МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА WEB-РЕСУРСУ ІНТЕРАКТИВНОГО СЛОВНИКА З СУЧАСНОЇ РОЗМОВНОЇ АНГЛІЙСЬКОЇ МОВИ ЗАСОБАМИ JAVASCRIPT»**

Виконав: студент 4 курсу, групи 6.1220
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)
освітньої програми Комп'ютерні науки
(назва освітньої програми)
В. Г. Плюснін
(ініціали та прізвище)
Керівник доцент кафедри комп'ютерних наук, к.т.н. Решевська К.С.,
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент завідувач кафедри фундаментальної та прикладної математики, професор, д.т.н. Гребенюк С.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра комп'ютерних наук

Рівень вищої освіти бакалавр

Спеціальність 122 Комп'ютерні науки
(шифр і назва)

Освітня програма Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук,
д.т.н., доцент

_____ Шило Г. М.
(підпис)

" 25 " грудня 2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Плюсніну Валентину Григоровичу
(прізвище, ім'я та по-батькові)

1. Тема роботи Розробка web-ресурсу інтерактивного словника з сучасної розмовної англійської мови засобами JavaScript

керівник роботи Решевська Катерина Сергіївна., к.т.н., доцент
(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 21 » грудня 2023 року № 2180-с

2. Строк подання студентом роботи 05.06.2024

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Визначити основні вимоги до веб-ресурсу, включаючи функціональність словника та інтерфейс користувача
2. Розробка дизайну інтерфейсу користувача.
3. Написати код для веб-ресурсу, використовуючи Javascript та обране API.
4. Провести тестування веб-ресурсу для перевірки його функціональності та виявлення можливих помилок.

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

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 25.12.23

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	25.12.23	
2.	Збір вихідних даних.	13.03.24	
3.	Обробка методичних та теоретичних	21.04.24	
	джерел.		
4.	Розробка першого розділу.	10.05.24	
5.	Розробка другого розділу.	16.05.24	
6.	Розробка третього розділу	25.05.24	
7.	Оформлення та нормоконтроль	04.06.24	
	кваліфікаційної роботи.		
8.	Захист кваліфікаційної роботи.	23.06.24	

Студент _____ В.Г. Плюснін
(підпис) (ініціали та прізвище)

Керівник роботи _____ К.С. Решевська
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О.Г. Спиця
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка web-ресурсу інтерактивного словника розмовної англійської мови засобами Javascript»: 62 с., 35 рис., 7 джерел, 3 додатки.

API, JAVASCRIPT, ВЕБ-РЕСУРС, ІНТЕРАКТИВНИЙ СЛОВНИК, РОЗМОВНА АНГЛІЙСЬКА МОВА, РОЗРОБКА.

Об’єкт дослідження – розробка веб-ресурсу інтерактивного словника засобами Javascript.

Мета роботи – створити інтерактивний веб-ресурс для вивчення розмовної англійської мови, використовуючи мову програмування JavaScript та API для надання функціоналу словника.

Метод дослідження – аналіз вимог до роботи, проєктування користувацького інтерфейсу, розробка веб-ресурсу, тестування

У цій кваліфікаційній роботі було створено веб-ресурс для вивчення англійської мови у інтерактивному форматі, що має на меті полегшення процесу вивчення нових слів для користувачів. Веб-ресурс створений за допомогою мови програмування JavaScript та WordsAPI. Робота включає аналіз вимог, проектування користувацького інтерфейсу, розробку та тестування. Результатом є веб-ресурс, який є ефективним інструментом для вивчення англійської мови.

SUMMARY

Bachelor’s Qualifying Theses «Development of a web-resource of an interactive dictionary of spoken English using Javascript»: 62 p., 35 figures, 7 sources, 3 appendices.

API, Javascript, web resource, interactive dictionary, spoken English, development.

The object of the study is the development of an interactive dictionary web resource using Javascript.

The aim of the study is to create an interactive web resource for learning spoken English using the JavaScript programming language and WordsAPI to provide dictionary functionality.

The Methods of research are analysis of work requirements, user interface design, web resource development, testing

In this qualifying paper, a web resource for learning English in an interactive format was created to facilitate the process of learning new words for users. The web resource was created using the JavaScript programming language and API. The work includes requirements analysis, user interface design, development, and testing. The result is a web resource that is an effective tool for learning English.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ, ТЕРМІНІВ

API	Application Programing Interface: набір правил і специфікацій, що дозволяють різним програмним системам взаємодіяти між собою
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
Веб-ресурс	сукупність веб-сторінок, об’єднаних під одним доменним ім’ям.
JavaScript	мова програмування, яка використовується для створення динамічного контенту на веб-сторінках.

ВСТУП

У сучасному світі, де англійська мова стає все більш важливою для міжнародного спілкування, потреба в ефективних інструментах для вивчення мови зростає. Одним з таких інструментів є інтерактивні словники, які дозволяють користувачам вивчати нові слова та фрази в контексті, що сприяє кращому засвоєнню.

Web-ресурси інтерактивних словників мають багато переваг перед традиційними паперовими словниками. Вони більш зручні у використанні, оскільки їх можна використовувати на будь-якому пристрої з підключенням до Інтернету. Крім того, web-ресурси постійно оновлюються новою інформацією, що робить їх більш актуальними.

Розробка web-ресурсу інтерактивного словника розмовної англійської мови засобами Javascript є актуальною темою дослідження, оскільки дозволяє створити новий ефективний інструмент для вивчення англійської мови. Цей інструмент може бути використаний як для самостійного навчання, так і для додаткової роботи на уроках англійської мови.

Об’єкт дослідження – розробка веб-ресурсу інтерактивного словника засобами Javascript.

Предмет дослідження – процес розробки веб-ресурсу інтерактивного словника розмовної англійської мови.

Мета роботи – створити інтерактивний веб-ресурс для вивчення розмовної англійської мови, який буде сприяти вивченню та засвоєнню лексики, використовуючи мову програмування JavaScript та API для надання функціоналу словника

Завдання на кваліфікаційну роботу:

– дослідити та засвоїти практики front-end розробки мовою програмування JavaScript;

- ознайомитися з технологією API та принципами її використання у front-end розробці;
- визначити основні вимоги до веб-ресурсу, включаючи функціональність словника та інтерфейс користувача;
- розробка дизайну інтерфейсу користувача;
- написати код для веб-ресурсу, використовуючи Javascript та обране API;
- провести тестування веб-ресурсу для перевірки його функціональності та виявлення можливих помилок.

1 АКТУАЛЬНІСТЬ ВИВЧЕННЯ РОЗМОВНОЇ АНГЛІЙСЬКОЇ МОВИ ТА АНАЛІЗ ВЖЕ ІСНУЮЧИХ ІНТЕРАКТИВНИХ СЛОВНИКІВ

1.1 Актуальність та значимість вивчення розмовної англійської мови

В сучасному світі англійська мова стає все більш важливою, а знання розмовної англійської мови відкривають перед людиною безліч можливостей. Англійська мова використовується в різних сферах життя: у бізнесі, політиці, науці, освіті, туризмі, IT-сфері тощо. Вміння вільно спілкуватися англійською мовою дозволяє налагодити контакт з людьми з усього світу, отримати доступ до кращих освітніх та професійних ресурсів, подорожувати без проблем, бути в курсі світових подій та новин.

Вивчення розмовної англійської мови стає все більш актуальним для людей різного віку та професій. Це пов’язано з наступними чинниками:

- розвиток інформаційних технологій. Інтернет та соціальні мережі відкривають доступ до величезної кількості інформації англійською мовою;
- збільшення можливостей для подорожей. Люди все частіше їздять за кордон і знання англійської мови стають необхідністю для комфортного перебування в іншій країні;
- розвиток міжнародного бізнесу. Англійська мова є мовою бізнесу, знання англійської мови дає можливість успішно вести справи на міжнародному рівні.

1.1.1 Статистика та факти, що підтверджують актуальність вивчення англійської мови

За даними British Council, у 2020 році англійська мова була мовою найвищого попиту в світі.

Близько 1,5 мільярда людей у світі володіють англійською мовою, це число постійно зростає.

Англійська мова є офіційною мовою 53 країн світу.

Англійська мова є мовою бізнесу та торгівлі. Близько 80% міжнародного бізнесу ведеться англійською мовою.

Знання англійської мови дає людям більше можливостей для працевлаштування. Згідно з дослідженням Oxford Economics, люди, які володіють англійською мовою, заробляють на 10% більше, ніж ті, хто нею не володіє.

1.1.2 Переваги використання інтерактивних словників для вивчення розмовної англійської мови

Інтерактивні словники мають ряд переваг перед традиційними паперовими словниками. Інтерактивні словники дозволяють користувачам не лише знаходити значення слів, але й слухати їхню вимову, переглянути транскрипцію, бачити приклади вживання, а також проходити різноманітні тести для ефективнішого засвоєння вивченого матеріалу. Інтерактивні словники доступні онлайн, що робить їх зручними у використанні в будь-якому місці та в будь-який час. Їх постійно оновлюють новою інформацією, що робить їх більш актуальними, ніж традиційні паперові словники.

Деякі інтерактивні словники дозволяють користувачам створювати власні списки слів, додавати закладки та нотатки, що робить їх більш персоналізованими та зручними у використанні.

1.2 Теоретичні та методичні аспекти викладання розмовної англійської мови

1.2.1 Особливості лексики та граматики розмовної англійської мови

Лексика розмовної англійської мови відрізняється від лексики офіційної англійської мови тим, що вона є більш неформальною та містить багато ідіом, фразеологічних зворотів, сленгу та скорочень. Граматика розмовної англійської мови також має свої особливості. Наприклад, у розмовній мові частіше використовуються скорочені форми дієслів, контракції, інверсія, а також не завжди дотримуються всіх правил часових форм.

1.2.2 Сучасні методи та технології навчання розмовної англійської мови

Викладання розмовної англійської мови зазнало значних змін в останні роки. Замість традиційних методів постійного вивчення правил граматики та довгих списків слів, зараз акцент робиться на розвитку навичок спілкування в реальних ситуаціях. На мою думку, це дійсно дієва стратегія, адже таким чином людині буде легше адаптуватися в англомовному середовищі, а також вона швидше заговорить та позбудеться мовного бар’єру.

Сьогодні все частіше можна зустріти комунікативний підхід до вивчення мови. Цей підхід робить акцент на розвитку навичок спілкування англійською мовою в реальних ситуаціях. Навчання будується на діалогах, рольових іграх, групових роботах та інших інтерактивних методах. Також під час викладання англійської часто практикується занурення в мовне середовище. Це метод, що передбачає постійне використання англійської мови на уроці. Вчителі можуть використовувати англійську мову для пояснення матеріалу, проведення ігор та організації роботи в класі.

При викладанні англійської мови в нагоді також стають автентичні матеріали – це тексти, аудіо та відеозаписи, які використовуються в реальному житті. Використання автентичних матеріалів допомагає учням звикнути до реальної англійської мови та розвиває їхні навички сприйняття мови на слух.

Ну і авжеж сучасні технології, такі як комп'ютерні програми, мобільні додатки, інтерактивні дошки та онлайн-ресурси, можуть зробити навчання розмовної англійської мови більш цікавим та ефективним.

1.3 Популярні інтерактивні словники

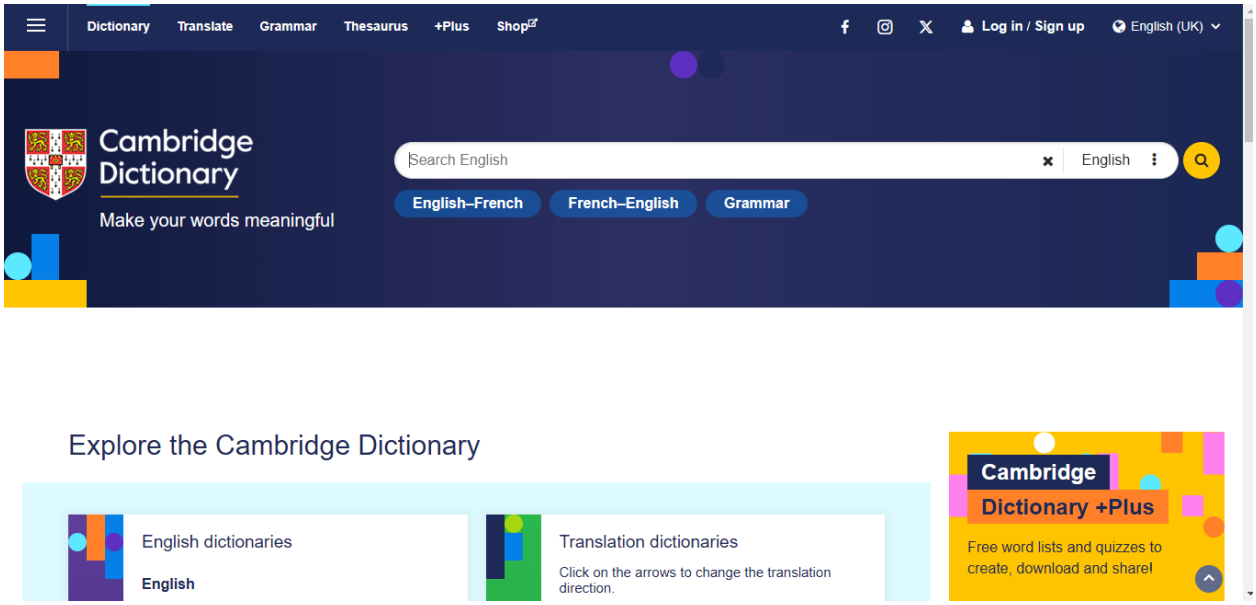

Рисунок 1.1 – Головна сторінка інтерактивного словнику Cambridge Dictionary Online

Cambridge Dictionary Online – це словник (див. рис. 1.1) що вражає багатством лексики та детальними поясненнями, роблячи його незамінним помічником для допитливих учнів. Різноманітні навчальні інструменти роблять процес вивчення мови цікавим та ефективним. Проте, деякі функції доступні лише за плату, а інтерфейс може здатися складним для початківців.

Сильні сторони:

– велика база слів та фраз;
– детальні пояснення та приклади;
– можна прослухати аудіо з вимовою слів;
– різноманітні навчальні інструменти, такі як тести та ігри.

Слабкі сторони:

– деякі функції доступні лише за платною підпискою;
– може бути складним для початківців;
– прослухати аудіо з вимовою можна не в усіх словах.

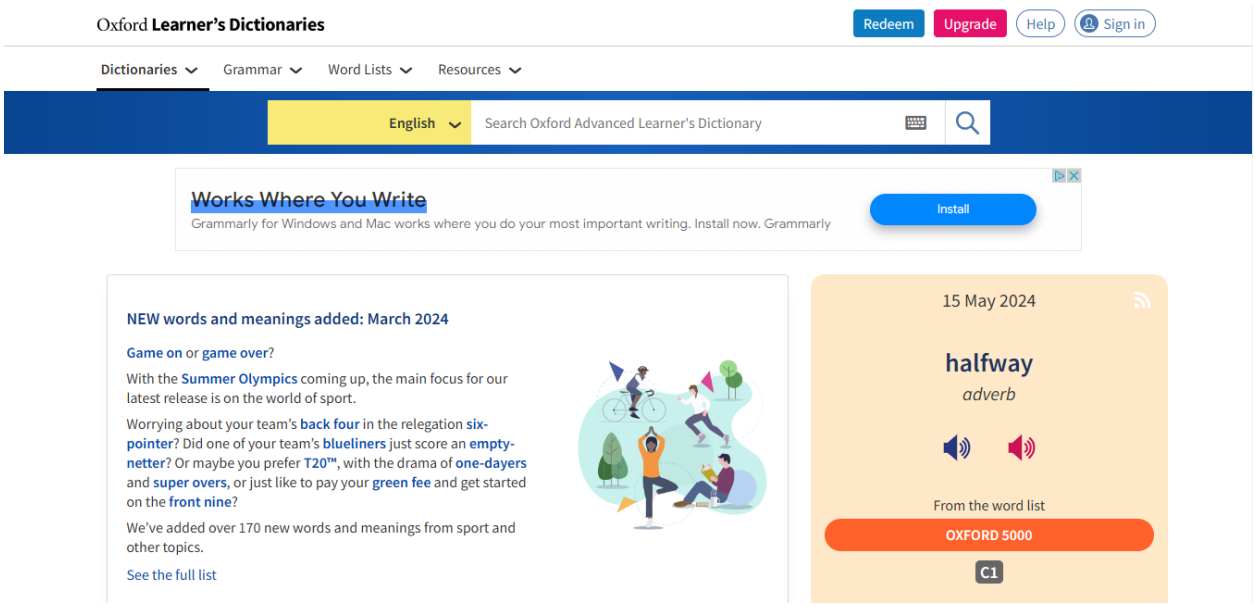

Рисунок 1.2 – Головна сторінка інтерактивного словнику Oxford Learner's Dictionaries Online

Oxford Learner's Dictionaries Online – це словник (див. рис. 1.2) що відрізняється чіткими та лаконічними поясненнями, що робить його ідеальним для тих, хто прагне швидко й зрозуміло засвоїти нову інформацію. Додаткові граматичні пояснення та навчальні інструменти роблять його цінним ресурсом для вивчення англійської мови. Проте, база слів трохи менша, ніж у Cambridge Dictionary Online, а деякі функції також доступні лише за підпискою.

Сильні сторони:

– чіткі та лаконічні пояснення;

– приклади, які відповідають рівню володіння мовою;
– граматичні пояснення;
– різноманітні навчальні інструменти, такі як тести та ігри;
– доступний на різних платформах.

Слабкі сторони:

– не така велика база слів, як у Cambridge Dictionary Online;
– деякі функції доступні лише за платною підпискою;
– не всі слова мають функцію для прослуховування вимови.

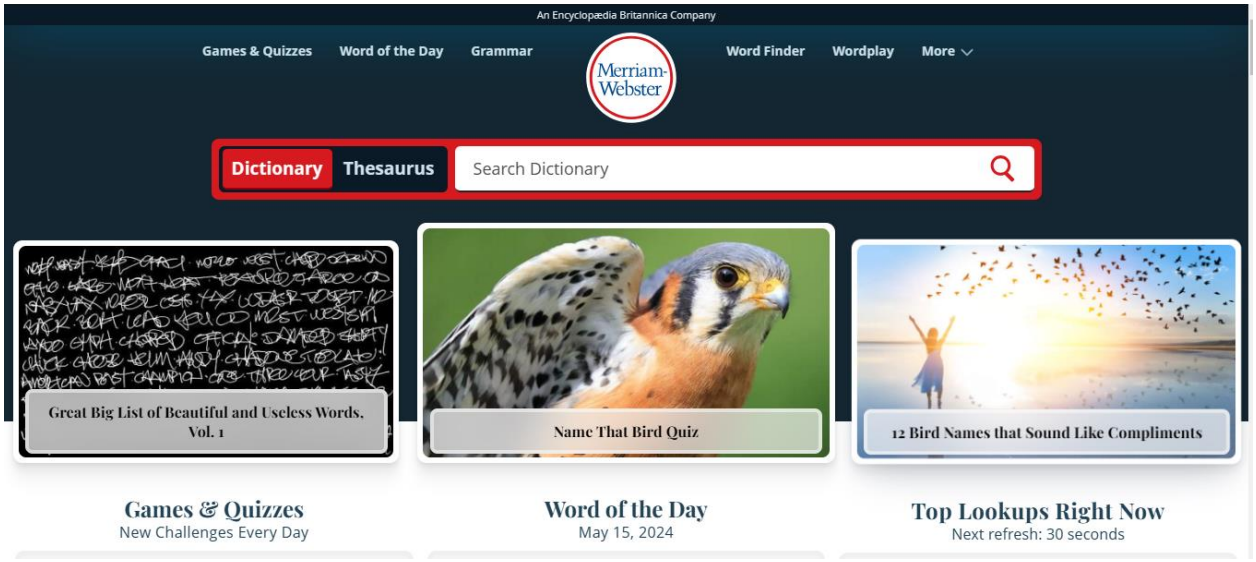

Рисунок 1.3 – Головна сторінка інтерактивного словнику Merriam-Webster Learner’s Dictionary

Merriam-Webster Learner’s Dictionary (див. рис. 1.3) фокусується на американському варіанті англійської мови, пропонуючи прості та зрозумілі пояснення з прикладами. Інформація про синоніми та антоніми допомагає розширити словниковий запас, а навчальні інструменти роблять процес вивчення мови динамічним. Проте, база слів не така велика, як у двох попередніх словниках,

Сильні сторони:

– прості та зрозумілі пояснення;

– приклади, які відповідають американському варіанту англійської мови;
– інформація про синоніми та антоніми;
– різноманітні навчальні інструменти, такі як тести та ігри;
– доступний на різних платформах.

Слабкі сторони:

– не така велика база слів, як у Cambridge Dictionary Online або Oxford Learner's Dictionaries Online;
– не всі слова мають функцію для прослуховування вимови.

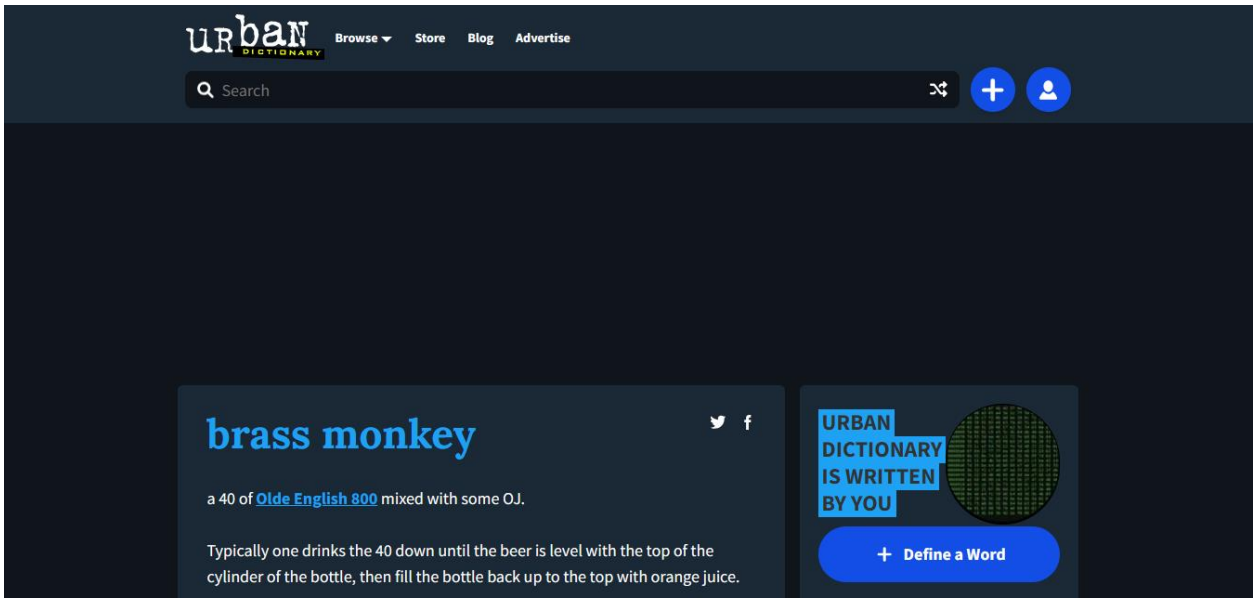

Рисунок 1.4 – Головна сторінка інтерактивного словнику Urban Dictionary

Urban Dictionary – це скарбниця сленгу, розмовних виразів та ідіом, що робить його незамінним для тих, хто хоче зануритися в атмосферу живої англійської мови (див. рис. 1.4). Можливість додавати власні пояснення та неформальний стиль роблять його цікавим та оригінальним. Проте, важливо пам'ятати, що пояснення не завжди точні, а контент може бути образливим.

Сильні сторони:

– містить сленг, розмовні вирази та ідіоми;
– пояснення написані користувачами словника;

– можливість додавати власні визначення;
– цікавий та неформальний стиль.

Слабкі сторони:

– пояснення не завжди точні або надійні;
– може містити образливий контент;
– не підходить для вивчення формальної англійської мови.

2 ТЕХНІЧНІ АСПЕКТИ РОЗРОБКИ ВЕБ-РЕСУРСУ ІНТЕРАКТИВНОГО СЛОВНИКА

2.1 Основи веб-розробки

Веб-розробка охоплює створення динамічних та інтерактивних веб-ресурсів, які забезпечують користувачам зручний доступ до інформації та функціоналу через інтернет. Основні принципи веб-розробки включають зручність користування, продуктивність та безпеку. Зручність користування означає, що інтерфейс повинен бути інтуїтивно зрозумілим і легким у навігації. Продуктивність передбачає високу швидкість завантаження та реагування веб-ресурсу. Безпека ж означає, що веб-ресурс повинен бути захищеним від різних атак, таких як XSS, CSRF та SQL-ін’єкції.

Основними елементами веб-розробки є HTML, CSS та JavaScript. HTML (HyperText Markup Language) використовується для створення структури веб-сторінок, CSS (Cascading Style Sheets) – для стилізації веб-сторінок, а JavaScript додає інтерактивності та динамічної поведінки. Ці мови разом утворюють основи фронтенд-розробки, які дозволяють створювати функціональні та привабливі веб-ресурси.

Сучасна фронтенд-розробка включає використання фреймворків та бібліотек, які значно спрощують процес розробки. Основні підходи включають компонентний підхід, односторінкові додатки (SPA) та реактивне програмування. Компонентний підхід передбачає побудову веб-додатків з незалежних блоків функціональності, що полегшує їх підтримку та розширення. Односторінкові додатки завантажуються один раз, а потім диманічно оновлюються без перезавантаження сторінки, що забезпечує швидшу та більш плавну взаємодію з користувачем. Реактивне програмування використовує бібліотеки, такі як RxJS, для обробки асинхронних подій та даних. Популярні фреймворки, такі як React, Angular та Vue.js, забезпечують

зручний спосіб створення компонентів та керування станом додатка, що сприяє покращенню продуктивності та підтримки коду.

2.2 Фронт-енд технології для створення інтерактивного веб-ресурсу

HTML (HyperText Markup Language) є основною мовою для створення структури веб-сторінок. Вона дозволяє визначати різні елементи сторінки, такі як заголовки, параграфи, посилання та зображення. HTML5, остання версія стандарту, додала нові елементи та атрибути, що дозволяють створювати більш семантичні та інтерактивні веб-сторінки. CSS (Cascading Style Sheets) використовується для стилізації веб-сторінок, дозволяючи налаштовувати кольори, шрифти, розташування елементів та багато іншого. CSS3 додала нові можливості, такі як анімації, трансформації та гнучкі макети, що дозволяють створювати більш складні та адаптивні дизайни.

JavaScript є мовою програмування, яка додає інтерактивності та динамічну поведінку до веб-сторінок. JavaScript дозволяє реагувати на дії користувача, змінювати вміст сторінки без її перезавантаження та взаємодіяти з веб-серверами. Розширення JavaScript включають ECMAScript (ES6 та пізніші версії), що додають нові синтаксичні можливості та покращують продуктивність.

React, створений компанією Facebook, є бібліотекою для створення користувацьких інтерфейсів. Він дозволяє створювати компоненти, які можуть повторно використовуватись та легко керувати станом додатка. React використовує віртуальний DOM, що дозволяє швидко оновлювати інтерфейс при зміні даних. Vue.js є прогресивним фреймворком для створення інтерфейсів користувача, який має низький поріг входу та потужні інструменти для створення складних додатків. Vue.js також використовує компонентний підхід та має вбудовані директиви для реактивного оновлення інтерфейсу.

API (Application Programming Interface) – набір правил та протоколів для взаємодії з зовнішніми системами. У фронтенд-розробці API використовуються для отримання та відправки даних до серверів, що дозволяє створювати динамічні додатки, які можуть взаємодіяти з різними сервісами та базами даних. Основні типи API включають REST (Representational State Transfer) та GraphQL. REST API використовує HTTP методи для взаємодії з ресурсами, є простим у використанні та підходить для CRUD-операцій (створення, читання, оновлення, видалення). GraphQL є мовою запитів для API, яка дозволяє клієнтам точно визначати, які дані їм потрібні, що зменшує обсяг переданих даних та покращує продуктивність додатків.

Безпека API є важливим аспектом веб-розробки. Практики безпечної роботи з API включають аутентифікацію та авторизацію, шифрування даних, обмеження доступу та захист від атак. Аутентифікація та авторизація використовують протоколи OAuth або JWT (JSON Web Tokens) для забезпечення безпеки. Шифрування даних забезпечується використанням HTTPS для передачі даних між клієнтом та сервером. Обмеження доступу включає використання механізмів контролю доступу, таких як обмеження за IP адресами або ролі та дозволи. Захист від атак включає використання методів захисту від CSRF (Cross-Site Request Forgery) та XSS (Cross-Site Scripting).

2.3 Методи та принципи проєктування користувацького інтерфейсу

Юзабіліті та UX/UI-дизайн є критичними аспектами розробки веб-додатків. Основні принципи юзабіліті включають простоту, зрозумілість та зворотній зв’язок. Простота означає, що інтерфейс повинен бути простим та інтуїтивно зрозумілим, щоб користувачі могли легко виконувати свої завдання. Зрозумілість передбачає, що елементи інтерфейсу повинні бути зрозумілими, з чіткими назвами та інструкціями. Зворотній зв’язок забезпечує користувачам інформацію про реакцію системи на їхні дії.

UX (User Experience) дизайн зосереджується на створенні позитивного досвіду користувача під час використання додатка. Це включає аналіз потреб користувачів, створення прототипів та тестування з користувачами. UX дизайнери працюють над тим, щоб забезпечити зручність, ефективність та задоволення від використання додатка. UI (User Interface) дизайн зосереджується на візуальному аспекті інтерфейсу, створюючи естетично привабливі інтерфейси, які відображають бренд та стиль додатка, а також забезпечують адаптивність інтерфейсу до різних пристроїв та розмірів екрану.

Оцінка зручності користування включає різні методи, такі як юзабіліті-тести, опитування користувачів, аналіз поведінки користувачів та A/B тестування. Юзабіліті-тести проводяться, коли користувачі виконують завдання з використанням додатка, а розробники спостерігають за їхньою поведінкою та виявляють проблеми. Опитування дозволяють отримати зворотний зв’язок від користувачів щодо їхнього досвіду використання додатка. Аналіз поведінки користувачів використовує інструменти аналітики для відстежування їхньої поведінки на веб-сторінках. A/B тестування порівнює дві версії інтерфейсу, показуючи їх різним групам користувачів, щоб визначити, яка версія краще задовольняє їхні потреби.

2.4 Використання WordsAPI для інтеграції даних

Для виконання кваліфікаційної роботи бакалавра я обрав WordsAPI, за допомогою якого буде реалізовано функціонал інтерактивного словника. WordsAPI – це потужний інструмент для роботи зі словниковими даними в інтерактивних веб-додатках. Він надає програмний інтерфейс для доступу до різних типів лінгвістичної інформації, таких як визначення, синоніми, антоніми, приклади вживання, частини мови та багато іншого. Використання WordsAPI значно спрощує процес отримання та обробки словникових даних,

що робить його ідеальним варіантом для розробки веб-ресурсу інтерактивного словника за темою.

Основні можливості WordsAPI включають пошук слів, отримання їхніх визначень та прикладів вживання, а також знаходження синонімів та антонімів. API підтримує різні типи запитів, що дозволяє отримувати як загальну інформацію про слово, так і більш детальну лінгвістичну інформацію. Використання цього API забезпечує високу якість та актуальність даних, що є важливим для створення надійних та точних словників.

Інтеграція WordsAPI у веб-додаток здійснюється за допомогою стандартних HTTP-запитів. При розробці можна використати методи JavaScript для відправки запитів до API та обробки отриманих даних. Це включає використання бібліотек, таких як Fetch API.

2.5 Інтерактивні функції у веб-додатках

Інтерактивні елементи у веб-додатках створюються за допомогою JavaScript та бібліотек, таких як jQuery. Основні техніки включають обробку подій, динамічне оновлення контенту та використання бібліотек. Обробка подій дозволяє реагувати на дії користувачів, такі як кліки, наведення миші та введення тексту. Динамічне оновлення контенту використовує AJAX (Asynchronous JavaScript and XML) для оновлення вмісту сторінки без її повного перезавантаження. Це дозволяє створювати інтерактивні елементи, такі як фільтри, сортування та пошук, які миттєво реагують на дії користувачів.

Анімації додають динамічності та візуальної привабливості до веб-додатків. Основні техніки включають CSS3 анімації та переходи, JavaScript анімації та динамічне оновлення контенту. CSS3 дозволяє створювати складні анімації та переходи за допомогою ключових кадрів та трансформацій. JavaScript використовується для створення складних анімацій, які не можуть

бути реалізовані за допомогою CSS. Динамічне оновлення контенту використовує AJAX для оновлення вмісту сторінки без її повного перезавантаження.

3 РОЗРОБКА ВЕБ-РЕСУРСУ

3.1 Проєктування користувацького інтерфейсу

3.1.1 Аналіз вимог та визначення функціоналу

Перед початком проєктування інтерфейсу інтерактивного словника було проведено детальний аналіз вимог до функціоналу. Цей процес включав вивчення схожих існуючих додатків, аналіз літературних джерел з розробки користувацьких інтерфейсів, а також врахування сучасних тенденцій в дизайні.

Основною цільовою аудиторією словника будуть студенти, викладачі, дослідники, а також ті, хто активно вивчає або планує вивчати англійську мову. На основі цього, були сформовані ключові вимоги до функціоналу, які відповідають потребам різних категорій користувачів.

По-перше, користувачам потрібен швидкий та зручний доступ до інформації про слова. Це означає, що інтерфейс має бути інтуїтивно зрозумілим, з мінімальною кількістю кроків для отримання результату.

По-друге, для користувачів буде важливим отримувати різноманітні види інформації про слова. Тож сайт має містити можливість пошуку значень, синонімів, антонімів, прикладів використання, рим та інших функцій які будуть описані у наступних підрозділах.

По-третє, інформація, що надається словником, повинна бути точною та оновленою. щоб користувачі могли покладатися на неї під час навчання або роботи.

3.1.2 Структура інтерфейсу

Після аналізу вже існуючих інтерактивних словників, я вирішив зробити простий інтерфейс, адже на багатьох веб-ресурсах, які я відвідав під час аналізу, доволі часто можна було зустріти дуже багато зайвих елементів. Наявність таких елементів може збентежити або заплутати користувача під час першого відвідування веб-ресурсу. Саме тому, було вирішено створити максимально дружній інтерфейс, у якому не буде зайвих елементів і користувачі могли легко, якісно та швидко отримати інформацію якою вони цікавляться.

Проєктування користувацького інтерфейсу для свого веб-ресурсу, для більшого розуміння та облегшення роботи у подальшому, я вирішив почати зі створення діаграми варіантів використання. Діаграма варіантів використання – це діаграма у якій показано усі можливі варіанти використання функціональної системи. Її створення значно покращить розуміння структури, а також полегшить реалізацію функціональну веб-ресурсу.

Користувач зможе зробити наступне:

- ввести слово;
- обрати яку саме інформацію про введене слово він хоче дізнатись;
- підтвердити введене слово та обрану функцію для отримання інформації;
- отримати інформацію про слово.

На основі визначених вище можливостей користувача, я побудував діаграму варіантів використання (див. рис. 3.1).

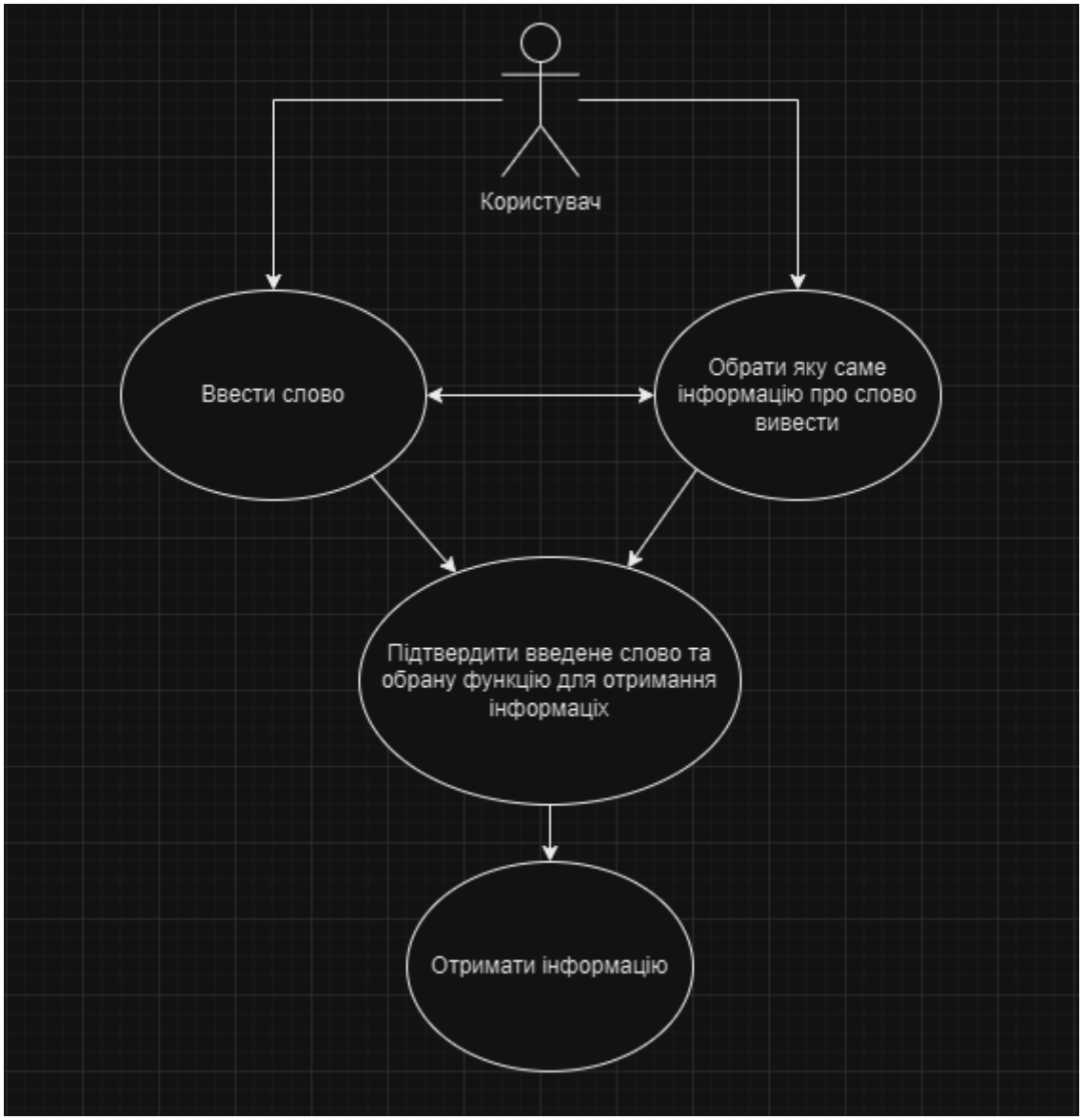

Рисунок 3.1 – Діаграма варіантів використання

Після створення діаграми варіантів використання, функціонал користувача став більш зрозумілим, тому можна перейти до планування самого інтерфейсу. Веб-ресурс має містити якийсь текст з привітанням для користувачів, а також текст, у якому буде знаходитися коротка інформація, яка допоможе користувачу з’ясувати мету веб-ресурсу та що на нього очікує. Одразу після текстової інформації планується основне поле для введення слова, у яке користувач вводитиме слово для якого бажає отримати інформацію. Після введення слова користувач переходить до меню вибору, яке містить різні варіанти функцій за допомогою яких користувач отримуватиме

результат, що містить інформацію, яка його цікавить. Після введення слова, ознайомлення з варіантами наявних функцій та вибору однієї з них. Користувач повинен натиснути на спеціальну кнопку для підтвердження свого вибору. Після натискання на кнопку, за допомогою реалізованого програмного коду, нижче користувач отримає усю необхідну інформацію про введене слово з обраної категорії.

Наступним етапом роботи над користувацьким інтерфейсом буде створення вайрфрейму для веб-ресурсу. Вайрфрейм(wireframe) – це дизайн низької точності, який показує структурну схему сторінок. По суті є макетом перед розробкою повноцінного дизайну сторінки.

Створений вайрфрейм веб-ресурсу інтерактивного словника (див. рис. 3.2)

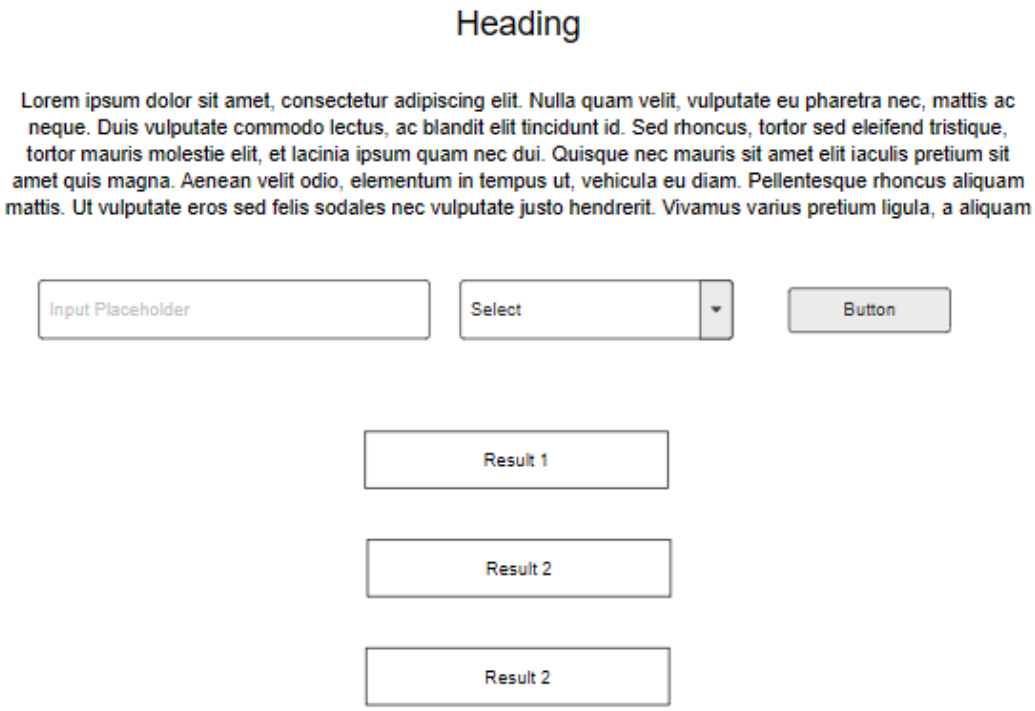

Рисунок 3.2 – Вайрфрейм веб-ресурсу інтерактивного словника

На рисунку 3.2 можемо побачити створений вайрфрейм для веб-ресурсу. На ньому маємо усі необхідні елементи для створення дружнього та зрозумілого інтерфейсу користувача. Елемент «Heading» відповідає за основний заголовок, у ньому буде розміщений текст з привітанням користувачів, які завітали на веб-ресурс. Далі, одразу ж під текстом з привітанням, бачимо абзац тексту, в ньому буде коротко описано, що це за сайт, що саме користувач може тут отримати, як цим скористуватися, тощо. Нижче розташовані основні елементи сторінки, за допомогою яких і буде здійснюватися отримання та надання необхідної інформації. Можемо бачити поле для введення слова, через яке, власне, буде здійснюватися отримання слова від користувача. Праворуч від нього розташоване select меню, призначенням якого є надання користувачу вибору різних функцій, які можуть стати йому в нагоді. Далі йде останній елемент у рядку – кнопка. За допомогою цієї кнопки буде здійснюватися підтвердження слова та вибору функції, після чого буде отримано вхідні дані та почато процес отримання і виведення вихідних даних, у вигляді відповіді від API з наданням запрошеної користувачем інформації. Нижче бачимо три елементи, що показують результати запиту, якщо під час обробки запиту користувача не виникло ніяких помилок.

3.1.3 Принципи дизайну інтерфейсу

Проєктування інтерфейсу інтерактивного словника базувалося на сучасних принципах UX/UI дизайну, щоб забезпечити зручність та естетичну привабливість веб-ресурсу. Основні принципи, які використовувалися під час розробки:

- мінімалізм;
- інтуїтивність;
- відсутність зайвих елементів;

- доступність;
- легкість використання.

Інтерфейс був створений таким чином, щоб уникати надмірних елементів і зосередитися на основних функціях. Це допомагає користувачам не відволікатися та швидко знаходити потрібну інформацію. Всі елементи інтерфейсу розташовані таким чином, щоб користувачі легко розуміли їх призначення та не плуталися під час використання веб-ресурсу. Весь інтерфейс дотримується єдиного стилю візуального оформлення. Це включає використання однакових шрифтів, кольорів та стилів.

Інтерфейс забезпечує швидкий відгук на дії користувача, що підвищує ефективність використання словника. Наприклад, при натисканні на кнопку пошуку одразу починається обробка запиту і виведення результатів.

3.2 Розробка дизайну за допомогою Figma

Після створення діаграми варіантів та вайрфрейму для словника, наступним важливим кроком у створенні дизайну буде перехід у Figma.

Figma – це потужний інструмент для створення та проєктування інтерфейсів, який відзначається своєю зручністю та широкими можливостями для співпраці. Як ресурс, Figma надає дизайнерам можливість створювати інтерактивні прототипи, проводити тестування та отримувати зворотний зв’язок у режимі реального часу. Figma має інтуїтивний інтерфейс і великий набір інструментів, які дозволяють швидко створювати візуально привабливі та функціональні макети, експериментувати з різними дизайнами та легко впроваджувати зміни.

3.2.1 Вибір кольорової схеми та шрифтів

Вибір кольорової схеми та шрифтів є важливим етапом у процесі розробки дизайну, оскільки ці елементи визначають візуальну привабливість та загальний стиль додатка. Для інтерактивного словника було вирішено використовувати сучасну, чисту та професійну кольорову гаму, яка б забезпечувала комфортне використання та приємне сприйняття інтерфейсу. На задньому фоні словника буде застосований градієнт з двох кольорів: червоний у нижньому лівому куті та фіолетовий у верхньому правому куті. Ці кольори плавно перетікають з одного в інший, створюючи враження глибини та динамічності. Така кольорова схема не лише привертає увагу, але й надає інтерфейсу сучасного та стильного вигляду.

Акцентні кольори, такі як світло-сірий, були використані для виділення важливих елементів, таких як: поле введення, меню вибору та самі результати. Це дозволяє користувачам швидко орієнтуватися в інтерфейсі та швидко знаходити необхідну інформацію. Ці елементи будуть мати прозорість аби краще вписуватися у дизайн.

Щодо вибору шрифтів, було прийнято рішення використовувати прості та читабельні шрифти, які добре виглядають на екранах різних розмірів. Основним шрифтом був обраний «Open Sans», який поєднує сучасний вигляд з високою читабельністю. Цей шрифт забезпечує комфортне читання тексту та добре поєднується з обраною кольоровою схемою.

Також було враховано використання різних ваг шрифтів для створення ієрархії та виділення важливих елементів тексту. Наприклад, заголовок з привітанням для користувачів має більш жирний шрифт, тоді як основний текст, текст у полях, кнопках, меню вибору та результатах має стандартну вагу. Це допоможе користувачам швидко орієнтуватися в текстовій інформації та зосереджувати увагу на ключових елементах.

3.2.2 Створення дизайну

У попередньому підрозділі, я обрав кольори які будуть послуговуватися заднім фоном сторінки. Тож відкривши figma, я створив новий проєкт та додав градієнт з кольорів, описаний у попередньому підрозділі (див. рис. 3.3).

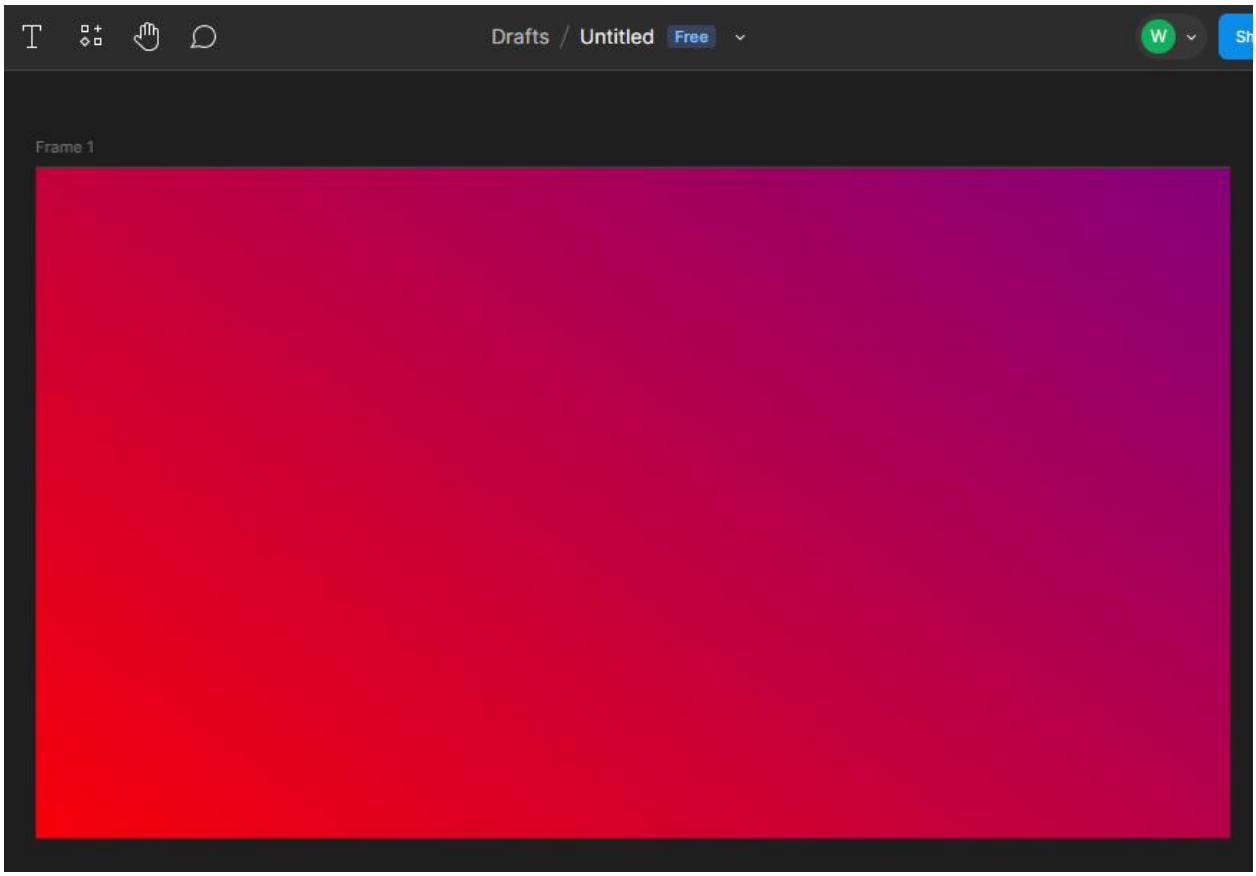

Рисунок 3.3 – Доданий градієнт

Наступним елементом, орієнтуючись на раніше створений вайфрейм, буде текст. Додаю заголовок у якому буде текст, що вітає користувачів на сайті. Задаю йому значення розміру 48, вагу шрифту виставляю SemiBold, ну і звісно ж, сам шрифт – Open Sans.

Слідом за текстом, що вітає користувачів, додаю текст, який коротко пояснює функціонал сайту та знайомить з ним користувача. Цьому тексту задаю значення розміру 25, вагу шрифту залишаю Regular, шрифт теж Open

Sans. Обидва тексти матимуть білий колір, а також вирівняні по центру (див. рис. 3.4).

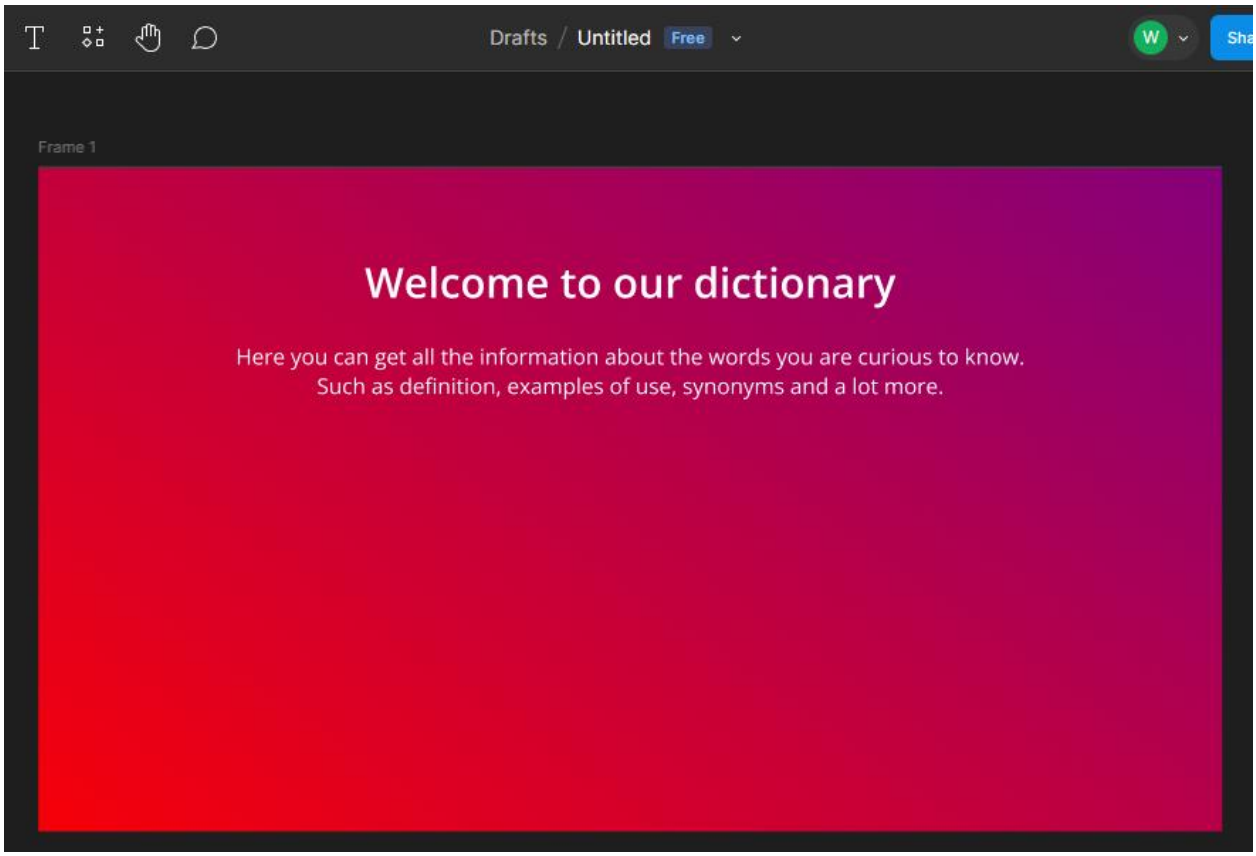

Рисунок 3.4 – Доданий текст

Далі мають бути розташовані основні елементи, а саме: поле для введення слова користувачем, меню для вибору потрібної функції та кнопка підтвердження введення та вибору. Почнемо з перших двох елементів, поля для вводу та меню з вибором функцій. Я додав два прямокутника, яким одразу змінив розмір та розташував відповідно до запланованої схеми дизайну, розробленого у вайрфреймі. Змінив колір прямокутників на світло-сірий та налаштував прозорість, що підкреслить їх та зробить дизайн більш виразним. Також змінив радіус кутів з 3 до 30, що перетворило кути прямокутників з гострих на згладжені та закруглені. До кожного з прямокутників додав відповідний текст, який показує, що це за меню. Цей текст також буде присутній, але трохи змінений на вже готовому сайті словника.

Додаю ще один прямокутник, який цього разу буде кнопкою. Змінюю радіус кутів, як це було з полем для вводу та меню вибору, з 3 до 30. Роблю кнопку трохи меншою за попередні два елементи, а також задаю їй помаранчевий колір та додаю текст (див. рис. 3.5).

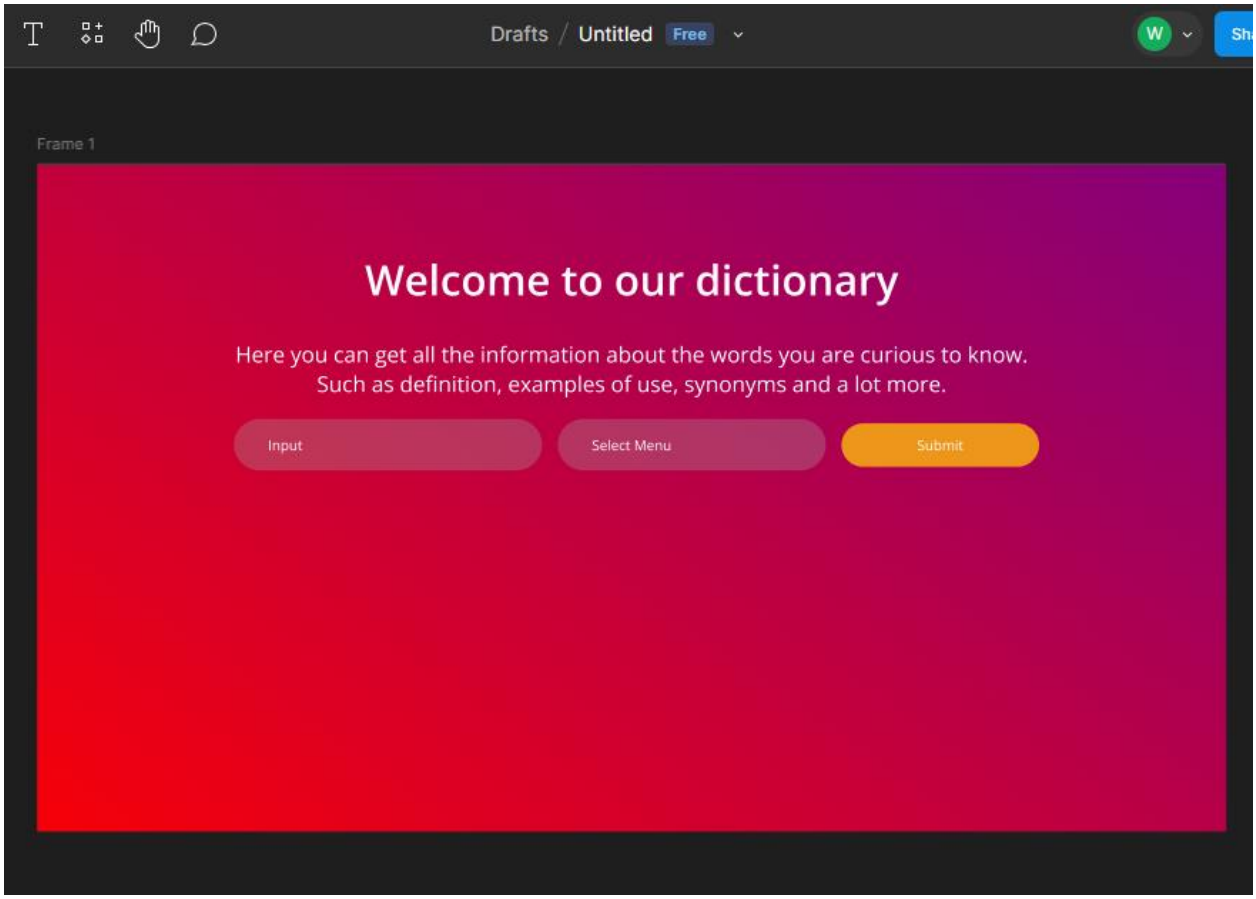

Рисунок 3.5 – Додані поле для введення, меню вибору функцій та кнопка

Залишилося додати тільки результати, котрі виводитимуться після виконання користувачем усіх необхідних кроків. Результати за стилем будуть як і поле для введення, тому я додав прямокутник, знову змінив радіус кутів, колір, прозорість, розмір та додав текст. Після чого скопіював це і додав ще два рази, зробивши таким чином 3 результати, що виводяться (див. рис. 3.6).

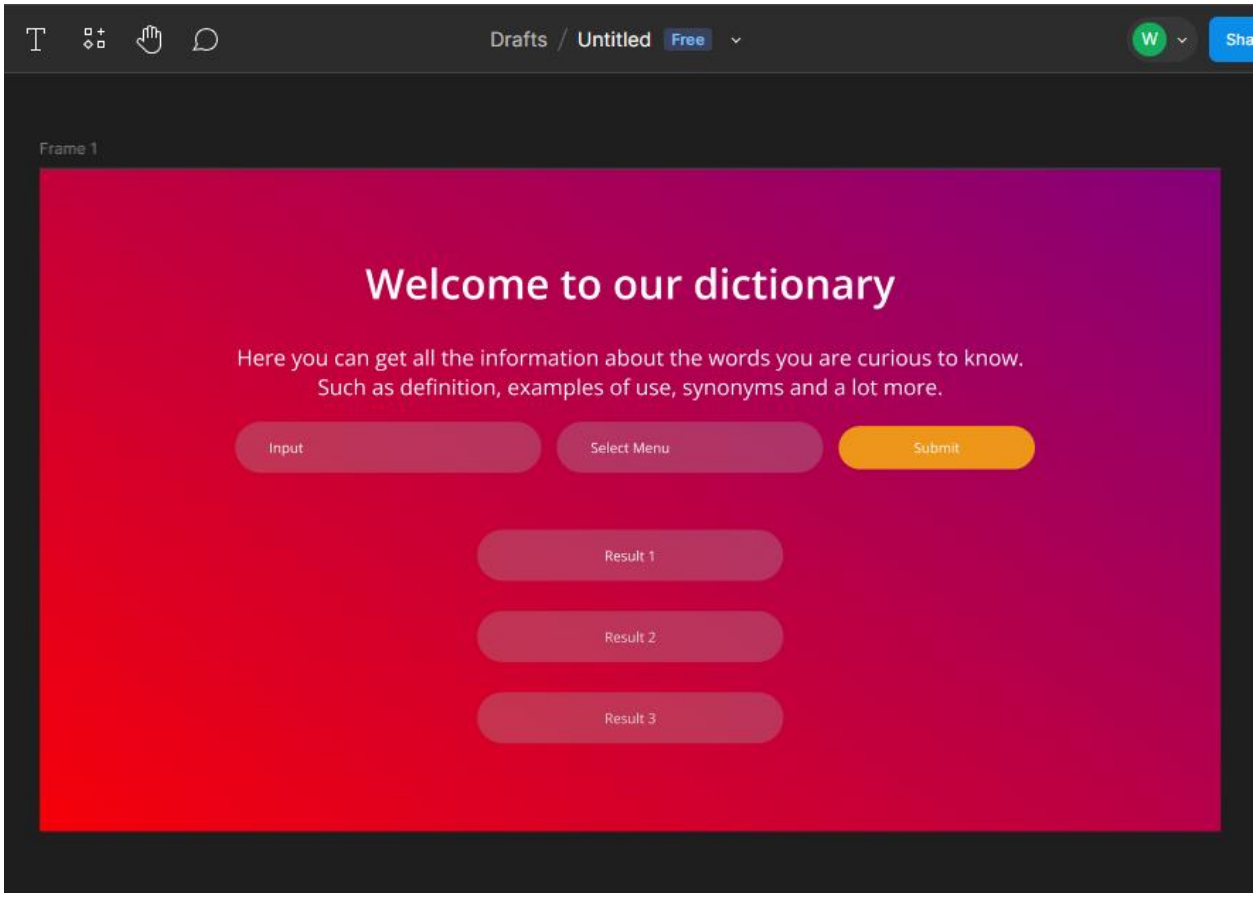

Рисунок 3.6 – Готовий дизайн

Дизайн завершено, тепер можна переходити до розробки веб-ресурсу словника.

3.3 Розробка веб-ресурсу інтерактивного словника

Перед початком розробки я перейшов на сайт RapidApi, щоб оформити підписку на WordsAPI, яке буде використовуватися під час розробки словника. Це API має тариф Freemium, що означає, що у API є можливість використання безкоштовно, але з певними обмеженнями. Оформивши підписку, я отримав усі необхідні дані, такі як унікальний API ключ, який використовується для підключення API у коді.

Створивши нову папку, спеціально, для самого веб-ресурсу та його розробки, я організував базову структуру веб-проєкту. Створив файл

Index.html, у якому буде зберігатися html код сторінки. Далі створив папки css та js, у яких зберігатимуться файл зі стилями style.css та файл з JavaScript кодом main.js.

Структура проєкту виглядає наступним чином (див. рис. 3.7):

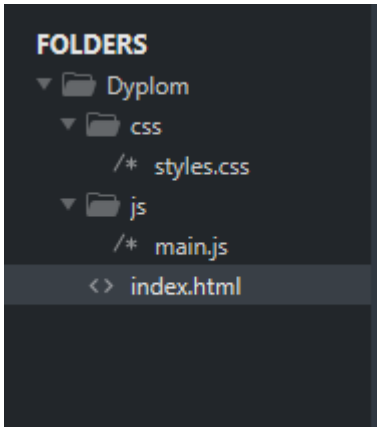

Рисунок 3.7 – Структура проєкту

3.3.1 Верстка веб-ресурсу

Почав розробку з верстки, одразу перейшов у html файл, де написав базову розмітку html файлу, підключив CSS та JavaScript файли (див. рис. 3.8).

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Dictionary Web-site</title>
    <link rel="stylesheet" type="text/css" href="css\styles.css">
</head>
<body>
    <script src="js/main.js"></script>
</body>
</html>
```

Рисунок 3.8 – Базова розмітка html файлу та підключення CSS і JavaScript

Далі відкрив тег header у якому, додав заголовок та текст за допомогою тегів h1 та p. У них вписав текст з макету, розробленого у Figma, у попередніх розділах (див. рис. 3.9). Після цього перейшов у CSS файл, щоб додати сторінці стилі. Для тегу body задаю градієнт з кольорів, відповідно до макету Figma. Далі задав стилі заголовку та тексту, задавши розміри, шрифти та колір. Вирівняв елементи по центру (див. рис. 3.10).

```
10      <header>
11          <h1>Welcome to our dictionary</h1>
12
13          <div>
14              <p>Here you can get all the information about the words you are curious to know.
15              <br>Such as definition, examples of use, synonyms and a lot more.</p>
16          </div>
17
18      </header>
```

Рисунок 3.9 – Доданий текст

```
1   body {
2     place-items: center;
3     min-height: 100vh;
4     background: linear-gradient(35deg, red, purple);
5   }
6
7   h1 {
8     font-family: 'Open Sans', sans-serif;
9     color: #fff;
10    font-size: 50px;
11    text-align: center;
12    padding-top: 2%;
13  }
14
15  p {
16    font-family: 'Open Sans', sans-serif;
17    color: #fff;
18    font-size: 25px;
19    text-align: center;
20  }
21
```

Рисунок 3.10 – Додані стилі

Далі я відкрив тег section, у якому почав створювати поле для введення за допомогою тегу input, меню вибору функції для обробки тексту за

допомогою теґу select та кнопку для відправки даних. У select меню додав три опції для вибору, інші додам під час написання коду на JavaScript. Наприкінці додав ul список, який буде виводити результати запитів користувачів (див. рис. 3.11). Для створених елементів додав стилі (див. рис. 3.12–3.15).

```
20      <section>
21          <form id="dictionary-form">
22              <input type="text" id="word-input" placeholder="Enter a word" required>
23              <select id="info-type" required>
24                  <option value="definitions">Definitions</option>
25                  <option value="synonyms">Synonyms</option>
26                  <option value="antonyms">Antonyms</option>
27              </select>
28              <button class="submit-button" type="submit">Submit</button>
29          </form>
30          <ul id="results"></ul>
31      </section>
```

Рисунок 3.11 – Додані у html файл основні елементи функціоналу

```
26  input {
27      appearance: none;
28      border: 0;
29      outline: 0;
30    font-size: 15px;
31    font-family: 'Open Sans', sans-serif;
32      width: 20rem;
33      padding: 1rem 4rem 1rem 1rem;
34      color: white;
35      background: no-repeat right 0.8em center / 1.4em,
36        linear-gradient(to left, var(--arrow-bg) 0em, var(--select-bg) 0em);
37      border-radius: 0.25em;
38      box-shadow: 0 0 1em 0 rgba(0, 0, 0, 0.2);
39      cursor: pointer;
40      border-radius: 100px;
41  }
42
43  input:focus, input:active {
44      outline: none;
45      color: white;
46      background: no-repeat right 0.8em center / 1.4em,
47          linear-gradient(to left, var(--arrow-bg) 0em, var(--select-bg) 0em);
48  }
49
50  ::placeholder {
51    font-size: 15px;
52    font-family: 'Open Sans', sans-serif;
53      color: white;
54      opacity: 1;
55  }
```

Рисунок 3.12 – Стилі для поля введення

```css
:root {
  --arrow-bg: rgba(255, 255, 255, 0.3);
  --arrow-icon: url(https://upload.wikimedia.org/wikipedia/commons/9/9d/Caret_down_font_awesome_whitevariation.svg);
  --option-bg: white;
  --select-bg: rgba(255, 255, 255, 0.2);
}

select {
  appearance: none;
  border: 0;
  outline: 0;
  font-size: 15px;
  font-family: 'Open Sans', sans-serif;
  width: 20rem;
  padding: 1rem 4rem 1rem 1rem;
  color: white;
  background: var(--arrow-icon) no-repeat right 0.8em center / 1.4em,
    linear-gradient(to left, var(--arrow-bg) 3em, var(--select-bg) 3em);
  border-radius: 0.25em;
  box-shadow: 0 0 1em 0 rgba(0, 0, 0, 0.2);
  cursor: pointer;
  border-radius: 100px;
}

select option {
    color: black;
    background-color: white;
}
```

Рисунок 3.13 – Додані стилі для select меню

```css
.submit-button {
  font-family: 'Montserrat', sans-serif;
  background: #ed9619;
  color: #fff;
  border-radius: 100px;
  display: block;
  width: 205px;
  padding: 15px 0;
  text-align: center;
  text-decoration: none;
  transition: all 0.5s ease;
  border: 0;
  cursor: pointer;
  display: inline;
  margin-left: 11px;
}

.submit-button:hover {
  background: #d9860e;
}
```

Рисунок 3.14 – Додані стилі для кнопки submit

```
110  ul {
111    display: grid;
112    justify-content: space-evenly;
113    list-style-type: none;
114  }
115
116  li {
117    appearance: none;
118    border: 0;
119    outline: 0;
120    font-family: 'Open Sans', sans-serif;
121    text-align: center;
122    width: 20rem;
123    padding: 1rem 4rem 1rem 4rem;
124    color: white;
125    background: no-repeat right 0.8em center / 1.4em,
126      linear-gradient(to left, var(--arrow-bg) 0em, var(--select-bg) 0em);
127    border-radius: 0.25em;
128    box-shadow: 0 0 1em 0 rgba(0, 0, 0, 0.2);
129    border-radius: 100px;
130    margin: 0.5rem 0;
131  }
```

Рисунок 3.15 – Стилі для виведення списку результатів

На цьому верстку завершено (див. рис. 3.16).

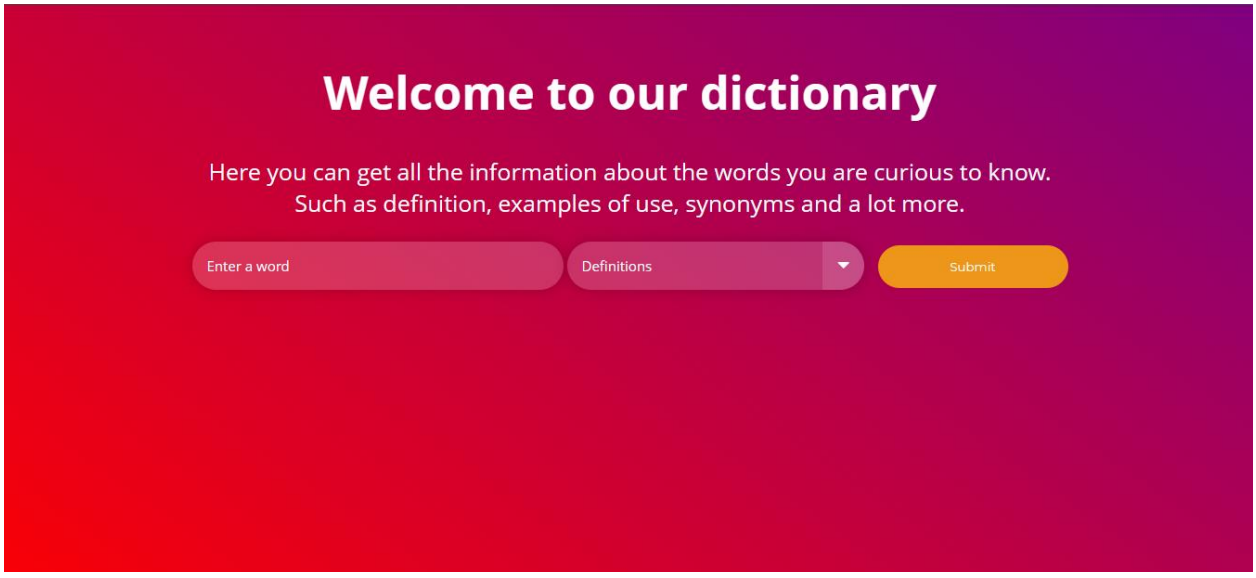

Рисунок 3.16 – Вигляд веб-ресурсу після верстки

Верстка усіх основних елементів завершена, словник виглядає так само як і на макетах створених під час проєктування та розробки користувацького інтерфейсу.

3.3.2 Написання функціональної частини за допомогою JavaScript

Перед тим як почати писати код для функціональної частини, я знову пройшовся по інформації щодо можливостей API та остаточно вирішив, що саме, я буду використовувати. Я обрав наступні типи інформації для обробки отриманих слів:

- definitions;
- synonyms;
- antonyms;
- examples;
- rhymes;
- syllables;
- pronunciation;
- similarTo;
- also.

Definitions надає значення введеного слова, synonyms надає синоніми, antonyms виводить антоніми до слова, examples виводить приклади використання слова у реченні, rhymes показує існуючі до слова рими, syllables розбирає слово на склади, pronunciation показує транскрипцію слова, similarTo виводить слова які схожі на введене слово, але не є синонімами. І остання опція Also виводить фрази частиною яких є введене слово, це може стати корисним під час пошуку фразових дієслів.

Визначившись який саме функціонал від API, я буду використовувати, я повернувся до html файлу та додав ще декілька тегів option яких не вистачало. (див. рис. 3.17)

```
<select id="info-type" required>
    <option value="definitions">Definitions</option>
    <option value="synonyms">Synonyms</option>
    <option value="antonyms">Antonyms</option>
    <option value="examples">Examples</option>
    <option value="rhymes">Rhymes</option>
    <option value="syllables">Syllables</option>
    <option value="pronunciation">Pronunciation</option>
    <option value="similarTo">Similar to, but not synonyms</option>
    <option value="also">Phrases of which the word is a part</option>
</select>
```

Рисунок 3.17 – Додані усі необхідні теґи option з типами інформації

Раніше у html коді, я створив форму, яка містить поле введення слова, select меню та кнопку, ця форма має id dictionary-form. У JavaScript файлі додав обробника подій submit, який спрацьовує при натисканні кнопки для відправлення даних введених користувачем. Одразу після обробника подій, я додав event.preventDefault(); що запобігає перезавантаженню сторінки після відправки форми. Далі прописане отримання даних з полів введення та вибору, а саме введене користувачем слово (word) з поля введення input з id word-input, тип інформації для обробки слова (infoType) з select меню з id infoType. Далі реалізовано очищення результатів, перед повторним пошуком вміст елемента resultList очищається (див. рис. 3.18).

```
document.getElementById('dictionary-form').addEventListener('submit', function(event) {
    event.preventDefault();

    const word = document.getElementById('word-input').value;
    const infoType = document.getElementById('info-type').value;
    const resultsList = document.getElementById('results');

    resultsList.innerHTML = '';
```

Рисунок 3.18 – Обробник подій для кнопки, отримання даних

Далі прописую запит до API за допомогою методу fetch. Запит виконується за допомогою отриманих від користувача даних word та infoType (див. рис. 3.19). Для здійснення запиту використовуються http-метод GET та

особистий API ключ та посилання на хоста. Одразу ж йде перевірка відповіді, якщо з відповіддю щось не так, користувач отримає помилку, в іншому випадку відповідь перетворюється на JSON

```
10      fetch(`https://wordsapiv1.p.rapidapi.com/words/${word}/${infoType}`, {
11          method: "GET",
12          headers: {
13              'x-rapidapi-key': 'My_API_Key',
14              'x-rapidapi-host': 'wordsapiv1.p.rapidapi.com'
15          }
16      })
17      .then(response => {
18          if (!response.ok) {
19              throw new Error(`HTTP error! status: ${response.status}`);
20          }
21          return response.json();
22      })
```

Рисунок 3.19 – Запит до API

Далі реалізовано логіку обробки даних в залежності від вибору користувача (див. рис. 3.20). У залежності від типу інформації (infoType), дані оброблюються по різному. Для definitions перевіряється наявність data.definitions і отримується список визначень. Для synonyms, antonyms, examples, rhymes, syllables, pronunciation, similarTo, та also перевіряється наявність відповідних полів у відповіді. Для pronunciation перевіряється, чи data.pronunciation.all це рядок або масив і відповідно обробляється.

```
23      .then(data => {
24          let items = [];
25
26          if (infoType === 'definitions' && data.definitions) {
27              items = data.definitions.map(def => def.definition);
28          } else if (infoType === 'synonyms' && data.synonyms) {
29              items = data.synonyms;
30          } else if (infoType === 'antonyms' && data.antonyms) {
31              items = data.antonyms;
32          } else if (infoType === 'examples' && data.examples) {
33              items = data.examples;
34          } else if (infoType === 'rhymes' && data.rhymes && data.rhymes.all) {
35              items = data.rhymes.all;
36          } else if (infoType === 'syllables' && data.syllables && data.syllables.list) {
37              items = data.syllables.list;
38          } else if (infoType === 'pronunciation' && data.pronunciation) {
39              if (typeof data.pronunciation.all === 'string') {
40                  items = [data.pronunciation.all];
41              } else if (Array.isArray(data.pronunciation.all)) {
42                  items = data.pronunciation.all;
43              } else {
44                  items = [data.pronunciation];
45              }
46          } else if (infoType === 'similarTo' && data.similarTo) {
47              items = data.similarTo;
48          } else if (infoType === 'also' && data.also) {
49              items = data.also;
50          } else {
51              throw new Error('No data found for the specified info type');
52          }
```

Рисунок 3.20 – Код обробки даних в залежності від вибору користувача

Вивід результатів: Якщо список items порожній, виводиться повідомлення "No results found". У іншому випадку кожен елемент списку додається до resultsList як елемент li. Якщо сталася помилка під час запиту або обробки даних, вона відображається у списку результатів (див. рис. 3.21).

```
54          if (items.length === 0) {
55              resultsList.innerHTML = '<li>No results found</li>';
56          } else {
57              items.forEach(item => {
58                  const listItem = document.createElement('li');
59                  listItem.textContent = item;
60                  resultsList.appendChild(listItem);
61              });
62          }
63      })
64      .catch(err => {
65          console.error(err);
66          resultsList.innerHTML = `<li>${err.message}</li>`;
67      });
68  });
```

Рисунок 3.21 – Код виводу результатів з обробкою помилок

3.3.3 Тестування роботи розробленого веб-ресурсу

Тепер коли сайт зі словником готовий, я проведу та продемонструю тестування реалізованого функціонала. Розроблений веб-ресурс має 9 різних типів обробки інформації, кожен з цих типів, доступний для вибору в select меню з випадаючим списком (див. рис. 3.22).

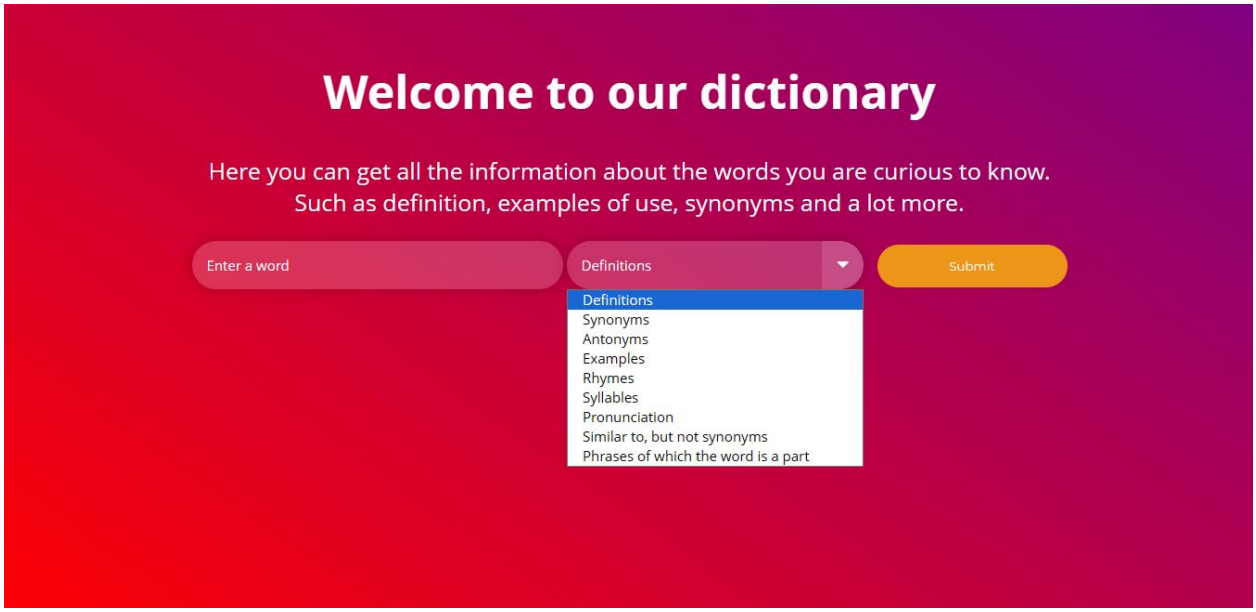

Рисунок 3.22 – Випадаючий список з типами інформації

Почнімо з перевірки першого типу обробки інформації – definitions (див. рис. 3.23). Цей тип вже обраний, тож мені залишається тільки вписати слово, нехай це буде слово – Dungeon

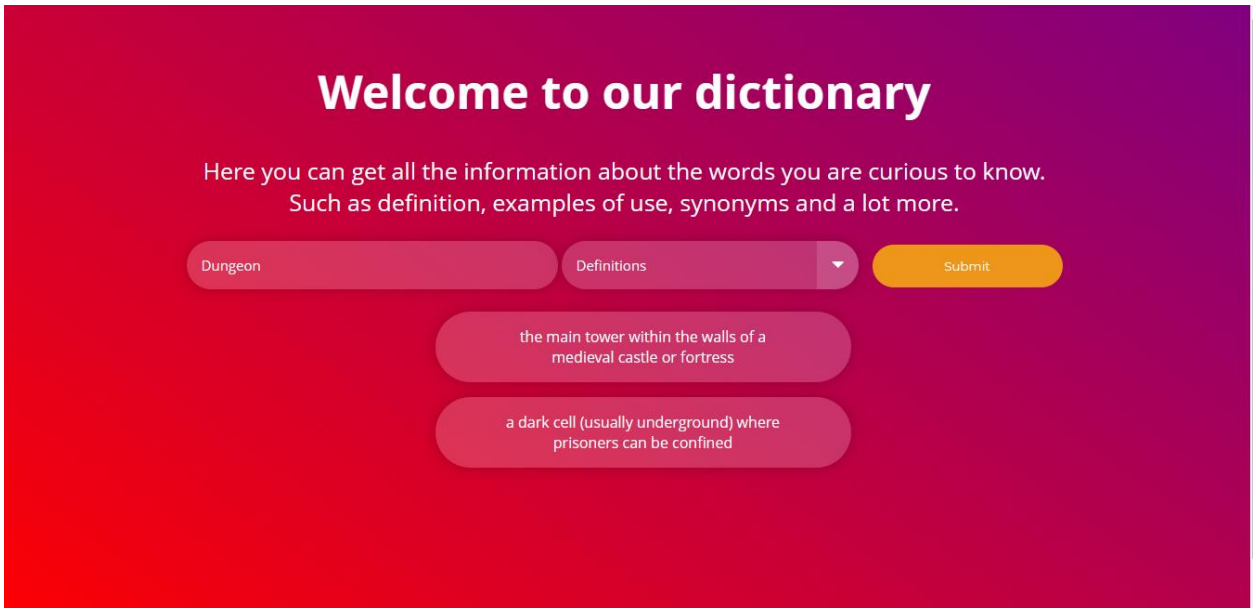

Рисунок 3.23 – Тестування роботи «Definitions»

Наступним буде тип даних для виведення синонімів – synonyms. Обираємо його вводимо наступне слово та перевіряємо результат (див. рис. 3.24).

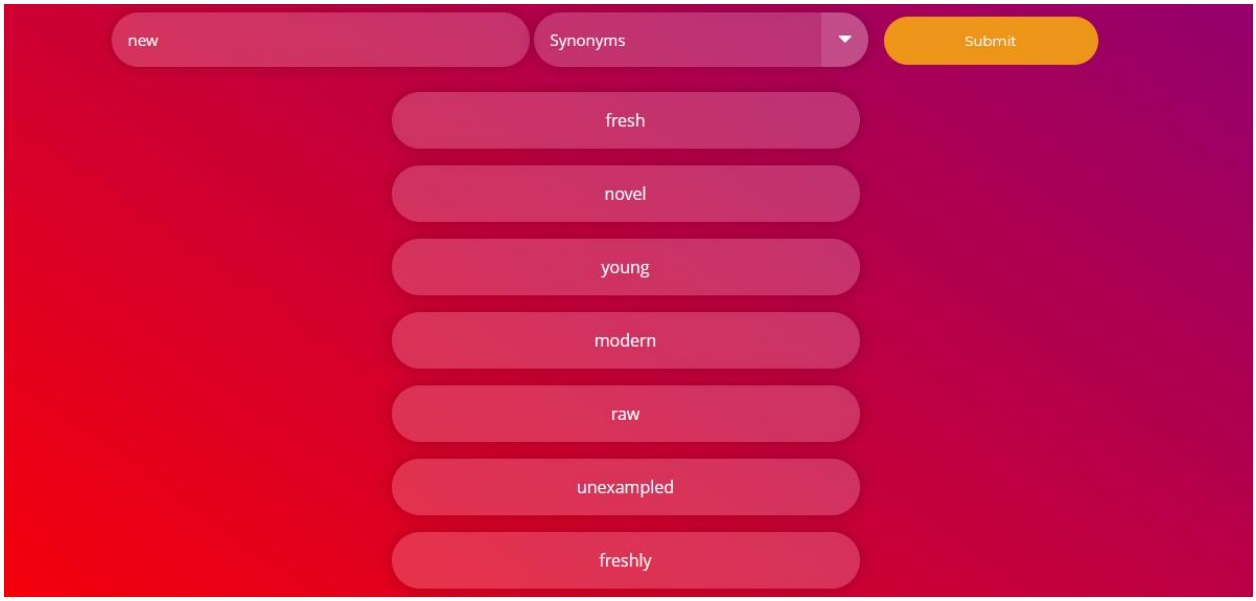

Рисунок 3.24 – Тестування роботи «Synonyms»

Далі поглянемо на виведення антонімів, перевіряючи – antonyms (див. рис. 3.25).

Рисунок 3.25 – Тестування роботи «Antonyms»

Тестування типу з прикладами у реченні – examples (див. рис. 3.26).

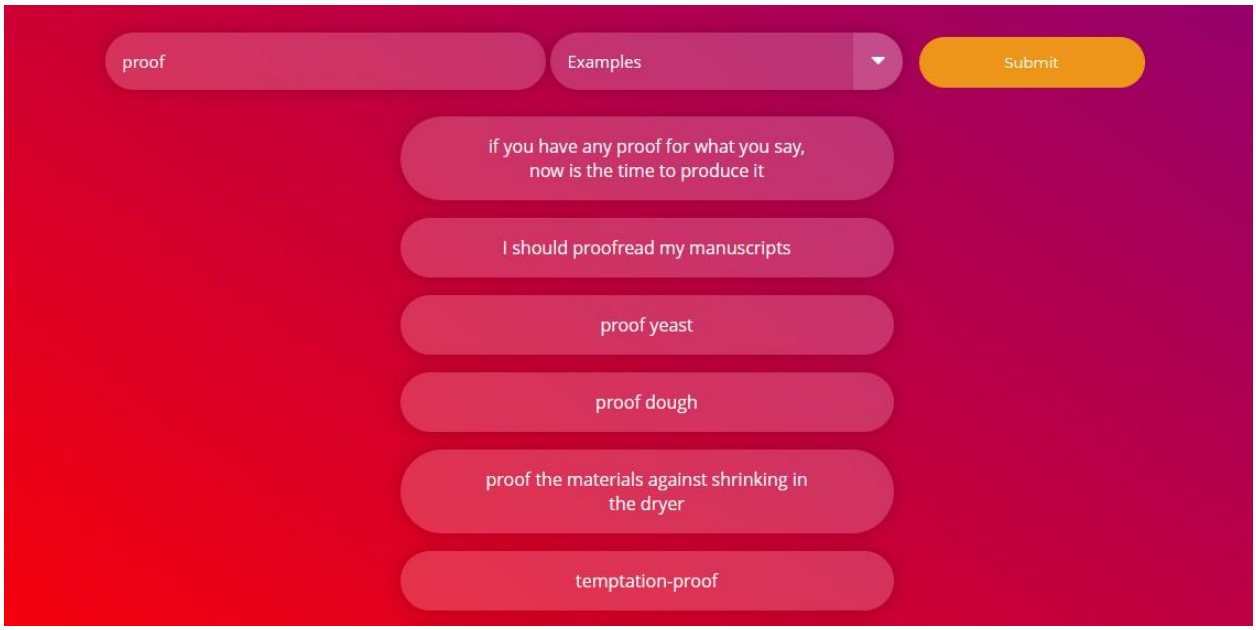

Рисунок 3.26 – Тестування роботи «Examples»

Тестування типу даних, що виводить існуючи до слова рими – rhymes (див. рис. 3.27–3.28).

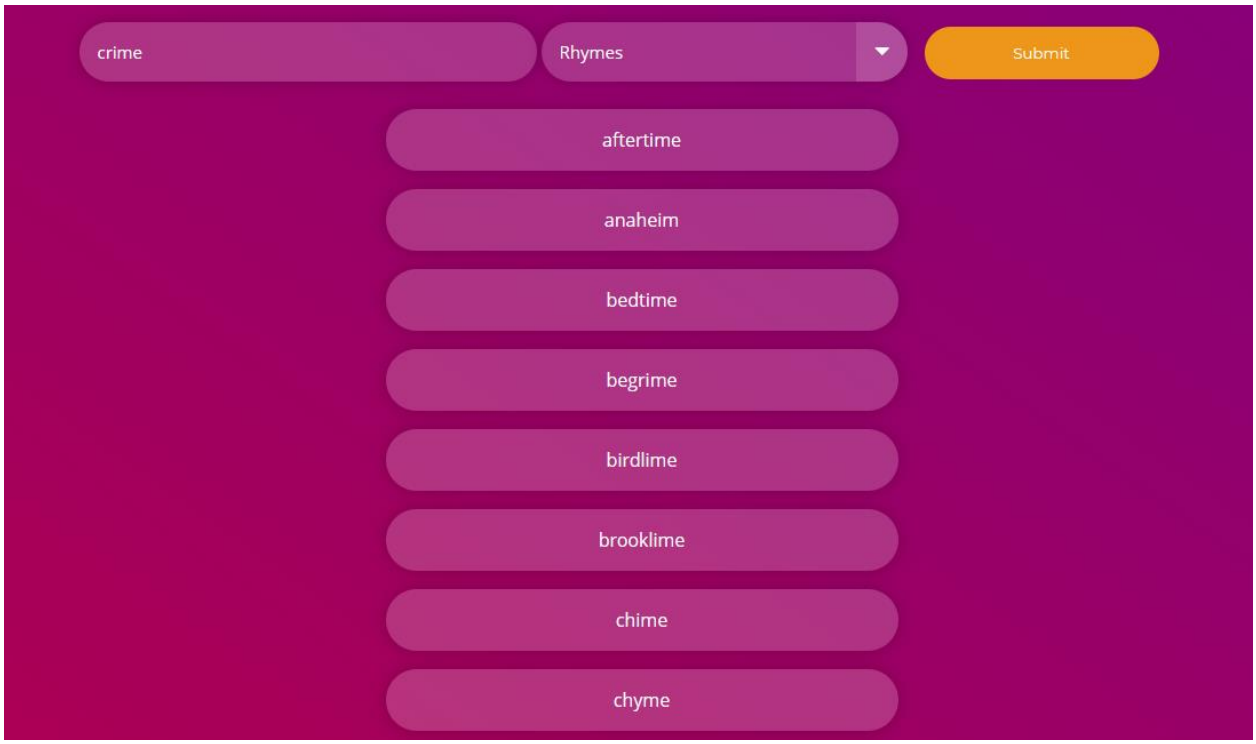

Рисунок 3.27 – Тестування роботи «Rhymes» №1

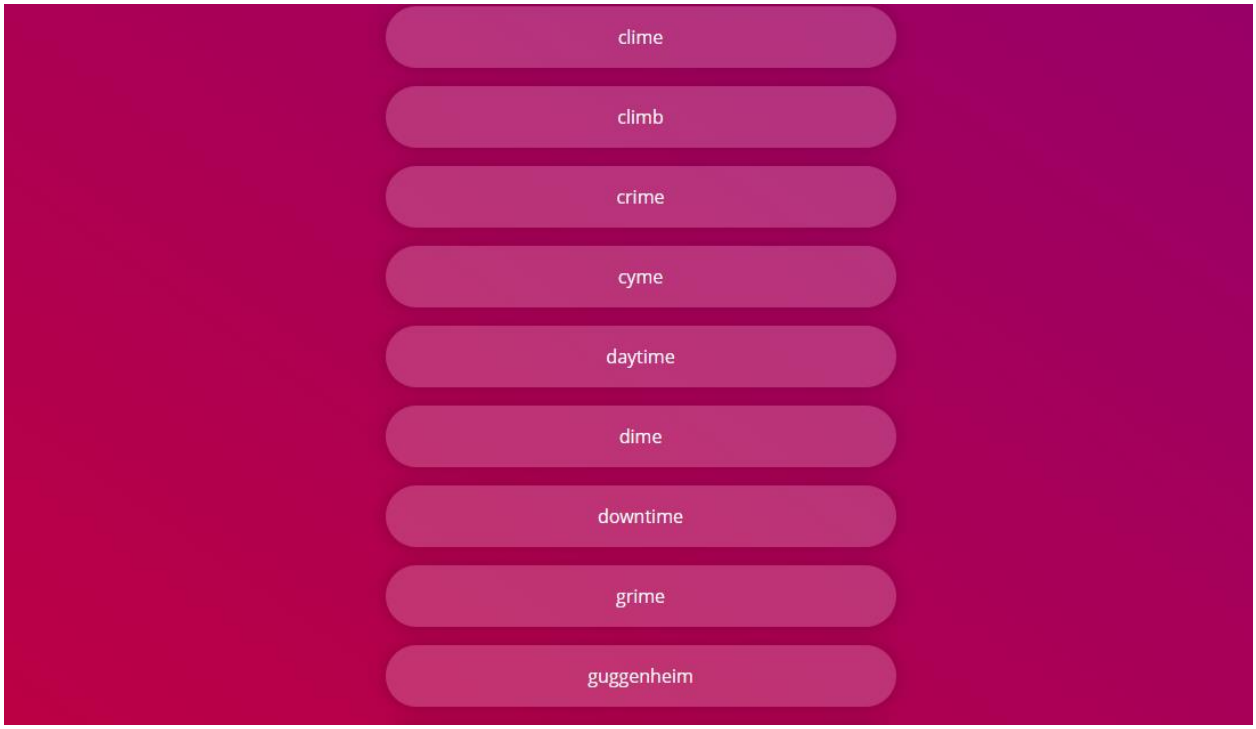

Рисунок 3.28 – Тестування роботи «Rhymes» №2

На черзі тестування типу «Syllables», який розкладає слово на склади (див. рис. 3.29).

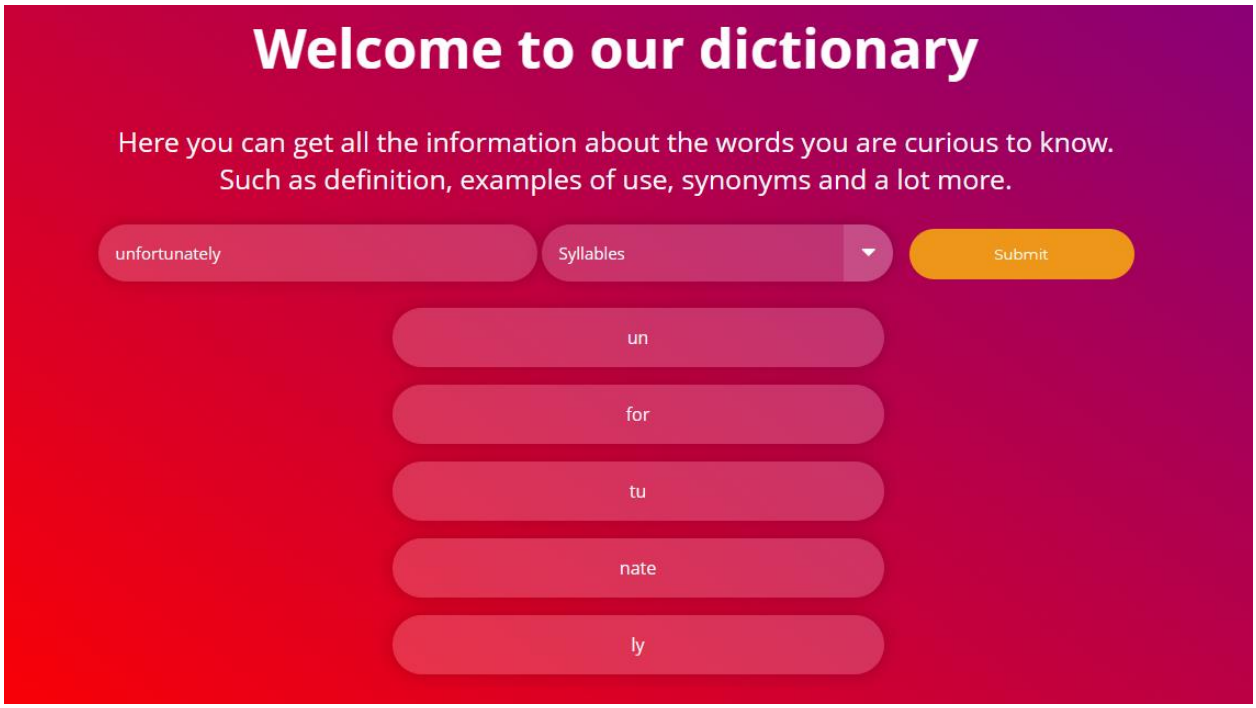

Рисунок 3.29 – Тестування роботи «Syllables»

Наступним типом є виведення транскрипції слова, яка сприяє кращому розумінню вимови слова (див. рис. 3.30).

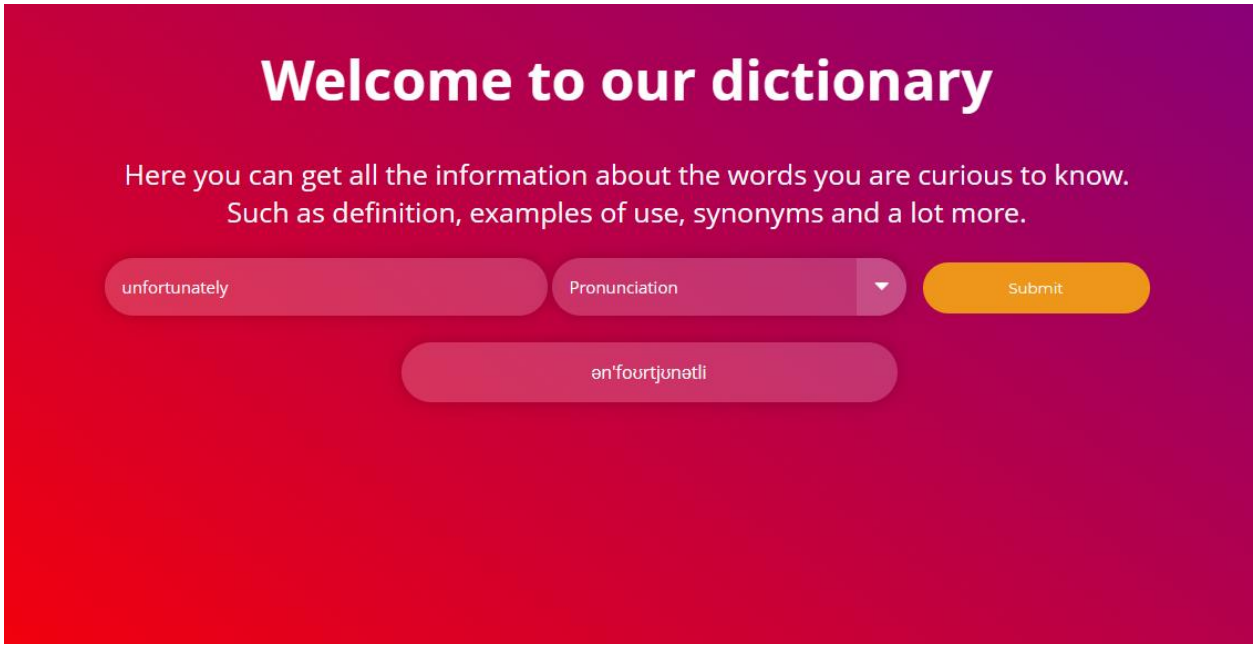

Рисунок 3.30 – Тестування роботи «Pronunciation»

Тип «SimilarTo» показує слова які схожі на введені, але при цьому не є синонімами цього слова. Результатів знову більше ніж може бути на зображенні, тому я залишу тільки одне зображення (див. рис. 3.31).

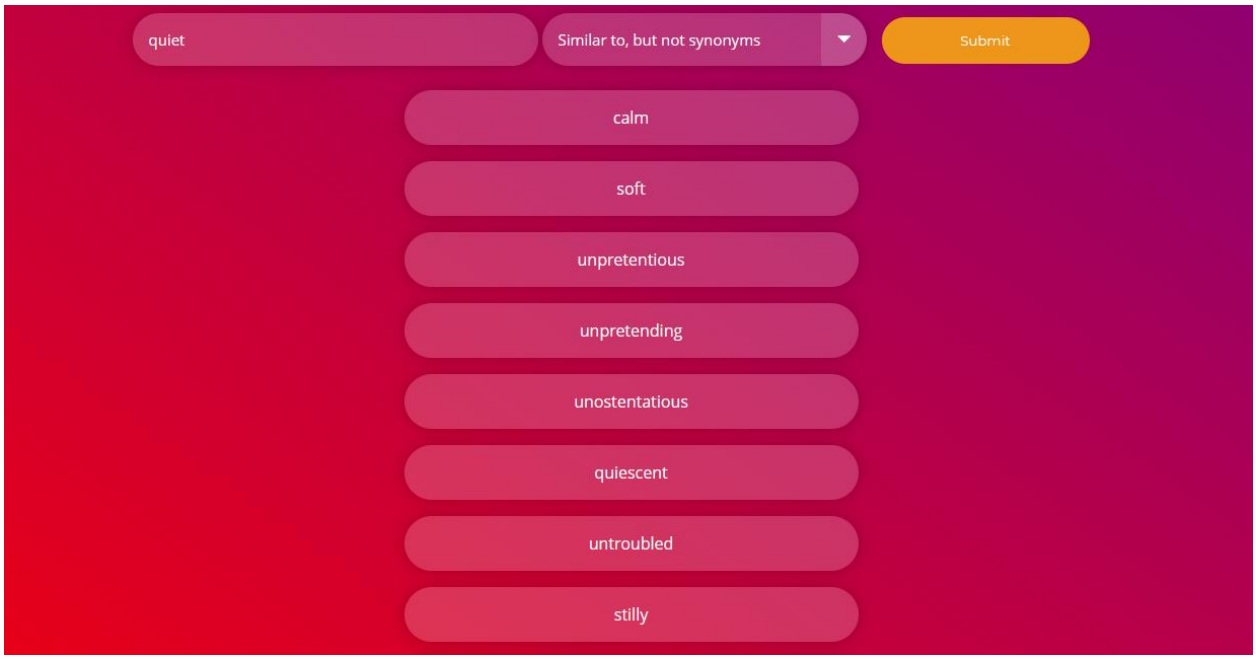

Рисунок 3.31 – Тестування роботи «Similar to, but not synonyms»

Наступний тип обробки даних, раніше згадувався як «also». У коді він теж так підписаний, але для більшої зрозумілості для користувачів, я трохи більше розписав його призначення у назві з меню вибору. Насправді ж цей тип показує чи є введене слово частиною якоїсь фрази. За допомогою цього типу можна легко шукати фразові дієслова (див. рис. 3.32).

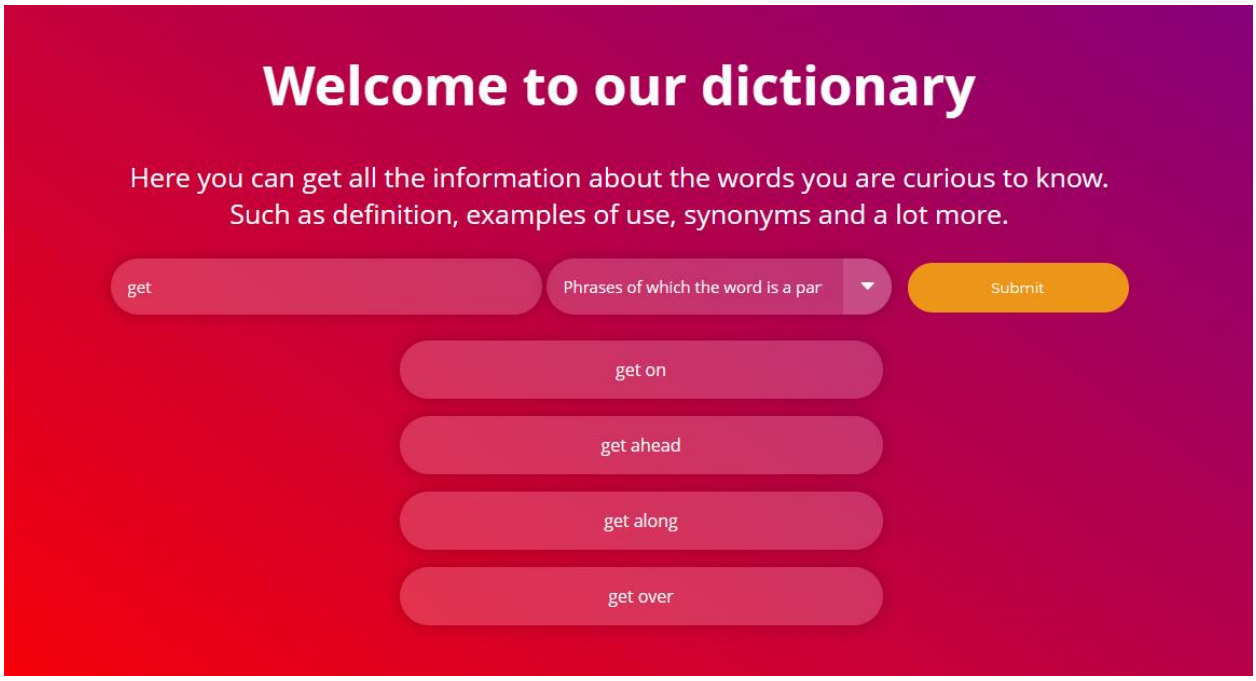

Рисунок 3.32 – Тестування роботи «Phrases of which the word is a part»

Отож усі тести пройшли перевірку та ви на власні очі побачили роботу всіх розроблених функцій, веб-ресурс працює та виконує усі поставленні задачі.

ВИСНОВКИ

Актуальність теми дослідження зумовлена зростаючою потребою у вивченні розмовної англійської мови. Вона необхідна для спілкування в подорожах, роботі, навчанні та інших сферах життя. Інтерактивні словники стають все більш популярними інструментами для вивчення мови, завдяки своїй інтерактивності, доступності та різноманітності функцій.

У роботі проаналізовано існуючі інтерактивні словники розмовної англійської мови, їх переваги та недоліки. Аналіз вже існуючих інтерактивних словників допоміг чітко визначити основні вимоги, а також необхідні функціональні можливості та особливості власного розробленого веб-ресурсу.

Завдяки навичкам здобутих на різних курсах під час навчання, я зміг підійти до питання проєктування користувацького інтерфейсу з іншого боку та спроєктував дизайн сайту, який буде дійсно зручним у використанні.

Протягом написання цієї роботи, я набув нових знань та навичок з реалізації front-end проєктів з використанням API. Проаналізувавши технічні аспекти створення таких веб-ресурсів, я заглибився в процес та зміг реалізувати задуманий проєкт для кваліфікаційної роботи бакалавра. Увесь процес створення веб-ресурсу інтерактивного словника розмовної англійської мови, я якомога детальніше розписав у третьому розділі.

Створений веб-ресурс має стати у нагоді багатьом користувачам, котрі вивчають або тільки планують вивчати англійську мову. Також розроблений інтерактивний словник має потенціал на удосконалення у майбутньому. Розширивши можливості сайту новими корисними функціями, які не були включені у цю роботу, можна значно покращити якість сайту та збільшити кількість активних користувачів, яким це дійсно буде корисно.

ПЕРЕЛІК ПОСИЛАНЬ

1. WordsAPI. RapidAPI. URL: https://rapidapi.com/dpventures/api/wordsapi (Дата звернення: 25.05.2024)
2. Introduction to web APIs. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Introduction (Дата звернення: 21.04.2024)
3. Working with APIs in JavaScript. Geeksforgeeks. URL: https://www.geeksforgeeks.org/working-with-apis-in-javascript/ (Дата звернення: 21.04.2024)
4. JavaScript Tutorial. W3Schools. URL: https://www.w3schools.com/js/ (Дата звернення: 11.03.2024)
5. Chris Bank. The guide to wireframing: guidebook for designers. San Francisco: Chronicle books, 2021. 187с. (Дата звернення: 13.05.2024)
6. Ian Collen. Language trends 2020. British Council. URL: https://www.britishcouncil.org/sites/default/files/language_trends_2020_0.pdf (Дата звернення: 10.05.2024)
7. The most spoken languages worldwide 2023. Statista. URL: https://www.statista.com/statistics/266808/the-most-spoken-languages-worldwide/ (Дата звернення: 11.05.2024)

ДОДАТОК А

HTML код веб-ресурсу

```
<!DOCTYPE html>
<html>
<head>
	<meta charset="utf-8">
	<meta name="viewport" content="width=device-width, initial-scale=1">
	<title>Dictionary Web-site</title>
	<link rel="stylesheet" type="text/css" href="css\styles.css">
</head>
<body>
	<header>
		<h1>Welcome to our dictionary</h1>

		<div>
			<p>Here you can get all the information about the words you are curious to know.
			<br>Such as definition, examples of use, synonyms and a lot more.</p>
		</div>

	</header>
	<section>

		<form id="dictionary-form">
			<input type="text" id="word-input" placeholder="Enter a word" required>
			<select id="info-type" required>
```

```
                    <option value="definitions">Definitions</option>
                    <option value="synonyms">Synonyms</option>
                    <option value="antonyms">Antonyms</option>
                    <option value="examples">Examples</option>
                    <option value="rhymes">Rhymes</option>
                    <option value="syllables">Syllables</option>
                    <option value="pronunciation">Pronunciation</option>
                    <option value="similarTo">Similar to, but not synonyms</option>
                    <option value="also">Phrases of which the word is a part</option>
                </select>
                <button class="submit-button" type="submit">Submit</button>
            </form>
            <ul id="results"></ul>

        </section>

        <script src="js/main.js"></script>
</body>
</html>
```

ДОДАТОК Б

CSS код веб-ресурсу

```
body {
  place-items: center;
  min-height: 100vh;
  background: linear-gradient(35deg, red, purple);
}

h1 {
        font-family: ‘Open Sans’, sans-serif;
        color: #fff;
        font-size: 50px;
        text-align: center;
        padding-top: 2%;
}

p {
        font-family: ‘Open Sans’, sans-serif;
        color: #fff;
        font-size: 25px;
        text-align: center;
}

form {
        text-align: center;
}

input {
```

```
        appearance: none;
        border: 0;
        outline: 0;
        font-size: 15px;
        font-family: ‘Open Sans’, sans-serif;
        width: 20rem;
        padding: 1rem 4rem 1rem 1rem;
        color: white;
        background: no-repeat right 0.8em center / 1.4em,
        linear-gradient(to left, var(--arrow-bg) 0em, var(--select-bg) 0em);
        border-radius: 0.25em;
        box-shadow: 0 0 1em 0 rgba(0, 0, 0, 0.2);
        cursor: pointer;
        border-radius: 100px;
}

input:focus, input:active {
    outline: none;
    color: white;
    background: no-repeat right 0.8em center / 1.4em,
        linear-gradient(to left, var(--arrow-bg) 0em, var(--select-bg) 0em);
}

::placeholder {
        font-size: 15px;
        font-family: ‘Open Sans’, sans-serif;
        color: white;
        opacity: 1;
}
```

```
/* select меню */

:root {
  --arrow-bg: rgba(255, 255, 255, 0.3);
  --arrow-icon:
url(https://upload.wikimedia.org/wikipedia/commons/9/9d/Caret_down_font_awes
ome_whitevariation.svg);
  --option-bg: white;
  --select-bg: rgba(255, 255, 255, 0.2);
}

select {
  appearance: none;
  border: 0;
  outline: 0;
  font-size: 15px;
  font-family: ‘Open Sans’, sans-serif;
  width: 20rem;
  padding: 1rem 4rem 1rem 1rem;
  color: white;
  background: var(--arrow-icon) no-repeat right 0.8em center / 1.4em,
    linear-gradient(to left, var(--arrow-bg) 3em, var(--select-bg) 3em);
  border-radius: 0.25em;
  box-shadow: 0 0 1em 0 rgba(0, 0, 0, 0.2);
  cursor: pointer;
  border-radius: 100px;
}

select option {
    color: black;
```

```
    background-color: white;
}


.submit-button {
        font-family: ‘Montserrat’, sans-serif;
        background: #ed9619;
        color: #fff;
        border-radius: 100px;
        display: block;
        width: 205px;
        padding: 15px 0;
        text-align: center;
        text-decoration: none;
        transition: all 0.5s ease;
        border: 0;
        cursor: pointer;
        display: inline;
        margin-left: 11px;
}

.submit-button:hover {
        background: #d9860e;
}

ul {
        display: grid;
        justify-content: space-evenly;
        list-style-type: none;
}
```

```
li {
        appearance: none;
        border: 0;
        outline: 0;
        font-family: ‘Open Sans’, sans-serif;
        text-align: center;
        width: 20rem;
        padding: 1rem 4rem 1rem 4rem;
        color: white;
        background: no-repeat right 0.8em center / 1.4em,
          linear-gradient(to left, var(--arrow-bg) 0em, var(--select-bg) 0em);
        border-radius: 0.25em;
        box-shadow: 0 0 1em 0 rgba(0, 0, 0, 0.2);
        border-radius: 100px;
        margin: 0.5rem 0;
}
```

ДОДАТОК В

JavaScript код веб-ресурсу

```
document.getElementById(‘dictionary-form’).addEventListener(‘submit’,
function(event) {
    event.preventDefault();

    const word = document.getElementById(‘word-input’).value;
    const infoType = document.getElementById(‘info-type’).value;
    const resultsList = document.getElementById(‘results’);

    resultsList.innerHTML = ‘‘;

    fetch(`https://wordsapiv1.p.rapidapi.com/words/${word}/${infoType}`, {
        method: "GET",
        headers: {
                    ‘x-rapidapi-key’: ‘API_KEY’,
                    ‘x-rapidapi-host’: ‘wordsapiv1.p.rapidapi.com’
                }
    })
    .then(response => {
        if (!response.ok) {
            throw new Error(`HTTP error! status: ${response.status}`);
        }
        return response.json();
    })
    .then(data => {
        let items = [];
```

```
if (infoType === ‘definitions’ && data.definitions) {
   items = data.definitions.map(def => def.definition);
} else if (infoType === ‘synonyms’ && data.synonyms) {
   items = data.synonyms;
} else if (infoType === ‘antonyms’ && data.antonyms) {
   items = data.antonyms;
} else if (infoType === ‘examples’ && data.examples) {
   items = data.examples;
} else if (infoType === ‘rhymes’ && data.rhymes && data.rhymes.all) {
   items = data.rhymes.all;
} else if (infoType === ‘syllables’ && data.syllables && data.syllables.list) {
   items = data.syllables.list;
} else if (infoType === ‘pronunciation’ && data.pronunciation) {
   if (typeof data.pronunciation.all === ‘string’) {
      items = [data.pronunciation.all];
   } else if (Array.isArray(data.pronunciation.all)) {
      items = data.pronunciation.all;
   } else {
      items = [data.pronunciation];
   }
} else if (infoType === ‘similarTo’ && data.similarTo) {
   items = data.similarTo;
} else if (infoType === ‘also’ && data.also) {
   items = data.also;
} else {
   throw new Error(‘No data found for the specified info type’);
}

if (items.length === 0) {
   resultsList.innerHTML = ‘<li>No results found</li>‘;
```

```
        } else {
            items.forEach(item => {
                const listItem = document.createElement('li');
                listItem.textContent = item;
                resultsList.appendChild(listItem);
            });
        }
    })
    .catch(err => {
        console.error(err);
        resultsList.innerHTML = `<li>${err.message}</li>`;
    });
});
```

Декларація
академічної доброчесності
здобувача ступеня вищої освіти ЗНУ

Я, Плюснін Валентин Григорович

студент(ка) __4__ курсу, ____денної_____ форми навчання, математичного факультету, спеціальності __122 комп'ютерні науки_____, адреса електронної пошти __valentynpliusnin@gmail.com_____, –

підтверджую, що написана мною кваліфікаційна робота магістра на тему «___Розробка web-ресурсу інтерактивного словника з сучасної розмовної англійської мови засобами Java Script____» відповідає вимогам академічної доброчесності та не містить порушень, що визначені у ст. 42 Закону України «Про освіту», зі змістом яких ознайомлений/ознайомлена;

– заявляю, що надана мною для перевірки електронна версія роботи є ідентичною її друкованій версії;

– згоден/згодна на перевірку моєї роботи на відповідність критеріям академічної доброчесності у будь-який спосіб, у тому числі за допомогою інтернет-системи, а також на архівування моєї роботи в базі даних цієї системи.

Студент

05.06.24		В.Г. Плюснін
(дата)	(підпис)	(прізвище, ініціали)

Науковий керівник

(дата)	(підпис)	(прізвище, ініціали)