка, корисного контенту та підпису. У заголовку зберігаються відомості про тип

маркера та метод шифрування для підпису. Корисний контент містить інформацію про користувача разом із додатковими даними, встановленими розробником програмного забезпечення. Підпис виникає в результаті об’єднання заголовка, корисного навантаження та конфіденційного ключа, який зберігається на сервері.

Перевірка на основі маркера JWT враховується у облікових даних безпеки. Враховуючи, що токен зберігається клієнтом і надсилається в зашифрованому вигляді, його можливе підроблення або модифікація зведено до мінімуму. Використання JWT не тільки полегшує навантаження на сервер (оскільки дозволяє уникнути зберігання сеансів на стороні сервера), але й відкриває шлях для масштабованості програми.

Після авторизації користувача відбувається перехід на головний екран, що представлено на рисунку 4.2. Головний екран демонструє слідуючу інформацію:

- навігаційне меню;
- розпочаті але не закінчені уроки;
- популярні словники зі словами для вивчення.

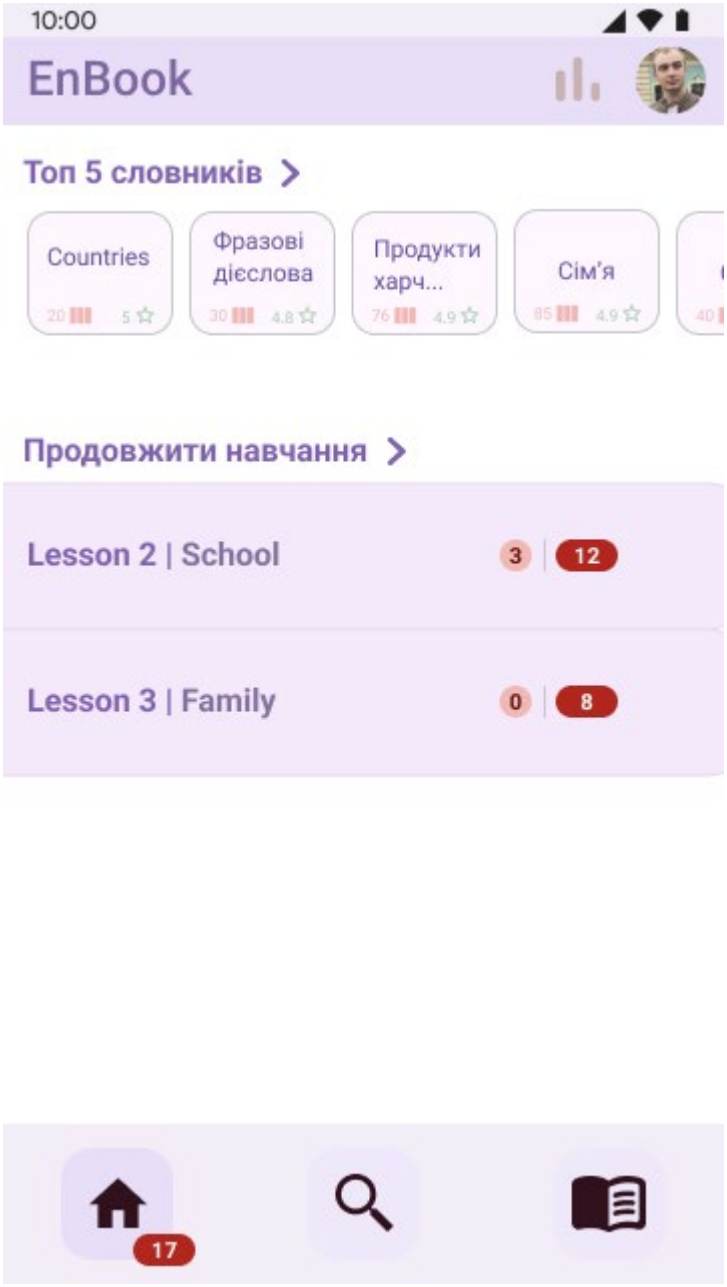

Рис. 4.2 Головний екран користувача

Далі розглянемо в практичний сценарій: навігація уроком, створеним викладачем на платформі. Цей урок є свідченням можливостей системи та педагогічного досвіду вчителя. У міру проходження вмісту користувачі ознайомлюються зі зручним та інтуїтивно зрозумілим інтерфейсом, розробленим для покращення досвіду навчання. Загальне відображення цього інтерфейсу, його функції та компонування, можна побачити на рисунку 4.3.

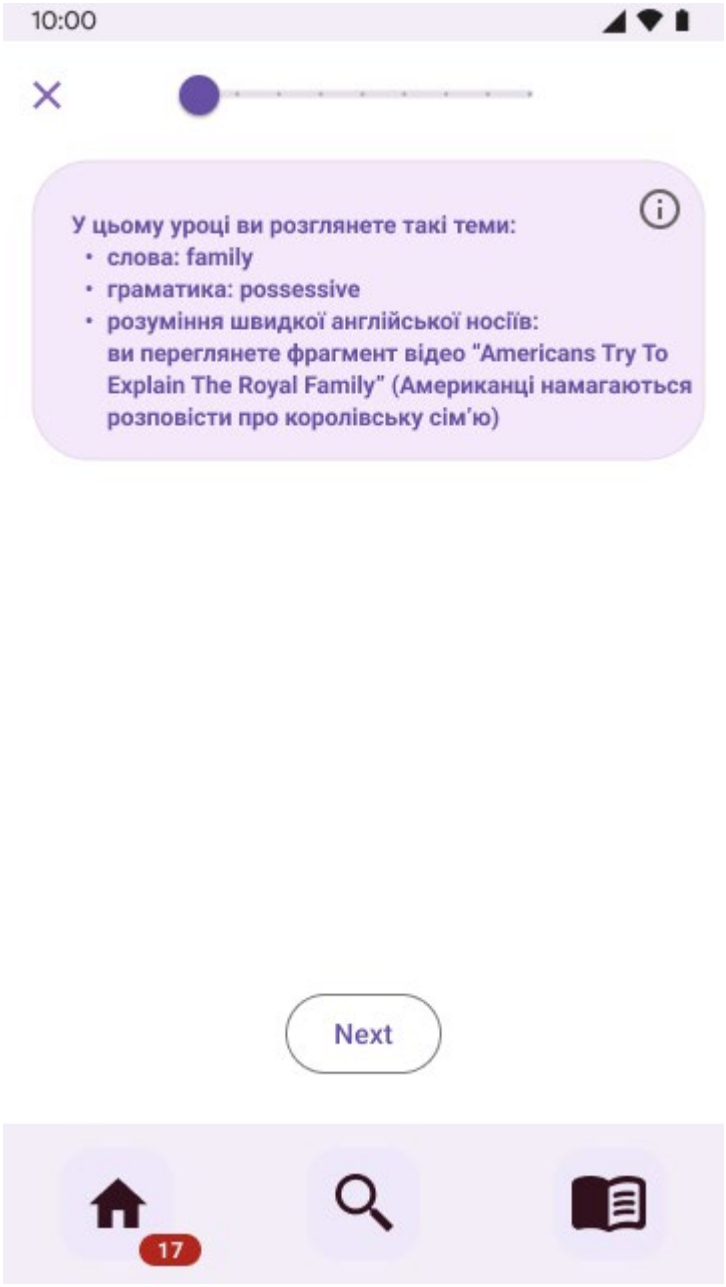

Рис. 4.3 Екран уроку (крок 1)

У рамках уроку зміст методично організовано в окремі сегменти або кроки. Кожен із цих кроків охоплює певний шматок матеріалу. Як показано на рисунку 4.3, початковий демонстраційний матеріал є текстовим абзацом, який служить інформативним повідомленням для скерування користувача. Цей формат забезпечує ясність і допомагає в систематичному прогресі змісту. Щоб перейти від цього моменту та перейти до наступного кроку уроку, користувачам пропонується натиснути кнопку «Next». Цей інтерактивний елемент забезпечує

плавний перехід, утримуючи зацікавленість учнів і гарантуючи, що вони засвоюють кожен сегмент інформації, перш ніж рухатися вперед.

Після початкової текстової інформації на уроці вводиться ще один формат інформативного контенту: відеопрезентація. Це мультимедійне доповнення слугує вдосконаленням процесу навчання, пропонуючи користувачам динамічний спосіб взаємодії та розуміння матеріалу. Візуальне представлення цього відеоконтенту можна переглянути на рисунку 4.4. Така інтеграція різноманітних форматів контенту не тільки збагачує освітній досвід, але й задовольняє різноманітні навчальні уподобання, гарантуючи, що всі користувачі, незалежно від того, віддають вони перевагу читанню чи візуальним демонстраціям, знайдуть матеріал доступним і привабливим.

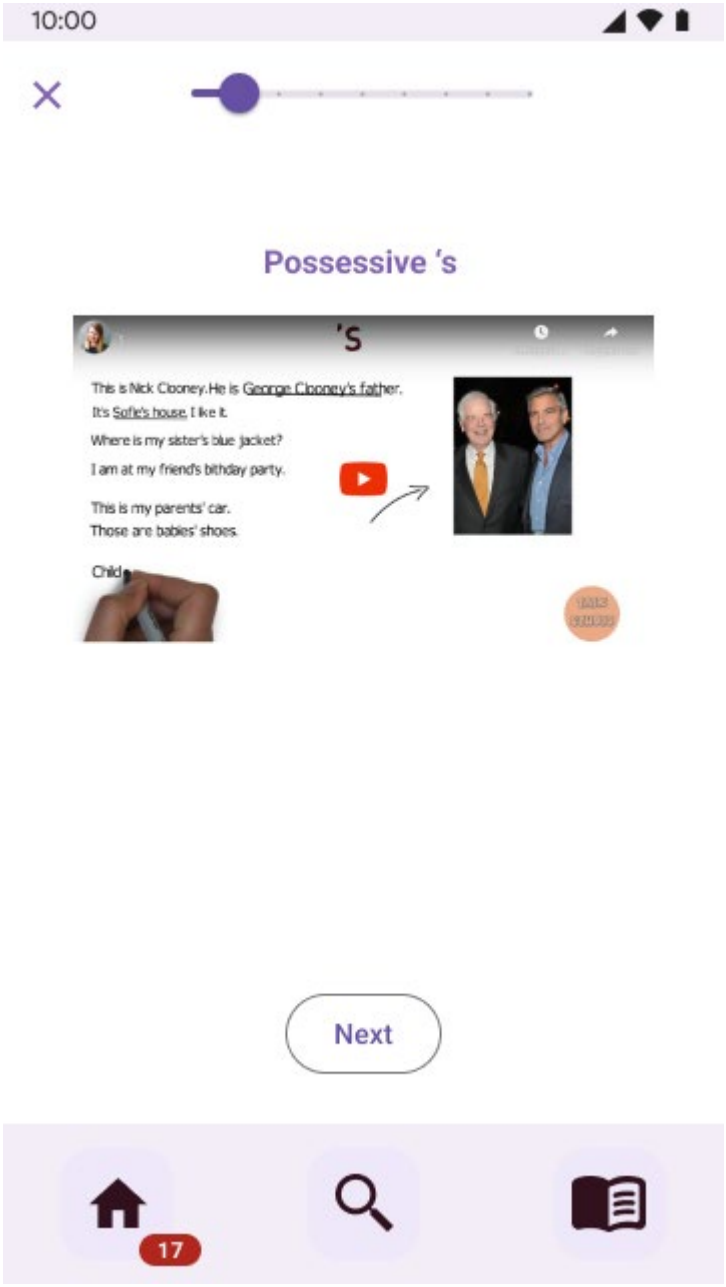

Рис. 4.4 Екран уроку з відеопрезентацією

Просуваючись далі до уроку, користувачі стикаються з більш інтерактивним сегментом: вікториною. Цей метод залучення пропонує практичний підхід до перевірки розуміння та закріплення матеріалу. На екрані представлена фотографія, яка проілюстрована на рисунку 4.5. Це зображення супроводжується набором варіантів або параметрів, які користувачі повинні оцінити та вибрати з них. Детальний макет і дизайн цього інтерфейсу вікторини, що висвітлює ці параметри, можна переглянути на рисунку 4.6.

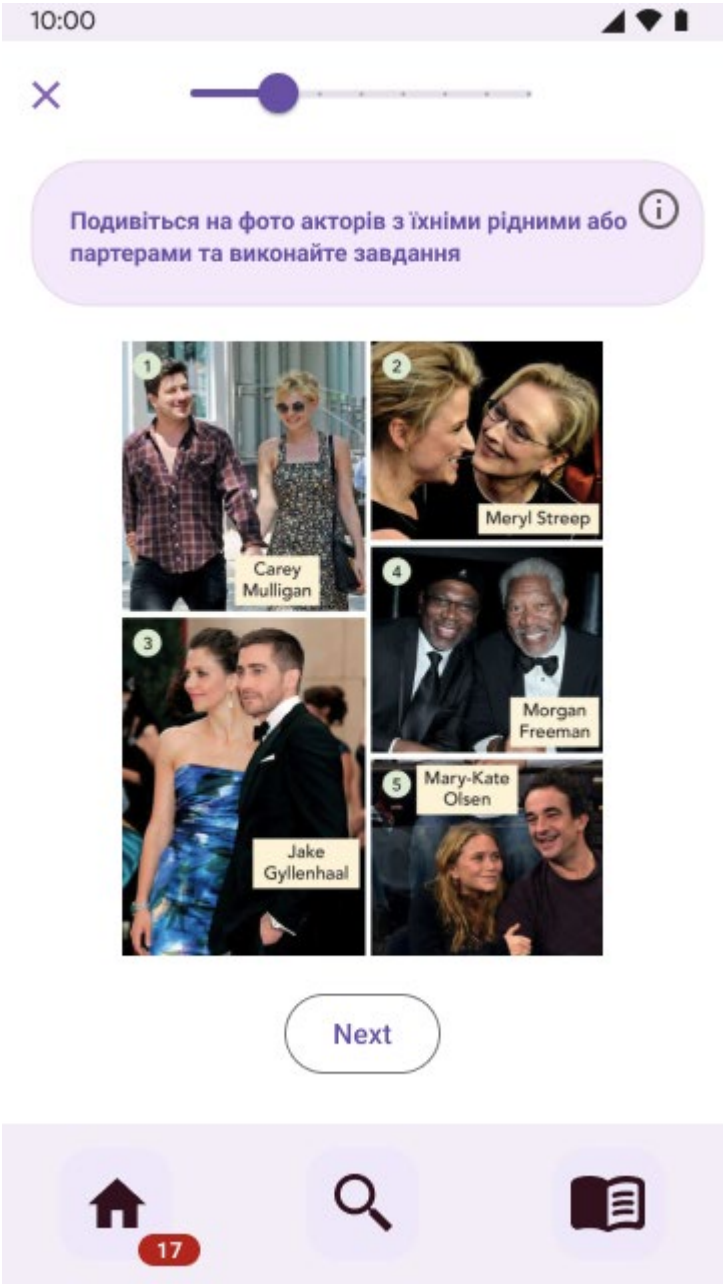

Рис. 4.5 Екран уроку з вступом до вікторини

Таке поєднання візуальних підказок та інтерактивних запитань не тільки забезпечує активну участь учнів, але й забезпечує негайний зворотний зв’язок щодо їхнього розуміння представленого матеріалу.

Рис. 4.6 Екран уроку з вікториною

Серед різноманітних завдань, доступних на платформі, особливо цікавим є вправа «вивчення слів». Ця вправа, розроблена для прискорення засвоєння словникового запасу, починається з орієнтованого на користувача підходу: людям пропонується відокремити знайомі їм слова від незнайомих, який представлено на рисунку 4.7. Цей процес вибору інтуїтивно зрозумілий – провевши пальцем вгору, користувачі вказують на бажання вивчити певне слово, тоді як пальцем вниз означає розпізнавання, минаючи слово.

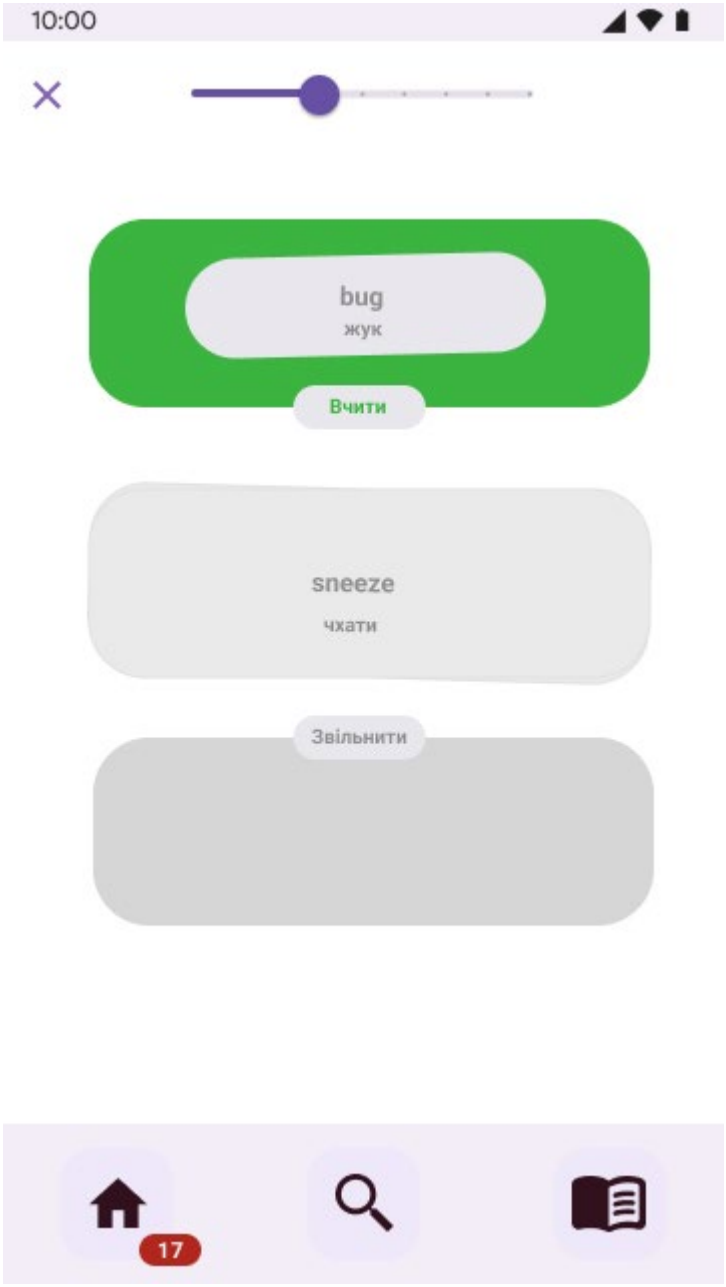

Рис. 4.7 Екран уроку з вправою “вивчення слів”

Після цього етапу вибору починається фактичне навчання, яке зображене на рисунку. Ось як алгоритм полегшує цей процес:

- Вибір слів: система генерує набір слів, що не перевищують десять, призначених для вивчення користувачем.
- Створення динамічного словника: слова, не обрані для негайного вивчення, каталогізуються в автоматизованому словнику, отриманому на основі уроку. Цей репозиторій виконує подвійну мету – він дозволяє

користувачам перекладати незнайомі слова або займатися вправою «заповнити порожні місця», додаючи пропущені літери в заданих словах.

- Ітераційні навчальні кола: процес навчання структурований у концентричні кола, кожне з яких служить рівнем майстерності. Користувачеві представлено три таких кола або фази. Успішне проходження всіх трьох без помилок означає, що слово було вивчено. Однак, якщо на будь-якому етапі станеться помилка, відповідне слово буде переміщено назад у згаданий раніше автоматично згенерований словник.
- Огляд після вивчення: слова, які не досягають майстерності після трьох кіл, поміщаються в цей спеціальний словник. Звідси користувачі мають можливість переглянути та опанувати їх у власному темпі.

По суті, ця діяльність «вивчення слів» завдяки своєму алгоритмічному дизайну та підходу, орієнтованому на користувача, забезпечує комплексний і гнучкий процес навчання. Це не тільки допомагає розширити словниковий запас, але й надає широкі можливості для повторення та закріплення.

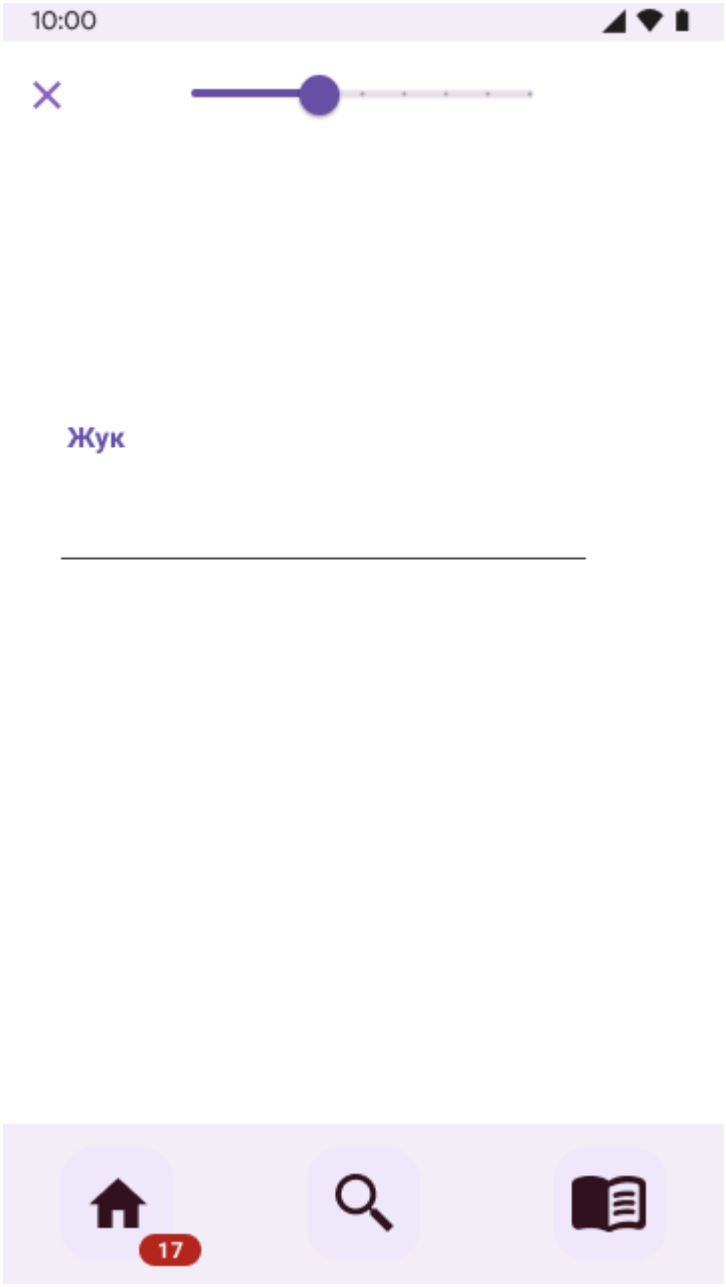

Рис. 4.8 Екран уроку з вправою “вивчення слів”

4.2 Результати аналітичного модуля

4.2.1 Naive Bayes Algorithm. Результатом розгортання структури NaivBaise представлена на рисунку 4.9.

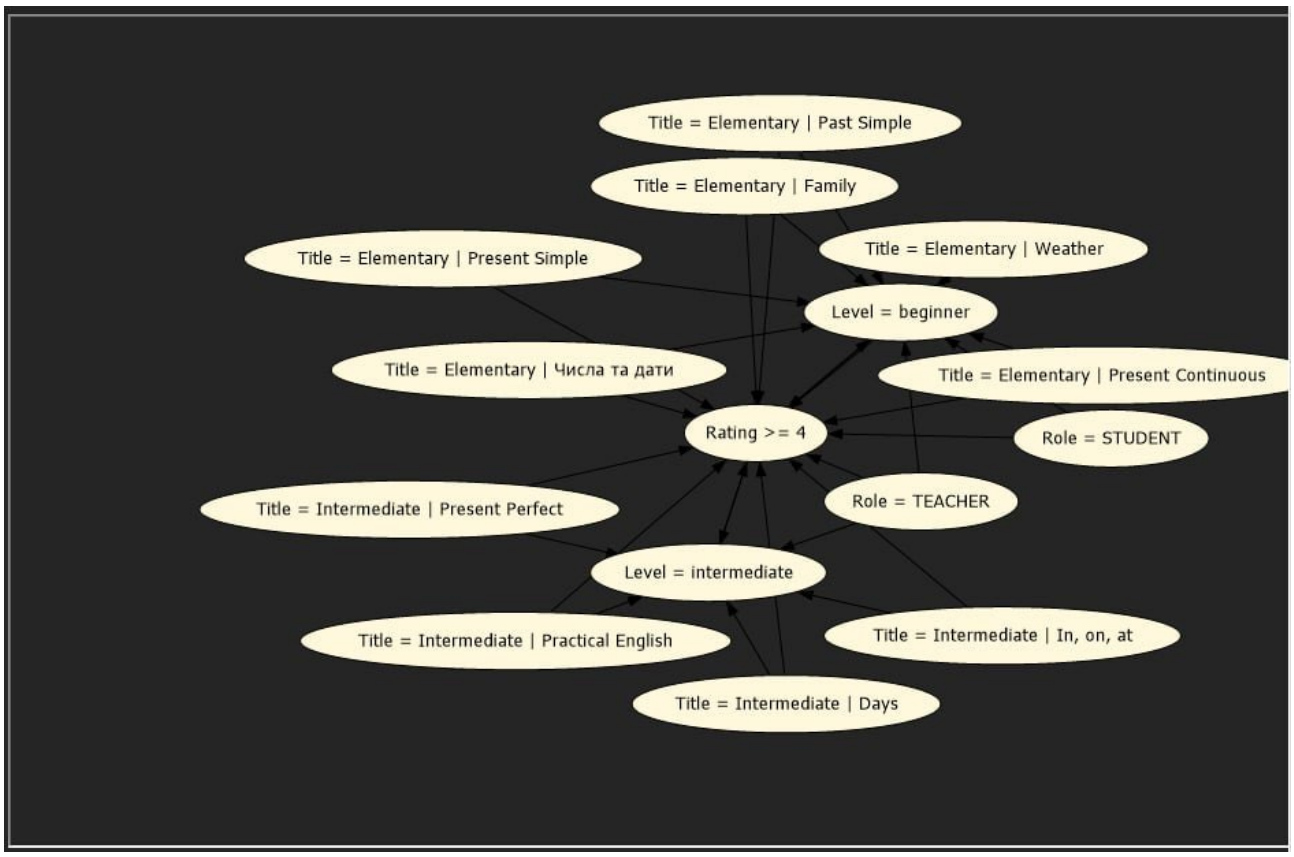

Рис. 4.9

4.2.3 Association Algorithm. Розширення структури ARule завершується компіляцією правил асоціації, як показано на рисунку 4.10. Ці правила супроводжуються двома критичними показниками: ймовірність і важливість. Імовірність відображає вірогідність того, що правило буде точним у прогнозованих сценаріях, тоді як важливість означає значимість або вагу правила в наборі даних. По суті, правила з вищими значеннями ймовірності та важливості вказують на сильнішу прогностичну силу та релевантність, припускаючи, що вони більш надійні для висновків або прийняття рішень на основі даних. Таке визначення пріоритетів допомагає зосередити аналітичні зусилля на найпотужніших і найдосконаліших правилах, тим самим підвищуючи ефективність і результативність процесу інтелектуального аналізу даних.

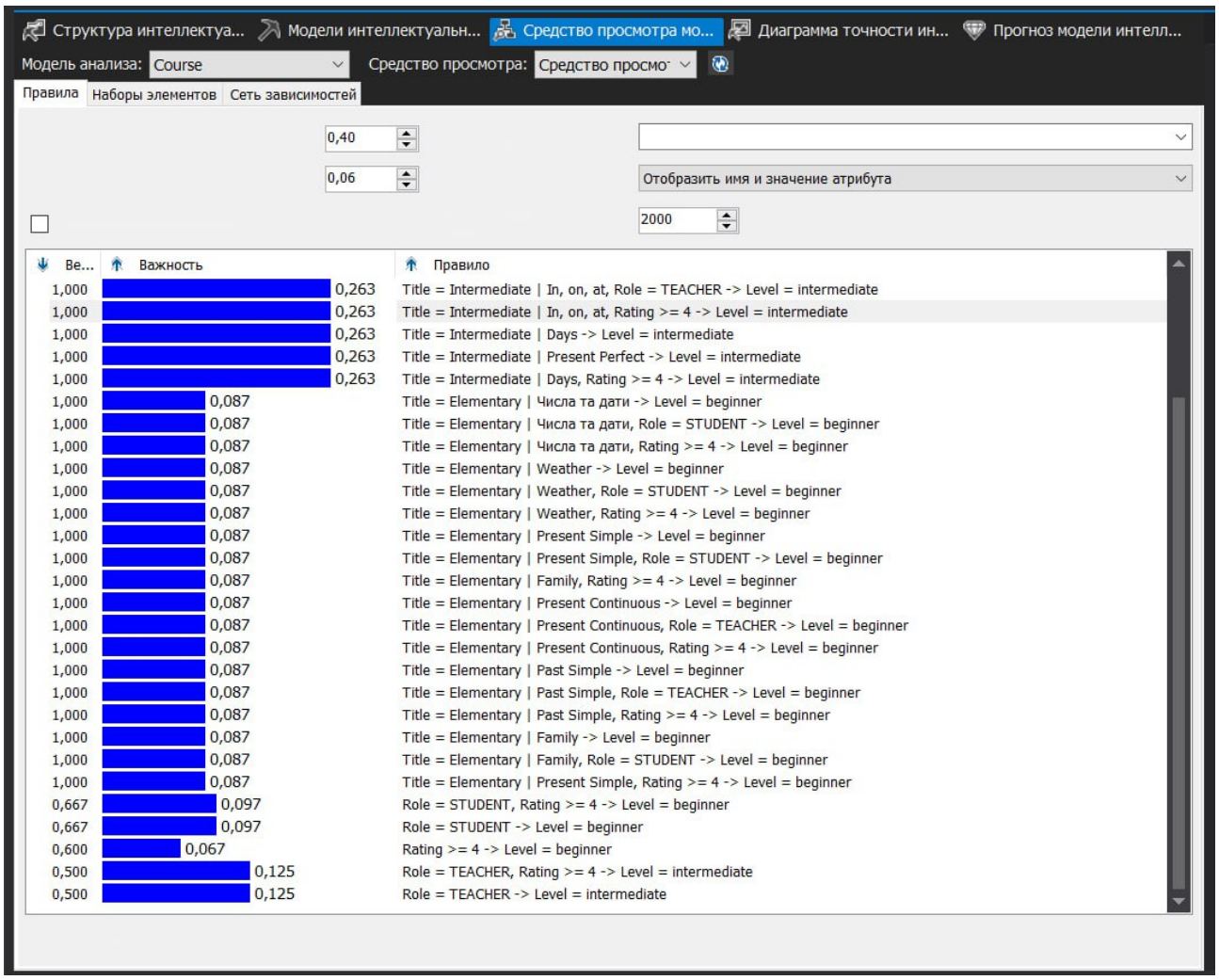

Рис. 4.10 Асоціативні правила структури ARule

Ці правила пов’язані з моделлю курсу з такими умовами, як назва вмісту, ролі (наприклад, ВЧИТЕЛЬ чи СТУДЕНТ) і рейтингові значення, які відповідають певним рівням знань (наприклад, початковий, середній). Чим вище значення важливості та ймовірності, тим більш значущим і надійнішим вважається правило.

Загалом структура забезпечує організоване уявлення про те, як різні умови пов’язані з рівнями кваліфікації, допомагаючи в інтерпретації даних і потенційно сприяючи процесу прийняття рішень, пов’язаних із розробкою курсу.

4.2.4 Clustering Algorithm. На рисунку 4.11 зображено структуру Cluster

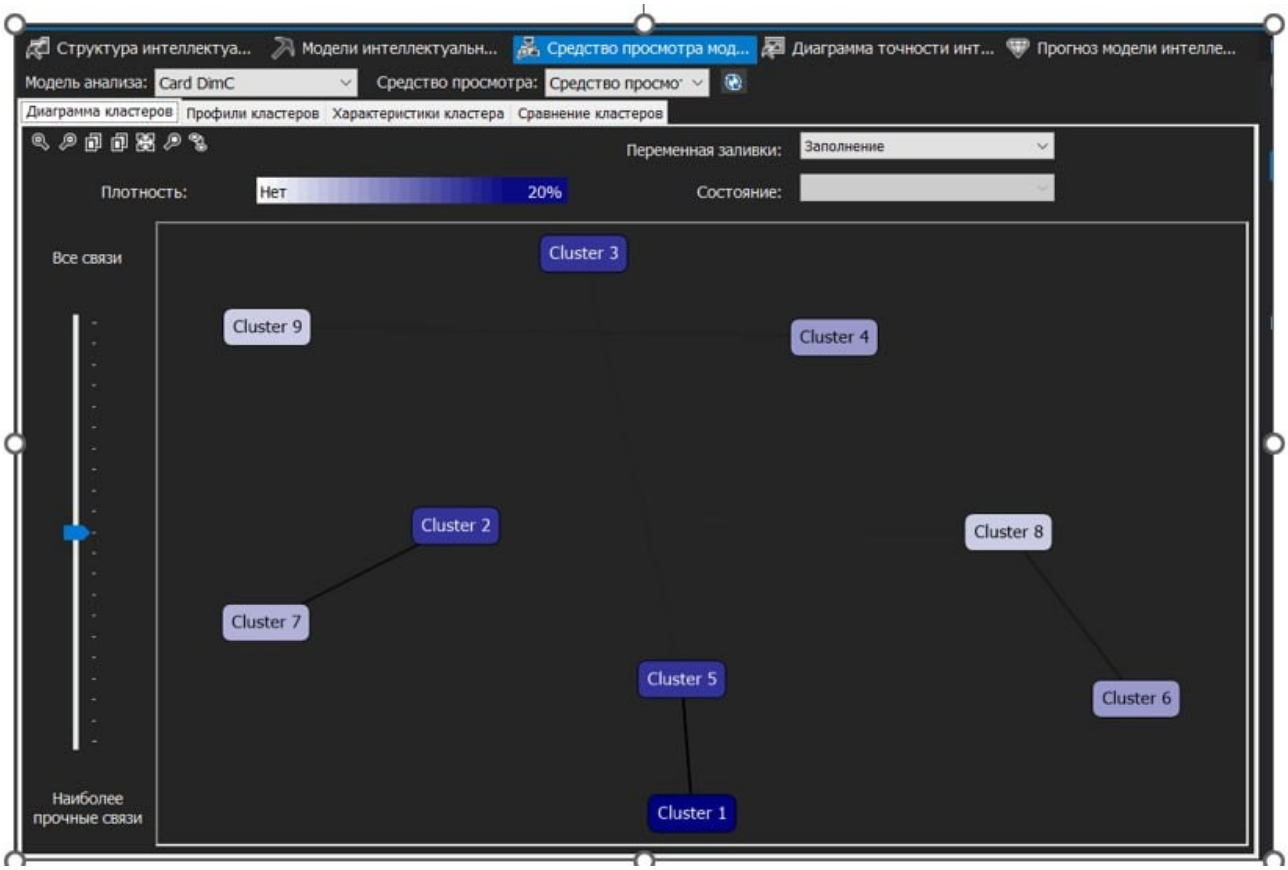

Рис. 4.11 Кластерна структура

Кластери здаються просторово розділеними, що вказує на чітке угруповання в проаналізованих даних.

Є лінії, що з’єднують деякі з кластерів, які можуть представляти зв’язки або тісні зв’язки між певними кластерами. Наприклад, «Кластер 2» з’єднаний з «Кластером 7» і «Кластером 5».

Загалом ця візуалізація кластеризації надає огляд того, як точки даних або випадки групуються на основі їх подібності, а зв’язки вказують на потенційні зв’язки між цими групами. Цей тип візуалізації може допомогти зрозуміти структуру та зв’язки в наборі даних.

ВИСНОВКИ

Підсумовуючи, цей проект є значним досягненням у розробці мобільного програмного забезпечення системи навчання англійської мови. У ході цієї роботи був застосований комплексний підхід, який передбачав використання різноманітних технологій та інструментів, таких як React Native, Java, Spring, MS SQL Server, OLAP та Data Mining. Систематичний аналіз аналогічних систем, їх переваг і недоліків допоміг закласти основу для цього інноваційного рішення.

Об'єктно-орієнтоване моделювання зіграло вирішальну роль у визначенні вимог до системи та операцій обробки, що зрештою проклало шлях до створення інформаційних і програмних компонентів.

Завершення цієї роботи завершилося презентацією інтерфейсу користувача мобільного додатку системи навчання англійської мови, результатами роботи аналітичного блоку та глибоким аналізом отриманих знань.

Рухаючись вперед, продовження розвитку цієї системи обіцяє отримати ще більше цінних знань, які можуть зробити значний внесок у сферу вивчення англійської мови та надати відчутну користь користувачам. Ця робота є свідченням потенціалу технологічних рішень у покращенні досвіду вивчення мови та отримання знань у професійному контексті.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Duolingo [Електронний ресурс] URL: https://uk.duolingo.com
2. Rosetta Stone [Електронний ресурс] URL: https://ru.rosettastone.com
3. Babbel [Електронний ресурс] URL: https://ua.babbel.com
4. Memrise [Електронний ресурс] URL: https://www.memrise.com
5. Oxford Online English [Електронний ресурс] URL: https://www.oxfordonlineenglish.com/home-mobile
6. VIPKid [Електронний ресурс] URL: https://www.vipkid.com/en-us
7. iTalki [Електронний ресурс] URL: https://www.italki.com/en
8. Tandem [Електронний ресурс] URL: https://www.tandem.net/ru/language-exchange/ukraine
9. Grammarly [Електронний ресурс] URL: https://www.grammarly.com
10. MondlyAR [Електронний ресурс] URL: https://www.mondly.com/ar
11. UML [Електронний ресурс] URL: https://www.lucidchart.com/pages/ru/uml
12. Діаграма прецедентів [Електронний ресурс] URL: https://www.wikidata.uk-ua.nina.az/%D0%94%D1%96%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%B0_%D0%BF%D1%80%D0%B5%D1%86%D0%B5%D0%B4%D0%B5%D0%BD%D1%82%D1%96%D0%B2.html
13. Діаграма станів [Електронний ресурс] URL: https://www.wikidata.uk-ua.nina.az/%D0%94%D1%96%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%B0_%D1%81%D1%82%D0%B0%D0%BD%D1%96%D0%B2.html
14. Логічна модель даних [Електронний ресурс] URL: https://www.wikiwand.com/uk/%D0%9B%D0%BE%D0%B3%D1%96%D1%87%D0%BD%D0%B0_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85

15. Microsoft SQL Server [Електронний ресурс] URL: https://www.microsoft.com/ru-ru/sql-server/sql-server-2019
16. Сховище даних [Електронний ресурс] URL: https://www.wikiwand.com/uk/%D0%A1%D1%85%D0%BE%D0%B2%D0%B8%D1%89%D0%B5_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85
17. Архітектура [Електронний ресурс] URL: https://wezom.com.ua/blog/arhitektura-programmnogo-obespecheniya
18. React Native [Електронний ресурс] URL: https://reactnative.dev/
19. Java [Електронний ресурс] URL: https://aws.amazon.com/ru/what-is/java/
20. Spring Framework [Електронний ресурс] URL: https://highload.today/spring-framework/
21. OpenAI API [Електронний ресурс] URL: https://openai.com/blog/openai-api
22. React [Електронний ресурс] URL: https://react.dev
23. IoC [Електронний ресурс] URL: https://spring-projects.ru/guides/lessons/lesson-2/
24. DI [Електронний ресурс] URL: https://highload.today/blogs/spring-framework/
25. AOP [Електронний ресурс] URL: https://foxminded.ua/ru/aspektno-orientirovannoe-programmirovanie
26. JDBC [Електронний ресурс] URL: https://proselyte.net/tutorials/jdbc/introduction/
27. ORM [Електронний ресурс] URL: https://www.hwlibre.com/uk/orm-object-relational-mapping/
28. Hibernate [Електронний ресурс] URL: https://hibernate.org/
29. JPA [Електронний ресурс] URL: https://spring.io/projects/spring-data-jpa
30. JDO [Електронний ресурс] URL: https://db.apache.org/jdo/

31. MVC Framework [Електронний ресурс] URL: https://www.cybermedian.com/what-is-mvc-framework/
32. JSP [Електронний ресурс] URL: https://metanit.com/java/javaee/3.1.php
33. Thymeleaf [Електронний ресурс] URL: https://www.thymeleaf.org/
34. Spring Security [Електронний ресурс] URL: https://spring.io/projects/spring-security
35. OLAP [Електронний ресурс] URL: https://support.microsoft.com/uk-ua/office/%D0%BE%D0%B3%D0%BB%D1%8F%D0%B4-%D0%BE%D0%BD%D0%BB%D0%B0%D0%B9%D0%BD%D0%BE%D0%B2%D0%BE%D1%97-%D0%B0%D0%BD%D0%B0%D0%BB%D1%96%D1%82%D0%B8%D1%87%D0%BD%D0%BE%D1%97-%D0%BE%D0%B1%D1%80%D0%BE%D0%B1%D0%BA%D0%B8-olap-15d2cdde-f70b-4277-b009-ed732b75fdd6