Міністерство освіти і науки України
Університет митної справи та фінансів

Факультет інноваційних технологій
Кафедра комп'ютерних наук та інженерії програмного забезпечення

Кваліфікаційна робота бакалавра

на тему: «Розробка компаньйонського застосунку до Spotify для пошуку тексту відтворюваної пісні»

Виконав: студент групи К-20-2
Спеціальність 122 «Комп'ютерні науки»
Кабанов О. В.
Керівник: к.ф.-м.н., доц. Рудянова Т. М.
Рецензент: Університет митної справи та фінансів
(місце роботи)
доцент кафедри транспортних технологій та міжнародної логістики
(посада)
к.т.н., доц. Разгонов С. А.
(науковий ступінь, вчене звання, прізвище та ініціали)

Дніпро – 2024

АНОТАЦІЯ

Кабанов О.В. “Розробка компаньйонського застосунку до Spotify для пошуку тексту відтворюваної пісні”.

Дипломна робота на здобуття освітнього ступеня бакалавр за спеціальністю 122 «Комп’ютерні науки». – Університет митної справи та фінансів, Дніпро, 2023.

Дипломна робота спрямована на розробку програмного додатка, який дозволяє користувачу за двома основними алгоритмами здійснювати пошук текстів пісень: перший алгоритм дозволяє шукати лірику за власним текстовим запитом, другий алгоритм є алгоритмом автоматичного пошуку, що дозволяє за натиском однієї кнопки знаходити тексти пісень. Автопошук працює з піснями, що відтворюються у музичному онлайн-плеєрі Spotify. Обидва алгоритми надають вибір, це дозволяє користувачу зручно та швидко отримувати бажаний текст навіть при використанні різних онлайн-плеєрів з боку юзерів. Цей додаток працює як супутня програма для Spotify, що дозволяє значно розширити функціональність оригінального музичного сервісу та покращити взаємодію юзерів з наявною у ньому музикою.

Метою дипломної роботи є аналіз застосунків для пошуку текстів пісень та створення власного застосунку на основі проведених досліджень. Проєкт розроблено за використанням мови програмування Python за використанням необхідних бібліотек, що якісно змінили, як процес розробки програмного забезпечення, так і його використання.

У розроблювальному додатку відсутня серверна частина, але програма потребує постійного інтернет підключення, оскільки для пошуку тексту використовуються відкриті інтернет джерела. Також, для роботи алгоритму автоматичного пошуку, необхідно мати на комп’ютері додаток Spotify.

Ключові слова: програма пошуку пісень, алгоритми пошуку, додаток Spotify, мова програмування Python.

ABSTRACT

Kabanov O.V. "Development of a companion application to Spotify for searching the lyrics of the song being played".

Thesis for obtaining a bachelor's degree in the specialty 122 "Computer Science". – University of Customs and Finance, Dnipro, 2023.

The diploma is aimed at the development of a software application that will allow the user to search for song texts using two main algorithms. The first algorithm will allow you to search for lyrics based on your own text query. The second algorithm is an automatic search algorithm that allows you to find lyrics at the push of a button, the automatic search works with songs that are played in the Spotify online music player. Both algorithms provide a choice, this allows the user to conveniently and quickly receive the desired text even when using different online players from the side of users. This application works as a companion program for Spotify, which allows you to significantly expand the functionality of the original music service and improve the interaction of users with the music available in it.

The purpose of the thesis is the analysis of applications for finding song texts and creating your own application based on the conducted research. The project was developed using the Python programming language using the necessary libraries, which qualitatively changed both the software development process and its use.

The development application does not have a server part, but the program requires a constant Internet connection, since open Internet sources are used for text search. Also, for the automatic search algorithm to work, you need to have the Spotify application on your computer.

Keywords: lyrics search program, search algorithms, Spotify application, Python programming language.

ЗМІСТ

ВСТУП

На сьогоднішній день неможливо уявити життя без досягнень технологічного прогресу людства. З кожним днем життя людини становиться все більш нерозривним з електронними девайсами та їх функціоналом. Ми використовуємо гаджети для роботи, для навчання, приготування їжі, здійснення покупок, для розмов та розваг, тощо. Технології вже стали другою половиною нашого життя, у час, коли було досягнуто все це, єдиним шляхом розвитку технологій є поліпшення досвіду від їх використовування.

Актуальність обраної теми дипломної роботи полягає у розробці додатку для пошуку тексту пісень. Актуальність зумовлена тим, що на ринку програмних продуктів майже немає якісних desktop-додатків, які допомагають користувачам знаходити тексти улюблених композицій.

Під час визначення цілей розробки було вирішено, що створювана програма має бути додатком-компаньйоном до онлайн-сервісу Spotify, оскільки він є найпопулярнішим та найзручнішим для прослухування музики. Додаток-компаньйон повинен вирішувати проблематику основного продукту, а у нього є свої суттєві недоліки, такі як відсутність текстів для композицій дуже багатьох виконавців. Також нещодавно Spotify змінив політику своєї компанії, після чого навіть невелика кількість наявних текстів стали недоступні для юзерів без версії “Premium”.

Цей додаток, що є продуктом для даної дипломної роботи, вирішує ці проблеми. Програма дозволяє здійснювати пошук тексту пісень, при знанні назви пісні та імені виконавця, також є доступним автоматичний режим пошуку, при якому не треба здійснювати письмовий ввід необхідної інформації, а лише включити пісню у відповідній програмі.

Процес розробки вимагав знання у роботі різних сервісів та інтеграції їх у своєму продукті. Також робота вимагала різні знання, треба розумітися у музичних стримінгових сервісах, у онлайн ресурсах з базою даних пісень,

принципу їх роботи, плануванні процесу розробки та знання мов програмування.

Важливими етапами розробки були пошук плагінів для взаємодії програми з сайтом, що є онлайн-ресурсом з базою текстів пісень, для взаємодії з онлайн-сервісом Spotify, планування дизайну, дослідження дизайну подібних додатків, та створення власного.

Метою роботи є розробка додатку, який дозволить користувачам здійснювати пошук текстів відтворюваних пісень в режимі реального часу, дозволить шукати тексти пісень за текстовим запитом, також забезпечить користувачів онлайн-сервісу Spotify, і не тільки, додатковою функціональністю для отримання поглибленого досвіду від прослуховування музики.

Об’єктом дослідження дипломної роботи є розробка корисного додатку для пошуку текстів пісень, а також його підтримка у майбутньому та розширення функціоналу.

Предметом дослідження є методи розробки без застосування інтеграції додатку з Spotify API та сторонніми API для текстів пісень, створення власного алгоритму пошуку текстів пісень на основі існуючих баз даних, дослідження принципів розробки користувацького інтерфейсу, відтворення їх у фінальному продукті, а також оптимізація та швидкість роботи застосунку.

Методи дослідження: абстрагування, аналіз та порівняння, метод розробки алгоритмів.

До теоретичних методів відноситься дослідження застосунку Spotify та інших ліцензійних онлайн-сервісів для прослуховування музики. Дослідження ринку схожих програм за функціоналом з розроблювальним компаньйонським додатком. Спостереження за ресурсами, що містять бази даних з текстами пісень, та пошук найліпшого для використання у роботі.

До практичних методів можна віднести прочитання інструкцій та документацій про роботі з різними пакетами для розробці застосунку на Python, ознайомлення з навчальними відео, та створення невеликих

застосунків для перевірки на придатність до роботи окремих функціональних рішень до майбутньої програми.

У відповідності до поставленої мети даної дипломної роботи, ставились та розв’язувались такі дослідницькі завдання:

1. Зробити аналіз обраних методів розробки та ідей, що застосовуються у програмі, обрати відповідний, оптимальний ресурс для завантаження тексту пісень з нього, дослідження стосовно реалізації окремих частин програми.

2. Розробити вимоги до продукту на фінальному етапі, вивчити ідеї, що були та не були використані аналогічними застосунками, дослідити вибір інших розробників.

3. Спроєктувати програмну структуру застосунку, спроєктувати приблизний візуальний інтерфейс, дослідити інтерфейс онлайн-сервісу Spotify, та відтворити його ідеї у своєму програмному застосунку

4. Проводити тестування системи на всьому етапі розробки, типізувати тестування на функціональні, графічні та виробничі. Оцінювати кожен з перелічених аспектів програми з боку розробника та користувача. Знаходження та впровадження спільних ідей.

У роботі усі поставлені завдання було вирішено. Практичним значенням одержаних результатів є:

1. Розширення функціональності Spotify:

Додаток надає юзерам можливість знаходити тексти пісень, що відтворюються в Spotify, у режимі реального часу, що покращить їхній досвід прослуховування музики.

2. Підвищення задоволеності користувачів:

Якісно створений інтерфейс зробив використання додатку зрозумілим та зручним, що сприяє збільшенню кількості користувачів. Можливість прямої інтеграції функції пошуку з найпопулярнішим онлайн-сервісом з прослуховування музики є однозначним плюсом для потенційних юзерів.

3. Культурне значення для користувачів:

Користувачі можуть детальніше ознайомитися з текстами прослуховуваних пісень, що сприяє кращому розумінню музичних творів. Окрім розважальних цілей, юзер може переслідувати навчальні цілі, тоді додаток може використовуватися як інструмент для вивчення іноземних мов.

Наукова новизна виконаної роботи полягає у розробці якісного додатку для пошуку тексту композицій. Порівняно з схожими додатками, унікальність вирізняється у реалізації деяких аспектів стосовно пошуку тексту та розробки без застосування сторонніх API.

Практичне значення результатів дослідження полягає у розробці програмного забезпечення, на їх основі. Проведенні дослідження та аналіз допомогли розробити

Результатом виконання дипломної роботи є підтвердженні результати навчання, які є відповідними до освітньої програмі спеціальності 122 «Комп’ютерні науки»:

- Застосувати знання основних форм і законів абстрактно-логічного мислення, основ методології наукового пізнання, форм і методів вилучення, аналізу, обробки та синтезу інформації в предметній області комп’ютерних наук.
- Розробляти програмні моделі предметних середовищ, вибирати парадигму програмування з позицій зручності та якості застосування для реалізації методів та алгоритмів розв’язання задач в галузі комп’ютерних наук.
- Володіти навичками управління життєвим циклом програмного забезпечення, продуктів і сервісів інформаційних технологій відповідно до вимог і обмежень замовника, вміти розробляти проєктну документацію.

Дипломна робота складається зі вступу, 3 розділів, висновків, переліку використаних джерел та додатку. Усього робота налічує 11 джерел. Додаток демонструє повний програмний код розроблювального проєкту та складається з 5 сторінок. Дипломна робота має об’єм у 61 сторінки та 25 рисунків.

РОЗДІЛ 1. ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ПОСТАНОВКА ЗАВДАННЯ ДОСЛІДЖЕННЯ

1.1 Аналіз предметної області та історія виникнення онлайн-сервісів для прослуховування музики

Потреба людства у прослуховуванні музики була завжди, навіть від тих часів, коли не існувало мов, як таких. У наш час сучасних інформаційних технологій ця потреба тільки виросла, але тепер мі слухаємо улюблені композиції користуючись додатками для гаджетів, онлайн-сервіси з потоковим транслюванням музики значно зросли в популярності за останні роки, і Spotify займає одну з провідних позицій серед них. У той же час зростає попит на додаткові функції, що покращують роботу користувачів, особливо на можливість перегляду текстів пісень прямо під час їх прослуховування. Розробка супутнього додатку, який інтегрується зі Spotify і надає текст до відтворюваної пісні, є актуальною темою в контексті розробки програмного забезпечення, створення сучасних та потрібних програм.

Для початку треба розглянути існуючі рішення і публікації, що спеціалізуються на розробці програмних додатків для пошуку текстів пісень. Основними джерелами є наукові дослідження, технічні звіти, статті в спеціалізованих IT журналах та документації по використовуванню плагінів та інших додатків для обраної мови програмування проєкту.

Історія онлайн-сервісів прослуховування музики розпочинається з появи Інтернету та розвитку цифрових технологій. Перші спроби розробити платформу для обміну музикою з’явилися в 1990-х роках. Одним з найвідоміших ранніх прикладів був Napster, випущений в 1999 році. Це була послуга обміну файлами, яка дозволяла користувачам порушувати авторські права та завантажувати безкоштовну музику. Популярність Napster призвела до численних судових позовів, які врешті-решт були закриті в 2001 році. На

фоні цієї події піратські сервіси продовжили існування, але суди з ними були лише частішим явищем

Потім розпочалася ера легальних музичних послуг. У 2003 році Apple запустила iTunes. Він дозволяв купувати та завантажувати окремі пісні й цілі альбоми. Це був великий крок вперед, для прослуховування ліцензійної музики, але користувачі шукали більш зручний спосіб отримати доступ до музики без необхідності купувати кожен трек окремо.

Першим по-справжньому відомим потоковим сервісом була Pandora, запущена в 2005 році. Pandora надавала юзерам радіо за запитом, яке дозволяло створювати власні станції відповідно до їхніх музичних уподобань. У 2008 році З'явився Spotify [1][2], який змінив все, ця революційна програма надавала доступ до величезної музичної бібліотеки безкоштовно за допомогою підписки. Spotify став дуже популярним завдяки своїй зручності, величезному каталогу та можливості створювати власні списки відтворення. Тут можна відстежити як вплив iTunes, у можливості прослуховування окремих пісень та альбомів у цілому, так і вплив Pandora, тому що однією з візитних карток сучасного Spotify є саме еволюціонована система радіо з Pandora, зараз юзери можуть створювати власні плейлисти, що доступні усім користувачам, або сам Spotify, завдяки новітнім алгоритмам сам створює підбірки музичних композицій за темами, типами музики, мовою співаків та навіть темпом і довжиною пісень.

Потреба в музичних онлайн-сервісах. Послуги онлайн-прослуховування музики стали необхідністю для мільйонів людей з кількох конкретних причин:

- Доступність: для прослуховування всіх пісень світу користувачам треба мати лише з’єднання з мережею інтернет, вони забезпечують миттєвий доступ до мільйонів треків з будь-якого місця і в будь-який час.
- Зручність: користувачі можуть слухати музику на різних пристроях без необхідності завантажувати та зберігати файли.

- Економічна ефективність: підписка на послугу потокового передавання, як правило, краща, ніж придбання одного окремого альбому чи треку.
- Просування нової музики: сучасні алгоритми, що взаємодіють напряму з історією прослуховування користувачів, часом прослуховування відповідних треків, дозволяють платформі здійснювати автоматичний пошук нової музики, що гарантовано сподобається користувачу.

Переваги та недоліки музичних онлайн-сервісів. У сучасних платформах для прослуховування музики є купи переваг, над старими платформами, такими як mp-3, cd та касетні плеєри, а саме:

- Широкий вибір: великий музичний каталог відомих і незалежних виконавців.
- Персоналізація: Алгоритм створює персоналізовані рекомендації та списки відтворення.
- Інтерактивний: користувачі можуть створювати списки відтворення та ділитися ними.
- Соціальні функції: Можливість відстежувати уподобання друзів і знайомих.

Одним з найбільших помічених, під час аналізу багатьох сервісів, недоліків деяких додатків є відсутність текстів пісень. Тексти пісень можуть бути важливими для користувачів, які хочуть співати пісню або хочуть краще зрозуміти її зміст. Відсутність текстів зменшує загальне задоволення від прослуховування і може стати перешкодою для людей, які хочуть вивчити нову мову за прослуховуванням композицій на ній або глибше зануритися в музику та творчість виконавців.

Але багато сервісів, таких як Spotify та Apple Music, додали текст у режимі реального часу, щоб покращити взаємодію з користувачем. Це показує еволюцію сервісу і його готовність задовольняти потреби користувачів.

Таким чином, онлайн-сервіси прослуховування музики мають значний вплив на те, як люди слухають музику, і стають все більш доступними, незважаючи на деякі недоліки, які поступово усуваються.

Нажаль зміна політики корпорації Spotify Technology S.A. заборонила користувачам без преміум підписки передивлятися тексти пісень, також ця проблема була раніше, оскільки Spotify мав тексти далеко не до всіх пісень

1.2 Дослідження застосунків для пошуку тексту пісень. Технічні аспекти інтеграції створюваного додатку зі Spotify

Згідно з аналізом публікацій, статей, та власними спостереженнями за подібними додатками, існуючі супутні програми до музичних сервісів часто мають обмежені можливості, такі як:

- Не всі з досліджених програм підтримують автоматичне відображення тексту пісні, яка відтворюється у конкретний момент прослуховування.
- Некоректність тексту пісень, погані джерела, відсутність модерації текстів, перелічені фактори сильно знижують зручність та бажання використання таких сервісів.
- Не всі тексти доступні, ця проблема не є такою критичною, як минула, оскільки на неї може не впливати розробник додатку або сервіс з зберіганням бази даних пісень, така ситуація може бути зумовлена політикою сервісів або більш глобальними факторами, наприклад геополітичною невирішеністю країн.

Варто відмітити, що деякі з цих програм мають деякі цікаві функції, такі як можливість ручного пошуку текстів пісень та надання додаткової інформації про пісні та виконавців, це може бути зручним функціоналом у випадку коли користувач бажає знайти текст пісні без її увімкнення.

Для розробки ефективного супутнього додатка важливо враховувати технічні можливості та обмеження API Spotify, а також можливість реалізації процесу розробки без їх використання.

Трьома основними функціями, які має надавати відповідний програмний застосунок, є:

- Можливість отримування інформації про відтворювану в даний момент пісню (назва, виконавець).
- Можливість дати користувачу змогу відтворювати ручний пошук тексту пісень.
- Можливість повторного пошуку, не обов’язково за таким самим методом.

Дослідження потреб користувача приводить до наступних висновків. Щоб розробити зручний і зрозумілий користувачеві додаток, необхідно проаналізувати потреби юзерів. Зокрема, для якісного аналізу слід уточнити такі питання:

- Які функції вони вважають найбільш важливими (автоматичне відображення тексту, синхронізація з музикою, можливість ручного пошуку тощо)?
- Які проблеми виникають при використанні існуючих функцій.
- Як використовують проблеми, та наскільки вони відштовхують користувача від майбутнього користування застосунком
- Такий аналіз можна провести, читаючи відгуки до програмних застосунків на таких платформах як Google Play та Microsoft Store.

Також варто ознакомитися з проблемами відтворення тексту у таких онлайн-сервісах як: Spotify, YouTube Music, Apple Music.

1.3 Особливість розроблюваної програми

У сучасному цифровому світі всі слухають музику, комусь вона подобається, комусь ні, але кожна людина колись певно хотіла дізнатись, що співається у якійсь конкретній пісні, для телефонів є багато додатків, які допоможуть з цією проблемою, на комп'ютер таких майже немає. Користувачам музичних сервісів на комп'ютері доводиться власноруч відкривати браузер на набирати назву пісні з ім'ям виконавця, що є довгим та не дуже зручним процесом. Тому розроблення додатку, що зможе знаходити текст пісень декількома способами є актуальним і потрібним рішенням для нашого життя.

Можна навести основні причини, у чому полягає необхідність створення програми з означеними характеристиками.

Покращення досвіду від прослуховування музики:

- Легкий доступ: користувачі можуть переглядати текст пісні, ввівши назву пісні та виконавця, або дозволити програмі автоматично розпізнавати трек, який зараз відтворюється у плеєрі.
- Зручність: наявність текстів пісень в одній програмі позбавляє користувача від необхідності вручну шукати тексти пісень в Інтернеті.
- Розуміння культури: музика часто відображає культурний, соціальний і політичний контекст, до чого відносить себе виконавець. Розуміння тексту пісень допомагає дізнатися більше про це, краще розуміти автора пісні.

Опишемо технологічні інновації, які доцільно використати у розробці додатку.

- Інтеграція з музичними службами: програми, інтегровані з такими платформами, як Spotify, можуть автоматично ідентифікувати пісню, що відтворюється, і відображати текст, що робить процес пошуку ще більш зручним.

- Відсутність API: відсутність API при розробці дозволяє користуватись додатком без реєстрації, також у таких додатки, як правило, більше швидкодія, що є однозначною перевагою, при пошуку.

Особливості взаємодії з найденим текстом.

- Обмін текстами: користувачі можуть ділитися знайденими текстами оскільки поле, що використовується для виводу дозволяє копіювати інформацію з себе

- Оформлення: тексти оформлені у схожому стилі до дизайн коду програми Spotify, це сприяє покращенню зчитування інформації при швидкому переключені з плеєра онлайн-сервісу на вікно знайденого тексту

Особливості готового продукту.

- Монетизація: розроблена програма є самостійним готовим продуктом, що може бути викладена на різні інтернет-маркети додатків. Створений проєкт може допомогти розробнику заробити на ньому, можливе додання реклами, та відповідних підписок для відключення цієї реклами.

Програми, які відображають тексти за запитом або роблять це автоматично, важливі з кількох важливих причин. Це не тільки покращує отримуваний досвід при прослуховуванні музики, але й сприяє вивченню мови, культурному розвитку користувача, можливістю поділитися цікавими рядками пісень, також подібні потрібні самим розробникам, оскільки створення подібних програм не тільки робить життя багатьох людей зручніше, а й може бути використано як спосіб заробітку, чи приклад у резюме для прийнятті на роботу. У сучасному світі, де технології та музика пов’язані нерозривно, такі програми є важливим інструментом для задоволення різноманітних потреб користувачів, згодом вони будуть ставати все більш розвиненими та технологічними, подібні додатки це лише перший крок, до майбутнього, де прослуховування музики приносить користь та задовольняє людей все більше.

1.4 Висновки до першого розділу

Зроблений аналіз дозволив визначити основні напрямки подальшої роботи по розробці програми. Треба відмітити, що одним із найважливіших аспектів є забезпечення зручності та певної інтерактивності використання програми, що досягається за рахунок інтеграції розроблюваної програми, що є супутньою до популярного потокового сервісу Spotify. Технічний аналіз показує можливості програми та обмеження, у рамки яких слід планувати майбутню розробку.

Основна мета дослідження – розробити додаток, який надає користувачам доступ до великої бази текстів пісень, це повинно відбуватися під час прослуховування музики, за основний додаток, на сумісну роботу з яким розрахована робота, є Spotify. Для досягнення поставленої мети було проведено детальний аналіз існуючих рішень та різних технологій їх реалізації, що використовуються в галузі розробки додатків на мові Python. Це дозволило визначитися з основними вимогами та завданнями, які необхідно враховувати при розробці додатку.

Перш за все, однією з важливих задач, що стоять перед розробкою програми, є забезпечення точності і своєчасності відображення тексту пісні.

Це вимагає використання надійних джерел даних та алгоритмів, які можуть автоматично знаходити текст за сформованим запитом. Також важливо передбачити можливість пошуку тексту пісень вручну, за текстовим запитом. Це дозволяє користувачеві знаходити тексти пісень, яких немає на платформі Spotify. Такий підхід повинен забезпечити юзерів відчуттям, що використання додатку приносить їм задоволення.

Процес розробки зручного та інтуїтивно зрозумілого інтерфейсу користувача є одним з найскладніших аспектів розробки додатків на Python. Інтерфейс повинен бути простим у використанні, з чіткою навігацією та мінімальним часом навчання у користуванні. Також важливо забезпечити сумісність з різними операційними системами, це не є великою проблемою,

оскільки розроблювальна програма не має ніяких функцій, що були б притаманні якимось окремим операційним системам. Адаптивний дизайн може привезти до неочікуваних проблем у користувача, тому від початку було прийнято рішення зробити простий та зрозумілий дизайн.

Не менш важливим фактором є забезпечення безпеки та конфіденційності даних юзерів, тому програма повинна не мати зайвих функцій, що будуть вимагати від користувача його даних.

Використання текстів пісень може бути обмежене умовами ліцензійної угоди, тому важливо переконатися, що додаток та сайт з базою даних текстів пісень відповідає законодавчим вимогам та законам про авторське право.

Забезпечення високої продуктивності додатку також є важливим аспектом, оскільки очікується, що кількість користувачів буде збільшуватися. Тому необхідно впроваджувати рішення, які дозволять системі ефективно працювати, треба впровадити дійсно потрібний функціонал, що зможе реалізувати всі потреби користувачів.

Загалом, розробка програми для відображення текстів пісень на Spotify є складним і багатогранним завданням, яке вимагає великої кількості знань, зацікавленості у процесі розробки та розгляду багатьох технічних, функціональних та юридичних аспектів. Проведений аналіз дозволив визначити основні аспекти та способи забезпечення успіху цього проєкту. Беручи їх до уваги, можна створювати конкурентоспроможні продукти, що відповідають потребам користувачів, і надавати високоякісні послуги для прослуховування музики з текстами пісень.

РОЗДІЛ 2. ОГЛЯД ІСНУЮЧИХ РІШЕНЬ, ЇХ АНАЛІЗ ТА ВИБІР МЕТОДІВ ВИРІШЕННЯ

2.1 Огляд існуючих рішень. Аналіз подібних застосунків

Для пошуку текстів пісень використовуються різні технології, кожна з яких має свої унікальні особливості, переваги та недоліки. Усього можна виділити три основних типи пошуку текстів пісень, які відповідають різним потребам та сценаріям використання:

Автоматичний пошук. Авто-пошук є найзручнішим та найшвидшим способом отримання тексти композицій. Цей тип пошуку передбачає автоматичне визначення пісні, яка зараз грає, і відображення її тексту в режимі реального часу. Основні технології, що використовуються для автоматичного пошуку, включають розпізнавання граючою пісні використовуючи інформацію з музичного плеєру, проводиться ідентифікація граючого музичного треку через спеціальну базу даних. Наприклад, такі сервіси, як Spotify або Musixmatch, здатні обробляти великі об’єми запитів текстів пісень за лічені секунди, маючи доступ до своєї бази даних з прив’язаними таймкодами вони відображають поточний граючий рядок.

Переваги автоматичного пошуку:

- Швидкість та зручність: користувачу не потрібно вручну вводити інформацію стосовно бажаної композиції.
- Точність пошуку: сучасні алгоритми розпізнавання музики мають високу точність і рідко допускають помилки.
- Інтеграція з різними музичними сервісами: автоматичний пошук може бути інтегрований з популярними музичними платформами, такими як Spotify чи Apple Music, це значно підвищує зручність для користувачів.

Недоліки автоматичного пошуку:

- Обмеження баз даних: не всі пісні можуть бути розпізнані, особливо якщо вони не входять до великих музичних бібліотек.
- Реалізація автоматичного пошуку з відображенням поточного рядку може бути напряму виконана лише з інтеграцією до спеціальних баз даних, до яких, наприклад Spotify, не дає доступу.

Текстовий пошук. Пошук за вводом – класичний метод пошуку текстів пісень, який передбачає введення користувачем ключових слів, фраз чи назви пісні у пошукове поле. Цей тип пошуку може здійснюватися як у веб-браузерах, так і в спеціалізованих додатках. Технології, що використовуються для пошуку за вводом, включають індексацію текстів пісень та застосування алгоритмів пошуку по базах даних, такий метод передбачає можливість використовувати декілька баз даних.

Переваги пошуку за вводом:

- Гнучкість пошуку: користувач може знайти текст пісні навіть за окремими фразами або словами, якщо не пам'ятає точну назву.
- Незалежність від онлайн-плеєрів: немає потреби у вмиканні пісні, це дозволяє шукати тексти у будь-яких умовах, без необхідності у запуску музичного плеєру.
- Доступність: цей метод доступний на всіх пристроях, що мають клавіатуру або сенсорний екран, тому для desktop-застосунків це найпопулярніший варіант реалізації пошуку лірики.

Недоліки пошуку за вводом:

- Потребує більше часу: порівняно з автоматичним пошуком, користувачу потрібно витратити більше часу на введення інформації та перевірку на правильність вводу.
- Правильність введення: помилки у введенні можуть призвести до відсутності результатів пошуку, або їх некоректності.

Пошук за звуком. Пошук з використанням мікрофону та звуків пристрою є інноваційним методом, який дозволяє знаходити тексти пісень, розпізнаючи їх за допомогою мікрофону. Цей тип пошуку використовує технології розпізнавання мови та аудіосигналів для ідентифікації пісні.

Переваги пошуку з використанням мікрофону:

- Зручність пошуку: користувачу не потрібно вводити текст вручну, достатньо просто розпочати пошук біля граючої пісні.
- Можливість розпізнавання мелодій: навіть якщо пісня є кавером або реміксом, мелодія може бути розпізнана.

Недоліки пошуку з використанням мікрофону:

- Залежність від якості виконання: точність розпізнавання може бути знижена, якщо мелодія звучить нерозбірливо, або грає далеко.
- Обов’язкова наявність мікрофону: наявність мікрофону є серйозним недоліком для даного методу пошуку.
- Відсутність підтримки desktop-застосунків: даний тип пошуку не буде зручним для користувачів комп’ютером, оскільки комп’ютер може не мати мікрофона для розпізнавання пісні, або не є таким портативним, як смартфони та планшети, для яких цей метод пошуку є більш доречним.

Таким чином, кожен з трьох основних типів пошуку текстів пісень має свої унікальні переваги та недоліки. Вибір конкретного методу залежить від умов та потреб користувача. Авто-пошук підходить для швидкого і точного визначення пісень, пошук за вводом – для гнучкості та незалежності від аудіо, аудіо-пошук – не підходить до формату desktop-застосунків, коли комп’ютер може розпізнавати звук тільки у кімнаті, де стоїть.

Розгянувши різноманітні методи пошуку текстів пісень приходиш до висновку, що існує велика кількість програм для пошуку текстів пісень, що використовують перелічені технології пошуку, є хороші та погані варіанти, пропонуються великі та малі, легкі та складні функції. Розглянемо деякі популярні програми для пошуку текстів пісень, які відрізняються за

функціональністю, якістю продуктивності та зручністю використання. Це допоможе зрозуміти аспекти, які слід враховувати при розробці програми, щоб зробити її конкурентоспроможною та корисною для майбутніх користувачів.

Огляд Spotify. Spotify не є програмою для пошуку текстів, її основний функціонал це саме програвання пісень, об’єднання їх у власні плейлисти, можливість ділитися ними тощо.

Але Spotify все ж має функцію пошуку текстів, лише за автоматичним режимом, після дослідження роботи даної функції було знайдено, що тексти є далеко не до всіх пісень (Рис. 2.1)

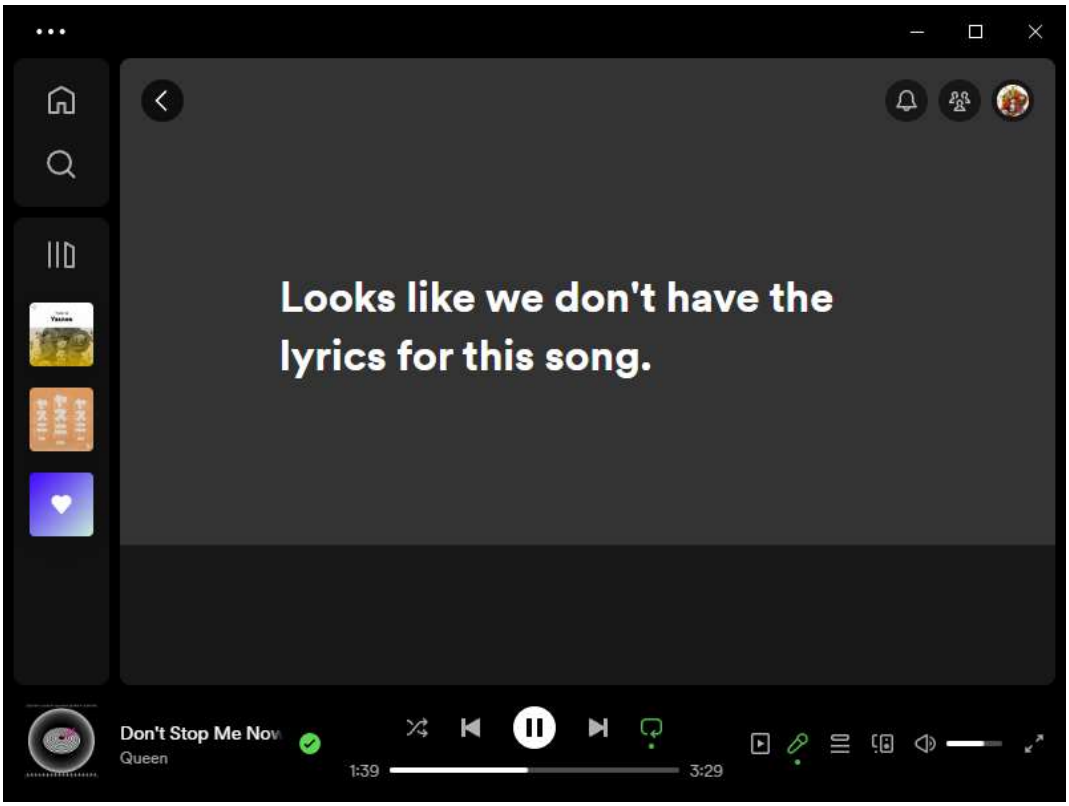


Рисунок 2.1 – Не знайдений текст пісні, у якої на платформі майже 2 мільярди прослуховувань

Також дослідження було виявлено, що після обновлення політики компанії, функція відображення текстів тепер доступна тільки після оформленні преміум-підписки. (Рис. 2.2)

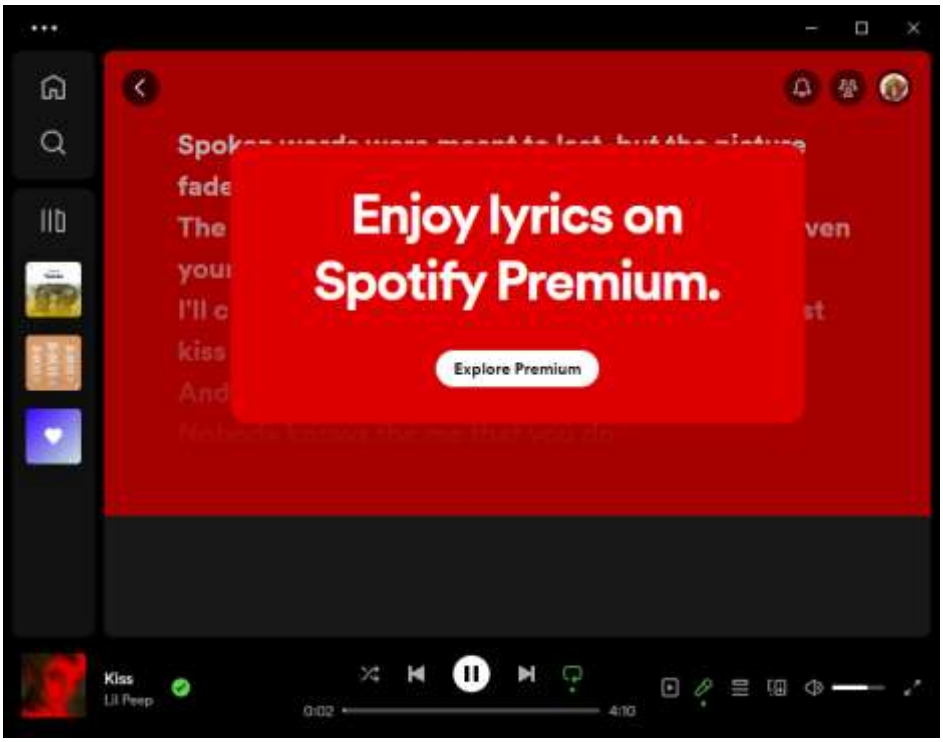


Рисунок 2.2 – Текст пісні знайдено, але він доступний лише після оформлення преміум-підписки

Серед значних плюсів відображення текстів пісень у Spotify треба відзначити приємний дизайн, що вписується до загального вигляду онлайн-сервісу, також приємною функцією її відображення рядка, що грає у даний момент.

Серед значних плюсів відображення текстів пісень у Spotify треба відзначити приємний дизайн, що вписується до загального вигляду онлайн-сервісу. Елементи інтерфейс додатку гармонійно поєднується між собою, забезпечуючи зручність та задоволення користувачів під час прослуховування музики та пошуку текстів до неї. Особливо приємною функцією є відображення рядка, що грає у даний момент, що дозволяє користувачам легко стежити за текстом пісні в режимі реального часу. Це не тільки підвищує зручність, але й додає інтерактивності та залученості читача тексту до прослуховування музики.

Огляд Versefy. Versefy – це програма для пошуку текстів пісень, серед всіх представлених варіантів вона має найбільший функціонал. Вона пропонує

широкі можливості для користувачів, але при дослідженні виникли деякі помилки, що негативно позначилися на фінальному баченні додатка. Однією з основних проблем є використання API та поганих джерел текстів пісень, через що тексти можуть знаходитися задовго, або взагалі залишитися незнайденими. Після перемикання з одного джерела на інше були спостережені проблеми у роботі, після чого текст пісень писався китайською мовою.

Незважаючи на ці проблеми, варто відмітити, що розробник додав до своєї програми багато налаштувань, такі як: зміна розміру тексту, налаштування товщини тексту, перемикання з світлої на темну тему. Ці функції дозволяють користувачам персоналізувати отриманий досвід та зробити висновки щодо використання додатку. Наприклад, можливість зміни розміру та товщини тексту може бути особливо корисною для людей з різними зоровими потребами, а перемикання між темною та світлою темами дозволяє адаптувати інтерфейс під різні умови освітлення, що значно покращує зручність використання додатку в різний час доби.

Таким чином, хоча Versefy і має свої недоліки, пов’язані з надійністю джерел текстів пісень, його багатофункціональність та налаштування забезпечують високий рівень персоналізації та зручності для користувачів.

Однак, для того щоб повністю відповідати очікуванням користувачів, розробникам потрібно зосередитися на покращенні якості джерел та стабільності роботи програми. (Рис. 2.3)

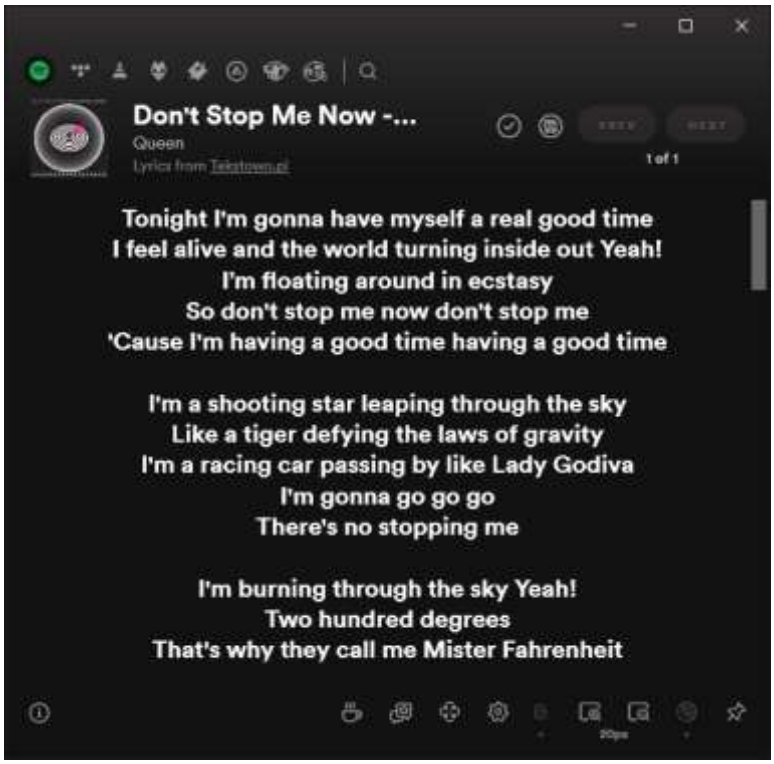


Рисунок 2.3 – Знайдений текст пісні, яку не відображає у себе Spotify

Огляд Lyrixound. Lyrixound – це додаток, що дозволяє шукати текст пісень за текстовим запитом. Він є другим за популярністю варіантом серед представлених у Microsoft Store. Проте, досвід від використання цього додатку не можна назвати однозначно позитивним.

Незважаючи на ці налаштування, основні функціональні проблеми та недоліки в дизайні програми залишаються значним мінусом. Для того щоб Lyrixound міг стати по-справжньому корисним та зручним інструментом для пошуку текстів пісень, але було помічено серйозні недоліки.

Дизайн програми виглядає гірше, аніж у конкурентів, що негативно впливає на загальне враження від її використання. Текст пісень відображається у непідходящих шрифтах, що робить читання незручним та неестетичним. Хоча програма має функцію пошуку тексту пісні, ця функція не працює належним чином, що значно знижує її корисність і викликає розчарування у користувачів (Рис. 2.4).

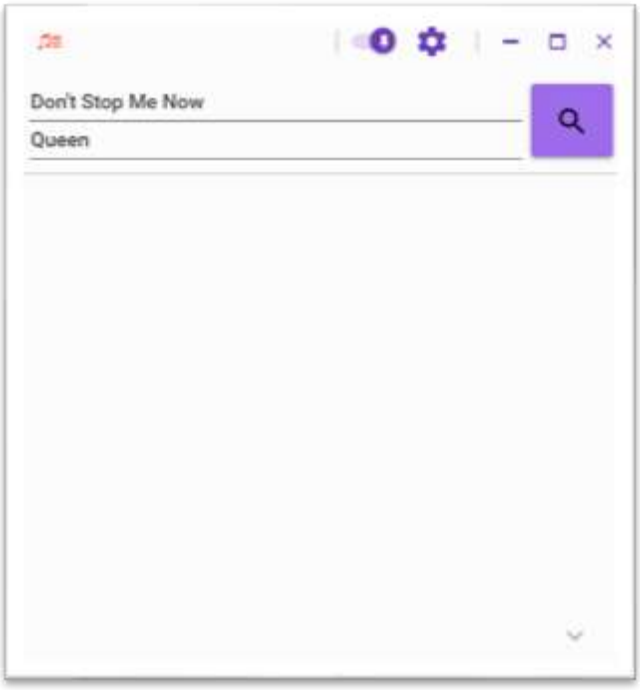


Рисунок 2.4 – Стався збій, через який у програми Lyrixound не працював навіть текстовий пошук

Серед плюсів програми можна відзначити можливість змінювати шрифти, розміри та кольори текстів пісні (Рис. 2.5). Наявні функції персоналізації тексту дозволяють користувачам налаштувати відображення текстів відповідно до своїх уподобань, що може частково компенсувати недоліки в дизайні. Можливість персоналізації відображення текстів є важливим аспектом, оскільки дозволяє адаптувати програму під індивідуальні потреби користувачів, підвищуючи таким чином їх загальну задоволеність від використовування програми.

Рисунок 2.5 – Можливості налаштування вигляду тексту

Огляд Genius. Genius це сайт, з найбільшою базою текстів пісень, серед усіх інших. Нажаль на даний момент немає додатку на персональний комп’ютер, але розробники Genius зробили додаток на телефони.

Перед аналізом варто відзначити, що це найбільша та найдетальніша база текстів пісень, тому інформативність, яку надає програма Genius є найбільшою.

При використанні мобільного додатку були помічені деякі проблеми, серед значних - несумісність з темною темою смартфона, що призводить до графічної помилки, текст та фон майже зливаються, під час цього тексти пісень дуже важко читати (Рис. 2.6).

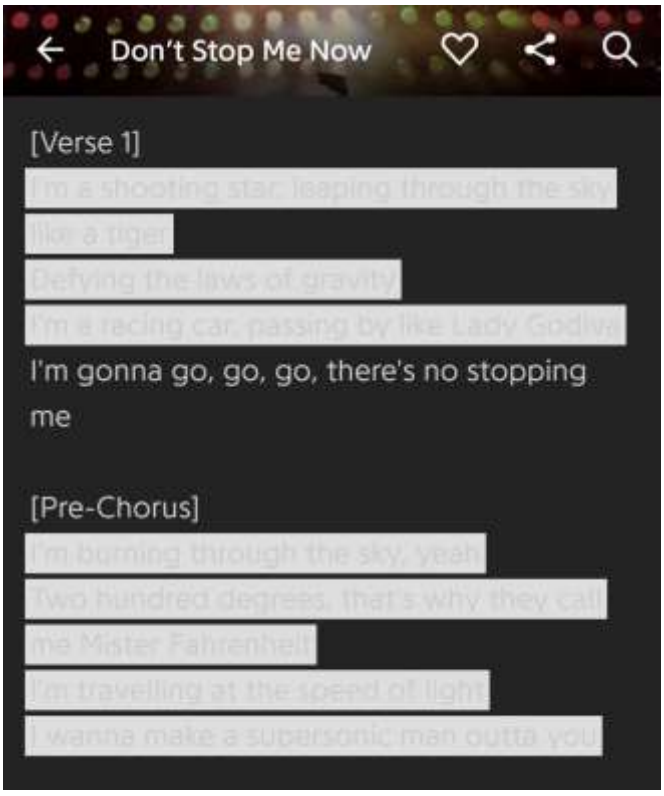


Рисунок 2.6 – Погана синхронізація додатку з темною темою смартфона призвела до помилки у дизайні, через що текст пісні дуже погано видно

Genius представив найбільш цікавий функціонал серед інших програм. Найцікавішою є можливість подивитись пояснення рядків, або окремих слів тексту. Їх можуть додавати як користувачі так і верефікований автор пісні.

Були проведені дослідження стосовно популярності додатків для пошуку текстів пісень, вважаючи що Microsoft Store не надає інформації про завантаження застосунків, інформація була взята з Google Play. Оскільки під час дослідження було важливим визначити кількість завантажень, враховуючи, що одна людина може завантажити застосунок лише один раз, було вирішено порівняти не саму кількість завантажень, а кількість відгуків (Рис. 2.7).

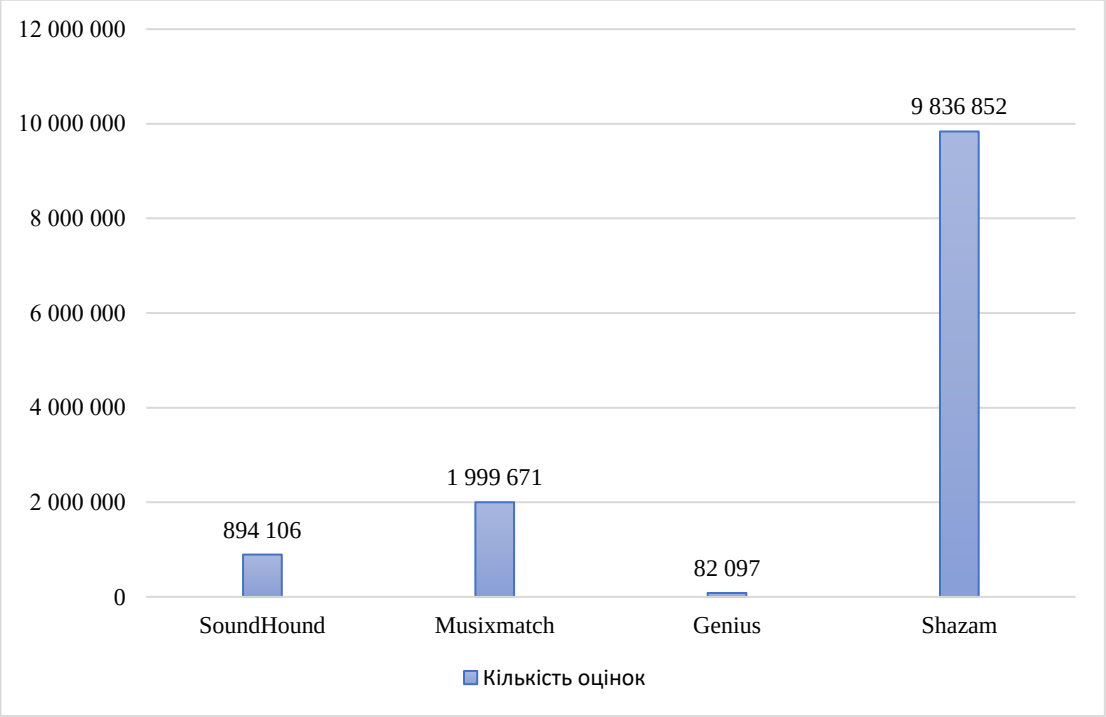


Рисунок 2.7 – Гістограма з порівнянням кількості відгуків, між найпопулярнішими мобільними застосунками для пошуку лірики

Дослідження відгуків дозволяє краще зрозуміти, які додатки користуються найбільшим попитом та задовольняють потреби користувачів смартфонів. Наведена статистика є важливою навіть для desktop-розробника, оскільки спираючись на дані, можна орієнтуватися на вимоги подібного ринку та приймати вірні рішення при розробленні власного застосунку, покращувати свій продукти відповідно до очікувань користувачів, які вже користуються подібними програмами на своїх мобільних пристроях.

2.2 Аналіз відмічених рішень та рекомендації до розробки

Аналіз після проведеної дослідницької роботи над декількома додатками показує, що одними з головних речей, на які розробнику подібної програми варто зосередити увагу це:

Цілісність бази даних, що містить тексти пісень. Велика кількість текстів та цілісність бази даних з ними є одними з найважливіших факторів оцінки додатку. Після аналізу можна прийти до висновку, що Genius, відомий своєю найповнішою базою даних, є лідером у цьому відношенні. А наприклад інші програми, такі як Spotify, мають обмежений доступ до тексту, що робить їх менш корисними для користувачів.

Рекомендація до розробки: надати широку та якісну базу даних текстів пісень, регулярно стежити за її оновленнями, перевіряти надійність та правильність.

Розробка дизайну. Після аналізу можна прийти до висновку, що дизайн та простота використання дуже важливі для користувачів. Spotify має гарний дизайн, але Lyrixound естетично неприємний на фоні іншого. Також у Genius є проблеми з темними темами, які ускладнюють читання тексту.

Рекомендація до розробки: звернути особливу увагу на дизайн інтерфейсу, перевірити що він залишається простим, зрозумілим, та працює коректно.

Особливості специфічних функцій. Після аналізу можна прийти до висновку, що Versify має велику кількість налаштувань, але має свої деякі технічні проблеми. Користувачі цінують можливість налаштовувати зовнішній вигляд тексту (шрифт, розмір, колір), але при зміні ресурсу для пошуку тексту виникали серйозні помилки.

Рекомендація до розробки: не додавати функції, що можуть бути причиною збоїв у роботі додатку.

Автоматичний пошук. Після аналізу можна прийти до висновку, що автоматичний пошук тексту - це найбажаніша функція, її наявність якісно

виділить програму серед конкурентів, але вона повинна бути справною на працювати коректно. У Lyrixound ця функція працює нестабільно, через що дана програма не користується великою популярністю.

Рекомендація до розробки: впровадити у додаток надійну функцію автоматичного текстового пошуку, яка буде працювати точно і швидко, для цього треба перевірити ресурс, що має базу даних текстів, та надалі продовжувати це робити, для уникнення проблем з роботою автоматичного пошуку.

2.3 Огляд бібліотек Python

Приймання запиту від користувача, пошук тексту та його відображення у окремому вікні, цей функціонал реалізований за допомогою мови програмування Python та ряду корисних бібліотек [3][4][5]. Основними інструментами, що використовуються для створення програми, є:

- Tkinter – це стандартна бібліотека Python, яка використовується для створення графічних інтерфейсів.
- Requests – це бібліотека для виконання HTTP-запитів.
- BeautifulSoup – це бібліотека, яка використовується для аналізу HTML та XML.
- Re – це модуль для роботи з регулярними виразами.
- Pygetwindow – це бібліотека, яка використовується для отримання інформації про активні вікна на робочому столі .
- Threading – це модуль для багатопоточності процесів.
- Time – це стандартний модуль для роботи з часом.

Розгянемо детальніше функціонал перелічених бібліотек у цьому проєкті:

Бібліотека Tkinter [6] використовується для створення головного вікна програми та його елементів. Весь графічний інтерфейс розробленого додатку був створений за використанням даної бібліотеки.

Опис функцій пов’язаних з Tkinter:

- clear_entry і restore_entry – функції очищення і відновлення тексту в полях введення.
- close_root_and_open_lyrics – можливість закрити головне вікно і відкрити вікно з текстом.
- show_lyrics_in_new_window – можливість перегляду текстів пісень в новому вікні.

Бібліотека Requests та BeautifulSoup [7] використовується для вилучення та обробки текстів з веб-сайту AZLyrics. Запити виконують HTTP-запит для вилучення HTML-сторінок, а потім використовують BeautifulSoup для аналізу цих сторінок та отримання необхідної інформації.

Опис функцій пов’язаних з Requests і BeautifulSoup:

- get_lyrics – зробить запит на веб-сайт, проаналзує сторінку та поверне текст пісні.

Бібліотека Re [8] використовується для обробки текстів, тобто видалення спеціальних символів з імен виконавців та назв пісень, що є необхідною складовою для реалізації функції пошуку тексту, коректний та швидкий пошук текстів пісень зобов’язаний саме цій бібліотеці.

Опис функцій пов’язаних з Re:

- remove_special_characters – видалить усі символи в тексті, крім букв і цифр.

Бібліотека Pygetwindow [9] використовується для ідентифікації активних вікон на робочому столі та дозволяє автоматично витягувати імена пісень та виконавців із заголовка вікна Spotify.

Бібліотека Threading [10] використовується для забезпечення багатопоточності в додатку. Це дозволяє окремому потоку виконувати тривалі

операції, такі як моніторинг активного вікна, не блокуючи основний інтерфейс користувача.

Опис функцій пов’язаних з Pygetwindow та Threading:

- watch_active_window – функція, яка відстежує активне вікно і отримує його заголовок.
- auto_spot_y – запускає потік для функції watch_active_window.

Бібліотека Time використовується для управління часом, наприклад, для створення затримок до виконання певної дії. У програмі, за допомогою цієї бібліотеки, юзерові надається відповідний час для запуску пісні після натискання кнопки автоматичного пошуку.

2.4 Висновки до другого розділу

У другому розділі було розглянуто аналогічні до розроблювального застосунку, їх дослідження допомогло у вирішенні реалізації функцій додатку. Дослідження переваг та недоліків допомогло зосередити увагу на розгляданні потенційних технологій для реалізації, а аналіз ринку мобільних застосунків з пошуку текстів пісень допоміг визначити тенденції серед популярних програм. Орієнтуючись на рішення, прийняті розробниками мобільних застосунків з пошуку текстів пісень, можна якісніше виконати поставленні задачі для desktop-додатку.

Також був проведений детальний аналіз інструментів та методів, які можна використовувати для впровадження алгоритмів з пошуку текстів пісень за різними типами їх формування. Результати проведеної дослідницької роботи цього розділу охоплюють деяку кількість програмних застосунків, що схожі за функціоналом з розроблювальним проєктом, були описані технології та бібліотеки, що будуть задіяні далі у розробці додатку.

Вивчивши кілька мов програмування та середовищ розробки, було вирішено зосередитись на мові Python. Цей вибір виправданий наступними причинами [3][4][5]:

- Гнучкість та універсальність: Python - це універсальна мова програмування, яка підтримує різні парадигми, включаючи об'єктно-орієнтоване, функціональне та процедурне програмування. Це полегшує адаптацію мови до різних завдань, що виникають в процесі розробки.
- Велика бібліотека: Python має великий набір бібліотек, що значно спрощує процес розробки. Зокрема, обробка текстів, взаємодія з користувачами та багатопотокові бібліотеки важливі для створюваного проєкту.
- Активна спільнота: велика та активна спільнота розробників Python забезпечує постійну підтримку та оновлення мови, її бібліотек тощо. Існує безліч інструкцій, якими можна керуватися при процесі розробки.

Основна частина проведеного аналізу полягала в оцінці та виборі бібліотек, що використовуються для розробки системи. Зокрема, варто відмітити ці 3 розглянуті бібліотеки:

- Re [8]: ця бібліотека є ключем до роботи з текстовими даними. Це дозволяє видаляти спеціальні символи з імен виконавців та назв пісень, що важливо для правильного формування URL запити для пошуку текстів. Обробка текстових даних за допомогою регулярних виразів, наданих Re, забезпечує високий рівень точності та швидкості.
- Pygetwindow [9]: бібліотека Pygetwindow дозволяє автоматично отримувати інформацію з активного вікна Spotify, включаючи назву пісень, яка автоматично там з’являється при увімкнені музики. Це значно спрощує процес автоматичного пошуку текстів пісень, оскільки користувачам не потрібно вручну вводити назву кожної пісні. Автоматизація цього процесу підвищує зручність використання програми.

- Threading [10]: багатопотоковість є важливою функцією для системи, оскільки вона дозволяє виконувати кілька операцій одночасно, не заважаючи основному інтерфейсу користувача. Бібліотека потоків дозволяє виконувати довгострокові операції, такі як пошук текстів у фоновому режимі, надаючи можливість створювати потоки та керувати ними.

Додатком до основної бібліотеки, були також використані додаткові інструменти для поліпшення функціональності і взаємодії з користувачем:

- Time: бібліотека time використовується для управління часом та створення затримок. Це особливо корисно, щоб дати користувачеві достатньо часу для запуску пісні перед початком автоматичного пошуку тексту. Налаштування функцій цієї бібліотеки дозволяє регулювати інтервал між запитами та оптимізувати поведінку та швидкодію системи.

На підставі аналізу було визначено, що використання обраних програмних засобів є найбільш оптимальним рішенням для реалізації поставленого завдання та мети застосунку. Інтеграція між інструментами в алгоритмах пошуку тексту пісень дозволяють досягти високої ефективності, швидкості і точності пошуку текстів пісень. Кожна обрана бібліотека відіграє свою унікальну роль і забезпечує комплексний підхід до вирішення проблем.

Можна зробити висновок, що вибрані рішення найкраще за все підходять для розробки алгоритмів пошуку тексту пісень, створення дизайну вікон до застосунку тощо. Інтегруючи ці інструменти, можна забезпечити високу ефективність, швидкість і точність роботи системи.

Також, підводячи підсумки, у другому розділі було підтверджено, що використання мови програмування Python та пов’язаних з нею бібліотек є правильним вибором для розробки майбутньої системи. Мова Python забезпечує не тільки гарну технічну реалізацію набору завдань, але і високий рівень зручності при процесі розробки, також забезпечує задоволеність користувачів від використовування фінальної версії продукту.

Python надає велику кількість бібліотек та фреймворків, які спрощують реалізацію складних функцій, таких як обробка запитів до різних ресурсів, що можуть слугувати базами даних. Технології обраної мови програмування роблять розробку не лише ефективною, але й більш продуктивною, дозволяючи розробнику зосередитися на вирішенні ключових задач, а не на окремих малих технічних деталях.

Крім того, Python має велику та активну спільноту розробників, що означає наявність великої кількості ресурсів, документації та підтримки. Це важливо для швидкого вирішення виникаючих проблем та обміну досвідом, що сприяє пришвидшенню процесу розробки та підвищенню якості кінцевого продукту.

Враховуючи перелічені переваги, Python стає оптимальним вибором для реалізації проекту, забезпечуючи необхідний баланс між продуктивністю розробки, функціональністю додатку та задоволеністю користувачів. Таким чином, використання Python не лише сприятиме успішному завершенню процесу розробки проєкту, але й забезпечить його подальший розвиток та підтримку на високому рівні.

РОЗДІЛ 3. РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАСТОСУНКУ “SPOT.Y” ДЛЯ ПОШУКУ ТЕКСТІВ ПІСЕНЬ

3.1 Загальна структура алгоритму для текстового пошуку

Алгоритм для текстового пошуку пісень дозволить користувачеві отримувати текст бажаної композиції, для отримання тексту пісні користувач повинен ввести її назву та ім’я виконавця.

Текстовий пошук реалізовано через створення відповідного URL-запиту, базою даних є сайт AZLyrics.com (Рис. 3.1).

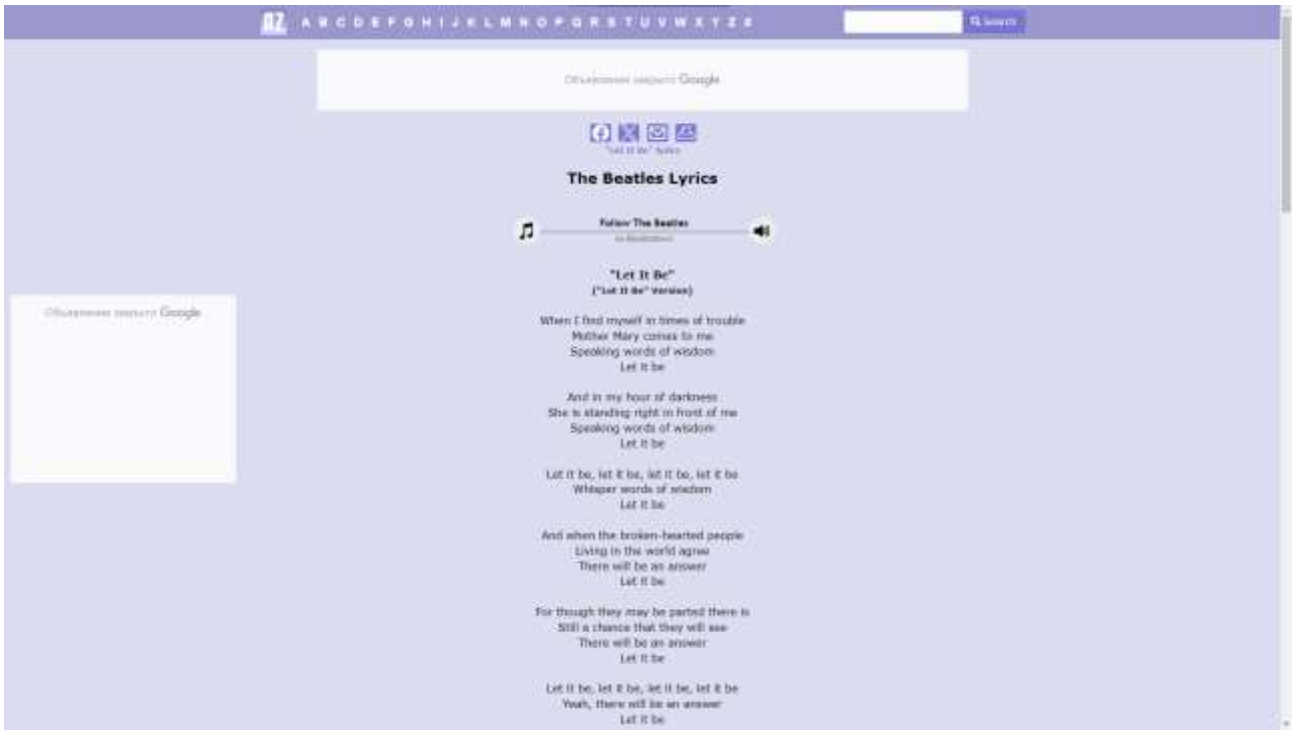


Рисунок 3.1 – Вигляд сайту AZLyrics при відкритій сторінці з текстом

Сайт AZLyrics був обраний через його велику базу даних пісень, також важливим фактором було те, що тексти з даного сайту пройшли граматичну перевірку та перевірку на відповідність.

Для розробки алгоритмів були використані такі Python бібліотеки, як:

- Requests – для відправки HTTP-запитів

- BeautifulSoup – для парсингу HTML
- Re – для роботи з регулярними виразами.

Наприклад, при створенні запиту для пісні “Don`t Stop Me Now – Queen” URL- посилання, для пошуку тексту на сайті, має виглядати саме так: “https://www.azlyrics.com/lyrics/queen/dontstopmenow.html”. Його можна розбити на 3 частини. Перша частина має у собі назву сайту, та частину з “lyrics”. Друга – ім’я виконавця, а третя – назва пісні. Коли користувач вводить ім’я виконавця та назву пісні (Рис. 3.2).

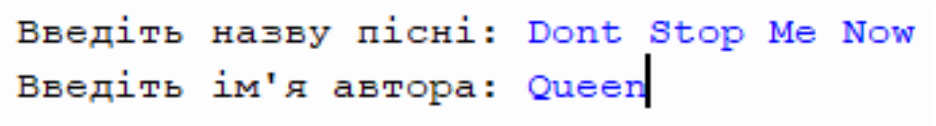


Рисунок 3.2 – Процес вводу інформації для пошуку

Програма використовує введені дані, та формує URL запит з ними у нижньому регістрі, видаляє всі спеціальні символи, залишаючи лише літери та цифри, оскільки вони часто використовуються у назвах композицій та псевдонімах виконавців.

Далі програма шукає на сторінці елементи, які зазвичай містять текст пісень. Знайшовши потрібний елемент, програма витягує з нього текст і видаляє непотрібні пробіли та символи. Якщо текст було знайдено, він з’явиться на екрані (Рис. 3.3).

```
Введіть назву пісні: Dont Stop Me Now
Введіть ім'я автора: Queen

Текст пісні:

Tonight
I'm gonna have myself a real good time
I feel alive
And the world, I'll turn it inside out
Yeah!
I'm floating around
In ecstasy
So don't stop me now, don't stop me
'Cause I'm having a good time, having a good time
I'm a shooting star leaping through the sky
Like a tiger defying the laws of gravity
I'm a racing car passing by
Like Lady Godiva
I'm gonna go, go, go
There's no stopping me
I'm burning through the sky
Yeah!
Two hundred degrees
That's why they call me Mister Fahrenheit
I'm traveling at the speed of light
I wanna make a supersonic man out of you
Don't stop me now
I'm having such a good time
I'm having a ball
```

Рисунок 3.3 – Успішно знайдений текст пісні через консоль

Програма також обробляє сценарії, коли текст не був знайдений, наприклад, якщо веб-сайт недоступний або виникають інші проблеми під час отримання чи обробки сторінки. У цьому випадку програма видає відповідну помилку (Рис. 3.4).

```
Введіть назву пісні: Dont Stap Mi Nau
Введіть ім'я автора: Kwin
Помилка: Не вдалося отримати сторінку. Код стану: 404

Текст песні не знайдено.
```

Рисунок 3.4 – Повідомлення про помилку, якщо текст не був знайдений.

3.2 Розроблення інтерфейсу програми

Після розробки робочих алгоритмів для текстового пошуку можна розпочати розробку графічного інтерфейсу. Для цього була використана бібліотека Tkinter, обрана вона була через свою велику кількість інструментів та різних документацій по їх використанню. Під час процесу розробки були використані знання з розробки дизайну на основі аналізу графічного дизайну Spotify та за допомогою отриманих знань під час навчання. Було визначено назву програми “Spot.y”, а також у програмах Adobe Illustrator та Adobe Photoshop було створено унікальну іконку до проєкту (Рис. 3.5)

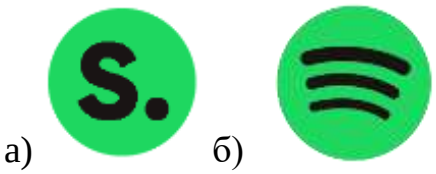


Рисунок 3.5 – а) Іконка розроблювального додатку “Spot.y”. б) Референсом при розробці була іконка програми “Spotify”

Наступним етапом після розробки іконки, було визначення кольорової схеми додатку (Рис. 3.6) та розмітка можливого розташування графічних елементів. На основі проведеної роботи було створено два вікна (Рис. 3.7).

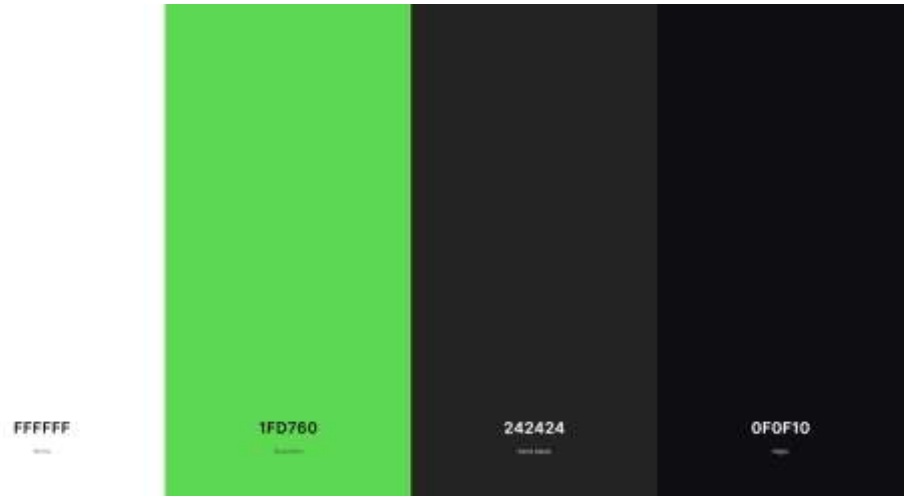
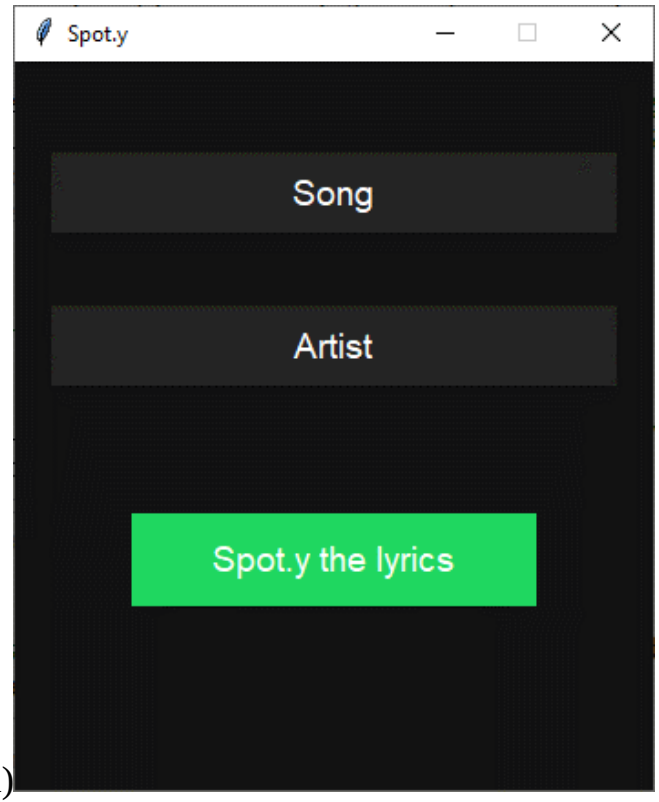


Рисунок 3.6 – Кольорова схема проєкту-додатку “Spot.y”

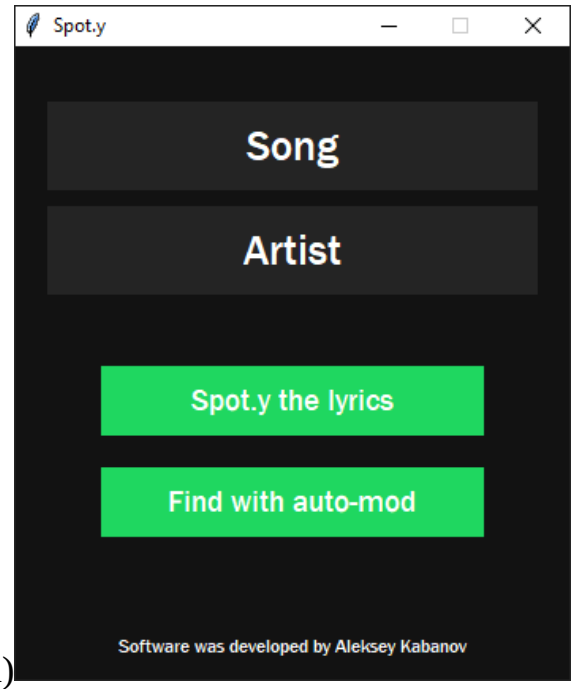
а) 

б) 

Рисунок 3.7 – а) Головне вікно. б) Вікно для виводу тексту

На головному вікні були створені два поля вводу та кнопка, що відкриває вікно для виводу тексту. Вивід тексту здійснюється на другому вікні, у спеціальному полі, а для переходу назад, була додана відповідна кнопка.

На основі дизайну програми для текстового пошуку пісень була проведена робота над оновленням дизайну, для реалізації автоматичного пошуку. При роботі над основним вікном була додана нова кнопка та було скореговано положення всіх елементів, також змінено шрифт, тепер він візуально більш схожий на той, що використовує компанія Spotify у своєму сервісі (Рис. 3.8). Виконані дії є не лише простою візуальна модернізацією, а й кроком до більш зручного, функціонального та приємного у використання додатку.

а) 

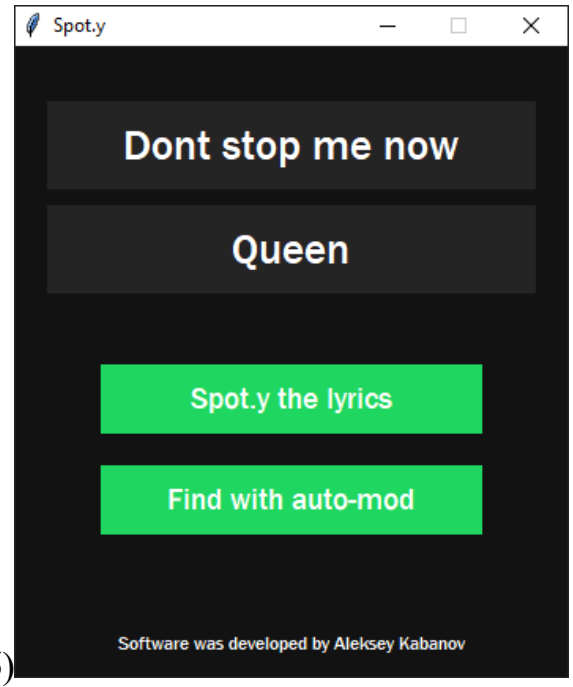
б) 

Рисунок 3.8 – а) Головне вікно до текстового вводу інформації. б) Головне вікно після вводу інформації у відповідні поля

3.3. Огляд на загальну структуру розроблених алгоритмів пошуку та приклади їх роботи

Запуск програми. При запуску програми створюється основне вікно за допомогою Tkinter. Це вікно включає два поля введення, де користувач може ввести назву пісні та ім'я виконавця. Ці поля мають стандартні значення за замовчуванням, які видаляються, коли користувач починає вводити текст, і відновлюються, якщо поле залишається порожнім після втрати фокусу.

Введення даних користувачем. Користувач вводить назву пісні та ім'я виконавця у відповідні поля. Рішення написати на полях текст “Song” та “Artist” допомагає уникнути випадкових помилок введення і робить інтерфейс більш інтуїтивно зрозумілим.

Пошук тексту пісні. Коли користувач натискає кнопку “Spot.y the lyrics”, програма бере введені дані та формує URL-посилання для веб-сайту AZLyrics. Для цього окрема функція очищає введені дані від спеціальних символів,

перетворюючи їх у формат, прийнятний для правильного запиту до бази даних. Потім здійснюється HTTP-запит до сформованого URL.

Обробка відповіді від веб-сайту. Якщо відповідь від веб-сайту є успішною (код відповіді 200), HTML-контент сторінки обробляється за допомогою бібліотеки BeautifulSoup. Програма знаходить відповідний блок HTML, що містить текст пісні, і витягує текст з цього блоку (Рис. 3.9).

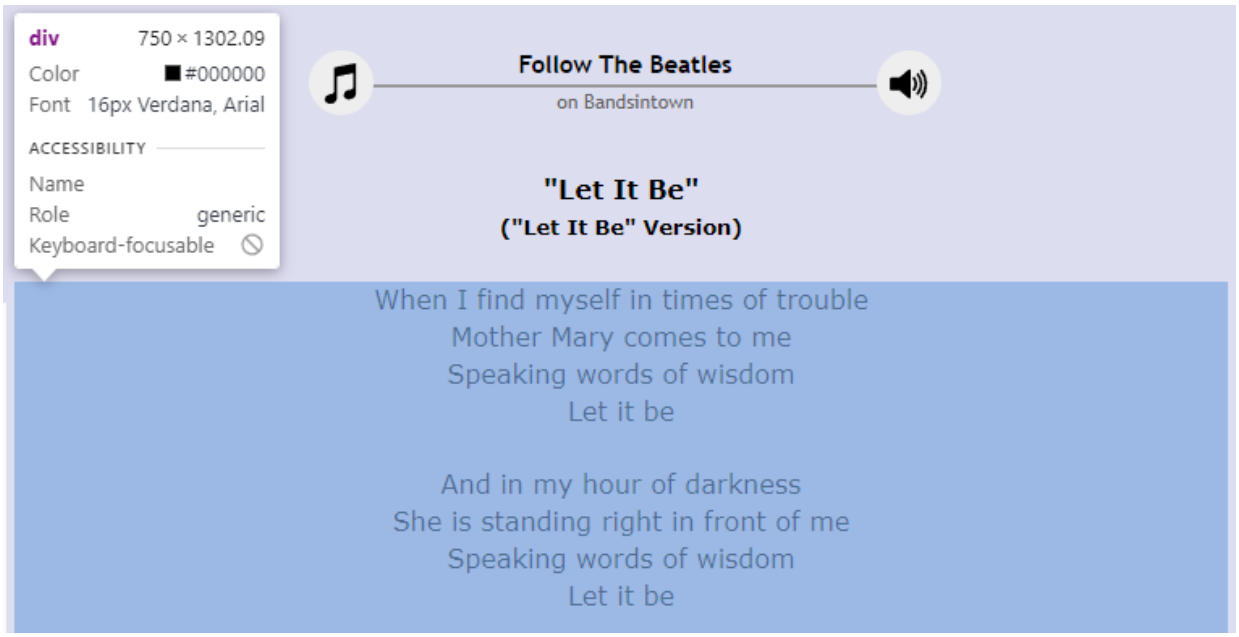


Рисунок 3.9 – Блок “div” з сайту AZLyrics, що містить текст пісні

Після цього текст обробляється для видалення зайвих розривів рядків і інших непотрібних символів.

Відображення тексту пісні. Отриманий текст пісні відображається у новому вікні, яке відкривається замість основного. Це вікно має кнопку, що дозволяє користувачеві закрити його та повернутися до основного вікна, що дозволяє здійснювати інший пошук без перезавантаження додатку.

Автоматичний пошук тексту пісні. Дана функціональність програми дозволяє автоматично визначати назву пісні та ім'я виконавця, без вводу з боку користувача.

Для цього використовується бібліотека pygetwindow, яка відстежує активні вікна і їх заголовки. При натисканні кнопки "Find with auto-mod", програма просить перейти юзера на вікно Spotify з граючою піснею (Рис. 3.10), тоді програма намагається отримати назву пісні з заголовка активного вікна, яке зазвичай відображає інформацію про відтворювану музику.

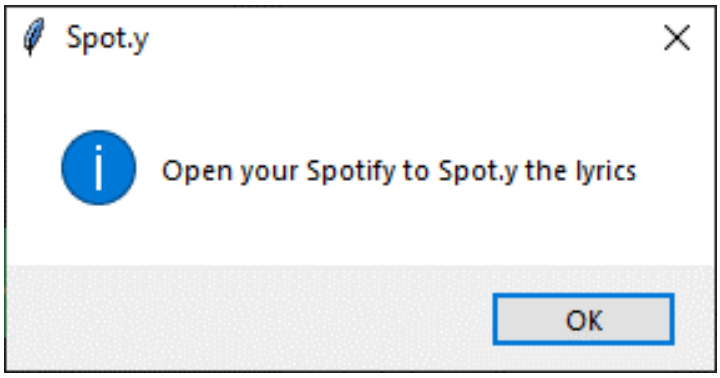


Рисунок 3.10 – Вікно сповіщення, яке інформує користувача, що для здійснення авто-пошуку, треба зробити плеєр Spotify активним вікном

Завершення роботи. Користувач може закрити вікно з текстом пісні після перегляду, також програма містить кнопку для повторного відкриття головного вікна, якщо користувач хоче продовжити пошук текстів пісень.

Таким чином, розроблена програма Spot.y пропонує зручні способи для пошуку текстів пісень з використанням сучасних технологій розробки з взаємодією з активними вікнами системи та обробки графічного інтерфейсу бази даних для отримання текстів композицій. Додаток надає користувачам можливість легко і швидко знаходити тексти улюблених пісень, забезпечуючи при цьому інтуїтивно зрозумілий інтерфейс за використанням функцій автоматичного та текстового пошуку.

Також програма, виводить відповідні вікна, коли їй не вдається знайти текст пісні (Рис. 3.11, 3.12). Причиною можуть бути різні проблеми під час пошуку тексту пісні

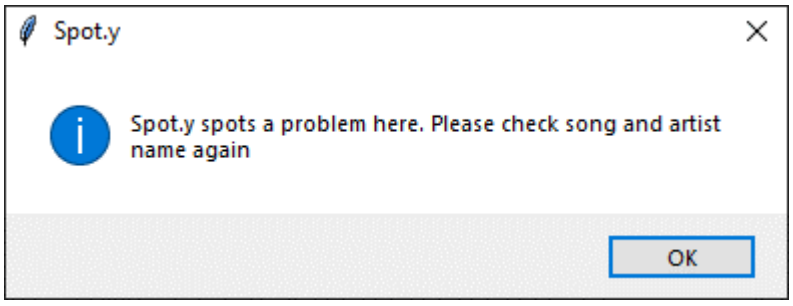


Рисунок 3.11 – Вікно сповіщення, яке з’являється після неможливості знаходження пісні за введеними даними

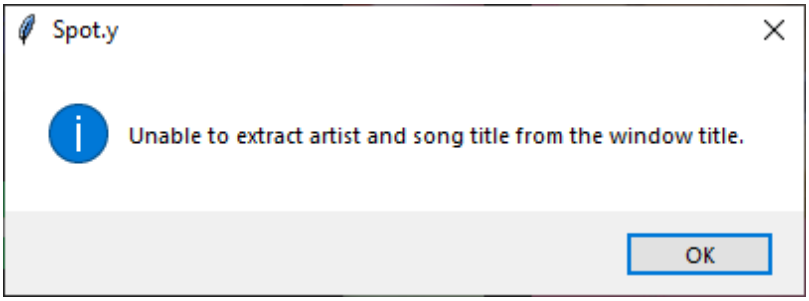


Рисунок 3.12 – Вікно сповіщення, яка з’являється після неможливості зчитування інформації про граючу пісню з активного вікна

Розглянемо детальніше повну структуру програмного застосунку з боку можливих дій користувача, на схемі також описано можливі вікна сповіщень, які будуть відображатися при виникненні помилок у знайденні тексту пісні (Рис. 3.13).

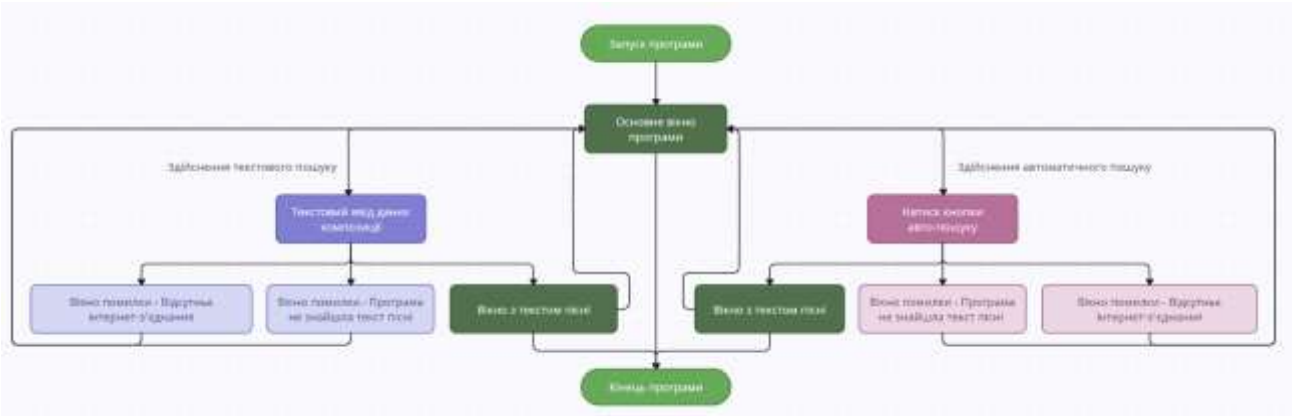


Рисунок 3.13 – Блок-схема програми Spot.y

Розгянемо приклад роботи алгоритму для здійснення пошуку за текстовим вводом:

1. Користувач вводить наступний текст:

- Ім'я виконавця: “Kanye West”

- Назва: “Good Morning”

2. Програма видаляє зайві символі:

- Ім'я виконавця змінено на "kanyewest".

- Назва пісні змінюється на "goodmorning".

3. Створено URL:

- "https://www.azlyrics.com/lyrics/kanyewest/goodmorning.html"

4. Програма надсилає запит і отримує HTML-код сторінки.

5. Програма аналізує розмітку HTML і знаходить текст пісні.

6. Знайдений текст виводиться для користувача у відповідному вікні (Рис. 3.14).

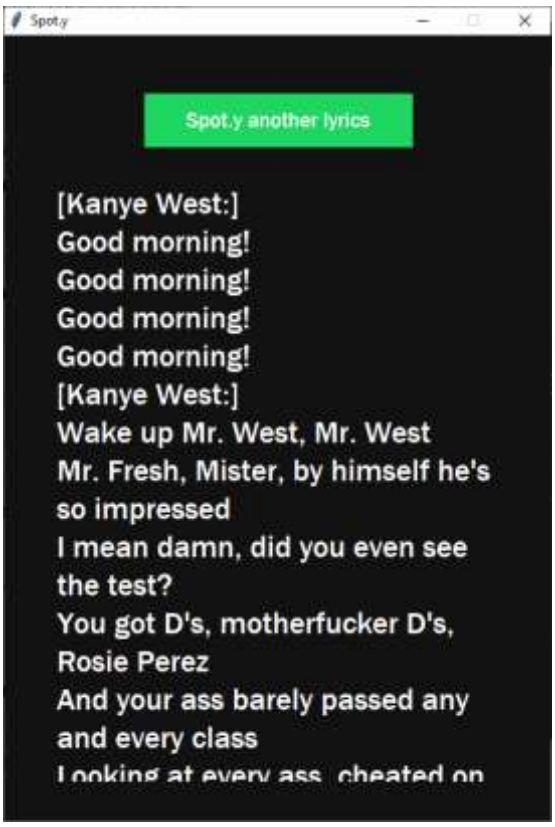


Рисунок 3.14 – Виведений текст пісні, яку програма знайшла за текстовим запитом користувача

Розгянемо приклад роботи алгоритму для здійснення пошуку автоматичним способом:

1. Користувач вмикає пісню у Spotify:

2. Програма просить користувача вибрати Spotify як головне вікно:

3. Програма зчитує з назви вікна плеєру автора та назву пісні (Рис. 3.15):

Рисунок 3.15 – Вікно онлайн-плеєра Spotify, у якому є інформація для пошуку тексту граючої пісні

- Ім'я виконавця: “Kanye West”

- Назва: “Huricane”

2. Програма видаляє зайві символі, у тому числі символ “-” між автором та назвою композиції:

- Ім'я виконавця змінено на "kanyewest".

- Назва пісні змінюється на "hurricane".

3. Створено URL:

- "https://www.azlyrics.com/lyrics/kanyewest/goodmorning.html"

4. Програма надсилає запит і отримує HTML-код сторінки.

5. Програма аналізує HTML і знаходить текст пісні.

6. Знайдений текст виводиться для користувача у відповідному вікні (Рис. 3.16).

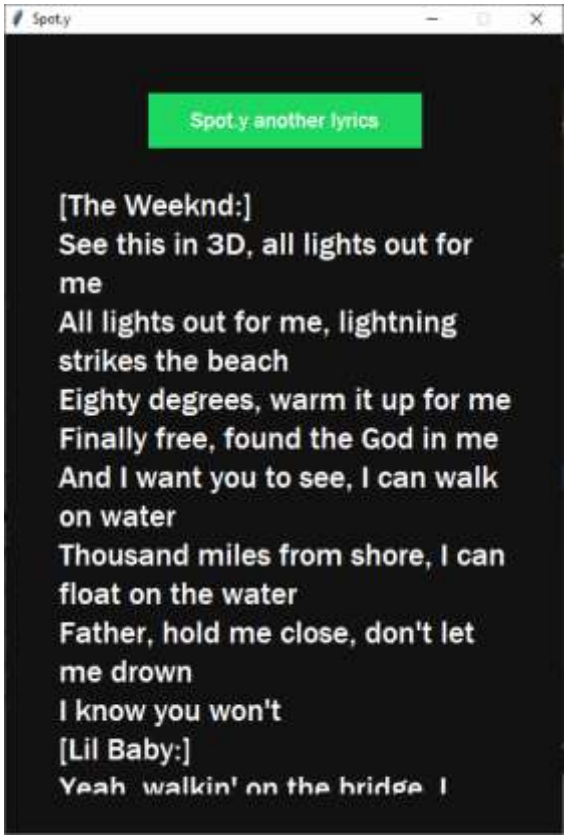


Рисунок 3.16 – Виведений текст пісні, яку програма знайшла за розробленим алгоритмом автоматичного пошуку

Алгоритм дозволяє автоматично знаходити та копіювати тексти пісень із сайту AZLyrics з великою базою даних за допомогою запитів HTTP та аналізу HTML-коду сторінки.

Варто відмітити, шо розроблений алгоритм не вимагає від користувача особливих знань у користуванні комп’ютером, усе що треба зробити це, або ввести інформацію бажаної пісні, або натиснути одну кнопку та переключити активне вікно на Spotify, робота над інтерфейсом є однією з найважливіших частин процесу розробки, оскільки якщо графічний інтерфейс буде зрозумілим та легким у використовуванні, користувачі самі будуть проводити більше часу користуючись даною програмою.

3.4 Переваги та недоліки розробленого додатку

Після проведення якісного аналізу програмного застосунку та досліджень на користувачах, що бачили подібний додаток вперше було зазначено присутність певних переваг та недоліків. Попри це, користувачі оцінювали додаток як корисний, інформативний та легкий у користуванні.

Серед переваг кінцевої версії програмного застосунку варто відмітити:

- Зрозумілість додатку: забезпечує користувача позитивним досвідом від використовування, а також спричиняє зниження порогу розуміння у користуванні, що забезпечує більшу кількість користувачів;
- Швидкодію створених алгоритмів: напряму впливає на користувача, через малу затрату часу на пошук тексту, що дозволяє більше насолоджуватися прослухуванням;
- Винахідливий та унікальний підхід до реалізації поставлених задач: є важливою перевагою для розробника, оскільки у повній мірі реалізує отримані їм теоретичні навички, та дозволяє застосувати творчий авторський підхід;
- Красивий і зрозумілий дизайн: як і зрозумілість сприяє на позитивний досвід від користування додатком, робить його більш пріоритетним у виборі між конкурентами;
- Точність знайдених текстів пісень: особливо впливає на поставленні цілі до розробки, оскільки точність знайдених текстів є пріоритетною серед інших реалізованих можливостей.

Також для легшого завантаження додатку та його просунення серед можливих користувачів, було створено Telegram-канал “Spot.y”, канал містить інформацію про додаток, інструкцію по завантаженню та інструкцію по користуванню (Рис. 3.17).

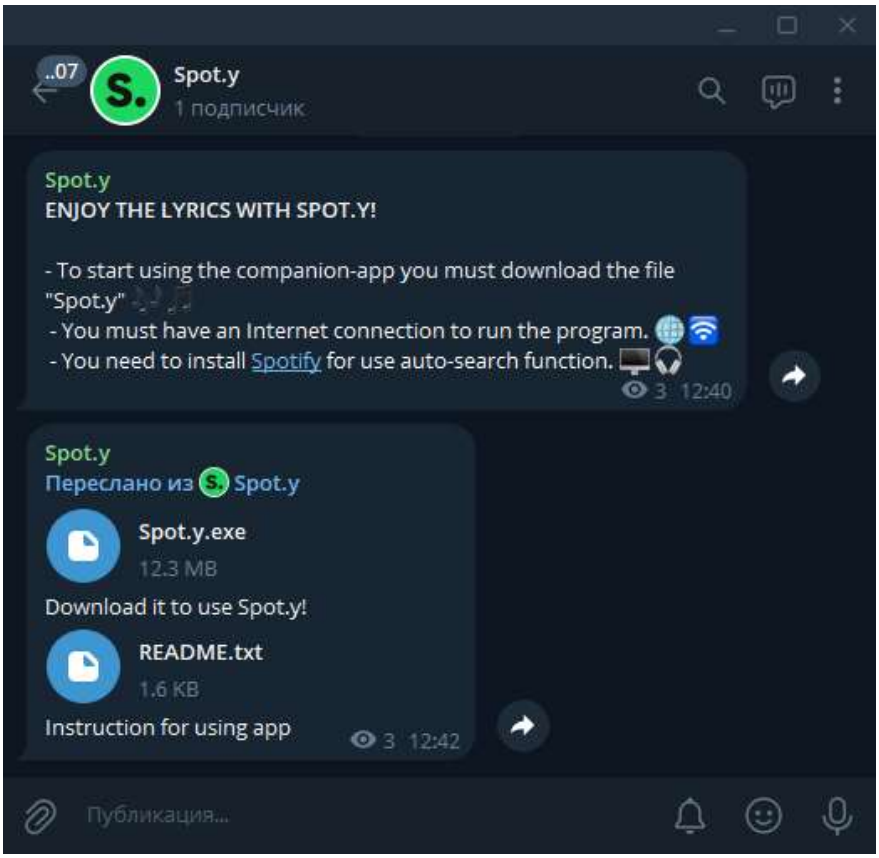


Рисунок 3.17 – Telegram-канал для розповсюдження додатку Spot.y

Інструкція по користуванню знаходиться у файлі “README.txt”, даний документ буде корисним для юзерів, що не ознайомленні з подібними застосунками. У файлі перераховано всі можливі дії для пошуку текстів композицій. Описані інструкції по виконанню автоматичного та текстового пошуку, повністю розписано алгоритм дій, вся інформація, для розширення кількості користувачів, була продубльована українською та англійською мовами (Рис. 3.18).

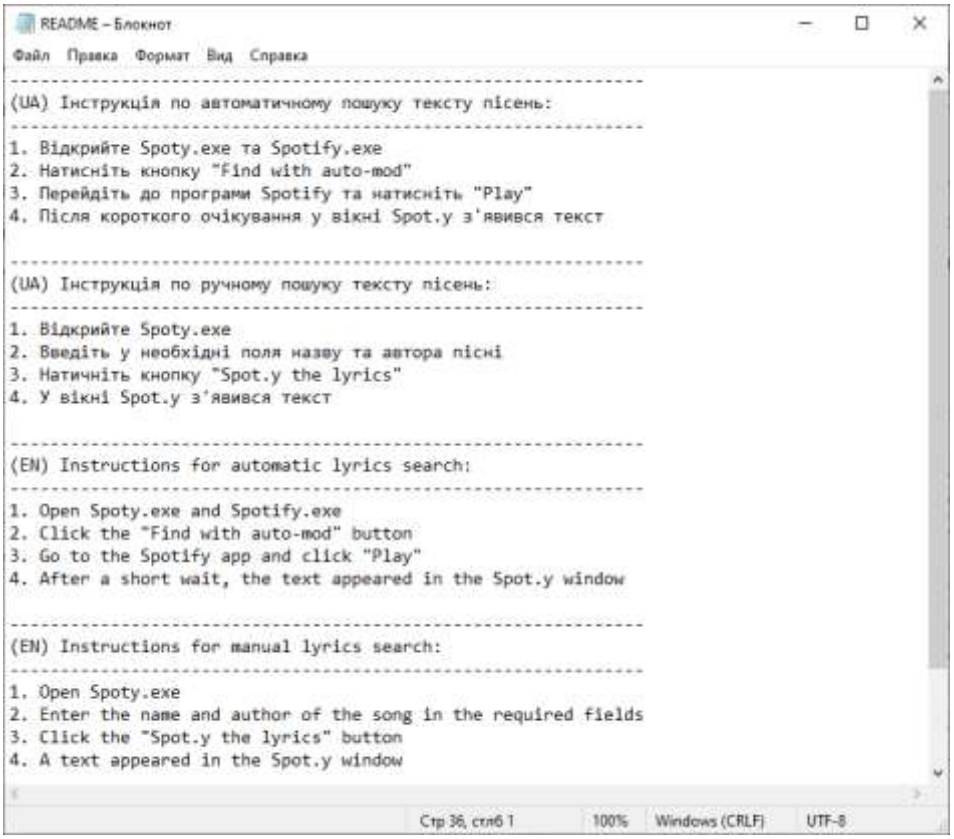


Рис. 3.18 – файл README, що має у собі інструкцію по користуванню алгоритмами пошуку

Після проведеного аналізу та дослідженням позитивних сторін треба звернути увагу на помічені недоліки:

- У наявності є лише одна базу даних з текстами;
- Відсутність вибору мови програми;
- Програму можна завантажити лише з спеціального Telegram-каналу;
- Додаток не підтримує автопошук з інших онлайн-плеєрів.

У ході тестування додатку було виявлено низку недоліків, які впливають на зручність користування. Ці недоліки плануються бути усуненими в найближчих оновленнях. Розроблений додаток планується розвивати, як окремий авторський продукт, який буде постійно підтримуватися та вдосконалюватися. Основною ціллю майбутнього процесу розробки є вдосконалення наявних функцій та розробка нових.

У майбутньому планується впровадити такі нововведення як:

- Підтримка різних баз даних: дозволить користувачам обирати різні бази, оскільки деякі можуть не підтримуватися у певних різних регіонах, або якість та точність тексту, можуть суттєво розрізнятись;
- Відображення обкладинки до знайденої пісні: вплине на обраний графічний стиль інтерфейсу, дозволить краще орієнтуватися користувачам з краще розвиненою візуальною пам’яттю, візуально прикрасить вигляд розробленого проєкт;
- Зберігання історії пошуку: допоможе користувачам повернутись до знайдених раніше текстів, а при підтримці відображенні обкладинок, швидко обирати текст бажаної композиції;
- Розробка сайту додатку: розроблений у майбутньому сайт знадобиться користувачам для отримання інформації про нововведення у програмі, а розробнику отримувати зворотній відгук від користувача.
- Викладення програми на платформу Microsoft Store: викладення проєкту на таку значну та велику платформу стане новим етапом у розробці та підтримки програмного додатку, це рішення сильно вплине на популярність розробленого проєкту, та напряму вплине на кількість користувачів.

3.5 Висновки до третього розділу

Підводячи підсумки до роботи, зробленої у третьому розділі, варто відмітити, що робота над ним була найскладнішою. Під час реалізації обраної мети та поставлених цілей було зроблену великий об’єм роботи:

- Розроблено алгоритм для текстового пошуку тексту пісень, що задіє надійну базу даних у вигляді перевіреного сайту з текстами пісень;
- Створено алгоритм, що є основною метою проєкту, алгоритм для автопошуку за натисканням однієї кнопки;
- Було проведено значну роботу щодо аналізу розроблених можливостей, шляхом опитувань визначено переваги та недоліки, помічені користувачами;
- Визначено поетапний план майбутнього проєкту, аналіз необхідного функціоналу для майбутніх версій;
- Розроблено, а потім реалізовано графічний дизайн, що є авторською розробкою на основі багатьох референсів та ідей;
- Вигадано назву для додатку та створено логотип, на основі визначеного дизайну та унікальній назві.

Креативне реалізування визначених цілей повністю зобов’язане мові програмування Python, її зручність та можливості, у вигляді великої кількості різноманітних бібліотек, повністю виправдали рішення, щодо обрання середовища розробки. Отримані під час навчання знання допомогли визначити необхідні бібліотеки та додатку, що були задіяні у процесі розробки. Так іконка проєкту була виконана у програмному застосунок Adobe Illustrator, за додатковим користуванням Adobe Photoshop, який у свою чергу використовувався для створення макету дизайну. А обрані додаткові пакети Python дозволили реалізувати створення додатку без інтеграції в проєкт сторонні API.

На закінчення слід зазначити, що у даному розділі було продемонстровано, що розроблена програма має великий потенціал, оскільки

показана результативність вказує на значний потенціал у майбутньому розвиненні програми, як окремого самостійного продукту, а не маленького проєкту. Актуальність зроблених дій підтверджується отриманням унікальних знань та досвіду зі сторони розробника, а винахідлива реалізація деяких функцій остаточно виділила програмний застосунок на фоні конкурентів, що є схожими за деякими своїми можливостями.

Користувачеві додаток значно спростить процес пошуку текстів пісень, особливо функцією автоматичного пошуку, яка була створена саме для користувачів Spotify, Інноваційний підхід до створення функцій автопошуку та зручного інтерфейсу робить цей додаток конкурентоспроможним та корисним інструментом для меломанів та звичайних користувачів.

Даний додаток є необхідністю для тих, хто бажає знаходити текст пісень автоматично, за натиском однієї кнопки. Це досягається за рахунок інноваційного підходу, який автоматизував процес пошуку. У підсумку варто зазначити, що результатом пророблені роботи є поява одного з небагатьох додатків, який для desktop-користувачів у майбутньому безпосередньо стане єдиним вірним варіантом у обранні програми для пошуку текстів пісень.

ВИСНОВКИ

У ході виконаного процесу розробки було створено корисна програма для щоденного користування юзерами з різних кутків Землі, розроблювальний проєкт є компаньйонським застосунком до музичного потокового онлайн-сервісу Spotify.

У результаті виконання дипломної роботи було досліджено можливості для розробки програмного забезпечення для пошуку текстів пісень. В процесі дослідження багатьох способів реалізації задуманого проєкту було прийнято розроблювати застосунок без використання API, лише використовуючи мову програмування Python та її бібліотеки, у результаті був створений програмний продукт “Spot.y”.

Проведені дослідження мови Python, та аналізи наявних до неї бібліотек показали, що вибір даної мови програмування був правильним рішенням. Обрані бібліотеки дозволили спланувати роботу програмних функцій, та реалізувати алгоритм автопошуку використовуючи творчий та нестандартний підхід.

Вивчення подібних, до розроблювальної, програм дозволив розробити правильний та функціональний план дизайну додатку. Для цього було завантажено багато онлайн-сервісів з прослуховування музики, найбільший вплив зумовив додаток Spotify, програма-компаньйон Spot.y має його кольорову палітру, іконка була розроблена з оглядом на приклад, а назва відсилає на оригінальний застосунок. Розробка макету інтерфейсу та розробка іконки виконувалася у додатках Adobe Photoshop та Adobe Illustrator. Варто відмітити, що розроблений дизайн, опитувані користувачі, знайшли приємним, мінімалістичним та інтуїтивно зрозумілим, таким, що дозволяє одразу зрозуміти, як їм користуватися.

Під час розробки особлива увага була присвячена розробці алгоритмів та простотою їх у користуванні, для цього були проведені опитування, результати котрих були використанні у вдосконаленні застосунку та

плануванні майбутнього його функціоналу. Програма пропонує юзерам два способи для знаходження текстів улюблених пісень:

- Ручний пошук пропонує знаходити тексти за вводом назви композиції та імені її виконавця, даний варіант працює у всіх користувачів, для його роботи не обов'язково мати якісь конкретні плеєри з програвання пісень.
- Автоматичний пошук пропонує знаходити тексти за натиском однієї кнопки, дана функція вимагає користувача мати завантаженим додаток Spotify.

Практичною ціною розробленого проєкту є розважання та збільшення світогляду користувачів шляхом ознайомлення з текстом їх найулюбленіших композицій. У майбутньому додаток розширить способи взаємодії з юзером, серед запланованих ідей варто відмітити такі як:

- Підтримка різних баз даних – дозволить користувачам обирати різні джерела з текстами пісень.
- Відображення обкладинок пісень – зображення обкладинок до композицій візуально прикрасить дизайн та вдосконалить досвід користувачів під час здійснюваного пошуку.
- Зберігання історії пошуку – дана функція дозволить побачити список останніх знайдених треки та переглянути їх текст.
- Сайт додатку – наявність сайту допоможе користувачам передивлятися інформацію про останні оновлення та надасть можливість звернутися до розробника.
- Просунення на платформі Microsoft Store – поява додатку у Microsoft Store сильно вплине на розповсюдженість програми.

На момент написання дипломної роботи, створений проєкт повністю відповідає меті, темі та всім поставленим цілям. Проведені тестування на інших користувачах демонструють повну актуальність розробленого програмного застосунку. На підставі проведеного аналізу розробленої програми та її тестуванням, можна прийти до висновків, що програма показала

себе як потужний продукт, що готовий вийти на ринок застосунків та бути підтримуваним у майбутньому. При оцінці корисності створеної програми також варто враховувати, що наданий додатком функціонал, що згодом буде тільки розширюватися, буде корисним, як для поціновувачів музики, так і для звичайних людей.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Erikkson M. Spotify Teardown: Inside the Black Box of Streaming Music / M. Erikkson., 2017.

2. Карлссон С. Електронна книга Spotify навиворіт. Як шведський стартап здійснив музичну революцію / Carlsson Sven., 2021.

3. Matthes E. Python Crash Course, 3rd Edition: A Hands-On, Project-Based Introduction to Programming / E. Matthes, 2023.

4. Reed M. Python Programmer's Toolkit: The Essential Tools And Libraries That Every Python Developer Should Know About / M. Reed., 2024.

5. Васильєв О. Програмування мовою Python / О. Васильєв., 2019.

6. Tkinter Manual. URL: https://docs.python.org/uk/3/library/tkinter.html.

7. BeautifulSoup Manual. URL: https://pypi.org/project/beautifulsoup4/.

8. Re Manual. URL: https://docs.python.org/uk/3/library/re.html..

9. Pygetwindow Manual. URL: https://pypi.org/project/PyGetWindow/.

10. Threading Manual. URL: https://docs.python.org/uk/3/library/threading.html.

11. Mihajlov M. HTML QuickStart Guide: The Simplified Beginner's Guide To HTML / ClydeBank Technology., 2015.

ДОДАТОК

Код розробленого додатку “Spot.y”:

```
import tkinter as tk
from tkinter import messagebox
import requests
from bs4 import BeautifulSoup
import re
import pygetwindow as gw
import threading
import time

lyrics_window = None
watching_thread = None

def get_lyrics(artist: str, title: str) -> str:
    artist = remove_special_characters(artist).lower()
    title = remove_special_characters(title).lower()
    url = f"https://www.azlyrics.com/lyrics/{artist}/{title}.html"

    try:
        response = requests.get(url)
        if response.status_code == 200:
            soup = BeautifulSoup(response.content, 'html.parser')
            lyrics_div = soup.find_all("div", class_=None, id=None,
limit=20)[0]
            lyrics = re.sub(r'[\r\n]+', '\n', lyrics_div.text.strip())
            return lyrics
        else:
            return None
    except Exception:
        return None

def remove_special_characters(text: str) -> str:
    return re.sub(r'[^a-zA-Z0-9]+', '', text)

def clear_entry(event: tk.Event, entry: tk.Entry, default_text: str) -> None:
    if entry.get() == default_text:
        entry.delete(0, tk.END)

def restore_entry(event: tk.Event, entry: tk.Entry, default_text: str) ->
None:
    if not entry.get():
```

```
            entry.insert(0, default_text)

    def close_root_and_open_lyrics() -> None:
        global lyrics_window
        root.withdraw()
        song_title = title_entry.get()
        artist_name = artist_entry.get()
        lyrics = get_lyrics(artist_name, song_title)

        if lyrics:
            lyrics_window = tk.Toplevel()
            lyrics_window.title("Spot.y")
            lyrics_window.geometry("487x700")
            lyrics_window.resizable(False, False)
            lyrics_window.config(bg="#121212")

            show_another_button = tk.Button(lyrics_window, text="Spot.y another
lyrics", font=("Franklin Gothic Medium", 14),
                                            command=close_lyrics_and_open_root,
bg="#1fd760", fg="#FFFFFF", bd=0, highlightthickness=0)
            show_another_button.place(relx=0.5,  y=52,  anchor=tk.N,  width=239,
height=48)

            lyrics_text = tk.Text(lyrics_window, wrap=tk.WORD, font=("Franklin
Gothic Medium", 20), bg="#121212", fg="#FFFFFF", bd=0, highlightthickness=0)
            lyrics_text.place(relx=0.5,    y=132,    anchor=tk.N,    width=397,
height=534)

            lyrics_text.insert(tk.END, lyrics)
            lyrics_text.config(state=tk.DISABLED)
        else:
            messagebox.showinfo("Spot.y", "Spot.y spots a problem here. Please
check song and artist name again")
            root.deiconify()

    def close_lyrics_and_open_root() -> None:
        lyrics_window.destroy()
        root.deiconify()

    def auto_spot_y() -> None:
        messagebox.showinfo("Spot.y", "Open your Spotify to Spot.y the lyrics")
        global watching_thread
        watching_thread      =      threading.Thread(target=watch_active_window,
daemon=True)
```

```
            watching_thread.start()

        def watch_active_window() -> None:
            time.sleep(5)
            active_window = gw.getActiveWindow()
            if active_window:
                title = active_window.title
                try:
                    artist, song_title = title.split(" - ", 1)
                    lyrics = get_lyrics(artist, song_title)
                    root.after(0,     lambda:     show_lyrics_in_new_window(artist,
song_title, lyrics))
                except ValueError:
                    messagebox.showinfo("Spot.y", "Unable to extract artist and song
title from the window title.")
            else:
                messagebox.showinfo("Spot.y", "No active window found.")

        def show_lyrics_in_new_window(artist: str, song_title: str, lyrics: str) ->
None:
            global lyrics_window
            root.withdraw()
            if lyrics:
                lyrics_window = tk.Toplevel()
                lyrics_window.title("Spot.y")
                lyrics_window.geometry("487x700")
                lyrics_window.resizable(False, False)
                lyrics_window.config(bg="#121212")

                show_another_button = tk.Button(lyrics_window, text="Spot.y another
lyrics", font=("Franklin Gothic Medium", 14),
                                                command=close_lyrics_and_open_root,
bg="#1fd760", fg="#FFFFFF", bd=0, highlightthickness=0)
                show_another_button.place(relx=0.5,  y=52,  anchor=tk.N,  width=239,
height=48)

                lyrics_text  =  tk.Text(lyrics_window,  wrap=tk.WORD,  font=("Franklin
Gothic Medium", 20), bg="#121212", fg="#FFFFFF", bd=0, highlightthickness=0)
                lyrics_text.place(relx=0.5,    y=132,    anchor=tk.N,    width=397,
height=534)

                lyrics_text.insert(tk.END, lyrics)
                lyrics_text.config(state=tk.DISABLED)
            else:
```

```
            messagebox.showinfo("Spot.y", "Spot.y spots a problem here. Please
check song and artist name again")
            root.deiconify()

    # Создаем главное окно
    root = tk.Tk()
    root.title("Spot.y")
    root.geometry("350x400")
    root.resizable(False, False)
    root.config(bg="#121212")

    input_frame = tk.Frame(root, padx=20, pady=30, bg="#121212")
    input_frame.pack()

    title_default_text = "Song"
    title_entry = tk.Entry(input_frame, font=("Franklin Gothic Medium", 20),
width=30, justify="center", bd=0, highlightthickness=0)
    title_entry.insert(0, title_default_text)
    title_entry.bind('<FocusIn>', lambda event, entry=title_entry:
clear_entry(event, entry, title_default_text))
    title_entry.bind('<FocusOut>', lambda event, entry=title_entry:
restore_entry(event, entry, title_default_text))
    title_entry.pack(pady=5, ipady=10)
    title_entry.config(bg="#242424", fg="#FFFFFF")

    artist_default_text = "Artist"
    artist_entry = tk.Entry(input_frame, font=("Franklin Gothic Medium", 20),
width=30, justify="center", bd=0, highlightthickness=0)
    artist_entry.insert(0, artist_default_text)
    artist_entry.bind('<FocusIn>', lambda event, entry=artist_entry:
clear_entry(event, entry, artist_default_text))
    artist_entry.bind('<FocusOut>', lambda event, entry=artist_entry:
restore_entry(event, entry, artist_default_text))
    artist_entry.pack(pady=5, ipady=10)
    artist_entry.config(bg="#242424", fg="#FFFFFF")

    button_frame = tk.Frame(root, bg="#121212")
    button_frame.pack(pady=(10, 20))

    search_button = tk.Button(button_frame, text="Spot.y the lyrics",
font=("Franklin Gothic Medium", 14), command=close_root_and_open_lyrics,
                              anchor="center", padx=20, pady=5, bd=0,
highlightthickness=0)
    search_button.pack(pady=(0, 20))
```

```
        search_button.config(bg="#1fd760", fg="#FFFFFF", width=20, height=1)

        auto_spot_y_button = tk.Button(button_frame, text="Find with auto-mod",
font=("Franklin Gothic Medium", 14), command=auto_spot_y,
                                       anchor="center", padx=20, pady=5, bd=0,
highlightthickness=0)
        auto_spot_y_button.pack()
        auto_spot_y_button.config(bg="#1fd760", fg="#FFFFFF", width=20, height=1)

        developer_label = tk.Label(root, text="Software was developed by Aleksey
Kabanov", font=("Franklin Gothic Medium", 9),
                                   bg="#121212", fg="#FFFFFF", justify="center")
        developer_label.pack(side=tk.BOTTOM, pady=(0, 10))

        root.mainloop()
```