ЗВО УНІВЕРСИТЕТ КОРОЛЯ ДАНИЛА

Факультет суспільних та прикладних наук

Кафедра інформаційних технологій

на правах рукопису

Тицький Роман Романович

УДК 004.4

Розробка платформи для інтелектуального навчання англійської мови за допомогою онлайн-флеш-карти та тестів

Спеціальність 121 – «Інженерія програмного забезпечення»

Кваліфікаційна робота на здобуття кваліфікації бакалавр

Нормоконтроль

_____Стисло О.В.

(підпис, дата, розшифрування підпису)

Студент

_____Тицький Р.Р.

(підпис, дата, розшифрування підпису)

Допускається до захисту

Завідувач кафедри

_____к.т.н., доц. Ващишак С.П.

(підпис, дата, розшифрування підпису)

Керівник роботи

_____ к.т.н., доц. Ващишак С.П.

(підпис, дата, розшифрування підпису)

Івано-Франківськ – 2024

ЗВО УНІВЕРСИТЕТ КОРОЛЯ ДАНИЛА

Факультет суспільних та прикладних наук

Кафедра інформаційних технологій

Освітній ступінь: «бакалавр»

Спеціальність: 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____

«____» _____ 2024 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Тицький Роман Романович

(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи:

Розробка платформи для інтелектуального навчання англійської мови за допомогою онлайн-флеш-карти та тестів

керівник роботи:

Ващишак Сергій Петрович, к.т.н., доцент.

затверджена наказом вищого навчального закладу від « 12 » березня 2024 року № 19/1

2. Термін подання студентом роботи 05.06.2024
3. Вихідні дані роботи: алгоритми адаптивного тестування, веб-технології для розробки навчальних платформ, бази даних, API для інтеграції.
4. Зміст кваліфікаційної роботи (перелік питань, які потрібно розробити)
   1. Характеристика типових освітніх веб-платформ
   2. Вибір технологій створення платформи засобами HTML, CSS, JavaScript, Django
   3. Етапи створення веб-платформи
   4. Створення макету та прототипу платформи "WordSnap"
5. Дата видачі завдання 14.03.2024

КОНСУЛЬТАНТИ РОЗДІЛІВ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Розділ	Консультант (прізвище, ініціали та посада)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1.	Збір та аналіз інформації для дослідження та розробки.	14.03.2024	Виконано
2.	Підготовка 1 розділу роботи.	28.03.2024	Виконано
3.	Проектування та розробка веб-платформи.	07.04.2024	Виконано
4.	Підготовка 2 розділу роботи.	21.04.2024	Виконано
5.	Програмна реалізація.	29.04.2024	Виконано
6.	Підготовка 3 розділу роботи.	15.05.2024	Виконано
7.	Оформлення квалифікаційної роботи та підготовка до захисту роботи	20.05.2024	Виконано

Студент _____ Тицький Р.Р.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Ващищак С.П.
(підпис) (прізвище та ініціали)

Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Сторінка	Опис графічного матеріалу	Сторінка	Опис графічного матеріалу
17	Quizlet створення флеш-карток	59	Головна сторінка
18	Quizlet створення тестів	62	Сторінка "Flashcards"
20	Anki створення флеш-карток	81	Сторінка "Create flashcards"
22	Brainscape кімната для створення флеш-карток	87	Сторінка "Dictionary"
22	Brainscape створення флеш-карток	90	Сторінка "tests"
26	Multilang головна сторінка веб-платформи	93	Результат тесту
31	Прототип Figma сторінка реєстрації	94	Сторінка "Vocabulary test"
33	Leancanvas		

АНОТАЦІЯ

Ця кваліфікаційна робота присвячена розробці онлайн-платформи для навчання англійської мови за допомогою флеш-карток та тестів. Розробка платформи здійснена з використанням сучасних технологій програмування, таких як Python, JavaScript, HTML, CSS, а також фреймворку Django для створення потужної серверної частини.

У рамках цього проекту було реалізовано ряд ключових функціональних можливостей, які дозволяють користувачам ефективно вивчати англійську мову. Використання Python та Django забезпечує надійність та швидкість роботи серверної частини платформи, а JavaScript дозволяє створити динамічний та інтерактивний інтерфейс для користувачів.

Крім того, платформа використовує сучасні методи та підходи до програмування, такі як асинхронне програмування та використання AJAX для забезпечення швидкодії та відзивчивості веб-додатку. Всі ці технології дозволяють створити зручне та ефективне середовище для навчання, яке відповідає сучасним вимогам програмістів та користувачів.

Результатом роботи є інноваційна онлайн-платформа, яка поєднує в собі найкращі практики програмування та навчання англійської мови, та може стати важливим інструментом для всіх, хто бажає покращити свої мовні навички в інтерактивному та захоплюючому форматі.

КЛЮЧОВІ СЛОВА: ОНЛАЙН-ПЛАТФОРМА, АНГЛІЙСЬКА МОВА, ФЛЕШ-КАРТКИ, ТЕСТИ, PYTHON, JAVASCRIPT, DJANGO, ВЕБ-РОЗРОБКА, АСИНХРОННЕ ПРОГРАМУВАННЯ, AJAX, ІНТЕРАКТИВНЕ НАВЧАННЯ, ПРОЕКТУВАННЯ АРХІТЕКТУРИ.

SUMMARY

This qualification work is devoted to the development of an online platform for teaching English using flashcards and tests. The platform was developed using modern programming technologies such as Python, JavaScript, HTML, CSS, as well as the Django framework to create a powerful backend.

As part of this project, a number of key functionalities have been implemented that allow users to learn English effectively. The use of Python and Django ensures the reliability and speed of the platform's backend, while JavaScript allows for a dynamic and interactive user interface.

In addition, the platform uses modern programming methods and approaches, such as asynchronous programming and AJAX, to ensure the performance and responsiveness of the web application. All these technologies allow us to create a convenient and effective learning environment that meets the modern requirements of programmers and users.

The result is an innovative online platform that combines the best practices of programming and English language learning, and can become an important tool for anyone who wants to improve their language skills an interactive and engaging format.

KEY WORDS: ONLINE PLATFORM, ENGLISH LANGUAGE, FLASHCARDS, TESTS, PYTHON, JAVASCRIPT, DJANGO, WEB DEVELOPMENT, ASYNCHRONOUS PROGRAMMING, AJAX, INTERACTIVE LEARNING, ARCHITECTURE DESIGN.

Зміст

ВСТУП

У сучасному світі, де інформаційні технології відіграють важливу роль у різних аспектах життя, володіння англійською мовою стає необхідністю для успішної кар'єри та професійного розвитку. Особливо це стосується фахівців у сфері інженерії програмного забезпечення, де спілкування та робота з документацією часто відбуваються англійською мовою.

З метою удосконалення мовних навичок та підвищення конкурентоспроможності студентів та фахівців у сфері програмного забезпечення, було розроблено та представлено платформу для інтелектуального навчання англійської мови за допомогою онлайн-флеш-карток та тестів. Ця платформа спрямована на забезпечення ефективного та захоплюючого навчання, враховуючи індивідуальні потреби та рівні користувачів.

У даній кваліфікаційній роботі детально описано процес розробки та функціонал веб-платформи для вивчення англійської мови за допомогою флеш карток, а також проведено аналіз її ефективності та можливостей застосування в навчальних цілях. Робота спрямована на вивчення можливостей застосування сучасних технологій для поліпшення процесу навчання та розвитку мовник навичок у сфері інженерії програмного забезпечення.

Актуальність теми. Актуальність дослідження полягає в усвідомленні швидко зростаючого попиту на якісне та ефективне навчання англійської мови серед студентів. Розвиток інформаційних технологій та поширення глобального спілкування надають особливого значення володінню англійською мовою в цій галузі. Розробка платформи для інтелектуального навчання англійської мови через онлайн-флеш-карти та тести відповідає потребам сучасного світу, де швидкість та доступність навчання мають велике значення. Цей підхід дозволить студентам та фахівцям в сфері програмного забезпечення покращити свої мовні навички, працюючи з матеріалами відповідно до свого рівня та індивідуальних потреб. Така платформа стане не лише зручним інструментом для навчання, але

й сприятиме підвищенню конкурентоспроможності в сучасному інформаційному середовищі.

Мета дослідження полягає у створенні та випробуванні онлайн-платформи для вивчення англійської мови через флеш-картки та тести. Основна ціль — надати студентам і користувачам зручний інструмент для покращення їхніх мовних навичок, що відповідають вимогам сучасного ринку праці. Також, мета включає визначення корисності цієї платформи для успішного навчання та професійного зростання.

Завдання дослідження, сформоване на основі мети, можна сформулювати наступним чином:

1. Визначення функціональних вимог до веб-платформи для інтелектуального навчання англійської мови з використанням флеш-карток та тестів, аналіз конкурентів.

2. Вибір технологій розробки, що найкраще відповідають вимогам проекту та забезпечують ефективність та зручність користування платформою.

3. Проектування архітектури системи, зокрема інтерфейсу користувача та бази даних, з метою забезпечення зручного та інтуїтивно зрозумілого середовища для навчання.

4. Розробка прототипу онлайн-платформи для інтелектуального навчання англійської мови та його тестування на відповідність визначеним вимогам.

5. Впровадження розробленої платформи для загального користування користувачами з метою надання доступу до матеріалів для навчання англійської мови та вдосконалення словникового запасу.

6. Оцінка ефективності та прийнятності розробленої платформи для інтелектуального навчання англійської мови шляхом збору та аналізу фідбеку від користувачів щодо її функціональності та зручності використання для вивчення англійської мови.

Об'єктом дослідження є процес створення веб-платформи для інтелектуального навчання англійської мови за допомогою технологій Python та JavaScript, фреймворк Django, а також мови розмітки HTML і CSS.

Предметом дослідження є аналіз та оптимізація архітектурних та програмних аспектів реалізації платформи для інтелектуального навчання англійської мови з використанням веб-технологій на базі Python.

Методи дослідження: методи дослідження, які були використані в ході дослідження та розробки платформи для інтелектуального навчання англійської мови за допомогою онлайн-флеш-карт та тестів:

1. Аналіз існуючих рішень: був проведений огляд різних веб-фреймворків, бібліотек та інструментів для веб-розробки на Python, таких як Django, Flask тощо. Мета аналізу полягає у виборі найбільш підходящих інструментів для реалізації проекту з урахуванням його вимог та потреб користувачів.

2. Проектування архітектури: була розроблена архітектура платформи, включаючи визначення модулів, компонентів та взаємодій між ними з урахуванням потреб користувачів та вимог проекту. Це допомагає забезпечити ефективну і стабільну роботу платформи.

3. Вибір інструментів та технологій: був здійснений вибір та оцінка інструментів, бібліотек та технологій, що найкраще підходять для реалізації окремих складових платформи, включаючи базу даних, мови які відповідають за візуалізацію сторінки, веб-сервер та фреймворк.

4. Розробка та тестування: було проведено розробку та тестування платформи, включаючи написання коду, створення інтерфейсів, реалізацію функціональності та перевірку її коректності та працездатності.

5. Оптимізація та масштабування: ула здійснена оптимізація роботи платформи з метою підвищення її продуктивності та ефективності, а також було сплановано її масштабування для роботи з великою кількістю користувачів.

6. Оцінка результатів: була проведена оцінка та аналіз результатів дослідження з метою визначення ефективності використаних методів та технологій і вибору найкращих практик для подальшого розвитку платформи.

Практичне значення одержаних результатів полягає в можливості створення та впровадження ефективної та інноваційної платформи для інтелектуального навчання англійської мови. Отримані результати дослідження

дозволять розробити оптимальну архітектуру та програмне забезпечення, які будуть відповідати вимогам користувачів та забезпечувати зручний та ефективний процес навчання. Це сприятиме покращенню мовних навичок студентів та фахівців у галузі програмного забезпечення, що має важливе значення для їхнього особистого та професійного розвитку. Крім того, створення такої платформи може мати широке застосування у навчальних закладах, компаніях та організаціях, які зацікавлені у підвищенні рівня англійської мови своїх співробітників або студентів.

Апробація результатів дослідження. Матеріали кваліфікаційної роботи були представлені на XI Міжнародна наукова конференція "Студентські наукові дискусії поза форматом, яка відбулась 11 квітня 2024 року в Університеті Короля Данила, також їх можна переглянути на сторінці університету.

Структура: розділи — 3. Загальний обсяг основної частини – 87 сторінок. Список використаних джерел містить — 20 позицій.

РОЗДІЛ 1. АНАЛІЗ КОНКУРЕНТІВ У СФЕРІ РОЗРОБКИ ПЛАТФОРМИ ДЛЯ ІНТЕЛЕКТУАЛЬНОГО НАВЧАННЯ АНГЛІЙСЬКОЇ МОВИ

У даному розділі проводиться аналіз та огляд наявних конкурентів у сфері розробки платформ для інтелектуального навчання англійської мови. Метою є вивчення їхніх технологічних рішень та підходів до вирішення схожих завдань, а також виявлення переваг і недоліків їхніх платформ. Це допоможе визначити ключові аспекти, які слід враховувати під час розробки власної платформи.

1.1 Вибір конкурентів та аналіз їхнього відкритого коду

У світі існує багато платформ для інтелектуального навчання англійської мови, які мають відкритий вихідний код і надають можливість співпраці та внесення власних внесків до розвитку програми. Декілька прикладів таких платформ включають:

1. Mnemosyne — це платформа для вивчення англійської мови за допомогою онлайн-флеш-карт, яка має відкритий вихідний код. Мова програмування Python, фреймворк Django.

2. AnkiDroid — це ще одна популярна платформа для навчання з відкритим вихідним кодом, аналогічно до Anki. Цей додаток призначений для використання на мобільних пристроях і дозволяє користувачам створювати та використовувати флеш-картки для ефективного навчання. Він пропонує користувачам можливість навчатися різних тем і предметів, запам'ятовуючи слова, фрази та інші важливі матеріали. Мова програмування Kotlin .

3. SuperMemo: це ще одна платформа, яка має відкритий вихідний код та спеціалізується на інтелектуальному навчанні. SuperMemo пропонує широкі можливості для створення та використання онлайн-флеш-карт, а також тестів для ефективного вивчення англійської мови. Мова програмування C#.

Mnemosyne — це онлайн платформа для навчання, що використовує онлайн-флеш-карти для ефективного запам'ятовування матеріалу. Mnemosyne реалізована мовою програмування Python, яка славиться своєю простотою в освоєнні та потужними інструментами для розробки програмного забезпечення. Python використовується для створення різних типів програм, включаючи веб-застосунки, наукові дослідження та інші. Його широка підтримка спільнотою, багатофункціональність і зручність розробки роблять його популярним вибором для різних проектів.

Щодо бібліотек, Mnemosyne використовує PyQt для створення графічного інтерфейсу користувача. PyQt — це обгортка Python для бібліотеки Qt, яка надає можливості для створення потужних інтерфейсів. PyQt має велику кількість готових елементів і зручну документацію. Однак, використання може вимагати додаткового навчання через складнішу структуру Qt.

Також, Mnemosyne використовує базу даних SQLite для зберігання даних. SQLite — це легка, вбудована база даних, яка не потребує окремого сервера і може працювати безпосередньо з файлами. Вона підтримує багато типів даних і має широкі можливості запитів. Проте, для великих обсягів даних або для роботи в розподілених середовищах, SQLite може бути не найкращим варіантом через обмежену масштабованість.

Отже, Mnemosyne використовує потужну комбінацію мови програмування Python та бібліотеки PyQt для створення зручного інтерфейсу користувача, а також бази даних SQLite для зберігання даних. Це дозволяє платформі забезпечувати швидку реакцію, простоту використання та надійність зберігання даних. Однак, слід бути обережними щодо масштабованості та розширюваності у випадку великих обсягів даних або складних інтерфейсів користувача.

AnkiDroid — це платформа для навчання, спеціалізована на використанні флеш-карток для ефективного запам'ятовування матеріалу, аналогічно до Anki. Вона реалізована мовою програмування Kotlin та використовує різноманітні технології для реалізації своєї функціональності. Щодо візуальної частини, AnkiDroid використовує HTML та CSS для створення структури та оформлення

віджетів інтерфейсу користувача. PyQt — це бібліотека Python, що використовується для створення графічного інтерфейсу. Вона надає доступ до різноманітних елементів інтерфейсу, таких як вікна, кнопки, поля введення тощо, і дозволяє їх налаштовувати та розміщувати на екрані. Серед інших компонентів використовуються SQLite для зберігання даних та інші бібліотеки для роботи з мережею, аудіо та зображеннями.

Однією з переваг AnkiDroid є простота використання та можливість налаштування функціоналу відповідно до потреб користувача. Вона також надає користувачам можливість створювати власні колоди флеш-карток та налаштовувати їх для оптимального навчання. Однак, слід зазначити, що AnkiDroid може мати обмежені можливості розвитку через недостатню активність спільноти та обмежені ресурси, а також обмежену підтримку користувачів в їх потребах.

Загалом, AnkiDroid є потужною платформою для навчання з відкритим вихідним кодом, яка надає користувачам можливість ефективно вивчати матеріал за допомогою флеш-карток та різноманітних інструментів. Однак, слід розглядати його можливості та обмеження при виборі платформи для навчання.

SuperMemo — це платформа для навчання, яка використовує метод повторення з інтервалами для ефективного запам'ятовування матеріалу. Ця платформа реалізована мовою програмування C# та використовує різноманітні технології для своєї функціональності.

Щодо візуальної частини, SuperMemo використовує HTML, CSS та WPF (Windows Presentation Foundation) для створення і оформлення графічного інтерфейсу користувача. HTML та CSS використовуються для створення структури та оформлення елементів інтерфейсу, таких як кнопки, поля введення тощо. WPF забезпечує більшу гнучкість у створенні графічного інтерфейсу для програм під управлінням операційної системи Windows.

Однією з переваг SuperMemo є його висока ефективність завдяки методу повторення з інтервалами, що дозволяє оптимізувати процес запам'ятовування

матеріалу. Крім того, платформа має розширені можливості для налаштування інтервалів повторення та ведення статистики навчання.

Однак, слід зазначити, що SuperMemo може мати обмежену підтримку для інших операційних систем, оскільки WPF підтримується тільки в операційних системах Windows. Також, розвиток платформи може бути обмежений через використання мови програмування C#, яка не так популярна серед розробників, як, наприклад, Python.

Загалом, SuperMemo є потужною платформою для навчання, яка використовує метод повторення з інтервалами для ефективного запам'ятовування матеріалу. Вона має ряд переваг, таких як висока ефективність та розширені можливості налаштування, але може мати обмеження у розвитку через обмежену підтримку операційних систем та мови програмування.

Під час аналізу трьох конкурентів у сфері платформ для навчання з відкритим вихідним кодом (Mnemosyne, AnkiDroid і SuperMemo) було виявлено, що кожна платформа має свої переваги та недоліки, залежно від мови програмування, на якій вона була реалізована.

Mnemosyne використовує мову програмування Python, що спрощує розробку і підтримку завдяки широкій підтримці спільноти та доступу до різноманітних бібліотек і фреймворків. AnkiDroid, натомість, використовує Java і Kotlin, що дозволяє використовувати специфічні бібліотеки для платформи Android, але може призвести до обмежень у розвитку через обмежену підтримку спільноти. SuperMemo, побудований на мові програмування C#, відзначається ефективністю методу повторення з інтервалами, але може мати обмежену підтримку для інших операційних систем та обмежені можливості розвитку через вибір мови програмування.

У візуальній частині Mnemosyne та AnkiDroid використовують HTML, CSS і JavaScript для створення красивого, динамічного та інтуїтивно зрозумілого інтерфейсу користувача. Ці технології полегшують навчання та забезпечують більшу гнучкість у створенні інтерфейсу. SuperMemo також може

використовувати ці технології, але обмеження мови програмування C# може призвести до складнощів у розробці.

Отже, використання HTML, CSS і JavaScript для візуальної частини платформи для навчання забезпечує більшу гнучкість та можливості для створення привабливого та інтуїтивно зрозумілого інтерфейсу користувача, що сприяє ефективному навчанню. У випадку вибору мови програмування для реалізації платформи для навчання, мова Python має переваги у зручності розробки та доступі до ресурсів, що сприяють подальшому розвитку програмного забезпечення.

1.2 Огляд існуючих конкурентів з вивченням англійської мови за допомогою флеш-карток

В моїй кваліфікаційній роботі я дослідив популярні платформи для навчання англійської мови, такі як Quizlet, Anki, Brainscape та Multilang. Моя мета полягає в аналізі переваг і недоліків кожної з цих платформ з метою використання цієї інформації для розробки власної платформи для інтелектуального навчання англійської мови за допомогою онлайн-флеш-карток та тестів. Під час дослідження я був зосереджений на виявленні недоліків та переваг конкурентів і використаю ці знання для створення більш ефективної та користувацько-орієнтованої платформи.

Платформа Quizlet використовує різні технології для створення як веб-сайту, так і мобільного додатку, що дозволяє користувачам ефективно вивчати матеріали за допомогою флеш-карток (рис. 1.1.) На веб-сайті використовується комбінація мов програмування, таких як JavaScript, HTML і CSS. JavaScript відповідає за динамічну взаємодію з користувачем, HTML структурує вміст сторінок, а CSS надає їм візуальне оформлення.

У мобільному додатку використовуються мови програмування Java або Kotlin для розробки на платформі Android і Swift або Objective-C для розробки на iOS. Ці мови дозволяють створювати функціональний інтерфейс користувача

та взаємодіяти з сервером. Для реалізації функціоналу мобільного додатку також можуть використовуватися різні бібліотеки та фреймворки, такі як Retrofit для роботи з мережею або Room для роботи з базою даних SQLite на Android.

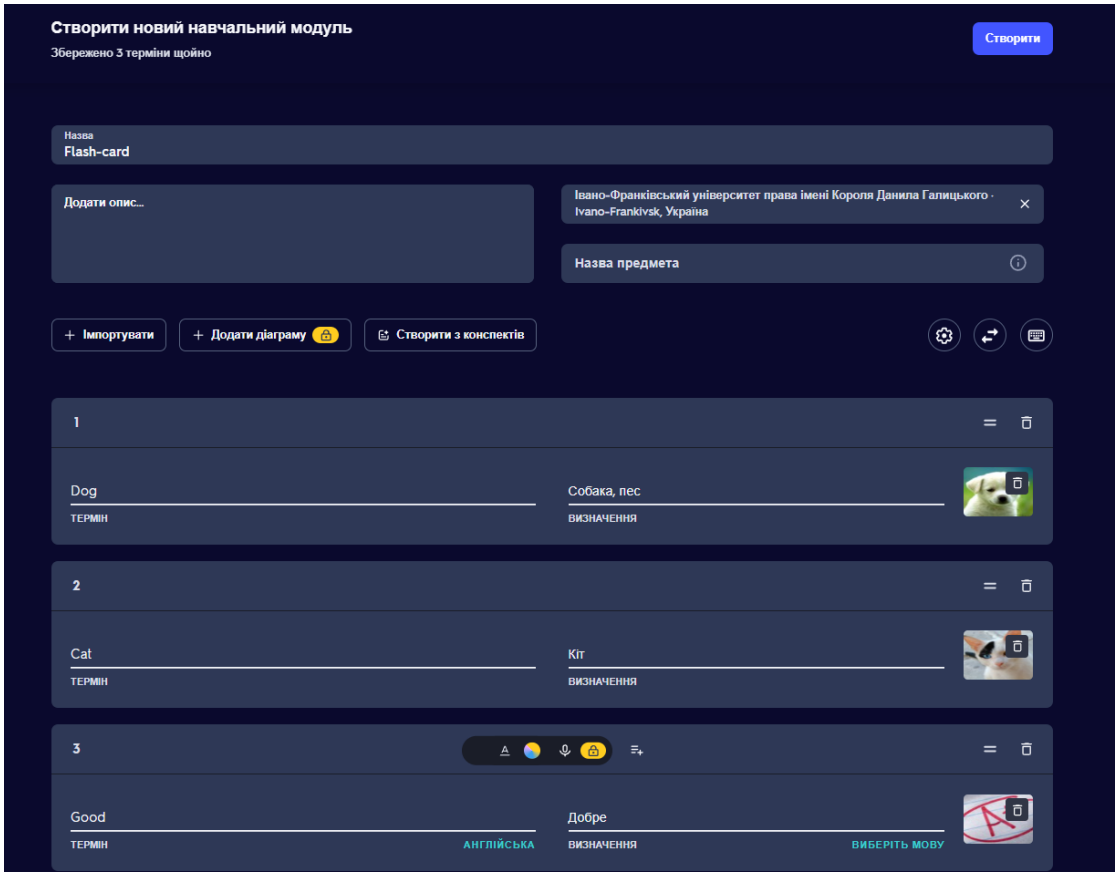


Рисунок 1.1 — Quizlet створення флеш-карток

Однією з основних функцій Quizlet є можливість створення та проходження тестів на основі навчальних карток (рис. 1.2), що допомагає у вивченні та запам'ятовуванні матеріалу. Крім того, платформа надає можливість використовувати готові навчальні матеріали, створені іншими користувачами, що робить процес навчання більш різноманітним та зацікавлюючим.

Quizlet також підтримує різноманітні навчальні режими, такі як письмові вправи, аудіо-візуальні підказки та ігри, що дозволяє користувачам обирати найбільш підходящий спосіб навчання.

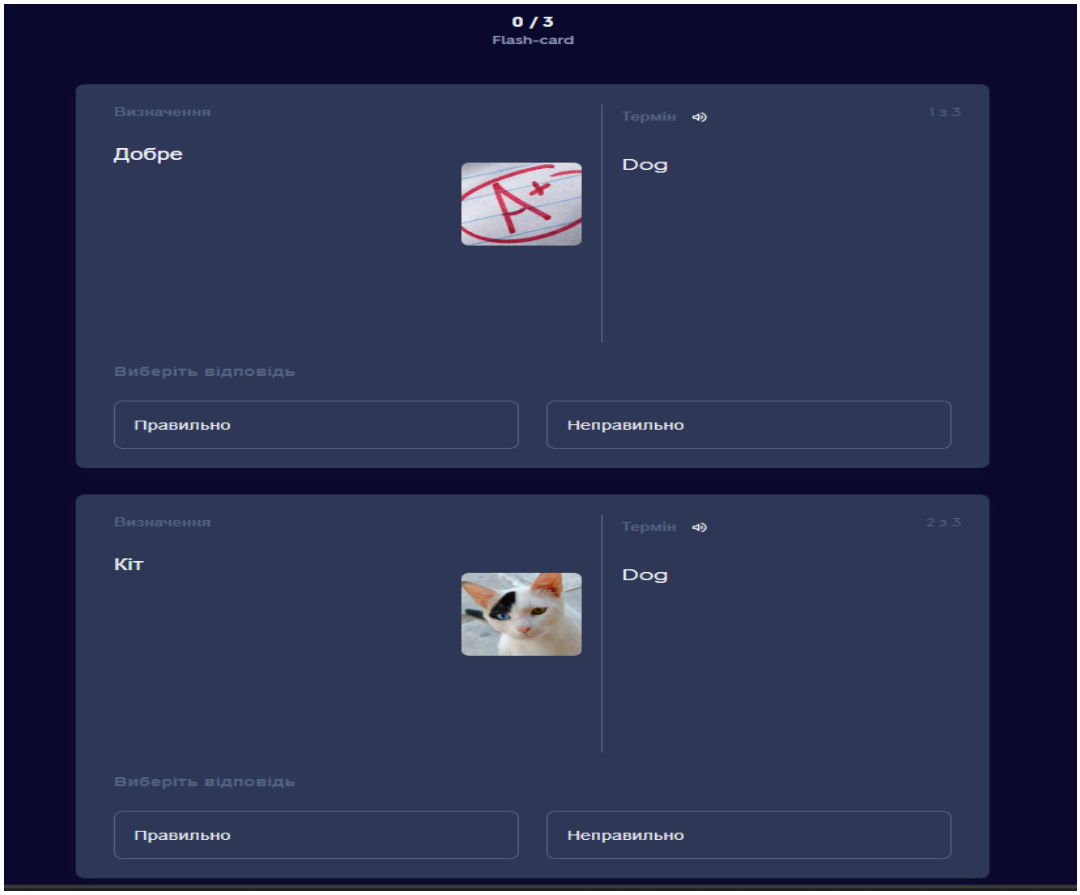


Рисунок 1.2 — Quizlet створення тестів

Quizlet є популярним серед студентів, вчителів та інших осіб, які прагнуть ефективно вивчати різноманітні предмети та теми. Він надає зручний та доступний інструмент для навчання та самонавчання, що робить його цінним ресурсом у сучасному освітньому середовищі.

Переваги Quizlet:

1. Зручний інтерфейс: Quizlet має простий і легкий у використанні інтерфейс, що робить його доступним для широкого кола користувачів.

2. Різноманітність матеріалів: платформа пропонує велику кількість готових навчальних матеріалів, які покривають різні предмети та теми.

3. Можливість спільного користування: користувачі можуть ділитися своїми навчальними колодами з іншими користувачами, що сприяє обміну знаннями та ідеями.

4. Наявність мобільного додатку: наявність мобільного додатку дозволяє користувачам вивчати матеріали в будь-який час та в будь-якому місці.

5. Тестування з миттєвими результатами: йункція проходження тестів на основі навчальних карток дозволяє швидко оцінити рівень знань та отримати миттєві результати.

Недоліки Quizlet:

1. Обмежена можливість редагування: у безкоштовній версії Quizlet обмежений функціонал редагування навчальних матеріалів, що може обмежити користувачів у створенні власних колод карток.

2. Необмежений доступ до матеріалів: не всі матеріали на Quizlet перевірені на точність та достовірність, що може призвести до використання неправильної або неповної інформації.

3. Залежність від Інтернет-підключення: для доступу до матеріалів та функцій Quizlet потрібне Інтернет-підключення.

4. Відсутність індивідуалізації: платформа не завжди надає індивідуальні рекомендації чи адаптується до потреб конкретного користувача, що може зменшити ефективність навчання для деяких осіб.

Розглядаючи другий конкурент Anki варто зазначити, що Anki є програмою для навчання за допомогою флеш-карток, доступною на різних платформах, включаючи комп'ютери, мобільні пристрої та планшети. Основна мета Anki полягає в покращенні запам'ятовування матеріалу через систематичне повторення, використовуючи індивідуально налаштовані алгоритми.

Що стосується мови програмування, на якій розроблено Anki, вона написана головним чином мовою програмування Python. Python використовується для створення і підтримки функціональності цієї програми, забезпечуючи її працездатність на різних платформах. Такий вибір мови програмування дозволяє Anki бути гнучким і доступним для користувачів на різних пристроях, сприяючи ефективному процесу навчання за допомогою флеш-карток (рис. 1.3).

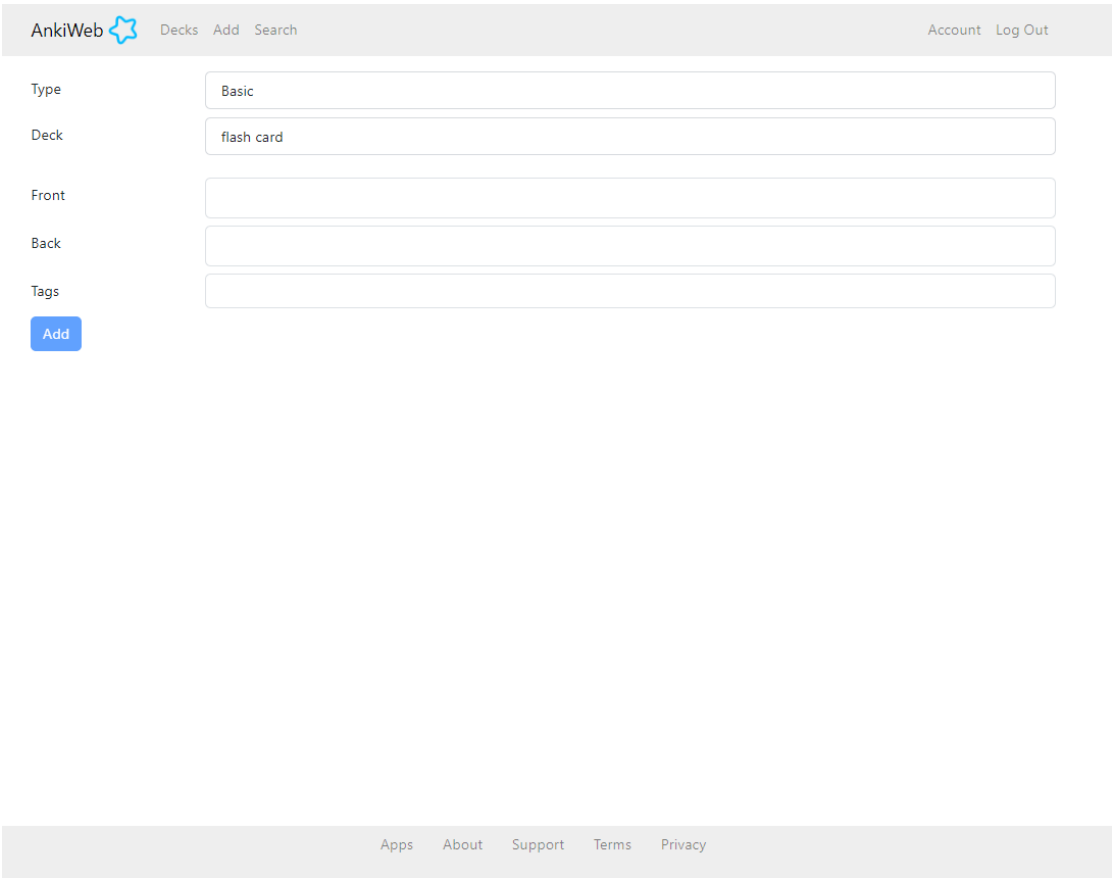


Рисунок 1.3 — Anki створення флеш-карток

Система Anki дозволяє користувачам створювати власні навчальні колоди карток, до яких можна додавати тексти, зображення, аудіо або відео матеріали. Ці картки можуть бути використані для вивчення різних тем, включаючи мови, науку, історію та інші області знань.

Переваги Anki:

1. Ефективність: Anki використовує систему повторення з використанням алгоритмів, які допомагають ефективніше запам'ятовувати матеріал.

2. Налаштованість: користувачі можуть індивідуально налаштовувати параметри повторення відповідно до своїх потреб та швидкості вивчення.

3. Гнучкість: Anki доступний на різних платформах, що дозволяє користувачам використовувати його на комп'ютерах, мобільних телфонах, смартфонах та планшетах.

Недоліки Anki:

1. Інтерфейс веб-платформи: веб-версія Anki може бути менш зручною та погано виглядати порівняно з мобільним додатком, що може вплинути на користувачів, які використовують його через веб-браузер.

2. Дизайн: дизайн веб-платформи Anki може виглядати менш привабливо порівняно з іншими аналогічними сервісами, що може вплинути на користувачів, які звикли до більш естетичного інтерфейсу.

3. Навчальна крива: для деяких користувачів може знадобитися певний час для засвоєння методу Anki та розуміння його системи повторення.

З урахуванням цього, для розробки власного веб-платформи, планується використання гнучкості підходу Anki до налаштувань, щоб дозволити користувачам індивідуально налаштовувати параметри навчання відповідно до їхніх потреб та швидкості вивчення.

Brainscape — це онлайн-платформа, яка спеціалізується на створенні та використанні інтерактивних флеш-карток для навчання (рис. 1.4). Щодо мов програмування, на яких написаний веб-сайт Brainscape, основні технології включають JavaScript, HTML та CSS для реалізації клієнтської частини. JavaScript використовується для динамічної взаємодії з користувачем, HTML — для структуризації вмісту сторінок, а CSS — для візуального оформлення.

Щодо бекенду, він написаний на одній з серверних мов програмування, таких як Python, а також може використовувати різноманітні фреймворки та бібліотеки для розробки веб-додатків. Використання Python дозволяє забезпечити надійність та масштабованість серверної частини, а такі фреймворки, як Django або Flask, можуть бути використані для швидкої розробки та підтримки веб-додатків. Крім того, Brainscape може використовувати бази даних, такі як PostgreSQL або MySQL, для зберігання та управління великими обсягами навчальних матеріалів. Інтеграція з різними API та сервісами також може бути реалізована для додаткового розширення функціональності платформи.

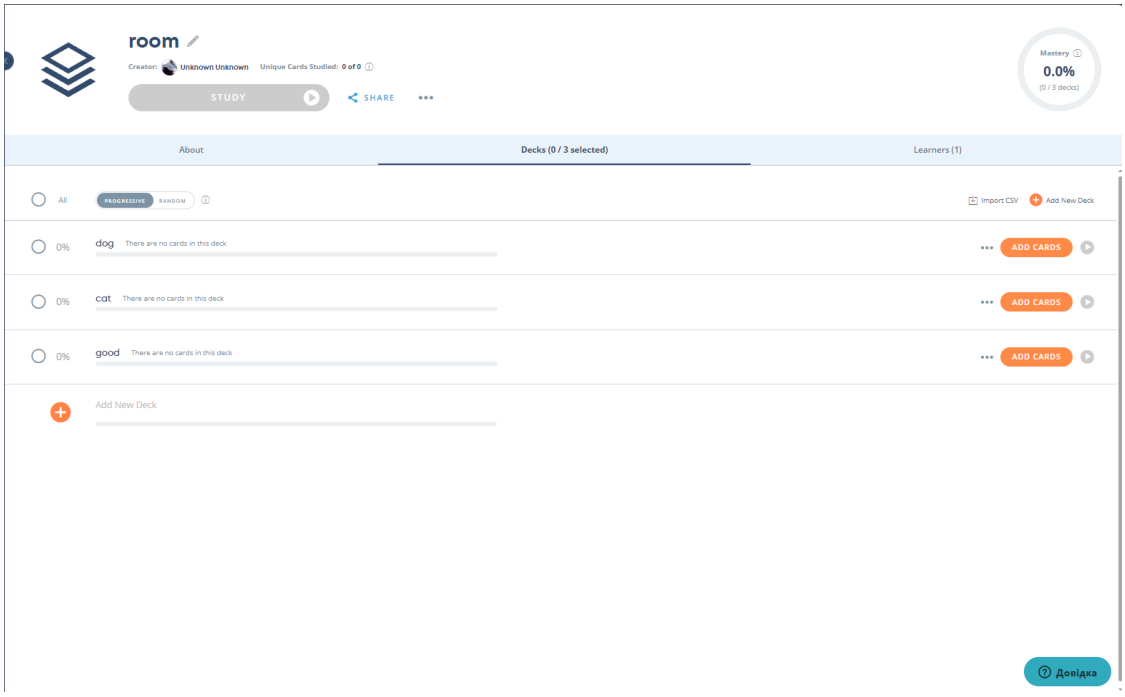


Рисунок 1.4 — Brainscape кімната для створення флеш-карток

Вона дозволяє користувачам створювати власні колоди карток або використовувати готові набори для вивчення різних предметів (рис. 1.5.), включаючи мови.

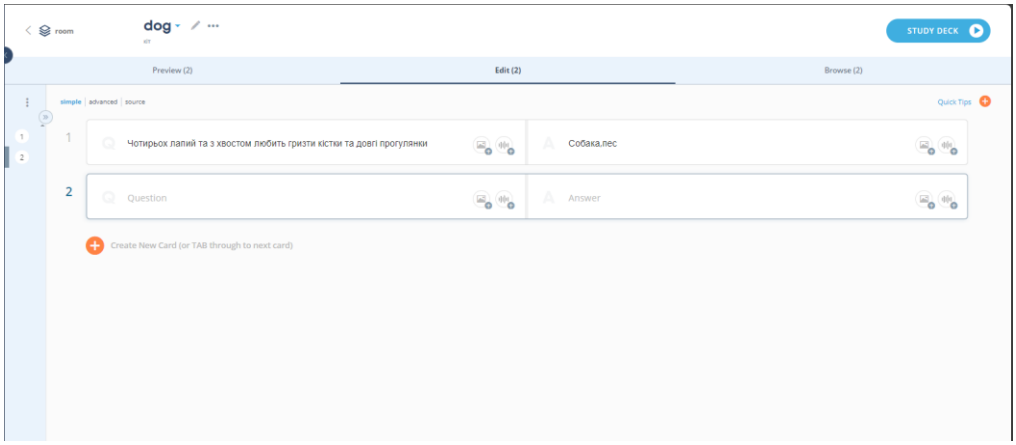


Рис. 1.5 - Brainscape створення флеш-карток

Платформа пропонує інноваційний підхід до навчання, використовуючи методику повторення, яка допомагає ефективно запам'ятовувати матеріал. Крім

того, Brainscape надає можливість використовувати інтерактивні тести та інші навчальні інструменти для покращення знань користувачів.

Переваги Brainscape:

1. Інтерактивні флеш-картки: Brainscape пропонує інтерактивний формат навчання за допомогою флеш-карток, що сприяє кращому запам'ятовуванню матеріалу та вивчення ангельської мови.

2. Гнучкість навчання: платформа дозволяє користувачам створювати власні колоди карток або використовувати готові набори для навчання.

3. Методика повторення: Brainscape використовує ефективні методи повторення, що допомагають оптимізувати процес запам'ятовування.

Недоліки Brainscape:

1. Обмежена безкоштовна версія: деякі функції та матеріали можуть бути недоступні у безкоштовній версії платформи.

2. Обмежена кількість навчальних матеріалів: хоча Brainscape пропонує готові набори для навчання, але їхня кількість може бути обмеженою порівняно з іншими платформами.

3. Відсутність інтерактивних тестів: у порівнянні з іншими платформами, Brainscape може не мати такої розширеної функціональності для проведення інтерактивних тестів.

Brainscape відзначається інтерактивним підходом до навчання та забезпечує ефективний спосіб вивчення за допомогою флеш-карток. Платформа дозволяє користувачам самостійно створювати картки або використовувати готові навчальні матеріали. Особливою перевагою Brainscape є алгоритм вивчення, який адаптується до індивідуальних потреб користувача та оптимізує процес запам'ятовування матеріалу.

Для покращення власної веб-платформи можна врахувати гнучкість налаштувань навчального процесу, яка дозволить користувачам більш ефективно адаптувати навчання до своїх потреб. Також варто звернути увагу на простоту та зручність інтерфейсу, щоб забезпечити комфортне користування платформою та вивченням.

Multilang з підходу, специфічного для кваліфікаційної роботи: Multilang — це веб-платформа, розроблена для навчання англійської мови, що надає різноманітні можливості для користувачів покращити свої мовні навички. Вона пропонує доступ до різноманітних вправ, тестів, відеоуроків, аудіоматеріалів та флеш-карток (рис. 1.6). Такий ресурс може бути корисним для тих, хто прагне покращити свої навички англійської мови, особливо з урахуванням його доступності онлайн. Інфраструктура та технології, використані в розробці Multilang, демонструють високий рівень сучасності та ефективності. Основною мовою програмування для розробки платформи є Python, що відомий своєю простотою та гнучкістю. Використання фреймворка FastAPI дозволяє реалізувати швидке та ефективне API.

SQLAlchemy використовується для роботи з базами даних, забезпечуючи надійну та продуктивну роботу з даними. Використання Hugging Face Transformers та Fluent:lang свідчить про використання сучасних методів обробки природної мови та моделей машинного перекладу для забезпечення високої якості перекладу та роботи з мовою.

Додатково, використання Docker для контейнеризації та розгортання, а також GitHub для зберігання коду та відстеження змін, свідчать про високий рівень автоматизації та організації розробницького процесу. Використання CI/CD для автоматизації тестування та розгортання дозволяє забезпечити стабільну та надійну роботу платформи Multilang.

Використання Docker значно спрощує процес розгортання додатків, забезпечуючи ізольоване середовище для кожного компонента системи, що покращує масштабованість та керованість платформи. GitHub дозволяє розробникам ефективно співпрацювати, відстежувати зміни в коді та інтегрувати нові функції без порушення роботи існуючих. CI/CD інструменти, такі як Jenkins або GitHub Actions, забезпечують автоматичне тестування та безперервне розгортання нових версій додатка, що дозволяє швидко виявляти та виправляти помилки, підвищуючи якість програмного забезпечення.

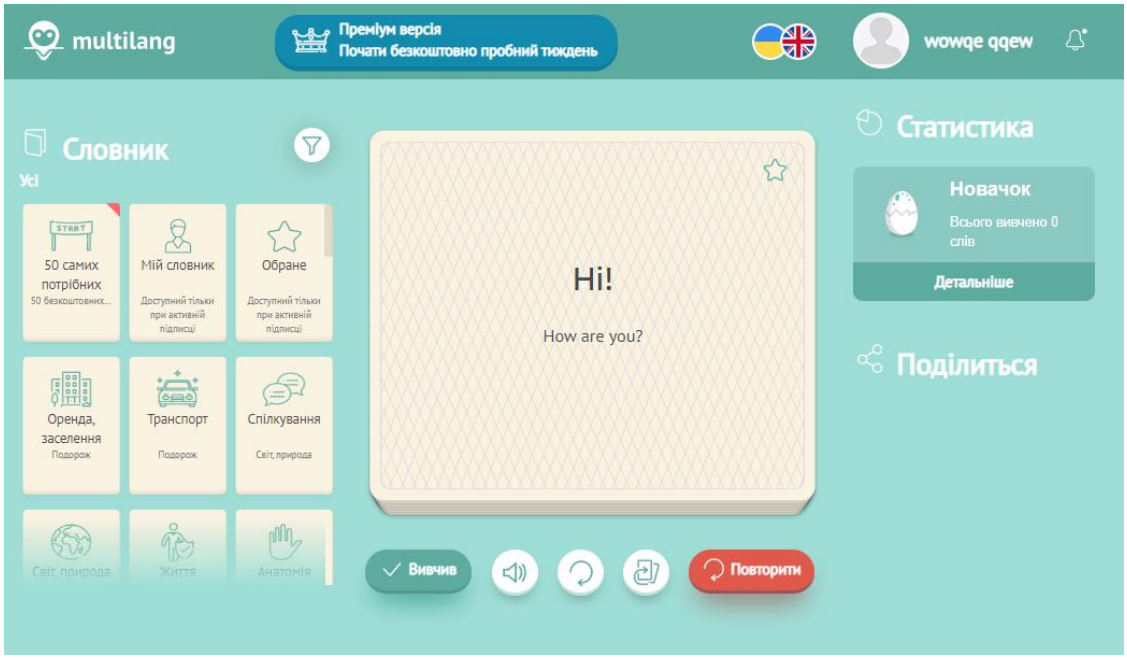


Рисунок 1.6 — Multilang головна сторінка веб-платформи

Переваги платформи Multilang полягають у можливості вивчення англійської мови через різноманітні вправи та інтерактивні завдання, а також у наявності вбудованого словника та інших інструментів для підтримки навчання. Однак у Multilang є недоліки, такі як відсутність можливості створення власних флеш-карток, що обмежує користувачів у персоналізації навчального матеріалу, а також обмежена кількість доступних навчальних ресурсів та вправ, що може ускладнити розширення знань користувачів.

Multilang використовує мову програмування Python з фреймворком FastAPI, що робить його гнучким і ефективним. Ця платформа дозволяє вивчати англійську мову через різноманітні вправи та інтерактивні завдання, надаючи доступ до вбудованого словника та інших інструментів навчання. Однак обмеженість можливостей створення власних флеш-карток може бути недоліком для користувачів, що шукають більш індивідуалізований підхід. Програмування на Python дозволяє легко розширювати можливості платформи, а FastAPI допомагає забезпечити швидку та ефективну роботу Multilang.

Висновки до розділу 1

У розділ 1 проаналізовано та проведено оглядає наявних конкурентів у сфері розробки платформ для інтелектуального навчання англійської мови за допомогою онлайн-флеш-карт та тестів. Наведені приклади конкурентів включають Mnemosyne, AnkiDroid та SuperMemo, які мають відкритий вихідний код і спеціалізуються на різних аспектах інтелектуального навчання.

Постановка задачі: з урахуванням аналізу конкурентів у галузі інтелектуального навчання англійської мови виникає необхідність в розробці веб-платформи з використанням власної унікальної концепції. Мета проекту полягає в створенні інноваційної платформи, яка надає користувачам можливість ефективно навчатися англійської мови шляхом використання інтерактивних методів та зручного інтерфейсу.

Недоліки конкурентів:

1. Mnemosyne: хоча Mnemosyne має простий і надійний інтерфейс, вона може зіткнутися з обмеженнями щодо масштабованості та розширюваності через використання PyQt та бази даних SQLite.

2. AnkiDroid: недостатня активність спільноти та обмежена підтримка можуть ускладнити розвиток AnkiDroid.

3. SuperMemo: незважаючи на широкі можливості, відсутність відкритого вихідного коду ускладнює співпрацю та внесення власних внесків до розвитку.

4. Multilang: неможливість створення власних флеш-карток може обмежити персоналізацію навчального матеріалу для користувачів.

5. Brainscape: обмеження у доступі до певних функцій за плату та відсутність можливості модифікації програмного забезпечення через закритий характер платформи ускладнюють використання Brainscape.

6. Quizlet: обмеження у функціоналі для безкоштовних користувачів та відсутність можливості модифікації додатка через закритий вихідний код ускладнюють використання Quizlet.

Отже, розробка веб-платформи для інтелектуального навчання англійської мови має потенціал вирішити недоліки існуючих конкурентів шляхом створення зручного та ефективного середовища для навчання, яке буде відкритим для розвитку та співпраці.

РОЗДІЛ 2. РОЗРОБКА ВЕБ-ПЛАТФОРМИ ДЛЯ ІНТЕЛЕКТУАЛЬНОГО НАВЧАННЯ АНГЛІЙСЬКОЇ МОВИ

У цьому розділі описано процес розробки веб-платформи для інтелектуального навчання англійської мови за допомогою онлайн-флеш-карт та тестів. Платформа була розроблена з метою надання користувачам можливості ефективного та зручного вивчення англійської мови через інтерактивні завдання та спеціально підібрані навчальні матеріали.

У процесі розробки платформи використовувалися різноманітні технології та бібліотеки, серед яких варто відзначити:

1. Фреймворк Django для створення веб-застосунків та логіки проекту, на мові програмування Python.

2. Бібліотека Bootstrap для швидкого та зручного розробки адаптивного та стильного інтерфейсу користувача.

3. Система управління базами даних SQLite3 для зберігання та керування даними, що використовуються в платформі.

4. Середовище розробки PyCharm для комфортної та продуктивної роботи над проектом.

5. Організації веб-сторінок та їхнього стильного оформлення використовуються мови та технології HTML, CSS і JavaScript.

Кожна з цих технологій та бібліотек відіграла важливу роль у забезпеченні функціональності та зручного інтерфейсу для користувачів. У наступних розділах буде надано детальний опис використаних технологій, їх роль у процесі розробки платформи та особливості їх використання. Також буде проаналізовано внесок кожної з цих технологій у створення функціоналу платформи та забезпечення її ефективності та зручності для користувачів.

Перед початком розробки платформи було створено чорно-білий прототип у Figma. Цей прототип дозволив визначити основну структуру та функціональність платформи, а також виявити потенційні проблеми та вирішити

їх на ранніх етапах розробки. Прототип відображають структуру та функціональність платформи і служать основою для подальшої розробки та уточнення дизайну інтерфейсу користувача. Крім цього, кожна ключова функціональна можливість платформи буде описана у вигляді "use case", що дозволяє краще зрозуміти, як користувачі використовуватимуть платформу та які операції вони будуть здійснювати для досягнення своїх цілей. Ці елементи доповнять розгляд кожної технології та бібліотеки, надаючи повніший уявлення про процес розробки та функціональність платформи.

2.1 Проектування і прототипування платформи

Першим кроком у розробці нашої платформи для інтелектуального навчання англійської мови було створення прототипу інтерфейсу користувача. Для цього ми використали інструмент Figma — потужний веб-додаток, призначений для дизайну, прототипування та спільної роботи над проектами. Figma дозволяє створювати візуальні прототипи, визначати основні елементи інтерфейсу та взаємодіяти з ними безпосередньо в браузері. В цьому підрозділі ми детально розглянемо процес створення чорно-білого прототипу в Figma, його структуру та функціональність, а також розглянемо важливі аспекти дизайну і взаємодії з користувачем, визначені у прототипі.

Зображення показує сторінку реєстрації для веб-сайту (рис. 1.8), де користувачі можуть створити свій обліковий запис. На сторінці присутні поля для введення адреси електронної пошти, пароля та його підтвердження, а також прапорець для прийняття умов використання веб-сайту. Крім того, є кнопка для завершення реєстрації та посилання на сторінку входу. Текстові елементи надають підказки щодо введення даних, а також інформують про можливість відновлення паролю.

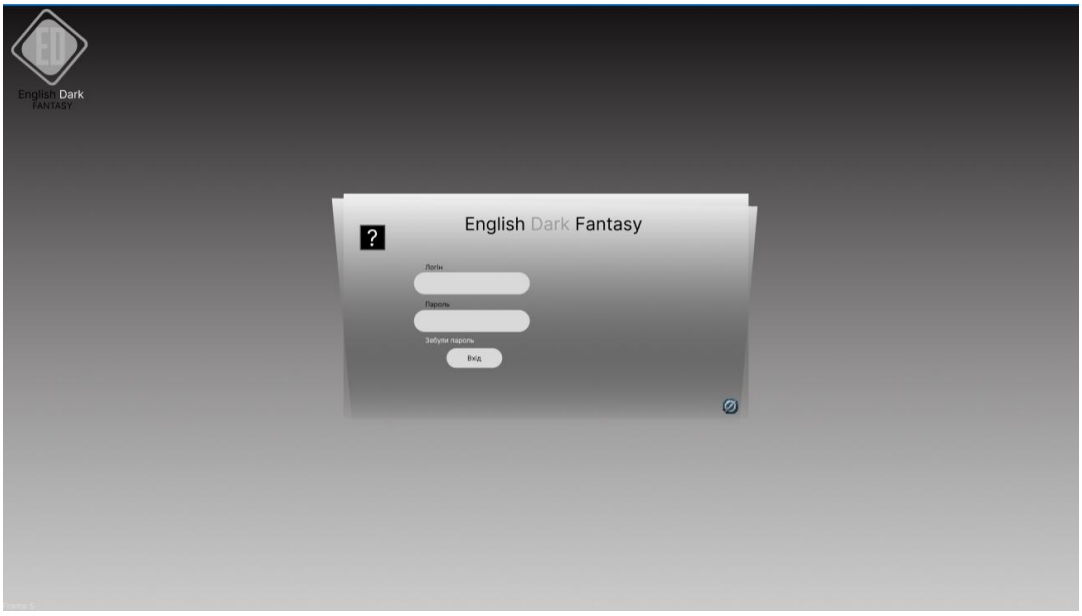


Рисунок 2.1 — Прототип Figma сторінка реєстрації

2.2 "Lean Canvas: стратегічне проектування платформи для інтелектуального навчання англійської мови"

Lean Canvas — це інструмент, спрямований на опис ключових аспектів бізнесу чи проекту, забезпечуючи компактну та ефективну стратегію. Використання Lean Canvas дозволяє чітко визначити бізнес-модель, цільову аудиторію, пропозицію вартості та інші ключові аспекти, що допомагає спрямувати зусилля на найважливіші напрямки розвитку та досягти успіху.

В контексті нашого проекту, який полягає у створенні освітнього веб-сайту з флеш-картками для вивчення англійських слів та тестів, Lean Canvas стає важливим інструментом для ефективного стратегічного проектування. Завдяки Lean Canvas ми можемо ідентифікувати ключові проблеми, з якими стикаються наші користувачі, та запропонувати цінні рішення, що відповідають їхнім потребам. Крім того, Lean Canvas допомагає визначити канали взаємодії з користувачами, структуру витрат та джерела доходу, що сприяє побудові стійкої та успішної бізнес-моделі. Це дозволяє оптимізувати ресурси та зусилля, зосереджуючись на важливих аспектах розвитку нашого освітнього продукту.

1. Розділ "Сегменти споживачів і ранні послідовники" містить аналіз різних груп користувачів за такими характеристиками, як стать, вік і рівень володіння мовою. У нашому випадку, різні групи класифікуються за рівнем володіння мовою, типом користувацького пристрою або метою вивчення.

2. У розділі "Проблема та існуючі альтернативи" виявлені проблеми, з якими стикаються користувачі, такі як труднощі в запам'ятовуванні слів, недостатність інтерактивності в існуючих методах навчання та високі витрати на курси англійської мови.

3. Розділ "Унікальна цінність" наводить інформацію про ключові переваги нашого продукту, зокрема інтерактивний метод вивчення слів за допомогою флеш-карток, можливість створення власного словника та персоналізовані тести для перевірки знань.

4. В "Рішенні проблеми" описано, як наш продукт ефективно вирішує проблеми користувачів шляхом інтерактивного вивчення слів та персоналізованих тестів для перевірки знань.

5. "Канали просування" включають створення спеціалізованої веб-платформи з флеш-картками та рекламні кампанії для забезпечення максимального охоплення цільової аудиторії.

6. "Джерела прибутку" описуються як підписки для доступу до додаткових функцій та рекламні доходи від незареєстрованих користувачів.

7. Розділ "Структура витрат" фокусується на витратах на обслуговування серверів, оренду технічної інфраструктури та рекламні кампанії.

8. "Ключові метрики" визначаються як частка ринку, трафік на сайті, конверсія користувачів та задоволеність користувачів для оцінки успішності свого проекту .

9. У розділі "Прихована перевага" підкреслюється інноваційний підхід до вивчення англійської мови через інтерактивні флеш-картки та персоналізовані тести, що робить навчання більш ефективним та захоплюючим.

Рисунок 2.2 — Leancanvas

2.3 Вибір мови програмування

Вибір мови програмування для проекту є критичним етапом, що впливає на його успішність. Обравши Python, я керувався кількома ключовими факторами, які відображають потреби та цілі проекту.

Простота вивчення та використання Python робить його ідеальним вибором для широкого кола розробників, включаючи початківців. Його читабельний синтаксис сприяє швидкому освоєнню основ програмування, що є важливим фактором для мене.

Розгалужена екосистема Python, що включає багато бібліотек та фреймворків, дозволяє ефективно реалізувати різноманітні функціональні можливості проекту. Наприклад, Django та Flask роблять розробку веб-додатків простою та ефективною.

Активна спільнота розробників Python надає широкий спектр ресурсів, від документації та уроків до форумів та підтримки. Це забезпечує доступ до якісної допомоги та рішень проблем під час розробки. Швидкість розробки є ще однією перевагою Python. Завдяки вбудованим структурам даних та великій кількості

бібліотек можна швидко створювати прототипи та розвивати функціонал. Додатково, підтримка асинхронного програмування у новіших версіях Python розширює можливості для реалізації веб-платформ з високою продуктивністю та швидкістю реакції, що важливо для проектів, де велике значення має швидкість.

2.4 Django — Фреймворк для Швидкої Розробки Веб-Додатків

Django — це високорівневий Python фреймворк, призначений для швидкої та ефективної розробки веб-додатків. Завдяки своїй потужній функціональності та простоті використання, Django здобув велику популярність серед розробників.

Простота використання: однією з ключових переваг Django є його простота використання. Фреймворк надає розробникам гнучкість та швидкість у створенні веб-додатків завдяки готовому набору інструментів та вбудованим функціям. Це дозволяє зосередитися на розробці функціоналу, не витрачаючи час на повторне вибудовування загальних компонентів [10].

Розширені можливості: Django надає широкі можливості для розробки веб-додатків будь-якої складності. За допомогою вбудованих модулів для автентифікації, керування адміністративними панелями та роботи з базами даних, розробники можуть швидко реалізувати різноманітні функції свого проекту та швидке написання.

Активна спільнота та ресурси: однією з переваг Django є його велика та активна спільнота розробників. Це забезпечує доступ до безлічі ресурсів, таких як документація, уроки, форуми та пакети розширень. Розробники можуть швидко знаходити відповіді на свої питання та ділитися досвідом з іншими учасниками спільноти.

Зручна адміністративна панель: однією з найпопулярніших особливостей Django є його вбудована адміністративна панель. Цей інструмент дозволяє адміністраторам легко керувати контентом веб-додатку, не потребуючи власного розроблення окремої адміністративної системи.

Звичайно, ще однією значущою перевагою Django є його висока захищеність. Фреймворк має вбудовані механізми захисту від типових вразливостей, таких як атаки SQL-ін'єкцій, Cross-Site Scripting (XSS) та Cross-Site Request Forgery (CSRF). Безпека є критичним аспектом будь-якого веб-додатку, і Django забезпечує розробників надійними засобами для захисту своїх проектів. Крім того, Django пропонує вбудовану підтримку міжнародних мов та локалізацію, що робить його ідеальним вибором для створення мультилінгвальних веб-додатків. Це означає, що розробники можуть легко адаптувати свої проекти для різних мовних аудиторій без значних зусиль.

Ще однією важливою особливістю Django є його масштабованість. Фреймворк надає зручні інструменти для роботи з великими обсягами даних та високими навантаженнями. Вбудовані можливості кешування, розподілені сесії та підтримка асинхронних запитів допомагають забезпечити високу продуктивність веб-додатків навіть при великому трафіку.

Отже, Django не лише пропонує швидку розробку та велику кількість ресурсів для розробників, але також забезпечує високий рівень безпеки, підтримку міжнародної локалізації та здатність масштабування, що робить його потужним інструментом для будь-яких веб-проектів.

2.5 Основи HTML та його структура

HTML, або HyperText Markup Language, є основною мовою для створення веб-сторінок. Це мова розмітки, яка визначає структуру та зміст веб-сторінок за допомогою спеціальних тегів. HTML відповідає за розташування елементів на сторінці, їх ієрархію та візуальний вигляд. Основними компонентами HTML-документу є теги, які визначають різні елементи сторінки, такі як заголовки, параграфи, посилання та зображення.

Основні теги HTML: у HTML є широкий набір тегів, кожен з яких має своє призначення. Наприклад, тег <html> визначає початок і кінець HTML-документу, теги <head> та <body> відповідають за метадані та зміст веб-сторінки

відповідно. Основні теги, такі як <h1>, <p>, <ul>, <a>, використовуються для визначення заголовків, параграфів, списків та посилань відповідно.

Структура HTML-документу: HTML-документ складається з різних елементів, таких як заголовок, тіло, метатеги тощо. Організація цих елементів у документі визначає його структуру. Теги розташовуються у документі відповідно до їхніх відносин та вкладеності. Наприклад, тіло документа зазвичай розпочинається тегом <body>, який містить усі видимі елементи сторінки.

Валідність та семантика HTML: валідність HTML відіграє важливу роль у забезпеченні правильної роботи та інтерпретації веб-сторінок браузерами. Використання семантичних тегів, які відповідають за значення та зміст елементів, допомагає пошуковим системам краще розуміти зміст сторінки та покращує її доступність для користувачів з обмеженими можливостями.

Форми та інтерактивні елементи: форми дозволяють користувачам взаємодіяти з веб-сторінками, надсилаючи дані на сервер. HTML надає різноманітні елементи форм, такі як текстові поля, кнопки, випадаючі списки, які дозволяють збирати різноманітні дані від користувачів.

Мультимедійні елементи: HTML також має можливості для вбудовування мультимедійного контенту, такого як зображення, відео та аудіо. Ці елементи дозволяють створювати більш привабливі та інтерактивні веб-сторінки для користувачів.

Інші розширені можливості HTML: HTML надає різноманітні можливості для створення складних інтерфейсів та відображення різноманітного контенту. Від тегів та , які використовуються для групування елементів, до таблиць, вставок і фреймів, HTML дозволяє створювати різноманітні веб-сторінки з різними типами контенту.

2.6 Основи CSS та його застосування

CSS, або Cascading Style Sheets, є мовою опису стилів, яка використовується для визначення зовнішнього вигляду веб-сторінок, наприклад,

кольору, шрифту, розміру та розміщення елементів. Використання CSS дозволяє відокремити вміст веб-сторінки від її представлення, що сприяє більш гнучкому та ефективному управлінню стилями та макетом сторінки.

Основні принципи CSS: CSS використовує селектори для вибору елементів на веб-сторінці та властивості, щоб задати їхні стилі. Наприклад, селектор p вибирає всі параграфи, а властивість color визначає колір тексту. Крім того, CSS дозволяє використовувати класи, ідентифікатори та псевдокласи для більш точного визначення стилів.

Типи CSS-властивостей: CSS має широкий спектр властивостей, які визначають різні аспекти вигляду елементів. Наприклад, властивості font-family, font-size та font-weight використовуються для визначення шрифту та його параметрів, тоді як властивості margin, padding та border використовуються для визначення зовнішнього вигляду елементів та їхнього розміщення на сторінці.

Каскадування та спадкування: каскадування в CSS визначає, які стилі застосовуються до конкретного елемента в залежності від його специфічності та пріоритетності. Спадкування дозволяє стилям елемента успадковувати значення властивостей від його батьківських елементів. Ці концепції дозволяють легко та ефективно управляти стилями на веб-сторінці.

Респонсивний дизайн та медіа-запити: респонсивний дизайн у CSS дозволяє сторінкам адаптуватися до різних розмірів екранів та пристроїв. Медіа-запити в CSS дозволяють встановлювати різні стилі в залежності від характеристик пристрою, таких як ширина екрану та орієнтація.

Анімація та переходи: CSS надає можливості для створення анімацій та переходів, що додають живий та привабливий вигляд до веб-сторінок. Використання ключових кадрів та функцій переходу дозволяє керувати рухом та змінами стилів елементів.

Браузерна сумісність та вендорні префікси: одним із викликів CSS є забезпечення сумісності стилів з різними веб-браузерами. Деякі функції CSS можуть потребувати вендорних префіксів для правильної роботи в різних браузерах, що потребує уважного управління стилями.

2.7 JavaScript: основи та можливості

JavaScript є мовою програмування, яка використовується для надання динаміки та інтерактивності на веб-сторінках. Він широко використовується для створення веб-додатків, веб-ігор, а також для розробки серверних застосунків та мобільних додатків за допомогою фреймворків, таких як Node.js та React Native.

Основні функції та можливості JavaScript: JavaScript надає ряд потужних функцій, включаючи можливість маніпулювання HTML-документами, керування подіями користувача, взаємодію з сервером за допомогою AJAX-запитів, а також створення анімацій та ефектів на веб-сторінці.

Сучасні стандарти та синтаксис: JavaScript постійно розвивається, і останні версії стандарту ECMAScript включають нові функції та покращення мови. Нові концепції, такі як стрілкові функції, деструктуризація та асинхронні функції, дозволяють писати більш чистий та ефективний код.[1]

Робота з DOM (Document Object Model): однією з ключових можливостей JavaScript є можливість взаємодіяти з DOM, тобто представленням HTML-документів у вигляді структури об'єктів. JavaScript дозволяє додавати, змінювати та видаляти елементи зі сторінки, обробляти події користувача та динамічно оновлювати вміст сторінки.

Функціональне програмування та об'єктно-орієнтоване програмування: JavaScript підтримує різні парадигми програмування, включаючи функціональне програмування та об'єктно-орієнтоване програмування. Це дозволяє розробникам писати більш структурований та модульний код, який є легшим у розумінні та розвитку.

Робота з веб-застосунками та API: JavaScript використовується для взаємодії з різними API, такими як API веб-серверів, соціальних мереж та картографічних сервісів. Це дозволяє розробникам створювати різноманітні веб-додатки та інтегрувати їх з різними сервісами та джерелами даних.

Фреймворки та бібліотеки: JavaScript має велику кількість фреймворків та бібліотек, таких як React, Angular та Vue.js, які спрощують розробку веб-додатків та забезпечують широкий функціонал для взаємодії з користувачем та обробки даних.

2.8 Bootstrap: інструмент для швидкого розвитку веб-інтерфейсів

Bootstrap — це потужний фреймворк для розробки веб-інтерфейсів та адаптивних мобільних сайтів. Він базується на HTML, CSS та JavaScript і надає набір готових компонентів та стилів, які допомагають створити сучасний та стильний дизайн веб-додатку.

Основні функції та можливості Bootstrap: Bootstrap має велику кількість готових елементів, таких як кнопки, форми, таблиці, навігаційні панелі та інші, що дозволяє швидко побудувати зручний та функціональний інтерфейс для веб-додатку. Крім того, він має вбудовану підтримку адаптивного дизайну, що дозволяє створювати сайти, які добре виглядають на будь-яких пристроях, від комп'ютерів до смартфонів.

Компоненти та класи Bootstrap: однією з ключових особливостей Bootstrap є його система класів, яка дозволяє швидко та легко додавати стилі до елементів HTML. Наприклад, класи "btn" використовуються для стилізації кнопок, а класи "container" та "row" використовуються для організації контенту на сторінці.

Підтримка розширень та тем: Bootstrap має велику кількість розширень та тем, які дозволяють розширити його функціонал та змінити вигляд за допомогою додаткових стилів та компонентів. Це дозволяє розробникам швидко налаштувати вигляд та функціонал своїх веб-додатків з урахуванням власних потреб та вимог проекту.

Підтримка спільноти та документація: Bootstrap має велику та активну спільноту розробників, яка надає безліч ресурсів, від документації та уроків до

шаблонів та розширень. Це дозволяє швидко знаходити відповіді на питання та вирішувати проблеми під час розробки на базі Bootstrap.

Шаблони та готові рішення: Bootstrap надає готові шаблони та компоненти, які можна використовувати для швидкого створення веб-сайтів та адаптації їх до конкретних потреб проекту. Це дозволяє ефективно використовувати час та зусилля при розробці веб-додатків.

2.9 SQLite3: легка та потужна вбудована база даних

SQLite3 — це компактна та легка використання вбудована база даних, яка надає можливість зберігати та керувати даними без необхідності окремого серверу баз даних. Вона широко використовується в різних типах додатків, від мобільних додатків до веб-серверів.

Основні функції та можливості SQLite3: SQLite3 підтримує багато стандартних функцій баз даних, таких як створення таблиць, виконання SQL-запитів, індексація даних та транзакційна безпека. Вона працює зі стандартом SQL, що дозволяє розробникам легко інтегрувати її в свої додатки.

Легкість використання та налаштування: однією з основних переваг SQLite3 є її простота використання та налаштування також зручність. Вона не потребує окремого сервера баз даних і може працювати безпосередньо з файлами даних, що робить її ідеальним вибором для невеликих проектів та додатків з обмеженими ресурсами.

Підтримка різних типів даних: SQLite3 підтримує різні типи даних, включаючи цілі числа, дійсні числа, рядки, дати та бінарні дані. Це дозволяє зберігати різноманітні дані в базі даних і виконувати різноманітні завдання, від зберігання текстової інформації до обробки складних структур даних.

Кросплатформеність та рентабельність: SQLite3 підтримується на багатьох платформах, включаючи Windows, macOS та різні дистрибутиви Linux, що робить її досить універсальною для різних типів додатків та середовищ розробки для користувачів.

Інструменти для роботи з SQLite3: щоб спростити роботу з SQLite3, існують різноманітні інструменти, такі як SQLite Studio, DB Browser for SQLite та інші, які надають зручний інтерфейс для роботи з базою даних SQLite3.

2.10 PyCharm: ідеальне середовище для розробки на Python

PyCharm — це інтегроване середовище розробки (IDE), призначене спеціально для роботи з мовою програмування Python. Розроблений компанією JetBrains, PyCharm надає розробникам широкий набір інструментів для продуктивної роботи над проектами будь-якої складності.

Переваги використання PyCharm

1. Повний функціонал для Python: PyCharm має всі необхідні інструменти для роботи з Python, включаючи підтримку структури проекту, відлагодження коду, автоматичне завершення коду, аналіз коду та багато іншого.

2. Широкий набір інструментів: у PyCharm є вбудовані інструменти для керування версіями за допомогою систем контролю версій, таких як Git, інтеграція з віртуальними середовищами, робота з базами даних, тестування коду та багато іншого.

3. Підтримка різних типів проектів: від веб-розробки до наукових обчислень, PyCharm підтримує різноманітні типи проектів, що робить його універсальним інструментом для будь-яких цілей.

4. Керування залежностями та пакетами: PyCharm надає зручний інтерфейс для керування залежностями та встановлення пакетів Python через pip, що спрощує роботу з зовнішніми бібліотеками та фреймворками.

5. Підтримка рефакторингу коду: за допомогою PyCharm можна легко виконувати рефакторинг коду, включаючи перейменування змінних, екстракцію функцій, оптимізацію імпортів та інші операції, що полегшують розробку та підтримку коду.

6. Розширення та налаштування: PyCharm підтримує розширення через плагіни, що дозволяє розробникам налаштувати середовище під свої потреби та використовувати додаткові інструменти.

PyCharm — це потужне та універсальне інтегроване середовище розробки, яке ідеально підходить для роботи з мовою програмування Python. З його допомогою розробники можуть продуктивно працювати над будь-якими проектами, від невеликих скриптів до великих програмних систем. Назва підрозділу: PyCharm: ваш надійний партнер у світі Python.

Висновки до розділу 2

Вибір технологій для кваліфікаційної роботи є ключовим аспектом, який впливає на її успішне виконання. Починаючи з мови програмування Python, яка обрана за її простоту та потужність, створення веб-додатків на цій мові стає більш доступним та зручним. Python здатний забезпечити потрібний функціонал для розробки як простих, так і досить складних веб-платформ.

Для швидкої та стабільної розробки веб-додатків обрано фреймворк Django. Django забезпечує готові рішення для багатьох аспектів веб-розробки, включити аутентифікацію, маршрутизацію та роботу з базами даних, що спрощує процес розробки та прискорює його. Його потужна система маршрутизації та вбудована ORM дозволяють ефективно взаємодіяти з базами даних, а також створювати безпечні та масштабовані веб-додатки.

Бібліотека Bootstrap обрана для створення інтерфейсу користувача через її широкий набір готових компонентів та стилів, що дозволяє швидко створювати адаптивний та привабливий дизайн. Bootstrap надає стандартні засоби для створення естетичного та легко використовуваного інтерфейсу, що сприяє покращенню користувацького досвіду та зручності взаємодії з веб-додатком.

Використання технології SQLite3 для управління базами даних є логічним вибором, оскільки вона є легкою та потужною вбудованою базою даних, що ідеально підходить для розробки веб-додатків. SQLite3 забезпечує можливість

простого та ефективного зберігання даних без необхідності встановлення та конфігурування складних серверів баз даних.

Середовище розробки PyCharm обрано для його зручності та продуктивності. PyCharm надає розробникам широкий набір інструментів для роботи з Python, включаючи підтримку версій, від лагодження коду та керування залежностями проекту. Це середовище дозволяє зосередитися на розробці програмного забезпечення та логіки, не витрачаючи час на налаштування середовища розробки.

Для організації веб-сторінок та їхнього стильного оформлення використовуються мови та технології HTML, CSS і JavaScript. HTML визначає структуру сторінки, CSS відповідає за її стиль та вигляд, а JavaScript забезпечує інтерактивність та динамічність веб-додатка. Використання цих технологій дозволяє створювати сучасні та інтерактивні веб-додатки, які задовольняють потреби користувачів.

Зважаючи на обрані технології та їхні можливості, можна визначити, що комбінація Python, Django, Bootstrap, SQLite3 та середовища розробки PyCharm є оптимальним вибором для розробки веб-додатків. Вони забезпечують не лише швидку і ефективну розробку, але й високу якість та зручність використання для як розробників, так і кінцевих користувачів. Використання мов та технологій HTML, CSS і JavaScript доповнює функціонал, дозволяючи створювати сучасні та інтерактивні веб-додатки, що задовольняють сучасні вимоги до веб-розробки.

РОЗДІЛ 3. ПРАКТИЧНА РЕАЛІЗАЦІЯ ВЕБ-ПЛАТФОРМИ ПО ВИВЧЕННЯ АНГЛІСЬКОЇ МОВИ ЗА ДОПОМОГУЮ ФЛЕШ-КАРТКО ТА ТЕСТІВ

Для реалізації платформи WordSnap, я використав ряд мов програмування та фреймворк Django. Python став основною мовою програмування для розробки серверної логіки, а Django забезпечив потужний інструментарій для швидкої та зручної розробки веб-додатків також були різні бібліотеки, також використав HTML, CSS та JavaScript для розробки візуально привабливого та інтерактивного веб-інтерфейсу. HTML використовується для створення структури веб-сторінок, CSS — для їхнього оформлення та стилізації, а JavaScript — для додавання динамічних ефектів та інтерактивності [8].

Розглянемо структуру та організацію коду, які допомогли мені побудувати функціональний, логічний та ефективний веб-додаток. Архітектура Django включає три основні компоненти: моделі (models), відображення (views) та шаблони (templates).

Моделі (Models): в рамках проекту "WordSnap" були створені моделі, які визначають структуру даних та взаємозв'язки між ними. Кожна модель є класом Python, який відображає таблицю в базі даних. Вони дозволяють ефективно зберігати та управляти інформацією про користувачів, питання, відповіді тощо.

Відображення (Views): функції відображень в Django відповідають за обробку HTTP-запитів та відображення відповідних сторінок або даних. Вони взаємодіють з моделями та передають дані до відповідних шаблонів для відображення. В моєму проекті, відображення виконують ключову роль у відображенні інформації користувачам та управлінні логікою додатку.

Шаблони (Templates): HTML-шаблони використовуються для відображення веб-сторінок та динамічного контенту. Вони містять HTML-код разом з Django-специфічними тегами та змінними для відображення динамічного

контенту. В нашому проекті, шаблони грають важливу роль у створенні зручного та естетичного інтерфейсу для користувачів.

3.2. Структура проекту та файлів

У моєму проекті існує головна папка з назвою "flashcards", яка виконує важливу роль у структурі всього додатку. У цій основній папці розміщені три додаткові папки, які відіграють ключову роль у правильному функціонуванні проекту. Перша з них — "management", призначена для зберігання файлів, які відповідають за управління адміністративними функціями системи. Друга папка — "migrations", є важливою складовою структури проекту, оскільки вона містить файли міграцій, які дозволяють автоматизовано керувати змінами в структурі бази даних. Нарешті, третя папка — "templates", призначена для зберігання HTML-шаблонів, які використовуються для відображення сторінок веб-додатку. Крім цих важливих папок, у головній папці також містяться основні файли з кодом, які відповідають за функціонал додатку.

У файлі "settings.py" містяться налаштування для веб-додатка на основі Django. Цей файл визначає параметри, такі як шляхи до статичних та медіафайлів, налаштування бази даних, мову та часовий пояс, а також налаштування аутентифікації та авторизації користувачів. Файл "init.py" в Django цей файл відповідає ініціалізацій для пакету flashcards.

У файлі "views.py" містяться функції, які відповідають за різноманітні аспекти роботи веб-додатка на основі Django. Кожна функція відповідає за певний функціонал та виконує конкретні завдання. Наприклад, функція "index(request)" відповідає за відображення головної сторінки додатку. Функція "flashcards(request)" відповідає за відображення сторінки з флеш-картками, а також забезпечує можливість додавання нових флеш-карток. Функція "login_view(request)" відповідає за обробку входу користувача до системи, перевіряючи введені дані та автентифікуючи користувача. Ці функції виконуються при отриманні відповідного запиту від користувача і забезпечують

взаємодію з додатком відповідно до встановленої логіки роботи. Крім того, в файлі "views.py" також містяться функції, які відповідають за обробку даних, валідацію, аутентифікацію користувачів та інші аспекти роботи веб-додатка.

У файлі "urls.py" визначені маршрути (URL-адреси), які відповідають за направлення запитів користувачів до відповідних функцій обробки веб-додатка на основі Django. Кожен маршрут визначається за допомогою функції path(), яка приймає шлях (URL-шаблон) та функцію обробки.

У файлі "models.py" визначені моделі, які використовуються для організації бази даних у веб-додатку на основі Django. Моделі визначають структуру даних, які будуть зберігатися в базі даних.

У файлі "forms.py" визначені форми, які використовуються для відображення веб-форм на сторінках додатка на основі Django. Форми використовуються для отримання даних від користувачів та їх подальшої обробки або збереження у базі даних.

У файлі "add_tests.py", який розташований у папці "management/commands" мого додатка, визначено керуючу команду Django, яка дозволяє додавати тести до системи. Керуючі команди є спеціальними скриптами, які можна викликати з командного рядка Django для виконання певних завдань або процесів.

У файлі "initial_data.py", розташованому у папці "management/commands" мого додатка, визначено керуючу команду Django, яка додає початкові дані до моєї системи. Було проведено дослідження та розроблено код для автоматизації створення початкових даних у вашій базі даних Django.

У файлі HTML-файл "index.html" є шаблоном для головної сторінки вашого веб-додатка. Він відображає заголовок "WordSnap" та дві кнопки для доступу до флеш-карток та тестів. Також у верхньому правому куті відображаються посилання для входу або виходу з системи в залежності від того, чи ввійшов користувач у систему.

У HTML-файлі "flashcards.html" є шаблоном для сторінки з флеш-картками вашого веб-додатка. Він відображає кнопки для вибору різних пакунків

флеш-карток, відображає саму флеш-картку з англійським та українським словом, а також зображенням, яке відповідає слову. Крім того, на цій сторінці є можливість видалення флеш-картки та збереження її до обраних фаворитів.

У HTML-файлі "dictionary.html" є шаблоном для сторінки словника мого веб-додатка. Він відображає таблицю з флеш-картками користувача, що містять українські та англійські слова, а також пак, до якого вони належать.

У HTML-файлі "create_flashcard.html" є шаблоном для сторінки створення нових флеш-карток у моєму веб-додатку. Він містить форму, де користувач може ввести українські та англійські слова для нової флеш-картки, а також вибрати пак, до якого вона буде належати.

У HTML-файлі "tests.html" представляє собою сторінку тестування рівня англійської мови у моєму веб-додатку. Він містить список питань з можливістю вибору однієї правильної відповіді або введення відповідного слова.

У файлі HTML-файл "vocabulary_test.html" представляє собою сторінку тестування словникового запасу у моєму веб-додатку. Користувач може вибрати слова з різних рівнів складності, і після натискання кнопки "Submit" відображається оцінка приблизного словникового запасу користувача.

Файл db.sqlite3 в моєму проекті є базою даних SQLite3, яка використовується для зберігання всієї структури даних та фактичних даних, що використовуються моїм веб-додатком.

3.3 Розробка функціоналу веб-платформи(models.py)

Основана частина розробки веб-платформи це в основних файлах як ‘view.py’, ‘models.py’ та ‘ursl.pyl’ також значна робота в папці ‘templates’ це весь вид сторінок.

```
from django.contrib.auth.models import User
from django.db import models
class Flashcard(models.Model):
```

```
        category = models.CharField(max_length=100, default='')
        pack = models.ForeignKey('Pack', on_delete=models.CASCADE,
related_name='flashcards')
        ukrainian_word = models.CharField(max_length=100)
        english_word = models.CharField(max_length=100)
        def __str__(self):
            return f"{self.ukrainian_word} - {self.english_word}"
```

Цей код визначає модель Flashcard для зберігання флеш-карток. Кожна флеш-картка має поля category (категорія), ukrainian_word (українське слово) і english_word (англійське слово), які є символьними рядками з максимальною довжиною 100 символів. Також вона належить до певного пакету, що вказується за допомогою зовнішнього ключа ForeignKey. Коли пакет видаляється, усі пов'язані з ним флеш-картки також видаляються завдяки параметру on_delete=models.CASCADE. Метод __str__ визначає, як об'єкт моделі буде виводитися у текстовому вигляді.

```
class Pack(models.Model):
    name = models.CharField(max_length=100)
class Word(models.Model):
    pack_id = models.ForeignKey(Pack, on_delete=models.CASCADE)
    ukrainian = models.CharField(max_length=100)
    english = models.CharField(max_length=100)
    def __str__(self):
        return f"{self.ukrainian} - {self.english}"
```

Цей фрагмент коду містить опис трьох моделей даних для Django-проекту:

модель Pack має одне поле name, яке є символьним рядком з максимальною довжиною 100 символів. Ця модель призначена для зберігання інформації про пакети слів.

Модель Word має поля pack_id, ukrainian і english, які також є символьними рядками з максимальною довжиною 100 символів. Поле pack_id є зовнішнім

ключем, який посилається на модель Pack і вказує на зв'язок між словом і пакетом. Метод __str__ визначає текстове представлення об'єкту моделі.

```
class FavoriteCard(models.Model):
    english = models.CharField(max_length=100)
    ukrainian = models.CharField(max_length=100)
    def __str__(self):
        return f"{self.english} - {self.ukrainian}"
```

Модель FavoriteCard має поля english і ukrainian, які також є символьними рядками з максимальною довжиною 100 символів. Ця модель використовується для зберігання улюблених слів або фраз.

3.3.2 Види веб-платформи(view.py)

У views.py описуються веб-сторінки, які користувач може переглядати, форми, які він може заповнювати, і дії, які будуть відбуваються під час взаємодії з цими елементами.

```
from django.contrib.auth import logout
from django.contrib.auth import authenticate, login
from django.shortcuts import render, redirect
from .forms import FlashcardForm, MyCustomLoginForm,
from .models import Flashcard, Pack
from django.shortcuts import render
from .forms import FlashcardForm, PackChoiceForm
from django.http import JsonResponse
from .models import FavoriteCard
```

У цьому коді відображенні різні імпорти які потрібні для цього файл.

```
def flashcards(request):
    beginner_pack = Pack.objects.get(name="Початковий")
```

```
        flashcards = Flashcard.objects.filter(pack=beginner_pack)
        if request.method == 'POST':
            form = FlashcardForm(request.POST)
            if form.is_valid(): form.save()
                return redirect('flashcards') else:
            form = FlashcardForm()
        return render(request,'flashcards.html',{'form': form,
'flashcards': flashcards})
```

У цьому фрагменті коду реалізовано дві функції виду «view» для обробки запитів HTTP.

Функція index(request) просто повертає відповідь на запит, яка включає в себе шаблон сторінки "index.html".

Функція flashcards(request) виконує обробку запитів для сторінки флеш-карток. Вона отримує пакет з бази даних за назвою "Початковий" і фільтрує флеш-картки за цим пакетом. Якщо отриманий запит є POST, вона перевіряє дані форми FlashcardForm. Якщо дані коректні, вони зберігаються у базі даних, і користувач перенаправляється на ту саму сторінку флеш-карток. Якщо запит є GET, то створюється новий екземпляр форми FlashcardForm, і сторінка відображається з цією формою та списком вже існуючих флеш-карток.

```
def create_flashcard(request):
    if request.method == 'POST':
        flashcard_form = FlashcardForm(request.POST)
        pack_form = PackChoiceForm(request.POST)
        if flashcard_form.is_valid() and pack_form.is_valid():
            flashcard = flashcard_form.save(commit=False)
            flashcard.user = request.user
            flashcard.save()
            pack = pack_form.cleaned_data['pack']
            pack.flashcards.add(flashcard)
            return redirect('flashcards')
```

У цьому фрагменті коду реалізована функція create_flashcard, яка відповідає за створення нової флеш-картки.

Якщо отриманий запит є POST, функція спробує створити нову флеш-картку на основі даних, отриманих з форми FlashcardForm, а також вибраний пакет, який отримується з форми PackChoiceForm. Флеш-картка зберігається в базі даних, і користувач перенаправляється на сторінку флеш-карток.

```
        else:
            flashcard_form = FlashcardForm()
            pack_form = PackChoiceForm()
        return render(request, 'create_flashcard.html',
{'flashcard_form': flashcard_form, 'pack_form': pack_form})
```

Якщо отриманий запит є GET, то створюються нові екземпляри форм FlashcardForm і PackChoiceForm, і сторінка відображається з цими формами для створення нової флеш-картки.

```
    def tests(request):
        return render(request, 'tests.html')
```

У цьому фрагменті коду реалізовані дві функції. Перша функція tests відповідає за відображення сторінки тестів. Коли користувач переходить на цю сторінку, вона просто відображає вміст шаблону "tests.html".

```
    def login_view(request):
        if request.method == 'POST':
            form = MyCustomLoginForm(request, request.POST)
            if form.is_valid():
                username = form.cleaned_data['username']
                password = form.cleaned_data['password']
                user = authenticate(request, username=username,
password=password)
                if user is not None:
```

```
                login(request, user)
                return redirect('index')
```

Друга функція login_view відповідає за відображення сторінки входу в систему і обробку введених користувачем даних. Якщо метод запиту — POST, то функція спробує автентифікувати користувача з введеними даними. Якщо автентифікація успішна, користувача ввійшов у систему і перенаправляється на головну сторінку "index".

```
    else:
        form = MyCustomLoginForm(request)
    return render(request, 'login.html', {'form': form})
```

Якщо метод запиту — GET, то відображається форма входу, де користувач може ввести свої дані.

```
def logout_view(request):
    logout(request)
    return redirect('index')
```

Функція logout_view приймає запит, виконує вихід користувача із системи за допомогою функції logout, і потім перенаправляє користувача на головну сторінку за допомогою функції redirect, яка перенаправляє на URL, пов'язаний з ім'ям index.

```
def signup_view(request):
    if request.method == 'POST':
        form = MyCustomUserCreationForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('login')
    else:
        form = MyCustomUserCreationForm()
    return render(request, 'signup.html', {'form': form})
```

Функція signup_view обробляє запити на реєстрацію нового користувача. Якщо запит є методом POST, створюється форма MyCustomUserCreationForm з даними, наданими у запиті. Зберігає нового користувача і перенаправляє на сторінку входу за допомогою функції redirect, яка перенаправляє на URL, пов'язаний з ім'ям login. Якщо запит не є методом POST, створюється порожня форма MyCustomUserCreationForm. В обох випадках функція повертає відображення шаблону signup.html з переданою у контексті формою.

```
    def mini_dictionary(request):
        flashcards = Flashcard.objects.all()
        return render(request, 'mini_dictionary.html', {'flashcards':
flashcards})
    def my_flashcards(request):
        user_flashcards = Flashcard.objects.filter(user=request.user)
        return render(request, 'my_flashcards.html',
{'user_flashcards': user_flashcards})
```

Функція mini_dictionary отримує всі об'єкти Flashcard з бази даних за допомогою Flashcard.objects.all() і повертає відображення шаблону mini_dictionary.html, передаючи в нього всі флеш-картки як контекст.

Функція my_flashcards отримує всі об'єкти Flashcard, які належать поточному користувачеві, за допомогою Flashcard.objects.filter(user=request.user) і повертає відображення шаблону my_flashcards.html, передаючи в нього флеш-картки користувача як контекст.

```
    def beginner_pack_words(request):
        beginner_pack = Pack.objects.get(name="Початковий")
        flashcards = Flashcard.objects.filter(category="Початковий")
        words_list = [{'ukrainian': flashcard.ukrainian, 'english':
flashcard.english} for flashcard in flashcards]
        return JsonResponse(words_list, safe=False)
```

Функція beginner_pack_words отримує об'єкт Pack з назвою "Початковий" за допомогою Pack.objects.get(name="Початковий"), потім отримує всі об'єкти Flashcard з категорією "Початковий" за допомогою Flashcard.objects.filter(category="Початковий"), створює список слів, що містять українські та англійські слова для кожної флеш-картки, і повертає цей список як JSON відповідь за допомогою JsonResponse.

```
def save_favorite(request):
if request.method == 'POST':   try:
data = json.loads(request.body)
english = data.get('english')
ukrainian = data.get('ukrainian')
if english and ukrainian:
if not FavoriteCard.objects.filter(english=english,
ukrainian=ukrainian).exists():
favorite_card = FavoriteCard(english=english, ukrainian=ukrainian)
```

Функція save_favorite обробляє POST запити, намагаючись зберегти нову улюблену картку. Вона отримує дані з тіла запиту, які містять англійське та українське слова. Якщо обидва слова надано, перевіряє, чи така картка вже існує в базі даних. Якщо не існує, створює новий об'єкт FavoriteCard з цими словами та зберігає його.

```
                favorite_card.save()
                  return JsonResponse({'status': 'success'}, status=201)
                      return JsonResponse({'status': 'error', 'message':
'Flashcard already exists'}, status=400)
                  return JsonResponse({'status': 'error', 'message':
'Invalid data'}, status=400)
             except json.JSONDecodeError:
                  return JsonResponse({'status': 'error', 'message':
'Invalid JSON'}, status=400)
         return JsonResponse({'status': 'error', 'message': 'Invalid
request method'}, status=405)
```

Після успішного збереження повертає JSON відповідь зі статусом успіху. Якщо картка вже існує або надано некоректні дані, то повертає JSON відповідь зі статусом помилки та відповідним повідомленням. У випадку помилки декодування JSON повертає JSON відповідь зі статусом помилки та повідомленням про не валідний JSON. Якщо метод запиту не POST, повертає JSON відповідь зі статусом помилки та повідомленням про неправильний метод згідно запиту.

```
    def get_favorites(request):
        if request.method == 'GET':
            favorites = FavoriteCard.objects.all().values('english',
'ukrainian')
            return JsonResponse(list(favorites), safe=False)
        return JsonResponse({'status': 'error', 'message': 'Invalid
request method'}, status=405)
```

Функція get_favorites обробляє GET запити, отримуючи всі улюблені картки з бази даних та повертаючи їх у форматі JSON. Якщо метод запиту не GET, повертає JSON відповідь зі статусом помилки та повідомленням про неправильний метод запиту [9].

```
    def english_level_test(request):
        if request.method == 'POST':
            q1 = request.POST.get('q1')
            q2 = request.POST.get('q2')
            return render(request, 'english_level_test.html',
{'test_results': 'Intermediate'})
        return render(request, 'english_level_test.html')
```

Функція english_level_test обробляє POST запити, отримуючи відповіді на запитання з форми, що містяться у запиті, та визначає рівень англійської мови користувача на основі цих відповідей. Результати тесту повертаються на

сторінку. Якщо метод запиту не POST, функція повертає порожню сторінку тесту на визначення рівня англійської мови.

```
def vocabulary_test(request):
    return render(request, 'vocabulary_test.html')
def packs(request):
    return render(request, 'packs.html')
```

Функція vocabulary_test відповідає за відображення сторінки тестування словникового запасу. Вона просто повертає шаблон сторінки vocabulary_test.html за допомогою функції render. Функція packs відповідає за відображення списку паків. Вона також просто повертає шаблон сторінки packs.html за допомогою функції render.

```
def add_pack(request):
    if request.method == 'POST':
        pack_name = request.POST.get('pack-name', '')
        pack_description = request.POST.get('pack-description', '')
        return JsonResponse({'success': True})
    else:
        return JsonResponse({'success': False})
def dictionary(request):
    return render(request, 'dictionary.html')
```

Функція add_pack обробляє запити для додавання нового паку. Якщо запит має метод 'POST', вона отримує дані форми з запиту, зокрема ім'я та опис паку, і зберігає їх у базі даних або виконує інші необхідні дії. Після цього функція повертає JSON-відповідь з підтвердженням успіху. Якщо метод запиту не 'POST', вона повертає JSON-відповідь з повідомленням про помилку. Функція dictionary відповідає за відображення сторінки словника і повертає шаблон dictionary.html за допомогою функції render.

3.3.4 Маршрути в веб-платформі(urls.py)

У файлі urls.py веб-платформи Django визначаються маршрути, які вказують, які види запитів будуть пов'язані з якими веб-сторінками. Кожен маршрут складається з шляху URL та функції виду (view function), яка буде викликатися для обробки запиту на цей URL.

```
from django.urls import path, include
from .views import beginner_pack_words
from .views import index, flashcards, create_flashcard, tests,
login_view, logout_view, signup_view, mini_dictionary, packs
from . import views
from django.contrib import admin
from django.conf import settings
from django.conf.urls.static import static
```

Цей код визначає URL-шляхи для маршрутизації запитів у веб-додатку Django. Його структура розпочинається з імпорту необхідних функцій та модулів, таких як path та include з django.urls, функції static для роботи зі статичними файлами, а також модуля адміністрування Django (admin) і модуля налаштувань Django (settings).

```
urlpatterns = [  path('admin/', admin.site.urls),   path('', index,
name='index'),  path('accounts/', include('allauth.urls')),
path('flashcards/', flashcards, name='flashcards'),
path('create_flashcard/', create_flashcard, name='create_flashcard'),
path('tests/', tests, name='tests'), path('login/', login_view,
name='login'), path('logout/', logout_view, name='logout'),
path('signup/', signup_view, name='signup'),
path('mini_dictionary/', mini_dictionary, name='mini_dictionary'),
```

Потім визначається список urlpatterns, що містить набір URL-шляхів і відповідних функцій обробників. Кожен URL-шлях визначає шлях та функцію

обробника для обробки запитів за цим шляхом. Наприклад, path('admin/', admin.site.urls) веде до сторінки адміністрування Django, а path('', index, name='index') вказує на головну сторінку додатку, яка викликає функцію index. Деякі URL-шляхи вказують на сторінки для реєстрації, входу та виходу користувачів, створення та відображення флеш-карток, тестів тощо.

```
path('api/beginner_pack_words/', beginner_pack_words,
name='beginner_pack_words'), path('save_favorite/', views.save_favorite,
name='save_favorite'), path('get_favorites/', views.get_favorites,
name='get_favorites'),path('english-level-test/',
views.english_level_test, name='english_level_test'),
path('vocabulary_test/', views.vocabulary_test, name='vocabulary_test'),
path('packs/', packs, name='packs'), path('add_pack/', views.add_pack,
name='add_pack'),path('dictionary/', views.dictionary,
name='dictionary'),] + static(settings.STATIC_URL,
document_root=settings.STATIC_ROOT)
```

Також деякі URL-шляхи призначені для взаємодії з API, як от api/beginner_pack_words/ для початкового набору слів. У кінці додається обробка статичних файлів у режимі розробки.

3.3.5 Визначення форм і їх функціональність(forms.py)

Управління користувачами та обробка даних — це важливі аспекти будь-якого веб-додатку. У веб-проектах Django форми грають ключову роль у зборі та обробці інформації, що надходить від користувачів. Вони визначають структуру взаємодії з веб-сторінками, дозволяючи користувачам реєструватися, входити в систему, надсилати дані та багато іншого.

```
class MyCustomLoginForm(AuthenticationForm):
    pass
class FlashcardForm(forms.ModelForm):
    class Meta:
```

```
        model = Flashcard
        fields = ['ukrainian_word', 'english_word', 'pack']
    def __init__(self, *args, **kwargs):
        super(FlashcardForm, self).__init__(*args, **kwargs)
        self.fields['pack'].queryset = Pack.objects.all()
```

Цей код визначає дві форми для веб-додатку Django.

MyCustomLoginForm: ця форма використовується для аутентифікації користувача. Вона базується на стандартній формі AuthenticationForm з модуля django.contrib.auth.forms. У цьому класі можна визначити будь-які специфічні для додатку налаштування форми аутентифікації.

FlashcardForm: ця форма призначена для створення та редагування флеш-карток. Вона базується на класі ModelForm з модуля django.forms, що дозволяє автоматично створювати форму на основі моделі Django. У класі Meta вказано модель Flashcard, з якою пов'язана форма, і перераховані поля цієї моделі, які мають бути відображені на формі. Крім того, у методі __init__ виконується налаштування поля pack, щоб забезпечити вибір паку з усіх наявних паків.

Ці дві форми дозволяють користувачам аутентифікуватися та створювати або редагувати флеш-картки у веб-платформі.

```
    class MyCustomUserCreationForm(UserCreationForm):
        class Meta(UserCreationForm.Meta):
            model = User
            fields = ('username', 'email')
    class PackChoiceForm(forms.Form):
        pack = forms.ModelChoiceField(queryset=Pack.objects.all(),
empty_label=None)
```

Клас MyCustomUserCreationForm є розширенням стандартної форми UserCreationForm з модуля django.contrib.auth.forms. Використовуючи клас Meta, він зазначає модель, яка буде використовуватися для створення нового користувача (у цьому випадку User з модуля django.contrib.auth.models). Також

вказуються поля, які будуть відображені на формі для реєстрації користувача, в даному випадку це поля username і email. Ви можете додати будь-які інші поля, які вважаєте потрібними для реєстрації.

Клас PackChoiceForm є простою формою, яка має одне поле pack, яке дозволяє користувачеві вибрати пак з вже існуючих. Це поле створюється за допомогою ModelChoiceField, яке базується на моделі Pack, і воно автоматично заповнюється всіма об'єктами Pack, які є в базі даних. Параметр empty_label=None вказує на те, що у цього поля не буде порожнього варіанту.

3.4 Templates вид сторінок

У цьому підрозділі я дослідив структуру та функціонал HTML-шаблонів, які використовуються для відображення різних сторінок у веб-платформи.

3.4.2 Головна сторінка

HTML-код цього шаблону відповідає за відображення головної сторінки веб-додатку "WordSnap". Він містить основну структуру HTML-документу з визначенням мови, заголовком сторінки, а також вбудованим CSS (рис 3.1).

Рисунок 3.1. — Головна сторінка

У цьому шаблоні використовуються класи CSS для стилізації елементів, таких як кнопки та контейнери. Наприклад, клас ".button-container" відповідає за відображення кнопок у вигляді вертикального списку з відступами зліва,

```
<style>
    .button-container {
        display: flex;
        flex-direction: column;
        align-items: flex-start;
        padding-left: 20px;
    }
    .button-container a {
        margin-bottom: 10px;
    }
```

а клас ".user-actions" визначає стилі для блоку з посиланнями "Login" та "Signup", які з'являються в правому верхньому куті сторінки, якщо користувач не увійшов у систему.

```
    .user-actions {
        position: absolute;
        top: 10px;
        right: 20px;
    }
    .user-actions a {
        margin-left: 10px;
    }
```

У розділі <body> містяться основні елементи сторінки. Заголовок <h1> виводить назву додатку "WordSnap".

```
<head>
    <meta charset="UTF-8">
```

```
        <meta name="viewport" content="width=device-width, initial-
scale=1.0">
        <title>WordSnap</title>
```

Блок містить посилання на вхід у систему або реєстрацію, в залежності від того, чи увійшов користувач у систему чи ні.

```
        <h1 style="font-family: 'Roboto', sans-serif;">WordSnap</h1>
        <div class="user-actions">
            {% if user.is_authenticated %}
                <a href="{% url 'logout' %}">Logout</a>
            {% else %}
                <a href="{% url 'account_login' %}">Login</a>
                <a href="{% url 'account_signup' %}">Signup</a>
            {% endif %}
```

Блок містить посилання на інші сторінки додатку, такі як "Flashcards" та "Tests". Кожна кнопка містить посилання на відповідну URL-адресу.

```
        </div>
        <div class="button-container">
            <a href="{% url 'flashcards'
%}"><button>Flashcards</button></a>
            <a href="{% url 'tests' %}"><button>Tests</button></a>
        </div>
    </body>
    </html>
```

Шаблон також використовує спеціальні теги шаблонізатора Django {% %}, які дозволяють вставляти динамічний контент. Наприклад, в тегу {% if user.is_authenticated %} перевіряється, чи ввійшов користувач у систему. Якщо так, відображається посилання на вихід з системи (Logout), в іншому випадку — посилання на вхід у систему (Login) та реєстрацію (Signup).

3.4.3 Сторінка Flashcards

Цей HTML-код відповідає за відображення сторінки з флеш-картками у веб-додатку "Flashcards" (Рис 3.2).

```
.button-container {
    text-align: left;
    margin-bottom: 20px; }
.btn.btn-primary {
    margin-right: 10px;
    padding: 10px 20px;
    border-radius: 5px;
    background-color: #800080;
    color: white;
}
```

Тут використовуються класи CSS для стилізації елементів, таких як кнопки та контейнери. Наприклад, класи ".button-container", ".btn.btn-primary" задають вигляд і розміщення кнопок, а класи ".button-container a" визначають стилі для звичайних посилань.

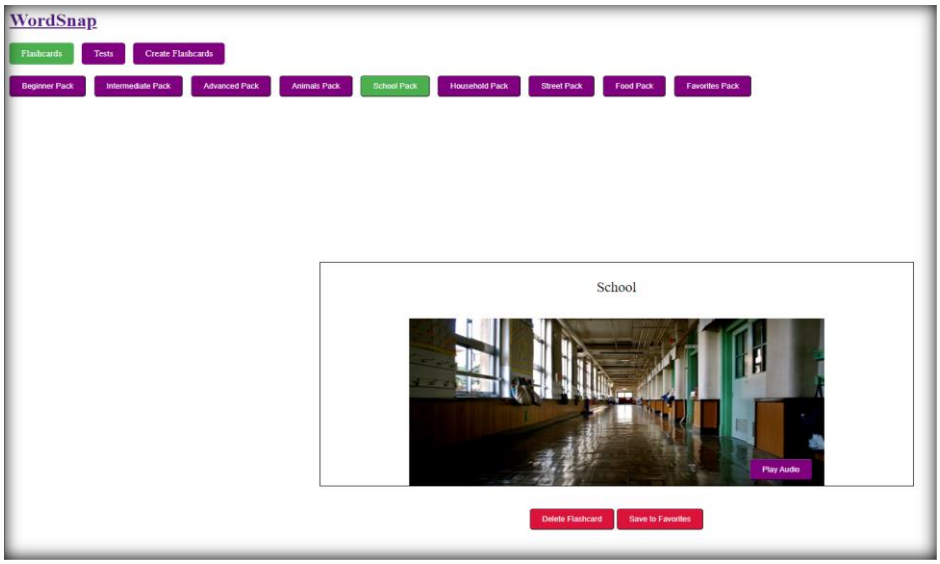


Рисунок 3.2 — Сторінка “Flashcards”

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Flashcards</title>
```

Код також включає в себе метатеги для встановлення кодування та масштабування сторінки, а також визначає заголовок сторінки. Тег <meta name="viewport" content="width=device-width, initial-scale=1.0"> встановлює ширину екрану відповідно до ширини пристрою, що дозволяє належним чином масштабувати сторінку на мобільних пристроях.

```
#flashcardContainer {
    width: 100%;
    height: 100vh;
    display: flex;
    justify-content: center;
    align-items: center;
    flex-direction: column;
}
```

Крім того, у коді є елементи CSS, які визначають розміщення та стиль елементів на сторінці. Наприклад, стиль #flashcardContainer встановлює висоту та ширину контейнера для флеш-карток та вирівнює їх по центру екрану.

```
#flashcard {
        width: 1000px;
        height: 400px;
        border: 2px solid black;
        display: flex;
        justify-content: center;
        align-items: center;
```

```
    flex-direction: column;
    cursor: pointer;
    margin-bottom: 20px;  }
```

Цей CSS-код визначає стилі для елемента з ідентифікатором flashcard. Він встановлює ширину та висоту цього елемента, обрамлення товщиною 2 пікселів та чорним кольором. Елемент відображається як гнучкий контейнер, який вирівнюється по центру як по горизонталі, так і по вертикалі. Текстовий вміст елемента розташовується по центру. Також для курсора встановлено властивість pointer, щоб вказати на те, що елемент є клікабельним. Нарешті, елемент має підсумкове значення відступу від нижньої границі 20 пікселів.

```
#wordContainer {
        flex: 1;
        display: flex;
        flex-direction: column;
        justify-content: center;
        align-items: center;
    }
```

Цей CSS-код визначає стилі для елемента з ідентифікатором wordContainer. Елемент розтягується, займаючи всю доступну область за допомогою властивості flex: 1. Внутрішній вміст елемента розташовується за допомогою гнучкої верстки (flexbox), при цьому напрямок основної осі встановлено вертикально (flex-direction: column). Вміст вирівнюється по центру як по горизонталі, так і по вертикалі завдяки властивостям justify-content: center та align-items: center. Це забезпечує центроване розташування дочірніх елементів всередині контейнера.

```
    #imageContainer {
        flex: 1;
```

```
    display: flex;
    justify-content: flex-start;
    align-items: center;
}
#wordImage {
    max-width: 100%;
    max-height: 100%;
}
```

Цей CSS-код стилізує два елементи: #imageContainer та #wordImage. Елемент #imageContainer налаштований на розтягнення, щоб зайняти всю доступну область, використовуючи властивість flex: 1. Він також використовує гнучку верстку (flexbox) для розміщення внутрішнього вмісту, причому дочірні елементи вирівнюються по лівому краю контейнера за допомогою властивості justify-content: flex-start і по центру вертикально за допомогою властивості align-items: center. Елемент #wordImage обмежений у розмірах так, щоб він не перевищував 100% ширини та висоти свого контейнера, що задається властивостями max-width: 100% та max-height: 100%. Це гарантує, що зображення всередині контейнера буде пропорційно масштабоване, не виходячи за межі контейнера.

```
#ukrainianWord, #englishWord {
    font-size: 24px;
}
.highlighted {
    background-color: #FFD700;
}
.extra-button-container {
    text-align: left;
}
```

Цей CSS-код встановлює стилі для елементів з ідентифікаторами ukrainianWord та englishWord, задаючи їм розмір шрифту 24 пікселі. Клас highlighted призначає фоновий колір #FFD700 (золотий) для виділених

елементів. Клас extra-button-container вирівнює текст по лівому краю всередині елементів, до яких він застосовується.

```
.delete-button, .save-button {
        margin-top: 20px;
        padding: 10px 20px;
        border-radius: 5px;
        background-color: #DC143C;
        color: white;
        cursor: pointer;
    }
```

Цей CSS-код задає стилі для елементів з класами delete-button і save-button. Він встановлює верхній відступ у 20 пікселів, внутрішні відступи по 10 пікселів зверху і знизу та по 20 пікселів з боків, що забезпечує додатковий простір всередині кнопок. Також кнопки мають закруглені кути з радіусом 5 пікселів, червоний фон із кодом кольору #DC143C і білий колір тексту. Курсор змінюється на вказівник при наведенні, що вказує на можливість натискання на кнопки.

```
@media (max-width: 600px) {
        .button-container a {
            display: block;
            margin-bottom: 10px;
        }
    }
```

Цей CSS-код визначає стилі для медіа-запиту, який застосовується, коли ширина вікна браузера не перевищує 600 пікселів. Усередині цього медіа-запиту вказано, що елементи a всередині контейнера з класом button-container будуть відображатися як блокові елементи. Це означає, що вони займуть всю ширину доступного простору.

```
</style>
```

```
    </head>
    <body>
        <h1><a href="{% url 'index' %}">WordSnap</a></h1>
        <div class="button-container">
            <a href="{% url 'flashcards' %}" class="{% if request.path
== '/flashcards/' %}active{% endif %}">Flashcards</a>
            <a href="{% url 'tests' %}" class="{% if request.path ==
'/tests/' %}active{% endif %}">Tests</a>
            <a href="{% url 'create_flashcard' %}" class="{% if
request.path == '/create_flashcard/' %}active{% endif %}">Create
Flashcards</a>
        </div>
```

Цей фрагмент HTML-коду використовується для створення частини веб-сторінки в Django, включаючи навігаційне меню. Він починається із закриття тегу <style> і відкриття тегів <head> і <body>, що вказує на структуру HTML-документа. Заголовок сторінки містить посилання на головну сторінку сайту з назвою "WordSnap", використовуючи тег <h1> і Django-шаблонний тег {% url 'index' %}, який динамічно створює URL для головної сторінки.

Під заголовком розташоване меню з трьома посиланнями: "Flashcards", "Tests" і "Create Flashcards". Ці посилання використовують шаблонні теги {% url '...' %} для динамічного створення відповідних URL-адрес. Крім того, за допомогою умовних виразів {% if request.path == '...' %}, активному посиланню додається клас active, що дозволяє стилізувати його і виділити як поточну сторінку. Таким чином, навігаційне меню забезпечує зручний перехід між різними розділами веб-платформи, відображаючи активну сторінку з особливим стилем та відображенням.

```
    <div class="button-row">
            <button class="btn btn-primary" id="beginnerPack">Beginner
Pack</button>
            <button class="btn btn-primary"
id="intermediatePack">Intermediate Pack</button>
```

```
            <button class="btn btn-primary" id="advancedPack">Advanced
Pack</button>
            <button class="btn btn-primary" id="animalsPack">Animals
Pack</button>
```

Цей HTML-код створює рядок кнопок із класом "button-row". У кожній кнопці є клас "btn btn-primary" для стилізації за допомогою CSS. Кожна кнопка має свій унікальний ідентифікатор ("beginnerPack", "intermediatePack", "advancedPack", "animalsPack"), який можна використовувати для ідентифікації кнопок у JavaScript або для встановлення взаємодії з ними у CSS.

```
    <button class="btn btn-primary" id="schoolPack">School
Pack</button>
            <button class="btn btn-primary"
id="householdPack">Household Pack</button>
            <button class="btn btn-primary" id="streetPack">Street
Pack</button>
            <button class="btn btn-primary" id="foodPack">Food
Pack</button>
            <button id="favoritesPack" class="btn btn-
primary">Favorites Pack</button>
```

У цьому HTML-коді є п'ять кнопок, кожна з яких має свій текстовий зміст та ідентифікатор. Класи btn та btn-primary використовуються для стилізації кнопок за допомогою CSS-стилів, що визначені у вашому проекті або бібліотеці стилів, наприклад, Bootstrap. Ці класи встановлюють певний вигляд для кнопок, щоб вони виглядали привабливо та відповідали загальному стилю вашого веб-сайту. Ідентифікатори, такі як schoolPack, householdPack, streetPack, foodPack та favoritesPack, призначені для ідентифікації кожної кнопки у JavaScript або jQuery коді. Це дозволить легко звертатися до кожної кнопки та виконувати різні дії в залежності від того, на яку кнопку натиснув користувач.

```
    <div id="flashcardContainer" style="display: flex;">
```

```
            <div id="flashcard">
                <div id="wordContainer">
                    <div id="englishWord">English Word</div>
                    <div id="ukrainianWord" style="display:
flex;">Слово по-українськи</div>
                </div>
```

Цей HTML-код представляє собою структуру веб-сторінки, яка містить два елементи.

Перший має ідентифікатор flashcardContainer і встановлено стиль display: flex;. Це вказує на те, що цей контейнер використовує гнучкість вирівнювання, що дає можливість розташовувати елементи в одному рядку.

У другому з ідентифікатором flashcard міститься контент, пов'язаний з флешкарткою. У ньому містяться ще два вкладені елементи: wordContainer і englishWord.

Елемент wordContainer призначений для розміщення слова англійською мовою (englishWord) та його відповідного перекладу українською мовою (ukrainianWord). Українське слово додатково вкладене у зі стилем display: flex;, що, ймовірно, використовується для гнучкого вирівнювання вмісту всередині нього.

```
                <div id="imageContainer" style="padding-top: 10px;
position: relative;">
                    <img id="wordImage" src="" style="max-width: 100%;
max-height: 100%;">
                    <button class="btn btn-primary"
onclick="playAudio()" style="position: absolute; bottom: 10px; right:
10px;">Play Audio</button>
                </div>
```

Цей HTML-код представляє собою контейнер для зображення та кнопки відтворення аудіо. Він має ідентифікатор imageContainer, який встановлює

відступи зверху контейнера. Стиль position: relative; визначає позиціонування відносно його вихідного місця в потоці документа.

У контейнері розміщується тег <img>, що має ідентифікатор wordImage. Він встановлюється за допомогою атрибуту src, а стилі max-width: 100%; та max-height: 100%; забезпечують масштабування зображення відносно розмірів контейнера так щоб зображення виглядало коректно.

Також у контейнері знаходиться кнопка з класом btn btn-primary, яка має текст "Play Audio". При натисканні на цю кнопку виконується JavaScript-функція playAudio(). Стиль position: absolute; розміщує кнопку абсолютно відносно внутрішнього контейнера, а властивості bottom: 10px; та right: 10px; встановлюють її позицію знизу та зправа відносно контейнера.

```
            </div>
            <div class="extra-button-container">
                <button class="delete-button" id="deleteButton">Delete
Flashcard</button>
                <button class="save-button"
id="saveToFavoritesButton">Save to Favorites</button>
            </div>
        <script
src="https://code.responsivevoice.org/responsivevoice.js?key=MZawgbhR"></
script>
```

Цей фрагмент коду є HTML-розміткою, яка складається з двох кнопок, розміщених всередині контейнера з класом extra-button-container. Перша кнопка має клас delete-button і має надпис "Delete Flashcard", а друга кнопка має клас save-button і надпис "Save to Favorites". Кожній кнопці також призначений унікальний ідентифікатор (id), щоб звертатися до них, якщо це необхідно.

Після цього HTML-коду імпортується зовнішній JavaScript-файл за допомогою тегу <script>. Цей файл містить скрипти для роботи з бібліотекою ResponsiveVoice, яка, використовується для функціоналу озвучування тексту.

```
                let currentPack = beginnerPack;
                let currentIndex = 0;
                let showingEnglish = true;
         document.getElementById("flashcard").addEventListener("click",
function () {  console.log("Clicked on flashcard.");
         if (!showingEnglish) {
             console.log("Showing Ukrainian word.");
        document.getElementById("englishWord").style.display = "block";
        document.getElementById("ukrainianWord").style.display = "none";
             currentIndex++;
```

Цей фрагмент JavaScript-коду налаштовує змінні, які використовуються для керування станом виведення флеш-карток на сторінці. Змінна currentPack встановлює початковий набір флеш-карток. Змінна currentIndex слідкує за поточною позицією флеш-картки в наборі. Змінна showingEnglish вказує, чи відображається англійське слово на флеш-картці.

Подія "click" на елементі з ідентифікатором "flashcard" відслідковується, і коли користувач клікає на флеш-картку, відбувається перевірка, чи відображається англійське слово. Якщо це так, то наступним кроком буде показ українського слова та перехід до наступної флеш-картки у наборі.

```
     if (currentIndex >= currentPack.length) {
                console.log("Reached end of pack.");
                currentIndex = 0;
            document.getElementById("flashcardContainer").style.display
= "none";
             } else {
                console.log("Showing next flashcard.");
                showFlashcard();
             }
```

Цей код є частиною JavaScript і виконується, коли користувач переглядає набір flashcards. Він перевіряє, чи досягнуто кінця набору flashcards. Якщо поточний індекс (currentIndex) більший або дорівнює довжині поточного набору

У протилежному випадку, якщо не досягнуто кінця набору, в консоль виводиться повідомлення "Showing next flashcard." (Показується наступна картка). Потім викликається функція showFlashcard(), яка відповідає за відображення наступної картки.

```
        } else {
            console.log("Showing English word.");
            document.getElementById("englishWord").style.display =
"none";
            document.getElementById("ukrainianWord").style.display =
"block";
         }
         showingEnglish = !showingEnglish;
     });
```

Цей код є частиною JavaScript і виконується, коли користувач переглядає набір flashcards. Він перевіряє, чи досягнуто кінця набору flashcards. Якщо поточний індекс (currentIndex) більший або дорівнює довжині поточного набору (currentPack.length), це означає, що вже показані всі картки у наборі. У такому випадку відбуваються наступні дії.

Цей фрагмент коду встановлює обробник подій для кліку на елемент "flashcard" і змінює відображення англійського та українського слова на флеш-картці. Початково відображається англійське слово. При кожному кліку на флеш-картку, вона перемикається між відображенням англійського та українського слова. Якщо на даний момент відображається англійське слово, воно приховується, а українське — показується. І навпаки, якщо на даний момент відображається українське слово, воно приховується, а англійське — показується. Також оновлюється поточний індекс слова у списку та, при досягненні кінця списку, приховується контейнер флеш-карток, якщо користувач прокрутив усі доступні флеш-картки.

```
            async function showFlashcard() {
```

```
const englishWord = currentPack[currentIndex].english;
const ukrainianWord = currentPack[currentIndex].ukrainian;
document.getElementById("englishWord").innerText = englishWord;
document.getElementById("ukrainianWord").innerText = ukrainianWord;
document.getElementById("englishWord").style.display = "block";
document.getElementById("ukrainianWord").style.display = "none";
                showingEnglish = true;
                await updateWordImage(englishWord);
                playAudio();
```

Ця функція відповідає за відображення флеш-картки. Вона отримує англійське та українське слово з поточного паку флеш-карток і встановлює їх як текст для відповідних елементів на сторінці. Після цього функція переключає відображення між англійським та українським словом, приховуючи одне та показуючи інше. Далі викликається функція updateWordImage, щоб оновити зображення слова, та playAudio, щоб програти аудіо для англійського слова. Функція використовує ключове слово async, щоб можна було використовувати оператор await для очікування завершення асинхронних операцій, таких як завантаження зображення.

```
function switchPack(pack) {
                currentPack = pack;
                currentIndex = 0;
                showFlashcard();
const packButtons = document.querySelectorAll(".button-row button");
packButtons.forEach(button => {
button.addEventListener("click", function(){
packButtons.forEach(btn => btn.classList.remove("active"));
this.classList.add("active");
const packId = this.id;
```

Цей код відповідає за переключення між різними паками флеш-карток при натисканні на відповідні кнопки. Функція switchPack приймає пак флеш-карток

як параметр і встановлює його як поточний пак, скидає індекс поточної картки на 0 і викликає функцію showFlashcard для відображення першої картки паку.

Потім код використовує метод querySelectorAll для вибору всіх кнопок, які є частинами класу .button-row.

```
switch (packId) {
                case 'beginnerPack':
                    switchPack(beginnerPack);
                    break;
                case 'intermediatePack':
                    switchPack(intermediatePack);
                    break;
                case 'advancedPack':
```

Для кожної кнопки додається слухач подій click, який викликається при натисканні на кнопку. При кожному натисканні спочатку знімається клас "active" з усіх кнопок, а потім додається до натиснутої кнопки.

```
                    switchPack(advancedPack);
                    break;
                case 'animalsPack':
                    switchPack(animalsPack);
                    break;
                case 'schoolPack':
```

В залежності від ідентифікатора кнопки викликається відповідна функція switchPack для зміни поточного паку або функція loadFavorites для завантаження паку "Favorites", якщо такий існує.

```
switchPack(foodPack);
                    break;
                case 'favoritesPack':
                    loadFavorites();
                    break;
```

```
                default:
        console.error("Unknown pack ID:", packId);
                    break;
            }
        });
```

Якщо ідентифікатор кнопки не відповідає жодному з наведених випадків, виводиться повідомлення про помилку у консоль.

```
function deleteFlashcard() {
        if (currentPack.length === 0) {
            alert("There are no flashcards to delete.");
            return;
        }
        currentPack.splice(currentIndex, 1);
        if (currentIndex >= currentPack.length) {
            currentIndex = 0;
        }
```

Ця функція відповідає за видалення флеш-картки з поточного паку. Спочатку вона перевіряє, чи існують флеш-картки у поточному паку. Якщо пак порожній, виводиться повідомлення про відсутність флеш-карток для видалення. Якщо флеш-картки є, видаляється флеш-картка за індексом currentIndex з поточного паку.

Після видалення флеш-картки перевіряється, чи індекс currentIndex не перевищує довжину поточного паку після видалення. Якщо він перевищує, currentIndex встановлюється на 0 для переходу до першої флеш-картки в оновленому паку.

```
        if (currentPack.length === 0) {
document.getElementById("flashcardContainer").style.display = "none";
        } else {
            showFlashcard();
        }
```

```
}
```

Якщо після видалення в паку залишаються флеш-картки, викликається функція showFlashcard() для відображення наступної флеш-картки після видалення. Якщо пак порожній після видалення, флеш-картка приховується.

```
document.getElementById("deleteButton").addEventListener("click", deleteFlashcard);
```

Ця частина коду встановлює обробник подій для кнопки з id "deleteButton". Коли ця кнопка клікнута, викликається функція deleteFlashcard(), яка видаляє поточну флеш-картку.

```
async function saveToFavorites() {
    let flashcard = {  english: document.getElementById("englishWord").innerText,  ukrainian: document.getElementById("ukrainianWord").innerText
    };
```

Функція saveToFavorites() відповідає за збереження флеш-картки в обрані. При кліку на відповідну кнопку викликається ця функція. Вона формує об'єкт flashcard з текстом англійського та українського слова, який береться з вмісту відповідних елементів DOM.

```
let response = await fetch('/save_favorite/', {
    method: 'POST',
    headers: {
        'Content-Type': 'application/json',
        'X-CSRFToken': getCookie('csrftoken')
    },
    body: JSON.stringify(flashcard)
});
```

Потім відбувається асинхронний запит до сервера за допомогою методу POST, щоб зберегти флеш-картку в обрані. У тілі запиту передається об'єкт flashcard, який перетворюється в рядок JSON за допомогою JSON.stringify(). У заголовках запиту вказуються тип вмісту ("Content-Type") та маркер CSRF ("X-CSRFToken"), який отримується за допомогою функції getCookie('csrftoken').

```
            if (response.ok) {
                alert("Flashcard saved to Favorites!");
            } else {
                alert("Failed to save flashcard.");
            }
        }
```

Якщо запит виконується успішно (response.ok), виводиться повідомлення про успішне збереження флеш-картки в обрані. У протилежному випадку виводиться повідомлення про невдачу у збереженні флеш-картки.

```
async function loadFavorites() {
            let response = await fetch('/get_favorites/');
            if (response.ok) {
                let favoritesPack = await response.json();
                switchPack(favoritesPack);
            } else {
                alert("Failed to load favorites.");
            }
        }
```

Функція loadFavorites() виконує асинхронний запит до сервера за допомогою методу fetch(), щоб отримати список улюблених флеш-карток. Якщо запит успішний (response.ok), то дані у форматі JSON зберігаються в змінну favoritesPack, і викликається функція switchPack(), яка відображає отриманий список флеш-карток [14].

```
        async function fetchImage(word) {
                    const clientId = 'BwffIMnCQdZEY0JqR-
r762jcLwEq2oHulRfqOKWzM8U';
                    const response = await
fetch(`https://api.unsplash.com/photos/random?query=${word}&client_id=${c
lientId}`);
                    const data = await response.json();
                    return data.urls.regular;
                }
```

Функція fetchImage(word) також виконує асинхронний запит до сервера, але вже за URL-адресою https://api.unsplash.com/photos/random?query=${word}&client_id=${clientId} для отримання випадкового зображення за ключовим словом word. Зображення отримується з ресурсу Unsplash. Після отримання даних у форматі JSON, URL зображення зберігається в змінній imageUrl.

```
                async function updateWordImage(word) {
                    const imageUrl = await fetchImage(word);
                    const wordImage =
document.getElementById('wordImage');
                    wordImage.src = imageUrl;
                    wordImage.style.width = '700px';
                    wordImage.style.height = '300px';
                }
```

Функція updateWordImage(word) викликає fetchImage(word) для отримання URL-адреси зображення за ключовим словом word. Після цього вона змінює атрибути src, width та height зображення з id wordImage таким чином, щоб відображати отримане зображення.

```
                async function updateImages() {
                    for (let i = 0; i < wordsToUpdate.length; i++) {
                        await
updateWordImage(wordsToUpdate[i].english);
```

```
        }
    }
    updateImages();
```

Функція updateImages() асинхронно оновлює зображення для всіх слів у масиві wordsToUpdate, викликаючи асинхронну функцію updateWordImage() для кожного слова.

Після цього викликається функція updateImages(), щоб розпочати оновлення зображень.

```
    document.getElementById("beginnerPack").addEventListener("click",
function () { switchPack(beginnerPack); }
    );
document.getElementById("intermediatePack").addEventListener("click",
function () { switchPack(intermediatePack); });
      document.getElementById("advancedPack").addEventListener("click",
function () { switchPack(advancedPack); });
```

Потім встановлюються прослуховувачі подій на кнопках пакунків. Кожна кнопка призначається прослуховувачу подій click, який викликає функцію switchPack() з відповідним пакунком при кліку на кнопці. Для кнопки "Favorites Pack" викликається функція loadFavorites() при кліку.

```
    document.getElementById("favoritesPack").addEventListener("click",
loadFavorites);
    document.getElementById("saveToFavoritesButton").addEventListener("
click", saveToFavorites);
```

Також встановлюється прослуховувач події на кнопці "Save to Favorites". Цей прослуховувач подій викликає функцію saveToFavorites() при кліку на кнопці "Save to Favorites".

```
function getCookie(name) {
```

```
                    let cookieValue = null;
                    if (document.cookie && document.cookie !== '') {
                        const cookies = document.cookie.split(';');
                        for (let i = 0; i < cookies.length; i++) {
                            const cookie = cookies[i].trim();
```

На останок, визначається функція getCookie(name), яка призначена для отримання значення cookie за заданим ім'ям. Функція перебирає всі cookies, розділяє їх на окремі елементи і порівнює кожен з них з ім'ям cookie, яке потрібно знайти якщо це потрібно.

```
    if (cookie.substring(0, name.length + 1) === (name + '='))
    {
                                cookieValue =
decodeURIComponent(cookie.substring(name.length + 1));
                                break;
                            }
                        }
                    }
                    return cookieValue;
                }
```

Якщо знайдено відповідне ім'я cookie, повертається його значення. Якщо ні, повертається null.

```
        function playAudio() {
                    var text =
document.getElementById('englishWord').innerText;
                    responsiveVoice.speak(text, "UK English Female");
                }
```

Ця функція призначена для відтворення аудіофайлу, що містить вимову англійського слова. Вона використовує бібліотеку ResponsiveVoice.js для відтворення звуку.

У функції спочатку отримується текст англійського слова з елементу з id 'englishWord'. Потім цей текст передається функції speak з бібліотеки ResponsiveVoice разом із параметром, який вказує на мову і стать, у якій потрібно відтворити вимову. У цьому випадку використовується мова "UK English" (англійська мова, яка використовується у Великій Британії) та стать "Female" (жіноча вимова).

Отже, при виклику цієї функції вона бере текст слова, яке потрібно вимовити, і передає його для аудіопрогравання бібліотеці ResponsiveVoice, яка відтворює звуковий файл з вимовою слова.

3.4.4 Сторінка створення flashcards

Ця сторінка відповідає за створення користувачам флеш карток та вибір куди ця флеш картка запишиться в який пак слів (Рис 3.3).

```
<body>
    <h1><a href="{% url 'index' %}
">WordSnap</a></h1>
   <div class="button-container">
    <a href="{% url 'flashcards' %}"
 class="{% if request.path
== '/flashcards/' %}active{% endif %}
">Flashcards</a>
    <a href="{% url 'tests' %}
" class="{% if request.path
== '/tests/' %}active{% endif %}
">Tests</a>
    <a href="{% url 'create_flashcard' %}
" class="{% if request.path ==
'/create_flashcard/' %}active{% endif %}
">Create Flashcards</a>
```

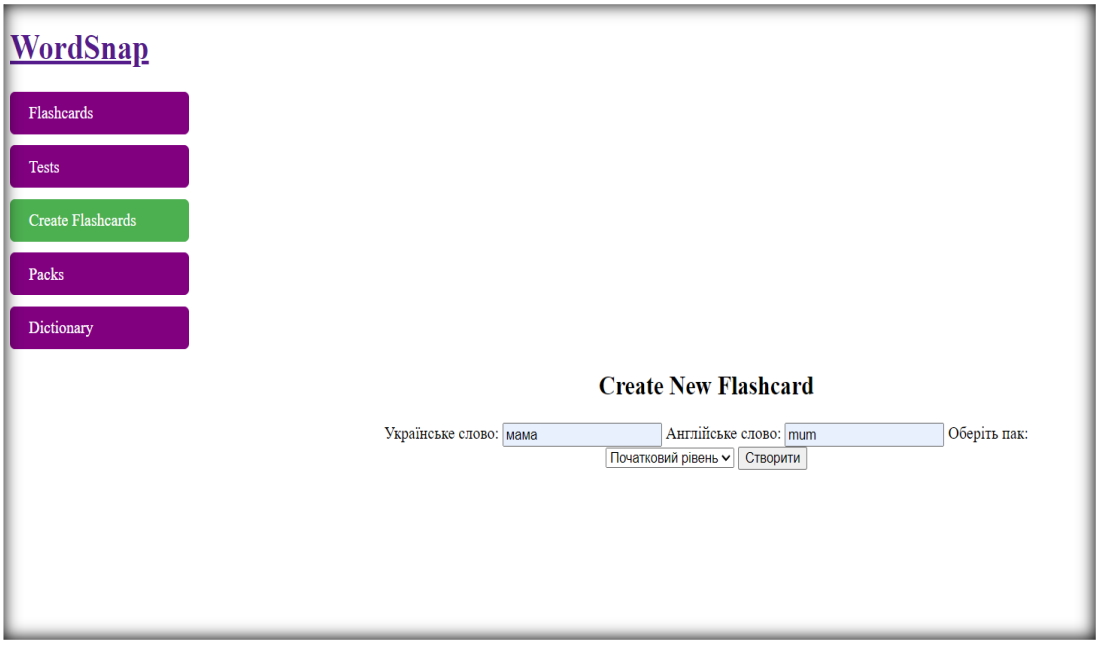


Рисунок 3.3 — Сторінка “Create flashcards”

Цей HTML-код представляє собою сторінку веб-додатка з навігаційним меню. На сторінці є заголовок першого рівня <h1>, який містить посилання на головну сторінку додатка. Нижче заголовка є контейнер з класом button-container, якому розміщені три кнопки: "Flashcards", "Tests" і "Create Flashcards".

```
        <a href="{% url 'packs' %}" class="{% if request.path ==
'/packs/' %}active{% endif %}">Packs</a>
        <a href="{% url 'dictionary' %}" class="{% if request.path ==
'/dictionary/' %}active{% endif %}">Dictionary</a>
    </div>
```

Кожне посилання має атрибут href, який вказує на URL-адресу, за якою слід перейти при кліці на посилання. Також кожне посилання має клас, який використовується для надання стилів за допомогою CSS. Умовні конструкції {% if ... %} використовуються для перевірки поточного шляху (request.path) і надання класу active активному посиланню, щоб воно виділялося на сторінці відповідно до поточної локації користувача.

```
    <div class="form-container">
```

```
        <h2>Create New Flashcard</h2>
        <form id="flashcard-form" action="{% url 'create_flashcard' %}"
method="post">
            {% csrf_token %}
<label for="ukrainian_word">Українське слово:</label>
<input type="text" id="ukrainian_word" name="ukrainian_word" required>
<label for="english_word">Англійське слово:</label>
```

Цей HTML-код представляє собою форму для створення нової картки для запам'ятовування. Вона містить різні елементи, такі як заголовок форми, поля вводу для українського та англійського слова, випадаючий список для вибору рівня складності паку, а також кнопку "Створити" для надсилання даних форми на сервер.

Кожен елемент форми зв'язаний з певним атрибутом name, який ідентифікує це поле при відправці форми. У випадку текстових полів це атрибути name="ukrainian_word" та name="english_word", а у випадку випадаючого списку це name="pack". Крім того, форма також містить захисний токен CSRF ({% csrf_token %}), який необхідно включити в Django-форми для захисту від атак CSRF та безпеки.

```
<input type="text" id="english_word" name="english_word" required>
<label for="pack">Оберіть пак:</label>
<select name="pack" id="pack">
<option value="beginner">Початковий рівень</option>
<option value="intermediate">Середній рівень</option>
<option value="advanced">Високий рівень</option>
            </select>
            <button type="submit">Створити</button>
        </form>
    </div>
```

Всі поля вводу вказуються з відповідними мітками (<label>), що підсилює їх розуміння для користувача. Крім того, форма містить заголовок <h2>, який надає коротку інструкцію для користувача щодо створення нової картки.

Кнопка "Створити" забезпечує можливість надіслати дані форми на сервер для обробки. Коли користувач натисне на цю кнопку, дані форми будуть відправлені на URL, який визначено в атрибуті action форми ({% url 'create_flashcard' %}), методом POST.

```
        window.onload = function() {
         document.getElementById('flashcard-
form').addEventListener('submit', function(event) {
             var form = this;
             event.preventDefault();
             var xhr = new XMLHttpRequest();
             xhr.open(form.method, form.action, true);
             xhr.setRequestHeader('X-Requested-With', 'XMLHttpRequest');
             xhr.onload = function () {
                 if (xhr.status === 200) {
```

Цей скрипт відповідає за обробку події подачі форми з ідентифікатором flashcard-form. При завантаженні сторінки (подія window.onload) встановлюється обробник подій для форми з ідентифікатором flashcard-form.

```
    var modal = document.getElementById('flashcard-created-modal');
                    modal.style.display = 'block';
                    setTimeout(function() {
                        modal.style.display = 'none';
                        updateFlashcardsList();
                    }, 3000);
                    form.reset();
```

Цей JavaScript-код виконує декілька дій у веб-додатку. По-перше, він отримує посилання на елемент з ідентифікатором flashcard-created-modal, який є модальним вікном або сповіщенням про створення нової картки. Потім він змінює стиль цього елемента, щоб він став видимим, встановлюючи його властивість display на значення 'block', що робить його видимим для користувача.

Після того як модальне вікно відображено протягом трьох секунд, виконується функція setTimeout, яка через три секунди знову змінює стиль модального вікна, встановлюючи його display на 'none', що приховує його від користувача. Також викликається функція updateFlashcardsList(), яка, ймовірно, оновлює список карток у веб-додатку.

```
} else {
                console.error(xhr.statusText);
            }
        };
        xhr.onerror = function () {
            console.error(xhr.statusText);
        };
```

Завершальним кроком є очищення форми (або елементу form), що найімовірніше відбувається за допомогою методу reset(), щоб після створення нової картки користувач міг швидко ввести інші дані.

```
         var formData = new FormData(form);
            formData.append('ukrainian_word',
form.elements['ukrainian_word'].value);
            formData.append('english_word',
form.elements['english_word'].value);
            xhr.send(formData);
        });
```

Цей код відповідає за формування даних та їх відправлення на сервер. Спочатку створюється об'єкт FormData для зберігання даних форми, які потім додаються до цього об'єкту за допомогою методу append(). Потім відправляються введені дані на сервер за допомогою об'єкта XMLHttpRequest та методу send(formData).

```
function updateFlashcardsList() {
```

```
            var xhr = new XMLHttpRequest();
            xhr.open('GET', '{% url 'flashcards' %}', true);
            xhr.onload = function () {
                if (xhr.status === 200) {
                    var flashcardsList =
document.getElementById('flashcards-list');
                    flashcardsList.innerHTML = xhr.responseText;
```

Деяка частина коду відноситься до функції updateFlashcardsList(), яка відповідає за оновлення списку флеш-карток на сторінці. Вона також використовує об'єкт XMLHttpRequest для виконання запиту типу GET на сервер. Після отримання відповіді вона очищає вміст елементу з ідентифікатором flashcards-list та заповнює його новим списком флеш-карток.

```
                } else {
                    console.error(xhr.statusText);
                }
            };
            xhr.onerror = function () {
                console.error(xhr.statusText);
            };
            xhr.send();
        }
        };
```

Якщо запит виявиться невдалим, помилка виводиться у консоль за допомогою console.error(xhr.statusText).

3.4.5 Сторінка словника Dictionary

Сторінка словника де користувач може бачити свої створенні флеш картки.

```
<div class="flashcards-container">
    <h2>Your Flashcards</h2>
```

```
<p>Total words: {{ user_flashcards|length }}</p>
<table>
    <thead>
        <tr>
            <th>#</th>
            <th>Українське слово</th>
            <th>Англійське слово</th>
            <th>Пак</th>
        </tr>
```

Цей код відображає сторінку з флеш-картками користувача. На сторінці є заголовок "Your Flashcards" і таблиця з відомостями про флеш-картки користувача, такими як українське слово, англійське слово та пак, до якого вони відносяться, також є номер слова (Рис 3.4).

```
<div class="flashcards-container">
    <h2>Your Flashcards</h2>
    <p>Total words: {{ user_flashcards|length }}</p>
    <table>
        <thead>
```

У таблиці виводяться усі флеш-картки, які належать поточному користувачу. Над таблицею знаходиться загальна кількість слів, яку відображається за допомогою фільтру |length.

Рисунок 3.4 — Сторінка “Dictionary”

```
        <div class="button-container">
<a href="{% url 'flashcards' %}" class="{% if request.path ==
'/flashcards/' %}active{% endif %}">Flashcards</a>
<a href="{% url 'tests' %}" class="{% if request.path == '/tests/'
%}active{% endif %}">Tests</a>
<a href="{% url 'create_flashcard' %}" class="{% if request.path ==
'/create_flashcard/' %}active{% endif %}">Create Flashcards</a>
<a href="{% url 'packs' %}" class="button">Flashcard Packs</a> <!--
<a href="{% url 'dictionary' %}" class="button {% if request.path ==
'/dictionary/' %}active{% endif %}">Mini Dictionary</a
    <div class="flashcards-container">
```

Вище таблиці розташовані посилання для навігації між сторінками додатку: "WordSnap" (посилання на головну сторінку), "Flashcards" (посилання на сторінку з флеш-картками, яке позначається як активне, якщо користувач знаходиться на цій сторінці), "Tests" (посилання на сторінку з тестами), "Create Flashcards" (посилання на сторінку для створення нових флеш-карток), "Flashcard Packs" (посилання на сторінку з паками флеш-карток) та "Mini Dictionary" (посилання на сторінку міні-словника, яке також позначається як активне, якщо користувач знаходиться на цій сторінці).

3.4.6 Сторінка тестів

На цій сторінці користувач проходить тест на рівень знання з англійської після проходження тесту користувачеві видають його рівень знань.

```
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-
scale=1.0">
        <title>WordSnap - English Level Test</title>
        <style>
            body {
                font-family: Arial, sans-serif;   }
```

```
.button-container {
    margin-bottom: 20px;}
```

Цей HTML-код визначає структуру сторінки для тестування рівня англійської мови у додатку "WordSnap". Він містить заголовок сторінки, контейнер для кнопок навігації та вбудований CSS для стилізації вигляду елементів сторінки.

Заголовок сторінки містить назву додатку "WordSnap", яка є посиланням на головну сторінку.

```
<div class="button-container">
    <button onclick="window.location.href='{% url 'flashcards' %}'">Flashcards</button>
    <button onclick="window.location.href='{% url 'tests' %}'">Tests</button>
    <button onclick="window.location.href='{% url 'vocabulary_test' %}'">Vocabulary Test</button> <!-- Нова кнопка -->
</div>
```

Контейнер для кнопок навігації містить три кнопки: "Flashcards" для переходу на сторінку з флеш-картками, "Tests" для переходу на сторінку з тестами та "Vocabulary Test" для переходу на сторінку з тестом словниковий запас слів.

```
.button-container button {
    display: inline-block;
    margin-right: 10px;
    text-decoration: none;
    padding: 10px 20px;
    border-radius: 5px;
    background-color: #800080;
    color: white;
    border: none;
    cursor: pointer; }
```

Вбудований CSS визначає стилізацію елементів сторінки, таких як кнопки навігації. Він задає вигляд кнопок, такий як колір фону, колір тексту, відступи тощо, щоб забезпечити їхню зручність і привабливий вигляд.

```
</div>
    <!-- English Proficiency Test -->
    <div id="test-container">
</div>
        <button id="submit-btn">Submit</button>
      <div id="result-container" style="display: none;">
    <!-- Результати тесту будуть відображені тут -->
</div>
```

Цей HTML-код представляє структуру сторінки для тесту на знання англійської мови. Він містить декілька елементів, кожен з яких має свій ідентифікатор для подальшого стилізування або взаємодії через JavaScript. Потім йде основний контейнер для тесту, позначений ідентифікатором test-container, в якому розміщені питання та відповіді тесту (хоча самі вони в цьому фрагменті відсутні). Після нього розташовується кнопка з ідентифікатором submit-btn, яка призначена для надсилання відповідей на тест. Під кнопкою знаходиться контейнер для відображення результатів тесту з ідентифікатором result-container, який за замовчуванням прихований завдяки інлайн-стилю display: none;. Цей контейнер буде використовуватися для показу результатів після того, як користувач натисне кнопку "Submit" (Рис 3.5).

```
    <script>
    document.getElementById("submit-btn").addEventListener("click",
function()
     { var correctAnswers = { q1: "He goes to school.",q2:
    "Beautiful or delicate",q3: "Ate",q4: "They don't like ice cream.",
q5:
```

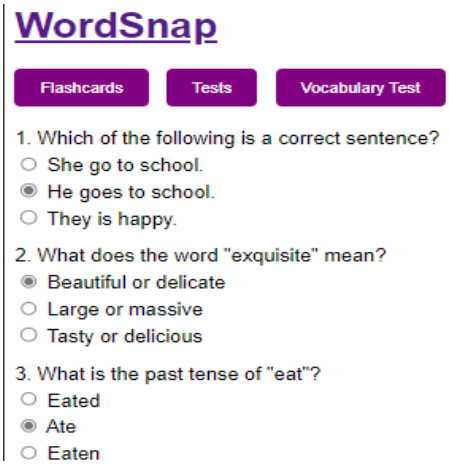


Рисунок 3.5 — Сторінка “Tests”

```
"Children",  q6: "Joyful",q7: "United States of America",q8: "They
were playing football.",q9: "Best",q10: "Pretty",q11: "went",q12:
lready",q13: "were",q14: "were",q15: "would find",q16: "will have
completed",q17: "had finished",q18: "were walking",q19: "will have
prepared", q20: "will have been waiting"};
        var answers = [];
```

Цей JavaScript-код призначений для обробки події натискання кнопки з ідентифікатором submit-btn. Коли користувач натискає цю кнопку, запускається функція, яка перевіряє відповіді на 20 питань. У коді спочатку визначається об'єкт correctAnswers, який містить правильні відповіді для кожного питання (від q1 до q20). Потім створюється порожній масив answers для збереження відповідей користувача.

```
    for (var i = 1; i <= 20; i++) {
            var input = document.querySelector("input[name='q" + i +
"']:checked");
            if (input) {
                answers.push({question: "q" + i, answer: input.value});
            }
        }
```

За допомогою циклу for програма проходить через всі питання, шукаючи вибрані відповіді (відповідні введені дані з типом radio, які мають бути позначені). Якщо відповідь знайдена, вона додається до масиву answers у вигляді об'єкта, що містить номер питання та відповідь користувача. Цей код допомагає зібрати відповіді користувача для подальшої обробки або перевірки.

```
      var correctAnswersCount = 0;
          for (var i = 0; i < answers.length; i++) {
         if (answers[i].answer === correctAnswers[answers[i].question]) {
                  correctAnswersCount++;
              }
          }
          var resultContainer = document.getElementById("result-
container");
          resultContainer.style.display = "block";
          resultContainer.innerHTML = "<p>Correct Answers: " +
correctAnswersCount + "</p>";
```

Цей JavaScript-код обчислює кількість правильних відповідей у масиві відповідей і відображає цей результат на веб-сторінці. Спочатку ініціалізується змінна correctAnswersCount значенням 0, щоб зберігати кількість правильних відповідей. Далі, в циклі for проходять через кожен елемент масиву answers. Для кожного елемента перевіряється, чи відповідає його значення правильній відповіді з масиву correctAnswers. Якщо відповідь правильна, змінна correctAnswersCount збільшується на одиницю. Після завершення циклу знаходиться HTML-елемент з ідентифікатором result-container, змінюється його стиль display на "block", що робить його видимим, і в його внутрішній HTML-код додається текст, який відображає кількість правильних відповідей.

```
          var englishLevel = "";
} else if (correctAnswersCount >= 11 && correctAnswersCount <= 15) {
} else if (correctAnswersCount >= 6 && correctAnswersCount <= 10) {
```

```
            englishLevel = "B2";
} else if (correctAnswersCount >= 4 && correctAnswersCount <= 5) {
            englishLevel = "B1";
} else if (correctAnswersCount >= 1 && correctAnswersCount <= 3) {
            englishLevel = "A2";
} else {        englishLevel = "A1"; } resultContainer.innerHTML +=
"<p>Your English Level: " + englishLevel + "</p>";
```

Цей JavaScript-код визначає рівень знання англійської мови на основі кількості правильних відповідей у тесті та відображає результат на веб-сторінці. Спочатку створюється змінна englishLevel, яка буде містити рівень англійської мови. Далі за допомогою серії умовних операторів if-else if перевіряється кількість правильних відповідей (correctAnswersCount). Залежно від цього значення змінна englishLevel присвоюється відповідний рівень від "A1" до "C2". Якщо кількість правильних відповідей знаходиться в межах від 16 до 20, рівень встановлюється як "C2"; від 11 до 15 – "C1"; від 6 до 10 – "B2"; від 4 до 5 – "B1"; від 1 до 3 – "A2"; у всіх інших випадках – "A1". Нарешті, результат додається до вмісту елемента resultContainer на веб-сторінці у вигляді параграфа з текстом "Your English Level: " та відповідним рівнем англійської мови (Рис 3.6) .

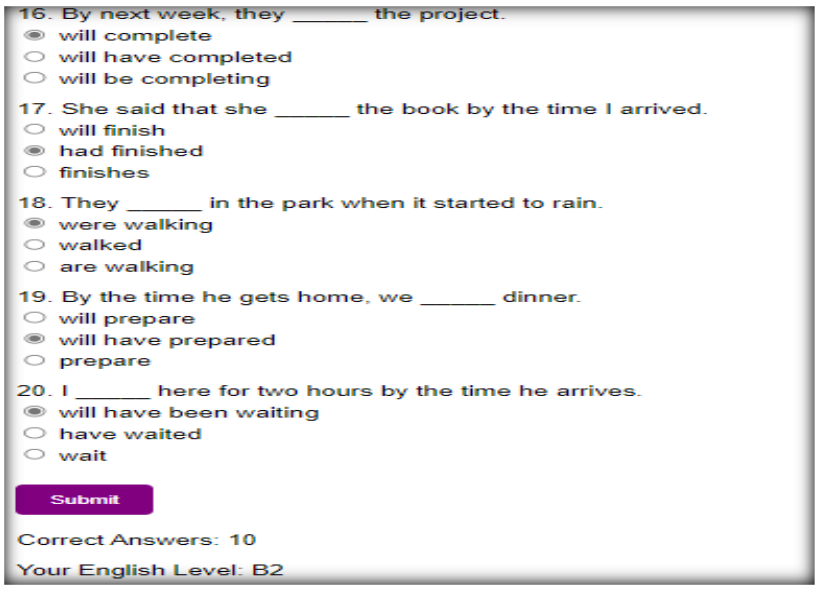


Рисунок 3.6 — Результат тесту

3.4.7 Сторінка тесту для визначення словникового запасу слів

На цій сторінці користувач проходить тест. Суть тесту така що користувач повинен вибрати слова зі списку які знає, слова є 5 рівня складності, після чого йому програма видає приблизне значення слів які він знає (Рис 3.7).

```
    <h1><a href="{% url 'index' %}">WordSnap</a></h1>
    <div class="button-container">
        <button onclick="window.location.href='{% url 'flashcards' %}'" class="{% if request.path == '/flashcards/' %}active{% endif %}">Flashcards</button>  <button onclick="window.location.href='{% url 'tests' %}'" class="{% if request.path == '/tests/' %}active{% endif %}">Tests</button>
        <button onclick="window.location.href='{% url 'vocabulary_test' %}'" class="{% if request.path == '/vocabulary_test/' %}active{% endif %}">Vocabulary Test</button>
```

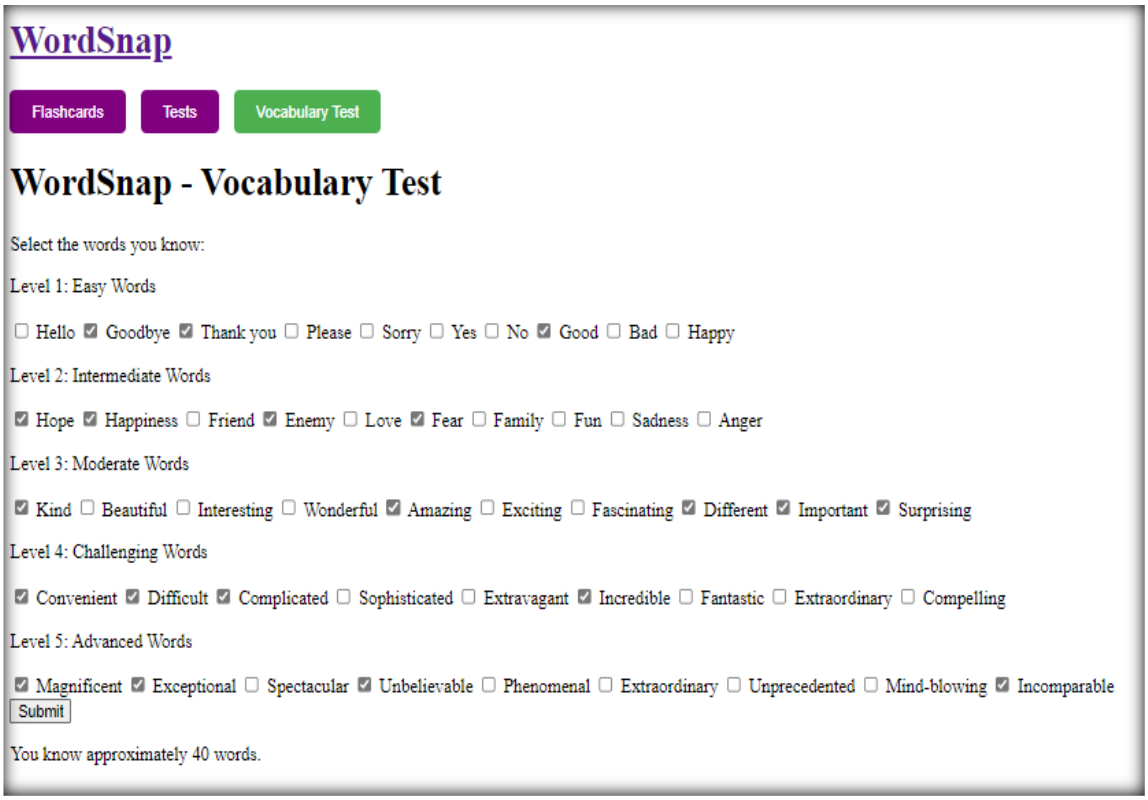


Рисунок 3.7 — Сторінка “Vocabulary test”

Цей HTML-код створює структуру сторінки для додатку "WordSnap". Він включає заголовок, контейнер з кнопками навігації ("Flashcards", "Tests", "Vocabulary Test") та відповідні CSS-стилі для визначення зовнішнього вигляду сторінки. Ці стилі також забезпечують адаптивність для різних розмірів екрану.

```
<h1>WordSnap - Vocabulary Test</h1>
    <div id="test-container">
        <p>Select the words you know:</p>
        <form id="vocabulary-test">
            <div class="question">
            </div>
            <button id="submit-btn" type="button">Submit</button>
        </form>
        <div id="result-container" style="display: none;">
            <!-- Results will be displayed here -->
        </div>
```

Цей HTML-код створює інтерфейс для тесту на словниковий запас під назвою "WordSnap — Vocabulary Test". Заголовок сторінки відображається як заголовок першого рівня. Основний контейнер для тесту містить параграф, який інформує користувача про необхідність вибору слів, які він знає. Далі йде форма з ідентифікатором vocabulary-test, всередині якої є порожній div з класом question, призначений для розміщення питань. Кнопка для відправки результатів тесту має ідентифікатор submit-btn і тип button, що запобігає її відправленню форми за замовчуванням. Після форми є контейнер для відображення результатів тесту з ідентифікатором result-container, який спочатку прихований за допомогою стилю display: none.

```
            document.getElementById("submit-
btn").addEventListener("click", function() {
                var levels = ["easy", "intermediate", "moderate",
"challenging", "advanced"];
```

```
                var totalWords = 0;
                levels.forEach(function(level) {
                    var selectedWords =
document.querySelectorAll("input[name='" + level + "']:checked").length;
                    totalWords += selectedWords;
                });
```

Цей код створює сторінку тестування словникового запасу. На сторінці є різні рівні складності слів, від простих до високих. Кожен рівень містить набір слів, до яких призначені прапорці, щоб вказати, чи знає користувач ці слова.

```
                var estimatedVocabularySize = Math.min(totalWords / 50
* 100, 10000); // Maximum 10000 words
                var resultContainer = document.getElementById("result-
container");
                resultContainer.style.display = "block";
                resultContainer.innerHTML = "<p>You know approximately
" + estimatedVocabularySize.toFixed(0) + " words.</p>";
            });
```

Цей JavaScript-код визначає змінну estimatedVocabularySize, яка обчислює приблизний розмір словникового запасу користувача на основі кількості вивчених слів (totalWords). Розрахунок відбувається шляхом ділення загальної кількості слів на 50 та множення результату на 100, з обмеженням максимального значення у 10 000 слів. Далі код отримує елемент з ідентифікатором result-container і змінює його стиль так, щоб він відображався як блоковий елемент. Після цього внутрішній HTML цього елемента оновлюється, щоб відобразити повідомлення з кількістю відомих слів, округленою до найближчого цілого числа.

3.4.8 Сторінки входу, реєстрацій, виходу

На цій сторінці користувач реєструє свій акаунт.

```
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Signup</title>
</head>
<body>
    <h1>Signup</h1>
    <form method="post"> {% csrf_token %} {{ form.as_p }}
        <button type="submit">Signup</button>
    </form>
```

Цей код створює сторінку реєстрації користувача в Django. Він використовує шаблон Django для відображення форми реєстрації. Основна структура сторінки включає в себе заголовок "Signup" (Реєстрація) та форму з полями, необхідними для реєстрації нового користувача.

У формі використовується метод POST, що означає, що дані з цієї форми будуть відправлені на сервер за допомогою HTTP-запиту POST при натисканні кнопки "Signup".{% csrf_token %} — це токен захисту від CSRF-атак (міжсайтова подібна атака), який вбудований у форму для забезпечення безпеки передачі даних.{{ form.as_p }} — це шаблонний тег Django, який відображає всі поля форми в HTML у вигляді параграфів (з використанням тега).

Після заповнення форми користувачем і натискання кнопки "Signup", дані будуть надіслані на сервер, де вони можуть бути оброблені Django для створення нового користувача в системі

```
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-
scale=1.0">
        <title>Logout</title>
    </head>
    <body>
        <h1>Logout</h1>
```

```
        <p>You have been logged out.</p>
    </body>
```

Цей код представляє сторінку виходу з системи (logout) в Django. Вона має просту структуру і містить заголовок "Logout" і повідомлення "You have been logged out."

Ця сторінка може використовуватись для відображення після того, як користувач вийшов з системи або здійснив вихід зі свого облікового запису. Вона може бути використана для повідомлення користувача про те, що його сеанс був успішно завершений.

Зазвичай ця сторінка буде використовуватись разом із функціоналом виходу з системи в Django, який забезпечує обробку виходу користувача з системи та відправку його на цю сторінку після виходу.

```
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Login</title>
</head>
<body>
    <h1>Login</h1>
   <form method="post" action="{% url 'account_login' %}">{% csrf_token
%}{{ form.as_p }}
        <button type="submit">Login</button>
    </form>
```

Цей HTML-код представляє сторінку виходу (logout) з системи. Його логіка полягає в тому, щоб просто відобразити текст "Logout" у заголовку першого рівня (<h1>) і повідомлення "You have been logged out." у параграфі (<p>). Коли користувач відвідує цю сторінку, він бачить ці тексти, що підтверджує його вихід з системи. На цій сторінці немає жодної додаткової логіки або дій, вона просто відображає повідомлення про вихід з системи.

Висновки до розділу 3

у розділі 3 було докладно описано процес розробки та реалізації веб-платформи "WordSnap" для вивчення англійської мови, яка базується на використанні флеш-карток та тестів. Використання сучасних технологій, таких як Python, фреймворк Django, а також HTML, CSS та JavaScript, дозволило створити ефективний інструмент для навчання англійської мови.

Розробка платформи включала створення моделей для управління даними, використання шаблонів для відображення інформації та впровадження динамічних елементів за допомогою JavaScript. Такий підхід забезпечив високу продуктивність та зручність використання як для мене, так і для користувачів.

Основні елементи архітектури Django, такі як моделі, відображення та шаблони, були інтегровані таким чином, щоб забезпечити ефективну роботу системи та легкість у її подальшому масштабуванні. Завдяки використанню фреймворка Django, розробники змогли зосередитися на створенні функціоналу, не витрачаючи багато часу на налаштування базових компонентів.

Впровадження інтерактивних елементів, таких як флеш-картки та тести, зробило процес вивчення мови більш захоплюючим та ефективним. Користувачі можуть створювати власні набори слів, проходити персоналізовані тести та відстежувати свій прогрес, що сприяє більшій мотивації та кращим результатам у навчанні.

У підсумку, розробка веб-платформи "WordSnap" продемонструвала можливості сучасних технологій у створенні ефективних освітніх інструментів. Ця платформа не лише полегшує процес вивчення англійської мови, але й робить його більш інтерактивним та адаптованим до потреб користувачів

ВИСНОВОК

У даній кваліфікаційній роботі виконано розробку платформи "WordSnap" для інтелектуального навчання англійської мови, яка базується на використанні флеш-карток та тестів. Проведено теоретичний аналіз наявних платформ для вивчення англійської мови, визначено їхні сильні та слабкі сторони, що дозволило сформулювати вимоги до власної розробки. Вивчено сучасні методи та технології, що використовуються у розробці освітніх платформ.

Розроблено архітектуру веб-платформи з використанням Python та фреймворка Django. Впроваджено інтерактивні елементи, такі як флеш-картки та тести, що сприяють покращенню засвоєння матеріалу. Реалізовано модулі для управління даними, користувачами та їхнім прогресом у навчанні.Оптимізовано роботу платформи для підвищення її продуктивності та ефективності. Проведено тестування платформи на групі користувачів для оцінки її ефективності та зручності у використанні. Визначено, що використання платформи сприяє значному покращенню мовних навичок користувачів. Отримані результати підтвердили високу ефективність використаних методів та технологій.

Розроблена платформа може бути використана в навчальних закладах, компаніях та організаціях для підвищення рівня англійської мови у студентів та працівників. Платформа сприятиме покращенню конкурентоспроможності користувачів у сфері програмного забезпечення завдяки вдосконаленню їхніх мовних навичок.

Робота над платформою "WordSnap" продемонструвала можливості сучасних технологій у створенні ефективних освітніх інструментів, які відповідають потребам сучасного ринку праці та сприяють професійному розвитку користувачів

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕД

1. Django Documentation. URL: https://docs.djangoproject.com/en/stable/ (дата звернення: 16.01.2024).

2. MDN Web Docs. URL: https://developer.mozilla.org/ (дата звернення: 18.03.2024).

3. Git Documentation. URL: https://git-scm.com/doc (дата звернення: 24.05.2024).

4. CSS Tricks. URL: https://css-tricks.com/ (дата звернення: 24.03.2024).

5. Ngrok Documentation. URL: https://ngrok.com/docs (дата звернення: 04.04.2024).

6. Django Packages. URL: https://djangopackages.org/ (дата звернення: 24.05.2024).

7. W3Schools Online Web Tutorials. URL: https://www.w3schools.com/ (дата звернення: 23.03.2024)

8. Python Advantages and Disadvantages – Step in the right direction вебсайт. URL: https://techvidvan.com/tutorials/python-advantages-and-disadvantages/ (дата звернення: 27.02.2023)..

9. Кнут Д. Мистецтво програмування. Том 1. Основні алгоритми. – М.: Вільямс, 2001. – 672 с.

10. HTML Documentation. URL: https://html.spec.whatwg.org/ (дата звернення: 04.04.2024).

11. PyCharm Features вебсайт. URL: https://www.jetbrains.com/pycharm/features/ (дата звернення: 01.03.2023).

12. Python вебсайт. URL: https://www.python.org/ (дата звернення: 28.02.2023).

13SQLite3 Documentation. URL: https://www.sqlite.org/docs.html (дата звернення: 24.05.2024).

14. JavaScript.info. URL: https://javascript.info/ (дата звернення: 24.05.2024).

15. Real Python: Python Tutorials. URL: https://realpython.com/ (дата звернення: 01.03.2024).

16 Platform basing Programing. Model-View-Template. URL: https://pbp-fasilkom-ui.github.io/ganjil-2023/en/assignments/tutorial/tutorial-1/ (дата звернення: 23.01.2024).

17. Python Package Index (PyPI). URL: https://pypi.org/ (дата звернення: 29.01.2024).

18. Python Documentation. URL: https://docs.python.org/3/ (дата звернення: 15.01.2024).

19. The Python Tutorial. URL: https://docs.python.org/3/tutorial/ (дата звернення: 16.03.2024).

20. Етапи життєвого циклу розробки ПЗ. URL: https://icstudio.online/post/etapi-zhittyevogo-ciklu-rozrobki-pz (дата звернення: 02.03.2023).

ДОДАТКИ

Додаток А

Файл models.py

```
from django.contrib.auth.models import User
from django.db import models

class Flashcard(models.Model):
    category = models.CharField(max_length=100, default='')  # 
Встановлюємо значення за замовчуванням
    pack = models.ForeignKey('Pack', on_delete=models.CASCADE, 
related_name='flashcards')
    ukrainian_word = models.CharField(max_length=100)
    english_word = models.CharField(max_length=100)

    def __str__(self):
        return f"{self.ukrainian_word} - {self.english_word}"

class Pack(models.Model):
    name = models.CharField(max_length=100)

class Word(models.Model):
    pack_id = models.ForeignKey(Pack, on_delete=models.CASCADE)
    ukrainian = models.CharField(max_length=100)
    english = models.CharField(max_length=100)

    def __str__(self):
        return f"{self.ukrainian} - {self.english}"

class FavoriteCard(models.Model):
    english = models.CharField(max_length=100)
    ukrainian = models.CharField(max_length=100)

    def __str__(self):
        return f"{self.english} - {self.ukrainian}"
```

Додаток Б

Файл views.py

```
from django.contrib.auth import logout
from django.contrib.auth import authenticate, login
from django.shortcuts import render, redirect
from .forms import FlashcardForm, MyCustomLoginForm,
MyCustomUserCreationForm
from .models import Flashcard, Pack
from django.shortcuts import render
from .forms import FlashcardForm, PackChoiceForm
from django.http import JsonResponse
from .models import FavoriteCard
import json

def index(request):
    return render(request, 'index.html')

def flashcards(request):
    beginner_pack = Pack.objects.get(name="Початковий")
    flashcards = Flashcard.objects.filter(pack=beginner_pack)
    if request.method == 'POST':
        form = FlashcardForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('flashcards')
    else:
        form = FlashcardForm()
    return render(request, 'flashcards.html', {'form': form,
'flashcards': flashcards})

def create_flashcard(request):
    if request.method == 'POST':
        flashcard_form = FlashcardForm(request.POST)
        pack_form = PackChoiceForm(request.POST)

        if flashcard_form.is_valid() and pack_form.is_valid():
```

```
            flashcard = flashcard_form.save(commit=False)
            flashcard.user = request.user
            flashcard.save()

            pack = pack_form.cleaned_data['pack']
            pack.flashcards.add(flashcard)
            return redirect('flashcards')

    else:
        flashcard_form = FlashcardForm()
        pack_form = PackChoiceForm()

    return render(request, 'create_flashcard.html', {'flashcard_form':
flashcard_form, 'pack_form': pack_form})
def tests(request):
    return render(request, 'tests.html')

def login_view(request):
    if request.method == 'POST':
        form = MyCustomLoginForm(request, request.POST)
        if form.is_valid():
            username = form.cleaned_data['username']
            password = form.cleaned_data['password']
            user = authenticate(request, username=username,
password=password)
            if user is not None:
                login(request, user)
                return redirect('index')
    else:
        form = MyCustomLoginForm(request)
    return render(request, 'login.html', {'form': form})


def logout_view(request):
    logout(request)
    return redirect('index')
```

```
def signup_view(request):
    if request.method == 'POST':
        form = MyCustomUserCreationForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('login')
    else:
        form = MyCustomUserCreationForm()
    return render(request, 'signup.html', {'form': form})


def mini_dictionary(request):
    flashcards = Flashcard.objects.all()
    return render(request, 'mini_dictionary.html', {'flashcards':
flashcards})


def my_flashcards(request):
    user_flashcards = Flashcard.objects.filter(user=request.user)
    return render(request, 'my_flashcards.html', {'user_flashcards':
user_flashcards})

def beginner_pack_words(request):
    beginner_pack = Pack.objects.get(name="Початковий")
    flashcards = Flashcard.objects.filter(category="Початковий")
    words_list = [{'ukrainian': flashcard.ukrainian, 'english':
flashcard.english} for flashcard in flashcards]
    return JsonResponse(words_list, safe=False)


def save_favorite(request):
    if request.method == 'POST':
        try:
            data = json.loads(request.body)
            english = data.get('english')
            ukrainian = data.get('ukrainian')
            if english and ukrainian:
```

```
                if not FavoriteCard.objects.filter(english=english,
ukrainian=ukrainian).exists():
                    favorite_card = FavoriteCard(english=english,
ukrainian=ukrainian)
                    favorite_card.save()
                    return JsonResponse({'status': 'success'},
status=201)
                return JsonResponse({'status': 'error', 'message':
'Flashcard already exists'}, status=400)
            return JsonResponse({'status': 'error', 'message': 'Invalid
data'}, status=400)
        except json.JSONDecodeError:
            return JsonResponse({'status': 'error', 'message': 'Invalid
JSON'}, status=400)
    return JsonResponse({'status': 'error', 'message': 'Invalid request
method'}, status=405)

def get_favorites(request):
    if request.method == 'GET':
        favorites = FavoriteCard.objects.all().values('english',
'ukrainian')
        return JsonResponse(list(favorites), safe=False)
    return JsonResponse({'status': 'error', 'message': 'Invalid request
method'}, status=405)

def english_level_test(request):
    if request.method == 'POST':
        q1 = request.POST.get('q1')
        q2 = request.POST.get('q2')
        return render(request, 'english_level_test.html',
{'test_results': 'Intermediate'})  # Змініть на свої дані
    return render(request, 'english_level_test.html')

def vocabulary_test(request):
    return render(request, 'vocabulary_test.html')


def packs(request):
```

```
    return render(request, 'packs.html')

def add_pack(request):
    if request.method == 'POST':
        pack_name = request.POST.get('pack-name', '')
        pack_description = request.POST.get('pack-description', '')
        return JsonResponse({'success': True})
    else:
        return JsonResponse({'success': False
def dictionary(request):
    return render(request, 'dictionary.html')
```

Додаток В

Файл index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>WordSnap</title>
    <style>
        .button-container {
            display: flex;
            flex-direction: column;
            align-items: flex-start;
            padding-left: 20px;
        }
        .button-container a {
            margin-bottom: 10px;
        }
        .user-actions {
            position: absolute;
            top: 10px;
            right: 20px;
        }
        .user-actions a {
            margin-left: 10px;
        }
    </style>
</head>
<body>
    <h1 style="font-family: 'Roboto', sans-serif;">WordSnap</h1>
    <div class="user-actions">
        {% if user.is_authenticated %}
            <a href="{% url 'logout' %}">Logout</a>
        {% else %}
```

```
            <a href="{% url 'account_login' %}">Login</a>
            <a href="{% url 'account_signup' %}">Signup</a>
        {% endif %}
    </div>
    <div class="button-container">
        <a href="{% url 'flashcards' %}"><button>Flashcards</button></a>
        <a href="{% url 'tests' %}"><button>Tests</button></a>
    </div>
</body>
</html>
```

Додаток Д

Файл flashcards.html

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Flashcards</title>
    <style>
        .button-container {
            text-align: left;
            margin-bottom: 20px;
        }

        .btn.btn-primary {
            margin-right: 10px;
            padding: 10px 20px;
            border-radius: 5px;
            background-color: #800080;
            color: white;
        }

        .btn.btn-primary.active {
            background-color: #4CAF50;
        }

        .button-container a {
            display: inline-block;
            margin-right: 10px;
            text-decoration: none;
            padding: 10px 20px;
            border-radius: 5px;
            background-color: #800080;
```

```
    color: white;
}

.button-container a.active {
    background-color: #4CAF50;
}

#flashcardContainer {
    width: 100%;
    height: 100vh;
    display: flex;
    justify-content: center;
    align-items: center;
    flex-direction: column;
}

#flashcard {
    width: 1000px;
    height: 400px;
    border: 2px solid black;
    display: flex;
    justify-content: center;
    align-items: center;
    flex-direction: column;
    cursor: pointer;
    margin-bottom: 20px;
}

#wordContainer {
    flex: 1;
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
}

#imageContainer {
    flex: 1;
```

```
        display: flex;
        justify-content: flex-start;
        align-items: center;
    }

    #wordImage {
        max-width: 100%;
        max-height: 100%;
    }

    #ukrainianWord, #englishWord {
        font-size: 24px;
    }

    .highlighted {
        background-color: #FFD700;
    }

    .extra-button-container {
        text-align: left;
    }

    .delete-button, .save-button {
        margin-top: 20px;
        padding: 10px 20px;
        border-radius: 5px;
        background-color: #DC143C;
        color: white;
        cursor: pointer;
    }

    @media (max-width: 600px) {
        .button-container a {
            display: block;
            margin-bottom: 10px;
        }
    }
</style>
```

```
</head>
<body>
    <h1><a href="{% url 'index' %}">WordSnap</a></h1>
    <div class="button-container">
        <a href="{% url 'flashcards' %}" class="{% if request.path ==
'/flashcards/' %}active{% endif %}">Flashcards</a>
        <a href="{% url 'tests' %}" class="{% if request.path ==
'/tests/' %}active{% endif %}">Tests</a>
        <a href="{% url 'create_flashcard' %}" class="{% if request.path
== '/create_flashcard/' %}active{% endif %}">Create Flashcards</a>
    </div>

    <div class="button-row">
        <button class="btn btn-primary" id="beginnerPack">Beginner
Pack</button>
        <button class="btn btn-primary"
id="intermediatePack">Intermediate Pack</button>
        <button class="btn btn-primary" id="advancedPack">Advanced
Pack</button>
        <button class="btn btn-primary" id="animalsPack">Animals
Pack</button>
        <button class="btn btn-primary" id="schoolPack">School
Pack</button>
        <button class="btn btn-primary" id="householdPack">Household
Pack</button>
        <button class="btn btn-primary" id="streetPack">Street
Pack</button>
        <button class="btn btn-primary" id="foodPack">Food Pack</button>
        <button id="favoritesPack" class="btn btn-primary">Favorites
Pack</button>
    </div>

    <div id="flashcardContainer" style="display: flex;">
        <div id="flashcard">
            <div id="wordContainer">
                <div id="englishWord">English Word</div>
                <div id="ukrainianWord" style="display: flex;">Слово по-
українськи</div>
```

```
            </div>
            <div id="imageContainer" style="padding-top: 10px; position:
relative;">
                <img id="wordImage" src="" style="max-width: 100%; max-
height: 100%;">
                <button class="btn btn-primary" onclick="playAudio()"
style="position: absolute; bottom: 10px; right: 10px;">Play
Audio</button>
            </div>
        </div>
        <div class="extra-button-container">
            <button class="delete-button" id="deleteButton">Delete
Flashcard</button>
            <button class="save-button" id="saveToFavoritesButton">Save
to Favorites</button>
        </div>
    </div>

    <script
src="https://code.responsivevoice.org/responsivevoice.js?key=MZawgbhR"></
script>
    <script>
        document.addEventListener("DOMContentLoaded", function() {
        let beginnerPack = [
        ];
        let intermediatePack = [
        ];
        let advancedPack = [
let currentPack = beginnerPack;
let currentIndex = 0;
let showingEnglish = true;

document.getElementById("flashcard").addEventListener("click", function
() {
    if (showingEnglish) {
        document.getElementById("englishWord").style.display = "none";
        document.getElementById("ukrainianWord").style.display = "block";
    } else {
```

```
        document.getElementById("englishWord").style.display = "block";
        document.getElementById("ukrainianWord").style.display = "none";
        currentIndex++;
        if (currentIndex >= currentPack.length) {
            currentIndex = 0;
            document.getElementById("flashcardContainer").style.display =
"none";
        } else {
            showFlashcard();
        }
    }
    showingEnglish = !showingEnglish;
});

async function showFlashcard() {
    const englishWord = currentPack[currentIndex].english;
    const ukrainianWord = currentPack[currentIndex].ukrainian;

    document.getElementById("englishWord").innerText = englishWord;
    document.getElementById("ukrainianWord").innerText = ukrainianWord;

    if (showingEnglish) {
        document.getElementById("englishWord").style.display = "block";
        document.getElementById("ukrainianWord").style.display = "none";
    } else {
        document.getElementById("englishWord").style.display = "none";
        document.getElementById("ukrainianWord").style.display = "block";
    }

    await updateWordImage(englishWord);
    playAudio();
}

function switchPack(pack) {
    currentPack = pack;
    currentIndex = 0;
    showingEnglish = true; // Починаємо з англійського слова при
перемиканні пачки
```

```
    showFlashcard();
}

const packButtons = document.querySelectorAll(".button-row button");

packButtons.forEach(button => {
    button.addEventListener("click", function() {
        packButtons.forEach(btn => btn.classList.remove("active"));
        this.classList.add("active");

        const packId = this.id;

        switch (packId) {
            case 'beginnerPack':
                switchPack(beginnerPack);
                break;
            case 'intermediatePack':
                switchPack(intermediatePack);
                break;
            case 'advancedPack':
                switchPack(advancedPack);
                break;
            case 'animalsPack':
                switchPack(animalsPack);
                break;
            case 'schoolPack':
                switchPack(schoolPack);
                break;
            case 'householdPack':
                switchPack(householdPack);
                break;
            case 'streetPack':
                switchPack(streetPack);
                break;
            case 'foodPack':
                switchPack(foodPack);
                break;
            case 'favoritesPack':
```

```
                loadFavorites();
                break;
            default:
                console.error("Unknown pack ID:", packId);
                break;
        }
    });
});

function deleteFlashcard() {
    if (currentPack.length === 0) {
        alert("There are no flashcards to delete.");
        return;
    }

    currentPack.splice(currentIndex, 1);

    if (currentIndex >= currentPack.length) {
        currentIndex = 0;
    }

    if (currentPack.length === 0) {
        document.getElementById("flashcardContainer").style.display =
"none";
    } else {
        showFlashcard();
    }
}

document.getElementById("deleteButton").addEventListener("click",
deleteFlashcard);

async function saveToFavorites() {
    let flashcard = {
        english: document.getElementById("englishWord").innerText,
        ukrainian: document.getElementById("ukrainianWord").innerText
    };
```

```
    let response = await fetch('/save_favorite/', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
            'X-CSRFToken': getCookie('csrftoken')
        },
        body: JSON.stringify(flashcard)
    });

    if (response.ok) {
        alert("Flashcard saved to Favorites!");
    } else {
        alert("Failed to save flashcard.");
    }
}

async function loadFavorites() {
    let response = await fetch('/get_favorites/');
    if (response.ok) {
        let favoritesPack = await response.json();
        switchPack(favoritesPack);
    } else {
        alert("Failed to load favorites.");
    }
}

async function fetchImage(word) {
    const clientId = 'BwffIMnCQdZEY0JqR-r762jcLwEq2oHulRfqOKWzM8U';
    const response = await
fetch(`https://api.unsplash.com/photos/random?query=${word}&client_id=${c
lientId}`);
    const data = await response.json();
    return data.urls.regular;
}

async function updateWordImage(word) {
    const imageUrl = await fetchImage(word);
    const wordImage = document.getElementById('wordImage');
```

```
    wordImage.src = imageUrl;
    wordImage.style.width = '700px';
    wordImage.style.height = '300px';
}

async function updateImages() {
    for (let i = 0; i < wordsToUpdate.length; i++) {
        await updateWordImage(wordsToUpdate[i].english);
    }
}

updateImages();

document.getElementById("beginnerPack").addEventListener("click",
function () { switchPack(beginnerPack); });
document.getElementById("intermediatePack").addEventListener("click",
function () { switchPack(intermediatePack); });
document.getElementById("advancedPack").addEventListener("click",
function () { switchPack(advancedPack); });
document.getElementById("animalsPack").addEventListener("click", function
() { switchPack(animalsPack); });
document.getElementById("schoolPack").addEventListener("click", function
() { switchPack(schoolPack); });
document.getElementById("householdPack").addEventListener("click",
function () { switchPack(householdPack); });
document.getElementById("streetPack").addEventListener("click", function
() { switchPack(streetPack); });
document.getElementById("foodPack").addEventListener("click", function ()
{ switchPack(foodPack); });
document.getElementById("favoritesPack").addEventListener("click",
loadFavorites);
document.getElementById("saveToFavoritesButton").addEventListener("click"
, saveToFavorites);

function getCookie(name) {
    let cookieValue = null;
    if (document.cookie && document.cookie !== '') {
        const cookies = document.cookie.split(';');
```

```
        for (let i = 0; i < cookies.length; i++) {
            const cookie = cookies[i].trim();
            if (cookie.substring(0, name.length + 1) === (name + '=')) {
                cookieValue =
decodeURIComponent(cookie.substring(name.length + 1));
                break;
            }
        }
    }
    return cookieValue;
}

function playAudio() {
    var text = document.getElementById('englishWord').innerText;
    responsiveVoice.speak(text, "UK English Female");
}

showFlashcard();
        }
);
    </script>
</body>
</html>
```

Додаток И

Файл create_flashcards.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Create Flashcards - WordSnap</title>
    <style>
        .button-container a {
            display: block;
            margin-bottom: 10px;
            text-decoration: none;
            padding: 10px 20px;
            border-radius: 5px;
            width: 150px;
            background-color: #800080;
            color: white;
        }

        .button-container a.active {
            background-color: #4CAF50;
            color: white;
        }

        .button-container {
            text-align: left;
        }

        .form-container {
            width: 50%;
            margin: 0 auto;
            text-align: center; */
        }
```

```
        .modal {
            display: none;
            position: fixed;
            z-index: 1;
            left: 0;
            top: 0;
            width: 100%;
            height: 100%;
            overflow: auto;
            background-color: rgba(0,0,0,0.4 */
        }

        .modal-content {
            background-color: #fefefe;
            margin: 15% auto; та вертикалі */
            padding: 20px;
            border: 1px solid #888;
            width: 80%;
        }
    </style>
</head>
<body>
    <h1><a href="{% url 'index' %}">WordSnap</a></h1>
   <div class="button-container">
    <a href="{% url 'flashcards' %}" class="{% if request.path ==
'/flashcards/' %}active{% endif %}">Flashcards</a>
    <a href="{% url 'tests' %}" class="{% if request.path == '/tests/'
%}active{% endif %}">Tests</a>
    <a href="{% url 'create_flashcard' %}" class="{% if request.path ==
'/create_flashcard/' %}active{% endif %}">Create Flashcards</a>
    <a href="{% url 'packs' %}" class="{% if request.path == '/packs/'
%}active{% endif %}">Packs</a>
    <a href="{% url 'dictionary' %}" class="{% if request.path ==
'/dictionary/' %}active{% endif %}">Dictionary</a>
</div>


    <!-- Форма створення флеш карток -->
```

```
<!-- Форма створення флеш карток -->
<div class="form-container">
    <h2>Create New Flashcard</h2>
    <form id="flashcard-form" action="{% url 'create_flashcard' %}"
method="post">
        {% csrf_token %}
        <label for="ukrainian_word">Українське слово:</label>
        <input type="text" id="ukrainian_word" name="ukrainian_word"
required>

        <label for="english_word">Англійське слово:</label>
        <input type="text" id="english_word" name="english_word"
required>

        <!-- Випадаючий список для вибору паку -->
        <label for="pack">Оберіть пак:</label>
        <select name="pack" id="pack">
            <option value="beginner">Початковий рівень</option>
            <option value="intermediate">Середній рівень</option>
            <option value="advanced">Високий рівень</option>
            <!-- Додайте інші паки за потребою -->
        </select>

        <button type="submit">Створити</button>
    </form>
</div>
<!-- Список флеш-карток -->
<div id="flashcards-list">
    <!-- Тут буде відображений список флеш-карток, отриманий через AJAX -
->
</div>


    <!-- Спливаюче вікно -->
    <div id="flashcard-created-modal" class="modal">
        <div class="modal-content">
            <h4>Flashcard Created</h4>
            <p>Your flashcard has been successfully created!</p>
```

```
        </div>
    </div>

    <script>
   window.onload = function() {
    document.getElementById('flashcard-form').addEventListener('submit',
function(event) {
        var form = this;
        event.preventDefault(); // Зупиняємо дійсну подію submit

        var xhr = new XMLHttpRequest();
        xhr.open(form.method, form.action, true);
        xhr.setRequestHeader('X-Requested-With', 'XMLHttpRequest');
        xhr.onload = function () {
            if (xhr.status === 200) {
                // Після успішного відправлення форми, показуємо
спливаюче вікно
                var modal = document.getElementById('flashcard-created-
modal');
                modal.style.display = 'block';
                setTimeout(function() {
                    modal.style.display = 'none';
                    updateFlashcardsList();
                }, 3000);
                form.reset();
            } else {
                console.error(xhr.statusText);
            }
        };
        xhr.onerror = function () {
            console.error(xhr.statusText);
        };

        var formData = new FormData(form);
        formData.append('ukrainian_word',
form.elements['ukrainian_word'].value);
        formData.append('english_word',
form.elements['english_word'].value);
```

```
        // Відправляємо дані на сервер
        xhr.send(formData);
    });

function updateFlashcardsList() {
    var xhr = new XMLHttpRequest();
    xhr.open('GET', '{% url 'flashcards' %}', true);
    xhr.onload = function () {
        if (xhr.status === 200) {
            var flashcardsList = document.getElementById('flashcards-
list');
            // Очищаємо список
            flashcardsList.innerHTML = xhr.responseText;
        } else {
            console.error(xhr.statusText);
        }
    };
    xhr.onerror = function () {
        console.error(xhr.statusText);
    };
    xhr.send();
}

};

    </script>
</body>
</html>
```

метадані

Заголовок

Розробка платформи для інтелектуального навчання англійської мови за допомогою онлайн-флеш-карти та тестів

Автор
Тицький Р.

Науковий керівник / Експерт
кандидат технічних наук Сергій Ващишак

підрозділ
King Danylo University

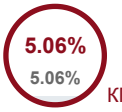
Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

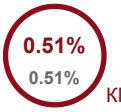
Заміна букв	ß	0
Інтервали	A→	0
Мікропробіли	·	1
Білі знаки	ß	0
Парафрази (SmartMarks)	a	52

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.



КП 1



КП 2

25
Довжина фрази для коефіцієнта подібності 2

16239
Кількість слів

132420
Кількість символів

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)	
1	https://stackoverflow.com/questions/10663446/post-method-always-return-403-forbidden	29	0.18 %
2	https://www.pythontutorial.net/django-tutorial/django-login/	28	0.17 %
3	https://essuir.sumdu.edu.ua/bitstream/123456789/91629/1/Koropets_mag_rob.pdf	26	0.16 %
4	https://www.evkova.org/kursovye-raboty/razrabotka-sajta-gornodobyivayuschej-kompanii-granit	23	0.14 %
5	https://www.educba.com/django-reverse/	22	0.14 %
6	https://ela.kpi.ua/bitstream/123456789/60274/1/Kozlenko_bakalavr.pdf	20	0.12 %