НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

«На правах рукопису»
УДК _____

До захисту допущено:
Завідувач кафедри
Сергій Стіренко
«___»_____2021 р.

Магістерська дисертація

на здобуття ступеня магістра
за освітньо-науковою програмою «Інженерія програмного забезпечення комп'ютерних систем»
зі спеціальності 121 «Інженерія програмного забезпечення»
на тему: «Система для дистанційного навчання англійської мови»

Виконав:
студент VI курсу, групи ІП-93мн
Литвинюк Дмитро Романович _____

Керівник:
Доц., к. т. н., доц.
Долголенко Олександр Миколайович _____

Консультант з нормоконтролю:
Проф., д.т.н., проф.
Кулаков Юрій Олексійович _____

Рецензент:
Доц., к. т. н., доц.
Писаренко Андрій Володимирович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2021 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет (інститут) *Інформатики та обчислювальної техніки*
(повна назва)

Кафедра *Обчислювальної техніки*
(повна назва)

Освітньо-кваліфікаційний ступінь *магістр*
(назва ОКР)

Спеціальність *121. Інженерія програмного забезпечення*
(код і назва)

Спеціалізація *121. Інженерія програмного забезпечення комп'ютерних систем*

ЗАТВЕРДЖУЮ
Завідувач кафедри
Стіренко С.Г.
(підпис) (ініціали, прізвище)
« » 2021 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Литвинюку Дмитру Романовичу
(прізвище, ім'я, по батькові)

1. Тема дисертації Система для дистанційного навчання англійської мови

Науковий керівник дисертації Долголенко О.М., доц., к. т. н.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від ***« 12» 03 2021 р. № 809-с***

2. Строк подання студентом дисертації:

3. Об'єкт дослідження: Система для дистанційного навчання англійської мови.

4. Предмет дослідження: Використання сучасних технологій для навчання.

5. Перелік завдань, які потрібно виконати: аналіз існуючих систем для дистанційного навчання англійської мови, порівняння сучасних підходів та інструментів, проектування системи, розробка системи, тестування розробленої системи, аналіз отриманих результатів.

6. Консультанти розділів дисертації:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
нормконтроль	Кулаков Ю.О.		

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів дисертації	Примітка
1	Затвердження теми роботи	05.03.2021	виконано
2	Дослідження предметної області, огляд літератури	11.03.2021	виконано
3	Аналіз існуючих рішень	17.03.2021	виконано
4	Проектування програмного забезпечення	29.03.2021	виконано
5	Розробка програмного забезпечення	13.04.2021	виконано
6	Тестування програмного забезпечення	15.04.2021	виконано
7	Оформлення пояснювальної записки	23.04.2021	виконано
8	Подання МД на попередній захист	26.04.2021	виконано
9	Основний захист	17.05.2021	

Студент _____ Д. Р. Литвинюк
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ О. М. Долголенко
(підпис) (ініціали, прізвище)

РЕФЕРАТ

на магістерську дисертацію

виконану на тему: Система для дистанційного навчання англійської мови

студентом: Литвинюком Дмитром Романовичем

Робота складається із вступу та чотирьох розділів. Загальний обсяг роботи: 80 аркуші основного тексту, 33 ілюстрації, 22 таблиць. При підготовці використовувалася література з 26 різних джерел.

Актуальність теми. З кожним роком дистанційне навчання набуває все більшої популярності у всьому світі. Дистанційне навчання надає велике різноманіття навчальних матеріалів для учнів, що дозволяє кожному бажаючому знайти, той навчальний матеріал, який він потребує. Усі в кого є бажання, мають можливість отримати доступ до багатьох курсів з різних дисциплін найкращих університетів світу, з будь-якого місця на Землі. Також дистанційне навчання надає можливість учням не бути прив’язаним до розкладу занять, а навчатись тоді, коли їм зручно та у темпі, який їм підходить найбільше. Дистанційне навчання англійської мови є одним з найрозповсюдженіших способів її вивчення. Використання сучасних технологій для персоналізації навчання та адаптації навчальних матеріалів для кожного окремого студента, дозволить покращити швидкість засвоювання матеріалів учнями та отримувати кращу якість навчання англійської мови.

Мета і завдання дослідження. Метою магістерської дисертації є збільшення ефективності вивчення англійської мови з використанням сучасних технологій.

Завдання дослідження:

1. Аналіз існуючих систем для дистанційного навчання англійської мови.
2. Порівняння сучасних підходів та інструментів для розробки програмного забезпечення.

3. Проектування системи для дистанційного навчання англійської мови.
4. Розробка системи для дистанційного навчання англійської мови.
5. Тестування системи для дистанційного навчання англійської мови.
6. Аналіз отриманих результатів.

Об’єкт дослідження. Система для дистанційного навчання англійської мови

Предмет дослідження. Використання сучасних технологій для навчання

Методи досліджень. Для досягнення поставлених в магістерській роботі задач, використано проектування програмного забезпечення, дослідження, обчислення, обробка природної мови.

Особистий внесок здобувача. Магістерське дослідження є самостійно виконаною роботою, в якій відображено особистий авторський підхід та особисто отримані теоретичні та прикладні результати, що відносяться до проектування та розробки системи для дистанційного навчання англійської мови. Формулювання мети та завдань дослідження проводилось спільно з науковим керівником.

Практичне значення одержаних результатів. Розроблено метод для автоматичного створення завдань для вивчення англійської мови. Розроблено спосіб для персоналізації навчання англійської мови.

Апробація роботи. Основні результати роботи за темою дисертації опубліковані та були обговорені: на Міжнародній науково-технічній конференції “The International Conference on Security, Fault Tolerance, Intelligence” (ICSFTI2021), 12-13 травня 2021, м. Київ;

Ключові слова.

ВЕБ-ДОДАТОК, ОБРОБКА ПРИРОДНОЇ МОВИ, АВТОМАТИЧНЕ СТВОРЕННЯ ЗАВДАНЬ, ПЕРСОНАЛІЗАЦІЯ, ДИСТНАЦІЙНЕ НАВЧАННЯ

ABSTRACT

Structure and scope of master's thesis: master's thesis consists of an introduction, four chapters and conclusions. Total volume of work: 80 pages of the main text, 33 illustrations, 22 tables. Literature from 26 different sources was used.

Subject relevance. Every year, the distance learning is growing more and more popular among the whole world. Distance learning provides a wide variety of learning materials for students, allowing anyone to find the learning material they need. Everyone who has the desire can access many courses from various disciplines of the best universities in the world, from different places. Distance learning also allows students not to be tied to a class schedule, but to learn when it is convenient for them and at a pace that suits them best. Distance learning English is one of the most common ways to learn it. The use of modern technologies to personalize learning and adapt learning materials for each individual student, will improve the speed of learning by students and get the best quality of English language learning.

Purpose and research objectives. The purpose of the master's thesis is to increase the efficiency of learning English with usage of modern technologies.

Research objectives:

1. Review existing distance learning systems for English language leaning.
2. Compare modern approaches and tools for software development.
3. Design of distance learning system for English language learning.
4. Development of distance learning system for English language learning.
5. Testing of distance leaning system for English language learning.
6. Analysis of the received results.

Object of research. Distance learning system for English language learning.

Subject of research. Usage of modern technologies for education.

Research methods. To achieve the objectives of the master's thesis, used software design, research, calculation, natural language processing.

Personal contribution of the applicant. Master's thesis is a self-performed work, which reflects the personal author's approach and personally obtained theoretical and applied final results related to designing and developing of distance learning system for English language learning.

The practical significance. Developed the method for automated task creation. Developed approach for personalization of English language learning.

Approbation of work. The main work results of the dissertation topic were reported and discussed at:

- International Scientific and Technical Conference "The International Conference on Security, Fault Tolerance, Intelligence" (ICSFTI2021), May 12-13, 2021, Kyiv;

Keywords.

WEB APPLICATION, NATURAL LANGUAGE PROCESSING, AUTOMATED TASK CREATION, PERSONALIZATION, DISTANCE LEARNING

ЗМІСТ

ВСТУП

З кожним роком дистанційне навчання набуває все більшої популярності у всьому світі. Дистанційне навчання надає велике різноманіття навчальних матеріалів для учнів, що дозволяє кожному бажаючому знайти, той навчальний матеріал, який він потребує. Усі в кого є бажання, мають можливість отримати доступ до багатьох курсів з різних дисциплін найкращих університетів світу, з будь-якого місця на Землі. Також дистанційне навчання надає можливість учням не бути прив’язаним до розкладу занять, а навчатись тоді, коли їм зручно та у темпі, який їм підходить найбільше.

Все частіше для навчання використовуються сучасні технології, які допомагають людині навчатися краще та досягати своїх цілей навчання. Однією з найбільших проблем в освіті є те, що люди навчаються по-різному і з різними темпами. Студенти проходять через систему освіти з різним рівнем здатності до навчання. Одні більш спритні в «лівому мозку» мислення з навичками аналітичної думки, а інші – більш кваліфіковані у «правому мозку» мислення з творчими, літературними та комунікативними здібностями.

Сучасні технології використовуються для адаптації та персоналізації навчання для кожного окремого учня. Завдяки гіперперсоналізації, створюється спеціальний профіль навчання для кожного учня та адаптації навчальних матеріалів для кожного учня, виходячи з його здібностей, бажаного способу навчання та досвіду. Використовуючи сучасні технології для навчання англійської мови, можна значно підвищити якість та продуктивність навчання учнів.

Англійська мова є однією з найбільших поширених мов у світі. Англійська мова офіційна у 53 країнах світу та найпоширеніша друга мова у світі, тому уміння спілкуватись англійською мовою, дозволяє спілкуватись з людьми з різних куточків планети. Знання англійської мови надає можливість отримати доступ до найактуальніших знань. Це найбільш актуально для галузей, де зміни дуже швидкі, наприклад у інформаційних технологіях.

Також знання англійської мови надає доступ до багатьох видів розваг, які створенні на англійській мові – книжок, фільмів, серіалів, телевізійних шоу та інше.

Дистанційне навчання англійської мови є одним з найрозповсюдженіших способів її вивчення. Використання сучасних технологій для персоналізації навчання та адаптації навчальних матеріалів для кожного окремого студента, дозволить покращити швидкість засвоювання матеріалів учнями та отримувати кращу якість навчання англійської мови.

РОЗДІЛ 1
ДОСЛІДЖЕННЯ СИСТЕМ ДЛЯ ДИСТАНЦІЙНОГО НАВЧАННЯ АНГЛІЙСЬКОЇ МОВИ

1.1 Характеристика систем для дистанційного навчання англійської мови

Дистанційне навчання описує будь-яке навчання, яке відбувається без фізичної присутності учнів на уроку [1]. Історично це описувало заочні курси, на яких студенти спілкувалися зі своїми вчителями поштою. З розвитком інформаційних технологій дистанційна освіта перейшла в інтернет, зробивши доступним величезний спектр систем та методів навчання практично на будь-якому пристрої підключеному до мережі.

Синхронне дистанційне навчання відноситься до типу навчання, коли навчальна група, до складу якої входять як викладач, так і студенти, взаємодіє одночасно, хоча і з різних географічних локацій. За допомогою цієї моделі студенти, як правило, повинні брати участь у навчальних заходах у визначений час. Залучення студентів одночасно, може допомогти забезпечити взаємодію всередині групи. Приклади синхронної навчальної діяльності включають групові чати, вебінари та інші форми відеоконференцій.

Асинхронне дистанційне навчання – це вид дистанційного навчання, де навчання відбувається на більш індивідуальній основі, а не через спільні сеанси в групі. Як правило, структура все ще надається у формі чітких термінів виконання завдань, але студенти виконують їх у зручний для них час. Асинхронне навчання, як правило, забезпечує більше гнучкості, оскільки учні не зобов’язані брати участь у заняттях одночасно, однак це може зменшити взаємодію в навчальній групі. Навчальні матеріали можуть мати форму письмового змісту, попередньо записаних відеоуроків, аудіозаписів та листування електронною поштою.

Самостійне дистанційне навчання відноситься до моделей дистанційного навчання, де студентам надається свобода йти у своєму темпі. Це означає, що загальний час, витрачений на курс, може суттєво відрізнятися

від одного студента до іншого, залежно від таких факторів, як здібність, кількість вільного часу та загальна зайнятість. Основними перевагами є свобода та гнучкість [2]. Самостійні курси також допомагають уникнути ситуацій, коли учні з високим рівнем зацікавленості та здібностей стримуються. При самостійному навчанні акцент на процесі навчання, а не на результатах навчання. Також, немає тиску при вивчені необхідного матеріалу за певний час, кінцевою метою є отримання знань, а не високої оцінки будь-яким способом.

На сьогоднішній день вивчення англійської мови є дуже важливим. Англійська мова найпоширеніший засіб спілкування між людьми з різних країн світу. Англійська мова є мовою спілкування в освіті, науці та техніці, дипломатичних відносинах, мистецтві, бізнесу, музиці та спорту [3]. Мистецькі та музичні фестивалі, в яких беруть участь міжнародні митці, проводяться переважно англійською мовою, а глобальні спортивні змагання, такі як Олімпійські ігри, використовують англійську мову як основну мову спілкування, незалежно від того, де вони відбуваються.

Мова також відіграє важливу роль у цифрову епоху, в якій ми живемо. Зростання інтернету свідчить про важливість вивчення англійської мови. Понад 50% усіх веб-сторінок, доступних в інтернеті на англійській мові. Вивчення англійської мови важливе, оскільки воно дає доступ до величезного обсягу інформації в інтернеті. Вміння читати англійською мовою, дозволяє отримати доступ до мільярдів сторінок інформації.

Якщо людина зацікавлена у зростанні своєї кар’єри на міжнародній арені, існує необхідність вивчати англійську мову. Надзвичайно важко досягти успіху на світовій арені, навіть не маючи базового розуміння англійської мови. Оскільки більшість міжнародних компаній мають відділення в багатьох країнах, англійська стає мовою, яку майже завжди можна використовувати для ефективного спілкування з клієнтами та іншими зацікавленими сторонами. Кожен, хто хоче працювати на міжнародній арені, повинен мати певні знання

з англійської мови. Знання мови, як правило, є обов'язковою умовою для тих, хто намагається зайняти посаду, яка передбачає роботу з колегами з інших країн та з міжнародними клієнтами.

На сьогоднішній день вивчення англійської мови за допомогою веб-сайтів є одним з найпоширеніших способів вивчення англійської мови так, як самостійне дистанційне навчання набагато доступніше ніж традиційні заняття з фізичною присутністю учнів. Опанування мови за допомогою веб-сайтів надає більше вільного простору учню в навчанні, та коштує набагато дешевше. Системи дистанційного навчання надають можливість студенту вчитись у найбільш зручному для нього темпі й не залежати від інших учнів. Кожен студент засвоює новий матеріал у своєму темпі, якщо студент швидко опановує нову тему, він раніше може перейти до наступної теми, якщо ж студенту навпаки новий матеріал дається важко, він спокійно може витратити на його вивчення більше часу і оволодіти знаннями на тому ж рівні, що і студент, який набагато швидше засвоїв новий матеріал. При навчанні у звичайних навчальних класах усі учні навчаються за планом, тому самостійне дистанційне навчання дає можливість кожному учню досягати поставлених цілей у навчанні за час, який необхідний кожному учню особисто, не маючи залежності від інших. Також дистанційне навчання надає можливість навчатись у час, який найбільш зручний для студента та виділяти на навчання час, залежно від обставин. Сучасні системи для дистанційного навчання використовують персоналізацію при подачі навчального матеріалу, що дає можливість кожному студенту отримувати знання, які їм необхідні та виявляти і усувати прогалини у знаннях.

Веб-сайти для дистанційного навчання англійської мови використовують різні інструменти для навчання. Більшість сайтів не мають широкого вибору навчального матеріалу й містять лише тести для вивчення граматики. Інша частина сайтів має свої власні підходи для подачі навчального матеріалу, які роблять навчання більш ефективними. Наприклад це

використання відеоматеріалів для вивчення слів та аудіо завдання для вивчення граматики. Веб-сайти, які використовують нестандартні підходи для навчання, привабливіші та більш популярні серед користувачів ніж звичайні сайти без унікальних особливостей.

Система для дистанційного навчання повинна надавати можливість учню тренувати усі необхідні навички: читання, письмо, слухання, вимову. Завдання можуть бути у різних формах. Для читання, як правило, використовуються завдання, які надають деякий текст та тестові завдання до нього. Для покращення навиків письма використовуються тести у різних формах: тести з варіантам відповідей, ввести пропущене слово, поставити слова в правильному порядку і т.д. Слухання тренується за допомогою аудіо та відео, до яких можуть додаватись завдання за прослуханим та побаченим матеріалом. Завдання для покращення вимови можуть бути у формі надання тексту, який студенту потрібно буде сказати. Те, що скаже студент буде записано та може бути проаналізовано системою, а також студент може прослухати свій аудіозапис та проаналізувати свої помилки. Також можуть бути окремі завдання для вивчення та повторення нових слів, що дозволяє студенту розширювати словниковий запас.

1.2 Огляд існуючих систем

1.2.1 USA Learns

Сайт пропонує багато різних завдань з вивчення мови, які охоплюють основи: граматику, правопис, вимову, словниковий запас, аудіювання. Сайт пропонує три курси для тих, хто вивчає англійську мову на початковому та середньому рівнях: перший курс, який розрахований для початківців, другий курс англійської мови для студентів, які мають середній рівень знань та розширений курс з практики англійської та читання.

Перший курс англійської мови – це відео-курс, призначений для тих, хто вивчає англійську мову на початковому рівні. Курс містить 20 тематичних уроків. Один урок складається з кількох коротких трьох хвилинних відео.

Теми уроків охоплюють основні навички, наприклад, як представити себе, запитати адресу, повідомити про хворобу та повідомити про надзвичайну ситуацію. Другий курс англійської мови призначений для тих, хто вивчає англійську мову середнього рівня. Цей курс містить послідовність із 20 уроків з п’яти тем, які представлені в серії історій про різних персонажів. Теми уроків: робота, сім’я, податки, розваги, освіта. Курс з практичного використання англійської містить серію історій, які допомагають учням англійської мови середнього рівня практикувати аудіювання, читання, письмо та розмовні навички. Історії, більшість з яких взяті з новин, адаптовані з урахуванням потреб тих, хто вивчає англійську мову. Всього є 44 історії, які містять завдання й можуть бути виконані в будь-якому порядку. Історії доступні в текстовому та аудіо форматах (Рис. 1.1).

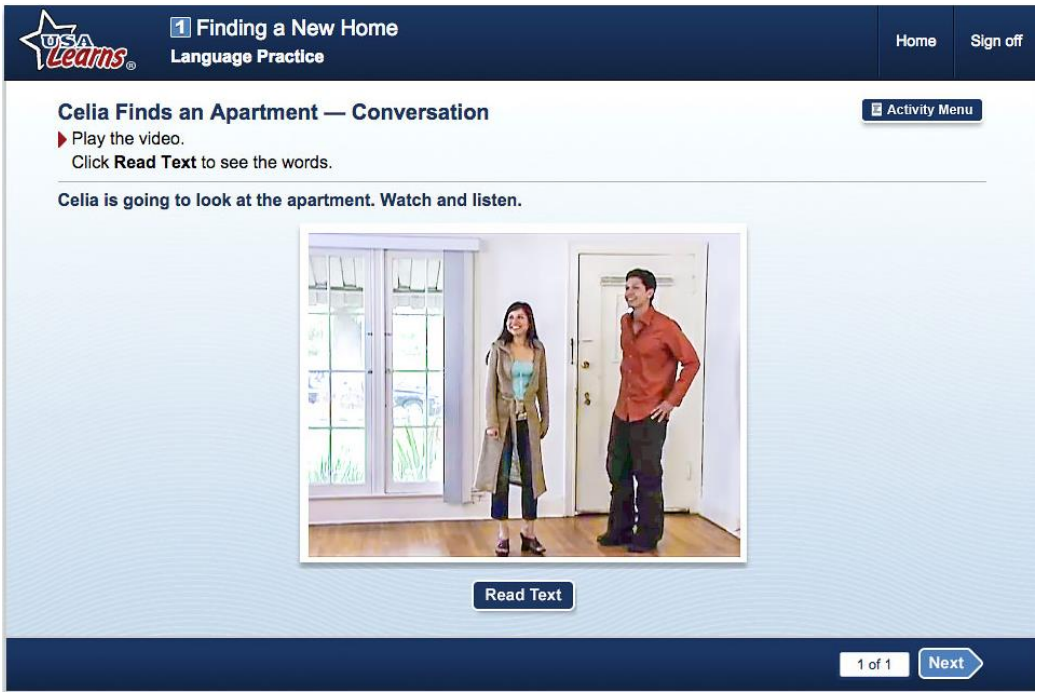


Рис. 1.1. Інтерфейс USA Learns [4]

Завдання для «USA Learns» розроблені Лос-Анджелеським об’єднаним шкільним округом. «USA Learns» є абсолютно безкоштовним веб-сайтом для вивчення англійської мови.

1.2.2 EnglishClass101

Веб-сайт «EnglishClass101» створений для студентів з різними рівнями знань. «EnglishClass101» має понад 1600 уроків для всіх рівнів навчання. «EnglishClass101» є однією з небагатьох систем для дистанційного навчання англійської мови, яка має окремі уроки для британської англійської та американської англійської. Заняття поєднують відео та аудіо для всебічного підходу, а також письмові нотатки для підведення підсумків кожного уроку. Усі уроки супроводжуються докладними шпаргалками, в яких викладається те, що було вивчено на уроку, і додається відповідна культурна чи суспільна інформація, яка може стосуватися уроку. Також є завдання для тренування словникового запасу. Для тренування вимови є завдання, які дозволяють записувати голос і програвати його, що дозволяє проаналізувати помилки та покращити вимову (Рис. 1.2).

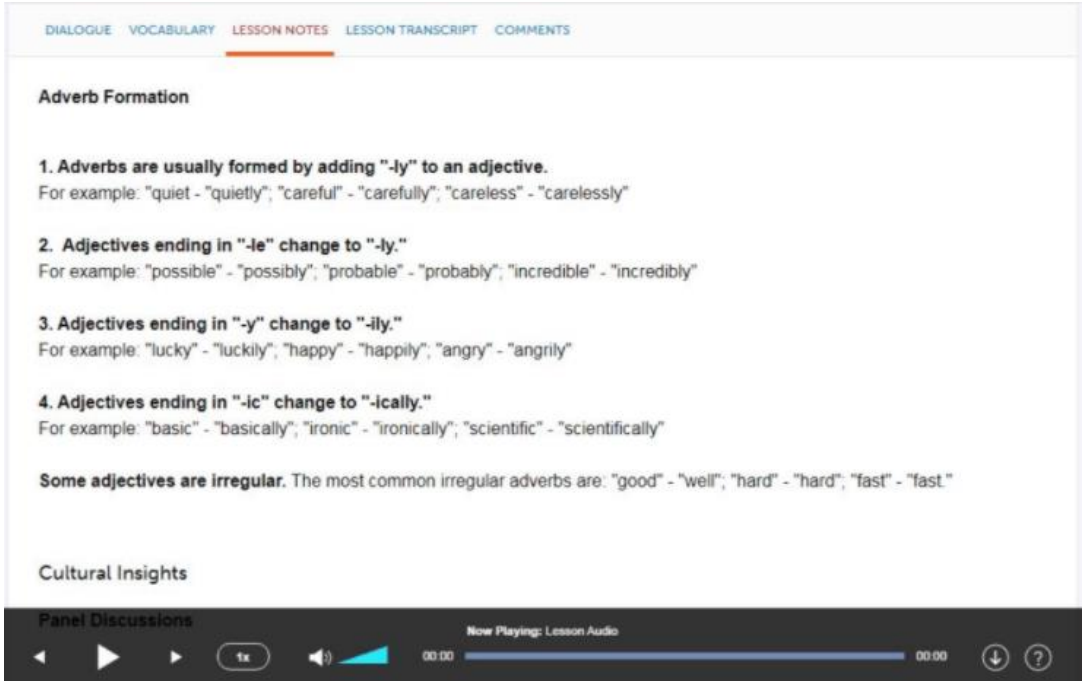


Рис. 1.2. Інтерфейс EnglishClass101 [5]

«EnglishClass101» має різні шляхи для навчання. Це дозволяє студенту точно налаштувати навчання, таким способом, яким він хоче вивчати англійську мову. Бібліотека уроків має можливість фільтрування за типом

навичок: розмова, читання, письмо, словниковий запас, граматика. Існує форум спільноти, де студенти можуть взаємодіяти з іншими учнями та допомагати одне одному в практиці. Ціна за «EnglishClass101» висока так, як має доволі велику команду викладачів.

1.2.3 British Council

«British Council» – це система для дистанційного навчання англійської мови, яка розрахована на різні вікові групи. «British Council» має три окремі розділи: для дітей, для підлітків та для дорослих. Кожен розділ містить велику кількість інтерактивних уроків, відео, ігор та подкастів, для вивчення англійської мови.

Для навчання дітей англійської мови використовується пісні, історії, відео, граматичні та словникові ігри. Завдання призначені для дітей, які вони можуть робити самостійно або з друзями, батьками та вчителями. Такий підхід дозволяє утримувати увагу дітей на навчанні. Для дорослих є багато аудіо та відео матеріалів для практики англійської мови. Матеріали включають подкасти, аудіокниги, серію телевізійних програм для вивчення англійської мови, створених разом з BBC, та відео про те, що треба говорити в різних ситуаціях. Також сайт містить багато граматичних тестів для різних рівнів знань (Рис. 1.3).

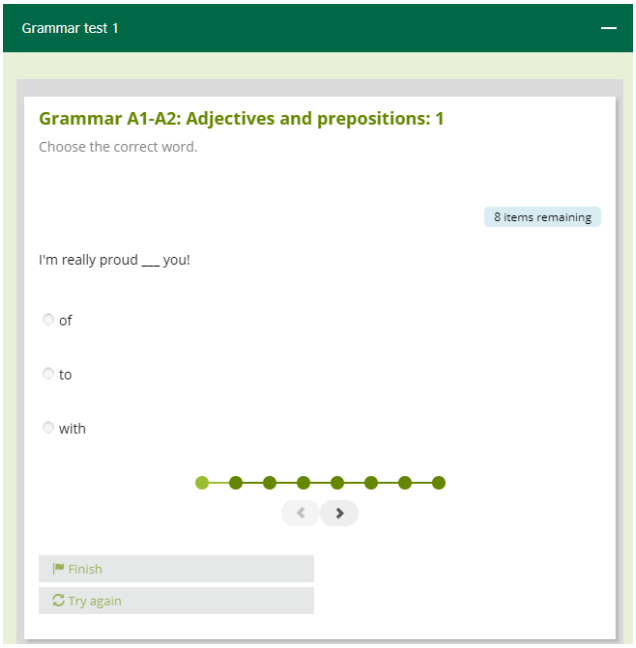


Рис. 1.3. Інтерфейс British Council [6]

«British Council», порівняно з іншими системами для дистанційного навчання англійської мови, є недорогими, також частина навчальних матеріалів доступна безкоштовно.

1.2.4 BBC Learning English

«BBC Learning English» розрахований для людей з певним досвідом володіння англійською мовою, а не для початківців. Сайт «BBC Learning English» розроблений для тих, хто має середній або просунутий рівень знань з англійської мови. Для навчання англійської мови «BBC Learning English» використовує відео та подкасти у стилі BBC. Вони зосереджуються на таких темах, як новини, англійські слова та фрази та серія відео, які детально зосереджуються на англійській вимові (Рис. 1.4).

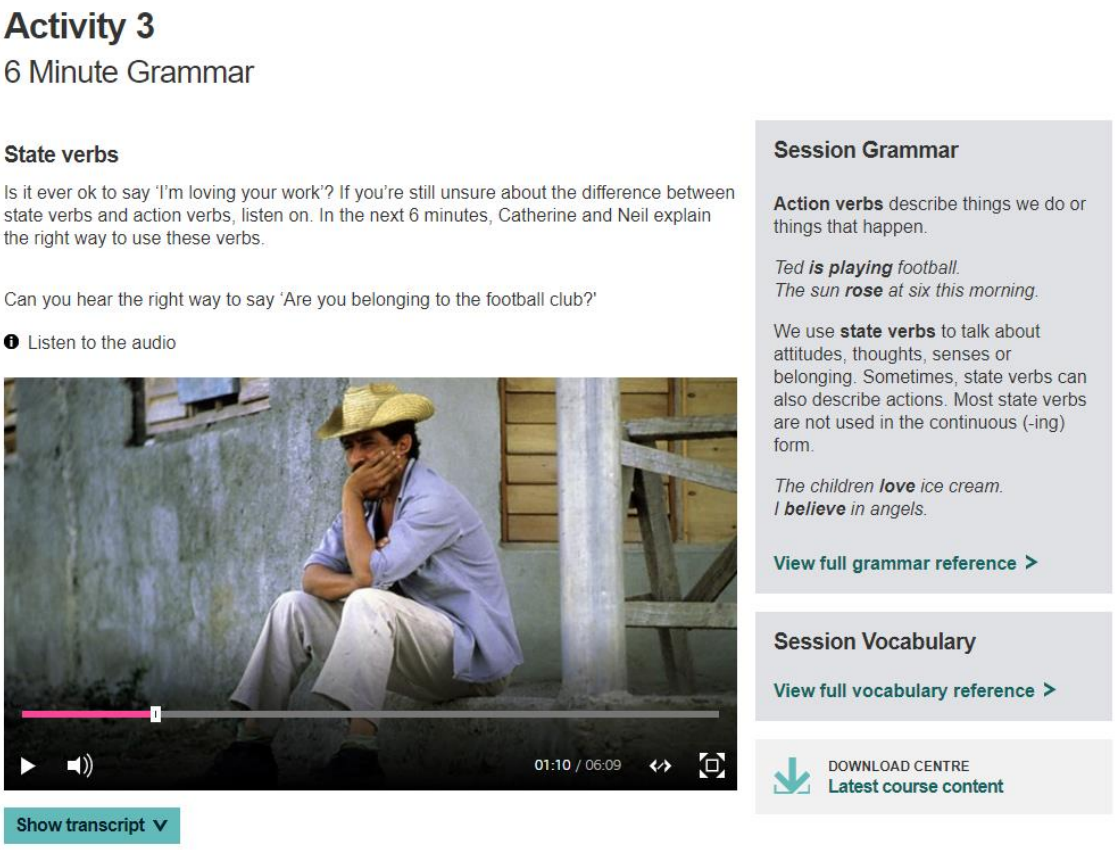


Рис. 1.4. Інтерфейс BBC Learning English [7]

Кожен їх подкаст та відео включає в себе завдання після прослуховування, щоб перевірити засвоювання нового матеріалу. Сервіс є абсолютно безкоштовним і підходить для відпрацювання навичок аудіювання.

1.2.5 Duolingo

«Duolingo» розрахований для вивчення лексики та граматики за допомогою інтерактивних тестів. Наголос робиться на вимові, з використанням великої кількості аудіо завдань, які дозволяють тренувати вимову слів. «Duolingo» використовує невеликі уроки, кожен з яких має свою специфічну тему, для подачі навчального матеріалу (Рис. 1.5). Курси розбиті на категорії, менші вправи розміщені всередині уроків. Виконання уроків розблоковує наступні, більш складні уроки. Така система розроблена для мотивування виконання уроків.

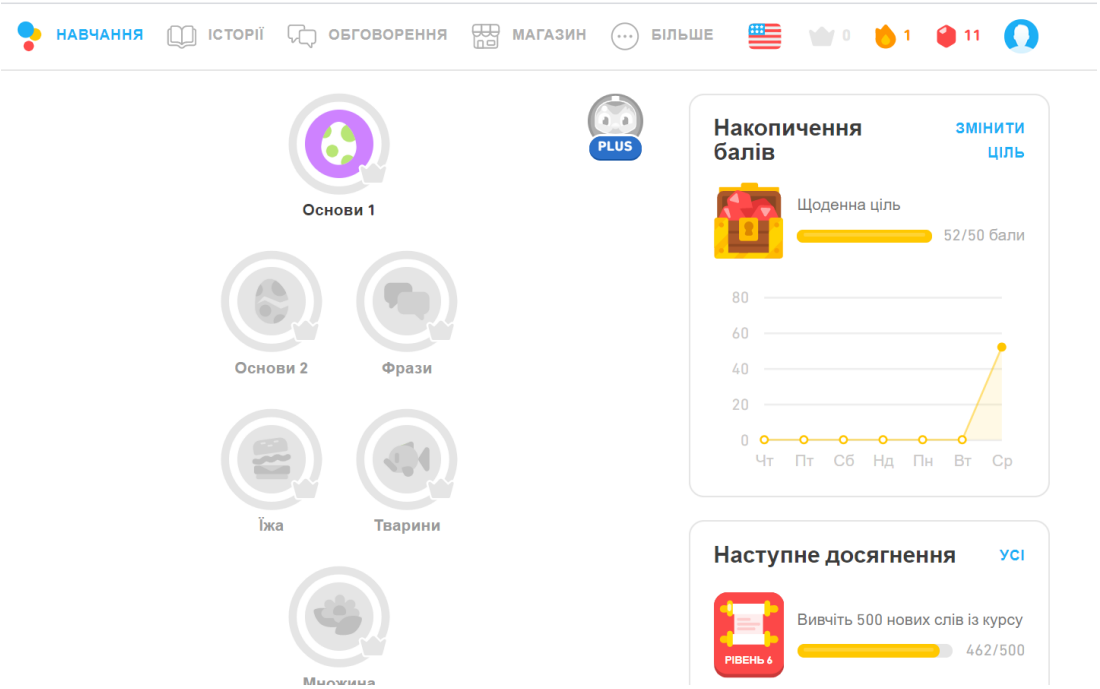


Рис. 1.5. Інтерфейс Duolingo [8]

«Duolingo» першими використали підхід, який використовує машинне навчання та обробку природніх мов для створення завдань для оцінювання знань з англійської мови. Такий підхід розроблений для спрощення створення навчальних завдань так, як при використані ML (Machine Learning) та NLP (Natural Language Processing) немає необхідності в пілотному тестуванні з людьми. Також використання ML та NLP дозволяє створювати велику кількість унікальних завдань, це дозволяє використовувати такий підхід для сертифікованого оцінювання рівня знань з англійської мови [4]. Завдання розроблені для оцінювання здатності розуміти письмову та розмовну англійську мову з різних тем, жанрів, різної складності, а також писати чи говорити на різні теми та з різними цілями.

Для оцінювання словникового запасу використовуються тести у форматі так/ні. У цих тестах потрібно відповісти «так», якщо слово є справжнім, або «ні» якщо надане слово насправді є англоподібним псевдословом (морфологічно та фонологічно правдоподібне, але в англійській мові значення не мають). Псевдослова були згенеровані за допомогою нейронної мережі, яка була натренована на англійському словнику.

C-тести використовуються для оцінювання навичок читання та писання. Такі завдання містять фрагменти тексту в яких деякі слова мають пропущенні літери, ціль завдання заповнити пропущені літери. Завдання показують наскільки добре студенти можуть обробляти тексти різної абстрактності та складності порівняно з більш короткими, більш конкретними текстами.

Для оцінювання навичок слухання та письма, використовується диктант. Для того, щоб успішно відповісти, потрібно проаналізувати окремі слова та зрозуміти їхні граматичні взаємозв'язки, перш ніж вводити почуте. Частина письмового завдання диктанту вимірює знання орфографії та граматики.

Навички мовлення, оцінюються за допомогою завдання, яке вимагає вимовити надане речення вголос. Таке завдання оцінює точність та вміння усно використовувати англійську мову. Це завдання націлене на мовні навички на рівні речення, які включають прості та складні компоненти. Крім того, як диктант, так і мовлення також вимірюють словниковий запас студента.

Завдання створюються за допомогою ML та NLP. Для створення завдань спочатку відбувається накопичення речень, які оцінені за рівнем складності за шкалою CEFR [10]. Для оцінювання речень використовується лінійна регресія на основі натренованої моделі. Для тренування класифікаційної моделі машинного навчання були підготовлені речення, кожне з яких було позначено рівнем складності за шкалою CEFR. Створена модель використовується для оцінювання речення зі статей з Вікіпедії та з фрагментів художньої літератури, які пройшли перевірку експертами на те, що їх вміст є допустимим. Оцінені речення використовуються для диктантів та оцінки мовлення. C-тести також генеруються на основі опрацьованих речень за допомогою NLP.

Висновки до розділу 1

В даному розділі було досліджено системи для дистанційного навчання англійської мови. Були розглянуті переваги дистанційного навчання та актуальність створення системи для дистанційного навчання англійської мови. Досліджено існуючі системи для дистанційного навчання англійської мови. Проаналізовані переваги та недоліки існуючих систем для дистанційного навчання англійської мови.

Існуючі системи мають ряд недоліків. Системи, які мають великий вибір навчального матеріалу коштують доволі дорого, що робить їх менш доступними, а безкоштовні системи мають доволі вузьку спеціалізацію і не мають завдань для розвитку усіх необхідних навичок. У більшості своїй сервіси для дистанційного навчання англійської мови створюють навчальний матеріал за допомогою групи вчителів та не надають широкого вибору персоналізації навчання. Лише сервіс «Duolingo» використовує засоби машинного навчання та обробку природніх мов для створення завдання, але створені завдання таким способом використовуються тільки для сертифікованого оцінювання знань учнів. Також є лише чотири види завдань, які створені таким способом.

За результатами аналізу проведеного дослідження систем для дистанційного навчання англійської мови, було зроблено висновок, що на сьогоднішній день є актуальним створення системи для дистанційного навчання англійської мови, яка б широко використовувала сучасні технології для персоналізації навчання та використовувала б машинне навчання та обробку природніх мов для створення великого набору різноманітних завдань.

РОЗДІЛ 2
ПРОЕКТУВАННЯ СИСТЕМИ ДЛЯ ДИСТАНЦІЙНОГО НАВЧАННЯ АНГЛІЙСЬКОЇ МОВИ

2.1 Вимоги до системи для дистанційного навчання англійської мови

Система для дистанційного навчання англійської мови повинна забезпечити потенційних користувачів достатньою кількістю навчального матеріалу для того, щоб користувач міг досягти успіхів у навчанні англійської мови. Система повинна надати користувачу можливість отримати необхідні знання з англійської мови. Система повинна надавати навчальний матеріал для покращення користувачами навичок читання, писання, слухання та для розширення словникового запасу.

Для покращення навичок писання повинні бути створені тестові завдання на різні теми та з різними рівнями складності. Тестові завдання повинні бути у різних формах: звичайні тести с трьома варіантами відповідей, з вибором правильної відповіді з двох варіантів, з відкриттям дужок. Для покращення навичок читання повинні бути надані тексти різних типів та з різними рівнями складності. Для покращення навичок слухання, повинні бути надані відеоматеріали на різні теми та з різними рівнями складності. Також для покращення навичок слухання повинні бути надані завдання на введення почутого речення. Для розширення словникового запасу повинна бути надано можливість пошуку тлумачення слів на англійській мові та можливість додавання слів до персонального словника користувача. Також повинні бути створені вправи для тренування знань слів.

Для персоналізації навчання користувачів повинен бути створений розділ з курсами для навчання. Курси повинні бути для кожного рівня знань. Курси повинні містити вправи для тренування всіх навичок. Усі вправи повинні відповідати рівню знань користувача. Тематика вправ повинна відповідати тому, що необхідно знати на відповідно рівні знань. Завдяки курсам, користувачу буде легше орієнтуватись у тому, що він повинен

вивчити, також користувачу буде легше відслідковувати свій прогрес у навчанні.

Для того, щоб забезпечити велику кількість граматичних вправ, повинен бути розроблений метод для автоматичного створення нових вправ з використанням обробки природніх мов. Метод для автоматичного створення нових вправ повинен забезпечувати створення завдань з певного тексту на різні теми й з різним рівнем складності. Такий метод дозволить швидко отримати велику кількість різноманітних вправ, що необхідно користувачам для практики різних навичок.

2.2 Огляд способів створення системи для дистанційного навчання англійської мови

Для того, щоб система дистанційного вивчення англійської мови, була успішною і популярною серед користувачів, вона повинна бути реалізована, як веб-додаток. Так, як мережа інтернет доступна у всьому світі й до неї підключенні мільярди пристроїв, тому це найкращий спосіб для поширення інформації [11]. Для дистанційної системи для навчання англійської мови це важливо тому, що це надає можливість залучити велику кількість користувачів.

Веб-додатки – це програмне забезпечення, яке використовує веб-браузер для виконання певної функції [12]. Інтернет є чудовим і недорогим каналом для зберігання та відображення всіх необхідних даних користувачам. Веб-додатки можуть бути розроблені для різних цілей і використовуватися компаніями чи приватними особами. Сучасні веб-додатки можуть бути використані для складних задач, наприклад, за допомогою веб-додатків користувачі можуть працювати над спільними документами та проектами [13]. Користувачі можуть створювати звіти, файли та обмінюватися інформацією з будь-якого місця та з будь-якого пристрою. Користувачу для використання веб-додатку потрібен лише пристрій на якому встановлений веб-браузер та який має доступ до інтернету.

Як правило, веб-додаток має клієнтську та серверну частини. Клієнтська частина – це частина додатку, яку користувач безпосередньо використовує для керуванням додатку [14]. Серверна частина відповідає за обробку та зберігання даних, які були введені користувачем. Клієнтська та серверна частини обмінюються даними через комп'ютерну мережу. Клієнти ініціюють сеанси зв'язку із серверами, які очікують на вхідні запити. Прикладами комп'ютерних програм, що використовують модель клієнт-сервер, є інтернет-магазини, електронна пошта та соцмережі.

Система для дистанційного навчання англійської мови також може бути реалізована як нативний додаток. Нативний додаток – це програмне забезпечення, яке розроблено спеціально для певної платформи та вимагає встановлення на пристрій користувача [15]. Такі додатки вимагають створення окремих програм для кожної платформ, що ускладнює розробку програмного забезпечення. Їх перевагою є те, що вони можуть працювати без доступу інтернету. До недоліків можна віднести те, що їх важче оновлювати ніж веб-додатки так, як оновлення вимагає дій від користувача.

Ще одним способом створення програмного забезпечення є гібридні додатки. Такі додатки поєднують в собі веб-додатки та нативні додатки. Гібридні додатки встановлюються на пристрій користувача, як і нативні. Встановлений додаток є «клієнтом» у клієнт-серверній архітектурі й використовує серверну частину для отримання та зберігання даних, як і веб-додатки. Гібридні додатки можуть використовувати ту саму серверну частину, що і веб-додатки.

При розробці програмного забезпечення спочатку можна створити серверну частину та клієнтську частину для веб-браузера, яка буде доступна на всіх платформах. Потім поступово можна створювати нативні додатки для різних платформ, які будуть використовувати ту саму серверну частину, що і веб-додаток. Такий підхід до розробки програмного забезпечення дозволяє як

найшвидше створити додаток, який буде доступний широкій аудиторії й дозволяє, відносно, швидко його покращувати.

2.3 Архітектура програмного забезпечення

2.3.1 Огляд архітектури веб-додатків

Архітектура програмного забезпечення – це шаблони та техніки, що використовуються для проектування та створення програмного забезпечення [16]. Архітектура програмного забезпечення надає найкращі практики, яких слід дотримуватися під час створення додатків, для того щоб додаток вийшов добре структурований. Шаблони дизайну програмного забезпечення описують рішення для типових проблем. Шаблони можна зв’язати, щоб створити більш загальну архітектуру програмного забезпечення. Замість того, щоб повністю створювати архітектуру самостійно, можуть бути використані існуючі шаблони дизайну, які гарантують, що все працюватиме так, як слід. Як частина архітектури додатків розглядаються як клієнтські, так і серверні частини. Архітектура програмного забезпечення є відправною точкою для побудови програми.

Добре побудована архітектура програмного забезпечення надає такі переваги [17]:

- Стабільно висока продуктивність додатку. Мета веб-додатку – правильно виконувати призначені функції та легко адаптуватися до вимог. Залежно від призначення додатку, може бути знайдено ефективне рішення, яке витримує різні навантаження та зберігає високу продуктивність.
- Масштабованість та гнучкість. У міру розвитку технологій програмне забезпечення повинно бути готове використовувати їх, щоб зберегти конкурентоспроможність. При розробці архітектури веб-додатків насамперед треба зосереджуватись на забезпеченні простих способів модифікації та масштабування програми. Якщо

додаток технічно важко масштабувати, підприємства стикаються з простоєм, що призводить до значних втрат часу та грошей.

– Проста інтеграція нових функцій. Поділивши цілу систему на більш дрібні частини, команда розробників може легко інтегрувати нову функціональність, не впливаючи на структуру та роботу всієї програми.
– Менший час розробки. Архітектура додатку повинна дозволяти розробникам працювати паралельно над розробкою. Чистий код та зрозуміла структура дозволяють залучити до команди більше спеціалістів з програмного забезпечення та значно скоротити час розробки.

Характеристики поганої архітектури програмного забезпечення:

– Додаток важко модифікувати. З кожною зміною однієї частини програми, в інших частинах може виникнути занадто багато залежностей.
– Крихкість. Додаток є крихким, якщо він не працює належним чином або перестає працювати після модифікацій.
– Відсутність можливості повторно використовувати код при створені подібних функцій у додатку, що збільшує час розробки.

Усі веб-додатки базуються на клієнт-серверній архітектурі. В рамках цієї моделі клієнт – це браузер або програма, яка надсилає запити на сервер для того, щоб зробити деяку дію. Сервер визначає, чи можливо зробити якусь дію, і надсилає відповідь клієнту. Є декілька способів того, як може бути організована клієнт-серверна система.

Дворівнева модель – це традиційна структура веб-додатків, що складається з двох основних частин:

– Клієнтська частина – браузер, який отримує необхідні дані, надані сервером. Він відображає користувальницький інтерфейс, відправляє запити на сервер і обробляє відповіді з нього.

– Серверна частина – веб-сервер, який надсилає дані на сторону клієнта, що дозволяє клієнту виконати певне завдання.

Інтернет забезпечує зв'язок між браузером та веб-сервером за допомогою HTTP-запитів та відповідей на них. Цей тип архітектури є найбільш актуальним для невеликих веб-додатків. У рамках цієї моделі небажана велика кількість користувачів та надсилання великої кількості запитів на сервер. Окремий сервер може не витримати значного навантаження. Якщо він виходить з ладу, додатком стає неможливо користуватись.

Багаторівнева архітектура – це традиційна архітектура, яка часто використовується для побудови корпоративних програм, і часто асоціюється зі застарілими програмами [18]. У багаторівневій архітектурі існує кілька рівнів, часто три, але може бути й більше, що утворюють додаток, кожен рівень має свою відповідальність. Рівні допомагають управляти залежностями та керувати великою кількістю програмного коду. У багаторівневій архітектурі рівні розташовані горизонтально, тому вони можуть викликати лише нижчий рівень. Рівень може викликати інший рівень безпосередньо під ним, або він може викликати будь-який із рівнів під ним. Рівень ніколи не може викликати інші рівні, які знаходиться вище за нього.

Як правило, розглядають три найпоширеніших рівня:

– Рівень презентації. Відповідає за відображення даних користувачеві та за введення даних.
– Рівень бізнес логіки. Відповідає за головну функціональність додатку.
– Рівень даних. Відповідає за збереження та доступ до даних.

Монолітна архітектура – це також архітектура пов’язана із застарілими системами, вона передбачає, що усі функціональні можливості знаходяться в межах однієї програми [19]. Це тісно пов’язано зі взаємодією між сервісами та в тому, як вони розробляються та надаються. Оновлення або масштабування однієї частини монолітної програми має наслідки для всієї програми та її

базової інфраструктури. Зміна коду в одній функціональності програми вимагає релізу всієї програми. Через це оновлення та нові релізи зазвичай можуть відбуватися лише один-два рази на рік і можуть включати лише загальне технічне обслуговування замість нових функцій. На відміну від цього, більш сучасні архітектури намагаються розподілити сервіси за функціональністю, щоб забезпечити більшу гнучкість.

Мікросервісна архітектура – це і архітектура, і підхід до написання програмного забезпечення [20]. За допомогою мікросервісів програми розбиваються на найдрібніші компоненти, незалежні один від одного. Кожен із цих компонентів або процесів є мікросервісом. Мікросервіси розподілені та слабко зв’язні, тому вони не впливають один на одного. Це надає переваги як для динамічної масштабованості, так і для стійкості до несправностей: окремі сервіси можна масштабувати за потребою, не вимагаючи великої інфраструктури, також якщо один з сервісів перестане працювати, це не вплине на інші сервіси. Метою використання мікросервісною архітектури є швидше створення якісного програмного забезпечення. Команда розробників може одночасно розробляти кілька мікросервісів. Оскільки сервіси розгортаються самостійно, не має потреби розгортати весь додаток, коли вносяться зміни в одному сервіси. Це дозволяє більшій кількості розробників одночасно працювати над окремими сервісами, замість того, щоб вносити зміни у весь додаток, що призводить до зменшення часу, витраченого на розробку, і можливості частіше випускати нові функції.

Подійно-орієнтована архітектура – це архітектура в якій фіксація, зв’язок, обробка та збереження подій є основною структурою рішення [21]. Це відрізняється від традиційної моделі, керованої запитом. Подія – це будь-яка значна зміна стану для системного обладнання чи програмного забезпечення. Джерело події може бути з внутрішніх або зовнішніх сервісів. Подійно-орієнтована архітектура забезпечує мінімальну зв’язність між компонентами системи, що робить її хорошим варіантом для сучасних розподілених додатків.

Подійно-орієнтована архітектура складається з джерела подій та споживачів подій. Джерело події виявляє подію і представляє подію як повідомлення. Джерело не знає про споживача події або про результат події. Після виявлення події вона передається від джерела події споживачам події через канал подій, де платформа обробки подій асинхронно обробляє подію. Подійно-орієнтована архітектура може базуватися на моделі видавець/підписник або на моделі потоку подій. Модель видавець/підписник заснована на підписках на потік подій. За цією моделлю, після того, як подія публікується видавцем, вона надсилається підписникам, які оброблюють подію. Це відрізняється від моделі потокової передачі подій, коли споживачі подій не підписуються на потік подій. Натомість вони можуть читати з будь-якої частини потоку і можуть приєднатися до потоку в будь-який час. Події фіксуються, з різних джерел, це можуть бути пристрої або програми, що дозволяє видавцям подій та споживачам подій обмінюватися інформацією про стан та результати подій в режимі реального часу.

2.3.2 Опис архітектури сервісу для дистанційного навчання англійської мови

Для розробки системи для дистанційного навчання англійської мови використовується клієнт-серверна архітектура.

Для клієнтської частини використовується односторінковий застосунок, який містить на одній HTML сторінці увесь вміст клієнтської частини, що дозволяє забезпечити досвід робити з додатком близький до користування нативним додатком.

Для серверної частини використовується мікросервісна архітектура з використання багаторівневої архітектури у кожному окрему сервісі та з використанням подійно-орієнтованої архітектури в деяких сервісах для забезпечення асинхроної взаємодії.

Мікросервісна архітектура були вибрана тому, що вона має ряд переваг, які можна використати при розробці системи для дистанційного навчання англійської мови:

- Ізоляція несправностей. Несправності в одному сервісу, у більшості випадків, не будуть впливати на роботу інших сервісів.
- Можливість вибирати різні технології для різних сервісів. Мікросервіси забезпечують гнучкість для використання нового стеку технологій за потреби в окремому сервісі. Проблем із залежністю буде не так багато, і відкат змін набагато простіший. З меншою кількістю коду можна отримати більшу гнучкість.
- Простіша розробка. Завдяки тому, що сервіси невеликі й в них набагато менше коду ніж в монолітних додатках, розробникам легше орієнтуватись у функціональності сервісу.
- Легше й швидше розгортання. Менша кодова база і область застосування сервісів, дозволяє набагато швидше розгортати їх.
- Масштабованість. Оскільки сервіси є окремими, вони можуть легше масштабуватись в необхідніший час, на відміну від монолітного додатку. Якщо все зробити правильно, це може вплинути й на економію коштів.

Серверна частина була розділена на мікросервіси за їх призначенням (Рис. 2.1):

- сервіс, який перенаправляє усі запити з клієнтської частини на конкретний сервіс
- сервіс, який відповідає за авторизацію та аутентифікацію користувачів
- сервіс, який керує завданнями
- сервіс, який керує відео та аудіо матеріалами
- сервіс, який керує статистикою користувачів
- сервіс, який керує курсами

– сервіс, який керує словником
– сервіс, який займається управлінням файлів
– сервіс, який займається аналізом текстів для генерації завдань

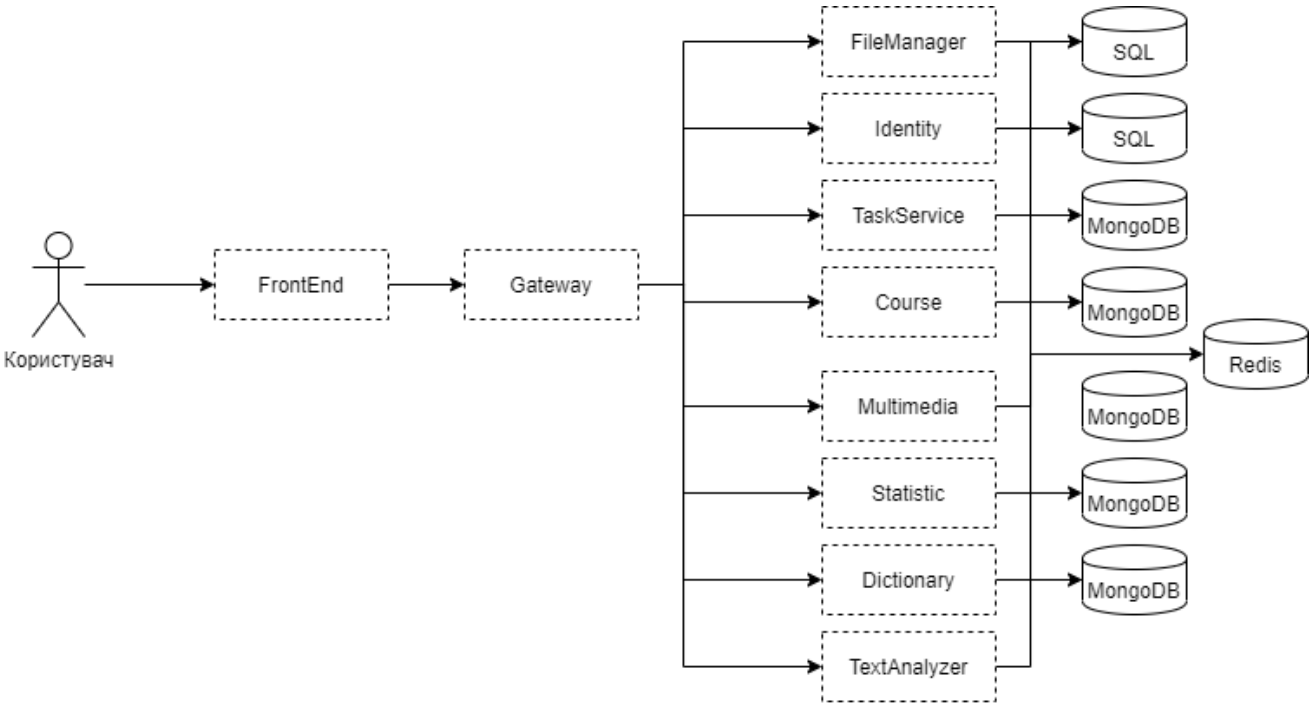


Рис. 2.1. Компоненти системи

2.4 Вибір технологій для розробки програмного забезпечення

2.4.1 Вибір фреймворку для розробки мікросервісів

Веб-фреймворк – це архітектура, що містить інструменти, бібліотеки та функціональні можливості, придатні для побудови та підтримки веб-проектів за допомогою швидкого та ефективного підходу. Вони призначені для впорядкування програм та сприяють повторному використанню коду.

Розглянемо фреймворки, які найбільш популярні для розробки веб-додатків з мікросервісною архітектурою:

– Django – це високоякісний веб-фреймворк для мови програмування Python, який дозволяє швидко створювати безпечні та надійні веб-додатки. Django піклується про більшу частину клопоту веб-розробки, тому розробники можуть зосередитись на написанні свого

додатку, не вимагаючи вирішення поширених проблем, рішення для яких надає Django. Це безкоштовний фреймворк з відкритим кодом, має велику та активну спільноту, документацію та підтримку. Django надає багато інструментів для розробки веб-додатків.

– Flask – це легкий мікрофреймворк для мови програмування Python, заснований на Werkzeug, Jinja2. Його називають мікрофреймворком, оскільки він має на меті зберегти свою основну функціональність невеликою, але, як правило, розширюваною для того, щоб бути корисним як для малих так і для великих додатків. Flask чудово підходить для мікросервісної архітектури.
– ASP.NET Core – це сучасний фреймворк з відкритим кодом для мови програмування C#, яка забезпечує створення сучасних веб-додатків. Він був розроблений для забезпечення швидкої та легкої розробки додатків, які розгортаються в хмарному середовищі або працюють локально. Фреймворк складається з невеликих модулів з мінімальними накладними витратами, завдяки чому зберігається гнучкість при розробці додатків. Додатки створенні з використанням ASP.NET Core можуть розроблятись та запускатись на Windows, Mac та Linux.
– Spring Cloud – це фреймворк для створення веб-додатків для мови програмування Java. Фреймворк полегшує розробку програм, надаючи рішення для багатьох загальних проблем, з якими стикаються при переході до розподіленого середовища.

При розробці системи для дистанційного навчання англійської мови використовується ASP.NET Core фреймворк так, як він створений для мови програмування C#, також він швидкий, легкий та має багато різних бібліотек, які вирішують поширені проблеми.

EnglishLearning.TextAnalyzer сервіс розроблений з використанням мови програмування Python так, як для мови програмування Python створені

найпопулярніші бібліотеки для обробки природніх мов. Для EnglishLearning.TextAnalyzer використовувався фреймворк Flask так, як він підходить більше ніж Django для розробки невеликих мікросервісів.

2.4.2 Вибір фреймворку для розробки клієнтської частини

Для створення клієнтської частини використовується односторінковий додаток. Односторінковий додаток – це додаток, який працює всередині браузера і не вимагає перезавантаження сторінки під час використання. Приклади односторінкових додатків: Gmail, Google Maps, Facebook та GitHub [22].

Односторінкові додатки створенні для розробки великих інтерфейсів користувача, намагаючись імітувати «природнє» середовище у браузері – без перезавантаження сторінки, без додаткового часу очікування. Це лише одна веб-сторінка, увесь зміст якої завантажується під час першого відкриття сторінки за допомогою JavaScript – від якого односторінкові додатки сильно залежать. Односторінкові додатки допомагають утримувати користувача в одному, зручному веб-просторі, де вміст представляється користувачеві просто, легко та ефективно.

Переваги односторінкових додатків:

– Односторінкові додатки працюють швидко, оскільки більшість ресурсів (HTML + CSS + JS) завантажуються лише один раз протягом усього терміну роботи програми. Надалі передаються тільки запити на сервер.
– Згодом легше створити мобільний додаток, оскільки розробник може повторно використовувати ту саму серверну частину для нативного мобільного додатка.
– Односторінковий додаток може ефективно кешувати будь-яке локальне сховище. Додаток надсилає лише один запит та зберігає всі дані, тоді він може використовувати ці дані й може працювати навіть в автономному режимі.

Розглянемо найпопулярніші фреймворки для створення односторінкових додатків:

– React – це фреймворк розроблений компанією Facebook. React найпопулярніший фреймворк для розробки односторінкових додатків, який доступний сьогодні. React використовується такими компаніями, як Facebook, Netflix та Airbnb. Основною метою React є створення інтерактивних інтерфейсів користувача з універсальних компонентів. React використовує JSX (розширений синтаксис для JavaScript) для створення шаблонів та реалізує односторонню модель потоку даних для заповнення компонентів даними. Щоразу, коли дані компонентів змінюються, React використовує свій віртуальний DOM для швидкого та ефективного оновлення сторінки. Одним із недоліків є те, що React стосується лише рівня перегляду клієнтської частини, а все інше залишає розробнику. Комусь подобається ця свобода, яку пропонує React, але іншим, особливо новим розробникам, це може здатись неструктурованим підходом до розробки додатків з React. Через багаторазові та постійні оновлення у фреймворку, документація не завжди оновлена, і це впливає на криву навчання для початківців.
– Angular – це фреймворк для розробки односторінкових додатків, який створений компанією Google. На відміно від React, який надає інструменти для рівня перегляду, Angular пропонує комплексне рішення для створення односторінкових клієнтських програм. Компоненти Angular можуть реалізовувати двосторонню прив’язку даних, що дозволяє їм прослуховувати події та оновлювати значення одночасно між батьківськими та дочірніми компонентами. Шаблони – це фрагменти HTML, які дозволяють використовувати спеціальний синтаксис для багатьох функцій Angular. TypeScript є основною

мовою для розробки Angular. Angular має велику спільноту, що спрощує навчання.

– Vue – це фреймворк для створення односторінкових додатків, який на відмінно від Angular та React, створений не великою компанією, а групою ентузіастів. Vue використовується такими компаніями, як Alibaba, Gitlab та Adobe. Vue має доже добру документацію, що значно спрощує вивчення цього фреймворку. Він розроблений так, щоб його можна було застосовувати поступово. Це означає, що Vue можна використовувати на звичайних веб-сторінках, щоб покращити їх функціональність, або він може бути використаний для створення односторінкового додатку. Vue використовує синтаксис шаблону на основі HTML, який дозволяє легко прив'язувати атрибути до базової моделі даних. Він також пропонує окремі файлові компоненти, які зберігають шаблон, код JavaScript та CSS в одному файлі.

При розробці системи для дистанційного навчання англійської мови, було вирішено використовувати Angular так, як він надає широкій набір інструментів для розробки односторінкових додатків і він найкраще підходить для розробки компонентів, які можна використовувати повторно.

2.4.3 Вибір бази даних

База даних – це сукупність інформації, яка організована таким чином, що нею легко керувати та оновлювати. Комп'ютерні бази даних, як правило, містять сукупності записів даних або файлів, що містять інформацію про операції та взаємодії з конкретними клієнтами.

Розглянемо найпопулярніші види баз даних:

– Реляційна база даних – це база даних, яка найкраще підходить для структурованих даних, структура яких змінюється не часто [22]. Реляційні системи управління базами даних (СУБД), в більшості своїй базуються на структурованій мові запитів (SQL). SQL надає багато переваг для додатків, які використовують структуровані дані.

Доступно багато реалізацій SQL (Microsoft SQL Server, MySQL, Oracle, PostgreSQL тощо). Дані зберігаються в таблицях з рядками та стовпцями і, як правило, мають кількісний характер, тому дані легко зберігаються й запити до бази даних швидко обробляються. Недоліком є те, що такі бази даних важко масштабувати. Також реляційні бази даних не підходять для напівструктурованих та неструктурованих даних. Реляційні бази даних, зазвичай, коштують дорожче за нереляційні бази даних.

– Нереляційна база даних – це система управління даними, яка не потребує фіксованої структури. Нереляційні бази даних легко масштабується. Основною метою використання нереляційних баз даних є розподілене сховище даних з величезними потребами в зберіганні даних. Нереляційні бази даних використовується для великих даних та веб-додатків у режимі реального часу. Наприклад, такі компанії, як Twitter, Facebook та Google, збирають терабайти даних користувачів щодня та зберігають їх у нереляційних базах даних. Є різні типи нереляційних баз даних: документо-орієнтовані, ключ-значення, графи.

При розробці системи для дистанційного навчання англійської мови, було вирішено використовувати документо-орієнтовану базу даних MongoDB для сервісів дані яких є напівструктурованими або неструктурованими, реляційну базу даних Microsoft SQL Server для сервісів дані яких є структрурованими, для зберігання даних у вигляді «Ключ-значення» використовуватиметься база даних Redis.

Висновки до розділу 2

У цьому розділі було описано вимоги до системи для дистанційного навчання англійської мови. Був здійснений огляд способів створення системи для дистанційного навчання англійської мови.

Було розглянуті різні архітектурні підходи до розробки веб-додатків, які найбільше підходять для розробки системи для дистанційного навчання англійської мови. Для розробки системи для дистанційного навчання англійської мови використовується клієнт-серверна архітектура. Для клієнтської частини використовується односторінковий застосунок. Для серверної частини використовується мікросервісна архітектура з використання багаторівневої архітектури у кожному окрему сервісі та з використанням подійно-орієнтованої архітектури в деяких сервісах для забезпечення асинхроної взаємодії.

Було здійснено огляд технологій, які необхідні для розробки додатку. Для розробки серверної частини було вирішено використовувати мову програмування C# та фреймворк ASP.NET Core. Для розробки клієнтської частини, було вирішено використовувати фреймворк Angular. Для зберігання даних, було вирішено використовувати документо-орієнтовану базу даних MongoDB для даних, які напівструктуровані або неструктуровані, для структурованих даних було вирішено використовувати реляційну базу даних Microsoft SQL Server.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ СИТЕМИ ДЛЯ ДИСТАНЦІЙНОГО НАВЧАННЯ АНГЛІЙСЬКОЇ МОВИ

3.1 Засоби для розробки програмного забезпечення

Засоби для розробки програмного забезпечення мають велике значення при розробці додатків тому, що ефективні та корисні засоби для розробки можуть значно прискорити розробку додатків та зробити її легшою.

Для розробки системи для дистанційного навчання англійської мови були використані такі засоби для розробки:

- Microsoft Visual Studio – це інтегроване середовище розробки, яка використовується для редагування, налагодження та побудови коду, а потім для розгортання додатку. Інтегроване середовище розробки (IDE) – це багатофункціональна програма, яка може бути використана для багатьох аспектів розробки програмного забезпечення. Крім стандартного редактора коду та налагоджувача, які надають більшість середовищ розробки, Visual Studio включає компілятори, засоби доробки коду, графічні дизайнери та багато інших функцій для полегшення процесу розробки програмного забезпечення.
- PyCharm – це спеціальне інтегроване середовище для розробки додатків з використанням мови програмування Python, що забезпечує широкий спектр важливих інструментів для розробників. PyCharm надає широкий набір інструментів для продуктивної розробки веб-додатків з використанням мови програмування Python.
- Atom – це безкоштовний редактор тексту та програмного коду, розроблений компанією GitHub. Atom дозволяє користувачам встановлювати сторонні пакети та теми для налаштування функцій та зовнішнього вигляду редактора, тому кожен розробник може налаштувати його відповідно до своїх уподобань. Atom

надає пакети для розробки додатків з використанням фреймворку Angular, які містять інструменти для швидкої та зручної розробки Angular додатків.

– Robo 3T – це безкоштовний та легкий графічний інструмент для роботи з базою даних MongoDB. Це інструмент управління MongoDB, який має оболонку, орієнтовану на використання на різних платформах. Цей інструмент не є типовим для інших адміністративних інструментів MongoDB для користувальницького інтерфейсу. За допомогою цієї оболонки користувач може переглядати, редагувати та видаляти документи з бази даних MongoDB. Robo 3T – це проект з відкритим кодом, тому він абсолютно безкоштовний.

3.2 Опис методу автоматичного створення завдань

Для того, щоб забезпечити велику кількість граматичних завдань в системі для дистанційного навчання англійської мови, розроблено метод для автоматичного створення завдань. Розробка даного методу потрібна для того, щоб забезпечити велику кількість різноманітних вправ у системі, що необхідно користувачам для практики різних навичок. Для автоматичного створення завдань використовується обробка природної мови (NLP).

Обробка природної мови – це область штучного інтелекту, яка надає можливість комп’ютерам читати та розуміти природні мови [25]. Це дисципліна, яка зосереджена на взаємодії між наукою про дані та природньою мовою, і охоплює багато галузей. Сьогодні обробка природної мови процвітає завдяки величезному вдосконаленню доступу до даних та збільшенню обчислювальних потужностей, які дозволяють досягти значних результатів у таких сферах, як охорона здоров’я, медіа, фінанси та серед багатьох інших.

Природна мова надзвичайно складна і різноманітна. Ось чому обробка природної мови включає багато методів її інтерпретації, починаючи від статистичних методів та методів машинного навчання, закінчуючи

заснованими на правилах та алгоритмічними підходами. Є п’ять основних завдань обробки природніх мов:

- Позначення частини мови. У кожному реченні визначається частина мови для кожного слова. Частина мови – це категорія слів, що мають подібні граматичні властивості. Існує велика кількість слів, які можуть служити кількома частинами мови, що робить складним завданням позначення частин мови.
- Лематизація. Лематизація – це процес видалення закінчень і зведення слова до базової форми, яка також відома як «лема». Минулий час змінюється на теперішній, а синоніми уніфікуються. Наприклад, слово «бігав» у минулому часу змінюється на лему «біг», а «найкращий» змінюється на лему «хороший».
- Лексичний аналіз. Лексичний аналіз розділяє текст на менші частини, які називаються лексемами. Цей процес сегментує великий текст на окремі речення та слова, одночасно видаляючи певні символи, такі як розділові знаки.
- Усунення неоднозначності. Усунення неоднозначності пов’язане зі значенням слів, які використовуються в природній мові. Деякі слова мають більше одного значення, і під час читання люди підбирають значення, яке має найбільший сенс у даному контексті.
- Семантика. Люди спілкуються на основі значень та контексту. Семантика допомагає комп’ютерам визначити структуру речень та найважливіші елементи тексту, щоб зрозуміти тему, що обговорюється. Наприклад, якщо текст містить такі слова, як вибори, демократ і республіканець чи бюджет, податки та інфляція, комп'ютер розуміє, що обговорюваною темою є американська політика та економіка.

Обробка природної мови використовується для вирішення багатьох завдань й постійно збільшується кількість сфер застосування.

Ось декілька завдань для вирішення яких застосовується обробка природніх мов.

- Обробка природної мови дозволяє розпізнавати та передбачати захворювання на основі електронних медичних карт та обробки мовлення пацієнта. Ця можливість досліджується для пацієнтів з різними захворюваннями, від серцево-судинних захворювань до депресії і навіть шизофренії.
- Організації можуть дізнатись, що говорять клієнти про їх послуги та товари, отримуючи інформацію з таких джерелах, як соціальні мережі та відгуки на веб-сайтах.
- Обробка природної мови використовується для виявлення спаму в листах електронної пошти.
- Обробка природної мови використовується для виявлення неправдивих новин у соцмережах.

Для розробки методу для автоматичного створення завдань, обробка природної мови використовується для пошуку речень, які можна використовувати як основу для завдання з певної теми.

В систему для аналізу текстів надсилається файл для аналізу. Спочатку відбувається лексичний аналіз при якому вхідний текст розбивається на речення, а кожне речення у свою чергу розділяються на слова. Після цього відбувається перевірка на те, що речення не містить недопустимий вміст, наприклад сленгові слова. Потім для кожного слова відбувається процес визначення частини мови, визначення залежностей з іншими словами та визначення леми для слова. Для визначення речень, які можна використовувати для створення завдань, задані правила, які визначають чи відповідає речення вимогам, щоб використовуватись для завдання з певної теми.

Наприклад розглянемо правило для визначення речень, які можуть використовуватись в завданнях для стверджувальних речень з теми «Present

Simple». Для того, щоб стверджувальне речення могло бути використано для завдань на тему «Present Simple», воно повинно відповідати таким вимогам:

- Речення повинно починатись з іменника чи займенника, також перед іменниками може бути використаний артикль.
- Після цього, необов’язково, розташований прислівник.
- Потім повинно йти дієслово у третій формі для третьої особи або у першій для всіх інших.
- Після цього можуть йти другорядні члени речення.
- Речення повинно закінчуватись крапкою або знаком оклику.

Після виконаного аналізу, речення які можна використовувати для створення надсилаються сервісу, який відповідає за роботу з завданнями, за допомогою потоку подій. Сервіс, який відповідає за роботу з завданнями, перед зберіганням речень в своїй базі даних, надсилає запити у сервіс, який відповідає за роботу зі словами, на те який рівень складності має речення й зберігає цю інформацію разом з реченням. У збережених реченнях зберігається інформація про частину мови, залежності з іншими словами речення та лему для кожного слова. Рівень складності речень визначається за тим, який найскладніший рівень складності мають слова у реченні.

Збереженні речення можуть використовуватись сервісом для створення різних типів завдань. Для завдань, в яких потрібно ввести прослухане речення, ніякої додаткової обробки речень не потрібно й речення зразу можна використовувати для завдань такого типу. Для інших завдань потрібна додаткова обробка речення для того, щоб використовувати його у завданнях. Наприклад, для того, щоб стверджувальне речення для теми «Present Simple» можна було використовувати у завданні з розкриттям дужок, повинно бути визначено дієслово присудок, яке буде розташоване у дужках. Дієслово присудок можна визначити використовуючи інформацію, яка була надано після аналізу речення – до якої частини мови відноситься слово та які воно має залежності з іншими словами.

Завдання, які були створені автоматичним методом можуть бути використані в розділі завдань та у розділі курсів, що дозволить користувачам мати великий вибір різноманітних завдань для їх потреб.

Компоненти системи, які приймають участь в автоматичному створені завдань зображені на Рис. 3.1.

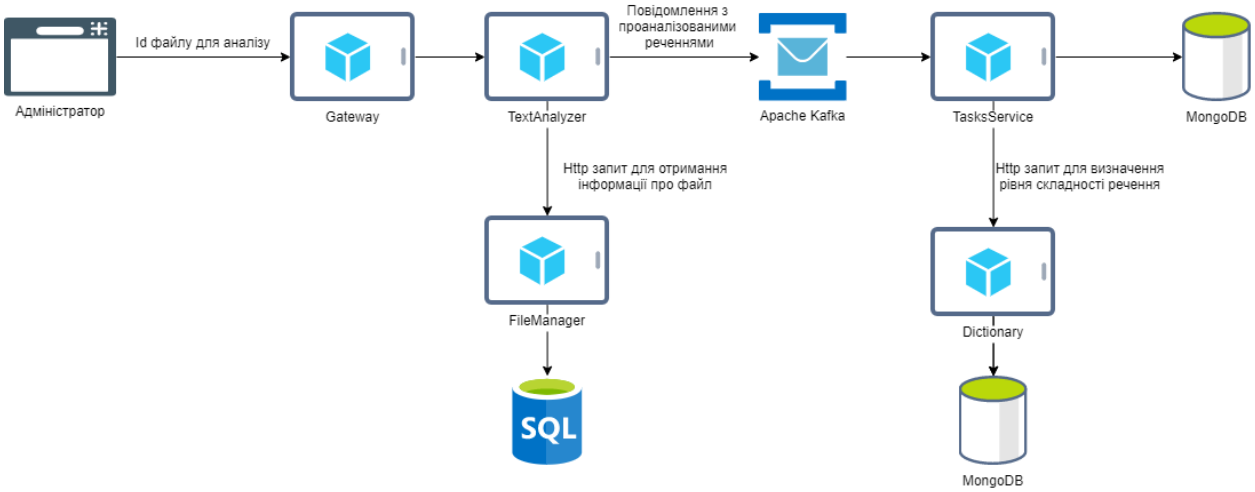


Рис. 3.1. Компоненти системи, які приймають участь в автоматичному створені завдань

3.3 Опис клієнтської частини

Для розробки клієнтської частини використовується фреймворк Angular.

Angular – це фреймворк для розробки односторінкових клієнтських додатків за допомогою HTML та TypeScript. Angular реалізує основну та додаткову функціональність, як набір бібліотек TypeScript, які імпортуються в додатки, коли вони необхідні.

Архітектура Angular додатків спирається на певні фундаментальні концепції. Основними будівельними елементами Angular є компоненти, які організовані в модулі [25]. Модулі збирають відповідний код у функціональні набори. Додаток Angular визначається набором модулів. Додаток завжди має кореневий модуль, який є основним у додатку й має назву «AppModule», як правило, додаток має багато функціональних модулів. Кожен модуль може імпортувати функціональність з інших модулів. Організація коду в окремі функціональні модулі допомагає керувати розробкою складних додатків та розробкою функціональності для повторного використання.

Кожен додаток Angular обов’язково має кореневий компонент, який з’єднує ієрархію компонентів із об’єктною моделлю документа сторінки (DOM). Компонентів може бути багато. Кожен компонент визначає клас, який містить певну функціональність та пов’язаний із ним шаблон HTML, який визначає, що відображатиметься у цільовому середовищі.

Шаблон поєднує HTML з розміткою Angular, яка може змінювати елементи HTML перед їх відображенням. Директиви шаблонів забезпечують прив’язку функціональності програми та розмітки, що пов’язує дані додатка та DOM. Перш ніж відображається подання, Angular оцінює директиви та аналізує прив’язки в шаблоні, щоб змінити елементи HTML та DOM відповідно до даних програми та логіки. Angular підтримує двосторонню прив’язку даних, тобто зміни в DOM, такі як вибір користувачів, також відображаються у даних додатку.

Для даних або логіки, які не пов’язані з певним представленням або для яких потрібно надати спільний доступ між компонентами, використовуються класи, які мають назву сервіси.

Angular модуль маршрутизатор (Router) надає сервіс, який дозволяє визначити шлях навігації між різними станами програми та різними представленнями. Він створений за зразком звичних правил навігації веб-браузера:

– Після вводу URL-адреси в адресний рядок, веб-браузер переходить на відповідну сторінку.
– Після натискання на посилання на сторінці, веб-браузер переходить на нову сторінку.
– Після натискання на кнопки браузера вперед і назад, веб-браузер переміщається вперед і назад по історії сторінок.

Маршрутизатор відображає URL-подібні шляхи до відображень замість сторінок. Коли користувач виконує дію, наприклад, натискання на посилання,

яка завантажує нову сторінку у браузері, маршрутизатор перехоплює поведінку браузера та показує або приховує ієрархію представлень.

Щоб визначити правила навігації, потрібно зв’язати шляхи навігації з компонентами. Шлях має URL-подібний синтаксис, який поєднує дані додатку, приблизно так само, як синтаксис шаблону поєднує представлення з даними додатку. Можна застосувати логіку програми, щоб вибрати, які подання відображати чи приховувати, в залежності від дій користувача та правил доступу.

Для того, щоб інтерфейс користувача мав привабливий вигляд використовується фреймворк Bootstrap. Bootstrap – це фреймворк, який робить створення веб-сторінок швидким та простим завдяки використанню попередньо відформатованих віджетів. Він містить величезну бібліотеку стилізованих віджетів та фреймів, які дозволяють користувачам швидко створювати привабливі веб-сторінки з дуже невеликими зусиллями. Використання попередньо відформатованих шаблонів для віджетів, користувачі Bootstrap економлять час, який витрачається на стилізацію кожного окремого елемента. Bootstrap надає добре розроблені шаблони віджетів і фреймів, які добре виглядають і працюють належним чином без будь-яких налаштувань. Базові віджети мають різноманітні варіанти, які можна додатково налаштувати за допомогою CSS.

Клієнтська частина розділена на два додатки: перший надає усю функціональність необхідну для користувачів, другий надає функціональність для керування системою, до якого мають доступи лише адміністратори системи.

Клієнтська частина для надсилання запитів на серверну частину, використовує протокол HTTP і отримує результати у форматі JSON. Усі запити з клієнтської частини надсилаються на сервіс Gateway, який потім перенаправляє запити на конкретні мікросервіси.

Модулі з яких складається додаток Angular, який доступний користувачам:

- Tasks – це модуль, який відповідає за відображення завдань.
- Authentication – відповідає за керування реєстрацією, аутентифікацією та авторизацією.
- Course – відповідає за відображення курсів.
- Dictionary – відповідає за відображення сторінок, які пов’язані з пошуком тлумачення слів та з додаванням слів до персонального словника.
- Multimedia – відповідає за відображення відео та аудіо матеріалів.
- Statistic – відповідає за відображення сторінок, що пов’язані з інформацією про статистику користувача.

Сторінки, які доступні в додатку Angular, який доступний користувачам описані в Таблиця 3.1.

Таблиця 3.1 – сторінки Angular додатку, який доступний користувачам

URL	Опис
/	Коренева сторінка з інформацією про додаток
/sign_in	Сторінка для аутентифікації у додатку
/sign_up	Сторінка для реєстрації у додатку
/course/list	Сторінка, яка надає список доступних вправ для користувача всередині курсу
/course/train/task/:grammarPart	Сторінка для граматичного завдання всередині курсу, grammarPart – тема завдання
/course/train/dictionary/:topic	Сторінка для тренування нових слів всередині курсу, topic – тематика слів
/course/dictionary/list/:topic	Сторінка, яка надає список слів, які необхідно вивчити користувачу для того, щоб пройти курс, topic – тематика слів
/dictionary/word	Сторінка для пошуку тлумачення слів
/dictionary/list	Сторінка, яка надає список слів, які користувач додав до свого персонального словника
/dictionary/details/:word	Сторінка з тлумаченням слова, word – слово для якого надається тлумачення

Модулі з яких складається додаток Angular, який використовуються для керування системою для дистанційного навчання англійської мови і доступний лише для адміністраторів системи:

– Grammar-Analyse – це модуль, що відповідає за відображення сторінок, які пов’язані з аналізом текстів для генерації завдань.
– Tasks – відповідає за відображення сторінок, які пов’язані з керуванням процесу автоматичного створення завдань.
– Uploaded-Files – відповідає за відображення сторінок, які пов’язані з керуванням завантажених файлів у систему.

Сторінки, які доступні в додатку Angular, для керування системою описані в Таблиця 3.2.

Таблиця 3.2 – сторінки Angular додатку, для керування системою

URL	Опис
/grammar-analyse/list	Сторінка, яка надає список виконаних аналізів файлів
/grammar-analyse/analyse/:id	Сторінка з результатами аналізу файлу, id – унікальний ідентифікатор проведеного аналізу файлу
/grammar-analyse/generate/task-items/:id	Сторінка з формою для генерації елементів завдань за результатами аналізу файлу, id – унікальний ідентифікатор проведеного аналізу файлу
/tasks/items	Сторінка, яка надає список елементів завдань
/tasks/create	Сторінка, яка надає форму для створення завдання
/tasks/list	Сторінка, яка надає список завдань
/uploaded-files/	Сторінка, яка надає список завантажених файлів у системі
/uploaded-files/analyse-form/:id	Сторінка, яка надає форму для початку аналізу файлу, id – унікальний ідентифікатор файлу

Таблиця 3.2 (продовження)

/uploaded-files/upload-file/:id	Сторінка, яка надає форму для завантаження файлу в систему, id – унікальний ідентифікатор папки, в яку буде завантажено файл

3.4 Опис серверної частини

Серверна частина складається з дев’яти мікросервісів:

- Gateway – сервіс, який перенаправляє усі запити з клієнтської частини на конкретний сервіс
- Identity – сервіс, який відповідає за авторизацію та аутентифікацію користувачів
- TaskService – сервіс, який керує завданнями
- Multimedia – сервіс, який керує відео та аудіо матеріалами
- Statistic – сервіс, який керує статистикою користувачів
- Course – сервіс, який керує курсами
- Dictionary – сервіс, який керує словником
- FileManager – сервіс, який займається управлінням файлів
- TextAnalyzer – сервіс, який займається аналізом текстів для генерації завдань

Усі сервіси, крім TextAnalyzer, розроблені з використанням мови програмування C# та фреймворком ASP.NET Core. Сервіс TextAnalyzer розроблений з використанням мови програмування Python та фреймворком Flask.

Для комунікації між собою, мікросервіси використовують HTTP запити та обмін повідомленнями через систему обміну повідомленнями Apache Kafka.

Apache Kafka – це розподілена система обміну повідомленнями, яка може обробляти великий обсяг даних і дозволяє передавати повідомлення з одного компоненту системи до іншого. Kafka підходить як для обробки повідомлень в режимі офлайн, так і в інтернеті. Повідомлення Kafka зберігаються на диску та дублікуються в кластері, щоб запобігти втраті даних.

Для Kafka може використовуватись модель видавець/підписник для обміну повідомленнями між сервісами.

Apache Kafka має такі переваги в порівнянні з іншими системами для обміну повідомленнями:

- Надійність – Kafka – це розподілена, розділена система, яка може масштабуватись та не допускає відмов.
- Масштабованість – система обміну повідомленнями Kafka легко масштабується без простою.
- Довговічність – Kafka використовує розподілений журнал логів, що означає, що повідомлення зберігаються на диску душе швидко і не втрачаються.
- Ефективність – Kafka має високу пропускну здатність як для публікації, так і для зчитування повідомлень. Kafka підтримує стабільну роботу навіть при зберіганні терабайтів повідомлень.

Для синтезу мовлення використовується Azure Speech Service, який надає можливість для перетворення тексту в аудіо файл.

3.4.1 Gateway

Сервіс Gateway реалізовує шаблон API Gateway. Сервіс Gateway розроблений для перенаправлення усіх HTTP запитів с клієнтської частини на конкретний мікросервіс. Це дозволяє зберігати в додатку клієнтської частини лише посилання на сервіс Gateway, усі запити до мікросервісів з клієнтської частини будуть проходити через сервіс Gateway.

При розробці сервісу використовувалась бібліотека Ocelot, яка надає інструменти для швидкої реалізації шаблону API Gateway в додатку, який розроблюється з використанням фреймворку ASP.NET Core. Перенаправлення запитів, відбувається за префіксом у шляху запита.

Префікси шляхів за яким здійснюється перенаправлення запитів наведено в Таблиця 3.3.

Таблиця 3.3 – Шляхи перенаправлення запитів

URL	Сервіс
/api/tasks/{everything}	TaskService
/api/multimedia/{everything}	Multimedia
/api/identity/{everything}	Identity
/api/statistic/{everything}	Statistic
/api/file-manager/{everything}	FileManager
/api/text-analyzer/{everything}	TextAnalyzer
/api/dictionary/{everything}	Dictionary
/api/course/{everything}	Course

3.4.2 Identity

Identity сервіс відповідає за аутентифікацію, авторизацію та реєстрацію користувачів.

Аутентифікація – це процес розпізнавання користувача. Це механізм асоціювання вхідного запиту з набором ідентифікаційних даних. Надані облікові дані порівнюються з даними, що містяться на сервері аутентифікації.

Авторизація – це механізм безпеки для визначення рівнів доступу або прав користувача, пов’язаних із системними ресурсами, включаючи файли, послуги, дані та функції додатків. Це процес надання або відмови у доступі до мережевого ресурсу, який дозволяє користувачеві отримати доступ до різних ресурсів на основі ідентифікації користувача.

Якщо запити на аутентифікацію користувача проходить успішно, Identity сервіс надсилає у відповіді JWT токен, який використовується для авторизація в різних компонентах систем.

Identity сервіс використовує Microsoft SQL Server для зберігання даних так, як дані якими керує Identity сервіс структуровані.

Identity сервіс використовує Apache Kafka для надсилання повідомлень про появу нових користувачів у системі, на ці повідомлення підписані TaskService та Course мікросервіси.

3.4.3 Utilities

Utilities – це набір бібліотек, які використовуються в різних мікросервісів, що розроблені з використанням фреймворку ASP.NET Core. Цей набір бібліотек створений для запобіганню написанню одного і того самого коду в різних мікросервісах, замість цього використовується набір бібліотек, які містять інструменти для вирішення поширених проблем.

Utilities складається з таких бібліотек:

- Configurations – містить класи, що використовуються для налаштування системи
- Http – містить методи розширення для роботи з HTTP запитами
- Identity – містить інструменти для налаштування авторизації та аутентифікації в системі
- Linq – включає в себе методи розширення для роботи з колекціями
- MessageBrokers – містить інструменти, що спрощують використання системи для обміну даних Apache Kafka
- Persistence – містить інструменти, що спрощують роботу з базами даних
- Speech – містить обгортку над Azure Speech Services для легшого використання синтезу мови в мікросервісах

3.4.4 Multimedia

Сервіс використовується для надання відеоматеріалів та текстових навчальних матеріалів.

Для зберігання даних Multimedia сервіс використовує базу даних MongoDB так, як дані якими керує сервіс Multimedia неструктуровані та не мають зв’язків між собою.

3.4.5 Statistic

Сервіс використовуються для збору та відображення статистики користування системою користувачем.

Для зберігання даних Statistic сервіс використовує базу даних MongoDB так, як дані якими керує сервіс Multimedia неструктуровані та не мають зв’язків між собою.

3.4.6 FileManager

Сервіс використовується для керування завантаженими файлами у системі.

FileManager сервіс використовує Microsoft SQL Server базу даних для зберігання інформації про файли.

База даних сервісу FileManager містить дві таблиці: Folders(Таблиця 3.4), яка містить інформацію про папки у яких відображуються файли у клієнтській частині та Files (Таблиця 3.5), яка зберігає інформацію про збережені файли.

Таблиця 3.4 – таблиця Folders

Назва колонки	Тип	Опис
Id	INT	Унікальний ідентифікатор папки
Name	NVARCHAR	Назва папки
ParentId	INT	Унікальний ідентифікатор батьківської папки

Таблиця 3.5 – таблиця Files

Назва колонки	Тип	Опис
Id	UNIQUEIDENTIFIER	Унікальний ідентифікатор файлу
Name	NVARCHAR	Назва файлу
Extension	NVARCHAR	Унікальний ідентифікатор батьківської папки
LastModified	DATETIME	Дата, коли файл останній раз змінений
FolderId	INT	Унікальний ідентифікатор батьківської папки
CreatedBy	UNIQUEIDENTIFIER	Унікальний ідентифікатор користувача, який створив файл
Metadata	NVARCHAR	Інформація про файл

Завантажені файли у систему зберігаються на диску.

HTTP маршрути, які надає FileManager сервіс, наведені у Таблиця 3.6.

Таблиця 3.6 – HTTP маршрути FileManager сервісу

Метод	URL	Опис
POST	/ api/file-manager/file	Завантаження файлу у систему
GET	/api/file-manager/file/{id}	Отримання файлу з унікальним ідентифікатором id
GET	/api/file-manager/folder/{id}/info	Отримання інформації про папку
GET	/api/file-manager/tree	Отримання дерева файлів

3.4.7 TextAnalyzer

Сервіс використовується для аналізу файлів і пошуку речень у великих файлів, які можна використовувати для автоматичного створення завдань.

Сервіс розроблений з використанням мови програмування Python та фреймворку Flask. Сервіс не використовує бази даних, лише файлову систему для читання файлів. Також сервіс використовує систему для обміну повідомлень Apache Kafka для надсилання повідомлень про знайдені речення, які можна використовувати для автоматичного створення завдань.

Для пошуку речень, які можна використовувати для автоматичного створення завдань, використовується обробка природної мови. При розробці сервісу використовувалась бібліотека spaCy. spaCy – це бібліотека, яка надає широкий набір інструментів для розробки додатків з використанням обробки природної мови. spaCy надає ефективні інструменти для лексичного аналізу, лематизації, визначення частини мови слова, визначення залежностей, які має слово з іншими словами речення.

TextAnalyzer невеликий сервіс і відповідає лише за одне завдання – пошуку речень, які можуть бути використані для автоматичного створення завдань.

3.4.8 TaskService

TaskService відповідає за керування завданнями.

Сервіс розроблений з використанням мови програмування C# та фреймворком ASP.NET Core.

Сервіс використовує базу даних MongoDB, так, як TaskService відповідає за напівструктуровані дані. Також TaskService використовує систему для обробки повідомлень Apache Kafka. TaskService робить HTTP запити на сервіс Dictionary для отримання інформації, про те який рівень складності мають слова.

Для створення аудіо завдань використовується сервіс Azure Speech Service, який перетворює текст завдання в аудіо.

База даних сервісу TaskService складається з таких колекцій: UserInformation, ParsedSents, GrammarFileAnalyzed, TaskItems, TaskGenerations. UserInformation (Таблиця 3.7) містить інформацію про вподобання зареєстрованих користувачів. ParsedSents (Таблиця 3.8) зберігає проаналізовані речення, які можна використати для створення завдання. GrammarFileAnalyzed (Таблиця 3.9) зберігає інформацію про виконані аналізи файлів. TaskItems (Таблиця 3.10) містить елементи завдань, з яких можна створити нове завдання, TaskGenerations (Таблиця 3.11) зберігає інформацію про згенеровані завдання.

Таблиця 3.7 – колекція UserInformation

Назва поля	Тип	Опис
Id	GUID	Унікальний ідентифікатор користувача
EnglishLevel	String	Рівень англійської користувача
FavouriteGrammarParts	String[]	Масив тем завдань, які найбільше цікавлять користувача

Таблиця 3.8 – колекція ParsedSents

Назва поля	Тип	Опис
Id	String	Унікальний ідентифікатор речення
AnalyzeId	Guid	Унікальний ідентифікатор проведеного аналізу файлу
Sent	String	Речення
SentType	String	Тип речення
EnglishLevel	String	Рівень складності
Tokens	SentToken	Слова у реченні

Таблиця 3.9 – колекція GrammarFileAnalyzed

Назва поля	Тип	Опис
Id	Guid	Унікальний ідентифікатор проведеного аналізу файлу
FileId	Guid	Унікальний ідентифікатор файлу, який використовувався для аналізу
Name	String	Ім’я
CreatedTime	DateTime	Час коли був зроблений аналіз
Path	String[]	Шлях файлу в внутрішній файловій системі сервісу
SentCount	Int32	Кількість речень, які було знайдено при аналізі файлу

Таблиця 3.10 – колекція TaskItems

Назва поля	Тип	Опис
Id	String	Унікальний ідентифікатор елемента завдання
TaskGenerationId	String	Унікальний ідентифікатор проведеної генерації завдань
GrammarPart	String	Тема завдання
TaskType	String	Тип завдання
SentType	String	Тип речення
EnglishLevel	String	Рівень складності
Content	BsonValue	Елемент завадання

Таблиця 3.11 – колекція TaskGenerations

Назва поля	Тип	Опис
Id	String	Унікальний ідентифікатор проведеної генерації завдань
AnalyzeId	Guid	Унікальний ідентифікатор проведеного аналізу файлу
Name	String	Ім’я
CreatedTime	DateTime	Час коли була зроблена генерація завдань
TaskType	String	Тип завдання
SentType	String	Тип речення
EnglishLevel	String	Рівень складності

TaskService оброблює повідомлення, які приходять з TextAnalyze та Identity сервісів, з використанням системи для обробки повідомлень Apache Kafka. Identity сервіс публікує повідомлення UserCreatedEvent (Таблиця 3.12) при створені нового користувача в системі. TaskService зберігає вподобання нового користувача в колекції UserInformation та використовує її для надання завдань користувачу, які будуть найбільше його цікавити.

Таблиця 3.12 – Структура повідомлення UserCreatedEvent

Назва поля	Тип	Опис
Id	GUID	Унікальний ідентифікатор користувача
Email	String	Електрона пошта користувача
EnglishLevel	String	Рівень англійської користувача
FavouriteGrammarParts	String[]	Масив тем завдань, які найбільше цікавлять користувача

TextAnalyze сервіс публікує повідомлення GrammarSentAnalyzedEvent (Таблиця 3.13), яке містить інформацію про речення, яке може використовуватись для створення завдання, та повідомлення GrammarFileAnalyzedEvent (Таблиця 3.14), яке містить інформацію про результати аналізу файлу. TaskService зберігає інформацію з повідомлення

GrammarSentAnalyzedEvent в колекції ParsedSents та інформацію з повідомлення GrammarFileAnalyzedEvent в колекції GrammarFileAnalyzed.

Таблиця 3.13 – структура повідомлення GrammarSentAnalyzedEvent

Назва поля	Тип	Опис
AnalyzeId	Guid	Унікальний ідентифікатор проведеного аналізу файлу
Sent	String	Речення
SentType	String	Тип речення
Tokens	SentToken	Слова у реченні

Таблиця 3.14 – структура повідомлення GrammarFileAnalyzedEvent

Назва поля	Тип	Опис
Id	Guid	Унікальний ідентифікатор проведеного аналізу файлу
FileId	Guid	Унікальний ідентифікатор файлу, який використовувався для аналізу
Name	String	Ім’я
CreatedTime	DateTime	Час коли був зроблений аналіз
Path	String[]	Шлях файлу у внутрішній файловій системі сервісу
SentCount	Int32	Кількість речень, які було знайдено при аналізі файлу

HTTP маршрути, які надає TaskService сервіс, наведені у Таблиця 3.15.

Таблиця 3.15 – HTTP маршрути TaskService сервісу

Метод	URL	Опис
POST	/api/tasks/audio	Створення аудіо завдання
GET	/api/tasks/audio/sent/{id}	Отримання аудіо для речення з ідентифікатором id
POST	/api/tasks/generate	Генерація елементів завдань на основі аналізу файлу
GET	/api/tasks/grammarFileAnalyzed/{id}	Отримання інформації про результати аналізу файлу. Id - унікальний ідентифікатор проведеного аналізу файлу

Таблиця 3.15 (продовження)

GET	/api/tasks/grammarFileAnalyzed/{id}/parsedSents	Отримання речень с проведеного аналізу файлів. Id - унікальний ідентифікатор проведеного аналізу файлу

3.4.9 Dictionary

Dictionary сервіс розроблений для завдань, які пов’язані з інформацією про слова.

Dictionary сервіс використовує базу даних MongoDB так, як не має пов’язаних між собою сутностей. Dictionary сервіс використовує сторонній сервіс WordsApi для отримання тлумачення слів. Dictionary сервіс надсилає HTTP запити у ситуаціях, коли потрібно отримати тлумачення слів.

База даних сервісу складається з колекцій EnglishWordMetadata та WordList. Колекція EnglishWordMetadata (Таблиця 3.16) зберігає інформацію про рівень складності та теми, до яких відносяться слова. Колекція WordList (Таблиця 3.17) містить інформацію про слова, які користувач додав до свого персонального словника.

Таблиця 3.16 - колекція EnglishWordMetadata

Назва поля	Тип	Опис
Id	String	Унікальний ідентифікатор слова
Word	String	Слово
GuideWord	String	Керівне слово
Level	String	Рівень складності
POS	String	Частина мови
Topics	String[]	Теми до яких відноситься слово

Таблиця 3.17 – колекція WordList

Назва поля	Тип	Опис
Id	String	Унікальний ідентифікатор слова
UserId	Guid	Унікальний ідентифікатор користувача
Word	String	Слово
IsLearned	Boolean	Флаг чи було слово вивчене користувачем
Definition	String	Тлумачення слово
PartOfSpeech	String	Частина мови
Synonyms	String[]	Синоніми до слова
Examples	String[]	Приклади використання слова

HTTP маршрути, які надає Dictionary сервіс, наведені у Таблиця 3.18.

Таблиця 3.18 – HTTP маршрути Dictionary сервісу

Метод	URL	Опис
GET	/api/dictionary/audio/word/{word}	Отримати аудіо файл для слова word
GET	/api/dictionary/word/{word}	Отримати тлумачення слова word
POST	/api/dictionary/list	Додавання слів до персонального словника користувача
GET	/api/dictionary/list	Отримати усі слова, які користувач додав до персонального словника
POST	/api/dictionary/list/learned	Помітити слово, як вивчене в персональному словнику користувача
GET	/api/dictionary/word-metadata/query	Отримати інформацію про слова

3.4.10 Course

Course сервіс розроблений для керування курсами користувача.

Мета сервісу надати користувачам навчальні матеріали, які відповідають рівню знань користувача. Сервіс надає завдання на теми, які необхідно знати користувачу, щоб підвищити свій рівень знань з англійської

мови. Також сервіс надає слова, значення яких необхідно знати користувачу для підвищення рівня знань. Сервіс надає інформацію клієнтській частині про прогрес у навчанні, завдяки чому користувач може відслідковувати свій прогрес та бачити на які теми він повинен звернути увагу.

Сервіс використовує базу даних MongoDB так, як сервіс зберігає неструктуровані дані. Для отримання списку завдань та слів, сервіс Course робить HTTP запити у сервіси TaskService та Dictionary відповідно.

База даних сервісу Course складається з трьох колекцій: UserCourse, CompletedTask, LearnedWords. UserCourse (Таблиця 3.19) містить інформацію про користувача, яка потрібна сервісу. CompletedTask (Таблиця 3.20) зберігає інформацію про виконані завдання користувача всередині курсу. LearnedWords (Таблиця 3.21) містить інформацію про слова, які були вивчені при проходженні курсу.

Таблиця 3.19 – колекція UserCourse

Назва поля	Тип	Опис
Id	Guid	Унікальний ідентифікатор користувача
EnglishLevel	String	Рівень знань користувача

Таблиця 3.20 – колекція CompletedTask

Назва поля	Тип	Опис
Id	String	Унікальний ідентифікатор користувача
UserId	Guid	Унікальний ідентифікатор користувача
GrammarPart	String	Тема завдання
ItemsCount	Int32	Кількість елементів у завданні
CorrectAnswers	Int32	Кількість правильних відповідей
TaskType	String	Тип завдання
Date	DateTime	Час виконання завдання

Таблиця 3.21 – колекція LearnedWords

Назва поля	Тип	Опис
Id	String	Унікальний ідентифікатор користувача
UserId	Guid	Унікальний ідентифікатор користувача
Topic	String	Тема слів
Words	String[]	Вивчені слова
CorrectAnswers	Int32	Кількість правильних відповідей
TaskType	String	Тип завдання
Date	DateTime	Час виконання завдання

Course сервіс оброблює UserCreatedEvent, який Identity сервіс надсилає при появі нового користувача в системі через систему для оброки повідомлень Apache Kafka. Course зберігає потрібну йому інформацію про користувача в колекції UserCourse.

HTTP маршрути, які надає Course сервіс, наведені у Таблиця 3.22.

Таблиця 3.22 – HTTP маршрути Course сервісу

Метод	URL	Опис
POST	/api/course/completed/task	Надсилання інформації про виконане завдання
POST	/api/course/completed/words	Надсилання інформації про вивчені слова
GET	/api/course/dictionary/words/{topic}	Отримання списку слів з теми topic, які користувачу потрібно знати для покращення рівня
GET	/api/course/train/task/{grammarPart}	Отримання завдання для теми grammarPart
GET	/api/course/train/dictionary/{topic }	Отримати слова для тренування з теми topic

Висновки до розділу 3

В цьому розділі був здійснений огляд інструментів для розробки системи для вивчення англійської мови.

Був описаний метод для автоматичного створення завдань з використанням обробки природної мови, який дозволяє створювати багато різнопланових завдань.

Була описана клієнтська частина та інструменти, які використовуються при розробці користувацьких інтерфейсів. Для розробки клієнтської частини був використаний фреймворк Angular.

Були описані компоненти серверної частини додатку та як вони взаємодіють між собою. Серверна частина розробленого додатку складається з дев’яти мікросервісів. Для розробки усіх мікросервісів, крім одного, була використана мова програмування C# та фреймворк ASP.NET Core. Для розробки мікросервісу, який використовував інструменти оброки природної мови, було використано мову програмування Python та фреймворк Flask. Були оглянуті інструменти, які використовувались при розробці мікросервісів такі, як spaCy, Apache Kafka та Azure Speech Service.

РОЗДІЛ 4
ОГЛЯД РОЗРОБЛЕНОЇ СИСТЕМИ ДЛЯ ДИСТАНЦІЙНОГО НАВЧАННЯ АНГЛІЙСЬКОЇ МОВИ

4.1 Огляд розробленого методу для автоматичного створення завдань

Розроблений метод для автоматичного створення завдань дозволяє використовувати у завданнях речення з великих текстових файлів таких, як сторінки з Вікіпедії та фрагменти художніх творів. Використання різних джерел інформації дозволяє створювати завдання в яких буде різний контекст, що дозволяє мати в системі цікаві завдання і це позитивно впливає на процес навчання користувачів. Наприклад, можна створювати завдання в яких використовується контекст космосу, фільмів та серіалів, популярних книг – це може приваблювати користувачів до виконання більшої кількості завдань.

Розроблений метод використовує обробку природної мови для лексичного аналізу, лематизації, семантичного аналізу. Дані інструменти дозволяють шукати у великих файлах речення, які відповідають певній граматичній темі та використовувати їх, як основу для створення завдань. Проаналізовані речення можуть використовуватись для завдань, де потрібно ввести прослухане речення. Також з проаналізованих речень можна утворювати більш складні типи завдань. Проаналізовані речення містять інформацію про те, до якої частини мови відноситься кожне слово, які зв’язки з іншими словами у реченні має кожне слова та лему кожного слова. Використовуючи ці дані, сервіс, який керує завданнями може автоматично створювати завдання різних типів. Наприклад, для створення завдання с відкриттям дужок для теми «Present Simple», системі необхідно знати лише дієслово присудок у реченні та його лему. Дієслово присудок система може знайти легко так, як це буде перше дієслово в реченні. У завдані, в дужках буде знаходитись лема для цього слова.

Рівень складності проаналізованого речення залежить від того, які слова будуть в ньому використовуватись. Рівень складності речення буде

відповідати рівню складності слова, яке має найвищий рівень складності серед усіх слів у реченні.

Переваги розробленого методу для автоматичного створення завдань порівняно з методом розробленим сервісом «Duolingo»:

- Завдання можуть бути створенні для конкретної граматичної теми, що дозволяє користувачам отримати велику кількість завдань для покращення їх рівня знань. Сервіс «Duolingo» автоматично створює лише загальні завдання для перевірки рівня знань.
- Дозволяє створювати різні типи завдань.
- Велика кількість завдань створених розробленим методом використовується для навчання користувачів. Розроблений метод сервісом «Duolingo» використовується лише для оцінювання рівня знань.

Недоліки розробленого методу:

- Оцінювання складності речення лише за складністю слів у реченні. Duolingo, крім цього, оцінює складність речень за кількістю слів у реченні та складністю зв’язків між словами у реченні.
- Недосконалий спосіб для виявлення речень з недопустимим змістом, які не можна використовувати для завдань.

4.2 Огляд інтерфейсу користувача розробленого додатку

4.2.1 Реєстрація

Половина функціональності додатку доступна лише для користувачів, які зареєстровані. Користувач повинен перейти на сторінку для реєстрації нових користувачів (Рис. 4.1) і ввести інформацію про себе: ім’я, прізвище, адресу електронної пошти, свій рівень знань англійської мови та додати свої вподобання до завдань. Також користувач повинен створити пароль для входу в систему.

Рис. 4.1. Сторінка для реєстрації нових користувачів

4.2.2 Аутентифікація

Сторінка для аутентифікації користувачів (Рис. 4.2) створена для того, щоб користувач в будь-який момент часу після реєстрації міг пройти процес аутентифікації та отримати повний доступ до додатку. Для того, щоб пройти процес аутентифікації в системі користувач повинен ввести свій пароль та свою адресу електронної пошти.

Рис. 4.2. Сторінка для аутентифікації користувачів

4.2.3 Керування системою

Для керування системою розроблений окремий Angular додаток, доступ до якого мають лише адміністратори сайту.

Сторінка зі списком завантажених файлів (Рис. 4.3) у системі дозволяє завантажувати нові файли (Рис. 4.4) та запускати аналіз файлів для пошуку речень (Рис. 4.5), які можна використовувати для створення завдань.

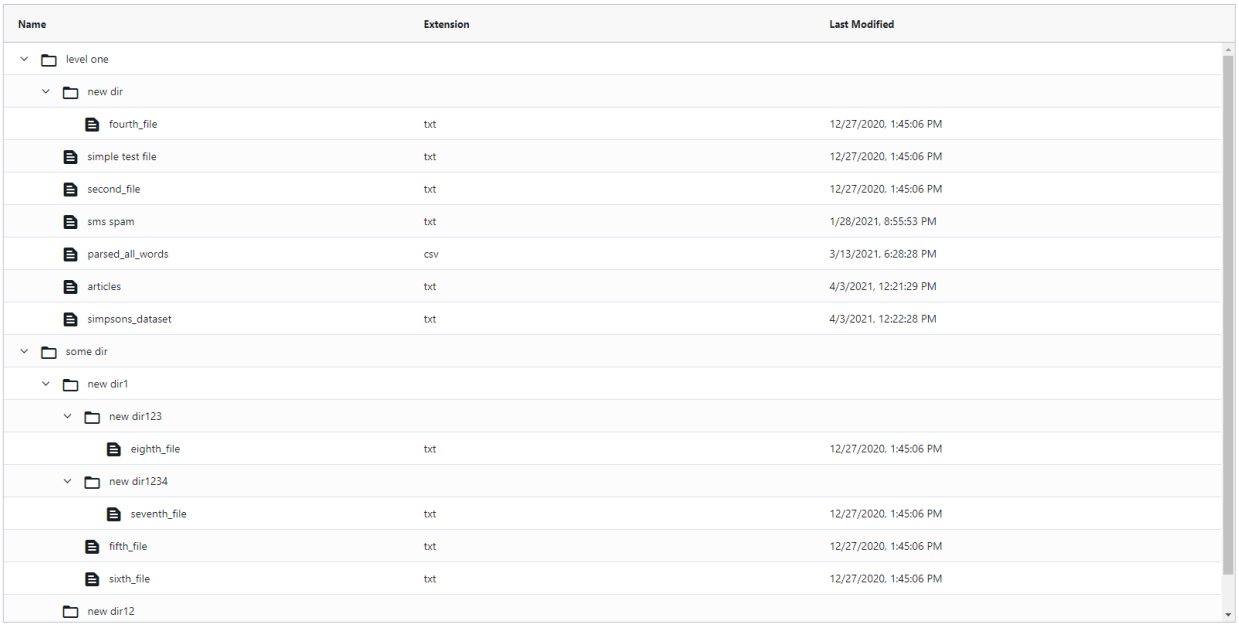


Рис. 4.3. Список завантажених файлів

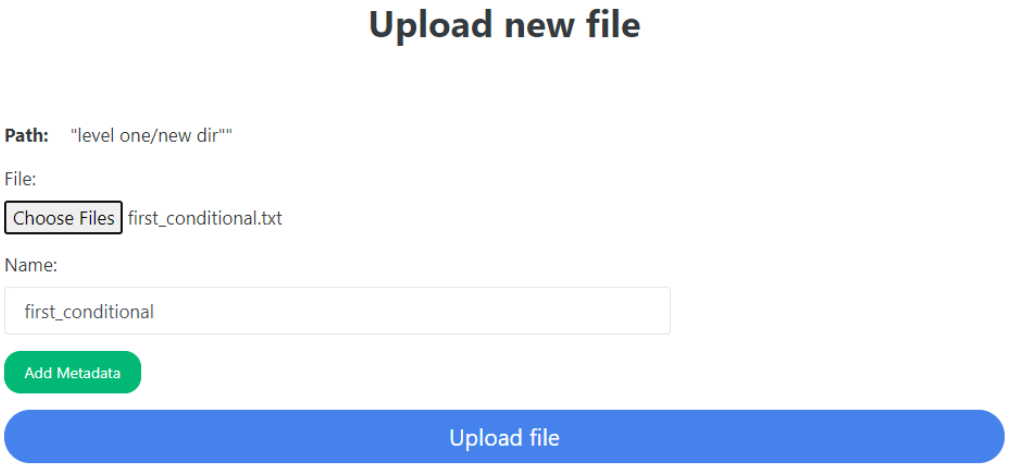


Рис. 4.4. Форма для завантаження нових файлів

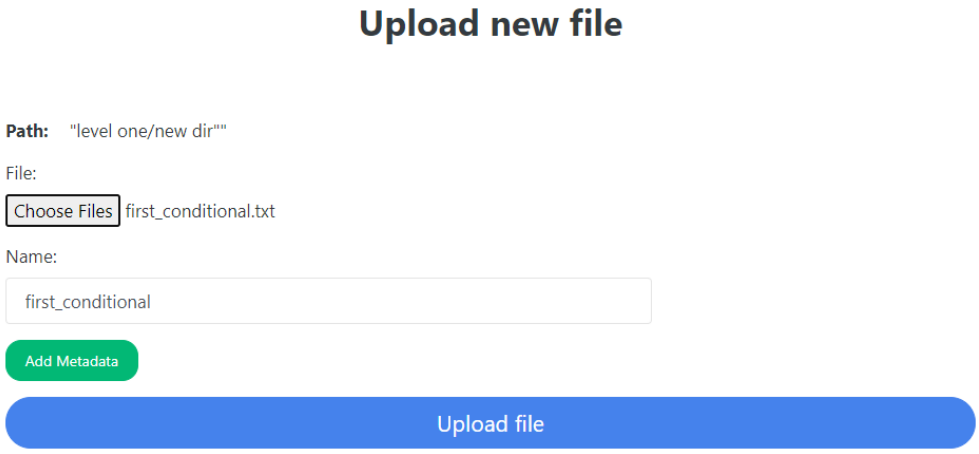


Рис. 4.5. Форма для запуску аналізу файлу

Доступні сторінки для перегляду списку виконаних аналізів файлів та з їх результатами.

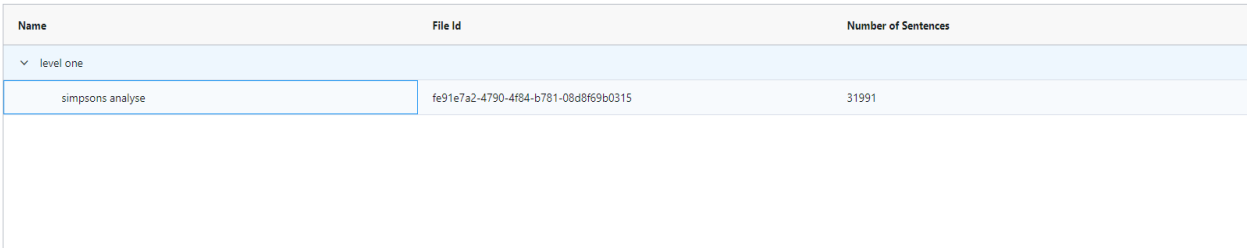


Рис. 4.6. Список виконаних аналізів файлів

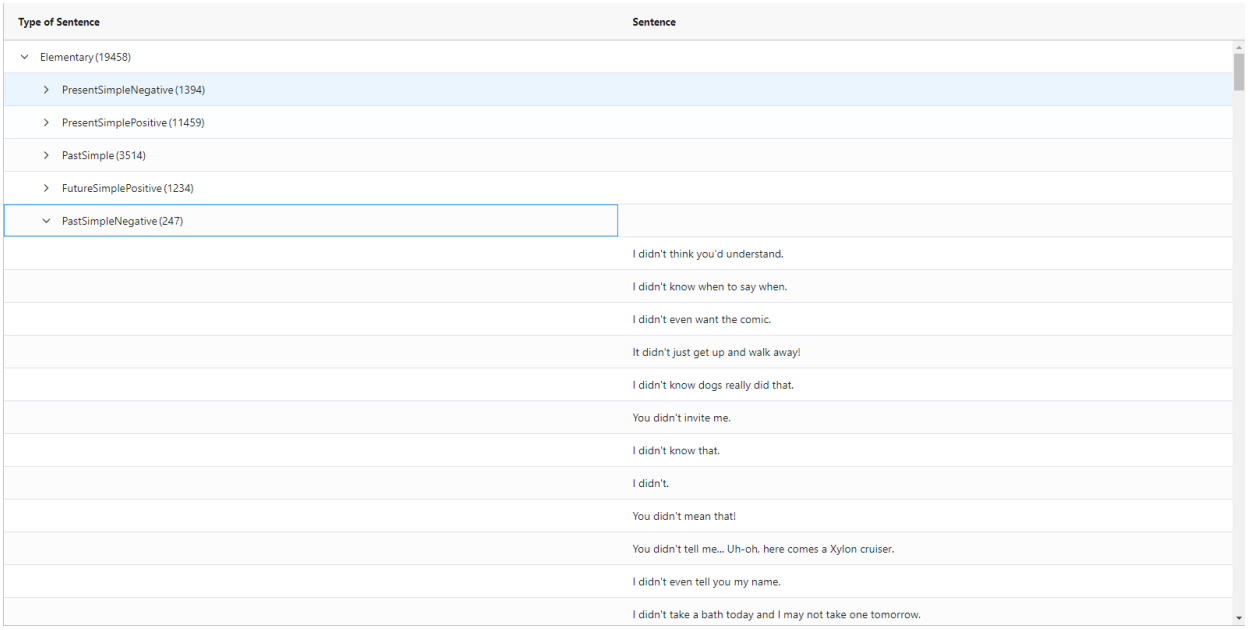


Рис. 4.7. Результати аналізу файлу

Форма для генерації завдань (Рис. 4.8) створена для початку процесу створення елементів завдань на основі проаналізованого файлу. Для початку процесу створення елементів завдань потрібно вибрати ім’я генерації елементів завдань, яке буде відображатись у відповідному списку, вибрати тему завдання та тип завдання, для якого будуть створені завдання. Результатом генерації елементів завдань, будуть створені елементи завдань вибраного типу і вибраної теми на основі речень з проаналізованого файлу, який був вибраний на сторінці списку виконаних аналізів файлів.

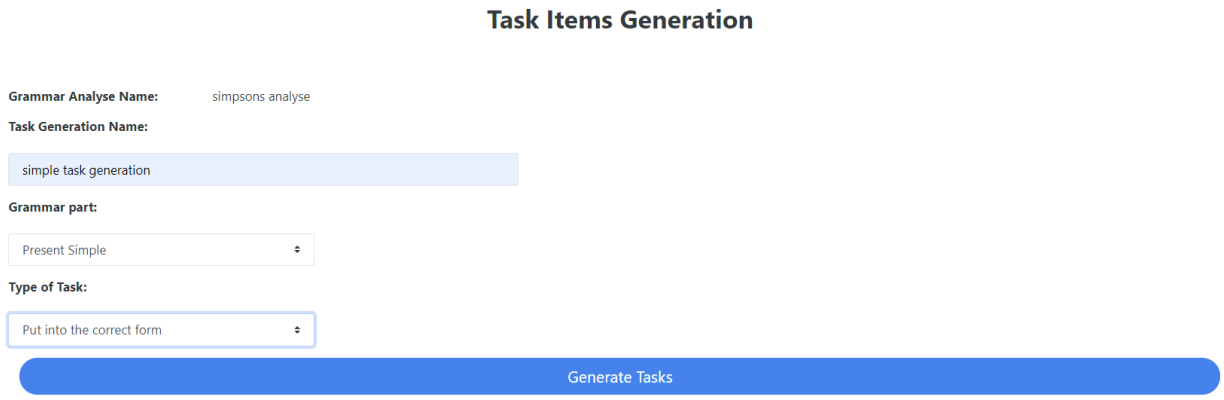


Рис. 4.8. Форма для генерації завдань

Елементи завдань, які були створені системою, доступні на сторінці зі списком елементів завдань (Рис. 4.9). Елементи завдань можуть бути використанні для створення повноцінних завдань. Форма для створення

завдань (Рис. 4.10) надає можливість створити нові завдання, вибравши елементи завдання, які будуть в ньому використовуватись, або для завдання будуть вибрані елементи випадковим чином за темою завдання, типом завдання та складністю завдання.

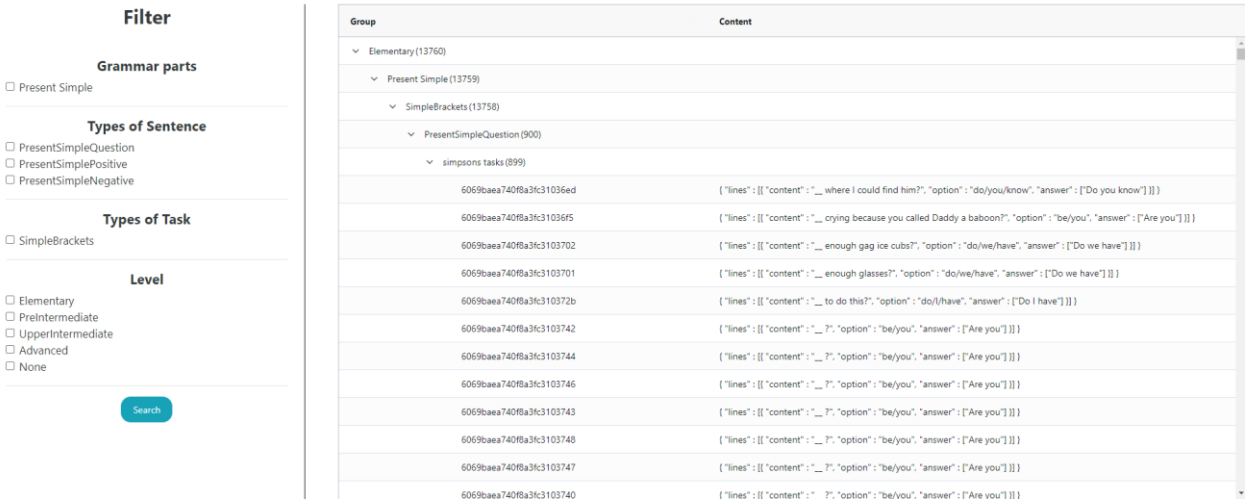


Рис. 4.9. Список елементів завдання

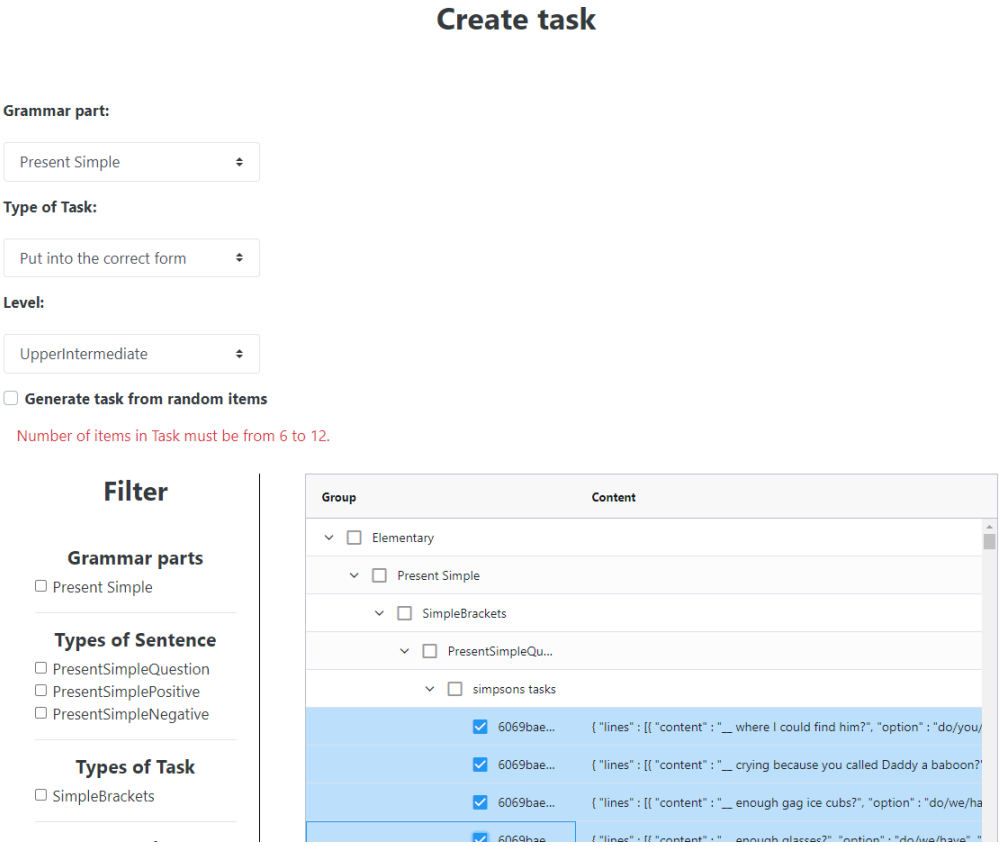


Рис. 4.10 Форма для створення завдань

4.2.4 Завдання

Система для розвитку навичок писання надає завдання на різні теми та з різними рівнями складності. На сторінці зі списком завдань доступі усі завдання. Їх можна відфільтрувати за бажанням користувача. Після того, як користувач вибере завдання, система перенаправить користувача на сторінку з самим завданням.

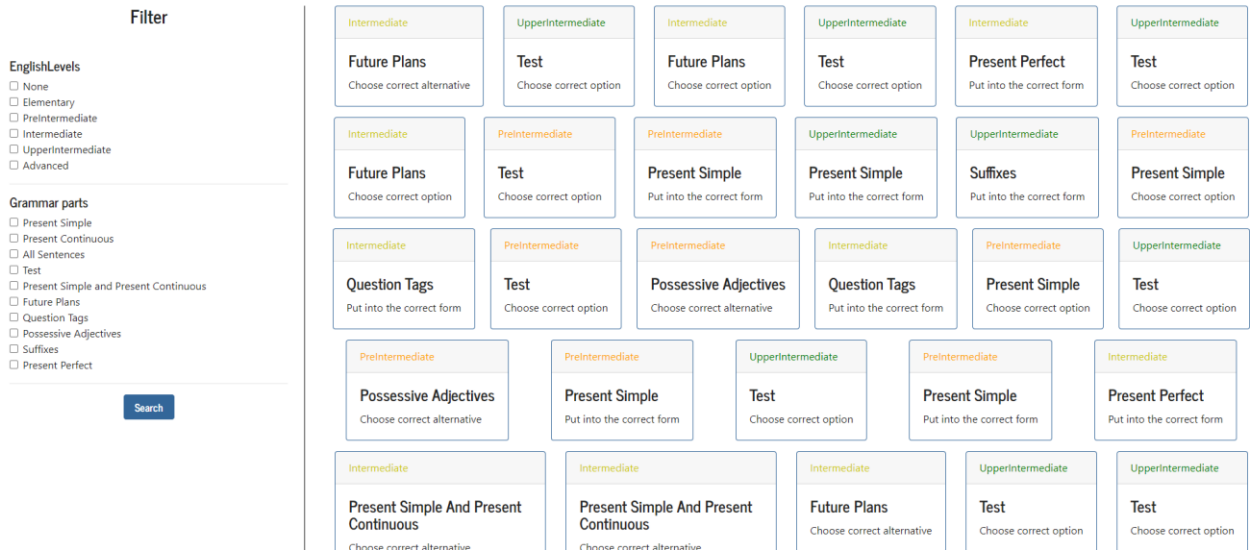


Рис. 4.11. Список завдань

Доступні чотири види завдань: звичайні тести з вибором правильної відповіді (Рис. 4.13), завдання з розкриттям дужок (Рис. 4.14), завдання з вибором однієї з двох відповідей (Рис. 4.14) та завдання на введення користувачем почутого речення (Рис. 4.15).

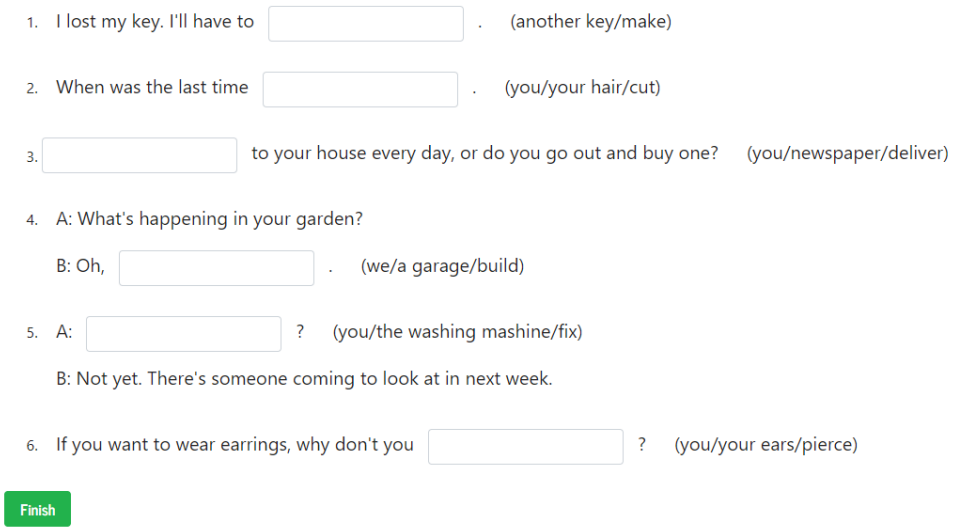


Рис. 4.12. Завдання з розкриттям дужок

1. Everybody needs _____ .

○ friends ○ the friends

2. Jane doesn't go to _____ very often.

○ parties ○ the parties

3. I wentshopping this morning. _____ were very busy.

○ Shops ○ The shops

4. Where's _____ ? It's in the fridge.

○ milk ○ the milk

5. I don't like _____ . I never drink it.

○ milk ○ the milk

6. Do you do any sports? Yes, I play _____ .

○ basketball ○ the basketball

7. An architect is a person who designs _____.

○ buldings ○ the buildings

8. We went for swim in the river. _____ was very cold.

○ Water ○ The water

9. I don't like swimming in _____.

○ cold water ○ the cold water

10. Excuse me? can you pass _____, please?

○ salt ○ the salt

Finish

Рис. 4.13. Тестове завдання

1. A: I 've never been to Hollywood.
   B: Haven't you? I went there last year.
2. A: How many films did you act in so far?
   B: I 've acted in seven films up to now.
3. A: He's only twenty-two years old? but he has travelled all around the world.
   B: Which countries did he visit ?.
   Incorrect answer
4. A: She [ ] four Oscars for her performances.
   B: That's right. She [ ] an award for Best Actress last month.
5. A: They [ ] in California for twenty years.
   B: When [ ] to Texas?
6. A: [ ] here long?
   B: Yes. I [ ] over an hour ago.
7. A: [ ] for forty years now.
   B: Do you remember the day we [ ] ?

Рис. 4.14. Завдання з вибором однієї з двох правильних відповідей.

1. 0:00 / 0:03
2. 0:00 / 0:02
3. 0:00 / 0:02
4. 0:00 / 0:02

Рис. 4.15. Завдання на введення почутого речення користувачем

4.2.5 Вивчення слів

Для розширення словникового запасу, доступний розділ для вивчення слів. На сторінці для пошуку слів (Рис. 4.16) можна знайти тлумачення слова, яке цікавить користувача. Також користувач може додавати слова до свого персонального словника (Рис. 4.17), а згодом вивчати їх.

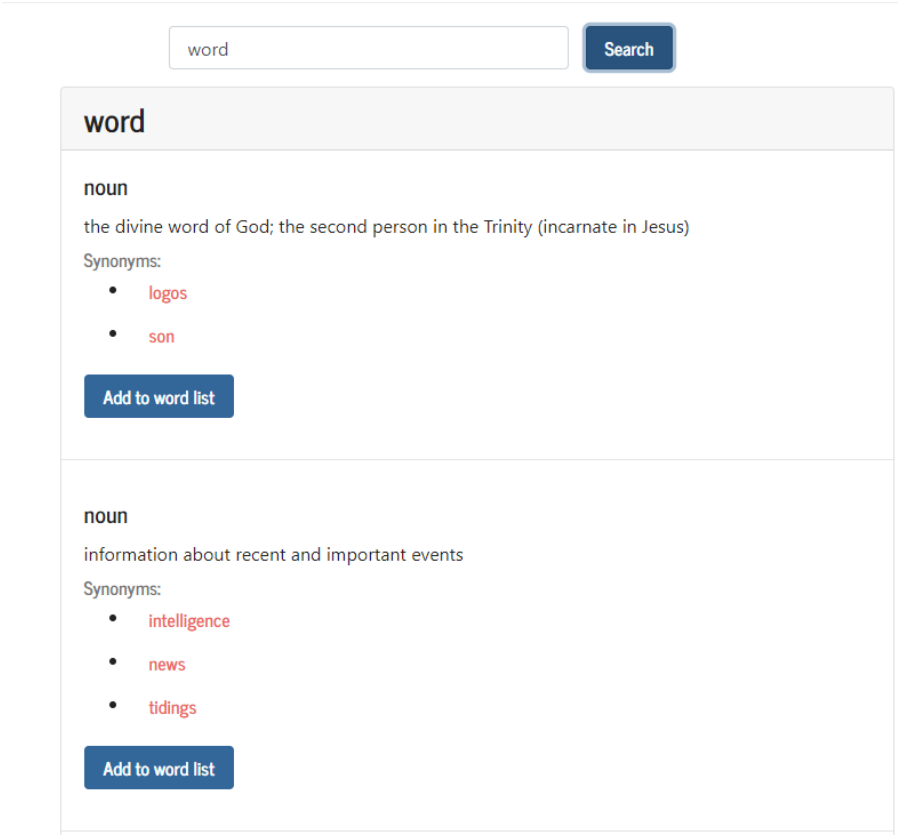


Рис. 4.16. Сторінка для пошуку слів

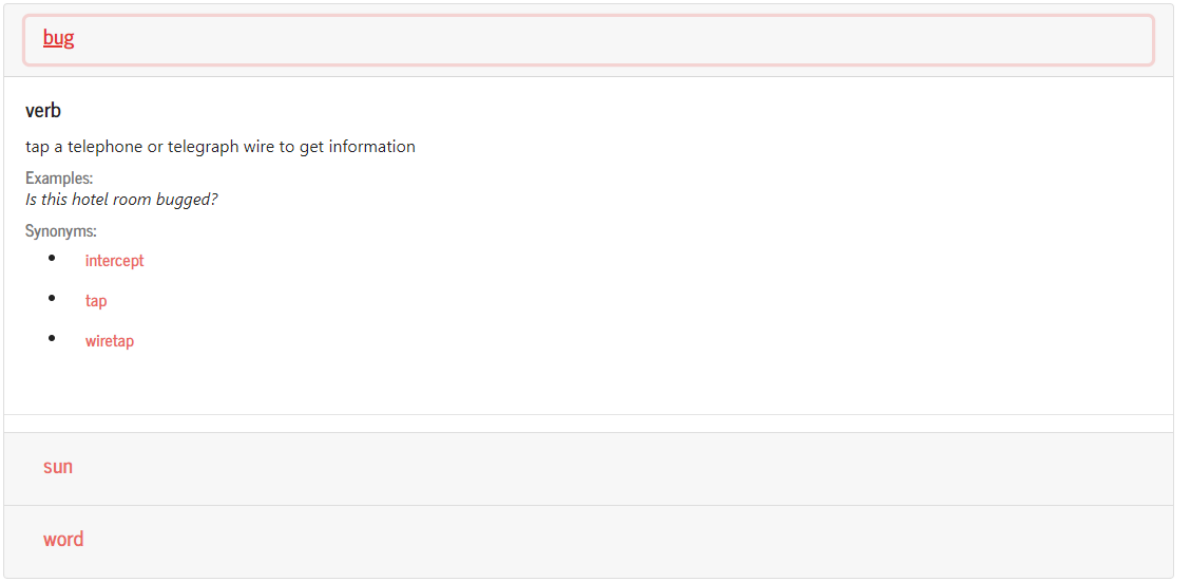


Рис. 4.17. Сторінка з персональним словником користувача

4.2.6 Відео

Для розвитку навичок слухання, система надає відеоматеріали. Користувач може знайти відео на тему, яка його найбільше цікавить та для його рівня складності. Сторінка зі списком відео (Рис. 4.18) відображає доступні для перегляду відео, який можна фільтрувати. Після вибору відео користувачем, система перенаправить на сторінку з відео (Рис. 4.19).

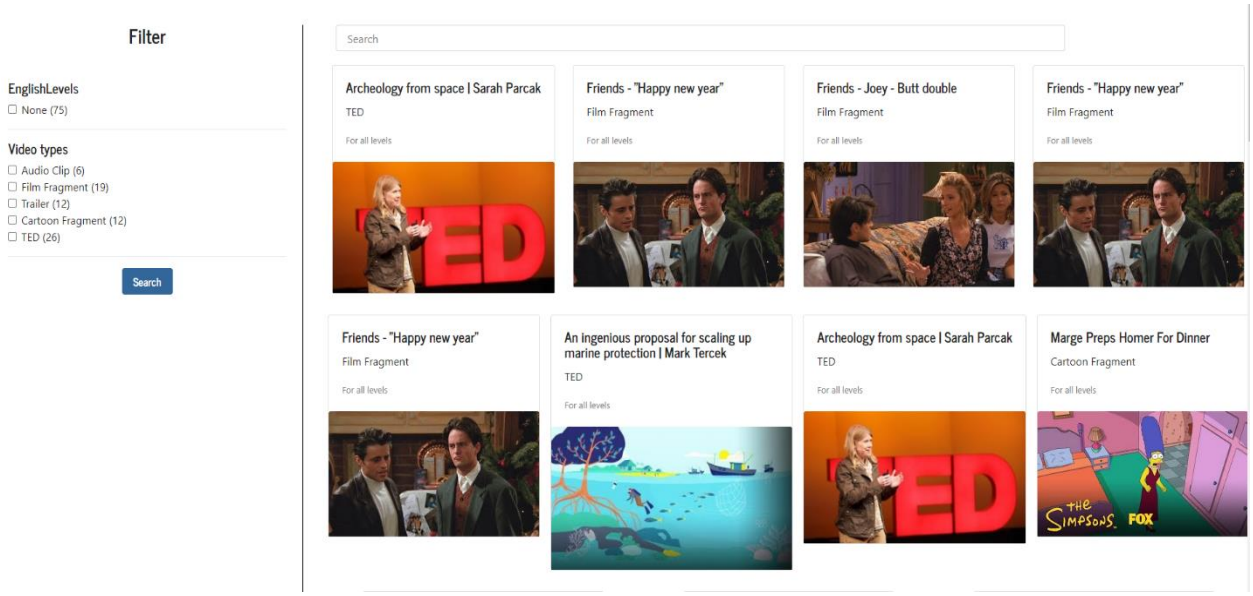


Рис. 4.18. Список відео

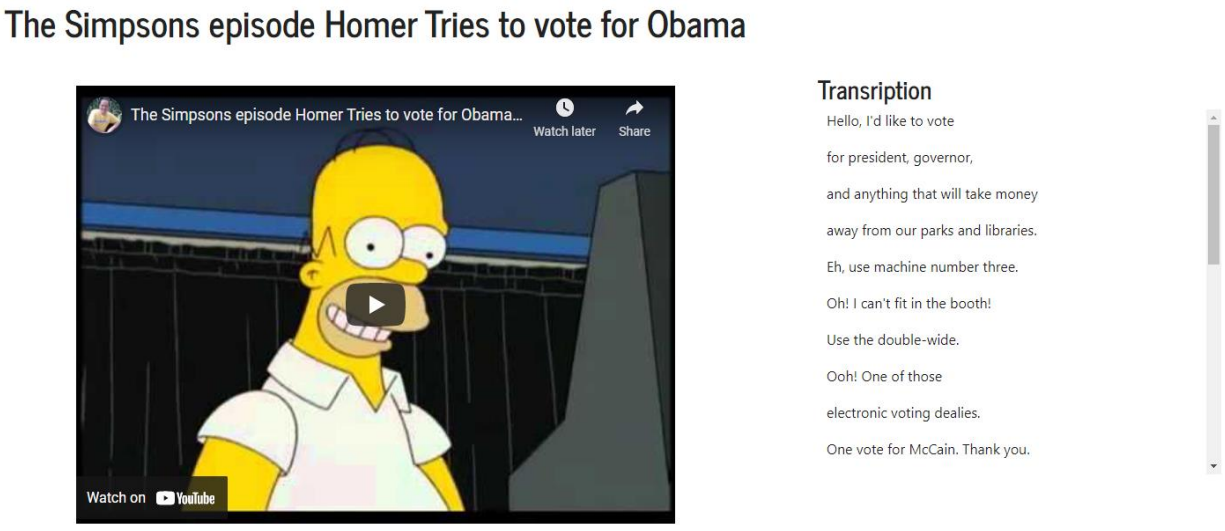


Рис. 4.19. Сторінка з відео

4.2.7 Тексти

Для розвитку навичок читання, система надає тексти для читання. Користувач може знайти тексти на тему, яка його найбільше цікавить та для його рівня складності. Сторінка зі списком текстів (Рис. 4.20) відображає доступні для читання тексти, який можна фільтрувати. Після вибору тексту користувачем, система перенаправить на сторінку з текстом (Рис. 4.21).

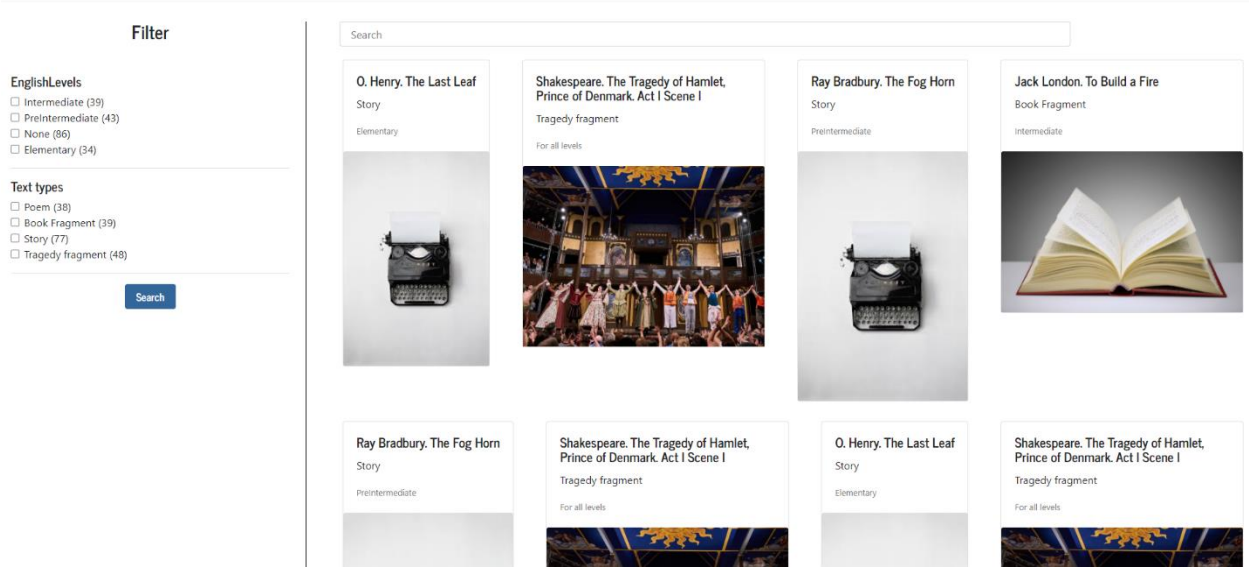
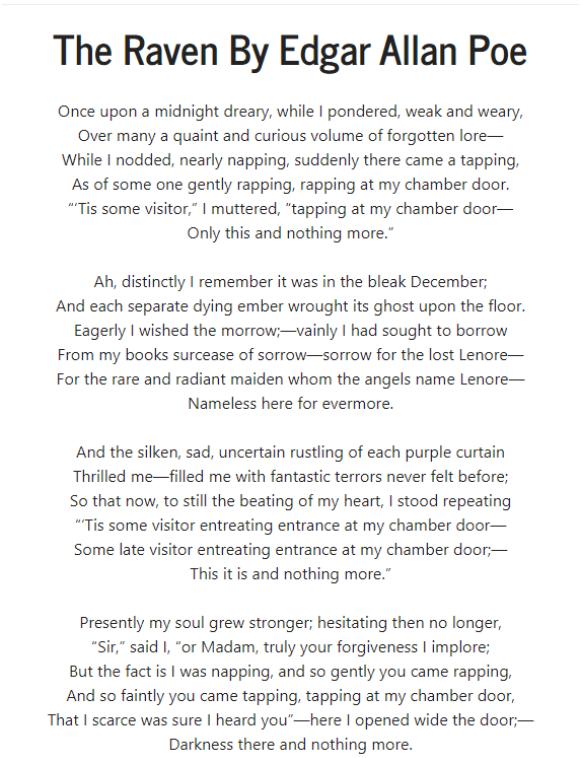


Рис. 4.20. Список текстів



The Raven By Edgar Allan Poe

Once upon a midnight dreary, while I pondered, weak and weary,
Over many a quaint and curious volume of forgotten lore—
While I nodded, nearly napping, suddenly there came a tapping,
As of some one gently rapping, rapping at my chamber door.
"'Tis some visitor," I muttered, "tapping at my chamber door—
Only this and nothing more."

Ah, distinctly I remember it was in the bleak December;
And each separate dying ember wrought its ghost upon the floor.
Eagerly I wished the morrow;—vainly I had sought to borrow
From my books surcease of sorrow—sorrow for the lost Lenore—
For the rare and radiant maiden whom the angels name Lenore—
Nameless here for evermore.

And the silken, sad, uncertain rustling of each purple curtain
Thrilled me—filled me with fantastic terrors never felt before;
So that now, to still the beating of my heart, I stood repeating
"'Tis some visitor entreating entrance at my chamber door—
Some late visitor entreating entrance at my chamber door;—
This it is and nothing more."

Presently my soul grew stronger; hesitating then no longer,
"Sir," said I, "or Madam, truly your forgiveness I implore;
But the fact is I was napping, and so gently you came rapping,
And so faintly you came tapping, tapping at my chamber door,
That I scarce was sure I heard you"—here I opened wide the door;—
Darkness there and nothing more.

Рис. 4.21. Сторінка з текстом

4.2.8 Курси

Для того, щоб користувачу було зручніше покращити свій загальний рівень, створений розділ курсів. Даний розділ доступний лише для авторизованих користувачів. Для кожного користувача формується список завдань на основі його рівня знань. На сторінці зі списком навчального матеріалу для курсу (Рис. 4.22) доступні завдання та слова на теми, які обов'язково треба знати для покращення рівня знань. Для кожної теми міститься інформація про прогрес користувача та скільки, ще завдань він повинен виконати для вивчення теми. На сторінці з вправами користувач може вибрати завдання, яке він хоче зробити. Також користувач може перевірити свої знання слів (Рис. 4.23).

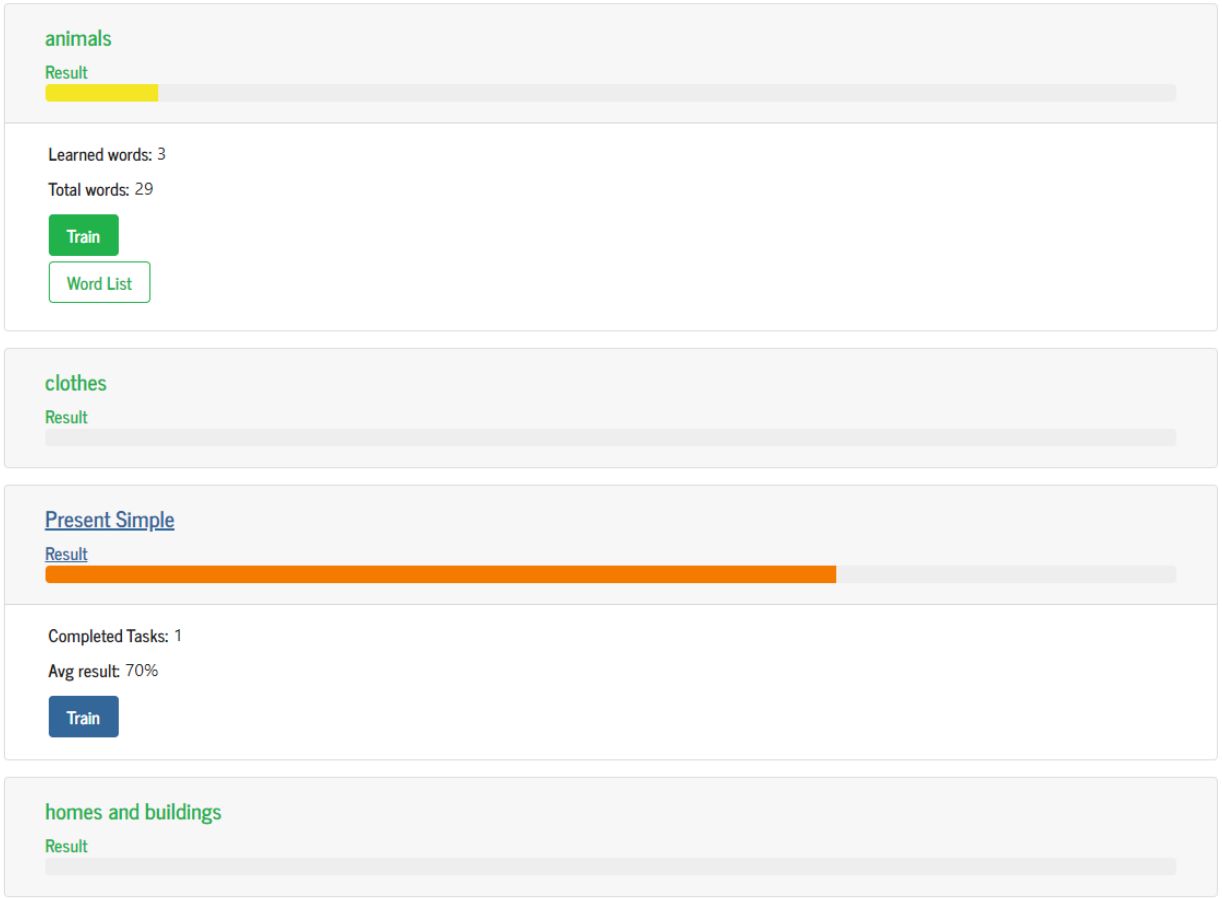


Рис. 4.22. Сторінка зі списком навчального матеріалу для курсу

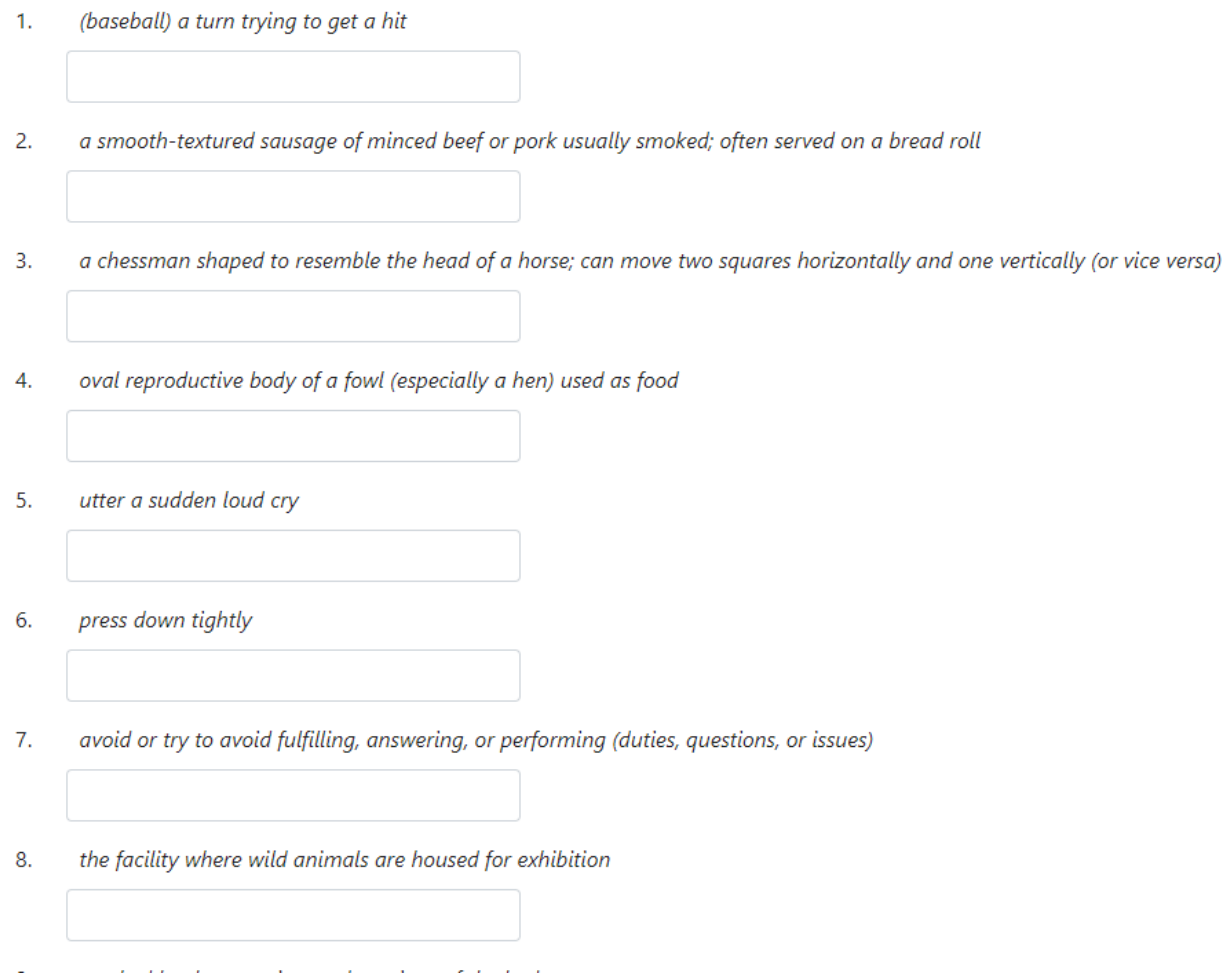


Рис. 4.23. Сторінка з завданням для перевірки знання слів

4.2.9 Статистика користувачів

Розділ статистки розроблений для того, щоб користувачі могли відслідковувати, як активно вони вивчають англійську мову та який в них прогрес у навчанні. Цей розділ доступний лише для авторизованих користувачів. У розділі статистики можна побачити, наскільки успішно користувач виконує завдання (Рис. 4.24). Також можна побачити інформацію про активність за останні 30 днів (Рис. 4.25) та список усіх виконаних завдань, переглянутих відео та прочитаних текстів (Рис. 4.26).

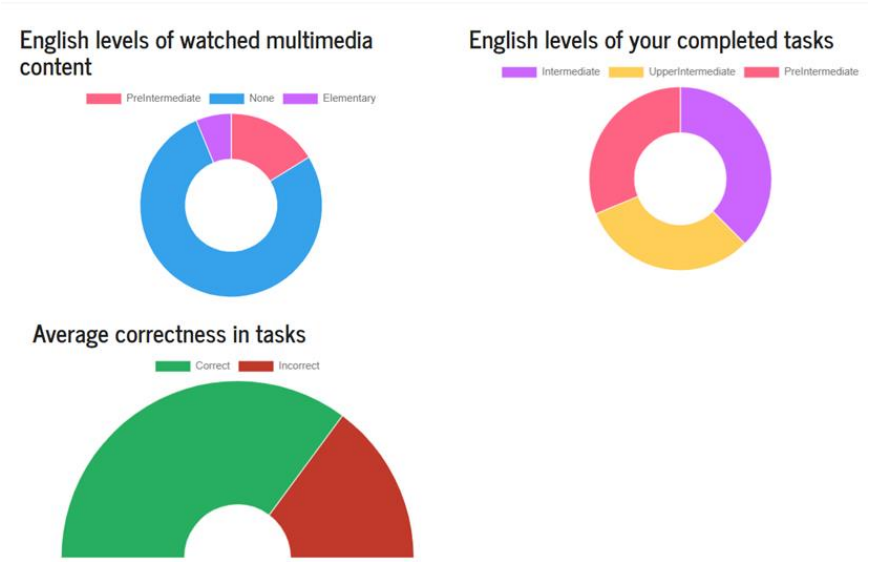


Рис. 4.24. Статистичні дані

Your favourites types of videos:
1. Cartoon Fragment
2. Film Fragment
3. Trailer

Your favourites types of texts:
1. Story
2. Tragedy fragment
3. Poem

Your last month activity

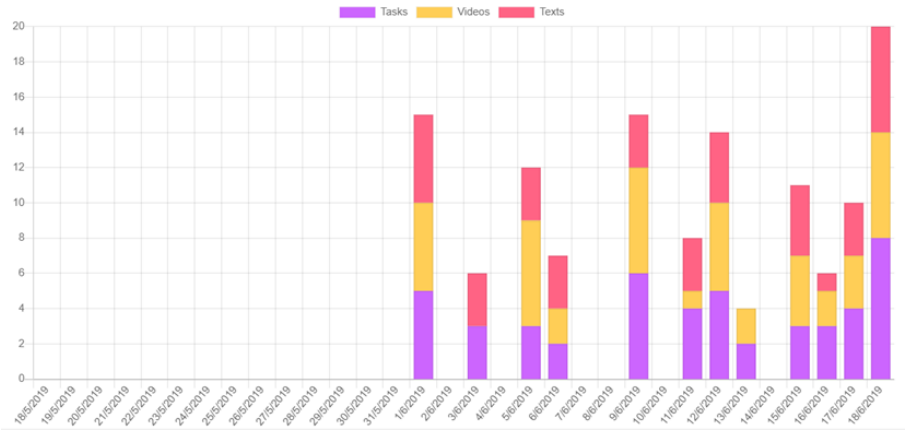


Рис. 4.25. Статистичні дані

Your completed tasks, and watched multimedia

29/5/2019

Text	PreIntermediate	Ray Bradbury. The Fog Horn
Video	None	Friends - Joey - Butt double
Video	None	The Simpsons episode Homer Tries to vote for Obama
Text	None	The Raven By Edgar Allan Poe
Task	Intermediate	Present Simple and Present Continuous Correct: 0. Incorrect: 0

Рис. 4.26. Статистичні дані

Висновки до розділу 4

В даному розділі був здійснений огляд розробленого додатку.

Був здійснений аналіз розробленого методу для автоматичного створення завдань. Були описані переваги та недоліки розробленого методу в порівняні з існуючими аналогами.

Розроблений сервіс для дистанційного навчання англійської мови забезпечує великий вибір завдань для покращення навичок, які необхідні для покращення знань з англійської мови. Також система курсів дозволяє персоналізувати навчання користувачів, що дозволяє покращити ефективність навчання.

Був описаний користувацький інтерфейс розробленої системи для дистанційного навчання англійської мови.

ВИСНОВКИ

У ході виконання магістерської дисертації було розроблено веб-додаток для вивчення англійської мови.

Під час виконання роботи було досліджено системи для дистанційного навчання англійської мови. Досліджено існуючі системи для дистанційного навчання англійської мови. Проаналізовані переваги та недоліки існуючих систем для дистанційного навчання англійської мови. За результатами аналізу проведеного дослідження систем для дистанційного навчання англійської мови, було зроблено висновок, що на сьогоднішній день є актуальним створення системи для дистанційного навчання англійської мови.

Також був здійснений огляд способів створення системи для дистанційного навчання англійської мови. Було розглянуті різні архітектурні підходи до розробки веб-додатків, які найбільше підходять для розробки системи для дистанційного навчання англійської мови. Було здійснено огляд технологій, які необхідні для розробки додатку.

Була описана реалізація системи для дистанційного навчання англійської мови. Для розробки клієнтської частини був використаний фреймворк Angular. Серверна частина розробленого додатку складається з дев’яти мікросервісів. Для розробки усіх мікросервісів, крім одного, була використана мова програмування C# та фреймворк ASP.NET Core. Для розробки мікросервісу, який використовував інструменти оброки природної мови, було використано мову програмування Python та фреймворк Flask.

Було розроблено метод для автоматичного створення завдань для вивчення англійської мови, що дозволяє надавати користувачам велику кількість різноманітних вправ на різні теми та з різними рівнями складності. Для розробки методу для автоматичного створення завдань, було використано інструменти обробки природної мови, для пошуку речень в текстових матеріалів, які можуть використовуватись для створення завдань. Був здійснений аналіз переваг та недоліків розробленого методу.

В результаті виконаної роботи, була розроблена система для дистанційного навчання англійської мови, що використовує сучасні технології для персоналізації навчання. Розроблена система надає користувачу можливість отримати необхідні знання для покращення рівня знань з англійської мови. Розроблена система надає навчальний матеріал для покращення користувачами навичок читання, писання, слухання та для розширення словникового запасу. Також розділ курсів дозволяє персоналізувати навчання кожного користувача, що підвищує ефективність навчання. Завдяки використанню розробленого методу для автоматичного створення завдань, система містить велику кількість різноманітних вправ на різні теми та з різними рівнями складності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Defining Distance Learning [Електронний ресурс] // Режим доступу: https://www.viewsonic.com/library/education/defining-distance-learning/
2. Why And How Self-Learning Is Important [Електронний ресурс] // Режим доступу: https://www.brightermonday.co.ke/blog/self-learning/
3. Why is Learning English is so Important? [Електронний ресурс] // Режим доступу: https://www.elc-schools.com/blog/4-reasons-why-learning-english-is-so-important/
4. USA Learns [Електронний ресурс] // Режим доступу: https://www.usalearns.org/
5. EnglishClass101 [Електронний ресурс] // Режим доступу: https://www.englishclass101.com/
6. British Council [Електронний ресурс] // Режим доступу: https://learnenglish.britishcouncil.org/
7. BBC Learning English [Електронний ресурс] // Режим доступу: https://www.bbc.co.uk/learningenglish/
8. Duolingo [Електронний ресурс] // Режим доступу: https://www.duolingo.com/
9. B.Settles, G.T. LaFlair, M. Hagiwara. Machine Learning–Driven Language Assessment [Електронний ресурс] // Режим доступу: https://research.duolingo.com/papers/settles.tacl20.pdf
10. English Vocabulary Profile Online - British English [Електронний ресурс] // Режим доступу: https://www.englishprofile.org/wordlists/evp
11. Principles of Modern Application Development [Електронний ресурс] // Режим доступу: https://www.nginx.com/blog/principles-of-modern-application-development/
12. What Is a Web Application? How It Works, Benefits and Examples [Електронний ресурс] // Режим доступу: https://www.indeed.com/career-advice/career-development/what-is-web-application

13. 5 Key Benefits of Web Applications for Business [Електронний ресурс] // Режим доступу: https://evergreencomputing.com/blog/5-key-benefits-of-web-applications-for-business/
14. What is a Web Application? [Електронний ресурс] // Режим доступу: https://blog.stackpath.com/web-application/
15. Native Apps, Web Apps or Hybrid Apps? What's the Difference? [Електронний ресурс] // Режим доступу: https://www.mobiloud.com/blog/native-web-or-hybrid-apps
16. What is an application architecture? [Електронний ресурс] // Режим доступу: https://www.redhat.com/en/topics/cloud-native-apps/what-is-an-application-architecture#:~:text=An%20application%20architecture%20describes%20the,you%20to%20build%20an%20application
17. Web App Architecture: Why Is It Important? [Електронний ресурс] // Режим доступу: https://light-it.net/blog/web-app-architecture-why-is-it-important/
18. Single-Tier Vs. Multi-Tier Architecture [Електронний ресурс] // Режим доступу: https://docs.bitnami.com/google-templates/singletier-vs-multitier/
19. Pattern: Monolithic Architecture [Електронний ресурс] // Режим доступу: https://microservices.io/patterns/monolithic.html
20. Microservices [Електронний ресурс] // Режим доступу: https://martinfowler.com/articles/microservices.html
21. Pattern: Event-driven architecture [Електронний ресурс] // Режим доступу: https://microservices.io/patterns/data/event-driven-architecture.html
22. Single-page application vs. multiple-page application [Електронний ресурс] // Режим доступу: https://medium.com/@NeotericEU/single-page-application-vs-multiple-page-application-2591588efe58

23. What Is a Relational Database? [Електронний ресурс] // Режим доступу: https://www.oracle.com/database/what-is-a-relational-database/
24. What is NoSQL? [Електронний ресурс] // Режим доступу: https://aws.amazon.com/nosql/
25. Your Guide to Natural Language Processing (NLP) [Електронний ресурс] // Режим доступу: https://towardsdatascience.com/your-guide-to-natural-language-processing-nlp-48ea2511f6e1
26. Introduction to Angular concepts [Електронний ресурс] // Режим доступу: https://angular.io/guide/architecture

ДОДАТОК А

СИСТЕМА ДЛЯ ДИСТАНЦІЙНОГО НАВЧАННЯ АНГЛІЙСЬКОЇ МОВИ

ЛІСТИНГ ДОДАТКУ

Аркушів 19

Київ – 2021

```
using System;
using System.IO;
using System.Threading.Tasks;
using EnglishLearning.FileManager.Application.Abstract;
using EnglishLearning.FileManager.Application.Models;
using EnglishLearning.FileManager.Web.ViewModels;
using EnglishLearning.Utilities.Identity.Abstractions;
using Microsoft.AspNetCore.Mvc;
using static EnglishLearning.FileManager.Web.Constants.ContentTypeConstants;
using static EnglishLearning.FileManager.Web.Infrastructure.WebMapper;

namespace EnglishLearning.FileManager.Web.Controllers
{
    [Route("/api/file-manager/file")]
    [ApiController]
    public class FileController : Controller
    {
        private readonly IFileService _fileService;
        private readonly IJwtInfoProvider _jwtInfoProvider;
        private readonly IFileUpdateService _fileUpdateService;

        public FileController(
            IFileService fileService,
            IJwtInfoProvider jwtInfoProvider,
            IFileUpdateService fileUpdateService)
        {
            _fileService = fileService;
            _jwtInfoProvider = jwtInfoProvider;
            _fileUpdateService = fileUpdateService;
        }

        [EnglishLearningAuthorize(AuthorizeRole.Admin)]
        [HttpPost]
        public async Task<ActionResult> CreateFile([FromForm] FileCreateViewModel file)
        {
            var fileCreateModel = MapViewModelToApplicationModel(file, _jwtInfoProvider.U
serId);
            await using var memoryStream = new MemoryStream();
            await file.UploadedFile.CopyToAsync(memoryStream);

            await _fileService.CreateFileAsync(memoryStream, fileCreateModel);

            return Ok();
        }

        [EnglishLearningAuthorize(AuthorizeRole.Admin)]
        [HttpGet("{id}")]
        public async Task<ActionResult> GetFile(Guid id)
        {
            var fileInfo = await _fileService.GetFileDetailedModelAsync(id);
```

```
            var contentType = FileExtensionContentTypeMap[fileInfo.Extension];
            var fileName = $"{fileInfo.Name}.{fileInfo.Extension}";
            var fileStream = await _fileService.GetFileContentAsync(id);

            return new FileStreamResult(fileStream, contentType)
            {
                FileDownloadName = fileName,
            };
        }

        [EnglishLearningAuthorize(AuthorizeRole.Admin)]
        [HttpGet("{id}/details")]
        public async Task<ActionResult> GetFileInfo(Guid id)
        {
            var fileDetails = await _fileService.GetFileDetailedModelAsync(id);

            return Ok(fileDetails);
        }

        [EnglishLearningAuthorize(AuthorizeRole.Admin)]
        [HttpGet("~/api/file-manager/folder/file/info")]
        public async Task<ActionResult> GetFilesInFolderInfo([FromQuery] int? folderId)
        {
            var fileInfos = await _fileService.GetAllFromFolderAsync(folderId);

            return Ok(fileInfos);
        }

        [EnglishLearningAuthorize(AuthorizeRole.Admin)]
        [HttpPost("remove-text")]
        public async Task<ActionResult> RemoveTextFromFile([FromBody] RemoveTextFromFileM
odel removeTextFromFileModel)
        {
            await _fileUpdateService.RemoveTextFromFileAsync(removeTextFromFileModel);

            return Ok();
        }
    }
}

using System;
using System.Threading.Tasks;
using EnglishLearning.FileManager.Application.Abstract;
using EnglishLearning.Utilities.Identity.Abstractions;
using Microsoft.AspNetCore.Mvc;

namespace EnglishLearning.FileManager.Web.Controllers
{
    [Route("/api/file-manager/folder")]
    [ApiController]
```

```
    public class FolderController : Controller
    {
        private readonly IFolderService _folderService;

        public FolderController(IFolderService folderService)
        {
            _folderService = folderService;
        }

        [EnglishLearningAuthorize(AuthorizeRole.Admin)]
        [HttpGet("{id}/info")]
        public async Task<ActionResult> GetFolder(int id)
        {
            var folderInfo = await _folderService.GetFolderInfoAsync(id);

            return Ok(folderInfo);
        }
    }
}


using System.Threading.Tasks;
using EnglishLearning.FileManager.Application.Abstract;
using EnglishLearning.Utilities.Identity.Abstractions;
using Microsoft.AspNetCore.Mvc;

namespace EnglishLearning.FileManager.Web.Controllers
{
    [Route("/api/file-manager/tree")]
    [ApiController]
    public class TreeController : Controller
    {
        private readonly ITreeService _treeService;

        public TreeController(ITreeService treeService)
        {
            _treeService = treeService;
        }

        [EnglishLearningAuthorize(AuthorizeRole.Admin)]
        [HttpGet]
        public async Task<IActionResult> Get()
        {
            var tree = await _treeService.GetTreeAsync();

            return Ok(tree);
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Threading.Tasks;
using EnglishLearning.FileManager.Application.Abstract;
using EnglishLearning.FileManager.Application.Configuration;
using EnglishLearning.FileManager.Application.Infrastructure;
using EnglishLearning.FileManager.Application.Models;
using EnglishLearning.FileManager.Persistence.Abstract;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Options;
using static EnglishLearning.FileManager.Application.Infrastructure.ApplicationMapper;

namespace EnglishLearning.FileManager.Application.Services
{
    internal class FileService : IFileService
    {
        private readonly IFileEntityRepository _fileRepository;

        private readonly IFolderService _folderService;

        private readonly FileShareConfiguration _fileShareConfiguration;

        private readonly ILogger<FileService> _logger;

        private readonly IFileManipulationServiceFactory _fileManipulationServiceFactory;

        public FileService(
            IFileEntityRepository fileRepository,
            IFolderService folderService,
            IOptions<FileShareConfiguration> fileShareConfiguration,
            IFileManipulationServiceFactory fileManipulationServiceFactory,
            ILogger<FileService> logger)
        {
            _fileRepository = fileRepository;
            _folderService = folderService;
            _fileShareConfiguration = fileShareConfiguration.Value;
            _fileManipulationServiceFactory = fileManipulationServiceFactory;
            _logger = logger;
        }

        public async Task<IReadOnlyList<FileModel>> GetAllByFolderId(int? folderId)
        {
            var files = await _fileRepository.FindAllAsync(x => x.FolderId == folderId);

            return files.MapFileEntitiesToModels();
        }

        public async Task<Stream> GetFileContentAsync(Guid id)
```

```
        {
            var fileName = id.ToString().ToUpper();
            var filePath = Path.Combine(_fileShareConfiguration.Path, fileName);
            var memoryStream = new MemoryStream();

            using (var stream = File.OpenRead(filePath))
            {
                await stream.CopyToAsync(memoryStream);
            }

            memoryStream.Seek(0, SeekOrigin.Begin);

            return memoryStream;
        }

        public async Task<FileDetailedModel> GetFileDetailedModelAsync(Guid id)
        {
            var file = await _fileRepository.GetAsync(id);
            IReadOnlyList<string> path;

            if (!file.FolderId.HasValue)
            {
                path = Array.Empty<string>();
            }
            else
            {
                var folderInfo = await _folderService.GetFolderInfoAsync(file.FolderId.Va
lue);
                path = folderInfo.Path.Concat(new [] { folderInfo.Name }).ToList();
            }

            return MapFileEntityToDetailedModel(file, path);
        }

        public async Task<IReadOnlyList<FileModel>> GetAllFromFolderAsync(int? folderId)
        {
            var allChildFolders = await _folderService.GetAllChildFoldersAsync(folderId);
            var folderIds = allChildFolders
                .Select(x => x.Id as int?)
                .ToList();
            folderIds.Add(folderId);

            var files = await _fileRepository
                .FindAllAsync(x => folderIds.Contains(x.FolderId));

            return files.MapFileEntitiesToModels();
        }

        public async Task CreateFileAsync(Stream fileStream, FileCreateModel fileCreateMo
del)
```

```
        {
            var fileManipulationService = _fileManipulationServiceFactory.GetFileManipula
tionService(fileCreateModel);

            await fileManipulationService.CreateFile(fileStream, fileCreateModel);
        }
    }
}

using System;
using System.IO;
using System.Threading.Tasks;
using EnglishLearning.FileManager.Application.Abstract;
using EnglishLearning.FileManager.Application.Configuration;
using EnglishLearning.FileManager.Application.Models;
using Microsoft.Extensions.Options;

namespace EnglishLearning.FileManager.Application.Services.FileManipulation
{
    internal abstract class BaseFileManipulationService : IFileManipulationService
    {
        protected readonly FileShareConfiguration _fileShareConfiguration;

        public BaseFileManipulationService(IOptions<FileShareConfiguration> fileShareConf
iguration)
        {
            _fileShareConfiguration = fileShareConfiguration.Value;
        }

        public abstract Task CreateFile(Stream fileStream, FileCreateModel fileCreateMode
l);

        protected async Task SaveFile(Stream fileStream, Guid id)
        {
            var fileName = id.ToString().ToUpper();
            var filePath = Path.Combine(_fileShareConfiguration.Path, fileName);

            using (var stream = File.Create(filePath))
            {
                if (fileStream.CanSeek)
                {
                    fileStream.Seek(0, SeekOrigin.Begin);
                }

                await fileStream.CopyToAsync(stream);
            }
        }
    }
}
```

```
using System;
using System.Linq;
using EnglishLearning.FileManager.Application.Abstract;
using EnglishLearning.FileManager.Application.Models;
using Microsoft.Extensions.DependencyInjection;
using static EnglishLearning.FileManager.Application.Constants.FileConstants;

namespace EnglishLearning.FileManager.Application.Services.FileManipulation
{
    internal class FileManipulationServiceFactory : IFileManipulationServiceFactory
    {
        private readonly IServiceProvider _serviceProvider;

        public FileManipulationServiceFactory(IServiceProvider serviceProvider)
        {
            _serviceProvider = serviceProvider;
        }

        public IFileManipulationService GetFileManipulationService(FileCreateModel fileCr
eateModel)
        {
            switch (fileCreateModel)
            {
                case { } model when model.Extension == Csv && !string.IsNullOrWhiteSpace(
model.CsvColumnToRead):
                    return _serviceProvider.GetRequiredService<FromCsvColumnFileManipulat
ionService>();
                case { } model when TextFileExtensions.Contains(model.Extension):
                    return _serviceProvider.GetRequiredService<TextFileManipulationServic
e>();
                case { } model when model.Extension == Zip:
                    return _serviceProvider.GetRequiredService<ZipFileManipulationService
>();
                default:
                    throw new ArgumentException("Incorrect fileExtension", nameof(fileCre
ateModel));
            }
        }
    }
}


using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using EnglishLearning.FileManager.Application.Abstract;
using EnglishLearning.FileManager.Application.Models;
using EnglishLearning.FileManager.Application.Models.Tree;
using EnglishLearning.FileManager.Persistence.Abstract;
```

```
using EnglishLearning.FileManager.Persistence.Entities;
using static EnglishLearning.FileManager.Application.Infrastructure.ApplicationMapper;

namespace EnglishLearning.FileManager.Application.Services
{
    public class FolderService : IFolderService
    {
        private readonly IFolderEntityRepository _folderRepository;

        public FolderService(IFolderEntityRepository folderRepository)
        {
            _folderRepository = folderRepository;
        }

        public async Task<IReadOnlyList<FolderModel>> GetChildFoldersAsync(int? folderId)
        {
            var folders = await _folderRepository.FindAllAsync(x => x.ParentId == folderI
d);

            return folders.MapFolderEntitiesToModels();
        }

        public async Task<FolderModel> CreateAsync(FolderCreateModel folderModel)
        {
            var folderEntity = MapFolderCreateModelToEntity(folderModel);
            var createdEntity = await _folderRepository.AddAsync(folderEntity);

            return MapFolderEntityToModel(createdEntity);
        }

        public async Task<FolderInfoModel> GetFolderInfoAsync(int folderId)
        {
            var folders = await _folderRepository.GetAllAsync();
            var foldersDictionary = folders.ToDictionary(x => x.Id);
            var folder = foldersDictionary[folderId];

            return new FolderInfoModel
            {
                Id = folder.Id,
                Name = folder.Name,
                Path = GetFolderPath(folder.ParentId, foldersDictionary),
            };
        }

        public async Task<IReadOnlyList<FolderModel>> GetAllChildFoldersAsync(int? folder
Id)
        {
            static IReadOnlyList<FolderEntity> GetAllChildFolders(
                int? currentFolderId,
                ILookup<int?, FolderEntity> folderLookup)
```

```
    {
        var allFolders = new List<FolderEntity>();
        IReadOnlyList<FolderEntity> currentFolders = Array.Empty<FolderEntity>();
        if (folderLookup.Contains(currentFolderId))
        {
            currentFolders = folderLookup[currentFolderId].ToList();
        }

        allFolders.AddRange(currentFolders);
        foreach (var folder in currentFolders)
        {
            var childFolders = GetAllChildFolders(folder.Id, folderLookup);
            allFolders.AddRange(childFolders);
        }

        return allFolders;
    }

    var folders = await _folderRepository.GetAllAsync();
    var folderLookup = folders
        .ToLookup(x => x.ParentId);

    var allFolders = GetAllChildFolders(folderId, folderLookup);

    return allFolders.MapFolderEntitiesToModels();
}

public async Task<IReadOnlyList<FolderTreeItem>> GetFolderTreeItemsAsync()
{
    var folders = await _folderRepository.GetAllAsync();

    var folderDictionary = folders.ToDictionary(x => x.Id);
    return folders
        .Select(x => new FolderTreeItem
        {
            Id = x.Id,
            Name = x.Name,
            Path = GetFolderPath(x.ParentId, folderDictionary),
        })
        .ToList();
}

private static IReadOnlyList<string> GetFolderPath(
    int? parentId,
    IReadOnlyDictionary<int, FolderEntity> folderDictionary)
{
    var path = new List<string>();
    while (parentId.HasValue)
    {
        var parent = folderDictionary[parentId.Value];
```

```
                path.Add(parent.Name);
                parentId = parent.ParentId;
            }

            path.Reverse();
            return path;
        }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using EnglishLearning.FileManager.Application.Abstract;
using EnglishLearning.FileManager.Application.Models.Tree;
using EnglishLearning.FileManager.Persistence.Abstract;
using EnglishLearning.FileManager.Persistence.Entities;

namespace EnglishLearning.FileManager.Application.Services
{
    internal class TreeService : ITreeService
    {
        private readonly IFolderService _folderService;

        private readonly IFileEntityRepository _fileRepository;

        public TreeService(
            IFolderService folderService,
            IFileEntityRepository fileRepository)
        {
            _folderService = folderService;
            _fileRepository = fileRepository;
        }

        public async Task<FileTree> GetTreeAsync()
        {
            var files = await _fileRepository.GetAllAsync();

            var folderItems = await _folderService.GetFolderTreeItemsAsync();
            var fileItems = GetFileItems(files, folderItems);

            return new FileTree
            {
                Folders = folderItems,
                Files = fileItems,
            };
        }

        private static IReadOnlyList<FileTreeItem> GetFileItems(
```

```
            IReadOnlyList<FileEntity> files,
            IReadOnlyList<FolderTreeItem> mappedFolders)
        {
            var folderDictionary = mappedFolders.ToDictionary(x => x.Id);

            return files
                .Select(x => new FileTreeItem
                {
                    Id = x.Id,
                    Name = x.Name,
                    Extension = x.Extension,
                    LastModified = x.LastModified,
                    CreatedBy = x.CreatedBy,
                    Metadata = x.Metadata,
                    Path = GetFilePath(x.FolderId, folderDictionary),
                })
                .ToList();
        }

        private static IReadOnlyList<string> GetFilePath(
            int? folderId,
            IReadOnlyDictionary<int, FolderTreeItem> folderDictionary)
        {
            if (folderId.HasValue)
            {
                var folder = folderDictionary[folderId.Value];
                return folder
                    .Path
                    .Concat(new[] { folder.Name })
                    .ToList();
            }
            else
            {
                return Array.Empty<string>();
            }
        }
    }
}

using System.Threading.Tasks;
using EnglishLearning.Dictionary.Application.Abstract;
using Microsoft.AspNetCore.Mvc;

namespace EnglishLearning.Dictionary.Web.Controllers
{
    [Route("/api/dictionary/audio")]
    [ApiController]
    public class AudioController : Controller
    {
        private readonly IAudioService _audioService;
```

```
        public AudioController(IAudioService audioService)
        {
            _audioService = audioService;
        }

        [HttpGet("word/{word}")]
        public async Task<IActionResult> Get([FromRoute] string word)
        {
            var stream = await _audioService.GetAudioAsync(word);

            return new FileStreamResult(stream, "audio/wav")
            {
                FileDownloadName = "audio.wav",
            };
        }
    }
}


using System.Collections.Generic;
using System.Threading.Tasks;
using AutoMapper;
using EnglishLearning.Dictionary.Application.Abstract;
using EnglishLearning.Dictionary.Domain.Models;
using EnglishLearning.Dictionary.Web.Contracts;
using Microsoft.AspNetCore.Mvc;

namespace EnglishLearning.Dictionary.Web.Controllers
{
    [Route("/api/dictionary/word")]
    [ApiController]
    public class WordController : Controller
    {
        private readonly IWordQueryService _queryService;

        private readonly IMapper _mapper;

        public WordController(IWordQueryService queryService, IMapper mapper)
        {
            _queryService = queryService;
            _mapper = mapper;
        }

        [HttpGet("{word}")]
        public async Task<IActionResult> Get([FromRoute] string word)
        {
            var searchModel = await _queryService.SearchAsync(word);

            var webModel = _mapper.Map<WordSearch>(searchModel);
```

```
            return Ok(webModel);
        }

        [HttpPost("query")]
        public async Task<IActionResult> FindAll([FromBody] WordSearchQuery query)
        {
            var queryModel = _mapper.Map<WordSearchQueryModel>(query);

            var words = await _queryService.FindAllAsync(queryModel);

            var webModels = _mapper.Map<IReadOnlyList<WordDetails>>(words);

            return Ok(webModels);
        }
    }
}


using System.Collections.Generic;
using System.Threading.Tasks;
using AutoMapper;
using EnglishLearning.Dictionary.Application.Abstract;
using EnglishLearning.Dictionary.Domain.Models;
using EnglishLearning.Dictionary.Web.Contracts;
using EnglishLearning.Utilities.Identity.Abstractions;
using Microsoft.AspNetCore.Mvc;

namespace EnglishLearning.Dictionary.Web.Controllers
{
    [Route("/api/dictionary/list")]
    [ApiController]
    public class WordListItemController : Controller
    {
        private readonly IWordListItemCommandService _commandService;

        private readonly IWordListItemQueryService _queryService;

        private readonly IJwtInfoProvider _jwtInfoProvider;

        private readonly IMapper _mapper;

        public WordListItemController(
            IWordListItemCommandService commandService,
            IWordListItemQueryService queryService,
            IJwtInfoProvider jwtInfoProvider,
            IMapper mapper)
        {
            _commandService = commandService;
            _queryService = queryService;
```

```
            _jwtInfoProvider = jwtInfoProvider;
            _mapper = mapper;
        }

        [EnglishLearningAuthorize]
        [HttpPost]
        public async Task<IActionResult> Create([FromBody] AddWordListDefinitionCommand c
ommand)
        {
            var commandModel = _mapper.Map<AddWordListDefinitionCommandModel>(command);
            commandModel.UserId = _jwtInfoProvider.UserId;

            await _commandService.AddDefinitionAsync(commandModel);

            return Ok();
        }

        [EnglishLearningAuthorize]
        [HttpGet]
        public async Task<IActionResult> Get()
        {
            var userId = _jwtInfoProvider.UserId;
            var models = await _queryService.FindAllAsync(userId);

            var webModels = _mapper.Map<IReadOnlyList<WordListItem>>(models);
            return Ok(webModels);
        }

        [EnglishLearningAuthorize]
        [HttpPost("learned")]
        public async Task<IActionResult> AddLearnedWords([FromBody] LearnedWordsCommand c
ommand)
        {
            var userId = _jwtInfoProvider.UserId;
            var model = _mapper.Map<LearnedWordsCommandModel>(command);
            model.UserId = userId;

            await _commandService.AddLearnedWordsAsync(model);

            return Ok();
        }

        [EnglishLearningAuthorize]
        [HttpGet("learned/random/{count}")]
        public async Task<IActionResult> GetRandomWordsToLearn([FromRoute] int count)
        {
            var userId = _jwtInfoProvider.UserId;

            var words = await _queryService.GetRandomWordsToLearnAsync(userId, count);
```

```
            var webModels = _mapper.Map<IReadOnlyList<WordListItem>>(words);
            return Ok(webModels);
        }
    }
}


using System.Collections.Generic;
using System.Threading.Tasks;
using AutoMapper;
using EnglishLearning.Dictionary.Application.Abstract;
using EnglishLearning.Dictionary.Domain.Models.Metadata;
using EnglishLearning.Dictionary.Web.Contracts.Metadata;
using EnglishLearning.Utilities.Identity.Abstractions;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Logging;

namespace EnglishLearning.Dictionary.Web.Controllers
{
    [Route("/api/dictionary/word-metadata")]
    [ApiController]
    public class WordMetadataController : Controller
    {
        private readonly ILogger<WordMetadataController> _logger;

        private readonly ICreateMetadataService _createMetadataService;

        private readonly IWordMetadataQueryService _wordMetadataQueryService;

        private readonly IMapper _mapper;

        public WordMetadataController(
            ILogger<WordMetadataController> logger,
            ICreateMetadataService createMetadataService,
            IWordMetadataQueryService wordMetadataQueryService,
            IMapper mapper)
        {
            _logger = logger;
            _createMetadataService = createMetadataService;
            _wordMetadataQueryService = wordMetadataQueryService;
            _mapper = mapper;
        }

        [EnglishLearningAuthorize(AuthorizeRole.Admin)]
        [HttpPost("command")]
        public async Task<IActionResult> Create([FromBody] CreateWordMetadataCommand crea
teCommand)
        {
            var createModel = _mapper.Map<CreateWordMetadataCommandModel>(createCommand);
            await _createMetadataService.CreateWordMetadataAsync(createModel);
```

```
            return Ok();
        }

        [HttpPost("query")]
        public async Task<IActionResult> Get([FromBody] WordMetadataQuery query)
        {
            var queryModel = _mapper.Map<WordMetadataQueryModel>(query);
            var words = await _wordMetadataQueryService.GetAsync(queryModel);

            var webModels = _mapper.Map<IReadOnlyList<WordMetadata>>(words);

            return Ok(webModels);
        }

        [HttpPost("query/filter")]
        public async Task<IActionResult> Get([FromBody] WordMetadataFilterQuery query)
        {
            var words = await _wordMetadataQueryService.FindAllAsync(query.Topic, query.L
evel);

            var webModels = _mapper.Map<IReadOnlyList<WordMetadata>>(words);

            return Ok(webModels);
        }

        [EnglishLearningAuthorize(AuthorizeRole.Admin)]
        [HttpGet("topics")]
        public async Task<IActionResult> GetTopics()
        {
            var topics = await _wordMetadataQueryService.GetAllTopicsAsync();

            return Ok(topics);
        }
    }
}


using System.IO;
using System.Threading.Tasks;
using EnglishLearning.Dictionary.Application.Abstract;
using EnglishLearning.Utilities.Speech.Contracts;

namespace EnglishLearning.Dictionary.Application.Services
{
    internal class AudioService : IAudioService
    {
        private readonly ITextToSpeechService _textToSpeechService;

        public AudioService(ITextToSpeechService textToSpeechService)
```

```
        {
            _textToSpeechService = textToSpeechService;
        }

        public Task<Stream> GetAudioAsync(string str)
        {
            return _textToSpeechService.SpeakTextAsync(str);
        }
    }
}


using System.Linq;
using System.Threading.Tasks;
using EnglishLearning.Dictionary.Application.Abstract;
using EnglishLearning.Dictionary.Domain.Models;
using EnglishLearning.Dictionary.Domain.Repositories;

namespace EnglishLearning.Dictionary.Application.Services
{
    internal class WordListItemCommandService : IWordListItemCommandService
    {
        private readonly IWordListItemRepository _repository;

        public WordListItemCommandService(IWordListItemRepository repository)
        {
            _repository = repository;
        }

        public Task AddDefinitionAsync(AddWordListDefinitionCommandModel command)
        {
            return _repository.AddWordListItemDefinitionAsync(command);
        }

        public async Task AddLearnedWordsAsync(LearnedWordsCommandModel command)
        {
            var items = await _repository.FindAllAsync(command.UserId, command.Words);
            foreach (var item in items)
            {
                item.IsLearned = true;
            }

            await _repository.UpdateAllAsync(items);
        }
    }
}


using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Threading.Tasks;
using EnglishLearning.Dictionary.Application.Abstract;
using EnglishLearning.Dictionary.Domain.Models;
using EnglishLearning.Dictionary.Domain.Repositories;
using EnglishLearning.Utilities.Linq.Extensions;

namespace EnglishLearning.Dictionary.Application.Services
{
    internal class WordListItemQueryService : IWordListItemQueryService
    {
        private readonly IWordListItemRepository _repository;

        public WordListItemQueryService(IWordListItemRepository repository)
        {
            _repository = repository;
        }

        public async Task<IReadOnlyList<WordListItemModel>> FindAllAsync(Guid userId)
        {
            var items = await _repository.FindAllAsync(userId);
            return items
                .OrderBy(x => x.Word)
                .ToList();
        }

        public async Task<IReadOnlyList<WordListItemModel>> GetRandomWordsToLearnAsync(Gu
id userId, int count)
        {
            var items = await _repository.GetNotLearnedAsync(userId);

            var randomItems = items
                .GetRandomCountOfElements(count)
                .ToList();

            return randomItems;
        }
    }
}


using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using EnglishLearning.Dictionary.Application.Abstract;
using EnglishLearning.Dictionary.Application.Constants;
using EnglishLearning.Dictionary.Common.Models;
using EnglishLearning.Dictionary.Domain.Models.Metadata;
using EnglishLearning.Dictionary.Domain.Repositories;
using Microsoft.Extensions.Logging;
```

```
namespace EnglishLearning.Dictionary.Application.Services
{
    internal class WordMetadataQueryService : IWordMetadataQueryService
    {
        private readonly IWordMetadataRepository _metadataRepository;

        private readonly ILogger<WordMetadataQueryService> _logger;

        public WordMetadataQueryService(
            IWordMetadataRepository metadataRepository,
            ILogger<WordMetadataQueryService> logger)
        {
            _metadataRepository = metadataRepository;
            _logger = logger;
        }

        public async Task<IReadOnlyList<WordMetadataModel>> GetAsync(WordMetadataQueryMod
el query)
        {
            var words = query.Words
                .Select(x => MapWordsWithApostrophe(x.ToLower()))
                .ToList();

            var wordsMetadata = await _metadataRepository.FindAllAsync(words);
            var foundedWords = wordsMetadata.Select(x => x.Word);
            var notFoundedWords = words.Except(foundedWords).ToList();
            foreach (var word in notFoundedWords)
            {
                _logger.LogWarning($"Not founded word: {word}");
            }

            return wordsMetadata;
        }

        public Task<IReadOnlyList<WordMetadataModel>> FindAllAsync(string topic, Dictiona
ryEnglishLevel level)
        {
            return _metadataRepository.FindAllAsync(topic, level);
        }

        public async Task<IReadOnlyCollection<string>> GetAllTopicsAsync()
        {
            var words = await _metadataRepository.GetAllAsync();
            var topics = words
                .SelectMany(x => x.Topics)
                .ToHashSet();

            return topics;
        }
```

```
        private string MapWordsWithApostrophe(string word)
        {
            if (AbbreviationConstants.WordsWithApostropheMap.TryGetValue(word, out string
 mapped))
            {
                return mapped;
            }

            return word;
        }
    }
}


using System.Collections.Generic;
using System.Threading.Tasks;
using EnglishLearning.Dictionary.DB.Abstract;
using EnglishLearning.Dictionary.DB.Entities;
using EnglishLearning.Utilities.Persistence.Mongo.Contexts;
using EnglishLearning.Utilities.Persistence.Mongo.Repositories;
using MongoDB.Driver;

namespace EnglishLearning.Dictionary.DB.Repositories
{
    internal class WordListItemMongoRepository : BaseMongoRepository<WordListItemEntity,
string>, IWordListItemMongoRepository
    {
        public WordListItemMongoRepository(MongoContext mongoContext)
            : base(mongoContext)
        {
        }


        public async Task UpdateAllAsync(IReadOnlyList<WordListItemEntity> entities)
        {
            foreach (var entity in entities)
            {
                await UpdateAsync(entity);
            }
        }
    }
}
```