КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра теорії та технології програмування

Кваліфікаційна робота
на здобуття ступеня бакалавра

за спеціальністю 122 Комп’ютерні науки

на тему:

РОЗРОБКА ВЕБ-ЗАСТОСУНКУ

ДЛЯ ВИВЧЕННЯ ІНОЗЕМНИХ МОВ

Виконала студентка 4-го курсу
Анна АЛЄКСЄЄНКО


(підпис)

Науковий керівник:
доцент, кандидат технічних наук
Олексій ТКАЧЕНКО


(підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент


(підпис)

Роботу розглянуто й допущено до захисту на засіданні кафедри математичних основ комп’ютерних наук
«05» червня 2023 р.,
протокол № 18
Завідувач кафедри
Микола НІКІТЧЕНКО

(підпис)

Київ – 2023

РЕФЕРАТ

Обсяг роботи 54 сторінки, 26 ілюстрацій, 11 таблиць, 38 джерел посилань, 1 додаток.

ВЕБ-ЗАСТОСУНОК, ВИВЧЕННЯ ІНОЗЕМНИХ МОВ, СЛОВНИК, ФЛЕШ-КАРТКА, GOOGLE CLOUD TRANSLATION API, JAVASCRIPT, POSTGRESQL, REACT, REST API, SPRING BOOT.

Об’єктом дослідження є процес вивчення іноземних мов з використанням інтерактивних флеш-карток. Предметом дослідження є системи для вивчення правил і слів іноземних мов. Метою дослідження є створення веб-застосунку для формування флеш-карток, що містять правила іноземної мови, та словників з автоматичним перекладом слів, завантаження та читання книжок іноземною мовою з можливістю додавання слів з книги у наявні словники, вивчення створених карток і слів та відслідковування прогресу навчання.

Методами дослідження є аналіз, порівняння, комп’ютерне моделювання, розробка програмного продукту, метод вивчення флеш-карток з адаптивною складністю. Інструменти дослідження: інтегровані середовища розробки WebStorm та Intellij IDEA, мови програмування JavaScript та Java, мова таблиць стилів CSS, фреймворк Spring Boot з Spring Security, бібліотеки React, Redux та Redux-Saga, реляційна база даних PostgreSQL, прикладний програмний інтерфейс Google Cloud Translation API.

Результати роботи: виконано загальний огляд застосунків для вивчення іноземних мов на ринку з аналізом переваг та недоліків. Спроєктовано та розроблено веб-застосунок “Vocab” для полегшення та систематизації процесу вивчення іноземних мов. Застосунок використовує алгоритм для короткочасного запам’ятовування з використанням адаптивної складності, дозволяючи швидко опановувати нові мови, а також надає засоби спільного доступу до наборів карток та словників іншим зареєстрованим користувачам.

Програмний застосунок “Vocab” може застосовуватися в навчальному процесі в школах та університетах для опанування іноземних мов учнями чи студентами, а також усіма людьми, які хочуть пришвидшити процес вивчення мов.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

СУБД – система управління базами даних;

ACID – atomicity, consistency, isolation, durability, атомарність, узгодженість, ізольованість, довговічність;

API – Application Programming Interface, прикладний програмний інтерфейс;

DOM – Document Object Model, об'єктна модель документа;

HTML – HyperText Markup Language, мова розмітки гіпертексту;

HTTP – Hypertext Transfer Protocol, протокол передачі гіпертексту;

JSON – JavaScript Object Notation, запис об'єктів JavaScript;

JWT – JSON Web Token, токен доступу на основі формату JSON;

ORM – Object-Relational Mapping, об'єктно-реляційна проєкція;

REST – Representational State Transfer, передача репрезентативного стану;

URI – Uniform Resource Identifier, уніфікований ідентифікатор ресурсів;

URL – Uniform Resource Locator, уніфікований локатор ресурсів;

UX – User Experience, досвід користування.

ВСТУП

Оцінка сучасного стану об’єкта розробки. У наш час невід’ємною складовою щоденного життя є комунікація. Через процеси глобалізації ми маємо спілкуватися з людьми з інших культур, що найчастіше розмовляють іноземними мовами, а також користуватися іноземними джерелами інформації. Це зумовлює зростання попиту на вивчення іноземних мов та потребу у системах, що допомагають вивчати мови за якомога менший проміжок часу. Все більше людей розглядають веб-застосунки як основний засіб вивчення іноземних мов через доступність та легкість використання, а також можливість навчатися онлайн без прив’язки до визначеної локації навчального центру.

Актуальність роботи та підстави для її виконання. На сьогоднішній день велика кількість людей в Україні співпрацює з іноземними замовниками або ж комунікує з колегами з інших країн. Наприклад, англійська мова є основною у дипломатії, бізнесі, інформаційних технологіях, політиці. Вивчення нової мови розширює кар’єрні можливості людини. Впровадження зручного веб-застосунку для вивчення іноземних мов допомагає вирішити низку проблем, пов’язаних з необхідністю постійно мати поряд конспекти чи книги, намаганням вивчити неструктуровану інформацію, яка важко запам’ятовується візуально, та відсутністю статистики вивчення окремих термінів мови чи правил. Застосунок може допомогти учням та студентам швидко освоїти матеріал, не витрачаючи багато часу на написання конспектів, а дорослим людям завжди знаходити час на заняття. Користувачі також можуть бути впевнені, що їхні дані не будуть втрачені при зміні пристрою, адже при використанні веб-застосунку всі дані зберігаються на віддаленому сервері.

Серед наявних на ринку продуктів є багато систем для вивчення мов, проте більшість з них використовує алгоритми для довгострокового запам’ятовування для вивчення матеріалу користувачами. Згідно з дослідженням [1], такі алгоритми не є ефективними для досягнення швидких результатів в умовах нестачі часу (наприклад, при підготовці до контрольних робіт та іспитів). Розробка застосунку для вивчення іноземних мов, що використовує алгоритм для короткочасного

запам’ятовування дозволяє вирішити проблему швидкого та ефективного засвоєння матеріалу.

Мета й завдання роботи. Метою кваліфікаційної роботи є створення веб-застосунку “Vocab” для вивчення іноземних мов з використанням флеш-карток. Для досягнення цієї мети поставлено такі завдання:

- дослідити алгоритми запам'ятовування з використанням флеш-карток;
- проаналізувати існуючі веб-застосунки для вивчення іноземних мов та визначити функціонал, якого не вистачає більшій кількості користувачів;
- обрати засоби та інструменти розробки застосунку;
- спроєктувати архітектуру застосунку та схему бази даних;
- розробити серверну частину застосунку;
- розробити користувацький інтерфейс;

Об'єкт, методи й інструменти дослідження. Об’єктом дослідження є процес вивчення іноземних мов з використанням інтерактивних флеш-карток. Предметом дослідження є системи для вивчення правил і слів іноземних мов.

Методами дослідження є аналіз, порівняння, комп’ютерне моделювання, розробка програмного продукту, метод вивчення флеш-карток з адаптивною складністю.

Для розробки користувацького інтерфейсу веб-застосунку було використано мову програмування JavaScript, мову таблиць стилів CSS, бібліотеки React, Redux та Redux-Saga. Для розробки серверної частини застосунку використовувалася мова програмування Java, фреймворк Spring Boot разом з Spring Security, реляційна база даних PostgreSQL, а також Google Cloud Translation API для імплементації функціоналу автоматичного перекладу. Робота виконувалася в IDE WebStorm та Intellij IDEA.

Можливі сфери застосування. Розроблений веб-застосунок “Vocab” можна використовувати у таких сферах:

- вивчення іноземних мов у середніх та вищих навчальних закладах освіти. Застосунок буде гарним доповненням до вже існуючих підручників та може

полегшити виконання домашнього завдання і підготовку до контрольних робіт учням/студентам;

- вивчення іноземних мов на мовних курсах. Функція спільного доступу до наборів карток та словників може допомогти викладачам у зручній формі надавати доступ до пройдених на уроці матеріалів;
- самостійне опанування іноземної мови людиною. Застосунок надає основні засоби для пришвидшеного вивчення мови без вчителя.

РОЗДІЛ 1. ОГЛЯД АЛГОРИТМІВ ВИВЧЕННЯ ФЛЕШ-КАРТОК

1.1 Огляд алгоритмів для довготривалої пам’яті

Дослідження на тему покращення довготривалого запам’ятовування інформації почалися з гіпотези німецького психолога Германа Еббінґауза, який вивчав відсоток забування інформації людиною в залежності від часу [2]. Згідно з гіпотезою, необхідно робити декілька повторень вивченого, щоб не забувати інформацію, а після кожного повторення проміжок часу до наступного можна збільшувати. Окрім цього, складання розкладу повторень інформації з урахуванням проміжків між навчанням є набагато більш ефективним у довгостроковій перспективі, ніж часті повторення інформації за один чи декілька разів протягом малого періоду часу.

На основі гіпотези Еббінґауза утворилася найпопулярніша на сьогодні методика довгострокового засвоєння матеріалу – розподілене повторення. Дана методика базується на встановленні інтервалів між повтореннями вивчення карток на основі ступеня їх вивчення. Таким чином, з кожним запам’ятовуванням флеш-картки інтервали між її показом збільшуються, дозволяючи при цьому вивчати весь матеріал поступово та планово по днях [3].

Виділяють такі види алгоритмів, що використовують розподілене повторення:

- система Лейтнера;
- алгоритми SuperMemo з різними параметрами;
- алгоритми з використанням нейронних мереж, тощо.

Дані алгоритми відрізняються способом визначення довжини інтервалу повторення для карток.

1.1.1 Система Лейтнера

Система Лейтнера була розоблена німецьким науковим журналістом Себастьяном Лейтнером у 1972 році. Вона є доволі простою у порівнянні з більш

сучасними системами і направлена здебільшого на подовження періоду, на який запам’ятовується інформація [4].

Система Лейтнера оперує флеш-картками для засвоєння інформації. Картки розподіляються по упорядкованим групам за рівнем знань – чим краще вивчена інформація, тим в вищу за рівнем групу потраляє картка. Якщо людина дає неправильну відповідь при показі флеш-картки, дана картка потряпляє у групу нижче, при правильній відповіді – у групу вище.

Оригінальна система Лейтнера мала п’ять рівнів груп карток, для кожної з яких були визначені статичні інтервали повторення інформації: 1 день, 2-3 дні, 3-7 днів, 4-15 днів та 5-20 днів для кожного рівня відповідно. Дана система є досить гнучкою та дозволяє змінювати кількість груп для карток, через що активно використовувалася у застосунках для вивчення матеріалу до появи алгоритмів з використанням машинного навчання.

1.1.2 Алгоритми SuperMemo

Сім’я алгоритмів SuperMemo була розроблена Петром Возняком у 1985 році та постійно покращувалася після цього [5]. Імплементація SuperMemo є дещо складнішою від системи, запропонованої Лейтнером. Найперший алгоритм в цій сім’ї має назву SM-0 та є версією алгоритму для використання на папері. Першою комп’ютерною імплементацією став алгоритм SM-2. Даний алгоритм використовував поняття коефіцієнту легкості, що показував легкість запам’ятовування та відтворення певної порції інформації з пам’яті. Початкове значення коефіцієнта обиралося на рівні 2.5, а згодом зменшувалося з кожною помилкою при згадуванні інформації. Якість відповіді на питання оцінювалося від 0 (повне забування) до 5 (ідеальне відтворення). Повторення інформації тривали, допоки кожна з карток не отримувала коефіцієнт легкості мінімально на рівні 4. Більш детальний опис алгоритму та дослідження його ефективності наведено у джерелі [6]. Згідно з даними дослідженнями автора Петра Возняка, алгоритм сприяє довгостроковому запам’ятовуванню близько 92% початково вивченої інформації.

1.2 Алгоритми для короткочасної пам’яті

Існує багато досліджень, що підтверджують більшу ефективність алгоритмів для довгострокового запам’ятовування у порівнянні з короткостроковим при регулярних заняттях. Проте доволі часто люди не мають часу на повне засвоєння матеріалу з використанням інтервалів і потребують більш швидкого способу вивчення. Використання алгоритмів для довгострокової пам’яті у таких умовах є неефективним. У дослідженні [1] було порівняно ефективність навчання всього об’єму інформації з використанням інтервалів і вивчення інформації послідовно з розбиттям на частини. Було отримано такі результати: при проведенні декількох сесій вивчення матеріалу, навчання з інтервалами було ефективнішим у 90% випадків, але після першої сесії для 72% учасників дослідження вивчення інформації з розбиттям на частини виявилося значно результативнішим. Саме вивчення інформації з розбиттям на частини було покладено в основу алгоритму даної кваліфікаційної роботи.

Серед алгоритмів для короткострокового запам’ятовування інформації виділяють вивчення у довільному порядку, вивчення з пріоритезацією матеріалу, при якому найважливіші питання задаються частіше за менш важливі, змішування запитань та відповідей, при якому учень не просто повинен відтворити відповідь, а і за відповіддю згадати питання, організацію питань за категоріями, а також вивчення з адаптивною складністю [7]. При навчанні з адаптивною складністю кожна картка має складність, що вираховується в залежності від відповідей учня. Чим менше правильних відповідей дано на питання – тим більше його складність і тим частіше картку буде запропоновано для вивчення знову.

1.3 Використаний в роботі алгоритм

Оскільки застосунок, розроблений у контексті даної кваліфікаційної роботи, сфокусований на учнях та студентах освітніх навчальних закладів та людях, що бажають швидко вивчити іноземну мову, алгоритми для довготривалого запам’ятовування не підходили б більшості користувачів через наявність чітких крайніх термінів вивчення матеріалу. Окрім цього, за даними одного з

найпопулярніших застосунків з флеш-картками для засвоєння матеріалу Quizlet, 95% учнів виділяють не більше 4 днів на вивчення набору карток, що є занадто малим інтервалом для ефективного застосування більшості алгоритмів для довготривалого запам’ятовування [8]. Щоб задовольнити потреби переважної більшості користувачів, було обрано імплементувати один з варіантів алгоритму для короткострокової пам’яті з використанням адаптивної складності. Даний алгоритм забезпечить швидке засвоєння інформації та дозволить користувачам бачити прогрес з першого дня використання застосунку.

На вхід алгоритму задається максимальний коефіцієнт вивчення, при досягненні якого одиниця матеріалу вважається вивченою. Рівень вивчення є числом, що характеризує частоту правильних відповідей з врахуванням неправильних. Для кожної одиниці матеріалу рівень починається з 0. При правильній відповіді до рівня додається 1, при неправильній – віднімається 1. Рівень вивчення є завжди невід’ємним числом, не більшим за заданий максимальний, тобто у випадку неправильної відповіді для одиниці матеріалу з нульовим рівнем, рівень все одно залишиться нульовим. Відсоток вивчення рахується як рівень вивчення, поділений на максимальний рівень та помножений на 100. При навчанні картки сортуються за рівнем вивчення. Якщо рівень вивчення для карток однаковий, вони показуються у довільному порядку, щоб уникнути запам’ятовування учнем відповідей за сусідніми картками.

Застосунок надає можливість користувачеві створювати велику кількість наборів карток. Алгоритм є найбільш ефективним, коли кожен набір карток має невеликий розмір.

РОЗДІЛ 2. ОГЛЯД НАЯВНИХ ПРОГРАМНИХ РІШЕНЬ

Наразі найбільшу частину ринку займають застосунки, які використовують розподілене повторення як основний алгоритм. Також присутні варіації алгоритмів з використанням машинного навчання. Розглянемо основні застосунки, наявні на ринку.

2.1 Quizlet

Програма Quizlet є системою навчання з використанням флеш-карток, що доступна у веб версії та на мобільних платформах Android та iOS. В одних з перших версій застосунку Quizlet використовував простий алгоритм навчання, описаний детально у джерелі [8]: користувач відповідав на одне питання для кожної картки, потім для всіх карток, для яких була дана неправильна відповідь. Цикл повторювався, допоки відповіді на всі питання не ставали правильними. Проте після детального дослідження команда розробників виявила, що після першого вивчення картки шанс відтворення інформації вдруге складав лише близько 87%. Якщо ж користувачам необхідно було декілька спроб для вивчення матеріалу, ймовірність повторного відтворення падала до 70%.

Недостатня ефективність існуючого алгоритму спричинила перехід до навчання з використанням розподілених повторень, а згодом до використання машинного навчання. Версія застосунку на сьогодні вираховує ймовірність забування кожної картки на основі минулих відповідей користувача та часу, що пройшов між повтореннями матеріалу.

Quizlet фокусується переважно на вивченні матеріалу з використанням флеш-карток. У застосунку можна створювати набори карток, ділитися ними з окремими користувачами, в межах створеного користувачем класу або публічно для всіх користувачів. Нещодавно було додано функціонал зі створення робочих зошитів з тестами [9].

Перевагами застосунку Quizlet є алгоритм засвоєння інформації, що враховує індивідуальні особливості навчання користувача, а також наявність

автоматичного перекладу карток. Quizlet має багатий функціонал, проте UX не завжди є зручним: додавання нових карток вимагає багато дій зі сторони користувача, що значно сповільнює користування застосунком. Окрім цього, велика частина функціоналу (додавання форматованого тексту, користування робочими зошитами для проходження тестів, тощо) доступна лише користувачам з платною підпискою.

2.2 Duolingo

Duolingo є популярною мобільною та веб-платформою для вивчення іноземних мов. Основною концепцією платформи є легке навчання з елементами гейміфікації, при яких користувач отримує віртуальні нагороди за виконання задач, запропонованих застосунком. Задачі мають динамічну форму: доповнення речення, вибір правильного варіанту з запропонованих або ж написання речення чи перекладу після прослуховування тексту [10].

Перша версія застосунку Duolingo використовувала систему Лейтнера як основний алгоритм навчання [11]. Проте користувачі помітили, що при використанні даної системи прогрес вивчення не є достовірним відображенням їхніх справжніх знань, через що перед командою розробників постала задача розробити новий підхід. Версія застосунку на сьогоднішній день використовує алгоритм розподіленого повторення з використанням машинного навчання. Для прогнозування рівня запам’ятовування інформації учнями, розробники Duolingo запровадили новий алгоритм – регресію напіврозпаду, що поєднує психолінгвістичну модель людської пам’яті з сучасними техніками машинного навчання та є узагальненням системи Лейтнера.

Перевагами застосунку Duolingo є динамічні завдання, що підбираються для кожного студента окремо в залежності від прогресу, а також краще запам’ятовування матеріалу через задіяння не тільки зору, а ще й слуху для виконання завдань. Проте в якості недоліків можна виділити неможливість створити власні завдання та переглянути весь список можливих завдань для

швидкого повторення вивченого матеріалу. Даний застосунок є гарним вибором як допоміжний засіб для довготривалого вивчення іноземної мови.

2.3 Anki

Anki є застосунком з відкритим вихідним кодом для вивчення матеріалу з використанням флеш-карток [12]. Застосунок доступний в мобільній, десктопній та веб-версії. Флеш-картки в Anki є гнучкими та підтримують зображення, відео, а також різноманітні форматування тексту. Окрім цього, можливе написання власних плагінів під застосунок та використання існуючих.

У якості алгоритму навчання Anki використовує алгоритм розподіленого повторення SM-2, що був описаний у розділі 1.1.2 [13]. Початковий інтервал між повтореннями визначається довжиною в 1 день, згодом в 6 днів. Під час показу флеш-картки користувачі самі визначають, наскільки гарно вони засвоїли матеріал. Для цього можливе використання шкали з трьох рівнів (погано / не дуже / добре) або ж налаштування власної з більшою чи меншою кількістю рівнів.

Основними перевагами застосунку Anki є гнучкість та велика кількість конфігурацій та плагінів, доступних користувачеві. Проте у застосунку відсутній автоматичний переклад слів, а використаний алгоритм навчання має недоліки у засвоєнні матеріалу. Окрім цього, не можна пройти весь набір карток за один раз, потрібно чекати декілька днів, доки відповідно до алгоритму до навчальних сесій додаватимуться нові картки з набору.

2.4 Memrise

Memrise є навчальною мобільною та веб-платформою, що пропонує онлайн-курси для вивчення мов. Наразі на платформі доступний вибір серед 23 іноземних мов [14]. Подібно до Duolingo, Memrise теж використовує гейміфікацію для більш динамічного досвіду навчання та підвищення мотивації користувачів. Memrise робить акцент на вправи з аудіювання, читання та письма для отримання знань, всі курси є послідовними та розділені за темами. Окрім професійних викладачів, курси можуть завантажувати також і звичайні користувачі.

Для вивчення матеріалу користувачами Memrise використовує розподілене повторення з підрахунком ймовірності забування для кожної одиниці матеріалу [15]. У разі правильної відповіді користувача картка буде повторена через довший період часу, якщо ж відповідь була дана неправильно – прогрес картки зникає і вона буде повторена знову через найменший проміжок часу. Застосунок використовує наступні інтервали: 4 години, 12 годин, 24 години, 6 днів, 12 днів, 48 днів, 96 днів, 6 місяців, тобто кожний наступний інтервал буде приблизно в 2 рази більший за попередній.

Перевагою застосунку Memrise є послідовність курсів, перехід від базових речей до більш складних. Навчання є динамічним, присутній великий набір курсів для вивчення різноманітних мов. Проте одним з недоліків застосунку є недостатня кількість конфігурацій та можливостей змінити вигляд картки. До карток можна додати лише звичайний неформатований текст, картки не підтримують зображення, відео, заголовки, тощо.

У таблиці 2.1 наведено порівняння основного функціоналу описаних вище застосунків.

Таблиця 2.1 – Порівняльна таблиця функціоналу застосунків

Назва	**Наявність динамічних завдань**	**Створення власних карток**	**Наявність плагінів**	**Вивчення карток за темами**
Quizlet	Ні	Так	Ні	Ні
Duolingo	Так	Ні	Ні	Так
Anki	Ні	Так	Так	Ні
Memrise	Так	Так	Ні	Так

РОЗДІЛ 3. ОГЛЯД ВИКОРИСТАНИХ ТЕХНОЛОГІЙ

3.1 Серверна частина

Для написання серверної частини використовувалися технології, що вже багато років застосовуються для імплементації надійних HTTP серверів з великою кількістю запитів від клієнтів.

3.1.1 Java

Java є поширеною об’єктно-орієнтовною мовою програмування, що була вперше випущена у травні 1995 року. Java має велику екосистему, підтримує кросплатформеність і багатопотоковість та часто застосовується для написання клієнт-серверних застосунків.

Виділяють такі чотири платформи мови програмування Java [16]:

- Java SE – Java Standard Edition, що містить базові об’єкти та типи мови програмування, а також високорівневі класи для мережевої взаємодії, конфігурації безпеки застосунку, взаємодії з базою даних тощо;
- Java ME – Java Micro Edition, що була спеціально розроблена під вбудовані системи і мобільні пристрої з обмеженими ресурсами. Java ME надає API і зоптимізовану невелику віртуальну машину для запуску Java застосунків і використовується здебільшого в мікроконтролерах, датчиках, мобільних телефонах, телевізійних приставках тощо;
- Java EE – Java Enterprise Edition (наразі відома як Jakarta EE), що побудована на основі Java SE та надає середовище виконання і програмні інтерфейси для імплементації багаторівневих масштабованих систем, основними вимогами до яких є надійність. Java EE найбільш відома завдяки своїй специфікації сервлетів, на основі яких було побудовано такі сервери як Tomcat [17] та Jetty [18], специфікації Java Persistance API, що є ORM стандартом, а також специфікації обміну повідомленнями Java Message Service, що підтримує асинхронну, надійну та слабкозв’язану передачу повідомлень [19];

- JavaFX, що є кросплатформеним фреймворком для створення клієнтських застосунків для мобільних, десктопних та вбудованих систем.

Для написання даної кваліфікаційної роботи було обрано Java SE. Платформа Java SE включає в себе наступні основні компоненти [20]:

- JRE – Java Runtime Environment, що надає бібліотеки, віртуальну машину Java та інші компоненти, необхідні для запуску програм, написаних мовою програмування Java;
- JDK – Java Development Kit, що містить JRE та додаткові програми, необхідні розробникам, такі як компілятори та відладчики.

Мову програмування Java було обрано через підтримку багатопотоковості, що необхідна для обробки запитів великої кількості користувачів одночасно, а також розвиненої екосистеми, що містить велику кількість фреймворків для написання надійних HTTP серверів та налаштування взаємодії з базами даних.

3.1.2 Spring Boot

Spring Boot є фреймворком з відкритим вихідним кодом для розробки застосунків на мовах Java та Kotlin, що базується на платформі Spring Framework [21]. Даний фреймворк значно полегшує налаштування залежностей застосунку, надаючи функціонал автоконфігурації для багатьох компонентів та бібліотек, пов’язаних з базами даних, безпекою, обміном повідомленнями, логуванням, тощо.

В основі Spring Boot лежить принцип ін’єкції залежностей, який дозволяє об’єктам визначати власні залежності та ін’єктувати їх згодом з використанням Spring контейнера. Даний підхід забезпечує модульність та слабку зв’язність компонентів застосунку, дозволяючи перевикористовувати компоненти та легко додавати нові, а також значно полегшує тестування [22]. Компоненти, що управляються Spring контейнером, називаються бінами.

Spring Boot має інтеграцію з великою кількістю технологій Java EE, таких як Java Persistence API, полегшуючи конфігурацію рівня доступу до даних та визначення таблиць завдяки автоконфігурації, Java Messaging Services,

конфігуруючи сервіси відправки та отримання повідомлень, сервлетами, дозволяючи написаним на Spring Boot застосункам запускатися на серверах Tomcat та Jetty. Окрім цього, фреймворк надає потужні засоби HTTP взаємодії, такі як RestController і RequestMapping анотації для визначення класів, відповідальних за прийом повідомлень та відправку відповіді, ExceptionHandler анотацію для визначення власного автоматичного обробника помилок, Valid анотацію для валідації запиту користувача, тощо.

Spring Boot було використано у даній роботі через потужні засоби, які фреймворк надає для написання API, заснованого на архітектурі REST, а також функціонал автоконфігурації взаємодії з базою даних.

3.1.3 Spring Security та JWT

Spring Security є фреймворком, що забезпечує автентифікацію, авторизацію, керування сесіями та захист від типових безпекових атак на застосунки [23].

Автентифікація є процесом підтвердження ідентифікаційних даних користувача, щоб переконатися, що особа, яка розпочала сесію, дійсно є тою, за кого себе видає. Spring Security надає такі основні механізми автентифікації:

- автентифікація з використанням імені користувача та пароля. Фреймворк підтримує автентифікацію на основі форм, надаючи вбудовану підтримку для надсилання форм входу користувачеві, перевірки введених облікових даних та керування сесіями, а також базову автентифікацію, при якій облікові дані користувача кодуються та надсилаються у складі HTTP заголовків запиту;
- OAuth/OIDC автентифікація. Spring Security має інтеграцію з OAuth 2.0 і OpenID Connect для автентифікації з використанням сторонніх постачальників ідентифікаційної інформації. За допомогою цього автентифікація користувачів можлива за посередництвом таких платформ, як Google, Facebook, GitHub та інших;

- LDAP (Lightweight Directory Access Protocol) автентифікація, що використовується для автентифікації користувачів на LDAP сервері або в службі каталогів;
- remember-me автентифікація, або автентифікація постійного входу, що є механізмом запам’ятовування користувачів між сесіями без необхідності повторного введення облікових даних. Зазвичай такий підхід імплементується з використанням згенерованих сервером токенів, що зберігаються у файлах кукі користувача та передаються серверу у подальших запитах для перевірки.

Авторизація є процесом визначення доступів користувача після проходження автентифікації. Spring Security надає велику кількість засобів для налаштування авторизації, таких як конфігурацію ролей та рівнів доступу, моделювання на рівні HTTP запиту відповідно до URL схеми, динамічну авторизацію на основі прописаної бізнес-логіки, а також гнучку обробку помилки при несанкціонованому доступі.

Для аутентифікації в застосунку було обрано автентифікацію постійного входу з використанням JSON Web Tokens (JWT).

JSON Web Token є відкритим стандартом RFC 7519 для безпечної передачі інформації між сторонами у форматі JSON [24]. JWT складається з таких трьох частин [25]:

- заголовок, в якому вказується алгоритм підписання токену і тип токену, що зазвичай відмічається як “JWT”;
- корисне навантаження, що містить твердження про користувача та додаткові дані;
- підпис, що створюється шляхом поєднання закодованого заголовка, закодованого корисного навантаження та секретного ключа за допомогою зазначеного алгоритму. В якості алгоритму найчастіше використовують ECDSA (Elliptic Curve Digital Signature Algorithm) або RSA (Rivest-Shamir-Adleman).

Особливостями JWT є його надійність завдяки використанню цифрових підписів, компактність, що дозволяє передавати менше даних мережею, та відсутність необхідності зберігати будь-які дані про сесії клієнтів на стороні сервера, що робить його гарним вибором для імплементації API з REST архітектурою.

3.1.4 PostgreSQL

PostgreSQL є об'єктно-реляційною СУБД з відкритим вихідним кодом, що початково була розроблена на факультеті комп'ютерних наук Каліфорнійського університету в Берклі. PostgreSQL підтримує складні запити, зовнішні ключі, транзакційну цілісність, багатоверсійний контроль паралельності, а також дозволяє розширення функціоналу користувачем за допомогою додавання нових типів даних, функцій, операторів, тощо [26].

PostgreSQL має широку систему типів, адже підтримує нумеричні типи даних (цілі, числа довільної точності, типи з плаваючою комою та серійні типи), грошові типи даних (тип money довжиною 8 байтів), символьні, бінарні, типи дати та часу, інтервальні, логічні, перелічувані та багато інших. Окрім цього, PostgreSQL надає транзакційність з підтримкою ACID, забезпечує високу продуктивність завдяки оптимізації запитів, паралельному виконанню, кешуванню та іншим технікам, а також містить вбудовані можливості повнотекстового пошуку.

Для виконання кваліфікаційної роботи було обрано PostgreSQL через високу продуктивність, легкість у налаштуванні та гнучкий функціонал, що підходить для написання більшості застосунків.

3.1.5 Google Cloud Translation API

Google Cloud Translation API є сервісом для перекладу тексту, що використовує технологію нейронного машинного перекладу від Google та підтримує понад сотню мов [27].

Google Cloud Translation API надає можливість користування такими чотирма видами сервісів:

- Translation Hub (дозволяє переклад на 135 мов, надаючи легкий інтуїтивний користувацький інтерфейс. Translation Hub дає змогу підприємствам налаштовувати та керувати робочим навантаженням перекладу за раніше недосяжного масштабу та вартості);
- AutoML Translation (розробники, перекладачі та експерти з локалізації, які знають предметну область, можуть створювати власні моделі перекладу з використанням технології AutoML без написання коду);
- Translation API (Translation API Basic використовує технологію нейронного машинного перекладу Google для миттєвого перекладу текстів понад сотнею мов);
- Media Translation API (забезпечує аудіопереклад у реальному часі безпосередньо до переданого контенту із підвищеною точністю та спрощеною інтеграцією).

Плата за використання API нараховується відповідно до кількості символів, включену в кожен запит. Google Cloud Translation API, окрім простого тексту, також дозволяє перекладати HTML сторінки.

Google Cloud Translation API було обрано для виконання роботи через великий перелік підтримуваних мов, можливість перекладати текст з врахуванням контексту та навченість мовної моделі, що використовується сервісом, розпізнавати спеціалізовані та технічні терміни.

3.2 Клієнтська частина

Для написання клієнтської частини використовувалися бібліотеки з розвиненою екосистемою та високою швидкістю і продуктивністю виконання.

3.2.1 JavaScript

JavaScript є інтерпретованою або компільованою “на льоту” мовою програмування, що є реалізацією стандарту ECMAScript та використовується для

створення інтерактивного динамічного веб-контенту. JavaScript підтримує об’єктно-орієнтовану, імперативну і декларативну парадигми програмування та є скриптовою мовою з динамічною типізацією [28].

JavaScript використовує модель паралелізму, засновану на циклі подій. Як тільки на сторінці відбувається якась подія (натискання на кнопку, надходження відповіді від сервера), вона додається у чергу. Двигун JavaScript підтримує стек викликів, який відстежує поточні виконувані функції. Цикл подій постійно перевіряє стек викликів та чергу подій. Якщо стек викликів порожній, цикл обирає першу подію з черги та додає відповідну їй функцію зворотнього виклику (callback) в стек подій. Завдяки даній моделі JavaScript може обробляти кілька операцій одночасно та виконувати трудомісткі або тривалі за часом операції, не блокуючи основний потік.

JavaScript було обрано для написання роботи через високу швидкість виконання, поширеність технології на ринку, а також розширені можливості взаємодії з DOM.

3.2.2 React

React є JavaScript бібліотекою з відкритим вихідним кодом для побудови користувацьких інтерфейсів, що здобула значну популярність в останні роки [29].

React дотримується архітектури на основі компонентів, усі елементи інтерфейсу розробляються як окремі незалежні компоненти, що можуть бути перевикористані. Модульний підхід значно полегшує розробку з використанням бібліотеки та покращує масштабованість. Бізнес-логіку можна розділяти на комоненти на основі принципу єдиної відповідальності, логіки побудови дизайну та інших підходів.

Окрім цього, React використовує віртуальний DOM [30], що є репрезентацією користувацького інтерфейсу, яка зберігається в пам’яті і синхронізується зі справжнім DOM з використанням бібліотек. Дана абстракція дозволяє React бібліотеці ефективно оновлювати та показувати лише необхідні компоненти, в яких відбулися зміни, замість оновлення всього DOM.

У бібліотеці React використовується односпрямований потік даних від батьківських компонентів до дочірніх [31]. Дана концепція значно полегшує керування станом застосунку та подальшу розробку.

Для виконання кваліфікаційної роботи було обрано React через ефективну роботу з DOM, що забезпечує високу продуктивність, модульний підхід до розробки, а також велику екосистему бібліотек для React застосунків.

3.2.3 Redux та Redux-Saga

Redux є бібліотекою, що надає контейнер передбачуваного стану для програм на мові JavaScript [32]. Таким чином стан застосунку зберігається в об’єкті-сховищі і є централізованим, а сховище приймається як єдине джерело правдивих даних виконуваної програми. Це значно полегшує тестування та відлагоджування застосунків та забезпечує узгодженість даних.

Redux складається з таких основних концепцій [33]:

- стан, що відображає стан застосунку в певний момент часу та доступний тільки для читання. Єдиним способом оновити стан є використання дій;
- редуктори, що є чистими функціями, які приймають поточний стан та викликану дію і повертають новий стан. На основі дій редуктори визначають, як саме оновлювати поточний стан;
- дії, що репрезентують події в застосунку, які викликали зміну стану. Дії є звичайними об’єктами мови JavaScript, що містять поле типу (type), а також опціонально корисне навантаження (payload).

Redux дозволяє використання проміжного програмного забезпечення (middleware) між моментом створення дії та досягненням дії редуктора. Проміжне програмне забезпечення може перехоплювати та змінювати дії, виконувати асинхронні операції, такі як відправлення HTTP запитів, тощо. У застосунку через гарну інтеграцію з Redux було обрано використати Redux-Saga.

Redux-Saga є бібліотекою проміжного програмного забезпечення для Redux, націлена на полегшення керування побічними ефектами застосунків, такими як асинхронними викликами чи доступами до кешу браузера, та покращити обробку

помилок при виконанні таких операцій [34]. Побічні ефекти оброблюються в окремомих потоках, що можуть бути розпочаті, призупинені та відмінені за допомогою викликів з головного застосунку.

У Redux-Saga виділяють такі основні концепції:

- саги, що є функціями-генераторами, які інкапсулюють логіку побічних ефектів;
- ефекти, що є об’єктами, які повертають саги. Ефекти містять інформацію для інтерпретації проміжним програмним забезпеченням, по суті вони є інструкціями для виконання певної дії, наприклад, виклику API.

Окрім цього, Redux-Saga надає способи тестування саг та ефектів в ізоляції.

3.2.4 CSS

CSS (Cascading Style Sheets) є мовою таблиць стилів, що використовується для опису вигляду HTML документів. CSS дозволяє відділяти структуру сторінки (вміст) від стилів (презентації), що дозволяє прописувати гнучку логіку відображення стилів в залежності від змісту [35]. CSS стандартизовано для веб-браузерів відповідно до специфікацій W3C.

Виділяють такі основні концепції мови таблиць стилів CSS:

- селектори, що визначають певний елемент або групу елементів, для яких треба застосувати стилі;
- оголошення стилів, що складаються з пари властивість-значення та визначають візуальні властивості (кольори, відступи, розмір, тощо);
- медіа-запити, що дозволяють застосовувати стилі в залежності від розміру екрану користувача;
- анімації та переходи, що дозволяють задавати ефекти руху та трансформації об’єктів на сторінці та визначати швидкість зміни властивостей об’єктів.

CSS широко використовується для розробки веб-застосунків, підтримується усіма сучасними браузерами та забезпечує ефективну стилізацію веб-сторінок, тому дана мова таблиць стилів була обрана для виконання кваліфікаційної роботи.

РОЗДІЛ 4. РЕАЛІЗАЦІЯ СИСТЕМИ

4.1 Призначення застосунку

Призначенням веб-застосунку “Vocab” є полегшення процесу вивчення іноземних мов з використанням флеш-карток та алгоритму навчання з адаптивною складністю. Застосунок повинен надавати можливості додавання та управління наборами флеш-карток, словниками та книжками, а також засоби для вивчення нового матеріалу.

4.2 Вимоги до системи

Система повинна підтримувати дві мови – українську та англійську, що можуть бути змінені в налаштуваннях.

Можна виділити такі основні функції системи:

- реєстрація користувача та вхід в систему. Система повинна забезпечувати можливість введення персональних даних для створення нового користувача, а також автентифікації існуючого користувача для подальшого користування застосунком;
- управління наборами карток. Користувач повинен мати змогу створювати нові набори карток, надавати спільний доступ до набору іншим зареєстрованим в системі користувачам, переглядати доступні набори та редагувати і видаляти власні набори;
- управління флеш-картками в наборі. Система повинна забезпечувати можливість створення нових карток в наборі, редагування існуючих, а також видалення карток;
- управління словниками. Користувач повинен мати можливість створювати нові словники, надавати спільний доступ іншим зареєстрованим в системі користувачам, переглядати доступні словники та редагувати і видаляти існуючі словники;
- управління книжками. У системі повинні бути присутні функції завантаження нових книжок у форматі epub, редагування імені та автора

завантажених книжок, видалення книжок, а також читання книжок з можливістю додавання виділених слів у власні словники;

- вивчення матеріалу. Користувачеві повинні бути доступні функції перегляду існуючих флеш-карток з можливістю оцінити правильність власної відповіді, а також функція перевірки правильності написання слова;
- налаштування мови та параметрів навчання. Користувач повинен мати змогу змінити мову інтерфейсу застосунку в налаштуваннях, а також визначити кількість карток, що буде показана у кожному циклі навчання.

Більш детальний опис можливостей зареєстрованого користувача представлено у діаграмі прецедентів, наведеній у додатку А.

4.3 Архітектура системи

Для розробки застосунку було використано клієнт-серверну архітектуру, адже вона має велику кількість переваг. Даний тип архітектури підтримує комунікацію декількох клієнтів з сервером у мережі, відділяє бізнес-логіку від представлення, дозволяє перевикористовувати код та легко розширювати API [36].

Клієнтом у системі виступає застосунок, написаний на мові JavaScript з використанням бібліотеки React, що містить компоненти користувацького інтерфейсу, правила їх відображення, а також правила накладання стилів.

Сервером є застосунок, написаний на мові Java з використанням фреймворку Spring Boot, що відповідає за логіку управління сутностями (користувачами, картками, словниками, книжками, тощо) та взаємодіє з базою даних. API сервера відповідає принципам REST архітектури, що допомагає клієнту та серверу розвиватися незалежно один від одного, забезпечує універсальність та незалежність клієнта і сервера. Виділяють такі основні принципи REST архітектури, що були реалізовані в застосунку [37]:

- клієнт-серверна архітектура: клієнт, бажаючи виконати роботу, надсилає запит серверу. Сервер або відхиляє запит, або обробляє його та надсилає відповідь клієнту;

- відсутність стану: кожен запит клієнта до сервера повинен мати достатньо інформації для обробки запиту. З точки зору сервера, клієнта не існує, коли він не робить запит. Сервер не повинен зберігати інформацію про сесію між запитами;
- кешування: відповідно до інструкцій сервера, клієнти можуть кешувати дані для подальшого використання закешованої копії;
- багатошарова система: комунікація клієнта та сервера може проходити з використанням посередників, наприклад проксі, проте клієнт не повинен знати, з ким саме відбувається процес комунікації – безпосередньо з сервером чи з одним з посередників;
- уніфікований інтерфейс: URI має визначати ресурс, з яким виконується певна дія. Маніпуляція ж ресурсом має відбуватися через його репрезентацію. Окрім цього, накладається обмеження на HTTP повідомлення – вони мають містити всю необхідну інформацію для коректної інтерпретації клієнтом (наприклад, тип переданого контенту, такий як JSON чи мультимедіа). Двигуном застосунку відповідно до принципу уніфікованого інтерфейсу є гіпермедіа, що дозволяє клієнту визначати всі можливі подальші дії завдяки включеним до відповіді сервера гіперпосиланням.

Загальна архітектура системи слідує шаблону багатошарової архітектури, в якому кожен рівень має свою чітко визначену сферу відповідальності [38]. У застосунку було виділено наступні рівні:

- презентаційний рівень, що відповідає за представлення інформації користувачу та обробку взаємодії користувача з системою;
- рівень бізнес-логіки, що виконує всі операції, пов’язані з бізнес-правилами та функціоналом системи;
- перзистентний рівень, що відповідає за роботу з даними за допомогою запитів;
- рівень бази даних, що відповідає за зберігання, управління та доступ до даних системи.

На рисунку 4.3.1 представлено схему архітектури системи.

Рисунок 4.3.1 – Архітектура системи “Vocab”

4.4 Організація інформаційної бази

Для проєктування бази даних було виділено основні сутності застосунку: користувач, набір карток, словник, книга, картка, дозвіл на набір карток та дозвіл на словник. На основі виділених сутностей було створено таблиці з основними полями. Згодом було створено проміжні таблиці для реалізації наявних відношень між сутностями.

Детальна схема бази даних з відношеннями між таблицями показана на рисунку 4.4.1. У таблиці 4.4.1 наведено опис кожної таблиці у базі даних.

Рисунок 4.4.1 – Схема бази даних

Таблиця 4.4.1 – Опис структури бази даних

Назва таблиці	**Опис**
users	Таблиця, в якій зберігаються дані про зареєстрованих користувачів системи
folders	Таблиця з даними про додані набори карток
dictionaries	Таблиця з даними про додані словники
cards	Таблиця з даними флеш-карток
books	Таблиця з даними про завантажені книги. Фізичні файлі з книгами зберігаються у файловій системі сервера, база даних містить лише інформацію про шлях до файлу
folder_permissions	Таблиця з доступами користувачів до наборів з картками, необхідна для забезпечення функціоналу спільного доступу
dictionary_permissions	Таблиця з доступами користувачів до словників, необхідна для забезпечення функціоналу спільного доступу
folders_cards	Таблиця, що містить відповідність флеш-карток наборам, в яких користувач створив дані картки
dictionaries_cards	Таблиця, що містить відповідність флеш-карток зі словами словникам, до яких ці слова було додано користувачем

У наступних таблицях подано назву, тип та детальний опис полів кожної таблиці в базі даних. Кожна таблиця, окрім допоміжних таблиць для імплементації відношень між сутностями, містить поле унікального ідентифікатору, а також час створення та оновлення поля.

Таблиця 4.4.2 – Таблиця users

Назва поля	**Тип поля**	**Опис**
id	uuid	Первинний ключ, ідентифікатор користувача
created_at	timestamp	Час реєстрації користувача в системі
updated_at	timestamp	Час останнього оновлення даних про користувача
email	varchar	Електронна адреса користувача
first_name	varchar	Ім’я користувача
last_name	varchar	Прізвище користувача
password	varchar	Закодований пароль користувача

Таблиця 4.4.3 – Таблиця folders

Назва поля	**Тип поля**	**Опис**
id	uuid	Первинний ключ, ідентифікатор набору карток
created_at	timestamp	Час створення набору карток
updated_at	timestamp	Час останнього оновлення набору карток
name	varchar	Назва набору карток, введена користувачем

Таблиця 4.4.4 – Таблиця dictionaries

Назва поля	**Тип поля**	**Опис**
id	uuid	Первинний ключ, ідентифікатор словника
created_at	timestamp	Час створення словника
updated_at	timestamp	Час останнього оновлення словника
name	varchar	Назва словника, введена користувачем

Таблиця 4.4.5 – Таблиця cards

Назва поля	**Тип поля**	**Опис**
id	uuid	Первинний ключ, ідентифікатор картки
created_at	timestamp	Час створення картки
updated_at	timestamp	Час останнього оновлення картки
front	text	Текст передньої сторони картки
back	text	Текст зворотньої сторони картки
rate	integer	Поточний рівень вивчення картки, що характеризує, наскільки гарно учень засвоїв дану одиницю матеріалу
max_rate	max_integer	Максимальний рівень вивчення картки, при досягненні якого матеріал вважатиметься вивченим

Таблиця 4.4.6 – Таблиця books

Назва поля	**Тип поля**	**Опис**
id	uuid	Первинний ключ, ідентифікатор книги
created_at	timestamp	Час завантаження книги в систему
updated_at	timestamp	Час останнього оновлення книги
name	varchar	Назва книги, введена користувачем
file_name	varchar	Назва файлу у файловій системі сервера, що відповідає даній книзі
author	varchar	Автор книги, введений користувачем
user_id	uuid	Ідентифікатор користувача, який завантажив дану книгу

Таблиця 4.4.7 – Таблиця folder_permissions

Назва поля	**Тип поля**	**Опис**
id	uuid	Первинний ключ, ідентифікатор доступу
created_at	timestamp	Час відкриття доступу до набору карток користувачеві
updated_at	timestamp	Час останнього оновлення даного доступу
type	smallint	Число, що репрезентує вид доступу (читання чи редагування)
folder_id	uuid	Ідентифікатор набору карток, до якого було відкрито доступ
user_id	uuid	Ідентифікатор користувача, якому надано доступ до набору карток

Таблиця 4.4.8 – Таблиця dictionary_permissions

Назва поля	**Тип поля**	**Опис**
id	uuid	Первинний ключ, ідентифікатор доступу
created_at	timestamp	Час відкриття доступу до словника користувачеві
updated_at	timestamp	Час останнього оновлення даного доступу
type	smallint	Число, що репрезентує вид доступу (читання чи редагування)
dictionary_id	uuid	Ідентифікатор словника, до якого було відкрито доступ
user_id	uuid	Ідентифікатор користувача, якому надано доступ до словника

Таблиця 4.4.9 – Таблиця folders_cards

Назва поля	**Тип поля**	**Опис**
folder_id	uuid	Ідентифікатор набору карток
card_at	uuid	Ідентифікатор картки, що пов’язана з набором карток

Таблиця 4.4.10 – Таблиця dictionaries_cards

Назва поля	**Тип поля**	**Опис**
dictionary_id	uuid	Ідентифікатор словника
card_at	uuid	Ідентифікатор картки, що пов’язана зі словником

4.5 Реалізація сервера

4.5.1 Автентифікація та авторизація

Автентифікація та авторизація у застосунку імплементована з використанням remember-me підходу, описаного в розділі 3.1.3. При успішній автентифікації та авторизації користувача на сервері у відповідь формується JWT токен, який згодом передається клієнтом у складі заголовків кожного запиту. Завдяки використанню даного підходу серверу не потрібно зберігати стан сесії для кожного клієнта. Це покращує продуктивність застосунку, а також надає можливості масштабування при збільшенні кількості користувачів.

Доступ до ресурсів системи дозволений лише автентифікованим користувачам. Єдине виключення було зроблено для безпосередньо входу в систему та реєстрації, адже до них повинні мати доступ всі користувачі без винятку. На рисунку 4.5.1.1 представлено безпекову конфігурацію на стороні сервера.

```java
/**
 * Configures HTTP security (cors, endpoints protection)
 */
@Bean
public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
    return http
            .cors().and() HttpSecurity
            .csrf().disable()
            .authorizeHttpRequests((requests) -> requests
                    .requestMatchers( ...patterns: "/", "/auth/**") AuthorizeHttpRequestsConfigurer<...>.AuthorizedUrl
                    .permitAll() AuthorizeHttpRequestsConfigurer<...>.AuthorizationManagerRequestMatcherRegistry
                    .anyRequest().authenticated()
            )
            .addFilterBefore(jwtRequestFilter, UsernamePasswordAuthenticationFilter.class)
            .sessionManagement() SessionManagementConfigurer<HttpSecurity>
            .sessionCreationPolicy(SessionCreationPolicy.STATELESS)
            .and() HttpSecurity
            .build();
}
```

Рисунок 4.5.1.1 – Безпекова конфігурація сервера

4.5.2 Взаємодія з базою даних

Для взаємодії з базою даних було використано класи репозиторіїв, які є частиною пакету org.springframework.data.jpa.repository. На рисунку 4.5.2.1 наведено приклад репозиторію для взаємодії з сутністю книжок, який містить як запит, написаний на мові HQL (Hibernate Query Language), так і запит, що генерується з ключових слів.

```java
public interface BookRepository extends JpaRepository<Book, UUID> {

    @Query("SELECT B FROM Book B " +
            "WHERE B.user.id = ?1 " +
            "AND LOWER(B.name) LIKE LOWER(concat(?2, '%')) " +
            "ORDER BY B.createdAt DESC")
    List<Book> getByUserAndName(UUID userId, String name);

    List<Book> getByUserIdOrderByCreatedAtDesc(UUID userId);
}
```

Рисунок 4.5.2.1 – Приклад репозиторію для сутності книжок

4.5.3 Обробка HTTP запитів

Обробка HTTP запитів на стороні сервера відбувається за допомогою контроллерів. Контроллери є класами, що помічені анотацією Controller або RestController та надають функціонал навігації запитів, серіалізації та десеріалізації повідомлень. На рисунку 4.5.3.1 наведено приклад контроллера для обробки запитів маніпуляції ресурсом книг.

```
@RestController
@RequestMapping("/books")
@RequiredArgsConstructor
@Slf4j
public class BookController {

    private final BookService bookService;

    private final FileStorageService storageService;

    @PostMapping
    public BookDto uploadBook(@AuthenticationPrincipal String principal,
                              @RequestParam String name,
                              @RequestParam String author,
                              @RequestParam MultipartFile file) {
        return bookService.saveBook(UUID.fromString(principal), name, author, file);
    }

    @GetMapping("/{id}")
    public BookDto getBook(@PathVariable UUID id) { return bookService.getBook(id); }

    @PutMapping("/{id}")
    public BookDto editBook(@PathVariable UUID id, @RequestBody BookPropertiesDto dto) {
        return bookService.editBook(id, dto);
    }

    @DeleteMapping("/{id}")
    public void deleteBook(@PathVariable UUID id) { bookService.deleteBook(id); }

    @GetMapping("/files/{filename:.+}")
    @ResponseBody
    public ResponseEntity<Resource> getFile(@PathVariable String filename) {
        log.info("File requested: " + filename);
        Resource file = storageService.load(filename);
        return ResponseEntity.ok()
                .header(HttpHeaders.CONTENT_DISPOSITION,
                        ...headerValues: "attachment; filename=\"" + file.getFilename() + "\"").body(file);
    }

}
```

Рисунок 4.5.3.1 – Приклад контроллера для ресурсу книг

Для налаштування повідомлень помилок при некоректних запитах від клієнта було імплементовано власний обробник помилок, що наслідує абстрактний клас ResponseEntityExceptionHandler та позначений анотацією ControllerAdvice. Код обробника помилок зображено на рисунку 4.5.3.2.

```
@ControllerAdvice
public class CustomExceptionHandler extends ResponseEntityExceptionHandler {

    private final DictionaryProperties dictionaryProperties;

    public CustomExceptionHandler(DictionaryProperties dictionaryProperties) {
        this.dictionaryProperties = dictionaryProperties;
    }

    @ExceptionHandler(EntityNotFoundException.class)
    public ResponseEntity<Object> handleEntityNotFound(EntityNotFoundException e) {
        return ResponseEntity.status(HttpStatus.NOT_FOUND).body(
                MessageFormat.format( pattern: "No such entity: {0}", e.getMessage())
        );
    }

    @ExceptionHandler(CredentialsException.class)
    public ResponseEntity<Object> handleLoginError(CredentialsException e) {
        return ResponseEntity.status(HttpStatus.CONFLICT).body(
                MessageFormat.format( pattern: "Invalid credentials: {0}", e.getMessage())
        );
    }
```

Рисунок 4.5.3.2 – Обробник помилок на стороні сервера

4.6 Реалізація клієнта

Для реалізації клієнта було створено функціональні React компоненти. Розбиття на компоненти виконувалося за логікою побудови дизайну: кожна сторінка є окремим компонентом, також в компоненти було винесено і менші структурні одиниці, такі як репрезентація наборів карток, словників, книжок, тощо.

Оновлення стану відбувається у редукторах. Дані стану передаються в компоненти за допомогою властивостей компонент. На рисунку 4.6.1 зображено редуктор для оновлення стану сторінки аутентифікації.

```
const authData = (state = initialState, action) => {
    switch (action.type) {
        case signUpRoutine.TRIGGER:
        case signInRoutine.TRIGGER:
            return {
                ...state,
                loading: true,
            }
        case signUpRoutine.SUCCESS:
        case signInRoutine.SUCCESS:
            return {
                loading: false,
                ...action.payload,
            }

        case signUpRoutine.FAILURE:
        case signInRoutine.FAILURE:
            return {
                ...state,
                loading: false,
            }
        default: {
            return state;
        }
    }
}
```

Рисунок 4.6.1 – редуктор для сторінки аутентифікації

РОЗДІЛ 5. РОБОТА З СИСТЕМОЮ “VOCAB”

Першою сторінкою, яку бачить неавтентифікований у системі користувач, є головна сторінка, зображена на рисунку 5.1. Вона містить опис основних переваг застосунку та посилання на вхід до системи і реєстрацію.

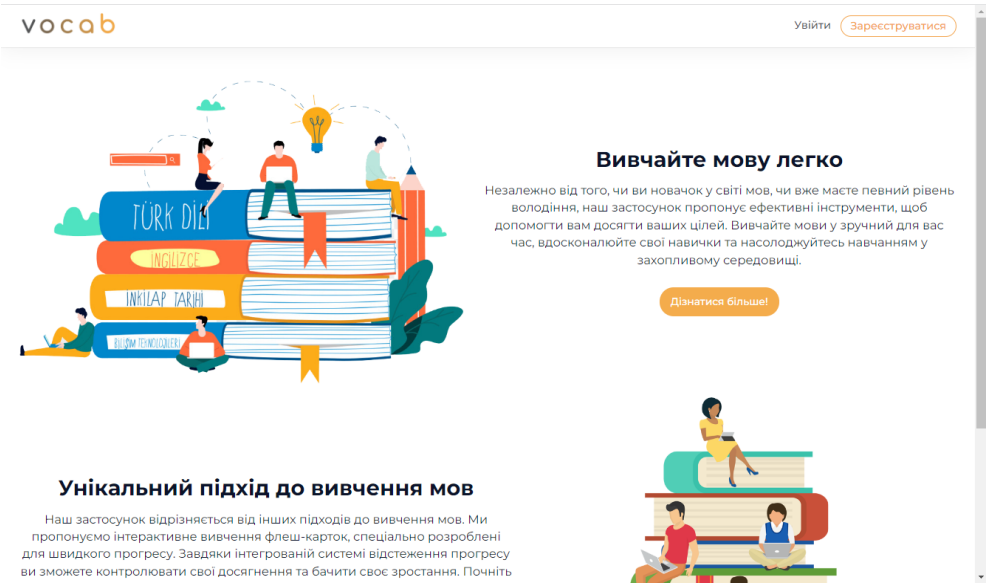

Рисунок 5.1 – Головна сторінка застосунку

Якщо у користувача ще немає облікового запису в системі, треба натиснути на кнопку “Зареєструватися” та ввести необхідні для реєстрації дані (ім’я, прізвище, електронну адресу та пароль). Сторінка реєстрації зображена на рисунку 5.2.

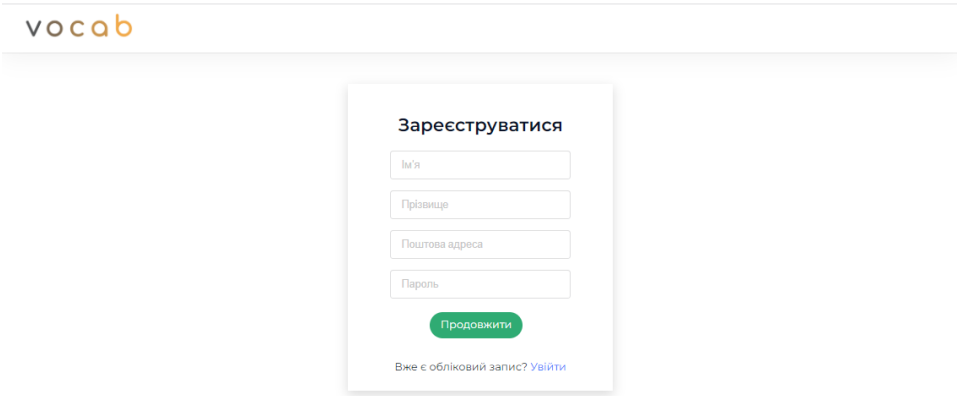

Рисунок 5.2 – Сторінка реєстрації користувача

Якщо у користувача вже є обліковий запис, на головній сторінці треба натиснути кнопку “Увійти” та ввести поштову адресу і пароль для входу в систему на сторінці входу, зображеній на рисунку 5.3.

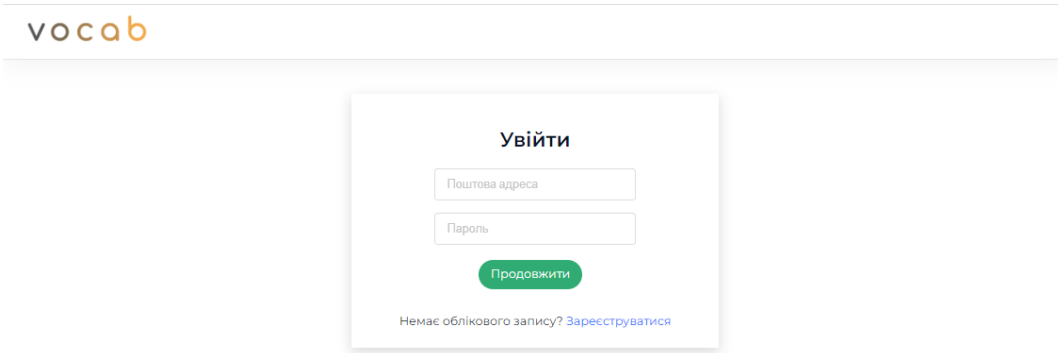

Рисунок 5.3 – Сторінка входу в систему

Після реєстрації або входу користувач перенаправляється на сторінку з наборами карток, зображеній на рисунку 5.4. На сторінці з наборами карток присутній пошук, показано власні набори користувача, а також набори, до котрих поточний користувач має доступ. Для створення нового набору треба натиснути на кнопку у правому нижньому кутку екрану, показану на рисунку 5.4.

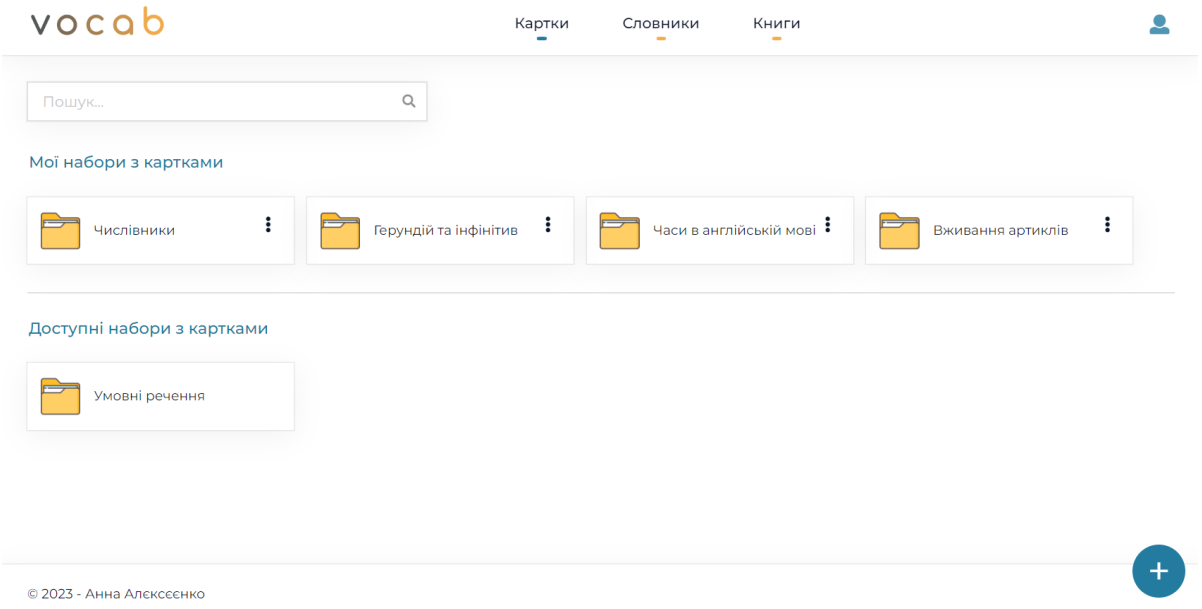

Рисунок 5.4 – Сторінка з наборами карток

Модальне вікно додавання набору карток, показане на рисунку 5.5, містить поле з назвою набору. Після введення назви стане активною кнопка збереження набору.

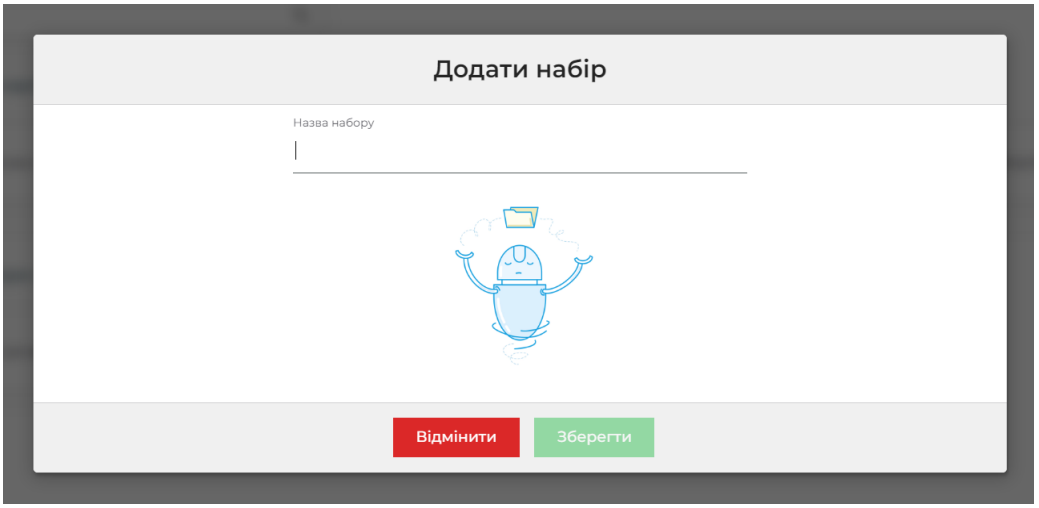

Рисунок 5.5 – Модальне вікно створення набору

Набори карток можна редагувати та видаляти, також до наборів можна надавати спільний доступ за допомогою меню, що присутнє для кожного набору та показане на рисунку 5.6.

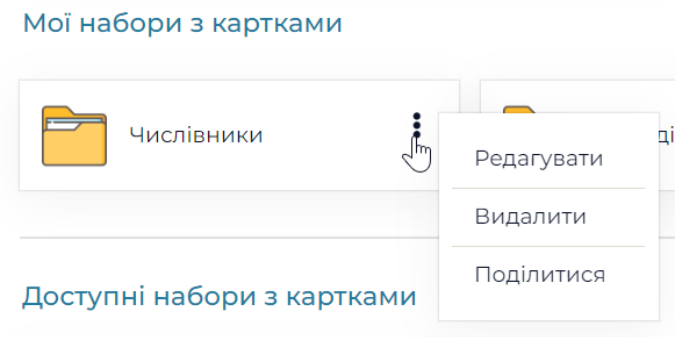

Рисунок 5.6 – Меню управління наборами карток

При натисканні на набір карток користувач переходить на сторінку з картками, зображену на рисунку 5.7. На даній сторінці доступний перегляд картки з подальшим редагуванням або видаленням, перехід в режим навчання, а також додавання нової картки. Для кожної картки відображено прогрес вивчення.

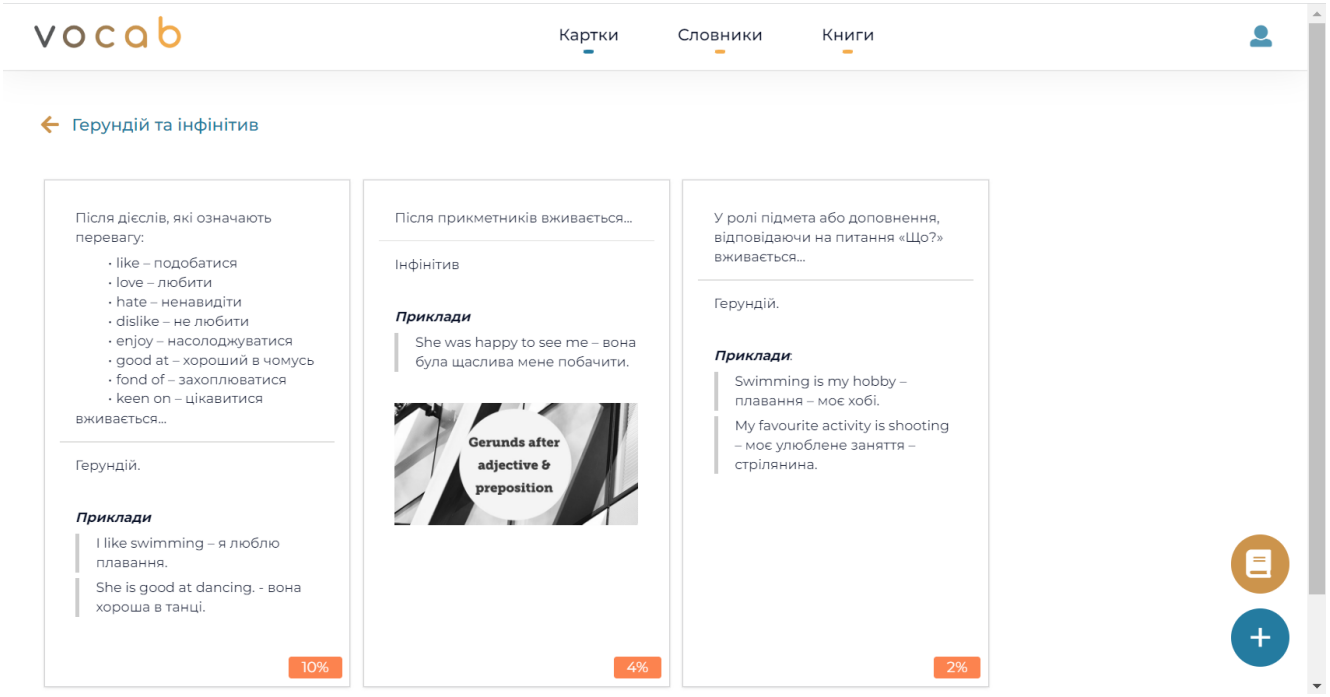

Рисунок 5.7 – Сторінка з картками

При натисканні на кнопку для додавання нової картки, що розташована в правому нижньому кутку та має синій колір (рисунок 5.7), відкривається модальне вікно для додавання картки, зображене на рисунку 5.8. У цьому вікні необхідно ввести передню та зворотню сторону картки, а також максимальну кількість правильних повторів, при якій картка вважатиметься вивченою.

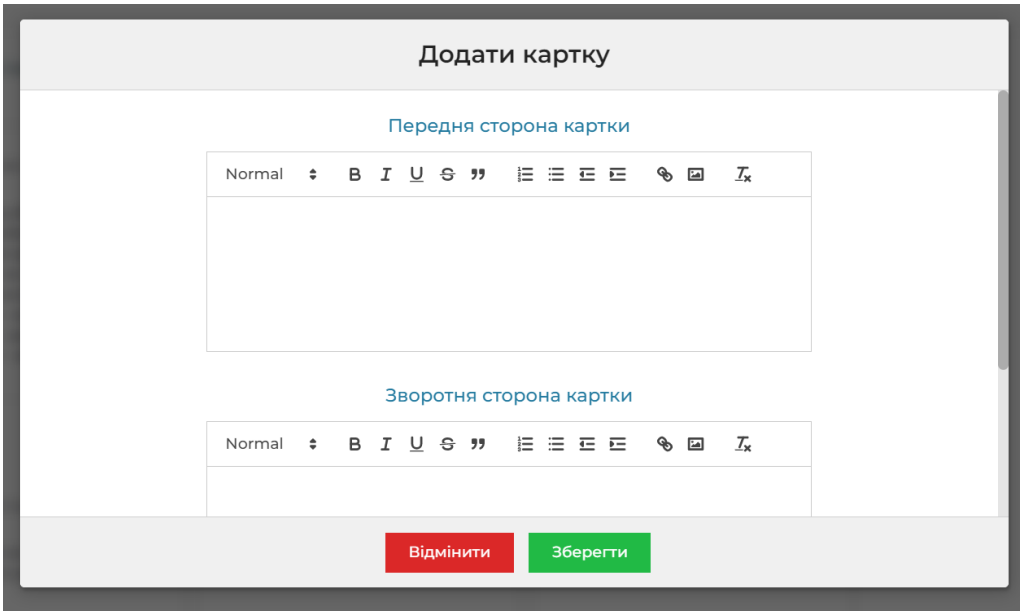

Рисунок 5.8 – Модальне вікно для додавання картки

При натисканні на конкретну картку користувач може перейти на сторінку управління карткою (рисунок 5.9), на якій будуть доступні опції редагування та видалення картки.

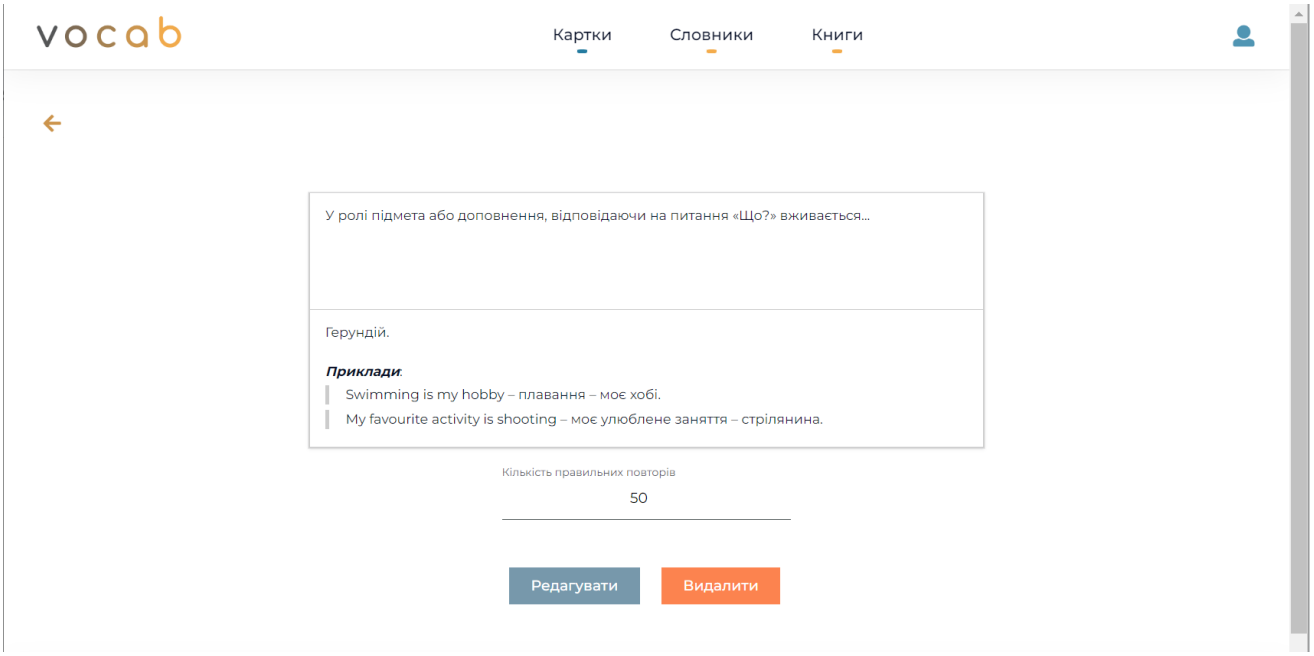

Рисунок 5.9 – Сторінка управління карткою

При натисканні на кнопку для переходу в режим навчання, що розташована в правому нижньому кутку та має помаранчевий колір (рисунок 5.7), відбувається переадресація на сторінку вивчення карток, зображену на рисунку 5.9. Користувачеві показується передня сторона картки та дається три опції:

- червона кнопка позначає, що картка ще не вивчена користувачем або погано запам’яталася. При натисканні на неї відсоток вивчення картки зменшиться та картка буде показана раніше за краще вивчені картки;
- зелена кнопка позначає, що користувач зміг пригадати вміст зворотньої сторони картки. Відсоток вивчення картки відповідно буде збільшено та картка буде показуватися після карток з меншим відсотком вивчення;
- синя кнопка необхідна, щоб перевернути картку. Даний функціонал використовується для самоперевірки або вивчення картки у перший раз.

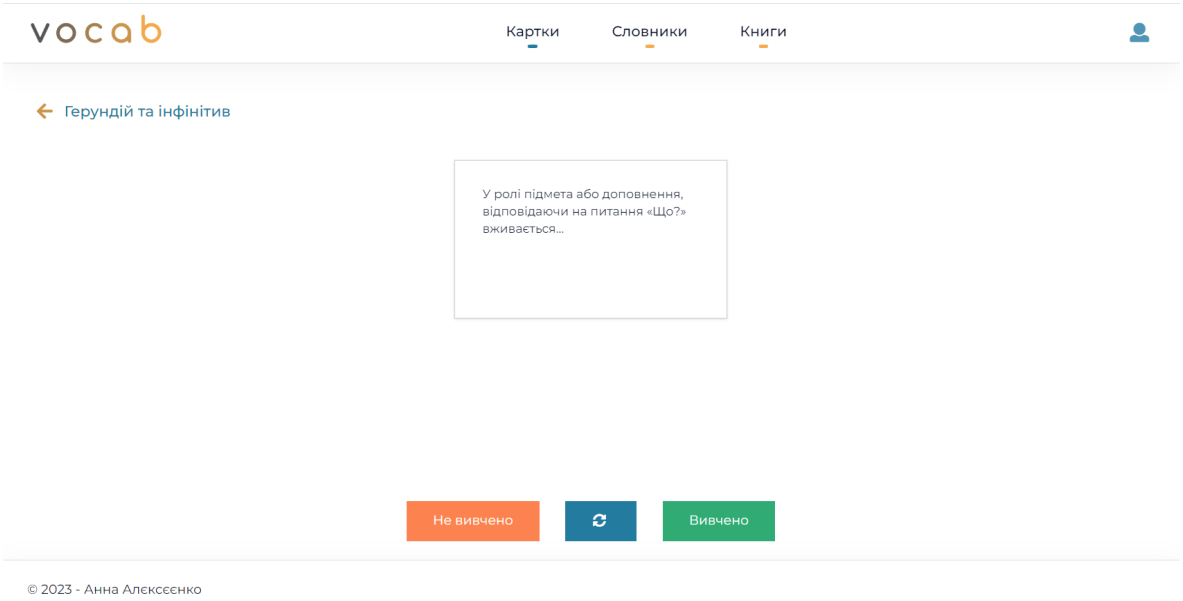

Рисунок 5.9 – Сторінка вивчення картки

На сторінці зі словниками (рисунок 5.10), на яку можна перейти, натиснувши відповідну кнопку в заголовку сторінки, відображаються власні словники користувача та словники, до яких користувачеві було надано доступ. Управління словниками відбувається за допомогою такого ж меню, як і для наборів карток.

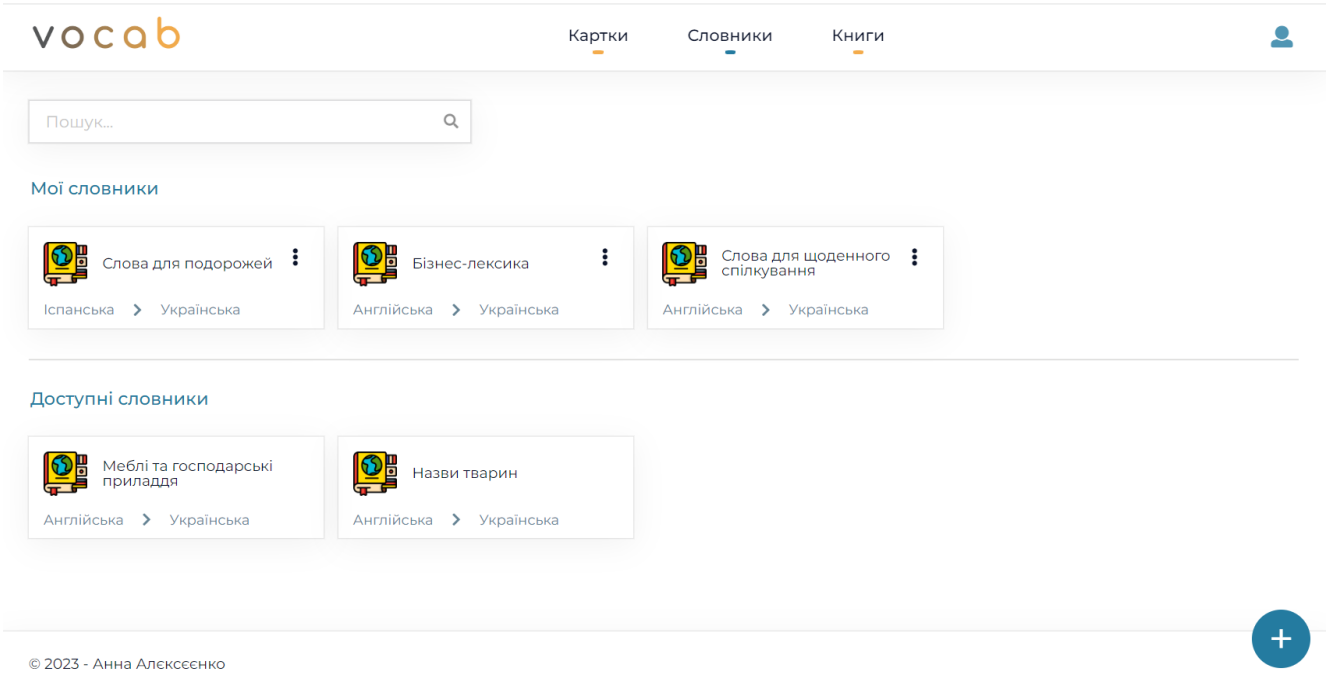

Рисунок 5.10 – Сторінка зі словниками

При натисканні на синю кнопку додавання словника у правому нижньому кутку відкриється модальне вікно (рисунок 5.11), в якому необхідно ввести назву словника, а також обрати мову оригіналу і мову перекладу.

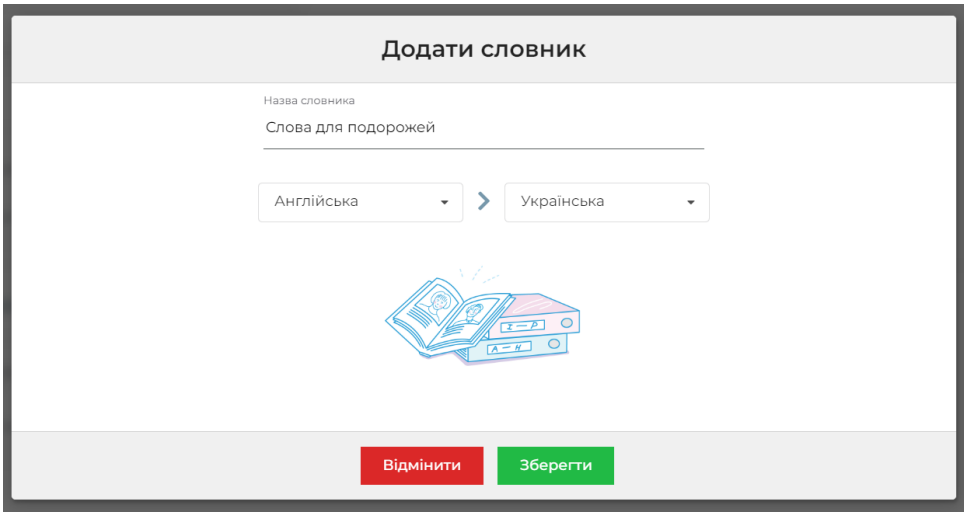

Рисунок 5.11 – Модальне вікно для додавання словника

При виборі словника зі списку на сторінці зі словниками користувач потрапляє на сторінку зі словами у даному словнику (рисунок 5.12). На даній сторінці можна додавати, редагувати та видаляти слова, а також розпочинати навчання.

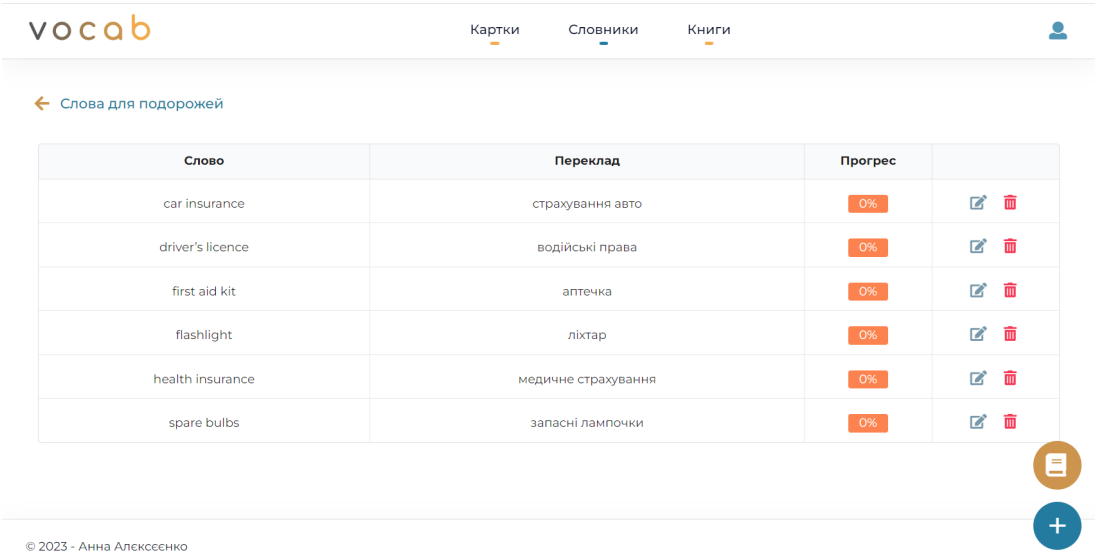

Рисунок 5.12 – Сторінка з словами у словнику

Для додавання нового слова у словник можна натиснути на синю кнопку в правому нижньому кутку (рисунок 5.12). Відкриється модальне вікно, зображене на рисунку 5.13, для введення слова, перекладу та максимальної кількості правильних повторів.

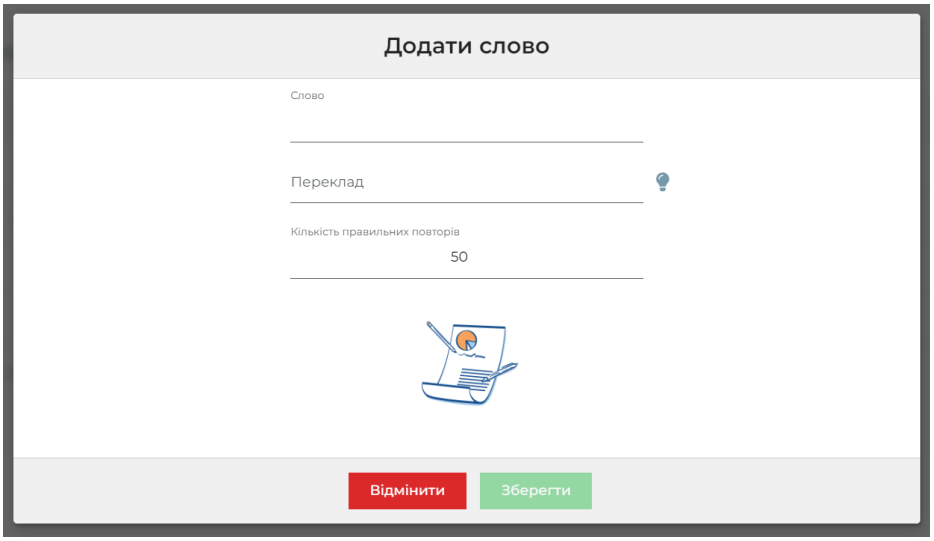

Рисунок 5.13 – Модальне вікно для додавання слова в словник

При натисканні на зображення блакитної лампи біля поля перекладу (рисунок 5.13) користувачеві буде запропоновано автоматичний переклад, який можна застосувати або ж відхилити (рисунок 5.14).

Рисунок 5.14 – Поле, що містить запропонований переклад

При натисканні на помаранчеву кнопку в правому нижньому кутку для початку навчання (рисунок 5.12) відкривається сторінка вивчення слів (рисунок

5.15), що містить такі ж опції для кожного слова, як і сторінка вивчення карток, а також поле для самоперевірки, в яке користувач може вводити переклад слова.

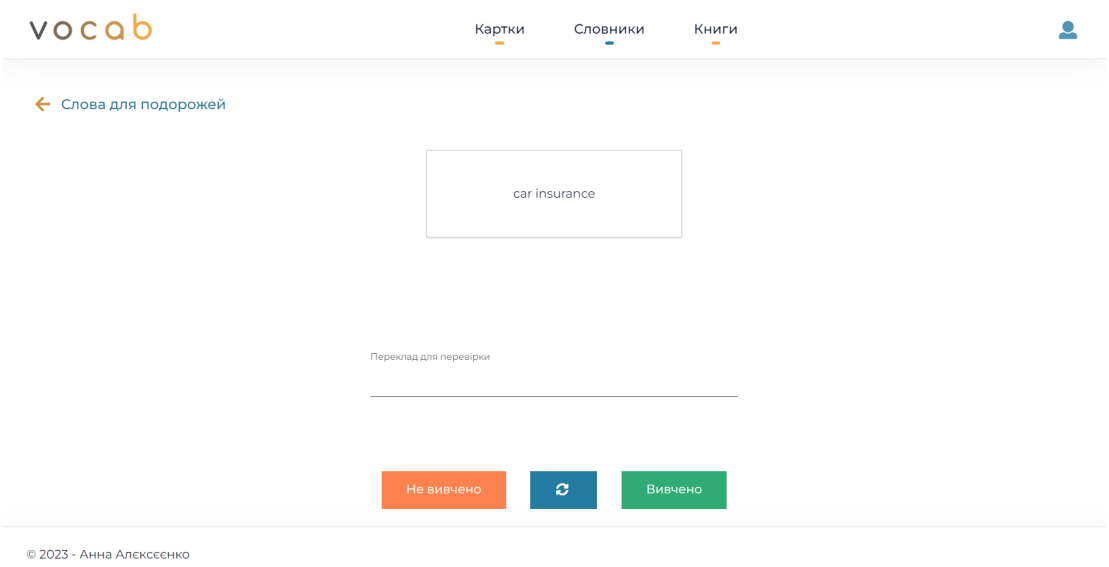

Рисунок 5.15 – Сторінка вивчення слів

При переході на сторінку книжок при натисканні на відповідну кнопку в заголовку сторінки користувачеві відображається сторінка зі списком завантажених книжок, зображена на рисунку 5.16.

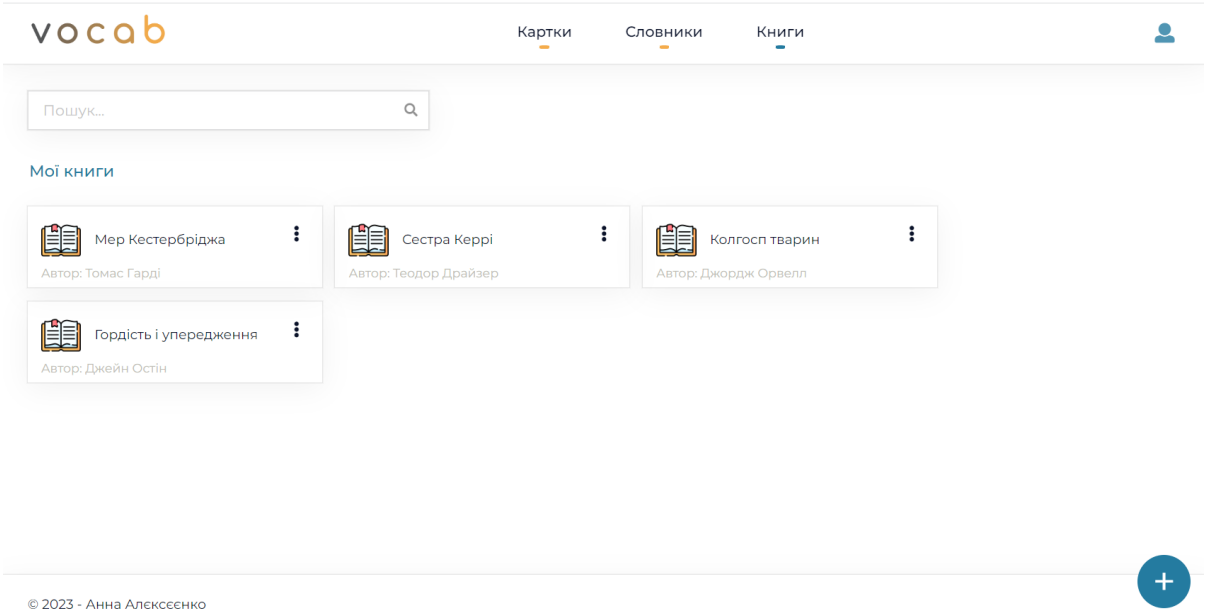

Рисунок 5.16 – Список завантажених книжок

Для кожної книги присутнє таке ж меню, як на сторінках з наборами карток та словниками. При виборі книги користувачеві відкривається сторінка з вмістом книги, зображена на рисунку 5.17, на якій він може гортати сторінки, переходити по заголовках книги, а також виділяти слова для подальшого додання в словники.

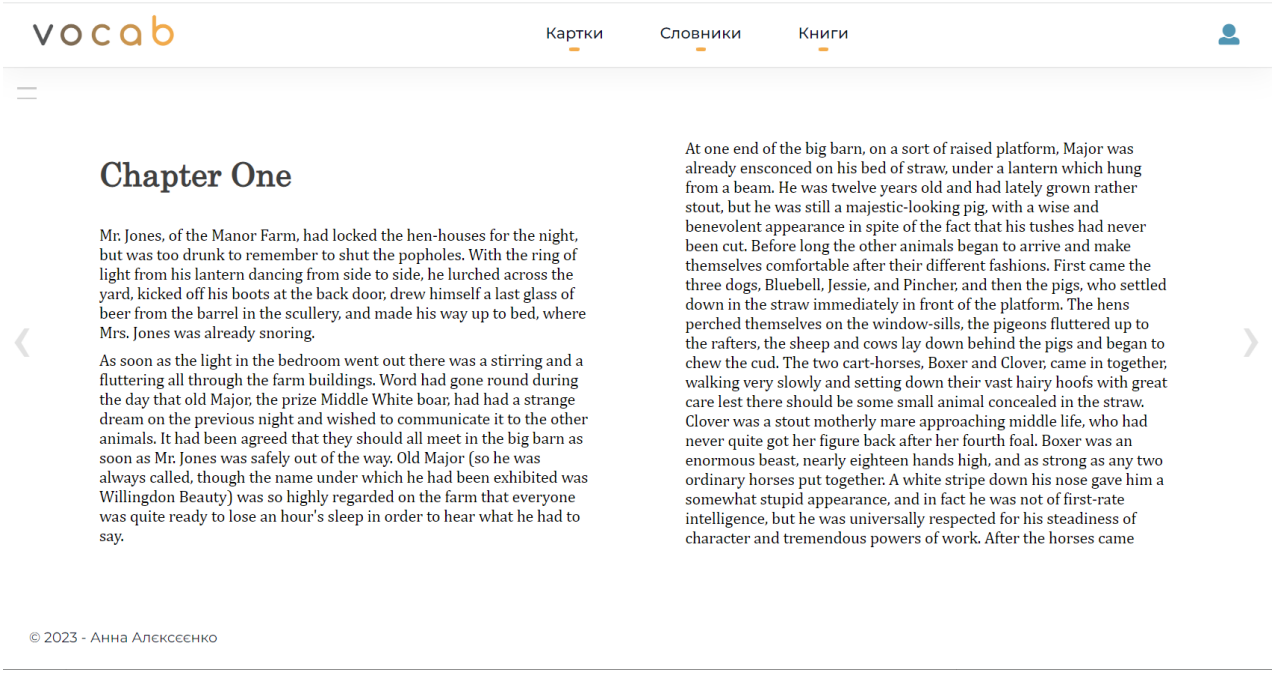

Рисунок 5.17 – Сторінка читання книги

При виділенні слова у книзі відкриється модальне вікно з вибором словника, зображене на рисунку 5.18, після чого користувач зможе ввести переклад слова і додати його в обраний словник.

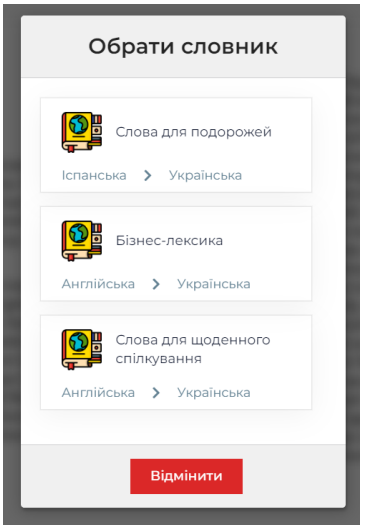

Рисунок 5.18 – Модальне вікно вибору словника

ВИСНОВКИ

Досліджено алгоритми навчання з використанням флеш-карток, проаналізовано веб-застосунки, наявні на ринку, спроєктовано архітектуру системи та схему бази даних застосунку, розроблено застосунок для вивчення іноземних мов з використанням флеш-карток, оптимізований під якнайшвидше вивчення матеріалу.

При розробці системи було досліджено сучасні методи та технології написання веб-застосунків.

Виявлено потребу в застосунках, націлених на якомога швидше засвоєння матеріалу в короткочасній пам’яті з можливістю вивчати необмежену кількість карток щоденно. Система, що була розроблена, заснована на алгоритмі для короткочасної пам’яті з використанням адаптивної складності. Окрім цього, система надає гнучкі інструменти для створення карток, підтримку автоматичного перекладу та інтегровану функцію читання книжок прямо в застосунку.

Розроблений застосунок може використовуватися для виконання домашніх завдань, підготовки до контрольних робіт та іспитів, вивчення нових слів під час читання книжок.

Перспективи подальшої розробки: портування застосунку на інші платформи – Windows, Linux, MacOS. Окрім цього, до наявного застосунку можна додавати більше режимів проведення навчання, в тому числі з задіянням слуху для запам’ятовування інформації та використанням досвіду гейміфікації для більш легкого засвоєння матеріалу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Kornell N. Optimising learning using flashcards: Spacing is more effective than cramming [Електронний ресурс] / Nate Kornell // Applied Cognitive Psychology. – 2009. – Т. 23, № 9. – С. 1297–1317. – Режим доступу: https://doi.org/10.1002/acp.1537.
2. Ebbinghaus H. Memory: A contribution to experimental psychology / Hermann Ebbinghaus. – New York City: Teachers College, Columbia University, 1913. – 123 с.
3. Spitzer H. F. Studies in retention. [Електронний ресурс] / H. F. Spitzer // Journal of Educational Psychology. – 1939. – Т. 30, № 9. – С. 641–656. – Режим доступу: https://doi.org/10.1037/h0063404.
4. Australasian Society for Computers in Learning in Tertiary Education Deakin University, Geelong, Australia. Conference Proceedings [Електронний ресурс] / Australasian Society for Computers in Learning in Tertiary Education Deakin University, Geelong, Australia // ASCILITE 2018 : Міжнар. наук. конф., Geelong, 25–28 листоп. 2018 р. – [Б. м.], 2018. – С. 570. – Режим доступу: https://2018conference.ascilite.org/wp-content/uploads/2018/12/ASCILITE-2018-Proceedings-Final.pdf.
5. The true history of spaced repetition [Електронний ресурс] // SuperMemo. – Режим доступу: https://www.supermemo.com/en/blog/the-true-history-of-spaced-repetition.
6. Application of a computer to improve the results obtained in working with the SuperMemo method [Електронний ресурс] // SuperMemo. – Режим доступу: https://www.supermemo.com/en/blog/application-of-a-computer-to-improve-the-results-obtained-in-working-with-the-supermemo-method.
7. Using Adaptive Flashcards for Automotive Maintenance Training in the Wild [Електронний ресурс] / Daphne E. Whitmer [та ін.] // Adaptive Instructional Systems. Design and Evaluation. – Cham, 2021. – С. 466–480. – Режим доступу: https://doi.org/10.1007/978-3-030-77857-6_33.

8. Spaced Repetition for All: Cognitive Science Meets Big Data in a Procrastinating World [Електронний ресурс] // Quizlet. – Режим доступу: https://quizlet.com/blog/spaced-repetition-for-all-cognitive-science-meets-big-data-in-a-procrastinating-world.
9. Textbook Solutions with Expert Answers [Електронний ресурс] // Quizlet. – Режим доступу: https://quizlet.com/explanations.
10. Our Teaching Approach [Електронний ресурс] // Duolingo. – Режим доступу: https://www.duolingo.com/approach.
11. Settles B. A Trainable Spaced Repetition Model for Language Learning [Електронний ресурс] / Burr Settles, Brendan Meeder // Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Berlin, Germany. – Stroudsburg, PA, USA, 2016. – Режим доступу: https://doi.org/10.18653/v1/p16-1174.
12. Anki - powerful, intelligent flashcards [Електронний ресурс] // Anki. – Режим доступу: https://apps.ankiweb.net.
13. Anki [Електронний ресурс] // SuperMemo Guru. – Режим доступу: https://supermemo.guru/wiki/Anki.
14. Memrise [Електронний ресурс] // Memrise. – Режим доступу: https://www.memrise.com.
15. How does the spaced repetition system work? [Електронний ресурс] // Memrise. – Режим доступу: https://memrise.zendesk.com/hc/en-us/articles/360015889057-How-does-the-spaced-repetition-system-work-.
16. Transition from Java EE to Jakarta EE [Електронний ресурс] // Oracle. – Режим доступу: https://blogs.oracle.com/javamagazine/post/transition-from-java-ee-to-jakarta-ee.
17. Apache Tomcat Versions [Електронний ресурс] // Apache Tomcat. – Режим доступу: https://tomcat.apache.org/whichversion.html.
18. Eclipse Jetty [Електронний ресурс] // The Eclipse Foundation. – Режим доступу: https://www.eclipse.org/jetty/.

19. Getting Started with Java Message Service (JMS) [Електронний ресурс] // Oracle. – Режим доступу: https://www.oracle.com/technical-resources/articles/java/intro-java-message-service.html.
20. Java Platform Overview [Електронний ресурс] // Oracle. – Режим доступу: https://docs.oracle.com/javase/8/docs/technotes/guides/index.html.
21. Spring Boot [Електронний ресурс] // Spring. – Режим доступу: https://spring.io/projects/spring-boot.
22. What is Java Spring Boot? [Електронний ресурс] // IBM - Deutschland | IBM. – Режим доступу: https://www.ibm.com/topics/java-spring-boot.
23. Spring Security Documentation [Електронний ресурс] // Spring. – Режим доступу: https://docs.spring.io/spring-security/reference/.
24. JWT [Електронний ресурс] // JSON Web Tokens - jwt.io. – Режим доступу: https://jwt.io/.
25. Jones M. JSON Web Token (JWT) [Електронний ресурс] / M. Jones, J. Bradley, N. Sakimura. – [Б. м.] : RFC Editor, 2015. – Режим доступу: https://doi.org/10.17487/rfc7519.
26. PostgreSQL: Documentation [Електронний ресурс] // PostgreSQL. – Режим доступу: https://www.postgresql.org/docs/.
27. Cloud Translation documentation [Електронний ресурс] // Google Cloud. – Режим доступу: https://cloud.google.com/translate/docs.
28. JavaScript [Електронний ресурс] // MDN Web Docs. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/JavaScript.
29. Thinking in React [Електронний ресурс] // React. – Режим доступу: https://react.dev/learn/thinking-in-react.
30. Virtual DOM and Internals [Електронний ресурс] // React. – Режим доступу: https://legacy.reactjs.org/docs/faq-internals.html.
31. State and Lifecycle [Електронний ресурс] // React. – Режим доступу: https://legacy.reactjs.org/docs/state-and-lifecycle.html#the-data-flows-down.

32. Redux - A predictable state container for JavaScript apps [Електронний ресурс] // Redux. – Режим доступу: https://redux.js.org/.
33. Redux Essentials, Part 1: Redux Overview and Concepts [Електронний ресурс] // Redux. – Режим доступу: https://redux.js.org/tutorials/essentials/part-1-overview-concepts
34. About Redux-Saga [Електронний ресурс] // Redux-Saga. – Режим доступу: https://redux-saga.js.org/docs/About.
35. Cascading Style Sheets [Електронний ресурс] // World Wide Web Consortium (W3C). – Режим доступу: https://www.w3.org/Style/CSS/Overview.en.html.
36. Saternos C. Client-Server Web Apps with JavaScript and Java / Casimir Saternos. – [Б. м.] : O'Reilly Media, Incorporated, 2014.
37. Richardson L. RESTful Web APIs / Leonard Richardson, Sam Ruby, Mike Amundsen. – [Б. м.] : O'Reilly Media, Incorporated, 2013.
38. Software Architecture Patterns [Електронний ресурс] // O'Reilly Online Learning. – Режим доступу: https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch01.html.

ДОДАТОК А

Діаграма прецедентів системи “Vocab”