МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Львівський національний університет імені Івана Франка

Факультет електроніки та комп'ютерних технологій

Кафедра радіоелектронних і комп'ютерних систем

Допустити до захисту
Завідувач кафедри
_____ проф. Оленич Ігор Богданович
«___» _____ 2023 р.

Кваліфікаційна робота

Бакалавр
(освітній ступінь)

РОЗРОБЛЕННЯ ЗАСТОСУНКУ «НАВЧАЛЬНИЙ СЛОВНИК» З ВИКОРИСТАННЯМ ВЕБ-ФРЕЙМВОРКІВ DJANGO ТА REACT

Виконала:
студентка IV курсу групи ФЕП-41
спеціальності 121 – Інженерія програмного забезпечення
_____ Павлюк Єлизавета Олегівна
Науковий керівник:
_____ асист. Павлик Михайло Романович
«___» _____ 2023 р.

Рецензент:
_____ _____
(підпис) (ПІБ)

Львів 2023

АНОТАЦІЯ

Навчальний словник «EduDictionary» – це комплексний онлайн-інструмент, розроблений для надання користувачам інтерактивного та освітнього досвіду у вивченні та розумінні різних слів англійської мови. Додаток використовує комбінацію веб-фреймворків Django та React для створення безшовного та інтуїтивно зрозумілого користувацького інтерфейсу. Django, потужний та універсальний фреймворк на основі Python, відповідає за логіку на стороні сервера, тоді як React, бібліотека JavaScript, покращує користувацький інтерфейс завдяки своїм динамічним та адаптивним компонентам.

Для ефективного зберігання та управління даними додаток використовує базу даних Firestore. Це гнучка та масштабована хмарна база даних NoSQL від Firebase, яка пропонує синхронізацію в режимі реального часу та безшовну інтеграцію з іншими сервісами цієї платформи. Це дозволяє безперешкодно отримувати дані та оновлювати їх, забезпечуючи користувачам доступ до найактуальнішої інформації.

Аутентифікація та авторизація користувачів здійснюється за допомогою Firebase, безпечної та надійної служби. Лише авторизовані користувачі можуть отримати доступ до певних функцій та можливостей навчального словника (наприклад, особистий словник збережених слів), що підвищує безпеку та конфіденційність даних.

Крім того, додаток інтегрується з API для розширення функціональності словника. Через API здійснюється пошук слів, синонімів, антонімів, дозволяє отримувати додаткову інформацію із зовнішніх джерел, розширюючи обсяг і глибину змісту словника. Ця інтеграція надає користувачам доступ до величезного масиву знань і забезпечує актуальність та сучасність.

Загалом, веб-додаток "Навчальний словник", створений на Django та React, з базою даних Firestore, авторизацією Firebase та інтеграцією API, пропонує комплексну та зручну платформу для вивчення та розширення своїх знань з англійської мови.

ABSTRACT

EduDictionary is a comprehensive online tool designed to provide users with an interactive and educational experience in learning and understanding various words of the English language. The application uses a combination of Django and React web frameworks to create a seamless and intuitive user interface. Django, a powerful and versatile Python-based framework, is responsible for server-side logic, while React, a JavaScript library, enhances the user experience with its dynamic and responsive components.

For efficient data storage and management, the application uses the Firestore database. This is a flexible and scalable cloud-based NoSQL database from Firebase that offers real-time synchronization and seamless integration with other services on the platform. This allows for seamless data retrieval and updating, providing users with access to the most up-to-date information.

User authentication and authorization is performed using Firebase, a secure and reliable service. Only authorized users can access certain features and capabilities of the learning dictionary (e.g., personal dictionary of saved words), which increases data security and confidentiality.

In addition, the application integrates with the API to extend the functionality of the dictionary. The API allows you to search for words, synonyms, antonyms, and receive additional information from external sources, expanding the scope and depth of the dictionary content. This integration provides users with access to a vast array of knowledge and ensures that the dictionary is up-to-date and relevant.

All in all, the Learning Dictionary web application, built on Django and React, with a Firestore database, Firebase authorization, and API integration, offers a comprehensive and convenient platform for learning and expanding your English language skills.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

API (*англ. Application Programming Interface*) – прикладний програмний інтерфейс, інтерфейс програмування застосунків, інтерфейс прикладного програмування

CSRF (*англ. Cross-site request forgery*) – міжсайтова підробка запиту

CSS (*англ. Cascading Style Sheets*) – каскадні таблиці стилів

DOM (*англ. Document Object Model*) – об’єктна модель документів

HTML (*англ. HyperText Markup Language*) – мова розмітки гіпертексту

HTTP (*англ. HyperText Transfer Protocol*) – протокол передачі гіпертексту

ID (*англ. identity document*) – ідентифікатор

IDE (*англ. Integrated Drive Electronics*) – інтегроване середовище розробки

JSON (*англ. JavaScript Object Notation*) – запис об'єктів JavaScript

JSX – розширення JavaScript.

LDOCE (*англ. Longman Dictionary of Contemporary English*) – словник сучасної англійської мови Longman

MFA (*англ. multi-factor authentication*) – багатофакторна автентифікація

MVT (*англ. Model View Template*) – шаблон представлення моделі

NLP (*англ. Natural language processing*) – обробка природної мови

NoSQL (*англ. non SQL*) – нереляційна база даних

ORM (*англ. Object-relational mapping*) – об'єктно-реляційна проєкція

SDK (*англ. Software Development Kit*) – набір із засобів розробки, утиліт і документації

SMS (*англ. Short Message Service*) – послуга обміну короткими повідомленнями

URL (*англ. Uniform Resource Locator*) – єдиний вказівник на ресурс

XSS (*англ. Cross Site Scripting*) – міжсайтовий скриптинг

БД – база даних

ООП – об’єктно-орієнтоване програмування

ВСТУП

У сучасному світі вивчення англійської мови набуває все більшого значення завдяки її статусу провідної мови. Володіння іноземною мовою відкриває двері до різноманітних освітніх та кар'єрних можливостей, дозволяючи людям ефективно спілкуватися та взаємодіяти з людьми по всьому світу. Як наслідок, зростає потреба в доступних і універсальних інструментах, які допоможуть тим, хто вивчає англійську. Ця дипломна робота присвячена розробці веб-додатку навчального словника «EduDictionary» з використанням веб-фреймворки Django та React

Актуальність цієї роботи полягає у вирішенні проблем, з якими стикаються люди, що вивчають англійську, у доступі до надійних та зручних ресурсів для вивчення мови. Існуючим словникам часто бракує інтерактивності і вони не забезпечують цілісного навчального процесу. «EduDictionary» має на меті заповнити цю прогалину, надаючи універсальну платформу, де користувачі можуть отримати доступ до великої колекції англійських слів, визначень, прикладів, синонімів та пов'язаних з ними ресурсів в інтерактивній та захоплюючій формі.

Метою цієї дипломної роботи є проєктування та розробка веб-додатку навчального словника «EduDictionary», що включає в себе передові технології та методології для покращення навчального процесу користувачам, які вивчають англійську мову. Використовуючи веб-фреймворки Django та React, базу даних Firestore, авторизацію Firebase та різноманітні API, додаток має на меті забезпечити безперебійну та інтуїтивно зрозумілу платформу, яка задовольняє різноманітні потреби учнів на різних рівнях володіння мовою.

Об'єктом дослідження в цій дипломній роботі є розробка веб-додатку «Навчальний словник», включаючи його особливості, функціональні можливості та дизайн користувацького інтерфейсу. Додаток слугуватиме комплексним ресурсом для тих, хто вивчає англійську мову, пропонуючи широкий спектр навчальних термінів, визначень, прикладів та супутніх матеріалів.

Предметом дослідження є використання веб-фреймворків Django та React, бази даних Firestore, авторизація Firebase та використання API при розробці веб-додатку. Дослідження буде зосереджене на розумінні можливостей цих технологій та їх

інтеграції для створення надійного та зручного для користувача освітнього інструменту.

Методи дослідження, використані в цій дипломній роботі, включатимуть поєднання практик розробки програмного забезпечення, принципів дизайну, орієнтованого на користувача, та аналізу даних. Процес розробки програмного забезпечення буде здійснюватися з використанням гнучких методологій для ефективного управління проєктами.

Елементи новизни цієї дипломної роботи полягають в інтеграції передових технологій, таких як веб-фреймворки Django і React, база даних Firestore, авторизація Firebase і використання API, для створення навчального словника, який перевершує обмеження традиційних словників. Сфера застосування поширюється користувачів, які вивчають англійську мову, прагнуть розшити свій словниковий запас, шукають інтерактивну платформу для вивчення англійської мови.

З точки зору подальшого розвитку, навчальний словник може бути розширений у кількох напрямках. По-перше, додаток може включати в себе елементи гейміфікації для підвищення залученості та мотивації користувачів. Інтерактивні вікторини, словникові завдання та функції відстеження прогресу можуть бути впроваджені, щоб зробити процес навчання більш приємним та корисним. Крім того, інтеграція алгоритмів обробки природної мови (NLP) може забезпечити розширені функції пошуку, контекстні рекомендації та персоналізований досвід навчання. Зокрема, зміст словника можна розширити, включивши в нього спеціалізовану термінологію з різних галузей, щоб задовольнити специфічні потреби учнів у різних сферах.

Отже, метою цієї дипломної роботи є розробка веб-додатку "Навчальний словник" з використанням веб-фреймворків Django та React, бази даних Firestore, авторизації Firebase та API. Розглядаючи важливість вивчення англійської мови в сучасному світі, актуальність роботи, а також мету, об'єкт та предмет дослідження, ця дипломна робота створює основу для розробки інноваційного освітнього інструменту. Методи дослідження, елементи новизни та запропоновані напрямки подальших розробок створюють основу для створення всебічного та динамічного навчального процесу для тих, хто вивчає англійську мову.

РОЗДІЛ 1. ОСОБЛИВОСТІ ВИВЧЕННЯ АНГЛІЙСЬКОЇ МОВИ ЗА ДОПОМОГОЮ НАВЧАЛЬНИХ СЛОВНИКІВ

1.1 Важливість вивчення англійської мови, як однієї з складових освіти

В сучасному світі освіта – це невід'ємна частина людського розвитку, що дає можливість кожній особистості і суспільству в цілому розкрити свій потенціал. Вона забезпечує нас необхідними знаннями, навичками та перспективами, щоб орієнтуватися в складному світі. Слугує каталізатором особистісного зростання, просвітництва та розширення можливостей.

У все більш взаємопов'язаному світі англійська мова стала лінгва франка спілкування, комерції та науки. Володіння іноземною мовою дає людині численні переваги як на особистому, так і на професійному рівні.

По-перше, англійська є провідною мовою спілкування, що дозволяє людям з різним мовним походженням ефективно взаємодіяти та обмінюватися ідеями. Люди можуть долати мовні бар'єри, спілкуватися з іншими по всьому світу та брати участь у глобальній розмові. Ця здатність полегшує культурний обмін, сприяє взаєморозумінню та співпраці на міжнародному рівні.

По-друге, володіння англійською мовою відкриває доступ до широкого спектру кар'єрних можливостей. На сучасному глобалізованому ринку праці багато роботодавців шукають кандидатів, які вільно володіють англійською мовою. Володіння іноземною мовою підвищує шанси на працевлаштування та відкриває двері до міжнародних вакансій, всесвітніх компаній та міжнародних організацій. Це збільшує шанси на кар'єрний ріст і дає людині конкурентну перевагу в професійній сфері.

Вивчення англійської мови розширює горизонти, надаючи доступ до величезного сховища знань та інформації. Більшість наукових досліджень, навчальної літератури та технологічних досягнень поширюються саме цією мовою. Опанувавши англійську, люди можуть отримати доступ до цих безцінних ресурсів, бути в курсі останніх подій у різних галузях і зробити свій внесок у світову скарбницю знань.

Крім того, володіння англійською мовою розширює культурне сприйняття та сприяє глибшому розумінню різних суспільств та їхніх традицій. Воно дозволяє людям взаємодіяти з літературою, фільмами та видами мистецтва з усього світу, тим самим сприяючи міжкультурному розумінню та співпереживанню.

Освіта є наріжним каменем особистого і суспільного прогресу, дозволяючи нам жити повноцінним життям і робити значущий внесок у розвиток своїх громад. У сфері освіти вивчення англійської мови стало незамінною навичкою в сучасному взаємопов'язаному світі. Володіння якою, не лише сприяє ефективному спілкуванню, але й відкриває двері до безлічі можливостей, як особистих, так і професійних. Тому, визнаючи важливість освіти та вивчаючи англійську мову, люди можуть стати на шлях самопізнання, розширення можливостей та глобального залучення.

1.2 Методи вивчення англійської мови

Вивчення англійської мови – це корисна справа, яка відкриває двері до численних можливостей. Щоб ефективно опанувати мовою, люди часто використовують різні методи навчання. Розглянемо різні підходи до вивчення англійської мови, висвітлимо їхні переваги та недоліки, розкриємо важливість застосування навчальних словників як цінного ресурсу в процесі вивчення мови.

Навчання в класі:

Переваги:

- Структурований підхід: навчання в класі передбачає структурований навчальний план з чітким розвитком тем і навичок.
- Взаємодія з однолітками: учні можуть брати участь у дискусіях, практикувати розмовну мову та отримувати зворотній зв'язок як від викладачів, так і від однокласників.
- Поради від експертів: кваліфіковані викладачі можуть надавати пояснення, правильну вимову та задовольняти індивідуальні навчальні потреби.

Недоліки:

- Обмежена індивідуальна увага: у класі вчителі можуть мати обмежений час, щоб задовольнити конкретні потреби кожного учня.

- Брак гнучкості: темп і зміст навчання в класі можуть не відповідати індивідуальним стилям навчання або інтересам.
- Недостатня розмовна практика: великі розміри класів та часові обмеження можуть обмежувати можливості для активної участі та мовленнєвої практики.

Самостійне навчання та онлайн-ресурси:

Переваги:

- Гнучкість: самостійне навчання дозволяє студентам встановлювати власний темп, зосереджуватися на сферах, що їх цікавлять, і адаптувати свій навчальний графік відповідно до своїх потреб.
- Доступні широкі ресурси: онлайн-платформи пропонують безліч матеріалів, таких як відео, подкасти, інтерактивні вправи та додатки для вивчення мови.
- Незалежність і автономія: самостійне навчання заохочує учнів брати на себе відповідальність за свій навчальний процес і розвивати самодисципліну.

Недоліки:

- Відсутність керівництва: без належного керівництва учні можуть мати труднощі з розумінням складних граматичних правил, вимови або ідіоматичних виразів.
- Обмежений зворотній зв'язок: самостійне навчання може не надавати негайного зворотного зв'язку щодо помилок, що ускладнює виправлення помилок і покращення точності.
- Можливість відволікання: відсутність структурованого середовища може призвести до зниження мотивації або відволікання від інших обов'язків.

Занурення та розмовна практика:

Переваги:

- Автентичний вплив мови: занурення в англомовне середовище забезпечує постійний вплив мови, покращуючи навички аудіювання та говоріння.
- Культурне розуміння: занурення сприяє розумінню культури, ідіоматичних виразів та невербальних комунікативних сигналів.
- Покращення вільного володіння мовою: розмовна практика з носіями мови покращує вимову, інтонацію та загальну плавність мови.

Недоліки:

- Обмежений доступ: можливості занурення можуть бути географічно обмеженими або фінансово недоступними для деяких учнів.
- Початковий мовний бар'єр: перехід до англомовного середовища може бути складним на початковому етапі, що може спричинити труднощі у спілкуванні та потенційне розчарування.
- Відсутність формального навчання: занурення саме по собі може не забезпечити структурованого навчання в таких сферах, як граматика або розширення словникового запасу.

Навчальні словники:

Переваги:

- Розширення словникового запасу: словники пропонують широкий спектр слів, визначень, синонімів та прикладів, що сприяє розвитку словникового запасу.
- Точність мови: словники забезпечують правильне написання, вимову та граматичне вживання слів, гарантуючи точність письмової та розмовної англійської мови.
- Контекстуальне розуміння: багато словників містять приклади речень, які допомагають учням зрозуміти, як слова використовуються в різних контекстах.
- Поради щодо вимови: багато навчальних словників містять фонетичні транскрипції або символи, які допомагають правильно вимовляти слова. Ці вказівки допомагають покращити свою розмовну англійську, надаючи рекомендації щодо наголосу, інтонації та поділу на склади.
- Інструмент для самостійного навчання: словники дають учням можливість стати незалежними користувачами мови. використовуючи їх, учні можуть досліджувати нові слова, перевіряти значення та вдосконалювати свої мовні навички без необхідності постійної допомоги чи покладання на інших.
- Самокорекція та усвідомлення помилок: словники дають змогу учням самостійно виправляти та усвідомлювати свої мовні помилки. Звертаючись до словників, учні можуть виявляти помилки, розуміти правильне вживання та покращувати точність своєї мови.

- Ресурс для навчання впродовж життя: словники є цінним ресурсом протягом усього процесу вивчення мови. Навіть досвідчені носії англійської мови продовжують користуватися словниками, щоб розширити свій словниковий запас, вдосконалити мовні навички та забезпечити точність у спілкуванні.

Недоліки:

- Надмірна довіра: покладання виключно на словники може перешкоджати розвитку незалежних мовних навичок і обмежувати творчий підхід до використання мови.
- Забирає багато часу: постійне звернення до словників під час розмови або читання може порушити потік спілкування і сповільнити процес навчання.
- Відсутність використання в режимі реального часу: словники не завжди відображають найновіші ідіоматичні вирази, сленг або використання мови, що розвивається.

Різні методи вивчення англійської мають свої унікальні переваги та недоліки. Навчання в класі забезпечує структуру та керівництво, в той час, як самонавчання пропонує гнучкість та незалежність. Занурення в мовне середовище та розмовна практика покращують вільне володіння та розуміння культури.

Навчальні словники відіграють вирішальну роль у вивченні англійської. Вони надають вичерпні мовні довідки, допомагають розширити словниковий запас, пояснюють значення слів, керують вимовою, підтримують розуміння граматики та сприяють самостійному навчанню. Ефективно використовуючи словники, люди можуть підвищити свій рівень володіння, точність і загальну компетенцію в англійській мові.

1.3 Аналіз існуючий словників

Нижче наведено деякі з найпопулярніших одномовних словників для вивчення англійської мови:

Oxford Learner's Dictionaries [1]

Це надійний ресурс для тих, хто вивчає англійську мову. Він пропонує вичерпні визначення, приклади речень, синоніми та аудіо-вимови. Словники орієнтовані саме

на студентів, надаючи інформацію про вживання слів, словосполучення та граматичні моделі. Переваги та недоліки наведено у таблиці 1.3.1.

Оксфордський словник призначено для студентів різних рівнів, від початкового до просунутого. Орієнтований на учнів і доступний стиль викладу робить його особливо придатним для учнів середнього рівня, які прагнуть розширити свій словниковий запас, поліпшити розуміння вживання слів і підвищити загальний рівень володіння мовою. Початківці також можуть скористатися його зручним для користувача макетом і простими визначеннями, тоді як просунуті студенти можуть використовувати його як швидкий довідник для пошуку нюансів у значенні та вживанні слів.

Таблиця 1.3.1

Переваги	Недоліки
– всебічне охоплення; – ясність і доступність; – підтримка вимови; – багаті приклади речень; – функції, орієнтовані на студента.	– обмежена глибина; – відсутність контекстної варіативності.

Cambridge English Dictionary [2]

Кембриджський словник широко використовується тими, хто вивчає англійську мову в усьому світі. Він містить чіткі визначення, приклади речень та аудіозаписи вимови. Він також пропонує додаткові функції, такі як вправи на розширення словникового запасу та граматичні підказки. Переваги та недоліки наведено у таблиці 1.3.2.

Словник англійської мови підходить для тих, хто вивчає англійську мову на всіх рівнях володіння мовою. Він надає підтримку та ресурси для початківців, пропонуючи чіткі визначення, вказівки щодо вимови та приклади речень. Користувачі середнього рівня можуть скористатися додатковою інформацією про мовні рівні та позначками, які допоможуть їм зрозуміти, як правильно використовувати мову в різних контекстах. Просунуті студенти можуть покладатися на словник для отримання детальних визначень, більш нюансованих значень та точних рекомендацій щодо вимови.

Таблиця 1.3.2

Переваги	Недоліки
– надійна та точна інформація; – всеобічне охоплення лексики; – мовні рівні та позначки щодо вживання; – поради щодо вимови; – приклади речень та вживання; – доступний та зручний інтерфейс.	– обмежена розмовна мова; – відсутність оновлень у режимі реального часу.

Collins Dictionary [3]

Пропонує всеосяжний онлайн-ресурс для тих, хто вивчає англійську мову. Він містить детальні визначення, приклади речень та аудіо вимови. Він також включає поради щодо використання мови, граматичні пояснення та різноманітні навчальні ресурси. Переваги та недоліки наведено у таблиці 1.3.3.

Онлайн-словник Collins English Dictionary підходить для тих, хто вивчає англійську мову на різних рівнях. Він підходить для початківців, пропонуючи чіткі визначення та прості мовні пояснення. Людям середнього рівня отримують вигоду від великого словникового запасу, допомоги з вимовою та прикладів використання. Просунуті знавці мови можуть використовувати словник для подальшого вдосконалення своїх навичок, вивчення відтінків значень та доступу до синонімів і антонімів для більш точного використання мови.

Таблиця 1.3.3

Переваги	Недоліки
– широке охоплення слів; – чіткі та стислі пояснення; – допомога з вимовою; – приклади вживання; – синоніми та антоніми; – регулярні оновлення.	– відсутність мовних варіацій у режимі реального часу; – обмежена контекстуальна інформація.

Longman Dictionary of Contemporary English (LDOCE) [4]

Словник сучасної англійської мови Longman Dictionary of Contemporary English відомий своїми зручними для людей функціями. Він пропонує чіткі визначення,

приклади речень, аудіо вимови та примітки до вживання. Він також надає мовні ресурси, такі як інформація про словосполучення, граматичні пояснення та інтерактивні вправи. Переваги та недоліки наведено у таблиці 1.3.4.

Longman підходить для тих, хто вивчає англійську мову на всіх рівнях. Зручний інтерфейс, чіткі визначення та велика кількість прикладів роблять його доступним для початківців. Студенти середнього та просунутого рівнів можуть скористатися його всеосяжним змістом, порадами щодо словосполучень та детальними поясненнями вживання слів. Професіонали та викладачі англійської мови також знаходять LDOCE цінним завдяки детальній інформації та точним мовним довідникам.

Таблиця 1.3.4

Переваги	Недоліки
– вичерпний зміст; – чіткі та доступні визначення; – підтримка вимови; – поради щодо словосполучень та вживання; – розширення словникового запасу; – граматика та примітки до вживання; – цифрова доступність.	– незліченна кількість інформації.

Macmillan [5]

Словник Macmillan є популярним вибором для тих, хто вивчає англійську мову. Він містить детальні визначення, приклади речень, аудіо вимови та примітки щодо вживання слів. Він також пропонує мовні ресурси, включаючи граматичні посібники та інтерактивні вікторини. Переваги та недоліки наведено у таблиці 1.3.5.

Чіткі пояснення, вказівки щодо вимови та приклади роблять словник цінним ресурсом для тих, хто вивчає мову, на будь-якому етапі їхньої мовної подорожі. Початківці можуть скористатися простими визначеннями та прикладами речень, тоді як більш просунуті студенти можуть заглибитися в нюанси використання мови та розширити свій словниковий запас завдяки всебічному охопленню.

Таблиця 1.3.5

Переваги	Недоліки
– всеобічне охоплення; – актуальна інформація; – чіткі визначення та приклади; – посібник з вимови; – контекстні приклади.	– обмежена кількість прикладів використання; – технічна термінологія.

Merriam-Webster Learner's Dictionary [6]

Merriam-Webster Learner's Dictionary спеціально розроблений для тих, хто вивчає англійську мову. Він містить визначення, приклади речень, аудіо-вимови та вправи на розширення словникового запасу. Він також пропонує функцію "Слово дня" для покращення щоденного поповнення словникового запасу. Переваги та недоліки наведено у таблиці 1.3.6.

Онлайн-словник Merriam-Webster підходить для тих, хто вивчає англійську мову на різних рівнях, від початкового до просунутого. Зручні визначення, чіткі пояснення та вказівки щодо вимови роблять його доступним для початківців, які формують свій базовий словниковий запас та мовні навички.

Таблиця 1.3.6

Переваги	Недоліки
– повнота та авторитетність; – допомога з вимовою; – зручні визначення; – синоніми та антоніми; – слово дня; – додаткові мовні ресурси.	– обмежене контекстне використання; – брак розмовної мови.

Ці онлайн-словники спеціально розроблені для тих, хто вивчає англійську мову, і надають цінні ресурси для поповнення словникового запасу, розуміння мови та покращення вимови. Їх популярність зумовлена надійністю, зручним інтерфейсом та функціями, орієнтованими на учня, що робить їх основними джерелами інформації для учнів з різним рівнем володіння мовою.

РОЗДІЛ 2. ВЕБ-ФРЕЙМВОРКИ DJANGO ТА REACT ТА ПЛАТФОРМА РОЗРОБКИ FIREBASE

2.1 Мова програмування JavaScript

JavaScript – це потужна мова сценаріїв на стороні клієнта. JavaScript в першу чергу використовується для покращення взаємодії користувача з веб-сайтом. Іншими словами, за допомогою JavaScript можна зробити веб-сайт більш живим та інформативнішим; також широко використовується для рендерингу та розробки мобільних пристроїв. На даний момент ця мова програмування є найпопулярнішою в світі. [7]

Існують також більш складні варіанти JavaScript на стороні сервера, такі як Node.js, які дають змогу додати до веб-сайту більше можливостей, ніж просто завантажувати файли (наприклад, зв'язок у реальному часі між кількома комп'ютерами).

JavaScript широко використовується для створення веб-сторінок, а також у деяких небраузерних додатках також використовується. JS можна використовувати як для клієнтських, так і для серверних розробок. JavaScript містить низку базових об'єктів, таких як послідовність, дата та математика, а також основний набір лінгвістичних структур, таких як оператори, керуючі структури та фрази.

Ключові моменти про JavaScript

Написання сценаріїв на стороні клієнта: JavaScript в основному використовується для написання сценаріїв на стороні клієнта, тобто запускається у веб-браузері користувача. Він може маніпулювати вмістом веб-сторінки, реагувати на дії користувача та динамічно оновлювати сторінку, не вимагаючи звернення до сервера в обидва боки.

Синтаксис і структура: JavaScript має C-подібний синтаксис і часто згадується як ECMAScript, що є офіційною специфікацією мови. Він використовує крапку з комою для розділення операторів і фігурні дужки для визначення блоків коду. Вона чутлива до регістру і підтримує як однорядкові, так і багаторядкові коментарі.

Типи даних: JavaScript підтримує різні типи даних, включаючи числа, рядки, булеві, масиви, об'єкти тощо. Він також має спеціальне значення *null*, яке означає

відсутність будь-якого значення об'єкта, і *undefined*, яке означає відсутність визначеного значення.

Динамічна типізація: JavaScript має динамічну типізацію, це означає, що не потрібно оголошувати тип даних змінної. Змінні можуть зберігати значення будь-якого типу, а їхні типи можуть змінюватися під час виконання програми.

Функції: вони є фундаментальною частиною JavaScript. Можна визначати функції за допомогою ключового слова *function* і використовувати їх для інкапсуляції блоків коду, які можна повторювати. JavaScript також підтримує анонімні функції, функції зі стрілками та функції вищого порядку.

Маніпуляції з DOM: об'єктна модель документа (DOM) представляє структуру HTML-документа як деревоподібну структуру. JavaScript дозволяє отримувати доступ до DOM і маніпулювати нею, що дає змогу динамічно змінювати вміст, стиль і поведінку веб-сторінки.

Обробка подій: JavaScript може реагувати на дії користувача, такі як кліки, рухи миші, натискання клавіш та заповнення форм, за допомогою обробки подій. Можна прикріпити слухачі подій до елементів HTML і визначити код JavaScript, який буде виконуватися при виникненні подій.

Асинхронне програмування: JavaScript підтримує асинхронне програмування за допомогою *callback, promise* та синтаксису *async/await*. Це дозволяє обробляти трудомісткі операції, такі як мережеві запити або читання файлів, не блокуючи виконання іншого коду.

Бібліотеки та фреймворки: JavaScript має велику екосистему бібліотек та фреймворків, які розширюють його функціональність. Популярні бібліотеки, такі як jQuery, спрощують маніпуляції з DOM, тоді як фреймворки, такі як React, Angular та Vue.js, надають потужні інструменти для створення складних веб-додатків.

Розробка на стороні сервера: хоч JavaScript в основному використовується для написання сценаріїв на стороні клієнта, він також може бути використаний для розробки на стороні сервера. Node.js, середовище виконання JavaScript, побудоване на Chrome V8, дозволяє запускати JavaScript на сервері, забезпечуючи повноцінну розробку на JavaScript.

Клієнтська частина [7]

JavaScript надає об'єкти для використання користувачем та його об'єктну модель документа (DOM). Завдяки розширенням з боку клієнта, програма може розміщувати елементи на HTML-формі та реагувати на події користувача, такі як клацання мишею, введення даних та навігацію по сторінці. Використання повноцінних клієнтських фреймворків таких як AngularJS, ReactJS, VueJS та багато інших.

Серверна частина [7]

JavaScript надає об'єкти, необхідні для запуску JavaScript на сервері. Таким чином серверні розширення дозволяють додатку взаємодіяти з базою даних, щоб забезпечувати безперервність даних від одного виклику до іншого або виконувати маніпуляції з файлами на сервері. Найпопулярнішим фреймворком для роботи з серверною частиною на сьогодні є node.js.

JavaScript постійно розвивається, і до мови регулярно додаються нові функції та вдосконалення. Ця мова програмування стала важливим інструментом для веб-розробників, що дозволяє їм створювати динамічні, інтерактивні та адаптивні сайти.

2.2 Бібліотека React

React – це популярна JavaScript бібліотека для створення користувацьких інтерфейсів. Розроблена компанією Facebook і випущена у світ у 2013 році, широко використовується для створення інтерактивних, багаторазових компонентів інтерфейсу користувача та лежить в основі одних з найпоширеніших додатків, зокрема, Facebook та Instagram, а також незліченної кількості інших застосунків. React дотримується компонентної архітектури, де інтерфейс розділений на невеликі, самодостатні компоненти, які можна комбінувати для побудови складних користувацьких інтерфейсів. [8, 9]

Ключові моменти про React

Компонентно-орієнтований: React заохочує модульний підхід до побудови інтерфейсу користувача. Компоненти – це будівельні блоки React-додатків, що представляють собою незалежні частини користувацького інтерфейсу, які можна

використовувати повторно. Кожен компонент може мати власний стан та властивості (пропси).

Віртуальний DOM: React використовує віртуальне представлення реального DOM, яке називається «Віртуальний DOM». Він підтримує полегшену копію DOM і ефективно оновлює реальний, коли це необхідно. Такий підхід покращує продуктивність, мінімізуючи прямі маніпуляції з DOM.

JSX: JSX – це синтаксичне розширення для JavaScript, яке дозволяє писати HTML-подібний код безпосередньо в JavaScript. Воно поєднує в собі можливості JavaScript та HTML, полегшуючи визначення структури та зовнішнього вигляду React-компонентів.

Односторонній потік даних: React використовує односпрямований потік даних. Це означає, що дані передаються від батьківських компонентів до дочірніх через пропси. Дочірні компоненти не можуть безпосередньо змінювати дані, отримані від пропсів, що забезпечує передбачуваний і контрольований потік даних.

Реактивні оновлення: React ефективно оновлює інтерфейс користувача, автоматично визначаючи зміни в стані або реквізитах компонента. Коли стан або пропси змінюються, React повторно рендерить лише відповідні компоненти або частини інтерфейсу, мінімізуючи непотрібні повторні візуалізації та покращуючи продуктивність.

Методи життєвого циклу: React надає методи життєвого циклу, які дозволяють підключатися до різних етапів життєвого циклу компонента. Можна виконувати ініціалізацію, оновлення стану та очищення за допомогою таких методів, як *componentDidMoun, componentWillUnmount* та інших.

Декларативний синтаксис: React використовує декларативний синтаксис, це означає, що розробник описує, як має виглядати інтерфейс на основі поточного стану, а React дбає про оновлення фактичного інтерфейсу відповідно до бажаного стану. Такий підхід спрощує код і полегшує міркування.

React Router: це популярна бібліотека, яка надає можливості маршрутизації для React-додатків. Вона дозволяє створювати декілька сторінок або представлень в

межах односторінкового додатку, забезпечуючи навігацію між різними компонентами без повного перезавантаження сторінки.

Хуки React: функції, які дозволяють використовувати стан та інші можливості React у функціональних компонентах. У багатьох випадках хуки усувають потребу в компонентах класів і надають більш простий та гнучкий спосіб управління станами та побічними ефектами.

Екосистема та спільнота: React має велику та активну спільноту, що призвело до створення процвітаючої екосистеми бібліотек, інструментів та ресурсів. Можна знайти численні сторонні бібліотеки, бібліотеки компонентів інтерфейсу користувача (наприклад, Material-UI та Ant Design), рішення для управління станом (наприклад, Redux та MobX) та багато іншого для покращення вашого досвіду розробки React.

Отже, React – це потужна JavaScript бібліотека для створення ефективних користувацьких інтерфейсів. Її компонентна архітектура, віртуальний DOM, декларативний синтаксис та широка екосистема роблять її популярним вибором для фронтенд-розробки. React надає розробникам потужний та ефективний спосіб створення інтерактивних та динамічних користувацьких інтерфейсів.

2.3 Мова програмування Python

Python – це інтерпретована, об'єктно-орієнтована мова програмування високого рівня з динамічною семантикою. Високорівневі вбудовані структури даних у поєднанні з динамічною типізацією та динамічним зв'язуванням роблять її дуже привабливою для швидкої розробки додатків, а також для використання в якості мови сценаріїв або мови для з'єднання існуючих компонентів між собою. Простий, легкий для вивчення синтаксис Python підкреслює читабельність, а отже, зменшує витрати на підтримку програм. Python підтримує модулі та пакети, що заохочує модульність програм та повторне використання коду. [10]

Ключові моменти про Python

Читабельність і простота: Python підкреслює читабельність коду завдяки чистому та інтуїтивно зрозумілому синтаксису. Він використовує відступи та пробіли

для визначення блоків коду, що робить його легким для розуміння та зменшує потребу в надмірній пунктуації.

Мова загального призначення: Python – це мова загального призначення, означає, що її можна використовувати для широкого спектру застосувань. Вона широко використовується для веб-розробки, наукових обчислень, аналізу даних, штучного інтелекту, автоматизації, написання сценаріїв тощо.

Інтерпретована мова: Python є інтерпретованою мовою, її не потрібно компілювати перед виконанням. Інтерпретатор Python читає і виконує код рядок за рядком, що робить його зручним для розробки та експериментів.

Велика стандартна бібліотека: Python постачається з великою стандартною бібліотекою, яка надає широкий спектр модулів та функцій для різних завдань. Вона пропонує функції для файлового вводу/виводу, роботи в мережі, регулярних виразів, серіалізації даних, багатопотоковості та багато іншого, що робить її потужною мовою *out of the box*.

Динамічна типізація: Python є динамічно типізованою мовою, не потрібно оголошувати тип даних змінної явно. Типи змінних визначаються під час виконання програми, що забезпечує більшу гнучкість, але також вимагає ретельної уваги до роботи з типами. [11, 12]

Об'єктно-орієнтоване програмування (ООП): Python підтримує об'єктно-орієнтоване програмування, що дозволяє визначати класи, створювати об'єкти та застосовувати такі принципи, як успадкування, інкапсуляція та поліморфізм. ООП у Python забезпечує багаторазове використання коду та модульність.

Широкі сторонні бібліотеки: Python має багату екосистему сторонніх бібліотек та фреймворків, які розширюють її функціональність. Популярні бібліотеки включають NumPy для чисельних обчислень, Pandas для маніпулювання даними, Django та Flask для веб-розробки, TensorFlow та PyTorch для машинного навчання та багато інших. [11]

Крос-платформна сумісність: Python є крос-платформенною мовою, яка працює на основних операційних системах, таких як Windows, macOS та Linux. Це

дозволяє розробникам писати код один раз і запускати його на різних платформах без значних модифікацій. [11]

Спільнота та підтримка: Python має велику та активну спільноту розробників, які роблять свій внесок у її розвиток. Надає велику документацію, навчальні посібники, форуми та проєкти з відкритим вихідним кодом, що дозволяє легко знаходити допомогу, навчатися та співпрацювати з іншими розробниками Python.

Легка інтеграція: Python легко інтегрується з іншими мовами, такими як C, C++ та Java. Ця гнучкість дозволяє розробникам використовувати існуючі бібліотеки та застосовувати Python для написання сценаріїв і завдань вищого рівня, одночасно використовуючи критичний до продуктивності код з інших мов.

Популярність мови програмування Python продовжує зростати завдяки його простоті, універсальності та широкому спектру додатків, які він підтримує. Вона широко використовується в різних галузях і сферах, що робить її чудовим вибором як для початківців, так і для досвідчених розробників.

2.4 Фреймворк Django

Django – це безкоштовний фреймворк з відкритим вихідним кодом, який може прискорити розробку веб-додатків, створених на мові програмування Python. [13]

Як працює Django?

Django слідує шаблону проєктування MVT (Model View Template – шаблон представлення моделі).

- Модель – дані, які ви хочете представити, зазвичай це дані з бази даних.
- Представлення – обробник запиту, який повертає відповідний шаблон і вміст - на основі запиту користувача.
- Шаблон – текстовий файл (наприклад, HTML-файл), що містить макет веб-сторінки з логікою відображення даних. [13]

Django був вперше випущений у 2005 році, і на той час більшість веб-сайтів складалися з однієї великої монолітної кодової бази. «Внутрішня частина» складалася з моделей баз даних, URL-адрес і представлень, які взаємодіяли з «зовнішніми»

шаблонами HTML, CSS і JavaScript, що контролювали презентаційний макет кожної веб-сторінки. [14]

Однак в останні роки підхід «API-first» став, мабуть, домінуючою парадигмою у веб-розробці. Цей підхід передбачає формальне відокремлення бек-енду від фронтенду. Це означає, що Django стає потужною базою даних та API, а не просто фреймворком для створення веб-сайтів. [14]

Ключові моменти про Django

Швидка розробка: Django призначений для швидкої розробки веб-додатків. Він надає набір інструментів, бібліотек і вбудованих функцій, які вирішують типові завдання веб-розробки, дозволяючи розробникам зосередитися на логіці програми, а не на низькорівневих деталях реалізації.

Batteries Included: Django слідує філософії "batteries included", що означає, що він включає в себе багато можливостей і функцій з коробки. Він надає ORM (об'єктно-реляційне відображення) для керування базами даних, механізм шаблонів, обробку форм, автентифікацію користувачів та багато іншого. Це зменшує потребу в зовнішніх залежностях і спрощує процес розробки.

Інтерфейс адміністрування: Django включає потужний інтерфейс адміністрування, що налаштовується. За допомогою мінімального коду розробники можуть створити повнофункціональну панель адміністрування, яка дозволить їм керувати та маніпулювати даними в базі даних додатку.

Маршрутизація URL-адрес: Django надає надійну систему маршрутизації URL-адрес, яка зіставляє URL-адреси з *views*. Це дозволяє розробникам визначати чисті та читабельні URL-адреси для своїх веб-додатків і пов'язувати їх з певними функціями або поданнями.

Підтримка ORM та баз даних: Django включає потужне об'єктно-реляційне відображення (ORM), яке абстрагує рівень бази даних і дозволяє розробникам взаємодіяти з базою даних за допомогою коду на Python. Він підтримує безліч баз даних, включаючи такі популярні, як PostgreSQL, MySQL, SQLite та Oracle.

Функції безпеки: Django включає декілька функцій безпеки, які допомагають розробникам створювати безпечні веб-додатки. Вони включають захист від

поширених веб-уразливостей, таких як міжсайтовий скриптинг (XSS), підробка міжсайтових запитів (CSRF), SQL- injection та багато іншого.

Масштабованість та продуктивність: Django розроблений для роботи з веб-сайтами з високим трафіком та масштабованими додатками. Він надає такі функції, як кешування, пул з'єднань з базами даних та асинхронне виконання завдань, щоб оптимізувати продуктивність та впоратися зі збільшеним користувацьким навантаженням.

Спільнота та екосистема: Django має велику та активну спільноту розробників, які роблять свій внесок у її розвиток. Надає обширну документацію, навчальні посібники, багаторазові пакунки та підтримує екосистему. Багато популярних веб-сайтів, включаючи Instagram та Pinterest, було створено з використанням Django.

Універсальність та інтеграція: Django можна інтегрувати з іншими бібліотеками та фреймворками Python, що робить його гнучким та універсальним. Його можна поєднувати з фронтенд-фреймворками, такими як React або Angular, для створення сучасних веб-додатків, або інтегрувати з бібліотеками науки про дані, такими як NumPy та Pandas, для аналізу та візуалізації даних.

Поєднання продуктивності, масштабованості та надійності Django зробило його популярним вибором для розробки веб-додатків. Зосередженість на чистоті коду, безпеці та швидкій розробці робить його придатним для проєктів будь-якого розміру, від невеликих прототипів до великомасштабних виробничих додатків.

2.5 Платформа розробки Firebase

Firebase – це платформа для розробки додатків, яка допомагає створювати та розвивати програми та ігри. Підтримується Google та користується довірою мільйонів компаній по всьому світу. [15]

Ключові особливості:

- Підтримує автентифікацію за допомогою паролів, телефонних номерів, Google, Facebook, Twitter тощо. Firebase Authentication (SDK) можна використовувати для ручної інтеграції одного або декількох методів входу в додаток.

- Дані синхронізуються між усіма клієнтами в режимі реального часу і залишаються доступними, навіть коли додаток переходить в офлайн.
- Firebase Hosting забезпечує швидкий хостинг для веб-додатків; контент кешується в мережах доставки контенту по всьому світу
- Додаток тестується на віртуальних і фізичних пристроях, розташованих в дата-центрах Google.
- Сповіщення можна надсилати за допомогою Firebase без додаткового кодування.

2.5.1 Автентифікація в Firebase

Автентифікація в Firebase – це процес перевірки особи користувачів, які отримують доступ до вашого додатку або сервісів. Firebase надає комплексну і просту у використанні систему автентифікації, яка дозволяє розробникам додавати функції автентифікації та авторизації користувачів до своїх додатків з мінімальними зусиллями.

Методи автентифікації користувачів

Автентифікація Firebase підтримує різні методи автентифікації, щоб задовольнити різні уподобання користувачів. До них відносяться:

- Електронна пошта/пароль – користувачі можуть реєструватися і входити в систему, використовуючи свою електронну пошту і комбінацію пароля.
- Вхід через соціальні мережі – Firebase пропонує інтеграцію з популярними соціальними мережами, такими як Google, Facebook, Twitter, GitHub та іншими, що дозволяє користувачам входити в систему, використовуючи свої існуючі облікові записи на цих платформах.
- Аутентифікація за номером телефону – вбудована система для перевірки телефонних номерів користувачів за допомогою SMS-кодів.
- Анонімна автентифікація – користувачі можуть входити в систему анонімно, без необхідності надавати будь-які облікові дані. Це корисно для сценаріїв, коли користувачам дозволений доступ до певних функцій, не вимагаючи від них створення облікового запису.

Firebase Authentication обробляє весь процес автентифікації, включаючи безпечне зберігання облікових даних користувача, шифрування конфіденційної інформації та захист від поширених загроз безпеки, таких як міжсайтовий скриптинг (XSS) і підробка міжсайтових запитів (CSRF).

Firebase Authentication пропонує набір API та інструментів для управління обліковими записами користувачів. Розробники можуть створювати, оновлювати та видаляти облікові записи користувачів програмно. Додаткова інформація про користувача, така як імена та зображення профілю, може зберігатися в об'єкті користувача Firebase.

Firebase надає функції для роботи з ролями та дозволами користувачів. Можна визначати користувацькі ролі та обмежувати доступ до певних ресурсів або функціональних можливостей на основі ролей користувачів або певних умов. Це допомагає забезпечити дотримання правил авторизації і гарантувати, що тільки авторизовані користувачі можуть виконувати певні дії у вашому додатку.

Firebase Authentication легко інтегрується з іншими службами та функціями Firebase. Можна використовувати облікові дані автентифікованого користувача для контролю доступу до інших служб Firebase, таких як база даних в режимі реального часу або хмарне сховище Firestore, і персоналізувати взаємодію з користувачами на основі їх статусу автентифікації.

Firebase пропонує готові бібліотеки інтерфейсу користувача та SDK для різних платформ та мов програмування. Ці бібліотеки надають готові до використання компоненти інтерфейсу користувача для автентифікації та допоміжні методи, що полегшує реалізацію потоків автентифікації користувачів у вашому додатку.

Firebase Authentication надає функції для підвищення безпеки, такі як багатофакторна автентифікація (MFA), яка додає додатковий рівень перевірки, вимагаючи від користувачів надати додатковий фактор, наприклад, SMS-коди або біометричну автентифікацію.

Firebase Authentication інтегрується з Firebase Analytics, що дозволяє відстежувати події, пов'язані з автентифікацією користувачів, контролювати дії

користувачів при вході в систему, а також отримувати інформацію про поведінку та активність користувачів.

Firebase Authentication спрощує процес впровадження автентифікації користувачів і дозволяє розробникам зосередитися на створенні безпечного та персоналізованого користувацького досвіду. Вона позбавляє від необхідності створювати складні процеси автентифікації з нуля і надає масштабоване і надійне рішення для управління користувачами у додатках.

2.5.2 База даних Firestore

Firestore – це гнучка та масштабована база даних документів NoSQL, що надається Firebase, платформою, розробленою Google. Вона призначена для створення веб- та мобільних додатків у режимі реального часу, дозволяючи розробникам зберігати, запитувати та синхронізувати дані в режимі реального часу на різних пристроях і платформах.

Особливості та концепції, пов'язані з Firestore:

- документно-орієнтована модель даних;
- синхронізація даних у реальному часі;
- масштабованість та продуктивність;
- потужні запити;
- індекси;
- безпека та автентифікація;
- атомарні операції та транзакції;
- безсерверне та автоматичне масштабування;
- інтеграція з екосистемою Firebase;
- SDK та бібліотеки.

Firestore широко використовується для створення додатків для спільної роботи в режимі реального часу, чатів, мобільних і веб-додатків, які потребують синхронізації даних у реальному часі та масштабованості. Завдяки гнучкій моделі даних, потужним можливостям запитів і бездоганній інтеграції з екосистемою

Firebase, Firestore спрощує процес розробки і дозволяє розробникам зосередитися на створенні цікавих і адаптивних додатків.

2.5 Що таке API?

API, або інтерфейс прикладного програмування, – це набір правил і протоколів, які дозволяють різним програмним додаткам спілкуватися один з одним. Він визначає, як повинні взаємодіяти різні компоненти програмних систем, вказуючи методи, формати даних і механізми автентифікації, що використовуються для комунікації. [16]

API дозволяють розробникам отримувати доступ і використовувати функціональні можливості та дані інших програмних додатків, сервісів або платформ без необхідності розуміти або модифікувати базову реалізацію. Вони надають стандартизований інтерфейс, який абстрагується від складнощів системи, полегшуючи інтеграцію різних програмних компонентів і створення додатків більш ефективно.

API можуть слугувати різним цілям, зокрема

- отримання даних;
- інтеграція функціональності;
- взаємодія зі службами;
- розширення додатків.

API можна класифікувати на різні типи, такі як

- *Веб-інтерфейси*, також відомі як HTTP-інтерфейси або REST-інтерфейси, доступні через Інтернет за допомогою протоколу HTTP. Зазвичай вони надають ресурси та операції, доступні за допомогою URL-адрес і методів HTTP, таких як GET, POST, PUT і DELETE. Веб-API часто повертають дані у форматах JSON або XML.
- *Бібліотеки або фреймворки* надають API для взаємодії з їхніми функціональними можливостями та ресурсами. Ці API зазвичай специфічні для мови програмування або фреймворку і дозволяють розробникам

використовувати готовий код або сервіси без необхідності розуміти базову реалізацію.

- *Операційні системи* пропонують API, які дозволяють розробникам взаємодіяти з системними ресурсами, апаратними пристроями або специфічними функціями, що надаються операційною системою. Ці API дозволяють розробникам додатків створювати програмне забезпечення, яке працює на різних платформах.
- *Бази даних* надають API для взаємодії з їхніми даними та виконання операцій з базами даних, таких як запити, вставка, оновлення або видалення записів. Ці API дозволяють розробникам отримувати доступ до даних, що зберігаються в системі баз даних, і маніпулювати ними.

API стали фундаментальною частиною сучасної розробки програмного забезпечення, надаючи інтеграцію, сумісність та співпрацю між різними програмами та сервісами. Вони дають можливість розробникам використовувати існуючі функції, отримувати доступ до ресурсів і створювати потужні додатки, об'єднуючи різні програмні компоненти разом.

2.6 Редактор Visual Studio Code

Visual Studio Code, часто згадуваний як VS Code, є популярним редактором вихідного коду, розробленим компанією Microsoft. Він надає легке та універсальне середовище для редагування, налагодження та керування кодом на різних мовах програмування та платформах. Ось деякі ключові особливості та аспекти Visual Studio Code:

- *Міжплатформна підтримка*

  Visual Studio Code доступний для Windows, macOS та Linux, що робить його доступним для розробників на різних операційних системах.
- *Розширюваність та налаштування*

  VS Code має потужну екосистему розширень, яка дозволяє розробникам покращувати та налаштовувати свій досвід кодування. Існує широкий спектр налаштувань для різних мов програмування, фреймворків та процесів розробки.

Розширення можуть надавати додаткову мовну підтримку, фрагменти коду, інтеграцію зі сторонніми інструментами тощо.

- *Інтуїтивно зрозумілий інтерфейс*

  Visual Studio Code має зручний та інтуїтивно зрозумілий інтерфейс. Він пропонує чистий макет з бічною панеллю для навігації файлами, гнучку область редактора та багатий набір функцій, доступних через меню, кнопки та комбінації клавіш.

- *Інтегрований термінал*

  VS Code включає інтегрований термінал, який дозволяє розробникам запускати команди, виконувати скрипти та виконувати різні завдання, не виходячи з редактора. Це усуває необхідність перемикатися між редактором і зовнішніми терміналами.

- *Можливості редагування коду*

  Visual Studio Code надає широкий спектр можливостей редагування коду для підвищення продуктивності. Він пропонує інтелектуальне завершення коду, підсвічування синтаксису, форматування, навігацію та можливості рефакторингу. Підтримує інтеграцію з контролем версій Git для керування коду.

- *Інтегрований Git та контроль версій*

  Visual Studio Code має вбудовану інтеграцію з Git, що робить його зручним для роботи з системами контролю версій. Розробники можуть виконувати операції з Git'ом, такі як фіксація змін, розгалуження, злиття та вирішення конфліктів, не виходячи з редактора.

- *Автоматизація завдань*

  VS Code підтримує автоматизацію завдань за допомогою інтегрованого *task runner*. Розробники можуть створювати власні завдання або використовувати готові для автоматизації повторюваних завдань, таких як компіляція коду, запуск тестів або розгортання додатків.

- *Інтелектуальна та мовна підтримка*

  Visual Studio Code пропонує інтелектуальні підказки та автоматичне завершення коду, відомі як Intellisense, для різних мов програмування. Вона надає специфічні для кожної мови функції, такі як перевірка синтаксису, фрагменти коду та інтеграція мовних служб, щоб підвищити ефективність розробки.

- *Інтегроване середовище розробки (IDE)*

  Будучи легким редактором, Visual Studio Code пропонує багато можливостей, які зазвичай асоціюються з повноцінними IDE. До них відносяться: інтегроване налагодження, лінтування, профілювання коду, інтегрований термінал і підтримка систем складання; що робить його придатним для широкого спектра робочих процесів розробки.

Visual Studio Code набула популярності серед розробників завдяки своїй гнучкості, широкому набору функцій та потужній підтримці спільноти. Платформа продовжує розвиватися завдяки регулярним оновленням та внескам від розробників відкритого коду, що робить його вибором для багатьох програмістів.

РОЗІДІЛ 3. РЕАЛІЗАЦІЯ ТА РОЗРОБКА ВЕБ-ЗАСТОСУНУ «EDUDICTIONARY»

Для реалізації веб-застосунку навчального словника «EduDictionary» використано мови програмування Python для серверної частини програми та JavaScript для користувацького інтерфейсу. При розробці було застосовано технології:

- Django;
- React JS;
- Firebase;
- Firestore.

Щоб отримати потрібну інформацію, React JS надсилає запит до Django, який в свою чергу перетворює інформацію з API та виводить результати за допомогою першого. Звертаючись до бази даних можна отримати слово з колекції конкретного користувача, додати нове, видалити або ж створити новий акаунт. Результати попередніх дій виводяться через React JS. Схема проєкту наведена на рисунку 3.1.

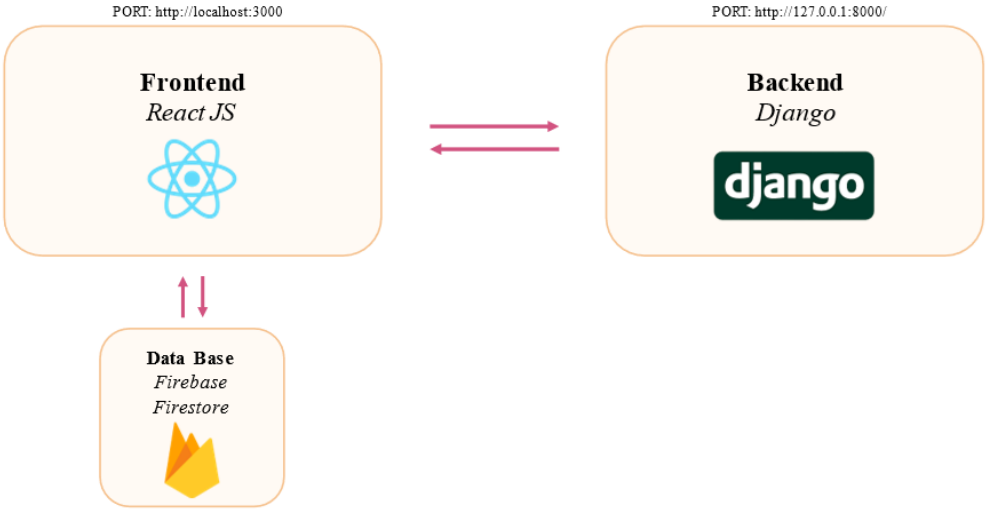

Рис. 3.1 – схема проєкту «Навчальний словник»

3.1 Створення Frontend частини проєкту

Першим кроком буде створення React-додатку у середовищі Visual Studio Code. Для цього в терміналі вводимо команду ***npm create react-app frontend***. Встановлюємо необхідні компоненти за допомогою наступних команд почергово:

1. **cd frontend** – перехід до папки frontend;
2. **npm install react-bootstrap** – встановлення React-Bootstrap (*використовує CSS фреймворк Bootstrap і замінює будь-який існуючий JavaScript на суто React-компоненти*); [17, 18, 19]
3. **npm install react-router-dom** – встановлення React Router DOM (*пакет, що містить стандартні компоненти та функції для реалізації маршрутизації в React-додатках*); [20]
4. **npm install react-router-bootstrap** – встановлення React Router Bootstrap (інтеграція між React Router та React-Bootstrap).

3.1.1 Опис роботи файлів з фрагментами коду

Header.js – реалізує верхнє статичне меню, де знаходяться назва веб-застосунку, посилання на вхід/реєстрацію, сторінку користувача. При авторизованому акаунті навігаційна панель змінюється, зникає «Log In» та додається «Dictionary» (рис. 3.1.1.1).

```
function Header() {
  const [authUser, setAuthUser] = useState(null);

  useEffect(() => {···
  }, []);
  return (
    <header style={{backgroundColor: '#D25380'}}>
    <Navbar variant="dark">
      <Container>
        <LinkContainer to='/'>
            <Navbar.Brand><h1>EduDictionary</h1></Navbar.Brand>
        </LinkContainer>

        <Navbar.Toggle aria-controls="basic-navbar-nav" />
              <Navbar.Collapse id="basic-navbar-nav">
                 {!authUser?<Nav className="mr-auto">
                <LinkContainer to='/signin'>
                  <Nav.Link><i className="fas fa-sign-in"></i> Log in</Nav.Link>
                </LinkContainer>
                  </Nav>:<></>}

                  <Nav className='ms-auto'>
                  {authUser ?(
                  <LinkContainer to='/dictionary'>
                  <Nav.Link ><i className="fas fa-book"></i> Dictionary</Nav.Link>
                </LinkContainer>):<></>
}
                  <LinkContainer to='/user'>
                  <Nav.Link><i className="fas fa-user "></i> User</Nav.Link>
                </LinkContainer>
                </Nav>
              </Navbar.Collapse>
        </Container>
    </Navbar>
  </header>
  )
}

export default Header
```

Рис. 3.1.1.1 – верхня панель

Footer.js – виводить внизу веб-застосунку фіксований текст (рис. 3.1.1.2).

```
<footer>
  <Container>
    <Row>
      <Col className='text-center py-3'>Copyright &copy; Yelyzaveta Pavliuk</Col>
    </Row>
  </Container>
</footer>
```

Рис. 3.1.1.2 – текст внизу застосунку

Search.js – пошук слова на головній сторінці; при натисканні на кнопку «Search», користувача скеровує на сторінку з результатом (рис. 3.1.1.3).

```
function Search() {
  const [word, setWord] = useState('')
  const navigate = useNavigate()
  const handleWordChange = (event) => {
    setWord(event.target.value);
  };

  const navigateToWordScreen = () => {
    navigate('/word', { state: { word: word } });
  }

  return (

    <Container className="input-group pr">
      <Form.Control size="lg" ···
      style={{borderColor: '#D25380', borderRadius: 50}}/>
      <Button ···
      </Button>
    </Container>
  );
}
```

Рис. 3.1.1.3 – пошук

Article.js – виводить на домашній сторінці посилання на статті зі вказаними заголовками та картинками (рис. 3.1.1.4). При натисканні направляє до публікації.

```
function Article({ article }) {
  return (
    <Card className="my-3 p-3" style={{ borderColor: '#D25380', borderRadius: 25, height: '100%' }}>
      <Link to={`/article/${article._id}`}>
        <Card.Img src={article.image} className="card-img-top embed-responsive-item" style={{ height: '200px', objectFit: 'cover' }} />
      </Link>

      <Card.Body>
        <Link to={`/article/${article._id}`} style={{textDecoration: 'none'}}>
          <div style={{height: '65px', justifyContent:'center', display:'flex', color: '#D25380'}}><Card.Title as="div">
            <h4><strong>{article.name}</strong></h4>
          </Card.Title>
          </div>
        </Link>
      </Card.Body>
    </Card>
```

Рис. 3.1.1.4 – статті на головному екрані

ArticleScreen.js – сторінка статті; відповідає за розміщення заголовку, картинки, тексту на сторінці. Автоматично через ID публікації підтягує її назву, картинку, текст.

Окрім цього, присутня кнопка «Go back», яка повертає на головну сторінку. Частину коду наведено на рисунку 3.1.1.5.

```
function ArticleScreen({ match }) {
  const { id } = useParams();
  const article = articles.find((p) => p._id === id);
  return (
    <div>
      <Link to='/' className='btn btn-sm my-3 ' style={{ background: '#E08E6D', borderRadius: 25, color: 'white' }}><strong>Go back</strong></Link>
      <Row>
        <Col md={6} style={{ justifyContent: 'center', marginLeft: '350px', marginRight: '350px' }}>
          <ListGroup.Item style={{ color: '#D25380', textDecoration: 'underline #F6C391' }}><h1><strong>{article.name}</strong></h1></ListGroup.Item>
          <Image style={{ marginTop: '10px' }} src={article.image} alt={article.name} fluid />
          <ListGroup.Item className="lex justify-center items-center h-screen w-screen"
          style={{ justifyContent: 'center', display: 'flex', marginTop: '20px' }}>{article.description}</ListGroup.Item>
        </Col>
```

Рис. 3.1.1.5 – вигляд сторінки зі статтею

DictionaryScreen.js – реалізує власний словник авторизованого користувача (рис. 3.1.1.6). Головною задачею цієї сторінки є зберігання доданих слів, при натисканні на них – користувача направляє на сторінку з відповідною інформацією. Якщо натиснути на × – збережений термін видалиться з персонального словника та з колекції користувача в базі даних. Кнопка «Go back» виконує такі ж функції, як і в попередньому пункті.

```
return (
  <div>
    <Link to='/' className='btn btn-sm' style={{ background: '#E08E6D', borderRadius: 25, color: 'white' }}><strong>Go back</strong></Link>
    <div className="flex justify-center items-center h-screen w-screen" style={{ justifyContent: 'center', display: 'flex'}}>
      <h1 style={{ color: '#D25380', textDecoration:'underline #F6C391' }}><strong>Your Dictionary</strong></h1>
    </div>
    <ul style={{}}>
      {words.map((wordObj, index) => (
        <li key={index} style={{ listStyle: 'none' }} className="btn-group" role='group'>
          <div class="btn-group btn-group-lg" style={{ marginTop: '10px', marginBottom: '10px', marginLeft: '10px', marginRight: '10px' }}>
            <Button
              className="btn btn-dark"
              type="button"
              style={{ borderColor: '#E08E6D', backgroundColor: '#E08E6D', borderLeft: 0, borderTopLeftRadius: 10, borderBottomLeftRadius: 10 }}
              onClick={() => navigateToWordScreen(wordObj.word)}>
              {wordObj.word}
            </Button>

            {authUser && (

              <Button
                onClick={() => deleteWord(wordObj.id)}
                style={{ borderColor: '#F6C391', backgroundColor: '#F6C391', borderTopRightRadius: 10, borderBottomRightRadius: 10 }}
                className="btn btn-outline-light"
                type='button'>
                <label className='fas fa-close' />
              </Button>
```

Рис. 3.1.1.6 – персональний словник користувача

HomeScreen.js – головний екран веб-застосунку, на ньому відображається пошук слова та статті (рис. 3.1.1.7).

```
function HomeScreen() {
  return (
    <div >
      <Search />
      <Row>
        {articles.map(article => (
          <Col key={article._id} sm={16} md={6} lg={4} xl={4}>
            <Card.Title>
              <Article article={article} />
            </Card.Title>
          </Col>
        ))}
      </Row>
    </div>
```

Рис. 3.1.1.7 – головна сторінка

UserScreen.js – сторінка користувача. Якщо акаунт не авторизований, тоді відображається надпис «Sign In or Sign Up». Натиснувши на «Sign In», користувача направить до «Log In» (входу), «Sign Up» – скеровує на сторінку реєстрації. Якщо відвідувачі сайту авторизувались, відображатиметься підтвердження про це та з'явиться кнопка «Sign Out» (вийти). Кнопка «Go back» виконує такі ж функції, що й в пункті 5. Фрагмент коду наведено на рисунку 3.1.1.8.

```
function UserScreen() {
  const [authUser, setAuthUser] = useState(null);
  const navigate = useNavigate();
  useEffect(() => {
    const listen = onAuthStateChanged(auth, (user) => {
      if (user) {
        setAuthUser(user);
      } else {
        setAuthUser(null);
      }
    });

    return () => {
      listen();
    };
  }, []);
  const userSignOut = () => {
    signOut(auth)
      .then(() => {
        console.log("sign out successful");
        navigate('/signin')
      })
      .catch((error) => console.log(error));
  };
```

Рис. 3.1.1.8 – сторінка користувача

WordScreen.js – файл, що відображає результати пошуку слова та оформлення екрану (рис. 3.1.1.9). На цій сторінці можна переглянути:

- фонетику;
- прослухати вимову слова;
- частину мови;
- приклади використання слова;

- визначення;
- синоніми;
- антоніми.

Крім вищевказаної інформації, можна подивитись на асоціативну картинку, додати слово до персонального словника натиснувши кнопку «*Add to Dictionary*». Кнопка «*Go back*» повертає на головний екран.

```
<Link to='/' className='btn btn-sm' style={{ background: '#E08E6D', borderRadius: 25, color: 'white' }}><strong>Go back</strong></Link>
<div className="text-center text-muted"><h5>Word Definition</h5></div>
<div className="text-center" style={{ color: '#D25380', marginTop: '20px' }}> <h1> <strong>{word}</strong></h1> </div>

{authUser ?
  <div className="lex justify-center items-center h-screen w-screen" style={{ justifyContent: 'center', display: 'flex' }}>
    <Button className='button' onClick={addToFavorites} style={{ width: '200px', borderRadius: 25, background: '#F6C391', borderColor: '#F6C391', marginTop: '20p
  </div> : <></>}

{isLoading ? (
  <div className="text-center text-muted" style={{ marginTop: '40px' }}><p>Loading...</p></div>
) : (
  <>
    {image && (
      <div className="flex justify-center items-center h-screen w-screen" style={{ justifyContent: 'center', display: 'flex' }}>
        <img src={image} alt={word} style={{ maxWidth: '60%', height: 'auto' }} />
      </div>
    )}
    {phonetics.length > 0 && (
      <div>
        <h3 style={{ textDecoration: 'underline #F6C391 5px', marginTop: '40px' }}><strong>Phonetics:</strong></h3>
        {phonetics.map((phonetic, index) => (
          <div key={index}>
            <h4 style={{ textDecoration: 'underline #D25380' }}>Text: {phonetic.text}</h4>
            <h4><audio controls src={phonetic.audio} /></h4>
```

Рис. 3.1.1.9 – фрагмент коду сторінки з результатами пошуку слова

SignInScreen.jsx – вхід в акаунт (рис. 3.1.1.10). У цьому файлі формується інтерфейс вхідної сторінки, який складається з полів для введення електронної пошти, паролю, кнопки «Log In», яка скеровує на сторінку користувача. Посилання «*Don’t have an account yet? Register now*» направляє на сторінку реєстрації. Кнопка «*Go back*» повертає на головний екран.

```
return (
  <div><Link to='/' className='btn btn-sm my-3 ' style={{ background: '#E08E6D', borderRadius: 25, color: 'white' }}><strong>Go back</strong></Link>
    <div className="flex justify-center items-center h-screen w-screen" style={{ justifyContent: 'center', display: 'flex', color: '#D25380', marginTop: '65px' }}>
      <div className="card w-400 p-3 bg-white" style={{ borderRadius: 25 }}>
        <div className="flex flex-col">
          <div className="flex">
            <h1 className="text-2xl"><strong>Log in to your account</strong><i class="ri-login-circle-line"></i></h1>
          </div>

          <Form onSubmit={signIn} layout="vertical" className="mt-2">
            <div className="form-group">
              <label for="exampleInputEmail1" style={{ marginLeft: '13px' }}>Email address</label>
              <input
                style={{ borderRadius: 25 }}
                className="form-control"
                id="exampleInputEmail1"
                aria-describedby="emailHelp"
                type="email"
                placeholder='Enter your email'
                value={email}
                onChange={(e) => setEmail(e.target.value)}>
              </input>
              <small id="emailHelp" className="form-text text-muted" style={{ marginLeft: '13px' }}>We'll never share your email with anyone else.</small>
            </div>
```

Рис. 3.1.1.10 – фрагмент коду, що відповідає за екран входу

SignUpScreen.jsx – сторінка реєстрації, яка виконує майже такі ж функції, що й SignInScreen.jsx, але є деякі відмінності. На цьому екрані відсутнє посилання «*Don't have an account yet? Register now*», кнопка «Go back» повертає на сторінку входу.

App.js – файл, в якому прописані шляхи веб-застосунку. Фрагмент коду наведено на рисунку 3.1.1.14.

```
function App() {
  return (
    <Router>
      <Header />
      <main className="py-3">
        <Container>
          <Routes>
            <Route path='/' element={<HomeScreen />} />
            <Route path="/word" element={<WordScreen />} />
            <Route path="/article/:id" element={<ArticleScreen />} />
            <Route path="/signin" element={<SignInScreen />} />
            <Route path="/signup" element={<SignUpScreen />} />
            <Route path="/user" element={<UserScreen />} />
            <Route path="/dictionary" element={<DictionaryScreen />} />
          </Routes>
        </Container>
      </main>
      <Footer />
    </Router>

  );
}

export default App;
```

Рис. 3.1.1.14 – фрагмент коду App.js

article.js – файл, в якому вказана вся важлива інформація до статей та тексти, які приховано в . Кожна публікація має унікальний ID, назву, картинку та опис. Фрагмент коду наведено на рисунку 3.1.1.15.

```
const articles
  = [
    {
      '_id': '1',
      'name': 'How to speak English fluently - 10 key steps',
      'image': '/images/1.jpg',
      'description':
        <div>···
        </div>
    },
```

Рис. 3.1.1.15 – фрагмент коду article.js

3.2 Створення Backend частини проєкту

Реалізуємо Django частину навчального словника. Насамперед потрібно створити додаток, для цього виконаємо наступні команди:

1. **pip install virtualenv** – створення віртуального середовища;
2. **virtualenv myenv** – даємо назву середовищу;
3. **myenv\scripts\activate** – активація;
4. **pip install django** – встановлюємо Django;
5. **django-admin startproject backend** – створює проєкт, який називається backend;
6. **cd backend** – перехід до папки;
7. **python mahage.py startapp api** – створюємо додаток з назвою api.

Віртуальне середовище – це середовище, яке використовується Django для виконання програми. Рекомендується реалізовувати Django-додаток в окремому середовищі. Python надає інструмент virtualenv для створення ізольованого середовища Python.

3.2.1 Опис роботи файлів

Одним з головних файлів у частині Django є **views.py**, що знаходиться в додатку **api**. Цей код визначає функцію перегляду *word_definition*, яка обробляє запит на отримання визначення, синонімів, антонімів, частини мови, звукового відтворення, прикладів використання слова.

Наведена нижче частина коду імпортує необхідні модулі: *requests* для створення HTTP-запитів і *JsonResponse* з Django для повернення JSON-відповідей (рис. 3.2.1.1).

```
import requests
from django.http import JsonResponse
```

Рис. 3.2.1.1 – імпорт необхідних модулів

word_definition отримує в якості параметра об’єкт *request*, який представляє одержаний вхідний HTTP-запит. Функція отримує значення параметра *word* з GET-

параметрів запиту, якщо параметр відсутній, то за замовчуванням він дорівнює порожньому рядку (рис. 3.2.1.2).

```
def word_definition(request):
    word = request.GET.get('word', '')
```

Рис. 3.2.1.2 – функція *word_definition*

У коді використано три URL-адреси (рис. 3.2.1.3):

- url_definition – отримання визначення слова, його фонетики, частини мови, прикладів використання зі словника API;
- url_synonyms – отримання синонімів слова з API синонімів;
- url_antonyms – отримання антонімів слова з API антонімів.

```
url_definition = f'https://api.dictionaryapi.dev/api/v2/entries/en/{word}'
url_synonyms = f'https://api.datamuse.com/words?rel_syn={word}'
url_antonyms = f'https://api.datamuse.com/words?rel_ant={word}'
```

Рис. 3.2.1.3 – URL-адреси

HTTP GET-запити звертаються до трьох посилань за допомогою функції *requests.get* та зберігає відповіді у змінних *response_definitio*, response_synonyms і *response_antonyms* (рис. 3.2.1.4).

```
response_definition = requests.get(url_definition)
response_synonyms = requests.get(url_synonyms)
response_antonyms = requests.get(url_antonyms)
```

Рис. 3.2.1.4 – GET-запити до посилань

Наступна частина коду перевіряє, чи всі три відповіді API мають код статусу 200, тобто успішний запит (рис. 3.2.1.5).

```
if response_definition.status_code == 200 and response_synonyms.status_code == 200 and response_antonyms.status_code == 200:
    data_definition = response_definition.json()[0]
```

Рис. 3.2.1.5 – код доступу 200

Якщо відповіді успішні, наступна частина коду переходить до вилучення та обробки даних з відповідей API (рис. 3.2.1.6), аналізуючи JSON. Цей фрагмент файлу **views.py** витягує інформацію про фонетику (з аудіо відтворенням слова) (рис. 3.2.1.7) та визначення (включаючи частину мови і приклади речень) (рис. 3.2.1.8).

```
phonetics = []
for phonetic in data_definition['phonetics']:
    text = phonetic.get('text', '')
    audio = phonetic.get('audio', '')
    phonetics.append({
        'text': text,
        'audio': audio,
    })
```

Рис. 3.2.1.7 – phonetics

```
meanings = []
for meaning in data_definition['meanings']:
    part_of_speech = meaning['partOfSpeech']
    definitions = []
    for definition in meaning['definitions']:
        text = definition['definition']
        example = definition.get('example', '')
        definitions.append({
            'text': text,
            'example': example,
        })
    meanings.append({
        'part_of_speech': part_of_speech,
        'definitions': definitions
    })
```

Рис. 3.2.1.8 –meanings

Далі код витягує синоніми та антоніми з відповідей API Datamuse (рис. 3.2.1.9).

```
data_synonyms = response_synonyms.json()
data_antonyms = response_antonyms.json()

synonyms = [item['word'] for item in data_synonyms]
antonyms = [item['word'] for item in data_antonyms]
```

Рис. 3.2.1.9 – синоніми та антоніми

Якщо всі попередні дані витягнуті успішно, код повертає відповідь у форматі JSON за допомогою функції *JsonResponse*. Відповідь містить фонетику, значення, синоніми та антоніми (рис. 3.2.1.10).

```
return JsonResponse({'phonetics': phonetics,  'meanings': meanings, 'synonyms': synonyms, 'antonyms': antonyms })
```

Рис. 3.2.1.10 – повернення інформації у форматі JSON

Якщо будь-яка з відповідей API не спрацьовує або якщо слово не знайдено, код повертає відповідь у форматі JSON з повідомленням про помилку (рис. 3.2.1.11).

```
else:
    return JsonResponse({'error': 'Word not found'})
```

Рис. 3.2.1.11 – повідомлення про помилку

Можна підсумувати, що код з файлу **views.py** інтегрується із зовнішніми API для отримання визначення, синонімів та антонімів слова і повертає результати у вигляді JSON-відповіді.

Розглянемо детальніше файл **urls.py**, що знаходиться в папці **backend** (рис. 3.2.1.12).

```
from django.contrib import admin
from django.urls import include, path

urlpatterns = [
    path('admin/', admin.site.urls),
    path('api/', include("api.urls")),
]
```

Рис. 3.2.1.12 – код файлу urls.py

Ця частина відповідає за визначення шаблонів URL для проєкту. Код імпортує модуль *admin* з *django.contrib*, що надає функціональність для сайту адміністрування Django, та функції *include* (використовується для включення інших шаблонів URL з модуля *api.urls*) і *path* (використовується для визначення шаблонів URL) з *django.urls*.

Список *urlpatterns* містить шаблони URL для проєкту. ***Path('admin/', admin.site.urls)*** відображає URL з префіксом *admin/* на сайт адміністрування Django. Коли користувач відвідує URL з цим префіксом, він буде перенаправлений на сайт адміністрування, де він може керувати різними аспектами проєкту.

Другий шаблон ***path('api/', include(«api.urls»))*** зіставляє URL з префіксом *api/* з URL, визначеними в модулі *api.urls*. Це робиться за допомогою функції *include*, яка включає шаблони URL-адрес з модуля *api.urls* з префіксом *api/*. Це дозволяє модульну обробку URL-адрес, коли в модулі *api.urls* можуть бути визначені конкретні шаблони URL-адрес, пов'язані з функціональністю API.

Підсумовуючи, можна сказати, що код встановлює шаблони URL для проєкту. Він включає шаблони URL-адрес для сайту адміністрування Django з префіксом *admin/* та шаблони URL-адрес, визначені у модулі *api.urls* з префіксом *api/*. За необхідності можна додати додаткові шаблони URL-адрес.

У файлі setting.py, що знаходиться в папці backend, додамо стрічку коду, яка наведена на рисунку 3.2.1.13.

```
CORS_ALLOW_ALL_ORIGINS = True
```

Рис. 3.2.1.13 – код

У Django параметр **CORS_ALLOW_ALL_ORIGINS** використовується для керування поведінкою CORS (*механізм, який дозволяє веб-браузерам робити запити з одного домену до іншого домену*). Якщо встановлено значення ***True***, як у наведеному коді, це дозволяє запитам з будь-якого походження отримувати доступ до ресурсів у проєкті Django. Це означає, що будь-який веб-сайт або веб-додаток, незалежно від його домену, може робити перехресні запити до кінцевих точок проєкту Django.

3.3 Платформа розробки Firebase

Першим кроком буде створення авторизації/реєстрації. Для цього необхідно виконати попередні кроки:

- створимо новий проєкт у Firebase, де вкажемо спосіб реєстрації – електронна пошта та пароль;
- у терміналі VS Code введемо наступну команду: *npm install firebase*;
- ініціалізуємо Firebase у React-додатку, створивши новий файл – **firebase.js**, вміст файлу наведено на рисунку 3.3.1.

```
// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app";
import { getAuth } from "firebase/auth";
import{ getFirestore } from "firebase/firestore"
// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration
const firebaseConfig = {
  apiKey: "AIzaSyCl-d0ihJz1FxzUs2yWq5UPrQISHEC-ChU",
  authDomain: "edudictionary-8cdf2.firebaseapp.com",
  projectId: "edudictionary-8cdf2",
  storageBucket: "edudictionary-8cdf2.appspot.com",
  messagingSenderId: "1027465848853",
  appId: "1:1027465848853:web:e8605fc6e51655ed3ac61c"
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);

const auth = getAuth(app);
const db = getFirestore(app);
export {auth,db};
```

Рис. 3.3.1 – файл firebase.js

Далі до файлів SignInScreen.jsx та SignUpScreen.jsx додамо функції входу до акаунту (рис. 3.3.2) та реєстрації (рис. 3.3.3).

```jsx
import React, { useState } from 'react'
import { Button, Form } from 'react-bootstrap'
import { useNavigate, Link } from 'react-router-dom'
import { auth } from '../firebase'
import { signInWithEmailAndPassword } from 'firebase/auth'



const SignInScreen = () => {
  const [email, setEmail] = useState('')
  const [password, setPassword] = useState('')
  const navigate = useNavigate()

  const signIn = (e) => {
    e.preventDefault()
    signInWithEmailAndPassword(auth, email, password)
      .then((userCredential) => {···
      })
      .catch((error) => {···
      })
  }
```

Рис. 3.3.2 – вхід в акаунт

```jsx
import React, { useState } from 'react'
import { Button, Form } from 'react-bootstrap'
import { useNavigate, Link } from 'react-router-dom'
import { auth } from '../firebase'
import { createUserWithEmailAndPassword } from 'firebase/auth'

const SignUpScreen = () => {
  const [email, setEmail] = useState('')
  const [password, setPassword] = useState('')
  const navigate = useNavigate()


  const signUp = (e) => {
    e.preventDefault()
    createUserWithEmailAndPassword(auth, email, password)
      .then((userCredential) => {···
      })
      .catch((error) => {···
      )
  }
```

Рис. 3.3.3 – реєстрація

У Firebase кожному користувачу присвоюється ID, яке генерується автоматично. Вигляд акаунтів у базі даних Firebase наведено на рисунку 3.3.4.

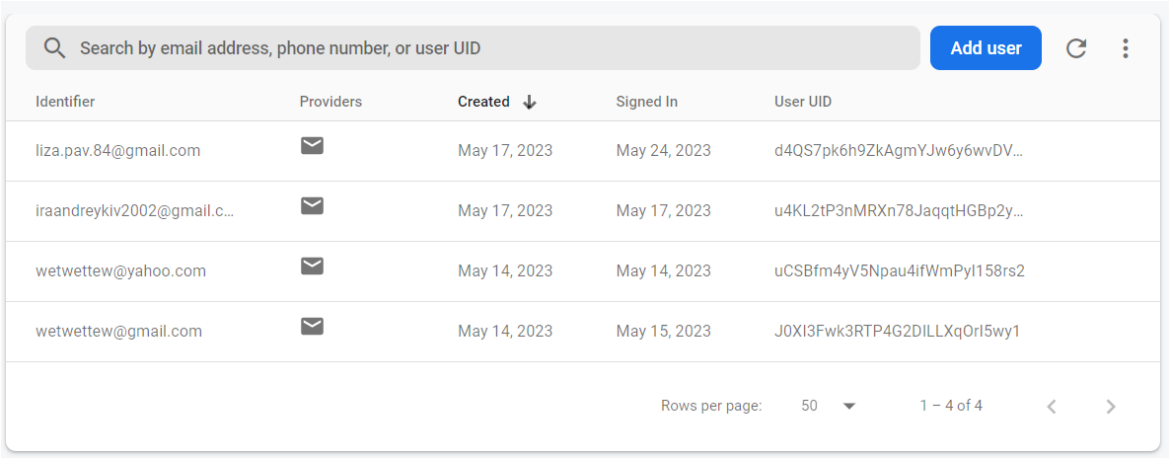

Рис. 3.3.4 – збережена інформація про користувача у БД

Реалізуємо можливість додавання слова в особистий словник для кожного авторизованого користувача. Для цього у Firestore створюється колекція з прототипом документа, де вказується: електронна пошта користувача, слово та ID до нього (генерується автоматично та випадково, щоб запобігти помилок). При додаванні в особистий словник формується новий об’єкт у базі даних Firestore, де вказана вся інформація (рис. 3.3.5).

```
dbWord: "love"
id: "d4QS7pk6h9ZkAgmYJw6y6wvDVam2"
useremail: "liza.pav.84@gmail.com"
```

Рис. 3.3.5 – вигляд доданого слова в особистий словник у базі даних

Окрім налаштувань в Firestore, у файлі WordScreen.js додаються функції **wordRef** (присвоює посилання на колекцію) та асинхронна **addToFavorites** (додає слово до власного словника користувача) (рис. 3.3.6).

```
const wordRef = collection(db, "words")
const addToFavorites = async () => (
  await addDoc(wordRef, {
    dbWord: word,
    useremail: authUser.email,
    id: authUser.uid,
  }).then(alert('word added'))
)
```

Рис. 3.3.6 – додавання слова до словника та БД

РОЗДІЛ 4. ТЕСТУВАННЯ ЗАСТОСУНКУ «EDUDICTIONARY»

4.1 Запуск застосунку навчального словника

Щоб запустити проєкт потрібно ввести в дію *backend* та *frontend* частини. Розпочнемо з серверу, у терміналі потрібно по черзі ввести наступні команди (рис. 4.1.1):

- *myenv\scripts\activate* – активація скриптів у віртуальному середовищі;
- *cd backend* – перехід до папки;
- *python mahage.py runserver* – запуск серверу.

Після успішного запуску, з'явиться адреса *backend* частини – *http://127.0.0.1:8000/*.

```
PS C:\Users\lizap\Desktop\english_dictionary_project - копія> myenv\scripts\activate
(myenv) PS C:\Users\lizap\Desktop\english_dictionary_project - копія> cd backend
(myenv) PS C:\Users\lizap\Desktop\english_dictionary_project - копія\backend> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
June 07, 2023 - 14:08:54
Django version 4.2, using settings 'backend.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Рис. 4.1.1 – запуск серверної частини

Перейдемо до клієнтської складової проєкту, у новому терміналі вводимо почергово такі команди:

- *cd frontend* – перехід до папки;
- *npm start* – запуск.

Після цього з'явиться повідомлення про успішну компіляцію (рис. 4.1.2).

```
Compiled successfully!

You can now view frontend in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.111.1:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

Рис. 4.1.2 – запуск клієнтської частини

4.2 Можливості застосунку

Розглянемо можливості навчального словника (рис. 4.2.1). У веб-застосунку головною ціллю є пошук слів. Тому після натискання на кнопку пошуку, користувача переводить на сторінку з результатом, де можна переглянути всю важливу інформацію про нього.

При авторизації в акаунт у користувачів з'являється можливість додавати слова до власного словника, а при потребі їх можна видалити. Окрім цього, на головній сторінці є статті на різні теми пов'язані з вивченням англійської мови.

Рис. 4.2.1 – можливості застосунку

Розглянемо детальніше кожен екран. Розпочнемо з домашньої сторінки, на ній присутнє верхнє меню, яке складається з назви, входу до акаунту та іконки користувача. Якщо користувач увійшов в акаунт, навігаційна панель змінюється, зникає «*Log In*» та додається «*Dictionary*» (рис. 4.2.2). Нижче можна побачити поле для пошуку слова та статті. Вигляд сторінки зображено на рисунку 4.2.3.

Рис. 4.2.2 – зміни у верхньому меню

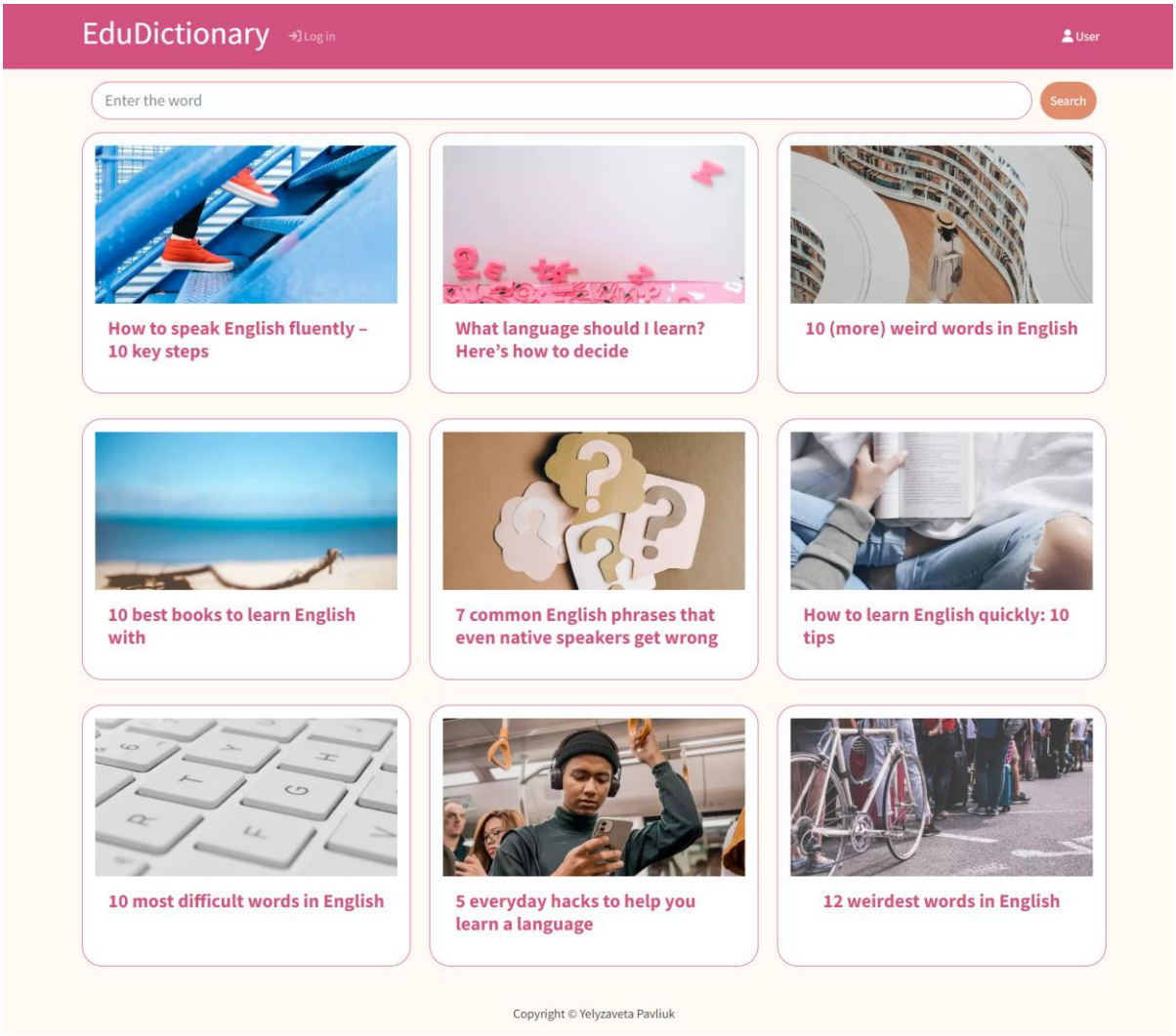

Рис. 4.2.3 – домашня сторінка застосунку

Після введення слова у поле пошуку та натискання на кнопку «Search», користувача направляє на сторінку з результатом (рис. 4.2.4). Тут знаходиться уся інформація про слово, а саме:

- картинка, яка пов'язана зі словом;
- фонетика;
- аудіо прослуховування;
- частина мови;
- визначення;
- приклади використання слова;
- синоніми;
- антоніми, якщо вони є до слова.

Варто підкреслити те, що на сторінці присутня кнопка «*Add to Dictionary*»,яка доступна лише авторизованим акаунтам. Натиснувши на неї, зявиться повідомлення про додавання слова до власного словника (рис. 4.2.5).

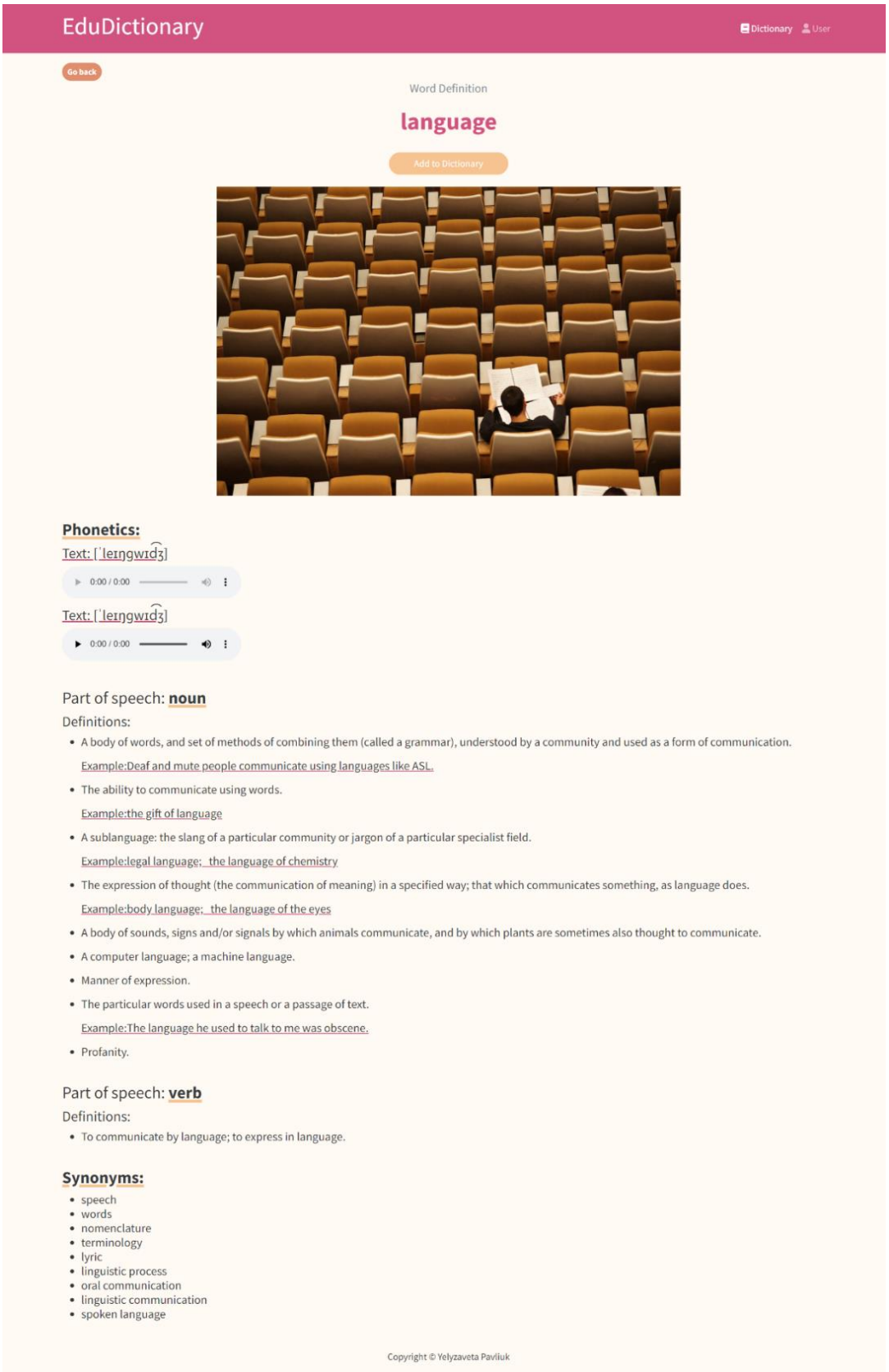

Рис. 4.2.4 – результат пошуку слова

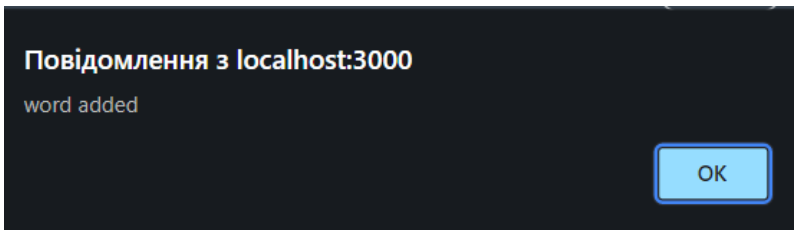

Рис. 4.2.5 – повідомлення про додавання слова в словник

Персональний словник містить в собі лише додані авторизованим користувачем слова, які при необхідності можна видалити. При натисканні на вираз, застосунок перенаправить на сторінку з результатами пошуку (рис. 4.2.6).

Рис. 4.2. – особистий словник

Сторінка користувача виглядає по-різному при авторизованому користувачі (рис. 4.5.7) та навпаки (рис. 4.2.8).

Рис. 4.2.7 – авторизований користувач

Рис. 4.2.8 – неавторизований користувач

Сторінки входу та реєстрації виглядають майже ідентично, обидві мають поля для вводу електронної пошти та паролю. Єдина відмінність між ними полягає в тому, що на сторінці авторизації є посилання «*Don't have an account yet? Register now*», яке направляє на сторінку реєстрація (рис. 4.2.9). Як тільки користувач увійшов в акаунт, з'явиться про це повідомлення (рис. 4.2.10). При реєстрації воно теж з'являється (рис. 4.2.11). Якщо користувач вже зареєстрований на таку електронну пошту, про це теж з'явиться попередження та акаунт не створиться (рис. 4.2.12).

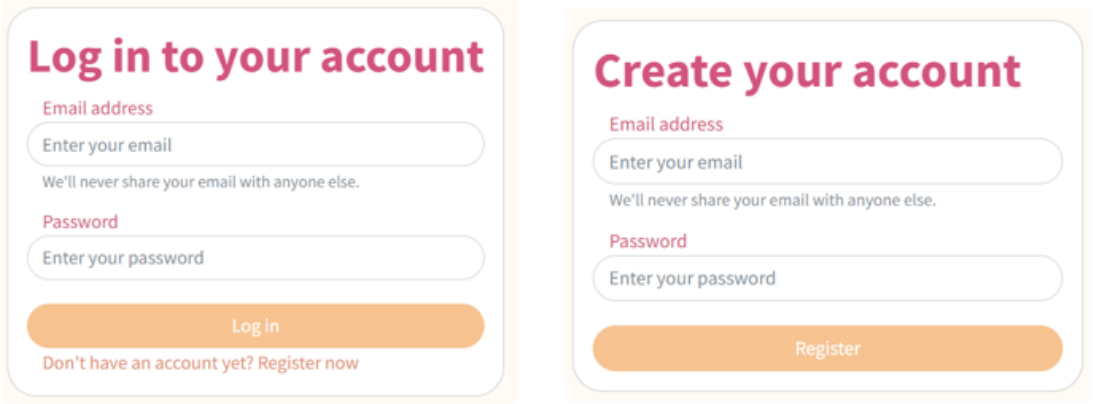

Рис. 4.2.9 – сторінки входу та реєстрації

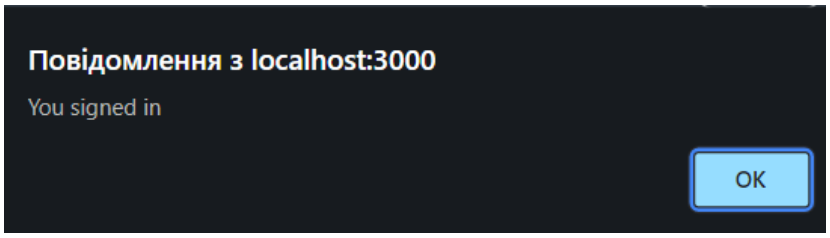

Рис. 4.2.10 – повідомлення про вхід

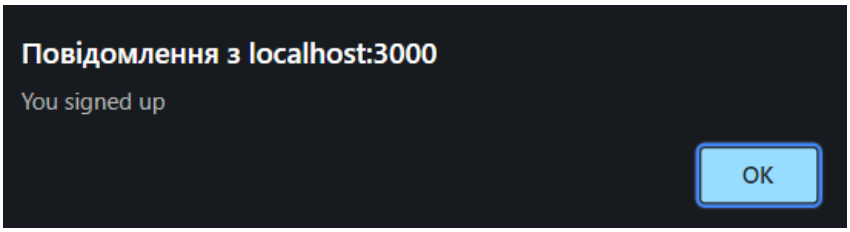

Рис. 4.2.11 – повідомлення про реєстрацію

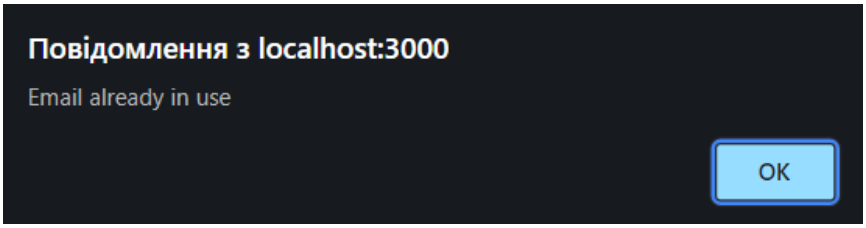

Рис. 4.2.12 – попередження про те, що така пошта вже зареєстрована

Сторінка статті складається з її назви, картинки та тексту. Вигляд екрану зображено на рисунку 4.2.13.

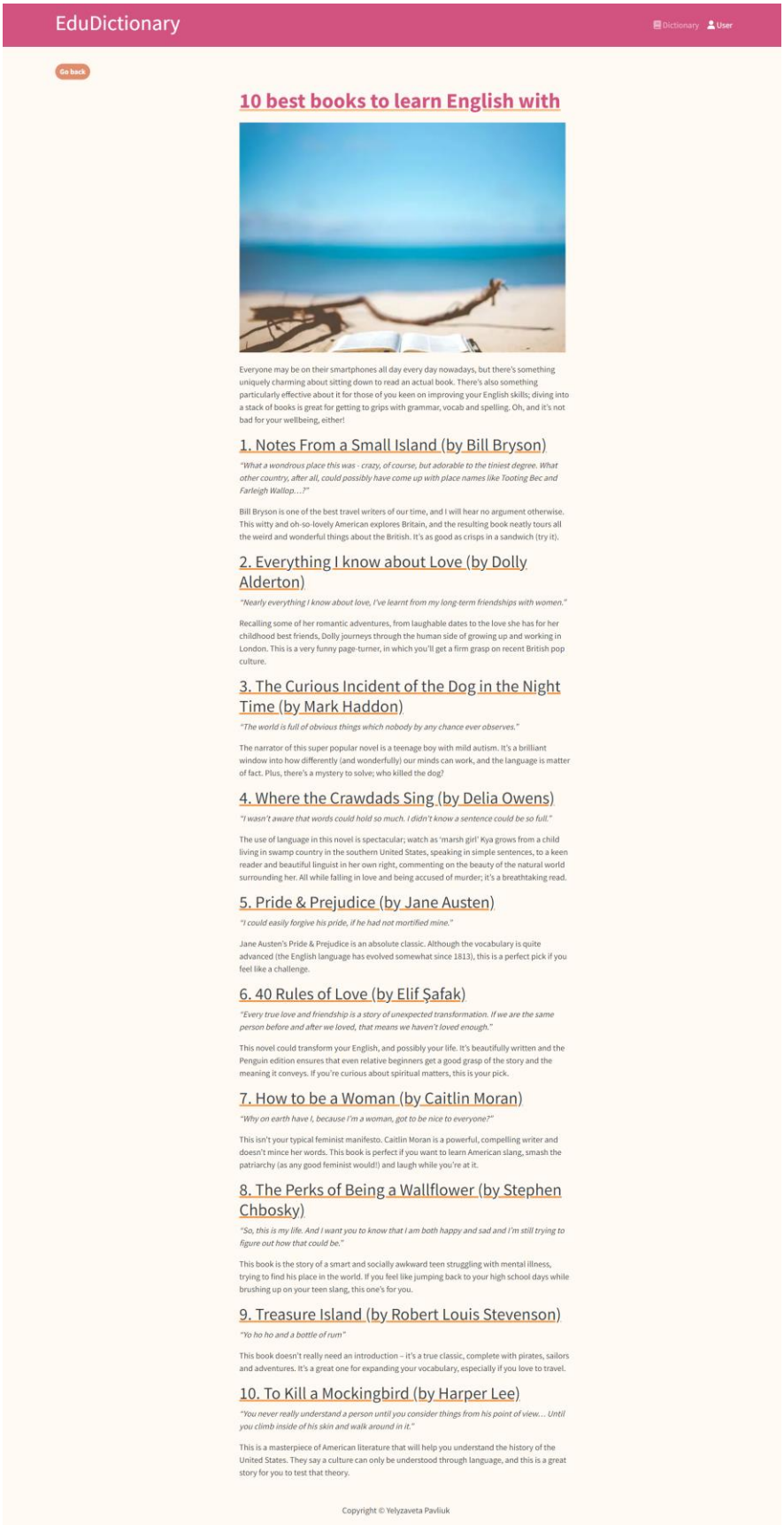

Рис. 4.2.13 – сторінка статті

ВИСНОВКИ

Розроблено веб-додаток «EduDictionary» з використанням веб-фреймворків Django та React. Додаток надає користувачам, які вивчають англійську мову, платформу для доступу до величезної колекції слів, визначень та пов'язану з ними інформацію.

Додаток використовує API, що дало можливість інтегрувати зовнішні джерела даних та отримати з них додаткову інформацію, синоніми, антоніми, приклади, аудіовідтворення, асоціативні картинки та супутні матеріали для кожного слова. Цей функціонал підвищив цінність та зручність використання додатку, зробивши його комплексним освітнім ресурсом для тих, хто вивчає мову.

Використання веб-фреймворку React дало можливість створити зручний, динамічний та інтерактивний користувацький інтерфейс, що полегшує вивчення нових слів.

Аутентифікація за допомогою Firebase забезпечила безпечний доступ користувачів та управління обліковими записами. Використання бази даних Firestore сприяло ефективному зберіганню інформації щодо приватних словників користувачів.

Аналіз існуючих навчальних онлайн словників виявив їхні недоліки: не зручний користувацький інтерфейс; відсутність асоціативних зображень; відсутність можливості авторизації користувачів та створення власних словників вивчених слів; відсутність додаткових навчальних матеріалів. У розробленому веб-додатку всі ці недоліки були враховані та виправленні.

Прогнозні припущення щодо розвитку навчального словника вказують на можливі напрямки майбутніх удосконалень, такі як елементи гейміфікації, збільшення статей, впровадження різних методів аутентифікації через соціальні мережі або Google, розширення спеціалізованої термінології з різних галузей і т.д.

Застосунок є значним внеском у сферу вивчення англійської мови, надаючи можливість тим, хто вивчає мову, покращити свої знання в простий, зрозумілий та ефективний спосіб.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Oxford Learner's Dictionaries | Find definitions, translations, and grammar explanations at Oxford Learner's Dictionaries. Oxford Learner's Dictionaries. URL: https://www.oxfordlearnersdictionaries.com/ (дата звернення: 01.05.2023).
2. Cambridge Dictionary | English Dictionary, Translations & Thesaurus. Cambridge Dictionary. URL: https://dictionary.cambridge.org/ (дата звернення: 01.05.2023).
3. Collins English Dictionary | Definitions, Translations, Example Sentences and Pronunciations. Collins Online Dictionary. URL: https://www.collinsdictionary.com/dictionary/english (дата звернення: 01.05.2023).
4. Longman Dictionary of Contemporary English | LDOCE. Longman Dictionary of Contemporary English | LDOCE. URL: https://www.ldoceonline.com/ (дата звернення: 01.05.2023).
5. Macmillan Dictionary | Free English Dictionary and Thesaurus. Macmillan Dictionary. URL: https://www.macmillandictionary.com/ (дата звернення: 01.05.2023).
6. Dictionary by Merriam-Webster: America's most-trusted online dictionary. Merriam-Webster. URL: https://www.merriam-webster.com/ (дата звернення: 01.05.2023).
7. Robert Kean JavaScript Programming for Beginners: JavaScript from A to Z. Independently Published, 2019. 77 с.
8. The React Handbook. flaviocopes.com. URL: https://flaviocopes.com/book/react/ (дата звернення: 02.05.2023).
9. What Is React & How Does It Actually Work?. Hostinger Tutorials. URL: https://www.hostinger.com/tutorials/what-is-react (дата звернення: 04.05.2023).
10. What is Python? Executive Summary. Python.org. URL: https://www.python.org/doc/essays/blurb/ (дата звернення: 11.05.2023).
11. Thomas J. Python Programming: The Fundamental Beginner's Guide to Learning Python. Independently Published, 2019. 110 с.
12. Richard L. Halterman Fundamentals of Python programing. Southern Adventist University, 2019. 669 с.
13. Introduction to Django. W3Schools Online Web Tutorials. URL: https://www.w3schools.com/django/django_intro.php (дата звернення: 02.05.2023).

14. Vincent W. S. Django for APIs: Build web APIs with Python & Django. Independently published, 2018. 190 с.
15. Firebase. Firebase. URL: https://firebase.google.com/ (дата звернення: 13.04.2023).
16. What is an Application Programming Interface (API)? | IBM. IBM - Deutschland | IBM. URL: https://www.ibm.com/topics/api (дата звернення: 04.05.2023).
17. Diwedi A. React Bootstrap - Scaler Topics. Scaler Topics. URL: https://www.scaler.com/topics/react/react-bootstrap/ (дата звернення: 30.04.2023).
18. React Bootstrap | React Bootstrap. React Bootstrap | React Bootstrap. URL: https://react-bootstrap.github.io/ (дата звернення: 29.03.2023).
19. Bootstrap. Bootstrap · The most popular HTML, CSS, and JS library in the world. URL: https://getbootstrap.com/ (дата звернення: 01.04.2023).
20. React Router vs. React Router DOM | Syncfusion Blogs. Syncfusion. URL: https://www.syncfusion.com/blogs/post/react-router-vs-react-router-dom.aspx (дата звернення: 31.05.2023).
21. Derks R. React Projects: Build 12 real-world applications from scratch using React, React Native, and React 360. Packt Publishing, 2019. 474 с.

ДОДАТОК А

1. views.py

```
import requests
from django.http import JsonResponse

def word_definition(request):
    word = request.GET.get('word', '')
    url_definition = f'https://api.dictionaryapi.dev/api/v2/entries/en/{word}'
    url_synonyms = f'https://api.datamuse.com/words?rel_syn={word}'
    url_antonyms = f'https://api.datamuse.com/words?rel_ant={word}'
    response_definition = requests.get(url_definition)
    response_synonyms = requests.get(url_synonyms)
    response_antonyms = requests.get(url_antonyms)
    if response_definition.status_code == 200 and response_synonyms.status_code == 200
and response_antonyms.status_code == 200:
        data_definition = response_definition.json()[0]
        phonetics = []
        for phonetic in data_definition['phonetics']:
            text = phonetic.get('text', '')
            audio = phonetic.get('audio', '')
            phonetics.append({
                'text': text,
                'audio': audio,
            })
        meanings = []
        for meaning in data_definition['meanings']:
            part_of_speech = meaning['partOfSpeech']
            definitions = []
            for definition in meaning['definitions']:
                text = definition['definition']
```

```
            example = definition.get('example', '')
            definitions.append({
                'text': text,
                'example': example,
            })
        meanings.append({
            'part_of_speech': part_of_speech,
            'definitions': definitions
        })
    data_synonyms = response_synonyms.json()
    data_antonyms = response_antonyms.json()
    synonyms = [item['word'] for item in data_synonyms]
    antonyms = [item['word'] for item in data_antonyms]
    return JsonResponse({'phonetics': phonetics, 'meanings': meanings, 'synonyms': synonyms, 'antonyms': antonyms })
  else:
    return JsonResponse({'error': 'Word not found'})
```

2. urls.py

```
from django.contrib import admin
from django.urls import include, path
urlpatterns = [
  path('admin/', admin.site.urls),
  path('api/', include("api.urls")),
]
```

ДОДАТОК Б

1. App.js

```
import React from 'react';
import { Container } from "react-bootstrap";
import { BrowserRouter as Router, Routes, Route } from "react-router-dom";
import Header from "./components/Header";
import Footer from "./components/Footer";
import HomeScreen from './screens/HomeScreen';
import WordScreen from "./screens/WordScreen";
import ArticleScreen from './screens/ArticleScreen'
import SignInScreen from './screens/SignInScreen'
import SignUpScreen from './screens/SignUpScreen'
import DictionaryScreen from './screens/DictionaryScreen'
import UserScreen from './screens/UserScreen'

function App() {
  return (
    <Router>
      <Header />
      <main className="py-3">
        <Container>
          <Routes>
            <Route path='/' element={<HomeScreen />} />
            <Route path="/word" element={<WordScreen />} />
            <Route path="/article/:id" element={<ArticleScreen />} />
            <Route path="/signin" element={<SignInScreen />} />
            <Route path="/signup" element={<SignUpScreen />} />
            <Route path="/user" element={<UserScreen />} />
            <Route path="/dictionary" element={<DictionaryScreen />} />
          </Routes>
```

```
        </Container>
      </main>
      <Footer />
    </Router>
  );
}
export default App;
```

2. Footer.js

```
import React from 'react'
import { Container, Row, Col } from 'react-bootstrap'
function Footer() {
  return (
        <footer>
          <Container>
            <Row>
              <Col className='text-center py-3'>Copyright &copy; Yelyzaveta Pavliuk</Col>
            </Row>
          </Container>
        </footer>
  )
}
export default Footer
```

3. Header.js

```
import { Navbar, Nav, Container } from 'react-bootstrap'
import { LinkContainer } from 'react-router-bootstrap'
import { onAuthStateChanged } from "firebase/auth";
import React, { useEffect, useState } from "react";
import { auth } from "../firebase";
```

```
function Header() {
  const [authUser, setAuthUser] = useState(null);
  useEffect(() => {
    const listen = onAuthStateChanged(auth, (user) => {
      if (user) {
        setAuthUser(user);
      } else {
        setAuthUser(null);
      }
    });
    return () => {
      listen();
    };
  }, []);
  return (
    <header style={{backgroundColor: '#D25380'}}>
    <Navbar variant="dark">
      <Container>
        <LinkContainer to='/'>
            <Navbar.Brand><h1>EduDictionary</h1></Navbar.Brand>
        </LinkContainer>

        <Navbar.Toggle aria-controls="basic-navbar-nav" />
            <Navbar.Collapse id="basic-navbar-nav">
              {!authUser?<Nav className="mr-auto">
              <LinkContainer to='/signin'>
                <Nav.Link><i className="fas fa-sign-in"></i> Log in</Nav.Link>
              </LinkContainer>
                </Nav>:<></>}
```

```
            <Nav className='ms-auto'>
            {authUser ?(
            <LinkContainer to='/dictionary'>
            <Nav.Link ><i className="fas fa-book"></i> Dictionary</Nav.Link>
          </LinkContainer>):<></>
}
            <LinkContainer to='/user'>
            <Nav.Link><i className="fas fa-user "></i> User</Nav.Link>
          </LinkContainer>
          </Nav>
        </Navbar.Collapse>
      </Container>
    </Navbar>
  </header>


  )
}

export default Header
```

4. Search.js

```
import React, { useState } from 'react'
import { Form, Container, Button, InputGroup } from 'react-bootstrap'
import { useNavigate } from 'react-router-dom'
function Search() {
  const [word, setWord] = useState('')
  const navigate = useNavigate()
  const handleWordChange = (event) => {
    setWord(event.target.value);
```

```
  };
  const navigateToWordScreen = () => {
    navigate('/word', { state: { word: word } });
  }
  return (
    <Container className="input-group pr">
      <Form.Control size="lg"
      aria-describedby="basic-addon2"
      placeholder="Enter the word"
      aria-label="Enter the word"
      onChange={handleWordChange}
      style={{borderColor: '#D25380', borderRadius: 50}}/>
      <Button
      variant="outline"
      id="button-addon2"
      className="btn btn-light"
      type="submit"
      style={{borderColor: '#E08E6D', backgroundColor: '#E08E6D', borderRadius: 50,
color:'white', marginLeft:'10px'}}
      onClick={navigateToWordScreen}>
        Search
      </Button>
    </Container>
  );
}
export default Search;
```