Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка веб-застосунку DictionaryUp для вивчення слів англійської мови

Виконав: студент IV курсу, групи СНс-42
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

_____ Стодола Н.М.
(підпис) (прізвище та ініціали)

Керівник _____ Дмитроца Л.П.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Марценко С.В,
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ _____
(підпис) (прізвище та ініціали)

Тернопіль
2024

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«____» _червня_ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Стодола Назарій Михайлович
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка веб-застосунку DictionaryUp для вивчення слів англійської мови

Керівник роботи Дмитроца Леся Павлівна, к.т.н., доцент кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*29*» *квітня* 2024 року № *4/7-47Х*

2. Термін подання студентом завершеної роботи 24 червня 2024р.

3. Вихідні дані до роботи Літературні та інтернет джерела щодо розробки користувацької та серверної частини сайту

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. Розділ 1. Аналіз предметної області та постановка завдання. Розділ 2. Проектна частина. Розділ 3. Практична частина. Розділ 4. Безпека життєдіяльності, основи охорони праці. Висновки. Перелік джерел.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Титульний слайд. 2. Мета кваліфікаційної роботи. 3. Об'єкт та предмет дослідження. 4. Переваги та недоліки освітніх платформ. 5. Популярні рішення та конкуренти. 6. Основні можливості. 7. Варіанти використання. 8. Використані технології. 9. Архітектура додатку. 10. Стадії та етапи розробки. 11. Структура бази даних. 12. Серверна частина програми. 13. Клієнтська частина програми. 14. Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Сенчишин В.С., доцент кафедри МТ		

7. Дата видачі завдання 29 січня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи		
2.	Підбір джерел про розробку веб-застосунку для		
	Вивчення слів англійської мови		
3.	Опрацювання джерел щодо розробки веб-застосунку		
	для вивчення слів англійської мови		
4.	Виконання дослідження щодо розробки застосунку		
	для вивчення слів англійської мови		
	Розроблення веб-застосунку для вивчення слів		
	англійської мови		
5.	Оформлення розділу «АНАЛІЗ ПРЕДМЕТНОЇ		
	ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ»		
6.	Оформлення розділу «ПРОЕКТНА ЧАСТИНА»		
7.	Оформлення розділу «ПРАКТИЧНА ЧАСТИНА»		
8.	Виконання завдання до підрозділу «Безпека життєдіяльності»		
9.	Виконання завдання до підрозділу «Основи охорони		
	праці»		
10.	Оформлення кваліфікаційної роботи		
11.	Нормоконтроль		
12.	Перевірка на плагіат		
13.	Попередній захист кваліфікаційної роботи		
14.	Захист кваліфікаційної роботи		

Студент _____ Стодола Н.М
(підпис) (прізвище та ініціали)

Керівник роботи _____ Дмитроца Л.П.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка веб-застосунку DictionaryUp для вивчення слів англійської мови // Кваліфікаційна робота освітнього рівня «Бакалавр» // Стодола Назарій Михайлович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СНс – 42 // Тернопіль, 2024 // С. – 67, рис. – 16, додат. – 5, бібліогр. – 35.

Ключові слова: навчання, англійські слова, словник, сучасний веб-додаток, програмне забезпечення повного циклу.

Кваліфікаційна робота присвячена розробці сучасного веб-додатку в сфері освіти.

В першому розділі кваліфікаційної роботи описано аналіз додатків зі сфери освіти та конкурентів. Висвітлено основні функціональні вимоги до розроблюваного додатку. Розглянуто сучасний стек технологій для реалізації додатку. Проаналізовано основні стадії розробки.

В другому розділі кваліфікаційної роботи досліджено структуру додатку. Подано вибір архітектури для додатку. Спроєктовано базу даних для ефективного зберігання даних. Створено основний логотип бренду та проектовано основні веб-сторінки клієнтської частини.

В третьому розділі кваліфікаційної роботи описано розробку всіх функціональних частин додатку. Проаналізовано процес інтеграції клієнтської та серверної частин. Проведено тестування додатку з дотриманням сучасних правил розробки.

Об'єкт дослідження: веб-додаток для вивчення слів англійської мови.

Предмет дослідження: процес розробки веб додатку.

ANNOTATION

Development of "DictionaryUp" Web Application for Learning English Words // Qualification work of the educational level "Bachelor" // Stodola Nazarii // Ternopil Ivan Pulyu National Technical University, Computer and Information Systems and Software Engineering Faculty, Computer Sciences Department, group SNs-42 // Ternopil, 2024 // P. – 67, fig. – 16, annexes. – 5, references – 35 .

Keywords: learning, english words, dictionary ,modern web app, full cycle software.

The qualification work is dedicated to the development of a modern web application in the field of education.

The goal of the work is to create an effective tool for learning English words. The first section of the qualification paper considered the analysis of educational applications and competitors. It highlighted the main functional requirements for the developed application, reviewed the modern technology stack for its implementation, and analyzed the main stages of development.

In the second section of the qualification work, it is considered the structure of the application. The selection of architecture for the application was made, the database was designed for effective data storage, the main brand logo was created, and the main web pages of the client-side were designed.

The third section of the qualification work describes the development of all functional parts of the application. It analyzed the process of integrating the client-side and server-side and conducted testing of the application in accordance with modern development rules.

Object of the study: web application for learning English words.

Subject of the study: process of developing a web application.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

AI – Artificial Intelligence.

API – Application Programming Interface.

CRUD – Create, Read, Update, Delete peration.

JSON – JavaScript Object Notation.

JWT – JSON Web Token.

MVC – Model-View-Controller.

ORM – Object-Relational Mapping.

ЗМІСТ

ВСТУП

Актуальність теми. Сьогодні в умовах глобалізації та інтеграції в міжнародну спільноту знання англійської мови є необхідністю. Однак, ефективне вивчення мови вимагає постійного поповнення та практики словникового запасу. Незважаючи на величезну кількість існуючих освітніх платформ та додатків, які спрямовані на вивчення мов, залишається недостатньо сервісів, що дозволяють якісно менеджити та практикувати власний словник слів.

Багато людей все ще використовують паперові словники або електронні таблиці для збереження нових слів та виразів, що є неефективним та незручним методом. Таблиці не призначені для таких завдань і не забезпечують можливість інтерактивної практики, повторення та автоматичного оновлення даних. Це створює значні труднощі у підтримці актуального та організованого словника, що, у свою чергу, може сповільнити процес навчання.

Мета і задачі дослідження. Метою даної кваліфікаційної роботи освітнього рівня «Бакалавр» є підвищення якості послуг у сфері вивчення англійської мови шляхом створення ефективного інструменту для управління та практики власного словникового запасу.

Для досягнення поставленої мети потрібно виконати ряд завдань, зокрема:

- проаналізувати стан досліджень у сфері освітніх технологій, зокрема інструментів для вивчення та менеджменту словникового запасу;
- вивчити існуючі методи та підходи до створення флеш-карток та інших інструментів для запам'ятовування слів;
- дослідити потреби та вимоги користувачів до інструментів для управління словниковим запасом;

- розробити концептуальну модель інструменту, яка відповідатиме потребам користувачів та сучасним тенденціям у сфері освітніх технологій;
- створити прототип інструменту для управління словниковим запасом з основним функціоналом;
- провести тестування прототипу з метою оцінки його ефективності та зручності у використанні;
- внести необхідні коригування до прототипу на основі результатів тестування та відгуків користувачів;
- оцінити вплив розробленого інструменту на процес вивчення англійської мови та його ефективність у порівнянні з існуючими рішеннями;
- підготувати рекомендації щодо подальшого розвитку та вдосконалення інструменту для управління словниковим запасом.

Практичне значення одержаних результатів. Розроблений інструмент забезпечить користувачам можливість легко створювати, редагувати та практикувати свої словники слів. Це рішення повністю вирішує поставлену задачу підвищення якості послуг у сфері вивчення англійської мови.

У разі успіху та вдалого просування та помірного масштабування, розроблений продукт має значний комерційний потенціал. Його можна монетизувати різними способами як-от підписка на преміум акаунт, інтеграції з іншими освітніми сервісами тощо. Розроблений інструмент може стати важливим внеском у сферу освітніх технологій, підвищуючи ефективність навчання та надаючи користувачам зручний та ефективний спосіб управління своїм словниковим запасом. Крім того, продукт має потенціал для подальшого розвитку та вдосконалення, що відкриває нові можливості для його комерційного використання.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ.

1.1 Аналітичний огляд існуючих рішень

Даний період у процесі розробки програмного забезпечення має велике значення, оскільки він включає етапи, що допомагають визначити та проаналізувати оптимальні шляхи для втілення конкретних завдань. Цей етап дозволяє виокремити основні та додаткові процеси, головною метою яких є забезпечення своєчасного та якісного випуску продукту.

Аналіз предметної області є фундаментальним етапом у підготовці дипломної роботи, визначальним напрямом всього дослідження та її кінцевих результатів. Цей процес не тільки допомагає глибше зрозуміти досліджувану тему, але й виявляє ключові питання, завдання та гіпотези, які лежать в основі роботи.

Освітні платформи є поширеними і ефективними на сьогодні, особливо у даній області продуктів, які пропонують вивчати і тестувати слова. Розробка подібного інструменту є частиною цифровізації [1] навчального процесу. Використання подібних продуктів також суттєво може допомогти при здобуванні професійної освіти в сучасних умовах, покращуючи процес вивчення професійної лексики [2]. У цьому розділі буде розглянуто декілька основних конкурентів, їхні переваги та недоліки, а також проведено порівняльний аналіз. Основними конкурентами є:

1. Quizlet [3] (див. рис. 1.1). Quizlet була заснована у 2005 році з метою створення інструменту для створення, обміну та вивчення флеш-карток. Вона швидко стала популярною завдяки зручному інтерфейсу та великій базі користувачів. Платформа надає функціонал для створення та редагування флеш-карток, інтерактивні тести та ігри для закріплення знань. Основні переваги Quizlet включають величезну базу користувацьких наборів, зручний

інтерфейс та підтримку мобільних додатків. Недоліками є обмежена функціональність у безкоштовній версії та залежність від інтернет-з'єднання.

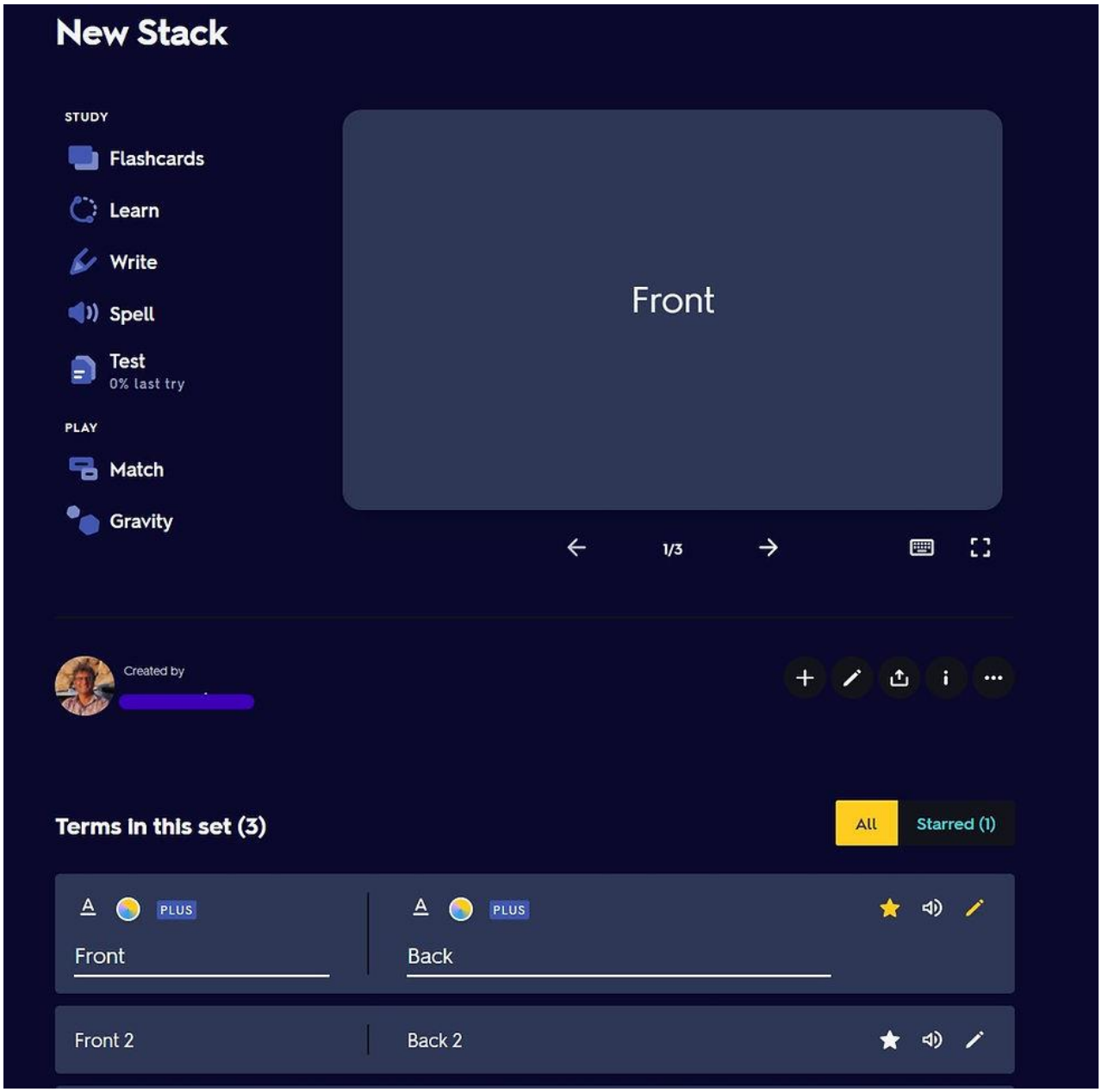

Рисунок 1.1 – Основний інтерефейс Quizlet

2. Anki [4] (див. рис. 1.2). Anki була заснована у 2006 році і спеціалізується на використанні алгоритмів спейст репетіціон для ефективного запам'ятовування. Платформа широко використовується серед студентів та професіоналів завдяки своїй ефективності. Anki дозволяє додавати

мультимедійний контент до карток, що підвищує їхню корисність. Основні переваги включають ефективність алгоритмів запам'ятовування та можливість створення складних карток. Недоліки полягають у менш зручному інтерфейсі порівняно з іншими сервісами та складності налаштувань.

Рисунок 1.2 – Основний інтерефейс Anki

3. Memrise [5]. Memrise була заснована у 2010 році і використовує мультимедійні засоби для запам'ятовування нових слів та виразів. Платформа має курсову структуру навчання з використанням відео та аудіо контенту, а також інтерактивні вправи. Memrise приваблює користувачів великою кількістю готових курсів та інтерактивними вправами, доступними через

мобільний додаток. Недоліками є обмежена функціональність у безкоштовній версії та потреба в підписці для доступу до більшості курсів.

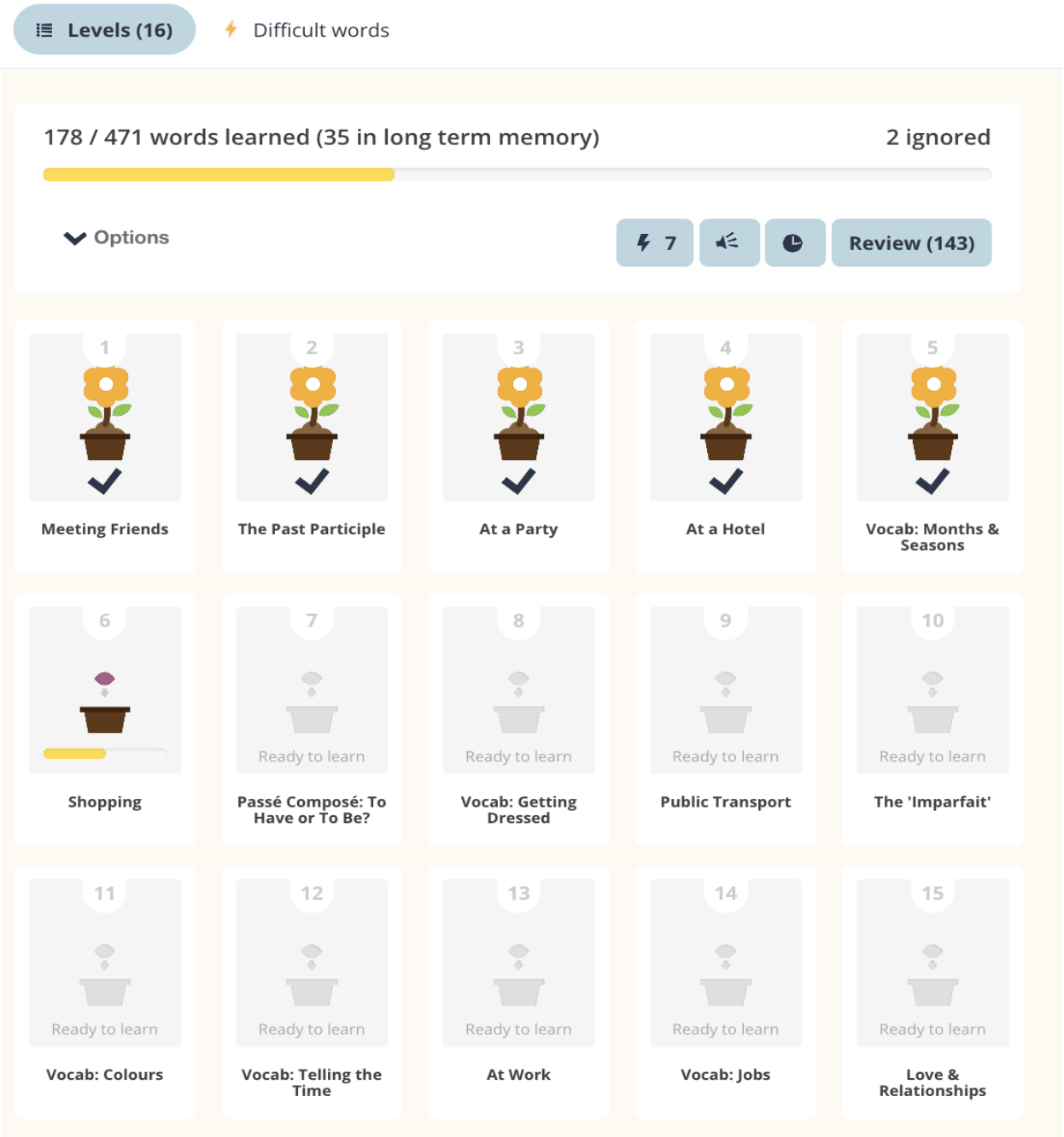

Рисунок 1.3 – Основний інтерефейс Memrise

4. Brainscape [6]. Brainscape була заснована у 2011 році з метою покращення процесу запам'ятовування за допомогою флеш-карток. Платформа дозволяє створювати та редагувати картки, синхронізувати їх між пристроями

та аналізувати прогрес навчання. Brainscape відома своєю простою у використанні платформою та можливістю детального аналізу прогресу. Основні недоліки включають потребу у підписці для доступу до повного функціоналу та відсутність інтерактивних ігор.

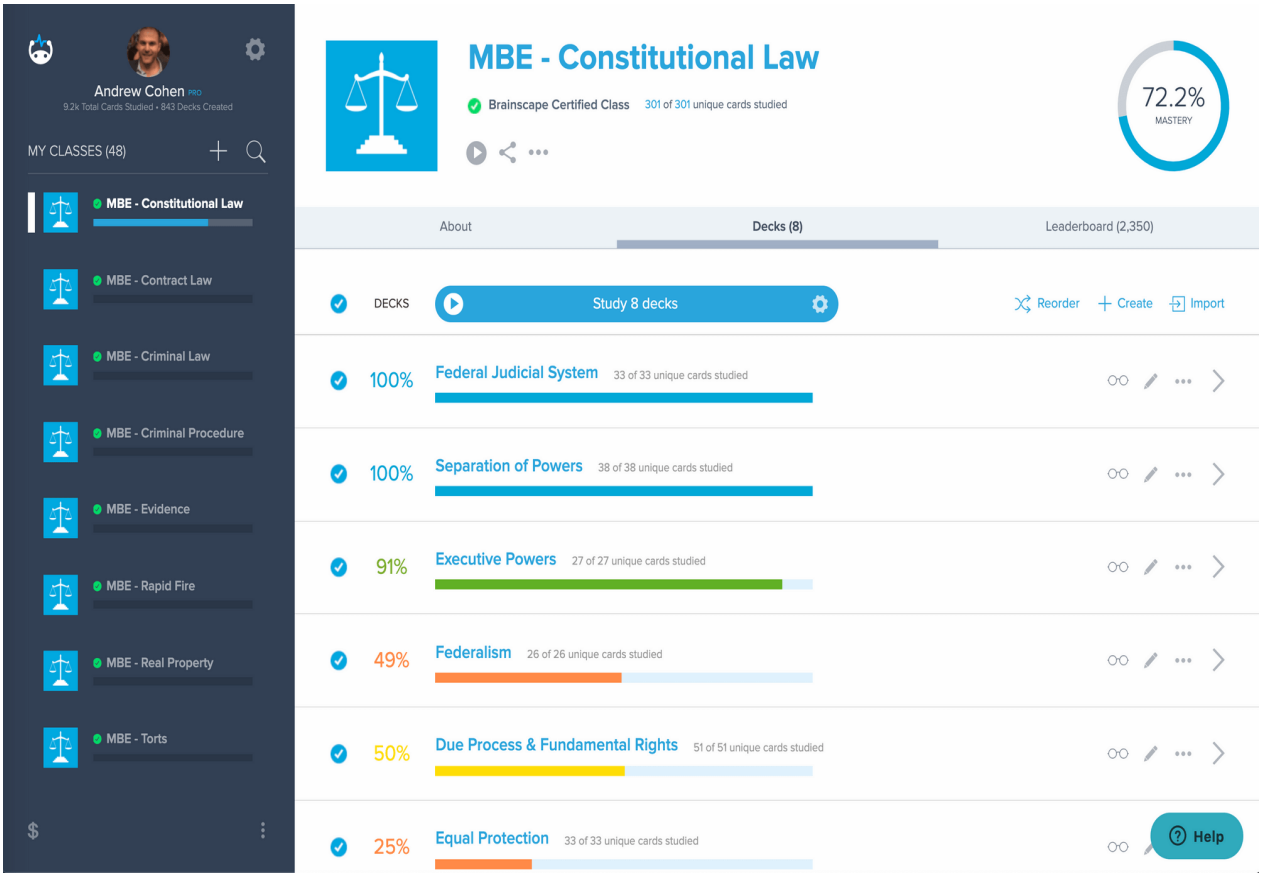

Рисунок 1.4 – Основний інтерефейс Brainscape

Порівняльний аналіз існуючих рішень для управління словниковим запасом показує, що кожна платформа має свої унікальні особливості, переваги та недоліки.

Quizlet та Memrise пропонують найбільшу кількість інтерактивних функцій, включаючи тести та ігри, що робить навчання більш цікавим та ефективним. Anki виділяється своїм підходом до запам'ятовування, використовуючи алгоритми спейст репетіціон, що є ефективним для довготривалого зберігання інформації. Brainscape пропонує простий у

використанні інтерфейс та детальну аналітику прогресу, що може бути корисним для відстеження результатів.

Більшість платформ мають обмежену функціональність у безкоштовній версії. Anki є винятком, оскільки пропонує повний функціонал безкоштовно. Платформи, такі як Quizlet та Memrise, пропонують додатковий платний контент та функції, що можуть бути корисними для користувачів, готових інвестувати в своє навчання фінансово. Ці сервіси пропонують кілька варіантів спільного навчання, що дозволяє учням працювати разом і вчитися у своїх однолітків. Такі функції, як навчальні групи, спільні навчальні набори та навчальні сеанси в реальному часі, сприяють командній роботі та створюють відчуття спільності. Цей спільний підхід покращує розуміння через обговорення та обмін знаннями.

Зручність використання. Quizlet та Memrise мають зручні та інтуїтивно зрозумілі інтерфейси, що робить їх привабливими для більш широкого кола користувачів. Вони також аналізують успішність учня та визначають пріоритетні сфери, які потребують вдосконалення. Anki може бути складнішим для новачків через складність налаштувань, але пропонує гнучкість та потужні інструменти для тих, хто готовий витратити час на навчання. Brainscape, своєю чергою, забезпечує простоту використання та корисні аналітичні інструменти, що робить його зручним для відстеження прогресу користувача.

1.2 Вимоги до функціоналу

Вимоги до функціоналу є ключовим етапом у розробці програмного забезпечення, оскільки визначають, як саме буде працювати продукт та які завдання він буде виконувати. У даному розділі ми розглянемо основні вимоги до функціоналу "DictionaryUp".

Доступність для широкої аудиторії – інструмент повинен бути доступним для широкого спектру користувачів, включаючи школярів, студентів та

дорослих, які прагнуть покращити свої знання англійської мови. Простота та зручність використання роблять його підходящим для людей різного віку та рівня підготовки. Важливо, щоб додаток був доступний на різних платформах, таких як комп'ютери та мобільні пристрої, щоб користувачі могли займатися навчанням у будь-який час та в будь-якому місці.

Ефективність навчання – завдяки інтерактивним функціям та можливості персоналізації навчання, користувачі зможуть більш ефективно запам'ятовувати нові слова та вирази. Інтерактивні флеш-картки та тести сприятимуть кращому засвоєнню матеріалу, що є ефективним методом запам'ятовування.

Підвищення мотивації – інтерактивний підхід та зручний інтерфейс стимулюють користувачів до регулярного навчання та підтримання свого словникового запасу в актуальному стані. Впровадження елементів гейміфікації, таких як нагороди, досягнення та рейтинги, допоможе підтримувати мотивацію користувачів. Соціальні функції, такі як можливість ділитися успіхами з друзями та створювати групи для спільного навчання, також сприятимуть підвищенню мотивації.

Конфіденційність та захист персональних даних – забезпечення конфіденційності та захисту персональних даних користувачів є пріоритетом. Це включає впровадження заходів для захисту даних, таких як шифрування, захист паролів та безпечне зберігання інформації.

Важливо також забезпечити можливість користувачам контролювати доступ до своїх даних та налаштовувати рівень конфіденційності. Відповідність міжнародним стандартам та нормативним вимогам щодо захисту даних, таким як GDPR [7], є обов’язковою умовою.

Продуктивність – висока продуктивність програми є ключовою вимогою для забезпечення стабільної та надійної роботи. Це включає швидке завантаження, обробку запитів та виконання операцій пов’язаних із опрацьовуванням навчального матеріалу. Затримки у завантаженні та збільшення часу очікування може призвести до отримання негативного досвіду

та можуть призвести до втрати користувачів. Програма повинна бути масштабованою, щоб зростаюча кількість користувачів та даних не спричиняли втрату продуктивності.

Надійність та стабільність роботи також є важливими аспектами, оскільки вони забезпечують безперебійну роботу. Оптимізація використання апаратних та програмних ресурсів допоможе знизити витрати на інфраструктуру та забезпечити оптимальну продуктивність.

1.3 Стадії та етапи розробки

Процес розробки програмного забезпечення включає кілька ключових етапів, кожен з яких допомагає організувати роботу, розділити її на підзадачі та забезпечити своєчасний та якісний випуск продукту.

1. Формування вимог та їхній аналіз. Першим етапом є формування вимог до продукту та їх аналіз. Це включає вибір технологій, які будуть використовуватися для розробки, визначення функціональних та нефункціональних вимог, а також складання специфікацій. Важливо також визначити цільову аудиторію продукту та її потреби, щоб забезпечити максимально ефективне вирішення завдань користувачів.

2. Проектування. На етапі проектування здійснюється налаштування середовища розробки, вибір відповідних технологій та інструментів, а також ініціалізація та конфігурація проекту. Проектування включає створення структури бази даних, визначення схем збереження даних та їхньої взаємодії. Важливим аспектом є також розробка дизайну користувацького інтерфейсу, який буде зручним та привабливим для користувачів.

3. Реалізація. Етап реалізації включає написання програмного коду для клієнтської та серверної частини програми. Це також передбачає налаштування комунікації між різними компонентами системи, такими як клієнтська частина, сервер та база даних. Використання патерну MVC допомагає розділити логіку

програми на окремі частини, що спрощує розробку та підтримку коду [8], повторне використання компонентів у процесі розробки зробить кодову базу більш структурованою [9].

4. Тестування. Тестування є критично важливим етапом для окремих компонентів програми, а також мануальне тестування для перевірки загальної функціональності. Мета тестування – виявити та виправити помилки, забезпечити стабільність та надійність роботи програми.

5. Впровадження. Останній етап – впровадження, включає перевірку функціональності продукту за межами середовища розробки, проведення тестування у реальних умовах використання. Це дозволяє виявити та виправити можливі проблеми, які можуть виникнути під час використання програми. Після успішного впровадження продукт випускається для кінцевих користувачів, забезпечуючи технічну підтримку та подальше вдосконалення.

1.4 Обґрунтування використовуваних технологій

Правильний вибір технологій є критично важливим для успішної розробки додатка, оскільки він визначає продуктивність, масштабованість, безпеку та зручність підтримки проекту. Вибір технологій має відповідати конкретним потребам додатка та забезпечувати оптимальні умови для його розвитку.

Для проєкту обрано архітектуру MVC та REST [10], оскільки ці підходи є найбільш поширеними та перевіреними рішеннями на 2024 рік. Структура проекту буде монорепо [11], тобто клієнтська і серверна частини зберігатимуться в одній папці та спільному репозиторію контролю версій, що спрощує управління кодом та спільну роботу над проектом.

Основною мовою програмування обрано об’єктно-орієнтований TypeScript замість JavaScript [12]. TypeScript забезпечує строгішу типізацію, що дозволяє зменшити кількість помилок на етапі розробки. Хоча старт розробки

може бути повільнішим, у довгостроковій перспективі розробники отримають кращий досвід та ефективність роботи над проектом значно зросте на пізніх етапах.

База даних буде реляційною, і обрано MySQL. Вибір був між MySQL та PostgreSQL [13], але оскільки не планується використовувати складні функціональні можливості баз даних такі як можливість асинхронної реплікації, MySQL є оптимальним варіантом через свою простоту та ефективність.

Серверним фреймворком буде Express, який відомий своєю простотою та ефективністю. Він дозволяє швидко налаштувати серверну частину та забезпечити надійну роботу з мінімальними витратами часу та ресурсів. Мінімалістичний дизайн фреймворку завдяки невеликій кількості конфігурацій чудово підходить для проектів малого та середнього розміру та при збільшенні кількості активних користувачів може легко маштабуватися горизонтально.

На клієнтській частині буде використано Vue.js. Цей фреймворк простіший у використанні "з коробки" порівняно з Angular [14], який є більш комплексним і вимагає значного налаштування перед початком розробки. Angular використовує TypeScript за замовчуванням, що може бути перевагою, але його складність часто перевищує потреби менших проектів.

Порівняно з React, Vue.js має декілька важливих переваг [15]. Vue.js пропонує більш інтуїтивно зрозумілий та легший у використанні глобальний стейт-менеджер Pinia [16], що значно спрощує управління станом додатка. Директиви у Vue.js дозволяють легко маніпулювати DOM [17], а computed properties надають простий спосіб обчислення властивостей на основі стану, що робить код чистішим та зручнішим для підтримки [18]. У React подібний функціонал реалізується через використання хуків, які є складнішими у правильній реалізації та тестуванню.

Vue.js надає потужний віджет для дебагу та відстежування стану компонентів – Vue Developer Tools (див. рис. 1.5), який включено за

замовчуванням у кожен ініціалізований проект [19]. Немає потреби встановлювати його як окремий пакет або розширення в браузері, як у випадку з React.js. Він тісно інтегрований з ініціалізованим проектом, що полегшує його використання та налаштування. Інтерфейс інтуїтивно зрозумілий та зручний для розробників, дозволяє швидко відстежувати стани, події, дерево компонентів та будь-які інші зміни у додатку, значно спрощуючи процес дебагу. Віджет включає в себе інструменти для огляду компонентів, відстежування стану та мутацій, аналізу продуктивності та інші корисні функції. Vue Developer Tools не потребує окремої вкладки та інтегрується прямо на вебсторінку, тому є сумісним з різними браузерами та операційними системами, що робить його універсальним інструментом для розробників та демонструє значну перевагу у зручності та ефективності відслідковування помилок та відстежування стану компонентів порівняно з React.js.

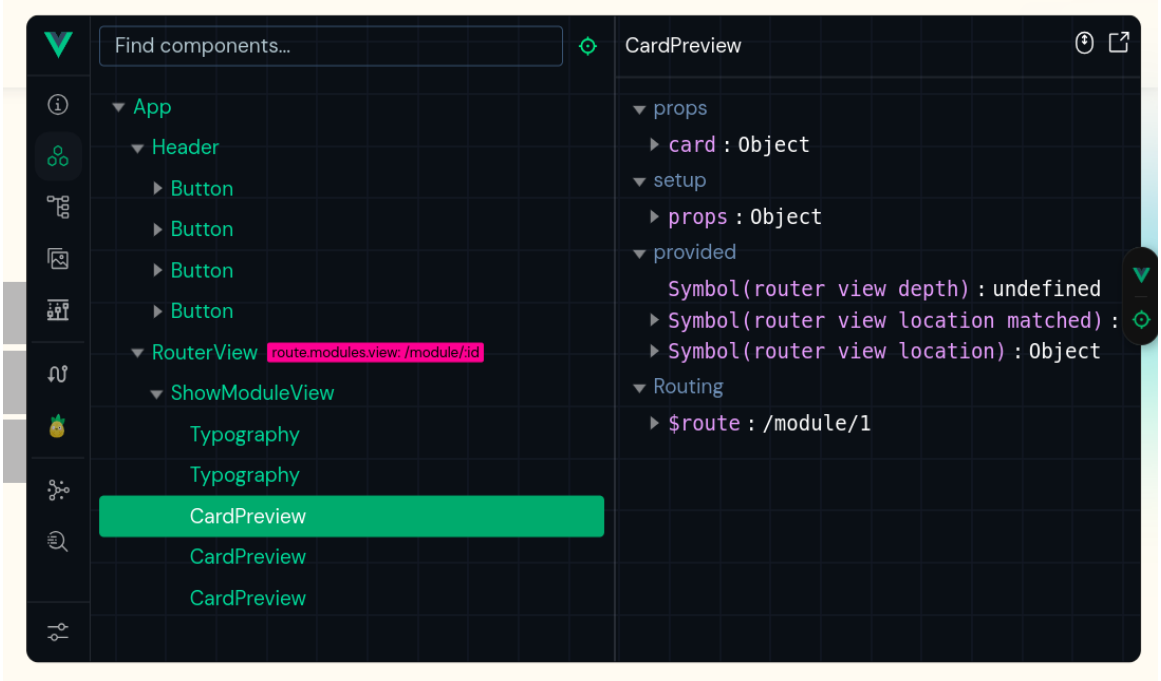

Рисунок 1.5 – Інтерфейс Vue Developer Tools

Для користувацького інтерфейсу буде використано бібліотеку Ant Design. Основний вибір був між Material UI від Google та Ant Design. Ant Design має

менший розмір пакету та пропонує велику кількість готових стилів та компонентів [20]. Крім того, враховано попередній досвід роботи розробника з Ant Design, що забезпечить швидше впровадження та налаштування.

Для збірки клієнтської частини обрано Vite. Цей інструмент забезпечує кращий досвід розробки завдяки hot reload [21], що дозволяє миттєво відображати зміни в коді без необхідності перезавантаження сторінки. Крім того, Vite забезпечує швидші збірки та ефективний розподіл на модулі значно перевершуючи Webpack за швидкістю та зручністю використання [22].

Уся інфраструктура буде запускатися в Docker, що дозволить легко керувати та масштабувати проект. Буде створено три контейнери: для бази даних, клієнтської частини та сервера. Це забезпечить ізоляцію компонентів, полегшить розгортання та підтримку додатка, а також підвищить стабільність та безпеку системи [23].

Використання цих технологій забезпечить оптимальну продуктивність, зручність розробки та підтримки, а також дозволить створити якісний та ефективний продукт для управління словниковим запасом.

1.5 Висновок до першого розділу

В першому розділі кваліфікаційної роботи було проведено комплексний аналіз та визначено ключові аспекти, необхідні для успішної розробки проекту "DictionaryUp".

Було виконано аналітичний огляд існуючих рішень для управління словниковим запасом, таких як Quizlet, Anki, Memrise та Brainscape. Було визначено основні функціональні можливості, переваги та недоліки кожного з них, що дозволило виявити сильні та слабкі сторони конкурентів та визначити оптимальні шляхи для розробки власного продукту.

На основі цього аналізу були сформульовані вимоги до функціоналу "DictionaryUp". Важливими аспектами стали доступність для широкої

аудиторії, ефективність навчання, підвищення мотивації користувачів, а також забезпечення конфіденційності та захисту персональних даних. Крім того, було підкреслено необхідність високої продуктивності та надійності системи.

Були визначені стадії та етапи розробки проекту, включаючи формування вимог, проектування, реалізацію, тестування та впровадження. Цей підхід дозволяє організувати процес розробки таким чином, щоб забезпечити своєчасний та якісний випуск продукту.

У розділі обґрунтування використовуваних технологій було прийнято рішення про використання сучасного стека технологій, включаючи MySQL як реляційну базу даних, Express як серверний фреймворк, Vue.js для клієнтської частини, та TypeScript для забезпечення строгої типізації та зменшення кількості помилок. Такий вибір технологій забезпечить високу продуктивність, надійність та зручність розробки, що сприятиме створенню якісного та ефективного програмного продукту.

Таким чином, перший розділ закладає міцну основу для подальшої розробки "DictionaryUp", визначаючи чіткі вимоги, етапи розробки та обґрунтовані технологічні рішення, що забезпечують успіх проекту.

РОЗДІЛ 2. ПРОЕКТНА ЧАСТИНА.

2.1 Розробка структури додатку

Оскільки для проекту "Dictionary Up" обрано архітектуру MVC важливо зрозуміти, як кожна з цих складових реалізується в нашій програмі:

- Model (Модель): Це частина додатку, яка відповідає за взаємодію з базою даних MySQL. Модель обробляє та конвертує в потрібний формат дані, отримані з бази, та готує їх для подальшого використання контролером і представленням. У проєкті модель буде реалізована на сервері за допомогою Express та ORM бібліотеки для роботи з MySQL;
- View (Представлення): Це частина додатку, яка відповідає за відображення даних користувачеві. У даному випадку представлення буде реалізовано за допомогою Vue.js, що дозволить створити динамічний та інтерактивний інтерфейс користувача. Vue.js компоненти будуть відображати дані, отримані від контролера, та забезпечуватимуть взаємодію з користувачем
- Controller (Контролер): Це частина додатку, яка відповідає за обробку запитів від користувача, взаємодію з моделлю та передання даних представленню. У проєкті контролер буде реалізований на сервері за допомогою Express, який оброблятиме HTTP-запити, викликатиме методи моделі для роботи з базою даних та надсилатиме відповіді клієнтській частині у форматі JSON [24].

2.1.1 Структура проєкту

Проект "DictionaryUp" буде організовано як монорепо, тобто і клієнтська, і серверна частини зберігатимуться в одному репозиторії та хоститимуться на одному сервері для спрощення процесу розробки та зменшення витрат на старті проекту та під час перших релізів. Така структура

дозволяє легко керувати кодом, спільно працювати над різними частинами проекту та швидше впроваджувати зміни. Монорепо забезпечує централізоване управління залежностями та дозволяє уникнути проблем з сумісністю версій різних частин проекту.

У майбутньому, коли проект зростатиме та вимагатиме більшої масштабованості, монорепо можна легко розділити на окремі репозиторії для клієнтської та серверної частин.

І для серверної, і для клієнтської частини буде використовуватися структура проекту за замовчуванням. Оскільки використовуються фреймворки, поділ на структури та модулі є визначеним наперед. Фреймворки, такі як Vue.js і Express, надають багато визначених компонентів, структур, модулів та підходів "з коробки". Вони вже гарно налаштовані для взаємодії один з одним, що дозволяє розробникам зосередитися на створенні кастомної бізнес-логіки, а не витрачати час на налаштування базової інфраструктури.

Фреймворки значно спрощують розробку завдяки своїм вбудованим функціям і стандартам. Наприклад, Vue.js надає прості у використанні компоненти та директиви, що робить розробку інтерфейсу швидшою та ефективнішою. До глобального об’єкту також можна прив’язати глобальні властивості та змінні, зареєструвати компоненти та плагіни. Express, зі свого боку, забезпечує легку інтеграцію з базою даних та іншими сервісами, що зменшує необхідність ручного налаштування серверної частини.

Використання цих фреймворків не тільки прискорює процес розробки, але й забезпечує високу якість та надійність коду завдяки перевіреним часом і спільнотою підходам. Змінювати структуру, задану фреймворками, має сенс лише у випадках вирішення специфічних проблем чи задач конкретного проекту, що не є типовими для більшості розробок. На додачу усі фреймворки забезпечені детальною документацією та навіть туторіалами із використання, тому відхилення від базової структури можу ускладнити написання коду та використання існуючих засобів розробки.

2.2 Розробка структури бази даних

Проектування бази даних є важливим етапом у розробці додатку, оскільки від цього залежить ефективність зберігання та доступу до даних. Для "DictionaryUp" було вирішено використовувати реляційну базу даних Mysql. Спочатку необхідно визначити ключові таблиці, які будуть зберігати основні сутності додатку. Головна і найважливіша сутність у додатку – це модулі, які об'єднують навколо себе багато карток. Кожна картка складається з оригінального іноземного тексту та перекладу на українську мову, а також має додаткову мітку, що вказує на її тип.

1. Модулі (modules) є головною сутністю в додатку, оскільки вони об'єднують групи карток. Кожен модуль представляє собою набір слів, який буде вивчати користувач. Вони організують словниковий матеріал у логічні блоки, що дозволяє користувачам ефективно структурувати своє навчання. Наприклад, модулі можуть бути тематичними (професії, подорожі) або граматичними (іменники, прикметники). Також у модулів є ще одна важлива характеристика – це публічність, тобто чи буде він доступний для інших користувачів, чи ні. Така властивість забезпечить можливість давати доступ до свої персональних модулів іншим користувачам. Для захисту від надмірного використання ресурсів програми та бази даних, розмір одного модулі обмежено в 100 карток. На основі викладених вимог до таблиці отримуємо наступну структуру:

- id (PRIMARY KEY) – унікальний ідентифікатор модуля;
- name – назва модуля;
- description – опис модуля;
- is_draft – поле, яке вказує, чи є модуль чернеткою;
- is_public – поле, яке вказує, чи є модуль публічним;
- user_id (FOREIGN KEY) – ідентифікатор користувача, який створив модуль;

- created_at – Дата створення модуля;
- updated_at – Дата останнього оновлення модуля.

2. Картки (cards) є основними елементами вивчення в модулі. Кожна картка складається з оригінального тексту іноземною мовою та перекладу на українську мову. Картки можуть містити окремі слова, словосполучення, і навіть цілі речення, що дозволяє користувачам вивчати не лише окремі слова, а й контекст їх використання. Картки мають мітки, що вказують на їх тип. Тип може бути визначений як частина мови (іменник, прикметник) або тематична категорія (професії, подорожі). Це допомагає користувачам сортувати і фільтрувати картки за різними критеріями. Також картки можуть мати приклади використання слова в реченнях. На основі викладених вимог до таблиці отримуємо наступну структуру:

- id (PRIMARY KEY) – унікальний ідентифікатор картки;
- origin – оригінальний текст іноземною мовою;
- translate – переклад на українську мову;
- module_id (FOREIGN KEY) – ідентифікатор модуля, до якого належить картка;
- label_id (FOREIGN KEY) – ідентифікатор позначки, що вказує на тип картки;
- created_at – дата створення картки;
- updated_at – дата останнього оновлення картки.

3. Мітки (labels) використовуються для класифікації карток. Кожна мітка має поле, яке вказує на її активність, тобто чи доступна вона для вибору чи ні. Мітки дозволяють точно класифікувати картки за типами та категоріями, що полегшує користувачам навігацію і пошук необхідних карток. Враховуючи, що однакові слова в іноземній мові можуть мати різні значення та належати до різних частин мови, мітки забезпечують точність і зручність у використанні карток. Додаткове поле активності міток дозволяє редагувати чи видаляти їх

без втрати користувацьких даних. На основі викладених вимог до таблиці отримуємо наступну структуру:

- id (PRIMARY KEY) – унікальний ідентифікатор позначки;
- name – назва позначки;
- is_active – поле, яке вказує, чи активна позначка;
- created_at – дата створення позначки;
- updated_at – дата останнього оновлення позначки.

4. Користувачі (users) – таблиця, яка зберігає дані користувачів, що включає нікнейм, електронну пошту та пароль. Нікнейм може бути вигаданим або справжнім ім'ям користувача і буде видимим для інших користувачів. Електронна пошта використовується для можливих розсилок та підтвердження реєстрації. Пароль забезпечує доступ до системи. Кожен користувач може створювати необмежену кількість модулів. На основі викладених вимог до таблиці отримуємо наступну структуру:

- id (PRIMARY KEY) – унікальний ідентифікатор користувача;
- name – ім'я або нікнейм користувача;
- email – електронна пошта користувача;
- password – пароль для доступу до системи;
- created_at – дата створення облікового запису;
- updated_at – дата останнього оновлення облікового запису.

5. Результати (results) – призначені для зберігання даних про проходження модулів користувачами. Ця таблиця фіксує інформацію про те, який користувач проходив який модуль, який результат він здобув, а також зберігає копію модуля із введеними відповідями у форматі JSON. Результати проходження модулів включають детальну інформацію про кожну спробу користувача. Зберігається ідентифікатор користувача, ідентифікатор модуля, час проходження, отриманий бал або результат, а також копія модуля у форматі JSON з усіма введеними відповідями. Це дозволяє зберігати історію

проходження для кожного користувача, що є важливим для аналізу прогресу та оцінки ефективності навчання. Збереження копії модуля у форматі JSON важливо для забезпечення коректності історичних даних, оскільки модулі можуть змінюватися з часом. Це дозволяє зберігати результати, прив'язані до конкретної версії модуля, що дає можливість відновити контекст навчання на момент проходження. На основі викладених вимог до таблиці отримуємо наступну структуру:

- id (PRIMARY KEY) – унікальний ідентифікатор результату.
- module_id (FOREIGN KEY) – ідентифікатор модуля, який було пройдено.
- user_id (FOREIGN KEY) – ідентифікатор користувача, який пройшов модуль.
- score – результат, отриманий користувачем.
- module_entity – копія модуля із введеними відповідями у форматі JSON.
- created_at – дата створення результату.
- updated_at – дата останнього оновлення результату.

Процес нормалізації бази даних включає розділення даних на таблиці таким чином, щоб мінімізувати дублювання інформації та забезпечити цілісність даних [25]. Основні принципи нормалізації були дотримані для забезпечення ефективності бази даних. У структурі бази даних використано первинні та зовнішні ключі для встановлення зв'язків між таблицями, а також розділення даних на логічні блоки. Наприклад, таблиця modules містить основну інформацію про модулі, а таблиця cards зберігає картки, пов'язані з конкретними модулями за допомогою зовнішнього ключа module_id. Це дозволяє уникнути дублювання даних і забезпечує чітку організацію інформації. Аналогічно, таблиця labels містить інформацію про позначки, які можуть бути використані для класифікації карток, а таблиця results зберігає історію проходження модулів користувачами, зберігаючи ідентифікатори

модулів та користувачів для встановлення зв'язків. Таке розділення даних на логічні блоки дозволяє ефективно керувати інформацією та забезпечує гнучкість у використанні даних.

Для забезпечення швидкого доступу до даних у базі будуть використовуватися індекси. Основні індекси будуть створені для полів user_id, module_id та label_id, щоб забезпечити швидкий пошук та фільтрацію даних. Наприклад, індекси на полі user_id у таблицях modules та results дозволять швидко знаходити всі модулі, створені певним користувачем, та всі результати його проходження модулів. Індекси на полі module_id у таблицях cards та results забезпечать швидкий доступ до карток певного модуля та результатів його проходження. Індекси на полі label_id у таблиці cards дозволять швидко фільтрувати картки за їх типами або категоріями. Використання індексів значно покращує продуктивність запитів до бази даних, особливо при роботі з великими обсягами даних.

Цілісність даних буде забезпечена за допомогою обмежень на рівні бази даних, таких як обов'язковість полів (NOT NULL), унікальні ключі (UNIQUE), а також зовнішні ключі (FOREIGN KEY) для забезпечення коректних зв'язків між таблицями. Наприклад, у таблиці users поля email та password будуть обов'язковими для заповнення, а поле email буде унікальним, щоб уникнути дублювання облікових записів. У таблиці modules поле user_id буде зовнішнім ключем, що посилається на таблицю users, забезпечуючи, що кожен модуль створений конкретним користувачем. Аналогічно, у таблиці cards поле module_id буде зовнішнім ключем, що посилається на таблицю modules, а поле label_id буде зовнішнім ключем, що посилається на таблицю labels, забезпечуючи коректні зв'язки між картками, модулями та позначками. У таблиці results поля module_id та user_id будуть зовнішніми ключами, що посилаються на відповідні таблиці, забезпечуючи зберігання історії проходження модулів користувачами. Такі обмеження дозволяють зберігати цілісність даних, запобігати некоректним записам і забезпечувати зв'язність

інформації між різними таблицями бази даних. На рисунку 2.1 зображено структуру бази даних графічно:

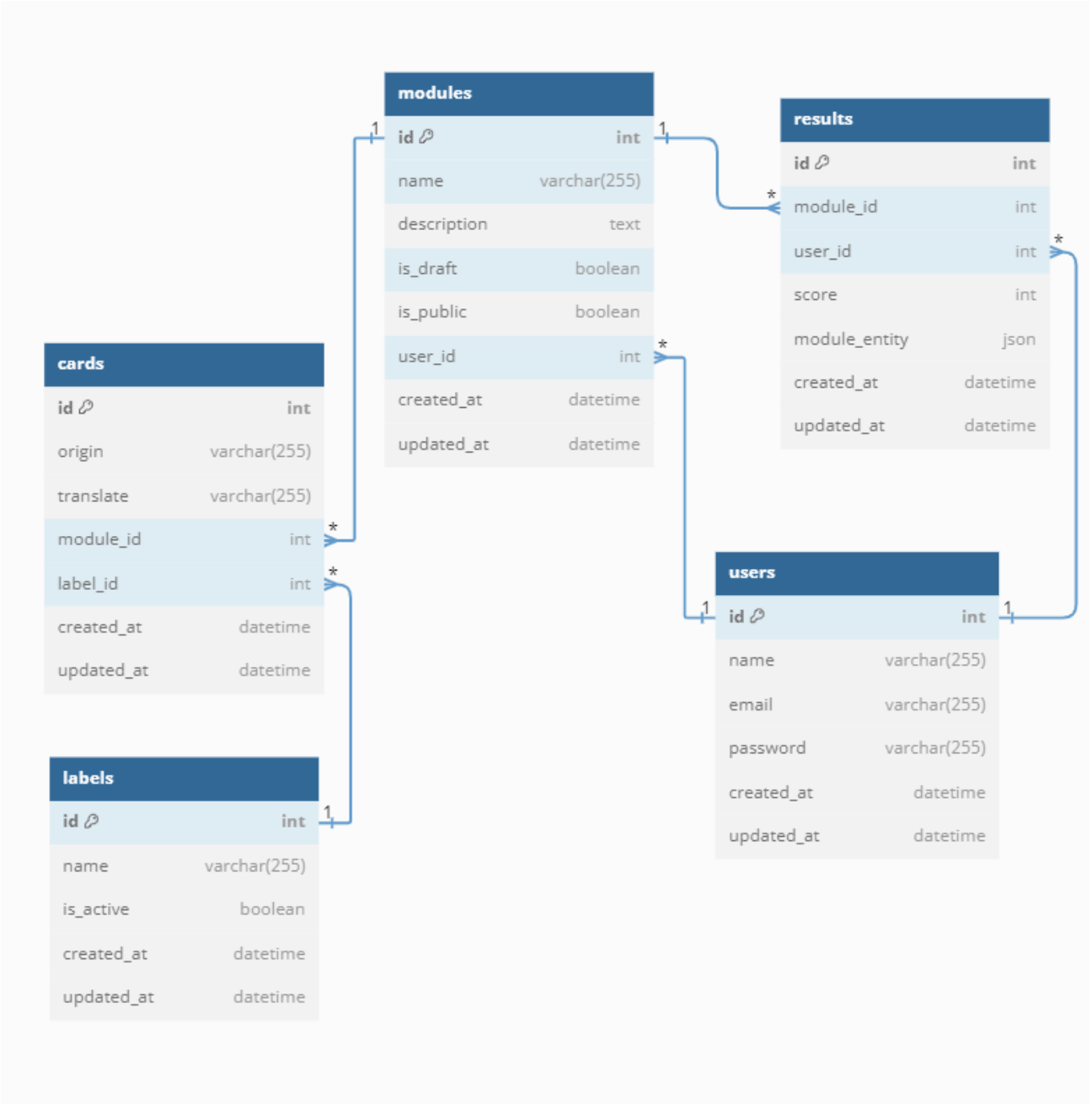

Рисунок 2.1 – ER модель бази даних

Такий підхід до структурування бази даних забезпечує гнучкість у використанні даних і створює основу для подальшого розвитку і розширення функціональності додатку.

2.3 Створення логотипу

Для додатку "DictionaryUp" було вирішено створити професійний логотип, що відображає суть та місію проєкту. Для цього використовували сервіс LOGO.com (див. рис. 2.2) , який є потужним інструментом для швидкого та якісного створення логотипів за допомогою штучного інтелекту [26]. LOGO.com надає простий та інтуїтивно зрозумілий процес створення логотипу. Для початку потрібно ввести назву компанії, слоган та обрати сферу діяльності. Після цього сервіс генерує кілька варіантів логотипів, які можна переглядати та налаштовувати відповідно до власних потреб.

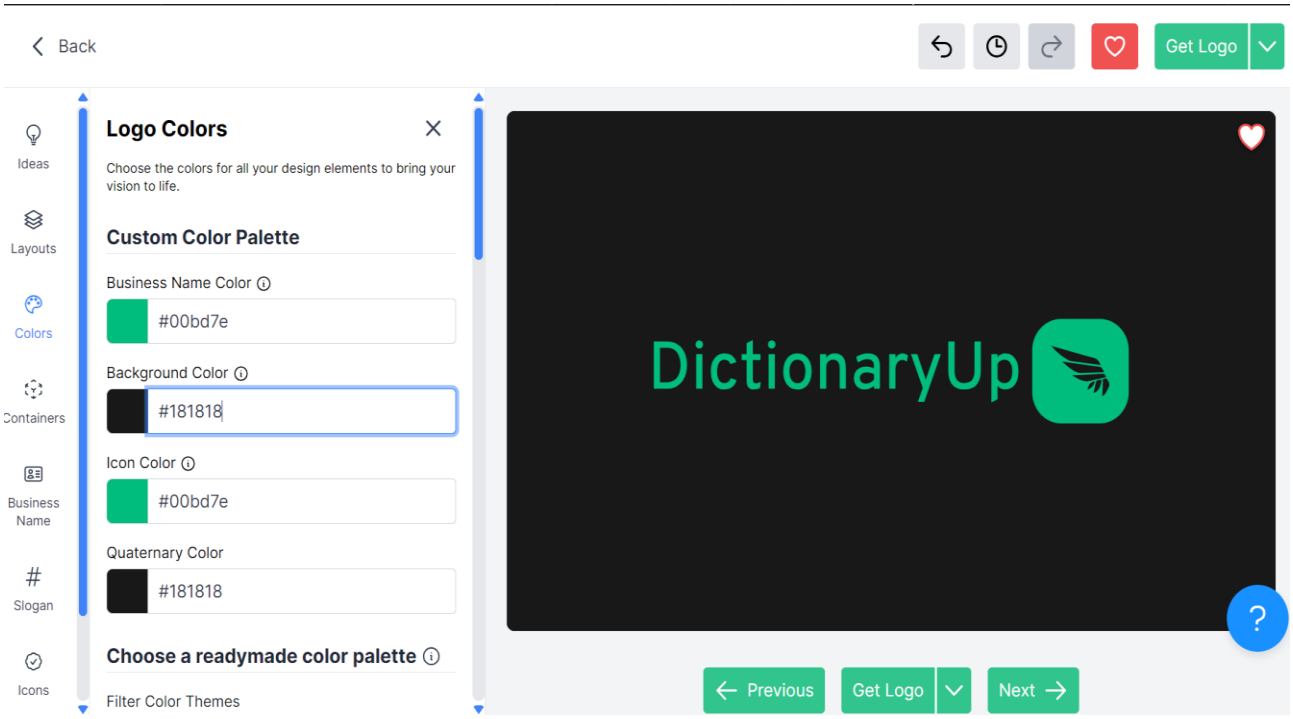

Рисунок 2.2 – Інтерфейс користувача у програмі LOGO.com

Переваги використання LOGO.com:

- Швидкість та простота – процес створення логотипу займає лише кілька хвилин завдяки інтуїтивно зрозумілому інтерфейсу та потужним інструментам налаштування;

- Інтуїтивне налаштування – можливість налаштувати всі аспекти логотипу, включаючи шрифти, кольори та іконки, дозволяє створити унікальний дизайн, що відповідає вимогам проекту;
- Висока якість – генеровані логотипи мають високу роздільну здатність і готові до використання в різних форматах (PNG, SVG, JPG).
- Економія ресурсів – використання AI-сервісу для створення логотипу є значно дешевшим і швидшим рішенням порівняно з найманням професійного дизайнера;

Для додатку "DictionaryUp" було створено два варіанти логотипу:

- Скорочений варіант (див. рис. 2.3), який буде використовуватися в шапці додатку. Він включає символ темного крила на зеленому фоні.
- Повний варіант (див. рис. 2.4), який буде розміщено на головній сторінці сайту. Він також використовує символ темного крила на зеленому фоні, але має більше деталей і текстових елементів.

Основні кольори бренду були обрані таким чином:

- Основний колір бренду: Зелений (#00bd7e);
- Основний колір фону: Темний (#181818);

Рисунок 2.3 – Скорочений варіант логотипу

Рисунок 2.4 – Повний варіант логотипу

Основні кольори бренду були обрані таким чином:

- Основний колір бренду: Зелений (#00bd7e);
- Основний колір фону: Темний (#181818).

Вибрані кольори використані не тільки в логотипі, а й будуть основними при розробці клієнтської частини програми. Завдяки ним, "DictionaryUp" тепер має впізнаваний і привабливий бренд, що підсилює його візуальну ідентичність та робить його привабливим для користувачів.

2.4 Проєктування інтерфейсу користувача та веб сторінок

Структура даного сайту буде стандартною для більшості застосунків, написаних на Vue.js. Це означає, що в корені проекту відпрацьовує головний кореневий компонент, який служить відправною точкою для всіх інших компонентів, і саме з нього визначається структура нашого сайту. Лістинг компонента наведено в додатку А.

Існує три основні типи моделей встановлення зв’язків між сторінками:

- лінійна;
- ієрархічна;
- довільна.

З вищезазначених моделей, ми розуміємо, що на даному проекті структура є довільною з ієрархічними елементами [27]. Це пов’язано з тим, що

основна роль шапки сторінки – це навігація. Весь її вміст пов’язаний із навігацією, і знаходиться на всіх сторінках, окрім головної сторінки, реєстрації та авторизації. Звідси отримуємо висновок, що з будь-якої сторінки є можливість перейти на майже будь-яку іншу сторінку. Така властивість притаманна більшості веб сайтів в Інтернеті.

Для взаємодії з додатком користувачі повинні бути поділені на авторизованих та неавторизованих. Це реалізується за допомогою Vue Router, який є найпоширенішим засобом навігації в екосистемі Vue.js. Vue Router монополізує навігацію, дозволяючи легко організувати маршрутизацію в додатку, опрацьовуючи доступ до певних сторінок на рівні роутів та забезпечуючи управління станом URL [28].

Відкриті сторінки включають головну сторінку, сторінку "Про Нас", авторизацію та реєстрацію. Головна сторінка служить першою сторінкою, яку бачить користувач. Вона містить заголовок і кнопку в центрі екрану, яка при натисканні перенаправляє користувача на форму авторизації. Заголовок описує основну функцію додатку і запрошує користувача приєднатися до платформи. Сторінка "Про Нас" надає інформацію про мету проекту та контактну інформацію, забезпечуючи загальний огляд проекту. Сторінка авторизації дозволяє користувачам увійти в систему, вводячи електронну пошту і пароль, надаючи доступ до закритих сторінок. Сторінка реєстрації дозволяє новим користувачам створити обліковий запис, вводячи необхідні дані для доступу до платформи.

Закриті сторінки доступні тільки для авторизованих користувачів і включають сторінку бібліотеки, створення модуля, перегляду модуля та проходження тесту. На сторінці бібліотеки відображаються всі користувацькі модулі. Користувачі можуть переглядати, редагувати, запускати тести та шукати модулі. Меню на цій сторінці містить кнопки для різних дій, таких як перегляд, редагування і запуск тестів, а пошуковий рядок дозволяє знаходити потрібні модулі.

Сторінка створення модуля дозволяє користувачам створити новий модуль, вводячи назву та опис, а також додаючи картки зі словами. Картки містять оригінальний текст іноземною мовою та переклад на українську. Кожна картка має функції видалення та автоперекладу, що дозволяє користувачам вводити текст англійською мовою і отримувати переклад за допомогою стороннього сервісу.

Сторінка перегляду модуля дозволяє користувачам переглядати всі картки у модулі, редагувати їх за потреби та переглядати результати проходження модулів. Це забезпечує можливість оновлення інформації та додавання нових прикладів використання слів.

На сторінці проходження тесту система показує випадкові картки, і користувач повинен ввести переклад. Після завершення тесту система обчислює результат і показує його користувачу, дозволяючи оцінити рівень знань і прогрес у вивченні слів.

На веб сторінках будуть форми введення даних, і всі форми та елементи форм будуть реалізовані за допомогою готових компонентів з Ant Design. Цей вибір обумовлений тим, що бібліотека чудово працює з Vue.js. Готові Компоненти забезпечують високу якість та зручність у використанні, що дозволяє створювати інтерактивні та функціональні форми без зайвих зусиль.

Використання Ant Design для форм має кілька переваг:

- Зручне опрацювання стану форм: компоненти підтримують реактивне управління станом форм, що дозволяє легко відстежувати та оновлювати дані форм у реальному часі;
- Вбудована валідація: бібліотека надає широкий спектр вбудованих засобів валідації елементів форми, що дозволяє легко реалізувати перевірку правильності введених даних без необхідності написання додаткового коду;
- Великий вибір компонентів: Бібліотека Ant Design включає багато готових компонентів для створення форм, таких як текстові поля, випадаючі списки, кнопки, чекбокси та інші елементи інтерфейсу.

Всі інші компоненти, включаючи компоненти макету та бізнес-логіки, будуть створюватися вручну. Це дозволить налаштувати їх відповідно до специфічних вимог проекту і забезпечити необхідну функціональність.

Така структура сайту забезпечує зручність у використанні, чітку навігацію та доступ до необхідних функцій для всіх користувачів. З використанням готових бібліотек організація коду стає простішою та ефективнішою, що дозволяє за короткий час створити інтуїтивно зрозумілий інтерфейс для взаємодії з додатком.

2.5 Проєктування серверної частини

Серверна частина "DictionaryUp" відповідає за обробку запитів від клієнтської частини, взаємодію з базою даних, а також за забезпечення безпеки та цілісності даних. Для реалізації серверної частини використовується Node.js, Express та TypeScript. Node.js забезпечує високу продуктивність та ефективність обробки запитів завдяки асинхронній обробці, тоді як Express надає простий і зручний API для створення серверних додатків. Використання TypeScript додає строгий контроль типів, що допомагає уникати багатьох помилок під час розробки, забезпечуючи більш стабільний і надійний код.

Контролери відповідають за бізнес-логіку додатку. Вони обробляють запити від клієнтської частини та взаємодіють із базою даних через моделі, визначені за допомогою TypeORM. Майже всі маршрути у програмі це стандартний набір CRUD операцій для роботи з різними сутностями, такими як модулі, картки тощо. Контролери забезпечують обробку створення, читання, оновлення та видалення даних, а також виконують додаткову бізнес-логіку, необхідну для функціонування додатку (див. рис. 2.5).

Авторизація здійснюється за допомогою JWT. При вході в систему користувач отримує токен, який використовується для доступу до захищених маршрутів. На кожен запит, який вимагає авторизацію, спрацьовує middleware,

що перевіряє валідність токена. Якщо токен валідний, запит обробляється далі, інакше користувач отримує помилку доступу. Middleware відіграють ключову роль у забезпеченні безпеки, оскільки вони перехоплюють запити і перевіряють права доступу до ресурсів.

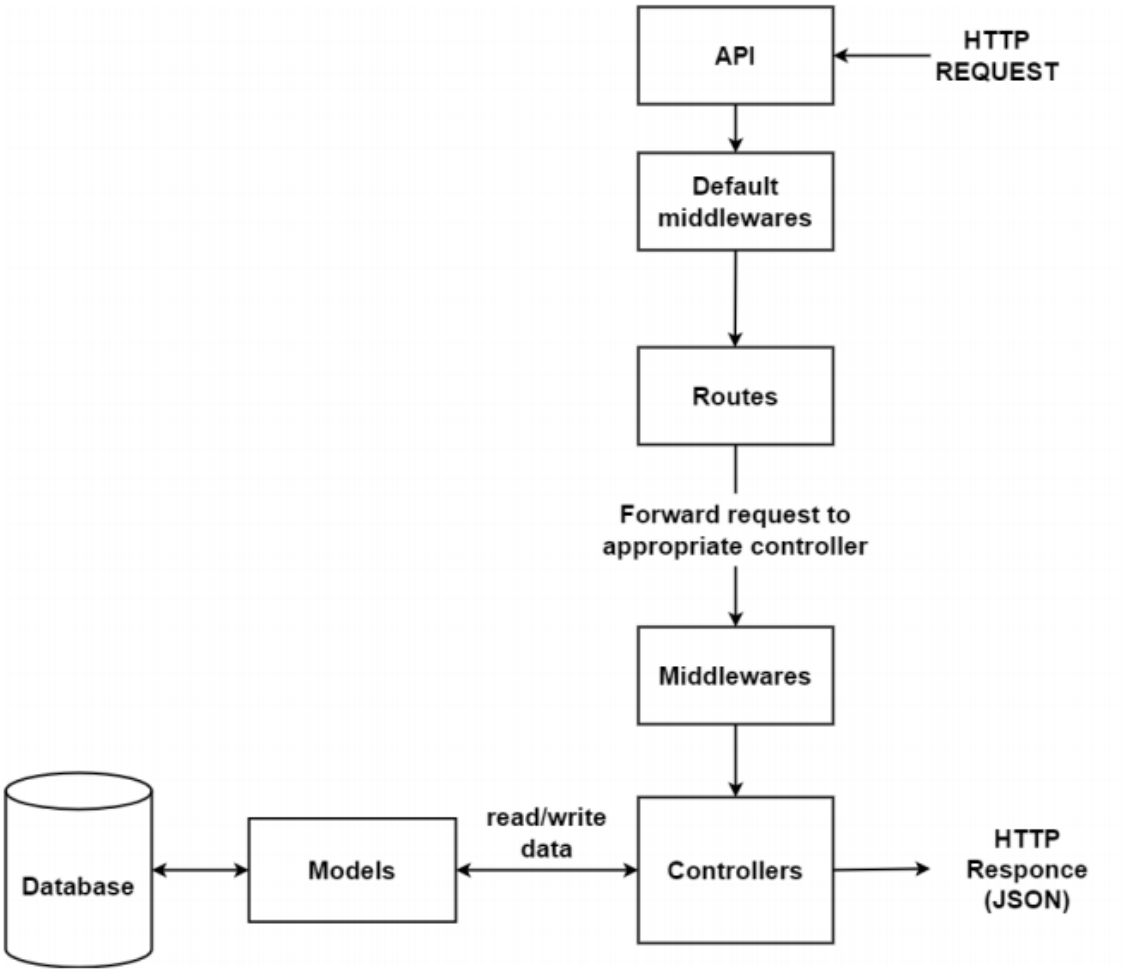

Рисунок 2.5 – Потік даних на сервері від запиту до відповіді

Структура проєкту організована таким чином, що сутності таблиць з бази даних зберігаються в окремих папках у корені проекту. Ці сутності описані за допомогою TypeORM, який є ORM бібліотекою для TypeScript і JavaScript. Вона дозволяє визначати сутності та їхні зв'язки у вигляді класів, що спрощує роботу з базою даних.

Кожна сутність відповідає таблиці в базі даних і описує її структуру, включаючи типи даних та зв'язки між таблицями. Використання TypeORM

забезпечує типізацію всіх сутностей, що допомагає уникати помилок при вставленні даних або редагуванні.

Для управління змінами у структурі бази даних використовуються міграції, які також описані засобами використовуваної ORM. Міграції дозволяють автоматично застосовувати зміни до бази даних, такі як створення нових таблиць, додавання або зміна стовпців, видалення таблиць тощо. Використання міграцій допомагає підтримувати цілісність бази даних і полегшує розгортання змін у різних середовищах, розробницьких чи продакш.

Цей підхід забезпечує структурованість та модульність коду, що дозволяє створити надійний і масштабований додаток, що відповідає всім вимогам сучасного веб-розробки.

2.6 Використання AI для перекладу

У проєкті, основна робота якого пов’язана із перекладом, потрібно додати функцію для перекладу. Для цього використовуємо DeepL [29].

DeepL – це один з провідних онлайн-перекладачів, який використовує штучний інтелект для забезпечення високої точності перекладів. Він відомий своєю здатністю розуміти контекст і надавати переклади, які є не лише граматично правильними, але й природними та зрозумілими для носіїв мови. Він підтримує багато мов і пропонує додаткові функції, такі як налаштування тональності та стилю перекладу.

Коли користувач створює картку і вводить текст, у правій верхній частині екрану з'являється кнопка перекладу. Натискаючи на цю кнопку, програма відправляє запит на сервер, який використовуючи відповідний API ключ робить запит до DeepL API і повертає переклад. Це дозволяє користувачам швидко отримати переклад введеного тексту без необхідності самостійно шукати переклад вручну. Основні переваги використання DeepL для автоматичного перекладу полягають у його здатності розуміти контекст

речення, що дозволяє забезпечити більш точні і природні переклади порівняно з іншими сервісами. Завдяки глибоким нейронним мережам, інструмент здатен враховувати контекст всього тексту, що перекладається, а не лише окремих слів або фраз, що значно покращує якість перекладу.

Однією з головних переваг його над Google Translate API є простота підключення та використання. Для початку роботи з DeepL достатньо зареєструватися і отримати свій API ключ. У той час як Google Translate API вимагає створення проекту в Google Cloud, що включає більше кроків, таких як налаштування доступів, дозволів та інших параметрів з'єднання може бути складніше для нових користувачів. Крім того, DeepL надає безкоштовний доступ з певними обмеженнями, які важко досягти у звичайному використанні, що робить його більш доступним для широкого кола користувачів. Використання DeepL для автоматичного перекладу в додатку забезпечує легкість отримання перекладу, покращуючи загальний досвід використання додатку.

2.7 Висновок до другого розділу

В другому розділі кваліфікаційної роботи було детально описано та спроектовано основні частини програми. Розробка структури додатку включала визначення компонентів клієнтської частини з використанням бібліотеки готових компонентів. Для роботи із базою даних використовується MySQL з TypeORM для опису сутностей і міграцій. Логотип створено за допомогою AI-сервісу LOGO.com, що дозволило швидко розробити професійний візуальний ідентифікатор для додатку. Серверна частина програми буде побудована засобами Node.js. Використання TypeORM забезпечує типізацію та надійність роботи з базою даних. Авторизація здійснюється за допомогою JWT. Функція автоматичного перекладу реалізована за допомогою DeepL, що забезпечує високоякісні переклади завдяки розумінню контексту нейромережею.

РОЗДІЛ 3. ПРАКТИЧНА ЧАСТИНА.

3.1 Ініціалізація клієнтської частини та розробка веб сторінок

У цьому розділі детально описується процес створення клієнтської частини програми на основі Vue.js, із використанням Vite для швидкої ініціалізації проєкту, Typescript для забезпечення типізації та надійності коду, Pinia для забезпечення зручного доступу до глобального стану додатку та Ant Design для використання готових компонентів тощо.

Першими створеними компонентами є основні кастомні елементи форми, такі як Input (див. додаток А) для введення користувацьких даних при взаємодії із програмою, Button (див. додаток Б) для кнопок та Typography для відображення текстових елементів і заголовків [30]. Ці компоненти формують основу для побудови більш складних інтерфейсів у додатку, забезпечуючи узгодженість та зручність використання.

Наступним кроком є налаштування глобального стейту для зберігання загальних даних у програмі. Це дозволяє ефективно керувати станом додатку та обмінюватися даними між різними компонентами.

Глобальний стейт реалізується за допомогою бібліотеки Pinia, в лістингу 3.1 наведенно створення глобального стейту для міток.

Лістинг 3.1 – Глобальний стейт міток

```
import { defineStore } from 'pinia';
      import { axiosInstance } from '@/plugins/axiosPlugin';
      export const useGlobalStore = defineStore('global', {
        state: () => ({ labels: [] }),
        getters: {
          getLabels: state => state.labels,
        },
        actions: {
          fetch() {
            axiosInstance.get('/store').then(res => {
              this.labels = res.data;
            });
```

```
    },
  },
});
```

Для здійснення запитів до сервера ми використовуємо Axios, який ініціалізується як Vue плагін (див. лістинг 3.2). Це дозволяє зберігати його у глобальному об'єкті Vue, що забезпечує легкий доступ до функцій Axios з будь-якого компонента додатку. Налаштування Axios включає базовий URL сервера та інші конфігурації для обробки запитів.

Лістинг 3.2 – Axios плагін Vue для запитів на сервер

```
import axios, { type AxiosInstance } from 'axios';
import type { App } from 'vue';
export const axiosInstance: AxiosInstance = axios.create({
  baseURL: 'http://localhost:3000',
});
export const axiosPlugin = {
  install: (app: App) => {
    app.config.globalProperties.$axios = axiosInstance;
    app.provide('axios', axiosInstance);
  },
};
export default axiosPlugin;
```

Для управління стилями у програмі створюється SCSS файл із загальними змінними стилів. Це дозволяє централізовано керувати стилями та забезпечувати єдиний вигляд і відчуття всіх компонентів у додатку. Змінні стилів можуть включати кольори, шрифти, розміри та інші параметри, які використовуються у різних частинах програми.

У головному файлі клієнтської частини програми всі ці елементи збираються разом (див. лістинг 3.3). Підключаються всі глобальні функції та компоненти, ініціалізується глобальний стейт та налаштовується Axios.

Лістинг 3.3 – Конфігурування глобального об’єкту Vue

```
import './assets/main.css';
```

```
import { createApp } from 'vue';
import { createPinia } from 'pinia';
import 'vue-toast-notification/dist/theme-default.css';
import { axiosPlugin } from './plugins/axiosPlugin';
import App from './App.vue';
import router from './router';
import Button from './components/Button.vue';
import AppLogo from './icons/AppLogo.vue';
import Typography from './components/Typography.vue';
import Input from './components/Input.vue';
import { useGlobalStore } from './stores/global';
const app = createApp(App);
app.use(createPinia());
app.use(router);
app.use(axiosPlugin);
app.component('Button', Button);
app.component('Logo', AppLogo);
app.component('Typography', Typography);
app.component('Input', Input);
const globalStore = useGlobalStore();
globalStore.fetch();
app.mount('#app');
```

Це забезпечує єдину точку входу для програми та гарантує, що всі необхідні залежності та налаштування доступні з самого початку, що робить процес розробки більш організованим і ефективним.

Створення веб-сторінок у додатку "DictionaryUp" базується на єдиній структурі, що використовує одні й ті самі компоненти. Це робить процес розробки макету значно швидшим та ефективнішим, оскільки компоненти можуть бути повторно використані в різних частинах програми. Використання єдиної структури забезпечує узгодженість у вигляді та функціональності всіх сторінок, що сприяє покращенню користувацького досвіду.

Найважливішими сторінками у додатку є сторінка створення модуля (див. рис. 3.1), перегляду модулів (див. рис. 3.2) та проходження тесту (див. рис. 3.3). Результати проходження модуля будуть відображені на сторінці проходження модуля одразу після його завершення (див. рис. 3.4).

Сторінка створення модуля дозволяє користувачам вводити назву та опис модуля, додавати нові картки з оригінальним текстом і перекладом, а також використовувати функції видалення карток та автоперекладу.

Ця сторінка забезпечує зручний інтерфейс для користувачів, щоб вони могли легко створювати нові навчальні модулі.

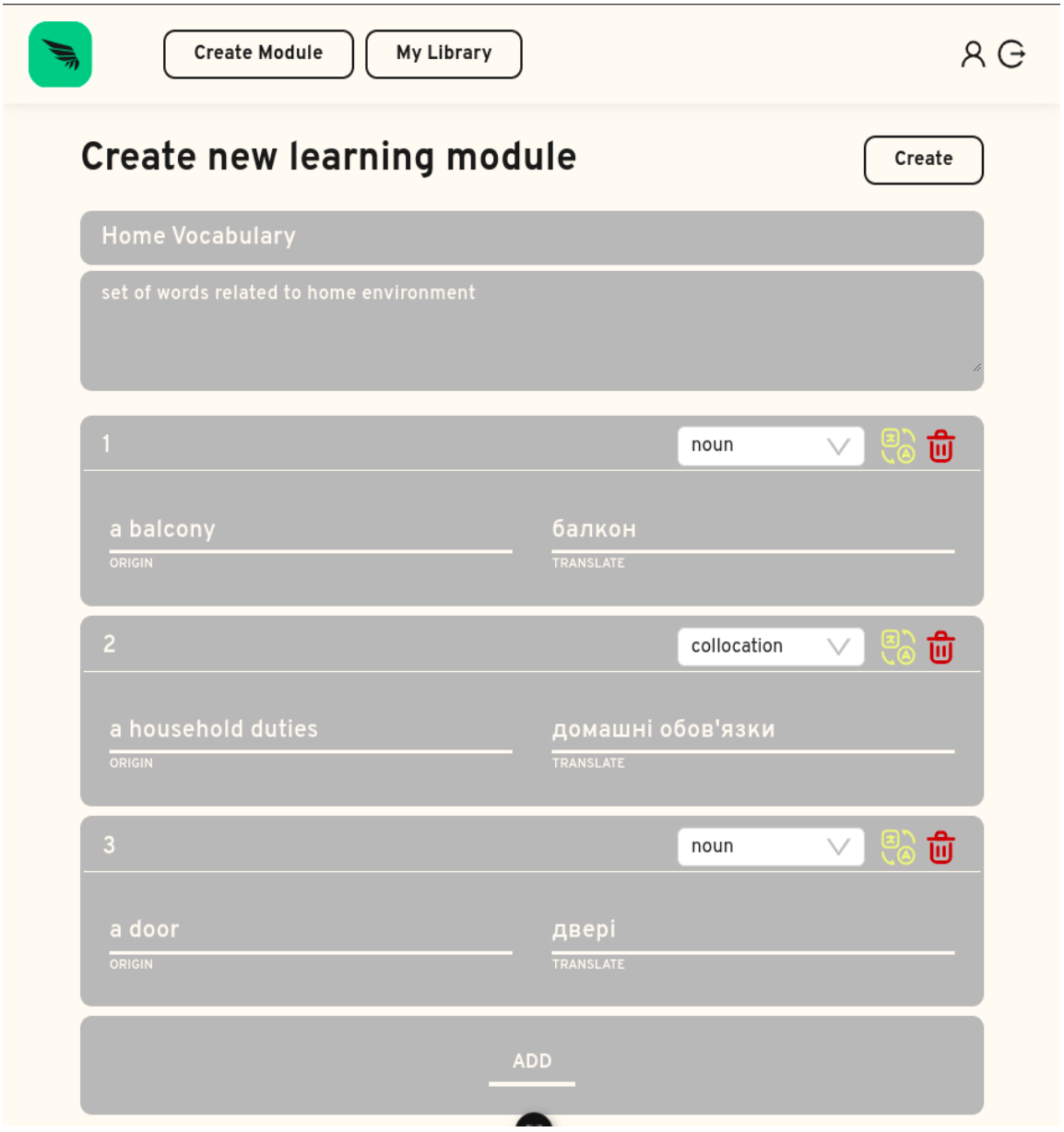

Рисунок 3.1 – Сторінка створення модуля

Сторінка перегляду модулів дозволяє користувачам переглядати всі створені ними модулі, редагувати існуючі картки та переглядати результати проходження тестів.

Ця сторінка надає зручні інструменти для управління та організації навчального контенту.

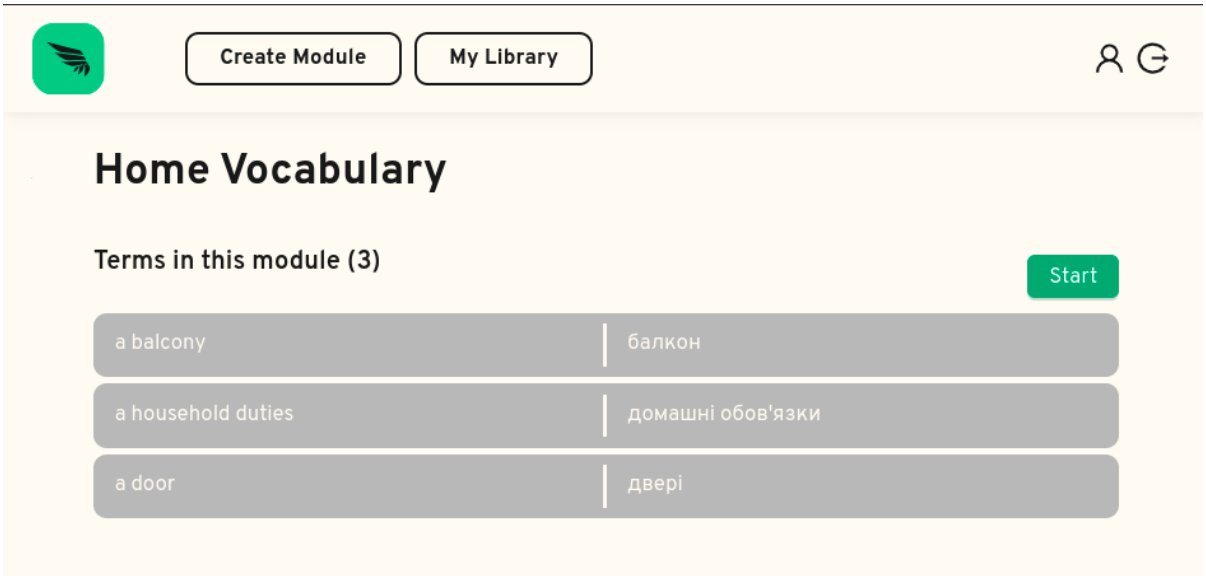

Рисунок 3.2 – Сторінка перегляду модулів

Сторінка проходження тесту надає користувачам можливість перевірити свої знання, відповідаючи на випадкові питання, що з'являються на основі карток з обраного модуля. Після завершення тесту система обчислює результат і відображає його користувачу, що дозволяє оцінити прогрес у вивченні.

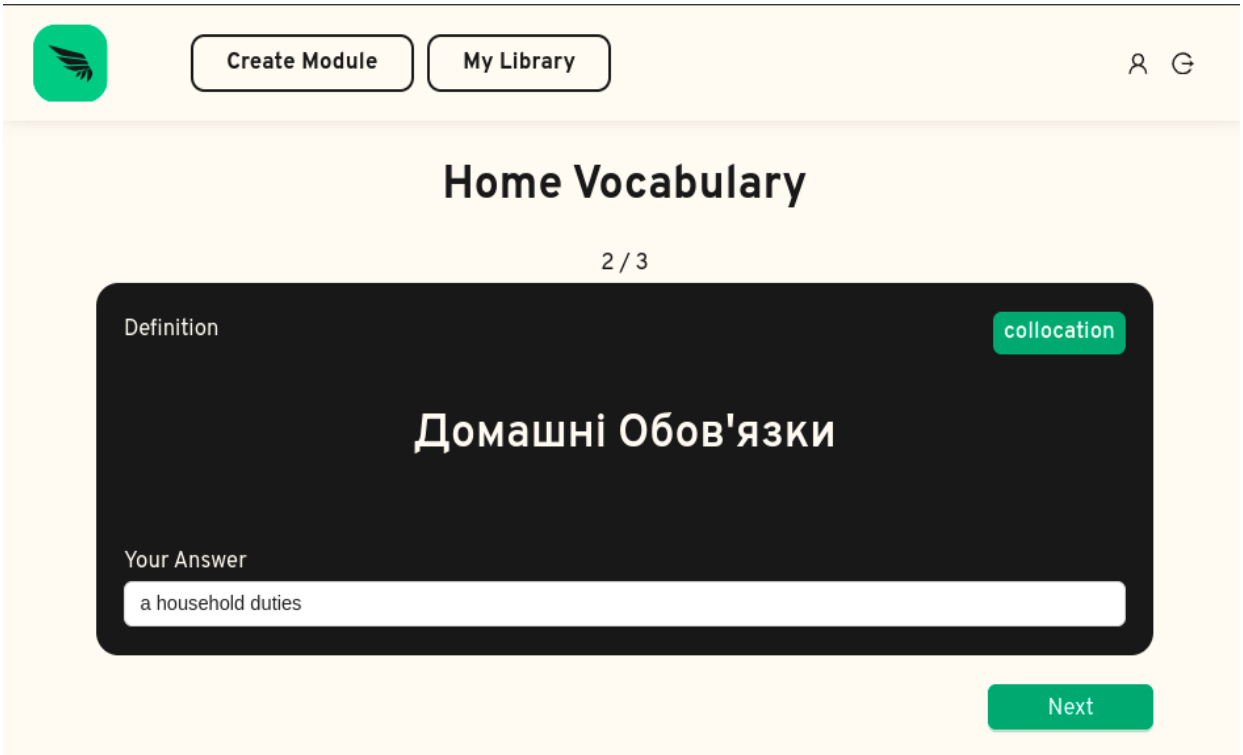

Рисунок 3.3 – Сторінка проходження тесту

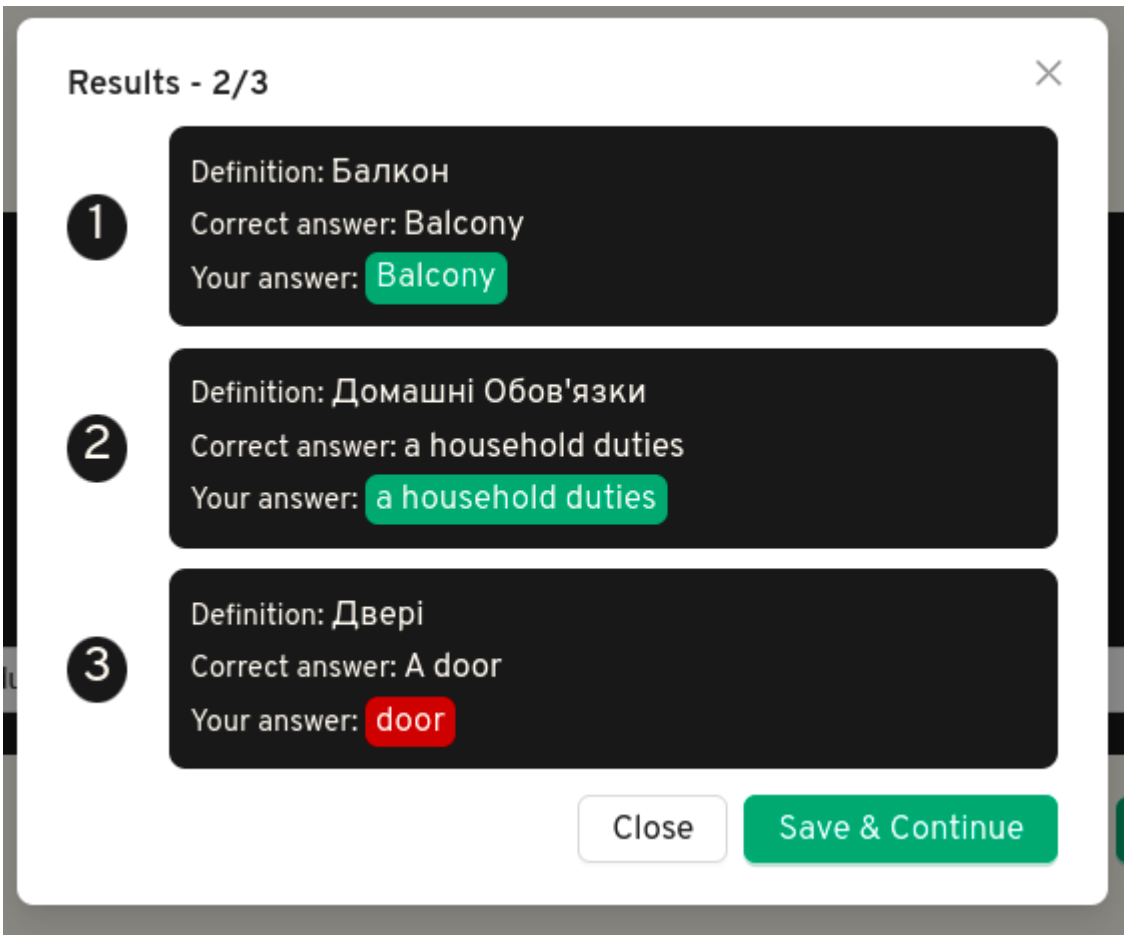

Рисунок 3.4 – Модальне вікно із результатами проходження модуля

Всі ці сторінки використовують спільні компоненти, такі як форми введення даних, кнопки, заголовки та інші елементи інтерфейсу, що забезпечує єдність дизайну та функціональності. Це дозволяє розробникам створювати нові сторінки швидко і безпомилково, забезпечуючи високу якість та зручність використання додатку "DictionaryUp".

3.2 Розробка серверної частини

Для реалізації серверної частини програми "DictionaryUp" використовуються кілька важливих бібліотек. Спочатку встановлюються всі необхідні залежності, що забезпечують функціональність та ефективність роботи серверу:

1. express – фреймворк для Node.js, який забезпечує простий і гнучкий спосіб створення серверних додатків і API. Express є дуже популярним завдяки своєму простому API та великій кількості плагінів, які полегшують розробку.

2. cors – бібліотека використовується для дозволу CORS (Cross-Origin Resource Sharing) запитів. Вона необхідна для того, щоб сервер міг обробляти запити з різних доменів, що є важливим для веб-додатків, які взаємодіють з API на іншому сервері.

3. dotenv – бібліотека дозволяє зберігати налаштування середовища в .env файлі. Це зручно для конфігурації проєкту, оскільки всі важливі параметри, такі як налаштування бази даних або секретні ключі, можуть бути зберігатися в одному місці.

4. tsx – інструмент забезпечує підтримку TypeScript для Node.js, дозволяючи використовувати TypeScript у проєктах Node.js без необхідності попередньої компіляції.

5. typeorm / mysql2 – це TypeScript засоби, що дозволяє працювати з базами даних у стилі об'єктно-орієнтованого програмування. MySQL2 – це клієнт для роботи з базою даних MySQL. Разом вони забезпечують ефективну роботу з базою даних, включаючи створення міграцій та моделей.

6. reflect-metadata – ця бібліотека необхідна для роботи з TypeORM, оскільки вона забезпечує підтримку метаданих, які використовуються для визначення типів даних у моделях.

7. deepl-node – це клієнтська бібліотека для роботи з API DeepL, що забезпечує автоматичний переклад текстів. Вона використовується для реалізації функції перекладу карток у додатку.

Після встановлення залежностей, налаштовується TypeScript під користувацькі потреби для розробника [31]. Основні правила, які необхідно додати – це підтримку синтаксису останніх версій мови програмування та імпортів. Конфігурацію TypeScript наведено у лістингу 3.5.

Лістинг 3.4 – TypeScript конфігурація проєкту

```
{
  "compilerOptions": {
    "target": "ESNext",
    "module": "ESNext",
    "moduleResolution": "node",
    "emitDecoratorMetadata": true,
    "experimentalDecorators": true,
    "esModuleInterop": true,
    "allowSyntheticDefaultImports": true,
    "allowJs": true,
    "resolveJsonModule": true,
    "isolatedModules": true,
    "skipLibCheck": true,
    "outDir": "./dist",
    "baseUrl": "."
  },
  "include": ["**/*"],
  "exclude": ["node_modules"]
}
```

Наступним кроком створюється під'єднання до бази даних та виконується налаштування міграцій. Міграції дозволяють автоматично застосовувати зміни до структури бази даних, такі як створення нових таблиць або зміна існуючих. Це забезпечує зручне управління структурою бази даних та підтримку її цілісності. Наступним кроком є опис моделей, що відповідають сутностям у базі даних.

Модель таблиці модулів наведено у додатку В. Моделі визначають структуру таблиць і зв'язки між ними, що полегшує роботу з даними на рівні коду. Далі ініціалізується сервер та налаштовуються маршрути за допомогою Express.

Код ініціалізації серверу наведено у додатку Г. До всіх маршрутів, окрім авторизаційних, додається middleware для перевірки авторизації користувача. Це забезпечує захист закритих ресурсів і гарантує, що до них мають доступ тільки авторизовані користувачі. У програмі є два основні набори маршрутів та контролерів, які забезпечують CRUD операції для модулів та карток. Функція, яка відповідає за вибір модулів із бази даних, наведено в лістингу 3.5.

Лістинг 3.5 – Контролер, який робить вибірку модулів із бази даних та повертає їх користувачеві

```
export async function getAll(req: Request, res: Response) {
       try {
         const result = await modulesRepo
           .createQueryBuilder('modules')
           .leftJoin('modules.cards', 'cards')
           .select([
             'modules.id as id',
             'modules.name as name',
             'modules.created_at as created_at',
           ])
           .addSelect('COUNT(cards.id)', 'cardsCount')
           .groupBy('modules.id')
           .orderBy('modules.created_at', 'DESC')
           .getRawMany();

         res.status(201).json(result);
       } catch (err) {
         console.log('Errror $$$', err);
         res.status(500).json({
           message: 'Internal Server Error!',
           error: err,
         });
       }
     }
```

Функція автоматичного перекладу у додатку забезпечується завдяки використанню Deepl API. API ключ для доступу до Deepl зберігається у змінних оточення, що забезпечує безпеку та зручність управління конфіденційними даними. У коді створюється інстанс перекладача, який використовується як сервіс під час звернення до функції перекладу. Це дозволяє легко інтегрувати функціонал перекладу в різні частини додатку, забезпечуючи зручний і швидкий спосіб отримання перекладів для користувачів.

Процес включає наступні кроки:

1. Збереження API ключа Deepl у змінних оточення.

2. Створення інстансу перекладача у коді (див. лістинг 3.6).

3. Виклик перекладача як сервісу під час звернення на переклад (див. лістинг 3.7).

Лістинг 3.6 – Інстанс перекладача Deepl Api

```
import * as deepl from 'deepl-node';
     const DEEPL_API_KEY = process.env.DEEPL_API_KEY || '';
     const translator = new deepl.Translator(DEEPL_API_KEY, {
       maxRetries: 2,
     });
     export default translator;
```

Лістинг 3.7 – Функція, яка опрацьовує запит на переклад

```
export async function translateCard(req: Request, res: Response) {
  try {
    const origin: string = req.body.origin;
    const result = await translator.translateText(origin, 'en',
'uk');
    res.status(200).json(result);
  } catch (err) {
    console.log('Errror $$$', err);
    res.status(500).json({
      message: 'Internal Server Error!',
      error: err,
    });
  }
}
```

Ці маршрути забезпечують можливість створення, читання, оновлення та видалення модулів і карток, що є основною функціональністю "DictionaryUp".

3.3 Тестування та впровадження

Валідація та тестування є критично важливими етапами розробки, що забезпечують надійність та якість роботи додатку. Потрібно дотримуватися найкращих практик і стандартів під час розробки, як серверної частини, так і клієнтської. Під час розробки використано Docker, що дозволило створити незалежне середовище для кожного компонента. Це забезпечує ізоляцію залежностей та конфігурацій, що значно спрощує процес розробки, тестування та розгортання додатку. Використання Docker дозволяє швидко налаштувати середовище розробки на будь-якому комп'ютері, забезпечуючи консистентність

та стабільність роботи додатку в різних середовищах. Конфігурацію середовища наведено у додатку Ґ.

Для клієнтської частини забезпечено чисту архітектуру, де код структурований і розділений на логічні компоненти. У консолі браузера не повинно бути жодних помилок або попереджень. Усі форми та інші елементи інтерфейсу мають інтегровану систему обробки помилок користувачів, що включає валідацію введених даних та повідомлення про некоректні дії. Окрім цього, реалізована система сповіщень, яка інформує користувачів про успішні операції (див. рис. 3.5) або виникнення проблем (див. рис. 3.6) через спливаючі повідомлення або модальні вікна.

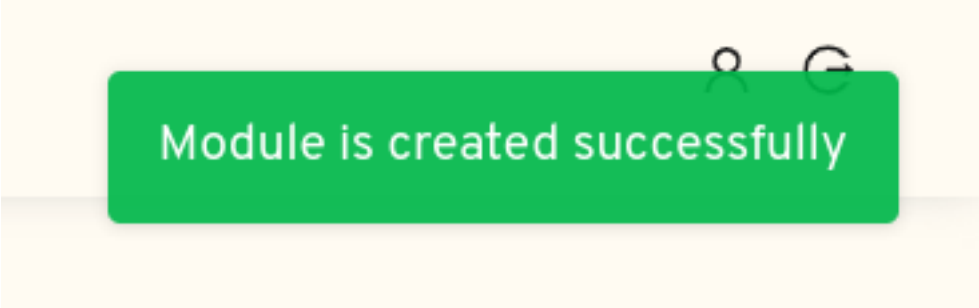

Рисунок 3.5 – Сповіщення про успішну операцію

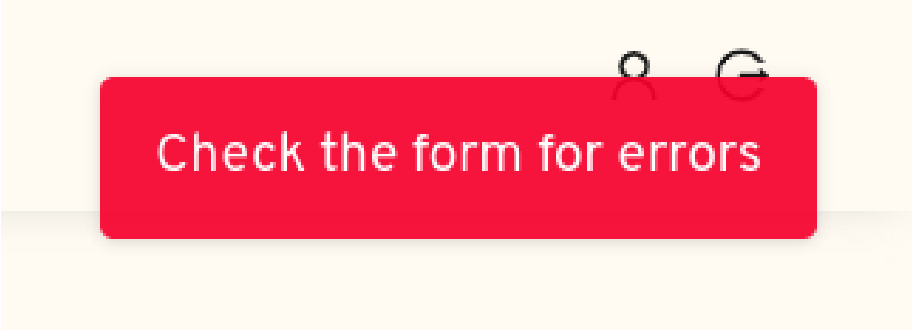

Рисунок 3.6 – Сповіщення про виникнення помилки

У серверній частині забезпечено правильну структуру відповідей для всіх маршрутів, що взаємодіють з клієнтською частиною. Всі помилки

обробляються належним чином: сервер не переривається свою роботу при їх виникненні, а записи про помилки ведуться з відповідними повідомленнями відправляються клієнту. Це підтримує стабільність системи і дозволяє користувачам отримувати необхідну інформацію про статус їхніх запитів. Для забезпечення надійності, всі маршрути ретельно тестуються за допомогою інструменту Postman прямо в процесі розробки. Це дозволяє перевірити коректність роботи API, валідацію введених даних та обробку помилок у різних сценаріях.

Завдяки дотриманню цих принципів, забезпечуєтсья висока якість розроблюваного продукту, мінімізацію помилок та покращення загального користувацького досвіду в додатку "DictionaryUp".

Після завершення розробки веб-додатку, необхідно здійснити його реліз і публікацію в Інтернеті, щоб забезпечити доступність для користувачів. Для цього чудово підійде DigitalOcean, який надає розподілені сервери, що забезпечують високу продуктивність і надійність.

Для першого тестового релізу нам знадобиться два сервери. Один сервер буде використовуватися для запуску самої програми, а інший - для хостингу бази даних. Це розділення дозволить ефективніше управляти ресурсами та забезпечить кращу продуктивність і безпеку. Налаштування включає встановлення всіх необхідних залежностей, таких як середовище виконання для Node.js, інструменти для управління пакетами, серверна конфігурація для Express та налаштування бази даних на другому сервері. Після завершення налаштувань, додаток буде готовий до публікації та доступний для користувачів. Таким чином, використання DigitalOcean дозволить забезпечити стабільну та ефективну роботу веб-додатку, роблячи його доступним для широкої аудиторії.

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Надзвичайні ситуації екологічного характеру.

Надзвичайна екологічна ситуація – це серйозне порушення природного стану навколишнього середовища на певній території. Вона потребує застосування особливих заходів з боку держави для усунення наслідків та відновлення природного балансу [32]. Надзвичайні ситуації можуть виникати через природні явища, такі як землетруси, повені, пожежі, а також антропогенні фактори, як-от промислові аварії, забруднення води і повітря, нераціональне використання природних ресурсів. Основні види надзвичайних ситуацій екологічного характеру:

1. Надзвичайна (напружена) ситуація характеризується порушенням функціонування екосистеми, перевищенням гранично допустимих концентрацій (ГДК) шкідливих речовин, зниженням біомаси та продуктивності екосистеми;

2. Кризова ситуація, яка спричиняє значне руйнування складових екосистеми, існує реальна небезпека виходу ситуації з-під контролю, що може потребувати термінового втручання;

3. Катастрофічна ситуація, яка є найбільш серйозною, оскільки викликає глибокі, незворотні зміни в природному середовищі, що веде до втрати природних ресурсів і значного погіршення умов життя населення, створені певною екологічною обстановкою на території різного ступеня безпечності.

Зони обмеженої екологічної безпеки характеризуються короткочасними негативними змінами.

Зони підвищеної екологічної небезпеки включають довготривалі негативні зміни, що можуть суттєво впливати на навколишнє середовище і здоров'я населення.

Зони екологічної катастрофи спостерігають стійкі незворотні зміни, які можуть вимагати евакуації населення та заборони будь-якої господарської діяльності. Критерії оголошення надзвичайної екологічної ситуації:

1. Значне перевищення гранично допустимих норм якості навколишнього середовища;

2. Виникнення реальної загрози життю та здоров'ю великої кількості людей;

3. Значна матеріальна шкода юридичним та фізичним особам або навколишньому природному середовищу;

4. Негативні зміни, які неможливо усунути без застосування надзвичайних заходів;

5. Суттєве обмеження або виключення можливості проживання населення та провадження господарської діяльності.

Законодавче регулювання надзвичайних екологічних ситуацій в Україні забезпечується відповідними законами та нормативними актами, які визначають порядок оголошення надзвичайного стану, заходи з ліквідації наслідків та відновлення постраждалих територій [33]. Надзвичайні екологічні ситуації можуть мати різноманітні причини, включаючи природні явища, такі як землетруси, повені, урагани, а також техногенні фактори, як-от аварії на промислових підприємствах, транспорті, хімічне та радіаційне забруднення.

Прогнозування та моніторинг надзвичайних екологічних ситуацій є важливою складовою запобігання їх виникненню та мінімізації наслідків. Це включає використання сучасних технологій для оцінки ризиків, моделювання можливих сценаріїв розвитку ситуацій та розробку ефективних стратегій реагування. Управління надзвичайними ситуаціями екологічного характеру потребує чіткої координації між різними державними та місцевими органами, підприємствами і громадськими організаціями. Важливо забезпечити оперативну взаємодію, обмін інформацією та залучення всіх доступних ресурсів для ефективного реагування на кризи.

4.2 Пожежна профілактика на дільниці (в цеху).

Пожежна профілактика на дільниці (в цеху) включає комплекс заходів, спрямованих на запобігання виникненню пожеж та мінімізацію їхніх наслідків.

Одним із ключових аспектів є захист виробничих приміщень від перевантаження горючими речовинами, що передбачає встановлення гранично допустимих норм горючого навантаження приміщень [34]. Це враховує продуктивність обладнання та організацію технологічного процесу. Важливим є також постійний контроль та зменшення кількості горючих відходів, що накопичуються під час виробництва. Це досягається шляхом впровадження технологічних рішень, які дозволяють мінімізувати утворення відходів або використовувати їх повторно.

Для підвищення пожежної безпеки можливо замінити використовувані горючі речовини на менш горючі або негорючі аналоги, що знижує ймовірність виникнення пожежі та зменшує кількість горючих матеріалів у виробництві. Під час нормальної експлуатації важливо обмежити кількість горючих речовин, що перебувають у приміщенні, шляхом обмеження обсягів виробів, рідин та твердих речовин, які одночасно зберігаються в цеху.

Приміщення та будівлі повинні бути класифіковані за їх пожежною небезпекою, що дозволяє визначити необхідні заходи безпеки для кожної категорії приміщень та забезпечити відповідний рівень захисту. Необхідно розробити та впровадити заходи, що забезпечують пожежну безпеку під час проведення технологічних процесів, включаючи контроль над можливими джерелами запалювання, забезпечення герметичності обладнання та встановлення систем вентиляції.

Забезпечення безпеки при виході горючих речовин із технологічних апаратів передбачає нейтралізацію горючих газів, герметизацію обладнання, встановлення систем відведення газів та інші заходи. Для попередження аварійних ситуацій необхідно постійно контролювати стан технологічного

обладнання та проводити своєчасні ремонти, використовуючи якісні матеріали та дотримуючись технологічних регламентів.

Приміщення повинні бути обладнані сучасними системами виявлення та гасіння пожеж, такими як пожежні сигналізації та спринклерні системи. Дотримання цих заходів допоможе значно знизити ризик виникнення пожежі на виробничій дільниці та забезпечити безпеку працівників та виробництва в цілому.

Відповідальним за стан протипожежного захисту на підприємстві є його керівник, який своїм наказом призначає посадових осіб щодо забезпечення пожежної безпеки підприємства, відповідальних за пожежну безпеку окремих будівель, споруд, приміщень, дільниць, технологічного та інженерного устаткування, а також за утримання та експлуатацію технічних засобів протипожежного захисту.

Обов'язки щодо забезпечення пожежної безпеки, утримання та експлуатації засобів протипожежного захисту мають бути відображені у відповідних посадових інструкціях, статутах підприємств. Працівники підприємства повинні бути ознайомлені з цими вимогами на інструктажах, під час проходження пожежно-технічного мінімуму.

Пожежна профілактика та заходи щодо забезпечення пожежної безпеки на виробничих об'єктах в Україні регулюються рядом законів, нормативно-правових актів та державних стандартів України [35]. Ця система нормативного регулювання спрямована на створення безпечних умов праці, попередження виникнення пожеж та мінімізацію їхніх наслідків.

ВИСНОВКИ

В першому розділі кваліфікаційної роботи проведено аналіз додатків із сфери освіти та проаналізовано основних конкурентів. Визначено основні функціональні вимоги до розроблюваного додатку та вибрано сучасний стек технологій для його реалізації. Також визначено основні стадії розробки, що включають підготовку, розробку, тестування та впровадження продукту.

В другому розділі було сформовано основну структуру додатку, як файлову, так і програмну, вибрано архітектуру MVC. Спроектовано базу даних, яка забезпечить ефективне зберігання даних, та створено основний логотип, який є обличчям бренду. Було також розроблено основні веб-сторінки клієнтської частини та функції контролерів серверної частини, а також визначено, як вони мають взаємодіяти для забезпечення належного функціонування додатку.

В третьому розділі було розроблено всі функціональні частини додатку. Було проведено тестування з дотриманням сучасних правил розробки веб-застосунків, що включало валідацію, обробку помилок та забезпечення безперебійної роботи серверу. Усі маршрути тестувалися за допомогою Postman для перевірки коректності роботи API. Використання незалежного середовища забезпечило для різних компонентів додатку, що сприяло стабільності та спрощенню процесу розробки і розгортання.

У розділі «Безпека життєдіяльності, основи охорони праці» висвітлено заходи безпеки та вимоги щодо охорони праці. Окрему увагу приділено надзвичайним ситуаціям екологічного характеру та пожежній профілактиці на дільниці (в цеху). Ці заходи забезпечують захист здоров'я та безпеки працівників під час розробки та експлуатації програмного забезпечення.

ПЕРЕЛІК ДЖЕРЕЛ

1 Strutynska, I., Kozbur, H., Dmytrotsa, L., Hlado, O., Kozbur, I. and Gashchyn, N., 2023. Analysis of the SMEs' Digitalization State Using HIT Index and Machine Learning Technique. 13th International Conference on Advanced Computer Information Technologies (ACIT). Wroclaw, Poland: IEEE, Institute of Electrical and Electronics Engineers Inc., pp. 332-337.

2 Мацюк, С. і Мацюк, Г.Р., 2022. Розвиток професійної технічної освіти в сучасних умовах. У: Збірник тез ІІІ Міжнародної науково-практичної конференції молодих учених та студентів „Філософські виміри техніки", 1-2 грудня 2022 року. Т., ТНТУ, с. 133-134. (Професійна технічна освіта в сучасних умовах).

3 Quizlet Overview. URL: https://quizlet.com/latest. (date of access: 02.04.2024).

4 AnkiApp - The best flashcard app to learn languages and more. URL: https://www.ankiapp.com/ (date of access: 02.02.2024).

5 Ling-app.com. Best Anki vs Memrise Review: Which One is the #1 App?. URL: https://ling-app.medium.com/best-anki-vs-memrise-review-which-one-is-1-app-f3b53f0f0e0e. (date of access: 02.02.2024).

6 Ling-app.com. (2023). Brainscape Review: Is it worth it?. URL: https://ling-app.medium.com/brainscape-review-is-it-worth-it-8c73e4f64c5d. (date of access: 02.02.2024).

7 General Data Protection Regulation - Wikipedia. URL: https://en.wikipedia.org/wiki/General_Data_Protection_Regulation. (date of access: 07.02.2024).

8 Model–view–controller - Wikipedia. URL: https://en.wikipedia.org/wiki/Model–view–controller. (date of access: 09.02.2024).

9 Михайлишин, М.С. і Лотоцький, В., 2018. Місце повторного використання компонентів у загальному процесі розробки програмного

забезпечення**.** У: Матеріали V науково-технічної конференції „Інформаційні моделі, системи та технології“, 1-2 лютого 2018 року. Т., ТНТУ, с. 95. (Секція 4. Програмна інженерія та моделювання складних розподілених систем).

10 Wikipedia, 2023. Representational state transfer. URL: https://en.wikipedia.org/wiki/Representational_state_transfer. (date of access: 11.02.2024).

11 SitePen, 2023. The Basics of a Monorepo: Where Projects Go to Meet. URL: https://www.sitepen.com/blog/the-basics-of-a-monorepo-where-projects-go-to-meet. (date of access: 11.02.2024).

12 Toptal. TypeScript vs. JavaScript: Your Go-to Guide. URL: https://www.toptal.com/javascript/typescript-vs-javascript (date of access: 11.02.2024).

13 Kinsta, 2024. PostgreSQL vs MySQL: Explore Their 12 Critical Differences. URL: https://kinsta.com/blog/postgresql-vs-mysql/ (date of access: 11.02.2024).

14 Medium. Vue vs Angular: A Comprehensive Comparison. URL: https://medium.com/digitalya-ops/vue-vs-angular-a-comprehensive-comparison. (date of access: 11.02.2024).

15 Medium. Vue vs React: The Ultimate Comparison for the Year. URL: https://medium.com/nerd-for-tech/vue-vs-react-the-ultimate-comparison-for-the-year. (date of access: 11.02.2024).

16 Geek Culture, 2023. Getting Started with Pinia: State Management in Vue.js. URL: https://medium.com/geekculture/getting-started-with-pinia-state-management-in-vue-js. (date of access: 20.03.2024).

17 Medium, 2023. Power of Custom Directives in Vue.js. URL: https://matifzia.medium.com/power-of-custom-directives-in-vue-js-3b00a421f02a. (date of access: 21.03.2024).

18 Medium, 2023. How Computed Properties Can Greatly Improve Your Vue.js Code. URL: https://medium.com/coddyfingers/how-computed-properties-can-greatly-improve-your-vue-js-code. (date of access: 23.03.2024).

19 Medium, 2023. Best Vue Developer Tools for 2024. URL: https://medium.com/simform-engineering/best-vue-developer-tools-for-2024-591d9317d242. (date of access: 30.03.2024).

20 DEV Community, 2023. The Ultimate Comparison: Ant Design vs Material UI. URL: https://dev.to/frontendmag/the-ultimate-comparison-ant-design-vs-material-ui-which-react-ui-library-to-choose-3mld. (date of access: 03.04.2024).

21 Medium, 2023. Vite: The Fastest Frontend Tooling for Modern Web Projects. URL: https://medium.com/@vitejs/vite-the-fastest-frontend-tooling-for-modern-web-projects-8fbf1737ddc3. (date of access: 03.04.2024).

22 Medium, 2023. Vite vs Webpack: The New Age of Frontend Build Tools. URL: https://medium.com/@vitejs/vite-vs-webpack-the-new-age-of-frontend-build-tools-9d52e5df04e8. (date of access: 03.04.2024).

23 Medium, 2023. How to Build and Dockerize a Node.js Application. URL: https://medium.com/@docker/how-to-build-and-dockerize-a-node-js-application-925d4bfb1b64. (date of access: 09.04.2024).

24 FreeCodeCamp, 2023. How to Create a CRUD API – NodeJS and Express Project for Beginners. URL: https://www.freecodecamp.org/news/how-to-create-a-crud-api-nodejs-and-express-for-beginners/. (date of access: 13.04.2024).

25 Medium, 2023. A Complete Database Normalization Tutorial. URL: https://medium.com/swlh/a-complete-database-normalization-tutorial-732df3748d0e. (date of access: 14.04.2024).

26 Logo Design: Create The Perfect Logo For Your Business. URL: https://logo.com/blog/logo-design-create-perfect-logo. (date of access: 18.04.2024).

27 Linear, Hierarchical, and Network Website Structures Explained. URL: https://dzone.com/articles/linear-hierarchical-and-network-website-structures-explained. (date of access: 25.04.2024).

28 LogRocket Blog, 2023. The ultimate guide to Vue Router. URL: https://blog.logrocket.com/the-ultimate-guide-to-vue-router/ (date of access: 30.04.2024).

29 AltexSoft, 2023. DeepL vs. Google Translate: Which Translator Is Better?. URL: https://www.altexsoft.com/blog/deepl-vs-google-translate/ (date of access: 06.05.2024).

30 Interaction Design Foundation, 2023. The UX Designer’s Guide to Typography. URL: https://www.interaction-design.org/literature/article/the-ux-designers-guide-to-typography (date of access: 16.05.2024).

31 TypeScript, 2023. Creating a TypeScript Node.js Express Application. URL: https://www.typescriptlang.org/docs/handbook/migrating-from-javascript.html (date of access: 20.05.2024).

32 Артем’єв, С. Р., Андронов, В. А., Андронов, А. І., Антонов, О. В. і Бригада, О. В. (2021) Екологія надзвичайних ситуацій: Частина І. 2 - ге Видання друге. Харків: Національний університет цивільного захисту України.

33 Верховна Рада України Закон України "Про зону надзвичайної екологічної ситуації". URL: https://zakon.rada.gov.ua/laws/show/1908-14#Text (дата звернення: 30.05.2024).

34 Український науково-дослідний і проектний інститут цивільного будівництва (2016) ДСТУ Б В.1.1-7:2016 Пожежна безпека об'єктів будівництва. Загальні вимоги. URL: https://dbn.co.ua/load/normativy/dstu_b_v_1_1_7_2016_pozhezhna_bezpeka_ob_ektiv_budivnictva_zagalni_vimogi/1-1-0-172 (дата звернення: 30.05.2024).

35 Волянський, П., Михайлов, В. і Васильєв, І. (2021) Наукове обґрунтування методичного забезпечення організації заходів з оповіщення та евакуації людей з будинків та споруд при пожежі. Державна служба України з надзвичайних ситуацій, Інститут державного управління та наукових досліджень з цивільного захисту. № держреєстрації 0119U002483.

ДОДАТКИ

Додаток А

Елемент форми – Input

```
<template>
  <div
    :class="{
      base_input: true,
      textarea: props.useTextarea,
      primary: props.type === 'primary',
      second: props.type === 'second',
    }">
    <input
      v-if="!useTextarea"
      type="text"
      class="input"
      :placeholder="props.placeholder"
      :value="props.modelValue"
      @input="handleInput" />
    <textarea
      v-else
      class="input"
      :placeholder="props.placeholder"
      :value="props.modelValue"
      @input="handleInput"></textarea>
    <div class="bottom">
      <label class="label">{{ labelBottom }}</label>
      <label
        class="error"
        v-if="error"
        >{{ error }}</label
      >
    </div>
  </div>
</template>

<script setup lang="ts">
  import { ref } from 'vue';
  import validation from '@/utils/validator';

  const error = ref<string>('');

  const props = defineProps({
    rules: {
      type: String,
      default: '',
    },
    name: {
      type: String,
      default: 'Field',
    },
    useTextarea: {
```

```
      type: Boolean,
      default: false,
    },
    type: {
      type: String,
      default: 'second',
    },
    placeholder: {
      type: String,
      default: '',
    },
    labelBottom: {
      type: String,
      default: '',
    },
    modelValue: {
      type: String,
      default: '',
    },
  });

  defineExpose({
    validate: () => {
      try {
        validation(props.rules, props.modelValue, props.name);
        return false;
      } catch (e: any) {
        error.value = e.message;
        return { error: e.message };
      }
    },
  });

  const emit = defineEmits(['update:modelValue']);

  const handleInput = (event: Event) => {
    error.value = '';
    emit('update:modelValue', event.target?.value);
  };
</script>
```

Додаток Б

Елемент форми – Button

```
<template>
  <div>
    <button
      v-if="!linkTo"
      @click="onClick"
      :type="submit ? 'submit' : 'button'"
      :class="styleClass">
      <slot></slot>
    </button>
    <RouterLink
      :to="linkTo"
      :class="styleClass"
      v-else>
      <slot></slot>
    </RouterLink>
  </div>
</template>

<script setup lang="ts">
  import { RouterLink } from 'vue-router';
  import { computed } from 'vue';

  const emit = defineEmits(['click']);

  const onClick = () => {
    emit('click');
  };

  const props = defineProps({
    type: {
      type: String,
      default: null,
    },
    className: {
      type: String,
      default: null,
    },
    linkTo: {
      type: Object,
      default: null,
    },
    submit: {
      type: Boolean,
      default: false,
    },
  });

  const styleClass = computed(() => {
```

```
      return ['base_button', props.type, props.className].join('
');
    });
</script>

<style lang="scss" scoped>
  .base_button {
    appearance: none;
    box-sizing: border-box;
    cursor: pointer;
    outline: none;
    user-select: none;
    -webkit-user-select: none;
    touch-action: manipulation;
    will-change: transform;
    text-decoration: none;
    transition: all 300ms cubic-bezier(0.23, 1, 0.32, 1);
    background-color: transparent;
    padding: 0px;
    letter-spacing: 0.2px;
  }
  .second {
    border: 0.125em solid var(--base);
    border-radius: var(--radius);
    box-sizing: border-box;
    color: var(--base);
    display: inline-block;
    font-size: 16px;
    font-weight: 700;
    line-height: normal;
    padding: 0.5em 1.5em;
    text-align: center;

    &:disabled {
      pointer-events: none;
    }
    &:hover {
      color: #fff;
      background-color: var(--base);
      box-shadow: rgba(0, 0, 0, 0.25) 0 8px 15px;
      transform: translateY(-2px);
    }
    &:active {
      box-shadow: none;
      transform: translateY(0);
    }
  }
</style>
```

Додаток В

Модель таблиці модулів

```
import {
  Entity,
  Column,
  PrimaryGeneratedColumn,
  CreateDateColumn,
  UpdateDateColumn,
  OneToMany,
} from 'typeorm';

import { Cards } from './cards.entity';

@Entity()
export class Modules {
  @PrimaryGeneratedColumn()
  id: number;

  @Column('varchar', { length: 40, nullable: false })
  name: string;

  @Column('varchar', { nullable: true })
  description: string;

  @CreateDateColumn({ name: 'created_at' })
  createdAt: Date;

  @UpdateDateColumn({ name: 'updated_at' })
  updatedAt: Date;

  @OneToMany(() => Cards, cards => cards.module)
  cards: Cards[];
}
```

Додаток Г

Файл ініціалізації сервера

```
import 'reflect-metadata';
import dotenv from 'dotenv';
import express from 'express';
import { Request, Response } from 'express';
import cors from 'cors';
import { Database } from './database';
import moduleRoter from 'src/routers/module.router';
import cardRouter from 'src/routers/card.router';
import storeRouter from 'src/routers/store.router';

dotenv.config();
const APP_PORT = Number(process.env.APP_PORT) || 3000;
const whitelist = ['http://localhost',
'http://localhost:5173'];
const corsOptions = {
  origin: function (origin: string, callback) {
    if (whitelist.indexOf(origin) !== -1 || !origin) {
      callback(null, true);
    } else {
      callback(new Error('Not allowed by CORS'));
    }
  },
  optionsSuccessStatus: 200,
  credentials: true,
} as cors.CorsOptions;

Database.initialize()
  .then(() => {
    console.log('Data Source is ready!!!');
    const app = express();
    app.use(express.json());
    app.use(cors(corsOptions));

    app.get('/users', function (req: Request, res: Response) {
      res.json({ users: [] });
    });

    app.use('/modules', moduleRoter);
    app.use('/cards', cardRouter);
    app.use('/store', storeRouter);

    app.listen(APP_PORT, () => {
      console.log(`Server started! { ${APP_PORT} }`);
    });
  })
  .catch(err => {
    console.error('Error during launching:', err);
});
```

Додаток Ґ

docker-compose.yaml

```
version: '3.8'

networks:
  s5_network:
    driver: bridge

services:
  mysql:
    container_name: mysql
    build:
      context: ./mysql
    environment:
      - MYSQL_DATABASE=${MYSQL_DATABASE}
      - MYSQL_USER=${MYSQL_USER}
      - MYSQL_PASSWORD=${MYSQL_PASSWORD}
      - MYSQL_ROOT_PASSWORD=${MYSQL_ROOT_PASSWORD}
      - TZ=UTC
    ports:
      - "${MYSQL_PORT}:3306"
    networks:
      - s5_network
    volumes:
      - db:/var/lib/mysql
  server:
   container_name: server
   build:
     context: .
   ports:
      - '3001:3001'
  volumes:
       - ./:/s5

  volumes:
db:
```