НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
Факультет інформаційних технологій

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ
Завідувач кафедри
Комп'ютерних наук
_____ Голуб Б.Л.

"02" червня 2025 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА
на тему
Програмне забезпечення веб орієнтованої системи для вивчення англійської мови
Спеціальність 121 – «Інженерія програмного забезпечення»

Гарант освітньої програми
К.т.н., доцент _____ Вайганг Г.О.

Керівник бакалаврської кваліфікаційної роботи

_____Степанов О. В._
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ _Дейнеко Андрій Андрійович_
(підпис) (ПІБ студента)

КИЇВ – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри
Комп'ютерних наук
_____ Голуб Б.Л.

"16" грудня 2024 р.

ЗАВДАННЯ
на виконання бакалаврської кваліфікаційної роботи студенту
_____ Дейнеко Андрію Андрійовичу _____
(прізвище, ім'я, по батькові)

Спеціальність 121 – «Інженерія програмного забезпечення»
Тема бакалаврської кваліфікаційної роботи Програмне забезпечення веб орієнтованої системи для вивчення англійської мови
затверджена наказом ректора НУБіП України від "16" грудня 2024р. №2249С

Термін подання завершеної роботи на кафедру 2025. 05. 25
(рік, місяць, число)

Вихідні дані до бакалаврської кваліфікаційної роботи:
опис програмного забезпечення

Перелік питань, які потрібно розробити
- системний аналіз предметної області;
- проєктування інформаційного та програмного забезпечення;
- розробка інформаційного та програмного забезпечення;
- рекомендації щодо впровадження та експлуатації системи;

Дата видачі завдання "16" грудня 2025 р.
Керівник бакалаврської кваліфікаційної роботи
_____ _____ Степанов О.В.
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Завдання прийняв до виконання _____ Дейнеко А.А.
(підпис) (ПІБ студента)

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

CEFR – Common European Framework of Reference

CSS – Cascading Style Sheets

CTE – Common Table Expression

ER-model – Entity-Relationship model

HTML – HyperText Markup Language

HTTPS – Hypertext Transfer Protocol Secure

IELTS – International English Language Testing System

JSON – JavaScript Object Notation

MVC – Model-View-Controller

MVT – Model-View-Template

ORM – Object-Relational Mapping

SSL – Secure Sockets Layer

SQL – Structured Query Language

TOEFL – Test of English as a Foreign Language

UML – Unified Modeling Language

XML – eXtensible Markup Language

XMI – XML Metadata Interchange

OOSE – Object-Oriented Software Engineering

OMT – Object Modeling Technique

RUP – Rational Unified Process

WCAG – Web Content Accessibility Guidelines

НФ – Нормальна форма

ВСТУП

У сучасному глобалізованому світі англійська мова стала необхідним інструментом для спілкування, освіти, кар'єрного зростання та розширення культурного обміну. Це універсальний засіб комунікації, що поєднує людей з різних країн і допомагає долати виклики, пов’язані з професійною діяльністю, навчанням і повсякденним життям. [14]

З огляду на стрімкий розвиток цифрових технологій та збільшення попиту на дистанційне навчання, зростає потреба у створенні гнучких, зручних і доступних веб-орієнтованих систем, що підтримуватимуть самостійне вивчення англійської мови. Такі системи повинні надавати користувачам інтерактивні навчальні матеріали, інструменти для тестування знань та засоби для відстеження прогресу.

Метою бакалаврської роботи є розробка функціональної платформи для вивчення англійської мови з використанням сучасних веб-технологій. Для досягнення цієї мети планується виконати такі завдання: провести аналіз існуючих програмних рішень у цій галузі та визначити їхні переваги і недоліки; змоделювати предметну область з урахуванням потреб потенційних користувачів; обґрунтувати вибір інструментів розробки та архітектури; розробити ключові функціональні модулі системи; протестувати працездатність продукту; сформулювати рекомендації щодо впровадження та розвитку платформи.

На основі зібраної інформації було спроєктовано структуру даних системи та створено UML-діаграми, що відображають її архітектуру. Для реалізації програмного забезпечення використано фреймворк Django (Python) для бекенду, систему керування базами даних PostgreSQL, мова запитів SQL, HTML/CSS/JavaScript для фронтенду, а також бібліотеки Bootstrap для стилізації інтерфейсу. У межах реалізації створено модулі для реєстрації та авторизації користувачів, управління матеріалами, проходження тестів, збереження результатів і динаміки навчання.

Структура роботи охоплює чотири основні розділи.

У розділі «Системний аналіз предметної області» розглянуто актуальність дослідження, визначено цільову аудиторію, сформульовано вимоги до системи, а також проведено аналіз аналогічних програмних рішень для виявлення їхніх сильних і слабких сторін. Це дозволило сформувати основу для проєктування системи з урахуванням потреб користувачів.

У розділі «Проєктування інформаційного та програмного забезпечення» представлено UML-діаграми, логічну модель бази даних, обґрунтовано вибір архітектури та інструментів для реалізації. Розроблено структуру даних і визначено основні компоненти системи, зокрема модулі для керування навчальними матеріалами, тестування, обліку користувачів.

У розділі «Розробка інформаційного та програмного забезпечення» описано реалізацію ключових функціональних модулів: інтерфейс користувача, обробка даних, робота з базою, реєстрація, авторизація, управління матеріалами та тестування. Проведено тестування системи для перевірки її повноти та стабільності.

У розділі «Рекомендації щодо впровадження та експлуатації системи» наведено технічні характеристики продукту та результати тестування.

Окремі результати дослідження представлені у формі тез, опублікованих у збірнику наукових праць VII Всеукраїнської науково-практичної конференції студентів і аспірантів «Теоретичні та прикладні аспекти розробки комп’ютерних систем 2025», що відбулася 23 травня 2025 року в місті Києві. У публікації представлено концепцію, технологічну базу та ключові рішення, реалізовані в межах проєкту. [15]

Таким чином, дана робота реалізує повний цикл створення прикладного програмного забезпечення — від аналізу предметної області, проєктування та реалізації до тестування і апробації функціональної веб-системи, орієнтованої на підтримку процесу вивчення англійської мови.

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Опис предметної області

Загальна характеристика предметної області

Вивчення англійської мови є надзвичайно важливим процесом у сучасному глобалізованому світі, де англійська мова є міжнародною мовою бізнесу, науки, техніки та міжкультурного спілкування.

Предметна область вивчення англійської мови охоплює широкий спектр лінгвістичних, педагогічних і психологічних аспектів, спрямованих на розвиток мовних компетенцій. Ці компетенції традиційно поділяються на чотири основні навички: аудіювання (слухове сприйняття), говоріння (усне мовлення), читання та письмо. Кожна з цих навичок вимагає специфічних методів викладання та оцінювання, які враховують різні рівні володіння мовою, визначені Загальноєвропейськими рекомендаціями з мовної освіти (ЗЄР) - від початкового рівня A1 до професійного рівня C2. Ефективне вивчення мови вимагає системного підходу, регулярної практики та безперервного зворотного зв'язку, що дозволяє учням відстежувати свій прогрес і відповідно коригувати навчальний процес.

Цільова аудиторія та її потреби

Цільова аудиторія системи вивчення англійської мови дуже різноманітна і включає учнів різного віку, рівня освіти та професійної орієнтації.

Серед основних категорій користувачів - школярі та студенти, які вивчають англійську мову в рамках навчальної програми; професіонали, яким англійська необхідна для кар'єрного зростання; особи, які готуються до міжнародних іспитів, таких як TOEFL, IELTS або Cambridge English; а також ті, хто вивчає мову для особистого розвитку або подорожей. Кожна з цих груп має специфічні потреби та мотивації, які впливають на вибір навчальних матеріалів та методів викладання.

Вивчення англійської мови є особливо актуальним для українців в умовах

євроінтеграції та розширення міжнародного співробітництва, що створює додатковий попит на ефективні інструменти вивчення мови, адаптовані до специфічних потреб україномовних користувачів.

Проблеми традиційного вивчення мови

Традиційні підходи до вивчення англійської мови стикаються з низкою суттєвих обмежень, які знижують їхню ефективність у сучасних умовах.

Серед ключових проблем - обмежена доступність якісної освіти через високу вартість курсів та приватних репетиторів, географічні бар'єри та жорсткий розклад занять, що ускладнює інтеграцію навчання у повсякденне життя. Групові заняття часто не враховують індивідуальні особливості учнів, такі як початковий рівень знань, темп навчання та конкретні цілі вивчення мови.

Самостійне навчання пропонує більшу гнучкість, але йому часто бракує структурованого підходу, постійного зворотного зв'язку та чіткої системи відстеження прогресу, що призводить до зниження мотивації та ефективності навчання. Ці проблеми особливо актуальні для українських користувачів, оскільки більшість міжнародних навчальних платформ не враховують специфічні труднощі, з якими стикаються україномовні студенти при вивченні англійської мови.

Роль технологій у вирішенні проблем мовної освіти

Сучасні технології відкривають нові можливості для подолання традиційних обмежень у вивченні англійської мови.

Веб-платформи для вивчення мови надають доступ до якісних навчальних матеріалів незалежно від місця та часу, значно розширюючи можливості для самоосвіти. Інтерактивні технології дозволяють створювати адаптивні навчальні системи, які автоматично підлаштовуються під рівень знань, темп навчання та індивідуальні потреби кожного користувача.

Мультимедійні можливості сучасних платформ пропонують різноманітні формати контенту (текст, аудіо, відео), які сприяють всебічному розвитку всіх мовних навичок. Автоматизовані системи тестування та оцінювання забезпечують миттєвий зворотній зв'язок, дозволяючи учням відстежувати свій

прогрес і відповідно коригувати навчальний процес. Ці технологічні рішення безпосередньо пов'язані з проблемами, описаними в попередньому пункті, і покликані ефективно їх вирішувати.

Інноваційні підходи до вивчення мови

Розроблена веб-платформа для вивчення англійської мови реалізує інноваційні підходи, що поєднують кращі практики традиційної педагогіки з можливостями сучасних технологій.

Система базується на комунікативному підході, який ставить практичне використання мови в реальних життєвих ситуаціях в центр навчального процесу - відповідно до потреб цільової аудиторії, описаних вище. Платформа використовує принципи мікронавчання, розбиваючи контент на невеликі, легко засвоювані одиниці, що дозволяє учням інтегрувати навчальні сесії у свій повсякденний розпорядок дня.

Гейміфікація навчального процесу за допомогою систем досягнень, рівнів та винагород підвищує мотивацію та залученість користувачів. Адаптивні алгоритми аналізують прогрес кожного учня і автоматично підлаштовують складність і тип завдань, забезпечуючи оптимальний рівень виклику. Ці інноваційні підходи безпосередньо пов'язані з технологічними можливостями, описаними в попередньому пункті, і спрямовані на повне використання їхнього потенціалу для ефективного вивчення мови.

Специфіка вивчення англійської мови для українців

Вивчення англійської мови українськими користувачами має свої особливості, які вимагають індивідуального підходу до розробки навчальних матеріалів та методик.

Українські студенти стикаються з особливими фонетичними проблемами, зокрема, з вимовою специфічних англійських звуків, яких немає в українській мові (наприклад, міжзубних звуків [θ] та [ð], а також звука [w]). Граматичні відмінності між мовами, включаючи систему часів, вживання артиклів, прийменників та порядок слів, також становлять особливі труднощі. Лексична інтерференція між українською, російською та англійською мовами може

призвести до поширених помилок у використанні лексики.

Культурні відмінності та реальний контекст ще більше впливають на те, як учні сприймають і розуміють мовний матеріал. Розроблена платформа враховує ці специфічні аспекти, пропонуючи цілеспрямовані вправи та пояснення, які вирішують типові труднощі, з якими стикаються українські студенти, що безпосередньо відповідає потребам цільової аудиторії, описаним вище.

Комплексне вирішення проблем мовної освіти

Розроблена веб-платформа для вивчення англійської мови пропонує комплексне вирішення проблем, з якими стикаються користувачі традиційної мовної освіти.

Система забезпечує доступ до якісного викладання мови через онлайн-формат, дозволяючи користувачам навчатися в будь-який зручний час і з будь-якого місця, де є доступ до Інтернету. Персоналізація процесу навчання досягається завдяки адаптивним алгоритмам, які враховують рівень знань, темп навчання та конкретні цілі кожного користувача. Структурований підхід реалізується через чітку організацію матеріалів відповідно до рівнів Загальноєвропейських рекомендацій з мовної освіти та тематичних модулів.

Інтегрована система тестування та відстеження прогресу забезпечує регулярний зворотній зв'язок, що підтримує мотивацію та підвищує ефективність навчання. Адаптація до потреб українських користувачів здійснюється за допомогою спеціально розробленого контенту, який враховує типові труднощі, з якими стикаються україномовні студенти. Таким чином, платформа об'єднує всі інноваційні підходи та технологічні рішення, описані в попередніх параграфах, утворюючи цілісну систему для ефективного вивчення англійської мови.

1.2. Аналіз вимог до програмної системи

Функціональні вимоги

Функціональні вимоги є основою успішної розробки програмного забезпечення та системи. Вони точно визначають, що повинен робити продукт, щоб відповідати потребам користувачів і бізнесу. Визначаючи функції та поведінку системи, функціональні вимоги гарантують, що кожна функція відповідає очікуванням користувачів і цілям проекту. Проекти загрожують дорогими переглядами, затримками у термінах і незадоволеними зацікавленими сторонами без чітких, чітко визначених функціональних вимог. [8]

Реєстрація та авторизація користувачів

Система має забезпечувати надійний механізм реєстрації нових користувачів та авторизації існуючих. Процес реєстрації повинен бути інтуїтивно зрозумілим, збираючи необхідну інформацію для персоналізації навчання: ім'я, електронну пошту, пароль та рівень володіння мовою. Авторизація має підтримувати стандартний вхід через email/пароль.

Система повинна підтримувати різні ролі користувачів (студенти, редактори контенту, адміністратори) з відповідними рівнями доступу. Профіль користувача має містити особисту інформацію, історію навчання, досягнення та статистику прогресу, з можливістю редагування та налаштування параметрів приватності.

Управління навчальними матеріалами

Платформа має надавати інструменти для створення, редагування та організації різноманітних навчальних матеріалів. Інтерфейс для викладачів повинен бути зручним, не вимагати спеціальних технічних знань, підтримувати форматування тексту, вставку медіа-файлів та створення інтерактивних елементів.

Навчальні матеріали мають бути логічно організовані за рівнями складності, темами та типами мовних навичок, з підтримкою ієрархічної структури: курси, модулі, уроки, вправи. Система повинна забезпечувати

можливість персоналізації матеріалів для різних рівнів та стилів навчання, а також підтримувати версійність для відстеження змін.

Проходження навчальних матеріалів

Інтерфейс для проходження матеріалів має бути інтуїтивним, естетично приємним та адаптивним до різних пристроїв. Користувач повинен мати можливість переглядати доступні курси, отримувати інформацію про їх зміст та складність, а також отримувати персоналізовані рекомендації.

Процес навчання має бути структурованим, з чітким відображенням прогресу та можливістю повернення до попередніх матеріалів. Система повинна забезпечувати інтерактивність через різноманітні вправи з миттєвим зворотним зв'язком, адаптуючись до темпу навчання та особливостей користувача.

Тестування знань

Система має підтримувати різні типи тестів: діагностичні, формативні та сумативні, охоплюючи всі аспекти мовної компетенції. Тести повинні включати різноманітні типи завдань, відповідні для оцінювання різних навичок, з автоматичною перевіркою де можливо та інструментами для ручної перевірки складніших завдань.

Після проходження тесту користувач має отримувати детальний зворотний зв'язок з поясненнями помилок та рекомендаціями. Система повинна підтримувати різні формати тестування та забезпечувати об'єктивність через випадковий вибір запитань з банку завдань.

Відстеження прогресу

Платформа має надавати інструменти для відстеження та аналізу прогресу навчання. Користувачі повинні бачити наочну інформацію про свій прогрес у різних аспектах: загальний прогрес у курсі, розвиток окремих навичок, засвоєння лексики та граматики.

Система має відстежувати динаміку навчання, порівнюючи поточні результати з попередніми, та надавати рекомендації на основі аналізу прогресу. Для викладачів мають бути доступні інструменти аналізу прогресу окремих користувачів та груп для виявлення тенденцій та оптимізації навчального

процесу.

Нефункціональні вимоги

Нефункціональні вимоги або NFR (non-functional requirements) – це набір специфікацій, які описують робочі можливості та обмеження системи та намагаються покращити її функціональність. Це в основному вимоги, які визначають, наскільки добре працюватиме продукт якщо врахувати, наприклад, швидкість, безпеку, надійність, цілісність даних тощо. [22]

Продуктивність

Система має забезпечувати високу швидкодію з мінімальним часом відгуку. Для більшості операцій час відгуку не повинен перевищувати 1-2 секунди, а для складніших операцій система має надавати зворотний зв'язок про процес виконання.

Платформа повинна ефективно використовувати ресурси сервера та клієнта, з особливою увагою до оптимізації для мобільних пристроїв. Система має зберігати прийнятну швидкодію при збільшенні навантаження через кешування, балансування навантаження та оптимізацію запитів до бази даних.

Безпека даних

Система має забезпечувати надійну автентифікацію користувачів через сильні паролі, можливість двофакторної автентифікації та обмеження спроб входу. Вся комунікація між клієнтом та сервером повинна бути зашифрована за допомогою HTTPS, а чутливі дані мають зберігатися в захищеному вигляді.

Платформа повинна реалізувати управління доступом на основі ролей та забезпечувати захист від поширених типів атак через параметризовані запити, валідацію вхідних даних та інші механізми безпеки. Система має вести журнали критичних дій для аудиту та моніторингу безпеки.

Масштабованість

Система повинна підтримувати горизонтальне масштабування через додавання нових серверів для розподілу навантаження. Архітектура має бути модульною, з можливістю незалежного масштабування окремих компонентів.

Платформа має ефективно працювати з великими обсягами даних через

оптимізацію бази даних, кешування на різних рівнях та автоматичне масштабування ресурсів залежно від навантаження.

Зручність використання

Інтерфейс системи має бути інтуїтивно зрозумілим, логічним та послідовним, з чіткою навігацією та структурою. Дизайн повинен бути адаптивним для різних пристроїв та доступним для користувачів з особливими потребами відповідно до стандартів WCAG.

Система має мінімізувати когнітивне навантаження на користувача через простий інтерфейс з чіткою візуальною ієрархією, забезпечувати ефективний зворотний зв'язок на дії користувача та можливість персоналізації інтерфейсу. Платформа повинна ефективно обробляти помилки, надаючи зрозумілі повідомлення та допомогу користувачам через контекстні підказки, навчальні матеріали та технічну підтримку.

1.3. Моделювання предметної області

Вибір інструменту для моделювання

Draw.io (diagrams.net)— це онлайн-програмне забезпечення для настільних комп’ютерів із відкритим кодом. Це програмне забезпечення для створення блок-схем і схем, розроблене відповідно до сучасних обов’язків і чутливості професіоналів. Крім того, ця програма може справити на користувачів гарне враження завдяки своєму інтуїтивно зрозумілому інтерфейсу, який дозволяє їм розміщувати свої дані в більш зручному вигляді. Це тому, що його інтерфейс містить параметри та інструменти, які легко доступні та зрозумілі користувачам будь-якого рівня. Крім того, можна очікувати, що програма для створення блок-схем може бути універсальною програмою. Тому що окрім того, що користувачі можуть використовувати онлайн і офлайн, що робить Draw.io безкоштовним інструментом, він також постачається з різними шаблонами та макетами для будь-яких художніх вимог, які можуть знадобитися користувачеві. [23]

Enterprise Architect — це професійний інструмент моделювання, який надає повний набір можливостей для проектування та розробки програмного забезпечення. Він підтримує всі типи UML-діаграм відповідно до стандарту UML 2.5, а також інші методології, такі як BPMN, SysML, TOGAF. Завдяки цьому його можна застосовувати як для класичного об’єктно-орієнтованого проектування, так і для бізнес-аналізу чи системного інжинірингу. Інструмент дозволяє детально налаштовувати елементи діаграм, що робить його гнучким для різних сфер і потреб.

Крім моделювання, Enterprise Architect підтримує генерацію коду та зворотне проектування, охоплюючи широкий спектр мов програмування. Він також містить функції для управління вимогами, тестування, створення документації та звітів, що робить його корисним для команд, які працюють над великими та складними проектами. Однак через складний інтерфейс і високу вартість ліцензії цей інструмент може бути менш зручним для індивідуальних розробників або невеликих проєктів.

Visual Paradigm — це професійний інструмент для моделювання та проєктування програмного забезпечення, який поєднує широкі функціональні можливості з інтуїтивним інтерфейсом. Його головною перевагою є інтеграція з популярними IDE, такими як Eclipse, IntelliJ IDEA та NetBeans, що дозволяє розробникам працювати з моделями безпосередньо в середовищі кодування. Інструмент підтримує командну роботу, контроль версій і спільний доступ до моделей, що робить його зручним для великих проєктних команд.

Visual Paradigm також вирізняється розширеними можливостями для автоматичної генерації технічної документації, створення діаграм та підтримки бізнес-моделювання (BPMN, DFD тощо). Це спрощує аналіз і документування систем, а також забезпечує актуальність проєктної інформації. Водночас, інструмент вимагає значних ресурсів системи та має високу вартість ліцензії, що може бути перешкодою для невеликих проєктів або індивідуальних користувачів. [1]

Обґрунтування вибору інструменту моделювання

Для моделювання системи вивчення англійської мови було обрано інструмент **Draw.io** після аналізу альтернативних варіантів. У порівнянні з більш складними і платними рішеннями, такими як Visual Paradigm та Enterprise Architect, він виявився найбільш зручним та доступним. Інструмент забезпечує усі необхідні можливості для створення діаграм, має простий інтерфейс, не потребує встановлення і дозволяє легко зберігати й обмінюватися результатами роботи. Це особливо важливо для швидкої візуалізації структури системи та ефективної командної роботи.

У процесі моделювання було визначено ключові елементи майбутньої системи, їхні ролі та взаємозв'язки. Основна увага приділялася структурі навчального контенту, його класифікації за темами та рівнями складності, організації користувачів і взаємодії між ними. Важливими аспектами стали збереження авторства, можливість керування публікацією матеріалів, підтримка додаткових ресурсів та логічна побудова структури, яка дозволяє ефективно організувати навчальний процес і забезпечити зручний доступ до матеріалів.

1.4. Огляд інформаційних джерел та існуючих рішень

Duolingo — це одна з найвідоміших платформ для вивчення мов, яка поєднує освітній процес із гейміфікацією. Вона побудована на принципі коротких, інтерактивних уроків, які охоплюють ключові аспекти мови: лексику, граматику, аудіювання, читання та вимову. Система адаптує складність завдань під рівень користувача, завдяки чому процес навчання залишається ефективним і персоналізованим. Уроки розроблені як ігрові рівні з системою очок, досягнень, лінготів і щоденних цілей, що підтримує мотивацію користувача. [19]

Платформа активно використовує соціальні механізми: змагання з іншими користувачами, прогрес друзів та командні завдання. Duolingo також інтегрує машинне навчання для покращення навчального контенту на основі статистики користувачів. Незважаючи на простоту та привабливість, сервіс має недоліки: слабке занурення в граматику та обмежені можливості для розвитку розмовної

мови. Проте Duolingo залишається потужним інструментом для початкового та регулярного самостійного вивчення мов завдяки зручності, адаптивності й захоплюючому формату.

BBC Learning English — це освітній проєкт BBC World Service, заснований у 1943 році, який надає безкоштовні ресурси для вивчення англійської мови. Сервіс пропонує інтерактивні курси, відео, подкасти та вправи, охоплюючи різні аспекти мови: граматику, вимову, лексику та розуміння на слух. Матеріали розроблені для учнів різних рівнів — від початкового до просунутого — та адаптовані для самостійного навчання або використання в класі. BBC Learning English також пропонує серії, такі як "6 Minute English", які допомагають розширити словниковий запас і покращити навички спілкування. Цей проєкт отримав численні нагороди за інновації в навчанні англійської мови.

Платформа орієнтована на британський варіант англійської та особливо корисна для тих, хто хоче покращити вимову, граматику, словниковий запас та аудіювання. Однак BBC Learning English не надає персоналізованого навчального шляху, функцій взаємодії між користувачами чи створення власного контенту. Незважаючи на це, вона залишається одним із найкращих джерел для тих, хто шукає надійні, професійно підготовлені матеріали з англійської мови. [3]

Аналіз системи для вивчення англійської мови

Розроблена платформа для вивчення англійської мови представляє собою інноваційне рішення на базі фреймворку Django. В основі архітектури лежить модульний підхід, що забезпечує гнучкість та масштабованість системи. Використання сучасних технологій, таких як TinyMCE для форматування контенту, дозволяє створювати якісні навчальні матеріали з багатим мультимедійним наповненням.

Ключовою особливістю платформи є можливість для викладачів створювати власний навчальний контент. На відміну від більшості існуючих рішень, які пропонують лише готові матеріали, дана система надає повну свободу у формуванні навчальної програми. Викладачі отримують інструменти

для створення, редагування та структурування матеріалів відповідно до потреб своїх студентів.

Реалізована чітка категоризація навчальних матеріалів за рівнями CEFR, що дозволяє ефективно організувати процес навчання та забезпечити послідовний розвиток мовних навичок. Кожен матеріал може бути доповнений мультимедійними елементами, що робить процес навчання більш інтерактивним та ефективним.

Таблиця. 1.1

Порівняння з конкурентами

Характеристика	**Розроблена система**	**Duolingo**	**BBC Learning English**
Створення власного контенту	✓	✗	✗
Підтримка CEFR	✓	Частково	✓
Мультимедійний контент	✓	✓	✓
Гейміфікація	Планується	✓	✗
Адаптивне навчання	✓	✓	✗
Робота з автентичними матеріалами	✓	✗	✓
Можливість розширення функціоналу	✓	✗	✗
Інтеграція з навчальними закладами	✓	✗	✗
Ціна	Безкоштовно	Freemium	Безкоштовно
Офлайн доступ	✓	✓	✗

Особлива увага приділяється модулю тестування знань, який забезпечує створення різноманітних типів завдань для перевірки засвоєння матеріалу. Система автоматично обробляє результати тестів та надає детальну статистику успішності, що допомагає викладачам відстежувати прогрес студентів.

Важливою перевагою розробленої системи є її відкритість та можливість інтеграції з існуючими освітніми платформами. Завдяки використанню Django, система легко розширюється новими функціями та може бути адаптована під специфічні потреби навчальних закладів.

1.5. Діаграма прецедентів

Діаграма прецедентів (або діаграма варіантів використання) (англ. Use case diagram) — в UML, діаграма, на якій зображено відношення між акторами та прецедентами в системі.

Діаграма прецедентів показує різні варіанти використання та різні типи користувачів системи і часто супроводжується іншими типами діаграм. Варіанти використання представлені колами або еліпсами. Актори (дійові особи) часто зображуються у вигляді паличок.

Діаграма прецедентів є графом, що складається з множини акторів, прецедентів (варіантів використання) обмежених межею системи (прямокутник), асоціацій між акторами та прецедентами, відношень серед прецедентів, та відношень узагальнення між акторами. Діаграми прецедентів відображають елементи моделі варіантів використання.

Суть діаграми прецедентів полягає в тому, що проєктована система подається у вигляді множини сутностей чи акторів, що взаємодіють із системою за допомогою так званих варіантів використання. Варіант використання (use case) використовують для описання послуг, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, який виконує система під час діалогу з актором. При цьому нічого не говориться про те, яким чином буде реалізовано взаємодію акторів із системою. [18]

У розробленій системі для вивчення англійської мови визначено два основних актори та їх взаємодію з системою через різні варіанти використання (прецеденти), сама продемонстрована на рис 1.1.

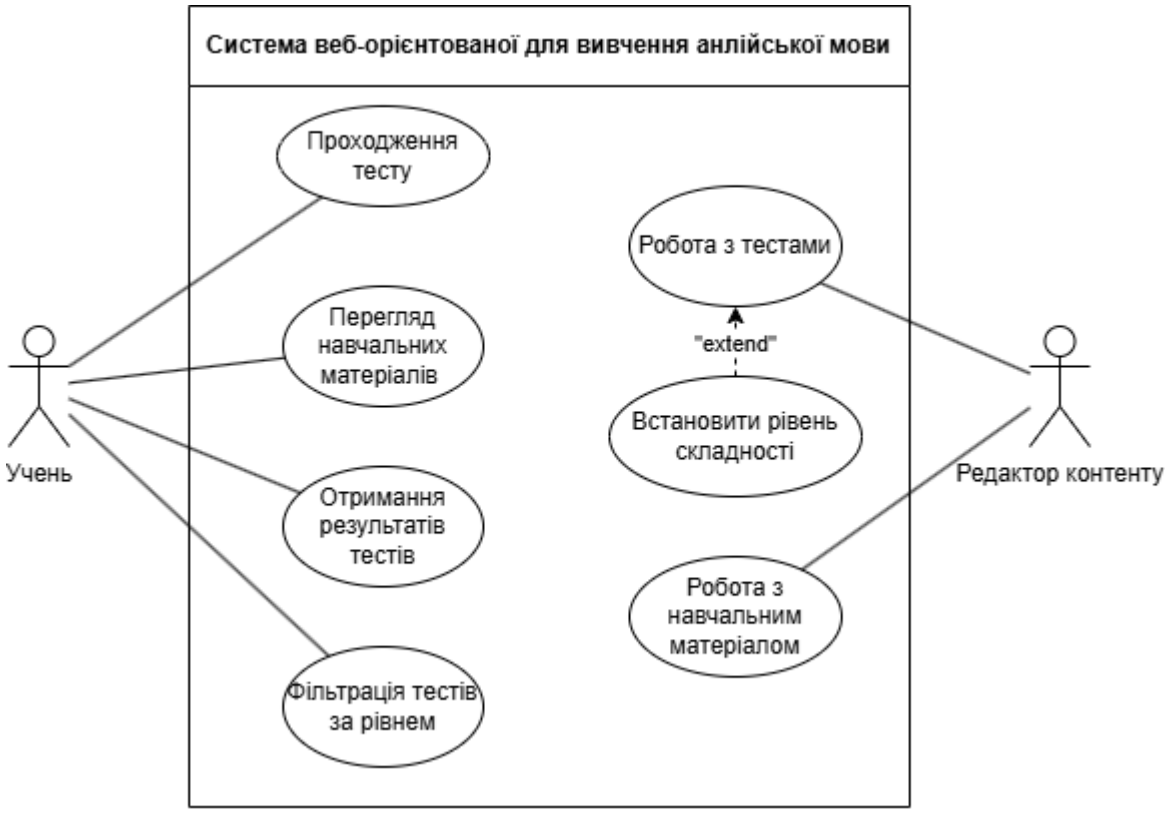

Рис 1.1 Діаграма прецедентів

Актори системи:

Учень - основний користувач системи, який використовує її для навчання

Редактор контенту - користувач з розширеними правами для управління навчальними матеріалами

Варіанти використання для Учня:

Проходження тесту - дозволяє учню проходити тести для перевірки знань, включає відповіді на питання різних типів, автоматична перевірка відповідей

Перегляд навчальних матеріалів - доступ до структурованих навчальних матеріалів, можливість читання текстів та перегляду мультимедіа, навігація по темах та рівнях складності

Отримання результатів тестів - перегляд результатів пройдених тестів, аналіз помилок та правильних відповідей, відстеження прогресу навчання

Фільтрація тестів за рівнем - вибір тестів відповідного рівня складності, сортування за категоріями, пошук потрібних тестових завдань

Варіанти використання для Редактора контенту:

Робота з тестами - створення нових тестів, редагування існуючих тестових завдань, видалення застарілих тестів, має розширення "Встановити рівень складності"

Встановити рівень складності - визначення рівня CEFR для тесту, маркування тестів відповідними позначками складності, забезпечення відповідності тестів стандартам

Робота з навчальним матеріалом - створення нових навчальних матеріалів, редагування існуючого контенту, додавання мультимедійних файлів, категоризація матеріалів

Відношення між прецедентами:

Extend-відношення - "Встановити рівень складності" розширює функціонал "Роботи з тестами", це означає, що при роботі з тестами редактор може додатково встановлювати їх рівень складності

Така структура прецедентів забезпечує чітке розмежування ролей користувачів та їх можливостей у системі, що сприяє ефективній організації навчального процесу та управлінню контентом.

1.6. Постановка завдання

Мета розробки системи

Метою розробки веб-орієнтованої системи для вивчення англійської мови є створення ефективного, доступного та інтерактивного освітнього середовища, яке дозволить користувачам різних рівнів підготовки самостійно вивчати англійську мову, розвивати мовні навички та відстежувати свій прогрес. Система має забезпечити структурований підхід до навчання, поєднуючи теоретичні матеріали з практичними вправами та тестами, що сприятиме комплексному засвоєнню мови.

Розроблена система має стати інструментом, який зробить процес вивчення англійської мови більш ефективним, цікавим та персоналізованим, адаптуючись до індивідуальних потреб та рівня знань кожного користувача.

Вона має надавати можливість вивчати мову в зручному темпі, з будь-якого місця та в будь-який час, що відповідає сучасним тенденціям дистанційного навчання.

Конкретні завдання, які необхідно вирішити

Для досягнення поставленої мети необхідно вирішити наступні завдання:

1 Розробити архітектуру системи, яка забезпечить модульність, масштабованість та надійність функціонування всіх компонентів
2 Створити систему реєстрації та авторизації користувачів з різними ролями (учні, редактори контенту)
3 Розробити модуль управління навчальними матеріалами з інтеграцією редактора TinyMCE
4 Реалізувати модуль тестування знань з підтримкою різних типів тестів
5 Створити інтуїтивно зрозумілий та адаптивний інтерфейс для проходження навчальних матеріалів
6 Розробити механізми відстеження прогресу користувачів
7 Забезпечити надійне зберігання даних з використанням системи управління базами даних
8 Реалізувати механізми захисту системи від несанкціонованого доступу

Система розробляється на основі сучасного фреймворку Django з використанням принципів об'єктно-орієнтованого програмування та шаблону проектування MVC.

Очікувані результати

В результаті виконання проєкту буде створено повнофункціональну веб-орієнтовану систему для вивчення англійської мови, яка включатиме:

1 Модуль реєстрації та авторизації користувачів з підтримкою різних ролей та рівнів доступу
2 Модуль управління навчальними матеріалами з інтегрованим редактором TinyMCE
3 Модуль тестування знань з підтримкою різних типів тестів та запитань
4 Адаптивний інтерфейс для проходження навчальних матеріалів та

тестів

5 Механізми відстеження прогресу користувачів з візуалізацією результатів

Система буде розгорнута на веб-сервері та доступна через інтернет з будь-якого пристрою з веб-браузером. Вона забезпечуватиме надійне зберігання даних, захист від несанкціонованого доступу та стабільну роботу при різних навантаженнях.

Критерії успішності проєкту

Проєкт вважатиметься успішним, якщо будуть досягнуті наступні критерії:

Функціональна повнота: система реалізує всі заплановані функціональні можливості, включаючи реєстрацію користувачів, управління навчальними матеріалами, тестування знань та відстеження прогресу.

Якість користувацького досвіду: інтерфейс системи є інтуїтивно зрозумілим, естетично приємним та адаптивним для різних пристроїв. Користувачі можуть легко орієнтуватися в системі та ефективно використовувати всі її функції.

Надійність та стабільність: система працює стабільно при різних навантаженнях, забезпечує надійне зберігання даних та захист від несанкціонованого доступу. Час відгуку системи не перевищує прийнятних меж.

Масштабованість та розширюваність: архітектура системи дозволяє легко додавати нові функціональні можливості та масштабувати систему при збільшенні кількості користувачів та обсягу даних.

Відповідність технічним вимогам: система розроблена з використанням сучасних технологій та відповідає встановленим технічним вимогам. Код є чистим, добре структурованим та документованим.

Обмеження та припущення

При розробці системи необхідно враховувати наступні обмеження та припущення:

1. Технологічні обмеження: система розробляється з використанням фреймворку Django та інших технологій, зазначених у технічному завданні
2. Часові обмеження: розробка системи має бути завершена у встановлені терміни
3. Ресурсні обмеження: розробка здійснюється з обмеженими людськими та технічними ресурсами
4. Припущення щодо користувачів: передбачається, що користувачі мають базові навички роботи з веб-інтерфейсами
5. Припущення щодо навантаження: очікується, що система буде обслуговувати обмежену кількість одночасних користувачів на початковому етапі
6. Припущення щодо безпеки: передбачається, що користувачі дотримуватимуться базових правил безпеки
7. Обмеження щодо контенту: система призначена для вивчення англійської мови і не підтримуватиме інші мови на початковому етапі

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1. Логічна модель даних у вигляді ER-діаграми

Модель «сутність-зв'язок» (ER-модель) (англ. Entity-relationship model або entity-relationship diagram) — модель даних, яка дозволяє описувати концептуальні схеми за допомогою узагальнених конструкцій блоків. ER-модель — це метамодель даних, тобто засіб опису моделей даних. Існує ряд моделей для представлення знань, але одним з найзручніших інструментів уніфікованого представлення даних, незалежного від програмного забезпечення, що його реалізує, є модель «сутність-зв'язок». Важливим є той факт, що з моделі «сутність-зв'язок» можуть бути породжені всі існуючі моделі даних (ієрархічна, мережева, реляційна, об'єктна), тому вона є найзагальнішою. [21]

Типи зв'язків в ER-діаграмах

ER-діаграми використовують різні типи зв'язків для відображення відносин між сутностями:

Один до одного (1:1): Кожен екземпляр однієї сутності пов'язаний з одним екземпляром іншої сутності, і навпаки. Наприклад, один співробітник має одне службове авто.

Один до багатьох (1:N): Один екземпляр сутності може бути пов'язаний з багатьма екземплярами іншої сутності, але кожен екземпляр другої сутності пов'язаний лише з одним екземпляром першої. Наприклад, один викладач викладає багато курсів.

Багато до багатьох (M:N): Кожен екземпляр однієї сутності може бути пов'язаний з багатьма екземплярами іншої сутності, і навпаки. Наприклад, студенти можуть записуватись на багато курсів, а курси можуть мати багато студентів.

Рекурсивний зв'язок: Сутність може бути пов'язана сама з собою. Наприклад, співробітник може бути керівником іншого співробітника.

Обов'язковий зв'язок: Участь сутності у зв'язку є обов'язковою.

Наприклад, кожен студент повинен бути зареєстрований принаймні на один курс.

Необов'язковий зв'язок: Участь сутності у зв'язку є необов'язковою. Наприклад, деякі викладачі можуть не мати призначених курсів. [11]

Опис ER-діаграми системи для вивчення англійської мови

На рис 2.1 представлена ER-діаграма системи для вивчення англійської мови, яка відображає всі сутності та зв'язки між ними. Діаграма демонструє структуру бази даних, що складається з трьох основних модулів: Core, Learning Materials та Tests Module.

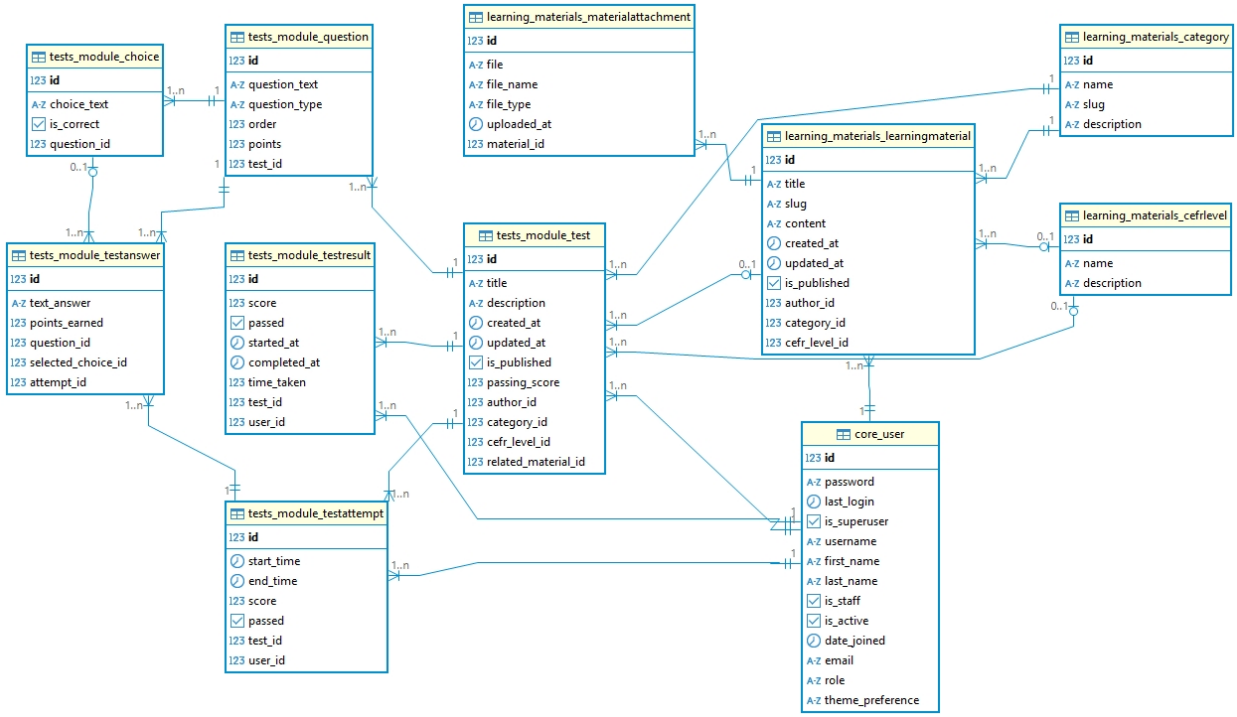

Рис 2.1 Логічна модель даних у вигляді ER-діаграми

Сутність **core_user** представляє користувачів системи для вивчення англійської мови. Кожен користувач має унікальний ідентифікатор (id), унікальне ім'я користувача для входу в систему (username), електронну пошту (email), зашифрований пароль (password), ім'я (first_name), прізвище (last_name), індикатор активності облікового запису (is_active), індикатор доступу до адміністративної панелі (is_staff), індикатор прав суперкористувача (is_superuser), дату та час реєстрації (date_joined), дату та час останнього входу (last_login).

Сутність **learning_materials_category** представляє категорії навчальних матеріалів. Кожна категорія має унікальний ідентифікатор (id), назву (name), опис (description), посилання на батьківську категорію, якщо є (parent_id), порядок відображення (order).

Сутність **learning_materials_cerflevel** представляє рівні володіння англійською мовою за Загальноєвропейською рамкою мовних компетенцій (CEFR). Кожен рівень має унікальний ідентифікатор (id), код рівня (code) - A1, A2, B1, B2, C1, C2, назву (name), опис (description).

Сутність **learning_materials_learningmaterial** представляє навчальні матеріали для вивчення англійської мови. Кожен матеріал має унікальний ідентифікатор (id), заголовок (title), вміст у форматі HTML з форматуванням TinyMCE (content), дату та час створення (created_at), дату та час останнього оновлення (updated_at), індикатор публікації (is_published), посилання на категорію (category_id), посилання на рівень CEFR (cefr_level_id).

Сутність **learning_materials_matterialattachment** представляє вкладення (файли) до навчальних матеріалів. Кожне вкладення має унікальний ідентифікатор (id), шлях до файлу (file), назву (title), опис (description), тип файлу - аудіо, відео, документ тощо (file_type), посилання на навчальний матеріал (material_id).

Сутність **tests_module_test** представляє тести для перевірки знань. Кожен тест має унікальний ідентифікатор (id), назву (title), опис (description), обмеження часу на проходження у хвилинах (time_limit), прохідний бал (passing_score), індикатор публікації (is_published), дату та час створення (created_at), дату та час останнього оновлення (updated_at), посилання на рівень CEFR (cefr_level_id).

Сутність **tests_module_question** представляє питання в тестах. Кожне питання має унікальний ідентифікатор (id), текст питання (text), тип питання - одиночний вибір, множинний вибір, відкрите питання (question_type), кількість балів за правильну відповідь (score), порядок відображення питання в тесті (order), посилання на тест (test_id).

Сутність **tests_module_choice** представляє варіанти відповідей на питання. Кожен варіант відповіді має унікальний ідентифікатор (id), текст варіанту відповіді (text), індикатор правильності відповіді (is_correct), пояснення до варіанту відповіді (explanation), посилання на питання (question_id).

Сутність **tests_module_testattempt** представляє спроби проходження тестів користувачами. Кожна спроба має унікальний ідентифікатор (id), дату та час початку проходження (started_at), дату та час завершення проходження (completed_at), індикатор завершення спроби (is_completed), посилання на користувача (user_id), посилання на тест (test_id).

Сутність **tests_module_testresult** представляє результати проходження тестів. Кожен результат має унікальний ідентифікатор (id), набраний бал (score), витрачений час у секундах (time_spent), індикатор досягнення прохідного балу (is_passed), дату та час завершення (completed_at), посилання на користувача (user_id), посилання на тест (test_id), посилання на спробу (attempt_id).

Сутність **tests_module_testanswer** представляє відповіді користувача на питання тесту. Кожна відповідь має унікальний ідентифікатор (id), текст відповіді для відкритих питань (text_answer), індикатор правильності відповіді (is_correct), посилання на питання (question_id), посилання на результат тесту (result_id), посилання на вибраний варіант відповіді для питань з вибором (choice_id).

База даних проєкту приведена до третьої нормальної форми (3НФ), що забезпечує оптимальну організацію даних, мінімізує надлишковість та запобігає аномаліям при модифікації даних.

Відповідність першій нормальній формі (1НФ) забезпечується тим, що всі атрибути містять атомарні (неподільні) значення, кожна таблиця має первинний ключ (id), а повторювані групи атрибутів відсутні.

Відповідність другій нормальній формі (2НФ) досягається завдяки тому, що база даних відповідає 1НФ, а всі неключові атрибути повністю функціонально залежать від первинного ключа. Часткові функціональні залежності відсутні.

Відповідність третій нормальній формі (3НФ) забезпечується тим, що база даних відповідає 2НФ, а транзитивні залежності між неключовими атрибутами відсутні.

Правильне розділення даних на логічні сутності (користувачі, категорії, рівні CEFR, навчальні матеріали, тести, питання, варіанти відповідей, спроби та результати) дозволяє уникнути дублювання інформації. Використання зовнішніх ключів замість дублювання даних забезпечує цілісність інформації та спрощує її оновлення.

Відсутність транзитивних залежностей проявляється в тому, що інформація про категорії, рівні CEFR та тести зберігається лише в відповідних таблицях і не дублюється в інших таблицях.

Оптимальні зв'язки між таблицями (один-до-багатьох, багато-до-одного) забезпечують гнучкість структури та ефективність запитів.

Така структура бази даних забезпечує надійне зберігання даних, ефективну роботу з ними та можливість масштабування проєкту в майбутньому, що є критично важливим для освітньої платформи з різноманітними типами контенту та взаємодії користувачів.

Розроблена ER-діаграма дозволить ефективно структурувати базу даних системи для вивчення англійської мови, забезпечуючи оптимальну організацію даних та взаємозв'язків між ними. Чітко визначені сутності та їх атрибути створюють надійну основу для розробки функціональних модулів системи, що відповідають за управління користувачами, навчальними матеріалами та тестуванням знань. Реалізація цієї моделі даних сприятиме створенню гнучкої та масштабованої системи, здатної ефективно задовольняти потреби користувачів у вивченні англійської мови.

2.2. Діаграма класів та кооперацій

Діагра́ма кла́сів (англ. *Class diagram*) — статичне представлення структури моделі в UML. Відображає статичні (декларативні) елементи, такі як:

класи, типи даних, їх зміст та відношення. Діаграма класів може містити позначення для пакетів та може містити позначення для вкладених пакетів. Також, діаграма класів може містити позначення деяких елементів поведінки, однак їх динаміка розкривається в інших типах діаграм. [16]

Зв’язки між класами на діаграмі показують, як об’єкти взаємодіють між собою: наприклад, через асоціацію (звичайний зв’язок), наслідування (успадкування властивостей), агрегацію або композицію (цілісні частини). Діаграми класів широко використовуються на етапі проєктування для планування архітектури програмного забезпечення.

Діаграма класів для систем зображено на рис 2.2

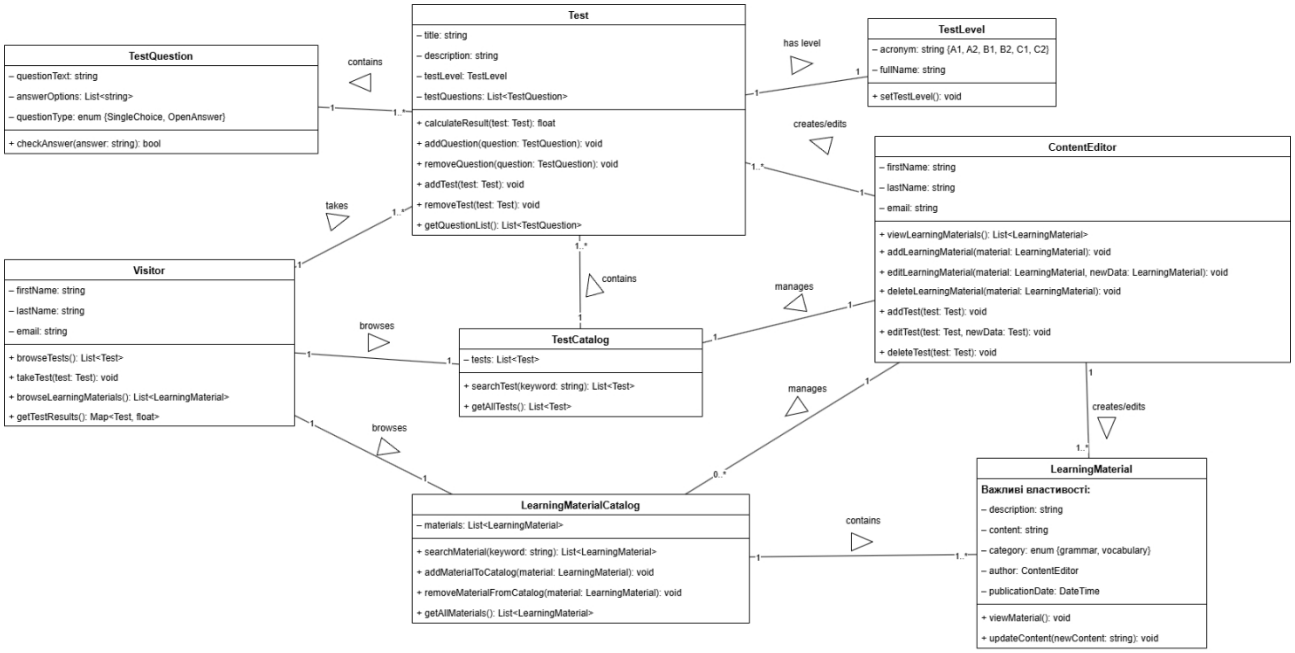

Рис 2.2 Діаграма класів та кооперацій

Розроблена система для вивчення англійської мови має модульну архітектуру, що складається з набору взаємопов'язаних класів, які забезпечують повний функціонал для навчання користувачів.

Клас "Test" відповідає за зберігання та управління тестами. Містить атрибути: title (string), description (string), testLevel (TestLevel), testQuestions (List<TestQuestion>). Надає методи: calculateResult(test: Test): float, addQuestion(question: TestQuestion): void, removeQuestion(question: TestQuestion): void, addTest(test: Test): void, removeTest(test: Test): void, getQuestionList():

List<TestQuestion>.

Клас "TestQuestion" представляє окремі питання в тесті. Містить атрибути: questionText (string), answerOptions (List<string>), questionType (enum {SingleChoice, OpenAnswer}). Надає метод: checkAnswer(answer: string): bool.

Клас "TestLevel" визначає рівень складності тесту відповідно до Загальноєвропейської рамки мовних компетенцій (CEFR). Містить атрибути: acronym (string: {A1, A2, B1, B2, C1, C2}), fullName (string). Надає метод: setTestLevel(): void.

Клас "TestCatalog" керує колекцією тестів та забезпечує доступ до них. Містить атрибут: tests (List<Test>). Надає методи: searchTest(keyword: string): List<Test>, getAllTests(): List<Test>.

Клас "LearningMaterial" представляє навчальний контент. Містить атрибути: description (string), content (string), category (enum {grammar, vocabulary}), author (ContentEditor), publicationDate (DateTime). Надає методи: viewMaterial(): void, updateContent(newContent: string): void.

Клас "ContentEditor" забезпечує функціональність для створення та редагування навчальних матеріалів і тестів. Містить атрибути: firstName (string), lastName (string), email (string). Надає методи: viewLearningMaterials(): List<LearningMaterial>, addLearningMaterial(material: LearningMaterial): void, editLearningMaterial(material: LearningMaterial, newData: LearningMaterial): void, deleteLearningMaterial(material: LearningMaterial): void, addTest(test: Test): void, editTest(test: Test, newData: Test): void, deleteTest(test: Test): void.

Клас "LearningMaterialCatalog" керує колекцією навчальних матеріалів. Містить атрибут: materials (List<LearningMaterial>). Надає методи: searchMaterial(keyword: string): List<LearningMaterial>, addMaterialToCatalog(material: LearningMaterial): void, removeMaterialFromCatalog(material: LearningMaterial): void, getAllMaterials(): List<LearningMaterial>.

Клас "Visitor" представляє користувача системи, що вивчає англійську мову. Містить атрибути: firstName (string), lastName (string), email (string). Надає

методи: browseTests(): List<Test>, takeTest(test: Test): void, browseLearningMaterials(): List<LearningMaterial>, getTestResults(): Map<Test, float>.

Розроблена архітектура забезпечує ефективну взаємодію між класами: Користувач може переглядати Навчальні матеріали з Каталогу Навчальних Матеріалів та проходити Тести з Каталогу Тестів; Редактор контенту дозволяє створювати та редагувати як Навчальні матеріали, так і Тести; Тести можуть бути пов'язані з Навчальними матеріалами; Рівень тесту забезпечує категоризацію Тестів за складністю.

Прості кооперації

Прості кооперації в UML, відомі також як діаграми співпраці або комунікаційні діаграми, є структурними елементами, що описують взаємодію між об'єктами системи для виконання певної функціональності. Вони фокусуються на ролях об'єктів та їхніх зв'язках, необхідних для досягнення спільної мети, що дозволяє спростити розуміння складних систем, розбиваючи їх на логічні компоненти.

У UML 2.x ці діаграми відомі як комунікаційні діаграми і використовуються для моделювання динамічної поведінки системи, ілюструючи потік повідомлень між об'єктами під час певного сценарію або варіанту використання. Вони є важливим інструментом для візуалізації взаємодій між об'єктами або компонентами в системі та показують, як вони співпрацюють для досягнення конкретних завдань або сценаріїв. [5]

Основними елементами діаграми співпраці є:

Об'єкти: представлені як прямокутники з мітками, що вказують на ім'я об'єкта та його клас.

Актори: ініціюють взаємодію в діаграмі та мають визначені ролі.

Зв'язки: показують асоціації між об'єктами та акторами, зображуються суцільними лініями.

Повідомлення між об'єктами: позначаються стрілками з мітками, що вказують на послідовність та зміст повідомлень. [4]

Тестова підсистема для вивчення англійської мови, зображена на рис 2.3, складається з чотирьох взаємопов'язаних класів, що забезпечують повний функціонал для створення, управління та проходження тестів.

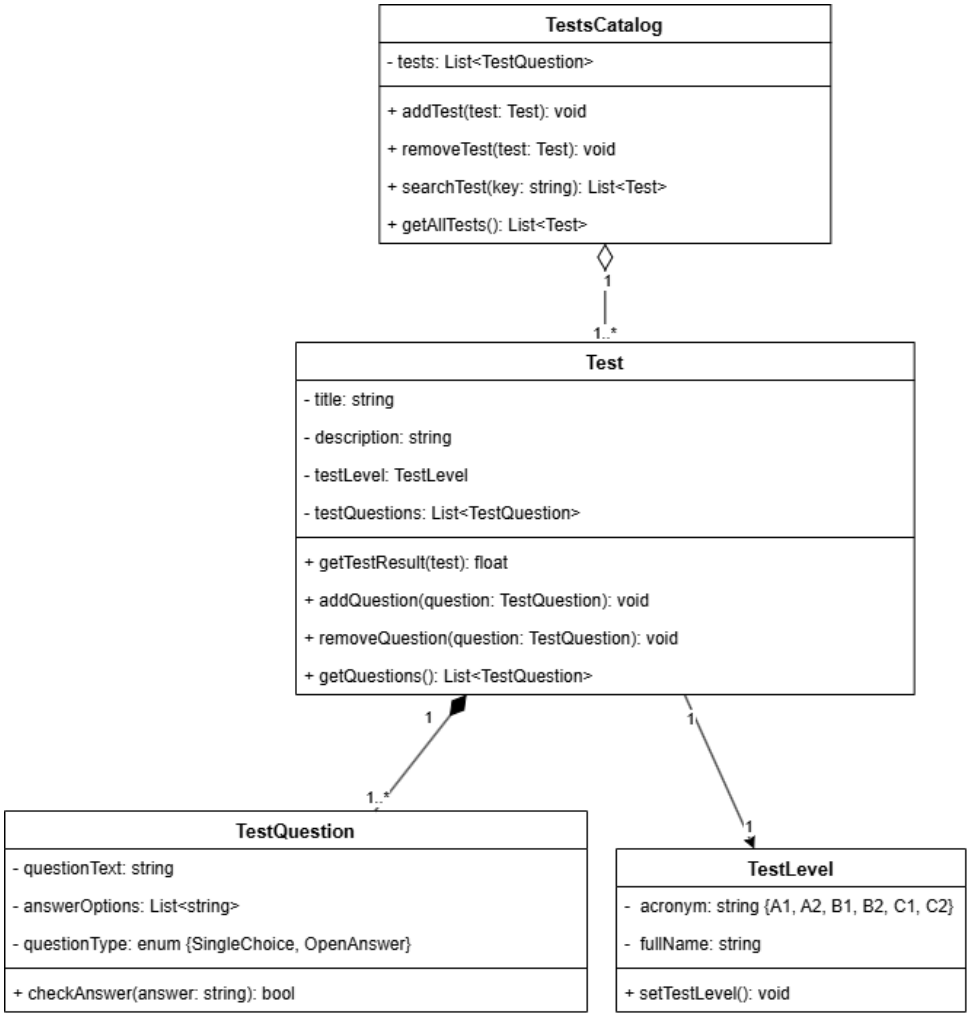

Рис 2.3 Проста кооперація модуля "Тести"

Клас "**TestCatalog**" виступає контейнером для зберігання колекції тестів; містить атрибут tests: List<TestQuestion>; надає методи: addTest(test: Test): void, removeTest(test: Test): void, searchTest(key: string): List<Test>, getAllTests(): List<Test>.

Клас "**Test**" є центральним елементом підсистеми; містить атрибути: title: string, description: string, testLevel: TestLevel, testQuestions: List<TestQuestion>; надає методи: getTestResult(test): float, addQuestion(question: TestQuestion): void, removeQuestion(question: TestQuestion): void, getQuestions(): List<TestQuestion>.

Клас "**TestQuestion**" представляє окремі питання в тесті; містить атрибути:

questionText: string, answerOptions: List<string>, questionType: enum (SingleChoice, OpenAnswer); надає метод checkAnswer(answer: string): bool.

Клас **"TestLevel"** визначає складність тесту; містить атрибути: acronym: string (A1, A2, B1, B2, C1, C2), fullName: string; надає метод setTestLevel().

Діаграма демонструє зв'язки між класами: TestCatalog агрегує множину об'єктів класу Test (зв'язок 1 до багатьох); Test композиційно включає множину об'єктів класу TestQuestion (зв'язок 1 до багатьох); Test асоціюється з одним об'єктом класу TestLevel (зв'язок 1 до 1). Така архітектура забезпечує гнучкість, масштабованість та ефективність тестової підсистеми для вивчення англійської мови.

Підсистема навчальних матеріалів для вивчення англійської мови, зображена на рис 2.4, складається з двох взаємопов'язаних класів, що забезпечують повний функціонал для створення, управління та доступу до навчальних матеріалів.

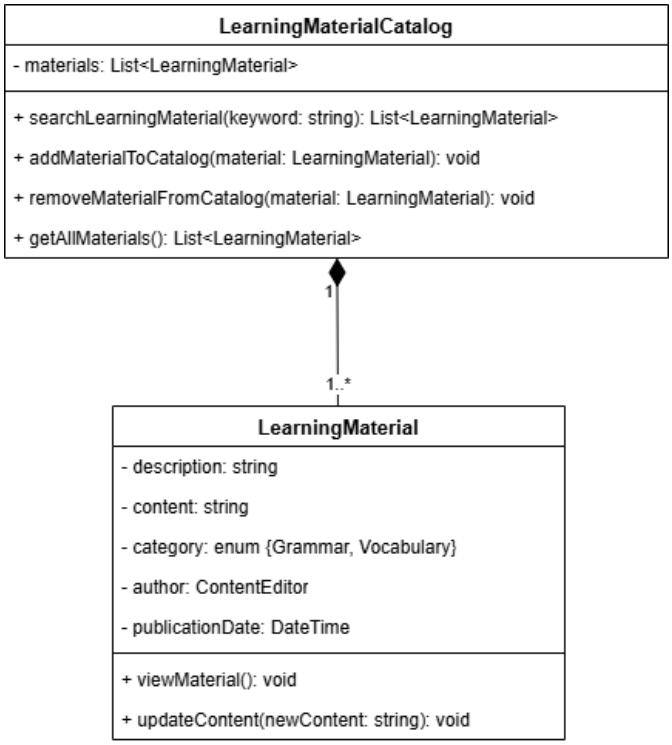

Рис 2.4 Проста кооперація модуля "Навчальні матеріали"

Клас **"LearningMaterialCatalog"** виступає контейнером для зберігання колекції навчальних матеріалів; містить атрибут materials:

List<LearningMaterial>; надає методи searchLearningMaterial(keyword: string): List<LearningMaterial>, addMaterialToCatalog(material: LearningMaterial): void, removeMaterialFromCatalog(material: LearningMaterial): void, getAllMaterials(): List<LearningMaterial>.

Клас **"LearningMaterial"** є центральним елементом підсистеми; містить розділ "Важливі властивості" з атрибутами description: string, content: string, category: enum (Grammar, Vocabulary), author: ContentEditor, publicationDate: DateTime; надає методи viewMaterial(): void, updateContent(newContent: string): void.

Діаграма демонструє зв'язок між класами: **LearningMaterialCatalog** агрегує множину об'єктів класу **LearningMaterial** (зв'язок 1 до багатьох). Така архітектура забезпечує гнучкість, масштабованість та ефективність підсистеми навчальних матеріалів для вивчення англійської мови, дозволяючи зручно організовувати, шукати та використовувати різноманітні навчальні ресурси.

Підсистема взаємодії користувача з навчальними матеріалами та тестами, зображена на рис 2.5, складається з трьох взаємопов'язаних класів, що забезпечують повний функціонал для навчання користувачів.

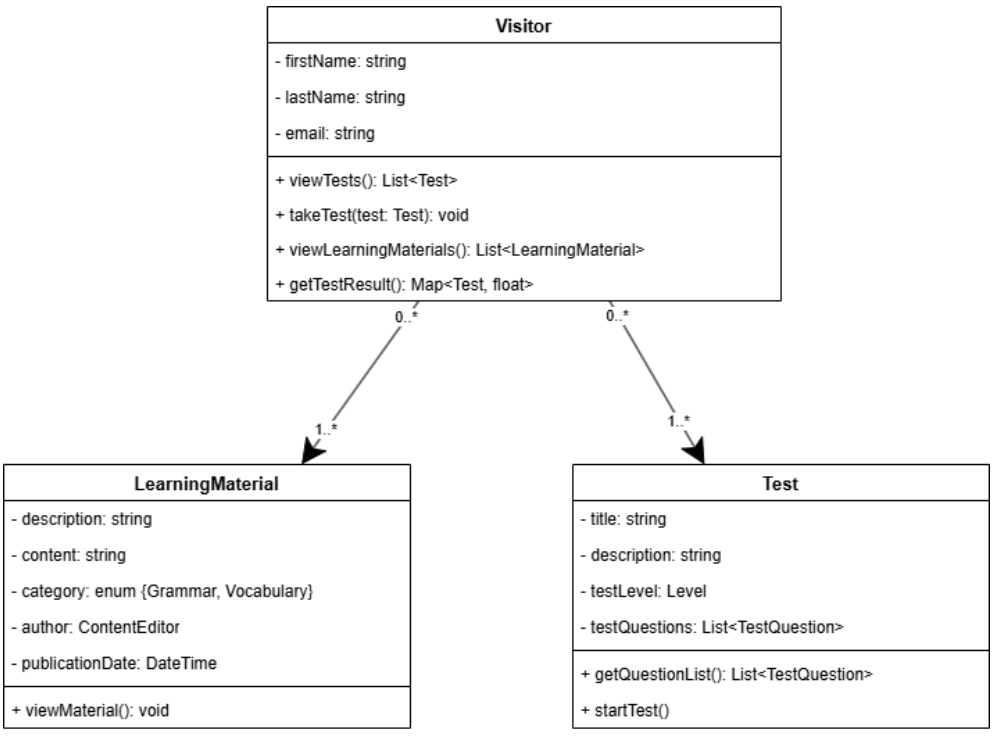

Рис 2.5 Проста кооперація модуля "Взаємодія користувача"

Клас **"Visitor"** представляє користувача системи; містить атрибути: firstName, lastName, email; надає методи: viewTests(): List<Test>, solveTest(test: Test): void, viewLearningMaterials(): List<LearningMaterial>, getTestResults(): Map<Test, float>.

Клас **"LearningMaterial"** містить розділ "Важливі властивості" з атрибутами: description: string, content: string, category: enum (Grammar, Vocabulary), author: ContentEditor, publicationDate: DateTime; надає метод viewMaterial(): void.

Клас **"Test"** містить атрибути: title: string, description: string, testLevel: Level, testQuestions: List<TestQuestion>; надає методи: getQuestionList(): List<TestQuestion>, startTest().

Діаграма демонструє зв'язки між класами: **Visitor** може переглядати багато **LearningMaterial** (зв'язок 0..* до 1..) та проходити багато **Test** (зв'язок 0.. до 1..*). Така архітектура забезпечує гнучкість та ефективність взаємодії користувача з системою для вивчення англійської мови, дозволяючи зручно отримувати доступ до навчальних ресурсів та проходити тестування для перевірки знань.

2.3. Діаграма пакетів

Діаграма пакетів (Package Diagram) в UML є структурною діаграмою, яка відображає організацію системи на рівні модулів або компонентів. Вона демонструє логічне групування елементів системи та залежності між ними. Діаграми пакетів особливо корисні для візуалізації архітектури великих систем, оскільки дозволяють представити систему на високому рівні абстракції, показуючи основні компоненти та їх взаємозв'язки без деталізації внутрішньої структури кожного компонента. [13]

Архітектура системи для вивчення англійської мови, зображена на рис 2.6, побудована за принципом багаторівневої архітектури та складається з двох основних шарів:

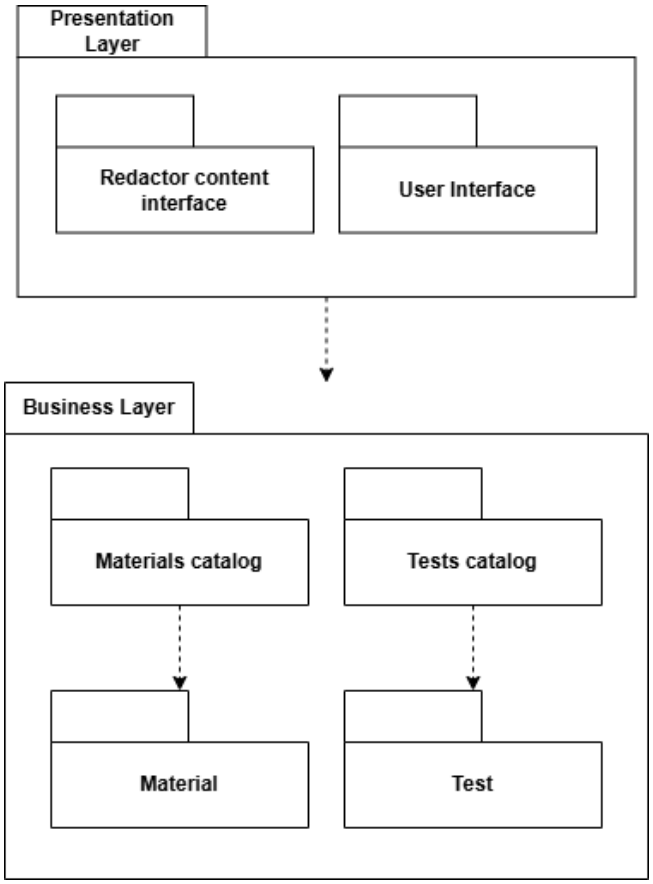

Рис 2.6 Діаграма пакетів

1 **Шар представлення (Presentation Layer)** - відповідає за взаємодію з користувачем та відображення інформації. Цей шар містить два основні компоненти:
    1 Інтерфейс редактора контенту (Redactor content interface) - забезпечує функціональність для створення та редагування навчальних матеріалів і тестів адміністраторами системи.
    2 Інтерфейс користувача (User Interface) - надає доступ до навчальних матеріалів та тестів для звичайних користувачів системи.
2 **Прикладний шар (Application Layer)** - містить основну бізнес-логіку системи та складається з чотирьох основних компонентів:
    1 Каталог матеріалів (Materials catalog) - відповідає за зберігання та управління колекцією навчальних матеріалів.
    2 Матеріал (Material) - представляє окремі навчальні матеріали з їх

вмістом та метаданими.

3 Каталог тестів (Tests catalog) - відповідає за зберігання та управління колекцією тестів.

4 Тест (Test) - представляє окремі тести з питаннями та варіантами відповідей.

Діаграма демонструє залежності між компонентами: Шар представлення залежить від Прикладного шару, Каталог матеріалів залежить від компонента Матеріал, а Каталог тестів залежить від компонента Тест. Така архітектура забезпечує чіткий розподіл відповідальності між компонентами, що сприяє гнучкості, масштабованості та підтримуваності системи для вивчення англійської мови.

2.4. Діаграма компонентів

Діаграма компонентів (англ. Component diagram) — в UML, діаграма, на якій відображаються компоненти, залежності та зв'язки між ними.

Діаграма компонентів відображає залежності між компонентами програмного забезпечення, включаючи компоненти вихідних кодів, бінарні компоненти, та компоненти, що можуть виконуватись. Модуль програмного забезпечення може бути представлено як компоненту. Деякі компоненти існують під час компіляції, деякі — під час компонування, а деякі під час роботи програми.. [17]

Діаграма компонентів системи для вивчення англійської мови, зображена на рис 2.7, складається з трьох основних підсистем:

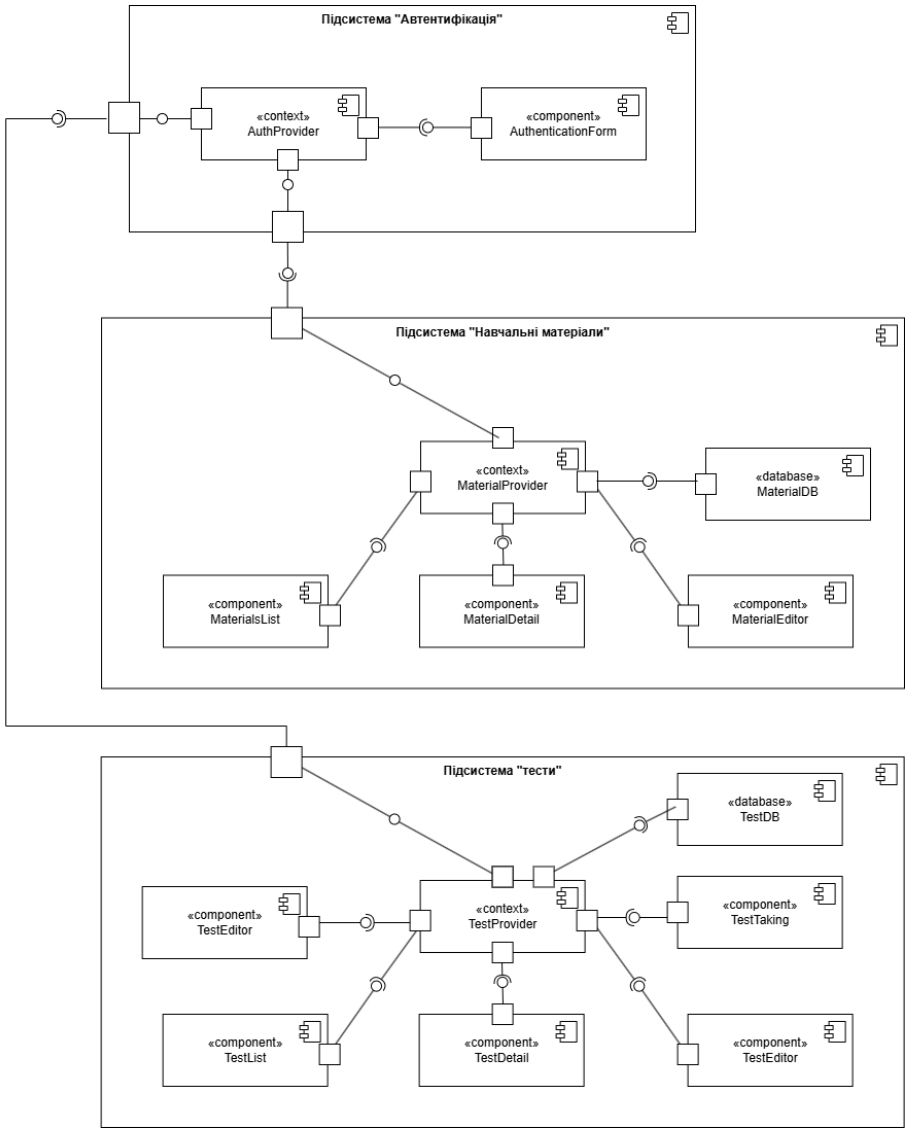

Рис 2.7 Діаграма компонентів

Підсистема "**Автентифікація**" - відповідає за управління доступом користувачів до системи. Включає наступні компоненти:

1. AuthProvider (контекст) - забезпечує логіку автентифікації та авторизації користувачів.
2. AuthenticationForm (компонент) - надає інтерфейс для входу користувачів у систему.

Підсистема **"Навчальні матеріали"** - відповідає за управління навчальним контентом. Включає наступні компоненти:

1 MaterialProvider (контекст) - забезпечує доступ до даних навчальних матеріалів.

2 MaterialDB (база даних) - зберігає інформацію про навчальні матеріали.
3 MaterialsList (компонент) - відображає список доступних навчальних матеріалів.
4 MaterialDetail (компонент) - відображає детальну інформацію про конкретний навчальний матеріал.
5 MaterialEditor (компонент) - надає інтерфейс для створення та редагування навчальних матеріалів.

Підсистема **"Тести"** - відповідає за управління тестами для перевірки знань. Включає наступні компоненти:

1 TestProvider (контекст) - забезпечує доступ до даних тестів.
2 TestDB (база даних) - зберігає інформацію про тести та результати їх проходження.
3 TestList (компонент) - відображає список доступних тестів.
4 TestDetail (компонент) - відображає детальну інформацію про конкретний тест.
5 TestEditor (компонент) - надає інтерфейс для створення та редагування тестів.
6 TestTaking (компонент) - забезпечує інтерфейс для проходження тестів користувачами.

Розроблена діаграма компонентів забезпечує чітке структурування архітектури системи для вивчення англійської мови, демонструючи логічну організацію та взаємозв'язки між ключовими підсистемами. Виділення трьох основних підсистем – "Автентифікація", "Навчальні матеріали" та "Тести" – дозволяє ефективно розподілити функціональність та відповідальність між компонентами, що сприяє модульності та масштабованості рішення.

3 РОЗРОБКА ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1. Система управління інформаційною базою

Вибір PostgreSQL як СУБД для проєкту "English Learning"

Для проєкту "English Learning" було обрано PostgreSQL як систему управління базами даних. Це рішення базується на ряді важливих факторів, які роблять PostgreSQL оптимальним вибором для даного типу веб-додатку.

PostgreSQL відома своєю надійністю та стабільністю роботи. Ця СУБД має понад 30 років розвитку та постійно вдосконалюється. Вона забезпечує цілісність даних на рівні транзакцій, що критично важливо для освітньої платформи, де необхідно зберігати прогрес користувачів та результати тестування.

PostgreSQL повністю відповідає вимогам ACID (Atomicity, Consistency, Isolation, Durability), що гарантує надійність операцій з даними. Це особливо важливо для системи навчання, де втрата даних про прогрес користувача може негативно вплинути на досвід використання платформи.

Проєкт "English Learning" передбачає роботу з різними типами контенту (текст, аудіо, відео) та складними структурами даних. PostgreSQL пропонує широкий спектр типів даних, включаючи JSON, масиви та геометричні типи, що дозволяє ефективно моделювати предметну область.

PostgreSQL має потужну систему розширень, яка дозволяє додавати нові функціональні можливості без зміни ядра СУБД. Це дає можливість у майбутньому розширювати функціональність проєкту, наприклад, додавати повнотекстовий пошук по навчальних матеріалах або географічну прив'язку користувачів.

Система навчання англійської мови потребує виконання складних запитів для аналізу прогресу користувачів, генерації статистики та рекомендацій. PostgreSQL має потужний оптимізатор запитів, який забезпечує високу продуктивність при роботі зі складними запитами та великими обсягами даних.

З ростом кількості користувачів та обсягу навчальних матеріалів виникає потреба в масштабуванні бази даних. PostgreSQL підтримує різні стратегії масштабування, включаючи реплікацію та шардинг, що дозволяє системі зростати разом із збільшенням навантаження.

Django, фреймворк на якому побудовано проєкт, має відмінну підтримку PostgreSQL через свій ORM (Object-Relational Mapping). Це дозволяє ефективно використовувати всі переваги PostgreSQL, зберігаючи при цьому простоту розробки, яку надає Django.

PostgreSQL має розвинену систему безпеки з підтримкою різних методів автентифікації, керування доступом на рівні ролей та шифрування даних. Це особливо важливо для освітньої платформи, де зберігаються персональні дані користувачів.

PostgreSQL має широку активну спільноту розробників та користувачів, а також відмінну документацію. Це спрощує процес розробки та вирішення потенційних проблем.

PostgreSQL є безкоштовною СУБД з відкритим кодом, що дозволяє знизити витрати на розробку та підтримку проєкту, не жертвуючи при цьому функціональністю та продуктивністю. [10]

Конфігурація PostgreSQL у проєкті

У файлі налаштувань проєкту визначено конфігурацію для підключення до бази даних PostgreSQL, яка включає використання PostgreSQL як СУБД, назву бази даних (english_learning_db), користувача бази даних (postgres), пароль для доступу, а також хост та порт для підключення. Така конфігурація забезпечує надійне та ефективне зберігання даних для проєкту "English Learning", створюючи міцну основу для розвитку функціональності системи.

3.2. Розробка інформаційної бази

При розробці проєкту "English Learning" було застосовано об'єктно-реляційне відображення (ORM) Django, що дозволило значно спростити роботу

з базою даних PostgreSQL та підвищити ефективність розробки.

Django ORM є потужним інструментом, який забезпечує абстракцію над реляційною базою даних, дозволяючи працювати з даними як з об'єктами Python, а не писати SQL-запити вручну. Це надає ряд переваг, які були використані в проєкті:

Моделі даних у проєкті визначені як класи Python, що успадковуються від базового класу models.Model . Кожна модель відповідає таблиці в базі даних, а атрибути моделі - колонкам таблиці. Наприклад, модель User розширює стандартну модель користувача Django, додаючи поля для ролі та налаштувань теми. Реалізація моделей представлена в додатку А.

Використання ORM дозволило автоматично створювати та оновлювати схему бази даних через систему міграцій Django. Це забезпечило контрольовані зміни структури бази даних та можливість відстежувати історію змін.

Зв'язки між таблицями реалізовані через поля зовнішніх ключів (ForeignKey), багато-до-багатьох (ManyToManyField) та один-до-одного (OneToOneField). Наприклад, навчальні матеріали пов'язані з категоріями та рівнями CEFR через зовнішні ключі.

ORM Django надає зручний API для виконання CRUD-операцій (створення, читання, оновлення, видалення) без необхідності писати SQL-запити. Це значно прискорило розробку та зменшило кількість потенційних помилок. [9]

Система запитів Django ORM (QuerySet API) дозволила ефективно фільтрувати, сортувати та агрегувати дані. Наприклад, отримання всіх опублікованих матеріалів певного рівня або підрахунок кількості успішних спроб проходження тесту.

Валідація даних відбувається на рівні моделей, що забезпечує цілісність даних. Наприклад, перевірка унікальності імен користувачів або обмеження на довжину полів.

Використання ORM забезпечило незалежність від конкретної СУБД. Хоча проєкт використовує PostgreSQL, перехід на іншу СУБД потребуватиме

мінімальних змін у коді.

Система сигналів Django дозволила реагувати на події, пов'язані з моделями (наприклад, створення нового користувача або зміна статусу публікації матеріалу).

Адміністративний інтерфейс Django автоматично генерується на основі моделей, що дозволило швидко створити панель адміністратора для управління даними.

Використання Django ORM також забезпечило захист від SQL-ін'єкцій, оскільки параметризація запитів відбувається автоматично. [9]

Таким чином, застосування ORM у проєкті "English Learning" дозволило значно прискорити розробку, підвищити якість коду та забезпечити надійну роботу з базою даних PostgreSQL.

Додатково, було розроблено фізичну модель бази даних у вигляді ER-діаграми, представлену на рис 3.1.

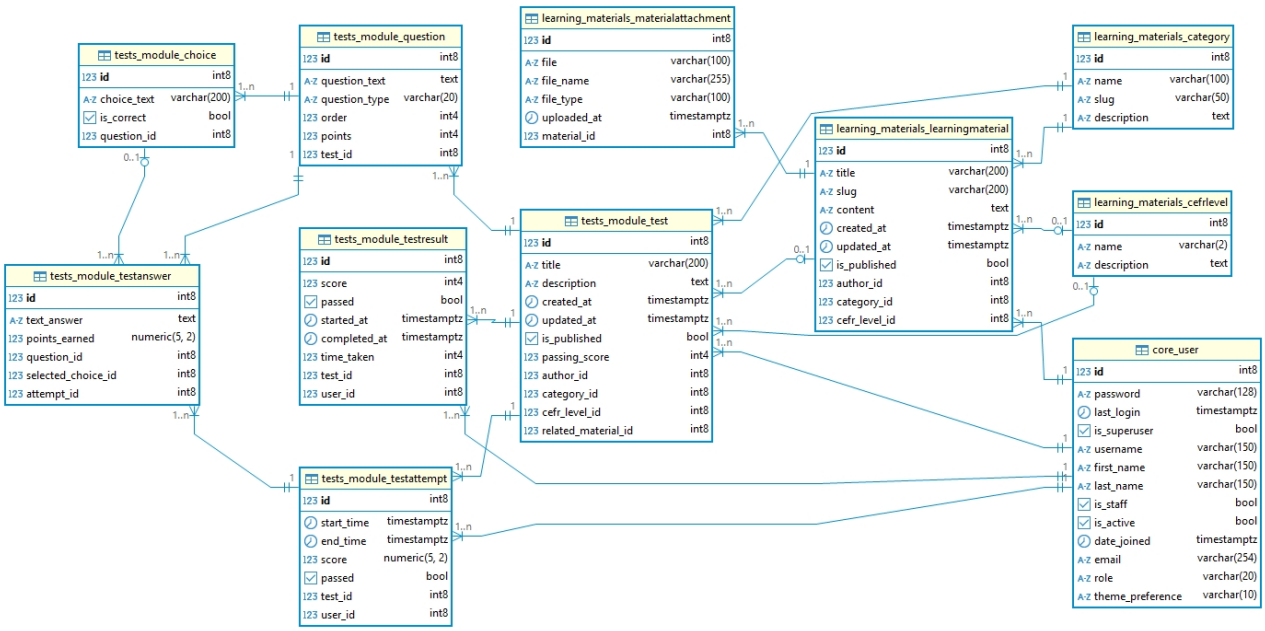

Рис 3.1 Фізична модель даних в Dbeaver

Таблиця: core_user

id – integer – унікальний ідентифікатор користувача, первинний ключ; username – varchar(150) – унікальне ім’я користувача для входу в систему; email – varchar(254) – електронна адреса користувача для комунікації та відновлення

паролю; first_name – varchar(150) – ім’я користувача; last_name – varchar(150) – прізвище користувача; password – varchar(128) – хешований пароль для безпечного зберігання; role – varchar(20) – роль користувача в системі (студент, викладач, адміністратор); theme_preference – varchar(10) – налаштування теми інтерфейсу (світла/темна); is_active – boolean – статус активності облікового запису; is_staff – boolean – прапорець доступу до адміністративної панелі; is_superuser – boolean – прапорець для надання повних прав доступу; date_joined – timestamp – дата та час реєстрації користувача; last_login – timestamp – дата та час останнього входу в систему.

Таблиця: learning_materials_category

id – integer – унікальний ідентифікатор категорії, первинний ключ; name – varchar(100) – назва категорії навчальних матеріалів; slug – varchar(100) – URL-дружнє представлення назви; description – text – детальний опис категорії.

Таблиця: learning_materials_cefrlevel

id – integer – унікальний ідентифікатор рівня, первинний ключ; name – varchar(5) – код рівня CEFR (A1, A2, B1, B2, C1, C2); description – text – детальний опис рівня володіння мовою.

Таблиця: learning_materials_learningmaterial

id – integer – унікальний ідентифікатор навчального матеріалу, первинний ключ; title – varchar(200) – назва навчального матеріалу; slug – varchar(200) – URL-дружнє представлення назви; content – text – HTML-вміст матеріалу; category_id – integer – зовнішній ключ до категорії; cefr_level_id – integer – зовнішній ключ до рівня CEFR; author_id – integer – зовнішній ключ до користувача (автора); created_at – timestamp – дата створення; updated_at – timestamp – дата оновлення; is_published – boolean – статус публікації.

Таблиця: learning_materials_materialattachment

id – integer – унікальний ідентифікатор вкладення, первинний ключ; material_id – integer – зовнішній ключ до навчального матеріалу; file – varchar(100) – шлях до файлу; file_name – varchar(255) – оригінальна назва файлу; file_type – varchar(50) – тип файлу (аудіо, відео, документ, зображення);

uploaded_at – timestamp – дата завантаження.

Таблиця: tests_module_test

id – integer – унікальний ідентифікатор тесту, первинний ключ; title – varchar(200) – назва тесту; description – text – опис тесту; category_id – integer – зовнішній ключ до категорії; cefr_level_id – integer – зовнішній ключ до рівня CEFR; related_material_id – integer – зовнішній ключ до навчального матеріалу; author_id – integer – зовнішній ключ до користувача (автора); created_at – timestamp – дата створення; updated_at – timestamp – дата оновлення; is_published – boolean – статус публікації; time_limit – integer – обмеження часу (у хвилинах); passing_score – integer – мінімальний відсоток для проходження.

Таблиця: tests_module_question

id – integer – унікальний ідентифікатор питання, первинний ключ; test_id – integer – зовнішній ключ до тесту; question_text – text – текст питання; question_type – varchar(20) – тип питання (множинний вибір, відкрите, правда/неправда); order – integer – порядковий номер; points – integer – кількість балів.

Таблиця: tests_module_choice

id – integer – унікальний ідентифікатор варіанту, первинний ключ; question_id – integer – зовнішній ключ до питання; choice_text – text – текст варіанту; is_correct – boolean – ознака правильності.

Таблиця: tests_module_testattempt

id – integer – унікальний ідентифікатор спроби, первинний ключ; user_id – integer – зовнішній ключ до користувача; test_id – integer – зовнішній ключ до тесту; start_time – timestamp – час початку; end_time – timestamp – час завершення; score – integer – набрані бали; passed – boolean – статус проходження.

Таблиця: tests_module_testanswer

id – integer – унікальний ідентифікатор відповіді, первинний ключ; attempt_id – integer – зовнішній ключ до спроби; question_id – integer – зовнішній ключ до питання; selected_choice_id – integer – зовнішній ключ до вибраного варіанту; text_answer – text – відповідь користувача (для відкритих питань);

points_earned – integer – отримані бали.

Таблиця: tests_module_testresult

id – integer – унікальний ідентифікатор результату, первинний ключ; user_id – integer – зовнішній ключ до користувача; test_id – integer – зовнішній ключ до тесту; score – integer – загальний бал; passed – boolean – статус проходження; started_at – timestamp – час початку; completed_at – timestamp – час завершення; time_taken – integer – витрачений час (у секундах).

3.3. Вибір інструментарію для створення прикладного програмного забезпечення

Для розробки проєкту "English Learning" було обрано комплексний стек технологій, який забезпечує ефективну розробку, надійність та масштабованість системи. Вибір кожного компонента технологічного стеку був зумовлений конкретними потребами проєкту та перевагами, які надають ці технології.

Django було обрано як основний фреймворк для розробки серверної частини додатку завдяки його потужній архітектурі, що базується на принципі "batteries included". Цей фреймворк надає готові рішення для аутентифікації користувачів, адміністративну панель, систему маршрутизації URL та ORM для роботи з базою даних. Django дотримується принципу DRY (Don't Repeat Yourself) та забезпечує швидку розробку безпечних веб-додатків. Для проєкту з вивчення англійської мови це особливо важливо, оскільки дозволяє швидко створювати функціональні модулі для управління навчальними матеріалами, тестами та прогресом користувачів. Реалізація моделей представлена в додатку А. [2, 12]

PostgreSQL було обрано як систему управління базами даних через її надійність, відповідність стандартам ACID та підтримку складних запитів. Ця СУБД відмінно працює з Django через вбудований ORM та забезпечує високу продуктивність при роботі з великими обсягами даних. PostgreSQL підтримує розширені типи даних, що важливо для зберігання різноманітного контенту

навчальних матеріалів, включаючи текст, аудіо та відео. Крім того, PostgreSQL має потужні можливості для повнотекстового пошуку, що дозволяє ефективно шукати інформацію в навчальних матеріалах.

AJAX (Asynchronous JavaScript and XML) використовується для забезпечення асинхронної взаємодії між клієнтом і сервером без перезавантаження сторінки. Це критично важливо для інтерактивних елементів навчальної платформи, таких як проходження тестів, миттєва перевірка відповідей та динамічне оновлення контенту. Реалізація таких елементів показана в додатку Б. AJAX дозволяє створити плавний користувацький досвід, що особливо важливо для освітніх платформ, де комфорт користувача безпосередньо впливає на ефективність навчання.

JavaScript використовується для створення інтерактивних елементів інтерфейсу та обробки подій на стороні клієнта. Ця мова програмування дозволяє реалізувати динамічні компоненти, такі як таймери для тестів, інтерактивні вправи, анімації та валідацію форм. JavaScript також використовується для роботи з AJAX-запитами та маніпуляцій з DOM-елементами, що забезпечує швидку реакцію інтерфейсу на дії користувача. Фрагменти реалізації наведено в додатку Б. [20]

HTML та **CSS** є основою для створення структури та стилізації веб-сторінок. HTML5 надає семантичні елементи, які покращують доступність та SEO-оптимізацію сайту, а також підтримку мультимедійного контенту без сторонніх плагінів. CSS3 дозволяє створювати адаптивний дизайн, який коректно відображається на різних пристроях, від мобільних телефонів до настільних комп'ютерів, що важливо для забезпечення доступу до навчальних матеріалів з будь-якого пристрою.

Додатково у проєкті використовується **TinyMCE** — потужний WYSIWYG-редактор, який інтегрований для створення та редагування навчальних матеріалів. Він дозволяє авторам контенту форматувати текст, додавати медіафайли та створювати структуровані матеріали без знання HTML. Конфігурація TinyMCE у проєкті включає широкий набір інструментів для

форматування тексту, вставки зображень, таблиць та емоджі.

Widget Tweaks — бібліотека Django, яка спрощує роботу з формами у шаблонах, дозволяючи гнучко налаштовувати відображення полів форм без необхідності перевизначати їх у Python-коді. Це особливо корисно для створення зручних форм реєстрації, входу та форм для створення навчальних матеріалів і тестів. Реалізація на стороні серверної логіки продемонстрована в додатку В. [6, 24]

Обраний стек технологій забезпечує оптимальний баланс між швидкістю розробки, продуктивністю, безпекою та масштабованістю. Django та PostgreSQL формують надійний фундамент для серверної частини, тоді як JavaScript, AJAX, HTML та CSS дозволяють створити сучасний та інтуїтивно зрозумілий інтерфейс користувача. Додаткові бібліотеки, такі як TinyMCE та Widget Tweaks, розширюють функціональність основного стеку та спрощують розробку специфічних компонентів системи для вивчення англійської мови.

3.4. Алгоритмізація та програмування програмних модулів

Розробка веб-додатку для вивчення англійської мови вимагає ретельного проектування та реалізації програмних модулів, які забезпечують основну функціональність системи. У цьому розділі розглянуто основні алгоритми та програмні рішення, які були використані при розробці ключових модулів системи.

Одним із центральних модулів системи є модуль управління навчальними матеріалами. Він відповідає за створення, редагування, категоризацію та відображення навчальних матеріалів різних рівнів складності. Розглянемо основні алгоритми, реалізовані в цьому модулі.

Алгоритм відображення навчальних матеріалів передбачає отримання даних про категорії та рівні CEFR з бази даних, фільтрацію матеріалів за цими параметрами та відображення їх у зручному для користувача форматі. Цей алгоритм реалізовано у функції materials_list , яка обробляє HTTP-запити до

сторінки з навчальними матеріалами. Метод перевіряє параметри запиту, формує відповідний QuerySet з використанням ORM Django та передає дані у шаблон для відображення.

Алгоритм створення та редагування навчальних матеріалів є більш складним і включає кілька етапів: валідацію даних форми, збереження основної інформації про матеріал, обробку прикріплених файлів та встановлення зв'язків з категоріями та рівнями CEFR. Цей алгоритм реалізовано у функціях create_material та edit_material , які обробляють HTTP-запити до відповідних сторінок.

Особливу увагу приділено інтеграції WYSIWYG-редактора TinyMCE, який дозволяє авторам створювати форматований контент без знання HTML. Для цього використовується поле content моделі LearningMaterial , яке зберігає HTML-код, згенерований редактором. Для безпечного відображення цього контенту використовується фільтр safe у шаблонах Django.

Алгоритм управління вкладеннями до навчальних матеріалів реалізовано у функції upload_attachment , яка обробляє AJAX-запити на завантаження файлів. Цей алгоритм передбачає валідацію типу та розміру файлу, збереження файлу на сервері, створення запису в базі даних та повернення JSON-відповіді з інформацією про завантажений файл.

Для забезпечення безпеки та контролю доступу реалізовано механізм перевірки прав користувачів, який використовує декоратор permission_required . Цей декоратор перевіряє, чи має користувач необхідні права для виконання певних дій, таких як створення, редагування або видалення навчальних матеріалів.

Іншим важливим модулем системи є модуль тестування знань. Він відповідає за створення тестів, генерацію питань різних типів, проведення тестування та аналіз результатів. Розглянемо основні алгоритми цього модуля.

Алгоритм проходження тесту реалізовано у функції take_test , яка обробляє HTTP-запити до сторінки тестування. Цей алгоритм передбачає перевірку доступності тесту для користувача, створення запису про спробу проходження

тесту, відображення питань та варіантів відповідей, а також збереження відповідей користувача.

Особливістю реалізації є використання AJAX для асинхронного збереження відповідей користувача без перезавантаження сторінки. Це дозволяє забезпечити плавний користувацький досвід та зберегти прогрес у випадку технічних проблем або випадкового закриття сторінки.

Алгоритм перевірки відповідей та розрахунку результатів тесту реалізовано у функції submit_test , яка обробляє AJAX-запити на завершення тесту. Цей алгоритм передбачає порівняння відповідей користувача з правильними відповідями, розрахунок кількості балів за кожне питання та загального результату, а також збереження результатів у базі даних.

Для питань з множинним вибором реалізовано алгоритм перевірки, який враховує частково правильні відповіді. Якщо користувач вибрав деякі, але не всі правильні варіанти, або вибрав правильні варіанти разом з неправильними, він отримує частину балів пропорційно до кількості правильно вибраних варіантів.

Для відкритих питань реалізовано алгоритм перевірки на основі ключових слів. Відповідь користувача порівнюється з набором ключових слів, і якщо вона містить достатню кількість цих слів, відповідь вважається правильною. Цей алгоритм також враховує синоніми та різні форми слів.

Важливим аспектом системи є відстеження прогресу користувачів. Для цього реалізовано алгоритми збору та аналізу статистики, які дозволяють відображати інформацію про пройдені тести, отримані бали та рекомендації щодо подальшого навчання.

Для зручності користувачів реалізовано функцію адаптивного навчання, яка аналізує результати тестів та пропонує матеріали та тести відповідного рівня складності. Ця функція використовує алгоритм рекомендацій, який враховує історію навчання користувача, його сильні та слабкі сторони, а також загальну статистику проходження тестів іншими користувачами.

Таким чином, розроблені алгоритми та програмні рішення забезпечують ефективне функціонування системи для вивчення англійської мови, надаючи

користувачам зручний доступ до навчальних матеріалів та інструментів для перевірки знань.

4 РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ СИСТЕМИ

4.1. Тестування системи

Тестування є невід'ємною частиною процесу розробки програмного забезпечення, що забезпечує якість та надійність кінцевого продукту. В рамках розробки системи вивчення англійської мови було проведено комплексне тестування всіх компонентів та функціональних можливостей.

Загальний підхід до тестування

Для забезпечення високої якості системи було застосовано багаторівневий підхід до тестування, що включав:

1. Модульне тестування (Unit Testing) - перевірка окремих компонентів системи на відповідність їх функціональним вимогам. Кожен модуль тестувався окремо для виявлення помилок на ранніх етапах розробки.

2. Інтеграційне тестування - перевірка взаємодії між різними компонентами системи, такими як модуль навчальних матеріалів, модуль тестування, система авторизації тощо.

3. Функціональне тестування - перевірка відповідності системи функціональним вимогам, визначеним на етапі проектування.

4. Тестування користувацького інтерфейсу - перевірка зручності використання, адаптивності та естетичного вигляду інтерфейсу на різних пристроях.

5. Тестування продуктивності - перевірка швидкодії системи при різних навантаженнях та кількості користувачів.

6. Тестування безпеки - перевірка захищеності системи від несанкціонованого доступу та вразливостей.

Результати тестування основних компонентів

Сторінка **«Реєстрація»** є важливим компонентом системи, що дозволяє новим користувачам створити обліковий запис для доступу до функціоналу платформи вивчення англійської мови. Ця сторінка містить форму з полями для

введення основної інформації користувача, необхідної для створення облікового запису.

Форма реєстрації зображена на рис 4.1.

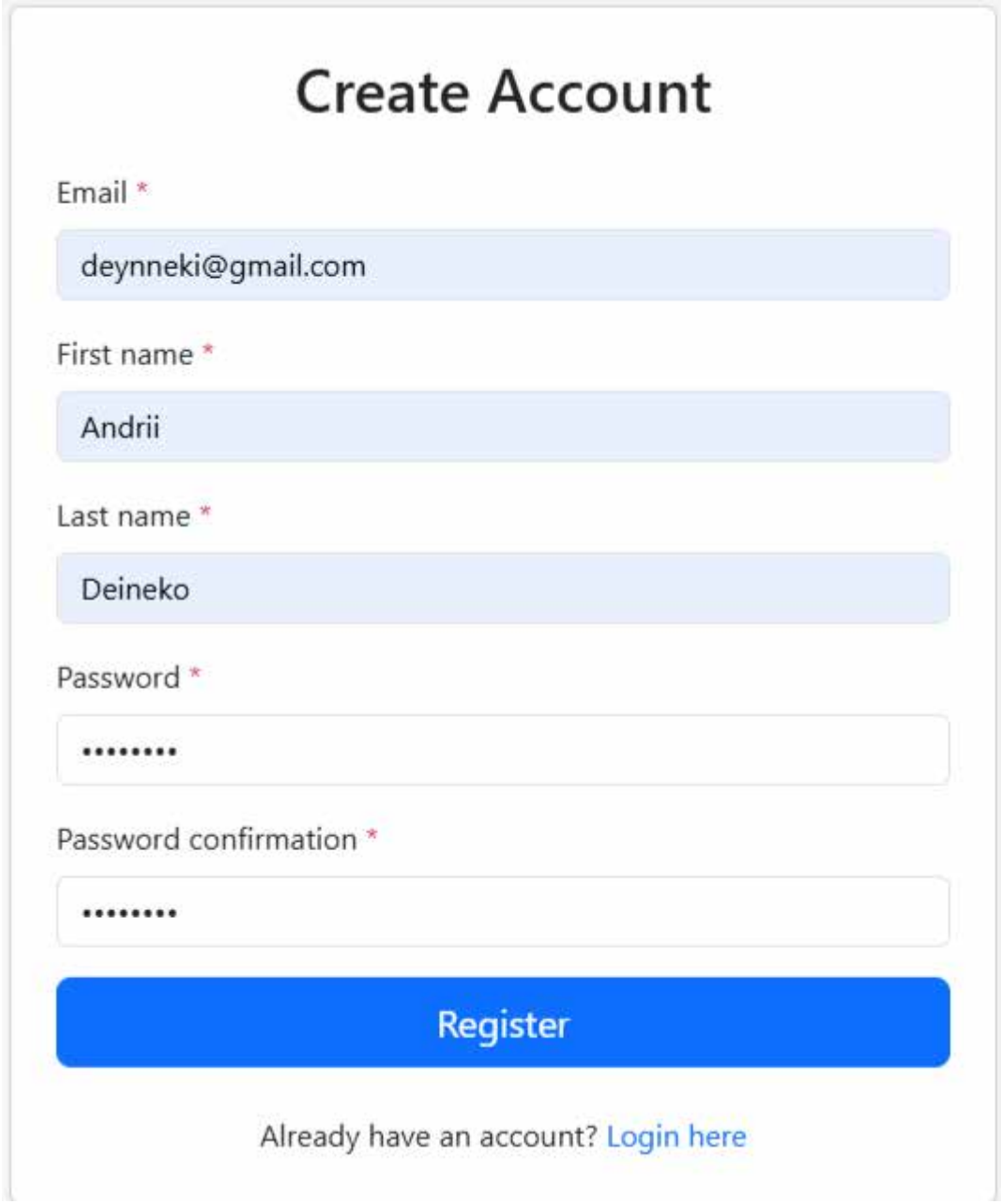

Рис 4.1 Форма «Реєстрація»

Тестування сторінки реєстрації пройшло успішно. Всі поля форми коректно відображаються та функціонують відповідно до вимог. Система належним чином перевіряє введені дані, зокрема валідність електронної пошти та відповідність паролів. Після успішної реєстрації користувач отримує повідомлення про необхідність підтвердження електронної пошти, що відповідає

запланованому сценарію використання. Інтерфейс сторінки інтуїтивно зрозумілий та відповідає загальному дизайну системи.

Сторінка **«Вхід в акаунт»** також є ключовим елементом системи, що забезпечує авторизацію користувачів та надає доступ до персоналізованих функцій платформи вивчення англійської мови. Ця сторінка містить форму з полями для введення електронної пошти та паролю, а також додаткові опції для зручності користувачів.

Форма входу в аккаунт зображена на рис 4.2.

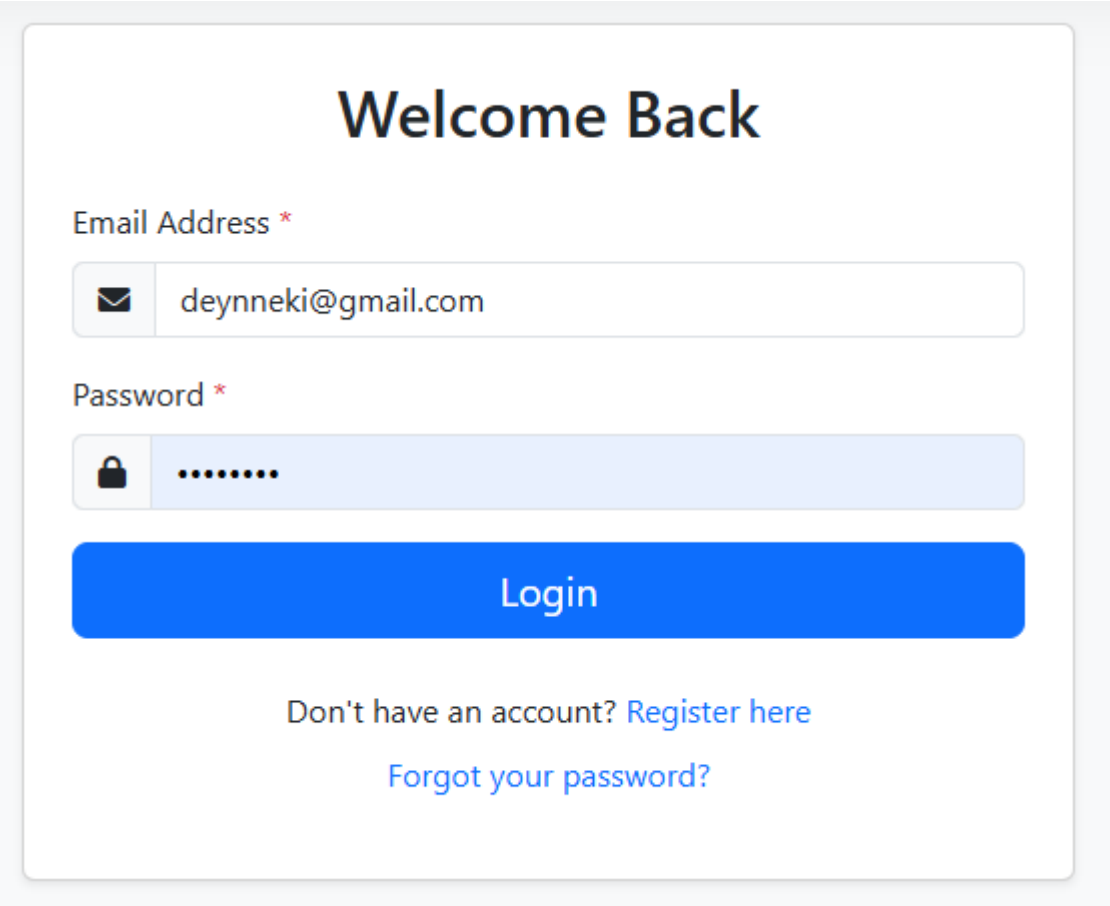

Рис 4.2 Форма «Вхід в акаунт»

Тестування сторінки входу в аккаунт пройшло успішно. Форма коректно відображається та функціонує, система належним чином перевіряє введені дані та надає відповідні повідомлення про помилки. Додаткові функції працюють коректно, а інтерфейс інтуїтивно зрозумілий та відповідає загальному дизайну системи.

Сторінка **"Редактор тестів"** надає користувачам з правами редактора

зручний інтерфейс для управління навчальними тестами. Вона дозволяє переглядати список створених тестів, створювати нові, редагувати існуючі та керувати їх статусом публікації.

Інтерфейс сторінки редактора тестів зображений на рис 4.3.

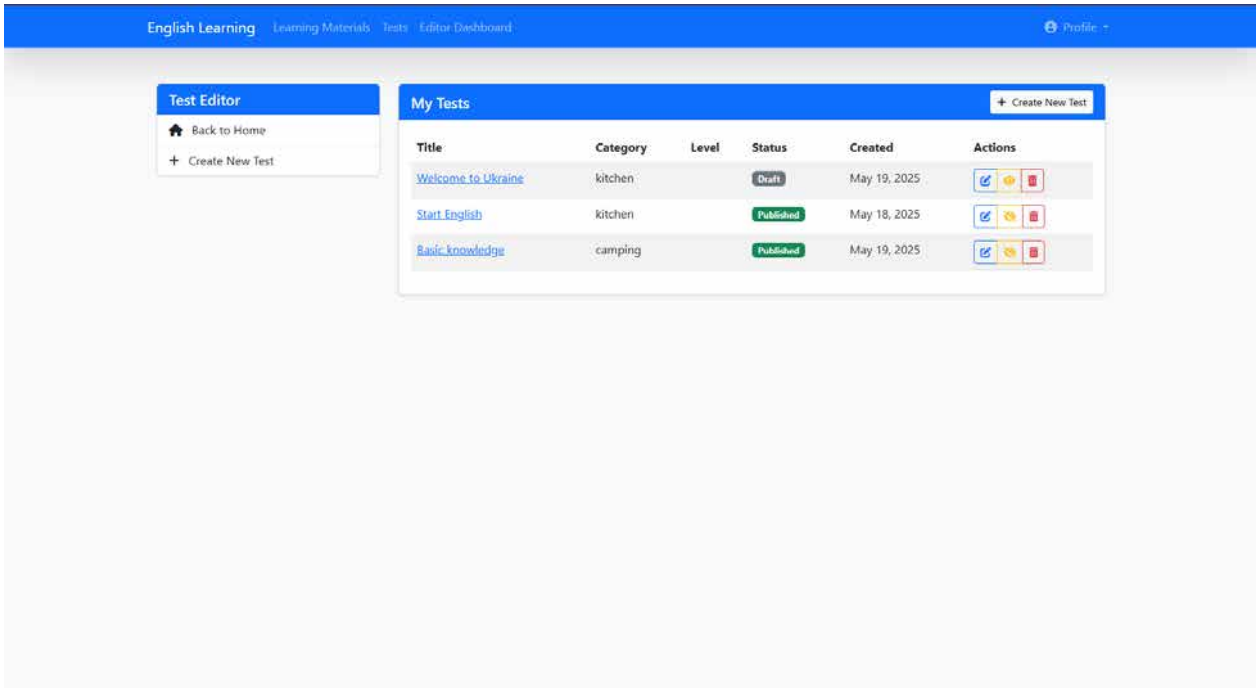

Рис 4.3 Сторінка "Створені тести"

Тестування сторінки редактора тестів продемонструвало високу якість реалізації з адаптивним дизайном, що коректно відображається на різних пристроях. Всі елементи інтерфейсу розташовані логічно та інтуїтивно зрозуміло для користувача. Таблиця тестів має чіткі кольорові індикатори статусу, а функціональність керування працює без затримок із системою підтвердження при видаленні. Загалом, сторінка повністю відповідає вимогам зручності використання та функціональності.

Сторінки **"Редагування тесту"** та **"Створення тесту"** надають користувачам з правами редактора зручний інтерфейс для управління тестами в системі вивчення англійської мови. Вони дозволяють налаштовувати всі параметри тесту, включаючи назву, опис, категорію, рівень складності та питання різних типів.

Інтерфейс сторінок редагування та створення тесту зображений на рис 4.4

та рис. 4.5.

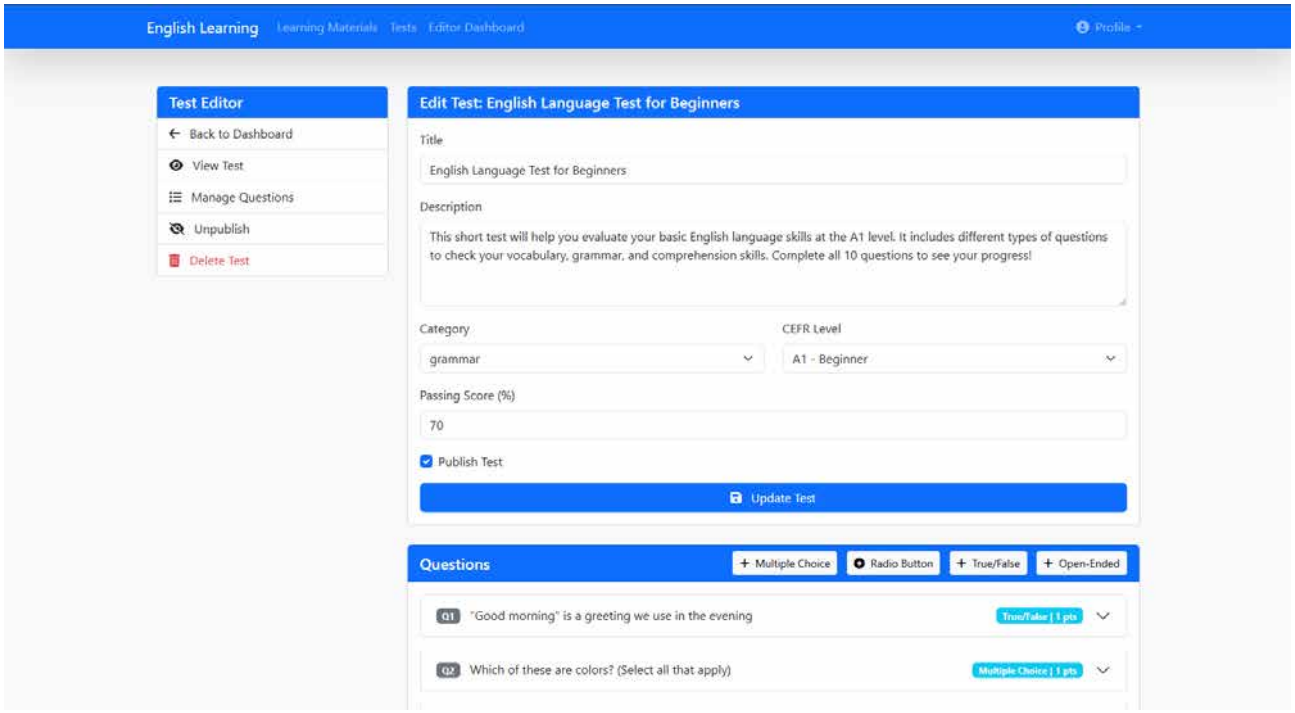

Рис 4.4 Сторінка "Редагувати тест"

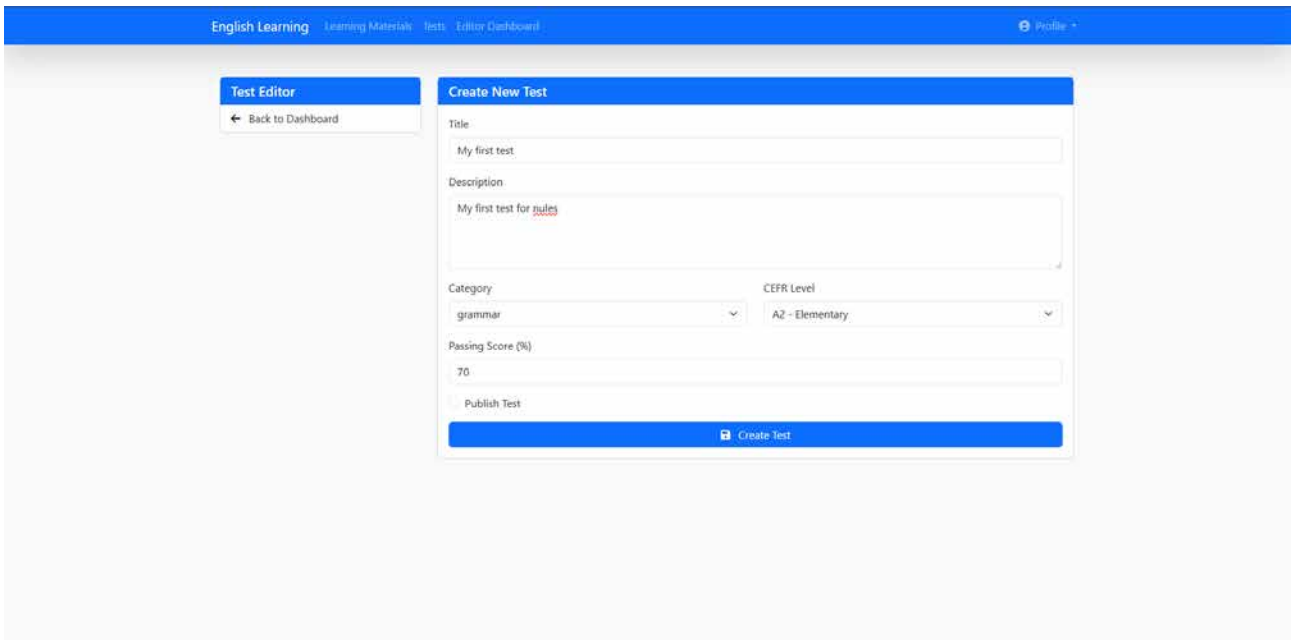

Рис 4.5 Сторінка "Створити тест"

Тестування сторінок редагування та створення тесту продемонструвало високу якість реалізації з адаптивним дизайном, що коректно відображається на різних пристроях. Всі елементи інтерфейсу розташовані логічно та інтуїтивно зрозуміло для користувача.

Редактор питань підтримує різні типи завдань (з вибором однієї відповіді, з множинним вибором, з введенням тексту та встановленням відповідностей), дозволяє налаштовувати часові обмеження, прохідний бал та порядок відображення питань. Система автоматично зберігає чернетки змін та забезпечує зручне керування статусом публікації тесту. Загалом, сторінки повністю відповідають вимогам зручності використання та функціональності для ефективного створення тестових завдань.

Інтерфейс сторінки "Editor Dashboard" зображений на рис 4.6.

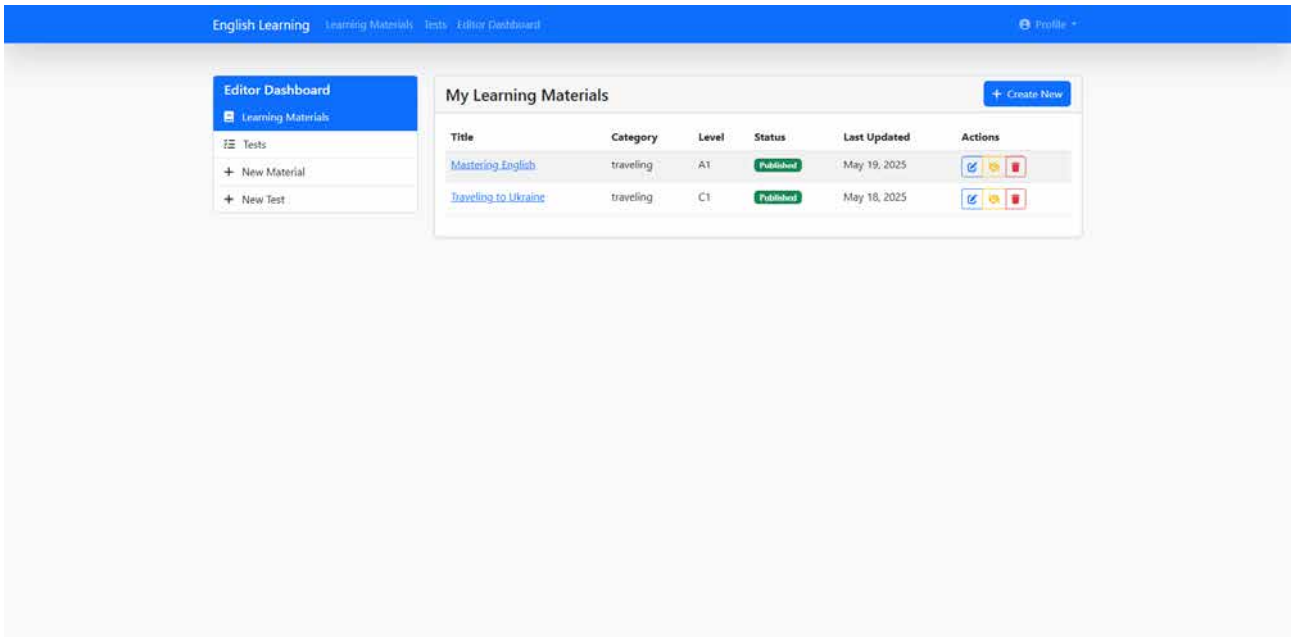

Рис 4.6 Сторінка "Панель редагування (Для навчальних матеріалів)"

У випадку відсутності створених матеріалів система відображає інформаційне повідомлення та пропонує створити перший матеріал. Кнопка "Create New" завжди доступна у верхній частині сторінки для швидкого доступу до форми створення нового навчального матеріалу. Такий формат представлення інформації є інтуїтивно зрозумілим та дозволяє ефективно управляти навчальними матеріалами в системі.

Сторінки **"Edit Learning material"** та **"Create Learning Material"** надають користувачам з правами редактора зручний інтерфейс для управління навчальними матеріалами в системі вивчення англійської мови. Вони дозволяють налаштовувати всі параметри матеріалу, включаючи назву, зміст,

категорію, рівень складності та додаткові файли.

Інтерфейс сторінок редагування та створення навчального матеріалу зображений на рис 4.7 та рис. 4.8.

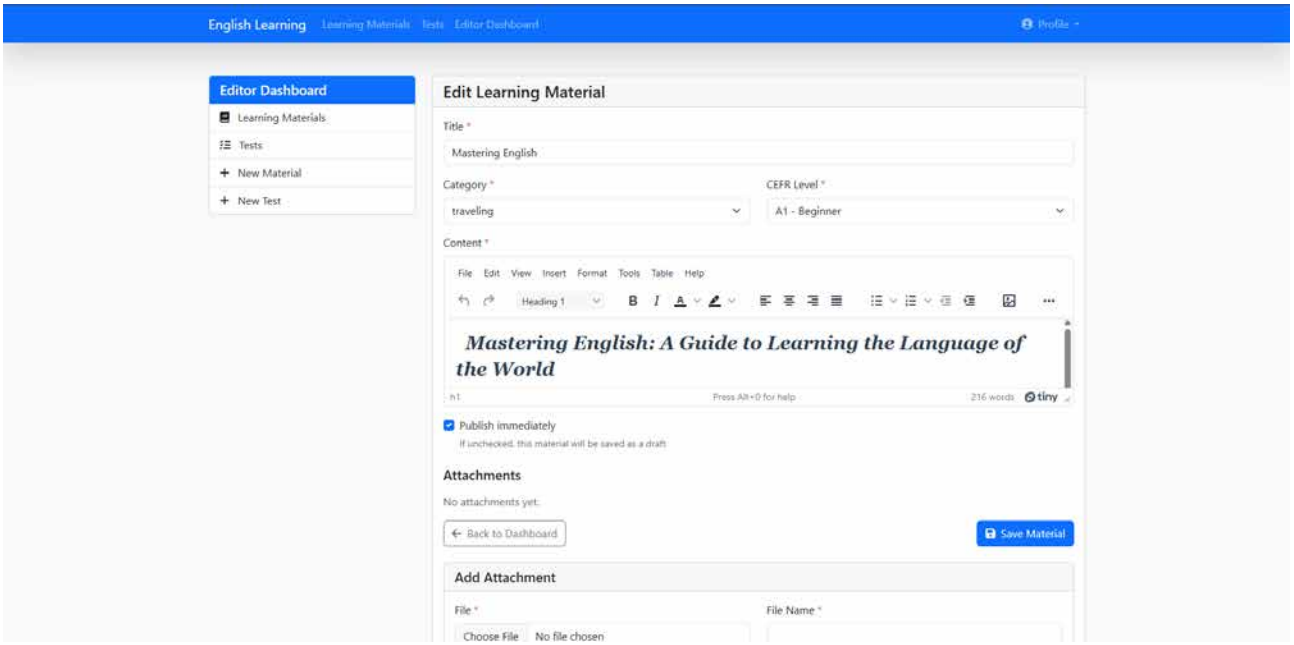

Рис 4.7 Сторінка "Редагування навчального матеріалу"

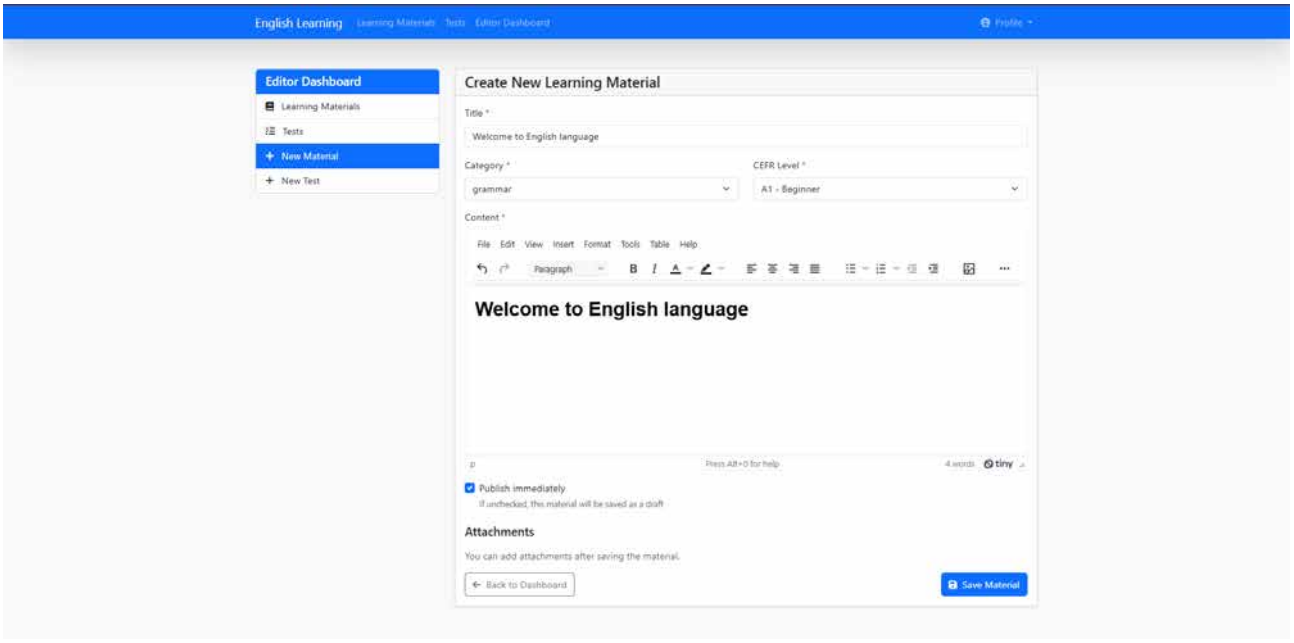

Рис 4.8 Сторінка "Створити навчальний матеріал"

Тестування сторінок редагування та створення навчального матеріалу продемонструвало високу якість реалізації з адаптивним дизайном, що коректно відображається на різних пристроях. Всі елементи інтерфейсу розташовані

логічно та інтуїтивно зрозуміло для користувача.

Редактор контенту дозволяє форматувати текст, додавати зображення та інші елементи, а також прикріплювати файли до матеріалу. Система автоматично генерує унікальні URL-адреси, забезпечує перевірку прав доступу та надає зручні інструменти для керування статусом публікації. Загалом, сторінки повністю відповідають вимогам зручності використання та функціональності для ефективного створення навчальних ресурсів.

Сторінка **"Профіль користувача"** надає зареєстрованим користувачам зручний інтерфейс для перегляду та редагування особистої інформації в системі вивчення англійської мови. Вона дозволяє налаштовувати основні параметри облікового запису, переглядати статистику навчання та керувати особистими даними.

Інтерфейс сторінки профілю користувача зображений на рис 4.9.

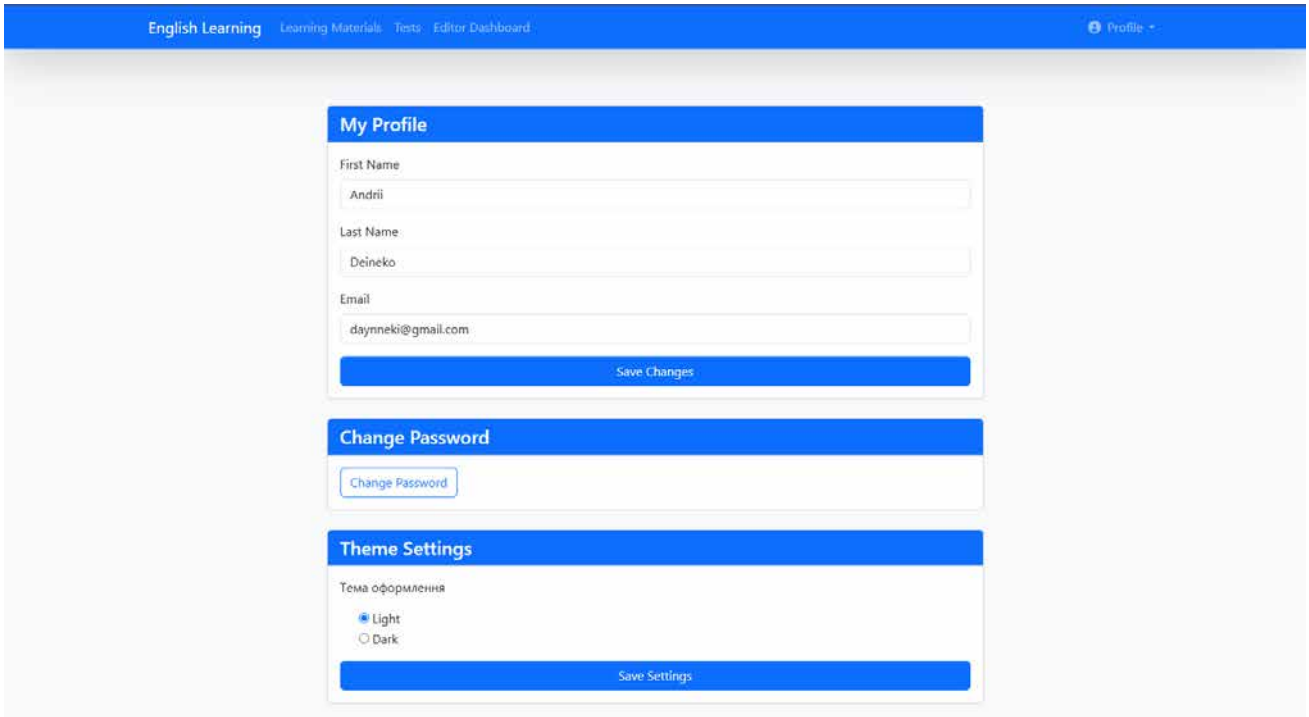

Рис 4.9 Сторінка "Профіль"

Тестування сторінки продемонструвало високу якість реалізації з адаптивним дизайном, що коректно відображається на різних пристроях. Всі елементи інтерфейсу розташовані логічно та інтуїтивно зрозуміло для користувача.

Профіль містить основну інформацію користувача (ім'я, прізвище, електронну пошту, дату реєстрації та рівень володіння англійською) з можливістю редагування особистих даних, завантаження аватару та оновлення паролю. На сторінці також відображається загальна статистика навчання та швидкі посилання на основні розділи системи. Загалом, сторінка профілю користувача повністю відповідає вимогам зручності використання та функціональності.

Сторінка **"Доступні тести"** надає користувачам зручний інтерфейс для перегляду та проходження тестів з англійської мови в системі навчання. Вона дозволяє фільтрувати тести за категоріями, рівнями складності та ключовими словами, що значно спрощує пошук необхідних завдань для перевірки знань.

Інтерфейс сторінки Доступні тести зображений на рис 4.10

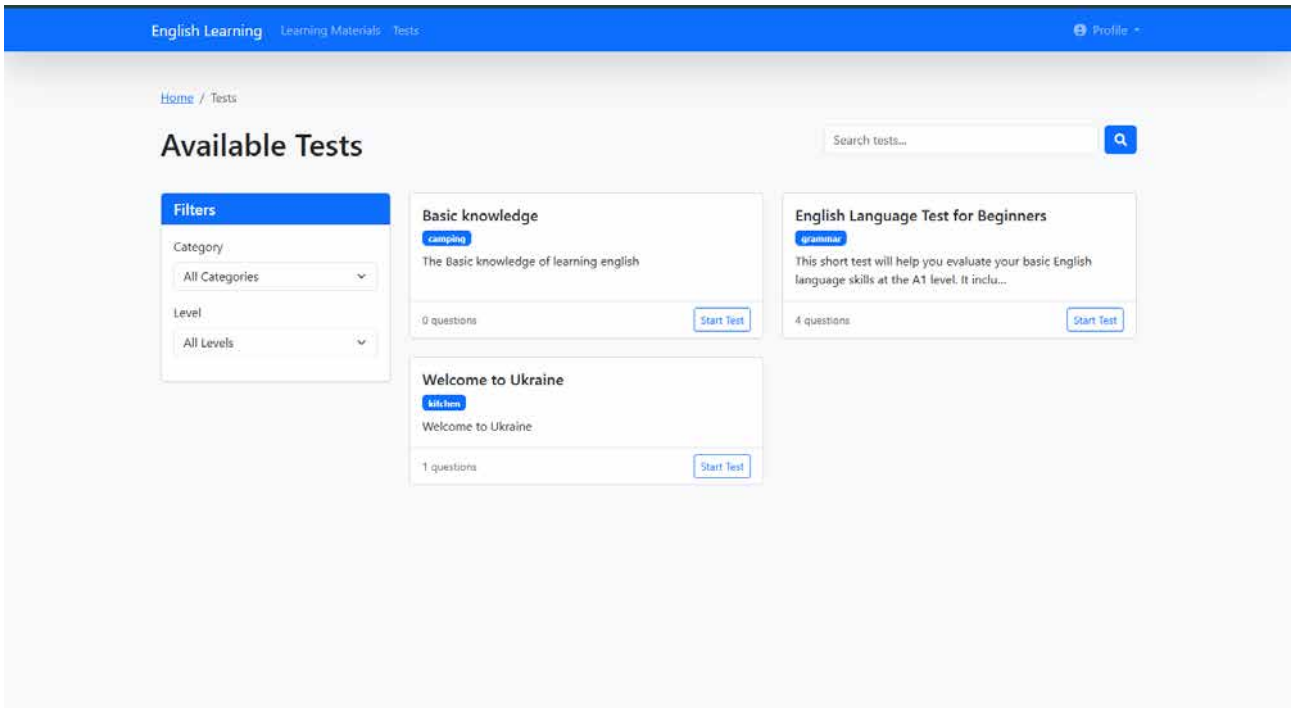

Рис 4.10 Сторінка "Доступні тести"

Тестування сторінки доступних тестів показало високу ефективність її реалізації. Сторінка швидко завантажується та адаптується до різних розмірів екрану. Система фільтрації працює коректно, дозволяючи користувачам швидко знаходити потрібні тести. Пагінація забезпечує зручну навігацію при великій кількості тестів, а інтерфейс є інтуїтивно зрозумілим для користувачів різного

рівня підготовки.

Сторінка **«Навчальні матеріали»** надає користувачам зручний інтерфейс для перегляду та пошуку доступних навчальних ресурсів у системі вивчення англійської мови. Вона дозволяє фільтрувати матеріали за категоріями, рівнями складності та ключовими словами, що значно спрощує пошук необхідної інформації.

Інтерфейс сторінки **«Навчальні матеріали»** зображений на рис 4.11.

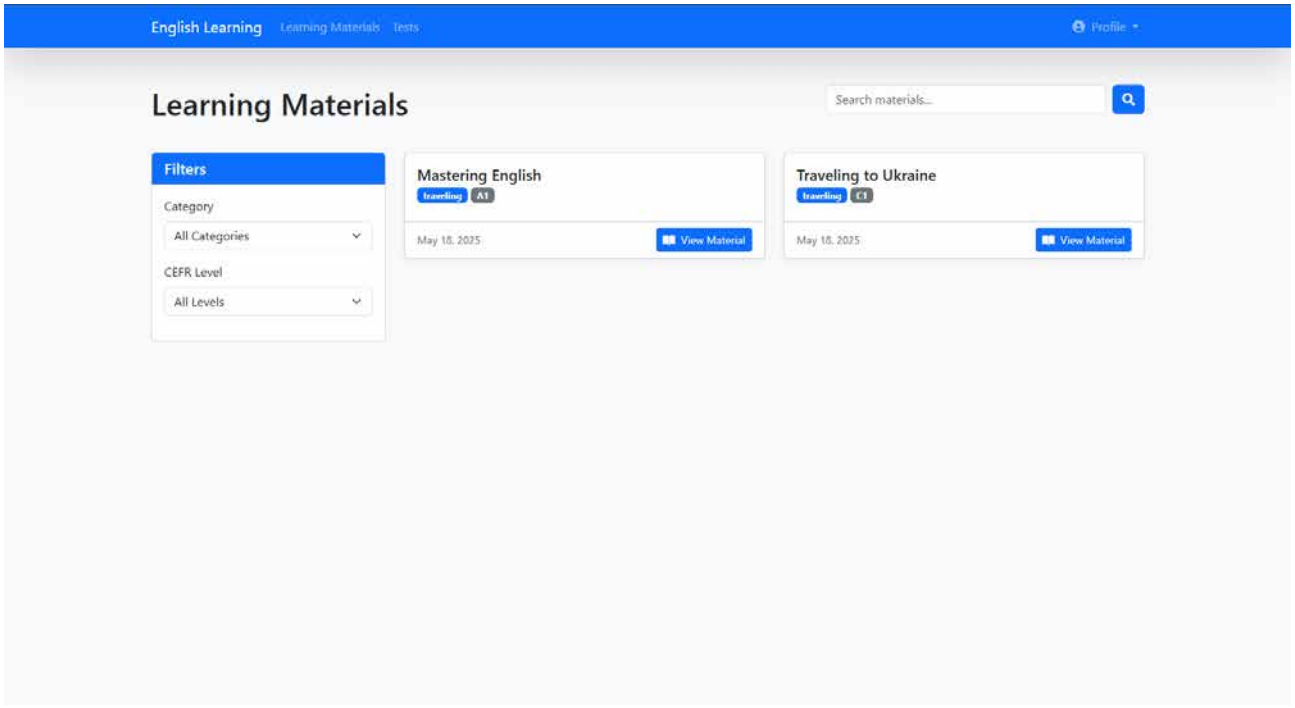

Рис 4.11 Сторінка "Навчальні матеріали"

Тестування сторінки навчальних матеріалів показало високу ефективність її реалізації. Сторінка швидко завантажується та адаптується до різних розмірів екрану. Система фільтрації працює коректно, дозволяючи користувачам швидко знаходити потрібні матеріали. Пагінація забезпечує зручну навігацію при великій кількості матеріалів, а інтерфейс є інтуїтивно зрозумілим для користувачів різного рівня підготовки.

Сторінка **«Перегляд конкретного навчального матеріалу»** надає користувачам зручний інтерфейс для детального ознайомлення з обраним навчальним ресурсом у системі вивчення англійської мови. Вона відображає повний зміст матеріалу, включаючи текст, зображення, таблиці та інші

мультимедійні елементи.

Інтерфейс сторінки «Перегляд навчального матеріалу» зображений на рис. 4.12

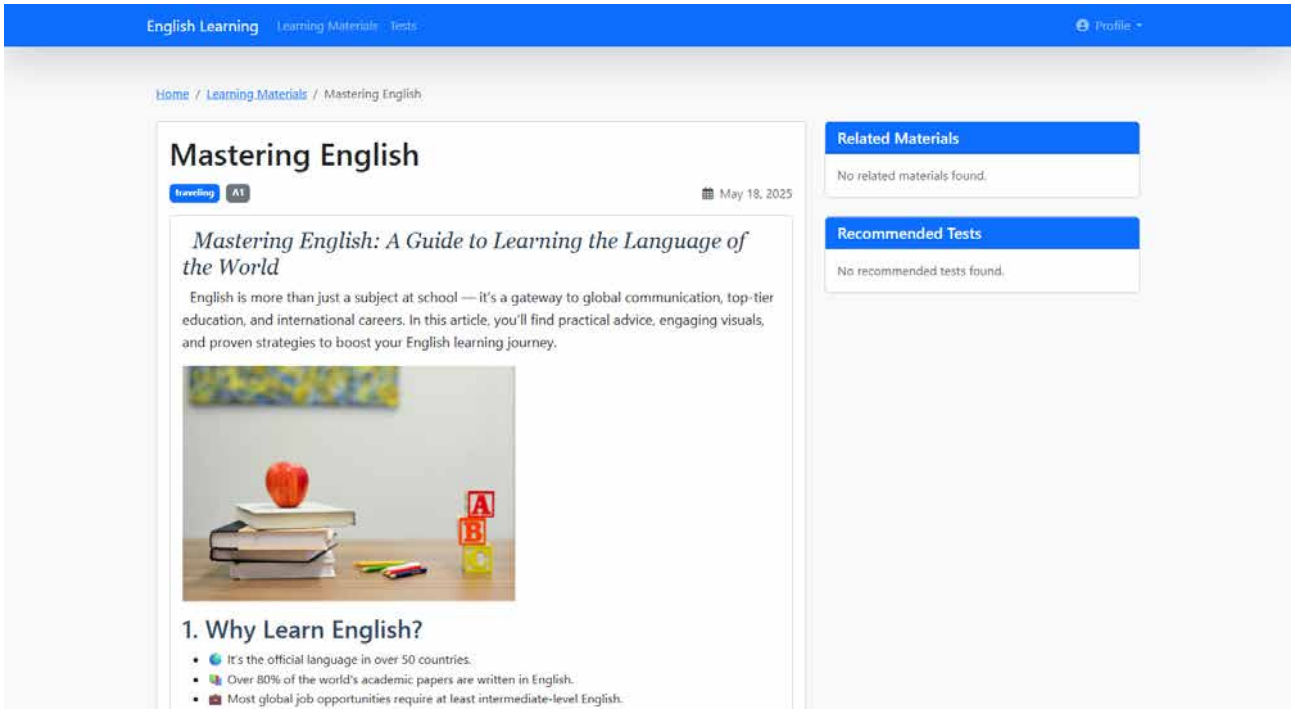

Рис 4.12 Сторінка "Перегляд навчального матеріалу"

Тестування сторінки перегляду матеріалу показало високу ефективність її реалізації. Сторінка швидко завантажується та адаптується до різних розмірів екрану. Контент відображається коректно, з належним форматуванням та структуруванням. Система також забезпечує зручний доступ до прикріплених файлів та пов'язаних матеріалів. Інтерфейс є інтуїтивно зрозумілим, з чіткою навігацією та можливістю повернення до списку матеріалів.

Сторінка **«Передперегляд тесту»** надає користувачам з правами редактора зручний інтерфейс для перевірки тесту перед його публікацією у системі вивчення англійської мови. Вона відображає тест точно так, як його побачать студенти, включаючи всі питання, варіанти відповідей та інструкції.

Інтерфейс сторінки «Передперегляд тесту» зображений на рис. 4.13.

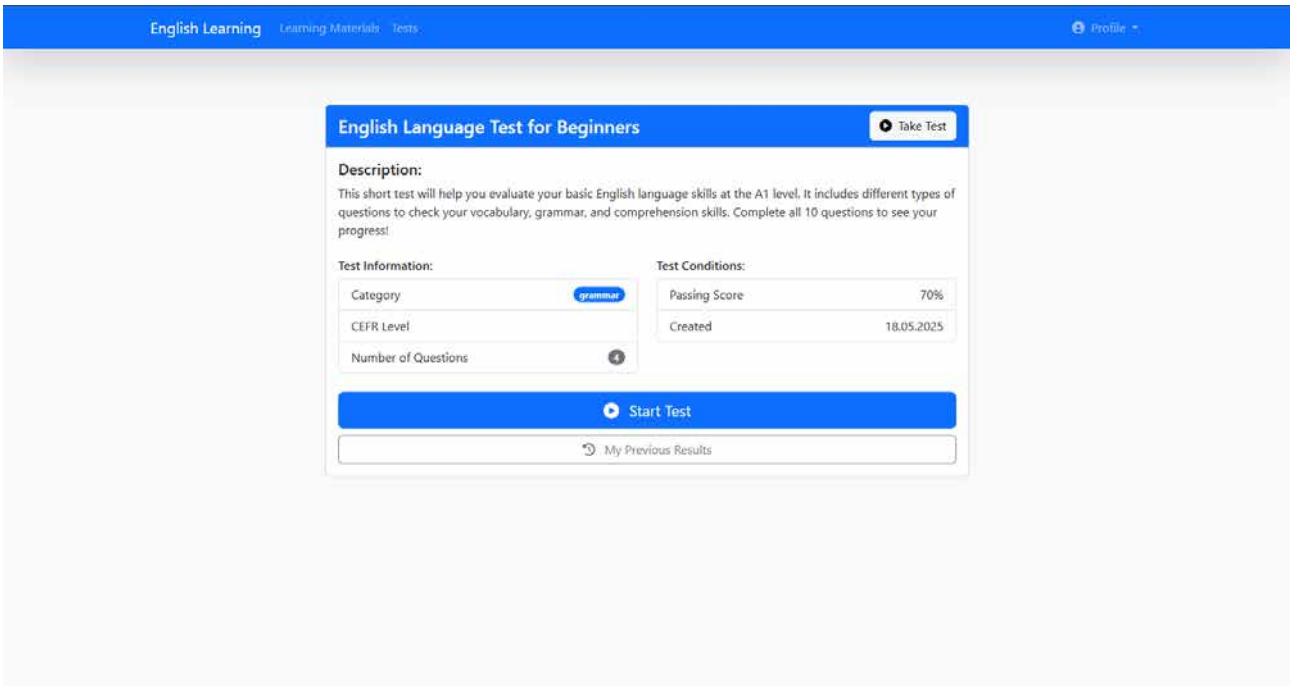

Рис 4.13 Сторінка "Передперегляд тесту"

Тестування сторінки передперегляду тесту показало високу ефективність її реалізації. Сторінка швидко завантажується та адаптується до різних розмірів екрану. Всі елементи тесту відображаються коректно, з належним форматуванням та структуруванням. Система також забезпечує можливість перевірки функціональності тесту без збереження результатів у базі даних.
Інтерфейс є інтуїтивно зрозумілим, з чіткою навігацією між питаннями та можливістю повернення до редагування тесту. Редактор може легко перевірити правильність налаштувань тесту, коректність питань та відповідей, а також загальний вигляд тесту з точки зору студента перед його публікацією.

4.2. Вимоги до апаратного та програмного забезпечення

Апаратне забезпечення

Серверна частина

Для забезпечення стабільної роботи системи вивчення англійської мови серверна частина повинна відповідати наступним мінімальним вимогам: процесор з щонайменше 4 ядрами (рекомендовано 8 ядер), оперативна пам'ять об'ємом не менше 8 ГБ (рекомендовано 16 ГБ), та дисковий простір не менше

100 ГБ на SSD-накопичувачі для забезпечення швидкого доступу до даних.

Важливою вимогою є наявність стабільного та швидкого підключення до мережі Інтернет з гігабітною пропускною здатністю для обслуговування великої кількості одночасних користувачів системи.

Клієнтська частина

Для комфортної роботи з системою клієнтські пристрої повинні мати сучасний процесор (Intel Core i3/AMD Ryzen 3 або вище), щонайменше 4 ГБ оперативної пам'яті та мінімум 1 ГБ вільного місця на диску для кешування даних.

Необхідне стабільне підключення до Інтернету зі швидкістю від 5 Мбіт/с для забезпечення плавної роботи з інтерактивними елементами системи та мультимедійним контентом.

Мобільні пристрої

Для використання системи на мобільних пристроях необхідна операційна система Android 7.0+ або iOS 12.0+, щонайменше 2 ГБ оперативної пам'яті та дисплей з діагоналлю від 5 дюймів з роздільною здатністю не менше 1280x720 пікселів.

Програмне забезпечення

Серверна частина

Для розгортання серверної частини системи рекомендується використовувати операційну систему Ubuntu Server 20.04 LTS або новішу версію, веб-сервер Nginx 1.18+ або Apache 2.4+, систему управління базами даних PostgreSQL 12+ або MySQL 8.0+.

Середовище виконання має базуватися на Python 3.8+ з використанням фреймворку Django 3.2+ або 4.0+. Для підвищення продуктивності рекомендується використовувати систему кешування Redis 6.0+.

Опціонально можливе використання технологій контейнеризації Docker 20.10+ та Docker Compose 2.0+ для спрощення розгортання та масштабування системи.

Клієнтська частина

Система підтримує роботу в сучасних веб-браузерах: Google Chrome 90+, Mozilla Firefox 88+, Safari 14+, Microsoft Edge 90+ та Opera 76+. Необхідна підтримка технологій HTML5, CSS3 та JavaScript ES6+.

Для коректної роботи всіх функцій системи в браузері користувача повинні бути увімкнені JavaScript та cookies, а також забезпечена підтримка технології WebSockets для реалізації функцій реального часу.

Мобільні додатки

Для розробки мобільних додатків системи використовуються технології, що забезпечують підтримку Android з мінімальною версією API 24 (Android 7.0) та iOS з мінімальною версією 12.0.

Рекомендовано використання кросплатформних фреймворків React Native 0.65+ або Flutter 2.5+ для забезпечення єдиної кодової бази та спрощення підтримки додатків.

Вимоги до безпеки

Система повинна забезпечувати шифрування даних при передачі за допомогою протоколу TLS 1.3, підтримувати двофакторну автентифікацію користувачів та шифрування чутливих даних у базі даних.

Важливою вимогою є наявність системи щоденного резервного копіювання з можливістю швидкого відновлення даних у разі виникнення непередбачуваних ситуацій.

Масштабованість

Архітектура системи повинна підтримувати горизонтальне масштабування через додавання нових серверів та балансування навантаження між ними. Рекомендовано використання багаторівневої системи кешування для оптимізації продуктивності.

Система розроблена з урахуванням можливості розгортання як у хмарному середовищі (AWS, Google Cloud, Azure), так і на власній інфраструктурі замовника, що забезпечує гнучкість у виборі варіанту розміщення залежно від потреб та можливостей.

4.3. Склад інсталяційного пакету

Загальна структура проєкту

Проєкт "English Learning Platform" розроблено на основі фреймворку Django та складається з кількох взаємопов'язаних модулів, що забезпечують повноцінне функціонування системи для вивчення англійської мови. Інсталяційний пакет містить наступні основні компоненти:

1. Модуль core - відповідає за автентифікацію користувачів, управління профілями, верифікацію електронної пошти та базові функції системи.

2. Модуль learning_materials - забезпечує створення, зберігання та відображення навчальних матеріалів різних типів.

3. Модуль tests_module - дозволяє створювати та проходити тести для перевірки знань з різними типами запитань.

4. Шаблони та статичні файли - забезпечують інтерфейс користувача та візуальне оформлення платформи.

5. Конфігураційні файли - містять налаштування проєкту, URL-маршрутизацію та інші системні параметри.

Отримання вихідного коду

Першим кроком розгортання проєкту є отримання його вихідного коду. Для цього необхідно клонувати репозиторій проєкту з системи контролю версій. Після успішного клонування репозиторію потрібно перейти до директорії проєкту. Надалі для ізоляції залежностей проєкту важливо створити віртуальне середовище Python. Після активації віртуального середовища командний рядок відображатиме назву віртуального середовища, що свідчить про його успішну активацію.

Налаштування віртуального середовища

Для зручності управління залежностями проєкту рекомендується використовувати віртуальне середовище Python, що дозволяє ізолювати залежності проєкту від інших проєктів на тій самій системі. Це забезпечує стабільність роботи додатку та запобігає конфліктам між різними версіями

бібліотек. Проєкт містить файл requirements.txt, який перелічує всі необхідні залежності. Виконання команди з взаємодією з ним дозволить інсталювати всі необхідні бібліотеки Python, включаючи Django, драйвер для роботи з PostgreSQL, бібліотеки для роботи з формами, TinyMCE для редагування тексту та інші компоненти, необхідні для функціонування веб-додатку.

Налаштування бази даних

Для роботи веб-додатку необхідно створити базу даних PostgreSQL. Спочатку необхідно встановити сам PostgreSQL, якщо він ще не встановлений в системі. Після встановлення потрібно створити базу даних та користувача для проєкту. У файлі налаштувань проєкту вже вказані параметри підключення до бази даних, які можна змінити відповідно до локальних налаштувань.

Конфігурація змінних середовища

Для безпечного зберігання чутливих даних, таких як паролі та ключі API, проєкт використовує змінні середовища. Для цього потрібно створити файл .env у кореневій директорії проєкту та додати до нього відповідні змінні. Детальніше налаштування будуть описані нижче

Налаштування параметрів середовища (environment variables):

SECRET_KEY – your_secret_key

DEBUG – True

ALLOWED_HOSTS – localhost, 127.0.0.1

Параметри бази даних:

DB_NAME – english_learning_db

DB_USER – postgres

DB_PASSWORD – your_password

DB_HOST – localhost

DB_PORT – 5432

Параметри електронної пошти:

EMAIL_HOST – smtp.gmail.com

EMAIL_PORT – 587

EMAIL_HOST_USER – your_email@gmail.com

EMAIL_HOST_PASSWORD – your_app_password

EMAIL_USE_TLS – True

DEFAULT_FROM_EMAIL – your_email@gmail.com

Міграція бази даних

Після налаштування бази даних та змінних середовища необхідно виконати міграції для створення необхідних таблиць у базі даних. Міграції - це спосіб, яким Django відстежує зміни в моделях даних та застосовує їх до схеми бази даних. Виконання міграцій забезпечить створення всіх необхідних таблиць та зв'язків між ними.

Створення суперкористувача

Щоб мати доступ до адміністративної панелі веб-додатку потрібно створити суперкористувача. У командному рядку будуть відповідні інструкції, щоб ввести ім'я користувача, електронну пошту та пароль для суперкористувача. Створений суперкористувач матиме повний доступ до адміністративної панелі та всіх функцій системи.

Збір статичних файлів

Для правильного відображення стилів, скриптів та зображень у веб-додатку необхідно зібрати статичні файли. Надалі всі вони з різних додатків скопіюються у єдину директорію, що спрощує їх обслуговування веб-сервером. Це особливо важливо при розгортанні проєкту в виробничому середовищі.

Запуск сервера розробки

Для тестування веб-додатку в середовищі розробки потрібно запустити вбудований сервер Django. Після запуску сервера веб-додаток буде доступний за адресою http://127.0.0.1:8000/ . Адміністративна панель доступна за адресою http://127.0.0.1:8000/admin/ . Через адміністративну панель можна керувати користувачами, навчальними матеріалами, тестами та іншими компонентами системи.

Особливості налаштування для виробничого середовища

При розгортанні проєкту в виробничому середовищі необхідно внести додаткові налаштування для забезпечення безпеки та продуктивності:

1. Встановити параметр DEBUG=False у файлі .env для вимкнення режиму відлагодження.

2. Налаштувати ALLOWED_HOSTS відповідно до доменного імені сервера.

3. Налаштувати веб-сервер (наприклад, Nginx) для обслуговування статичних файлів.

4. Налаштувати WSGI-сервер (наприклад, Gunicorn) для запуску Django-додатку.

5. Налаштувати SSL-сертифікати для забезпечення шифрованого з'єднання.

Заходи безпеки

Для забезпечення безпеки системи інсталяційний пакет включає ряд заходів:

1. Використання змінних середовища - чутливі дані (паролі, ключі API) зберігаються у змінних середовища, а не в коді.

2. Налаштування HTTPS - інструкції з налаштування SSL-сертифікатів для забезпечення шифрованого з'єднання.

3. Налаштування брандмауера - рекомендації з налаштування брандмауера для обмеження доступу до сервера.

4. Регулярні оновлення - інструкції з оновлення залежностей для усунення вразливостей.

5. Обмеження прав доступу - рекомендації з налаштування прав доступу до файлів та каталогів.

6. Верифікація електронної пошти - система вимагає підтвердження електронної пошти для активації облікового запису.

7. Безпечне відновлення паролю - реалізовано механізм безпечного відновлення паролю через електронну пошту.

Дотримання цих рекомендацій забезпечить надійну роботу платформи для вивчення англійської мови та захист даних користувачів.

ВИСНОВКИ

У процесі виконання дипломної роботи було реалізовано повнофункціональну веб орієнтовану систему для вивчення англійської мови. Система створена з урахуванням сучасних освітніх потреб, що зумовлено широким використанням інформаційних технологій у навчальному процесі та зростанням попиту на доступні, гнучкі й персоналізовані платформи для вивчення іноземних мов.

На основі системного аналізу предметної області було визначено основні проблеми традиційного мовного навчання, потреби користувачів та особливості засвоєння англійської мови українською аудиторією. Це дозволило сформувати обґрунтовані функціональні та нефункціональні вимоги до системи. Було враховано ключові аспекти: адаптивність навчання, структурованість матеріалів, інтерактивність, підтримка тестування, збереження результатів та захист персональних даних.

У ході роботи було спроєктовано архітектуру системи за допомогою діаграм UML (прецедентів, класів, пакетів, компонентів та кооперацій), що дозволило детально описати структуру, логіку взаємодії компонентів і функціонування користувацьких сценаріїв. Для зберігання навчальних матеріалів, тестів і результатів навчання створено логічну ER-модель даних, яка забезпечує масштабованість та надійну роботу з базою даних.

На етапі реалізації система була побудована з використанням фреймворку Django, що забезпечило швидку розробку, підтримку модульної архітектури та відповідність сучасним стандартам безпеки. Було реалізовано модулі для реєстрації, авторизації, управління навчальними матеріалами, проходження тестів, перегляду статистики прогресу. Інтерфейс системи є адаптивним і зручним для користувачів різного рівня підготовки.

У процесі тестування система продемонструвала стабільну роботу, відповідність заданим функціональним характеристикам, швидкість реакції на дії користувача та захищеність даних. Платформа надає можливість створювати

власні навчальні курси, проходити тести з автоматичною перевіркою результатів, а також відслідковувати прогрес у навчанні за допомогою зручної системи звітності.

Отримані результати свідчать про досягнення всіх поставлених завдань. Система повністю відповідає технічному завданню та має значний потенціал для практичного використання в освітніх установах, на онлайн-курсах або як засіб самостійного навчання.

Таким чином, створена платформа є сучасним, ефективним та доступним інструментом для вивчення англійської мови, що поєднує педагогічні принципи з технологічними можливостями та відповідає реальним потребам користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. About Us. Visual Paradigm. URL: https://www.visual-paradigm.com/aboutus/ (дата звернення: 10.03.2025).
2. Advanced Features of Django for Professional Projects. Nucamp. URL: https://www.nucamp.co/blog/coding-bootcamp-back-end-with-python-and-sql-advanced-features-of-django-for-professional-projects (дата звернення: 14.04.2025).
3. BBC Learning English. Wikipedia. URL: https://en.wikipedia.org/wiki/BBC_Learning_English (дата звернення: 26.02.2025).
4. Collaboration diagram. TechTarget. URL: https://www.techtarget.com/searchsoftwarequality/definition/collaboration-diagram (дата звернення: 04.01.2025).
5. Collaboration Diagrams (Unified Modeling Language). GeeksforGeeks. URL: https://www.geeksforgeeks.org/collaboration-diagrams-unified-modeling-languageuml/ (дата звернення: 08.03.2025).
6. django-widget-tweaks. PyPI. URL: https://pypi.org/project/django-widget-tweaks/ (дата звернення: 30.01.2025).
7. Full-Text Search. Django Documentation. URL: https://docs.djangoproject.com/en/5.2/ref/contrib/postgres/search/ (дата звернення: 22.02.2025).
8. Functional Requirements. Visure Solutions. URL: https://visuresolutions.com/uk/alm-guide/functional-requirements/ (дата звернення: 19.03.2025).
9. Models. Django Documentation. URL: https://docs.djangoproject.com/en/5.1/topics/db/models/ (дата звернення: 01.02.2025).
10. PostgreSQL проти MySQL: яку СУБД обрати? DreamHost. URL: https://www.dreamhost.com/blog/uk/postgresql-proti-mysql/ (дата звернення: 25.04.2025).
11. Structural Constraints of Relationships in ER Model. GeeksforGeeks.

URL: https://www.geeksforgeeks.org/structural-constraints-of-relationships-in-er-model/ (дата звернення: 17.01.2025).

12. What is Django Web Framework? GeeksforGeeks. URL: https://www.geeksforgeeks.org/what-is-django-web-framework/ (дата звернення: 03.02.2025).

13. What is Package Diagram? Visual Paradigm. URL: https://www.visual-paradigm.com/cn/guide/uml-unified-modeling-language/what-is-package-diagram/ (дата звернення: 02.04.2025).

14. Важливість вивчення англійської мови у сучасному світі. BUKI. URL: https://buki.com.ua/blogs/vazlivist-vivcennya-angliiskoyi-movi-u-sucasnomu-sviti/ (дата звернення: 06.05.2025).

15. Дейнеко А. А. Програмне забезпечення веб орієнтованої системи для вивчення англійської мови. Збірник наукових праць за матеріалами VII Всеукраїнської науково-практичної конференції студентів і аспірантів «Теоретичні та прикладні аспекти розробки комп'ютерних систем 2025», м. Київ, 23 травня 2025 р. НУБіП України. Київ, 2025.

16. Діаграма класів. URL: https://uk.wikipedia.org/wiki/Діаграма_класів (дата звернення: 12.01.2025).

17. Діаграма компонентів. URL: https://uk.wikipedia.org/wiki/Діаграма_компонентів (дата звернення: 09.02.2025).

18. Діаграма прецедентів. URL: https://uk.wikipedia.org/wiki/Діаграма_прецедентів (дата звернення: 28.03.2025).

19. Мовою мільйонів: історія створення та розвитку популярного сервісу Duolingo. Investory News. URL: https://investory.news/movoyu-miljoniv-istoriya-stvorennya-ta-rozvitku-populyarnogo-servisu-duolingo/ (дата звернення: 20.04.2025).

20. Мови програмування для веб-розробки та супутні технології. Друкарня.com.ua. URL: https://drukarnia.com.ua/articles/movi-programuvannya-dlya-vebrozrobki-ta-suputni-tekhnologiyi-wzeAr (дата звернення: 29.03.2025).

21. Модель «сутність — зв'язок». URL:

https://uk.wikipedia.org/wiki/Модель_«сутність_—_зв'язок» (дата звернення: 04.02.2025).

22. Нефункціональні вимоги: приклади, типи, підходи. QATestLab Training Center. URL: https://training.qatestlab.com/blog/technical-articles/non-functional-requirements-examples-types-approaches/ (дата звернення: 05.01.2025).

23. Огляд Draw.io: функції, переваги та недоліки. MindOnMap. URL: https://www.mindonmap.com/uk/blog/drawio-review/ (дата звернення: 07.03.2025).

24. Посібник по Django для початківців. DevZone. URL: https://devzone.org.ua/post/posibnyk-po-django-dlia-pochatkivtsiv-chastyna-3 (дата звернення: 11.04.2025).

ДОДАТОК А

Моделі Django ORM тестового модуля

```
from django.db import models
from django.conf import settings
from learning_materials.models import Category, CEFRLevel, LearningMaterial

class Test(models.Model):
    title = models.CharField(max_length=200)
    description = models.TextField()
    category = models.ForeignKey(Category, on_delete=models.CASCADE)
    cefr_level = models.ForeignKey(CEFRLevel, on_delete=models.SET_NULL, null=True)
    related_material = models.ForeignKey(LearningMaterial, on_delete=models.SET_NULL, null=True, blank=True)
    author = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
    is_published = models.BooleanField(default=False)
    passing_score = models.IntegerField(help_text="Minimum score to pass (percentage)", default=70)

    def __str__(self):
        return self.title

class Question(models.Model):
    MULTIPLE_CHOICE = 'multiple_choice'
    OPEN_ENDED = 'open_ended'
    TRUE_FALSE = 'true_false'
    RADIO_BUTTON = 'radio_button'

    QUESTION_TYPES = [
        (MULTIPLE_CHOICE, 'Multiple Choice'),
        (OPEN_ENDED, 'Open Ended'),
```

```
            (TRUE_FALSE, 'True/False'),
            (RADIO_BUTTON, 'Radio Button'),
        ]

        test = models.ForeignKey(Test, on_delete=models.CASCADE, related_name='questions')
        question_text = models.TextField()
        question_type = models.CharField(max_length=20, choices=QUESTION_TYPES, default=MULTIPLE_CHOICE)
        order = models.IntegerField(default=0)
        points = models.IntegerField(default=1)

        class Meta:
            ordering = ['order']

        def __str__(self):
            return f"{self.test.title} - Question {self.order}"

    class Choice(models.Model):
        question = models.ForeignKey(Question, on_delete=models.CASCADE, related_name='choices')
        choice_text = models.CharField(max_length=200)
        is_correct = models.BooleanField(default=False)

        def __str__(self):
            return self.choice_text

    class TestResult(models.Model):
        user = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE)
        test = models.ForeignKey(Test, on_delete=models.CASCADE)
        score = models.IntegerField()
        passed = models.BooleanField(default=False)
        started_at = models.DateTimeField(auto_now_add=True)
        completed_at = models.DateTimeField(null=True, blank=True)
        time_taken = models.IntegerField(help_text="Time taken in seconds", null=True,
```

```
blank=True)

        def __str__(self):
            return f"{self.user.email} - {self.test.title} - {self.score}%"

    class TestAttempt(models.Model):
        user = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE, related_name='test_attempts')
        test = models.ForeignKey(Test, on_delete=models.CASCADE, related_name='attempts')
        start_time = models.DateTimeField(auto_now_add=True)
        end_time = models.DateTimeField(null=True, blank=True)
        score = models.DecimalField(max_digits=5, decimal_places=2, null=True, blank=True)
        passed = models.BooleanField(null=True, blank=True)

        def save(self, *args, **kwargs):
            super().save(*args, **kwargs)

        def __str__(self):
            return f"{ self.user.username} - {self.test.title} ({self.start_time.strftime('%Y-%m-%d %H:%M')})"

        def calculate_score(self):
            total_points = sum(q.points for q in self.test.questions.all())
            if total_points == 0:
                return 0

            earned_points = 0
            processed_questions = set()

            for answer in self.answers.all():
                if answer.question.id in processed_questions:
                    continue

                processed_questions.add(answer.question.id)
```

```
                earned_points += answer.points_earned

            percentage = (earned_points / total_points) * 100
            self.score = round(percentage, 2)
            self.passed = self.score >= self.test.passing_score
            return self.score

        class Meta:
            ordering = ['-start_time']

    class TestAnswer(models.Model):
        attempt = models.ForeignKey(TestAttempt, on_delete=models.CASCADE, related_name='answers')
        question = models.ForeignKey(Question, on_delete=models.CASCADE)
        selected_choice = models.ForeignKey(Choice, on_delete=models.CASCADE, null=True, blank=True)
        text_answer = models.TextField(null=True, blank=True)
        points_earned = models.DecimalField(max_digits=5, decimal_places=2, default=0)

        def __str__(self):
            return f"Answer for {self.question} by {self.attempt.user.username}"

        def evaluate(self):
            if self.question.question_type == 'multiple_choice':

                user_answers = TestAnswer.objects.filter(attempt=self.attempt, question=self.question)

                correct_choices = self.question.choices.filter(is_correct=True)
                correct_choice_ids = set(choice.id for choice in correct_choices)

                selected_choice_ids = set(answer.selected_choice.id for answer in user_answers if answer.selected_choice)

                all_selected_are_correct = selected_choice_ids.issubset(correct_choice_ids)
```

```
    all_correct_are_selected = correct_choice_ids.issubset(selected_choice_ids)

    has_selected_choices = len(selected_choice_ids) > 0

    if has_selected_choices and all_selected_are_correct and all_correct_are_selected:
        self.points_earned = self.question.points
    else:
        self.points_earned = 0

    if user_answers.count() > 1:
        points = self.points_earned
        for answer in user_answers:
            if answer.id != self.id:
                answer.points_earned = points
                answer.save()
elif self.question.question_type in ['true_false', 'radio_button']:
    if self.selected_choice and self.selected_choice.is_correct:
        self.points_earned = self.question.points
    else:
        self.points_earned = 0
elif self.question.question_type == 'open_ended':
    self.points_earned = self.question.points

self.save()
return self.points_earned
```

Моделі Django ORM Навчального модуля

```
from django.db import models
from django.conf import settings
from django.utils.text import slugify
from django.urls import reverse
from tinymce.models import HTMLField

class Category(models.Model):
    name = models.CharField(max_length=100)
    slug = models.SlugField(unique=True)
    description = models.TextField(blank=True)

    class Meta:
        verbose_name_plural = "Categories"

    def __str__(self):
        return self.name

    def save(self, *args, **kwargs):
        if not self.slug:
            self.slug = slugify(self.name)
        super().save(*args, **kwargs)
class CEFRLevel(models.Model):
    LEVELS = [
        ('A1', 'A1 - Beginner'),
        ('A2', 'A2 - Elementary'),
        ('B1', 'B1 - Intermediate'),
        ('B2', 'B2 - Upper Intermediate'),
        ('C1', 'C1 - Advanced'),
        ('C2', 'C2 - Proficiency'),
    ]
    name = models.CharField(max_length=2, choices=LEVELS, unique=True)
    description = models.TextField(blank=True)
    def __str__(self):
        return dict(self.LEVELS)[self.name]
class LearningMaterial(models.Model):
    title = models.CharField(max_length=200)
    slug = models.SlugField(max_length=200, unique=True)
    content = HTMLField()  # Змініть це поле
    category = models.ForeignKey(Category, on_delete=models.CASCADE)
    cefr_level = models.ForeignKey(CEFRLevel, on_delete=models.SET_NULL, null=True)
    author = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
    is_published = models.BooleanField(default=False)

    def __str__(self):
        return self.title

    def save(self, *args, **kwargs):
        if not self.slug:
```

```
            self.slug = slugify(self.title)

            original_slug = self.slug
            count = 1
            while LearningMaterial.objects.filter(slug=self.slug).exists():
                self.slug = f"{original_slug}-{count}"
                count += 1

        super().save(*args, **kwargs)

    def get_absolute_url(self):
        return reverse('material_detail', kwargs={'slug': self.slug})

class MaterialAttachment(models.Model):
    material = models.ForeignKey(LearningMaterial, on_delete=models.CASCADE,
related_name='attachments')
    file = models.FileField(upload_to='materials/', null=True, blank=True)
    file_name = models.CharField(max_length=255)
    file_type = models.CharField(max_length=100)
    uploaded_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return self.file_name
```

Моделі Django ORM «Ядра» системи

```
from django.contrib.auth.models import AbstractUser
from django.db import models
from django.utils import timezone

class User(AbstractUser):
    VISITOR = 'visitor'
    CONTENT_EDITOR = 'content_editor'

    ROLE_CHOICES = [
        (VISITOR, 'Visitor'),
        (CONTENT_EDITOR, 'Content Editor'),
    ]

    THEME_LIGHT = 'light'
    THEME_DARK = 'dark'

    THEME_CHOICES = [
        (THEME_LIGHT, 'Light'),
        (THEME_DARK, 'Dark'),
    ]

    email = models.EmailField(unique=True)
    role = models.CharField(max_length=20, choices=ROLE_CHOICES, default=VISITOR)
    theme_preference = models.CharField(max_length=10, choices=THEME_CHOICES,
default=THEME_LIGHT)
    is_email_verified = models.BooleanField(default=False)
    email_verification_token = models.CharField(max_length=100, blank=True, null=True)
    email_verification_token_created = models.DateTimeField(null=True, blank=True)

    USERNAME_FIELD = 'email'
    REQUIRED_FIELDS = ['username']

    def save(self, *args, **kwargs):
        if not self.username:
            self.username = self.email

        if self.role == self.CONTENT_EDITOR:
            self.is_staff = True

        super().save(*args, **kwargs)

    def is_content_editor(self):
        return self.role == self.CONTENT_EDITOR
```

ДОДАТОК Б

JavaScript в файлі base.html

```
document.addEventListener('DOMContentLoaded', function() {
    tinymce.init({
        selector: 'textarea.rich-editor',
        height: 400,
        plugins: [
            'advlist', 'autolink', 'lists', 'link', 'image', 'charmap', 'preview',
            'anchor', 'searchreplace', 'visualblocks', 'code', 'fullscreen',
            'insertdatetime', 'media', 'table', 'help', 'wordcount'
        ],
        toolbar: 'undo redo | blocks | ' +
            'bold italic forecolor backcolor | alignleft aligncenter ' +
            'alignright alignjustify | bullist numlist outdent indent | ' +
            'image | removeformat | help',
        content_style: 'body { font-family:Helvetica,Arial,sans-serif; font-size:16px }',
        image_advtab: true,
        image_dimensions: true,
        automatic_uploads: false,
        convert_urls: false,
        relative_urls: false,
        remove_script_host: false
    });

    const themeToggle = document.getElementById('themeToggle');
    if (themeToggle) {
        themeToggle.addEventListener('click', function(e) {
            e.preventDefault();

            fetch('/toggle-theme/', {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/json',
                 'X-   CSRFToken': document.querySelector('[name=csrfmiddlewaretoken]').value
                },
                body: JSON.stringify({})
            })
            .then(response => response.json())
            .then(data => {
                if (data.success) {
                    document.body.classList.toggle('dark-theme');
                    document.body.classList.toggle('bg-light');

                    const themeIcon = themeToggle.querySelector('i');
                    if (themeIcon.classList.contains('fa-moon')) {
                        themeIcon.classList.remove('fa-moon');
                        themeIcon.classList.add('fa-sun');
                        themeToggle.querySelector('i + span').textContent = ' Light Theme';
                    } else {
```

```
                themeIcon.classList.remove('fa-sun');
                themeIcon.classList.add('fa-moon');
                themeToggle.querySelector('i + span').textContent = ' Dark Theme';
            }
        }
    })
    .catch(error => console.error('Error:', error));
});

if (document.body.classList.contains('dark-theme')) {
    const themeIcon = themeToggle.querySelector('i');
    themeIcon.classList.remove('fa-moon');
    themeIcon.classList.add('fa-sun');
    if (!themeToggle.querySelector('i + span')) {
        const textSpan = document.createElement('span');
        textSpan.textContent = ' Light Theme';
        themeToggle.appendChild(textSpan);
    } else {
        themeToggle.querySelector('i + span').textContent = ' Light Theme';
    }
} else {
    if (!themeToggle.querySelector('i + span')) {
        const textSpan = document.createElement('span');
        textSpan.textContent = ' Dark Theme';
        themeToggle.appendChild(textSpan);
    } else {
        themeToggle.querySelector('i + span').textContent = ' Dark Theme';
    }
}
    }
});
```

JavaScript в test_form.html:

```
<script>
        document.addEventListener('DOMContentLoaded', function() {
            const addRadioOptionBtn = document.querySelector('#addRadioButtonModal button[data-action="add-radio-option"]');
            const radioOptionsContainer = document.querySelector('#addRadioButtonModal .options-container');

            if (addRadioOptionBtn && radioOptionsContainer) {
                let radioOptionCounter = 1;

                const firstRadioOptionRemoveBtn = document.querySelector('#addRadioButtonModal .option-item:first-child .remove-option');
                if (firstRadioOptionRemoveBtn) {
                    firstRadioOptionRemoveBtn.addEventListener('click', function() {
                        const optionItems = radioOptionsContainer.querySelectorAll('.option-item');
                        if (optionItems.length > 1) {
                            this.closest('.option-item').remove();
```

```
                    } else {
                        alert('Не можна видалити останній варіант відповіді');
                    }
                });
            }

            addRadioOptionBtn.addEventListener('click', function(e) {
                e.preventDefault();

                radioOptionCounter++;

                const newOption = document.createElement('div');
                newOption.className = 'mb-3 option-item';
                newOption.innerHTML = `
                    <div class="input-group">
                        <span  class="input-group-text">Option
${radioOptionCounter}</span>
                        <inputtype="text"name="      choice_text_${radioOptionCounter}"
class="form-control" required>
                        <div class="input-group-text">
                            <input  type="radio"  name="                              is_correct"
value="${radioOptionCounter}" class="form-check-input mt-0">
                            <span class="ms-2">Correct</span>
                        </div>
                        <button type="button" class="    btn btn-outline-danger remove-option">
                            <i class="fas fa-times"></i>
                        </button>
                    </div>
                `;

                radioOptionsContainer.appendChild(newOption);

                const removeBtn = newOption.querySelector('.remove-option');
                removeBtn.addEventListener('click', function() {
                    newOption.remove();
                });
            });
        }
    });
</script>

    document.addEventListener('DOMContentLoaded', function() {
        const addOptionBtn = document.querySelector('#addMultipleChoiceModal button[data-
action="add-option"]');
        const optionsContainer = document.querySelector('#addMultipleChoiceModal .options-
container');

        if (addOptionBtn && optionsContainer) {
            let optionCounter = 1;

            addOptionBtn.addEventListener('click', function(e) {
                e.preventDefault();
```

```
                optionCounter++;

                const newOption = document.createElement('div');
                newOption.className = 'mb-3 option-item';
                newOption.innerHTML = `
                    <div class="input-group">
                        <span class="input-group-text">Option ${optionCounter}</span>
                        <input type="text" name="choice_text_${optionCounter}" class="form-control" required>
                        <div class="input-group-text">
                            <input type="checkbox" name="is_correct_${optionCounter}" class="form-check-input mt-0">
                            <span class="ms-2">Correct</span>
                        </div>
                        <button type="button" class="btn btn-outline-danger remove-option">
                            <i class="fas fa-times"></i>
                        </button>
                    </div>
                `;

                optionsContainer.appendChild(newOption);

                const removeBtn = newOption.querySelector('.remove-option');
                removeBtn.addEventListener('click', function() {
                    newOption.remove();
                });
            });

                document.querySelectorAll('#addMultipleChoiceModal .remove-option').forEach(function(btn) {
                btn.addEventListener('click', function() {
                    this.closest('.option-item').remove();
                });
            });
        }
    });


    document.addEventListener('DOMContentLoaded', function() {
                const addRadioOptionBtn = document.querySelector('#addRadioButtonModal button[data-action="add-radio-option"]');
                const radioOptionsContainer = document.querySelector('#addRadioButtonModal .options-container');

                if (addRadioOptionBtn && radioOptionsContainer) {
                    let radioOptionCounter = 1;

                    const firstRadioOptionRemoveBtn = document.querySelector('#addRadioButtonModal .option-item:first-child .remove-option');
                    if (firstRadioOptionRemoveBtn) {
                        firstRadioOptionRemoveBtn.addEventListener('click', function() {
```

```
                const optionItems = radioOptionsContainer.querySelectorAll('.option-item');
                if (optionItems.length > 1) {
                    this.closest('.option-item').remove();
                } else {
                    alert('Не можна видалити останній варіант відповіді');
                }
            });
        }

        addRadioOptionBtn.addEventListener('click', function(e) {
            e.preventDefault();

            radioOptionCounter++;

            const newOption = document.createElement('div');
            newOption.className = 'mb-3 option-item';
            newOption.innerHTML = `
                <div class="input-group">
                    <span  class="input-group-text">Option ${radioOptionCounter}</span>
                    <input type="text" name="      choice_text_${radioOptionCounter}" class="form-control" required>
                    <div class="input-group-text">
                        <input  type="radio"  name="                    is_correct" value="${radioOptionCounter}" class="form-check-input mt-0">
                        <span class="ms-2">Correct</span>
                    </div>
                    <button type="button" class="    btn btn-outline-danger remove-option">
                        <i class="fas fa-times"></i>
                    </button>
                </div>
            `;

            radioOptionsContainer.appendChild(newOption);

            const removeBtn = newOption.querySelector('.remove-option');
            removeBtn.addEventListener('click', function() {
                newOption.remove();
            });
        });
    }
});
document.addEventListener('DOMContentLoaded', function() {
    const addOptionBtn = document.querySelector('#addMultipleChoiceModal button[data-action="add-option"]');
    const optionsContainer = document.querySelector('#addMultipleChoiceModal .options-container');

    if (addOptionBtn && optionsContainer) {
        let optionCounter = 1;
```

```
                addOptionBtn.addEventListener('click', function(e) {
                    e.preventDefault();

                    optionCounter++;

                    const newOption = document.createElement('div');
                    newOption.className = 'mb-3 option-item';
                    newOption.innerHTML = `
                        <div class="input-group">
                            <span class="input-group-text">Option ${optionCounter}</span>
                            <input type="text" name="        choice_text_${optionCounter}"  class="form-
control" required>
                            <div class="input-group-text">
                                <input type="checkbox" name="                is_correct_${optionCounter}"
class="form-check-input mt-0">
                                <span class="ms-2">Correct</span>
                            </div>
                            <button type="button" class="btn btn-outline-danger remove-option">
                                <i class="fas fa-times"></i>
                            </button>
                        </div>
                    `;
                    optionsContainer.appendChild(newOption);
                    const removeBtn = newOption.querySelector('.remove-option');
                    removeBtn.addEventListener('click', function() {
                        newOption.remove();
                    });
                });

                                    document.querySelectorAll('#addMultipleChoiceModal     .remove-
option').forEach(function(btn) {
                    btn.addEventListener('click', function() {
                        this.closest('.option-item').remove();
                    });}); }});
```

ДОДАТОК В

Python представлення Навчального модуля:

```
from django.shortcuts import render, redirect, get_object_or_404
from django.contrib.auth.decorators import login_required, user_passes_test
from django.contrib import messages
from django.core.paginator import Paginator
from .models import Category, CEFRLevel, LearningMaterial, MaterialAttachment
from .forms import MaterialForm, MaterialAttachmentForm, CategoryForm
from django.utils.text import slugify
from django.utils.crypto import get_random_string
import datetime

def is_content_editor(user):
    return user.is_authenticated and user.is_content_editor()

# Visitor views
def material_list(request):
    # Get all published materials
    materials = LearningMaterial.objects.filter(is_published=True).order_by('-created_at')

    # Ensure all materials have valid slugs
    for material in materials:
        if not material.slug:
            material.save()  # This will trigger the save method to generate a slug

    categories = Category.objects.all()
    levels = CEFRLevel.objects.all()

    category_filter = request.GET.get('category')
    level_filter = request.GET.get('level')
    search_query = request.GET.get('search')

    if category_filter:
        materials = materials.filter(category__slug=category_filter)

    if level_filter:
        materials = materials.filter(cefr_level__name=level_filter)

    if search_query:
        materials = materials.filter(title__icontains=search_query) |
materials.filter(content__icontains=search_query)

    paginator = Paginator(materials, 10)
    page_number = request.GET.get('page')
    page_obj = paginator.get_page(page_number)

    return render(request, 'learning_materials/material_list.html', {
        'page_obj': page_obj,
        'categories': categories,
```

```
        'levels': levels,
        'category_filter': category_filter,
        'level_filter': level_filter,
        'search_query': search_query,
    })

def material_detail(request, slug):
    material = get_object_or_404(LearningMaterial, slug=slug)

    # If not published, only allow content editors to view
    if not material.is_published and not is_content_editor(request.user):
        messages.error(request, "This material is not available.")
        return redirect('material_list')

    return render(request, 'learning_materials/material_detail.html', {
        'material': material,
    })

# Content Editor views
@login_required
@user_passes_test(is_content_editor)
def editor_dashboard(request):
    materials = LearningMaterial.objects.filter(author=request.user).order_by('-updated_at')

    for material in materials:
        if not material.slug:
            material.save()

    return render(request, 'learning_materials/editor/dashboard.html', {
        'materials': materials,
    })

@login_required
@user_passes_test(is_content_editor)
def material_create(request):
    if request.method == 'POST':
        if 'save_material' in request.POST:
            form = MaterialForm(request.POST)
            if form.is_valid():
                material = form.save(commit=False)

                slug = slugify(material.title)

                if not slug:
                    now = datetime.datetime.now()
                    slug = f"material-{now.strftime('%Y%m%d')}-{get_random_string(5)}"

                original_slug = slug
                count = 1
                while LearningMaterial.objects.filter(slug=slug).exists():
                    slug = f"{original_slug}-{count}"
                    count += 1
```

```
            material.slug = slug
            material.author = request.user
            material.save()
            messages.success(request, "Material created successfully!")
            return redirect('material_detail', slug=material.slug)
    else:
        form = MaterialForm()

    return render(request, 'learning_materials/editor/material_form.html', {
        'form': form,
        'title': 'Create New Learning Material',
    })

@login_required
@user_passes_test(is_content_editor)
def material_edit(request, slug):
    material = get_object_or_404(LearningMaterial, slug=slug)

    if material.author != request.user and not request.user.is_content_editor():
        messages.error(request, "You don't have permission to edit this material.")
        return redirect('material_detail', slug=slug)

    if request.method == 'POST':
        if 'save_material' in request.POST:
            form = MaterialForm(request.POST, instance=material)
            if form.is_valid():
                material = form.save(commit=False)
                material.save()
                messages.success(request, "Material updated successfully!")
                return redirect('material_detail', slug=material.slug)
        elif 'add_attachment' in request.POST:
            attachment_form = MaterialAttachmentForm(request.POST, request.FILES)
            if attachment_form.is_valid():
                if 'file' in request.FILES and attachment_form.cleaned_data.get('file_name'):
                    attachment = attachment_form.save(commit=False)
                    attachment.material = material
                    attachment.save()
                    messages.success(request, "Attachment added successfully!")
                return redirect('material_edit', slug=material.slug)
    else:
        form = MaterialForm(instance=material)
        attachment_form = MaterialAttachmentForm()

    attachments = material.attachments.all()

    context = {
        'title': 'Edit Learning Material',
        'form': form,
        'attachment_form': attachment_form,
        'material': material,
        'attachments': attachments,
```

```
    }

    return render(request, 'learning_materials/editor/material_form.html', context)

@login_required
@user_passes_test(is_content_editor)
def material_delete(request, slug):
    material = get_object_or_404(LearningMaterial, slug=slug)

    if material.author != request.user and not request.user.is_staff:
        messages.error(request, "You don't have permission to delete this material.")
        return redirect('material_detail', slug=slug)

    if request.method == 'POST':
        title = material.title

        try:
            material.delete()
            messages.success(request, f'"{title}" has been deleted successfully.')
            return redirect('editor_dashboard')
        except Exception as e:
            messages.error(request, f'Error deleting material: {str(e)}')
            return redirect('material_detail', slug=slug)

    return render(request, 'learning_materials/editor/material_confirm_delete.html', {'material': material})

@login_required
@user_passes_test(is_content_editor)
def material_toggle_publish(request, slug):
    material = get_object_or_404(LearningMaterial, slug=slug)

    if material.author != request.user and not request.user.is_superuser:
        messages.error(request, "You don't have permission to modify this material.")
        return redirect('editor_dashboard')

    material.is_published = not material.is_published
    material.save()

    status = "published" if material.is_published else "unpublished"
    messages.success(request, f"Material '{material.title}' {status} successfully.")

    return redirect('editor_dashboard')
```