МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютеризованих систем управління

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри

_____Литвиненко О.Є.
"_____"_____ 2022 р.

ДИПЛОМНИЙ ПРОЕКТ
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ
"БАКАЛАВР"

Тема: Веб-додаток пошуку медіа-контенту за голосом.

_____

_____

Виконавець: _____ Анісімов М.П

Керівник: _____ к.т.н., доцент Халімон Н. Ф.

Нормоконтролер: _____Тупота Є.В.

Київ 2022

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії
Кафедра комп’ютеризованих систем управління
Спеціальність 123 «Комп'ютерна інженерія»
(шифр, найменування)

Освітньо-професійна програма «Системне програмування»
Форма навчання денна

ЗАТВЕРЖУЮ
Завідувач кафедри

Литвиненко О. Є.

« ____ »_____2022 р.

З А В Д А Н Н Я
на виконання дипломної роботи (проекту)

Анісімова Максима Павловича
(прізвище, ім’я, по батькові випускника в родовому відмінку)

1. Тема дипломної роботи (проекту): Веб-додаток пошуку медіа-контенту за голосом

затверджена наказом ректора від "04" лютого 2022 року № 135/ст.

2. Термін виконання роботи (проекту): з 17.05.2022 до 20.06.2022

3. Вихідні дані до роботи (проекту): база даних реального часу *Firebase Realtime Database*, інтегроване середовище розробки *Visual Studio Code*, мова *JavaScript*, мова *HTML*, мова *CSS*.

4. Зміст пояснювальної записки:
1) Особливості пошуку медіа-контенту.
2) Проектування веб-додатку пошуку медіа-контенту за голосом.
3) Розробка веб-додатку пошуку медіа-контенту за голосом.

5. Перелік обов’язкового графічного (ілюстративного) матеріалу:

1) Головна сторінка веб-додатку пошуку медіа-контенту за голосом.

2) Сторінка пошуку фільмів веб-додатку пошуку медіа-контенту за голосом.

3) Сторінка пошуку книг веб-додатку пошуку медіа-контенту за голосом.

4) Сторінка пошуку музикальних композицій веб-додатку пошуку медіа-контенту за голосом.

6. Календарний план-графік

№ пор.	Завдання	Термін виконання	Відмітка про виконання
1	Ознайомитись з постановкою задачі дипломного проектування	18.05.22	
2	Вивчити спеціальну літературу і технічну документацію	14.05.22	
3	Дослідити додатки для пошуку медіа-контенту, допоміжне програмне забезпечення, спеціальне програмне забезпечення для обробки аудіо.	16.05.22	
4	Написати розділ 1.	18.05.22	
5	Спроектувати веб-додаток пошуку медіа-контенту за голосом.	19.05.22	
6	Написати розділ 2.	21.05.22	
7	Розробити веб-додаток пошуку медіа-контенту за голосом.	25.05.22	
8	Написати розділ 3.	04.06.22	
9	Оформити пояснювальну записку	08.06.22	
10	Підготувати графічний демонстраційний матеріал та доповідь	09.06.22	

7. Дата видачі завдання: “22” грудня 2022 р.

Керівник дипломної роботи (проекту) _____ Халімон Н.Ф.
(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання _____ Анісімов М.П.
(підпис випускника) (П.І.Б.)

РЕФЕРАТ

Пояснювальна записка до дипломного проекту «Веб-додаток пошуку медіа-контенту за голосом»: 50 с., 11 рис., 15 літературних джерел.

JAVASCRIPT, *FIREBASE*, *VUE.JS*, ДОСТУПНІСТЬ, *МЕДІА-КОНТЕНТ*, ВЕБ-ДОДАТОК ПОШУКУ ЗА ГОЛОСОМ, АНАЛІЗ МОВЛЕННЯ.

Об’єкт проектування – інтернет-ресурси медіа-контенту.

Предмет проектування – розробка веб-додатку пошуку медіа-контенту за голосом.

Мета дипломного проекту – розробка веб-додатку, що дозволить користувачам здійснювати пошуку медіа-контенту за голосом.

Метод проектування – застосування крос-платформного програмного засобу для розробки веб-додатку.

ЗМІСТ

ВСТУП

Медіа-контент є одним з найпопулярніших видів інформації, що можна знайти в Інтернеті. Практично кожен користувач Інтернету сьогодні кожен день шукає або передивляється медіа-контент. Але не кожен додаток для пошуку медіа-контенту є доступним. Основа проблема з якою стикаються користувачі при використанні не доступних програм – неможливість використання людьми з обмеженими можливостями. Користувачі з проблемами зору, слуху і т.д можуть стикнутися з тим, що просто не зможуть використовувати додаток. Завданням є розробити додаток, що дозволить користувачам з обмеженими можливостями шукати різний тип медіа-контенту.

Список основних проблем, з якими стикаються люди з обмеженими можливостями, при використанні різного програмного забезпечення: низький контраст тексту, відсутність можливості навігації за допомогою клавіатури, відсутність тексту-пояснення для зображень, тощо.

Низький контраст тексту – це одна з найпопулярніших проблем. Цю проблему мають близько 85% веб-сторінок у мережі. Вона з'являється, коли колір тексту погано поєднується з кольором заливки. Наприклад, жовтий на білому. Такий текст дуже важко читати.

Відсутність можливості навігації за допомогою клавіатури характерна наступним. Сучасний стандарт *HTML5* за замовчуванням надає можливість навігації за допомогою клавіатури між інтерактивними тегами (*button, a, input* і т.д), але багато розробників веб-сайтів можуть створювати інтерактивні елементи без використання цих тих тегів, через це навігація може працювати некоректно).

Відсутність тексту-пояснення для зображень можна характеризувати таким чином. Цю проблему мають близько 68% веб-сторінок. Проблема пов'язана з тим, що зображення – це тип інформації, яку люди з поганим зором не можуть ефективно сприйняти. Тому розробники, які вставляють зображення на сайт, повинні постійно надавати текст-пояснення цьому зображенню.

Це не всі, але найпопулярніші проблеми, з якими стикаються люди з обмеженнями при використанні різного програмного забезпечення. На щастя, про

таких людей не забули і створили різноманітне програмне забезпечення, щоб допомагати їм. Таке програмне забезпечення називають допоміжним. Допоміжне програмне забезпечення має чимало реалізацій, наприклад: читач екрану, екранна лупа, помічник навігації, тощо.

Відео, музика, книги, зображення є невід'ємною частиною медіа-контенту. Звичайні люди можуть легко шукати таку інформацію за допомогою різних пошукових систем, наприклад: *Google, Yahoo, Bing*.

Найпопулярнішим видом аудіо інформації сьогодні є музика. Для її пошуку існує багато програм. Однією з відомих платформ є *Spotify*. Це постачальник потокових аудіо, заснований 23 квітня 2006 року. Цей сервіс використовується як сервіс для розповсюдження музики, але також має потужні інструменти пошуку. У *Spotify* музику можна шукати за різними параметрами, наприклад, виконавцем, альбомом, жанром, списком відтворення або лейблом звукозапису. Користувачі можуть створювати, редагувати та ділитися списками відтворення, ділитися треками в соціальних мережах та створювати списки відтворення з іншими користувачами. *Spotify* надає доступ до понад 70 мільйонів пісень, 2,2 мільйонів подкастів і 4 мільярдів списків відтворення.

В другому розділі було розглянуто проектування веб-додатку пошуку медіа-контенту за голосом. Було визначено наступні функціональні особливості: пошук фільмів; пошук композицій музики; пошук книг; реєстрація та авторизація; можливість додавання типу медіа-контенту (фільм, музику або книгу) до "обраного". Інтерфейс було спроектовано адаптивним, кросбраузерним та доступним.

При проектуванні веб-додатку пошуку медіа-контенту за голосом було визначено такі сторінки: '/' – головна сторінка; '/*movies'* – сторінка для пошуку інформації про фільми; '/*books'* – сторінка для пошуку інформації про книги; '/*music'* – сторінка для пошуку інформації про музику; '/*sign-up'* – сторінка для реєстрації акаунту користувача; '*/sign-in'* – сторінка для входу в акаунт користувача; *'/favorites'* – сторінка для перегляду "обраного" медіа-контенту.

В третьому розділі було описано вибір програмного забезпечення для реалізації додатку, розробку серверної частини та інтерфейсу додатку. Веб-додаток пошуку медіа-контенту складається з: сервісу авторизації та реєстрації, головної сторінки, сторінки пошуку фільмів, сторінки пошуку книг, сторінки пошуку музикальних композицій, сторінки для реєстрації, сторінки для авторизації, компоненту пошуку, сторінки “обраного” медіа-контенту. Для додатку було розроблено базу даних.

Об’єкт проектування – системи пошуку медіа-контенту за голосом.

Предмет проектування – веб-додаток пошуку медіа-контенту за голосом.

Мета дипломного проекту – створення веб-додатку для пошуку медіа-контенту за голосом, який дозволить користувачу знаходити медіа-контент різного типу, використовуючи голосовий пошук. Таким чином, мета проекту – розробити універсальний веб-додаток пошуку медіа-контенту за голосом, який збільшить ефективність користувача під час пошуку медіа-контенту, використовуючи сучасні технології та алгоритми обробки голосу.

Медіа-контент є одним з найпопулярніших видів контенту в мережі. Більшість програмного забезпечення завжди створювалось для людей, які не мають інвалідності та обмежених можливостей. Але є багато людей, які не мають можливості використовувати мишу, клавіатуру та монітор для пошуку інформації, тому тема дипломного проекту “Веб-додаток пошуку медіа-контенту за голосом” є актуальним завданням.

РОЗДІЛ 1
ОСОБЛИВОСТІ ПОШУКУ МЕДІА-КОНТЕНТУ

1.1. Програмне забезпечення для людей з обмеженими можливостями

Медіа-контент є одним з найпопулярніших видів контенту в мережі. Більшість програмного забезпечення завжди створювалось для людей, які не мають інвалідності та обмежених можливостей. Так як більша частина людства – люди, які не мають критичних проблем з зором, мають всі кінцівки, можуть розмовляти, тому інтерфейси програмного забезпечення в першу чергу моделюються під них. Але є багато людей, які не мають можливості використовувати мишу, клавіатуру та монітор для пошуку інформації. Люди, які мають проблеми з зором не можуть читати маленький текст, а є також люди, які просто втратили зір та взагалі не можуть використовувати монітор. Є також дальтоніки, які не можуть розрізняти кольори. Люди без кінцівок не мають змоги використовувати клавіатуру або мишу [1].

Список основних проблем, з якими стикаються люди з обмеженими можливостями, при використанні різного програмного забезпечення: низький контраст тексту, відсутність можливості навігації за допомогою клавіатури, відсутність тексту-пояснення для зображень, тощо. Низький контраст тексту – це одна з найпопулярніших проблем. Цю проблему мають близько 85% веб-сторінок у мережі. Вона з’являється, коли колір тексту погано поєднується з кольором заливки. Наприклад, жовтий на білому. Такий текст дуже важко читати. Відсутність можливості навігації за допомогою клавіатури характерна наступним. Сучасний стандарт *HTML5* за замовчуванням надає можливість

навігації за допомогою клавіатури між інтерактивними тегами (*button, a, input* і т.д), але багато розробників веб-сайтів можуть створювати інтерактивні елементи без використання цих тих тегів, через це навігація може працювати некоректно).

Відсутність тексту-пояснення для зображень можна характеризувати таким чином. Цю проблему мають близько 68% веб-сторінок. Проблема пов'язана з тим, що зображення – це тип інформації, яку люди з поганим зором не можуть ефективно сприйняти. Тому розробники, які вставляють зображення на сайт, повинні постійно надавати текст-пояснення цьому зображенню.

Це не всі, але найпопулярніші проблеми, з якими стикаються люди з обмеженнями при використанні різного програмного забезпечення. На щастя, про таких людей не забули і створили різноманітне програмне забезпечення, щоб допомагати їм. Таке програмне забезпечення називають допоміжним. Допоміжне програмне забезпечення має чимало реалізацій, наприклад: читач екрану, екранна лупа, помічник навігації, тощо.

Читач екрану − це програма, яка допомагає людям з проблемами зору отримувати текстову інформацію з документів. Найпопулярнішим читачем екрану для систем *Appl*e є *VoiceOver*. Використовуючи *VoiceOver*, користувач може отримати доступ до пристрою *Macintosh* або *iOS* на основі голосових описів і, у випадку *Mac*, до клавіатури. Ця функція розроблена для підвищення доступності для сліпих і слабозорих користувачів, а також для користувачів з дислексією.

VoiceOver розглядає інтерфейс користувача як ієрархію елементів, по яких можна переміщатися різними натисканнями клавіш, отримувати їх описання у вигляді звуку. Також можна взаємодіяти, наприклад, з текстовим полем, дозволяючи читати його текст і, якщо можливо, редагувати його; взаємодія зі смугою прокрутки дозволяє переміщати її за допомогою клавіатури. Підтримується більше 30 мов.

Стандартний читачем екрану для систем *Windows* є *Microsoft Narrator,* який розроблено професором Полом Бленкхорном у 1999 році, ця утиліта зробила операційну систему *Windows* більш доступною для сліпих і слабозорих користувачів. Екранний диктор входить у кожну копію *Microsoft Windows,*

забезпечуючи доступ до *Windows* без необхідності встановлення додаткового програмного забезпечення, якщо комп'ютер, що використовується, містить звукову карту та динаміки чи навушники. Програма дозволяє зчитувати все, що відбувається на екрані системи *Windows*, у вигляді звукової інформації; також є можливість переключатися між елементами за допомогою голосу або клавіатури.

Також, дуже популярним у використанні є екранна лупа – це програмне забезпечення, яке взаємодіє з графічним пристроєм комп'ютера для збільшення зображення на екрані. Таким чином, люди, що мають проблеми з зором і не можуть розглядіти якусь інформацію на екрані, зможуть її збільшити. Також це програмне забезпечення корисне для людей, які не можуть використовувати читачів екрану. Найпростіша форма збільшення представляє збільшену частину вихідного вмісту екрана. Ця збільшена частина повинна включати вміст, що цікавить користувача, а також збільшений курсор. Коли користувач переміщує курсор, екранна лупа повинна відслідковувати його та показувати нову збільшену частину.

Рис 1.1. Приклад використання екранної лупи

Програмне забезпечення для допомоги в навігації може бути використано людьми з обмеженими можливостями для розуміння, де вони знаходяться.

Прикладом такого програмного забезпечення є *RightHear*. Ця програма має багато корисних функцій: інформування користувача про своє поточне місцезнаходження за запитом і автоматично, через заздалегідь визначені проміжки часу; надання користувачеві посилання на відповідне місце призначення в Інтернеті (якщо є); збереження цікавих місць користувача у вигляді записів; автоматичні оголошення загальнодоступних визначних місць, перехрестя вулиць і точок, збережених користувачем; підтримка сторонніх додатків для громадського транспорту, таких як *Moovit*, *Uber*, *Lyft*, Gett; створення подорожі від свого поточного місцезнаходження до місця призначення та можливість переглядати розклади та маршрути до місця призначення; моделювання локацій, що дозволяє користувачам досліджувати віддалені місця перед тим, як поїхати туди; оголошення громадських точок та точок загального користування, перехресть, розташованих в напрямку, куди користувач спрямовує свій пристрій; оголошення напрямку, на який вказує користувач за допомогою пристрою; можливість навігації через *Bluetooth*-маяки [2].

Відео, музика, книги, зображення є невід'ємною частиною інформації. Звичайні люди можуть легко шукати таку інформацію за допомогою різних пошукових систем, наприклад: *Google*, *Yahoo*, *Bing*.

У пошуковій системі *Google* є спеціальний пошук зображень *Google Images*. *Google Images* − це пошукова система, яка дозволяє користувачам шукати зображення у всесвітній мережі. Під час пошуку зображення відображається ескіз кожного відповідного зображення. Коли користувач натискає ескіз, зображення відображається у більшому розмірі, і користувачі можуть відвідати веб-сторінку, на якій використовується зображення. Алгоритм цієї системи можна представити у такому вигляді: аналіз зображення (подане зображення аналізується, щоб знайти ідентифікатори, такі як кольори, точки, лінії та текстури), створення запиту (відмінні риси зображення використовуються для створення пошукового запиту), збіг зображення (запит співставляється із зображеннями у серверній частині *Google*), повернення результату (алгоритми пошуку та відповідності *Google* повертають користувачеві відповідні та візуально схожі зображення) [3].

Альтернативною системою пошуку зображень є *TinEye*. Це перша пошукова система зображень у мережі, яка використовує технологію ідентифікації зображень, а не ключові слова, метадані чи водяні знаки. *TinEye* дозволяє користувачам шукати не за ключовими словами, а за зображеннями. Після надсилання зображення *TinEye* створює «унікальний і компактний цифровий підпис або відбиток пальця» зображення та порівнює його з іншими проіндексованими зображеннями. Використання системи можна описати наступним чином: користувач завантажує зображення в пошукову систему (розмір завантаження обмежено 20 МБ) або надає *URL*-адресу зображення або сторінки, що містить зображення. Пошукова система шукатиме інше використання зображення в Інтернеті, включаючи модифіковані зображення на основі цього зображення, і повідомить про дату та час їх опублікування. *TinEye* не розпізнає контури об’єктів і не виконує розпізнавання обличчя, але розпізнає все зображення та деякі змінені версії цього зображення. Це включає менші, більші та обрізані версії зображення. Результати, згенеровані *TinEye*, включають загальну кількість збігів у базі даних, створених надісланим зображенням.

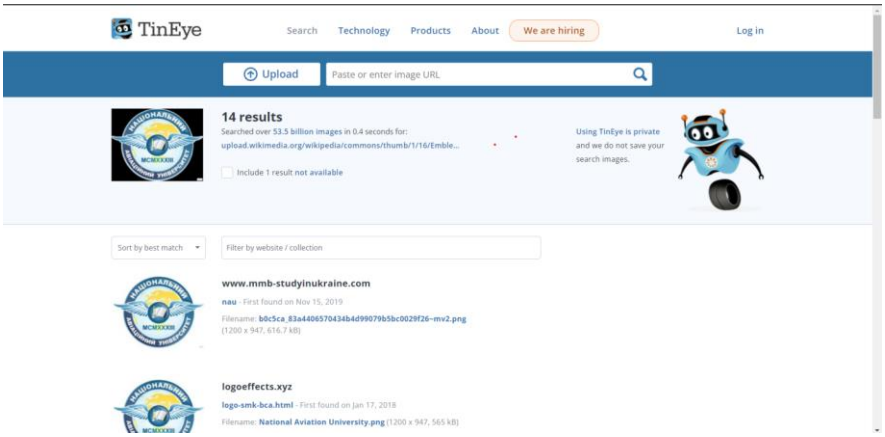

Рис 1.2 Приклад пошуку зображення за допомогою *TinEye*

Найпопулярнішою систему пошуку та обміну відео сьогодні є *YouTube*. Цей сервіс вже давно перетворися на соціальну мережу, але досі виступає часто вживаним інструментом для пошуку відео. Система дозволяє шукати відео за

допомогою тестового або аудіо пошуку, фільтрувати і сортувати видачу пошуку. Для керування пошуком наразі наявні такі фільтри:

- за датою завантаження;
- тип;
- тривалість;
- особливості.

Також можливе сортування за такими критеріями:

- за релевантністю;
- за датою завантаження;
- за кількістю переглядів;
- за рейтингом.

Альтернативним сервісом пошуку відео є *Google Video*. *Google Video* дозволяє шукати відео у всесвітній мережі. Для пошуку треба відкрити веб-сторінку *Google Video* та набрати в пошук ключові слова. Після цього, система надає результат пошуку.

Найпопулярнішим видом аудіо інформації сьогодні є музика. Для її пошуку існує багато програм. Однією з відомих платформ є *Spotify*. Це постачальник потокових аудіо, заснований 23 квітня 2006 року. Цей сервіс використовується як сервіс для розповсюдження музики, але також має потужні інструменти пошуку. У *Spotify* музику можна шукати за різними параметрами, наприклад, виконавцем, альбомом, жанром, списком відтворення або лейблом звукозапису. Користувачі можуть створювати, редагувати та ділитися списками відтворення, ділитися треками в соціальних мережах та створювати списки відтворення з іншими користувачами. *Spotify* надає доступ до понад 70 мільйонів пісень, 2,2 мільйонів подкастів і 4 мільярдів списків відтворення.

Не менш відомою платформою для пошуку аудіо є *Shazam* − це програма, яка може ідентифікувати музику на основі короткого аудіо-зразку, відтвореного за допомогою мікрофона на пристрої. Тобто користувач може за допомогою свого голосу проговорити слова пісні і *Shazam* спробує знайти її. *Shazam* ідентифікує пісні за допомогою створення аудіо-зразку. Він використовує вбудований

мікрофон смартфона або комп'ютера для запису такого зразку. *Shazam* зберігає каталог аудіо-зразків у базі даних. Користувач проспівує пісню протягом 10 секунд, і програма створює аудіо-зразок. *Shazam* аналізує записаний звук і шукає відповідність в базі даних мільйонів пісень. Якщо *Shazam* знаходить відповідність, то надсилає користувачеві інформацію у вигляді назви пісні та її виконавця.

1.2. Спеціальне програмне забезпечення для обробки аудіо

Для запису та відтворення аудіо існує чимало програм, наприклад:

- *Audacity*;
- *Adobe Audition*;
- *Vocaroo*;
- *Diction.io*.

Audacity − це безкоштовний цифровий аудіоредактор з відкритим вихідним кодом і програмне забезпечення для запису, доступне для Windows, *macOS*, *Linux* та інших *Unix*-подібних операційних систем. На додаток до запису аудіо з кількох джерел, *Audacity* можна використовувати для постобробки всіх типів аудіо, включаючи такі ефекти, як нормалізація, обрізка, а також затухання і зникнення. Серед корисних функцій *Audacity* можна виділити такі: запис і відтворення звуків, редагування, великий набір цифрових плагінів і ефектів, виявлення помилок відключення під час запису з перевантаженим ЦП, аналіз спектру звуку.

Adobe Audition − це програма для редагування аудіо та мікшування звуку, яка дозволяє редагувати та застосовувати ефекти до аудіо з відеоматеріалів. *Audition* включає в себе аудіо-доріжку, форму сигналу та спектральний дисплей для створення, мікшування, редагування та відновлення звукового вмісту. *Adobe Audition* розроблено, щоб прискорити робочі процеси виробництва відео та обробки аудіо. Цю програму буде корисно використовувати для: запису аудіо для лекцій або демонстрацій, редагування існуючого аудіо, запису аудіо для віртуальних зустрічей. З особливостей *Adobe Audition* можна виділити:

− запис, редагування аудіо;

− створення звуку професійної якості;

− створення подкастів;

− можливість створювати нове аудіо на базі поєднання інших.

Vocaroo − це програма, що дозволяє записувати аудіо за допомогою мікрофону користувача. Треба просто натиснути кнопку ‘Запис’, дозволити додатку використовувати мікрофон і проговорити слова. Після цього користувач отримає аудіо-файл, який зможе відтворити за допомогою плеєру. Ця програма буде корисною для ведення голосових щоденників, запису голосу для передачі іншим користувачам.

Також існують сервіси, що перетворюють аудіо у текст (*speech-to-text services*). За допомогою таких сервісів, користувачі можуть диктувати якусь інформацію, яка потім буде збережена у вигляді тексту. Одним з прикладів такого програмного забезпечення є сервіс *Diction.io*. За допомогою цього сервісу, користувачі можуть створювати документи, диктувати інформацію голосом, а сервіс буде переводити цю інформацію у текст, та записувати його у документ. Сервіс підтримує відображення тексту у вигляді сторінок, форматування тексту, збереження тексту.

1.3. Висновки до розділу

На сьогоднішній день існує багато варіантів програмного забезпечення для пошуку медіа-контенту. Серед програмного забезпечення для пошуку медіа-контенту можна знайти додатки для пошуку фільмів, зображень, музики, книжок, відео, тощо, наприклад: *IMDb, Spotify, Google Images, Google Books, Youtube*. Але не все таке програмне забезпечення є доступним.

Доступне програмне забезпечення призначене для роботи з людьми, які мають проблеми зору, слуху, моторики, і які можуть отримати доступ до програмного забезпечення за допомогою допоміжного пристрою. Допоміжне програмне забезпечення може включати: програми зчитування з екрана (читачі

екрану), клавіатури Брайля або програмне забезпечення для збільшення екрана (екранна лупа), програмне забезпечення для розпізнавання голосу, яке допомагає людям з обмеженими можливостями пересування орієнтуватися в Інтернеті та вводити текст, використовуючи лише свій голос.

Найпопулярнішим видом аудіо інформації сьогодні є музика. Для її пошуку існує багато програм. Однією з відомих платформ є *Spotify*. Це постачальник потокових аудіо, заснований 23 квітня 2006 року. Цей сервіс використовується як сервіс для розповсюдження музики, але також має потужні інструменти пошуку. У *Spotify* музику можна шукати за різними параметрами, наприклад, виконавцем, альбомом, жанром, списком відтворення або лейблом звукозапису. Користувачі можуть створювати, редагувати та ділитися списками відтворення, ділитися треками в соціальних мережах та створювати списки відтворення з іншими користувачами. *Spotify* надає доступ до понад 70 мільйонів пісень, 2,2 мільйонів подкастів і 4 мільярдів списків відтворення.

Програмне забезпечення для обробки аудіо має сьогодні багато реалізацій. Таке програмне забезпечення допомагає користувачам створювати нове аудіо, змінювати існуюче, покращувати якість аудіо, створювати записи, тощо, наприклад: *Audacity, Adobe Audition, Vocaroo.* Також користувачі можуть використовувати сервіси, які аналізують мовлення, для того щоб зберігати голосову інформацію у вигляді тексту. Одним з прикладів такого ПЗ є додаток *Diction.io*.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ВЕБ-ДОДАТКУ ПОШУКУ МЕДІА-КОНТЕНТУ ЗА ГОЛОСОМ

2.1 Проектування функцій веб-додатку.

При проектуванні веб-додатку пошуку медіа-контенту за голосом було визначено наступні функції:

- Пошук фільмів;
- Пошук композицій музики;
- Пошук книг;
- Реєстрація та авторизація;
- Можливість додавання типу медіа-контенту (фільм, музику або книгу) до “Обраного”.

Важливою вимогою до пошуку є те, що користувач повинен мати змогу створювати пошуковий запит за допомогою голосу, а не тільки за допомогою клавіатури. Тобто, додаток може перевести голосове повідомлення користувача у текстовий запит пошуку. Додаток дозволяє користувачу обирати мову введення пошукового запиту: англійську або українську.

Користувач може обирати між типом пошуку медіа-контенту. Після того, як користувач диктує або друкує пошуковий запит і здійснює пошук, додаток видає результат у вигляді списку. Кожен елемент списку вміщає в себе інформацію про тип медіа-контенту (наприклад: назву, рік видання, і т.д).

Також, користувач може додати тип медіа контенту в “Обране” для того, щоб мати до нього швидкий доступ. Для цього, додаток повинен надавати користувачу можливість створювати акаунт у системі.

Створення акаунту у системі проводиться за допомогою такої інформації:

- Електронна пошта;
- Пароль;
- Ім’я.

Після надання цієї інформації, користувач зможе увійти систему використовуючи:

- Електронну пошту;
- Пароль.

Після входу в систему, користувачу надається можливість керувати “обраним” медіа-контентом, та отримувати швидкий доступ до нього, без виконання пошукових запитів.

При здійснені пошуку фільмів, додаток надає таку інформацію про кожен фільм: назва, тривалість, рік, зображення. Також, додаток надає доступ користувачу до трейлеру цього фільму.

При здійснені пошуку музики, додаток надає таку інформацію про кожну композицію: назва (за шаблоном:{виконавець} – {назва композиції}), зображення. Додаток надає доступ користувачу до прослуховування електронної версії цієї композиції.

При здійснені пошуку книг, додаток надає таку інформацію про кожну книгу: назва, кількість сторінок, дата публікації, зображення. Додаток надає доступ користувачу до безкоштовної електронної версії цієї книги.

2.2 **Проектування інтерфейсу**

Інтерфейс додатку є сучасним та відповідає стандартам розробки веб-додатків, а це означає, що він є:

- Адаптивним;
- Кросбраузерним;
- Доступним;

Адаптивний інтерфейс користувача − це інтерфейс користувача, який адаптується, тобто змінює свій макет і елементи до потреб користувача. Це означає, що додаток повинен адекватно працювати і виглядати на різних пристроях, які сьогодні можуть бути використані для перегляду веб-додатків, наприклад:

- Настільний комп'ютер;
- Ноутбук;
- Смартфон;
- Планшет.

Всі ці пристрої мають різні операційні системи, розміри екрану, але додаток працює для кожного з них. Переваги адаптивного інтерфейсу користувача полягають у його здатності відповідати потребам користувачів. Властивості адаптивного інтерфейсу дозволяють показувати лише релевантну інформацію на основі поточного користувача. Це створює менше плутанини для менш досвідчених користувачів і забезпечує легкий доступ до всієї системи. Залежно від завдання, адаптивний інтерфейс може підвищити стабільність системи [4].

Кросбраузерний інтерфейс – це інтерфейс, який не змінює свого вигляду при використанні у різних браузерах. Це означає, що інтерфейс повинен однаково виглядати та працювати при використанні у різноманітних браузерах, наприклад:

- *Google Chrome*;
- *Microsoft Edge*;
- *Safari*;
- *Mozilla Firefox*.

Кросбраузерність описує проблеми та стратегії, які забезпечують узгоджений вигляд додатків у якомога більшій кількості браузерів і платформ. Оскільки існує багато операційних систем і браузерів, спроба підтримки всіх з них стала серйозною проблемою для розробників інтерфейсу. Програмісти платформ зрозуміли важливість стандартизації використання та поведінки елементів веб-браузеру як засобу для полегшення кросбраузерної взаємодії для розробників інтерфейсу. Чим точніше програмісти дотримуються стандартів, тим

більшою мірою програми будуть вести себе передбачуваним чином під час запуску на створених ними платформах. Органи стандартизації, такі як *WHATWG* та *W3C*, надають чітко визначені специфікації щодо того, як мають бути реалізовані загальні функції програми, щоб допомогти забезпечити узгоджений досвід. Для тестування кросбраузерності додатків існують емулятори веб-браузерів. Емулятор веб-браузеру − програмне забезпечення, яке імітує функціональність популярних веб-браузерів, доступних на ринку. Наприклад, емулятор *Firefox* забезпечить такий самий вигляд і відчуття, що й фактичний браузер *Firefox*, коли на ньому відкривається веб-сайт. Емулятори браузера допоможуть перевірити, як веб-додаток виглядає і працює в різних браузерах [5].

Доступний інтерфейс – інтерфейс, який виконує вимоги веб-стандартів (*W3C Web Accessibility Initiative*) щодо доступності. Доступність означає, що веб-сайти, інструменти та технології розроблені так, щоб люди з обмеженими можливостями могли ними користуватися. Доступність веб-сайтів залежить від окремих компонентів, серед яких:

− вміст - інформація на веб-сторінці чи веб-додатку, включаючи природну інформацію (наприклад, текст, зображення та звуки) та код чи розмітку, що визначає структуру, презентацію;

− веб-браузери, медіаплеєри та інші «агенти користувача;

− інструменти оцінки - аналізатори веб-доступності, валідатори HTML, валідатори CSS тощо [6].

Однією з головних функцій додатку є створення пошукового запиту через голос, для того, щоб люди без клавіатури, або люди з поганим зором могли використовувати інтерфейс, тобто додаток є доступним.

Інтерфейс додатку представляє систему з декількох сторінок:

− ‘/’ – головна сторінка;

− ‘/*movies’* – сторінка для пошуку інформації про фільми;

− ‘/*books’* – сторінка для пошуку інформації про книги;

− ‘/*music’* – сторінка для пошуку інформації про музику;

− ‘/*sign-up’* – сторінка для реєстрації акаунту користувача;

- *'/sign-in'* – сторінка для входу в акаунт користувача;
- *'/favorites'* – сторінка для перегляду "обраного" медіа-контенту.

Важливою частиною інтерфейсу є бічна панель – панель, яка містить навігаційне меню, за допомогою якого користувач зможе переходити між сторінками. На сторінках *'/movies'*, *'/books'*, *'/music'* наявна форма, за її допомогою користувач зможе написати запит, на основі якого буде здійснено пошук певного типу медіа-контенту (в залежності від сторінки). Головною особливістю цієї форми є те, що користувач зможе вводити дані не тільки з клавіатури, але й за допомогою голосу. Для цього у формі наявна кнопка, яка активізує введення пошукового запиту за допомогою голосу. Після натискання на кнопку активації введення пошукового запиту за допомогою голосу додаток відображає спливаюче меню, у якому можна обрати мову запиту. Доступні мови для диктування пошукового запиту відповідно такі: англійська (за замовчуванням) та українська. Після обирання мови, користувач натискає кнопку 'Говорити' і починається процес аналізу мовлення користувача. Після введення інформації голосом користувач зможе передивитеся те, що він надиктував у полі введення інформації, і при потребі відредагувати інформацію з клавіатури або ж знову натиснути кнопку та проговорити запит ще раз. Після відправлення форми з пошуковим запитом додаток відправить *HTTP* запит на отримання інформації, і система надасть результати пошуку.

Після отримання результатів пошуку система буде відображати результат у вигляді горизонтального списку блоків, які містять інформацію про тип медіа-контенту (книги, фільму або композиції).

У випадку пошуку фільмів, користувач побачить список блоків, де кожен блок буде містити відповідно вимогам:

- Назву;
- Тривалість;
- Рік;
- Зображення.

Також при натисканні на зображення користувач переходить на сторінку цього фільму у додатку *IMDb* і подивитися трейлер цього фільму.

У випадку пошуку книг, користувач побачить список блоків, де кожен блок буде містити:

- Назву;
- Кількість сторінок;
- Дату публікації;
- Зображення.

При натисканні на зображення користувач зможе перейти на сторінку книги у додатку *Google Books*, де отримає доступ до безкоштовної електронної версії книги.

У випадку пошуку композиції, користувач побачить список блоків, де кожен блок буде містити:

- Назву;
- Зображення.

При натисканні на зображення користувач зможе перейти на сторінку композиції у додатку *Spotify* і прослухати композицію.

Важливим моментом інтерфейсу є порядок відображення елементів у результатах пошуку: для широких екранів він буде горизонтальним, але якщо рядок з елементів не вміщується у ширину екрану користувача, елементи будуть переноситись на наступний рядок. Максимальна кількість елементів в одному рядку – 4, мінімальна – 1. Також, якщо кількість рядків за своєю висотою не вміщується в екран користувача, система буде відображати вертикальну полосу прокрутки, за допомогою якої користувач зможе прокрутити сторінку нижче.

Також дуже важливою функцію додатку є можливість додати медіа-файл у “обране”. Тобто, користувач може зберегти медіа-контент для себе у додатку, перейти на сторінку *‘/favorites’* та побачити його. Збережений медіа-контент відображається у вигляді вкладок:

- Фільми (включена вкладка за замовчуванням);
- Книги;

– Музика.

При активації однієї з вкладок, користувач бачить відповідний збережений медіа-контент у вигляді аналогічного списку, що використовується для відображання результатів пошуку. Але для цього користувач повинен зайти в існуючий акаунт, або ж створити його.

Для того щоб створити акаунт, користувач повинен зайти на сторінку *'/sign-up'* і заповнити форму. Форма вимагає заповнення таких даних:

– Пошта;

– Ім'я;

– Пароль.

Після введення цієї інформації, система відправить запит до сервісу і створить акаунт користувача. Для того щоб зайти в цей акаунт, користувач повинен перейти на сторінку */sign-in*, ввести форму, яка вимагає такі дані:

– Пошта;

– Пароль.

Після успішного введення цих даних додаток надсилає форму і авторизує користувача.

2.3 **Особливості проектування додатку для пошуку за голосом**

Головною особливістю проектування додатків для пошуку за голосом є наявність підтримки обробки голосу на платформі, що буде використовуватись для запуску додатку. Не всі платформи підтримують обробку голосу як готове рішення. Це означає, що розробники при проектуванні таких додатків можуть стикнутися зі складнощами реалізації цієї особливості. Наприклад, веб-платформа надає цю особливість, як готове рішення, пропонуючи розробнику інтерфейс мови програмування, який він може використовувати при написанні програми для браузеру. Цей інтерфейс називається *Web Speech API,* він є доволі простим, підтримується у багатьох сучасних браузерах. Все про що повинен думати розробник – це правильно викликати методи цього інтерфейсу. Для доступу до

мікрофону користувача, розробник повинен просто викликати метод цього інтерфейсу, після того як браузер виконає цю інструкцію, він запросить у користувача, чи дозволяє він надати доступ до мікрофону, і якщо користувач погоджується, браузер починає аналізувати голос користувача, і програма може отримати доступ до текстового формату мови користувача. Також важливим моментом є встановлення мови спілкування, адже якщо її не встановити, браузер не зможе виконати аналіз того, що говорить користувач. Можна зробити висновок, що опрацювання голосу при проектуванні додатків для сучасних платформ виконується в декілька етапів, які полягають в наступному

− отримання доступу до мікрофону;

− аналіз мовлення користувача;

− перетворення голосового представлення мовлення у текстове представлення.

Розробник при проектуванні системи з обробкою голосу почне стикатися з проблемами тоді, коли платформа, на якій буде запускатися додаток не буде надавати обробку голосу, як готове рішення. Це буде означати, що розробник повинен самостійно розробляти алгоритми для аналізу мовлення користувача, та алгоритми транслювання мовлення у текстове представлення, писати фрагменти коду для отримання доступу до мікрофону.

Для розпізнавання мовлення в тексті та підвищення точності транскрипції використовуються різні алгоритми та методи обчислення. Найбільш часто використовувані методи: обробка природної мови (*NLP*), приховані марковські моделі (*HMM*), *N*-грами, нейронні мережі, діаризація спікера.

Обробка природної мови − це область штучного інтелекту, яка зосереджується на взаємодії між людьми та машинами через мовлення та текст. Багато мобільних пристроїв включають у свої системи розпізнавання мовлення для здійснення голосового пошуку.

Прихована марківська модель − модель ланцюга Маркова, яка передбачає, що ймовірність наданого стану залежить від поточного стану, а не його попередніх станів. Хоча модель ланцюга Маркова корисна для подій, що

спостерігається, таких як введення тексту, приховані моделі Маркова дозволяють включати приховані події, такі як теги частини мови, в імовірнісну модель. Вони використовуються як моделі послідовності в рамках розпізнавання мовлення, призначаючи мітки кожному блоку: словам, складам, реченням, тощо. Ці мітки створюють зіставлення з наданим вхідним кодом, що дозволяє визначити найбільш відповідну послідовність міток. *N*-грама - це найпростіший тип моделі мовлення (*LM*), який присвоює ймовірності реченням або фразам.

N-грама − це найпростіший тип моделі мовлення (*LM*), який присвоює ймовірності реченням або фразам. Також *N*-граму можна описати як послідовність *N*-слів. Наприклад, «НАУ вище неба» — це триграма або 3 грами, а «університет НАУ вище неба» — 4 грами. Граматика та ймовірність певних послідовностей слів використовуються для покращення розпізнавання та точності.

Нейронні мережі в основному використовуються для алгоритмів глибокого навчання, обробляють навчальні дані, імітуючи взаємозв’язок людського мозку через вузли. Кожен вузол складається з входів, ваг, зміщення і виходу. Якщо це вихідне значення перевищує заданий поріг, він «запускає» або активує вузол, передаючи дані наступному вузлу в мережі. Нейронні мережі вивчають функцію аналізу мовлення за допомогою контрольованого навчання, коригуючи на основі функції втрат через процес градієнтного спуску. Хоча нейронні мережі, як правило, більш точні і можуть приймати більше даних, це пов’язано з ефективністю продуктивності, оскільки вони, як правило, повільніше навчаються в порівнянні з традиційними моделями мовлення.

Діаризація спікеру − алгоритми діаризації спікера ідентифікують і сегментують мовлення за ідентифікатором мовця. Це допомагає програмам краще розрізняти людей у розмові, і такі алгоритми часто застосовується у системах кол-центрів, які завдяки цьому можуть відрізняють клієнтів [7].

2.4 **Висновки до розділу**

При проектуванні додатку пошуку медіа-контенту за голосом було визначено головні функції і описано проектування інтерфейсу додатку. Додаток дозволяє користувачам здійснювати пошук медіа-контенту різного типу: фільмів, книжок, музики. Однією з головних особливостей додатку є можливість пошуку медіа-контенту за допомогою голосу. Тобто, додаток може перевести голосове повідомлення користувача у текстовий запит пошуку. Додаток повинен дозволяти користувачу обирати мову введення пошукового запиту: англійську або українську. Також додаток надає користувачу можливість створити аккаунт у системі, а потім увійти до нього. Це дозволить користувачу зберігати медіа-контенту різного типу, додаючи його до “обраного”. Після збереження медіа-контенту користувач зможе передивитись його на спеціальній сторінці (*/favorites).* Таким чином користувачі можуть отримувати доступ до медіа-контенту без здійснення пошуку.

Інтерфейс додатку було спроектовано таким чином, щоб він був кросбраузерним, адаптивним та доступним. Кросбраузерність стосується проблем та стратегій, які забезпечують узгоджений вигляд додатків у якомога більшій кількості браузерів і платформ. Оскільки існує багато операційних систем і браузерів, спроба підтримки всіх з них стала серйозною проблемою для розробників інтерфейсу. Адаптивний інтерфейс користувача − це інтерфейс користувача, який адаптується, тобто змінює свій макет і елементи до потреб користувача. Це означає, що додаток повинен адекватно працювати і виглядати на різних пристроях, які сьогодні можуть бути використані для перегляду веб-додатків. Доступний інтерфейс – інтерфейс, який відповідає вимогам веб-стандартів (*W3C Web Accessibility Initiative*) щодо доступності.

Головною особливістю проектування додатків для пошуку за голосом є наявність підтримки обробки голосу на платформі, що буде використовуватись для запуску додатку. Не всі платформи підтримують обробку голосу як готове

рішення. Це означає, що розробники при проектуванні таких додатків можуть стикнутися зі складнощами реалізації цієї особливості. Наприклад, веб-платформа надає цю особливість, як готове рішення, пропонуючи розробнику інтерфейс мови програмування, який він може використовувати при написанні програми для браузеру. Цей інтерфейс називається *Web Speech API*, він є доволі простим, підтримується у багатьох сучасних браузерах. Все про що повинен думати розробник – це правильно викликати методи цього інтерфейсу. Для доступу до мікрофону користувача, розробник повинен просто викликати метод цього інтерфейсу, після того як браузер виконає цю інструкцію, він запросить у користувача, чи дозволяє він надати доступ до мікрофону, і якщо користувач погоджується, браузер починає аналізувати голос користувача, і програма може отримати доступ до текстового формату мови користувача. Також важливим моментом є встановлення мови спілкування, адже якщо її не встановити, браузер не зможе виконати аналіз того, що говорить користувач.

Додатки, що мають особливість пошуку за голосом дозволяють користувачам ефективно шукати інформацію, але такі додатки мають складнощі з підтримкою платформ, що використовуються для запуску додатку. Розробник при проектуванні системи з обробкою голосу почне стикатися з проблемами тоді, коли платформа, на якій буде запускатися додаток, не буде надавати обробку голосу, як готове рішення. Це буде означати, що розробник повинен самостійно розробляти алгоритми для аналізу мовлення користувача, та алгоритми транслювання мовлення у текстове представлення, писати фрагменти коду для отримання доступу до мікрофону. Для розпізнавання мовлення в тексті та підвищення точності транскрипції використовуються різні алгоритми та методи обчислення. Найбільш часто використовувані підходи: обробка природної мови (*NLP*), приховані марковські моделі (*HMM*), *N*-грами, нейронні мережі, діаризація спікера.

РОЗДІЛ 3
РОЗРОБКА ВЕБ-ДОДАТКУ ПОШУКУ МЕДІА-КОНТЕНТУ ЗА ГОЛОСОМ

3.1 Налаштування середовища розробки та файлів проектів

Для розробки веб-додатку пошуку медіа-контенту за голосом спочатку встановлено відповідне середовище розробки, а також згенерувати проект с заздалегідь налаштованими бібліотеками, фреймворком і інструментами. Для реалізації додатку було обрано: мову програмування *JavaScript*, фреймворк для будування інтерфейсу *Vue.js*; базу даних *Firebase Realtime Database*; бібліотеку *Firebase Auth* для авторизації користувачів; бібліотеку програмних засобів для роботи з базою даних про фільми *IMDb API* (*Internet Movie Database Application Programming Interface*); бібліотеку сервісу *Spotify API* для отримання інформації про музику; бібліотеку *Google Books API* для отримання інформації про книги; мову розмітки *HTML* для створення структури веб-додатку; мову *CSS* для стилізації проекту; бібліотеку *Element-plus* для отримання готових компонентів інтерфейсу; бібліотеку *Axios* для отримання інформації за протоколом *HTTP*; бібліотеку *Speech Recognition API* для обробки голосу користувача; програмне забезпечення *Vite* для генерації, збірки, локальної розробки проекту; пакетний менеджер *Yarn* для запуску та завантаження пакетів додатку; фреймворк *Pinia* для керування глобальним станом додатку; бібліотеку *Vue Query* для управлінням асинхронними операціями; бібліотеку *Vue Router* для керування навігацією; бібліотеку *Tailwind CSS* – для спрощення стилізації проекту; бібліотеку *PostCSS* – для компіляції *.css* файлів додатку з *JavaScript* коду; *Visual Studio Code* – як інтегроване середовище розробки та редактор коду; програму *Eslint* для лінтингу коду додатку; програму *Prettier* – для форматування коду додатку.

JavaScript – мова програмування з динамічною системою типізації, використовується для створення динамічних веб-додатків у браузерах, а також для написання веб-серверів за допомогою *Node.js*, написання додатків для робочого стола за допомогою *Electron*. Мова *JavaScript* є однією з найпопулярніших мов сьогодні. Наразі *JavaScript* є стандартною мовою програмування для веб-браузерів, і кожен веб-браузер сьогодні має у своїй структурі движок для запуску *JavaScript* програм [8].

Vue.js – фреймворк для будування інтерактивних веб-додатків. Написаний на мові *JavaScript*. *Vue.js* абстрагує розробників від керування імперативним *DOM API* браузеру, і концентрує увагу на керуванні станом та даними додатку. Vue.js є дуже корисним у будуванні малих веб-додатків, і дозволяє бути інтегрованим на *HTML* сторінку як стороння бібліотека, а також може бути використаним для будування великих додатків за допомогою *Vue CLI* [9].

Firebase Realtime Database – нереляційна база даних, що має змогу працювати у режимі реального часу, тобто за допомогою цієї бази даних можна створювати додатки, що потребують миттєвого оновлення інтерфейсу, наприклад онлайн-чати. Дані у *Firebase Realtime Database* зберігаються у форматі *JSON*. База даних має реалізацію *API* для багатьох платформ:

- *Web* (*JavaScript*);
- *Android*;
- *iOS*;
- *Electron*.

Firebase Auth – сервіс авторизації, що може бути інтегрований для різних платформ для створення системи користувачів. *Firebase Auth* надає змогу розробникам не створювати свою систему авторизації користувачів, а використати вже готове рішення. Сервіс надає змогу авторизації користувачів як

за стандартним шляхом: електронна пошта і пароль, але й за допомогою сторонніх сервісів, наприклад: *Google, Facebook, Apple* та інших. [10]

IMDb API – сервіс для отримання інформації про фільми. Сервіс базується на базі даних додатку *IMDb*, що є одним з найпопулярніших веб-додатків з надання інформації про фільми. Наразі підтримуються методи для отримання інформації про фільми за ключовими словами, отримання інформації про конкретний фільм , отримання інформації про акторів.

Spotify API – сервіс для отримання інформації про музику. Сервіс базується на базі даних додатку *Spotify*, який є найпопулярнішим додатком для прослуховування та публікації музики. Підтримуються методи для отримання даних про музику за ключовими словами, отримання інформації про конкретну композицію, отримання інформації про виконавця або гурт.

Google Books API – сервіс для отримання інформації про книги. Сервіс базується на базі даних додатку *Google Books*, який є популярним засобом для читання книжок. Підтримуються методи для отримання даних про книги за ключовими словами, отримання інформації про конкретну книгу, отримання інформації про автора.

HTML – мова розмітки для структуризації веб-додатків. Ця мова є стандартом створення “скелету” веб-сторінок. Мова дозволяє описувати структуру у вигляді тегів. Розрізняють теги для візуального описання сторінки (*a, button, input)* , а також теги для описання мета-інформації додатку (*head, meta, link*) [11].

CSS – мова для стилізації веб-додатків. Це стандартна мова для опису стилів інтерфейсу веб-додатку. *CSS* дозволяє описувати стилі за допомогою системи селекторів. За допомогою селекторів розробник може описати елемент, який треба стилізувати, а потім описати стилі для цього елементу [12].

Element Plus – бібліотека готових компонентів для будування інтерфейсу веб-додатку. Написана на мові *JavaScript* з використанням фреймворку *Vue*.js. Бібліотека поставляє набір готових, стилізованих компонентів, таких як кнопки, поля для введення даних, форми, тощо.

Axios – бібліотека, яка представляю собою *HTTP* клієнт для передачі інформації за протоколом *HTTP*. Бібліотека написана на мові *JavaScript*.

Speech Recognition API – веб-*API*, що дозволяє опрацьовувати голос користувача, та переводити його у текстовий формат. Наразі цей програмний засіб підтримується у сучасних браузерах, та дозволяє програмному забезпеченню взаємодіяти з мікрофоном користувача. Є можливість обирати мову спілкування [13].

Vite – інструмент, який дозволяє згенерувати готові проекти, які базуються на мові JavaScript та таких фреймворках, як: *React.js, Vue.js, Svelte* та інші. *Vite* запроваджує автоматичну збірку проектів, а також веб-сервер для локальної розробки.

Yarn – менеджер пакетів для *JavaScript* додатків. За його допомогою розробники можуть керувати залежностями своїх додатків, створювати та встановлювати публічні або приватні пакети.

Pinia – бібліотека для управління глобальним станом *Vue.js* додатків. *Pinia* дозволяє створювати навколо компонентів додатку “шину даних”, за допомогою якої кожен компонент зможе отримати доступ до даних, або змінити ці дані.

Vue Query – бібліотека для управління асинхронними операціями та станом у *Vue.js* додатках. Ця бібліотека також вирішує проблеми кешування асинхронних даних.

Vue Router – бібліотека для управління навігацією у *Vue.js* додатках. Бібліотека дозволяє створювати сторінки та прикріпляти до них *URL* адреси, за допомогою яких користувач зможе отримати доступ до певної сторінки.

Tailwind CSS – бібліотека для стилізації додатків. Бібліотека полегшує процес стилізації, надаючи розробнику готові “утилітарні” *HTML* класи, які інкапсулюють *CSS* стилі. Розробник може конфігурувати ці стилі, за допомогою *Tailwind* конфігурації.

PostCSS – бібліотека для компіляції стилів за допомогою *JavaScript* плагінів. Плагіни можуть видаляти зайвий *CSS* код, реалізувати змінні та міксини в *CSS* і т.д.

Visual Studio Code – вбудоване середовище розробки та редактор коду для різних мов програмування. Підтримує встановлення різноманітних плагінів для розробки, є досить гнучким для налаштування.

Eslint – програма для лінтингу *JavaScript* програм, тобто ця програма аналізує іншу програму на наявні помилки.

Prettier – програма для форматування *JavaScript* програм. Підтримує налаштування через спеціальний конфігураційний файл. За допомогою *Prettier* розробник може не думати про такі речі, як: ставити ‘;’ в кінці кожної інструкції чи ні, скільки повинно бути пробілів перед кожним рядком, та інше. *Prettier* буде автоматично форматувати *JavaScript* програму замість розробника.

Додаток було згенеровану за допомогою *Vite*, для цього у термінал було введено команду: *yarn create vite web-dodatok-poshuku-media-contentu-za-golosom –template vue.* Після цього *Vite* згенерував додаток за шаблоном для *Vue.js* фреймворку.

Коренева файлова структура додатку виглядає так:

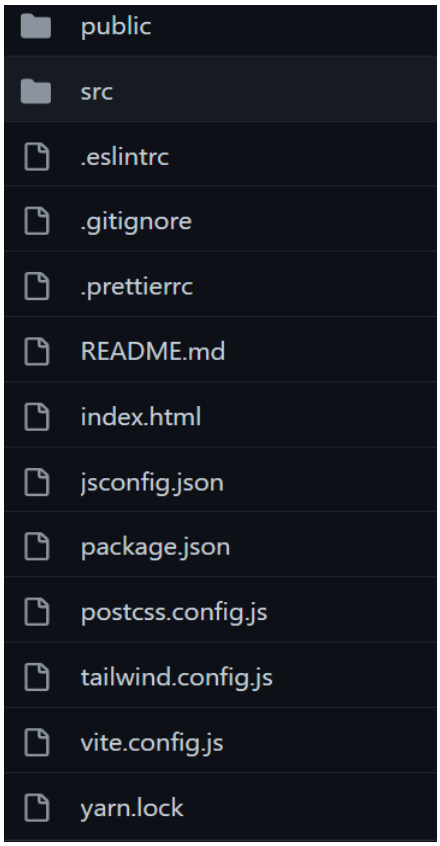

Рис 3.1 Файлова структура додатку

public – директорія, де зберігаються публічні файли для веб-браузеру, наприклад: *favicon.ico* (іконка веб-додатку, що відображається при його відкритті у веб-браузері).

src – директорія, в якій зберігаються вихідний код додатку (компоненти, сторінки, утиліти, сервіси та інше).

eslintrc - конфігураційний файл *Eslint*. За допомогою цього файлу, наприклад, можна встановлювати додаткові плагіни для *Eslint*, або ж додавати налаштування для правил *Eslint*.

prettierrc – конфігураційний файл *Prettier*. За допомогою цього файлу, можна, наприклад, встановити, чи додавати “;” в кінці кожної JavaScript інструкції, скільки повинно бути пробілів перед початком кожного рядку і т.д.

README.md – файл формату *Markdown*. Містить базову інструкції про додаток, як його запускати, збирати.

index.html – головний *HTML* файл додатку. За допомогою цього файлу можна змінювати мета інформацію додатку, наприклад: заголовок, опис, і т.д.

jsconfig.json – файл формату *JSON*, за допомогою якого описуються спеціальні налаштування додатку для *Visual Studio Code*.

package.json – файл, у якому описуються залежності додатку, їх назви та версії.

postcss.config.js – файл конфігурації *PostCSS*.

tailwind.config.js – файл конфігурації *TailwindCSS*.

vite.config.js – файл конфігурації *Vite*.

yarn.lock – автозгенерований файл пакетного менеджеру Yarn, який містить розширену інформацію про залежності.

Після генерації проекту у терміналі було виконано команду: *yarn install* для встановлення та налаштування обраних пакетів та бібліотек.

3.2 Розробка серверної частини

Серверна частина додатку потрібна для отримання даних про книги, музику, фільми, а також для реєстрації та авторизації користувачів, для збереження “обраного” медіа-контенту.

При реалізації серверної частини було виконано наступне: налаштовано систему аутентифікації *Firebase Auth,* створено проект у додатку *Firebase* , створено базу даних *Firebase Realtime Database* з такими сутностями: *movies, books, tracks.*

Для отримання інформації про книги, музику, фільми потрібний доступ до *Google Books API, IMDb API, Spotify API* відповідно. Ці *API* є публічними, тому все що потрібно це правильно виконувати *HTTP* запити до їх *URL* адрес.

Для реалізації реєстрації та авторизації користувачів було створено проект у додатку *Firebase*. Після створення проекту, у налаштуваннях було підключено сервіс *Firebase Auth*. З можливих варіантів реєстрації та авторизації було обрано тільки “за поштою та паролем”.

Для реалізації збереження медіа-контенту до проекту у *Firebase* було додано сервіс *Firebase Realtime Database*. Після цього, у базі даних було створено 3 сутності, у яких будуть зберігатися дані:

- *movies* – сутність, у якій будуть зберігатися фільми.
- *books* – сутність, у якій будуть зберігатися книги;
- *tracks* – сутність, у якій будуть зберігатися музикальні композиції.

3.3 Розробка інтерфейсу клієнтської частини

При реалізації клієнтської частини було виконано наступне: створено головний компонент *App.vue,* створено файл *firebase/index.js,* створено сторінку *SignUpPage.vue,* створено компонент *SignUpForm.vue,* створено сторінку *SignInPage.vue,* створено компонент *SignInForm.vue,* створено компонент *SearchInput.vue,* створено сторінку *MoviesPage.vue,* створено компонент *MoviesList.vue,* створено сторінку *BooksPage.vue,* створено компонент *BooksList.vue,* створено сторінку *MusicPage.vue,* створено компонент

TracksList.vue, створено сторінку *FavoritesPage.vue*, створено компонент *AddFavorite.vue.*

Для реалізації клієнтської частини, спочатку у директорії *src* було створено головний компонент *App.vue* – корінний компонент додатку. Він вміщує компоненти сторінок, які будуть перемикатися в залежності від поточної адреси навігації, а також макет, що містить: верхній блок зверху (*header)*, що містить назву проекту “Веб-додаток пошуку медіа-контенту за голосом”; навігаційне меню, що знаходиться зліва.

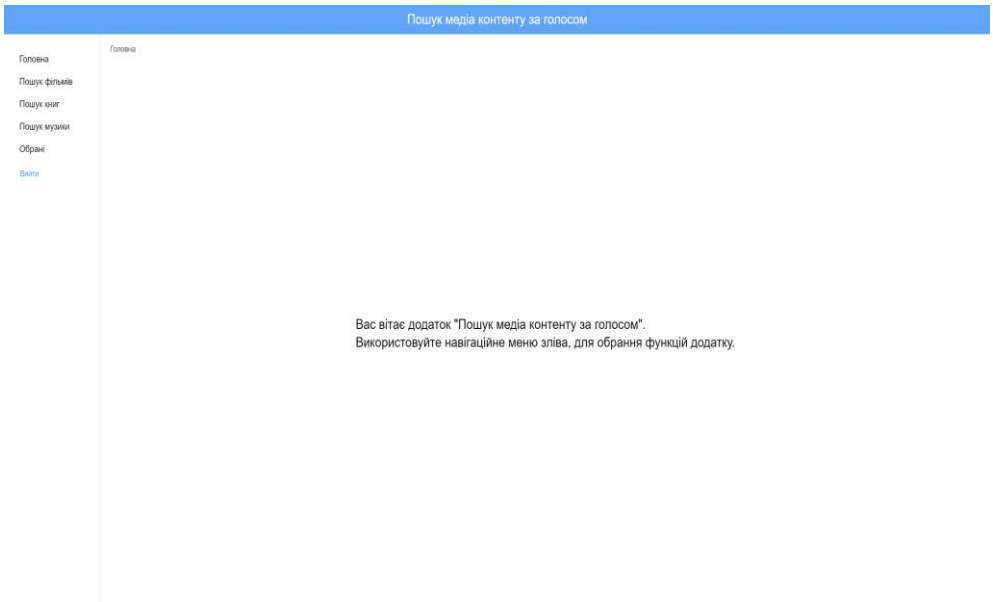

Рис 3.2 Головна сторінка веб-додатку пошуку медіа-контенту за голосом

Залежність певної сторінки від поточної *URL* адреси реалізована за допомогою *Vue Router* і описана у файлі *src/router.js*. Для здійснення запитів до *Firebase Realtime Database* & *Firebase Auth* було створено файл *firebase/index.js* де експортуються методи, що керують цими сервісами.

Для реалізації реєстрації було створено сторінку *SignUpPage.vue* (всі сторінки зберігаються у директорії pages), та прив’язано її до *UR*L адреси ‘/*sign-up*’. Потім було створено компонент *SignUpForm.vue* (всі компоненти зберігаються у директорії *components*), який було інтегровано в сторінку

SignUpPage. Таким чином сторінка *'/sign-up'* буде відображати форму з такими полями:

- Пошта;
- Ім'я;
- Пароль.

А також кнопку 'Зареєструватися' для відправки форми на сервер. Після натискання кнопки – додаток відправить *HTTP* запит до сервісу *Firebase Auth*, в свою чергу цей сервіс створить нового користувача.

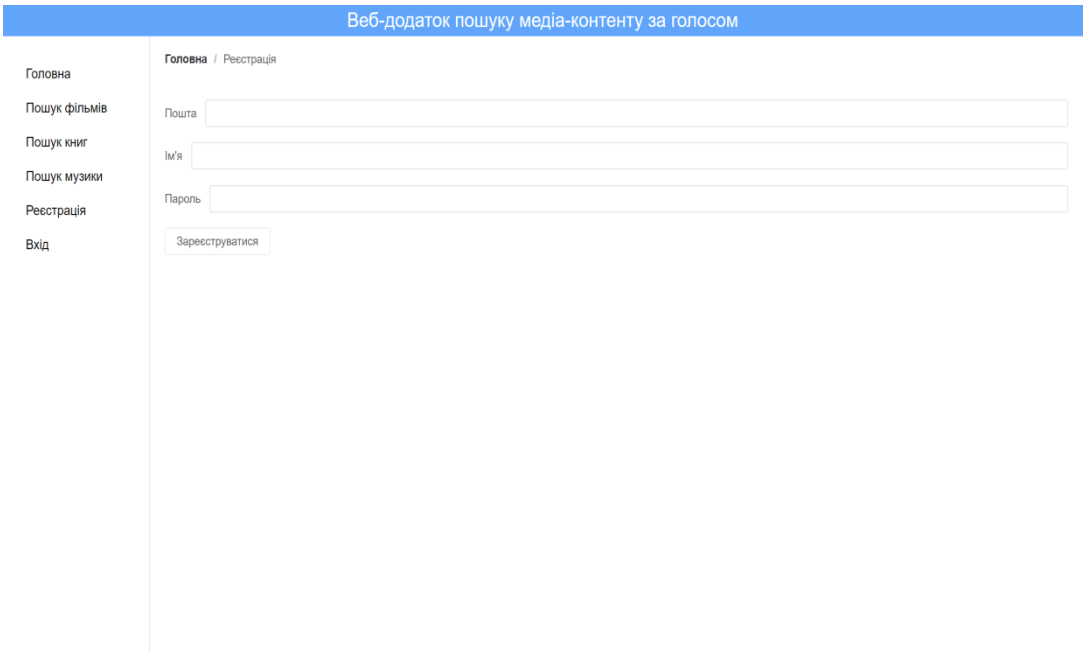

Рис 3.3 Сторінка реєстрації

Для реалізації авторизації було створено сторінку *SignInPage.vue*, та прив'язано її до *URL* адреси *'/sign-in'*. Потім було створено компонент *SignInForm.vue*, який було інтегровано в сторінку *SignUpPage.vue*. Таким чином сторінка '/sign-in' буде відображати форму з такими полями:

- Пошта;
- Пароль.

А також кнопку 'Вхід' для відправки форми на сервер. Після натискання кнопки – додаток відправить *HTTP* запит до сервісу *Firebase Auth*, після цього

сервіс авторизує користувача, і додаток збереже користувача у глобальне сховище даних *Pinia*, для того, щоб кожен компонент додатку міг отримувати доступ до інформації про користувача.

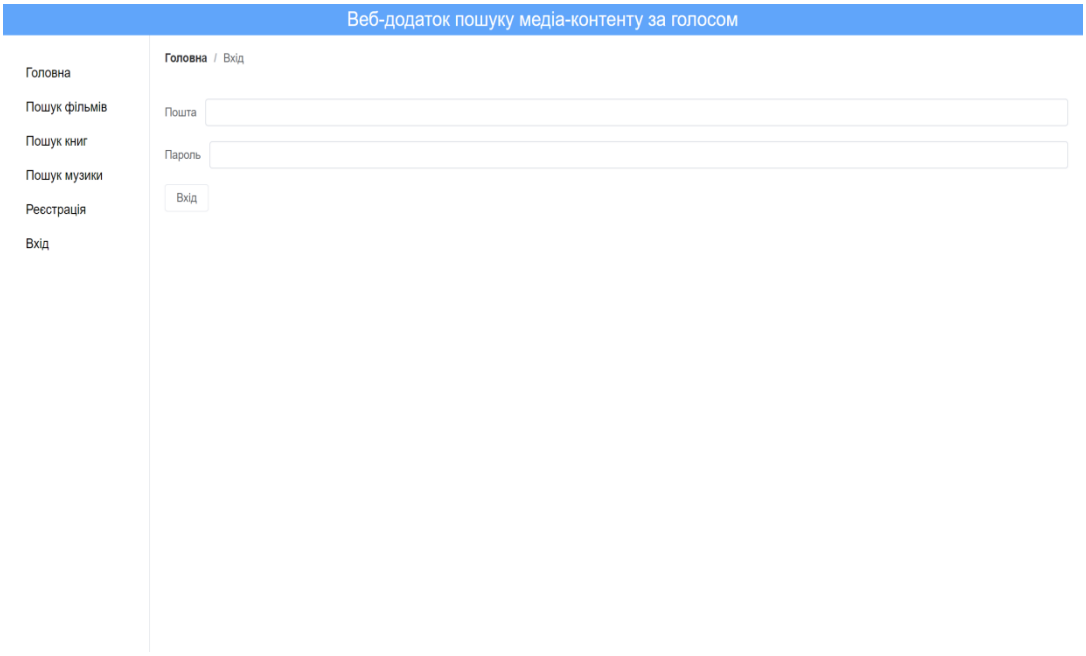

Рис 3.4 Сторінка авторизації

Для реалізації пошуку медіа-контенту було створено файл *SearchInput.vue*. Оскільки інтерфейс форми пошуку медіа-контенту однаковий для всіх типів, цей компонент використовується на сторінках для пошуку фільмів, книг і композицій музики. Компонент відображає форму з полем для вводу пошукового запиту, кнопкою ‘Пошук’ для здійснення пошукового запиту, а також кнопкою, яка відображає зображення мікрофону, за допомогою якої користувач зможе обрати мову та продиктувати пошуковий запит голосом. Після введення пошукового запиту та натиснення кнопки ‘Пошук’ компонент ініціює подію ‘*submit’*, яку інші компоненти зможуть перехопити, та на основі її виконати специфічну логіку.

```
const input = ref(‘’); // ініціалізація змінної пошукового запиту
function onInput(currentInput) {
    input.value = currentInput; // присвоєння змінної пошукового запиту
}
```

```
function onSearch() {
  if (!input.value) return;
  emit('search', input.value); // ініціалізація події 'submit'
}
<search-input class="mb-14" @search="onSearch" /> // приклад перехоплення
```
події '*submit*'

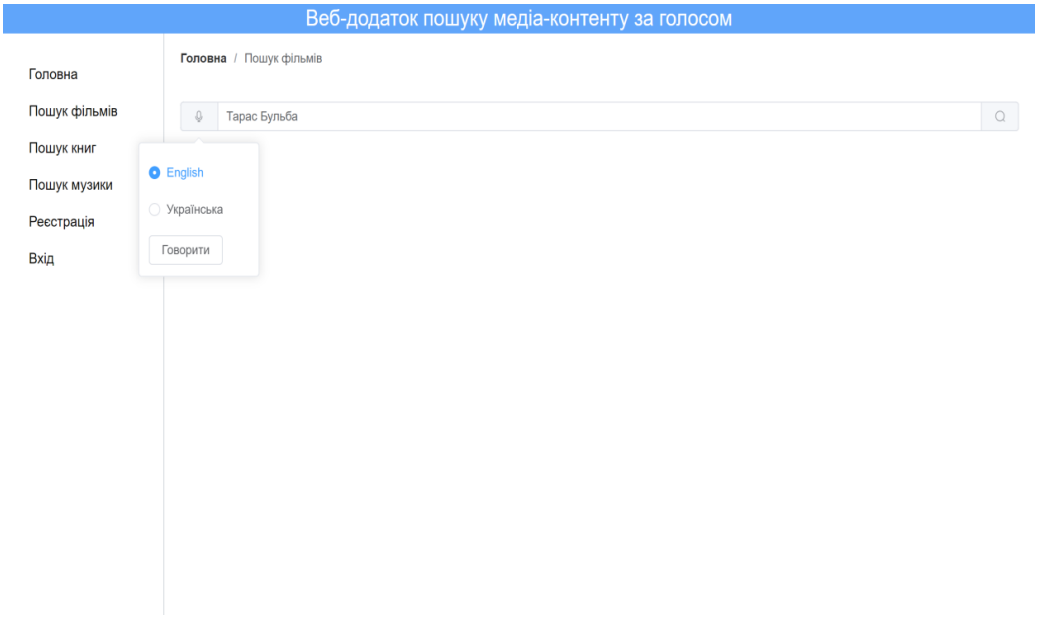

Рис 3.5 Приклад використання компоненту пошуку

Для реалізації пошуку фільмів було створено сторінку *MoviesPage.vue*, та прив'язано її до *URL* адреси *'/movies'*. Після цього в цю сторінку було інтегровано компонент *SearchInput.vue*, за допомогою якого користувач може зробити пошуковий запит. Після відправки форми пошукового запиту – компонент сторінки перехоплює подію *'submit'* і здійснює запит до *IMDb API*. Для відображення результату пошуку фільмів було створено компонент *MoviesList.vue*. Цей компонент відображає результат пошуку у вигляді списку блоків з інформацією про фільм: назва; тривалість; рік; зображення. Зображення обгорнуто у *HTML тег <a>* з посиланням на фільм у сервісі *IMDb*.

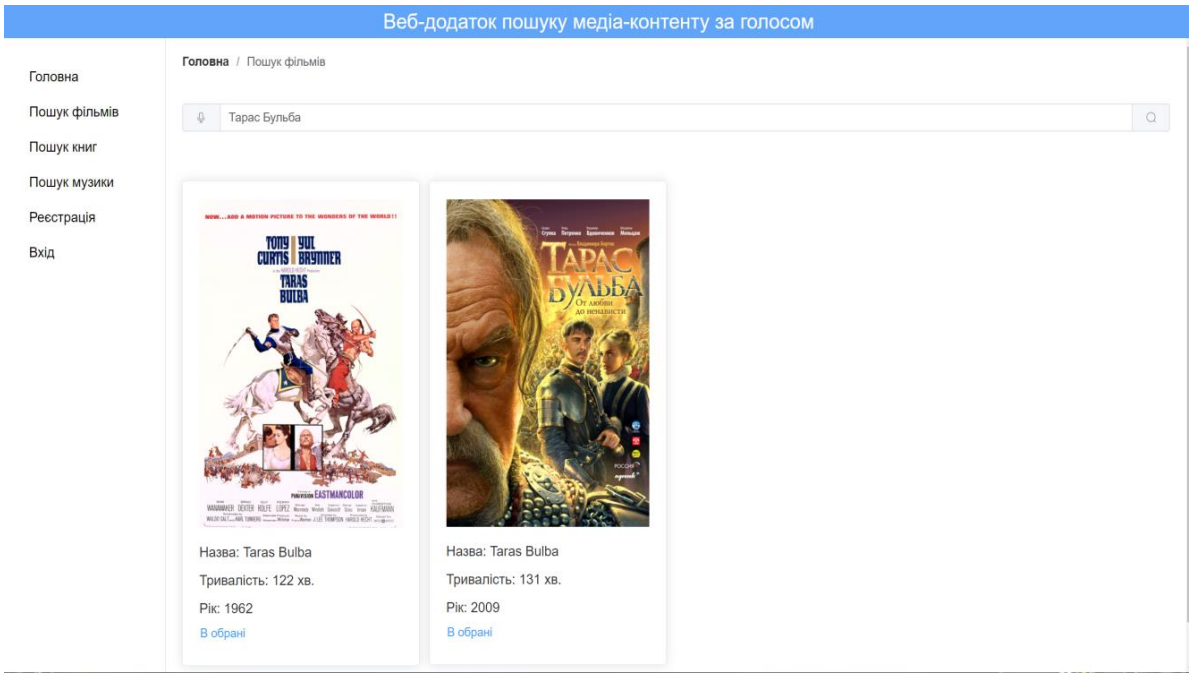

Рис 3.6 Сторінка пошуку фільмів веб-додатку пошуку медіа-контенту за голосом

Для реалізації пошуку книг було створено сторінку *BooksPage.vue*, та прив'язано її до *URL* адреси *'/books'*. Після цього в цю сторінку було інтегровано компонент *SearchInput.vue*. Після відправки форми пошукового запиту – компонент сторінки перехоплює подію *'submit'* і здійснює запит до *Google Books API*. Для відображення результату пошуку книг було створено компонент *BooksList.vue*. Цей компонент відображає результат пошуку у вигляді списку блоків з інформацією про книгу: назва; кількість сторінок; дата публікації; зображення. Зображення було обгорнуто у *HTML тег* <a> з посиланням на книгу у сервісі *Google Books*.

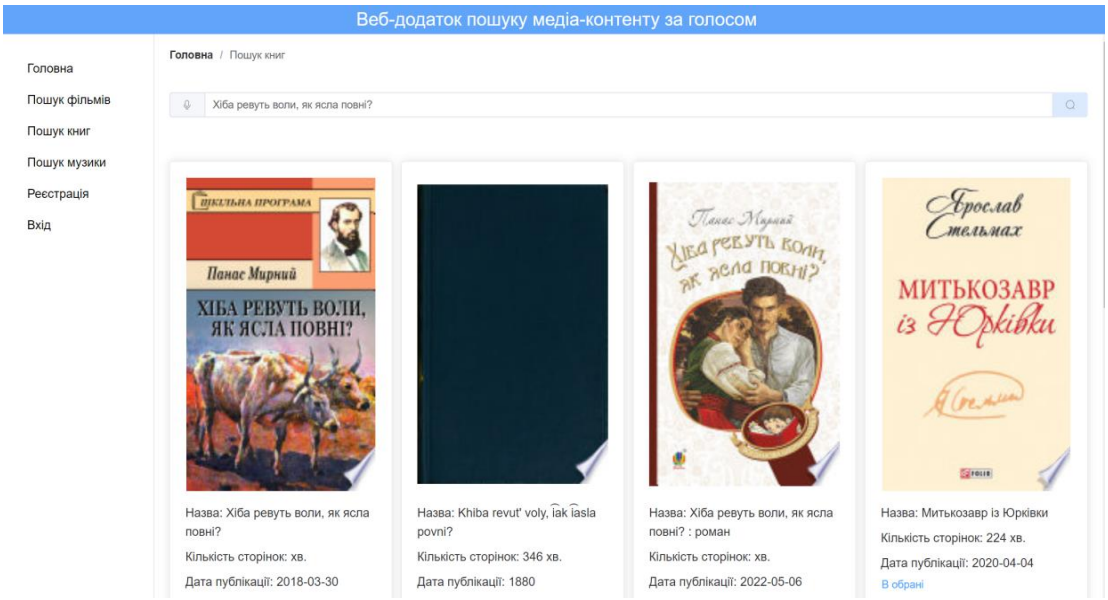

Рис 3.7 Сторінка пошуку книг веб-додатку пошуку медіа-контенту за голосом

Для реалізації пошуку музики було створено сторінку *TracksList.vue*, та прив'язано її до URL адреси *'/music*. Після цього в цю сторінку було інтегровано компонент *SearchInput.vue*. Після відправки форми пошукового запиту – компонент сторінки перехоплює подію *'submit'* і здійснює запит до *Spotify API*. Для відображення результату музикальних композицій було створено компонент *TracksList.vue*. Цей компонент відображає результат пошуку у вигляді списку блоків з інформацією про музикальну композицію: назва; зображення. Зображення було обгорнуто у *HTML тег <a>* з посиланням на книгу у сервісі *Spotify*.

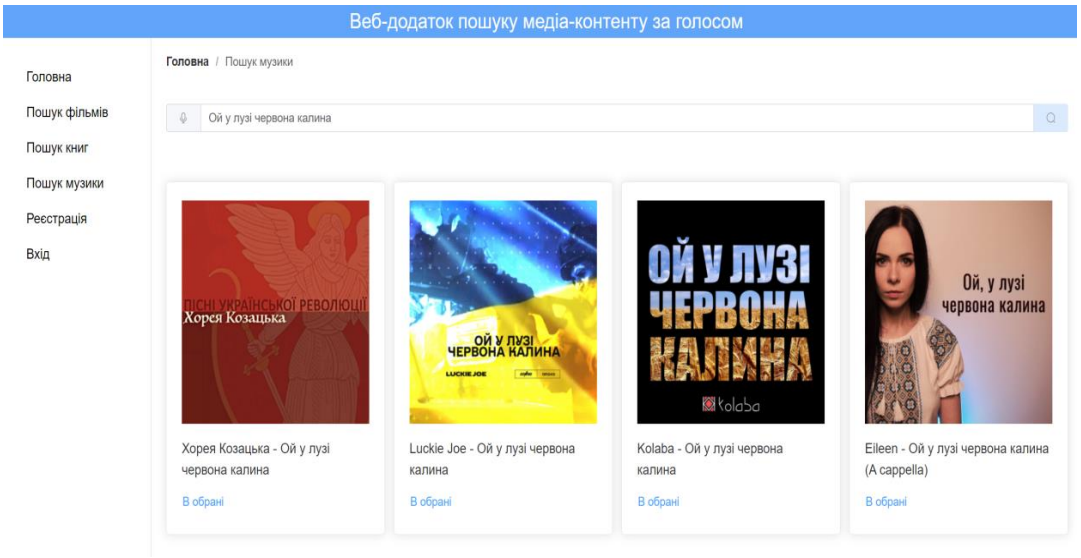

Рис 3.8 Сторінка пошуку музикальних композицій веб-додатку пошуку медіа-контенту за голосом

Для реалізації додавання медіа-контенту у ‘обране’ було створено компонент *‘AddFavorite.vue’*. Цей компонент інтегрується в кожен елемент списку результатів пошуку медіа-контенту. При натисканні кнопки ‘У обране’ відсилається запит до бази даних *Firebase Realtime Database* для оновлення відповідної сутності (в залежності від типу медіа-контенту). Також було створено сторінку *FavoritesPage.vue*, яку було прив’язано до *URL* адреси *‘/favorites’*. На цій сторінці користувач може переглянути список “обраного” медіа-контенту, переключаючи вкладки “Фільми”, “Книги”, “Музика”.

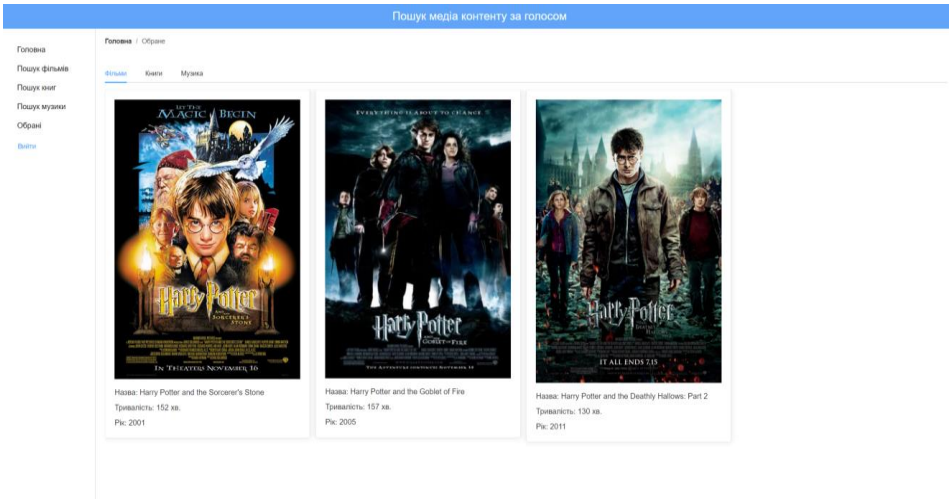

Рис 3.9 Сторінка ‘Обране’

3.4 Розробка модулів для роботи з голосом

Для реалізації аналізу мовлення користувача було створено спеціальний модуль *speech-service.js*. Сервіс експортує клас *SpeechRecognitionService*, у конструкторі якого створюється об'єкт *recognition*, що є екземпляром класу з *SpeechRecognition. SpeechRecognition* − це клас, який вбудований в інтерфейс *Web API* браузеру. Клас постачає різноманітні властивості, події та методи для контролю аналізу мовлення:

grammars – набір слів, які повинні розпізнаватися *Speech Recognition API.*

continuous – властивість, яка контролює, чи припиняти слухання після того, як користувач перестане говорити.

lang – встановлює мову для розпізнавання. Якщо ця властивість не вказана за замовчуванням, то використовується мова, що встановлена в браузері.

interimResults – властивість, яка зберігає проміжні результати аналізу, але не остаточні. Тобто це ті результати, які будуть зберігатися, доки користувач не закінчить розпізнавання його мовлення.

maxAlternatives – максимальна кількість альтернативних результатів аналізу мовлення. За замовчуванням дорівнює 1.

Також клас надає доступ до багатьох подій:

onaudiostart – подія, яка викликається, коли почався запис аудіо мовлення.

onaudioend – подія, яка викликається, коли закінчився запис аудіо мовлення.

onend – подія, яка викликається, коли процес розпізнавання мовлення закінчився.

onerror – подія, яка виклається, коли процес розпізнавання мовлення припинився помилкою.

onnomatch – подія, яка викликається, коли процес розпізнавання мовлення припинився без результату, тобто система не змогла розпізнати слова користувача.

onresult – подія, яка викликається, коли процес розпізнавання мовлення зміг розпізнати слова користувача.

onsoundstart – подія, яка викликається, коли система розпізнає будь-який звук, не обов'язково мовлення.

onsoundend – подія, яка викликається, коли система припинила розпізнавати будь-який звук.

onspeechstart – подія, яка викликається, коли система аналізує звук, який розпізнає як мовлення людини.

onspeechend – подія, яка викликається, коли система припиняє аналізувати звук, що був розпізнаний як мовлення людини.

Також клас *SpeechRecognition* надає доступ до таких методів: *abort, start, stop.*

abort – припиняє розпізнавання мовлення та не повертає результат розпізнавання.

start – запускає розпізнавання мовлення.

stop – закінчує розпізнавання мовлення, повертає результат розпізнавання.

Після створення об'єкту *recognition*, у конструкторі класу *SpeechRecognitionServic* властивості *recognition.interimResults* присвоюється *true.* Створений клас постачає один метод для аналізу мовлення користувача, що називається *recognize.* Цей метод приймає два параметри: *lang* – мову розпізнавання, а також параметр *callback,* який передає результат аналізу. При викликанні методу *recognize,* цей метод встановлює для властивості об'єкта *recognition.lang* значення параметра *lang,* встановлює значення події *onresult* рівною параметру *callback* і в кінці викликає метод об'єкту *recognition.start,* що починає розпізнавання мовлення. Цей клас використовується у згаданому вище компоненті *SearchInput.vue* для реалізації пошуку медіа-контенту за голосом.

3.5 Висновки до розділу

При розробці додатку пошуку медіа-контенту було встановлене інтегроване середовище розробки, обрано програмне забезпечення (фреймворк, бібліотеки та інструменти для розробки), згенеровано додаток, реалізовано всі спроектовані

функції. Для реалізації додатку було обрано: мову програмування *JavaScript*, фреймворк для будування інтерфейсу *Vue.js*; базу даних *Firebase Realtime Database*; *Firebase Auth* для авторизації користувачів; Бібліотека *IMDb API* для отримання інформації про фільми; *Spotify API* для отримання інформації про музику; *Google Books API* для отримання інформації про книги; мову розмітки *HTML* для створення структури веб-додатку; мову *CSS* для стилізації проекту; бібліотеку *Element-plus* для отримання готових компонентів інтерфейсу; бібліотеку *Axios* для отримання інформації за протоколом *HTTP*; *Speech Recognition API* для обробки голосу користувача; *Vite* для генерації, збірки, локальної розробки проекту; *Yarn* для запуску та завантаження пакетів додатку; *Pinia* для керування глобальним станом додатку; *Vue Query* для управлінням асинхронними операціями; *Vue Router* для керування навігацією; *Tailwind CSS* – для спрощення стилізації проекту; *PostCSS* – для компіляції *.css* файлів додатку з *JavaScript* коду; *Visual Studio Code* – як інтегроване середовище розробки та редактор коду; *Eslint* для лінтингу коду додатку; програму *Prettier* – для форматування коду додатку.

Для реалізації серверної частини додатку було використано сервіси *Firebase Auth і Firebase Realtime Database. Firebase Auth* було використано для реалізації реєстрації та авторизації користувачів. *Firebase Realtime Database* було використано для збереження медіа-контенту користувачів у розділі “обране”.

Для реалізації інтерфейсу було створено компоненти за допомогою фреймворку *Vue.js*: головну сторінку додатку *HomePage.vue*; сторінку *SearchInput.vue* – для реалізації форми пошуку; *MoviesPage.vue, MoviesList.vue* – для реалізації пошуку фільмів; *BooksPage.vue, BooksList.vue* – для реалізації пошуку книг; *MusicPage.vue, TracksList.vue* – для реалізації пошуку музики; *SignInPage.vue, SignInForm.vue, SignUpForm.vue, SignUpPage.vue* – для реалізації реєстрації та авторизації.

Проект було виконано з дотриманням діючих стандартів та положень [14], [15].

ВИСНОВКИ

На сьогоднішній день існує багато варіантів програмного забезпечення для пошуку медіа-контенту. Серед програмного забезпечення для пошуку медіа-контенту можна знайти додатки для пошуку фільмів, зображень, музики, книжок, відео, тощо, наприклад: *IMDb, Spotify, Google Images, Google Books, Youtube*. Але не все таке програмне забезпечення є доступним.

Доступне програмне забезпечення призначене для роботи з людьми, які мають проблеми зору, слуху, моторики, і які можуть отримати доступ до програмного забезпечення за допомогою допоміжного пристрою. Допоміжне програмне забезпечення може включати: програми зчитування з екрана (читачі екрану), клавіатури Брайля або програмне забезпечення для збільшення екрана (екранна лупа), програмне забезпечення для розпізнавання голосу, яке допомагає людям з обмеженими можливостями пересування орієнтуватися в Інтернеті та вводити текст, використовуючи лише свій голос.

Програмне забезпечення для обробки аудіо має сьогодні багато реалізацій. Таке програмне забезпечення допомагає користувачам створювати нове аудіо, змінювати існуюче, покращувати якість аудіо, створювати записи, тощо, наприклад: *Audacity, Adobe Audition, Vocaroo.* Також користувачі можуть використовувати сервіси, які аналізують мовлення, для того щоб зберігати голосову інформацію у вигляді тексту. Одним з прикладів такого ПЗ є додаток *Diction.io*.

При проектуванні додатку пошуку медіа-контенту за голосом було визначено головні функції і інтерфейс додатку. Додаток дозволяє користувачам здійснювати пошук медіа-контенту різного типу: фільми, книжки, музику. Однією з головних особливостей додатку є можливість пошуку медіа-контенту за допомогою голосу. Тобто, додаток може перевести голосове повідомлення користувача у текстовий запит пошуку. Додаток повинен дозволяти користувачу обирати мову введення пошукового запиту: англійську або українську. Також додаток надає користувачу можливість створити акаунт у системі, а потім увійти

до нього. Це дозволить користувачу зберігати медіа-контенту різного типу, додаючи його до “обраного”. Після збереження медіа-контенту користувач зможе передивитись його на спеціальній сторінці (*/favorites).* Таким чином користувачі отримують доступ до медіа-контенту без здійснення пошуку.

Інтерфейс додатку було спроектовано таким чином, щоб він був кроссбраузерним, адаптивним та доступним. Кросбраузерність описує проблеми та стратегії, які забезпечують узгоджений вигляд додатків у якомога більшій кількості браузерів і платформ. Оскільки існує багато операційних систем і браузерів, спроба підтримки всіх з них стала серйозною проблемою для розробників інтерфейсу. Адаптивний інтерфейс користувача − це інтерфейс користувача, який адаптується, тобто змінює свій макет і елементи до потреб користувача. Це означає, що додаток повинен адекватно працювати і виглядати на різних пристроях, які сьогодні можуть бути використані для перегляду веб-додатків. Доступний інтерфейс – інтерфейс, який виконує вимоги веб-стандартів (*W3C Web Accessibility Initiative*) щодо доступності.

Головною особливістю проектування додатків для пошуку за голосом є наявність підтримки обробки голосу на платформі, що буде використовуватись для запуску додатку. Не всі платформи підтримують обробку голосу як готове рішення. Це означає, що розробники при проектуванні таких додатків можуть стикнутися зі складнощами реалізації цієї особливості. Наприклад, веб-платформа надає цю особливість, як готове рішення, пропонуючи розробнику інтерфейс мови програмування, який він може використовувати при написанні програми для браузеру. Цей інтерфейс називається *Web Speech API*, він є доволі простим, підтримується у багатьох сучасних браузерах. Все про що повинен думати розробник – це правильно викликати методи цього інтерфейсу. Для доступу до мікрофону користувача, розробник повинен просто викликати метод цього інтерфейсу, після того як браузер виконає цю інструкцію, він запросить у користувача, чи дозволяє він надати доступ до мікрофону, і якщо користувач погоджується, браузер починає аналізувати голос користувача, і програма може отримати доступ до текстового формату мови користувача. Також важливим

моментом є встановлення мови спілкування, адже якщо її не встановити, браузер не зможе виконати аналіз того, що говорить користувач.

Додатки, що мають особливість пошуку за голосом дозволяють користувачам ефективно шукати інформацію, але такі додатки мають складнощі з підтримкою платформ, що використовуються для запуску додатку. Розробник при проектуванні системи з обробкою голосу почне стикатися з проблемами тоді, коли платформа, на якій буде запускатися додаток, не буде надавати обробку голосу, як готове рішення. Це буде означати, що розробник повинен самостійно розробляти алгоритми для аналізу мовлення користувача, та алгоритми транслювання мовлення у текстове представлення, писати фрагменти коду для отримання доступу до мікрофону. Для розпізнавання мовлення в тексті та підвищення точності транскрипції використовуються різні алгоритми та методи обчислення. Найбільш часто використовувані підходи: обробка природної мови (*NLP*), приховані марківські моделі (*HMM*), *N*-грами, нейронні мережі, діаризація спікера.

При розробці додатку пошуку медіа-контенту було встановлене інтегроване середовище розробки, обрано відповідне програмне забезпечення (фреймворк, бібліотеки та інструменти для розробки), згенеровано додаток, реалізовано всі спроектовані особливості.

Для реалізації серверної частини додатку було використано сервіси *Firebase Auth і Firebase Realtime Database. Firebase Auth* було використано для реалізації реєстрації та авторизації користувачів. *Firebase Realtime Database* було використано для збереження медіа-контенту користувачів у розділі “обране”.

Для реалізації інтерфейсу було створено компоненти за допомогою фреймворку *Vue.js*: головну сторінку додатку *HomePage.vue*; сторінку *SearchInput.vue* – для реалізації форми пошуку; *MoviesPage.vue, MoviesList.vue* – для реалізації пошуку фільмів; *BooksPage.vue, BooksList.vue* – для реалізації пошуку книг; *MusicPage.vue, TracksList.vue* – для реалізації пошуку музики; *SignInPage.vue, SignInForm.vue, SignUpForm.vue, SignUpPage.vue* – для реалізації реєстрації та авторизації.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Медіа-контент: види та особливості [Електронний ресурс] – Режим доступу: https://issuu.com/instituteofjournalusm/docs/mediacontent_book (дата звернення 20.05.2022)
2. What is assistive technology? [Електронний ресурс] – Режим доступу: https://www.washington.edu/accesscomputing/what-assistive-technology (дата звернення 20.05.2022)
3. Get the picture [Електронний ресурс] – Режим доступу: https://googleblog.blogspot.com/2005/02/get-picture.html (дата звернення 20.05.2022)
4. What is an Adaptive User Interface? [Електронний ресурс] – Режим доступу: https://elementor.com/resources/glossary/what-is-an-adaptive-user-interface/ (дата звернення 25.02.2022)
5. Daniel Herken. The Cross Browser Handbook. – Kindle Edition, 2014. – 109с.
6. Microsoft Corporation Engineering Software for Accessibility. – Microsoft Press, 2009. – 100с.
7. What is speech recognition? [Електронний ресурс] – Режим доступу: https://www.ibm.com/cloud/learn/speech-recognition (дата звернення 21.05.2022)
8. David Flanagan. JavaScript: The Definitive Guide. – O'Reilly Media, 2006. – 1032с.
9. Erik Hanchett. Vue.js in Action. – Manning, 2018. – 375с.
10. Houssem Yahiaoui. Firebase Cookbook: Over 70 recipes to help you create real-time web and mobile applications with Firebase. – Packt Publishing, 2017. – 288с.
11. Jon Duckett. HTML and CSS: Design and Build Websites. – Wiley, 2014. – 512с.
12. Eric Mayer. CSS: The Definitive Guide: Visual Presentation for the Web. – O'Reilly Media, 2017. – 1088с.

13. MDN Web Docs. Speech Recognition [Електронний ресурс] – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/API/SpeechRecognition (дата звернення 22.05.2022)
14. ДСТУ 3008-95. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення / Держстандарт України. – Вид. офіц. – [Чинний від 1995-02-23]. – Київ, 2007. – 86с.
15. Слободян О. Положення про дипломні роботи (проекти) випускників Національного авіаційного університету. – К.: Видавництво НАУ, 2017. – 63с.