Вінницький національний технічний університет
Факультет інформаційних інтелектуальних технологій та автоматизації
Кафедра комп'ютерних наук

Бакалаврська кваліфікаційна робота на тему:

РОЗРОБКА WEB-ДОДАТКУ ДЛЯ ВИВЧЕННЯ АНГЛІЙСЬКОЇ МОВИ

Виконала: студентка 4 курсу, групи 4КН-21б
спеціальності 122 Комп'ютерні науки
(шифр і назва напрямку підготовки, спеціальності)
Писарук О.О.
(прізвище та ініціали)
Керівник: к.т.н., доцент, асистент каф. КН
Богач І.В.
(прізвище та ініціали)
«04» червня 2025 р.
Рецензент: к.т.н., доцент, доцент каф. АІІТ
Севастьянов В.М.
(прізвище та ініціали)
«05» червня 2025 р.

Допущено до захисту
Зав. кафедри Яровий А.А.
«06» червня 2025 р.

Вінниця – 2025 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
проф., д.т.н. Яровий А.А
«05» травня 2025р

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Писарук Олександрі Олександрівні

1. Тема роботи: Розробка WEB-додатку для вивчення англійської мови.
Керівник роботи: к.т.н., доцент, асистент кафедри КН Богач І.В.
Затверджені наказом вищого навчального закладу "20.03" 2025 року №97
2. Термін подання студентом роботи 30 травня 2025 року
3. Вихідні дані до роботи : Кількість сутностей у базі даних – 12 од.; Час відгуку серверної частини – не більше 500 мілісекунд; Підтримка одночасного підключення – не менше 100 користувачів; Підтримка авторизації та захисту персональних даних; Можливість експорту словника у форматі PDF (для преміум-користувачів); Клієнтська частина додатку – 1 од.; Серверна частина додатку – 1 од.; Функціональні модулі: словник користувача, додавання слів, читання текстів, система досягнень, налаштування профілю, соціальна взаємодія – 6 од.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): Аналіз предметної області з вивчення англійської мови; Проектування WEB-додатку для вивчення англійської мови; Програмна реалізація WEB-додатку для вивчення англійської мови.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): архітектурне рішення додатку «Для вивчення англійської мови»; діаграма прецедентів взаємодії користувача з додатком; діаграма розгортання WEB-додатку для вивчення англійської мови; UML-діаграма компонентів WEB-додатку для вивчення англійської мови; Фрагмент реляційної моделі даних для веб-ресурсу «Вивчення англійської мови»; Граф-схема алгоритму додавання слова до словника.

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1-3	Богач І.В., к.т.н., доцент, асистент каф. КН		

7. Дата видачі завдання 05 травня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів проекту (роботи)	Примітк
1	Аналіз предметної області з вивчення англійської мови	05.05.25 - 08.05.25	вик
2	Проектування WEB-додатку для вивчення англійської мови	09.05.25 - 13.05.25	вик
3	Програмна реалізація WEB-додатку для вивчення англійської мови	14.05.25 - 20.05.25	вик
4	Тестування WEB-додатку для вивчення англійської мови	21.05.25 - 26.05.25	вик
5	Оформлення роботи	27.05.25 - 29.05.25	вик
6	Розробка інструкції користувача	30.05.25 - 01.06.25	вик
7	Оформлення матеріалів до захисту БКР	02.06.25 - 04.06.25	вик

Студентка _____ Писарук О.О.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Богач І.В.
(підпис) (прізвище та ініціали

АНОТАЦІЯ

Бакалаврська дипломна робота складається з 93 сторінок формату А4, на яких є 22 рисунків, список використаних джерел містить 24 найменувань.

У цій бакалаврській роботі представлено повний цикл розробки веб-додатку персонального англомовного словника, створеного для підтримки вивчення англійської мови. Система розроблена на базі фреймворку Django, який забезпечує серверну логіку та управління даними, із використанням SQLite для зберігання інформації та Celery для виконання асинхронних задач, таких як надсилання електронних листів. Веб-додаток дозволяє користувачам створювати власні словники, групувати слова, читати тексти з інтерактивним перекладом, відстежувати прогрес через систему досягнень, а також взаємодіяти з іншими через дружні зв’язки та спільний доступ до лексики. Реалізована функціональність сприяє підвищенню ефективності навчання шляхом персоналізації процесу, автоматизації оцінки знань та заохочення співпраці між користувачами.

Ключові слова: веб-додаток, персональний словник, навчання мов, Django, SQLite, Celery, інтерактивне навчання.

ABSTRACT

The bachelor's thesis consists of 93 A4 pages, containing 22 figures, with a list of references comprising 24 items.

This bachelor's thesis presents the complete development cycle of a web application for a personal English dictionary designed to support English language learning. The system is built on the Django framework, which provides server-side logic and data management, using SQLite for data storage and Celery for handling asynchronous tasks, such as sending emails. The web application enables users to create their own dictionaries, group words, read texts with interactive translation, track progress through an achievement system, and interact with others via friendships and shared vocabulary access. The implemented functionality enhances learning efficiency by personalizing the process, automating knowledge assessment, and encouraging collaboration among users.

Keywords: web application, personal dictionary, language learning, Django, SQLite, Celery, interactive learning.

ЗМІСТ

ВСТУП

Актуальність дослідження зумовлена потребою у сучасних цифрових засобах для ефективного вивчення англійської мови. У добу стрімкого розвитку інформаційних технологій особливої ваги набувають інструменти, що дозволяють опановувати іноземні мови в зручний, інтерактивний та персоналізований спосіб. Зростання ролі англійської як мови міжнародного спілкування, навчання, науки й бізнесу формує сталий попит на гнучкі програмні рішення, які адаптуються під індивідуальні потреби користувачів.

При цьому все більше користувачів очікують інтелектуальних функцій, наприклад адаптивного підбору вправ на основі їхнього рівня та темпу навчання. Безперервний доступ із будь-якого пристрою та можливість інтеграції з сучасними сервісами (штучний інтелект, розпізнавання голосу) стають ключовими факторами успішності таких продуктів.

Однак, сучасні додатки для вивчення англійської мови, попри широку популярність, часто мають низку обмежень. Більшість з них не дає можливості створювати власний словник, редагувати або групувати слова, обирати теми для читання або обмінюватися контентом із іншими користувачами.

Тому актуальним є розробка веб-додатку, що забезпечить персоналізоване навчальне середовище, дозволить поєднати функціональність навчального словника, інтерактивного читання, статистики прогресу та соціальної взаємодії між користувачами та значно підвищить ефективність засвоєння англійської мови за рахунок персоналізованого навчального середовища, гнучкого керування навчальним контентом, систему досягнень і соціальну взаємодію між користувачами.

Метою роботи є розширення функціональних можливостей web-додатку для вивчення англійської мови.

Об’єктом дослідження є процес вивчення іноземної мови засобами web-технологій.

Предметом дослідження є програмні засоби у вигляді web-додатку для організації вивчення англійської мови.

Задачі дослідження:

1. Провести аналіз предметної області з вивчення англійської мови, проаналізувати існуючі програмні рішення та обґрунтовати доцільність розробки.

2. Спроєктувати web додаток для вивчення англійської мови.

3. Реалізувати web додаток для вивчення англійської мови.

4. Виконати тестування розробленого web-додатку та провести аналіз отриманих результатів.

5. Розробити інструкцію користувача.

Апробація роботи.

Результати роботи було представлено на LIV Всеукраїнській науково-технічній конференції підрозділів Вінницького національного технічного університету (2025) та на Міжнародній науково-практичній Інтернет-конференції студентів, аспірантів та молодих науковців «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)».

Публікації: електронні тези конференції LIV Всеукраїнській науково-технічній конференції підрозділів Вінницького національного технічного університету (2025) [4]; Міжнародній науково-практичній Інтернет-конференції студентів, аспірантів та молодих науковців «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)».

Впровадження. Результати роботи планується використати для організації вивчення англійської мови в ТОВ «ІТІ» (додаток Д).

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ З ВИВЧЕННЯ АНГЛІЙСЬКОЇ МОВИ

1.1 Обґрунтування доцільності розробки WEB-додатку для вивчення англійської мови

Сучасний світ стрімко розвивається завдяки інформаційним технологіям, які охоплюють усі сфери життя, зокрема освіту. В умовах глобалізації англійська мова, як одна з найважливіших мов міжнародного спілкування, набуває все більшої популярності. Оволодіння нею відкриває доступ до світових інформаційних ресурсів, сприяє професійному зростанню та полегшує міжкультурний діалог. У зв’язку з цим створення додатків для вивчення англійської мови є важливим напрямом, оскільки такі інструменти роблять навчальний процес зручнішим, доступнішим і більш результативним.

Традиційні методи вивчення мов, такі як заняття з репетитором чи використання друкованих підручників, мають певні обмеження. Вони потребують значних часових і фінансових витрат, а також не завжди враховують індивідуальні потреби учнів. Водночас цифрові технології надають нові можливості для оптимізації навчання. Веб-додатки для вивчення англійської мови дозволяють користувачам навчатися у зручний час і в комфортному середовищі, адаптуючи матеріал до власного рівня знань і поєднуючи навчання з іншими повсякденними справами. Це підтверджує актуальність створення такого додатку та його відповідність сучасним потребам суспільства.

Однією з головних переваг таких програм є можливість персоналізації. Користувачі можуть самостійно обирати лексику для вивчення, формувати власні списки слів і відстежувати свій прогрес, що підвищує мотивацію та покращує засвоєння матеріалу. Додатки також можуть містити інтерактивні елементи, зокрема читання текстів із перекладом чи систему досягнень, які роблять навчання цікавішим і динамічнішим. Такі функції не лише сприяють

кращому запам’ятовуванню інформації, а й підтримують зацікавленість користувача впродовж тривалого часу.

Ще одним важливим аспектом розробки подібного додатку є можливість соціальної взаємодії. У сучасному світі люди прагнуть обмінюватися знаннями та співпрацювати. Додаток може надавати інструменти для обміну лексикою, створення навчальних груп і отримання зворотного зв’язку від інших користувачів. Це не тільки сприяє формуванню спільноти однодумців, а й додає мотивації завдяки соціальній залученості.

Аналіз ринку підтверджує перспективність розробки таких додатків. Згідно з дослідженнями, кількість користувачів, які опановують англійську мову за допомогою цифрових платформ, постійно зростає. Популярні сервіси, як-от Duolingo чи Quizlet, мають мільйони активних користувачів по всьому світу, що свідчить про високий попит на ефективні онлайн-інструменти для навчання. Водночас багато існуючих додатків пропонують стандартні набори вправ, тоді як сучасні користувачі все частіше потребують персоналізованих рішень, які враховують їхні інтереси та цілі.

Розробка веб-додатку для вивчення англійської мови також має економічне обґрунтування. Такі платформи не вимагають значних витрат на друк матеріалів чи організацію фізичних занять, що робить їх доступними для широкого кола користувачів. Використання сучасних технологій, наприклад, фреймворку Django, дозволяє швидко створювати стабільні, гнучкі та масштабовані системи, які легко адаптуються до нових вимог.

Отже, розробка додатку для вивчення англійської мови є доцільною з огляду на низку важливих факторів: зростаючу потребу в ефективних навчальних інструментах, можливість персоналізації та інтерактивності, соціальну взаємодію в освітньому процесі, а також економічні переваги використання веб-технологій. Такий додаток відповідатиме актуальним тенденціям у сфері освіти та стане корисним інструментом як для індивідуальних користувачів, так і для спільнот, що прагнуть покращити свої мовні навички.

1.2 Аналіз існуючих програмних рішень для вивчення англійської мови

Основною метою застосунку для вивчення англійської мови є забезпечення користувачів зручним і ефективним інструментом для засвоєння лексики, граматики та практичних навичок. Такі програми допомагають підвищити мотивацію до навчання, пропонують індивідуальний підхід та дозволяють відстежувати власний прогрес. Завдяки цьому процес опанування мови стає більш гнучким і адаптивним до швидкого темпу сучасного життя.

Сьогодні на ринку представлено багато програмних рішень, які тією чи іншою мірою сприяють вивченню англійської мови. Серед найпопулярніших платформ варто виділити Duolingo, Quizlet і Memrise.

Duolingo — це один із найвідоміших додатків для мовного навчання, який ґрунтується на гейміфікації. Він пропонує користувачам короткі інтерактивні уроки, що включають переклад, аудіювання та вибір правильних відповідей. Основна перевага Duolingo полягає в його простоті та гейміфікованому підході, який перетворює навчання на захопливий процес. Учасники можуть навчатися у зручний час, отримувати бали за правильні відповіді та змагатися з іншими користувачами. Крім того, система нагадує про необхідність регулярного навчання через push-сповіщення, що сприяє формуванню стійкої звички. Проте, попри зручність і популярність, Duolingo має низку обмежень. Користувачі не можуть створювати власні словники, об'єднувати слова в тематичні групи чи працювати з автентичними або адаптованими текстами. Функціонал платформи не передбачає можливості обміну контентом між користувачами або гнучкого налаштування навчального процесу під індивідуальні потреби. Це зменшує рівень персоналізації, що може бути критичним для тих, хто прагне глибшого та більш контрольованого вивчення мови. Саме ці обмеження стали поштовхом для розробки альтернативного веб-додатку з розширеними можливостями. Приклад інтерфейсу платформи Duolingo зображено на рисунку 1.1.[3]

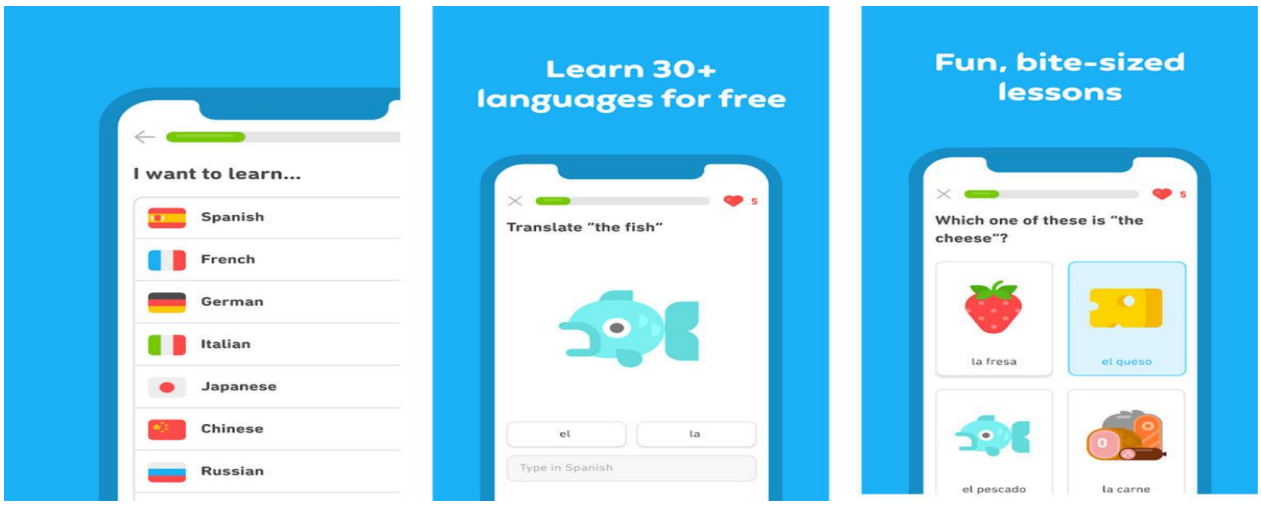


Рисунок 1.1 – Приклад інтерфейсу платформи Duolingo

Quizlet є платформою для роботи з навчальними картками, яка дозволяє користувачам створювати власні набори слів, додавати переклади та обмінюватися ними з іншими. Відмінною особливістю Quizlet є різноманітні режими навчання: тести, вправи на відповідність та адаптивний режим "Вивчення", що підлаштовується під рівень користувача. Додатково платформа підтримує аудіосупровід, що допомагає у розвитку навичок вимови та розуміння на слух. Завдяки інтуїтивному інтерфейсу та можливості колективного навчання цей сервіс популярний серед студентів і викладачів. Приклад інтерфейсу сайту Quizlet зображено на рисунку 1.2.[4]

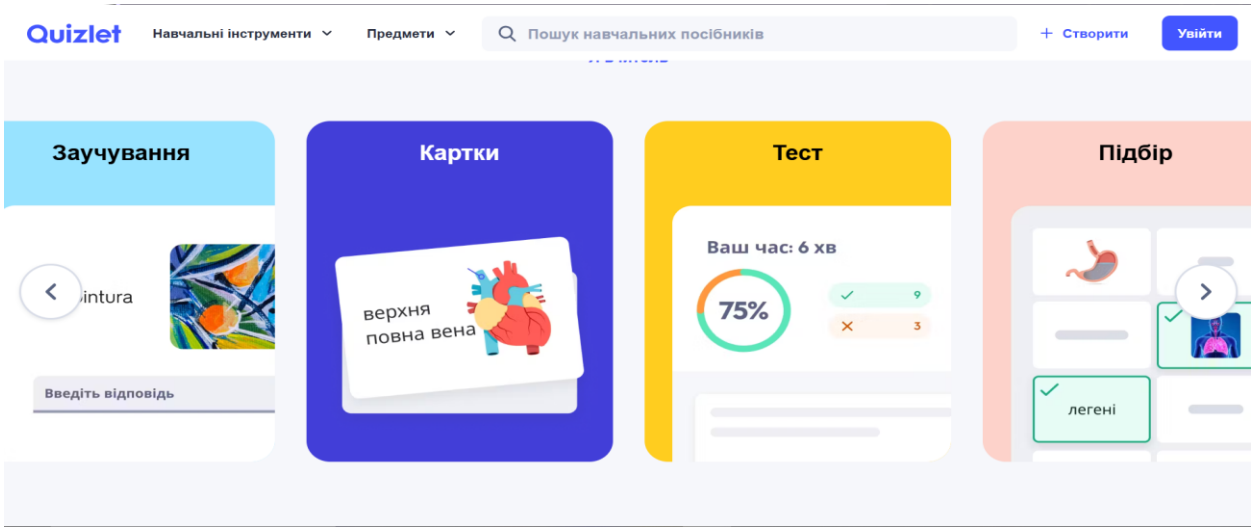


Рисунок 1.2 – Приклад інтерфейсу сайту Quizlet

Memrise спеціалізується на запам’ятовуванні слів за допомогою асоціативного методу та інтервального повторення. Такий підхід сприяє довготривалому засвоєнню нової лексики. Особливістю Memrise є інтеграція відео з носіями мови та можливість створення персоналізованих курсів, що робить навчання більш реалістичним і наближеним до природного мовного середовища. Крім того, сервіс надає детальну статистику прогресу, що дозволяє користувачам контролювати власні результати. Приклад інтерфейсу сайту Memrise зображено на рисунку 1.3.[5]

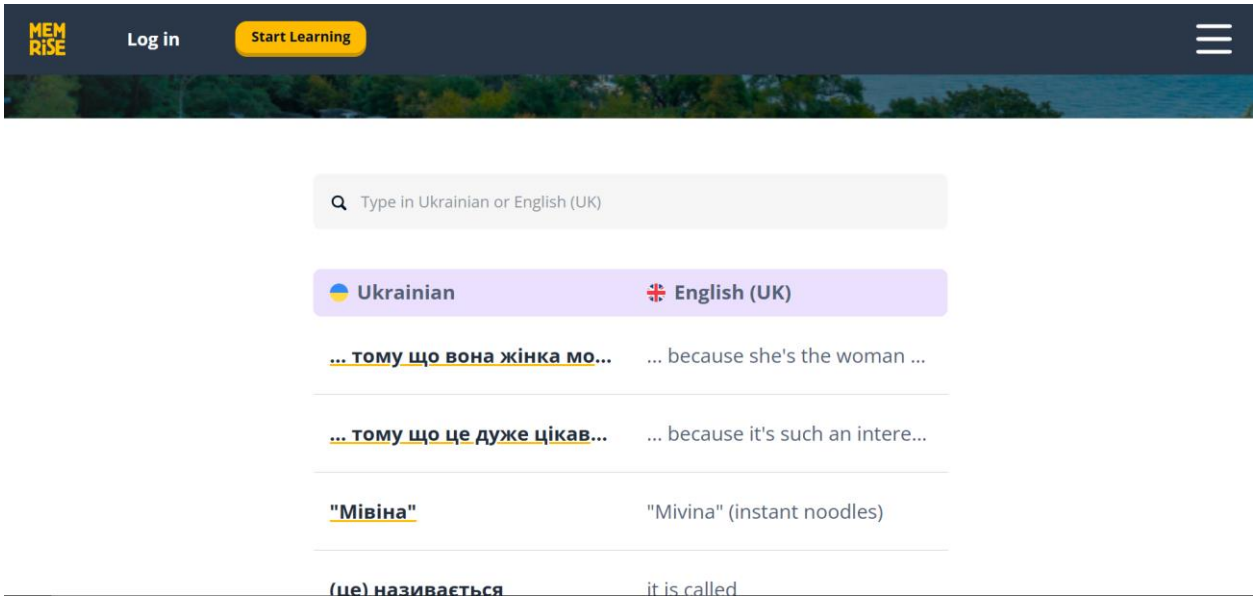


Рисунок 1.3 – Приклад інтерфейсу сайту Memrise

Попри численні переваги кожної з платформ, вони мають певні обмеження. Наприклад, Duolingo добре підходить для початкового рівня, але його вправи обмежені простими граматичними конструкціями та не забезпечують роботи з власними текстами. Quizlet фокусується переважно на лексичних вправах, однак не пропонує інструментів для розвитку навичок читання чи соціальної взаємодії між користувачами. Memrise ефективний для запам’ятовування слів, проте має недостатньо функцій для інтерактивного навчання в групах чи опрацювання складних граматичних тем.

З метою наочного порівняння основних характеристик зазначених платформ подано таблицю 1.1.

Таблиця 1.1 – Порівняння систем для вивчення англійської мови

	Duolingo	Quizlet	Memrise
Інтерфейс	Простий	Простий	Інтерактивний
Гейміфікація	+	+	+
Створення власної лексики	-	+	+
Читання текстів із перекладом	-	-	-
Соціальна взаємодія	-	+	-
Система досягнень	+	-	+
Інтервальне повторення	-	-	+

Таким чином, аналіз популярних платформ для вивчення англійської мови, зокрема Duolingo, Quizlet і Memrise, засвідчив їхню ефективність у певних аспектах навчання. Водночас жодна з цих систем не забезпечує комплексного підходу, що включав би персоналізовані словники, інтерактивне читання з перекладом та активну взаємодію між користувачами. Ці обмеження вказують на перспективність розробки нового додатку для вивчення англійської мови, який поєднував би розширені можливості персоналізації, інтерактивності та співпраці. Саме такі функції реалізовано в представленому проєкті на основі Django.

1.3 Постановка задачі та вимог до програмної реалізації WEB-додатку для вивчення англійської мови

Додаток для вивчення англійської мови має відповідати низці важливих вимог, серед яких – зручність використання, ефективність навчання та персоналізація процесу. Його ключові складові включають серверну частину, розроблену на основі Django, базу даних SQLite для збереження інформації та Celery для виконання асинхронних операцій.[6]

Django – це потужний веб-фреймворк з відкритим вихідним кодом, створений на Python, що забезпечує швидку розробку безпечних та масштабованих веб-застосунків. Він має вбудовану систему автентифікації, об’єктно-реляційне відображення (ORM) для роботи з базами даних і модульну архітектуру, що значно спрощує реалізацію складних функцій. [7]

SQLite – легка реляційна база даних, яка не потребує окремого серверного процесу. Вона вирізняється простотою інтеграції, високою швидкістю роботи з невеликими проєктами та мінімальними вимогами до налаштувань, що робить її зручним рішенням для зберігання словникових даних, інформації про користувачів та навчальних матеріалів.

Celery – це інструмент для обробки асинхронних задач у Python, який інтегрується з Django, дозволяючи виконувати операції у фоновому режимі. Його застосування підвищує продуктивність додатку, оскільки завдання, такі як надсилання повідомлень чи обробка великих обсягів даних, не уповільнюють роботу основного потоку.

Django був обраний для реалізації серверної частини завдяки простоті використання, широкому набору вбудованих інструментів безпеки та можливості швидкої розробки веб-додатків. У порівнянні з Flask, який вимагає більше ручного налаштування, або FastAPI, орієнтованого переважно на асинхронні API, Django надає комплексний підхід до створення функціоналу, включаючи інтегровані рішення для автентифікації та роботи з базами даних. Значна спільнота користувачів і доступна документація роблять цей фреймворк ще більш привабливим для розробників.

Вибір SQLite обумовлений її компактністю та достатньою продуктивністю для середньомасштабних проєктів. На відміну від PostgreSQL, яка пропонує розширені функції, але потребує більше ресурсів і складніших налаштувань, SQLite є оптимальним рішенням для організації структурованих даних. У порівнянні з нереляційними базами, такими як MongoDB, вона краще підходить для збереження даних зі строгими взаємозв’язками, що відповідає вимогам цього проєкту.

Використання Celery забезпечує ефективну обробку фонових процесів, таких як розсилання сповіщень або автоматичне оновлення навчальних матеріалів. Порівняно з іншими рішеннями, такими як RQ, Celery має ширший функціонал і краще взаємодіє з Django, що робить його оптимальним вибором для даного додатку.

Серед основних функцій системи слід виділити:

- реєстрацію та автентифікацію користувачів для безпечного та персоналізованого доступу;
- створення та керування персональними словниками, що дає змогу додавати, групувати та переглядати вивчені слова;
- інтерактивне читання текстів із вбудованим перекладом для ефективного засвоєння нової лексики;
- відстеження прогресу за допомогою системи досягнень, що сприяє мотивації користувачів;
- соціальну взаємодію, включаючи можливість створення дружніх зв’язків та спільного використання навчальних матеріалів.

Розробка додатку на основі Django із використанням SQLite та Celery дозволяє створити стабільну, гнучку та функціональну платформу для вивчення англійської мови. Такий підхід забезпечує поєднання персоналізації, інтерактивності та можливостей співпраці, що робить систему ефективним інструментом для користувачів із різними рівнями знань та навчальними потребами. [8]

1.4 Висновки

Розгляд ключових аспектів у першому розділі підтвердив актуальність створення додатку для вивчення англійської мови, який відповідав би сучасним освітнім викликам. Дослідження предметної області засвідчило, що в умовах швидкої цифрової трансформації та зростаючого попиту на англійську мову необхідні інноваційні рішення, що поєднують ефективність,

персоналізацію та інтерактивність навчання. Використання технологій Django, SQLite і Celery забезпечує розробку стабільної, гнучкої та продуктивної системи, що гармонійно поєднує простоту впровадження із широкими функціональними можливостями.

Аналіз наявних рішень, зокрема Duolingo, Quizlet і Memrise, підтвердив їхню результативність у певних аспектах навчання, але водночас виявив їхні обмеження щодо комплексного підходу до персоналізованого навчання, інтерактивного опрацювання текстів і соціальної взаємодії. Це свідчить про перспективність створення нового додатку, що компенсує ці недоліки, пропонуючи користувачам розширений набір можливостей для ефективного опанування мови. У результаті впровадження такого інструменту можна очікувати покращення якості мовної освіти, підвищення мотивації користувачів та сприяння розвитку їхніх навичок у зручному й доступному форматі.

2 ПРОЕКТУВАННЯ WEB-ДОДАТКУ ДЛЯ ВИВЧЕННЯ АНГЛІЙСЬКОЇ МОВИ

2.1 Аналіз та вибір архітектурних рішень WEB-додатку для вивчення англійської мови

Під час створення додатку для вивчення англійської мови ключову увагу приділено вибору архітектури системи, яка б відповідала всім вимогам щодо стабільності, масштабованості, гнучкості й зручності подальшої підтримки. Враховуючи функціональні потреби, особливості предметної області та специфіку взаємодії користувачів із системою, було обрано клієнт-серверну архітектуру з чітким розмежуванням між логікою програми, обробкою даних та інтерфейсною частиною.

Серверна частина реалізована за допомогою високорівневого веб-фреймворку Django, що базується на мові програмування Python. Django забезпечує зручне використання архітектурного підходу Model–View–Template (MVT), у межах якого чітко структурується логіка взаємодії з базою даних, обробки запитів та візуального подання інформації користувачеві. Такий підхід дозволяє створювати проєкти, які легко підтримувати, масштабувати та адаптувати до нових вимог. [1]

Одним із ключових факторів на користь Django є його вбудовані механізми авторизації, маршрутизації, обробки форм, кешування та захисту від типових веб-загроз. Це дозволяє значно скоротити час на реалізацію базової логіки та зосередитись на розробці функціоналу, специфічного для навчального процесу. Крім того, Django надає інструменти для реалізації багаторівневої структури даних, що дозволяє ефективно моделювати навчальні об’єкти, їх властивості та взаємозв’язки між ними.

Для зберігання інформації використано реляційну базу даних, що реалізована на основі SQLite. Такий вибір зумовлений простотою у використанні, мінімальними вимогами до налаштування та повною

інтеграцією з інструментами Django. База даних зберігає навчальний матеріал (словникові одиниці, тексти), дані про користувачів, їхній прогрес, досягнення, результати виконаних завдань, а також інформацію про взаємодію між користувачами (наприклад, систему друзів). [5]

Окрему роль у проєкті відіграє обробка фонових процесів, що не потребують миттєвого реагування. Для цього інтегровано систему асинхронних задач Celery, яка дозволяє виконувати певні операції у фоновому режимі — наприклад, оновлення статистики, обробку сповіщень або очищення кешу. Це суттєво знижує навантаження на основний потік обробки запитів і забезпечує стабільну роботу системи навіть за умов високої активності користувачів.

Загальна архітектура додатку створює надійну основу для реалізації широкого функціоналу: вивчення лексики, читання англомовних текстів, виконання інтерактивних вправ, накопичення балів і досягнень, формування дружніх зв’язків та збереження статистики навчання. Структура системи є відкритою для подальшого розширення — як у межах веб-платформи, так і через зовнішні інтерфейси, зокрема мобільні додатки чи API. Архітектурну модель проєкту представлено на рисунку 2.1.

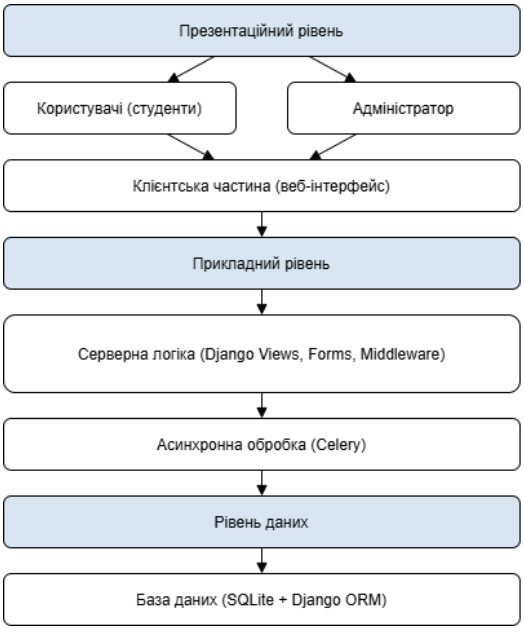


Рисунок 2.1 – Архітектурне рішення додатку «Для вивчення англійської мови»

Таким чином, обрана архітектура повністю відповідає цілям і завданням проєкту, забезпечує ефективну реалізацію навчального процесу у веб-середовищі та створює технічне підґрунтя для якісної, зручної й безпечної взаємодії з користувачем. [9]

2.2 Розробка UML-діаграм WEB-додатку для вивчення англійської мови

UML (Unified Modeling Language) — уніфікована мова моделювання, яка використовується для формалізації структури, поведінки та взаємодії елементів програмної системи. За допомогою UML-діаграм можна візуалізувати логіку системи до її реалізації, спростити етапи проєктування, аналізу й перевірки архітектурних рішень. UML-діаграми особливо корисні в розробці складних додатків, де присутні багато ролей користувачів, об'єктів і взаємодій між ними.

Діаграма прецедентів, зображена на рисунку 2.2, описує типові сценарії взаємодії користувачів із додатком. Основний актор — зареєстрований користувач, який має доступ до ключових функцій: перегляд і редагування профілю, робота з особистим словником, додавання нових слів, перегляд списку збережених слів, ознайомлення з досягненнями, додавання друзів, перегляд профілів інших користувачів. Окрему роль виконує адміністратор, який керує навчальним контентом та модерує систему загалом. Така діаграма дозволяє чітко окреслити межі функціоналу для кожної ролі та сформувати базу для побудови архітектури взаємодій.[10]

На діаграмі послідовностей змодельовано сценарій додавання нового слова до словника користувача. Користувач ініціює дію, відкриває відповідну форму, вводить нове слово та відправляє запит. Система перевіряє наявність цього слова у загальній базі, зберігає його в особистому словнику користувача або встановлює зв’язок із вже існуючим записом. Після завершення операції

користувач отримує оновлений список своїх слів. Така послідовність дій дозволяє уникати дублювання даних та підтримувати цілісність структури.

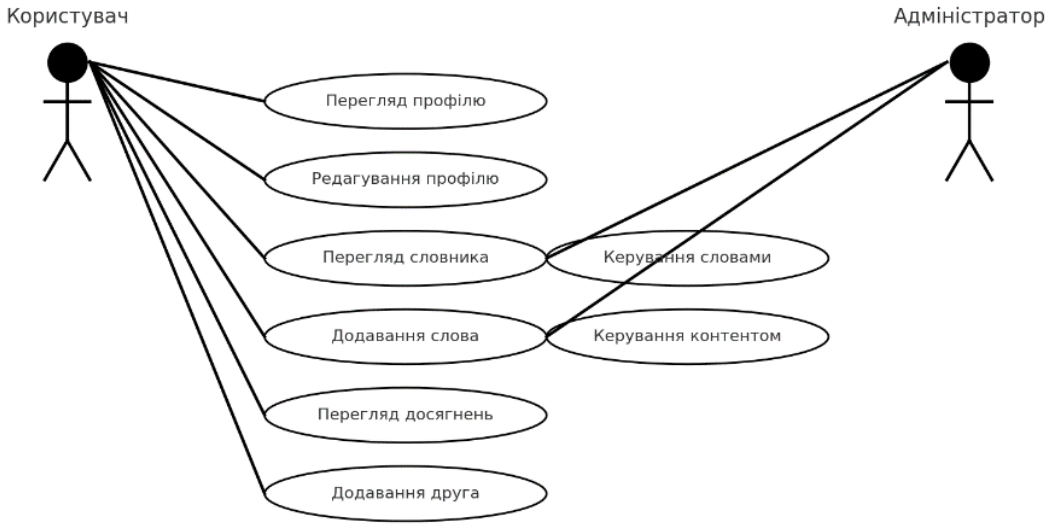


Рисунок 2.2 – Діаграма прецедентів взаємодії користувача з додатком

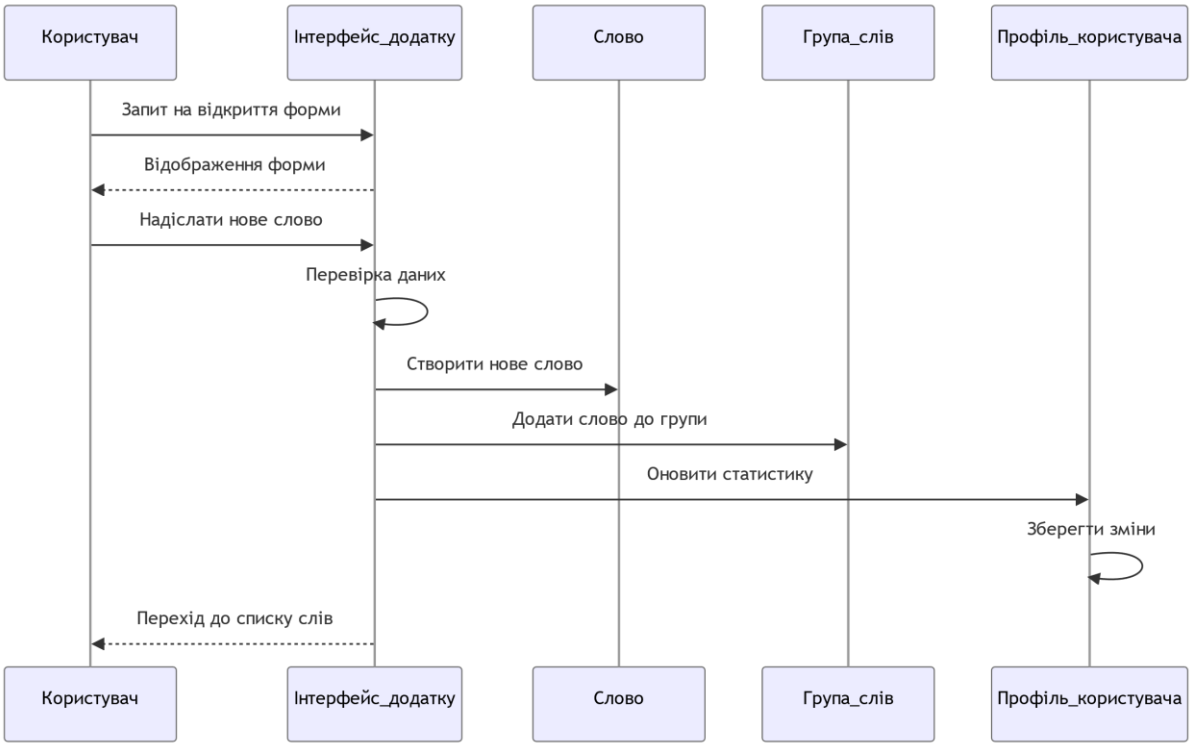


Рисунок 2.3 – Діаграма послідовностей сценарію додавання слова до словника

Ще один сценарій демонструє оновлення профілю. Користувач відкриває інтерфейс редагування, вносить зміни до персональних даних, після чого система перевіряє коректність введеної інформації, зберігає її у базі даних та повертає оновлений профіль для перегляду. У разі помилок користувач отримує повідомлення з описом проблеми. Ця діаграма дозволяє відстежити послідовність перевірок, збереження та оновлення даних у модулі профілю.

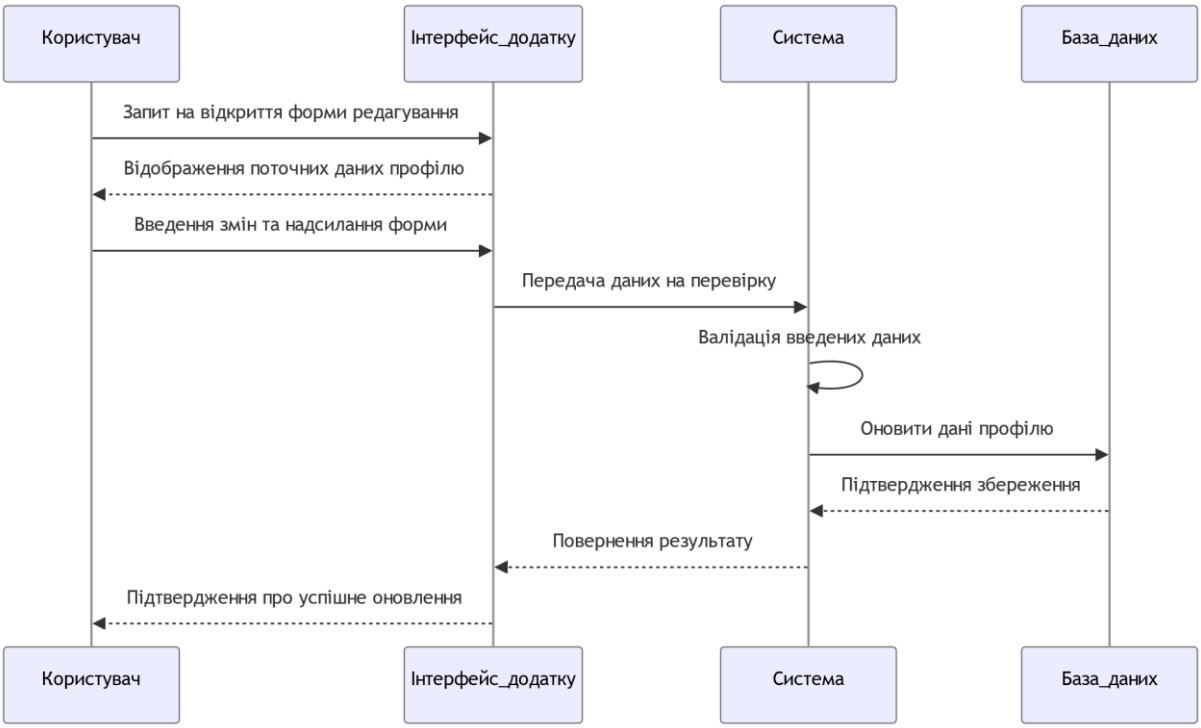


Рисунок 2.4 – Діаграма послідовностей сценарію оновлення профілю користувача

На діаграмі розгортання представлено фізичну структуру компонентів системи. Користувацький інтерфейс функціонує у веб-браузері, де відбувається взаємодія із серверною частиною. Сервер обробляє запити, відповідає за логіку взаємодії користувачів, обробку даних словника, досягнень, друзів і профілів. Усі дані зберігаються в централізованій базі даних. Для виконання асинхронних задач, таких як оновлення статистики чи присвоєння досягнень, застосовується окремий фоновий процес, що отримує задачі через посередника повідомлень. Дана структура забезпечує чітке

розділення функцій і дозволяє системі масштабуватися в умовах зростання навантаження.

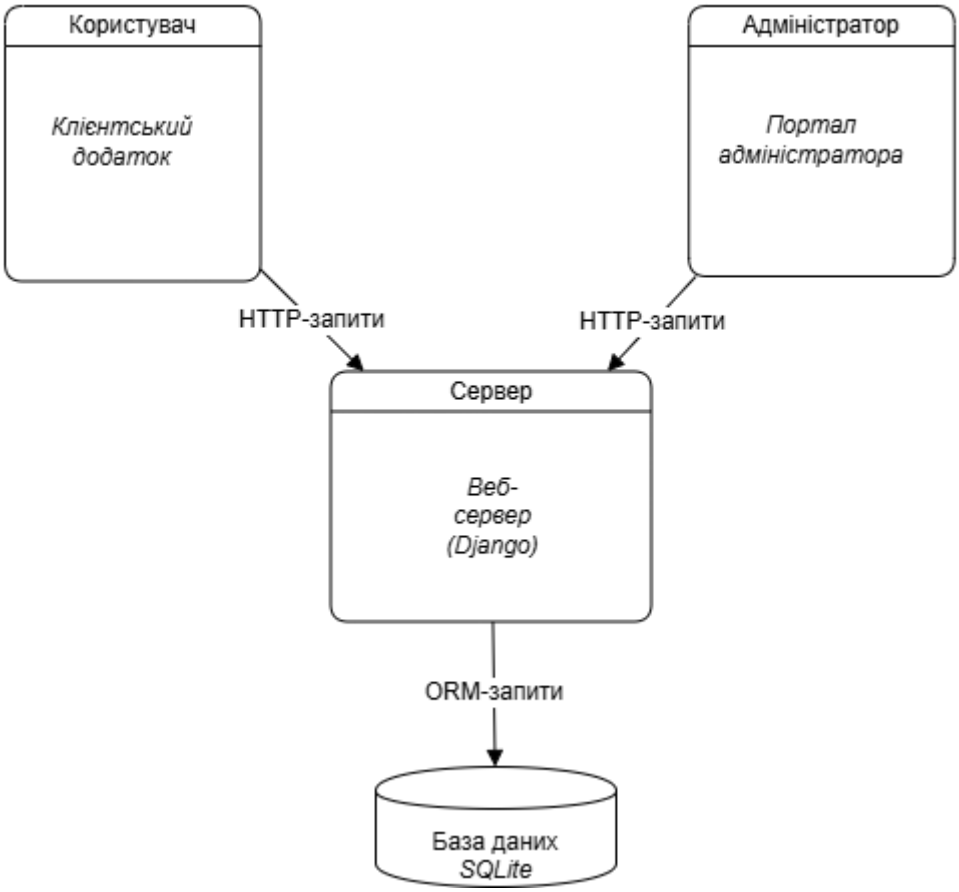


Рисунок 2.5 – Діаграма розгортання WEB-додатку для вивчення англійської мови

2.3 Розробка алгоритмів WEB-додатку для вивчення англійської мови

На даному етапі реалізуються основні алгоритми, що забезпечують функціонування системи. Їх завдання — забезпечити інтуїтивну взаємодію користувача з інтерфейсом, стабільну обробку запитів і логіку збереження навчальної інформації.

Одним з ключових є алгоритм додавання нового слова до словника користувача. У ході його виконання система перевіряє наявність слова у загальному словнику. Якщо запис існує, він прив’язується до поточного

профілю. У протилежному випадку створюється новий запис, де вказуються переклад, приклад використання та тематична категорія. Така логіка дозволяє уникнути дублювання інформації та оптимізувати структуру бази.

Паралельно формується алгоритм перегляду тексту з перекладом кожного слова. Після розбиття тексту на окремі лексеми, система перевіряє кожну з них на наявність у словнику користувача або загальному реєстрі, і виводить переклад у спливаючому вікні при натисканні. Завдяки цьому забезпечується інтерактивний і персоналізований процес навчання.

Опрацьовується також механізм редагування профілю користувача, зокрема — змінення особистих даних і аватара. Усі введені дані проходять перевірку на коректність, після чого зберігаються в базу даних, і користувач перенаправляється на оновлену сторінку профілю. [11]

Реалізується логіка соціальної взаємодії між користувачами, яка включає функції надсилання, підтвердження та скасування запитів на дружбу. Після підтвердження користувачі отримують можливість переглядати статистику і словники одне одного. Завдяки реалізації механізму дружби, користувачі можуть не лише обмінюватися лексичними матеріалами, а й надихатися до навчання шляхом взаємного спостереження за досягненнями. Така взаємодія підсилює мотивацію, сприяє формуванню освітньої спільноти та створює ефект соціального навчання. Передбачено також обмеження доступу до даних через налаштування приватності в профілі, що дозволяє користувачам самостійно визначати рівень відкритості свого навчального прогресу.

Крім того, система передбачає можливість копіювання слів із груп друзів до власного словника, що значно полегшує обмін навчальними матеріалами. Таке рішення підвищує ефективність взаємодії, сприяючи не лише особистому прогресу, а й колективному навчанню.

Структура компонентів додатку зображена на UML-діаграмі компонентів (рисунок 2.6), яка наочно демонструє розподіл відповідальностей між інтерфейсом, серверною логікою, базою даних і фоновими задачами (рисунок нижче).

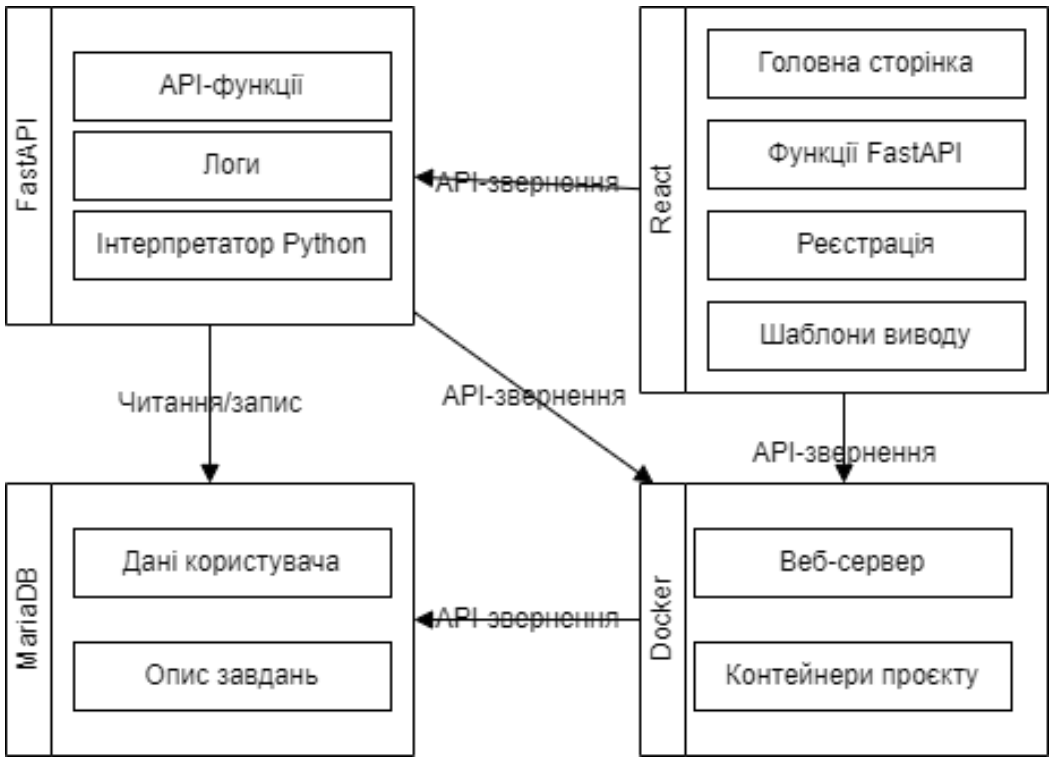


Рисунок 2.6 – UML-діаграма компонентів WEB-додатку для вивчення англійської мови

Алгоритм додавання слова до словника реалізує перевірку наявності відповідного запису в глобальній базі. Якщо слово вже існує, система не створює новий запис, а зв’язує його з поточним користувачем, враховуючи індивідуальні параметри: тип слова, групу, статус улюбленого тощо. У разі відсутності такого запису створюється новий, зберігаючи супутню інформацію — переклад, приклад використання, дату створення та зв’язок із текстами. Це дозволяє користувачу бачити контекст, у якому він зустрів слово, що сприяє глибшому засвоєнню лексики. Операції відбуваються із застосуванням асинхронної обробки, що забезпечує швидкість та зручність у роботі з додатком. Граф-схема алгоритму, що наочно демонструє вищезазначену логіку перевірки, створення запису або його зв’язування з користувачем, представлена на рисунку 2.7.

Рисунок 2.7 – Граф-схема алгоритму додавання слова до словника

Таким чином, реалізовані алгоритми спрямовані на забезпечення персоналізованого навчального процесу, збереження прогресу користувача та ефективну взаємодію між компонентами системи.

2.4 Розробка бази даних WEB-додатку для вивчення англійської мови

Розробка бази даних є важливим етапом створення веб-ресурсу, що забезпечує структуроване зберігання даних користувачів, словникового запасу, груп слів, досягнень та текстів для читання. У даному розділі описано вибір системи керування базами даних, структуру реляційної бази даних та представлено фрагмент моделі даних.

Для реалізації бази даних веб-ресурсу обрано SQLite — легковагову реляційну СКБД із відкритим вихідним кодом. Вибір SQLite обґрунтовано такими чинниками:

1. Простота інтеграції: SQLite не потребує окремого серверного процесу, що спрощує розгортання та тестування веб-ресурсу на етапі розробки.
2. Підтримка Django ORM: SQLite є стандартною СКБД для Django, що забезпечує швидку інтеграцію через об’єктно-реляційне відображення.
3. Достатня продуктивність: Для веб-ресурсу з обмеженою кількістю одночасних користувачів (до 100) SQLite забезпечує достатню швидкість обробки запитів.
4. Портативність: База даних зберігається в одному файлі, що полегшує перенесення даних між середовищами.

Альтернативи, такі як PostgreSQL або MySQL, відхилено через складність налаштування та надмірну функціональність для проєкту з обмеженими вимогами до масштабування.

База даних веб-ресурсу складається з 12 основних сутностей, що відповідають моделям, визначеним у файлі models.py. Кожна сутність забезпечує певну функціональність системи та має зв’язки з іншими сутностями. Нижче наведено перелік сутностей, їхнє призначення, ключові атрибути та зв’язки:

1. Word (Слово):

- Призначення: Зберігає інформацію про слова англійської мови для вивчення користувачами.
- Ключові атрибути: word (CharField), translation (CharField), is_favourite (BooleanField), word_type (CharField).
- Зв’язки: Один до багатьох із User, багато до багатьох із WordGroup.

2. WordGroup (Група слів):

- Призначення: Дозволяє користувачам групувати слова для зручного вивчення та обміну.
- Ключові атрибути: name (CharField), is_main (BooleanField), user (ForeignKey).
- Зв’язки: Багато до багатьох із Word, один до багатьох із User, один до одного з CommunityGroup.

3. CommunityGroup (Спільнота груп):

- Призначення: Керує статусом груп слів, які можуть бути публічними або очікувати затвердження.
- Ключові атрибути: group (OneToOneField), state (CharField).
- Зв’язки: Один до одного з WordGroup.

4. UserProfile (Профіль користувача):

- Призначення: Зберігає налаштування та статистику активності користувача.
- Ключові атрибути: user (OneToOneField), avatar (ImageField), words_num_in_prof (IntegerField), show_pie_chart (BooleanField).
- Зв’язки: Один до одного з User.

5. UserLogin (Логіни користувача):

- Призначення: Відстежує дати входу користувачів для аналізу активності.
- Ключові атрибути: user (ForeignKey), date (DateField).
- Зв’язки: Один до багатьох із User.

6. Friendship (Дружні зв’язки):

- Призначення: Керує запитами на дружбу між користувачами та їхнім статусом.
- Ключові атрибути: sender (ForeignKey), receiver (ForeignKey), status (CharField).
- Зв’язки: Один до багатьох із User (для sender та receiver).

7. ReadingText (Текст для читання):

- Призначення: Зберігає тексти англійською мовою з перекладами слів для практики читання.
- Ключові атрибути: title (CharField), content (TextField), words_with_translations (JSONField), eng_level (CharField).
- Зв’язки: Відсутні.

8. Achievement (Досягнення):

- Призначення: Містить інформацію про досягнення, які можуть отримати користувачі.
- Ключові атрибути: name (CharField), description (TextField), level (PositiveIntegerField).
- Зв’язки: Багато до багатьох із User через UserAchievement.

9. UserAchievement (Досягнення користувача):

- Призначення: Пов’язує користувачів із їхніми отриманими досягненнями.
- Ключові атрибути: user (ForeignKey), achievement (ForeignKey), time_get (DateTimeField).
- Зв’язки: Один до багатьох із User, один до багатьох із Achievement.

10. Category (Категорія):

- Призначення: Використовується для класифікації рейтингів користувачів.
- Ключові атрибути: name (CharField), last_update (DateTimeField).
- Зв’язки: Один до багатьох із Top.

11. Notification (Сповіщення):

- Призначення: Зберігає сповіщення для користувачів, наприклад, про нові дружні зв’язки чи досягнення.
- Ключові атрибути: user (ForeignKey), message (TextField), is_read (BooleanField).
- Зв’язки: Один до багатьох із User.

12. Top (Рейтинг):

- Призначення: Відображає рейтинги користувачів за різними категоріями.
- Ключові атрибути: category (ForeignKey), user (ForeignKey), place (PositiveIntegerField), points (PositiveIntegerField).
- Зв’язки: Один до багатьох із Category, один до багатьох із User.

Для наочності розроблено фрагмент реляційної моделі даних, який включає ключові сутності та їхні зв’язки. На рисунку 2.8 представлено діаграму, що ілюструє взаємозв’язки між сутностями User, Word, WordGroup, UserProfile, Friendship, ReadingText, Achievement, UserAchievement, Category, Notification та Top.

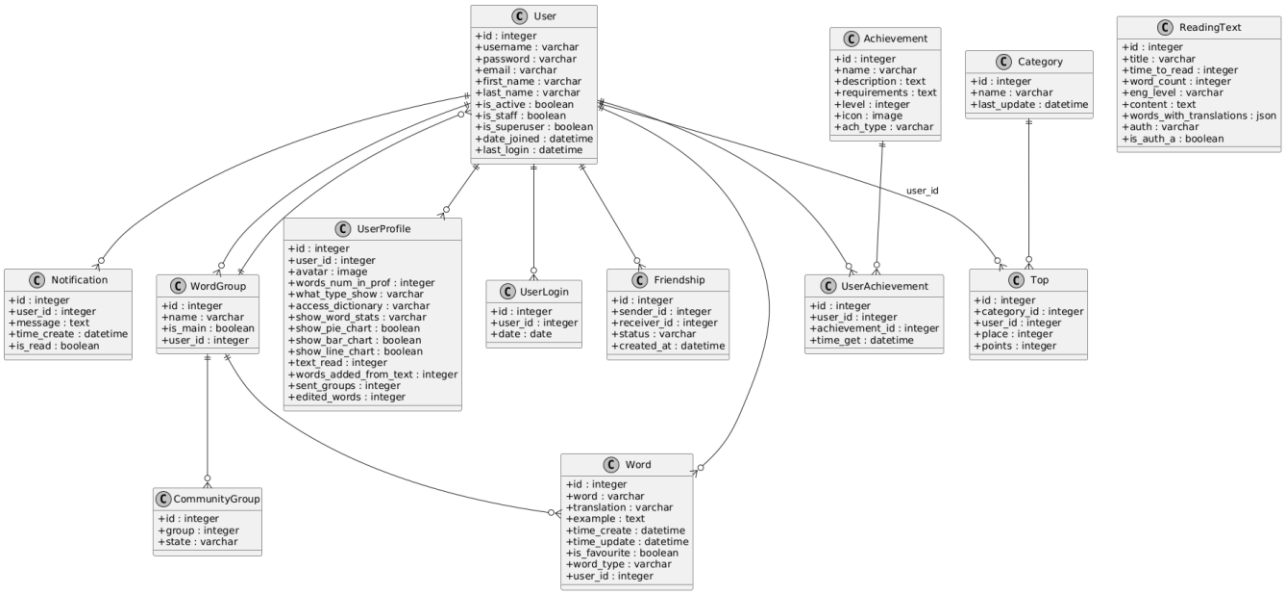


Рисунок 2.8 – Фрагмент реляційної моделі даних для веб-ресурсу «Вивчення англійської мови»

База даних реалізована за допомогою Django ORM, що забезпечує абстракцію над SQL-запитами та спрощує взаємодію з SQLite. Моделі, описані в models.py, автоматично створюють відповідні таблиці в базі даних із урахуванням визначених полів, індексів та зв’язків.[12]

2.5 Висновки

У даному розділі було здійснено всебічне проєктування структури та функціональних компонентів програмного продукту, що реалізує можливість вивчення англійської мови у веб-середовищі. На основі аналізу вимог до системи, її функціонального призначення та особливостей цільової аудиторії, було обґрунтовано вибір клієнт-серверної архітектури з трирівневою структурою. Такий підхід забезпечує логічне розмежування обов’язків між клієнтською частиною, серверною логікою та рівнем обробки даних, що в результаті підвищує гнучкість і масштабованість системи.

Серверна частина додатку реалізована за допомогою фреймворку Django, який забезпечує високу швидкість розробки, чітке структурування логіки та наявність вбудованих механізмів безпеки, авторизації, маршрутизації, кешування та роботи з формами. Це дозволяє сконцентрувати зусилля на розробці предметно-орієнтованого функціоналу, зокрема — механізмів роботи з лексичними одиницями, текстами, досягненнями та соціальною взаємодією між користувачами.

Особливу увагу приділено розробці UML-діаграм, які дозволили формалізувати структуру та поведінку системи, візуалізувати сценарії взаємодії користувачів з додатком та визначити ролі ключових компонентів. Діаграми прецедентів, послідовностей, розгортання та компонентів сприяли побудові цілісної логічної моделі додатку.

Крім того, були визначені основні алгоритми, що забезпечують функціонування основних сценаріїв використання — додавання нового слова, роботу з текстами, редагування профілю, а також управління соціальними

зв’язками між користувачами. Ці алгоритми спрямовані на забезпечення персоналізованого та зручного навчального процесу, що поєднує інтерактивність із системністю подачі навчального матеріалу.

Окремо було спроєктовано структуру бази даних, яка охоплює основні сутності, необхідні для повноцінної реалізації функціоналу: слова, групи, тексти, профілі користувачів, досягнення, рейтинги та сповіщення. Вибір реляційної бази даних SQLite є обґрунтованим рішенням для стартового етапу розробки, оскільки ця СКБД забезпечує швидку інтеграцію з Django ORM, має низький поріг входу та не потребує складного конфігурування.

Таким чином, проєктування додатку заклало надійне підґрунтя для наступного етапу — програмної реалізації, на якому спроєктовані компоненти будуть інтегровані в єдину систему для забезпечення повноцінної роботи додатку у реальному середовищі.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ WEB-ДОДАТКУ ДЛЯ ВИВЧЕННЯ АНГЛІЙСЬКОЇ МОВИ

3.1 Обґрунтування вибору мови програмування розробки WEB-додатку для вивчення англійської мови

У процесі розробки веб-додатку для вивчення англійської мови особливу увагу було приділено вибору відповідної мови програмування та супутніх технологій. З урахуванням вимог проєкту, таких як висока швидкість розробки, стабільність, розширюваність та безпека, для реалізації серверної частини було обрано мову Python у поєднанні з високорівневим фреймворком Django, а для клієнтської частини — стандартний стек веб-технологій: HTML, CSS та JavaScript.

Мова програмування Python вирізняється лаконічним та зрозумілим синтаксисом, що сприяє зниженню складності розробки й підвищенню читабельності коду. Django, як один із найпопулярніших веб-фреймворків для Python, забезпечує швидке створення функціональних веб-додатків завдяки вбудованим інструментам для роботи з базами даних, авторизації, маршрутизації, формами та захистом від типових веб-загроз (SQL-ін’єкцій, XSS тощо). У межах цього проєкту Django використовується для обробки запитів, управління словниковими записами, ведення статистики навчання, роботи з досягненнями та організації соціальної взаємодії між користувачами. Підтримка Django ORM дозволяє легко працювати з базою даних (у даному випадку SQLite), а бібліотека Celery — реалізовувати фонову обробку задач, як-от надсилання сповіщень чи оновлення статистики. [13]

Python також має переваги з точки зору кросплатформеності, що значно полегшує процес розгортання та тестування додатку на різних операційних системах. Завдяки широкій підтримці спільноти та великій кількості документації, виникаючі технічні труднощі можна вирішити оперативно, що позитивно впливає на загальні терміни реалізації проєкту.

Для забезпечення інтерфейсної частини застосовано HTML для структурування сторінок, CSS — для оформлення зовнішнього вигляду та адаптивного дизайну, і JavaScript — для реалізації динамічної взаємодії з користувачем. Зокрема, HTML дає змогу формувати сторінки авторизації, профілю, перегляду словника тощо. CSS (із залученням бібліотеки Bootstrap) забезпечує коректне відображення елементів на різних пристроях, що є критично важливим для користувацького досвіду. JavaScript, у свою чергу, використовується для реалізації інтерактивних елементів, таких як спливаючі підказки, асинхронна підвантажка даних (через AJAX) або динамічне оновлення сторінки без перезавантаження. Такий підхід забезпечує плавну і швидку взаємодію з додатком у реальному часі.[14]

Під час аналізу альтернатив для серверної частини розглядалися мови PHP (у поєднанні з фреймворком Laravel) та JavaScript (на базі Node.js). Попри широке застосування PHP, дана мова не забезпечує такої гнучкості та інтегрованості, як Python із Django, особливо у контексті побудови великих структурованих систем. Node.js, хоч і підходить для реалізації асинхронних додатків, потребує додаткових зусиль при налаштуванні базової безпеки, обробці форм, авторизації та реалізації ORM. Django ж надає повний набір необхідних інструментів «із коробки», що дозволяє зосередитись на логіці предметної області, а не на низькорівневій конфігурації. [15]

Порівняльна характеристика трьох мов програмування, що розглядалися для реалізації серверної частини, наведена у таблиці 3.1. З огляду на викладене, було здійснено порівняння Python із Django та альтернативних рішень — PHP з Laravel і JavaScript із Node.js — за низкою ключових критеріїв, таких як простота синтаксису, наявність бібліотек, рівень безпеки, інтеграція з базами даних та підтримка асинхронної обробки. Це дозволило об'єктивно оцінити можливості кожної мови у контексті створення серверної частини освітнього веб-додатку.

Таблиця 3.1 – Порівняння мов програмування для серверної частини

Характеристика	Python (Django)	PHP (Laravel)	JavaScript (Node.js)
Простота синтаксису	Висока	Середня	Середня
Екосистема бібліотек	Широка	Широка	Широка
Вбудовані механізми безпеки	Високий рівень	Середній рівень	Низький (зовнішні модулі)
Швидкість розробки	Висока	Середня	Середня
Інтеграція з базами даних	Проста (Django ORM)	Середня	Складна
Асинхронні задачі	Підтримується (Celery)	Середня (Queue)	Підтримується
Спільнота та документація	Висока	Висока	Висока
Масштабованість	Висока	Середня	Висока

Щодо клієнтської частини, альтернативно було проаналізовано можливість використання сучасних JavaScript-фреймворків — React та Vue.js. Проте в межах даного проєкту, зважаючи на відносно невелику складність інтерфейсу, використання стандартного стеку HTML/CSS/JavaScript у поєднанні з шаблонною системою Django є цілком достатнім і економічно доцільним рішенням. Перевагою такого підходу є легкість інтеграції, швидкість розробки та зменшення залежності від зовнішніх бібліотек. [16]

Порівняльну характеристику інструментів для клієнтської частини наведено в таблиці 3.2.

Мову програмування Python разом із фреймворком Django було обґрунтовано як оптимальний вибір для серверної частини веб-додатку, оскільки це рішення поєднує простоту, гнучкість, високу швидкість розробки та наявність вбудованих засобів безпеки. Для реалізації клієнтської частини найбільш раціональним виявилося використання стандартного HTML/CSS/JavaScript-стеку, що забезпечує достатню інтерактивність,

сумісність і швидке створення зручного інтерфейсу користувача без надмірної складності. У комплексі обрані інструменти дозволяють ефективно реалізувати навчальний функціонал додатку з мінімальними витратами ресурсів і часу.[17]

Таблиця 3.2 – Порівняння інструментів для клієнтської частини WEB-додатку

Характеристика	HTML/CSS/JavaScript	React	Vue.js
Простота реалізації	Висока	Середня	Середня
Сумісність із браузерами	Висока	Висока	Висока
Інтерактивність	Середня (з jQuery)	Висока	Висока
Швидкість розробки	Висока	Середня	Середня
Інтеграція з Django	Пряма (через шаблони)	Посередня (через API)	Посередня (через API)
Розмір бібліотеки	Малий (з Bootstrap)	Великий	Середній
Підтримка адаптивного дизайну	Висока	Висока	Висока

3.2 Обґрунтування вибору середовища розробки для WEB-додатку для вивчення англійської мови

Важливою складовою успішної розробки програмного забезпечення є вибір інструментів, які безпосередньо впливають на ефективність процесу створення продукту, його надійність, підтримуваність та швидкість розгортання. Одним із таких ключових інструментів є середовище розробки, або IDE (Integrated Development Environment), яке виконує роль базової платформи для написання, редагування, налагодження, запуску і тестування програмного коду. У ході проєктування і реалізації веб-додатку для вивчення

англійської мови було проаналізовано декілька популярних сучасних середовищ розробки з метою обрання найбільш придатного для потреб проєкту.

У результаті порівняльного аналізу було обрано Visual Studio Code — сучасне, легке та розширюване середовище розробки з відкритим кодом, створене корпорацією Microsoft. Цей вибір зумовлений його широкими можливостями, простотою у використанні, високою швидкодією, а також гнучкістю налаштувань, що дозволяє адаптувати середовище до конкретних потреб веброзробника. Visual Studio Code підтримує роботу з усіма ключовими мовами та технологіями, що використовувались у рамках проєкту: Python, HTML, CSS, JavaScript, а також забезпечує безшовну інтеграцію з такими інструментами, як Git, GitHub, Docker, віртуальне середовище Python (venv), а також із фреймворком Django.

Середовище Visual Studio Code забезпечило розробнику можливість одночасної роботи з серверною і клієнтською частинами системи в межах одного інтерфейсу. Інтегрований термінал, потужна система підсвічування синтаксису, підтримка автодоповнення (через розширення Pylance та Python extension), а також можливість запуску скриптів безпосередньо в середовищі значно підвищили швидкість і зручність реалізації функціоналу. Під час розробки особливо корисними виявилися вбудовані засоби налагодження (debugger), які дозволяли швидко локалізувати помилки у логіці додатку, перевіряти роботу представлень, моделей, форм і взаємодії з базою даних. Система контролю версій Git, інтегрована у VS Code, дала змогу організувати безпечну роботу з проєктом, зберігати всі етапи змін, а також проводити тестування без ризику втрати даних.

Окрім Visual Studio Code, було також розглянуто альтернативні середовища розробки, які потенційно могли бути використані в межах цього проєкту. Серед них: PyCharm Community Edition — спеціалізоване середовище для розробки на Python, Sublime Text — легкий текстовий редактор із розширюваними можливостями, а також Atom — гнучкий редактор коду з

відкритим вихідним кодом. Кожне з цих середовищ має свої переваги, проте жодне не продемонструвало такої гнучкості, швидкодії та сумісності з усіма використаними у проєкті технологіями, як Visual Studio Code.

Зокрема, PyCharm, хоча й забезпечує потужну підтримку Django, має суттєве обмеження — повний набір функцій доступний лише у платній версії (Professional Edition), тоді як безкоштовна Community Edition має обмежену підтримку вебтехнологій, що унеможливлює зручну роботу з HTML/CSS та JavaScript у межах одного середовища. Sublime Text, попри надзвичайно високу швидкість роботи, не має повноцінної інтеграції з інструментами Python та Django без встановлення великої кількості сторонніх плагінів, що ускладнює налаштування. Atom, незважаючи на зручний інтерфейс, значно поступається іншим середовищам у швидкодії та стабільності, особливо при роботі з великими проєктами та складними структурами файлів.

У таблиці 3.3 наведено порівняльну характеристику найбільш поширених середовищ розробки, які були розглянуті на етапі проєктування.

Для повного і об'єктивного аналізу були визначені ключові критерії, за якими проведено оцінювання кожного із розглянутих середовищ. До таких критеріїв належать: повнота підтримки основних технологій (Python, Django, HTML/CSS/JavaScript), наявність автодоповнення коду, інтеграція з системами контролю версій, підтримка плагінів, стабільність роботи та зручність налаштування середовища відповідно до потреб розробки. Оцінка середовищ за цими параметрами дала змогу систематизувати інформацію та аргументовано зробити остаточний вибір середовища розробки, який представлено в таблиці 3.3.

Таблиця 3.3 – Порівняння середовищ розробки для створення WEB-додатку

Характеристика	Visual Studio Code	PyCharm Community	Sublime Text	Atom
Підтримка Python/Django	Повна (через розширення)	Часткова	Часткова	Часткова
Інтеграція з Git	Так	Так	Через плагіни	Так
Автодоповнення коду	Так (через Pylance)	Так	Так	Так
Вбудований термінал	Так	Так	Ні	Так
Швидкодія	Висока	Середня	Висока	Низька
Підтримка розширень	Дуже широка	Обмежена	Помірна	Помірна
Зручність налаштування	Висока	Висока	Середня	Середня
Безкоштовна версія	Так	Так (без вебпідтримки)	Так	Так
Підтримка фронтенд-технологій	Повна	Часткова	Обмежена	Повна

Таким чином, Visual Studio Code було обґрунтовано як найкращий вибір середовища розробки для створення веб-додатку освітнього призначення. Це середовище забезпечує високий рівень зручності роботи, широкі можливості для розширення функціоналу, підтримку ключових технологій проєкту, а також дозволяє розробнику повністю зосередитися на реалізації навчального функціоналу без потреби у складному технічному конфігуруванні.

3.3 Реалізація програмних модулів WEB-додатку для вивчення англійської мови

3.3.1 Реалізація асинхронних задач із Celery, Redis і Docker

Розробка додатку для вивчення англійської мови включає в себе використання широкого спектру сучасних технологій, що забезпечують високий рівень продуктивності та зручності користування. Важливою складовою частиною такого додатку є ефективна обробка запитів та задач, а також підтримка масштабованості системи при збільшенні навантаження. Одним із ключових аспектів у реалізації цього є використання системи обробки фонових задач, що дозволяє ефективно управляти завданнями, які потребують значних ресурсів або мають бути виконані асинхронно. Для цього в моєму проекті було обрано використання Celery — популярної системи для обробки асинхронних задач, у поєднанні з Redis, який виконує роль брокера повідомлень.

Для оптимізації цієї інфраструктури та забезпечення стабільної роботи, я застосував технологію контейнеризації через Docker. Це дозволяє ізолювати сервіси в окремі контейнери, що знижує залежність між компонентами системи та полегшує її налаштування і обслуговування. Зокрема, для Redis, який є необхідним компонентом у процесі роботи Celery, був використаний окремий Docker-контейнер. Це дозволяє легко налаштовувати Redis, масштабувати систему та забезпечити її стабільність при зростанні кількості користувачів.

Варто зазначити, що хоча Docker Compose зазвичай використовується для організації взаємодії між кількома контейнерами, в моєму проекті не було потреби в ньому, оскільки система на даному етапі включає лише один контейнер для Redis. Такий підхід дозволяє спростити інфраструктуру без додаткових складнощів, що можуть виникнути при використанні Docker Compose для управління кількома контейнерами одночасно. Однак у разі необхідності додавання нових сервісів чи компонентів у майбутньому, цей

підхід можна легко розширити, без необхідності переписувати основну архітектуру додатку.

Використання Docker і Redis у поєднанні з Celery дозволяє забезпечити асинхронну обробку задач, що є важливим аспектом для таких функцій, як збереження прогресу користувачів, обробка запитів на реєстрацію, надсилання сповіщень та забезпечення інтерактивного навчання. Такий підхід сприяє створенню масштабованого, надійного та зручного додатку для вивчення англійської мови. [18]

Однією з головних переваг використання Docker у цьому проєкті є можливість швидко та без зайвих складнощів запускати необхідні сервіси. Для того щоб налаштувати Redis та Celery, достатньо виконати кілька простих команд у терміналі. Це дозволяє зекономити час і спростити процес налаштування, особливо на різних етапах розробки, тестування та розгортання.

Щоб запустити контейнер з Redis, необхідно виконати наступну команду:

```
docker run -d --name redis-server -p 6379:6379 redis
```

Для того щоб запустити Celery worker, потрібно виконати таку команду в терміналі:

```
celery -A mad worker -l info
```

Celery Worker — це процес, який виконує завдання, що передаються йому через систему повідомлень. Він є основною частиною Celery і відповідає за обробку фонових асинхронних задач.

Worker чекає на задачі, що надходять до нього через брокер повідомлень (у нашому випадку це Redis). Коли задача надходить, worker отримує її, обробляє і повертає результат (якщо це потрібно). Worker може виконувати різні типи задач, наприклад, відправку електронних листів, обробку великих файлів, виконання довготривалих операцій або збереження даних.

Завдяки worker, задачі можуть виконуватися у фоновому режимі, без блокування основного потоку програми. Це дозволяє веб-додатку працювати швидше і не гальмувати при виконанні важких операцій. Кілька worker-ів

можуть працювати паралельно, що дозволяє обробляти великий обсяг задач одночасно, підвищуючи ефективність і зменшуючи час обробки. [19]

Для того щоб запустити Celery beat, який відповідає за періодичне виконання задач, потрібно виконати команду:

celery -A mad beat -l info

Celery Beat — це компонент, який займається плануванням задач. Якщо ви хочете, щоб якісь завдання виконувались регулярно (наприклад, кожні 5 хвилин або один раз на день), для цього використовується Celery Beat.

Beat працює як планувальник задач. Він визначає, коли і які задачі повинні бути виконані, і передає їх worker-у для обробки. Beat має таблицю з планами задач, де зазначено час і інтервал для виконання кожної задачі. Наприклад, якщо ви хочете кожні 10 хвилин оновлювати статистику користувачів або відправляти сповіщення, Beat це забезпечить.

Задачі, які мають виконуватись за розкладом, можна автоматизувати. Це дозволяє зекономити час і ресурси, не потребуючи постійного втручання розробника. Beat дозволяє задати чіткий інтервал виконання задач, що дуже зручно для періодичних операцій без потреби вручну запускати їх.

Для контролю за роботою Celery worker і beat можна використовувати Flower — веб-інтерфейс для моніторингу. Щоб запустити Flower, потрібно виконати команду в іншому терміналі:

celery -A mad flower

Flower дає змогу моніторити стан worker-ів: ви можете бачити, які worker-и зараз активні, які задачі вони обробляють, які задачі виконано, а які знаходяться в черзі. Flower також дозволяє керувати задачами: можна перервати або перепочати певну задачу. У Flower є графіки і статистика, що дають змогу відслідковувати ефективність роботи вашої системи та вчасно помічати проблеми. [20]

3.3.2 Реалізація серверної частини додатку на Django

Серверна частина додатку для вивчення англійської мови побудована на основі популярного фреймворку Django, який є одним із найефективніших інструментів для розробки веб-додатків на Python. Вибір Django обумовлений його потужними вбудованими можливостями, що значно спрощують розробку, підтримку та масштабування додатків. Django дозволяє легко інтегрувати в себе всі необхідні елементи, такі як аутентифікація користувачів, створення бази даних, обробка запитів від користувачів та створення API для взаємодії з іншими сервісами. Це забезпечує високу продуктивність, безпеку та масштабованість додатку, що важливо для будь-якої освітньої платформи з великою кількістю користувачів.

Завдяки вбудованій підтримці ORM (Object-Relational Mapping), Django дозволяє не турбуватись про створення SQL запитів, автоматично генеруючи необхідні таблиці в базі даних на основі визначених моделей. Для зберігання даних в проекті було обрано легковаговий SQL-движок SQLite, який є ідеальним вибором для невеликих і середніх за розміром додатків, зокрема для тестування та локального використання. Всі користувацькі дані, результати тестів, прогрес у вивченні слів та інші важливі дані зберігаються в базі, що дає змогу швидко їх обробляти та оновлювати.

Серверна частина також інтегрує Celery, що дозволяє виконувати асинхронні завдання, такі як обробка даних, відправка електронних листів або обчислення, що потребують часу. Використання Celery дозволяє відкласти складні операції на окремі треди, зберігаючи основний процес серверу швидким та ефективним. Зокрема, для зручності планування регулярних задач, таких як оновлення контенту або звітності, в проекті також використовується Celery Beat, який дозволяє налаштувати періодичні завдання для виконання у фоновому режимі. [21]

Що стосується взаємодії серверної частини з клієнтським інтерфейсом, то Django забезпечує гнучку систему рендерингу HTML через шаблони, використовуючи Jinja, що дозволяє створювати динамічні сторінки, на яких

користувач може взаємодіяти з платформою, проходити тести, переглядати статистику та тренувати свої знання. Jinja дозволяє ефективно вставляти змінні, управляти умовами та циклами в HTML-шаблонах, що робить створення інтерфейсу зручним та гнучким. З таким підходом вдається легко інтегрувати дані з бази та динамічно оновлювати контент сторінок без необхідності перезавантаження. Це важливо для забезпечення зручного та інтуїтивно зрозумілого користувацького досвіду на платформі для вивчення мови.

Завдяки цим інструментам серверна частина проекту стає потужною та надійною, здатною ефективно обробляти великі обсяги запитів від користувачів, виконувати складні операції у фоновому режимі та забезпечувати стабільну та безпечну роботу платформи.

Перш за все варто розпочати з моделей проекту. Моделі в Django — це основний елемент, який відповідає за взаємодію з базою даних. Вони дозволяють нам визначати структуру таблиць у базі даних та правила зберігання інформації. Кожен клас моделі в Django є підкласом django.db.models.Model, і визначає поля, які мають бути збережені в таблиці, а також різноманітні методи для роботи з цими даними. Крім того, моделі в Django інтегровані з іншими важливими компонентами фреймворку, як-от форми, адміністративний інтерфейс і т. д.

Ключова ідея при роботі з моделями в Django — це визначення класу для кожної сутності, яку ви хочете зберігати в базі даних. Моделі автоматично створюють таблиці в базі, відповідні полям класів. Моделі є частиною того, що в Django називається "ORM" (Object-Relational Mapping), тобто перетворення об'єктів у базу даних.

У нашому проекті ми працюємо над створенням веб-додатку для управління персональним словником, де користувачі можуть зберігати слова, додавати їхні переклади, приклади використання та інші додаткові дані. Ключові функції додатку включають:

- Зберігання слів і їх перекладів.

- Визначення типів слів (іменник, дієслово тощо).
- Відзначення слів як улюблених для швидкого доступу.
- Можливість додавання прикладів використання слів у контексті.

Ось як було реалізовано модель Word для зберігання слів в базу даних:

```
from django.db import models
class Word(models.Model):
    TYPE_CHOICES = [
        ('noun', 'Noun'),
        ('verb', 'Verb'),
        ('adjective', 'Adjective'),
        ('adverb', 'Adverb'),
        ('pronoun', 'Pronoun'),
        ('phrasal verb', 'Phrasal verb'),
        ('other', 'Other')
    ]
    word = models.CharField(max_length=100)
    translation = models.CharField(max_length=100)
    example = models.TextField(blank=True)
    time_create = models.DateTimeField(auto_now_add=True)
    time_update = models.DateTimeField(auto_now=True)
    is_favourite = models.BooleanField(default=False)
    word_type = models.CharField(max_length=20, choices=TYPE_CHOICES, default='other')
    def __str__(self):
        return self.word
```

word: Основне слово, яке користувач хоче зберегти. Тип поля CharField забезпечує обмеження довжини для уникнення введення занадто довгих слів.

- translation: Переклад цього слова на іншу мову. Також є текстовим полем, щоб підтримувати короткі та точні переклади.
- example: Це поле дає можливість зберігати приклади використання слова в контексті. Це корисно для кращого розуміння, як слово може бути використане у реченні.

- time_create: Поле, яке автоматично заповнюється датою та часом створення запису. Це дає змогу відслідковувати, коли саме було додано слово.
- time_update: Поле для автоматичного оновлення часу, коли слово було змінено. Це корисно для відстеження змін.
- is_favourite: Це булеве поле дозволяє користувачам відзначати свої улюблені слова. Це додає зручності при пошуку та доступі до важливих слів.
- word_type: Тип слова, що дозволяє класифікувати його як іменник, дієслово тощо. Це поле використовує параметр choices, що обмежує можливі значення, забезпечуючи

У цьому проєкті концепція *стріку* (streak) стосується певного показника активності користувача, який відображає регулярність його дій на платформі. Наприклад, можна рахувати кількість днів, поспіль у які користувач додає нові слова або виконує певні завдання, такі як збереження перекладів, груп слів або інших дій, передбачених функціоналом додатку.

Для реалізації стріку необхідно відслідковувати кожну активність користувача і порівнювати її з попередніми діями. Наприклад, кожного разу, коли користувач додає нове слово система фіксує дату цієї активності. Якщо наступного дня після попереднього оновлення користувач виконує дію, стрік збільшується. Якщо ж він пропускає день, стрік обнуляється, і все починається заново.

Такий підхід дозволяє не тільки відслідковувати активність користувачів, але й мотивувати їх підтримувати регулярність у виконанні дій. Візуально це відображається в профілі, щоб користувачі бачили свій прогрес і мали додаткову мотивацію не переривати стрік. Цей механізм є потужним інструментом для залучення та утримання користувачів на платформі.

Досягнення в цьому проєкті також виконують важливу роль у створенні мотивації для користувачів і стимулюванні їх активності на платформі. Вони являють собою певні цілі або етапи, які користувач може досягти в процесі використання сайту. Ці досягнення пов'язані з різними діями: додаванням нових слів, створенням груп слів, досягненням певного рівня активності,

виконанням завдань як, наприклад, збереження певної кількості перекладів з текстів.

Однією з основних функцій досягнень є надання користувачам відчуття прогресу та досягнення мети. Це важливо, тому що на платформі для вивчення мови або роботи з даними користувачам може бути складно зрозуміти, наскільки ефективними є їх зусилля. Завдяки досягненням вони бачать конкретні результати своєї роботи, що додає їм впевненості та бажання продовжувати.

Для досягнень також є окрема модель Achievement. І окрема модель яка відповідає за збереження досягнень відповідно до користувачів UserAchievement. Тобто збереження конкретних досягеннь конкретних користувачів.

```
class Achievement(models.Model):
ACH_TYPE_CHOICES = [
('1', 'Words'),
('2', 'Groups'),
('3', 'Friends'),
('4', 'Reading'),
('5', 'Interaction'),
('6', 'Content Quality'),
('7', 'Special')
]
name = models.CharField(max_length=100)
description = models.TextField()
requirements = models.TextField(blank=True)
level = models.PositiveIntegerField()
icon = models.ImageField(upload_to='achievements/', blank=True)
ach_type = models.CharField(max_length=20, choices=ACH_TYPE_CHOICES)
def get_icon_url(self):
if self.icon:
return f'/media/achievements/{self.icon.name.split("/")[-1]}'
return self.name
def __str__(self):
```

```
return f"{self.name} (Level {self.level})"
class UserAchievement(models.Model):
user = models.ForeignKey(User, on_delete=models.CASCADE)
achievement = models.ForeignKey(Achievement, on_delete=models.CASCADE)
time_get = models.DateTimeField(auto_now_add=True)
def __str__(self):
return f"{self.user.username} - {self.achievement.name}"
class Meta:
ordering = ['-time_get']
indexes = [
models.Index(fields=['-time_get'])
]
```

У проєкті крім моделі, що відповідає за збереження та обробку даних, важливу роль відіграють представлення (views). Вони відповідають за відображення інформації користувачу та обробку його запитів. У файлі views.py зберігаються всі функції або класи, які визначають, як обробляти запити, які надходять від користувача, і як саме ці дані будуть відображатися на веб-сторінці. Представлення мають на меті зв'язок між моделями і шаблонами, виконуючи обробку бізнес-логіки та передаючи результат у вигляді HTML-сторінок.

Крім базових представлень, що просто передають дані з моделі на сторінку, є й більш складні варіанти представлень, які включають додаткові функціональні можливості. Наприклад, представлення, що обробляють додавання або редагування елементів, підрахунок досягнень користувачів або реалізація фільтрації й сортування даних. Особливу увагу варто приділити представленням, що взаємодіють з файлом або базою даних для збереження змін, як-от додавання нових слів, груп слів або перегляд статистики користувачів. [22]

Для того щоб автоматично оновлювати досягнення при зміні даних користувача, функції використовують сигнали Django. Наприклад, функції update_achievements_words, update_achievements_on_group_words_change, update_achievements_friends, і update_achievements_reading спрацьовують на

події збереження або змін у моделях, таких як створення слів, зміни в групах, додавання друзів або редагування профілю користувача. Це дозволяє оперативно оновлювати досягнення без необхідності додаткових запитів від користувача, що підвищує зручність і ефективність системи. Це відбувається за допомогою декоратора @receiver.

```
@receiver(post_save, sender=Word)
def update_achievements_words(sender, instance, **kwargs):
thresholds = [10, 50, 100]
process_words_achivments(instance.user, thresholds)
process_special_achivments(instance.user)
process_interaction_achivments(instance.user)

@receiver(post_save, sender=WordGroup)
def update_achievements_on_group_words_change(sender, instance, **kwargs):
thresholds = [1, 5, 10]
process_achievements(instance.user, achievement_type='2', thresholds=thresholds)
process_special_achivments(instance.user)
process_interaction_achivments(instance.user)

@receiver(m2m_changed, sender=WordGroup.words.through)
def update_achievements_on_group_words_change(sender, instance, action, **kwargs):
if action in ['post_add', 'post_remove', 'post_clear']:
thresholds = [1, 5, 10]
process_achievements(instance.user, achievement_type='2', thresholds=thresholds)
process_special_achivments(instance.user)
process_interaction_achivments(instance.user)

@receiver(post_save, sender=Friendship)
def update_achievements_friends(sender, instance, **kwargs):
thresholds = [5, 20, 50]
process_achievements(instance.sender, achievement_type='3', thresholds=thresholds)
process_achievements(instance.receiver, achievement_type='3', thresholds=thresholds)

@receiver(post_save, sender=UserProfile)
```

```
def update_achievements_reading(sender, instance, **kwargs):
thresholds = [(5, 1), (50, 10), (100, 20)]
process_achievements(instance.user, achievement_type='4', thresholds=thresholds)
process_special_achivments(instance.user)
process_interaction_achivments(instance.user)
```

Ці функції працюють разом, щоб забезпечити ефективне та автоматичне присвоєння досягнень користувачам на основі їхніх дій на платформі, при цьому дозволяючи змінювати рівень досягнень і підтримувати актуальність інформації.

Крім того з важливих функцій, що повинна бути доступна за преміум підпискою це експорт всіх слів словника у pdf файл. Вона реалізована у функції export_pdf. Ця функція є важливою, оскільки вона дозволяє користувачеві експортувати його словник у формат PDF. Це може бути корисним у багатьох ситуаціях, коли потрібно зберігати або друкувати особисті дані, наприклад, для перегляду без підключення до Інтернету або для архівування. [23]

```
def export_pdf(request):

if not request.user.user_profile.is_premium:
return redirect('soon')
words = Word.objects.filter(user=request.user)
response = HttpResponse(content_type='application/pdf')
response['Content-Disposition'] = f'attachment;
filename="{request.user.username}_dictionary.pdf"'
pdf = canvas.Canvas(response, pagesize=letter)
width, height = letter
left_margin = 50
right_margin = 50
max_width = width - left_margin - right_margin
y = height - 50

pdf.setFont("DejaVuSans", 12)
pdf.drawString(left_margin, y, f"Dictionary of {request.user.username}")
y -= 30
```

```
        pdf.setFont("DejaVuSans", 10)
        pdf.drawString(left_margin, y, "-" * 50)
        y -= 20

        for word in words:
        pdf.setFont("DejaVuSans-Bold", 10)
        pdf.drawString(left_margin, y, f"Word: {word.word}")
        y -= 15

        pdf.setFont("DejaVuSans", 10)
        translation_lines = wrap_text(f"Translation: {word.translation}", max_width,
"DejaVuSans", 10, pdf)
        for line in translation_lines:
        pdf.drawString(left_margin, y, line)
        y -= 15

        example_lines = wrap_text(f"Example: {word.example}", max_width, "DejaVuSans", 10,
pdf)
        for line in example_lines:
        pdf.drawString(left_margin, y, line)
        y -= 15

        y -= 10

        if y < 50:
        pdf.showPage()
        pdf.setFont("DejaVuSans", 10)
        y = height - 100

        pdf.save()
        return response
```

Перше, що виконується у функції — це перевірка, чи є у користувача преміум-статус. Якщо користувач не є преміум, його автоматично перенаправляють на сторінку, де він дізнається, що доступ до цієї функції йому

заблоковано. Це забезпечує монетизацію функції і дозволяє обмежити доступ до неї лише для тих, хто має платний акаунт.

Функція створює PDF-документ, в якому містяться дані про кожне слово, збережене в словнику користувача. Вся інформація, включаючи слово, переклад і приклад використання, відображається на окремих рядках. Для зручності використовується шрифт DejaVuSans, а сам документ форматується таким чином, щоб текст не виходив за межі сторінки.

Функція також використовує допоміжну функцію wrap_text, яка автоматично розбиває довгі рядки на декілька коротших, щоб текст не виходив за межі сторінки. Це важливо для збереження належного форматування і читаємості документу.

Нарешті, після виконання всіх етапів формування PDF-документу, користувач отримує готовий файл для завантаження. Це дає йому можливість зберігати свої дані на комп'ютері або роздруковувати їх для подальшого використання, що робить цю функцію надзвичайно корисною для тих, хто хоче мати доступ до свого словника в зручному вигляді.

3.3.3 Реалізація логіки представлень та обробки запитів

У Django, окрім функцій представлень (views), також активно використовуються класи представлень (class-based views, CBV). Ці класи часто є підкласами вже існуючих класів, що дозволяє розширювати їх функціональність. Використання класів дає більше гнучкості й можливостей для організації коду. Вони дозволяють зменшити дублювання коду та забезпечують більш чисту та розширювану архітектуру проекту. Клас представлення у Django — це особливий спосіб обробки HTTP-запитів, що дозволяє структурувати код у вигляді методів, таких як get(), post(), put(), і т.д. Це дозволяє зручніше працювати з різними типами запитів у рамках однієї логічної одиниці. Використання класів також забезпечує гнучкість, оскільки можна легко створювати нові класи, що наслідують від базових класів, і перевизначати лише необхідні методи.

Клас ProfileView є прикладом представлення, яке наслідується від класів LoginRequiredMixin і View. Це дозволяє поєднати перевірку автентифікації та обробку HTTP-запитів у рамках одного класу. Він відповідає за відображення профілю користувача та всіх пов'язаних з ним даних.

LoginRequiredMixin забезпечує, щоб лише авторизовані користувачі могли отримати доступ до цього представлення. Якщо користувач не увійшов в систему, він буде перенаправлений на сторінку входу.

View надає базову структуру для обробки HTTP-запитів у класах. У класі ProfileView використовується метод get(), який відповідає за обробку GET-запитів

Метод get_user_data() викликається у класі для збору всіх даних, необхідних для відображення профілю користувача. Тут виконується кілька важливих операцій:

1. Перевірка наявності профілю користувача через UserProfile.objects.get_or_create(user=user), що дозволяє автоматично створювати профіль, якщо він ще не існує.
2. Збір інформації про слова користувача, зокрема улюблені слова, їх кількість, а також дані про групи слів, до яких ці слова належать.
3. Обчислення статистики за типами слів, що дозволяє зручно візуалізувати розподіл за категоріями.

Клас також обробляє взаємодії між користувачами. Через Friendship клас отримує інформацію про поточні дружні запити, відправлені та отримані, і передає ці дані в шаблон. Це важлива частина логіки для соціальних мереж, де користувачі можуть мати друзів і керувати своїми запитами. Завдяки цим даним користувач може бачити, чи є він другом іншого користувача, чи є незавершені дружні запити.

Клас обчислює статистику досягнень користувача, зокрема його стрики — періоди безперервної активності. Для цього використовуються методи, що розраховують поточний та найдовший стрик користувача на основі даних у профілі. Це додає елемент мотивації для користувача, оскільки він може

слідкувати за своїм прогресом та ставити нові цілі для покращення своєї активності.

Після збору всіх необхідних даних клас передає їх у шаблон для рендерингу. Тут використовуються дані про профіль користувача, його досягнення, статистику за словами, групи, друзі та багато іншого. Завдяки цьому, користувач отримує детальну і персоналізовану інформацію, що дозволяє йому краще розуміти свій прогрес на платформі.[24]

```
class ProfileView(LoginRequiredMixin, View):
def get(self, request, user_name, **kwargs):
def get_user_data(user):
user_profile, created = UserProfile.objects.get_or_create(user=user)
is_favorite = user_profile.what_type_show == 'fav'

recent_words = Word.objects.filter(
user=user,
is_favourite=is_favorite
)[:user_profile.words_num_in_prof] if is_favorite else
Word.objects.filter(user=user)[:user_profile.words_num_in_prof]

words = Word.objects.filter(user=user)
word_count = words.count()
group_count = max((WordGroup.objects.filter(user=user).count() +
WordGroup.objects.filter(uses_users=self.request.user).count()) - 1, 0)

friends = User.objects.filter(
Q(friendship_requests_sent__receiver=user, friendship_requests_sent__status='accepted') |
Q(friendship_requests_received__sender=user,
friendship_requests_received__status='accepted')
).distinct()
```

В Django для виконання асинхронних задач, як було згаданао раніше, часто використовують бібліотеку Celery. Це дозволяє відкладати виконання важких або тривалих задач до окремих воркерів, що не блокують основний потік запитів. Завдяки цьому додаток стає більш масштабованим і швидким,

оскільки не потрібно чекати виконання кожної задачі перед тим, як користувач отримає відповідь.

Celery дозволяє створювати таски, які є функціями, що можуть бути виконані асинхронно. Таски можуть виконуватися у фоновому режимі або за певним розкладом. Вони можуть бути відкладені на деякий час, чи виконуватись повторно через певні інтервали.

Функція send_activation_email, що викликається під час реєстрації користувача.

```
@app.task
def send_activation_email(user_id):
try:
user = User.objects.get(pk=user_id)
token = default_token_generator.make_token(user)
uid = urlsafe_base64_encode(force_bytes(user.pk))
activation_link = f"{settings.SITE_URL}{reverse('activate_account', kwargs={'uidb64': uid, 'token': token})}"

subject = "Activate your account"
message = f"Hello {user.username},\n\nPlease click the link below to activate your account:\n{activation_link}"
send_mail(subject, message, settings.DEFAULT_FROM_EMAIL, [user.email])
except User.DoesNotExist:
pass
```

Ця конкретна таска призначена для надсилання електронного листа з посиланням для активації акаунту користувача. Функція send_activation_email приймає параметр user_id, який ідентифікує користувача в системі. Цей параметр є унікальним для кожного користувача та використовується для отримання об'єкта користувача через User.objects.get(pk=user_id).

Генерація токену активації:

default_token_generator.make_token(user) генерує токен, який є тимчасовим кодом для активації акаунту. Токен забезпечує безпеку процесу активації акаунту.

urlsafe_base64_encode(force_bytes(user.pk)) кодує унікальний ідентифікатор користувача у форматі Base64 для використання в URL.

Формування активаційного посилання: За допомогою reverse формуємо URL для активації акаунту. В URL передаються два параметри: uidb64 (кодування ідентифікатора користувача) та token (генерований токен).

Надсилання електронного листа: За допомогою функції send_mail, відправляється повідомлення користувачу з темою "Activate your account" і посиланням для активації акаунту.

Ця таска є прикладом класичного використання Celery для асинхронного надсилання листів, що є важливим для оптимізації роботи веб-додатків.

Посилання підтвердження реєстрації має вигляд activate/<uidb64>/<token>/. Django надає потужну та гнучку систему маршрутизації URL, яка дозволяє зручно створювати та обробляти URL-шляхи для вашого веб-додатка. Вся система побудована на використанні конфігураційного файлу urls.py, де вказується, як саме різні URL запити будуть оброблятися відповідними представленнями (views).

Django дозволяє створювати динамічні URL-шляхи за допомогою змінних частин у шляху. Вони записуються в <> і можуть приймати значення, передані в URL запиті. Вищезгаданий шлях activate/<uidb64>/<token>/ є прикладом такого динамічного маршруту, де uidb64 і token є параметрами, які передаються в функцію представлення. У цьому випадку uidb64 і token є змінними, які представляють закодований ідентифікатор користувача та токен для активації акаунту. Після того, як URL запит приходить до серверу, Django передає ці параметри в функцію представлення, яка буде обробляти їх.

В наданому нижче масиві надано приклади деяких посилань проєкту.

```
urlpatterns = [
path('', views.index, name='home'),
```

```
    path('words/<slug:user_name>', views.WordListView.as_view(), name='words'),
    path('login/', views.LoginUser.as_view(), name='login'),
    path('register/', views.RegisterUser.as_view(), name='register'),
    path('profile/<slug:user_name>/', views.ProfileView.as_view(), name='profile'),
    path('make_favourite/<int:word_id>/', views.make_favourite, name='make_favourite'),
    path('respond-friend-request/<int:friendship_id>/<str:response>/',
views.respond_to_friend_request, name='respond_friend_request'),
    path('practice/reading/<int:text_id>/', views.reading_text_view, name='reading_text'),
    path('word/<int:text_id>/<str:word>/', views.word_detail_view, name='word_detail'),
    path('practice/groups/<int:group_id>/',          views.PracticeGroupWordsListView.as_view(),
name='group_words_practice'),
    path('download/<path:file>/', views.download_file, name='download_file'),
    path('soon/', views.soon_page, name='soon'),
    path('send_grooup_request/<int:group_id>/', views.send_group_request,
name='send_group_request'),
    path('approve_group_request/<int:group_id>/', views.approve_group_request,
name='approve_group_request'),
    path('delete_group_request/<int:group_id>/', views.reject_group_request,
name='delete_group_request'),
    path('export-pdf/', views.export_pdf, name='export_pdf'),
    path('activate/<uidb64>/<token>/', views.activate_account, name='activate_account'),
    path('tops/', views.tops_by_category, name='tops_by_category'),
    path('api/get-tops-by-category/', views.api_get_tops_by_category,
name='api_get_tops_by_category'),
    path('api/notifications/', views.notifications_api, name='notifications_api'),
    path('notification/<int:notification_id>/read/', views.mark_notification_as_read,
name='mark_notification_read'),
    ] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

3.4 Тестування WEB-додатку для вивчення англійської мови

Для початку варто розглянути панель адміністратора. Веб-ресурс використовує стандартну адміністративну панель, надану фреймворком Django, для ефективного управління вмістом і взаємодією з користувачами.

Панель адміністрування Django є потужним вбудованим інструментом, який дозволяє адміністраторам керувати записами бази даних через зручний інтерфейс без необхідності додаткової розробки для базових операцій CRUD (створення, читання, оновлення, видалення).

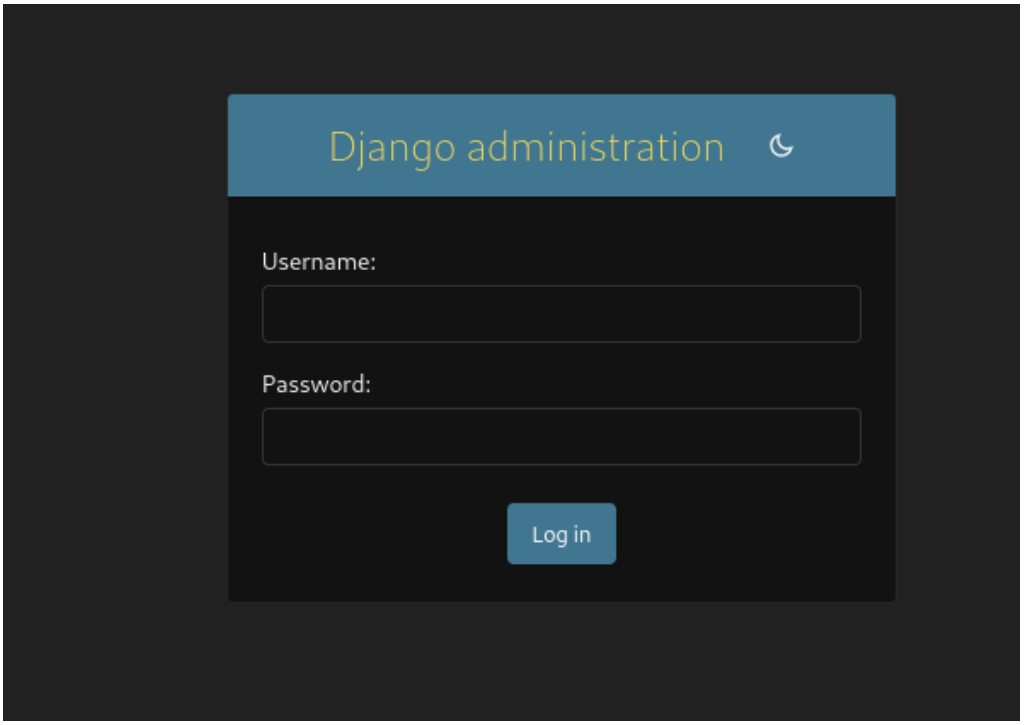


Рисунок 3.1 – Вхід користувача в адмінпанель WEB-ресурсу

Після входу користувач може переглядати та редагувати таку інформацію:

- користувачі;
- досягення;
- досягнення користувачів;
- дружні зв'язки;
- тексти для читання;
- топ користувачів по категоріям;
- слова;
- групи слів.

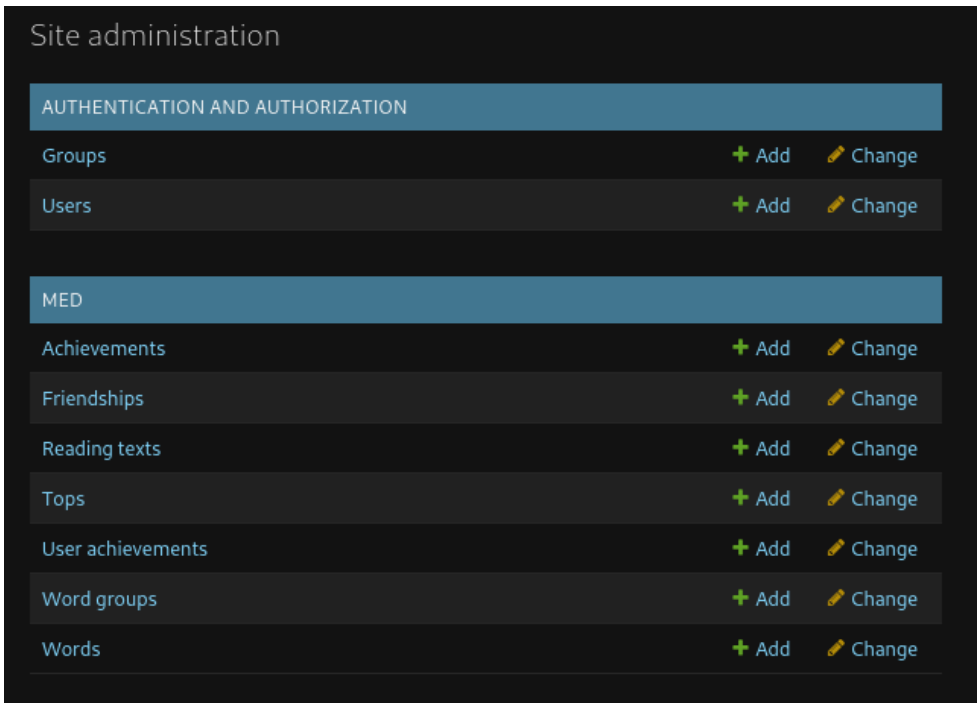


Рисунок 3.2 - Категорії інформації доступні адміністратору

Для зручності можна змінювати тему інтерфейсу нічна (темна) або денна (світла).

На кожній зі сторінок можна створювати нові екземпляри моделей, а також модифікувати та видаляти вже наявні.

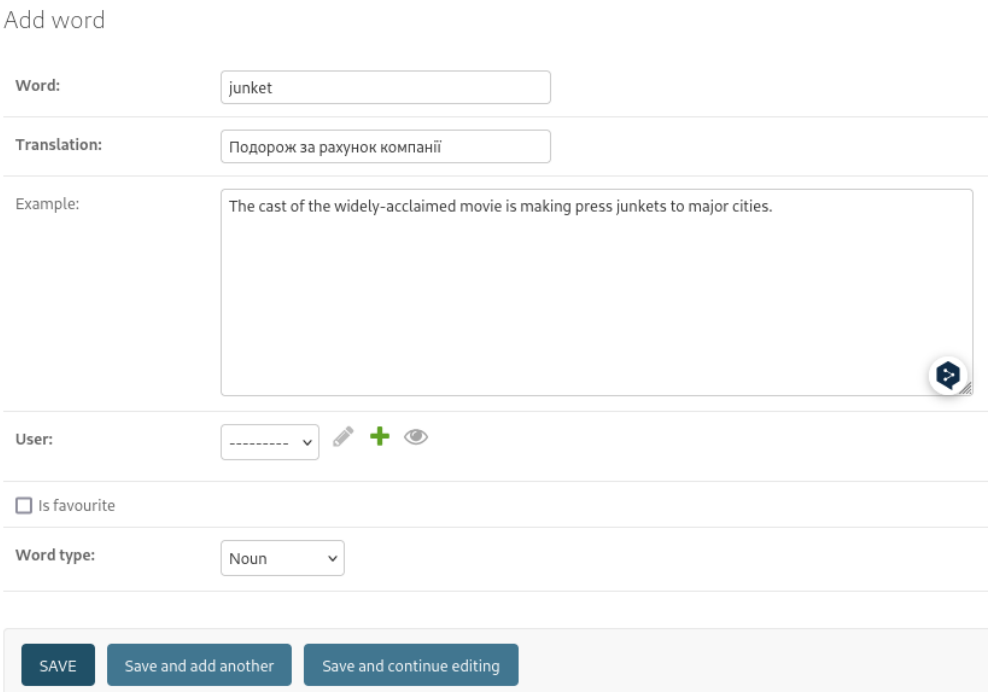


Рисунок 3.3 – Приклад створення нового слова

Після додавання нової інформації вона стане доступна на тій частині WEB-ресурсу, яка доступна для всіх клієнтів. Перш ніж увійти користувачу потрібно зареєструватись вказавши пошту, оскільки на неї буде відправлено лист, з підтвердженням реєстрації.

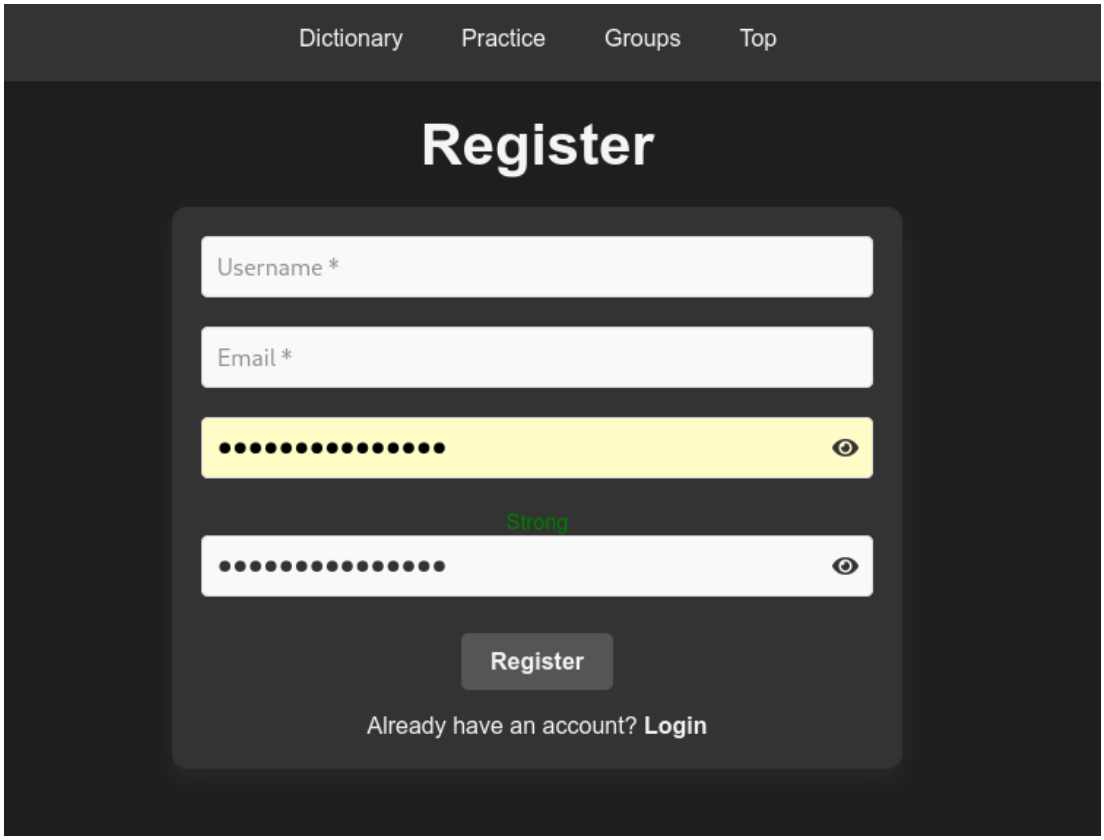


Рисунок 3.4 – Форма реєстрації користувача

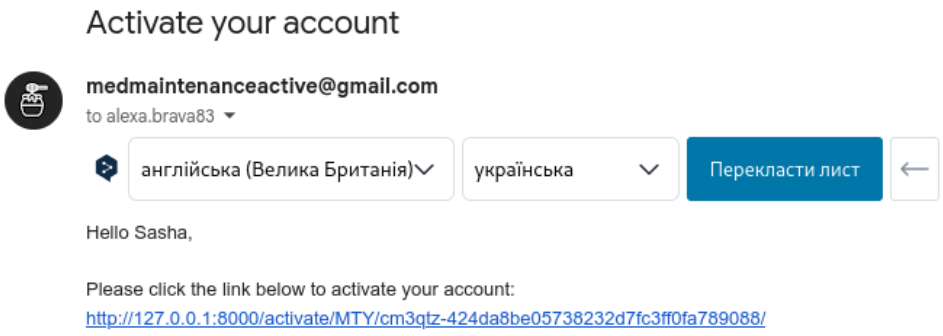


Рисунок 3.5 – Лист підтвердження електроної пошти

Після сторінки входу чи реєстрації, користувач потрапляє на свою стоірнку, де він може редагувати свій профіль, змінювати інформацію про себе в налаштуваннях, змінювати приватність своїх слів та груп, бачити статистику слів у вигялді різних діаграм, а також зверху є навігаційна панель, де є пошук, який виконує пошук користувачів, а також кнопки для переходу між різними стоірнками.

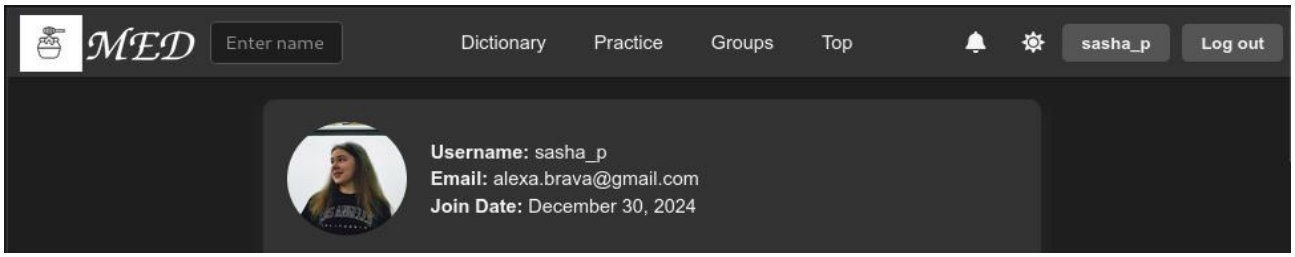


Рисунок 3.6 – Сторінка профілю користувача

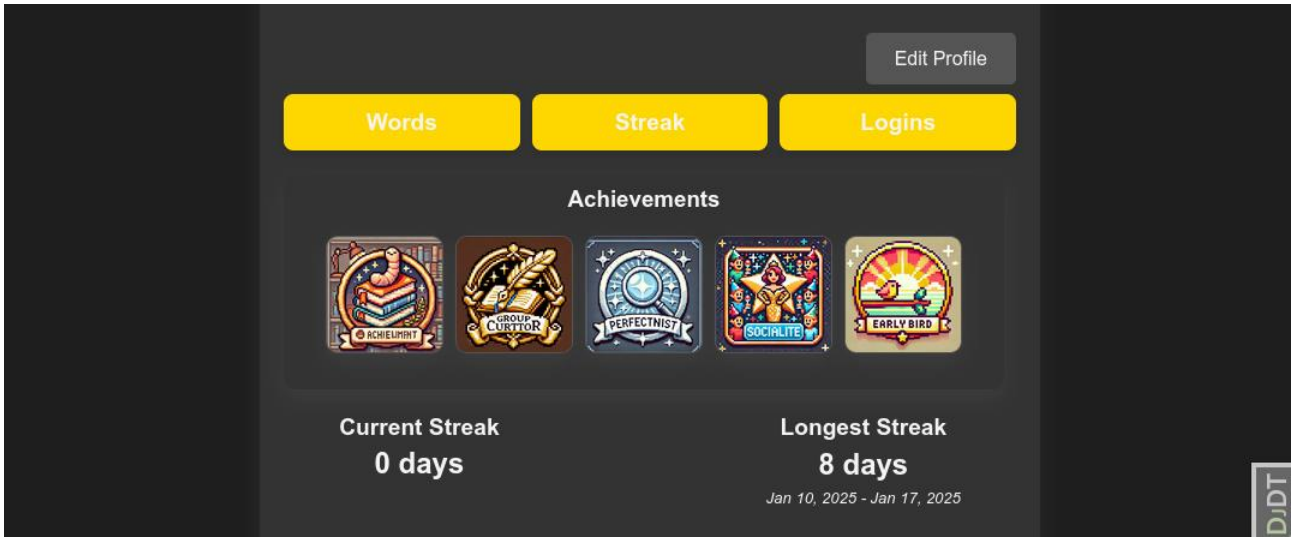


Рисунок 3.7 – Інформація про те чи користувач в топі, його досягення та стрік додавання слів

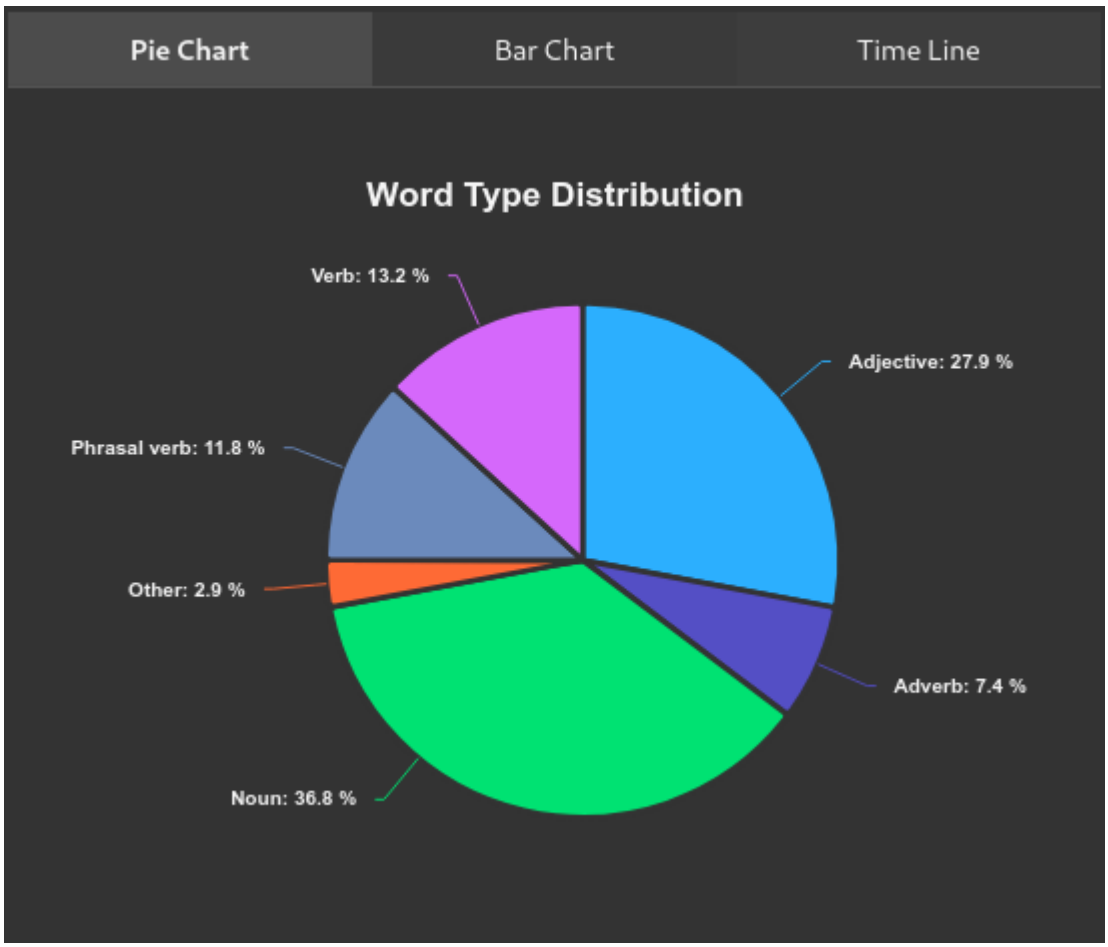


Рисунок 3.8 – Діаграма розподілу слів за типом

В налаштуваннях профіля можна змінювати порядок діаграм в профілі а також вмикати та вимикати різні з них. Доступні 3 типип діаграм:

1. кругова діаграма розподілу слів за типами;
2. стовпчаста діаграма розпоідлу слів за типами;
3. хронологія додаваня слів за останніій тиждень у вигляді графіка зображено на рисунку 3.9;

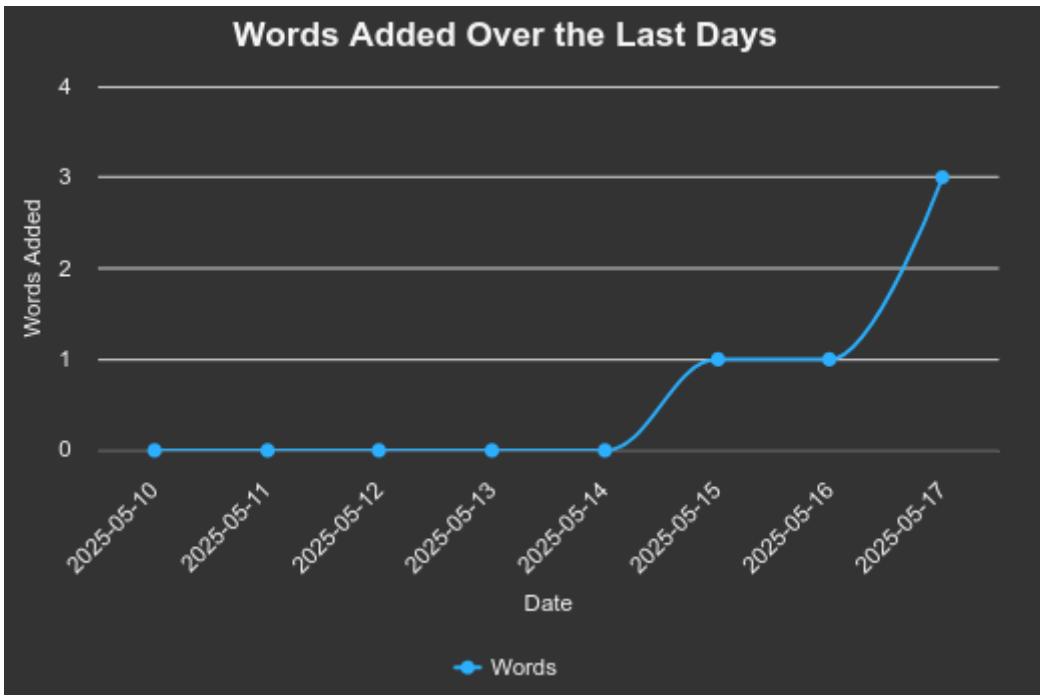


Рисунок 3.9 – Хронологія додавання слів за останні 7 днів

В навігаційні панелі зверху є 4 посилання:

- Словник
- Практика
- Групи
- Топ

Коли натискаємо на кнопку «словник», нас перекидає на сторінку де ми можемо побачити всі свої збережені слова зображення 3.10.

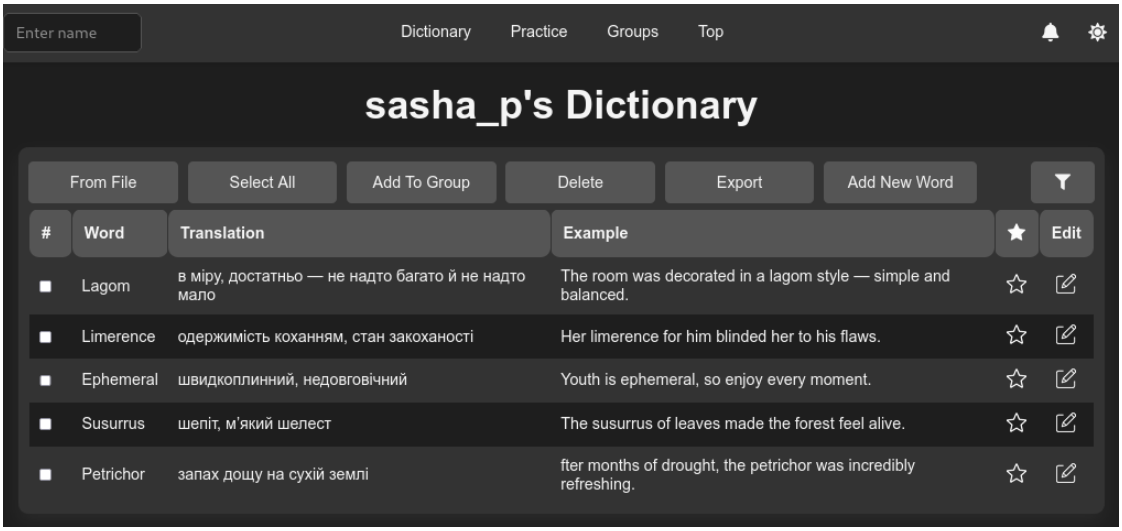


Рисунок 3.10 – Сторінка, що відображає наш словник

Тут можна фільтрувати слова, загружати ведику кількість слів з файлу (таблиці exel або json файлу), можна видаляти чи редагувати окремі слова, додавати слова в різні групи а також є функція, яка дозволяє скачати всі свої слова файлом в форматі pdf.

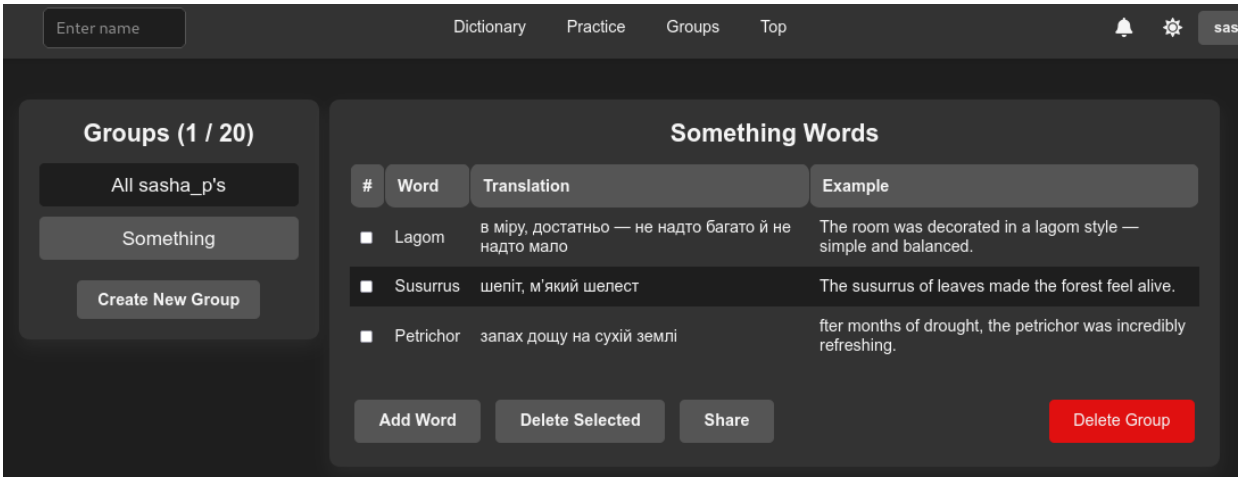


Рисунок 3.11 – Сторінка груп

На сторінці груп перш за все завжди є група All username ця група містить в собі всі слова користувача. Крім того можна створювати нові групи і додавати туди будь які слова з власного словника, це дозволяє створювати упорядковані системи слів для полегшення орієнтування в великому словнику, або просто для того щоб винести окремо якісь слова, наприклад ті що вам необіхдно вивчити чи запам'ятати.

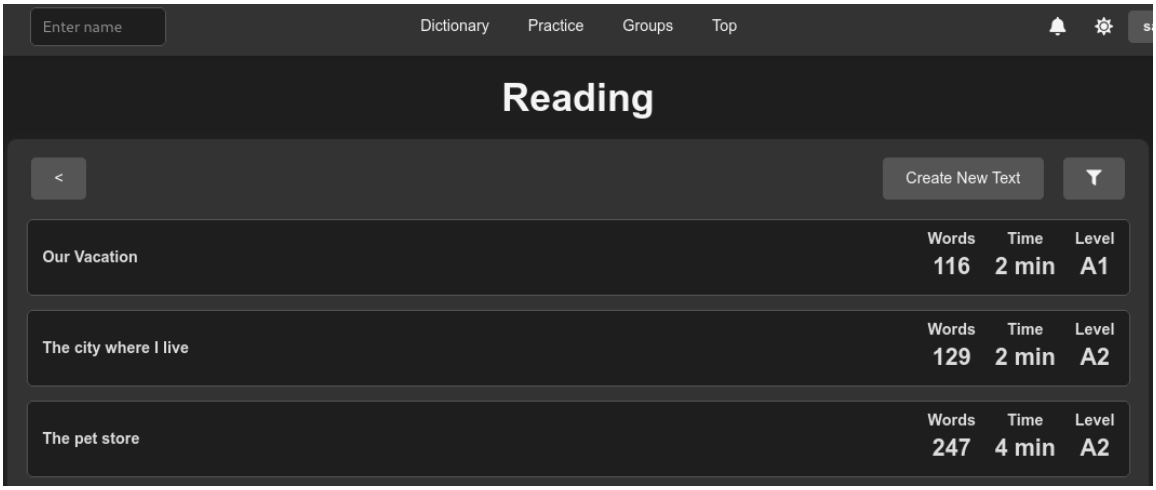


Рисунок 3.12 – Сторінка практики, а саме читання

На сторінці читання адміністратор може додавати нові тексти, а користувач може їх читати, якщо адміністратор вносить текст в правильному форматі, то майже кожне слово в тексті стає інтерактивним, на нього можна натиснути, побачити як воно пеерекладається, та речееня в якому воно є. Його можна зберегати в власний словник щоб краще запам'ятати. У таблиці 3.2 наведено порівняння функціональних можливостей реалізованого веб-додатку з популярними аналогами, що підтверджує його розширену функціональність та ефективність у контексті поставленої мети дослідження.

Таблиця 3.2 – Порівняння функціональності реалізованого веб-додатку з існуючими аналогами

Критерій	**Duolingo**	**Quizlet**	**Memrise**	**Реалізований web-додаток**
Персоналізований словник	-	+	+	+
Можливість створення груп слів	-	+-	-	+
Інтерактивне читання текстів	-	-	-	+
Відстеження досягнень	+	-	+	+
Соціальна взаємодія між користувачами	+-	-	-	+
Рейтингова система	+	-	+	+
Гнучке керування налаштуваннями профілю	-	-	-	+
Підтримка імпорту/експорту словника	-	+	-	+
Швидкість доступу до основного функціоналу	1.5хв	2хв	2.5хв	1хв

3.5 Висновки

У цьому розділі було реалізовано ключові програмні компоненти веб-додатку для вивчення англійської мови з урахуванням попередньо спроєктованої архітектури та функціональних вимог. Особливу увагу було приділено вибору технологій, які забезпечують ефективну реалізацію

функціоналу, стабільність роботи, гнучкість масштабування, а також зручність подальшої підтримки системи.

На основі порівняльного аналізу було обґрунтовано вибір мови програмування Python у поєднанні з високорівневим фреймворком Django для реалізації серверної частини додатку. Django забезпечив структуровану та швидку розробку завдяки вбудованим інструментам для роботи з базами даних, формами, авторизацією та захистом. Для клієнтської частини було обрано класичний стек веб-технологій — HTML, CSS, JavaScript — із використанням шаблонної системи Django. Такий підхід дав змогу реалізувати інтерактивний та адаптивний інтерфейс без зайвих залежностей від зовнішніх фреймворків.

Було обґрунтовано вибір Visual Studio Code як основного середовища розробки, що завдяки своїй гнучкості, швидкодії та широкому набору розширень дало змогу ефективно працювати з усіма необхідними технологіями в межах одного інтерфейсу. Для реалізації асинхронних задач використано Celery у поєднанні з Redis, що забезпечило можливість обробки задач у фоновому режимі, зокрема — надсилання листів, оновлення досягнень, збереження статистики тощо. Для контейнеризації Redis було застосовано технологію Docker, що значно спростило розгортання середовища.

У рамках програмної реалізації було створено модельну частину проєкту, яка охоплює ключові сутності системи: користувачів, слова, групи слів, тексти для читання, досягнення, дружні зв’язки, повідомлення та інші. Важливою частиною стала модель UserProfile, яка дозволяє гнучко налаштовувати параметри профілю, відображення статистики, а також реалізувати механізми приватності й преміум-функціоналу. Створені моделі дозволяють ефективно організувати структуру даних і взаємодію між компонентами системи.

Крім того, реалізовано повноцінну взаємодію між клієнтом і сервером через представлення (views), що обробляють основну бізнес-логіку: додавання слів, редагування профілю, взаємодію з досягненнями, формування PDF-експорту, авторизацію, соціальні функції тощо. За допомогою класів

представлення (class-based views) та сигналів (@receiver) було організовано обробку подій і автоматичне оновлення досягнень користувача.

Також у системі реалізовано важливу функцію експорту словника у формат PDF, доступну лише користувачам із преміум-доступом. Це дозволяє користувачам зберігати або друкувати свої дані для подальшого використання, що підвищує практичну цінність платформи.

Проведено базове тестування реалізованого функціоналу, зокрема через адміністративну панель Django, яка дає змогу керувати даними безпосередньо з інтерфейсу без написання додаткового коду. Фронтенд-частина також протестована на основні сценарії взаємодії користувача з додатком: реєстрація, робота зі словником, редагування профілю, перегляд статистики, використання функції читання текстів, керування групами слів та досягненнями.

Таким чином, програмна реалізація охопила весь основний функціонал, необхідний для повноцінної роботи освітнього веб-додатку. Обрані технології й підходи забезпечили гнучку та масштабовану архітектуру, готову до розширення й інтеграції нових модулів у майбутньому.

ВИСНОВКИ

Поставлені перед бакалаврською кваліфікаційною роботою задачі виконано в повному обсязі, а саме:

- проведено аналіз предметної області з вивчення іноземних мов та обґрунтовано доцільність створення веб-додатку для вивчення англійської мови;
- проаналізовано існуючі програмні рішення для мовного навчання та виявлено їх недоліки й обмеження;
- спроєктовано архітектуру веб-додатку, розроблено UML-діаграми, алгоритми та структуру бази даних;
- реалізовано клієнтську та серверну частини додатку з використанням фреймворку Django, мови Python, а також асинхронної обробки завдань за допомогою Celery і Redis;
- протестовано основні функціональні модулі веб-додатку, зокрема: словник, групи слів, тексти для читання, система досягнень, профіль користувача, соціальна взаємодія, рейтингова система та інше.

У процесі виконання бакалаврської кваліфікаційної роботи було спроєктовано та реалізовано повнофункціональний веб-додаток, що дозволяє користувачу створювати власний словник англійської мови, об’єднувати слова у групи, читати адаптовані тексти з перекладом, переглядати статистику, отримувати досягнення та взаємодіяти з іншими користувачами.

Розроблений web-додаток підтримує гнучке управління контентом, налаштування видимості інформації, імпорт та експорт словника, а також надає інструменти для спільного навчання. Було створено понад десять основних моделей бази даних, зокрема: словникові записи, групи, тексти, профілі, досягнення, сповіщення, дружні зв’язки, рейтинги.

Відповідно до поставленої мети, у дипломній роботі реалізовано веб-додаток, який забезпечує персоналізоване навчальне середовище, гнучке

керування навчальним контентом, систему досягнень і соціальну взаємодію між користувачами.

У порівнянні з найближчими аналогами, у розробленому додатку впроваджено розширені функції самостійної роботи зі словником, зручне читання текстів, перегляд активності та гейміфікацію навчального процесу, що значно підвищує ефективність засвоєння англійської мови.

Звіт про виконану роботу оформлено відповідно до встановлених вимог та методичних вказівок, що забезпечує його послідовність та відповідність академічним стандартам. [25]

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. РОЗРОБКА МОДУЛЯ СОЦІАЛЬНОЇ ВЗАЄМОДІЇ ДЛЯ ПЕРСОНАЛЬНОГО АНГЛОМОВНОГО СЛОВНИКА НА БАЗІ DJANGO /О.О. Писарук, І.В. Богач // Матеріали LIV науково-технічної конференції підрозділів Вінницького національного технічного університету (НТКП ВНТУ-2025). Збірник доповідей [Електронний ресурс], Вінниця : ВНТУ, 2025. Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2025/paper/view/24507

2. РОЗРОБКА ІНТЕРАКТИВНОГО МОВНОГО СЕРЕДОВИЩА ДЛЯ ПЕРСОНАЛІЗОВАНОГО НАВЧАННЯ АНГЛІЙСЬКОЇ МОВИ НА ОСНОВІ DJANGO / О.О.Писарук, І.В.Богач// Матеріали міжнародної науково-практичної Інтернет-конференції студентів, аспірантів та молодих науковців «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)». Збірник доповідей [Електронний ресурс], Вінниця: ВНТУ, 2025. Режим доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2025/paper/view/25424

3. StriveCloud. (2025). Duolingo Gamification Explained.Доступно за посиланням: https://strivecloud.io/blog/gamification-examples-boost-user-retention-duolingo/

4. Quizlet. (2025). Mission – Our mission is to help students practice and master whatever they are learning. [Електронний ресурс]. Режим доступу: https://quizlet.com/mission

5. Wikipedia. (2025). Memrise [Електронний ресурс]. Режим доступу: https://en.wikipedia.org/wiki/Memrise

6. Іванов, П. Django для початківців і не тільки. Київ: Ліра-К, 2022. – 296с.

7. Новоселець, О. Веб-програмування на Python: практичний посібник. Львів: Ліра-К, 2019. – 256 с.

8. Іванов, Т. Django: розробка веб-застосунків. Київ: Профіль, 2021. – 312 с.

9. Сергієнко, А. Бази даних SQLite: створення та управління. Харків: Основа, 2022. – 208 с.

10. Шевчук, Н. Методика розробки навчальних мовних додатків. Київ: Шкільний світ, 2020. – 224 с.

11. Мартін, Р. Чиста архітектура: мистецтво розроблення програмного забезпечення. Харків: Віват, 2021. – 416 с.

12. Бхаргава, А. Грокаємо алгоритми. Ілюстрований посібник для програмістів і допитливих. Київ: Наш Формат, 2019. – 280 с.

13. Гайджарж, В., Ізварін, І. Бази даних в інформаційних системах. Київ: Центр учбової літератури, 2018. – 320 с.

14. Козлов, Д. Веб-розробка з використанням Python і JavaScript. Харків: Фоліо, 2021. – 248 с.

15. Ковальчук, О., Петров, І. Асинхронне програмування на Python з Celery. Київ: Наукова думка, 2023. – 192 с.

16. Мартін, Р. Чистий код: створення, аналіз і рефакторинг. Харків: Віват, 2020. – 464 с.

17. Мельник, Р. Програмування веб-застосувань (фронт-енд та бек-енд). Львів: Видавництво Львівської політехніки, 2021. – 204 с.

18. Бородкіна, І., Бородкін, Г. Інженерія програмного забезпечення. Київ: Каравела, 2020. – 288 с.

19. Грицюк Ю. Аналіз вимог до програмного забезпечення. Київ: КНУБА, 2019. – 212 с.

20. Burston, J. (Ed.). Computer Assisted Language Learning, Vol. 34, Issue 7. Taylor & Francis, 2021. URL: https://www.tandfonline.com/toc/ncal20/34/7

21. TestDriven.io. The Definitive Guide to Celery and Django, updated Feb 3, 2025. URL: https://testdriven.io/courses/celery-django/

22. Appliku. “Django Celery Tutorial to Background Tasks.” Appliku.com, 2025. URL: https://appliku.com/tutorials/django-celery/

23. Mercer, M., Marchetti, D., & Meyer, F. Celery: Distributed Task Queue — Official Documentation, 2025. URL: https://docs.celeryproject.org/en/stable/

24. Garcia, M., & Li, X. “Cloud-Native Web Apps for Language Learning: Architectures and Case Studies.” IEEE Access, 2022. URL: https://ieeexplore.ieee.org/document/9525581

25. Яровий А. А., Сілагін О. В., Богач І. В. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів спеціальності 122 «Комп’ютерні науки» [Електронний ресурс]. Вінниця : ВНТУ, 2023. PDF, 54 с.).

Додаток А (обов'язковий)

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Розробка WEB-додатку для вивчення англійської мови

Тип роботи: бакалаврська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ кафедра комп'ютерних наук
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 1,52 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

✓ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

Яровий А.А., зав. каф. КН _____ _____
(прізвище, ініціали, посада) (підпис)

Колесницький О.К., проф. каф КН _____ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Озеранський В.С.
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Богач І.В.
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Писарук О.О.
(підпис) (прізвище, ініціали)

ДОДАТОК Б (обов'язковий)

Лістинг основних функцій програми

```
from datetime import datetime
import re
import string
from django.http import FileResponse, Http404, HttpResponseNotFound, HttpResponseRedirect, HttpResponse
from django.shortcuts import get_object_or_404, redirect, render
from django.urls import reverse, reverse_lazy
from med.forms import AddWordForm, ChengePasswordForm, RegisterUserForm, LoginUserForm, WordForm, GroupForm, EditProfileForm, AvatarUpdateForm, WordsShowForm, TextForm
from django.views.generic import ListView, CreateView
from django.contrib.auth.views import LoginView
from django.views import View
from django.views.decorators.cache import cache_page
from django.db.models import Min, Max, When, Case
import requests

from reportlab.pdfbase.ttfonts import TTFont
from reportlab.pdfbase import pdfmetrics
from reportlab.lib.pagesizes import letter
from reportlab.pdfgen import canvas

from django.utils.timezone import now, timedelta
from django.db.models.functions import TruncDay

from django.middleware.csrf import get_token
from django.core.files.base import ContentFile
```

```
import base64

from django.contrib.auth.decorators import login_required
from django.contrib import messages
from django.utils.decorators import method_decorator

from django.db import transaction
from django.db.models import Q, Count

from django.http import JsonResponse
import json

from django.core.paginator import Paginator

from urllib.parse import urlparse

import openpyxl

from django.contrib.auth.tokens import default_token_generator
from django.utils.http import urlsafe_base64_decode
from django.contrib.auth import logout, login, update_session_auth_hash
from django.contrib.auth.mixins import LoginRequiredMixin
import os
from med.models import *

from django.db.models.signals import post_save, m2m_changed
from django.dispatch import receiver
from django.conf import settings

from med.tasks import send_activation_email, update_top
```

```
MAX_GROUP_COUNT = 20

practice_cards = {
    'test': {
        'word': 'Test',
        'img': 'med/img/practice/dice_light.png',
        'dark_img': 'med/img/practice/dice_dark.png',
        'href': 'soon'
    },
    'reading': {
        'word': 'Reading',
        'img': 'med/img/practice/read_light.png',
        'dark_img': 'med/img/practice/read_dark.png',
        'href': 'practice_reading'
    },
    'groups': {
        'word': 'Groups',
        'img': 'med/img/practice/group_light.png',
        'dark_img': 'med/img/practice/group_dark.png',
        'href': 'practice_groups'
    },
}

processed_signals = {}
process_wrods_signals = {}

@cache_page(60 * 15)
def index(request):
    if request.user.is_authenticated:
```

```
        return redirect('profile', user_name=request.user.username)
    return render(request, 'med/index.html')

@cache_page(60 * 15)
def about(request):
    return render(request, 'med/about.html')

def add_to_main_group(request, word):
    group_name = f"All {request.user.username}'s "
    group, created = WordGroup.objects.get_or_create(
        name=group_name,
        is_main=True,
        user=request.user
    )

    group.save()

    if isinstance(word, str):
        word = Word.objects.get(word=word, user=request.user)

    group.words.add(word)


@method_decorator(login_required, name='dispatch')
class AddWordView(LoginRequiredMixin, CreateView):
    form_class = AddWordForm
    template_name = 'med/add_word.html'
    extra_context = {'title': 'Add Word'}

    def get_context_data(self, **kwargs):
```

```
        context = super().get_context_data(**kwargs)
        context['from_text'] = self.request.GET.get('from') == 'text'
        return context

    def get_success_url(self):
        return reverse_lazy('words', kwargs={'user_name': self.request.user.username})

    def form_valid(self, form):
        word = form.save(commit=False)
        word.user = self.request.user
        word.save()

        add_to_main_group(self.request, word)

        from_text = self.request.POST.get('from_text', '') == 'true'
        if from_text:
            user_profile = UserProfile.objects.get(user=self.request.user)
            user_profile.words_added_from_text += 1
            user_profile.save()

        return super().form_valid(form)


@method_decorator(login_required, name='dispatch')
class ConfirmDeleteView(View):
    def get(self, request, *args, **kwargs):
        word_ids = request.GET.getlist('word_ids')
        group_id = request.GET.get('group_id')
        text_id = request.GET.get('text_id')
```

```
        context = {}
        if text_id:
            text = get_object_or_404(ReadingText, id=text_id)
            context.update({
                'is_text': True,
                'text': 'Are you sure you want to delete this text?',
                'text_id': text_id,
                'text_title': text.title
            })

        elif group_id:
            group = get_object_or_404(WordGroup, id=group_id,
user=request.user)
            words = group.words.filter(id__in=word_ids) if word_ids else []
            context.update({
                'is_group': True,
                'group_name': group.name,
                'group_id': group_id,
                'words': words,
                'text': self._get_delete_message(words, group.name)
            })

            if word_ids:
                context['word_ids'] = word_ids

        elif word_ids:
            words = Word.objects.filter(id__in=word_ids, user=request.user)
            if not words:
                return redirect('words')
```

```
            context.update({
                'is_group': False,
                'words': words,
                'word_ids': word_ids,
                'text': self._get_delete_message(words)
            })
        return render(request, 'med/confirm_delete.html', context)

    def post(self, request, *args, **kwargs):
        word_ids = request.POST.getlist('word_ids')
        group_id = request.POST.get('group_id')
        text_id = request.POST.get('text_id')

        if text_id:
            get_object_or_404(ReadingText, id=text_id).delete()
            return redirect('practice_reading')
        elif group_id:
            group = get_object_or_404(WordGroup, id=group_id,
user=request.user)
            if word_ids:
                words = group.words.filter(id__in=word_ids)
                for word in words:
                    group.words.remove(word)

                return redirect('group_words', group_id=group_id)
            else:
                group.delete()
                return redirect('groups')
        elif word_ids:
            Word.objects.filter(id__in=word_ids, user=request.user).delete()
```

```
            return redirect('words', user_name=request.user.username)

        return redirect('profile', user_name=request.user.username)

    @staticmethod
    def _get_delete_message(words, group_name=None):
        if group_name:
            return ("Are you sure you want to delete this group?"
                    if len(words) == 0 else
                    f"Are you sure you want to delete these words from group {group_name}?"
                    if len(words) > 1 else
                    f"Are you sure you want to delete this word from group {group_name}?")
        return ("Are you sure you want to delete these words?"
                if len(words) > 1 else
                "Are you sure you want to delete this word?")


@method_decorator(login_required, name='dispatch')
class EditWordView(View):
    def get(self, request, word_id, *args, **kwargs):
        word = get_object_or_404(Word, id=word_id, user=request.user)
        form = WordForm(instance=word)
        return render(request, 'med/edit_word.html', {'form': form, 'word': word})

    def post(self, request, word_id, *args, **kwargs):
        word = get_object_or_404(Word, id=word_id, user=request.user)
        form = WordForm(request.POST, instance=word)
```

```
        if form.is_valid():
            form.save()

            if form.has_changed():
                user_profile = UserProfile.objects.get(user=request.user)
                user_profile.edited_words += 1
                user_profile.save()

            return redirect('words', user_name=request.user.username)
        return render(request, 'med/edit_word.html', {'form': form, 'word': word})


@method_decorator(login_required, name='dispatch')
class WordListView(ListView):
    model = Word
    template_name = 'med/words.html'
    context_object_name = 'words'
    paginate_by = 25
    def get(self, request, *args, **kwargs):
        if 'sort_alphabet' not in request.GET and 'sort_date' not in request.GET:
            query_params = request.GET.copy()
            query_params['sort_date'] = 'desc'
            return HttpResponseRedirect(f"{request.path}?{query_params.urlencode()}")
        return super().get(request, *args, **kwargs)

    def get_queryset(self):
        user_name = self.kwargs['user_name']
        user = get_object_or_404(User, username=user_name)
```

```
self.is_my_dict = user == self.request.user
queryset = Word.objects.filter(user=user)

word_type = self.request.GET.get('type')
if word_type:
    queryset = queryset.filter(word_type=word_type)

filter_word = self.request.GET.get('filter_word', '')
filter_translation = self.request.GET.get('filter_translation', '')

if filter_word:
    queryset = queryset.filter(word__icontains=filter_word)
if filter_translation:
    queryset = queryset.filter(translation__icontains=filter_translation)

sort_alphabet = self.request.GET.get('sort_alphabet')
sort_date = self.request.GET.get('sort_date')

if sort_alphabet == 'asc':
    queryset = queryset.order_by('word')
elif sort_alphabet == 'desc':
    queryset = queryset.order_by('-word')

if sort_date == 'asc':
    queryset = queryset.order_by('time_create')
elif sort_date == 'desc':
    queryset = queryset.order_by('-time_create')

return queryset
```

```
    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        user = get_object_or_404(User, username=self.kwargs['user_name'])
        request_user = self.request.user

        friends = User.objects.filter(
            Q(friendship_requests_sent__receiver=user,
friendship_requests_sent__status='accepted') |
            Q(friendship_requests_received__sender=user,
friendship_requests_received__status='accepted')
        ).distinct()

        is_friends = request_user in friends

        types = Word.TYPE_CHOICES

        context.update({
            'user_name': self.kwargs['user_name'],
            'user': user,
            'title': f"{self.kwargs['user_name']}'s Dictionary",
            'is_my_dict': getattr(self, 'is_my_dict', False),
            'is_dict': True,
            'logged_user': request_user,
            'access': UserProfile.objects.get(user=user).access_dictionary,
            'is_friends': is_friends,
            'filter_word': self.request.GET.get('filter_word', ''),
            'filter_translation': self.request.GET.get('filter_translation', ''),
            'sort_alphabet': self.request.GET.get('sort_alphabet', 'asc'),
            'sort_date': self.request.GET.get('sort_date', 'asc'),
```

ДОДАТОК В

(обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

«РОЗРОБКА WEB-ДОДАТКУ ДЛЯ ВИВЧЕННЯ АНГЛІЙСЬКОЇ МОВИ»

Виконала: студентка 4 курсу,групи 4КН-21б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Писарук О.О.
(прізвище та ініціали)

Керівник: к.т.н., доцент, асистент каф.КН
Богач І.В.
(прізвище та ініціали)
« 03 » червня 2025 р.

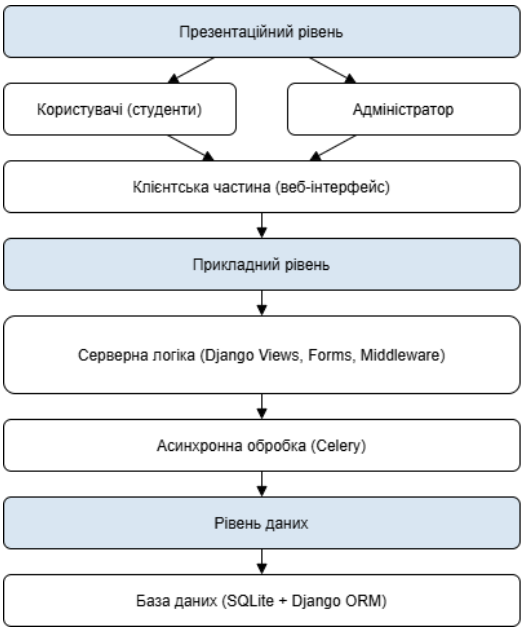


Рисунок В.1 – Архітектурне рішення додатку «Для вивчення англійської мови»

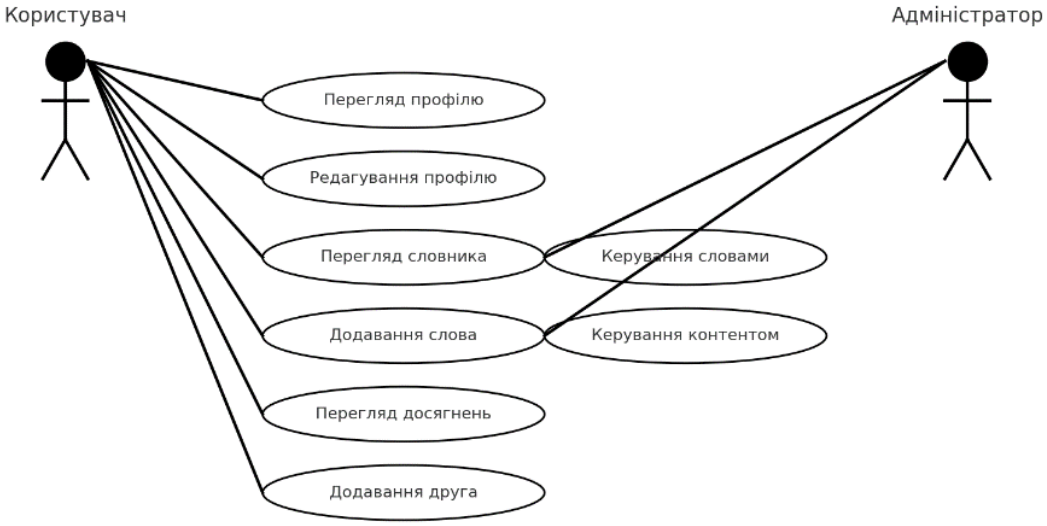


Рисунок В.2 – Діаграма прецедентів взаємодії користувача з додатком

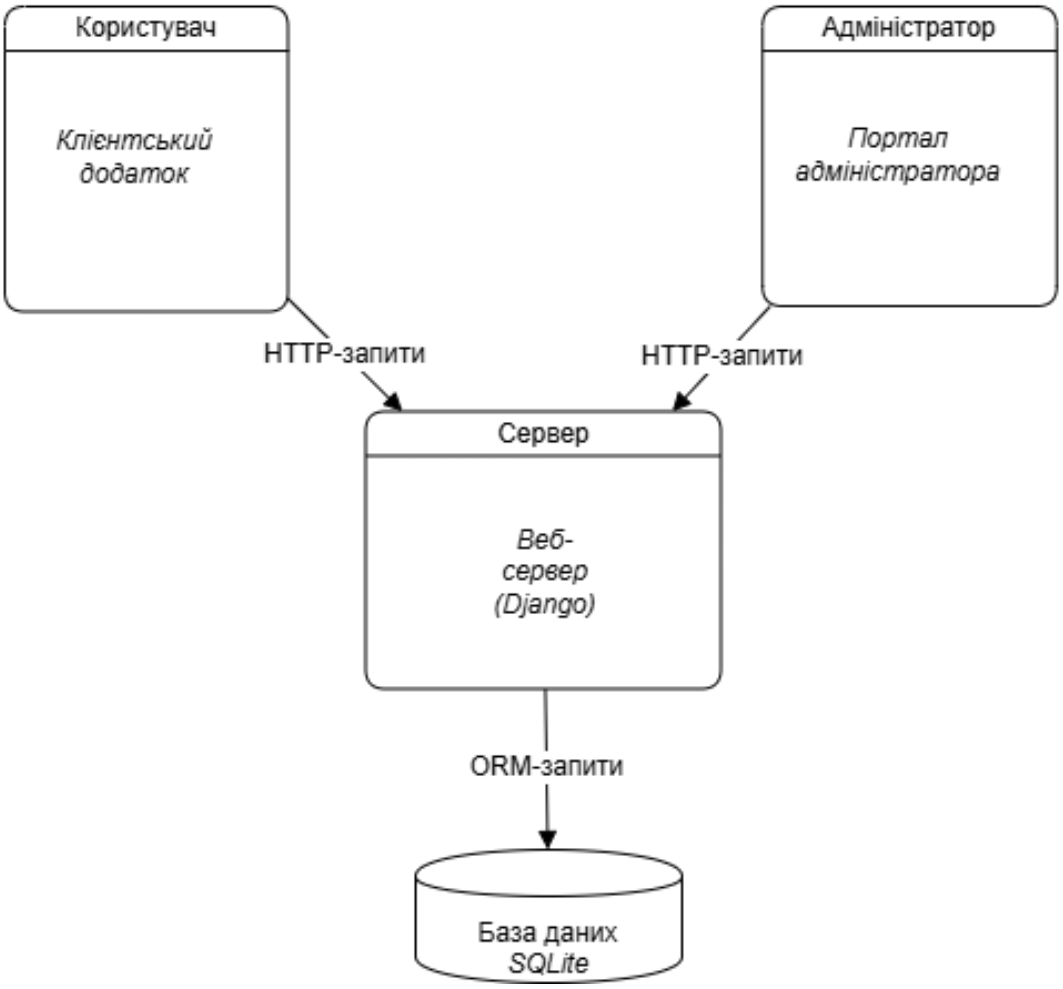


Рисунок В.3 – Діаграма розгортання WEB-додатку для вивчення англійської мови

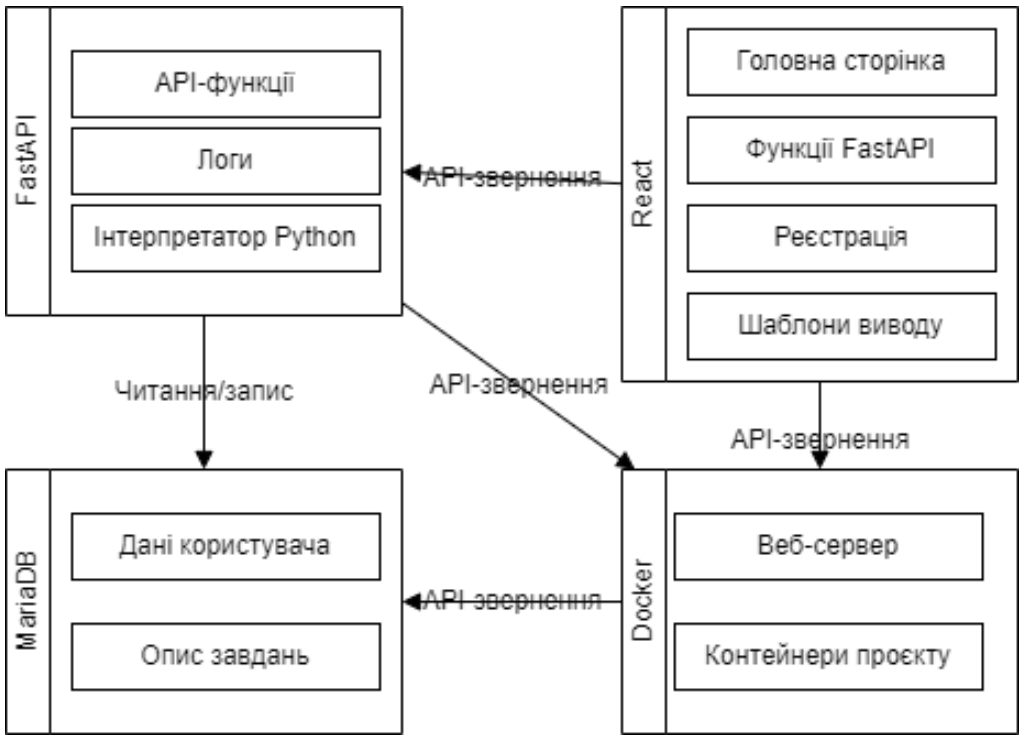


Рисунок В.4 – UML-діаграма компонентів WEB-додатку для вивчення англійської мови

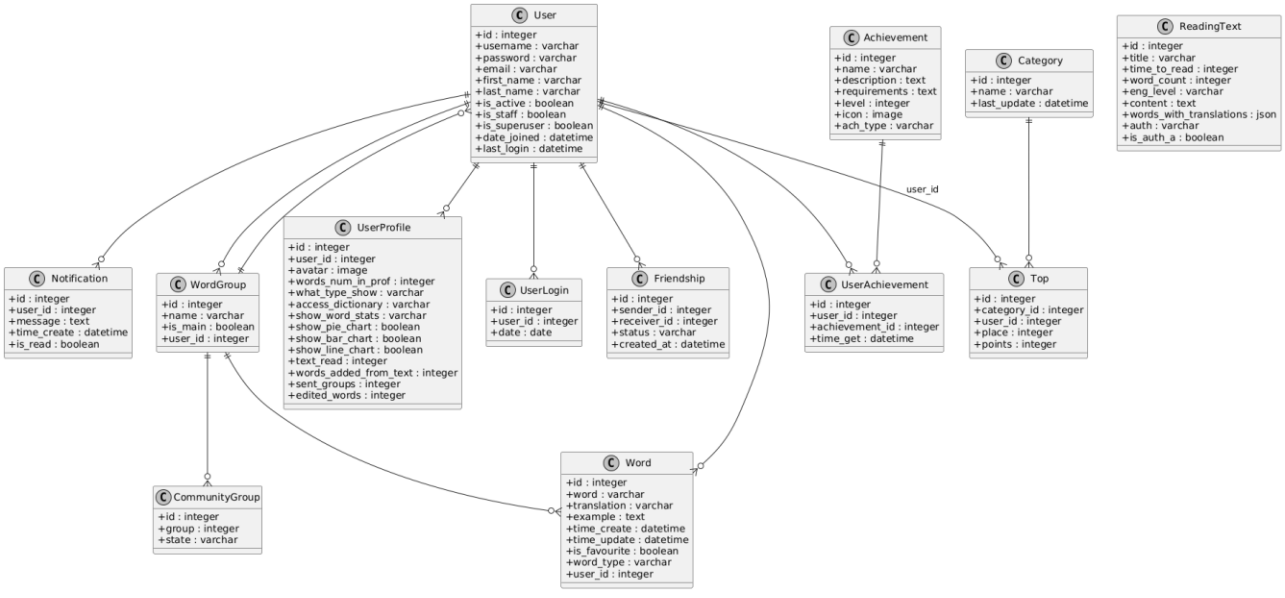


Рисунок В.5 – Фрагмент реляційної моделі даних для веб-ресурсу «Вивчення англійської мови»

Рисунок В.6 – Граф-схема алгоритму додавання слова до словника

ДОДАТОК Г (довідниковий)

Інструкція користувача

Для роботи з web-додатком потрібно виконати наступне:

1. Встановити необхідні програми

Python:

- Перейти на python.org.
- Завантажити та встановити Python (переконайтеся, що це версія 3.8 або новіша).
- Під час встановлення поставити галочку біля "Add Python to PATH".

Docker:

- Перейти на docker.com.
- Завантажити Docker Desktop для вашої операційної системи (Windows, macOS або Linux).
- Встановити і запустити Docker (переконайтеся, що програма працює).

Git:

- Перейти на git-scm.com.
- Завантажити та встановити Git для вашої операційної системи.
- Після встановлення перевірити, чи працює Git, ввівши в терміналі: git –version

2. Завантаження проєкту

1) Відкрити термінал (на Windows це може бути "Командний рядок" або "PowerShell", на macOS/Linux — "Термінал") в VSCode
2) Створити папку, куди завантажено проєкт. Наприклад:

```
mkdir med
cd med
```

3) Завантажити код проєкту з GitHub:

git clone https://github.com/HappyMaxxx/Med-My-English-Dictionary .

(Крапка в кінці означає, що код буде завантажено в поточну папку.)

3. Налаштування віртуального середовища

1) У терміналі, перебуваючи в папці проєкту (med), створити віртуальне середовище:

python3 -m venv venv

2) Активувати віртуальне середовище:

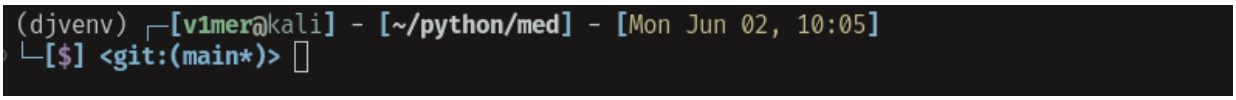
- Для **Linux/macOS**:

source venv/bin/activate

- Для **Windows**:

venv\Scripts\activate

Після цього в терміналі перед вашим рядком має з’явитися (venv).



Тут djvenv

3) Встановити необхідні бібліотеки:

pip install -r requirements.txt

4. Налаштування Docker і Redis

1) Переконатися, що Docker запущено на вашому комп’ютері (Docker Desktop має бути відкритим).

2) Запустити Redis за допомогою Docker:

docker run -d --name redis-server -p 6379:6379 redis

Ця команда завантажить і запустить Redis у фоновому режимі.

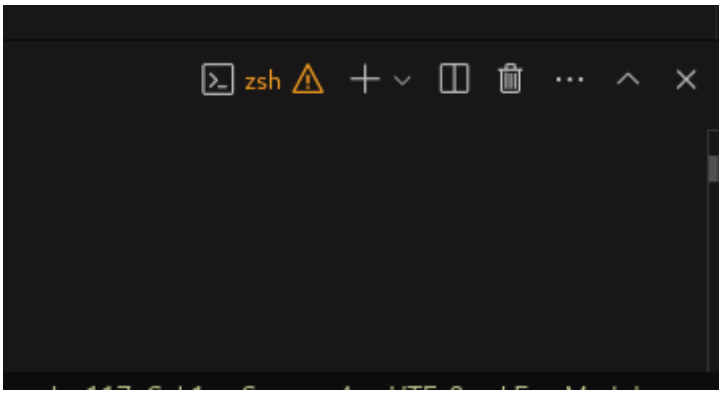
3) Перевірте, чи Redis працює:

```
docker ps
```

Ви маєте побачити контейнер із назвою redis-server у списку.

5. Налаштування Celery

1) Відкрити новий термінал (залишивши попередній відкритим) і перейти до папки проєкту (med).



2) Активувати віртуальне середовище (як у кроці 3.2).
3) Запустити Celery worker:

```
celery -A mad worker -l info
```

4) Відкрити ще один термінал, перейти до папки проєкту, активувати віртуальне середовище та запустити Celery beat:

```
celery -A mad beat -l info
```

5) (Необов’язково) Для моніторингу Celery можна запустити Flower у ще одному терміналі:

```
celery -A mad flower
```

6. Налаштування файлу .env

1) Знайти файл example.env у папці проєкту та перейменувати його на .env

2) Відкрити файл .env у текстовому редакторі
3) Налаштувати дані для пошти:
    - Зайти у налаштування Gmail-акаунта.
    - Увімкнути "Двоетапну перевірку" (якщо ще не ввімкнена).
    - Створити "Пароль для додатків" у розділі безпеки.
    - Скопіювати цей пароль і вставити його в .env у відповідне поле (наприклад, EMAIL_HOST_PASSWORD).

7. Запуск проєкту

1) У терміналі, перебуваючи в папці проєкту (med) з активованим віртуальним середовищем, запустити сервер:

```
python manage.py runserver
```

2) Відкрити браузер і перейти за адресою: http://127.0.0.1:8000.
3) Ви маєте побачити головну сторінку додатка **Med - My English Dictionary**.

8. Завершення роботи

1) Щоб зупинити сервер, потрібно натиснути Ctrl+C у терміналі, де запущено python manage.py runserver.
2) Зупинити Redis, якщо він більше не потрібен:

```
docker stop redis-server
docker rm redis-server
```

3) Деактивувати віртуальне середовище:

```
deactivat
```

МАЛЕ НАУКОВО-ВИРОБНИЧЕ ПІДПРИЄМСТВО "ТОВ "ІТІ"
Україна, 21021, м.Вінниця, вул. Келецька, 56

№ 42 від "6" 06 2025р.

ДОВІДКА

Дана Писарук Олександрі Олександрівні в тому, що результати, одержані нею в процесі виконання бакалаврської кваліфікаційної роботи, а саме алгоритми та програмні засоби для організації вивчення англійської мови, планується використати в розробках ТОВ «ІТІ».

Зам. директора ТОВ «ІТІ» Бодяк В.М.