Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Скорика Владислава Максимовича*
(ПІБ)

академічної групи *121-20-1*
(шифр)

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

освітньої програми *Інженерія програмного забезпечення*
(назва освітньої програми)

на тему: *Розробка вебдодатку для вивчення англійської мови з використанням штучного інтелекту на базі стеку технологій React, Node.js, MongoDB та фреймворку Express*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*доц. Ширін А.Л.*			
розділів:				
спеціальний	*доц. Ширін А.Л.*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент				
Нормоконтролер	*доц. Мартиненко А.А.*			

Дніпро
2024

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

М.О. Алексєєв
(підпис) (прізвище, ініціали)

« » 2024 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *121-20-1* *Скорика Владислава Максимовича*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка вебдодатку для вивчення англійської мови з використанням штучного інтелекту на базі стеку технологій React, Node.js, MongoDB та фреймворку Express*

затверджена наказом ректора НТУ «ДП» від *23.05.2024* № *469-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів проєктно-технологічної та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі. Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***01.06.2024 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***06.06.2024 р.***

Завдання видав _____ *доц. Ширін А.Л.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Скорик В.М.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2024 р.*

Термін подання кваліфікаційної роботи до ЕК: *24.06.2024 р.*

РЕФЕРАТ

Пояснювальна записка: 82 с., 31 рис., 3 дод., 25 джерел.

Об'єкт розробки: вебдодаток для вивчення англійської мови з використанням штучного інтелекту.

Мета кваліфікаційної роботи: розробка вебдодатку для вивчення англійської мови з використанням штучного інтелекту на базі стеку технологій React, Node.js, MongoDB та фреймворку Express.

У вступі розглядається важливість англійської мови у сучасному світі, труднощі у її вивченні, а також переваги використання цього вебдодатку для вдосконалення своєї англійської.

У першому розділі описується в подробицях нюанси у вивченні англійської мови, призначення розробки та постановка задачі – постановка технологій, що використовуються, і поверхневий опис функціональних можливостей додатку.

У другому розділі обговорюється в подробицях призначення платформи, права доступу користувачів, використовувані архітектури та шаблони програмування, послідовність алгоритмів функціонування сервера платформи, а також опис і пояснення вибору технологій для написання проекту.

В економічному розділі визначається рівень складності проекту, проводиться розрахунок вартості розроблення вебдодатку залежно від різних параметрів (кількості рядків коду, ресурсів, електрики, Інтернету тощо), а також визначається запланований час на його створення.

Практичне значення роботи полягає у розробці вебдодатку, який дозволить людям з будь-яким рівнем знання англійської покращувати свої навички у необхідних темах з допомогою штучного інтелекту.

Актуальність платформи обумовлюється великим попитом людей на вивчення англійської мови зручним та персоніфікованим способом.

Список ключових слів: ВЕБДОДАТОК, БРАУЗЕР, ШТУЧНИЙ ІНТЕЛЕКТ, АНГЛІЙСЬКА, JAVASCRIPT, NODEJS, НАВЧАННЯ.

ABSTRACT

Explanatory note: 82 p., 31 fig., 3 app., 25 sources.

Object of development: a web application for learning English using artificial intelligence.

The purpose of the qualification work: development of a web application for learning English using artificial intelligence based on the React, Node.js, MongoDB and Express framework technologies.

The introduction discusses the importance of the English language in today's world, the difficulties in learning it, and the benefits of using this web application to improve your English.

The first section describes in detail the nuances of learning English, the purpose of the development and the task statement - the statement of the technologies used and a superficial description of the functionality of the application.

The second section discusses in detail the purpose of the platform, user access rights, used architectures and programming patterns, the sequence of algorithms for the platform server, as well as a description and explanation of the choice of technologies for writing the project.

The economic section determines the level of complexity of the project, calculates the cost of developing a web application depending on various parameters (the number of lines of code, resources, electricity, the Internet, etc.), and determines the planned time for its creation.

The practical significance of the work is to develop a web application that will allow people with any level of knowledge of English to improve their skills in the necessary topics with the help of artificial intelligence.

The relevance of the platform is determined by the great demand of people to learn English in a convenient and personalized way.

Keywords list: WEB APPLICATION, BROWSER, ARTIFICIAL INTELLIGENCE, ENGLISH, JAVASCRIPT, NODEJS, LEARNING.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

HTML – HyperText Markup Language.

CSS – Cascading Style Sheets.

JS – JavaScript.

SQL – Structured Query Language.

JSON – JavaScript Object Notation.

JWT – JSON Web Token.

API – Application Programming Interface.

HTTP – Hypertext Transfer Protocol.

ЗМІСТ

ВСТУП

У сучасному світі англійська мова стала важливим ключем до багатьох можливостей – від академічного розвитку до кар'єрного зростання. Англійська мова відіграє визначальну роль як мова міжнародного спілкування, науки, бізнесу та культури. Знання англійської відкриває безліч можливостей, забезпечуючи доступ до інформації, освіти, робочих місць та спілкування з людьми з різних куточків планети. Знання англійської мови стає ключем до успіху в різних сферах життя та дозволяє реалізувати свій потенціал у глобальному світі.

Проте, навіть із зростаючою свідомістю про важливість вивчення англійської, багато людей стикаються з труднощами у цьому процесі. Серед найпоширеніших проблем можна виділити:

− відсутність мотивації: брак мотивації може затримати процес вивчення. Без внутрішньої мотивації важко зберігати інтерес та наполегливість у вивченні;

− відсутність часу: багато людей стикаються з обмеженим часом на вивчення мови через роботу, навчання та інші зобов'язання;

− несистематичне навчання: відсутність структурованого підходу до вивчення може призвести до неефективного використання часу та ресурсів;

− відсутність доступу до якісних навчальних ресурсів: не у всіх є доступ до якісних курсів, викладачів та навчальних матеріалів.

Додаток для вивчення англійської мови може вирішити багато з цих проблем. Він може надати структуровану програму навчання, яка пристосована до потреб користувача та його рівня знань. Також додаток може забезпечити мотивацію через систему відстеження прогресу, а також надати доступ до високоякісних навчальних ресурсів, включаючи тестування та інтерактивні завдання.

Завдяки технологіям штучного інтелекту, додаток для вивчення англійської може адаптуватися до потреб кожного користувача, враховуючи

його стиль навчання, міцні та слабкі сторони, а також надавати персоналізовані рекомендації та фідбек.

Метою даної кваліфікаційної роботи є створення зручного програмного додатку із зрозумілим функціоналом, що дозволить спростити вивчення англійської.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1. Загальні відомості з предметної галузі

Вивчення англійської мови на сьогоднішній день стає надзвичайно важливим аспектом освіти та професійного розвитку. Англійська є мовою міжнародного спілкування та бізнесу, а також надзвичайно важливою для отримання доступу до світових ресурсів та можливостей. Особливо у сфері науки, технологій та культури володіння англійською мовою є ключовим фактором успіху.

Багато людей стикаються з проблемами в процесі вивчення англійської мови через відсутність чіткої стратегії та ефективних методів навчання. Навіть при бажанні вивчити мову, може виникати відчуття розчарування через невдачі та відчуття, що прогрес стоїть на місці.

Однією з основних причин цього є відсутність індивідуалізованого підходу до навчання. Кожна людина має власний стиль навчання, рівень знань та цілі, і що працює для одного, може бути неефективним для іншого. Відсутність персоналізованого підходу у більшості навчальних програм та курсів може ускладнювати процес вивчення для багатьох людей.

Штучний інтелект став неодмінною складовою сучасних технологій, і його застосування у вивченні мов є об'єктом зростаючого інтересу. Штучний інтелект може ефективно використовуватися для індивідуалізованого навчання, автоматичної оцінки та корекції помилок, а також для створення інтерактивних та цікавих навчальних матеріалів.

Використання штучного інтелекту в додатку для вивчення англійської мови відкриває безліч можливостей для користувачів. Штучний інтелект може зручно замінити роль викладача, надаючи індивідуалізовані рекомендації та вправи, які відповідають потребам та рівню знань кожного користувача. Крім того, завдяки штучному інтелекту, додаток може генерувати нескінченну

кількість завдань та вправ для вивчення англійської мови, що дозволяє користувачам постійно підтримувати мотивацію та продовжувати розвиватися. Штучний інтелект також дозволяє автоматизувати процес перевірки виконаних завдань, забезпечуючи швидку та об'єктивну оцінку знань користувачів. Таким чином, використання штучного інтелекту робить процес вивчення англійської мови більш ефективним, цікавим та зручним для кожного користувача.

Розробка вебдодатку для вивчення англійської мови з використанням штучного інтелекту на базі стеку технологій React, Node.js, MongoDB та фреймворку Express відповідає потребам сучасного користувача, який шукає зручні та ефективні способи вивчення мов. Цей додаток має на меті поєднати найновітніші технології з сучасними методиками вивчення мови для створення ефективного та захоплюючого навчального досвіду.

1.2. Призначення розробки та галузь застосування

На сьогоднішній день існує безліч додатків для вивчення англійської мови, проте багато з них мають свої недоліки. Деякі з них можуть бути недостатньо структурованими, не надавати персоналізованого підходу або не мати достатньої кількості високоякісних навчальних матеріалів. Інші можуть бути недостатньо мотиваційними або не забезпечувати ефективних механізмів відстеження прогресу.

Проте додаток, який використовує штучний інтелект, може вирішити ці проблеми. Він може адаптуватись до потреб кожного користувача, надаючи персоналізовані рекомендації та завдання, які відповідають рівню знань та індивідуальним потребам. Крім того, він може використовувати аналітику та машинне навчання для визначення ефективних методів навчання та підвищення мотивації користувачів.

Такий додаток може забезпечити структуровану та систематичну програму навчання, включаючи різноманітні тести, інтерактивні завдання та механізми взаємодії з викладачем чи іншими користувачами. Він може надати

користувачеві доступ до якісних навчальних матеріалів, які постійно генеруються та адаптовані до потреб сучасного вивчення мови.

Тому додаток з використанням штучного інтелекту може бути ефективним інструментом для вивчення англійської мови, який допоможе подолати недоліки традиційних методів навчання та забезпечить ефективний та цікавий процес вивчення.

1.3. Підстави для розробки

Підставами для розробки та виконання кваліфікаційної роботи є:

− освітня програма 121 Інженерія програмного забезпечення;

− навчальний план та графік навчального процесу;

− наказ ректора Національного технічного університету «Дніпровська політехніка» № 469-с від 23.05.2023 р;

− завдання на кваліфікаційну роботу на тему «Розробка вебдодатку для вивчення англійської мови з використанням штучного інтелекту на базі стеку технологій React, Node.js, MongoDB та фреймворку Express».

1.4. Постановка завдання

Основною метою роботи є розробка інтерактивного вебдодатку, який забезпечить користувачам зручний і ефективний інструмент для вивчення англійської мови. Додаток буде використовувати штучний інтелект для персоналізації навчального процесу та оптимізації результатів користувачів.

Конкретні завдання включають:

− розробку інтерфейсу користувача, який буде інтуїтивно зрозумілим та забезпечить зручний доступ до всіх функцій додатку;

− інтеграцію системи штучного інтелекту для аналізу та оцінки навчальних досягнень користувачів, а також для персоналізації навчальних матеріалів та рекомендацій;

− розробку системи відстеження прогресу користувачів, яка буде забезпечувати статистику про виконані завдання, досягнуті результати та рекомендації щодо подальшого навчання;

− забезпечення безпеки та захисту даних користувачів;

− оптимізацію додатку для працездатності та швидкодії на різних пристроях та платформах;

− проведення тестування та виправлення помилок для забезпечення стабільної та безперебійної роботи додатку.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Вебплатформа повинна бути реалізована мовою програмування JavaScript у вигляді вебсайту, скористатись яким можна за допомогою будь-якого браузера (Chrome, Opera, Microsoft Edge, Safari, Mozilla та ін.).

Для написання платформи передбачається використання наступних технологій: HTML, CSS, JavaScript, Express, React, OpenAI API.

Всі дані, що використовуються всередині системи, а також відображаються користувачам, повинні бути розміщені в базі даних MongoDB.

Платформа має надавати такий функціонал:

− реєстрація та авторизація користувачів;

− можливість вивчати англійську мову, виконуючи завдання різних типів;

− відстеження навчального прогресу;

− використання штучного інтелекту;

1.5.2. Вимоги до інформаційної безпеки

Забезпечення інформаційної безпеки є одним з основних вимог у розробці вебзастосунку. Додаток повинен забезпечувати захист конфіденційної інформації користувачів, а також гарантувати цілісність та доступність даних.

Основні вимоги до інформаційної безпеки:

1) Захист даних користувачів:

− Забезпечення безпеки особистих даних користувачів, включаючи ім'я, електронну адресу, пароль та інші особисті дані;

− Використання шифрування для захисту передачі та збереження конфіденційної інформації.

2) Управління доступом:

− Реалізація механізмів автентифікації та авторизації користувачів з метою контролю доступу до функцій та даних додатку;

− Встановлення рівнів доступу для різних типів користувачів залежно від їх ролі в системі.

3) Захист від вразливостей:

− Проведення тестування на проникнення для виявлення та усунення потенційних вразливостей системи;

− Постійне оновлення компонентів та бібліотек, що використовуються в додатку, для зменшення ризику використання вразливих версій програмного забезпечення.

4) Резервне копіювання даних:

− Періодичне створення резервних копій даних для запобігання втраті інформації внаслідок технічних або людських помилок.

1.5.3. Вимоги до складу та параметрів технічних засобів

Для доступу до платформи може бути використаний будь-який пристрій: комп'ютер, ноутбук, телефон, планшет. Важливі такі характеристики браузера (як приклад використано Chrome та ноутбук):

− версія браузера Chrome 81.0.4044.20 або вище;

− підтримка 32- або 64-розрядного процесора та операційної системи;

− операційна система Windows 7 чи вище;

− наявність маніпулятора-миші/тачпаду та клавіатури;

− оперативна пам'ять об'ємом 2 ГБ або вище

− процесор із частотою 1 ГГц або вище;

− жорсткий диск HDD чи SSD об'ємом 10 ГБ або вище;

1.5.4. Вимоги до інформаційної та програмної сумісності

Додаток має бути реалізовано на мові програмування JavaScript із використанням фреймворку Express для backend частини, React, HTML і CSS для frontend частини та бази даних MongoDB.

Для роботи з додатком потрібно мати доступ до мережі Інтернет для роботи з OpenAI API і будь-який браузер.

Для розгортання додатку потрібно мати встановлений Docker, за допомогою якого в контейнерах будуть запускатись всі складові застосунку.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Функціональне призначення програми

Вебдодаток для вивчення англійської мови з використанням штучного інтелекту створений з метою забезпечення ефективного і доступного способу вивчення англійської мови для користувачів з різним рівнем підготовки. Система використовує сучасні технології штучного інтелекту для персоналізації навчального процесу, що дозволяє кожному користувачу отримувати індивідуально підібрані завдання та рекомендації.

Додаток реалізує ряд ключових функцій, які забезпечують повний цикл навчання англійській мові:

1. Реєстрація та автентифікація користувачів:

– Створення облікового запису;

– Вхід в систему за допомогою логіну та паролю;

– Система ролей користувач/адміністратор.

2. Інтерактивні завдання та вправи:

– Різні види завдань, включаючи тестування та письмо;

– Вправи, які враховують рівень знань користувача.

3. Система зворотного зв'язку:

– Негайний зворотний зв'язок щодо виконання завдань;

– Підказки та пояснення до помилок, що допомагають зрозуміти та засвоїти матеріал;

– Можливість перегляду виконаних завдань з отриманим зворотним зв’язком.

4. Моніторинг прогресу та досягнень:

– Відображення прогресу у вигляді графіків та карток;

– Перегляд статистики;

– Наявність історії виконаних завдань.

2.2. Опис застосованих математичних методів

Під час розробки вебдодатку для вивчення англійської мови з використанням штучного інтелекту, використовувалися стандартні математичні операції та можливості використаних технологій для підрахунків та сортування.

2.3. Опис використаної архітектури та шаблонів проектування

Архітектура вебдодатку для вивчення англійської мови побудована за принципами розподіленої системи з використанням клієнт-серверної моделі. Основними компонентами архітектури є фронтенд (клієнтська частина), бекенд (серверна частина) та база даних. Такий підхід забезпечує модульність, масштабованість та зручність підтримки додатку.

Фронтенд частина реалізована з використанням бібліотеки React, що дозволяє створювати динамічні та інтерактивні користувацькі інтерфейси. React використовує компонентний підхід, що забезпечує багаторазове використання коду та спрощує управління станом додатку. Інструмент Vite використовується для швидкої збірки та запуску проекту, що забезпечує високу продуктивність під час розробки. Vite надає гаряче перезавантаження модулів (HMR), що значно пришвидшує процес розробки та тестування.

Серверна частина додатку побудована на базі Node.js з використанням фреймворку Express. Express забезпечує легку та гнучку маршрутизацію запитів, підтримку middleware для обробки запитів та відповідей, а також інтеграцію з різними базами даних та іншими сервісами. База даних MongoDB використовується для зберігання даних користувачів, завдань, результатів тестів та інших важливих даних. Mongoose, об'єктно-документний менеджер (ODM) для MongoDB, забезпечує зручний інтерфейс для взаємодії з базою даних та валідацію даних.

У розробці додатку використані кілька шаблонів проектування для забезпечення ефективності та масштабованості системи. Паттерн "Модель-Вид-Контролер" (MVC) (рис. 2.1.) розділяє логіку додатку на три взаємодіючі компоненти:

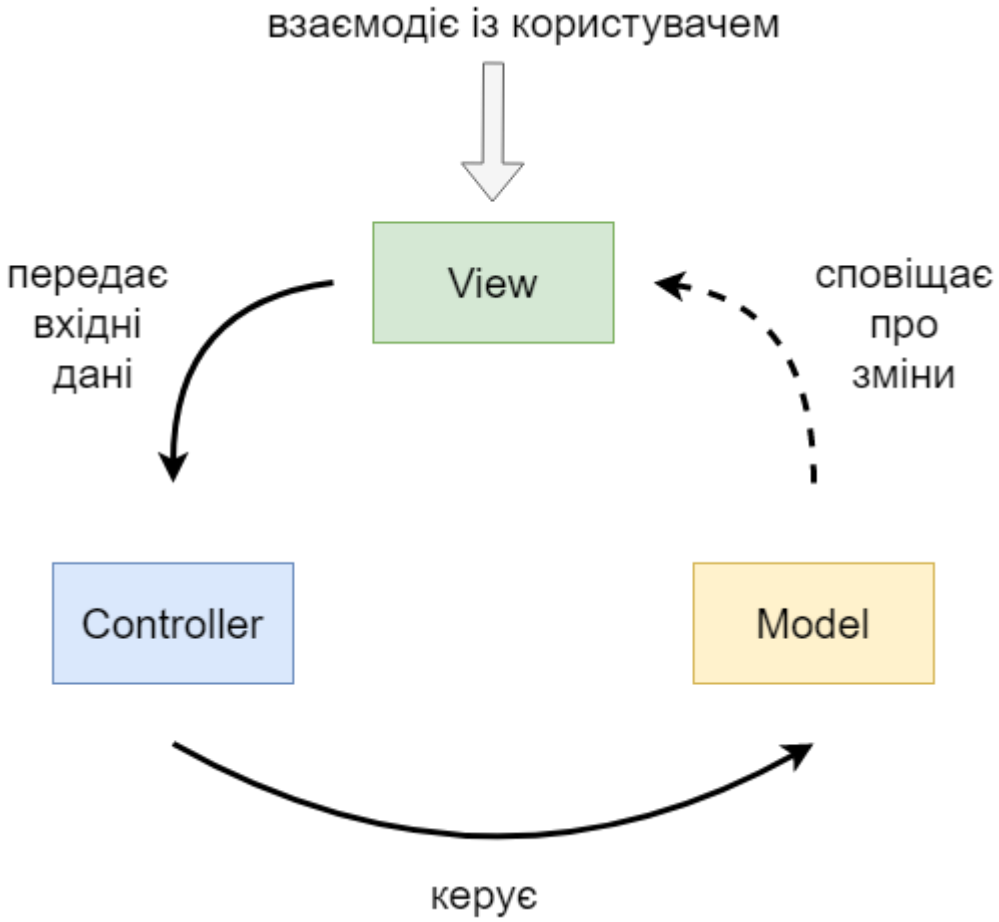

Рис. 2.1. Паттерн "Модель-Вид-Контролер"

Модель (Model): відповідає за роботу з даними додатку. Модель містить логіку для доступу, зберігання та маніпулювання даними, взаємодіє з базою даних і забезпечує цілісність та валідність даних. У проекті це реалізується за допомогою Mongoose, що дозволяє визначати схеми та моделі для MongoDB.

Вид (View): відповідає за відображення даних користувачеві. Це шар інтерфейсу користувача, який генерує HTML-код на основі даних моделі та надає його користувачеві. У нашому проекті View реалізується за допомогою React, що дозволяє створювати динамічні та інтерактивні інтерфейси.

Контролер (Controller): відповідає за обробку вхідних запитів користувача, взаємодію з моделлю та передачею даних до інтерфейсу. Контролер приймає запити від клієнта, обробляє їх, звертається до моделі для отримання або маніпуляції даними та повертає результати у інтерфейс для відображення. У проекті контролер реалізується за допомогою Express, що дозволяє легко визначати маршрути та обробляти запити.

Паттерн MVC спрощує підтримку та розширення додатку, забезпечуючи чітке розділення відповідальностей. Це дозволяє різним командам розробників працювати над різними частинами додатку одночасно, зменшуючи ймовірність конфліктів та підвищуючи продуктивність.

2.4. Опис використаних технологій та мов програмування

У процесі розробки програмного додатку було використано ряд сучасних технологій та мов програмування. Кожна з них відіграє важливу роль у забезпеченні функціональності, продуктивності та надійності системи.

REST

REST (Representational State Transfer) — це стиль архітектури для розподілених систем, зокрема для вебсервісів. Він був вперше описаний Роєм Філдінгом у його докторській дисертації в 2000 році. Основна ідея REST полягає в тому, щоб використовувати стандартні методи HTTP для взаємодії між клієнтом і сервером, що дозволяє створювати масштабовані і легкі у підтримці вебсервіси.

Основні принципи REST:

– Клієнт-серверна архітектура – розділення клієнтської та серверної частин додатку. Клієнт відповідає за інтерфейс користувача, а сервер — за обробку запитів і зберігання даних;

– Безстанова взаємодія – кожен запит від клієнта до сервера повинен містити всю необхідну інформацію для його обробки. Сервер не зберігає стан клієнта між запитами;

– Кешування – відповіді сервера можуть бути позначені як кешовані або некешовані. Це дозволяє зменшити навантаження на сервер та покращити продуктивність;

– Єдиний інтерфейс – взаємодія між клієнтом і сервером здійснюється через стандартизований інтерфейс (наприклад, URI для ресурсів і методи HTTP);

– Шарова система – можливість мати проміжні сервери (проксі, шлюзи), що можуть забезпечувати додаткову функціональність, таку як балансування навантаження або кешування.

Переваги REST:

– Простота та зручність – використання стандартних методів HTTP (GET, POST, PUT, DELETE), що робить API зрозумілим і легким у використанні;

– Масштабованість – легко масштабувати окремі компоненти (клієнт або сервер), що дозволяє підтримувати великий обсяг трафіку та даних;

– Гнучкість – підтримка різних форматів даних (наприклад, JSON, XML), що дозволяє легко інтегруватися з різними системами;

– Незалежність платформи та мови програмування – клієнти та сервери можуть бути реалізовані на різних платформах і мовах програмування, що забезпечує високу сумісність і гнучкість.

Застосування у проекті:

В рамках розробки вебдодатку для вивчення англійської мови з використанням штучного інтелекту REST використовується для організації взаємодії між фронтендом (React) і бекендом (Node.js та Express).

Клієнтські технології та мови програмування:

JavaScript

JavaScript — це високорівнева, динамічна мова програмування, яка широко використовується для розробки вебдодатків. Вона була створена Бренданом Айком у 1995 році та стала однією з трьох основних технологій, що складають основу сучасного вебу разом з HTML та CSS. JavaScript дозволяє створювати інтерактивні та динамічні вебсторінки, надаючи розробникам можливість виконувати скрипти на стороні клієнта.

Основні особливості JavaScript:

– Інтерпретована мова – JavaScript виконується безпосередньо у веббраузері, без потреби в попередній компіляції;

– Динамічна типізація – зміни типів даних можливі під час виконання програми, що забезпечує більшу гнучкість;

– Подієво-орієнтована – підтримка обробки подій дозволяє створювати інтерактивні вебсторінки, реагуючи на дії користувачів;

– Прототипне успадкування – об'єктна модель JavaScript базується на прототипах, що дозволяє створювати об'єкти без класів;

– Підтримка асинхронності – асинхронні операції (через колбеки, проміси, async/await) дозволяють ефективно обробляти події, виконувати запити до сервера та інші довготривалі задачі.

Переваги JavaScript для веброзробки:

– Висока інтерактивність – JavaScript дозволяє створювати інтерактивні елементи на вебсторінках, такі як динамічні форми, модальні вікна, слайдери тощо;

– Міжплатформеність – виконується у всіх сучасних веббраузерах (Chrome, Firefox, Safari, Edge), що забезпечує високу сумісність без потреби у додаткових плагінах;

– Швидкість розробки – завдяки великій кількості готових бібліотек та фреймворків (наприклад, React, Angular, Vue.js), JavaScript прискорює процес розробки;

– Потужні інструменти розробки – вбудовані інструменти браузера для розробки та відлагодження JavaScript-коду, такі як Chrome DevTools;

– Велика спільнота – активна спільнота розробників, багато ресурсів для навчання та підтримки (форумів, блогів, репозиторіїв на GitHub).

Застосування у проекті

У проекті JavaScript використовується як на стороні клієнта, так і на стороні сервера, забезпечуючи повний стек веброзробки.

Vite

Vite — це сучасний інструмент для побудови фронтенд проектів, створений Еваном Ю, автором Vue.js. Vite фокусується на швидкості розробки та оптимізації процесу збірки за рахунок використання новітніх вебтехнологій, таких як ESBuild і підтримка модулів ES (ESM). Він надає просте налаштування та миттєвий старт проектів з гарячим перезавантаженням модулів (HMR).

Основні особливості Vite:

– Швидка збірка – Vite використовує ESBuild, надзвичайно швидкий збирач та мінімізатор JavaScript, написаний на Go, що забезпечує миттєвий старт та швидку збірку проектів;

– Гаряче перезавантаження модулів (HMR) – Vite забезпечує миттєве оновлення змін у коді без необхідності повного перезавантаження сторінки, що значно прискорює процес розробки;

– Мінімальна конфігурація – Vite пропонує простий та зрозумілий процес налаштування проектів, що дозволяє розробникам швидко розпочати роботу.

– Підтримка сучасних вебстандартів – Vite підтримує новітні стандарти JavaScript, такі як модулі ES, що дозволяє використовувати сучасні можливості мови без додаткової конфігурації;

– Вбудована підтримка TypeScript – Vite має вбудовану підтримку TypeScript, що забезпечує зручну роботу з цією мовою без додаткових налаштувань;

– Гнучка архітектура – Vite легко інтегрується з різними бібліотеками та фреймворками, такими як React, Vue.js, Svelte тощо.

Переваги Vite для веброзробки:

– Швидкий старт – завдяки використанню ESBuild, Vite дозволяє майже миттєво запустити проект, що значно економить час розробників;

– Покращена продуктивність – гаряче перезавантаження модулів забезпечує миттєве оновлення змін, що підвищує ефективність роботи та знижує час очікування;

– Простота налаштування – Vite мінімізує потребу в складних конфігураціях, дозволяючи розробникам зосередитися на написанні коду, а не на налаштуванні інструментів;

– Сучасні можливості – підтримка останніх стандартів JavaScript та вбудована підтримка TypeScript роблять Vite ідеальним вибором для сучасних вебдодатків;

– Гнучкість – легкість інтеграції з різними фреймворками та бібліотеками дозволяє використовувати Vite в різних типах проектів, забезпечуючи високу адаптивність.

Застосування Vite у проекті:

У вебдодатку Vite використовується для швидкої збірки та запуску клієнтської частини, що реалізована на React.

React

React — це популярна бібліотека для створення користувацьких інтерфейсів, розроблена компанією Facebook. Вона дозволяє будувати компонентно-орієнтовані інтерфейси, що робить її потужним інструментом для розробки динамічних і інтерактивних вебдодатків. React забезпечує розробникам можливість створювати великі вебдодатки, в яких дані можуть змінюватися без перезавантаження сторінки.

Основні особливості React:

– Компонентний підхід – React базується на концепції компонентів, які є незалежними і багаторазово використовуваними блоками коду. Це дозволяє створювати складні інтерфейси з невеликих, ізольованих компонентів;

– Віртуальний DOM – React використовує віртуальний DOM для оптимізації оновлень інтерфейсу. Віртуальний DOM дозволяє React ефективно порівнювати поточний стан інтерфейсу з попереднім і оновлювати тільки ті частини, які змінилися;

– JSX (JavaScript XML) — це синтаксичне розширення JavaScript, що дозволяє писати HTML-подібний код у файлах JavaScript. Це спрощує створення компонентів і робить код більш читабельним;

– Однонаправлений потік даних – у React дані передаються з батьківських компонентів до дочірніх через властивості (props), що забезпечує передбачуваність і контрольованість потоку даних;

– Hooks – дозволяють використовувати стан та інші React-функціональні можливості в функціональних компонентах. Вони роблять код більш компактним і зрозумілим.

Переваги React для веброзробки

– Швидкість розробки – завдяки компонентному підходу та можливості багаторазового використання компонентів, розробка на React стає швидшою та ефективнішою;

– Продуктивність – віртуальний DOM забезпечує високу продуктивність оновлень інтерфейсу, знижуючи навантаження на реальний DOM і підвищуючи швидкість відображення змін;

– Гнучкість – React легко інтегрується з іншими бібліотеками та фреймворками, дозволяючи розробникам обирати найкращі інструменти для конкретних задач;

– Широка спільнота – велика та активна спільнота розробників забезпечує доступ до численних ресурсів, навчальних матеріалів, бібліотек та інструментів, що спрощує процес розробки та підтримки додатків;

– Розширюваність – React легко масштабувати для створення великих і складних додатків, що робить його ідеальним вибором для проектів будь-якого розміру.

Застосування у проекті:

У проекті з розробки вебдодатку для вивчення англійської мови React використовується для створення інтерфейсу користувача. Це забезпечує динамічність і інтерактивність додатку, роблячи процес навчання зручним та ефективним.

React Router Dom

React Router Dom — це бібліотека для управління маршрутизацією в додатках на базі React. Вона дозволяє створювати багато-сторінкові додатки з

єдиним інтерфейсом, де перемикання між сторінками здійснюється без повного перезавантаження сторінки. React Router Dom використовує концепцію маршрутів (routes) і компонентів для визначення та обробки різних URL в додатку.

Основні особливості React Router Dom:

– Маршрути (Routes) – дозволяють визначати різні шляхи (URL) в додатку та відповідні компоненти, які повинні бути відображені для кожного шляху;

– Посилання (Link) – компонент для створення навігаційних посилань, які дозволяють користувачам переміщуватися між сторінками без повного перезавантаження;

– Перенаправлення (Redirect) – механізм для автоматичного перенаправлення користувачів з одного шляху на інший;

– Динамічні маршрути – підтримка маршрутів з параметрами, що дозволяє створювати більш гнучкі та динамічні інтерфейси;

– Захищені маршрути – можливість обмежувати доступ до певних маршрутів на основі умов, таких як аутентифікація користувачів.

Переваги React Router Dom для веброзробки:

– Простота у використанні – React Router Dom забезпечує простий і зрозумілий інтерфейс для створення маршрутів та управління навігацією у додатку;

– Динамічні інтерфейси – можливість створювати динамічні інтерфейси з підтримкою параметрів URL та умовної навігації;

– Керована навігація – дозволяє легко керувати переходами між сторінками, включаючи програмне перенаправлення та захист маршрутів;

– SEO-переваги – підтримка серверного рендерингу (SSR) у поєднанні з React Router Dom може покращити індексацію сторінок пошуковими системами;

– Масштабованість – підтримка складних структур маршрутів, що дозволяє створювати великі додатки з багаторівневою навігацією.

Застосування у проекті:

У додатку React Router Dom використовується для управління маршрутизацією та навігацією між різними сторінками.

Axios

Axios — це популярна бібліотека JavaScript для здійснення HTTP-запитів, що дозволяє здійснювати асинхронні операції з сервером з браузера або Node.js. Axios підтримує всі основні методи HTTP, такі як GET, POST, PUT, DELETE, і дозволяє легко працювати з API.

Основні особливості Axios:

– Підтримка всіх методів HTTP – Axios дозволяє здійснювати всі основні методи HTTP-запитів, такі як GET, POST, PUT, DELETE, PATCH та інші;

– Підтримка промісів – Axios використовує проміси (Promises), що забезпечує зручну та зрозумілу обробку асинхронних операцій;

– Автоматичне перетворення даних – Axios автоматично перетворює JSON-дані у відповідь, що спрощує роботу з даними, отриманими з сервера;

– Підтримка перехоплювачів (interceptors) – можливість налаштування обробки запитів і відповідей через перехоплювачі, що дозволяє додатково обробляти дані або додавати логіку безпосередньо перед відправленням або після отримання відповіді;

– Обробка помилок – вбудована підтримка обробки помилок, що дозволяє легко керувати помилками під час здійснення HTTP-запитів;

– Підтримка запитів з параметрами – Axios дозволяє легко додавати параметри до запитів, що спрощує роботу з RESTful API;

– Підтримка скасування запитів – можливість скасування запитів, що є корисним для обробки ситуацій, коли запити більше не актуальні (наприклад, при навігації або автоматичному оновленні);

– Підтримка тайм-аутів – можливість налаштування часу очікування для запитів, що дозволяє уникнути довгих очікувань у разі проблем із сервером;

– Підтримка завантаження та відправки файлів – Axios дозволяє легко завантажувати файли на сервер та отримувати файли з сервера, що спрощує роботу з мультимедійними даними.

Переваги Axios для веброзробки:

– Простота у використанні – Axios забезпечує простий та зрозумілий синтаксис для здійснення HTTP-запитів, що спрощує взаємодію з сервером;

– Асинхронність – Axios дозволяє здійснювати асинхронні запити, що не блокують головний потік виконання, забезпечуючи швидкий відгук користувача;

– Обробка відповідей – Axios автоматично перетворює JSON-дані у відповідь, що спрощує роботу з даними, отриманими з сервера;

– Підтримка запитів з параметрами – Axios дозволяє легко додавати параметри до запитів, що спрощує роботу з RESTful API;

– Обробка помилок – вбудована підтримка обробки помилок, що дозволяє легко керувати помилками під час здійснення HTTP-запитів;

– Сумісність з Node.js – Axios можна використовувати як у браузері, так і на сервері з Node.js, що забезпечує універсальність бібліотеки.

Застосування у проекті:

У проекті Axios використовується для здійснення HTTP-запитів до бекенд-сервера, що дозволяє інтерактивно обмінюватися даними між клієнтською та серверною частинами додатку.

Zod

Zod — це TypeScript-перевірка схем та бібліотека валідатора, яка дозволяє розробникам визначати структури даних та перевіряти їх відповідність цим структурам. Zod забезпечує простий і зрозумілий спосіб валідації даних, що надходять від користувачів, з API або з інших джерел.

Основні особливості Zod:

– Декларативний синтаксис – простий і зрозумілий спосіб визначення схем даних з використанням декларативного синтаксису;

– Статична типізація – підтримка статичної типізації TypeScript, що дозволяє використовувати визначені схеми для забезпечення коректності типів у всьому коді;

– Композиція схем – можливість комбінувати та вкладати схеми для створення складних структур даних;

– Асинхронна валідація – підтримка асинхронної валідації, що дозволяє виконувати перевірки, які потребують асинхронних операцій;

– Автоматичне виведення типів – можливість автоматично виводити TypeScript-типи з визначених схем, що знижує кількість повторюваного коду та покращує підтримку типів;

– Розширюваність – легкість розширення та налаштування правил валідації відповідно до потреб проекту;

– Зрозумілі повідомлення про помилки – забезпечення зрозумілих та інформативних повідомлень про помилки валідації, що спрощує діагностику проблем.

Переваги Zod для веброзробки:

– Підвищена надійність коду – завдяки статичній типізації та валідації на етапі компіляції забезпечується висока надійність коду;

– Зниження кількості помилок – автоматична валідація даних дозволяє зменшити кількість помилок, пов'язаних з некоректними даними;

– Поліпшена підтримка типів – можливість автоматично виводити TypeScript-типи з визначених схем покращує підтримку типів у всьому коді;

– Зручність використання – декларативний синтаксис та зрозумілі повідомлення про помилки роблять Zod зручним інструментом для валідації даних;

– Гнучкість – можливість легко налаштовувати та розширювати правила валідації відповідно до вимог проекту.

Застосування у проекті:

У додатку Zod використовується для валідації даних, що надходять від користувачів та API, що забезпечує їх відповідність визначеним схемам.

Tailwind

Tailwind CSS — це утилітарний CSS-фреймворк, який дозволяє швидко створювати стильні та адаптивні інтерфейси без написання звичайного CSS. Замість того, щоб визначати стилі у файлах CSS, розробники використовують класи безпосередньо в HTML для застосування стилів.

Основні особливості Tailwind CSS:

– Утилітарні класи – набір низькорівневих класів, таких як text-center, mt-4, text-blue-500, які можна комбінувати для створення складних дизайнів без написання власного CSS;

– Конфігурація через файли – можливість налаштовувати фреймворк через конфігураційні файли, дозволяючи змінювати кольори, відступи, шрифти та інші параметри для відповідності дизайну проекту;

– Вбудована адаптивність – підтримка адаптивного дизайну через використання утилітарних класів, що робить розробку для різних розмірів екранів простою та інтуїтивною;

– Зручна кастомізація – можливість легко налаштовувати та розширювати стилі за допомогою плагінів або власних утилітарних класів;

– Висока продуктивність – автоматична оптимізація CSS під час збірки, що дозволяє видаляти невикористовувані стилі та зменшувати розмір файлів CSS;

– Інтеграція з популярними фреймворками – можливість легко інтегрувати Tailwind CSS з такими фреймворками як React, Vue.js, Angular та іншими.

Переваги Tailwind CSS для веброзробки:

– Швидкість розробки – використання утилітарних класів дозволяє швидко застосовувати стилі без написання власного CSS, що значно прискорює процес розробки;

– Гнучкість – можливість комбінування утилітарних класів для створення складних та унікальних дизайнів без обмежень;

– Узгодженість дизайну – використання одного набору утилітарних класів забезпечує узгодженість стилів у всьому проекті;

– Легка підтримка – зміни в дизайні можуть бути легко впроваджені шляхом редагування HTML-класів, без необхідності зміни CSS-файлів;

– Висока продуктивність – автоматичне видалення невикористовуваних стилів під час збірки зменшує розмір кінцевих файлів CSS, що покращує продуктивність завантаження сторінок.

Застосування у проекті:

У проекті Tailwind CSS використовується для швидкого створення стильних і адаптивних інтерфейсів. Завдяки використанню утилітарних класів, є можливість легко та швидко налаштовувати зовнішній вигляд додатку, забезпечуючи при цьому узгодженість дизайну та високу продуктивність.

DaisyUI

DaisyUI — це бібліотека компонентів для Tailwind CSS, яка забезпечує готові стилізовані компоненти для швидкого створення інтерфейсів користувача. DaisyUI значно спрощує процес розробки, надаючи набір готових стилів для компонентів, таких як кнопки, форми, модальні вікна, картки та багато іншого.

Основні особливості DaisyUI:

– Інтеграція з Tailwind CSS – DaisyUI розширює можливості Tailwind CSS, додаючи стилізовані компоненти, які легко інтегруються в існуючі проекти на базі Tailwind;

– Набір готових компонентів – забезпечує готові до використання компоненти, такі як кнопки, форми, алерти, модальні вікна, меню, картки та інші елементи інтерфейсу;

– Легка кастомізація – можливість налаштовувати зовнішній вигляд компонентів через зміну конфігураційних файлів Tailwind CSS або використання власних утилітарних класів;

– Підтримка тем – DaisyUI дозволяє легко змінювати теми дизайну додатку, використовуючи вбудовану систему тем;

– Висока продуктивність – забезпечує оптимізований розмір CSS-файлів завдяки автоматичному видаленню невикористовуваних стилів під час збірки;

– Документація та приклади – надає детальну документацію та приклади використання компонентів, що спрощує їх інтеграцію та налаштування.

Переваги DaisyUI для веброзробки:

– Швидкий старт – використання готових компонентів дозволяє швидко розпочати розробку інтерфейсу користувача без необхідності створення стилів з нуля;

– Узгодженість дизайну – забезпечує узгоджений та професійний вигляд інтерфейсу завдяки використанню єдиної стилістичної системи;

– Зниження кількості коду – зменшує кількість написаного CSS-коду, оскільки більшість стилів вже реалізовані у компонентах DaisyUI;

– Зручність кастомізації – легко налаштовувати зовнішній вигляд компонентів для відповідності специфічним вимогам проекту;

– Підтримка адаптивного дизайну – всі компоненти DaisyUI адаптовані для роботи на різних розмірах екранів, що забезпечує відмінний користувацький досвід на мобільних пристроях;

– Відкритий вихідний код – DaisyUI є бібліотекою з відкритим вихідним кодом, що дозволяє спільноті розробників вносити свої покращення та доповнення.

Застосування у проекті:

У додатку DaisyUI використовується для швидкого створення та стилізації інтерфейсів користувача. Використання готових компонентів DaisyUI дозволяє значно прискорити процес розробки, забезпечуючи при цьому узгодженість та професійний вигляд інтерфейсу. Завдяки можливості легкої кастомізації, є можливість адаптувати компоненти для відповідності специфічним вимогам нашого проекту, забезпечуючи користувачам зручний та естетично приємний досвід роботи з додатком.

Серверні технології та мови програмування:

Node.js

Node.js — це середовище виконання JavaScript на сервері, яке дозволяє розробникам використовувати JavaScript для серверної розробки. Node.js був створений Райаном Далем у 2009 році і базується на движку V8 від Google, який використовується в браузері Chrome. Node.js підтримує асинхронну, неблокуючу архітектуру вводу/виводу, що робить його ідеальним для створення масштабованих мережевих додатків.

Основні особливості Node.js:

– Асинхронна, неблокуюча модель вводу/виводу – забезпечує високу продуктивність та здатність обробляти велику кількість одночасних з'єднань;

– Однопоточний, подієво-орієнтований цикл – Node.js використовує однопоточну модель з циклом подій, що дозволяє ефективно керувати одночасними операціями;

– Велика екосистема модулів – NPM (Node Package Manager) є одним з найбільших репозиторіїв бібліотек та модулів, що дозволяє легко розширювати функціональність додатків;

– Підтримка різних протоколів – Node.js підтримує HTTP, HTTPS, TCP, UDP, DNS та інші протоколи, що робить його універсальним інструментом для мережевих додатків;

– Легка інтеграція з іншими технологіями – Node.js легко інтегрується з різними базами даних, фреймворками та сервісами, що забезпечує гнучкість у розробці додатків.

Переваги Node.js:

– Висока продуктивність – завдяки асинхронній моделі вводу/виводу та ефективному використанню ресурсів, Node.js забезпечує високу продуктивність додатків;

– Масштабованість – можливість обробляти велику кількість одночасних з'єднань робить Node.js ідеальним для створення масштабованих мережевих додатків;

– Єдина мова програмування – використання JavaScript як на клієнтській, так і на серверній стороні спрощує розробку та підтримку коду;

– Швидка розробка – велика кількість готових модулів та бібліотек в NPM дозволяє швидко додавати нові функціональні можливості;

– Активна спільнота – велика та активна спільнота розробників забезпечує підтримку та постійне вдосконалення платформи;

– Підтримка сучасних стандартів – Node.js підтримує сучасні стандарти JavaScript та ECMAScript, що дозволяє використовувати новітні можливості мови.

Застосування у проекті:

У проекті Node.js використовується для реалізації серверної частини додатку. Це забезпечує швидку та ефективну обробку запитів від клієнтів, взаємодію з базою даних, а також управління аутентифікацією та авторизацією користувачів.

Express

Express — це мінімалістичний вебфреймворк для Node.js, який забезпечує набір інструментів і функцій для побудови вебдодатків і API. Він був розроблений TJ Holowaychuk і випущений у 2010 році. Express забезпечує простий і гнучкий підхід до створення серверних додатків, дозволяючи розробникам швидко і легко створювати маршрути, обробляти запити та відповіді.

Основні особливості Express:

– Легка конфігурація – простий і зрозумілий синтаксис для налаштування маршрутизації та обробки запитів;

– Підтримка середовищ – можливість легко налаштовувати додаток для різних середовищ (розробка, тестування, продакшн);

– Інтеграція з middleware – підтримка додавання проміжних обробників для обробки запитів, відповідей та інших операцій;

– Широка підтримка шаблонів – можливість використовувати різні системи шаблонів для генерації HTML-сторінок;

– Потужна маршрутизація – гнучка система маршрутизації, що дозволяє визначати маршрути для обробки різних HTTP-запитів;

– Підтримка REST API – забезпечує простий спосіб створення RESTful API для взаємодії з клієнтами.

Переваги Express:

– Простота використання – мінімалістичний підхід і простий синтаксис роблять Express легким у використанні для розробників усіх рівнів;

– Гнучкість – можливість додавання middleware для обробки запитів і відповідей, що дозволяє легко розширювати функціональність додатку;

– Швидкість розробки – забезпечує швидку розробку серверних додатків завдяки своїй простоті та широкій екосистемі плагінів і модулів;

– Велика спільнота – активна спільнота розробників забезпечує доступ до великої кількості ресурсів, прикладів та документації;

– Масштабованість – можливість створення як невеликих, так і великих додатків з великою кількістю маршрутів і middleware;

– Інтеграція з іншими технологіями – легко інтегрується з різними базами даних, системами автентифікації та іншими інструментами.

Застосування у проекті:

У проекті Express використовується для побудови серверної частини вебдодатку. Це забезпечує обробку HTTP-запитів, маршрутизацію та взаємодію з базою даних.

TypeScript

TypeScript — це надбудова над JavaScript, яка додає статичну типізацію та сучасні можливості мови до стандартного JavaScript. Вона була розроблена компанією Microsoft і вперше випущена в 2012 році. TypeScript компілюється у звичайний JavaScript, що робить його сумісним з усіма існуючими JavaScript-рушіями та середовищами.

Основні особливості TypeScript:

– Статична типізація – дозволяє визначати типи змінних, функцій, об'єктів тощо, що допомагає виявляти помилки на етапі компіляції;

– Підтримка сучасних стандартів JavaScript – включає в себе всі новітні функції ECMAScript, що дозволяє використовувати сучасні можливості мови;

– Сумісність з JavaScript – TypeScript є надбудовою над JavaScript, тому будь-який код JavaScript є коректним кодом TypeScript;

– Інтерфейси та типи – можливість визначати інтерфейси та типи, що робить код більш зрозумілим та підтримуваним;

– Простори імен та модулі – підтримка просторів імен та модулів для організації коду та зменшення конфліктів між іменами;

– Розширені можливості ООП – підтримка класів, інтерфейсів, наслідування, абстрактних класів та інших концепцій об'єктно-орієнтованого програмування.

Переваги TypeScript:

– Підвищена надійність коду – статична типізація допомагає виявляти та виправляти помилки на етапі компіляції, що підвищує надійність коду;

– Поліпшена підтримка великих проектів – TypeScript полегшує роботу з великими кодовими базами завдяки можливості визначення чітких типів та інтерфейсів;

– Інтелектуальна допомога IDE – завдяки статичній типізації, багато IDE (наприклад, Visual Studio Code) можуть надавати більш точні підказки, автозаповнення та рефакторинг коду;

– Краща підтримка рефакторингу – статична типізація та інші можливості TypeScript полегшують процес рефакторингу, роблячи його більш безпечним та ефективним;

– Сумісність з існуючим JavaScript-кодом – можливість поступового впровадження TypeScript у проект, що вже існує, без необхідності переписувати весь код з нуля;

– Документація коду – використання типів та інтерфейсів робить код самодокументованим, що полегшує його розуміння та підтримку.

Застосування у проекті:

У проекті TypeScript використовується для розробки серверної частини вебдодатку. Це забезпечує надійність, зрозумілість та підтримуваність коду.

Dotenv

Dotenv — це невеликий модуль для Node.js, який завантажує змінні середовища з файлу .env у process.env. Цей модуль дозволяє зберігати налаштування конфігурації та секретні дані окремо від коду, що підвищує безпеку та зручність управління конфігурацією проекту.

Основні особливості Dotenv:

– Простота використання – Dotenv легко інтегрується у проект, вимагаючи лише підключення модуля та створення файлу .env;

– Зберігання конфіденційних даних – можливість зберігати конфіденційні дані, такі як ключі API, паролі та інші налаштування, поза кодом додатку;

– Різні конфігурації для різних середовищ – підтримка різних файлів конфігурації для різних середовищ (розробка, тестування, продакшн), що спрощує управління налаштуваннями;

– Відокремлення конфігурації від коду – зберігання налаштувань у файлі .env дозволяє відокремити конфігурацію від коду, що підвищує безпеку та спрощує налаштування проекту;

– Підтримка стандартних форматів – використання простого текстового формату для визначення змінних середовища, що полегшує їх читання та редагування.

Переваги Dotenv:

– Підвищена безпека – зберігання конфіденційних даних окремо від коду знижує ризик витоку даних;

– Спрощене управління конфігурацією – зручність зберігання та зміни налаштувань у файлі .env, що робить процес налаштування проекту більш організованим;

– Легка інтеграція – простий процес інтеграції Dotenv у проект, що дозволяє швидко почати використовувати цей інструмент для управління конфігурацією;

– Підтримка різних середовищ – можливість легко перемикатися між різними конфігураціями для розробки, тестування та продакшн середовищ;

– Чітка структура проекту – відокремлення конфігурації від коду допомагає підтримувати чисту та організовану структуру проекту.

Застосування у проекті:

У проекті Dotenv використовується для управління конфігурацією та зберігання конфіденційних даних, таких як налаштування бази даних, ключі API для OpenAI та інші змінні середовища.

Bcryptjs

Bcryptjs — це бібліотека JavaScript, яка забезпечує хешування паролів з використанням алгоритму bcrypt. Вона призначена для використання у Node.js-додатках і дозволяє розробникам легко створювати надійні хеші для паролів користувачів, що підвищує безпеку зберігання та обробки паролів.

Основні особливості Bcryptjs:

– Хешування паролів – забезпечує хешування паролів з використанням алгоритму bcrypt, що робить паролі захищеними від зламування;

– Сіль (salt) – автоматично генерує унікальну сіль для кожного пароля, що забезпечує додатковий рівень безпеки;

– Перевірка паролів – дозволяє порівнювати введений користувачем пароль з його хешем для перевірки аутентифікації;

– Конфігурація складності хешування – можливість налаштування складності (кількості раундів) хешування, що дозволяє балансувати між безпекою та продуктивністю;

– Відсутність залежності від компіляції – bcryptjs є чистою JavaScript-реалізацією, що не потребує компіляції, на відміну від інших реалізацій bcrypt.

Переваги Bcryptjs:

– Висока безпека – алгоритм bcrypt спеціально розроблений для безпечного хешування паролів, забезпечуючи захист від атак типу brute-force та rainbow table;

– Легкість використання – простий та зрозумілий API робить bcryptjs легким у використанні для розробників будь-якого рівня;

– Гнучкість – можливість налаштування складності хешування дозволяє адаптувати бібліотеку під конкретні вимоги проекту;

– Відсутність залежності від компіляції – реалізація на чистому JavaScript дозволяє використовувати бібліотеку без необхідності встановлення додаткових інструментів для компіляції;

– Сумісність – bcryptjs легко інтегрується з іншими бібліотеками та фреймворками, що використовуються у Node.js-додатках.

Застосування у проекті:

У проекті Bcryptjs використовується для забезпечення безпеки зберігання та обробки паролів користувачів. Це допомагає захистити дані користувачів від несанкціонованого доступу та атак.

Mongoose

Mongoose — це об'єктно-документний менеджер (ODM) для MongoDB та Node.js. Він надає простий та зручний інтерфейс для взаємодії з базою даних MongoDB, дозволяючи розробникам працювати з документами та колекціями як з об'єктами JavaScript. Mongoose підтримує валідацію схем, попередні middleware, а також різні методи для зручного маніпулювання даними.

Основні особливості Mongoose:

– Схеми – можливість визначення схем для документів, що дозволяє забезпечити структуру та валідацію даних у колекціях MongoDB;

– Моделі – використання моделей для взаємодії з документами в базі даних, що полегшує виконання CRUD операцій;

– Валідація – вбудована підтримка валідації даних при збереженні документів у базу даних, що забезпечує цілісність даних;

– Middleware – можливість визначення middleware для виконання додаткових операцій під час виконання методів збереження, оновлення або видалення документів;

– Реляційні посилання – підтримка референційних зв'язків між документами, що дозволяє створювати складніші структури даних;

– Запити – потужні можливості для створення запитів до бази даних з використанням методів моделей;

– Плагіни – можливість розширення функціональності Mongoose за допомогою плагінів, що дозволяє додавати нові можливості до моделей і схем.

Переваги Mongoose:

– Полегшена робота з MongoDB – Mongoose забезпечує зручний і зрозумілий інтерфейс для взаємодії з MongoDB, що значно спрощує розробку додатків;

– Валідація даних – вбудована підтримка валідації даних допомагає забезпечити цілісність і коректність даних у базі даних;

– Схеми та моделі – можливість визначення схем і моделей дозволяє створювати структуровані та підтримувані дані;

– Можливості middleware – використання middleware забезпечує додаткову гнучкість і контроль над операціями з документами;

– Розширюваність – підтримка плагінів дозволяє легко розширювати можливості Mongoose відповідно до потреб проекту;

– Потужні запити – зручні методи для створення складних запитів до бази даних дозволяють ефективно працювати з даними.

Застосування у проекті:

У проекті Mongoose використовується для взаємодії з базою даних MongoDB, забезпечуючи структуроване зберігання даних та зручний інтерфейс для виконання операцій CRUD.

Jsonwebtoken

Jsonwebtoken (JWT) — це стандарт для створення токенів доступу, що використовуються для передачі інформації між сторонами як JSON-об'єкт у безпечний та надійний спосіб. JWT широко використовується для аутентифікації та авторизації в вебдодатках. Бібліотека Jsonwebtoken для Node.js дозволяє легко створювати, підписувати та перевіряти JWT.

Основні особливості Jsonwebtoken:

– Створення токенів – можливість створення токенів для передачі інформації між клієнтом і сервером;

– Підписування токенів – забезпечення цілісності та автентичності токенів за допомогою підписування секретним ключем або RSA-ключем;

– Валідація токенів – можливість перевірки токенів на стороні сервера для аутентифікації та авторизації користувачів;

– Термін дії токенів – підтримка встановлення терміну дії токенів, що підвищує безпеку та контролює час доступу;

– Підтримка різних алгоритмів шифрування – можливість використання різних алгоритмів шифрування для підписування токенів, таких як HMAC, RS256 тощо.

Переваги Jsonwebtoken:

– Безпека – забезпечує безпечну передачу інформації між клієнтом і сервером за допомогою підписаних токенів;

– Простота використання – простий та зрозумілий API для створення, підписування та перевірки токенів;

– Відсутність необхідності зберігання стану – дозволяє реалізувати безстанову аутентифікацію, що знижує навантаження на сервери та спрощує масштабування;

– Гнучкість – підтримка різних алгоритмів шифрування та можливість встановлення різних атрибутів у токенах;

– Сумісність – широко підтримується у різних мовах програмування та платформах, що робить його універсальним рішенням для аутентифікації.

Застосування у проекті:

У проекті Jsonwebtoken використовується для реалізації аутентифікації та авторизації користувачів, забезпечуючи безпеку та зручність доступу до ресурсів вебдодатку.

Openai

OpenAI — це бібліотека для роботи з API OpenAI, яка надає розробникам зручний інтерфейс для інтеграції передових моделей штучного інтелекту у свої додатки. Цей пакет дозволяє легко взаємодіяти з сервісами OpenAI, такими як GPT (Generative Pre-trained Transformer), забезпечуючи доступ до потужних можливостей AI для генерації тексту, обробки природної мови та інших завдань.

Основні особливості пакета OpenAI:

– Інтеграція з API OpenAI – забезпечує зручний інтерфейс для взаємодії з API OpenAI, дозволяючи легко використовувати моделі штучного інтелекту у додатках;

– Підтримка різних моделей – пакет підтримує роботу з різними моделями OpenAI, такими як GPT-3 або GPT-4o, що дозволяє виконувати широкий спектр завдань;

– Генерація тексту – можливість створення зв'язного та якісного тексту на основі заданих вхідних даних;

– Обробка природної мови – здатність розуміти та генерувати текст на природній мові, що дозволяє використовувати AI для автоматизації текстових завдань;

– Гнучкість налаштувань – можливість налаштовувати параметри запитів для точного контролю над вихідними даними;

– Легкість використання – простий та зрозумілий API, що робить інтеграцію з OpenAI швидкою та ефективною.

Переваги пакета OpenAI:

– Швидка інтеграція – зручний API дозволяє швидко інтегрувати моделі OpenAI у додатки, забезпечуючи доступ до потужних можливостей AI;

– Висока якість та точність – моделі OpenAI забезпечують високу якість та точність виконання завдань завдяки передовим алгоритмам машинного навчання;

– Гнучкість – можливість налаштовувати параметри запитів для досягнення бажаних результатів у різних сценаріях використання;

– Автоматизація завдань – пакет дозволяє автоматизувати рутинні текстові завдання, що підвищує ефективність роботи та знижує навантаження на розробників;

– Підтримка різноманітних завдань – можливість виконувати широкий спектр завдань, від генерації тексту до обробки природної мови, робить пакет універсальним інструментом для розробки.

Застосування у проекті:

У проекті пакет OpenAI використовується для інтеграції інтелектуальних функцій, що забезпечують автоматичну генерацію навчальних матеріалів, відповіді на запити користувачів та аналіз текстів.

Nodemon

Nodemon — це інструмент для автоматичного перезавантаження Node.js додатків під час розробки. Він спостерігає за файлами у проекті і автоматично перезапускає сервер, коли виявляє зміни. Це дозволяє розробникам працювати більш ефективно, оскільки їм не потрібно вручну зупиняти і перезапускати сервер після кожної зміни в коді.

Основні особливості Nodemon:

– Автоматичне перезавантаження – Nodemon автоматично перезапускає сервер при виявленні змін у файлах проекту;

– Підтримка різних мов і конфігурацій – Nodemon може бути налаштований для спостереження за файлами різних типів і використовувати різні команди для перезапуску додатка;

– Легкість у використанні – простий у налаштуванні та інтеграції з існуючими проектами Node.js;

– Сповіщення про помилки – Nodemon повідомляє про помилки, що виникають під час виконання додатка, що допомагає швидше знаходити і виправляти помилки.

Переваги Nodemon:

– Підвищена продуктивність – автоматичне перезавантаження сервера значно прискорює процес розробки, оскільки розробникам не потрібно вручну перезапускати сервер після кожної зміни;

– Легкість налаштування – Nodemon легко інтегрується у будь-який проект Node.js з мінімальними зусиллями;

– Сумісність – Nodemon працює з усіма популярними фреймворками та бібліотеками для Node.js, що робить його універсальним інструментом для розробників.

Застосування у проекті:

У проекті Nodemon використовується для автоматизації процесу перезавантаження сервера під час розробки, що підвищує продуктивність розробника і спрощує відлагодження коду.

Docker

Docker — це платформа для розробки, доставки та запуску додатків у контейнерах. Контейнери дозволяють розробникам упакувати додаток з усіма його залежностями та конфігураціями в один ізольований блок, який може працювати однаково на будь-якому середовищі. Це забезпечує портативність, стабільність та ефективність роботи додатків.

Основні особливості Docker:

– Контейнери – забезпечують ізольоване середовище для додатків, що включає всі необхідні залежності та конфігурації;

– Docker Hub – репозиторій для зберігання та обміну контейнерами, що дозволяє легко розповсюджувати та використовувати готові контейнери;

– Docker Compose – інструмент для визначення та запуску багатоконтейнерних Docker-додатків, що спрощує управління складними додатками;

– Образи Docker – можливість створення образів (images) додатків, що включають всі необхідні компоненти для їх запуску;

– Оркестрація контейнерів – інтеграція з інструментами оркестрації, такими як Kubernetes та Docker Swarm, для автоматизації розгортання, управління та масштабування контейнерних додатків;

– Підтримка різних середовищ – забезпечує однакове середовище для розробки, тестування та продакшн, що гарантує стабільність і передбачуваність додатків.

Переваги Docker:

– Портативність – контейнери забезпечують однакове середовище для додатків незалежно від платформи або інфраструктури, що підвищує портативність додатків;

– Ізоляція – контейнери забезпечують ізоляцію додатків та їхніх залежностей, що знижує ризик конфліктів і підвищує безпеку;

– Швидке розгортання – контейнери дозволяють швидко розгортати додатки та їхні оновлення, що знижує час виходу на ринок;

– Масштабованість – Docker інтегрується з інструментами оркестрації, що дозволяє автоматизувати масштабування додатків відповідно до навантаження;

– Ефективність ресурсів – контейнери використовують системні ресурси ефективніше, ніж віртуальні машини, що знижує витрати на інфраструктуру;

– Легка інтеграція CI/CD – Docker спрощує інтеграцію з процесами безперервної інтеграції та безперервного розгортання (CI/CD), що підвищує ефективність розробки та розгортання додатків.

Застосування у проекті:

У проекті Docker використовується для створення ізольованих середовищ для розробки, тестування та розгортання вебдодатку, що підвищує стабільність та передбачуваність роботи додатку на різних етапах його життєвого циклу.

2.5. Опис структури програми та алгоритмів її функціонування

Структура клієнтської частини додатку представлена на рис. 2.2. Опис файлів та директорій:

– dist – містить згенеровані файли після зборки проекту;

– node_modules – містить встановлені пакети та залежності Node.js;

– src – основна директорія з вихідними файлами проекту:

– api – містить файли для взаємодії з бекендом (наприклад, функції для виконання API-запитів);

– assets – містить статичні файли, такі як зображення, шрифти, тощо;

– components – містить повторно використовувані компоненти React;

– helpers – містить допоміжні функції та утиліти;

– hooks – містить користувацькі хуки React;

– layouts – містить компоненти макетів сторінок;

– pages – містить компоненти сторінок додатку;

– schemas – містить схеми для валідації даних;

– App.jsx – головний компонент додатку, що визначає структуру додатку;

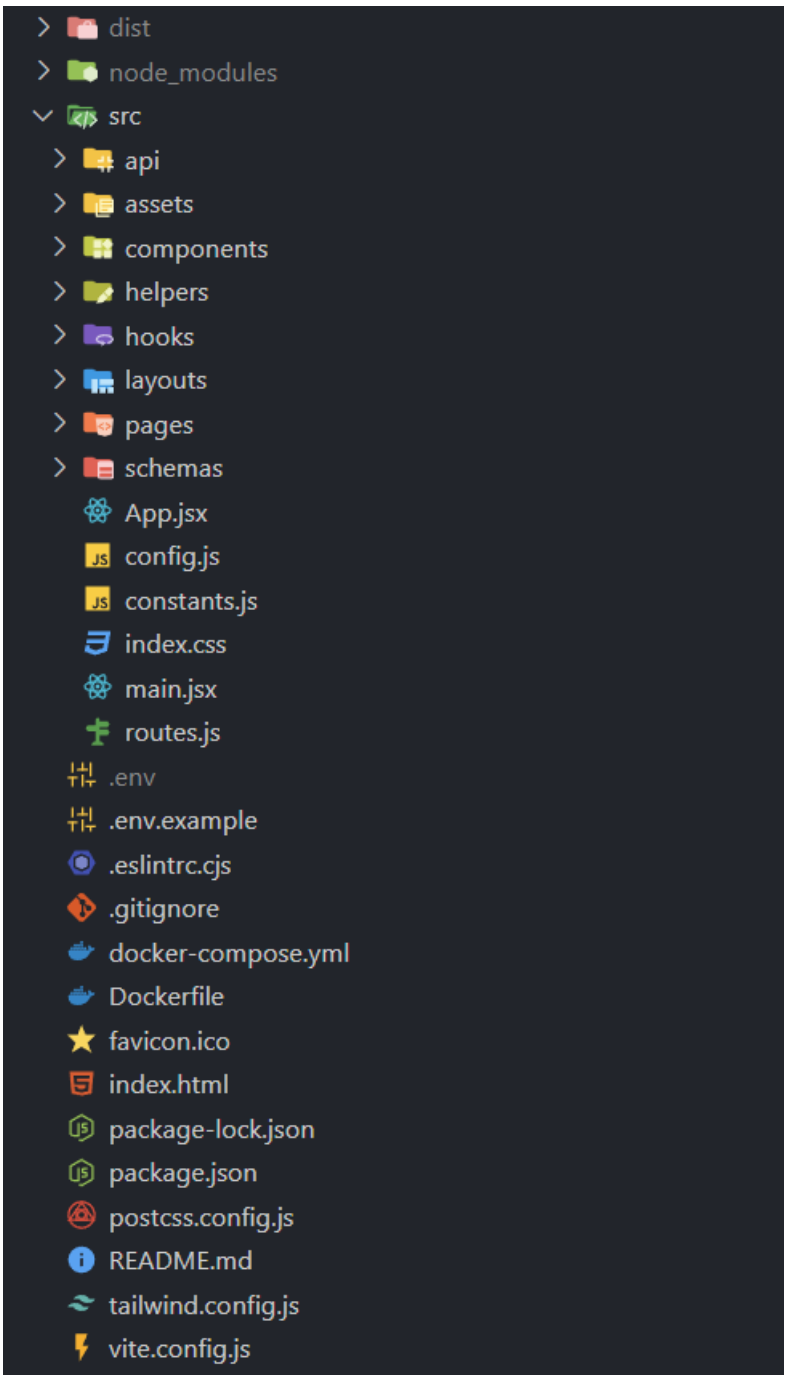

Рис. 2.2. Структура клієнтської частини додатку

– config.js – файл конфігурації, що містить налаштування додатку;

– constants.js – містить константи, які використовуються у додатку;

– index.css – головний файл стилів для додатку;

– main.jsx – початковий файл для запуску React-додатку;

– routes.js – містить маршрутизацію додатку.

– .env – файл налаштувань середовища, що містить конфіденційні дані та налаштування;

– .env.example – приклад файлу .env, що містить шаблон налаштувань середовища;

– .eslintrc.cjs – файл конфігурації ESLint для налаштування правил лінтингу;

– .gitignore – файл, що вказує на файли та директорії, які мають бути ігноровані Git;

– docker-compose.yml – файл конфігурації Docker Compose для запуску кількох контейнерів Docker;

– Dockerfile – файл конфігурації Docker для створення образу Docker;

– favicon.ico – файл з іконкою додатку;

– index.html – основний HTML файл додатку;

– package-lock.json – файл, що містить зафіксовані версії встановлених пакетів та їх залежностей;

– package.json – файл конфігурації Node.js проекту, що містить метадані проекту та залежності;

– postcss.config.js – файл конфігурації PostCSS для обробки CSS файлів;

– README.md – файл документації проекту, що містить інформацію про проект та інструкції з його використання;

– tailwind.config.js – файл конфігурації Tailwind CSS;

– vite.config.js – файл конфігурації Vite для налаштування процесу зборки проекту.

Алгоритм функціонування платформи:

1. В першу чергу налаштовується файл .env за зразком .env.example;
2. Далі виконується команда docker-compose up. За її допомогою створюється контейнер, встановлюються всі залежності та запускається сервер;

3. Додаток готовий приймати користувачів. При відкриті будь-якої сторінки головний файл index.html завантажує JavaScript, який рендерить необхідний контент;
4. Визначення та обробка маршрутів для навігації між різними сторінками додатку;
5. Виконання запитів до бекенду для взаємодії з базою даних та обробки даних.

Структура серверної частини додатку представлена на рис. 2.3. Опис файлів та директорій:

– node_modules – містить встановлені пакети та залежності Node.js;

– src – основна директорія з вихідними файлами проекту;

– controllers – містить контролери, які обробляють вхідні запити та взаємодіють з моделями та сервісами;

– errors – містить файли для обробки помилок та винятків у додатку;

– middlewares – містить проміжні обробники (middleware) для обробки запитів перед їх передачею до кінцевих обробників;

– models – містить моделі даних, які визначають структуру даних у базі даних;

– routes – містить визначення маршрутів додатку, які спрямовують запити до відповідних контролерів;

– schemas – містить схеми для валідації даних;

– services – містить бізнес-логіку та сервіси, що забезпечують функціональність додатку;

– types – містить визначення типів TypeScript, які використовуються у проекті;

– utils – містить допоміжні функції та утиліти для використання в проекті;

– constants.ts – містить константи, які використовуються у додатку;

– db.ts – містить налаштування та функції для взаємодії з базою даних;

– index.ts – початковий файл, який завантажує ініціалізує додаток;

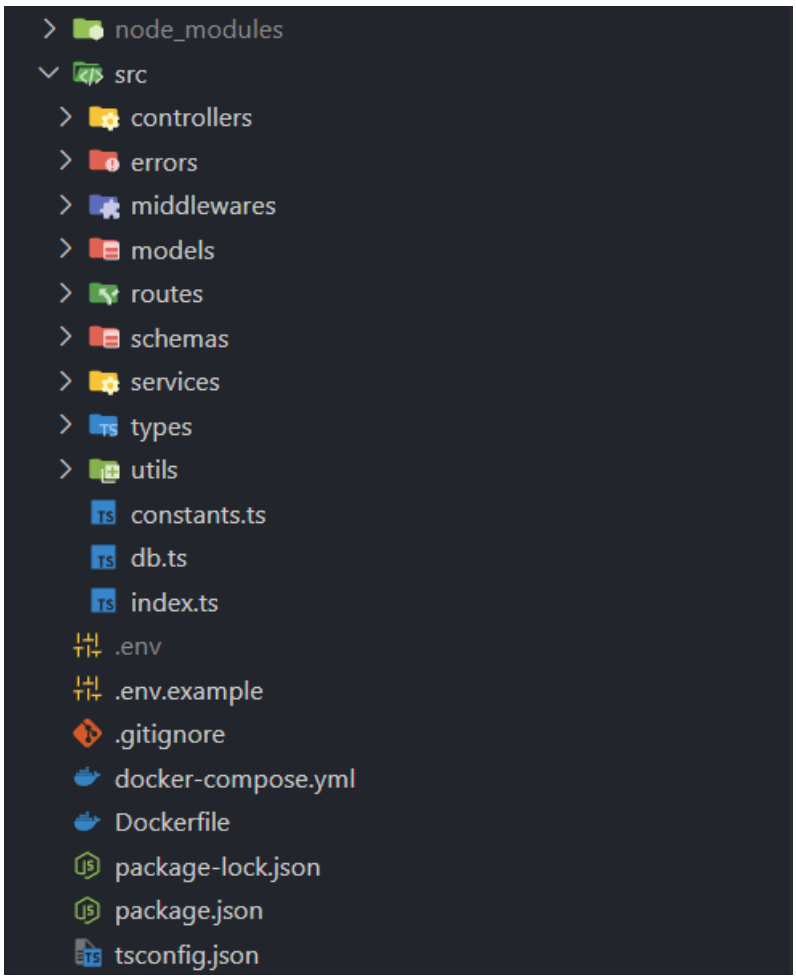

Рис. 2.3. Структура серверної частини додатку

– .env – файл налаштувань середовища, що містить конфіденційні дані та налаштування;

– .env.example – приклад файлу .env, що містить шаблон налаштувань середовища;

– .gitignore – файл, що вказує на файли та директорії, які мають бути ігноровані Git;

– docker-compose.yml – файл конфігурації Docker Compose для запуску кількох контейнерів Docker;

– Dockerfile – файл конфігурації Docker для створення образу Docker;

– package-lock.json – файл, що містить зафіксовані версії встановлених пакетів та їх залежностей;

– package.json – файл конфігурації Node.js проекту, що містить метадані проекту та залежності;

– tsconfig.json – файл конфігурації TypeScript, що визначає налаштування компіляції TypeScript коду.

Алгоритм функціонування платформи:

1. В першу чергу налаштовується файл .env за зразком .env.example;

2. Далі виконується команда docker-compose up. За її допомогою створюються контейнери додатку та бази даних, встановлюються всі залежності та запускається сервер;

3. Додаток готовий приймати запити;

4. Спеціально налаштовані роути обробляють запити і передають їх контролерам;

5. Контролери валідують вхідні дані і викликають з ними методи сервісів;

6. Сервіси виконують необхідну логіку і повертають вихідні дані, які проходять в зворотному порядку через контролери та роути.

Структура бази даних у вебдодатку організована на основі документо-орієнтованої бази даних MongoDB. Вона складається з декількох колекцій, кожна з яких відповідає за зберігання певного типу даних (рис 2.4.). Основні колекції включають:

– users – для зберігання інформації про користувачів;

– generated-exercises – для зберігання інформації про згенеровані та виконані завдання

– dictionaries – для словників на різні теми;

– words – для слів, що є складовими словників;

– grammar-rules – для зберігання граматичних тем;

– writing-tasks – для опису письмових завдань.

Кожна колекція має чітко визначену схему, що забезпечує цілісність даних та ефективність їх обробки. Використання Mongoose як об'єктно-документного менеджера (ODM) дозволяє визначати ці схеми, здійснювати валідацію та

швидку взаємодію з базою даних. Така структура бази даних забезпечує гнучкість, масштабованість та високий рівень продуктивності додатку.

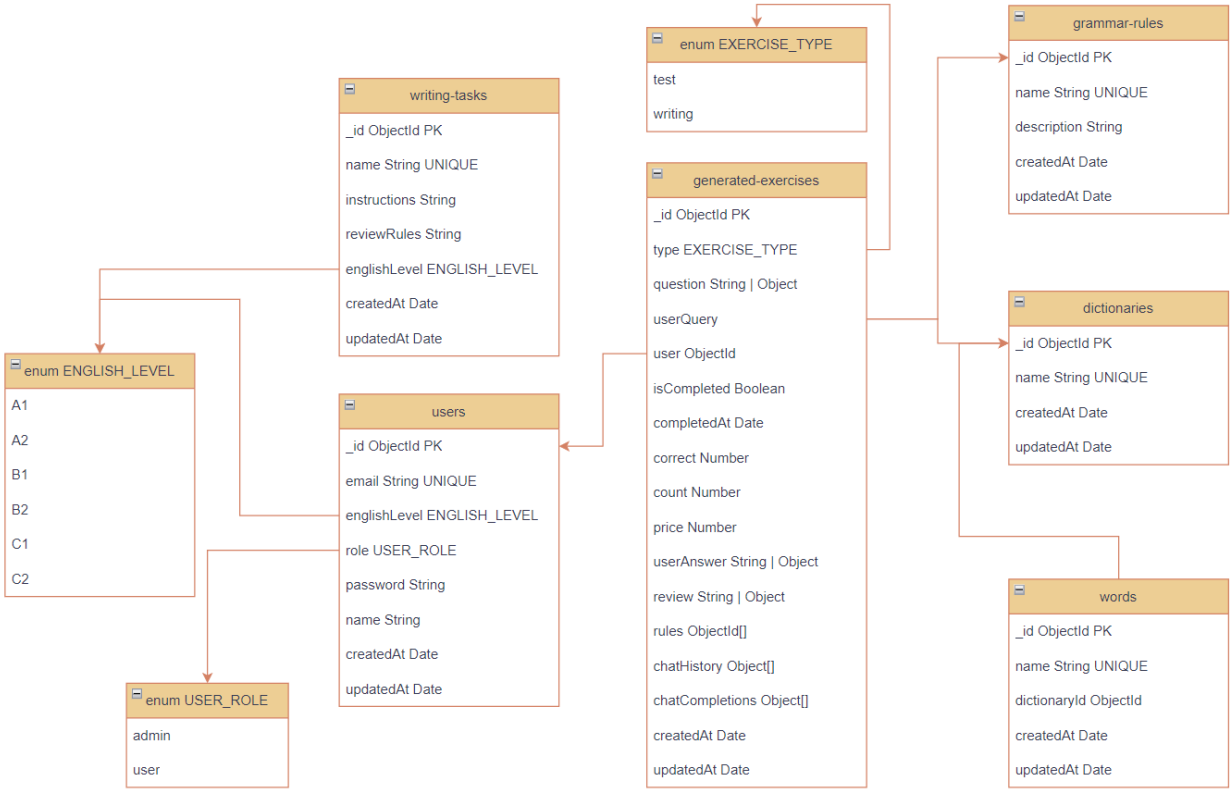

Рис. 2.4. Структура бази даних

2.6. Обґрунтування та організація вхідних та вихідних даних програми

Вхідні та вихідні дані є критичними компонентами будь-якого вебдодатку, забезпечуючи взаємодію між користувачами та сервером. У додатку для вивчення англійської мови вхідні дані надходять від користувачів через форми реєстрації, авторизації, виконання завдань, а також через API-запити. Вихідні дані включають результати тестів, інформацію про профіль користувача, навчальні матеріали та зворотний зв'язок.

Організація вхідних даних починається з їхньої валідації. Для цього використовуються бібліотеки, які забезпечують коректність та безпеку введених даних, запобігаючи можливим атакам та помилкам. Валідація здійснюється на

обох рівнях – клієнтському та серверному, що гарантує додатковий рівень захисту.

Вихідні дані формуються на сервері після обробки запитів та взаємодії з базою даних. Вони повертаються у форматі JSON, що забезпечує зручність їх обробки на клієнтській стороні.

Вибір технологій для організації вхідних та вихідних даних обґрунтований потребою у забезпеченні безпеки, ефективності та надійності додатку. Використання Node.js та Express дозволяє ефективно обробляти HTTP-запити, а MongoDB забезпечує гнучке зберігання даних. Інтеграція бібліотек для валідації, таких як zod, підвищує загальну безпеку додатку.

2.7. Опис розробленого програмного продукту

2.7.1. Використані технічні засоби

При створенні додатку було використано електронно-обчислювальну машину (персональний ноутбук) з наступними характеристиками:

- Процесор Intel Core i7-9750H;
- Відеокарта Nvidia GeForce GTX 1650;
- Оперативна пам’ять 16 ГБ.
- Пам’ять: 512GB.

2.7.2. Використані програмні засоби

У процесі розробки вебдодатку для вивчення англійської мови були використані такі програмні засоби, як Visual Studio Code та MongoDB Compass.

Visual Studio Code — це потужний та універсальний редактор коду, розроблений компанією Microsoft. Він підтримує безліч мов програмування, включаючи JavaScript та TypeScript, що є основними мовами проекту. VS Code забезпечує розширену функціональність за допомогою великої кількості плагінів

та розширень. Інтеграція з системами контролю версій, такими як Git, спрощує управління версіями коду та спільну роботу над проектом.

MongoDB Compass — це графічний інтерфейс для роботи з базою даних MongoDB. Він надає зручні інструменти для дослідження, візуалізації та управління даними у базі даних. За допомогою MongoDB Compass розробники можуть легко переглядати структуру колекцій, виконувати запити, аналізувати індекси та переглядати статистику продуктивності.

Окрім основних інструментів, у проекті використовувались й інші програмні засоби для забезпечення повного циклу розробки:

- Git та GitHub для контролю версій;
- Docker для контейнеризації додатку, що забезпечує консистентність середовища розробки;
- Postman для тестування API.

2.7.3. Виклик та завантаження програми

Для запуску додатку необхідно мати доступ в інтернет, щоб мати можливість взаємодіяти з API OpenAI та встановлений Docker для запуску частин додатку в контейнерах:

1. Перейти в кореневу директорію серверної частини додатку;
2. Налаштувати .env файл за прикладом .env.example;
3. Виконати команду docker-compose up для запуску контейнерів з базою даних та сервером;
4. Перейти в кореневу директорію клієнтської частини програми;
5. Налаштувати .env файл за прикладом .env.example;
6. Виконати команду docker-compose up для запуску контейнера з клієнтською частиною додатку;
7. Перейти за адресою, що буде вказана в консолі.

2.7.4. Опис інтерфейсу користувача

На головній сторінці додатку (рис. 2.5.) знаходиться опис програми з невеликою кількістю інформаційних блоків та кнопками «Get Started», які ведуть на сторінку авторизації (рис. 2.6. - 2.7.).

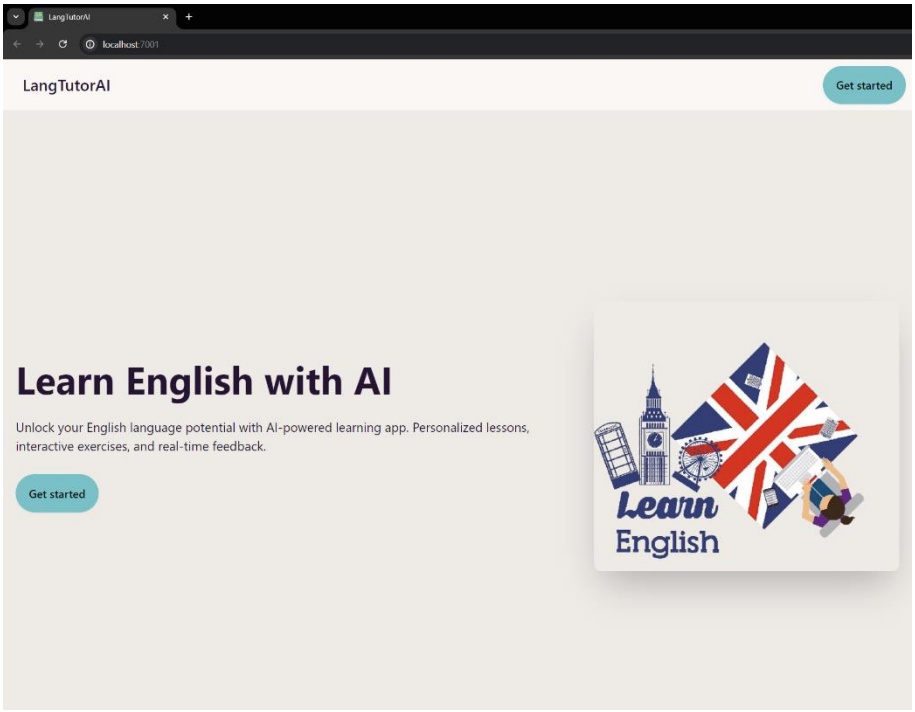

Рис. 2.5. Вигляд головної сторінки

Сторінка авторизації має форму з перемикачем на логін (рис. 2.6.) або реєстрацію (рис. 2.7.). Тут присутня валідація, тому при введенні некоректних даних, поля з помилкою підсвічуються та показують помилку (рис. 2.8.). Також є можливість переглянути введений пароль, натиснувши на іконку ока.

Після успішної авторизації у правому верхньому куті з’являється повідомлення (рис. 2.9.) про це. А у випадку помилки, з’являється нотифікація (рис. 2.10.) про помилку. Користувач переходить на сторінку дашборда (рис. 2.11.), на якій знаходиться статистика виконаних тестів, зроблених письмових завдань, середня правильність відповідей та 2 графіки, що показують кількість виконаних завдань та їх правильність по дням за останній тиждень.

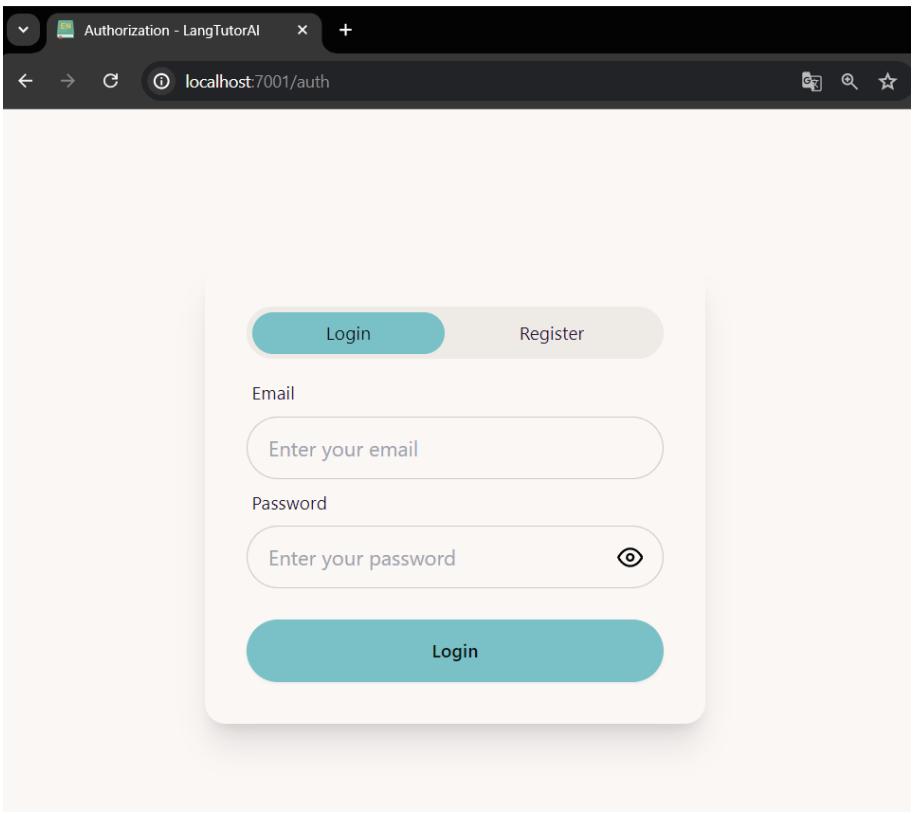

Рис. 2.6. Вигляд сторінки авторизації, коли обрано «Login»

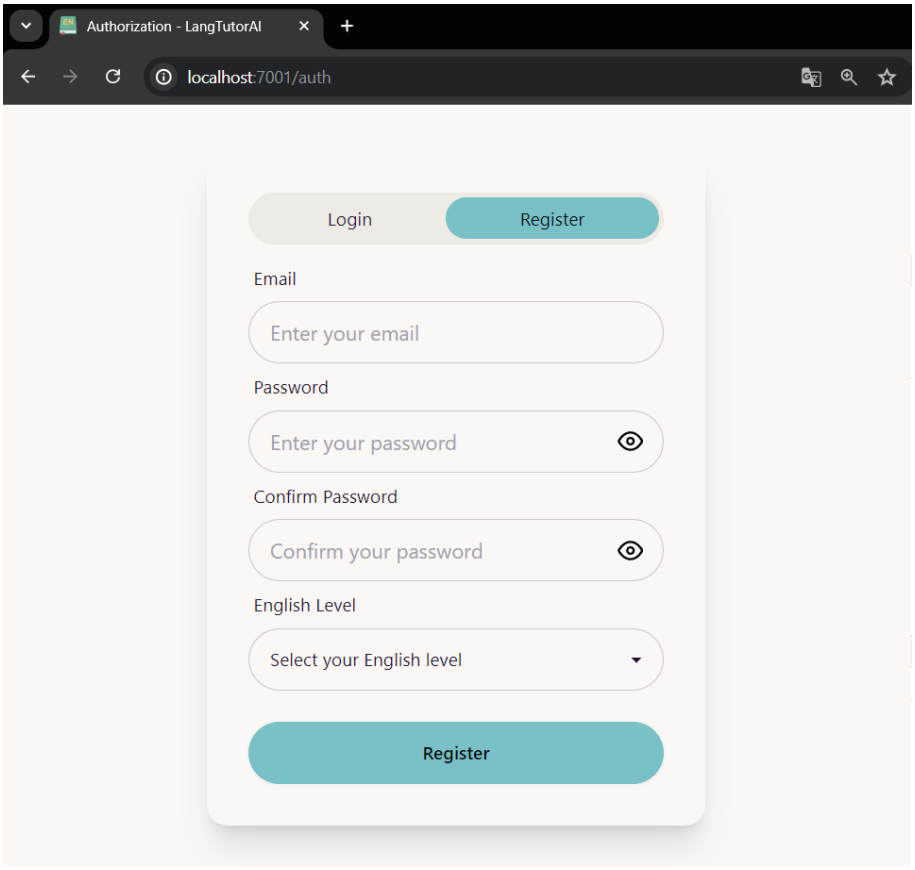

Рис. 2.7. Вигляд сторінки авторизації, коли обрано «Register»

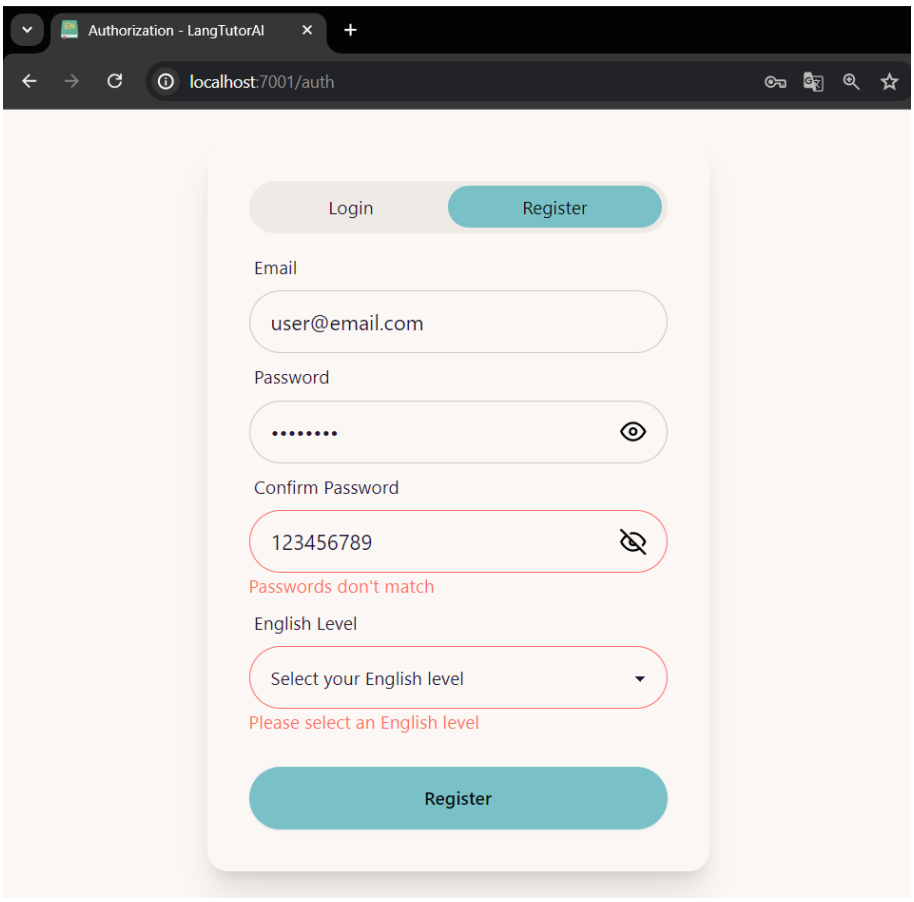

Рис. 2.8. Вигляд полів, що не пройшли валідацію

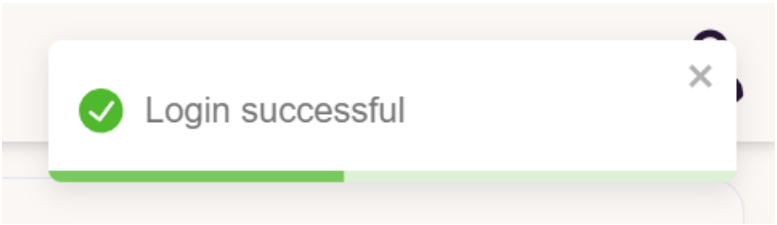

Рис. 2.9. Успішне повідомлення

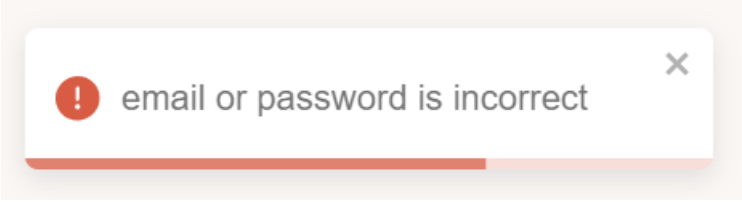

Рис. 2.10. Повідомлення з помилкою

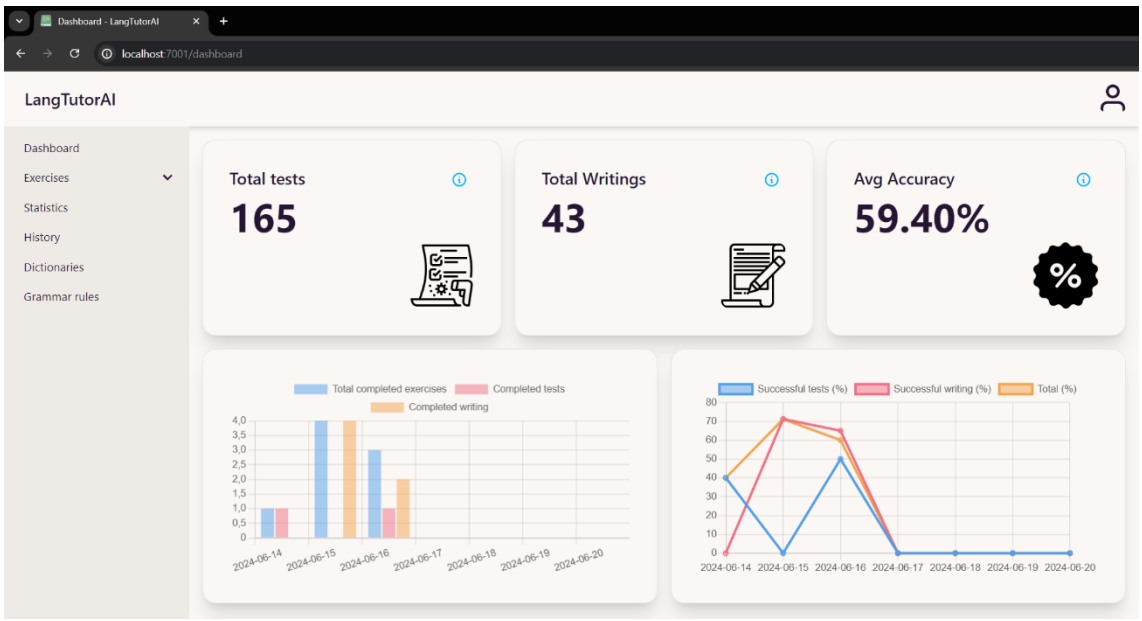

Рис. 2.11. Вигляд сторінки дашборда

Також на сторінці дашборда (рис. 2.11.) можна помітити загальну структуру сторінок додатку, які бачать авторизовані користувачі. Зверху знаходиться хедер з назвою додатку та іконкою користувача. Зліва знаходиться навігаційний список сторінок.

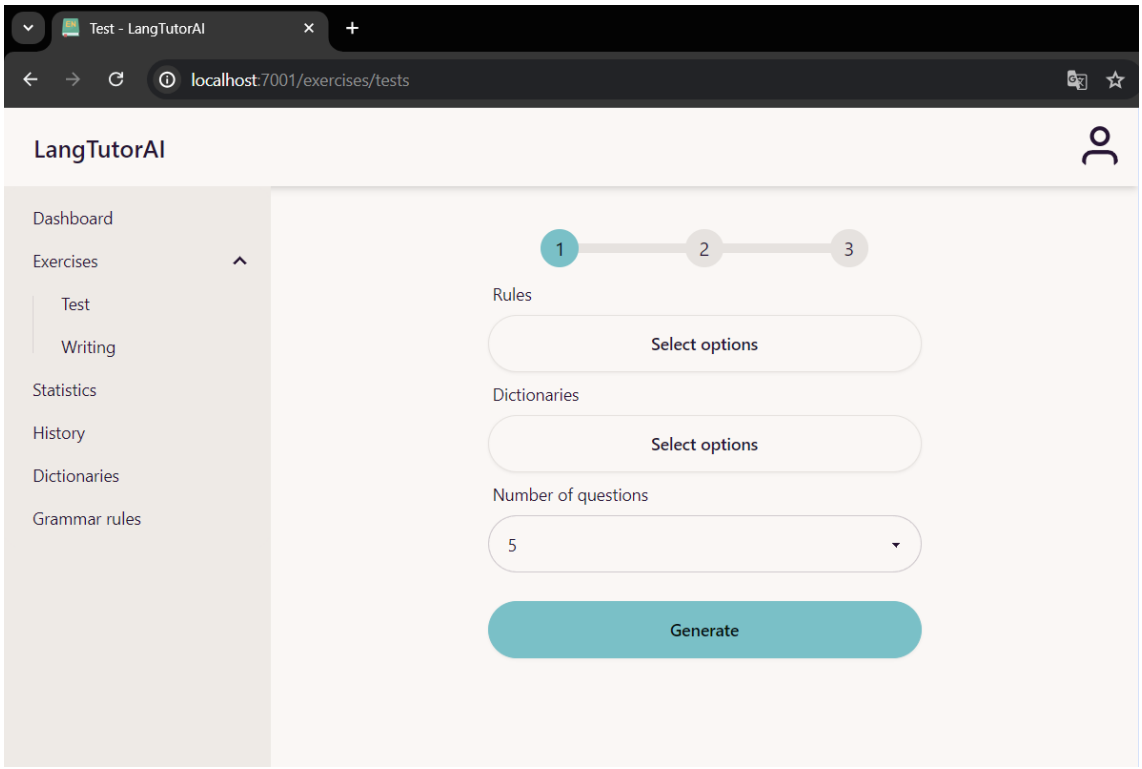

Рис. 2.12. Вигляд сторінки тестування

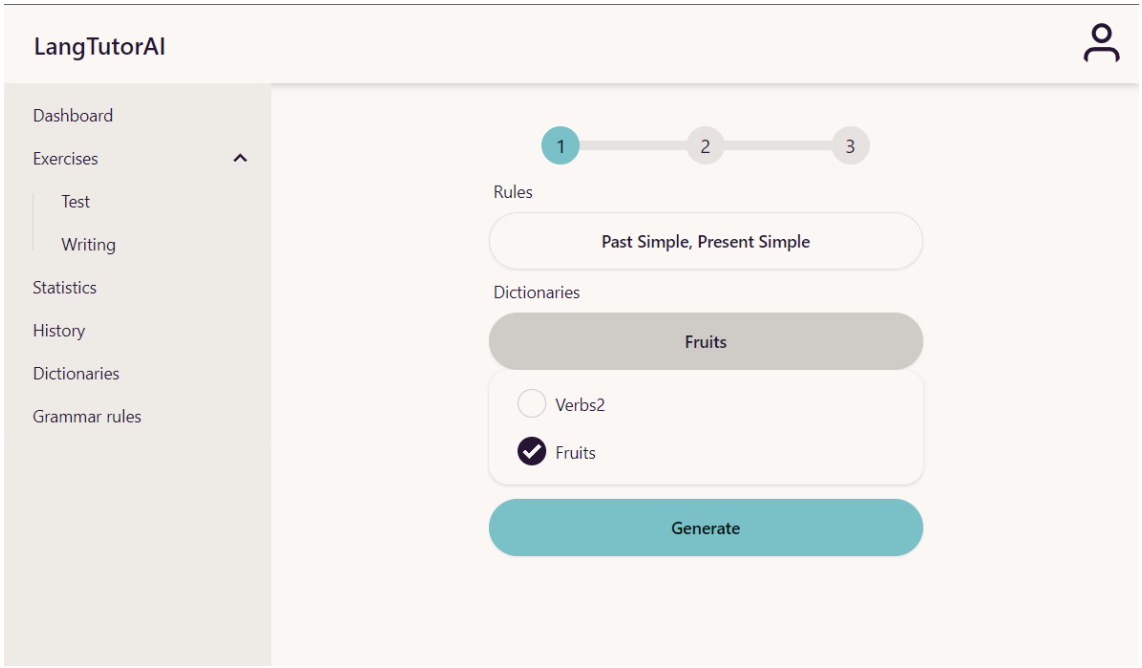

Рис. 2.13. Вигляд обирання тем та словників для тестування

Після натискання на кнопку «Test» у меню, відкривається сторінка виконання тесту (рис. 2.12.), де можна обрати необхідні теми, словники та кількість завдань (рис. 2.13.).

Натискання кнопки «Generate» починає створення тесту і показує екран завантаження (рис 2.14.).

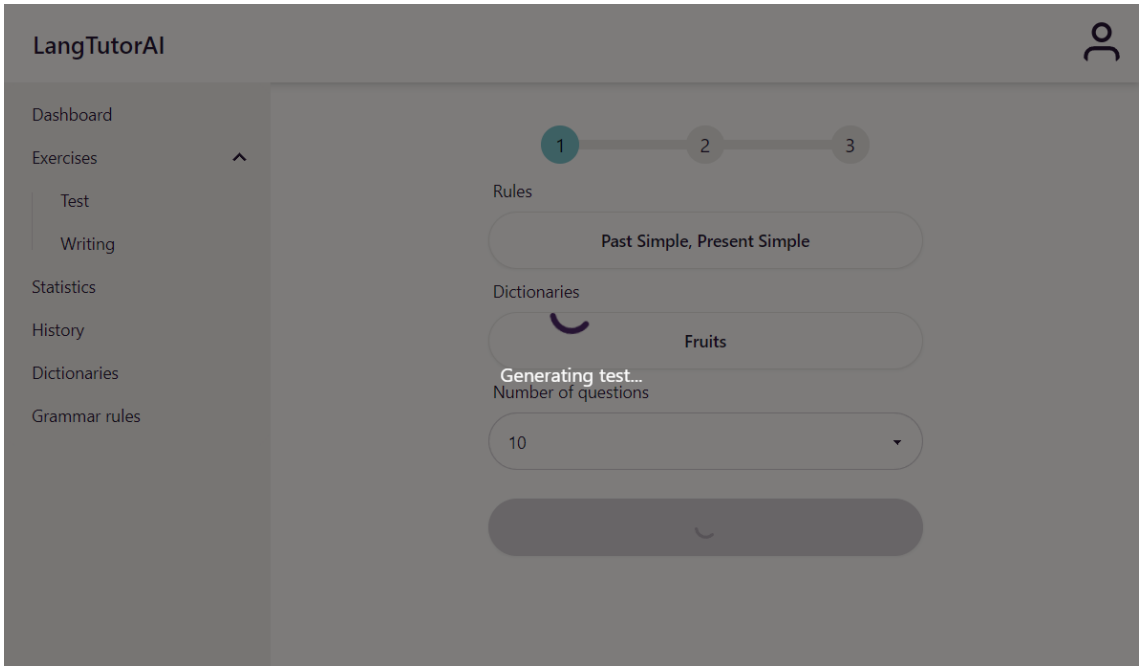

Рис. 2.14. Вигляд екрану завантаження

Після завершення генерації, користувачу відображається тест (рис 2.15.) з можливістю обирати варіанти та перевірити тест.

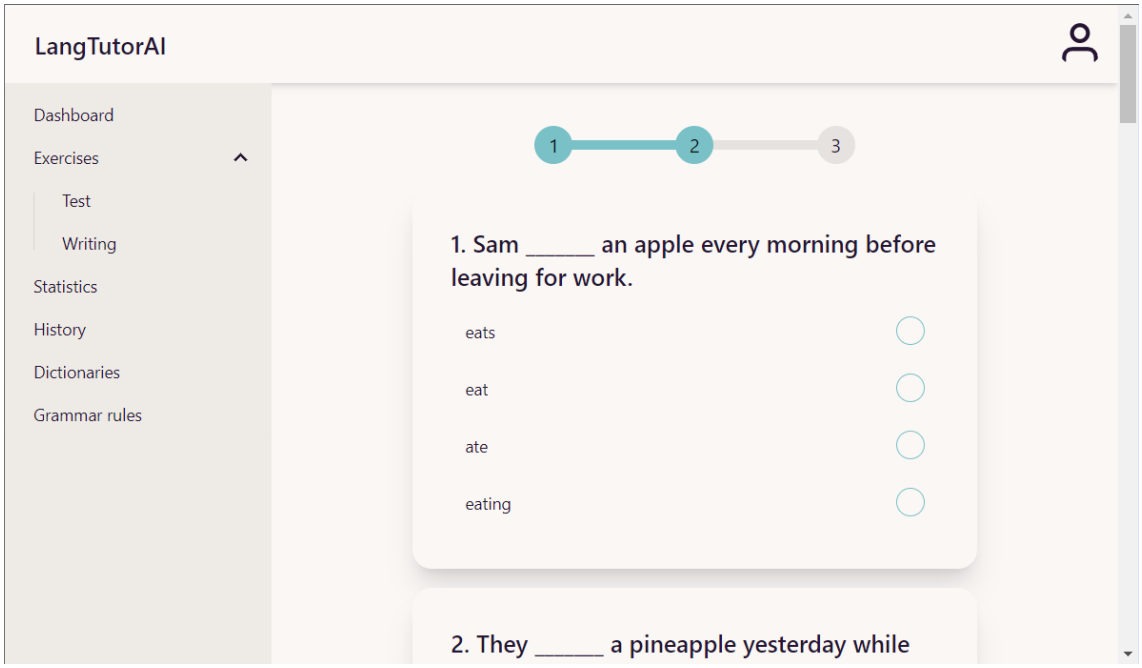

Рис. 2.15. Згенерований тесту

Рис. 2.16. Результати виконання тесту

Коли користувач завершує виконання тесту та натискає кнопку «Review», його відповіді відправляються на перевірку. Як тільки вона завершена, користувачу відображається загальний результат, відгук по кожному тесту та загальний коментар (рис. 2.16.).

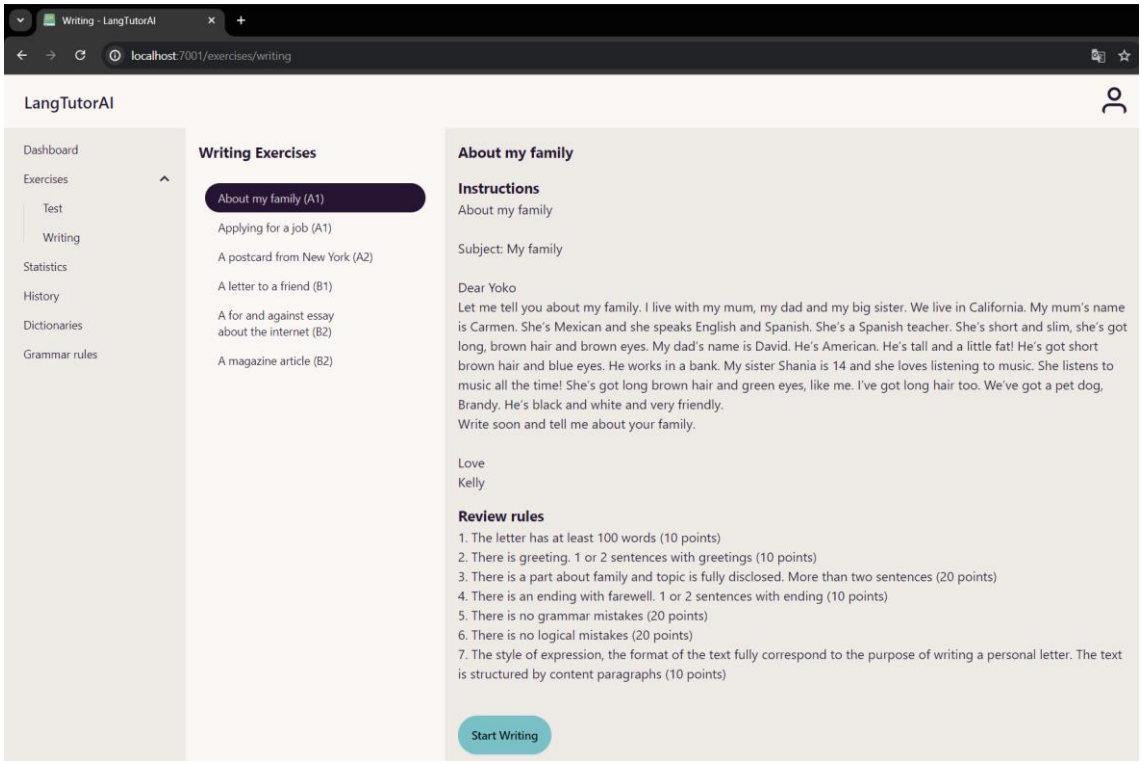

Рис. 2.17. Сторінка письмових завдань

На сторінці «Writing» відображається список можливих завдань з рівнем англійської мови (рис. 2.17.). Після обирання одного з них, справа можна побачити деталі.

Коли натиснуто кнопку «Start Writing» починається виконання обраної письмової вправи і знизу з’являється поле для введення тексту (рис. 2.18.).

Після написання тексту та натискання кнопки «Review Writing», користувачу відображається результат з загальною кількістю набраних балів на відгуком по виконаній роботі.

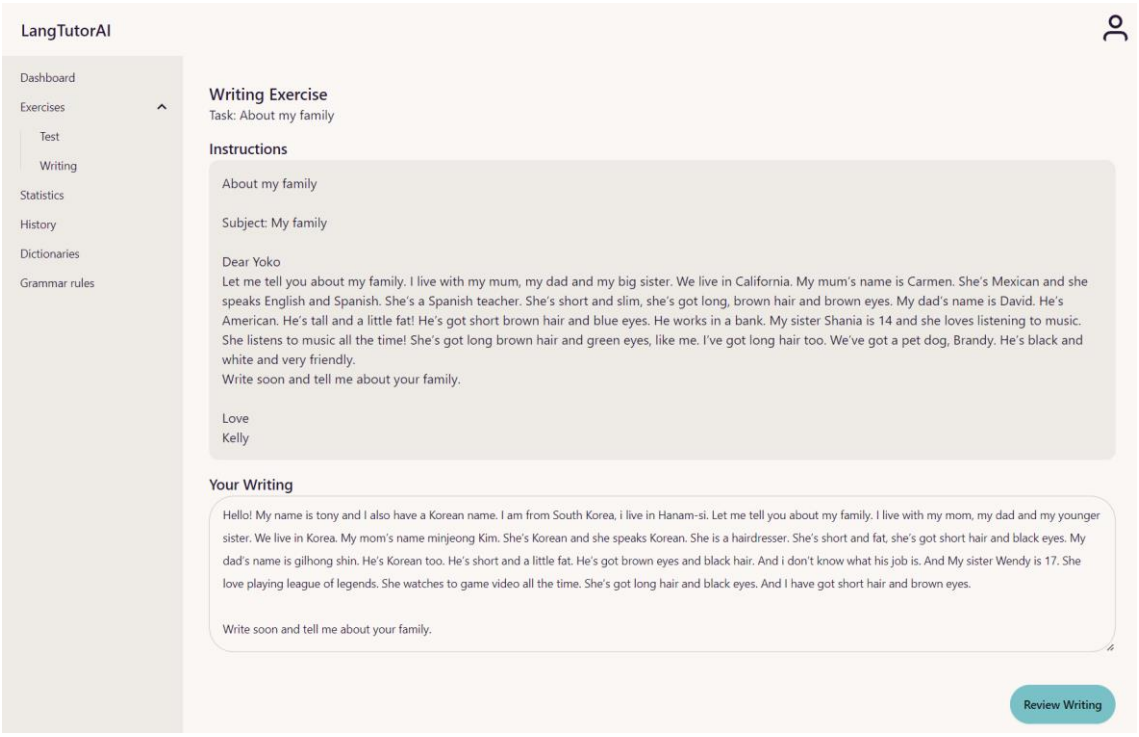

Рис. 2.18. Вигляд виконання завдання

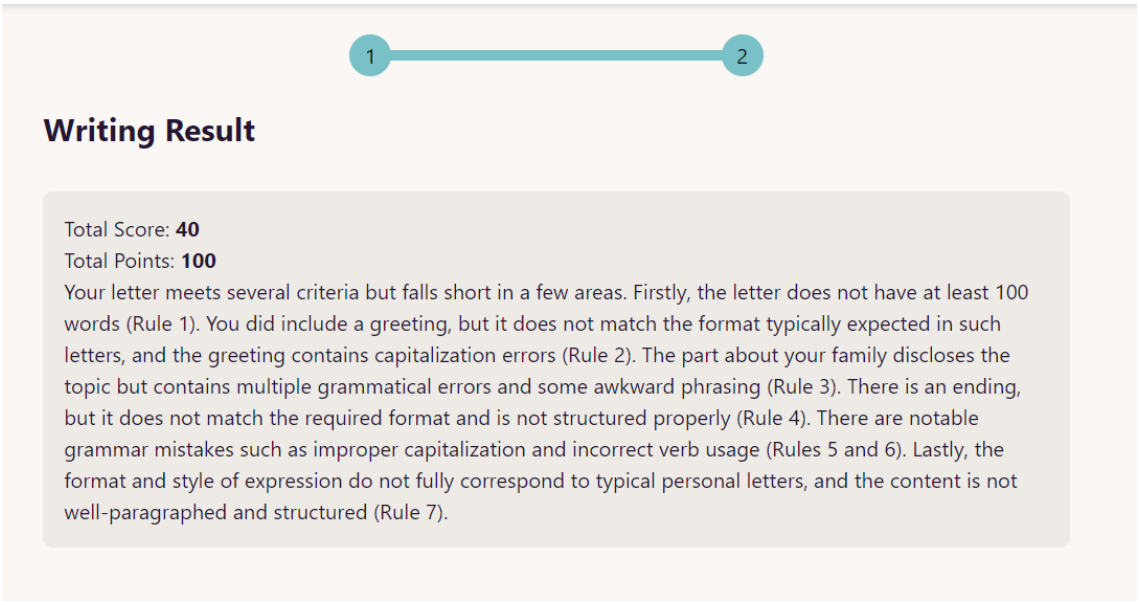

Рис. 2.19. Результати виконання письмового завдання

На сторінці «Statistics» можна побачити статистику виконання тестів за різними темами (рис. 2.20.) і побачити, що він знає добре, а над чим ще треба працювати.

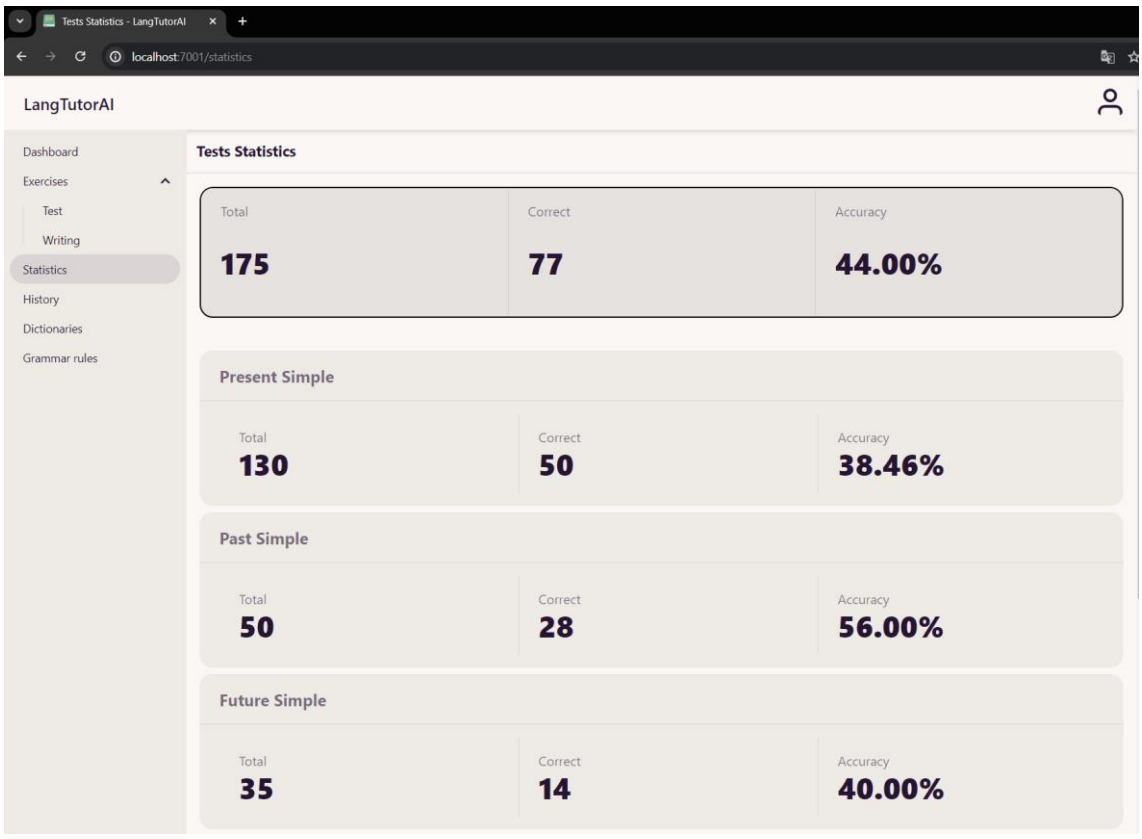

Рис. 2.20. Сторінка статистики

На сторінці «History» користувач може переглянути історію своїх виконаних завдань (рис. 2.21), відфільтрувати їх за типом та перейти на детальний перегляд кожного з них.

Таблиця історії відображає тему виконаного завдання, дату виконання та результат.

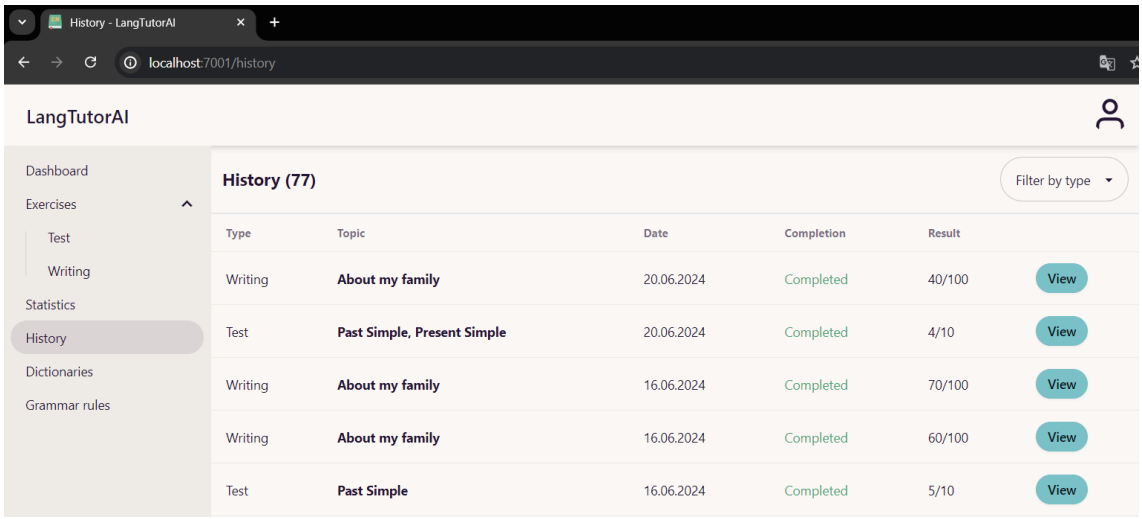

Рис. 2.21. Історія виконаних завдань

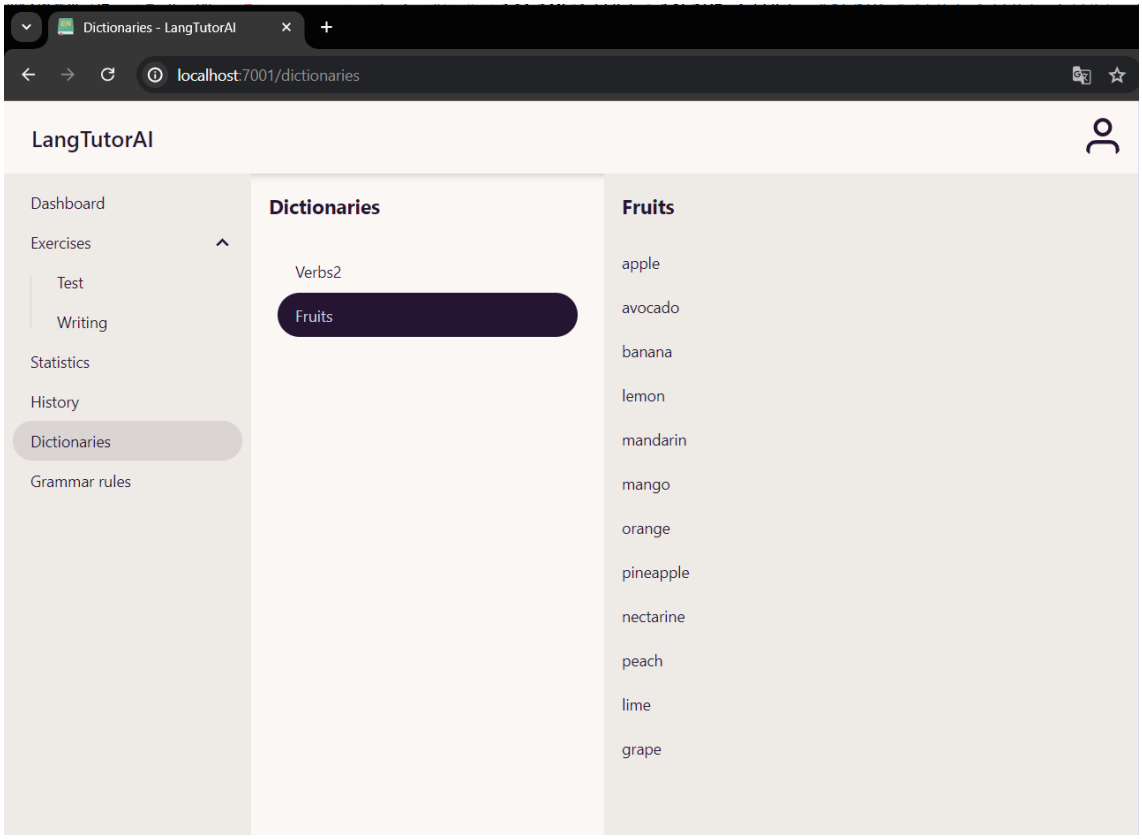

Рис. 2.22. Сторінка словників та слів

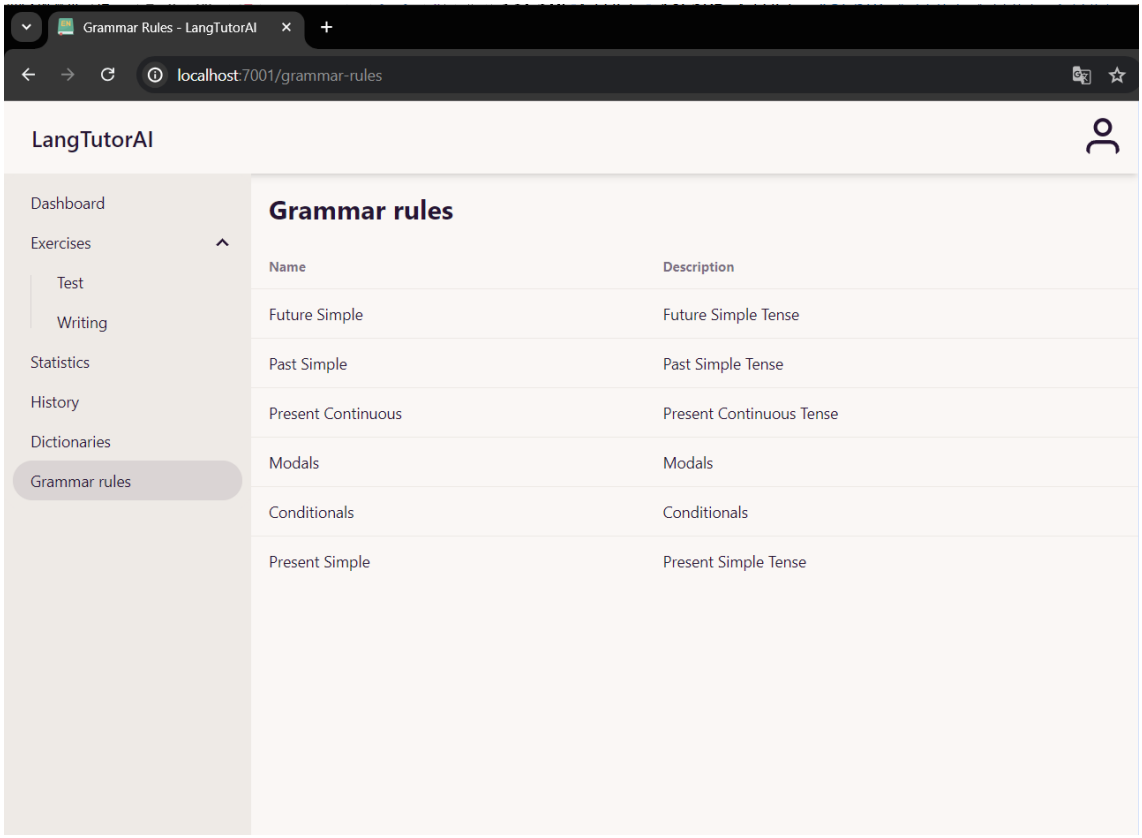

Рис. 2.23. Сторінка граматичних правил

Сторінка «Dictionaries» містить словники. При натисканні на один з них, можна переглянути слова (рис 2.22.).

На сторінці «Grammar Rules» знаходиться список граматичних правил, за якими можна виконувати тести та їх опис (рис. 2.23.).

Після натискання іконки користувача в хедері, з'являється випадаючий список (рис. 2.24.) з можливістю перейти в профіль, змінити тему (рис. 2.25.) та вийти з акаунту.

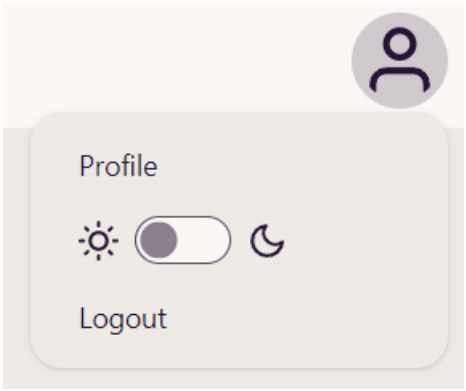

Рис. 2.24. Випадаючий список

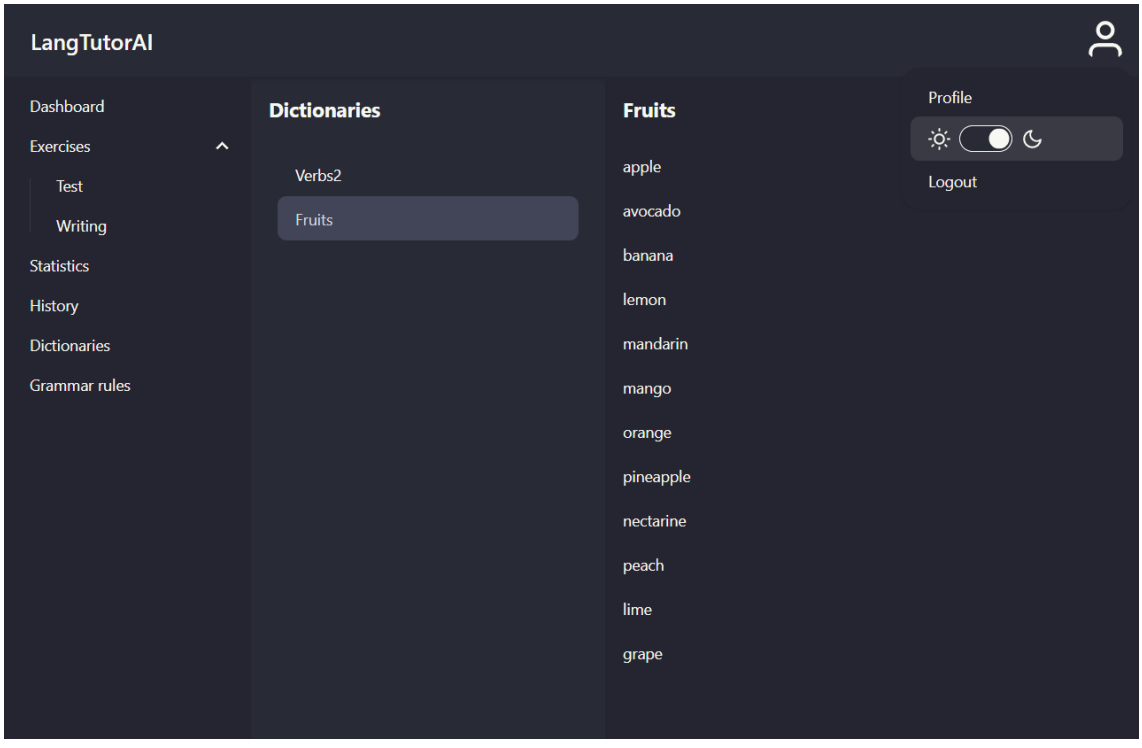

Рис. 2.25. Темна тема додатку

Сторінка профілю користувача (рис. 2.26.) дозволяє переглянути його дані (ім’я, пошту, рівень англійської мови) та змінити їх у разі необхідності (рис. 2.27.).

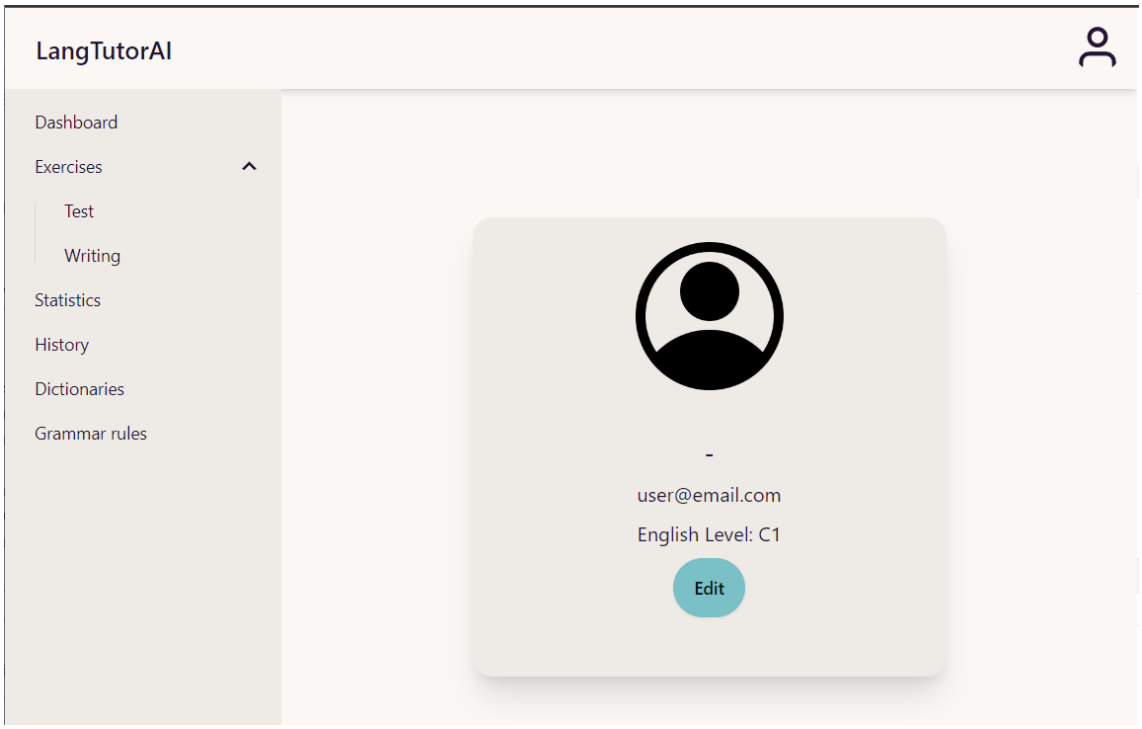

Рис. 2.26. Профіль користувача

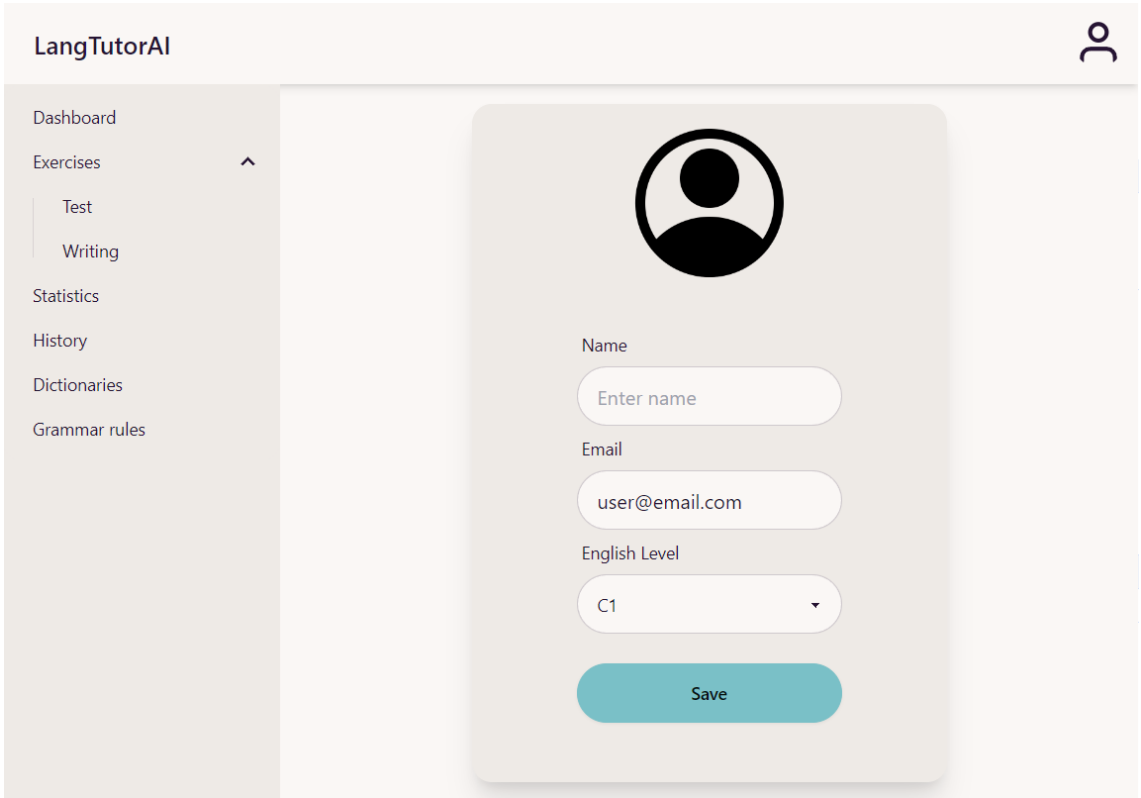

Рис. 2.27. Зміна даних

Також в додатку існує адміністратор, якому відображається більше сторінок та дається більше можливостей.

На дашборді адміністратору відображається додаткова картка з витратами на генерацію та перевірку завдань (рис. 2.28.). А інформація в інших картках та графіках відображається по всім користувачам додатку.

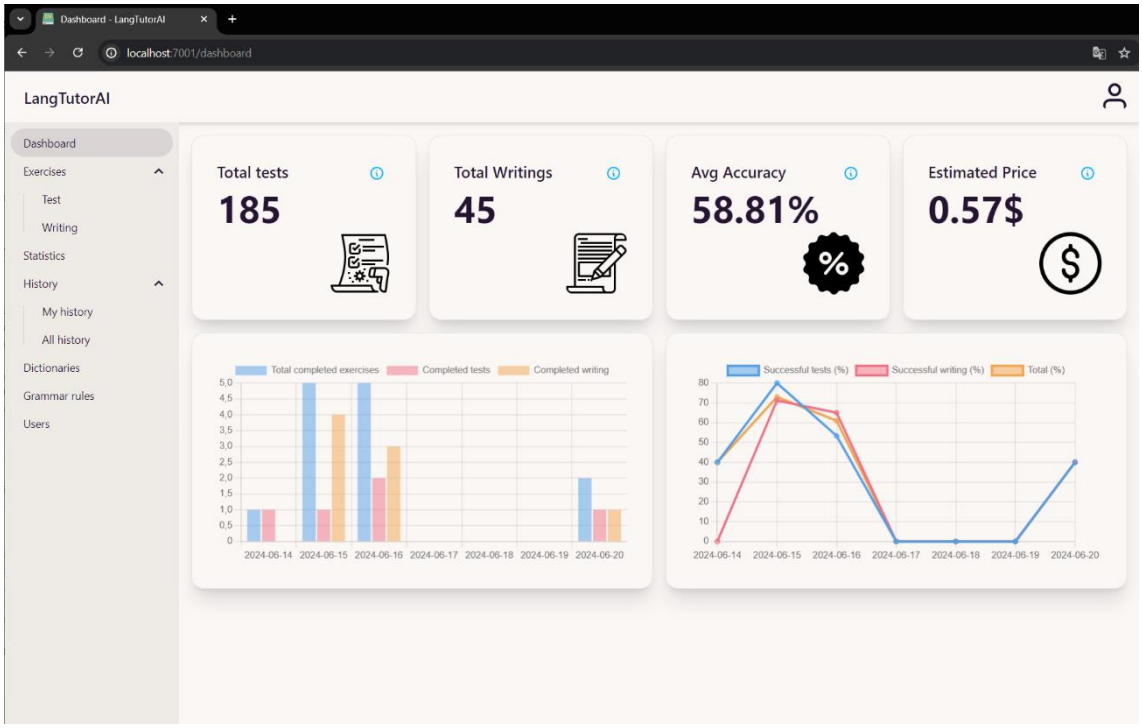

Рис. 2.28. Дашборд адміністратора

Адміністратору на вкладці «History» доступний перегляд окремо його історії та історії всіх користувачів.

Додатково з’являється сторінка «Users», яка містить інформацію про всіх користувачів у системі та дозволяє переглянути їх дані (рис. 2.29.).

Адміністратору на сторінці словників дозволяється керувати словниками та словами в них (рис. 2.30.). Він має можливість їх створювати, змінювати та видаляти.

На сторінці граматичних правил адміністратору дозволяється керувати ними (рис. 2.31.). Йому надається можливість створювати, змінювати та видаляти їх.

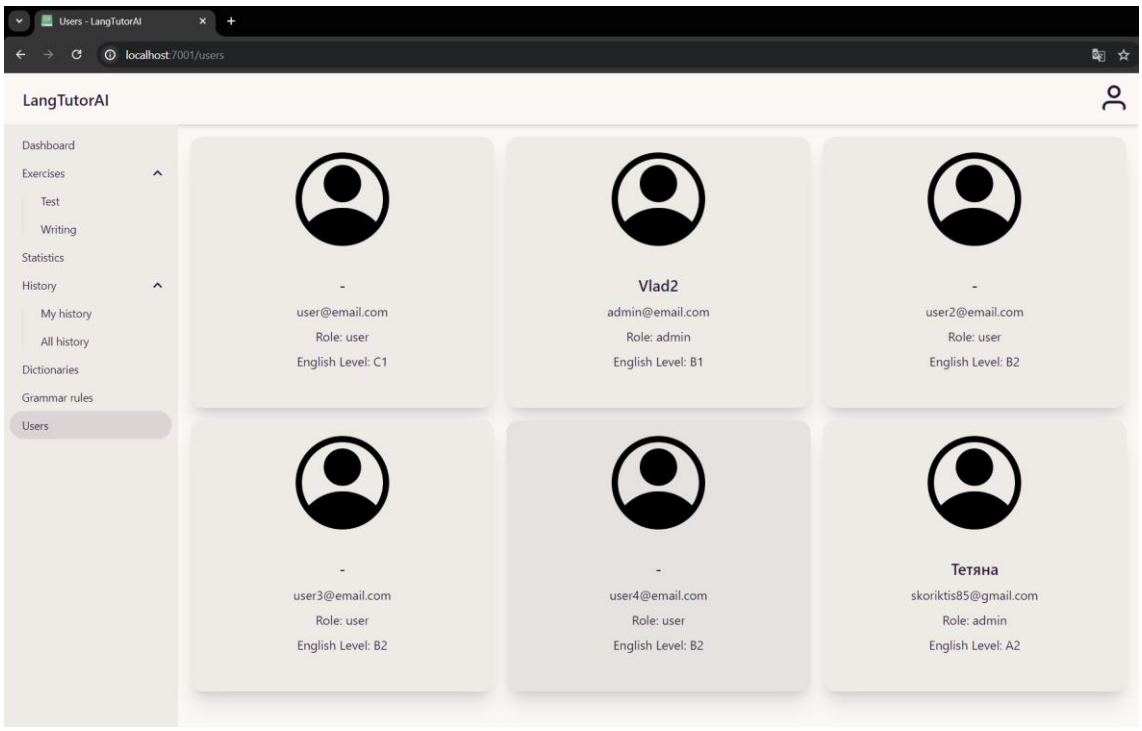

Рис. 2.29. Сторінка користувачів

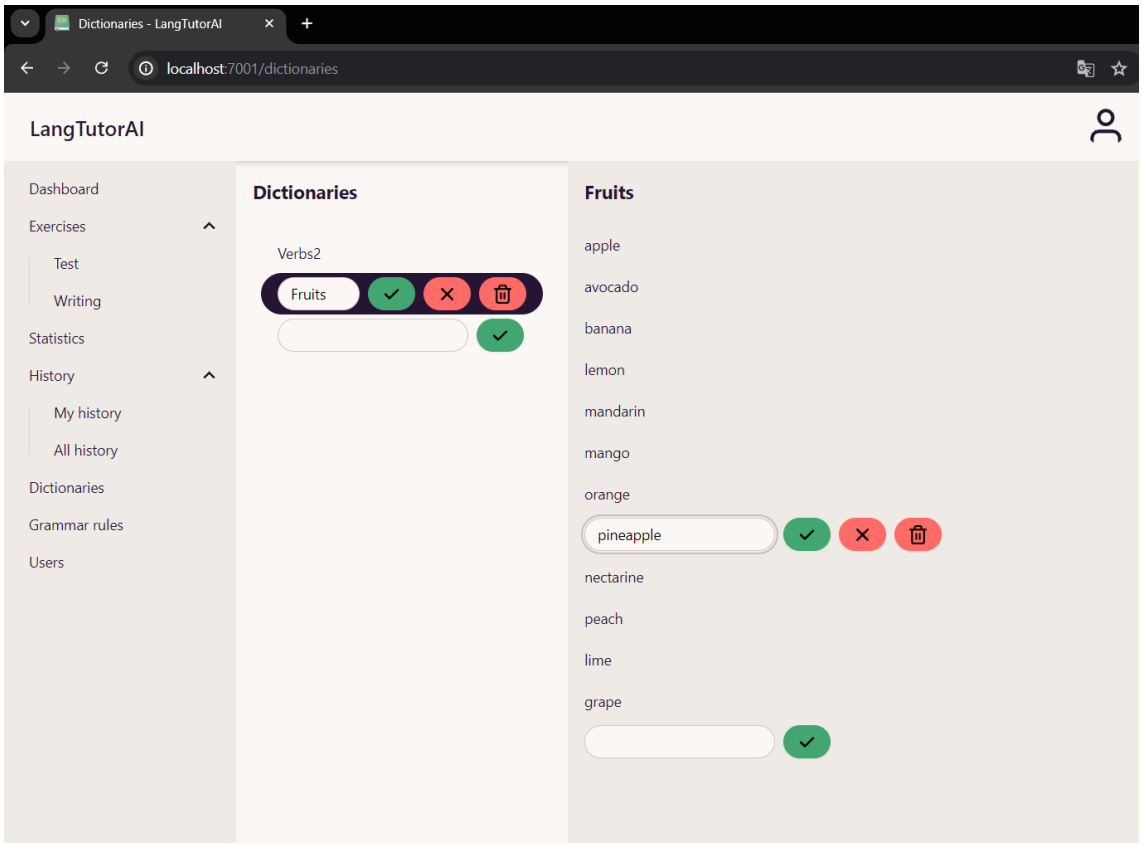

Рис. 2.30. Сторінка словників адміністратора

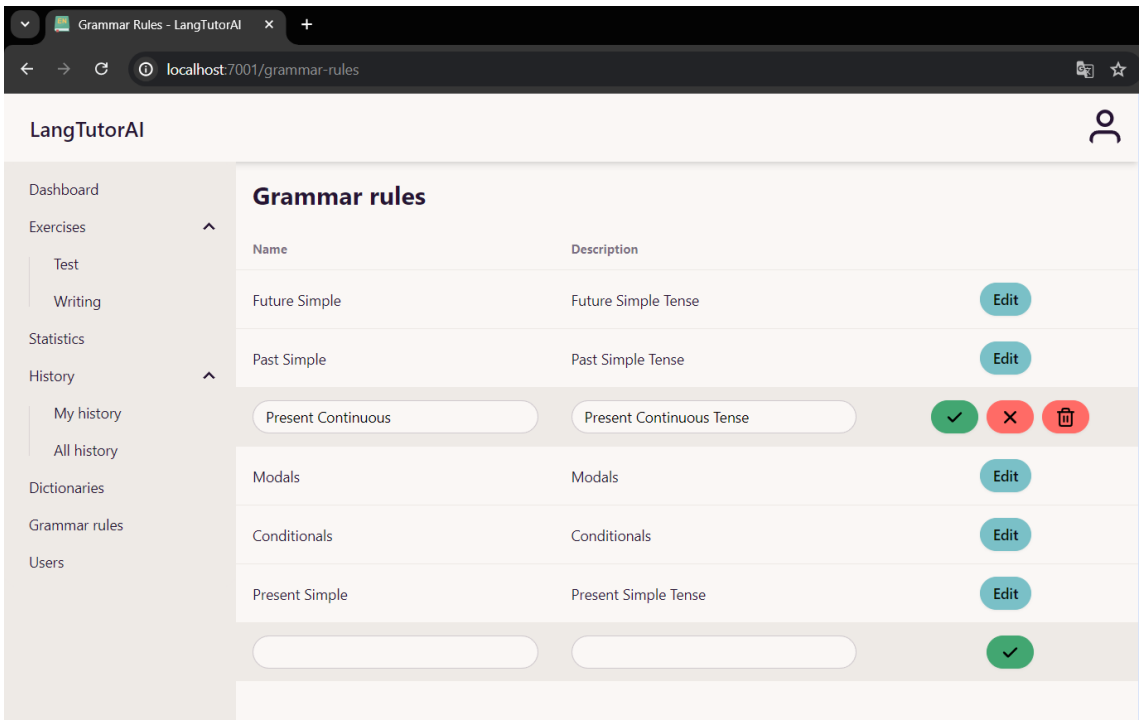

Рис. 2.31. Сторінка граматичних правил адміністратора

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Вихідні дані:

1. q – передбачуване число операторів: 4000;

2. C – коефіцієнт складності програми: 1,25;

3. p – коефіцієнт корекції програми в ході її розробки: 0,05;

4. $C_{пр}$ – середня годинна заробітна плата програміста: 185 грн/год;

За даними порталу «DOU» станом на грудень 2023 року заробітна плата Full-Stack JavaScript розробника рівня Junior складає від 550$ до 1046$ із медіанним значенням у 800$ на місяць [24]. У середньому, розробник має приблизно 22 робочих дня по 8 годин на місяць. З цього можна розрахувати погодинну ставку, як 800$ / (22 день * 8 годин) = 4,55 $/година. Курс долара США на момент середини червня 2024 року становить 40,64 грн згідно офіційного курсу валют НБУ [25]. Отже, годинна заробітна плата становить 185 грн/годину.

5. B – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,2;

6. k – коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1.1;

7. $C_{мч}$ – вартість машино-години ЕОМ – 5,97 грн/год;

Розрахунок вартості машино-години електронно-обчислювальної машини (ЕОМ) включає врахування таких ключових компонентів, як амортизація обладнання, електроенергія та інтернет:

Амортизація враховує зниження вартості обладнання протягом його експлуатаційного терміну і дорівнює відношенню його початкової вартості до часу використання. При значеннях початкової вартості обладнання (ноутбука) у 30000 грн та часу експлуатації у 4 роки маємо:

30000 грн / (4 р * 12 м * 176 год) = 3,55 грн/год

Розрахунок витрат на електроенергію включає споживання енергії обладнання та вартість електроенергії. При значеннях потужності блоку живлення ноутбука 135 Вт та вартості електроенергії 4,32 грн/кВт·год маємо:

0.135 кВт * 4,32 грн/кВт·год = 0,58 грн/год

Вартість інтернету на місць складає 323 грн, тому маємо:

323 грн / 176 год = 1,84 грн/год

Отже, вартість машино-години ЕОМ складає:

3,55 + 0,58 + 1,84 = 5,97 грн/год

8. B_k – число виконавців – 1 людина;

9. F_p – місячний фонд робочого часу (при 40 годинному робочому тижні) – 176 годин.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д \text{ , людино-годин,} \quad (3.1)$$

де t_o – витрати праці на підготовку й опис поставленої задачі (приймається 50);

t_u – витрати праці на дослідження алгоритму рішення задачі;

t_a – витрати праці на розробку блок-схеми алгоритму;

t_n – витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ – витрати праці на налагодження програми на ЕОМ;

$t_д$ – витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C \cdot (1 + p), \quad (3.2)$$

де q – передбачуване число операторів;

C – коефіцієнт складності програми;

p – коефіцієнт кореляції програми в ході її розробки.

$$Q = 4000 \cdot 1{,}25 \cdot (1 + 0{,}05) = 5250$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75\ldots85) \cdot k},\ \text{людино-годин}, \qquad (3.3)$$

де *B* – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

k – коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності;

$$t_u = \frac{5250 \cdot 1{,}2}{85 \cdot 1{,}1} = 67{,}38,\ \text{людино-годин}.$$

Витрати праці на розробку алгоритму рішення задачі:

$$t_a = \frac{Q}{(20...25) \cdot k} \qquad (3.4)$$

$$t_a = \frac{5250}{25 \cdot 1{,}1} = 190{,}9,\ \text{людино-годин}.$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20...25) \cdot k} \qquad (3.5)$$

$$t_n = \frac{5250}{25 \cdot 1{,}1} = 190{,}9,\ \text{людино-годин}.$$

Витрати праці на налагодження програми на ЕОМ:

– за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4...5) \cdot k} \qquad (3.6)$$

$$t_{отл} = \frac{5250}{5 \cdot 1{,}1} = 954{,}5,\ \text{людино-годин},$$

– за умови комплексного налагодження завдання:

$$t^к_{отл} = 1{,}5 \cdot t_{отл} \qquad (3.7)$$

$$t^к_{отл} = 1{,}5 \cdot 954{,}5 = 1431{,}75,\ \text{людино-годин}$$

Витрати праці на підготовку документації:

$$t_д = t_{др} + t_{до} \quad (3.8)$$

де $t_{др}$ – трудомісткість підготовки матеріалів і рукопису

$$t_{др} = \frac{Q}{(15...20) \cdot k} \quad (3.9)$$

$$t_{др} = \frac{5250}{20 \cdot 1,1} = 238,64, \text{ людино-годин.}$$

$t_{до}$ – трудомісткість редагування, печатки й оформлення документації

$$t_{до} = 0,75 \cdot t_{др} \quad (3.10)$$

$$t_{до} = 0,75 \cdot 238,64 = 178,98, \text{ людино-годин.}$$

Отже, витрати праці на підготовку документації за формулою (3.8):

$$t_д = 238,64 + 178,98 = 417,62, \text{ людино-годин.}$$

Розрахуємо трудомісткість розробки програмного забезпечення за формулою (3.1):

$t = 50 + 67,38 + 190,9 + 190,9 + 954,5 + 417,62 = 1871,3$, людино-годин.

Як результат отримано, що в загальній складності необхідно 2148,9 людино-годин для розробки даного програмного забезпечення.

3.2. Рахунок витрат на створення програми

Витрати на створення ПЗ $К_{по}$ включають витрати на заробітну плату виконавця програми $З_{зп}$ і витрат машинного часу, необхідного на налагодження програми на ЕОМ:

$$К_{по} = З_{зп} + З_{мв}, \text{ грн.} \quad (3.11)$$

Заробітна плата виконавців визначається за формулою:

$$З_{зп} = t \cdot С_{пр}, \text{ грн,} \quad (3.12)$$

де t – загальна трудомісткість, людино-годин;

$С_{ПР}$ – середня годинна заробітна плата програміста, грн/година

$$З_{зп} = 1871,3 \cdot 185 = 346190,5 \text{ грн.}$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{мв} = t_{отл} \cdot C_{мч}, \text{ грн,} \quad (3.13)$$

де $t_{отл}$ – трудомісткість налагодження програми на ЕОМ, год;

$C_{мч}$ – вартість машино-години ЕОМ, грн/год

$$З_{мв} = 954,5 \cdot 5,97 = 5698,37 \text{ грн.}$$

Звідси витрати на створення програмного продукту:

$$К_{по} = 346190,5 + 5698,37 = 351888,87 \text{ грн.}$$

Очікуваний період створення ПЗ:

$$Т = \frac{t}{B_k \cdot F_p}, \text{ міс.} \quad (3.14)$$

де B_k – число виконавців (дорівнює 1);

$F_р$ – місячний фонд робочого часу (при 40 годинному робочому тижні $F_р$=176 годин).

Звідси витрати на створення програмного продукту:

$$Т = \frac{1871,3}{1 \cdot 176} = 10,6 \text{ міс.}$$

Висновки: розробка даного програмного забезпечення потребує 1871,3 людино-годин. Ймовірна очікувана тривалість розробки складатиме 10,6 місяців при стандартному 40-годинному робочому тижні. Очікувані витрати на створення програмного забезпечення складатимуть 346190,5 грн.

ВИСНОВКИ

В ході виконання кваліфікаційної роботи було розроблено вебдодаток для вивчення англійської мови з використанням штучного інтелекту. Застосунок розроблено на базі стеку технологій React, Node.js, MongoDB та фреймворку Express.

Призначенням додатку є допомога людям у вивченні іноземної мови. Такий застосунок може забезпечити структуровану та систематичну програму навчання, включаючи різноманітні тести та інтерактивні завдання. Він може надати користувачеві доступ до якісних навчальних матеріалів, які постійно генеруються та адаптовані до потреб сучасного вивчення мови. Використанню штучного інтелекту може бути ефективним інструментом для підтримки у вивченні англійської мови користувачам, що допоможе подолати недоліки традиційних методів навчання та забезпечить ефективний та цікавий процес вивчення.

Актуальність платформи обумовлюється постійною необхідністю людей вивчати та підтримувати англійську мову зручним та персоніфікованим способом.

В ході виконання кваліфікаційної роботи були проведені підрахунки по встановленню трудомісткості розробки програмного забезпечення (1871,3 людино-годин), вартості розробки (346190,5 грн) та часу на його створення (10,6 міс.)

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Express 4.x - API Reference. [Електронний ресурс] URL: https://expressjs.com/en/4x/api.html (дата звернення: 11.05.2024)

2. React Reference Overview. [Електронний ресурс] URL: https://react.dev/reference/react (дата звернення: 11.05.2024)

3. Node.js Documentation. [Електронний ресурс] URL: https://nodejs.org/docs/latest/api (дата звернення: 12.05.2024)

4. MongoDB Documentation. [Електронний ресурс] URL: https://www.mongodb.com/docs (дата звернення: 12.05.2024)

5. Mongoose Documentation. [Електронний ресурс] URL: https://mongoosejs.com/docs/index.html (дата звернення: 12.05.2024)

6. API Reference - OpenAI API. [Електронний ресурс] URL: https://platform.openai.com/docs/api-reference (дата звернення: 13.05.2024)

7. Mongoose Documentation. [Електронний ресурс] URL: https://mongoosejs.com/docs/index.html (дата звернення: 17.05.2024)

8. JSON Web Tokens. [Електронний ресурс] URL: https://jwt.io (дата звернення: 17.05.2024)

9. Ranjan, Alok, Abhilasha Sinha, and Ranjit Battewad. JavaScript for modern web development: building a web application using HTML, CSS, and JavaScript. BPB Publications, 2020.

10. Vanderkam, Dan. Effective TypeScript. " O'Reilly Media, Inc.", 2024.

11. Rastogi, Aseem, et al. "Safe & efficient gradual typing for TypeScript." Proceedings of the 42Nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, 2015.

12. Arcos-Medina, Gloria, Jorge Menéndez, and Javier Vallejo. "Comparative Study of Performance and Productivity of MVC and MVVM design patterns." KnE Engineering (2018): 241-252.

13. Martin Robert. Clean Code: A Handbook of Agile Software Craftsmanship. — Pearson. 2008. — 464 p.

14. What is a RESTful API. [Електронний ресурс] URL: https://aws.amazon.com/what-is/restful-api (дата звернення: 21.05.2024)

15. Blischak, John D., Emily R. Davenport, and Greg Wilson. "A quick introduction to version control with Git and GitHub." PLoS computational biology 12.1 (2016): e1004668.

16. Dockerizing React Application Built with Vite: A Simple Guide. [Електронний ресурс] URL: https://thedkpatel.medium.com/dockerizing-react-application-built-with-vite-a-simple-guide-4c41eb09defa (дата звернення: 21.05.2024)

17. Express Error Handling. [Електронний ресурс] URL: https://expressjs.com/en/guide/error-handling.html (дата звернення: 24.05.2024)

18. Li, Yishan, and Sathiamoorthy Manoharan. "A performance comparison of SQL and NoSQL databases." 2013 IEEE Pacific Rim conference on communications, computers and signal processing (PACRIM). IEEE, 2013.

19. Bradshaw, Shannon, Eoin Brazil, and Kristina Chodorow. MongoDB: the definitive guide: powerful and scalable data storage. O'Reilly Media, 2019.

20. Pricing | OpenAI. [Електронний ресурс] URL: https://openai.com/api/pricing (дата звернення: 24.05.2024)

21. Zod Documentation. [Електронний ресурс] URL: https://zod.dev (дата звернення: 24.05.2024)

22. Teixeira, Pedro. Professional Node. js: Building Javascript based scalable software. John Wiley & Sons, 2012.

23. Hahn, Evan. Express in Action: Writing, building, and testing Node. js applications. Simon and Schuster, 2016.

24. Середня зарплата Junior TypeScript розробника. [Електронний ресурс] URL: https://jobs.dou.ua/salaries/?period=2023-12&position=Junior%20SE&technology=TypeScript&experience=0-1 (дата звернення: 14.06.2024)

25. Офіційний курс гривні щодо іноземних валют. [Електронний ресурс] URL: https://bank.gov.ua/ua/markets/exchangerates (дата звернення: 14.06.2024)

ДОДАТОК А

КОД ПРОГРАМИ

Server:
index.ts
```
import dotenv from "dotenv";
dotenv.config();
import express from "express";
import cors from "cors";
import db from "./db";
import { dictionariesRouter } from "./routes/dictionaries.routes";
import { usersRouter } from "./routes/users.routes";
import { wordsRouter } from "./routes/words.routes";
import { grammarRulesRouter } from "./routes/grammar-rules.routes";
import { exercisesRouter } from "./routes/exercises.routes";
import {
  errorHandler,
  invalidPathHandler,
} from "./middlewares/error-handler.middleware";
import { writingTasksRouter } from "./routes/writing-tasks.routes";
import { authGuard } from "./middlewares/auth.guard";
import { authRouter } from "./routes/auth.routes";

const port = process.env.PORT || 7000;
const app = express();

app.use(express.json());
app.use(cors());

app.use("/dictionaries", authGuard, dictionariesRouter);
app.use("/users", usersRouter);
app.use("/words", authGuard, wordsRouter);
app.use("/grammar-rules", authGuard, grammarRulesRouter);
app.use("/exercises", authGuard, exercisesRouter);
app.use("/writing-tasks", authGuard, writingTasksRouter);
app.use("/auth", authRouter);

app.use(errorHandler);
app.use(invalidPathHandler);

app.listen(port, async () => {
  await db;
  console.log(`App listening on port ${port}`);
});
```

.env.example
```
PORT=7000

OPENAI_API_KEY=openai-api-key

MONGODB_URI=mongodb://mongo_db:27017/eng-app

JWT_SECRET=jwt-secret
```

db.ts
```
import mongoose from "mongoose";

export default mongoose.connect(process.env.MONGODB_URI);
```

constants.ts
```
import { ChatCompletionMessageParam } from "openai/resources";
```

```
import { ENGLISH_LEVEL } from "./models/User";
import { WritingTask } from "./models/WritingTask";

export const generatedTestFormat = {
  questions: [
    {
      question: "*1. question*",
      options: { A: "option a", B: "option b", C: "option c", D: "option d" },
      correctAnswer: "*correctAnswer*",
    },
  ],
};

export const reviewAnswersFormat = {
  questions: {
    1: {
      isCorrect: true,
      review: "Review on this answer",
    },
  },
  correctAnswers: 3,
  totalReview: "Total review on test",
};

export interface IGenerateTestPrompt {
  count: number;
  topics: string[];
  words: string[];
  level: ENGLISH_LEVEL;
}

export interface IReviewAnswersPrompt {
  answers: string;
}

export const reviewAnswersPrompt = ({ answers }: IReviewAnswersPrompt) => {
  return `Review the answers: ${JSON.stringify(
    answers
  )}. Give review on these answers. Return result in this JSON format: ${JSON.stringify(
    reviewAnswersFormat
  )}.`;
};

export const generateTestPrompt = ({
  count,
  topics,
  words,
  level,
}: IGenerateTestPrompt) => {
  return `Create a ${count}-question test. Each question should have only one correct answer.
    Questions should be on these English grammar topics: ${topics.join(", ")}.
    Use ${level}-level vocabulary to generate questions.
    Also try to use these words: ${words}.
    Return result without any text, only in this JSON format: ${JSON.stringify(
      generatedTestFormat
    )}.`;
};

export const generateTestMessages = (
  options: IGenerateTestPrompt
): ChatCompletionMessageParam[] => {
  return [
```

```
    {
      role: "system",
      content:
        "You are a helpful assistant in learning English language. Act like an all-knowing English teacher. All of your answers should return only in JSON format. Do not add unnecessary symbols",
    },
    { role: "user", content: generateTestPrompt(options) },
  ];
};

export const reviewWritingFormat = {
  totalReview: "total review",
  totalPoints: 0,
  points: 0,
};

export interface IGenerateWritingPrompt {
  writingTask: WritingTask;
  answer: string;
  level: ENGLISH_LEVEL;
}

export const generateWritingPrompt = ({
  writingTask,
  answer,
  level,
}: IGenerateWritingPrompt) => {
  return `I should write good answer to this letter:
  ---start letter---
  ${writingTask.instructions}
  ---end letter---
  And your task is to review my letter in response. Take into account that I have ${level} level of English. To evaluate my letter give points for some rules. There is possible points for every rule. Rules:
  ---start rules---
  ${writingTask.reviewRules}
  ---end rules---
  In response return 'totalPoints' as sum of possible points for every rule and 'points' as sum of points given for rules out of all possible points. Give a review of the letter as a whole and tell what rules were broken in field 'totalReview'.
  So there is my letter:
  ---start letter---
  ${answer}
  ---end letter---
  And in the end return result only in this JSON format: ${JSON.stringify(
    reviewWritingFormat
  )}.`;
};

export const generateWritingMessages = (
  options: IGenerateWritingPrompt
): ChatCompletionMessageParam[] => {
  return [
    {
      role: "system",
      content:
        "You are a helpful assistant in learning English language. Act like an all-knowing English teacher. All of your answers should return only in JSON format. Do not add unnecessary symbols",
    },
    { role: "user", content: generateWritingPrompt(options) },
  ];
};
```

routes/auth.routes.ts

```
import { Router } from "express";
import { authGuard } from "../middlewares/auth.guard";
import { AuthController } from "../controllers/auth.controller";

const authRouter = Router();
const authController = new AuthController();

authRouter.post("/register", (req, res, next) =>
  authController.register(req, res).catch(next)
);
authRouter.post("/login", (req, res, next) =>
  authController.login(req, res).catch(next)
);
authRouter.get("/profile", authGuard, (req, res, next) =>
  authController.profile(req, res).catch(next)
);

export { authRouter };
```

routes/exercises.routes.ts

```
import { Router } from "express";
import { ExercisesController } from "../controllers/exercises.controller";

const exercisesRouter = Router();
const exercisesController = new ExercisesController();

exercisesRouter.get("/:id", (req, res, next) =>
  exercisesController.findById(req, res).catch(next)
);

exercisesRouter.post("/tests", (req, res, next) =>
  exercisesController.generateTestExercise(req, res).catch(next)
);
exercisesRouter.post("/tests/:id/review", (req, res, next) =>
  exercisesController.reviewTestAnswers(req, res).catch(next)
);
exercisesRouter.get("/tests/statistics", (req, res, next) =>
  exercisesController.getTestsStatistics(req, res).catch(next)
);

exercisesRouter.post("/writing/review", (req, res, next) =>
  exercisesController.reviewWritingExercise(req, res).catch(next)
);

exercisesRouter.get("/dashboard/total", (req, res, next) =>
  exercisesController.getDashboardData(req, res).catch(next)
);
exercisesRouter.get("/dashboard/charts/total", (req, res, next) =>
  exercisesController.getTotalExercisesChart(req, res).catch(next)
);
exercisesRouter.get("/dashboard/charts/percentage", (req, res, next) =>
  exercisesController.getCompletionPercentageChart(req, res).catch(next)
);

exercisesRouter.get("/", (req, res, next) =>
  exercisesController.getHistory(req, res).catch(next)
);

export { exercisesRouter };
```

routes/users.routes.ts

```
import { Router } from "express";
```

```
import { UsersController } from "../controllers/users.controller";
import { authGuard } from "../middlewares/auth.guard";
import { roleGuard } from "../middlewares/role.guard";
import { USER_ROLE } from "../models/User";

const usersRouter = Router();
const usersController = new UsersController();

usersRouter.get(
  "/",
  authGuard,
  roleGuard([USER_ROLE.ADMIN]),
  (req, res, next) => usersController.find(req, res).catch(next)
);
usersRouter.get(
  "/:id",
  authGuard,
  roleGuard([USER_ROLE.ADMIN]),
  (req, res, next) => usersController.findById(req, res).catch(next)
);
usersRouter.put("/:id", authGuard, (req, res, next) =>
  usersController.update(req, res).catch(next)
);

export { usersRouter };
```

controllers/auth.controller.ts

```
import { Request, Response } from "express";
import { zodParse } from "../utils/zod-parse";
import { loginUserSchema, registerUserSchema } from "../schemas/auth.schema";
import { AuthService } from "../services/auth.service";

export class AuthController {
  private readonly authService: AuthService;

  constructor() {
    this.authService = new AuthService();
  }

  async register(req: Request, res: Response) {
    const { body } = await zodParse(registerUserSchema, req);

    const result = await this.authService.register(body);
    return res.json(result);
  }

  async login(req: Request, res: Response) {
    const { body } = await zodParse(loginUserSchema, req);

    const result = await this.authService.login(body);
    return res.json(result);
  }

  async profile(_: Request, res: Response) {
    const authUser = res.locals.user;

    const result = await this.authService.profile(authUser);
    return res.json(result);
  }
}
```

Решта коду додається окремо на CD-диску.

ДОДАТОК Б

ВІДГУК

КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

на кваліфікаційну роботу бакалавра на тему:

«Розробка вебдодатку для вивчення англійської мови з використанням штучного інтелекту на базі стеку технологій React, Node.js, MongoDB та фреймворку Express»

Студента групи 121-20-1 Скорика Владислава Максимовича

Керівник економічного розділу

доц. каф. ПЕП та ПУ, к.е.н **Л.В. Касьяненко**

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	Опис
Пояснювальні документи	
Кваліфікаційна робота Скорик.docx	Пояснювальна записка до кваліфікаційної роботи. Документ Word
Кваліфікаційна робота Скорик.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
Скорик.rar	Архів. Містить коди програми і скомпільовану програму
Презентація	
Скорик.ppt	Презентація кваліфікаційної роботи