Пояснювальна записка
до курсової роботи

на тему: Голосовий помічник

КПІ.ІП-1315.045490.02.81

Київ – 2023

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

IDE	–	Integrated Development Environment – інтегроване середовище розробки.
JDK	–	Java development kit
JAR	–	Java archive
ПЗ	–	Програмне забезпечення
ГП	–	Голосовий помічник
ШІ	–	Штучний інтелект

ВСТУП

У сучасному цифровому світі голосові помічники (ГП) стають неодмінною складовою повсякденного життя, спрощуючи взаємодію людей з технологіями та надаючи широкі можливості для зручного отримання інформації та виконання різноманітних завдань. Актуальність вивчення та розробки ГП базується на швидкому розвитку галузі штучного інтелекту, зростанні інтересу до голосового, дистанційного інтерфейсу та збільшенні можливостей технологій обробки природної мови[1].

Світові тенденції показують постійне удосконалення ГП у напрямку поліпшення їхньої розпізнавальної точності, розширення функціональності та інтеграції з іншими інформаційними системами. Поява нових технологій, таких як машинне навчання та нейронні мережі, створює нові можливості для розробки більш ефективних та інтелектуальних ГП[2].

Однак в контексті даної курсової роботи акцент робиться на відмінності від наявних ГП. Велика частина наявних рішень є комерційними продуктами з обмеженим доступом до внутрішніх механізмів та алгоритмів. Даний ГП відрізняється тим, що він базується на принципах відкритого програмного забезпечення та вільної ліцензії, надаючи користувачам можливість активної участі в розвитку та вдосконаленні системи.

Метою даної курсової роботи є ретельний аналіз сучасного стану ГП, визначення перспектив розвитку та розробка концепції нового голосового помічника, спрямованого на задоволення потреб користувачів у контексті швидкозмінного технологічного середовища. У роботі буде розглянуто сфери застосування ГП та обґрунтовано його актуальність у різних галузях життя та бізнесу.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

В умовах сучасного інформаційного середовища розробка програмного забезпечення визнається як ключовий фактор для вирішення широкого спектра завдань та оптимізації бізнес-процесів. Початковий етап цього процесу — аналіз вимог, що визначає основні потреби та очікування від майбутнього продукту.

Аналіз вимог — це важливий крок, на якому встановлюються як функціональні, так і нефункціональні характеристики програмного продукту. Цей етап визначає ключові параметри та особливості продукту, що є визначальними для його успішної розробки та функціонування.

У даній курсовій роботі розглядаються загальні аспекти визначення вимог, їх класифікація та вплив на подальший процес розробки. Аналізується досвід теперішніх голосових помічників, зокрема їх функціональні можливості та нефункціональні характеристики, що стає основою для формулювання вимог до нового голосового помічника з відкритим вихідним кодом. Особливий акцент робиться на забезпеченні прозорості та спільної участі у процесі його розвитку.

1.1 Загальні положення

У цьому розділі представлено всебічний огляд голосових помічників, досліджено загальні положення, визначено ключові поняття та проаналізовано тенденції розвитку цієї галузі.

Поява голосових помічників є важливою віхою на перетині штучного інтелекту (ШІ) та людино-машинної взаємодії. Ці інтелектуальні програми стали повсюдними, вбудованими в різні пристрої - від смартфонів до розумних колонок і натільних пристроїв. З розвитком технологій можливості голосових помічників розвиваються, впливаючи на різні аспекти нашого повсякденного життя[3].

Концепція голосової взаємодії з машинами виникла кілька десятиліть тому, коли ранні експерименти та прототипи проклали шлях до складних голосових помічників, з якими ми стикаємося сьогодні. Простеження історичної траєкторії дає цінну інформацію про еволюційний шлях цих технологій.

Голосовий асистент (помічник) - це інтелектуальне програмне забезпечення, яке реагує на голосові команди, використовуючи штучний інтелект і машинне навчання[4].

Прикладами таких програм є Alexa, Siri, Google Assistant і Bixby, які можна знайти на таких пристроях, як смартфони та смарт-колонки. Вони виконують різні завдання - від дзвінків до керування пристроями розумного дому[5].

Ключові компоненти ГП включають перетворення мови в текст, тексту в мову, прийняття рішень і базову архітектуру[4].

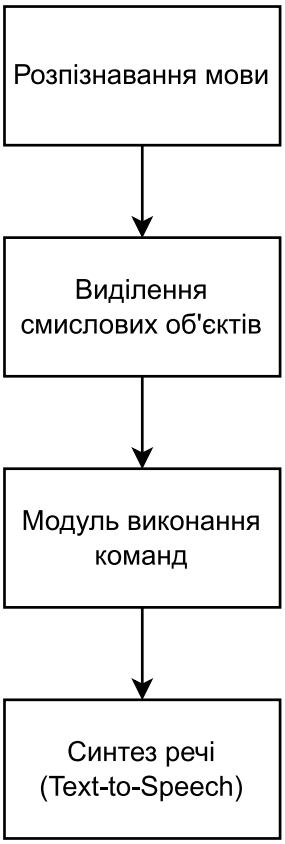

Рисунок 1.1. — Загальна схема роботи ГП

Голосові асистенти забезпечують зручну та ефективну взаємодію завдяки обробці природної мови.

Їх впровадження може стати значущим кроком у напрямку зручності та ефективності в щоденному житті, наприклад:

– Залишатися в курсі подій - доступ до широкого спектра інформації, доступної в Інтернеті, дізнатися про погоду, отримати останні новини, перевірити інформацію про дорожній рух тощо.

– Керування музикою - наказуємо їм відтворювати музику, яка відповідає нашому поточному настрою.
– Керування пристроями - підключення до інших розумних пристроїв і виконання команд, таких як "увімкнути телевізор".
– Фінансові оновлення - доступ до інформації про банківський рахунок, перевірка балансу, перегляд транзакцій і навіть запит продуктів, таких як картки.
– Керування розкладом - ефективна організація часу без використання мобільного пристрою, годинника чи щоденника.
– Підтримка доступності - сприяння людям з вадами зору, людям похилого віку та залежним особам за допомогою функції голосового пошуку.
– Посилена безпека - інтеграція домашніх камер безпеки з можливістю віддаленого перегляду прямих трансляцій на наших пристроях.

З іншого боку, голосові асистенти мають свої недоліки, які важливо враховувати при їхньому використанні. До числа ключових труднощів входять:

– Занепокоєння щодо конфіденційності - постійна функціональність означає постійне прослуховування, що підвищує ймовірність запису розмов.
– Доступ до персональних даних - надання доступу до облікових записів електронної пошти, списків контактів та іншої особистої інформації, необхідної для роботи асистентів.
– Знайомство з нашими звичками - вони вловлюють набагато більше, ніж те, що ми говоримо і просимо. Вони також можуть витягувати дані з підключених до них пристроїв і збирати інформацію про те, о котрій годині ми повертаємося додому, та інші дані.
– Ризик крадіжки особистих даних - попри те, що вони розпізнають лише наш голос, ризик проникнення хакерів та отримання ними доступу до персональних даних залишається.

1.2 Змістовний опис і аналіз предметної області

На сучасному етапі розвитку IT-технологій голосові помічники є важливою складовою нашого цифрового середовища. Однак існують певні недоліки та виклики, які потребують уваги та подальшого вдосконалення, серед них[6]:

- Низька точність розпізнавання мови - деякі голосові помічники можуть некоректно розпізнавати акценти чи нестандартне мовлення, що призводить до помилок у виконанні команд.
- Обмежені можливості розуміння контексту - голосові асистенти іноді не можуть ефективно узгоджувати різні запитання та відповіді у більшому контексті, що призводить до неповних чи неправильних відповідей.
- Питання конфіденційності та безпеки - збір та обробка великого обсягу особистих даних створює загрозу конфіденційності та можливість несанкціонованого доступу до цих даних.

В межах виконання курсової роботи обрано напрямок забезпечення безпеки та конфіденційності за допомогою відкритого вихідного коду, відмови від зберігання в базі даних інформації про користувачів та постійного контролю за прослуховуванням. Основний акцент робиться на забезпеченні користувачів контролем над процесом та відсутності комерційних аспектів[7].

Заходи забезпечення безпеки та конфіденційності включають:

- Використання відкритого вихідного коду - активне використання відкритого вихідного коду надає можливість експертам і громаді перевіряти код на наявність можливих вразливостей та документувати, як сам голосовий помічник обробляє дані користувачів.
- Відмова від зберігання в базі даних - інформація про користувачів не зберігається в базі даних, що мінімізує ризик витоку конфіденційної інформації через можливі порушення безпеки бази даних.
- Користувач-центричний підхід - користувачі мають повний контроль над процесом прослуховування. Вони можуть визначати момент активації голосового помічника та його зупинки.

– Відсутність комерційних цілей - проєкт фокусується на вирішенні завдань без наміру отримання прибутку, що убезпечує від використання даних користувачів у комерційних цілях.

1.3 Аналіз існуючих технологій та успішних IT-проєктів

Проаналізуємо відоме на сьогодні алгоритмічне забезпечення у даній області та технічні рішення, що допоможуть у реалізації голосового помічнику. Далі будуть розглянуті допоміжні програмні засоби, засоби розробки та готові програмні рішення.

1.3.1 Аналіз відомих алгоритмічних та технічних рішень

У світі голосових помічників визначені ключові алгоритми для розпізнавання мовлення та обробки голосових команд. Розглянемо основні алгоритми, які використовуються у відомих голосових помічниках.

– Google Speech-to-Text - Використовує глибоке навчання для розпізнавання мовлення.

  Забезпечує високу точність та широку підтримку мов, це надійний інструмент транскрипції для глобальної аудиторії з підтримкою понад 125 мов[8].

  Він пропонує розпізнавання потокового мовлення в реальному часі, адаптивність до специфічних термінів і завадостійкість. Користувачі можуть керувати інфраструктурою за допомогою локальної опції та вибирати з моделей для конкретного домену.

  Функції включають багатоканальне розпізнавання, фільтрацію контенту, автоматичну пунктуацію (бета-версія) та діалогізацію мовлення (бета-версія). Ціноутворення базується на кількості обробленого аудіо щомісяця, що вимірюється з кроком в одну секунду, забезпечуючи прозору та гнучку структуру витрат[9].

– CMUSphinx - Робить акцент на локальному розпізнаванні мовлення, що може бути корисним для деяких сценаріїв.

  Забезпечує можливість роботи без підключення до Інтернету. CMUSphinx є провідним рішенням для розпізнавання мови з численними перевагами.

Завдяки найсучаснішим алгоритмам, він вирізняється ефективним розпізнаванням мовлення, пристосованим до платформ з обмеженими ресурсами. Його гнучкий дизайн робить акцент на розробці практичних додатків, а не на чистих дослідженнях, що робить його цінним інструментом для різноманітних проектів.

CMUSphinx підтримує кілька мов, зокрема англійську, англійську для США, англійську для Великобританії, французьку, китайську, німецьку, голландську тощо, з можливістю побудови моделей для інших мов. Платформа працює за ліцензією, що дозволяє комерційне розповсюдження, і пропонує комерційну підтримку. Завдяки активній розробці та графіку випусків, активній спільноті з понад 400 користувачів у групі LinkedIn CMUSphinx, а також ряду інструментів для різних цілей розпізнавання мови, таких як визначення ключових слів, вирівнювання та оцінка вимови, CMUSphinx виділяється як надійне та універсальне рішення для потреб розпізнавання мови[10].

– DeepSpeech (Mozilla) - Використовує нейронні мережі для глибокого розпізнавання мовлення.

Open-source проект, що надає гнучкість для модифікацій та удосконалень.

Має на меті створити доступну та відкриту систему розпізнавання мовлення, призначену для роботи на різних платформах і не потребуючу висококласного обладнання.

Випущений під ліцензією Mozilla Public License, він використовує рекурентну нейронну мережу для обробки мовних спектрограм і генерування транскрипцій англійського тексту.

Завдяки наявності попередньо навчених моделей і можливості користувачам навчати свої власні, DeepSpeech підтримує різні мови і платформи. Документація охоплює створення бінарних файлів, декодування, підрахунок балів та методи оптимізації, що робить його універсальним і зручним у використанні рішенням. API надаються для C, .NET Framework, Java, JavaScript (NodeJS/ElectronJS) та Python, з прикладами використання для кожної мови, що сприяє досягненню мети DeepSpeech - широкому використанню[11].

– Wit.ai - Використовується для розпізнавання не тільки тексту, але і сутностей та інтентів у мовленні.

  Частина компанії Meta, пропонує зручну платформу для створення та навчання додатків для розпізнавання мови. Користувачі можуть навчати програму на прикладах, пов'язуючи наміри з висловлюваннями, щоб керувати процесом навчання. Система вдосконалюється з більшою кількістю прикладів і дозволяє безперервну перевірку.

  Користувачі можуть запитувати свій додаток через Wit.ai API, налаштовуючи рівні достовірності намірів для точного налаштування.

  Підтримуючи сутності для вилучення конкретної інформації, Wit.ai робить акцент на практичних кроках, таких як створення намірів та сутностей[12].

  Платформа полегшує безперешкодну розробку та навчання додатків для розпізнавання мови. Екосистема Wit.ai заохочує співпрацю спільноти, надаючи доступ до знань, ресурсів та інструментів для прискорення проектів з розпізнавання природної мови.

В якості технічного рішення було обрано CMUSphinx через його акцент на локальному розпізнаванні мовлення та можливості працювати без підключення до Інтернету. Такі особливості роблять CMUSphinx оптимальним вибором для проектів, де важливо забезпечити ефективне розпізнавання мовлення в умовах обмежених ресурсів та забезпечити незалежність від мережі для нормальної роботи. Крім того, гнучкий дизайн та активна розробка роблять CMUSphinx надійним та універсальним рішенням для різноманітних проектів.

1.3.2 Аналіз допоміжних програмних засобів та засобів розробки

В даному пункті розглядається низка допоміжних програмних засобів, які були вивчені та проаналізовані для використання у розробці проекту. До цих засобів включаються мови програмування, інтегровані середовища розробки (IDE) та бібліотеки для виконання завдань проекту.

1.3.2.1 Мови програмування

– Java

Обрана через свою платформенну незалежність, високу надійність, та велику спільноту розробників. Вище інших мов за стабільністю та широким спектром застосувань[13].

– Python

Зручна для роботи з різноманітними бібліотеками та широким спектром задач, включаючи обробку даних та штучний інтелект. Гнучка та зручна, але може вимагати додаткових оптимізацій для великих проектів[14].

1.3.2.2 Інтегровані середовища розробки (IDE)

– IntelliJ IDEA

Можливості автоматизації, висока продуктивність та зручний інтерфейс. Зручна та потужна, надає багато інструментів для комфортної розробки[15].

– Eclipse

Відкрите програмне забезпечення, широко використовується для розробки різноманітних застосунків. Має велику спільноту користувачів, але може бути менш продуктивною порівняно з IntelliJ IDEA[16].

1.3.2.3 Бібліотеки

– Sphinx4

Бібліотека для розпізнавання мови та обробки голосу. Висока точність розпізнавання та широкий функціонал[17].

– Google Speech Recognition API

Хмарна платформа для розпізнавання мови в реальному часі. Може вигідно використовуватися для великих обсягів даних та високої швидкості розпізнавання[18].

– Swing

  Бібліотека для створення графічного інтерфейсу користувача. Проста у використанні та добре інтегрується з Java, але існують більш сучасні альтернативи[19].

– JavaFX

  Сучасна бібліотека для створення графічного інтерфейсу та багатофункціональних додатків. Забезпечує більш сучасний та гнучкий інтерфейс, але може вимагати більше зусиль для вивчення[20].

– FreeTTS

  Бібліотека для генерації голосу. Використовується для створення текст-на-мову та імітації голосу. Додає можливість генерації голосу, що може бути корисним для реалізації голосового інтерфейсу та взаємодії з користувачем[21].

Після ретельного аналізу визначено, що оптимальним вибором для даного проекту є використання мови програмування Java, інтегрованої середовищі розробки IntelliJ IDEA, бібліотек Sphinx4 для розпізнавання мовлення, бібліотеки Swing для створення графічного інтерфейсу користувача, та бібліотеки FreeTTS для генерації голосу. Цей вибір базується на їхній високій продуктивності, надійності та специфічних функціях, необхідних для успішної реалізації голосового інтерфейсу та взаємодії з користувачем.

1.3.3 Аналіз відомих програмних продуктів

Розглянувши наявні успішні IT-проекти, складемо таблиці порівняння.

1.3.3.1 Amazon Alexa

Amazon Alexa - це голосовий асистент, розроблений компанією Amazon. Він взаємодіє з користувачем через голосові команди, надаючи інформацію, виконуючи завдання та контролюючи підключені пристрої[22].

Таблиця 1.1. — Порівняння з аналогом (Alexa)

Функціонал	Vocalia	Alexa	Пояснення
Розпізнавання голосових команд	Так	Так	Обидва продукти мають можливість розпізнавання голосу та виконання команд.
Забезпечення інформацією	Так	Так	Обидва можуть надавати інформацію користувачам з різних джерел.
Управління підключеними пристроями	Ні	Так	Alexa може керувати підключеними пристроями у домашній мережі, в той час як Vocalia - ні.

1.3.3.2 Siri

Siri - голосовий асистент, розроблений Apple для операційних систем (ОС) iOS, iPadOS, macOS та watchOS. Вона використовується для виконання завдань та отримання інформації через голосові команди[23].

Таблиця 1.2. — Порівняння з аналогом (Siri)

Функціонал	Vocalia	Siri	Пояснення
Розпізнавання голосових команд	Так	Так	Обидва продукти мають можливість розпізнавання голосу та виконання команд.
Інтеграція з Apple-продуктами	Ні	Так	Siri може взаємодіяти з іншими пристроями в екосистемі Apple, в той час як Vocalia - ні.
Особистий асистент	Так	Так	Обидва продукти можуть служити особистими асистентами для користувача.

1.3.3.3 Google Assistant

Google Assistant - голосовий асистент, розроблений Google. Він взаємодіє з користувачем через голосові команди, надаючи інформацію, керуючи пристроями та виконуючи завдання[24].

Таблиця 1.3. — Порівняння з аналогом (Google Assistant)

Функціонал	Vocalia	Google Assistant	Пояснення
Розпізнавання голосових команд	Так	Так	Обидва продукти мають можливість розпізнавання голосу та виконання команд.
Інтеграція з Google-продуктами	Частково	Так	Google Assistant може інтегруватись з іншими послугами та пристроями від Google, Vocalia також інтегрується з деякими Google-продуктами.
Пошук інформації онлайн	Так	Так	Обидва можуть шукати та надавати інформацію з Інтернету.

1.3.3.4 Bixby

Таблиця 1.4. — Порівняння з аналогом (Bixby)[25]

Функціонал	Vocalia	Bixby	Пояснення
Розпізнавання голосових команд	Так	Так	Як і дипломний проєкт, Bixby може розпізнавати голосові команди користувача.
Інтеграція з Samsung-продуктами	Ні	Так	Bixby спеціально розроблений для взаємодії зі смартфонами, телевізорами та іншими пристроями від Samsung, в той час як Vocalia – десктопний додаток на персональний комп’ютер.

Власно-розроблений голосовий асистент враховує найкращі практики та характеристики успішних аналогів, прагнучі створити продукт, що задовольнить стандарти якості та ефективності в галузі голосових технологій, і одночасно буде з відкритим вихідним кодом. Це стратегічне рішення відкриває можливості для широкого співробітництва та сприяє взаємодії з розробниками з усього світу.

1.4 Аналіз вимог до програмного забезпечення

Головною функцією програмного забезпечення є вдосконалення ефективності взаємодії з користувачем та автоматизація повсякденних завдань, більше функцій можна побачити на рисунку 1.2.

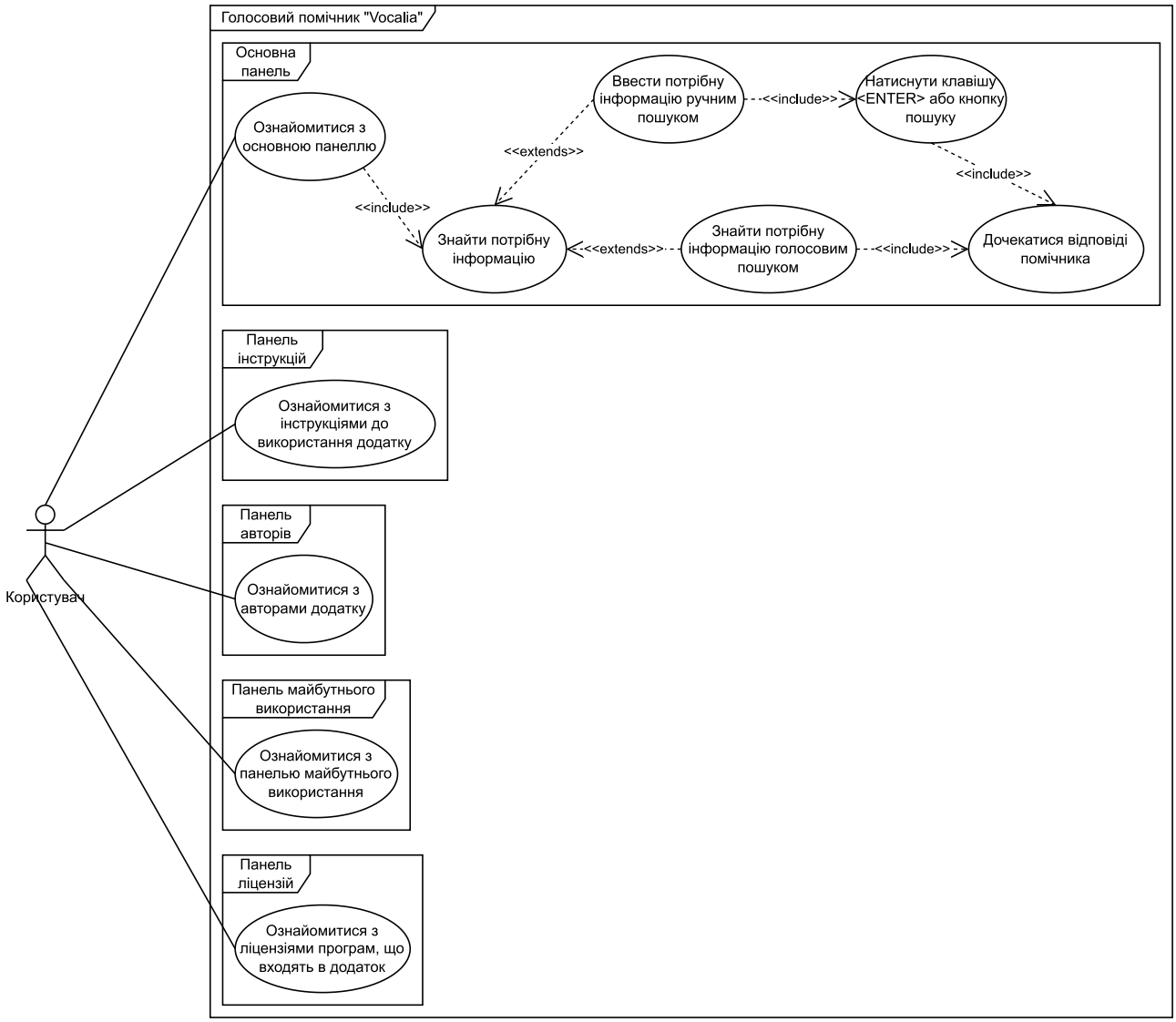

Рисунок 1.2. — Діаграма варіантів використання

В таблицях 1.5 – 1.11 наведені варіанти використання програмного забезпечення.

Таблиця 1.5. — Варіант використання UC-01

Use case name	Вхід користувача на головну сторінку
Use case ID	UC-01
Goals	Успішний вхід користувача на головну сторінку
Actors	Користувач
Trigger	Користувач бажає увійти на головну сторінку

Продовження таблиці 1.5.

Pre-conditions	У користувача встановлений додаток “Vocalia”
Flow of Events	Користувач відкриває додаток “Vocalia”, після чого відкривається головна сторінка та користувач опиняється на неї.
Extension	Відсутні.
Post-Condition	Завантаження голосового асистенту та інших модулів

Таблиця 1.6. — Варіант використання UC-02

Use case name	Вхід користувача на сторінку інструкцій
Use case ID	UC-02
Goals	Успішний вхід користувача на сторінку інструкцій
Actors	Користувач
Trigger	Користувач бажає увійти на сторінку інструкцій
Pre-conditions	Користувач успішно увійшов в додаток та знаходиться не на цій сторінці.
Flow of Events	Користувач переходить на сторінку інструкцій.
Extension	Відсутні.
Post-Condition	Перехід на сторінку інструкцій

Таблиця 1.7. — Варіант використання UC-03

Use case name	Вхід користувача на сторінку авторів
Use case ID	UC-03
Goals	Успішний вхід користувача на сторінку авторів
Actors	Користувач
Trigger	Користувач бажає увійти на сторінку авторів
Pre-conditions	Користувач успішно увійшов в додаток та знаходиться не на цій сторінці.
Flow of Events	Користувач переходить на сторінку авторів.
Extension	Відсутні.
Post-Condition	Перехід на сторінку авторів

Таблиця 1.8. — Варіант використання UC-04

Use case name	Вхід користувача на сторінку майбутнього використання
Use case ID	UC-04
Goals	Успішний вхід користувача на сторінку майбутнього використання
Actors	Користувач
Trigger	Користувач бажає увійти на сторінку майбутнього використання

Продовження таблиці 1.8.

Pre-conditions	Користувач успішно увійшов в додаток та знаходиться не на цій сторінці.
Flow of Events	Користувач переходить на сторінку майбутнього використання.
Extension	Відсутні.
Post-Condition	Перехід на сторінку майбутнього використання

Таблиця 1.9. — Варіант використання UC-05

Use case name	Вхід користувача на сторінку ліцензій
Use case ID	UC-05
Goals	Успішний вхід користувача на сторінку ліцензій
Actors	Користувач
Trigger	Користувач бажає увійти на сторінку ліцензій
Pre-conditions	Користувач успішно увійшов в додаток та знаходиться не на цій сторінці.
Flow of Events	Користувач переходить на сторінку ліцензій.
Extension	Відсутні.
Post-Condition	Перехід на сторінку ліцензій

Таблиця 1.10. — Варіант використання UC-06

Use case name	Пошук потрібної інформації ручним пошуком
Use case ID	UC-06
Goals	Успішне знаходження потрібної інформації ручним пошуком
Actors	Користувач
Trigger	Користувач бажає знайти потрібну інформацію ручним пошуком
Pre-conditions	Користувач успішно увійшов в додаток та знаходиться на головній сторінці.

Продовження Таблиці 1.10.

Flow of Events	Користувач вводить потрібний запит в поле для пошуку. Натискає клавішу <ENTER> або іконку пошуку. Чекає відповіді ГП.
Extension	У разі відсутності доступу в інтернет, пошук буде заблоковано та написано відповідне повідомлення про це. У разі натискання клавіші <ENTER> або іконці пошуку, поки ГП озвучує відповідь, пошук буде заблоковано та написано відповідне повідомлення про це.
Post-Condition	Створення відповіді на запит користувача. Її озвучення голосом.

Таблиця 1.11. — Варіант використання UC-07

Use case name	Пошук потрібної інформації голосовим пошуком
Use case ID	UC-07
Goals	Успішне знаходження потрібної інформації голосовим пошуком
Actors	Користувач
Trigger	Користувач бажає знайти потрібну інформацію голосовим пошуком
Pre-conditions	Користувач успішно увійшов в додаток та знаходиться на головній сторінці.
Flow of Events	Користувач натискає на іконку голосового пошуку. Вводить потрібний запит голосом. Чекає відповіді ГП.
Extension	У разі відсутності доступу в інтернет, пошук буде заблоковано та написано відповідне повідомлення про це. У разі натискання іконці голосового пошуку, поки ГП озвучує відповідь, пошук буде заблоковано та написано відповідне повідомлення про це.
Post-Condition	Створення відповіді на запит користувача. Її озвучення голосом.

1.4.1 Розроблення функціональних вимог

Програмне забезпечення розділене на модулі. Кожен модуль має свій певний набір функцій. На рисунку 1.3 наведено загальну модель вимог, а в таблицях 1.12 – 1.27 наведений опис функціональних вимог до програмного забезпечення. Матрицю трасування вимог можна побачити на рисунку 1.4.

Опис	Назва	Пріоритет	Ризик
1. Інтерфейс	FR-1	1	Низький
1.1. Панель контролю	FR-1		
1.1.1. Закриття програми	FR-1		
1.1.2. Зміна розміру програми	FR-2		
1.1.3. Згортання програми	FR-3		
1.1.4. Перетягування вікна програми	FR-4		
1.2. Вікно відповідей	FR-8		
1.2.1 Прокручування відповіді ГП	FR-8		
1.3. Головне меню	FR-9		
1.3.1. Перехід на основну панель	FR-9		
1.3.2. Перехід на панель інструкцій	FR-10		
1.3.3. Перехід на панель авторів	FR-11		
1.3.4. Перехід на панель майбутнього використання	FR-12		
1.3.5. Перехід на панель ліцензій	FR-13		
2. Система	FR-7	2	Високий
2.1. Надання текстового та голосового доступу до сервісів	FR-14		
2.2. Відтворення відповідей та інформації	FR-7		
3. Пошук	FR-5	1	Середній
3.1. Ручний пошук	FR-5		
3.1.1. Гарантування конфіденційності всіх запитів	FR-16		
3.2. Голосовий пошук	FR-6		
3.2.1 Розпізнавання англійської мови	FR-15		
3.2.2. Гарантування конфіденційності всіх запитів	FR-16		

Рисунок 1.3. — Модель вимог у загальному вигляді

Таблиця 1.12. — Функціональна вимога FR-1

Назва	Можливість закриття програми
Опис	Система повинна надавати можливість закриття програми.

Таблиця 1.13. — Функціональна вимога FR-2

Назва	Можливість зміни розміру програми
Опис	Система повинна надавати можливість зміни розміру програми.

Таблиця 1.14. — Функціональна вимога FR-3

Назва	Можливість згортання програми
Опис	Система повинна надавати можливість згортання програми.

Таблиця 1.15. — Функціональна вимога FR-4

Назва	Можливість перетягування вікна програми
Опис	Система повинна надавати можливість перетягування вікна програми.

Таблиця 1.16. — Функціональна вимога FR-5

Назва	Можливість ручного пошуку
Опис	Система повинна надавати можливість виконання ручного пошуку.

Таблиця 1.17. — Функціональна вимога FR-6

Назва	Можливість голосового пошуку
Опис	Система повинна надавати можливість голосового пошуку.

Таблиця 1.18. — Функціональна вимога FR-7

Назва	Відтворення відповідей та інформації
Опис	Система повинна надавати можливість відтворення голосом асистента та друкованого тексту у вікні відповіді.

Таблиця 1.19. — Функціональна вимога FR-8

Назва	Можливість прокручування відповіді ГП
Опис	Система повинна надавати можливість прокручування вмісту вікна відповіді ГП.

Таблиця 1.20. — Функціональна вимога FR-9

Назва	Можливість переходу на основну панель
Опис	Система повинна надавати можливість переходу на основну панель.

Таблиця 1.21. — Функціональна вимога FR-10

Назва	Можливість переходу на панель інструкцій
Опис	Система повинна надавати можливість переходу на панель інструкцій до використання.

Таблиця 1.22. — Функціональна вимога FR-11

Назва	Можливість переходу на панель авторів
Опис	Система повинна надавати можливість переходу на панель авторів.

Таблиця 1.23. — Функціональна вимога FR-12

Назва	Можливість переходу на панель майбутнього використання
Опис	Система повинна надавати можливість переходу на панель майбутнього використання.

Таблиця 1.24. — Функціональна вимога FR-13

Назва	Можливість переходу на панель ліцензій
Опис	Система повинна надавати можливість переходу на панель ліцензій.

Таблиця 1.25. — Функціональна вимога FR-14

Назва	Надання текстового та голосового доступу до сервісів
Опис	Система повинна надавати можливість текстового та голосового доступу до різних сервісів, таких як пошук, погода, переклад, музика тощо.

Таблиця 1.26. — Функціональна вимога FR-15

Назва	Забезпечення функції розпізнавання англійської мови
Опис	Система повинна надавати можливість розпізнавання англійської мови для забезпечення введення команд клавіатурою та голосом.

Таблиця 1.27. — Функціональна вимога FR-16

Назва	Гарантування конфіденційності всіх запитів
Опис	Система повинна забезпечувати конфіденційності всіх запитів для захисту приватності користувачів.

	FR-1	FR-2	FR-3	FR-4	FR-5	FR-6	FR-7	FR-8	FR-9	FR-10	FR-11	FR-12	FR-13	FR-14	FR-15	FR-16
UC-01									+							
UC-02										+						
UC-03											+					
UC-04												+				
UC-05													+			
UC-06					+		+	+	+					+		+
UC-07						+	+	+	+					+	+	+

Рисунок 1.4. — Матриця трасування вимог

1.4.2 Розроблення нефункціональних вимог

В таблицях 1.28 – 1.32 наведений опис нефункціональних вимог до програмного забезпечення.

Таблиця 1.28. — Нефункціональна вимога NFR-1

Назва	Надійність
Опис	Зміна розміру програми повинна відбуватися без будь-яких суттєвих затримок або відмов.

Таблиця 1.29. — Нефункціональна вимога NFR-2

Назва	Видимість
Опис	Згортання та розгортання програми повинні відбуватися з максимальною відмітністю для користувача.

Таблиця 1.30. — Нефункціональна вимога NFR-3

Назва	Продуктивність
Опис	Час від видачі команди на пошук до відображення результатів повинен бути менше 5 секунд.

Таблиця 1.31. — Нефункціональна вимога NFR-4

Назва	Масштабованість
Опис	Система має оптимізовано працювати на різних розмірах екрану, забезпечуючи оптимальний користувацький досвід.

Таблиця 1.32. — Нефункціональна вимога NFR-5

Назва	Ергономіка
Опис	Користувацький інтерфейс системи повинен бути інтуїтивно зрозумілим та зручним для використання. Результати пошуку мають бути представлені чітко та лаконічно, а користувач повинен мати можливість взаємодіяти з ними без зайвих ускладнень.

1.5 Постановка задачі

Створення голосового помічника розкриває великий потенціал в застосуванні, здатний успішно інтегруватися в різноманітні сфери. Основною метою є вдосконалення користувацького досвіду та автоматизація щоденних завдань, пов'язаних з використанням технологічних пристроїв та браузерів.

Голосовий помічник спрямований на широкий спектр користувачів, які бажають полегшити взаємодію з технічними пристроями та оптимізувати свої робочі процеси. Беззаперечно, програмне забезпечення має бути доступним та зручним для кінцевих користувачів, які використовують його для ефективного управління та взаємодії з різними технологічними пристроями.

Не менш важливим є виконання всіх визначених функціональних і нефункціональних вимог програмного забезпечення.

Висновки до розділу

У розділі визначено встановлення завдання та розглянуто наявні аналоги голосових помічників. Представлені альтернативи оцінені як високоякісні, але мають свої недоліки, такі як використання закритого коду або зберігання даних на серверах.

Крім того, в першому розділі здійснено аналіз вимог, складено таблицю для відповідного сценарію використання та кожної функціональної вимоги. Це надає більш детальне уявлення про вимоги до мови програмування та технологій, які слід використовувати.

Після ретельного аналізу було визначено, що оптимальним вибором для даного проєкту є використання мови програмування Java, інтегрованого середовища розробки IntelliJ IDEA, бібліотеки Sphinx4 для розпізнавання мовлення, бібліотеки Swing для створення графічного інтерфейсу користувача, та бібліотеки FreeTTS для генерації голосу. Цей вибір обґрунтовується їх високою продуктивністю, надійністю та специфічними можливостями, необхідними для успішної реалізації голосового інтерфейсу та ефективної взаємодії з користувачем.

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

Для опису бізнес процесу пошуку інформації в ручному режимі користувачем використовується BPMN модель (рисунок 2.1)[26].

Опис послідовності пошуку інформації в ручному режимі користувачем:

– користувач, знаходячись на головній сторінці, вводить команду для пошуку інформації;

– надісланий запит користувача обробляється на стороні обробника команд;

– у випадку наявності помилки в запиті, обробляється помилка, після чого процес зупиняється;

– підготовлений запит для пошуку відправляється до сервісу пошуку з боку обробника команд;

– сервіс пошуку знаходить інформацію за вказаним запитом і відправляє її до обробника команд;

– знайдена інформація передається користувачу через обробник команд та відображається в браузері.

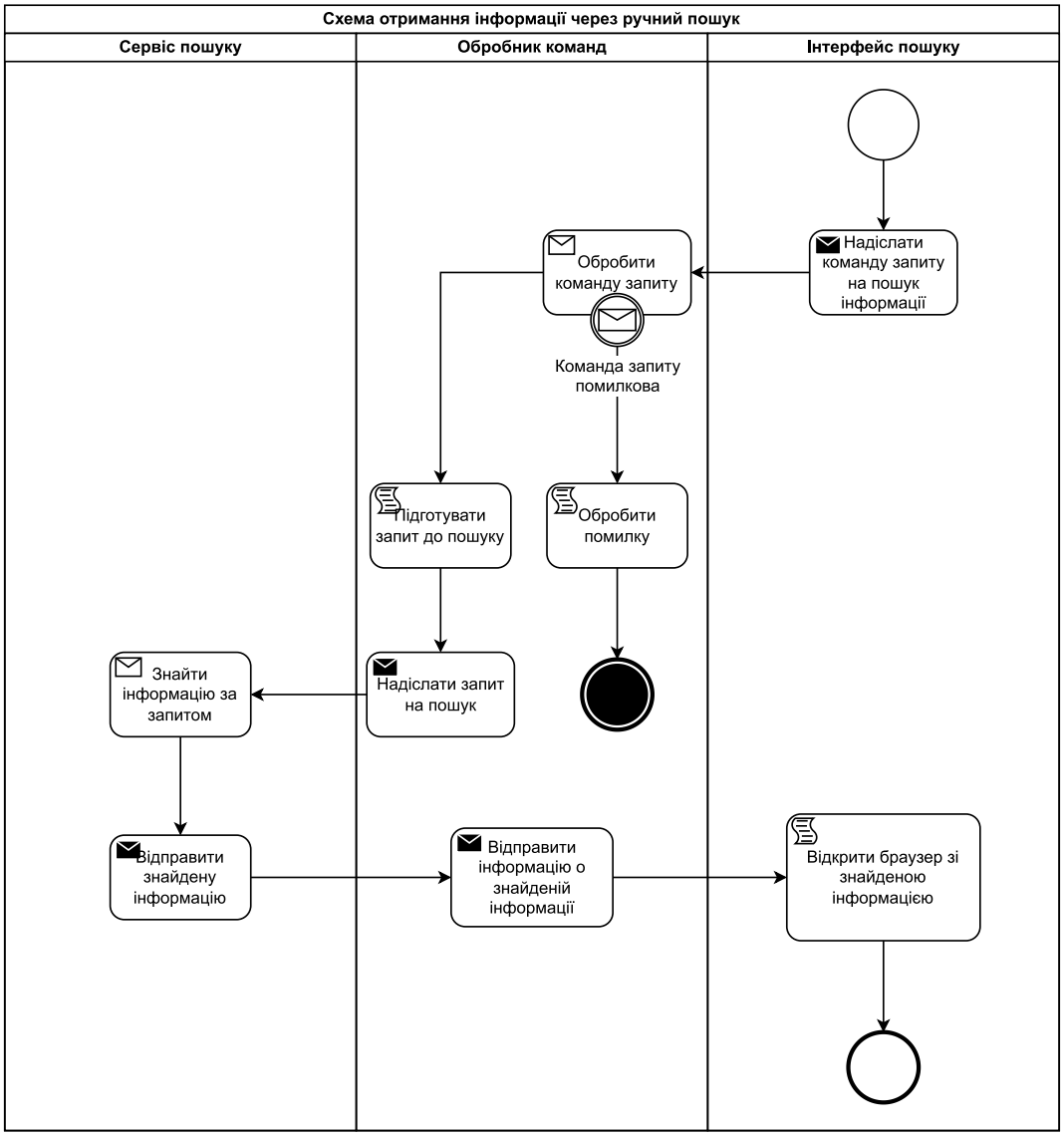

Рисунок 2.1. — Схема бізнес-процесу отримання інформації через ручний пошук

Для опису бізнес процесу пошуку інформації в голосовому режимі користувачем використовується BPMN модель (рисунок 2.2).

Опис послідовності пошуку інформації в голосовому режимі користувачем:

– користувач, знаходячись на головній сторінці, натискає на голосовий пошук та говорить команду для пошуку інформації;

– надісланий запит користувача обробляється на стороні обробника команд;

– у випадку наявності помилки в запиті, обробляється помилка, після чого процес зупиняється;

– підготовлений запит для пошуку відправляється до сервісу пошуку з боку обробника команд;

– сервіс пошуку знаходить інформацію за вказаним запитом і відправляє її до обробника команд;

– знайдена інформація передається користувачу через обробник команд та відображається в браузері.

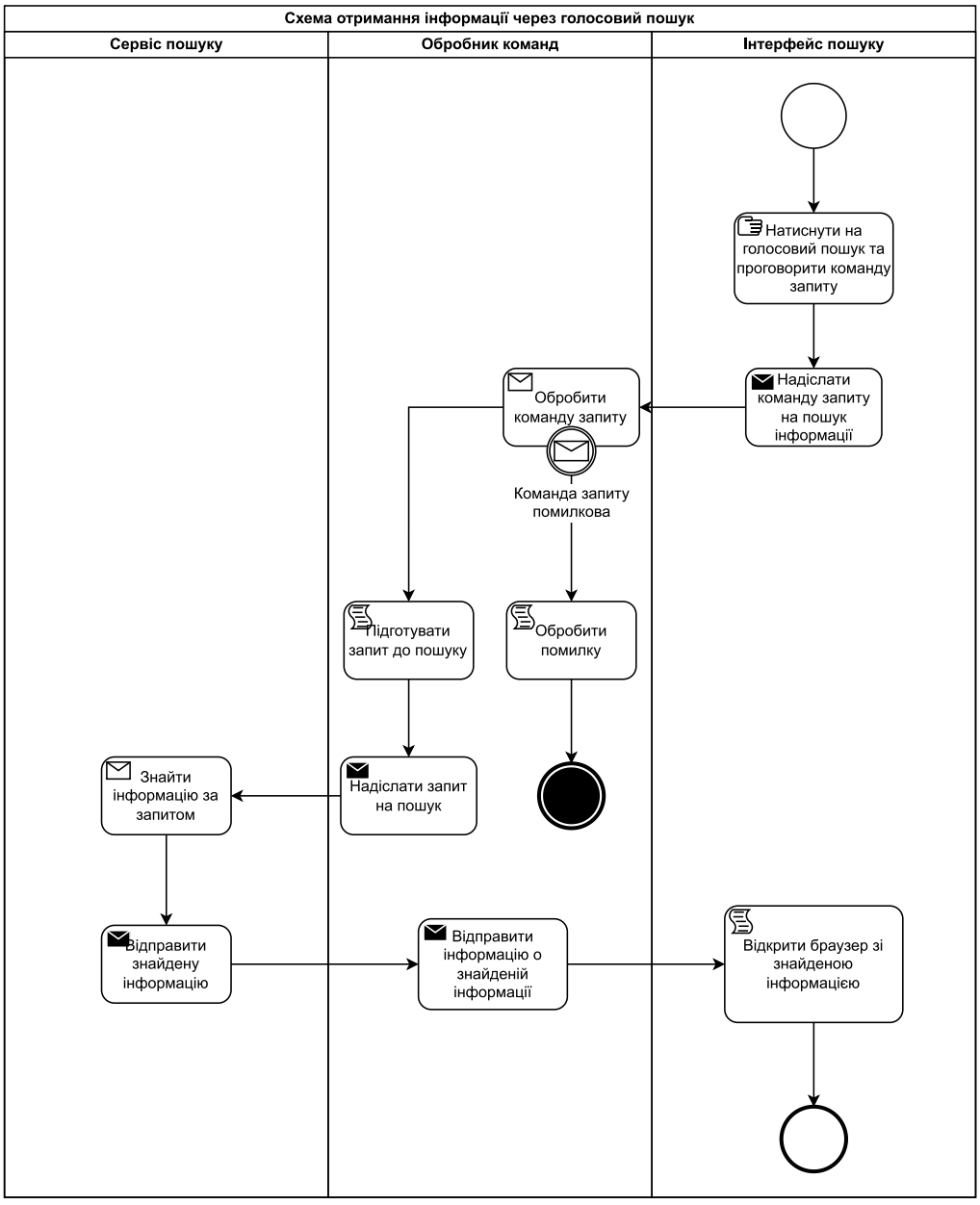

Рисунок 2.2. — Схема бізнес-процесу отримання інформації через голосовий пошук

2.2 Архітектура програмного забезпечення

При проєктуванні голосового помічника, використання патернів стає ключовим етапом для забезпечення ефективності, читабельності та легкості управління кодом. Одні із таких патернів об'єктноорієнтований підхід та синглтон патерн.

У розробці голосового помічника використовується об'єктноорієнтований підхід до програмування (ООП), що сприяє організації коду у вигляді класів та об'єктів для спрощення розробки та утримання коду. Наприклад, існує клас Synthesizer, який втілює об'єкт синтезатора тексту в мову. Крім того, для забезпечення єдиності екземпляра класу Synthesizer використовується синглтон патерн[27][28].

Структура класу Synthesizer включає приватний конструктор при створенні екземпляра класу, де встановлюється системна властивість та ініціалізується об'єкт голосу. Синглтон реалізовано через статичний вкладений клас SynthesizerHolder, який забезпечує ініціалізацію та утримання єдиного екземпляра синтезатора.

Додатково, програма використовує механізм wait() та notifyAll() для реалізації Observer Pattern, який дозволяє спостерігати за змінами та сповіщати зареєстровані об'єкти про ці зміни. Компонент LockConnection слугує об'єктом спостереження, використовується для синхронізації між потоками та використовує механізм wait() та notifyAll() для ефективного керування виконанням операцій та уникнення зайвого очікування.

Інші частини програми, що взаємодіють із LockConnection, можуть реєструватися як спостерігачі та отримувати сповіщення про зміни стану, що додає до загальної ефективності та гнучкості системи.

Заданий контекст натякає на те, що побудова UML-діаграми може значно полегшити розуміння архітектурних аспектів системи. Це важливий інструмент для візуалізації структури, поведінки та взаємодії класів у системі. Діаграми UML дозволяють команді розробників та іншим учасникам проекту легше сприймати та спілкуватися з архітектурними концепціями[29].

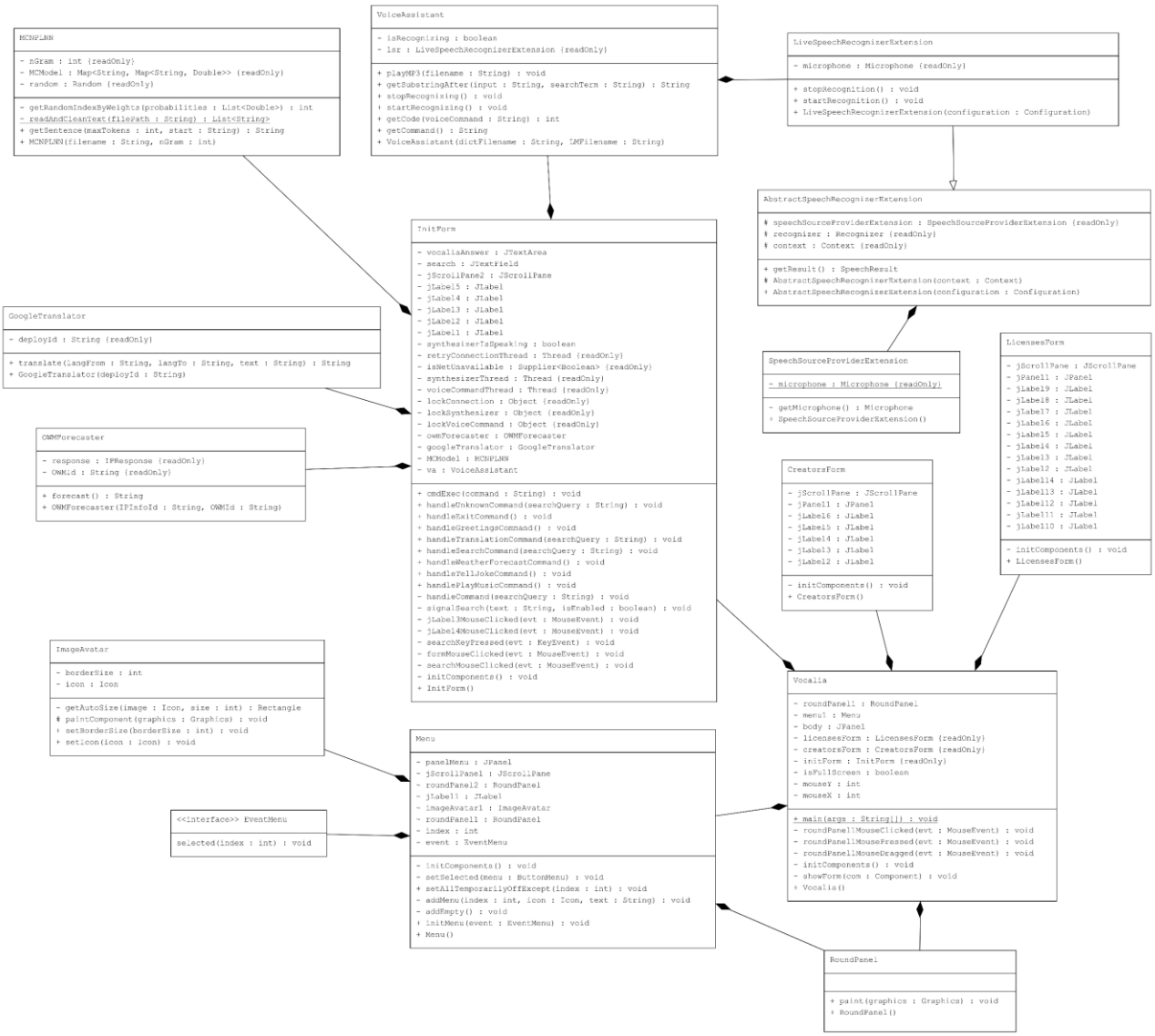

Рисунок 2.3. — UML-діаграма класів

Архітектура, побудована за патерном Singleton, дозволяє зберігати код чистим, оскільки вона гарантує, що у програмі існує лише один екземпляр класу і немає глобальних змінних чи складних систем управління життєвим циклом об'єктів.

Об'єктно-орієнтоване програмування дозволяє створювати код, який організований навколо об'єктів, які представляють конкретні екземпляри або класи об'єктів.

2.3 Конструювання програмного забезпечення

2.3.1 Оригінальні алгоритми та модифікації

Для досягнення паралельної анімації у Java Swing використовується підхід, де окремий потік відповідає за кожну окрему анімаційну частину. Наприклад, можна виділити окремий потік для обробки кожної кнопки меню на екрані, забезпечуючи гладкість та ефективне використання ресурсів[19].

Також розроблено можливість динамічного виділення потоків програмно, що дозволяє ефективно використовувати ресурси системи, адаптуючи кількість потоків під завдання[17].

У бібліотеці Sphinx4 використовується сінглтон-підхід для кожної ініціалізації об'єкта класу нової лінії мікрофону.

Це забезпечує ефективне управління ресурсами та уникнення зайвого створення екземплярів, зберігаючи стан мікрофону та структуру лише одного екземпляра.

Поміж іншими покращеннями до бібліотеки Sphinx4 введено оптимізації, які дозволяють ефективніше використовувати ресурси мікрофону та оптимізувати шлях обробки звукової інформації.

2.3.2 Структури даних та програмні структури

- Об'єктно-орієнтований підхід та сінглтон

  Клас Synthesizer реалізовано за об'єктно-орієнтованим підходом, що дозволяє організувати код у вигляді класів та об'єктів. Застосовано сінглтон патерн для забезпечення єдиності екземпляра синтезатора.
- Механізм wait() та notifyAll() для Observer Pattern

  Клас LockConnection використовує механізми wait() та notifyAll() для реалізації Observer Pattern, що дозволяє ефективно спостерігати та сповіщати зареєстровані об'єкти про зміни.
- Діаграми UML для візуалізації структури

Застосовано побудову UML-діаграм для візуалізації архітектурних аспектів системи, сприяючи легшому розумінню та спілкуванню з архітектурними концепціями.

2.3.3 Утиліти, бібліотеки та інше ПЗ

Опис утиліт, бібліотек та іншого стороннього програмного забезпечення, що використовується у розробці наведено в таблиці 2.22.

Таблиця 2.1. — Опис утиліт

№ п/п	Назва утиліти	Опис застосування
1	IntelliJ IDEA	Головне середовище розробки програмного забезпечення курсової роботи.
2	Apache NetBeans	Програмне забезпечення, необхідне для швидкого та зручного створення інтерфейсної частини програмного забезпечення курсової роботи.
3	Sphinx4	Бібліотека для розпізнавання мовлення, розробленою для використання в різних проектах. Вона базується на технологіях розпізнавання мовлення та надає інтерфейс для інтеграції з різними мовленнєвими застосунками. Sphinx4 може бути використаний для розпізнавання голосу в реальному часі, обробки команд голосового управління та інших завдань, пов'язаних із мовленням.

Продовження Таблиці 2.1.

№ п/п	Назва утиліти	Опис застосування
4	Swing	Бібліотека для розробки графічного інтерфейсу користувача (GUI) в мові програмування Java. Ця бібліотека надає набір компонентів і контейнерів, які дозволяють створювати візуально привабливі та інтерактивні програми. Swing використовується для розробки віконних додатків, де користувач може взаємодіяти з програмою за допомогою миші, клавіатури та інших пристроїв.
5	FreeTTS	Бібліотека для синтезу мовлення в мові програмування Java. Вона дозволяє розробникам створювати додатки, які можуть генерувати звукові файли або безпосередньо виводити мовлення на аудіо-пристрої.
6	IPinfo	Бібліотека (та сервіс) для отримання інформації про IP-адреси та геолокацію. Використовуючи цей сервіс у розробці, ви можете отримувати дані про місцезнаходження користувачів, що може бути корисним для точкового аналізу.

2.4 Аналіз безпеки даних

Ця курсова робота виконує функції голосового помічника та, важливо відзначити, не здійснює збір персональних даних користувача. Таким чином, аспекти безпеки даних не становлять частину цього проекту.

Висновки до розділу

В даному розділі висвітлено результати аналізу бізнес-процесів з пошуку інформації в ручному та голосовому режимах. Представлено деталі архітектури

програмного забезпечення голосового помічника, охоплюючи об'єктно-орієнтований підхід, синглтон патерн, використання механізмів wait() та notifyAll() для Observer Pattern та використання UML-діаграми класів для візуалізації структури.

Розглянуто оригінальні алгоритми та модифікації, спрямовані на паралельну анімацію та ефективне управління ресурсами. Подано опис структур даних та програмних конструкцій, що використовуються у процесі розробки, і визначено їхню раціональність та ефективність.

Висвітлено використання утиліт, бібліотек та іншого зовнішнього програмного забезпечення у розробці. Проведено аналіз безпеки даних, в результаті якого підтверджено, що дана курсова робота не здійснює збір персональних даних користувача.

Зазначено, що об'єктно-орієнтований підхід та синглтон патерн сприяють ефективній організації коду, спрощуючи розробку та утримання кодової бази. Механізм wait() та notifyAll() для Observer Pattern дозволяє ефективно спостерігати за змінами та інформувати зареєстровані об'єкти про ці зміни. Діаграми UML виявляються значущим інструментом для візуалізації структури, поведінки та взаємодії класів у системі. Паралельна анімація та ефективне управління ресурсами сприяють підвищенню продуктивності та ефективності системи.

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз якості ПЗ

Аналіз якості ПЗ було проведено за допомогою двох плагінів: MetricsReloaded та Qodana[30][31].

Основні метрики, отримані за допомогою MetricsReloaded, та їх опис:

– JLOC (Javadoc lines of code) – 205:

Кількість рядків коду, які містять коментарі Javadoc (документаційні коментарі для Java коду). Вказує на розмір документації в коді.

– LOC (Lines of code) – 1854:

Загальна кількість рядків вихідного коду. Ця метрика включає як код, так і коментарі. Вказує на обсяг реального виконуваного коду.

– v(G)avg (Average cyclomatic complexity) – 1.88:

Середнє значення цикломатичної складності коду. Цикломатична складність вимірює кількість ліній виходу (alternative paths) в коді. Низьке значення (менше 5) вказує на меншу складність коду.

Щодо результатів аналізу плагіном Qodana:

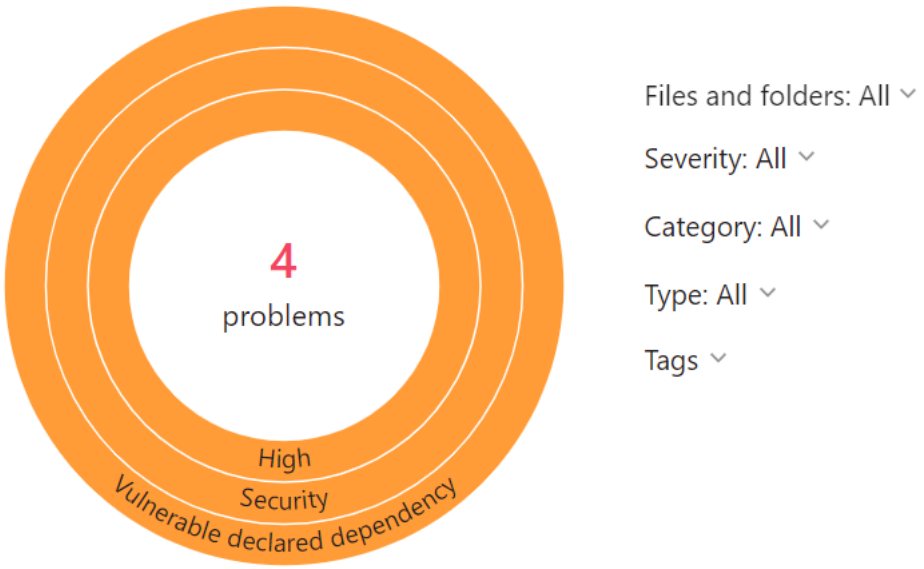

Рисунок 3.1. — Результат аналізу коду за допомогою плагіну Qodana

Знайдено тільки 4 проблеми, які є залежностями в Maven та ніяк не впливають на даний проект.

За результатами аналізу вказано, що код проекту в цілому має низьку цикломатичну складність, хорошу документацію (включаючи Javadoc), та містить дуже мало проблем, які, крім того, не впливають на функціональність проекту.

3.2 Опис процесів тестування

Було виконане мануальне тестування програмного забезпечення, опис відповідних тестів наведено у таблицях 3.1 – 3.8.

Таблиця 3.1. — Тест VO_1

Test Case ID	VO_1
Summary	Підтвердити, що переклад за запитом в додатку "Vocalia" ручним вводом працює без помилок
Priority	Високий
Precondition	Користувач має комп'ютер з JDK >= 8 із доступом в інтернет та встановлений додаток "Vocalia"

Продовження Таблиці 3.1.

Steps	Test Step	Test Data	Expected result
1	Відкрийте додаток "Vocalia" двічі клікнувши на нього ЛКМ		Додаток відкрився і ніяких помилок не сталося, при відкритті програми програвся звук про те, що додаток готовий для роботи
2	Натисніть на поле пошуку два рази ЛКМ		Поле пошуку активувалося для вводу
3	Введіть команду для перекладу тексту та сам текст після неї в поле пошуку клавіатурою	translate Hello I'm a Nickita	
4	Натисніть на клавіатурі <ENTER> або на кнопку пошуку		Асистент вивів на екран переклад фрази та програв звук що команда виконана успішно
5	Закрийте додаток "Vocalia"		Додаток закрився без помилок

Таблиця 3.2. — Тест VO_2

Test Case ID	VO_2		
Summary	Підтвердити, що прогноз погоди за запитом в додатку "Vocalia" голосовим вводом працює без помилок		
Priority	Високий		
Precondition	Користувач має комп'ютер з JDK >= 8 із доступом в інтернет та встановлений додаток "Vocalia"		
Steps	Test Step	Test Data	Expected result
1	Відкрийте додаток "Vocalia" двічі клікнувши на нього ЛКМ		Додаток відкрився і ніяких помилок не сталося, при відкритті програми програвся звук про те, що додаток готовий для роботи
2	Натисніть на кнопку голосового вводу		Іконка голосового вводу змінилася на активну

Продовження Таблиці 3.2.

Steps	Test Step	Test Data	Expected result
3	Скажіть команду для прогнозу погоди англійською	weather forecast	Іконка голосового вводу змінилася на пасивну та асистент вивів на екран прогноз погоди для локації зовнішнього IP
4	Закрийте додаток "Vocalia"		Додаток закрився без помилок

Таблиця 3.3. — Тест VO_3

Test Case ID	VO_3
Summary	Підтвердити, що програма правильно обрабляє ситуації, коли користувач без доступа в інтернет
Priority	Середній
Precondition	Користувач має комп'ютер з JDK >= 8 без доступу в інтернет та встановлений додаток "Vocalia"

Продовження Таблиці 3.3.

Steps	Test Step	Test Data	Expected result
1	Відкрийте додаток "Vocalia" двічі клікнувши на нього ЛКМ		Додаток відкрився і ніяких помилок не сталося, при відкритті програми програвся звук про те, що додаток готовий для роботи
2	Натисніть на поле пошуку		Додаток виведе повідомлення про відсутність інтернет з'єднання та спробує під'єднатиса до інтернету через 1.5 секунди
3	Натисніть на кнопку пошуку		Додаток виведе повідомлення про відсутність інтернет з'єднання та спробує під'єднатиса до інтернету через 1.5 секунди

Продовження Таблиці 3.3.

4	Натисніть на кнопку голосового пошуку		Додаток виведе повідомлення про відсутність інтернет з'єднання та спробує під'єднатиса до інтернету через 1.5 секунди
5	Ввімкніть доступ до інтернету		
6	Натисніть на поле пошуку		Повідомлення про відсутність інтернет з'єднання пропадає та поле пошуку стає активним для вводу
7	Закрийте додаток "Vocalia"		Додаток закрився без помилок

Таблиця 3.4. — Тест VO_4

Test Case ID	VO_4
Summary	Підтвердити, що в додатку "Vocalia" працює захист від спаму мишкою
Priority	Середній
Precondition	Користувач має комп'ютер з JDK >= 8 із доступом в інтернет та встановлений додаток "Vocalia"

Продовження Таблиці 3.4.

Steps	Test Step	Test Data	Expected result
1	Відкрийте додаток "Vocalia" двічі клікнувши на нього ЛКМ		Додаток відкрився і ніяких помилок не сталося, при відкритті програми програвся звук про те, що додаток готовий для роботи
2	Натисніть на іншу вкладку ЛКМ та швидко натисніть на інші		Додаток буде ігнорувати кліки ЛКМ декілька секунд після чого можна буде натискати інші вкладки
3	Натисніть на іншу вкладку один раз		Додаток відкриє натиснуту вкладку

Продовження Таблиці 3.4.

4	Натисніть на відкриту вкладку		Додаток ігнорує натискання на відкриту вкладку
5	Натисніть на іншу вкладку один раз		Додаток відкриє натиснуту вкладку
6	Закрийте додаток "Vocalia"		Додаток закрився без помилок

Таблиця 3.5. — Тест VO_5

Test Case ID	VO_5
Summary	Підтвердити, що пошук за запитом в додатку "Vocalia" ручним вводом працює без помилок
Priority	Високий
Precondition	Користувач має комп'ютер з JDK >= 8 з ОС Windows із доступом в інтернет >=7 та встановлений додаток "Vocalia"

Продовження Таблиці 3.5.

Steps	Test Step	Test Data	Expected result
1	Відкрийте додаток "Vocalia" двічі клікнувши на нього ЛКМ		Додаток відкрився і ніяких помилок не сталося, при відкритті програми програвся звук про те, що додаток готовий для роботи
2	Натисніть на поле пошуку два рази ЛКМ		Поле пошуку активувалося для вводу
3	Введіть команду для пошуку запиту в браузері та сам запит після неї в поле пошуку клавіатурою	search for youtube	
4	Натисніть на клавіатурі <ENTER> або на кнопку пошуку		Додаток відкрив браузер Google Chrome (якщо є, інакше Microsoft Edge) із відповідним запитом в ньому
5	Закрийте додаток "Vocalia"		Додаток закрився без помилок

Таблиця 3.6. — Тест VO_6

Test Case ID	VO_6		
Summary	Підтвердити, що генератор жартів в додатку "Vocalia" ручним вводом працює без помилок		
Priority	Низький		
Precondition	Користувач має комп'ютер з JDK >= 8 із доступом в інтернет та встановлений додаток "Vocalia"		
Steps	Test Step	Test Data	Expected result
1	Відкрийте додаток "Vocalia" двічі клікнувши на нього ЛКМ		Додаток відкрився і ніяких помилок не сталося, при відкритті програми програвся звук про те, що додаток готовий для роботи
2	Натисніть на поле пошуку два рази ЛКМ		Поле пошуку активувалося для вводу
3	Введіть команду для генерації жарту в поле пошуку клавіатурою	tell me a joke	

Продовження Таблиці 3.6.

4	Натисніть на клавіатурі <ENTER> або на кнопку пошуку		Асистент вивів на екран сгенерований жарт
5	Введіть команду для генерації жарту в поле пошуку клавіатурою знову	tell me a joke	
6	Натисніть на клавіатурі <ENTER> або на кнопку пошуку		Асистент вивів на екран новий сгенерований жарт
7	Закрийте додаток "Vocalia"		Додаток закрився без помилок

Таблиця 3.7. — Тест VO_7

Test Case ID	VO_7
Summary	Підтвердити, що сервіс музики в додатку "Vocalia" ручним вводом працює без помилок
Priority	Середній
Precondition	Користувач має комп'ютер з JDK >= 8 з ОС Windows із доступом в інтернет >=7 та встановлений додаток "Vocalia"

Продовження Таблиці 3.7.

Steps	Test Step	Test Data	Expected result
1	Відкрийте додаток "Vocalia" двічі клікнувши на нього ЛКМ		Додаток відкрився і ніяких помилок не сталося, при відкритті програми програвся звук про те, що додаток готовий для роботи
2	Натисніть на поле пошуку два рази ЛКМ		Поле пошуку активувалося для вводу
3	Введіть команду для музики в поле пошуку клавіатурою	play music	
4	Натисніть на клавіатурі <ENTER> або на кнопку пошуку		Асистент відкрив YouTube Music випадковий плейліст у браузері Google Chrome (якщо є, інакше Microsoft Edge) та ввімкнув його на прослуховування

Продовження Таблиці 3.7.

Steps	Test Step	Test Data	Expected result
5	Закрийте додаток "Vocalia"		Додаток закрився без помилок

Таблиця 3.8. — Тест VO_8

Test Case ID	VO_8
Summary	Підтвердити, що, поки асистент говорить, користувач не може користуватися пошуком
Priority	Високий
Precondition	Користувач має комп'ютер з JDK >= 8 з ОС Windows із доступом в інтернет >=7 та встановлений додаток "Vocalia"

Продовження Таблиці 3.8.

Steps	Test Step	Test Data	Expected result
1	Відкрийте додаток "Vocalia" двічі клікнувши на нього ЛКМ		Додаток відкрився і ніяких помилок не сталося, при відкритті програми програвся звук про те, що додаток готовий для роботи
2	Натисніть на поле пошуку два рази ЛКМ		Поле пошуку активувалося для вводу
3	Введіть будь яку команду в поле пошуку клавіатурою	weather forecast	
4	Натисніть на клавіатурі <ENTER> або на кнопку пошуку		Асистент виконав відповідний запит

Продовження Таблиці 3.8.

5	Поки асистент каже відповідь, натисніть на поле пошуку або на кнопки пошуку		Додаток повідомить про те, що асистент ще говорить відповідь на команду та не дасть користуватися пошуком
6	Закрийте додаток "Vocalia"		Додаток закрився без помилок

Висновки до розділу

В цьому розділі проведено аналіз якості та тестування програмного забезпечення додатку "Vocalia". Для оцінки якості ПЗ використано два плагіни: MetricsReloaded і Qodana. Результати показали, що код додатку має низьку цикломатичну складність, добру документацію (зокрема, Javadoc) і містить мало проблем, які не впливають на функціональність проекту.

Для тестування ПЗ виконано 8 мануальних тестів, що охоплюють основні функціональні можливості додатку. Усі тести пройдені успішно. Основний висновок аналізу полягає в тому, що код "Vocalia" характеризується низькою цикломатичною складністю, якісною документацією та мінімальною кількістю проблем, які не впливають на функціональність.

У результаті тестування виявлено, що додаток має високу якість коду, стабільність функціоналу та коректну роботу основних функцій. Знайдені проблеми із залежностями в Maven не мають впливу на роботу проекту, а усі мануальні тести пройшли успішно, що підтверджує надійність додатку.

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Для розгортання голосового додатку потрібно виконати наступні кроки:

– Встановити JDK 21 або більше з офіційного сайту: https://www.oracle.com/cis/java/technologies/downloads/.

– Створити новий проект в Intellij IDEA, чи в іншому зручному середовищі, для Java та склонувати в нього репозиторій Vocalia:
git clone https://github.com/TheMegistone4Ever/Vocalia.git

– Збудувати проєкт за допомогою Maven, для цього запустіть в локальному терміналі:
mvn install

– Створити JAR-артефакт:
Натиснути Cntrl+Alt+Shift+S (або пройти за шляхом: File -> Project Structure).
Зайти на вкладку артефактів.
Натиснути “+”, далі обрати JAR та From modules with dependencies.

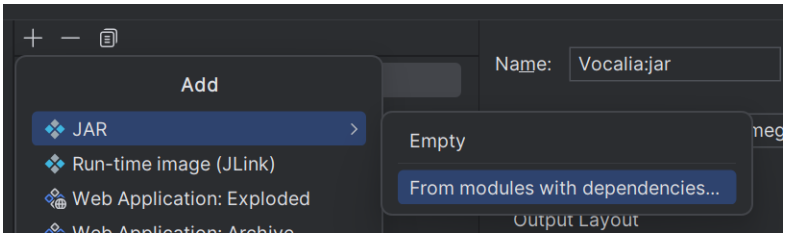

Рисунок 4.1. — Створення JAR-артефакту

Обрати модуль Vocalia та основний клас com.nickmegistone.vocaliamaven.Vocalia.
Натиснути “OK”.

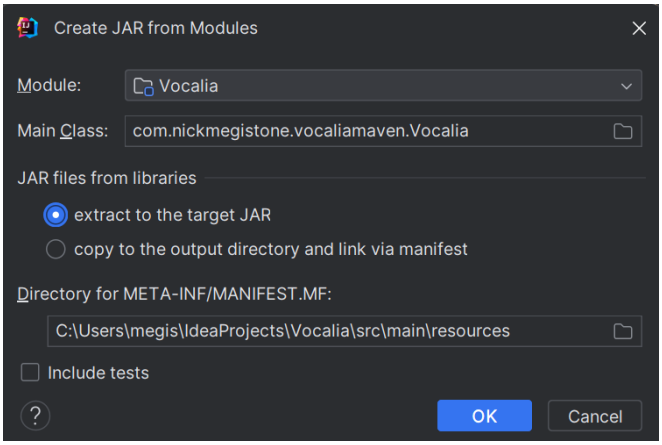

Рисунок 4.2. — Налаштування створеного JAR-артефакту

Натиснути “Apply”.

– Побудувати проект: пройти за шляхом: Build -> Build Artifacts -> Vocalia:jar -> Build)

– Після побудови скопіювати повний шлях JAR-файлу в out теки.

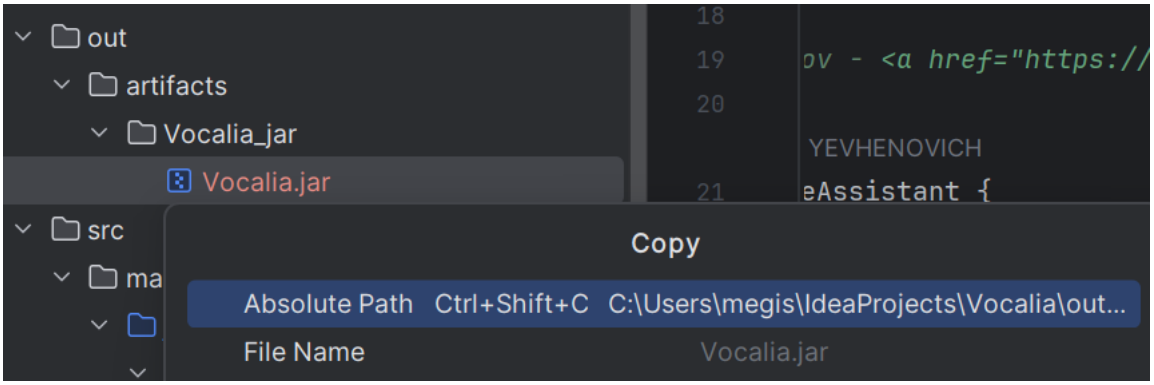

Рисунок 4.3. — Копіювання повного шляху створеного JAR-файлу

– Скомпілювати JAR-файл з кореневої папки проекту: “C:\Program Files\Java\jdk-<version>\bin\java.exe” -jar “ABSOLUTE PATH”

– Можна користуватися голосовим помічником.

4.2 Підтримка програмного забезпечення

Для ефективного управління ПЗ, проект буде зберігатися на GitHub, і кожен коміт буде представляти декілька пушів. Кожен коміт має тег у форматі "v*..", де "*" вказує на конкретну версію. Користувачі можуть завантажити архів і встановити додаток, дотримуючись інструкцій у файлі README.md. Цей підхід дозволяє легко відслідковувати версії та забезпечує зручний спосіб для користувачів встановлювати та оновлювати ПЗ.

Використання GitHub для зберігання ПЗ допомагає в створенні ефективного та спільного середовища розробки. Платформа надає можливість простоти відслідковування змін, зручного спілкування між розробниками та документування проекту. Система комітів і тегів "v*.." дозволяє легко розрізняти різні версії ПЗ, а також фіксує важливі зміни та оновлення.

Крім того, GitHub забезпечує можливість роботи з гілками, що дозволяє працювати паралельно над різними функціональностями чи виправленнями.

Вбудовані інструменти для ведення багтрекінгу, такі як Issues, дозволяють ефективно вирішувати проблеми та взаємодіяти з користувачами.

Висновки до розділу

У даному розділі було детально висвітлено процес впровадження та супроводу програмного забезпечення голосового додатку. Починаючи з встановлення JDK версії 21 або вище та створення проекту в Intellij IDEA, розгортання програми передбачало кілька кроків, таких як збірка проекту за допомогою Maven і створення JAR-артефакту.

Процедура створення JAR-артефакту включала в себе важливі кроки, такі як вибір основного класу та модуля, налаштування артефакту та його побудова. Результатом був готовий JAR-файл, який можна було використовувати для запуску голосового додатку.

Використання системи контролю версій GitHub для зберігання програмного забезпечення дозволило забезпечити прозорість та легкість завантаження ПЗ. Вона створює зручне середовище для відслідковування змін, вирішення проблем, спілкування та документування проекту. Використання гілок також сприяло паралельній роботі над різними аспектами програми. Успішне впровадження та супровід програмного забезпечення включає етапи встановлення середовища розробки, розгортання програми та ефективне управління версіями через систему контролю версій, зокрема GitHub.

ВИСНОВКИ

У ході виконання курсової роботи було розроблено голосовий помічник "Vocalia" як Desktop додаток. Додаток дозволяє користувачам здійснювати голосовий та ручний пошук інформації в Інтернеті, а також виконувати інші завдання, такі як відтворення музики, генерація жартів, прогноз погоди тощо.

У першому розділі курсової роботи було проведено аналіз вимог до додатка, визначено його функціональні та нефункціональні вимоги. На основі цього аналізу було обрано оптимальні мову програмування Java, середовище розробки Intellij IDEA та інші технології для реалізації додатка.

У другому розділі було розроблено архітектуру додатка, яка включає об'єктноорієнтований підхід, синглтон патерн, механізми wait() та notifyAll() для Observer Pattern, UML-діаграми класів, паралельну анімацію та ефективне управління ресурсами. Оптимізовано бібліотеку Sphinx-4 за допомогою багатопоточності та виправлено деякі проблеми з цією бібліотекою, наприклад, неефективне використання мікрофона пристрою.

У третьому розділі було проведено аналіз якості та тестування додатка. Для оцінки якості ПЗ були використані два плагіни: MetricsReloaded і Qodana. Результати показали, що код додатка має низьку цикломатичну складність, добру документацію та містить мало проблем, які не впливають на функціональність проєкту. У результаті тестування виявлено, що додаток має високу якість коду, стабільність функціонала та коректну роботу основних функцій.

У четвертому розділі було детально висвітлено процес впровадження та супроводу програмного забезпечення голосового додатка. Розгортання програми передбачало кілька кроків, таких як встановлення JDK версії 21 або вище та створення проєкту в Intellij IDEA, збірка проєкту за допомогою Maven і створення JAR-артефакту. Використання системи контролю версій GitHub для зберігання програмного забезпечення дозволило забезпечити прозорість та легкість завантаження ПЗ.

За результатами виконання курсової роботи можна зробити наступні висновки:

- Додаток "Vocalia" відповідає усім поставленим вимогам і може бути використаний для практичних цілей.
- Додаток має високу якість коду, стабільність функціонала та коректну роботу основних функцій.
- Розгортання та супровід додатка є ефективним та прозорим.

Для подальшого розвитку додатка можна розглянути такі напрямки:

- Розширення функціональності додатка, наприклад, додаванням підтримки інших мов, додаткових функцій пошуку, а також можливістю взаємодії з іншими пристроями.
- Покращення продуктивності та ефективності додатка, наприклад, шляхом використання нових алгоритмів та структур даних.
- Підвищення безпеки додатка, наприклад, шляхом використання сучасних методів шифрування та захисту від вразливостей.

Розроблений голосовий помічник має широкий спектр потенційних застосувань. Його можна використовувати для покращення доступності інформації та послуг для людей з обмеженими можливостями, а також для підвищення продуктивності праці та навчання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Everything you need to know about voice assistants [Електронний ресурс] // santander.com. - Режим доступу до ресурсу: https://www.santander.com/en/stories/everything-you-need-to-know-about-voice-assistants
2) Voice assistants: What they are and what they mean for marketing and commerce [Електронний ресурс] // insiderintelligence.com. - Режим доступу до ресурсу: https://www.insiderintelligence.com/insights/voice-assistants
3) What is artificial intelligence (AI)? [Електронний ресурс] // ibm.com. - Режим доступу до ресурсу: https://www.ibm.com/topics/artificial-intelligence
4) How do voice assistants work? [Електронний ресурс] // thisisdmg.com. - Режим доступу до ресурсу: https://www.thisisdmg.com/en/how-do-voice-assistants-work
5) GOOGLE ASSISTANT, SIRI, BIXBY, ALEXA, ЯКИЙ НАЙКРАЩИЙ? [Електронний ресурс] // gizchina.com.ua. - Режим доступу до ресурсу: https://gizchina.com.ua/uk/2022/12/26/google-assistant-siri-bixby-alexa-yakyy-naykraschyy
6) The Paradox of Intelligent Assistants: Poor Usability, High Adoption [Електронний ресурс] // nngroup.com. - Режим доступу до ресурсу: https://www.nngroup.com/articles/intelligent-assistants-poor-usability-high-adoption
7) Google's Privacy Policy [Електронний ресурс] // policies.google.com. - Режим доступу до ресурсу: https://policies.google.com/privacy?hl=en-US
8) Google's Speech-to-Text documentation [Електронний ресурс] // cloud.google.com. - Режим доступу до ресурсу: https://cloud.google.com/speech-to-text/docs
9) Google's Speech-to-Text [Електронний ресурс] // cloud.google.com. - Режим доступу до ресурсу: https://cloud.google.com/speech-to-text?hl=en
10) CMUSphinx Documentation [Електронний ресурс] // cmusphinx.github.io. - Режим доступу до ресурсу: https://cmusphinx.github.io/wiki

11) Welcome to DeepSpeech’s documentation! [Електронний ресурс] // deepspeech.readthedocs.io. - Режим доступу до ресурсу: https://deepspeech.readthedocs.io/en/r0.9
12) wit.ai - Build Natural Language Experiences [Електронний ресурс] // wit.ai. - Режим доступу до ресурсу: https://wit.ai
13) Java Documentation [Електронний ресурс] // docs.oracle.com. - Режим доступу до ресурсу: https://docs.oracle.com/en/java
14) Python 3.12.1 documentation [Електронний ресурс] // docs.python.org. - Режим доступу до ресурсу: https://docs.python.org/3
15) IntelliJ IDEA – the Leading Java and Kotlin IDE [Електронний ресурс] // jetbrains.com. - Режим доступу до ресурсу: https://www.jetbrains.com/idea
16) ECLIPSE IDE - The Leading Open Platform for Professional Developers [Електронний ресурс] // eclipseide.org. - Режим доступу до ресурсу: https://eclipseide.org
17) Sphinx-4 - Pure Java speech recognition library [Електронний ресурс] // github.com. - Режим доступу до ресурсу: https://github.com/cmusphinx/sphinx4
18) Transcribe speech to text by using client libraries [Електронний ресурс] // cloud.google.com. - Режим доступу до ресурсу: https://cloud.google.com/speech-to-text/docs/transcribe-client-libraries#client-libraries-install-java
19) Package javax.swing [Електронний ресурс] // docs.oracle.com. - Режим доступу до ресурсу: https://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html
20) JavaFX [Електронний ресурс] // openjfx.io. - Режим доступу до ресурсу: https://openjfx.io
21) FreeTTS 1.2.3 - A speech synthesizer written entirely in the JavaTM programming language [Електронний ресурс] // freetts.sourceforge.io. - Режим доступу до ресурсу: https://freetts.sourceforge.io
22) Amazon Alexa [Електронний ресурс] // alexa.com. - Режим доступу до ресурсу: https://www.alexa.com
23) Apple Siri [Електронний ресурс] // support.apple.com. - Режим доступу до ресурсу: https://support.apple.com/uk-ua/guide/iphone/iph83aad8922/ios

24) Google Assistant [Електронний ресурс] // assistant.google.com. - Режим доступу до ресурсу: https://assistant.google.com/intl/en
25) Samsung Bixby [Електронний ресурс] // samsung.com. - Режим доступу до ресурсу: https://www.samsung.com/ca/apps/bixby
26) BPMN [Електронний ресурс] // uk.wikipedia.org. - Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/BPMN
27) WHAT IS OBJECT-ORIENTED PROGRAMMING (OOP)? EXPLAINING FOUR MAJOR PRINCIPLES [Електронний ресурс] // softserveinc.com. - Режим доступу до ресурсу: https://career.softserveinc.com/en-us/stories/what-is-object-oriented-programming-oop-explaining-four-major-principles
28) Java Singleton Design Pattern Best Practices with Examples [Електронний ресурс] // digitalocean.com. - Режим доступу до ресурсу: https://www.digitalocean.com/community/tutorials/java-singleton-design-pattern-best-practices-examples
29) What is Unified Modeling Language (UML)? [Електронний ресурс] // visual-paradigm.com. - Режим доступу до ресурсу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/
30) MetricsReloaded - IntelliJ IDEs Plugin | Marketplace [Електронний ресурс] // plugins.jetbrains.com. - Режим доступу до ресурсу: https://plugins.jetbrains.com/plugin/93-metricsreloaded
31) Qodana Documentation [Електронний ресурс] // jetbrains.com. - Режим доступу до ресурсу: https://www.jetbrains.com/help/qodana/qodana-ide-plugin.html#UI+overview

ДОДАТОК А

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Керівник роботи

_____ Ілля АХАЛАДЗЕ

“_3_” ____грудня____ 2023 р.

ГОЛОСОВИЙ ПОМІЧНИК

Технічне завдання

КПІ.ІП-1315.045440.01.91

“ПОГОДЖЕНО”

Керівник роботи:

_____ Ілля АХАЛАДЗЕ

Консультант:

_____ Максим ГОЛОВЧЕНКО

Виконавець:

_____ Микита КИСЕЛЬОВ

Київ – 2023

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Голосовий Помічник.

Галузь застосування: Освіта та навчання.

Наведене технічне завдання поширюється на розробку голосового помічника “Vocalia” специфічного програмного забезпечення, котре використовується для надання голосової інтерфейсної можливості користувачам для виконання різноманітних завдань та отримання інформації. Призначена для особистого використання користувачами з метою надання їм зручного інструменту для отримання інформації, розваг та покращення їх продуктивності.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки голосового помічника є покращення користувацького досвіду, оптимізація часу людини, а також автоматизація повсякденних дій в браузері, що є завданням курсової роботи.

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для надання користувачам зручного та ефективного інтерфейсу для взаємодії з технологічними пристроями через голосові команди.

Метою розробки є полегшення використання електронних пристроїв, забезпечення швидкого доступу до інформації та функцій, а також поліпшення повсякденного життя користувачів через автоматизацію завдань та інтелектуальні відповіді на їхні запитання.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Програмне забезпечення повинно забезпечувати виконання наступних основних функції:

4.1.1 Користувацького інтерфейсу

- Можливість закриття програми натиснувши елемент №1 (Рисунок 4.1);
- Можливість зміни розміру програми натиснувши елемент №2 (Рисунок 4.1);
- Можливість згортання програми натиснувши елемент №3 (Рисунок 4.1);
- Можливість перетягування вікна програми;
- Можливість ручного пошуку натиснувши елемент №4 або елемент №5 на основній панелі (Рисунок 4.1);
- Можливість голосового пошуку натиснувши елемент №6 на основній панелі (Рисунок 4.1);
- Відтворення відповідей та інформації в елементі №7 на основній панелі голосом асистенту та друкованим текстом (Рисунок 4.1);
- Можливість прокручувати елемент №7 на основній панелі (Рисунок 4.1), якщо відповідь асистента або інформація велика та повністю не поміщається рамки елементу;
- Можливість перейти на основну панель (Рисунок 4.1), якщо на даний момент користувач знаходиться на іншій, натиснувши на елемент 9 в елементі №8 (Рисунок 4.1);
- Можливість перейти на панель інструкцій до використання (Рисунок 4.2), якщо на даний момент користувач знаходиться на іншій, натиснувши на елемент 10 в елементі №8 (Рисунок 4.1);
- Можливість перейти на панель авторів (Рисунок 4.3), якщо на даний момент користувач знаходиться на іншій, натиснувши на елемент 11 в елементі №8 (Рисунок 4.1);

– Можливість перейти на панель майбутнього використання (Рисунок 4.4), якщо на даний момент користувач знаходиться на іншій, натиснувши на елемент 12 в елементі №8 (Рисунок 4.1);
– Можливість перейти на панель ліцензій (Рисунок 4.5), якщо на даний момент користувач знаходиться на іншій, натиснувши на елемент 13 в елементі №8 (Рисунок 4.1).

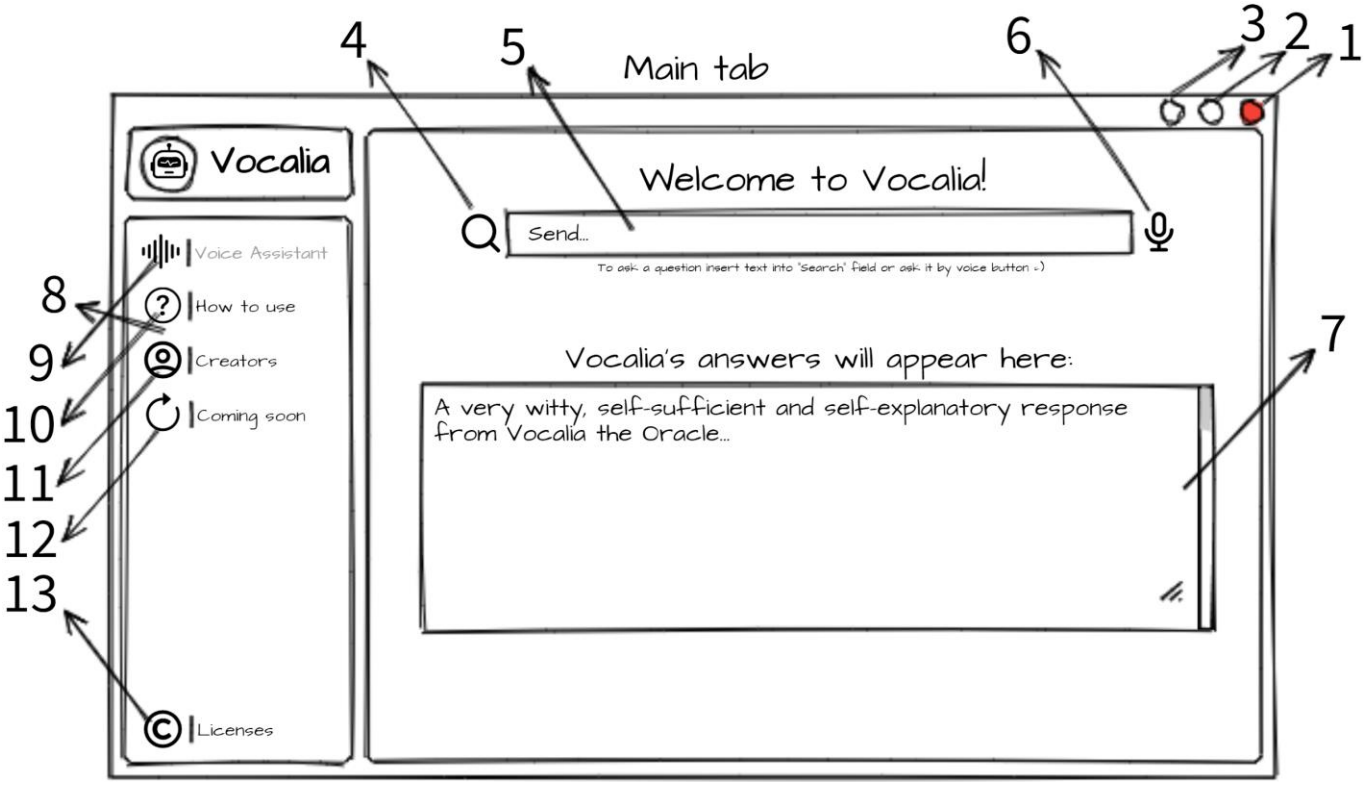

Рисунок 4.1. — Основна панель додатку.

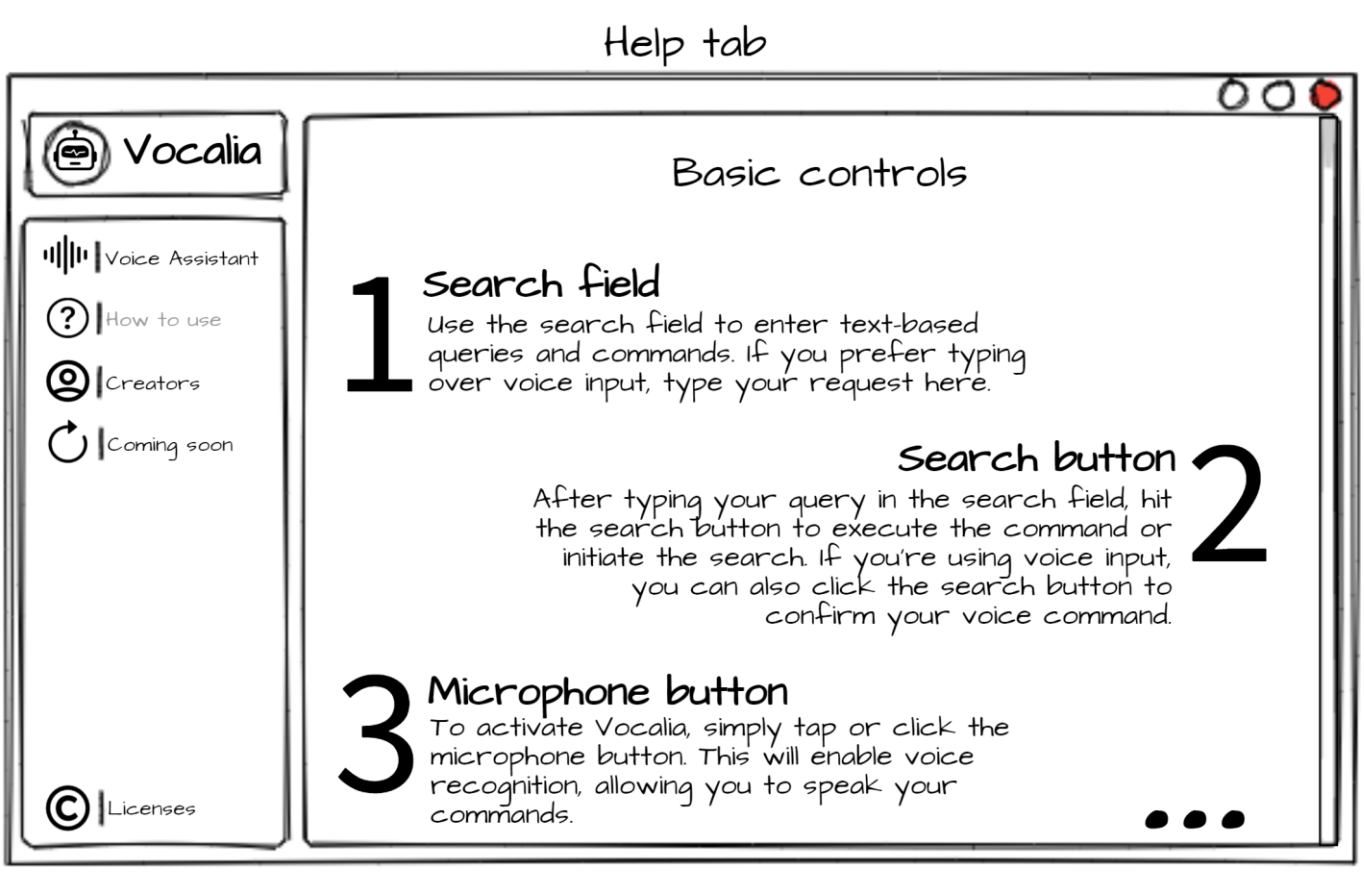

Рисунок 4.2. — Панель інструкцій до використання додатку.

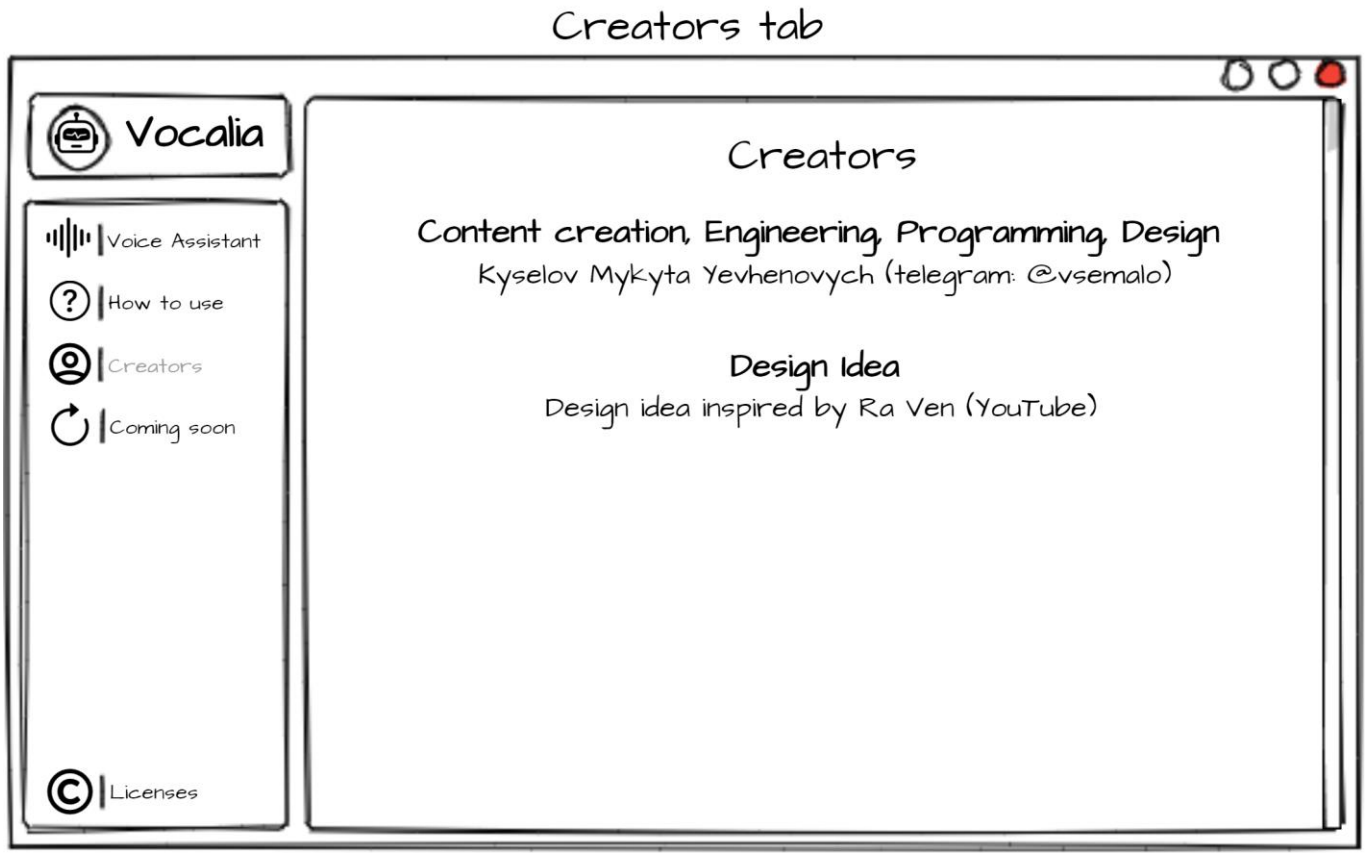

Рисунок 4.3. — Панель авторів додатку.

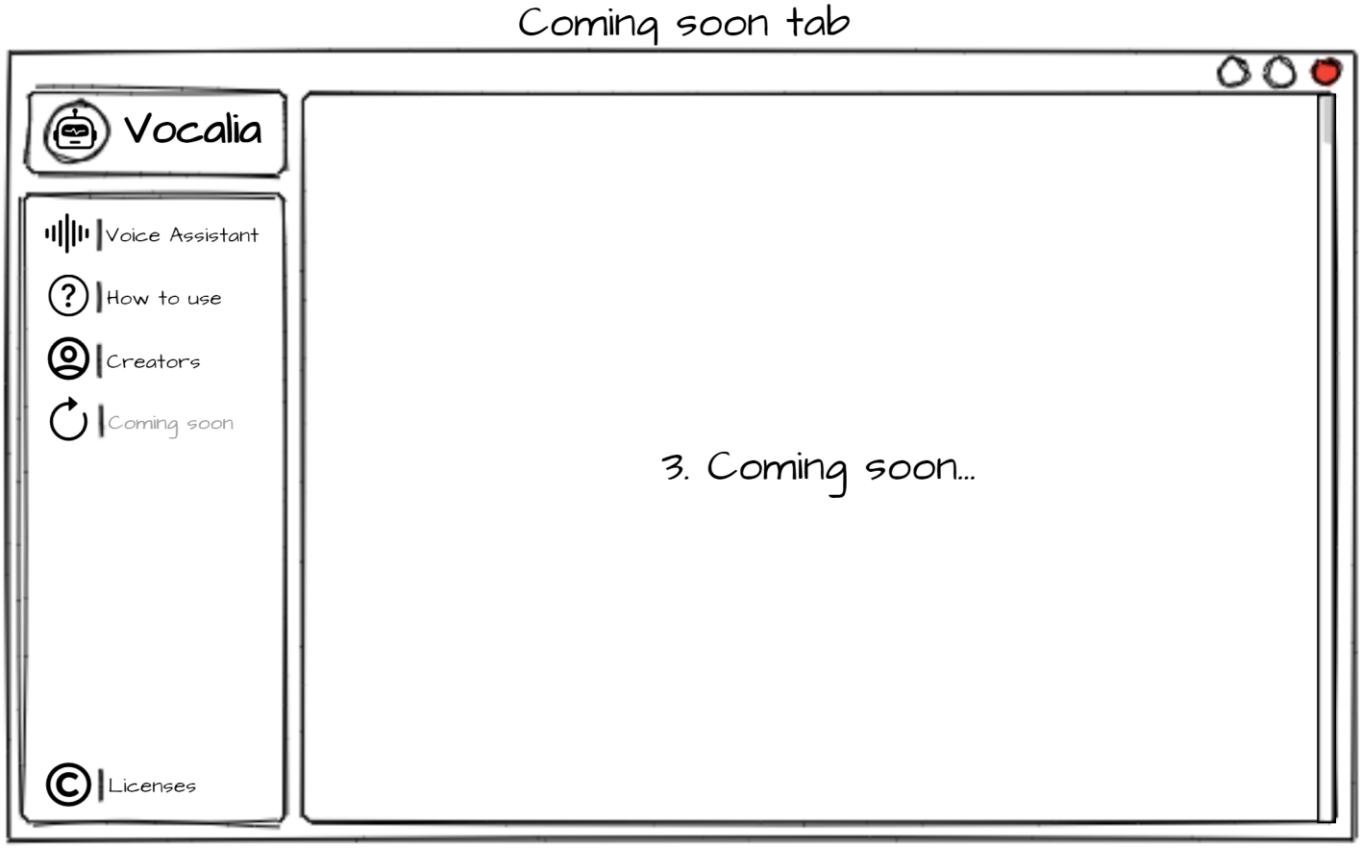

Рисунок 4.4. — Панель майбутнього використання додатку.

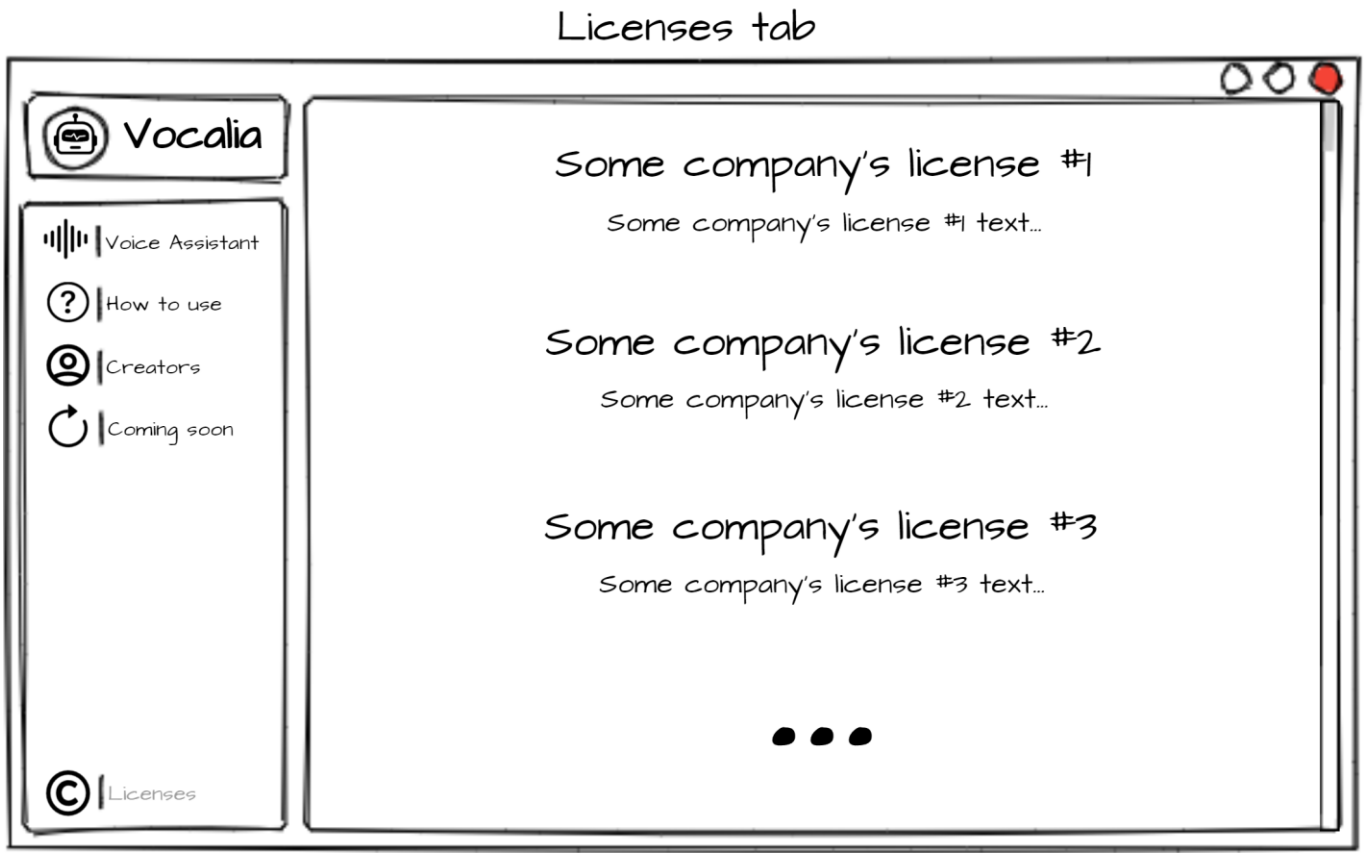

Рисунок 4.5. — Панель ліцензій додатку.

4.1.2 Для користувача:

– Надання текстового та голосового доступу до сервісів пошуку;
– Надання текстового та голосового доступу до сервісів погоди;
– Надання текстового та голосового доступу до сервісів перекладача;
– Надання текстового та голосового доступу до музичних сервісів;
– Забезпечення функції розпізнавання англійської мови;
– Можливість створення команд клавіатурою та голосом.

4.1.3 Додаткові вимоги:

– Гарантування безпеки даних та конфіденційності голосових запитів.

4.2 Вимоги до надійності

Передбачити контроль введення інформації та захист від некоректних дій користувача.

Забезпечити створення механізмів для швидкого відновлення послуг після відключення Інтернету.

Забезпечити захист від спаму командами та натисканнями миші.

4.3 Умови експлуатації

Умови експлуатації згідно СанПін 2.2.2.542 – 96.

4.3.1 Вид обслуговування

Вимоги до виду обслуговування не висуваються.

4.3.2 Обслуговуючий персонал

Вимоги до обслуговуючого персоналу не висуваються.

4.4 Вимоги до складу і параметрів технічних засобів

Програмне забезпечення повинно функціонувати на IBM-сумісних персональних комп‘ютерах.

Мінімальна конфігурація технічних засобів:

– тип процесору: Pentium 2 266 MHz;
– об‘єм ОЗП: 2 Гб;
– підключення до мережі Інтернет зі швидкістю від 2 мегабіт;
– версія JDK: від 20.0.2.

Рекомендована конфігурація технічних засобів:

– тип процесору: AMD Ryzen 9 6900HS;
– об‘єм ОЗП: 16 Гб;
– підключення до мережі Інтернет зі швидкістю від 100 мегабіт;
– версія JDK: від 21.0.1.

4.5 Вимоги до інформаційної та програмної сумісності

Програмне забезпечення повинно працювати під управлінням операційних систем сімейства WIN64 (Windows 7, Windows 8 і т.д.).

4.5.1 Вимоги до вхідних даних

Вхідні дані повинні бути представлені в наступному форматі: голосові команди, друкований текст в поле пошуку.

4.5.2 Вимоги до вихідних даних

Результати повинні бути представлені в наступному форматі: текстова відповідь на екрані, аудіо відповідь.

4.5.3 Вимоги до мови розробки

Розробку виконати на мові програмування Java.

4.5.4 Вимоги до середовища розробки

Розробку виконати на платформі Windows за допомогою середовища IntelliJ IDEA Ultimate.

4.5.5 Вимоги до представленню вихідних кодів

Вихідний код програми має бути представлений у вигляді JAR-файлу.

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не висуваються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не висуваються.

4.8 Спеціальні вимоги

Спеціальні вимоги не висуваються.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Попередній склад програмної документації

У склад супроводжувальної документації повинні входити наступні документи на аркушах формату А4:

– пояснювальна записка;
– технічне завдання;
– керівництво користувача;
– програма та методика тестування;
– текст програми.

Графічна частина повинна бути виконана на аркушах формату А3 та містити наступні документи:

– схема структурна варіантів використання;
– схеми взаємодії об’єктів, об’єктна декомпозиція;
– схема структурна класів програмного забезпечення;

5.2 Спеціальні вимоги до програмної документації

Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

№	Назва етапу	Строк	Звітність
1.	Вивчення літератури за тематикою роботи	12.09	
2.	Розробка технічного завдання	03.10	Технічне завдання
3.	Аналіз вимог та уточнення специфікацій	07.10	Специфікації програмного забезпечення
4.	Проектування структури програмного забезпечення, проектування компонентів	15.10	Схема структурна програмного забезпечення та специфікація компонентів (діаграма класів, схема алгоритму)
5.	Програмна реалізація програмного забезпечення	20.10	Тексти програмного забезпечення
6.	Тестування програмного забезпечення	25.10	Тести, результати тестування
7.	Розробка матеріалів текстової частини роботи	21.11	Пояснювальна записка
8.	Розробка матеріалів графічної частини роботи	25.12	Графічний матеріал проекту
9.	Оформлення технічної документації роботи	30.12	Технічна документація

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

ДОДАТОК Б

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Керівник роботи

_____ Ілля АХАЛАДЗЕ

“___” _____ 2023 р.

ГОЛОСОВИЙ ПОМІЧНИК

Текст програми

КПІ.ІП-1315.045490.03.12

“ПОГОДЖЕНО”

Керівник роботи:

_____ Ілля АХАЛАДЗЕ

Консультант:

_____ Максим ГОЛОВЧЕНКО

Виконавець:

_____ Микита КИСЕЛЬОВ

Київ – 2023

Файл VoiceAssistant.java:

```
/*
 * Click nbfs:\\nbhost\\SystemFileSystem\\Templates\\Licenses\\license-default.txt to change this license
 * Click nbfs:\\nbhost\\SystemFileSystem\\Templates\\Classes\\Class.java to edit this template
 */
package com.nickmegistone.ai;

import com.nickmegistone.ai.sphinxextextension.LiveSpeechRecognizerExtension;
import edu.cmu.sphinx.api.Configuration;
import javazoom.jl.decoder.JavaLayerException;
import javazoom.jl.player.advanced.AdvancedPlayer;
import org.jetbrains.annotations.NotNull;

import java.io.FileInputStream;
import java.io.IOException;

/**
 * This class represents a voice assistant that can perform various tasks based on voice commands.
 *
 * @author Mykyta Kyselov - <a href="https://github.com/TheMegistone4Ever">Github</a>
 */
public final class VoiceAssistant {

    private final LiveSpeechRecognizerExtension lsr;
    private boolean isRecognizing = false;
```

```
    /**
     * Constructs a VoiceAssistant object with the specified parameters.
     *
     * @param dictFilename A string representing the filename of the dictionary for
speech recognition.
     * @param LMFilename   A string representing the filename of the language
model for speech recognition.
     */
    public VoiceAssistant(String dictFilename, String LMFilename) {
        Configuration configuration = new Configuration();
        configuration.setAcousticModelPath(
                String.format("resource:%s", "/edu/cmu/sphinx/models/en-us/en-us"));
        configuration.setDictionaryPath(
                String.format("file:%s/src/main/java/com/nickmegistone/resources/%s",
System.getProperty("user.dir"), dictFilename));
        configuration.setLanguageModelPath(
                String.format("file:%s/src/main/java/com/nickmegistone/resources/%s",
System.getProperty("user.dir"), LMFilename));
        try {
            lsr = new LiveSpeechRecognizerExtension(configuration);
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }

    /**
     * Retrieves the command from the voice recognition result.
     *
     * @return The command extracted from the voice recognition result, converted to
lowercase.
```

```
     */
    public @NotNull String getCommand() {
        return lsr.getResult().getHypothesis().toLowerCase();
    }

    /**
     * Retrieves the code corresponding to the given voice command.
     *
     * @param voiceCommand The voice command to be checked for code mapping.
     * @return The code associated with the voice command. Returns -1 if no
matching code is found.
     */
    public int getCode(@NotNull String voiceCommand) {
        if (voiceCommand.contains("play music")) {
            return 0;
        } else if (voiceCommand.contains("tell me a joke")) {
            return 1;
        } else if (voiceCommand.contains("weather forecast")) {
            return 2;
        } else if (voiceCommand.contains("search for")) {
            return 3;
        } else if (voiceCommand.contains("translate")) {
            return 4;
        } else if (voiceCommand.contains("hey vocalia")) {
            return 5;
        } else if (voiceCommand.contains("bye vocalia")) {
            return 6;
        }
        return -1;
    }
```

```
    /**
     * Starts the speech recognition process.
     */
    public void startRecognizing() {
        lsr.startRecognition();
        isRecognizing = true;
        playMP3("greetings.mp3");
    }

    /**
     * Stops the speech recognition process.
     */
    public void stopRecognizing() {
        if (isRecognizing) {
            lsr.stopRecognition();
            isRecognizing = false;
        }
        playMP3("farewell.mp3");
    }

    /**
     * Retrieves the substring after a specified search term in a given input string.
     *
     * @param input      The input string to search within.
     * @param searchTerm The search term to find the substring after.
     * @return           The substring after the search term.
     */
    public @NotNull String getSubstringAfter(@NotNull String input, @NotNull
String searchTerm) {
```

```
        if (input.length() <= searchTerm.length()) return input;
        return input
                .substring(input.indexOf(searchTerm) + searchTerm.length())
                .trim()
                .replaceAll("\\s", "%20");
    }

    /**
     * Plays an MP3 file.
     *
     * @param filename A string representing the filename of the MP3 file to be
played.
     */
    public void playMP3(String filename) {
        try (FileInputStream in = new
FileInputStream(String.format("%s/src/main/java/com/nickmegistone/resources/%s",
System.getProperty("user.dir"), filename))) {
            new AdvancedPlayer(in).play();
        } catch (IOException | JavaLayerException e) {
            throw new RuntimeException(e);
        }
    }
}
```

Файл GoogleTranslator.java:

```
package com.nickmegistone.ai;

import static com.nickmegistone.apputils.AppUtils.getUrlContent;
import org.jetbrains.annotations.NotNull;
```

```
import java.net.URLEncoder;
import java.nio.charset.StandardCharsets;

/**
 * The GoogleTranslator class represents a translator that can translate text from one language to another using
 * Google Apps Script. It utilizes the Google Apps Script web application deployed with a specific deploy ID.
 *
 * @author Mykyta Kyselov - <a href="https://github.com/TheMegistone4Ever">Github</a>
 */
public class GoogleTranslator {

    private final String deployId;

    /**
     * Constructs a GoogleTranslator object with the specified deploy ID.
     *
     * @param deployId The deployment ID for the Google Apps Script web application.
     */
    public GoogleTranslator(String deployId) {
        this.deployId = deployId;
    }

    /**
     * Translates the given text from the source language to the target language.
     *
```

```
    * @param langFrom The source language code.
    * @param langTo   The target language code.
    * @param text     The text to be translated.
    * @return         The translated text.
    */
   public @NotNull String translate(String langFrom, String langTo, String text) {
      return getUrlContent(
            String.format(

"https://script.google.com/macros/s/%s/exec?q=%s&target=%s&source=%s",
                  deployId,
                  URLEncoder.encode(text, StandardCharsets.UTF_8),
                  langTo,
                  langFrom
            )
      );
   }
}
```

Файл MCNPLNN.java:

```
package com.nickmegistone.ai;

import org.apache.commons.lang3.StringUtils;
import org.jetbrains.annotations.NotNull;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.*;
```

```
import java.util.regex.Matcher;
import java.util.regex.Pattern;

/**
 * This class represents a Markov Chain Natural Language Processing Neural Network.
 *
 * @author Mykyta Kyselov - <a href="https://github.com/TheMegistone4Ever">Github</a>
 */
public class MCNPLNN {

    private final Random random;
    private final Map<String, Map<String, Double>> MCModel;
    private final int nGram;

    /**
     * This constructor creates a Markov Chain model object from a file of raw text.
     *
     * @param filename A filename that will be a list of cleaned words to be used as the basis for the model.
     * @param nGram    An integer representing the number of words in each state. Default value is 3.
     */
    public MCNPLNN(String filename, int nGram) {
        random = new Random();
        this.nGram = nGram;
        List<String> clearText = readAndCleanText(String.format(System.getProperty("user.dir") +
                "/src/main/java/com/nickmegistone/resources/%s", filename));
```

```
        MCModel = new HashMap<>();
        for (int i = 0; i <= clearText.size() - (nGram << 1); ++i) {
            String currState = String.join(" ", clearText.subList(i, i + nGram));
            String nextState = String.join(" ", clearText.subList(i + nGram, i + (nGram << 1)));
            MCModel.putIfAbsent(currState, new HashMap<>());
            MCModel.get(currState).put(nextState, MCModel.get(currState).getOrDefault(nextState, .0) + 1);
        }
        // Relative frequency
        for (Map.Entry<String, Map<String, Double>> entry : MCModel.entrySet()) {
            double totalNeighbours = entry.getValue().values().stream().mapToDouble(Double::doubleValue).sum();
            Map<String, Double> transition = entry.getValue();
            for (Map.Entry<String, Double> transitionEntry : transition.entrySet()) {
                transitionEntry.setValue(transitionEntry.getValue() / totalNeighbours);
            }
        }
    }

    /**
     * This function generates a string of text based on a Markov chain model.
     *
     * @param maxTokens An integer representing the maximum tokens in the generated text. Default value is 30.
     * @param start     A string representing the initial state to begin text generation.
     * @return          A processed string of generated text.
     */
    public String getSentence(int maxTokens, @NotNull String start) {
        if (start.chars().filter(ch -> ch == ' ').count() + 1 != nGram) {
```

```
            return "Count of words must be same as nGram!";
        }
        List<String> text = new ArrayList<>(Collections.singletonList(start));
        for (int i = 0; i < maxTokens; ++i) {
            Map<String, Double> possibleWords = MCModel.get(text.get(text.size() -
1));
            String token = (String)
possibleWords.keySet().toArray()[getRandomIndexByWeights(possibleWords.values
().stream().toList())];
            if (token.equals(start)) {
                break;
            }
            text.add(token);
        }
        return StringUtils.capitalize(String.join(" ", text) + ".")
                .replaceFirst(" Okay", " Okay,")
                .replaceFirst(" heres", " here's")
                .replaceFirst(" joke", " joke:")
                .replaceFirst(" it", ", it")
                .replaceFirst(" because", "? - because")
                .replaceFirst(" youre", ", you're")
                .replaceFirst(" haha", " - :) ha-ha ;),");
    }

    /**
     * This function reads text from a file and returns a list of cleaned words.
     *
     * @param filePath A string representing the path to the file to be read.
     * @return A list of cleaned words extracted from the file.
     */
```

```
    private static @NotNull List<String> readAndCleanText(String filePath) {
        List<String> cleanedWords = new ArrayList<>();
        Pattern pattern = Pattern.compile("\\b\\w+\\b");
        try (BufferedReader br = new BufferedReader(new FileReader(filePath))) {
            String line;
            while ((line = br.readLine()) != null) {
                Matcher matcher = pattern.matcher(line.replaceAll("[^\\w\\s\\a(){}-]",
"").toLowerCase());
                while (matcher.find()) {
                    cleanedWords.add(matcher.group());
                }
            }
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
        return cleanedWords;
    }

    /**
     * This function returns a random index based on a list of probabilities.
     *
     * @param probabilities A list of doubles representing the probabilities.
     * @return An integer representing the random index.
     */
    private int getRandomIndexByWeights(@NotNull List<Double> probabilities) {
        int i = 0;
        for (double cumulativeProbability = 0, rnd = random.nextDouble(); i <
probabilities.size(); ++i) {
            cumulativeProbability += probabilities.get(i);
            if (rnd < cumulativeProbability) {
```

```
                break;
            }
        }
        return i;
    }
}
```

Файл OWMForecaster.java:

```
package com.nickmegistone.ai;

import io.ipinfo.api.IPinfo;
import io.ipinfo.api.errors.RateLimitedException;
import io.ipinfo.api.model.IPResponse;
import org.json.JSONObject;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.URL;
import java.util.Locale;

import static com.nickmegistone.apputils.AppUtils.getUrlContent;

/**
 * This class represents an OWMForecaster that retrieves weather forecast information from the OpenWeatherMap API.
 *
 * @author Mykyta Kyselov - <a href="https://github.com/TheMegistone4Ever">Github</a>
```

```
 */
public class OWMForecaster {

    private final String OWMId;
    private final IPResponse response;

    /**
     * Constructs an OWMForecaster object with the specified IPInfo ID and
OpenWeatherMap ID.
     *
     * @param IPInfoId The IPInfo ID for retrieving location information.
     * @param OWMId    The OpenWeatherMap ID for accessing the weather API.
     */
    public OWMForecaster(String IPInfoId, String OWMId) {
        this.OWMId = OWMId;
        // TODO: Replace with a more reliable method of retrieving the IP address.
        try (BufferedReader br = new BufferedReader(new InputStreamReader(new
URL("https://checkip.amazonaws.com/").openStream()))) {
            response = new
IPinfo.Builder().setToken(IPInfoId).build().lookupIP(br.readLine());
        } catch (IOException | RateLimitedException e) {
            throw new RuntimeException(e);
        }
    }

    /**
     * Retrieves the weather forecast for the current location.
     *
     * @return The weather forecast as a formatted string.
     */
```

```
    public String forecast() {
        String forecast = "Hello there! It seems we've encountered a little hiccup in our weather system, and " +
                "unfortunately, we don't have your specific location information at the moment.";
        JSONObject json = new JSONObject(
                getUrlContent(
                        String.format(

"https://api.openweathermap.org/data/2.5/forecast?lat=%s&lon=%s&appid=%s&units=%s&lang=%s",
                                response.getLatitude(),
                                response.getLongitude(),
                                OWMId,
                                "metric",
                                "en"
                        )
                )
        );
        if (json.getString("cod").equals("200")) {
            JSONObject infoObj = json.getJSONArray("list").getJSONObject(1);
            JSONObject mainObj = infoObj.getJSONObject("main");
            JSONObject cityObj = json.getJSONObject("city");
            forecast = String.format(
                    "Good evening, folks! It's time for your weather update. In the beautiful town of %s, located " +
                            "in %s at coordinates %,.1f latitude and %,.1f longitude, we're experiencing next " +
                            "conditions. Currently, the temperature is %,.1f degrees Celsius, with a feels-like " +
```

```
                "temperature of %,.1f degrees Celsius. We're expecting a maximum
temperature of %,.1f " +
                "degrees Celsius and a minimum temperature of %,.1f degrees
Celsius. The atmospheric " +
                "pressure stands at %d millibars, and the humidity is %d%%. As for
the skies, %s. The " +
                "wind is blowing at a speed of %,.1f meters per second. That's all
for now from your " +
                "weather team. Stay tuned for more updates!",
            cityObj.getString("name"),
            new
Locale.Builder().setRegion(cityObj.getString("country")).build().getDisplayCountry(
Locale.ENGLISH),
            Double.parseDouble(response.getLatitude()),
            Double.parseDouble(response.getLongitude()),
            mainObj.getDouble("temp"),
            mainObj.getDouble("feels_like"),
            mainObj.getDouble("temp_max"),
            mainObj.getDouble("temp_min"),
            mainObj.getInt("pressure"),
            mainObj.getInt("humidity"),

infoObj.getJSONArray("weather").getJSONObject(0).getString("description"),
            infoObj.getJSONObject("wind").getDouble("speed")
        );
    }
    return forecast;
  }
}
```

Файл Synthesizer.java:

```
package com.nickmegistone.ai;

import com.sun.speech.freetts.Voice;
import com.sun.speech.freetts.VoiceManager;

/**
 * The Synthesizer class represents a text-to-speech synthesizer that can convert text into speech.
 * It uses the FreeTTS library for speech synthesis. <a href="https://cmusphinx.github.io/wiki/tutorial/">Tutorial</a>.
 *
 * @author Mykyta Kyselov - <a href="https://github.com/TheMegistone4Ever">Github</a>
 */
public class Synthesizer implements AutoCloseable {

    private final Voice voice;

    /**
     * Constructs a Synthesizer object with the specified voice name and pitch.
     *
     * @param voiceName The name of the voice to be used for speech synthesis.
     * @param pitch     The pitch of the synthesized speech (range: 0.0 to 500.0).
     */
    private Synthesizer(String voiceName, float pitch) {
        System.setProperty("freetts.voices", "com.sun.speech.freetts.en.us.cmu_us_kal.KevinVoiceDirectory");
        voice = VoiceManager.getInstance().getVoice(voiceName);
```

```
    voice.setPitch(pitch);
    voice.allocate();
}

private static final class SynthesizerHolder {
    private static final Synthesizer instance = new Synthesizer("kevin16", 120);
}

/**
 * Returns the singleton instance of the Synthesizer class.
 *
 * @return The Synthesizer instance.
 */
public static Synthesizer getInstance() {
    return SynthesizerHolder.instance;
}

/**
 * Converts the given message into speech.
 *
 * @param message The message to be spoken.
 */
public void speak(String message) {
    voice.speak(message);
}

@Override
public void close() {
    voice.deallocate();
```

```
    }
}
```

Файл EventMenu.java:

```
package com.nickmegistone.event;

public interface EventMenu {

    void selected(int index);
}
```

Файл Vocalia.java:

```
package com.nickmegistone.vocaliamaven;

import com.nickmegistone.form.CreatorsForm;
import com.nickmegistone.form.Form;
import com.nickmegistone.form.InitForm;
import com.nickmegistone.form.LicensesForm;
import org.jetbrains.annotations.NotNull;

import javax.swing.*;
import java.awt.*;
import java.awt.event.MouseEvent;

import static com.nickmegistone.apputils.AppUtils.*;

public class Vocalia extends javax.swing.JFrame {
```

```
private int mouseX, mouseY;
private boolean isFullScreen = false;
private final InitForm initForm;
private final CreatorsForm creatorsForm;
private final LicensesForm licensesForm;

public Vocalia() {
    initComponents();
    setBackground(MAIN_BACKGROUND_COLOR);
    initForm = new InitForm();
    creatorsForm = new CreatorsForm();
    licensesForm = new LicensesForm();
    menu1.initMenu(index -> {
        switch (index) {
            case 0 -> showForm(initForm);
            case 1 -> showForm(new Form(index)); // TODO: add help form
            case 2 -> showForm(creatorsForm);
            case 9 -> showForm(licensesForm);
            default -> showForm(new Form(index));
        }
    });
    showForm(initForm);
    menu1.setAllTemporarilyOffExcept(0);
}

private void showForm(Component com) {
    body.removeAll();
    body.add(com);
    body.revalidate();
    body.repaint();
```

```
    }

    // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-
BEGIN:initComponents
    private void initComponents() {

        roundPanel1 = new com.nickmegistone.swing.RoundPanel();
        menu1 = new com.nickmegistone.component.Menu();
        body = new com.nickmegistone.swing.RoundPanel();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setTitle("Vocalia");
        setBackground(new java.awt.Color(24, 24, 24));
        setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
        setMinimumSize(new java.awt.Dimension(940, 540));
        setPreferredSize(new java.awt.Dimension(WIDTH, HEIGHT));

        roundPanel1.setBackground(LIGHT_BACKGROUND_COLOR);
        roundPanel1.setPreferredSize(new java.awt.Dimension(FRAME_WIDTH,
FRAME_HEIGHT));
        roundPanel1.addMouseMotionListener(new
java.awt.event.MouseMotionAdapter() {
            public void mouseDragged(java.awt.event.MouseEvent evt) {
                roundPanel1MouseDragged(evt);
            }
        });
        roundPanel1.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                roundPanel1MouseClicked(evt);
            }
```

```
            public void mousePressed(java.awt.event.MouseEvent evt) {
                roundPanel1MousePressed(evt);
            }
        });

        menu1.setMinimumSize(new java.awt.Dimension(0, 0));
        menu1.setPreferredSize(new java.awt.Dimension(256, 680));

        body.setBackground(DARK_BACKGROUND_COLOR);
        body.setPreferredSize(new java.awt.Dimension(998, 680));
        body.setLayout(new java.awt.BorderLayout());

        javax.swing.GroupLayout roundPanel1Layout = new
javax.swing.GroupLayout(roundPanel1);
        roundPanel1.setLayout(roundPanel1Layout);
        roundPanel1Layout.setHorizontalGroup(

roundPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEAD
ING)
            .addGroup(roundPanel1Layout.createSequentialGroup()
                .addGap(10, 10, 10)
                .addComponent(menu1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(10, 10, 10)
                .addComponent(body, javax.swing.GroupLayout.DEFAULT_SIZE, 0,
Short.MAX_VALUE)
                .addContainerGap())
        );
        roundPanel1Layout.setVerticalGroup(
```

```
roundPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(roundPanel1Layout.createSequentialGroup()
            .addContainerGap()

.addGroup(roundPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(menu1, javax.swing.GroupLayout.DEFAULT_SIZE, 0, Short.MAX_VALUE)
                .addComponent(body, javax.swing.GroupLayout.DEFAULT_SIZE, 0, Short.MAX_VALUE))
            .addGap(10, 10, 10))
    );

    javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(roundPanel1, javax.swing.GroupLayout.DEFAULT_SIZE, 282, Short.MAX_VALUE)
    );
    layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(roundPanel1, javax.swing.GroupLayout.DEFAULT_SIZE, 16, Short.MAX_VALUE)
    );

    pack();
```

```
    }// </editor-fold>//GEN-END:initComponents

    private void roundPanel1MouseDragged(@NotNull MouseEvent evt) {//GEN-
FIRST:event_roundPanel1MouseDragged
        System.out.println(evt);
        setLocation(evt.getXOnScreen() - mouseX, evt.getYOnScreen() - mouseY);
    }//GEN-LAST:event_roundPanel1MouseDragged

    private void roundPanel1MousePressed(@NotNull MouseEvent evt) {//GEN-
FIRST:event_roundPanel1MousePressed
        System.out.println(evt);
        mouseX = evt.getX();
        mouseY = evt.getY();
    }//GEN-LAST:event_roundPanel1MousePressed

    private void roundPanel1MouseClicked(@NotNull MouseEvent evt) {//GEN-
FIRST:event_roundPanel1MouseClicked
        System.out.println(evt);
        if (evt.getClickCount() > 1) {
            GraphicsDevice device =
GraphicsEnvironment.getLocalGraphicsEnvironment().getDefaultScreenDevice();
            if (isFullScreen) {
                // Exit full screen mode
                device.setFullScreenWindow(null);
                isFullScreen = false;
            } else {
                // Enter full screen mode
                device.setFullScreenWindow(this);
                isFullScreen = true;
            }
```

```
        }
    }//GEN-LAST:event_roundPanel1MouseClicked

    public static void main(String[] args) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException | InstantiationException | IllegalAccessException |
                 UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(Vocalia.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
```

```
        java.awt.EventQueue.invokeLater(() -> {
            Vocalia vocalia = new Vocalia();
            vocalia.setIconImage(new ImageIcon(System.getProperty("user.dir") +
"/src/main/java/com/nickmegistone/resources/logo.png").getImage());
            vocalia.setVisible(true);
        });
    }

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JPanel body;
    private com.nickmegistone.component.Menu menu1;
    private com.nickmegistone.swing.RoundPanel roundPanel1;
    // End of variables declaration//GEN-END:variables
}
```

Файл AppUtils.java:

```
package com.nickmegistone.apputils;

import org.jetbrains.annotations.NotNull;

import java.awt.*;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.URI;
import java.net.URISyntaxException;
import java.net.URLDecoder;
import java.nio.charset.StandardCharsets;
```

```
public final class AppUtils {

    public static final int MENU_SLEEP_MILLIS = 1400;
    public static final int FRAME_WIDTH = 1280;
    public static final int FRAME_HEIGHT = 720;
    public static final int THUMB_SIZE = 80;
    public static final int INTERNET_TIMEOUT = 1500;
    public static final Color SEARCH_ENABLED_COLOR = new Color(0, 102, 102);
    public static final Color SEARCH_DISABLED_COLOR = new java.awt.Color(255,
102, 0);
    public static final Color SCROLLBAR_COLOR = new Color(130, 130, 130);
    public static final Color MAIN_BACKGROUND_COLOR = new Color(51, 51, 51);
    public static final Color LIGHT_BACKGROUND_COLOR = new Color(102, 102,
102);
    public static final Color DARK_BACKGROUND_COLOR = new Color(24, 24,
24);
    public static final Color AVATAR_BORDER_TEXT_COLOR = new Color(224,
224, 224);
    public static final String SYNTHESIZER_IS_SPEAKING = "Synthesizer is
speaking...";
    public static final String SEND = "Send...";
    public static final String SEARCH_WHEN_CLICKED = "";
    public static final String NO_INTERNET_CONNECTION_SEARCH =
String.format("No internet connection, retrying after %,.1f seconds...",
INTERNET_TIMEOUT / 1000.0);

    /**
     * Retrieves the content of a URL address.
     *
     * @param urlAddress The URL address from which to retrieve the content.
```

```
     * @return The content of the URL as a string.
     */
    public static @NotNull String getUrlContent(String urlAddress) {
        StringBuilder response = new StringBuilder();
        try {
            try (BufferedReader bufferedReader = new BufferedReader(
                    new InputStreamReader(
                            new URI(
                                    urlAddress
                            ).toURL().openConnection().getInputStream()
                    )
            )) {
                String line;
                while ((line = bufferedReader.readLine()) != null) {
                    response.append(line).append("\n");
                }
            }
        } catch (IOException | URISyntaxException e) {
            throw new RuntimeException(e);
        }
        return URLDecoder.decode(response.toString(), StandardCharsets.UTF_8);
    }
}
```

ДОДАТОК В

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Керівник роботи

_____ Ілля АХАЛАДЗЕ

“___” _____ 2023 р.

ГОЛОСОВИЙ ПОМІЧНИК

Програма та методика тестування

КПІ.ІП-1315.045490.04.51

“ПОГОДЖЕНО”

Керівник роботи:

_____ Ілля АХАЛАДЗЕ

Консультант:

_____ Максим ГОЛОВЧЕНКО

Виконавець:

_____ Микита КИСЕЛЬОВ

Київ – 2023

ЗМІСТ

1 ОБ’ЄКТ ВИПРОБУВАНЬ

Об’єктом випробування є голосовий помічник, що написаний на мові програмування Java з використанням бібліотек Swing, Sphinx-4 та FreeTTS.

2 МЕТА ТЕСТУВАННЯ

Метою тестування є перевірка наступного:

I. працездатність запитів до голосового помічника;

II. виконання функціональних вимог;

III. виконання нефункціональних вимог.

3 МЕТОДИ ТЕСТУВАННЯ

Для тестування програмного забезпечення використовуються мануальне тестування.

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконується мануально з метою знаходження помилок та недоліків як у функціональній частині програмного забезпечення так і в зручності його користування.

ДОДАТОК Г

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Керівник роботи

_____ Ілля АХАЛАДЗЕ

“___” _____ 2023 р.

ГОЛОСОВИЙ ПОМІЧНИК

Керівництво користувача

КПІ.ІП-1315.045490.05.34

“ПОГОДЖЕНО”

Керівник роботи:

_____ Ілля АХАЛАДЗЕ

Консультант:

_____ Максим ГОЛОВЧЕНКО

Виконавець:

_____ Микита КИСЕЛЬОВ

Київ – 2023

ЗМІСТ

5 ПРИЗНАЧЕННЯ ПРОГРАМИ

Розробка призначена для надання користувачам зручного та ефективного інтерфейсу для взаємодії з технологічними пристроями через голосові команди.

Метою розробки є полегшення використання електронних пристроїв, забезпечення швидкого доступу до інформації та функцій, а також поліпшення повсякденного життя користувачів через автоматизацію завдань та інтелектуальні відповіді на їхні запитання.

6 ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ

6.1 Системні вимоги для коректної роботи

Мінімальна конфігурація технічних засобів:

– тип процесору: Pentium 2 266 MHz;

– об‘єм ОЗП: 2 Гб;

– підключення до мережі Інтернет зі швидкістю від 2 мегабіт;

– версія JDK: від 20.0.2.

Рекомендована конфігурація технічних засобів:

– тип процесору: AMD Ryzen 9 6900HS;

– об‘єм ОЗП: 16 Гб;

– підключення до мережі Інтернет зі швидкістю від 100 мегабіт;

– версія JDK: від 21.0.1.

6.2 Завантаження застосунку

Потрібно встановити JDK 21 або більше з офіційного сайту, якщо ще немає:

https://www.oracle.com/cis/java/technologies/downloads/.

Далі, для того, щоб завантажити застосунок, необхідно завантажити архів з готовим голосовим асистентом, розпакувати його за допомогою сторонніх утиліт, таких як WinRAR (рисунок 2.1). Подальша установка голосового асистента не потрібна, все що потрібно для її запуску, це відкрити файл “run.vbs” у розпакованій папці (рисунок 2.2).

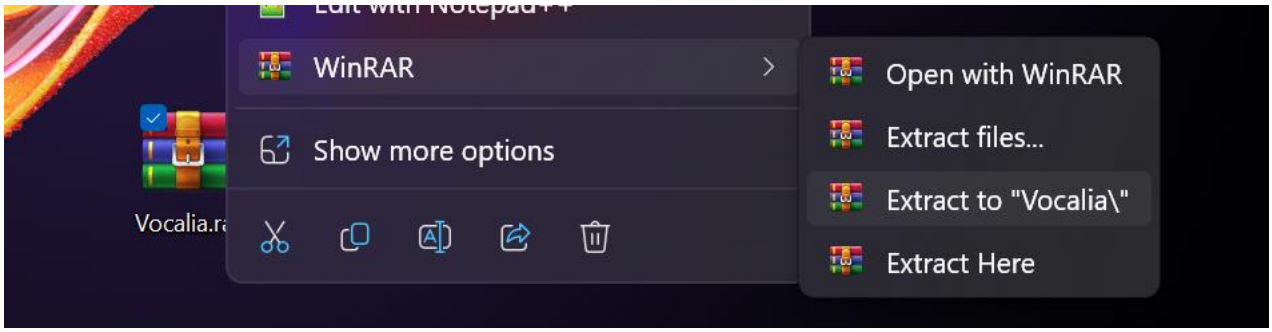

Рисунок 2.4. — Розпаковка голосового асистента на комп’ютер

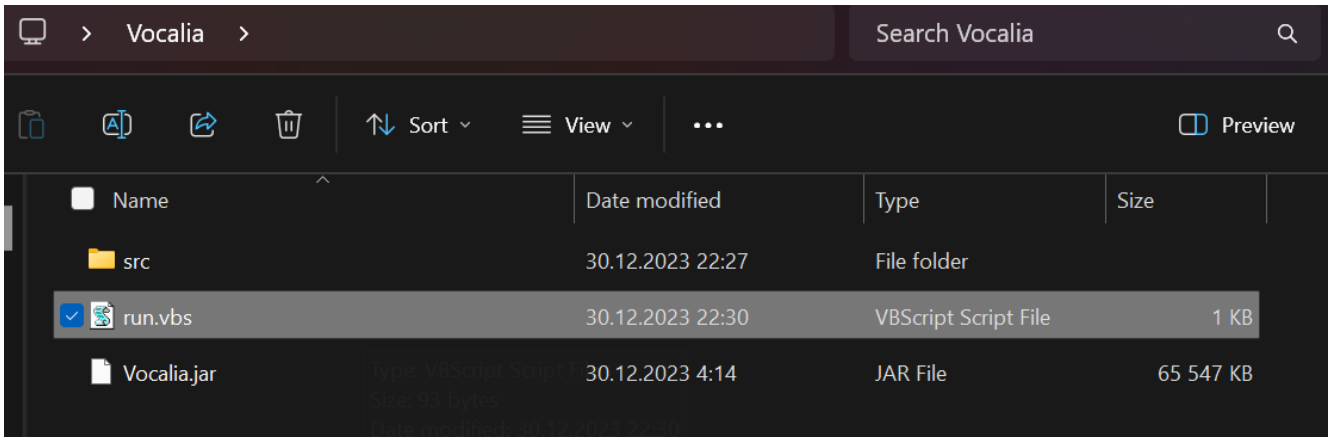

Рисунок 2.5. — Запуск голосового асистента

6.3 Перевірка коректної роботи

Після розархівування файлу очікується, що з'явиться папка з готовим до запуску голосовим асистентом. Якщо цього не сталося або виявлено відсутність необхідних файлів у папці, можна вважати, що процес розпакування відбувся некоректно через пошкоджений файл або помилку в роботі архіватора. У разі успішного розпакування вам буде доступна можливість запустити гру, відкривши файл з назвою "run.vbs" (рисунок 2.2).

7 ВИКОНАННЯ ПРОГРАМИ

Запускаємо голосового асистента та бачимо головну сторінку (рисунок 3.1).

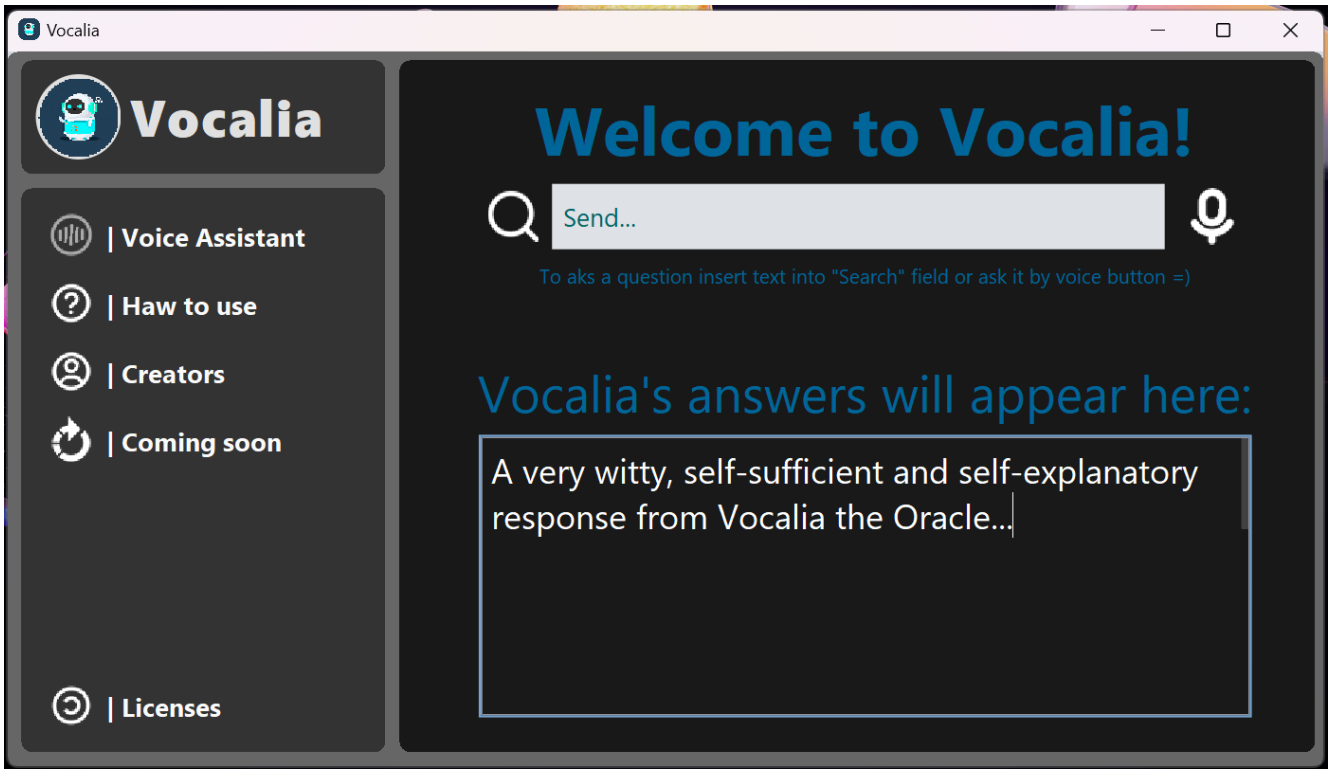

Рисунок 3.1. — Головна сторінка голосового асистента

Натискаючи на різні кнопки меню можна подивитися їх зміст, нижче наведено приклад для ліцензії (рисунок 3.2) та авторів (рисунок 3.3).

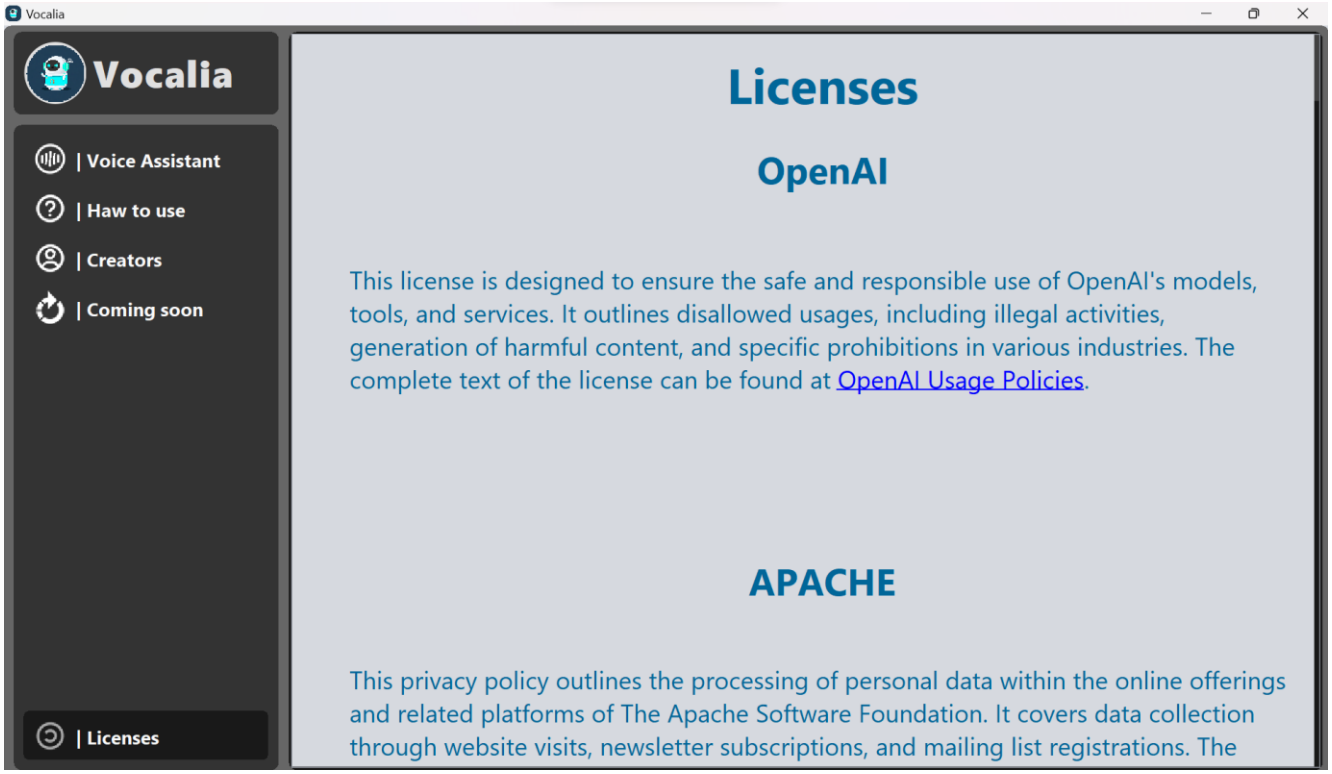

Рисунок 3.2. — Сторінка ліцензій голосового асистента

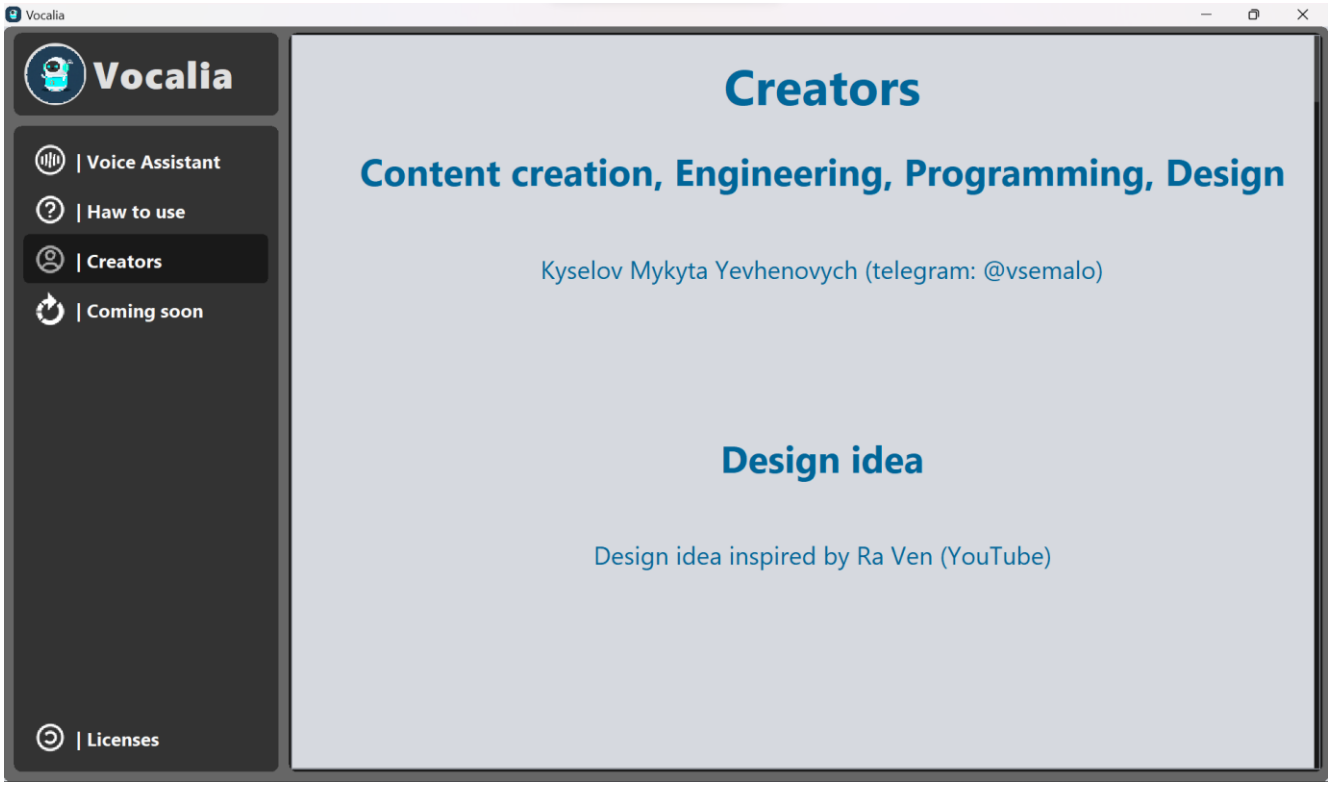

Рисунок 3.3. — Сторінка авторів голосового асистента

На головній панелі ви маєте наступні можливості:

Натиснувши на поле ручного пошуку, друкувати запит та надсилати його клавішею <ENTER> або іконку ручного пошуку (рисунок 3.4).

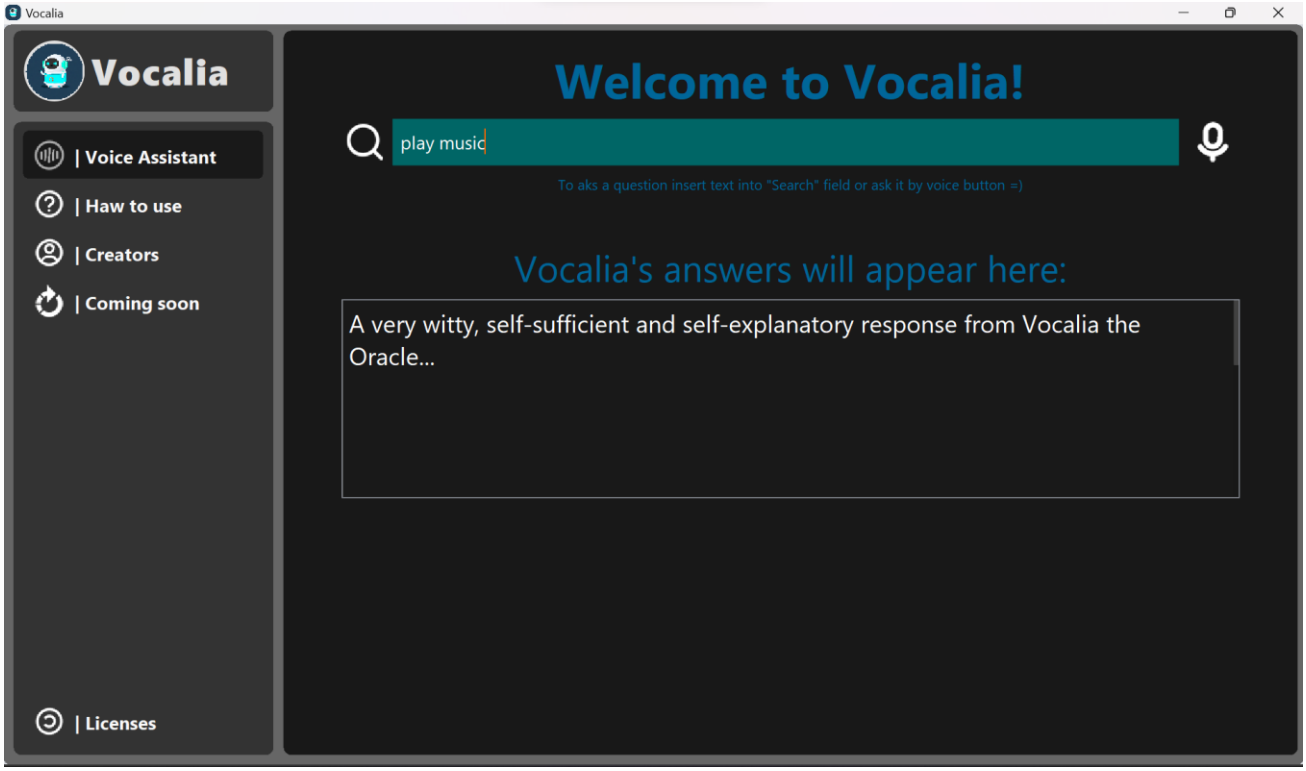

Рисунок 3.4. — Ручний пошук

Натиснувши на іконку голосового пошуку, говорити запит та надсилати його закінчивши говорити (рисунок 3.5).

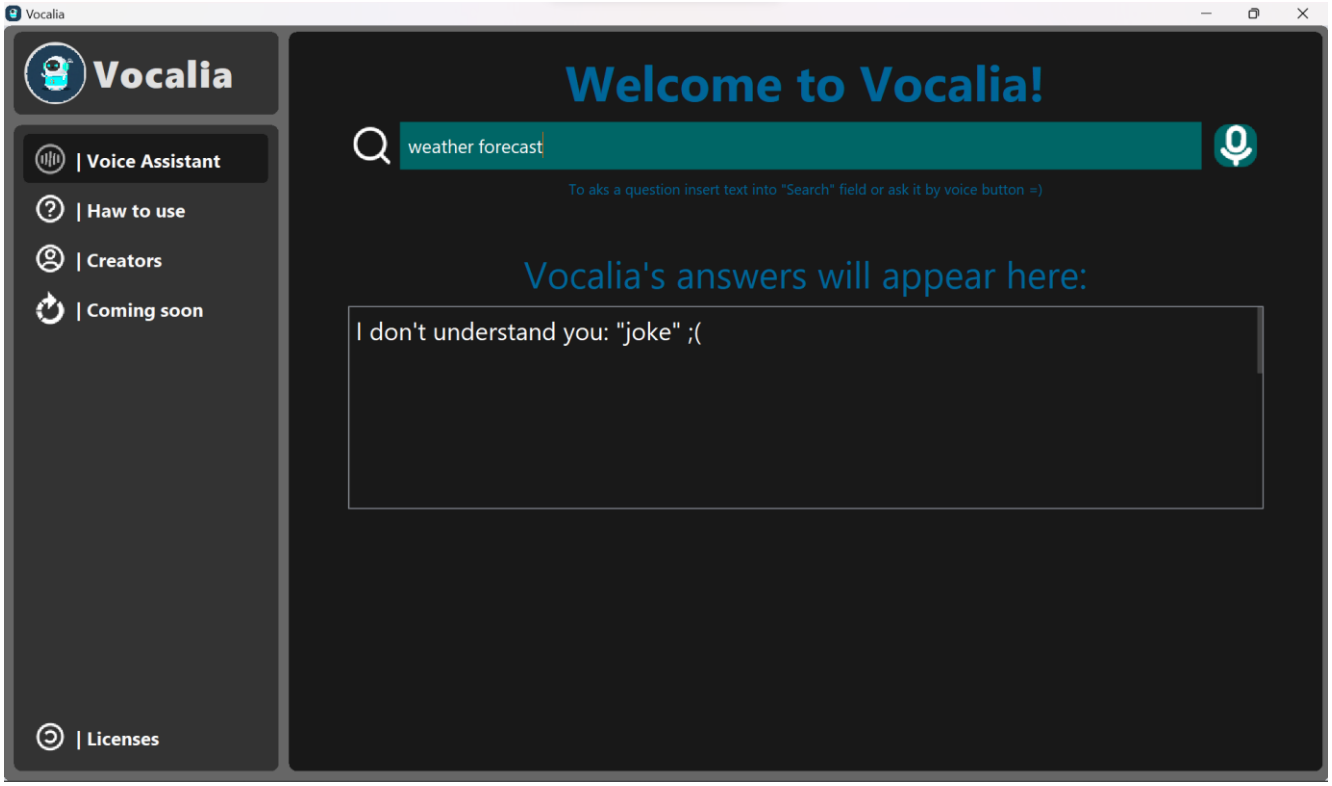

Рисунок 3.5. — Голосовий пошук

Ви маєте змогу застосувати наступні команди:

– play music – програти музику.
– tell me a joke – розповісти жарт.
– weather forecast – сказати прогноз погоди.
– search for <SEARCH TEXT> – знайти за текстом запиту.
– translate <TRANSLATE TEXT> – перевести за текстом запиту.
– hey vocalia – привітати голосового асистента.
– bye vocalia – попрощатися з голосовим асистентом.

Далі наведено приклад використання команд “search for” (рисунок 3.6) та “weather forecast” (рисунок 3.7).

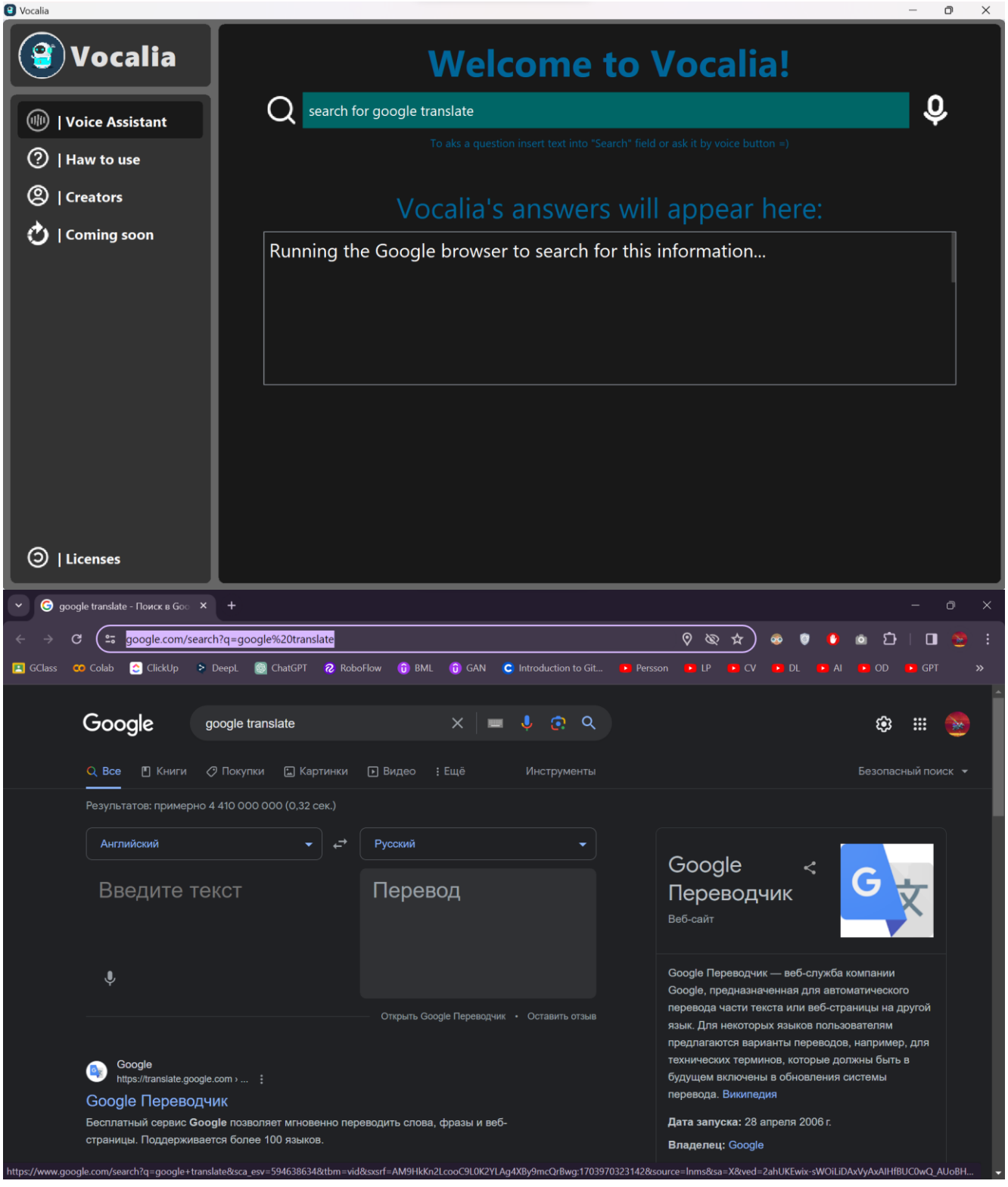

Рисунок 3.6. — Приклад використання команди “search for”

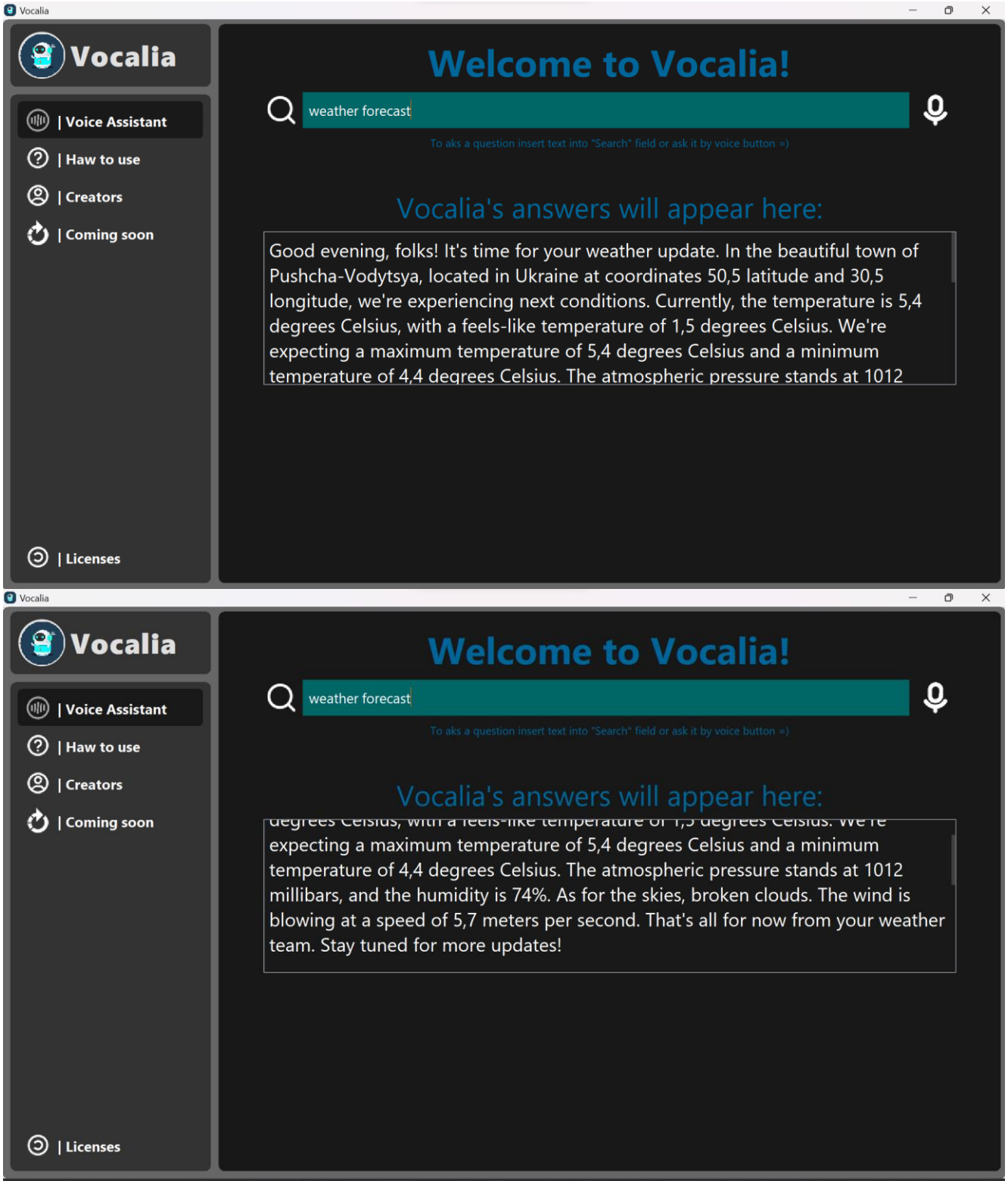

Рисунок 3.7. — Приклад використання команди “weather forecast”