Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2025 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи штучного інтелекту для роботи з аудіоданими”

Виконав здобувач вищої освіти
IV курсу, групи КІ-21-2
ОПП «Кібербезпека»
спеціальності 123 «Комп’ютерна інженерія»
_____ Мельник А.М.
«____» _____ 2025 р.

Керівник проекту
доктор технічних наук, професор
_____ Мелешко Є. В.
«____» _____ 2025 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *бакалавр*

Галузь знань ._12_____”Інформаційні технології”_____

Спеціальність *123 Комп’ютерна інженерія*

Освітньо-професійна (освітньо-наукова) програма “Компютерна інженерія"

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____О.А.Смірнов
« __ » _____ 2025 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Мельник Анні Михайлівній
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи штучного інтелекту для роботи з аудіоданими*

2.Керівник роботи *Мелешко Єлизавета Владиславівна, д-р техн. наук, професор*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу *№ 47-02 від 17.01.2025 року*

3. Строк подання студентом роботи до захисту *24.05.2025 р.*

4. Мета та завдання кваліфікаційної бакалаврської роботи*: Метою розробки є програмне забезпечення системи штучного інтелекту для роботи з аудіоданими*

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

1. Призначення та область використання.
2.Перегляд аналогічних існуючих систем.
3. Опис і обґрунтування проектних рішень.
4. Етапи програмування системи.
5. Впровадження системи в промислову експлуатацію.
6. Висновки

6. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

7. Дата видачі завдання « » 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної бакалаврської роботи	Строк виконання етапів кваліфікаційної бакалаврської роботи	Примітка
1.	Аналіз існуючих систем	10.03.2025 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2025 р.	
3.	Розробка моделі компонента	20.03.2025 р.	
4.	Розробка структур даних	25.03.2025 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2025 р.	
6.	Програмування алгоритмів	10.04.2025 р.	
7.	Оформлення ПЗ	17.04.2025 р.	
8.	Попередній захист роботи	22.05.2025 р.	

Дата видачі завдання
« » 2025 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
« » 2025 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Мельник А.М. Програмне забезпечення системи штучного інтелекту для роботи з аудіоданими. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2025.

У даній кваліфікаційній бакалаврській роботі розроблено програмне забезпечення, яке призначене для роботи з аудіоданими.

Метою роботи є створення системи штучного інтелекту для роботи з аудіоданими.

Результат роботи – програмна реалізація системи штучного інтелекту для роботи з аудіоданими.

В процесі роботи над реалізацією системи виконано дослідження існуючих методів, алгоритмів та програмних засобів. Розроблено та реалізовано власне програмне забезпечення, здійснено опис всіх його компонентів.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ архітектури IBM PC з ОС Windows 10/11.

Програму розроблено на мові програмування Python.

Ключові слова: комп'ютерна інженерія, обробка інформації, штучний інтелект, аудіодані.

ABSTRACT

Melnyk A.M. Artificial intelligence system for working with audio data. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2025.

This bachelor's qualification work presents the development of software designed for working with audio data.

The aim of the work is to create an artificial intelligence system for working with audio data.

The result of the work is the software implementation of an artificial intelligence system for working with audio data.

During the development process, research was conducted on existing methods, algorithms, and software tools. Custom software was developed and implemented, and a detailed description of all its components was provided.

A user-friendly interface was designed. Instructions for working with the software tools are included.

The program can be used on IBM PC architecture personal computers with Windows 10/11 operating systems.

The software was developed using the Python programming language.

Keywords: computer engineering, data processing, system, artificial intelligence, audio data.

ЗМІСТ

					ВКРБ-123.25.0036.00.00.ПЗ		
Вим	Арк.	№ докум.	Підп.	Дата			
Розроб.		Мельник А.М.			Програмне забезпечення системи штучного інтелекту для роботи з аудіоданими	Літ. Б	Аркуш 1 / Аркушів 67
Перев.		Мелешко Є.В.					
Н.контр.		Коваленко А.С.				КІ-21-2	
Затв.		Смірнов О.А.					

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

NLP – Natural Language Processing

AIVA – Artificial Intelligence Virtual Assistant

TTS – Text-to-Speech

FTP – File Transfer Protocol

SFTP – Secure File Transfer Protocol

PyPi – Python Package Index

SSH – Secure Shell

HTML – HyperText Markup Language

CSS – Cascading Style Sheets

AWS – Amazon Web Services

SVN – Subversion (Version Control System)

VSC – Visual Studio Code

MySQL – My Structured Query Language

AI – Artificial Intelligence

DS – Data Science

API – Application Programming Interface

ВСТУП

Сучасні технології штучного інтелекту (ШІ) суттєво змінили підхід до обробки та аналізу аудіоданих. Від автоматичного розпізнавання мовлення до генерації синтетичних голосів і музики — програмне забезпечення штучного інтелекту відкриває нові можливості для різних галузей, зокрема медицини, телекомунікацій, розваг та безпеки.

Аудіоаналіз на основі ШІ дозволяє не лише розпізнавати мовлення та ідентифікувати звуки, але й аналізувати емоції, фільтрувати шуми та відтворювати природне звучання голосу. Застосування глибоких нейронних мереж, алгоритмів обробки сигналів і технологій машинного навчання дає змогу створювати інтелектуальні системи, які здатні працювати в реальному часі та адаптуватися до різних сценаріїв.

У цій роботі розглядається програмне забезпечення для систем штучного інтелекту, що працюють з аудіоданими. Особлива увага приділяється принципам побудови таких систем, основним алгоритмам та їх застосуванню у практичних задачах, а також викликам, які постають перед розробниками в цій сфері.

Мета й завдання дослідження. Дослідити принципи розробки та функціонування програмного забезпечення систем штучного інтелекту для роботи з аудіоданими, а також визначити ефективні методи обробки, аналізу та використання таких систем у практичних застосуваннях.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Проаналізувати сучасні підходи до розробки програмного забезпечення на основі штучного інтелекту для обробки аудіоданих.

– Розглянути алгоритми розпізнавання мовлення, ідентифікації звуків, шумопригнічення та синтезу голосу.

– Оцінити ефективність використання нейронних мереж та машинного навчання в аудіосистемах.

– Дослідити прикладні аспекти застосування ШІ для роботи з аудіоданими у різних сферах (медицина, телекомунікації, безпека, розваги тощо).

– Визначити основні виклики та перспективи розвитку штучного інтелекту в обробці аудіоінформації.

– Запропонувати рекомендації щодо оптимізації програмного забезпечення для підвищення якості роботи аудіоаналітичних систем.

Практична цінність отриманих результатів Результати дослідження можуть бути корисними для розробників програмного забезпечення, дослідників у сфері штучного інтелекту, а також компаній, які працюють із технологіями обробки аудіоданих. Також результати в різних галузях можуть бути використані у сферах телекомунікацій, медицини (аудіодіагностика), безпеки (аудіоспостереження), а також у розважальних та освітніх сферах.

Запропоновані методи можуть сприяти покращенню точності розпізнавання мовлення та автоматичного аналізу аудіоінформації.

Ефективність машинного навчання у роботі з аудіо визначає оптимальні підходи до навчання нейронних мереж для роботи з голосовими даними, що допоможе розробникам створювати більш адаптивні та продуктивні моделі.

Отже, розробка та впровадження програмного забезпечення системи штучного інтелекту для роботи з аудіоданими є актуальною задачею, яка вимагає постійного вдосконалення і розробки нових рішень, вирішувалася у цій кваліфікаційній роботі.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Програмне забезпечення системи штучного інтелекту для роботи з аудіоданими призначене для аналізу, обробки та інтерпретації звукових сигналів з використанням сучасних методів машинного навчання та нейронних мереж. Основна мета системи – це автоматизація процесів розпізнавання, сегментації, класифікації та синтезу аудіофайлів.

Аналіз функціональних можливості системи:

– Розпізнавання мови – конвертація мовлення у текст із підтримкою кількох мов.

– Ідентифікація мовця – розпізнавання голосу конкретного користувача для автентифікації або персоналізації сервісів.

– Аналіз емоційного стану – визначення емоцій мовця на основі тональності та інтонації голосу.

– Фільтрація шуму – покращення якості звуку шляхом видалення фонових шумів та перешкод.

– Класифікація звуків – автоматичне розпізнавання різних типів аудіоданих, наприклад, музики, мовлення, навколишніх шумів.

– Синтез мовлення – генерація голосових повідомлень із тексту з використанням технологій Text-to-Speech (TTS).

– Музична обробка – аналіз та синтез мелодій, визначення жанру, темпу та гармонійної структури.

– Автоматичне створення субтитрів – генерація та синхронізація субтитрів для відеофайлів.

1.2 Область застосування

Програмне забезпечення на основі штучного інтелекту для роботи з аудіоданими може використовуватися в широкому спектрі галузей, від медіа та охорони здоров'я до безпеки та наукових досліджень. Завдяки сучасним технологіям машинного навчання та нейронних мереж, система здатна аналізувати, обробляти, розпізнавати та синтезувати аудіоінформацію з високою точністю.

Голосові асистенти та автоматизація взаємодії з користувачами. Ці системи використовують для обробки голосових команд, взаємодії з користувачем та виконання різноманітних завдань. Завдяки машинному навчанню, обробці природної мови (NLP) та синтезу мовлення, ці системи можуть розпізнавати людську мову, аналізувати контекст запиту та давати відповідь у вигляді голосу або тексту.

Автоматизовані системи обробки аудіо, що використовують штучний інтелект (ШІ), суттєво змінюють медіа та розважальну індустрію. Вони забезпечують якісний монтаж, адаптацію звуку, генерацію аудіоконтенту та інтерактивні можливості для користувачів..

Використання ШІ **у сфері охорони здоров'я та медичної діагностики.** відіграє ключову роль у розвитку медичних технологій, зокрема у сфері аналізу аудіоданих. Інтелектуальні системи здатні автоматично розпізнавати та аналізувати медичні звуки, що покращує діагностику, моніторинг стану пацієнтів і комунікацію між лікарями та пацієнтами.

Безпека та правоохоронна діяльність. Завдяки автоматизації аналізу аудіоданих, AI може виявляти загрози, аналізувати розмови, покращувати комунікацію силових структур та допомагати у розслідуваннях.

В освіті та наукових дослідженнях використовується для покращуючи якість навчання, персоналізуючи підхід до студентів і автоматизуючи обробку великих масивів даних у дослідженнях. Студенти можуть отримувати

автоматично розшифровані лекції, що зменшує необхідність вести конспекти та дозволяє краще зосередитися на матеріалі.

Соціальні мережі та аналіз аудіо-контенту. В цій сфері штучний інтелект дозволяє автоматизувати процеси транскрипції, аналізу емоцій, модерації контенту та створення аудіоматеріалів, що робить соціальні платформи ефективнішими, безпечнішими та більш персоналізованими.

Фінансові операції проводяться легше з використанням системи підвищуючи точність аналізу даних, автоматизуючи процеси та покращуючи безпеку фінансових операцій. Його використання охоплює банківський сектор, страхування, біржову торгівлю, кредитування та управління ризиками.

Система штучного інтелекту для роботи з аудіоданими є потужним інструментом для автоматизації та оптимізації роботи у багатьох сферах – від побутових застосунків до наукових досліджень та безпеки. Завдяки технологіям обробки природної мови (NLP), нейромережам та алгоритмам машинного навчання, вона здатна забезпечити високий рівень точності аналізу аудіо, що відкриває широкі можливості для бізнесу, освіти, охорони здоров'я та цифрової трансформації суспільства.

1.3 Можливості використання систем. Їх переваги та недоліки

Принцип роботи систем ШІ для генерації голосу. Генерація голосу починається з тексту. Спочатку потрібно прописати текст, який потім можна буде озвучувати. Це може бути будь-який текст, навіть створений за допомогою ШІ. Потім введений текст системи штучного інтелекту перетворять на людську мову, яка, по-суті, є набором звукових хвиль, що можна почути вухом або мікрофоном. Тобто штучний інтелект аналізує текст та згідно патернам, які вивчив перетворює текст на мову, використовуючи коливання хвиль. Також можна покращити вихідні дані, використовуючи різну швидкість, висоту голосу та навіть акцент.

Принцип роботи систем ШІ для розпізнавання голосу. Якщо говорити

про розпізнавання голосу, то системи ШІ працюють навпаки – аналізуються звукові хвилі та перетворюються на цифрові сигнали, що приблизно еквівалентні файлам WAV. Потім цей цифровий сигнал проходить стадію попередньї обробки, на якій видаляється фоновий шум, а будь-який розпізнаваний звук розбивається на фонеми. Далі ШІ використовує вже запрограмовані патерни, в яких прописані набори даних для розпізнавання, які порівнюються з вихідними даними після обробки цифрового сигналу.

Варіанти використання систем ШІ для генерації голосу. Існують різні методи використання систем генерації голосу, зокрема [1, 2]:

– **Створення матеріалів з підробленим голосом.** Іноді це може бути з позитивними намірами, наприклад, використання для імітації присутності померлої людини, щоб увіковічнити її голос, але є і негативні, наприклад підроблення голосу живої людини з закликами, які можуть нашкодити її репутації. Часто таке трапляється в політиці.

– **Допомога хворим.** Це один з прикладів використання нових технологій з користю. Використовують, коли є потреба у генерації голосу для людей, які не можуть говорити фізично. Наприклад, така технологія використовувалася англійським фізиком-теоретиком Стівеном Хокінгом для спілкування з оточуючими. У паралізованого фізика рухомим залишався лише вказівний палець правої руки, яким він набирав текст, що озвучувався системою штучного інтелекту для генерації голосу. Для реалізації технології у майбутньому, зокрема, може використовуватися й низка інших технологій, наприклад, імплантати мозку, які обробляють нейронні шаблони та ШІ, що перетворює ці шаблони на слова, які хоче сказати пацієнт, і генератор голосу, який говорить справжнім голосом пацієнта. Каліфорнійський університет розробляє технологію, яка допомагає людям, втратившим голос, можливість говорити [1]. Зокрема, голос пацієнта з бічним аміотрофічним склерозом був клонований із записів до того, як хвороба забрала його здатність говорити для подальшого використання у системах генерації голосу для нього. Також, у деяких формах терапії можуть

використовуватися генератори для створення голосів, що допомагають людині впоратися з емоційними або психологічними труднощами.

– **Голосові агенти для обслуговування клієнтів.** Такий метод значно полегшує життя працівників сфери кол-центрів, але тим самим і позбавляє їх роботи.

– **Зміна голосу для анонімізації.** Генератори можуть використовуватися для захисту анонімності під час розмов у публічних просторах або онлайн. Це важливо для журналістів, свідків або активістів, яким необхідно зберегти свою особистість у таємниці.

– **Освіта.** У навчальних програмах можуть використовуватися штучні голоси для створення інтерактивних віртуальних вчителів або помічників, які допомагають у процесі навчання, особливо для дітей або людей з вадами слуху.

– **Створення аудіокниг.** Програми з генерацією голосу можуть озвучити надруковані тексти, що значно полегшує та пришвидшує ознайомлення з великими об’ємами книг при навчанні або у рамках хобі.

Переваги використання ШІ для генерації голосу:

– **Анонімність і конфіденційність.** Зміна голосу дозволяє захистити свою ідентичність під час телефонних дзвінків або онлайн-комунікацій.

– **Допомога людям із вадами мовлення.** Застосунки для зміни голосу можуть бути інструментом для людей, які втратили голос через хворобу або травму. Вони можуть використовувати синтетичний голос, який звучить подібно до їхнього оригінального.

– **Автоматизація рутини.** Голосові генератори можуть застосовуватись у бізнесі для автоматизації телефонних служб і тим самим скорочення витрат на кол-центри.

– **Реалістичні дубляжі та локалізації.** Для дубляжу фільмів або створення аудіокниг ці інструменти дозволяють зменшити залежність від реальних акторів, одночасно підтримуючи високу якість і точність звучання.

– **Миттєва зміна та інтерактивність.** Багато застосунків дозволяють змінювати голос у реальному часі, що особливо зручно для ігор або спілкування в онлайн-чатах.

Недоліки використання ШІ для генерації голосу:

– **Натуральність і якість звуку.** Незважаючи на значний прогрес, багато застосунків для зміни голосу все ще не можуть досягти повністю натурального звучання. Згенеровані голоси можуть звучати штучно або механічно, що знижує загальне враження від результату.

– **Етичні питання та зловживання.** Технології зміни голосу можуть використовуватися для шахрайства, обману або дезінформації.

– **Залежність від якості вихідного сигналу.** Якість генерації голосу часто залежить від чистоти та якості оригінального запису, який використовувався у процесі створення патернів. Якщо початковий сигнал мав шум або спотворення, то і результат буде низької якості.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем

Google Cloud Speech-to-Text - це сучасні технології для автоматичного перетворення мовлення в текст і транскрипції. Цей сервіс мовлення від Google, дозволяє розробникам використовувати автовідповідачі в кол-центрах, дозволяє IoT-пристроям спілкуватися з користувачами та перетворювати текстові повідомлення на голосовий формат [5].

Які можливості відкриває технологія?

– Можна використовувати для розшифрування довго контенту, наприклад, відео. Це дає можливість зекономити час на споживання контенту та виділити головні аспекти.

– Можна легко перетворити мову на текст. Навіть якщо на відео декілька спікерів.

– Розпізнавання мовлення телефонного дзвінка. Тут ви можете транскрибувати звук будь-якої телефонної розмови.

– Технологія корисна у медичному секторі. Дає можливість розшифрувати нотатки або розмови з медичним працівником (коли ми чуємо що говорить лікар, але не запам’ятовуємо медичні терміни).

Які переваги розпізнавання мови?

– Інструмент підтримує понад 125 мов.

– Фільтрування ненормативної лексики. В текстових результатах фільтрується ненормативна лексика, тому на вихід ми можемо бачити лише професійний та етичний текст.

– Не потрібне шумозаглушення при записі аудіо. Розпізнавання може здійснювати навіть в шумі, Google Cloud може впоратися з цим.

– Простий та зручний інтерфейс. Важливо при користуванні інструментом інтуїтивно розуміти, яким чином його можна використовувати та чим він може полегшити життя.

– Висока мовленнєва адаптивність. Сервіс надає унікальні підказки підвищення точності транскрипції.

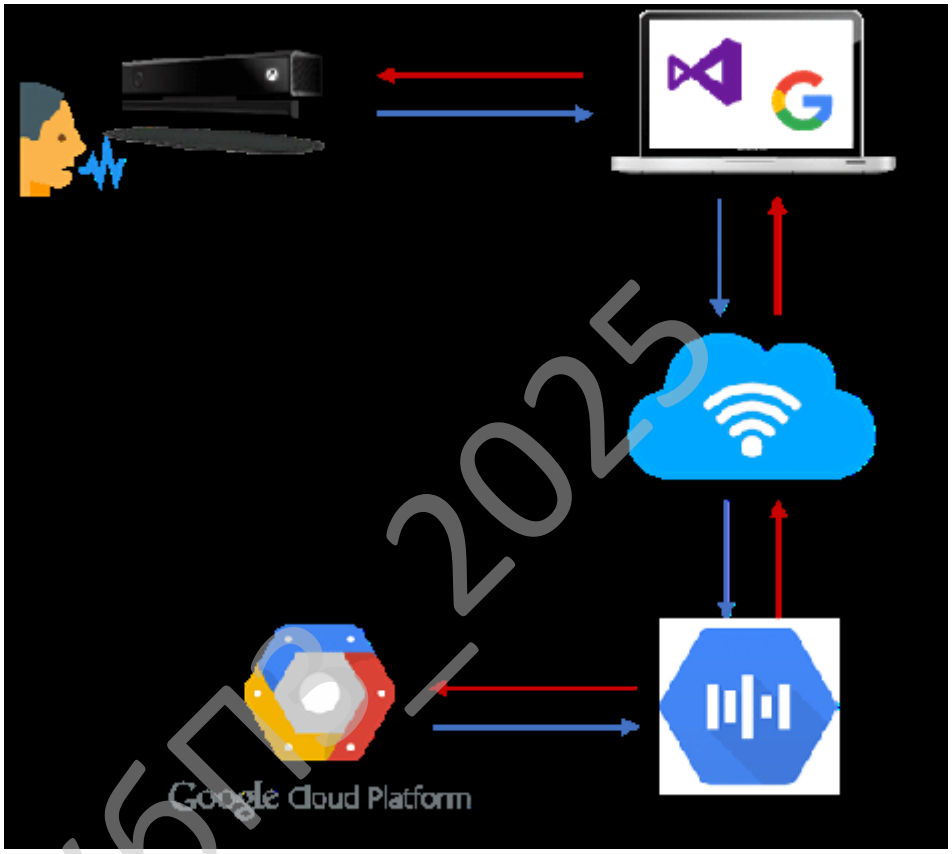

Рисунок 2.1 – Архітектура технології Google Cloud Speech-to-Text

Spotify AI DJ – інструмент на основі штучногоо інтелекту, створений для того, щоб аналізувати вподобання користувача та в майбутньому створювати плейлисти на їх основі [6].

Як працює Spotify AI DJ?

– Аналіз даних – штучний інтелект збирає інформацію про ваші музичні вподобання.

– Генерація плейлиста – Spotify створює мікс музики на основі аналізу.

– Додавання голосових коментарів – голосовий AI-асистент представляє треки та додає контекст.

– Адаптація в реальному часі – чим більше ви слухаєте, тим точніші рекомендації отримуєте.

Які можливості відкриває технологія?

– Знайти нові музичні вподобання

– Дає можливість познайомитись з новими виконавцями та створити плейлист з схожих треків, які вподобав користувач

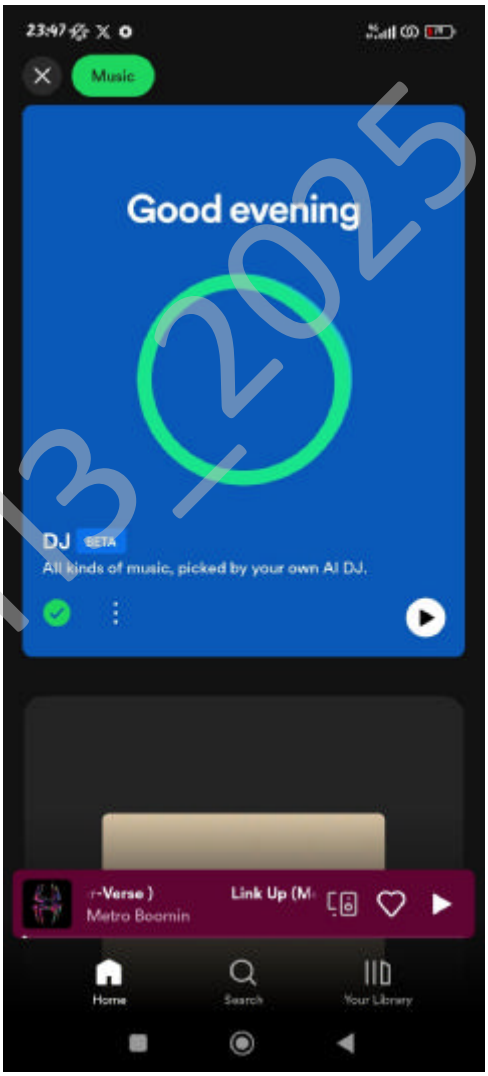

Рисунок 2.2 – Інтерфейс Spotify AI DJ

AIVA (Artificial Intelligence Virtual Artist) – інтрумент для створення

музики, на основі штучного інтелекту [7]. Дозволяє створювати музичні треки різних стилів. AIVA була розроблена компанією **Aiva Technologies** і стала відомою як перший ШІ, офіційно визнаний композитором (зареєстрований у авторському товаристві SACEM).

Які можливості відкриває технологія?

– Доступність музичної композиції для всіх.

– Навіть без знань у музиці можна створювати професійні композиції.

– Допомагає незалежним розробникам і контент-мейкерам отримати якісний саундтрек без дорогих студій.

– AIVA може допомагати музикантам знаходити нові ідеї та експериментувати з різними стилями.

– Використання ШІ як інструмента для колаборації.

– Спрощення процесу написання музики для комерційних проєктів.

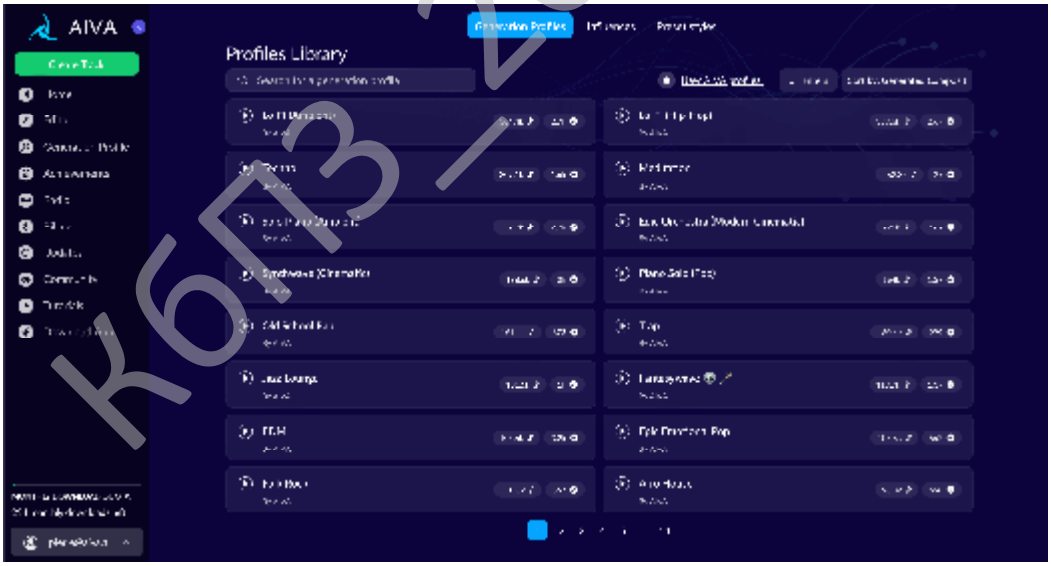

Рисунок 2.3 – Інтерфейс AIVA (Artificial Intelligence Virtual Artist)

Google Assistant – це віртуальний голосовий помічник, розроблений компанією **Google** на основі штучного інтелекту. Він використовує обробку природної мови (NLP) та машинне навчання для взаємодії з користувачами, виконання команд і надання інформації в режимі реального часу [8].

Основні можливості Google Assistant:

– Голосове управління пристроями.

– Доступний на смартфонах (Android та iOS), смарт-колонках (Google Nest, Google Home), телевізорах, автомобільних системах (Android Auto) і розумних гаджетах для дому.

– Відповідає на запитання, використовуючи пошуковий механізм Google.

– Надає актуальну інформацію (погода, новини, спортивні результати тощо) та надає доступ до керування розуьним будильником.

– Підключається до систем Google Home, Nest, Philips Hue, Xiaomi Smart Home та інших.

– Керує освітленням, камерами, розетками, замками тощо.

– Створення подій у календарі, нагадувань та будильників.

– Надсилання повідомлень та керування електронною поштою.

– Розпізнавання голосу та персоналізація.

– Вміє розпізнавати голос власника та адаптувати відповіді під користувача.

– Підтримує багатомовність та здатний перемикатися між мовами.

– Відтворення музики через Spotify, YouTube Music, Apple Music тощо.

– Працює з популярними сервісами, як-от WhatsApp, Telegram, Google Maps, Google Pay.

– Може замовляти таксі, надсилати повідомлення чи знаходити заклади поблизу.

– Миттєво перекладає розмови на 40+ мов, корисно для подорожей.

Nuance Dragon — це передова система розпізнавання мови на основі штучного інтелекту, розроблена компанією Nuance Communications. Вона призначена для точного перетворення голосу в текст, автоматизації документування та підвищення продуктивності роботи в різних сферах [9].

Де використовується Nuance Dragon?

– Медицина – лікарі швидко диктують історії хвороб, знижуючи час на заповнення документів.

– Юриспруденція – юристи автоматизують складання договорів, судових позовів.

– Бізнес – створення звітів, електронних листів, транскрипція зустрічей.

– Освіта – розпізнавання лекцій, допомога людям із порушеннями зору та моторики.

– Журналістика – швидкий запис інтерв'ю та автоматизація транскрипції.

Pindrop — це провідна технологія голосової безпеки, яка використовує штучний інтелект та аналіз аудіоданих для автентифікації користувачів та виявлення шахрайства. Ця система застосовується в фінансових установах, контакт-центрах та інших сферах, де важлива безпека голосової ідентифікації [10].

Основні можливості Pindrop:

– Інструменти для веброзробки з використанням фреймворку Django

– Аутентифікація за голосовим відбитком (Voice Fingerprinting)

– Використовує унікальні акустичні характеристики голосу для підтвердження особи.

– Виявляє фальшиві дзвінки, deepfake-аудіо та синтетичні голоси.

– Автоматизує процес входу в систему без необхідності паролів.

– Аналіз навколишнього середовища (Device & Call Metadata Analysis)

– Працює з VoIP, мобільними та стаціонарними мережами.

– Інтегрується в існуючі контакт-центри та CRM-системи.

– Підтримка глибокого аналізу аудіо.

Де використовується Pindrop?

– Банківський сектор – для безпечної ідентифікації клієнтів без PIN-кодів та паролів.

– Страхові компанії – для перевірки особи під час звернень у підтримку.

– Контакт-центри – для боротьби з шахрайством у телефонних сервісах.

– Фінансові сервіси – для аналізу голосових транзакцій та авторизації платежів.

– Державні установи – для підвищення безпеки телефонної комунікації.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Опис середовища PyCharm

PyCharm є одним із найбільш потужних середовищ для розробки на мові Python, розробленим компанією JetBrains. Воно надає повний набір інструментів, необхідних для написання, тестування, налагодження та оптимізації програм, зокрема для створення складних систем штучного інтелекту (ШІ). Оскільки Python є однією з основних мов для розробки в галузі штучного інтелекту, PyCharm пропонує розширену підтримку бібліотек для машинного навчання, обробки даних, комп'ютерного зору та аудіо. Завдяки своїм можливостям PyCharm є ідеальним інструментом для розробки систем, які обробляють аудіодані, в тому числі систем ШІ для розпізнавання мовлення, класифікації звуків або аналізу аудіофайлів [11].

Однією з основних переваг PyCharm є його інтуїтивно зрозумілий та потужний редактор коду, який включає підтримку автодоповнення, що зменшує ймовірність синтаксичних помилок і підвищує ефективність написання коду. Редактор підтримує синтаксичне підсвічування для Python, що допомагає швидко орієнтуватися в структурі коду. Крім того, можливість інтеграції з різними системами керування версіями, такими як Git, GitHub, GitLab та Bitbucket, дозволяє легко співпрацювати з іншими розробниками та відслідковувати зміни в проєкті. PyCharm також має можливість працювати з кількома гілками одночасно, що є важливим аспектом при розробці великих проєктів, зокрема в команді.

Для проектів, які потребують роботи з численними зовнішніми бібліотеками, PyCharm забезпечує зручний механізм управління залежностями через віртуальні середовища (наприклад, venv або Conda). Це дає змогу легко ізолювати пакети для кожного проєкту, знижуючи ризики конфліктів версій та забезпечуючи стабільну роботу системи.

PyCharm підтримує численні наукові та ШІ-бібліотеки, що робить його чудовим вибором для створення систем, які обробляють аудіодані. Бібліотеки, як-от Librosa (для аналізу аудіосигналів), SpeechRecognition (для розпізнавання мовлення), PyDub (для обробки аудіофайлів), дозволяють виконувати широкий спектр операцій з аудіо, починаючи від обробки сигналу та закінчуючи розпізнаванням мови. PyCharm дозволяє інтегрувати ці бібліотеки у проект із мінімальними труднощами, забезпечуючи зручне середовище для експериментів і розробки.

При розробці моделі ШІ для аудіоданих важливим аспектом є можливість інтеграції з бібліотеками для машинного навчання, такими як TensorFlow та PyTorch, а також засобами для глибинного навчання. PyCharm пропонує повну підтримку цих бібліотек, включаючи можливість використання GPU для пришвидшення навчання моделей. PyCharm автоматично налаштовує середовище для роботи з такими бібліотеками, а також пропонує спеціалізовані інструменти для відлагодження моделей машинного навчання. Це важливо, оскільки ШІ-системи для роботи з аудіоданими можуть бути складними, і на кожному етапі їхнього створення важливо перевіряти та оптимізувати їх ефективність.

У PyCharm є потужні інструменти для профілювання, які допомагають оцінити ефективність роботи алгоритмів ШІ. Це включає можливість тестування часу виконання певних частин коду та аналізу використання ресурсів, що є необхідним для оптимізації систем, які обробляють великі обсяги даних, зокрема аудіо. У режимі налагодження розробники можуть аналізувати логіку виконання моделей та змінювати параметри в реальному часі, що значно прискорює процес розробки та усунення помилок.

Інтеграція з Jupyter Notebook дає змогу розробникам працювати з інтерактивними блоками коду, що важливо при роботі з даними та моделями ШІ. Це дозволяє швидко експериментувати з різними підходами до обробки аудіо, переглядати графіки та візуалізувати результати роботи моделей. Таким чином, PyCharm надає все необхідне для ефективної розробки та тестування моделей ШІ, які обробляють аудіодані.

Підтримка роботи з базами даних також є важливим аспектом при розробці таких систем. PyCharm дозволяє інтегруватися з різними СУБД [12], такими як MySQL [13], PostgreSQL [14], SQLite [15], що дає змогу зберігати великі обсяги аудіоданих, а також метадані, пов’язані з аналізом або результатами роботи моделей ШІ. Інструменти для роботи з базами даних інтегровані безпосередньо в IDE, що дозволяє безперешкодно виконувати SQL-запити, аналізувати дані і підтримувати зв’язок між кодом і даними.

PyCharm також підтримує багато вбудованих тестових фреймворків, таких як unittest, pytest та doctest, що дозволяє створювати автоматизовані тести для перевірки роботи аудіоаналітичних моделей і забезпечує високий рівень стабільності розроблених систем. Окрім цього, PyCharm дає можливість виконувати інтеграційні тести, що критично важливо для забезпечення безпеки та ефективності систем, що працюють в реальних умовах.

Таким чином, PyCharm є одним із найбільш потужних і зручних інструментів для розробки систем ШІ для роботи з аудіоданими. Його багатофункціональність, підтримка великих наукових бібліотек, зручні інструменти для тестування і налагодження, а також потужна підтримка для роботи з базами даних і системами контролю версій робить його незамінним для розробників у галузі машинного навчання, обробки даних і комп’ютерного зору.

Особливості середовища

PyCharm має низку особливостей, які роблять його надзвичайно корисним інструментом для розробки програмного забезпечення, зокрема для створення систем штучного інтелекту та обробки аудіоданих.

Ось кілька ключових особливостей:

Інтелектуальний редактор коду: PyCharm підтримує автодоповнення, підсвічування синтаксису, виявлення помилок на лету, а також автоматичне виправлення помилок. Це дає можливість значно знизити кількість помилок при написанні коду та прискорити процес розробки.

Підтримка кількох мов програмування: Хоча PyCharm спеціалізується на Python, він також підтримує такі мови, як JavaScript, HTML, CSS, SQL, що дозволяє розробляти повноцінні веб-застосунки, які можуть включати інтеграцію з ШІ-системами.

Віртуальні середовища: PyCharm дозволяє створювати та використовувати віртуальні середовища для кожного проєкту, що допомагає уникнути конфліктів між бібліотеками та різними версіями Python. Це особливо корисно при роботі з різними проєктами, де використовуються різні версії бібліотек.

Інтеграція з Git та іншими системами контролю версій: PyCharm підтримує Git, GitHub, GitLab, Bitbucket, що дозволяє зручно працювати з репозиторіями та синхронізувати зміни. Вбудовані інструменти дозволяють швидко перевіряти, змінювати і зливати гілки, що є важливим для командної роботи над проєктами.

Налагодження та профілювання: Інструменти для налагодження в PyCharm дозволяють детально відстежувати виконання коду, змінні та їхні значення на кожному кроці. Також є можливість проводити профілювання програми, щоб оцінити її ефективність та оптимізувати використання ресурсів, що важливо для складних систем ШІ, які працюють з великими даними.

Інтеграція з науковими бібліотеками: PyCharm підтримує популярні бібліотеки для машинного навчання, такі як TensorFlow [16], PyTorch [17], scikit-learn [18], а також бібліотеки для обробки аудіо, як-от Librosa, SpeechRecognition, pydub. Це дозволяє без проблем створювати і тестувати складні моделі ШІ для аналізу та розпізнавання аудіо.

Інтерактивна робота з Jupyter Notebooks: PyCharm підтримує інтеграцію з Jupyter Notebooks, що дозволяє розробникам працювати з інтерактивними сесіями. Це особливо корисно при експериментах з алгоритмами машинного навчання або при виконанні складних обчислень.

Робота з базами даних: PyCharm має вбудовану підтримку баз даних, таких як MySQL, PostgreSQL, SQLite, MongoDB, що дозволяє інтегрувати бази даних у проєкти для зберігання великих обсягів аудіоданих та результатів обробки.

Автоматизація тестування: PyCharm має інструменти для написання юніт-тестів, інтеграційних тестів, а також для перевірки коду за допомогою фреймворків, таких як pytest або unittest. Це дозволяє підтримувати високу якість коду та забезпечувати коректну роботу систем ШІ в реальних умовах.

Підтримка Docker: PyCharm підтримує Docker [19], що дозволяє розробникам створювати контейнери для додатків, зокрема для моделей ШІ, з необхідними бібліотеками та залежностями, що забезпечує стабільність роботи на різних платформах.

Робота з великими даними: PyCharm може обробляти великі обсяги даних, що є важливим при роботі з аудіофайлами або з іншими великими наборами даних, які часто використовуються в системах ШІ.

Гнучкість у налаштуваннях: PyCharm надає широкі можливості для налаштування інтерфейсу користувача та функціоналу під потреби розробника, що робить роботу з ним максимально комфортною.

Допоміжні інструменти PyCharm

Окрім основного функціоналу, PyCharm дозволяє використовувати різні допоміжні інструменти, які значно покращують продуктивність роботи розробника.

1. Менеджер пакунків Python (Python Packages)

Менеджер пакунків у PyCharm дозволяє безпосередньо з IDE керувати залежностями проєкту. Він надає зручний інтерфейс для пошуку, встановлення,

оновлення та видалення бібліотек, що використовуються у проєкті. Для цього не потрібно вручну вводити команди в терміналі, все можна зробити через графічний інтерфейс:

Вкладка Python Packages доступна через File > Settings > Project: <Project Name> > Python Interpreter.

Користувач може встановлювати пакунки, здійснювати пошук бібліотек по репозиторіях PyPi, оновлювати існуючі бібліотеки до останніх версій або видаляти непотрібні залежності.

Цей інструмент полегшує роботу з проєктами, що мають багато зовнішніх залежностей, особливо якщо це комплексні наукові або машинно-навчальні проєкти, які використовують велику кількість бібліотек.

2. Інтеграція з FTP/SFTP та SSH

PyCharm підтримує віддалену роботу з файлами через FTP (File Transfer Protocol), SFTP (Secure FTP) [20] і SSH [21] (Secure Shell). Це дозволяє програмістам працювати над проєктами, що зберігаються на віддалених серверах, не покидаючи середовище IDE:

Віддалене підключення до серверів дає можливість відразу змінювати файли, здійснювати синхронізацію та завантажувати їх на сервер.

Крім того, є можливість запускати скрипти та виконувати команди безпосередньо через термінал в PyCharm, що полегшує тестування та налагодження.

Інтеграція з SSH дозволяє також налаштовувати безпечне підключення до серверів, що дає змогу здійснювати роботу з проектами без ризику порушення безпеки.

3. Task and Contexts

Цей інструмент дає можливість програмістам керувати завданнями та зберігати контекст роботи під час перемикання між різними частинами проєкту. Task and Contexts є корисними для організації багатозадачної роботи:

В PyCharm можна створювати завдання (Tasks) для визначення

конкретних цілей або проблем, які потрібно вирішити.

Крім того, можна зберігати контекст роботи, наприклад, відкриті файли, місця в коді, вибір рядків та інші параметри, і в будь-який момент відновлювати ці параметри.

Це особливо зручно при роботі над кількома частинами коду одночасно або при перемиканні між різними завданнями.

4. Інструменти аналізу продуктивності (Profiler)

Для оптимізації та покращення продуктивності коду PyCharm має вбудовані інструменти профілювання. Ці інструменти допомагають визначити місця, де код споживає найбільше часу або ресурсів, і надають детальну інформацію про використання процесора, пам'яті, I/O операцій тощо:

Профілювання дозволяє виявити неефективні частини коду, такі як нескінченні цикли, не оптимізовані алгоритми або зайві виклики функцій.

PyCharm підтримує інтеграцію з профайлерами, такими як cProfile та VisualVM, які можна використовувати для аналізу коду.

Інструмент дозволяє побудувати графічні звіти, що допомагають легко зрозуміти, де саме відбувається найвища затримка чи споживання ресурсів.

5. Jupyter Notebook

PyCharm має підтримку Jupyter Notebook, що є дуже корисним інструментом для аналізу даних, машинного навчання та досліджень. Jupyter дозволяє програмістам створювати інтерактивні блоки коду, що дає змогу комбінувати текстові пояснення та код у межах одного документа:

Всі зміни в Jupyter Notebook можна здійснювати в реальному часі, а також виконувати окремі частини коду без необхідності перезапуску всієї програми.

PyCharm дозволяє відкривати, редагувати та запускати .ipynb файли безпосередньо з інтерфейсу, що дає можливість працювати з даними, будувати графіки, здійснювати обчислення і тестувати моделі машинного навчання.

Це ідеальний інструмент для науковців, дослідників і студентів, які займаються аналізом даних, обробкою та візуалізацією інформації.

6. Плагіни та розширення

PyCharm підтримує велику кількість плагінів, які можуть бути додані до IDE для розширення її функціональності. Плагіни дозволяють налаштувати середовище розробки під специфічні потреби або додати нові можливості:

Для роботи з базами даних можна встановити плагіни для MySQL, PostgreSQL, MongoDB та інших.

Існують плагіни для підтримки таких фреймворків, як Django, Flask, FastAPI, а також для роботи з фронтенд-технологіями, такими як HTML, CSS, JavaScript.

Плагіни для роботи з контейнерами Docker та хмарними технологіями AWS чи Google Cloud дозволяють зручно керувати інфраструктурою безпосередньо в середовищі розробки.

Встановлення плагінів дозволяє програмістам налаштовувати PyCharm для роботи з різними мовами програмування (Java, JavaScript, C++, R тощо).

7. Інтеграція з системами контролю версій (VCS)

PyCharm має вбудовану підтримку для таких систем контролю версій, як Git, Subversion (SVN), Mercurial, GitHub та інші:

Інтеграція дозволяє безпосередньо з IDE здійснювати всі необхідні операції для роботи з репозиторіями: клонування, коміти, розв’язання конфліктів, створення гілок.

Через вбудовану панель можна переглядати зміни в коді, аналізувати історію версій, здійснювати злиття змін і зворотну сумісність.

Також можна здійснювати pull або push запити на сервери GitHub або GitLab.

Ці допоміжні інструменти роблять PyCharm потужним середовищем для розробки, яке значно покращує продуктивність розробників і надає широкий спектр функцій для ефективного управління проєктами.

2.3 Розгорнута постановка завдання

Я проаналізувала два затосунки: Google Календар та Notion. Для кожного створила асистента, щоб була можливість додавати, редагувати, видаляти або ознайомитись з подіями, за допомогою голосових команд

Система голосового асистента для створення подій у календарі розробляється для забезпечення природної взаємодії користувача з програмою через голосові команди. Головною метою є створення ефективного інструменту, що дозволяє без необхідності використання клавіатури чи графічного інтерфейсу планувати події, додавати їх до Google Календаря, а також отримувати зворотний зв'язок про виконані дії.

В основі системи лежить можливість розпізнавання мовлення, що дозволяє користувачеві формулювати команди в природній формі. Наприклад, користувач може сказати: «Створи зустріч на завтра о десятій ранку з описом наради», і система має правильно розпізнати зміст команди, визначити ключові параметри, такі як дата, час, назва події та її опис, після чого передати ці дані в Google Календар.

Одним із важливих аспектів роботи системи є підтримка природної взаємодії, що включає здатність асистента уточнювати інформацію, якщо вона була подана неповною або неточною. Наприклад, якщо користувач лише сказав «Створи подію завтра», система повинна запитати додаткові параметри, такі як точний час або назва заходу. Також передбачається можливість роботи в умовах шуму або неоднозначного формулювання, що вимагає застосування розширених алгоритмів розпізнавання голосу та корекції помилок.

Система повинна працювати в режимі інтерактивного діалогу. Це означає, що якщо користувач говорить недостатньо чітко, асистент може запропонувати альтернативні варіанти розпізнаної фрази або дозволити ввести інформацію вручну, якщо голосове розпізнавання не спрацювало належним чином. Таким чином, розробка включає реалізацію механізму дублювання команд голосом і

текстом, що дозволяє користувачеві вводити дані альтернативними способами без втрати ефективності роботи.

Оскільки система працює з датами та часом, необхідно передбачити алгоритми розпізнавання та інтерпретації часу в різних форматах. Наприклад, користувач може сказати як «двадцять восьмого лютого о п'ятій вечора», так і «через два дні о третій», і система має коректно конвертувати ці дані в стандартний формат ISO 8601, який використовується для взаємодії з Google Календарем.

Ще одним ключовим елементом є голосовий вихід. Після успішного створення події система має підтвердити її додавання, проте без надмірної деталізації (наприклад, без зачитування точного посилання на подію). Це забезпечує комфортніший досвід взаємодії, оскільки користувачеві важливо знати, що подія створена, але немає потреби отримувати всі технічні деталі.

З точки зору технічної реалізації, система передбачає використання API Google Calendar для безпосереднього створення подій, бібліотек для розпізнавання мовлення (зокрема, Google Speech Recognition) та модуля для синтезу мовлення (pyttsx3), що дозволяє асистенту відповідати голосом. Аутентифікація користувача в Google виконується через OAuth 2.0, що гарантує безпеку та збереження персональних даних.

Для зручності використання система повинна бути оптимізована для швидкого реагування на команди, а також забезпечувати можливість гнучкого налаштування, наприклад, вибір голосу асистента, зміна швидкості мови або активація певних додаткових можливостей, таких як повторне озвучення події перед її підтвердженням.

Загальна концепція даної системи базується на прагненні зробити процес створення календарних подій інтуїтивно зрозумілим та мінімізувати необхідність текстового введення, що дозволяє користувачеві взаємодіяти з нею в режимі реального часу.

Метою роботи з голосового асистента для Notion було створити систему, яка дозволить голосовому асистенту взаємодіяти з платформою Notion. Асистент повинен виконувати команди користувача, зокрема створення, редагування, видалення та пошук нотаток і завдань у Notion. Асистент повинен отримувати голосові команди користувача, обробляти їх та виконувати відповідні дії в Notion.

Користувач має можливість створювати нові нотатки. Для цього асистент запитує заголовок і зміст нотатки, після чого передає ці дані в Notion.

Якщо потрібно відредагувати нотатку, асистент спочатку знаходить її у Notion за заголовком або ключовими словами, після чого запитує, які саме зміни потрібно внести, і оновлює відповідний запис.

Для видалення нотатки асистент спочатку знаходить її у Notion, повідомляє користувачу інформацію про неї та запитує підтвердження перед видаленням. Якщо користувач підтверджує, запис видаляється.

Також передбачена можливість пошуку інформації. Користувач може запитати асистента про конкретні записи, наприклад, за ключовими словами або датою створення. Асистент отримує результати та озвучує знайдену інформацію.

Окрім нотаток, система повинна дозволяти керування завданнями. Користувач може створювати нові завдання, змінювати їх статус, переглядати список поточних завдань або отримувати нагадування про важливі події.

Для взаємодії з Notion використовується API, що дозволяє створювати, редагувати, видаляти та отримувати дані з бази. Асистент використовує систему розпізнавання голосу, яка перетворює мовлення на текст. Озвучення відповідей здійснюється за допомогою синтезу мовлення.

Щоб забезпечити стабільну роботу системи, передбачена обробка можливих помилок, таких як відсутність інтернет-з’єднання, неправильне розпізнавання голосової команди або ситуація, коли запитана нотатка не знайдена. Користувач отримує можливість керувати нотатками та завданнями в Notion за допомогою голосових команд. Всі зміни вносяться в реальному часі, а асистент надає зворотний зв’язок у вигляді голосових відповідей.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Вибір Календаря та Monday як основи для асистента зумовлений їхньою ключовою роллю в організації та управлінні подіями. Івент-менеджери постійно працюють з великою кількістю завдань, дедлайнів і комунікацій, тому ефективна система планування та координації є необхідною. Календар забезпечує автоматизацію процесів розподілу подій, нагадувань та планування, тоді як Notion дає змогу систематизувати інформацію про заходи, команду, клієнтів і завдання в одному місці.

Поєднання цих двох інструментів у межах асистента створює ефективну екосистему для управління подіями. Інтеграція з Календарем дозволяє відстежувати розклад, отримувати нагадування, знаходити вільні часові слоти для зустрічей і автоматично створювати події на основі голосових команд. Використання Notion додає гнучкість у роботі з інформацією: можна швидко занотовувати ідеї, вести бази даних щодо клієнтів, спонсорів, локацій та інших аспектів організації заходів.

Ця система дозволяє оптимізувати робочі процеси івент-менеджера, мінімізувати ризик втрати важливої інформації та зробити планування більш структурованим. У перспективі цей асистент може стати незамінним помічником у професійній діяльності, спрощуючи управління подіями та підвищуючи ефективність роботи.

Розроблена система Ця програма створена для того, щоб допомогти користувачу створювати події в Google Calendar за допомогою голосового асистента. Вона поєднує кілька ключових елементів: розпізнавання голосу, текстове озвучування, а також взаємодію з Google Calendar API [22] для створення подій. Нижче прописано детально опис функціонування програми:

Ініціалізація програми та налаштування голосу. Програма використовує бібліотеку pyttsx3 для перетворення тексту в мову (TTS — Text to Speech), що дозволяє асистенту озвучувати команди і запити. Налаштовується швидкість мовлення та гучність, а також вибирається голос за замовчуванням із доступних в системі.

Програма використовує speech_recognition для розпізнавання голосу користувача, що дає змогу взаємодіяти з асистентом за допомогою голосових команд.

Авторизація в Google API. Для того, щоб асистент міг взаємодіяти з Google Calendar, потрібно пройти авторизацію через Google API. Для цього програма використовує OAuth 2.0 [23] — стандартний механізм авторизації, який дозволяє безпечно взаємодіяти з даними користувача в Google.

Програма перевіряє наявність збережених облікових даних (token.pickle), і якщо вони відсутні або застарілі, автоматично ініціює процес авторизації через браузер.

Розпізнавання голосових команд. Коли програма запущена, асистент слухає голос користувача за допомогою мікрофона, використовуючи бібліотеку speech_recognition. Він реагує на команди типу "create event" або "stop". Якщо розпізнається команда для створення події, програма переходить до запиту необхідних даних.

Якщо голосовий ввід не був розпізнаний (наприклад, через шум або незрозуміле слово), програма не зациклюється на помилці, а просто запитує користувача знову.

Запит необхідних даних для події. Після того, як користувач дає команду на створення події, асистент просить ввести або сказати наступну інформацію:

Назва події — що за захід відбувається (наприклад, "Team Meeting").

Опис події — коротка інформація про захід (наприклад, "Discuss project updates").

Час початку та закінчення — ці значення користувач вводить вручну в

консоль у форматі ISO (наприклад, "2025-03-07T10:00:00" для початку та "2025-03-07T11:00:00" для завершення). Якщо користувач не може сказати дату або час, він може ввести їх вручну.

Парсинг та валідація дати та часу. Введені дати перевіряються і конвертуються в формат, який Google Calendar API може розпізнати (ISO 8601). Для цього використовуються інструменти з бібліотеки dateutil.parse [24], що дозволяє коректно парсити різні формати дат і часу.

Створення події в Google Calendar. Коли всі дані зібрані і перевірені, програма використовує Google Calendar API для створення події. Створена подія містить такі поля:

Заголовок події — назва, яку вказав користувач.

Опис події — текст, який пояснює, що саме буде відбуватися.

Час початку та завершення — введені дати й часи, які були перетворені в правильний формат.

Всі ці дані передаються до Google Calendar API, де подія створюється в календарі користувача.

Оповіщення користувача. Після створення події, асистент повідомляє користувача, що подія була успішно додана до календаря. Водночас, якщо щось пішло не так під час створення події (наприклад, через неправильний формат дати чи іншої помилки), програма виводить повідомлення про помилку і озвучує це.

Завершення роботи програми. Якщо користувач вирішує завершити роботу з програмою, він може сказати команду "stop" або "exit". Після цього програма припиняє свою роботу та прощається з користувачем.

Деталі взаємодії з користувачем:

Голосові запити: У всіх випадках, коли асистент потребує від користувача відповіді, він використовує голосові запити, щоб зробити процес більш природним і інтуїтивно зрозумілим.

Підтримка введення через клавіатуру: Якщо асистент не розпізнає

голосовий ввід, користувач може ввести відповідь вручну через консоль. Це дає можливість працювати безперервно навіть за умови, що голосове введення не вдалося.

При запуску програма ініціалізує голосовий асистент, який вітає користувача і пропонує йому можливість створити подію. Спочатку користувач може сказати команду "create event", і асистент запросить надані дані для створення події, зокрема назву, опис, початкову та кінцеву дату та час. У разі, якщо асистент не зможе зрозуміти голосовий ввід, він дозволяє користувачу ввести ці дані вручну через консоль.

Для обробки дати та часу програма використовує бібліотеку dateutil, яка дозволяє конвертувати текстову дату в формат, зрозумілий для Google Calendar API. Користувач вводить час вручну у стандартному форматі ISO (наприклад, "2025-03-07T10:00:00") для початку і завершення події. Після цього, програма намагається створити подію в Google Calendar.

При успішному створенні події програма повідомляє користувача про успіх, але не озвучує посилання на подію. Якщо ж виникає якась помилка під час створення події, вона буде виведена у консоль і озвучена.

Таким чином, програма забезпечує інтерактивну взаємодію з користувачем, дозволяючи створювати події в Google Calendar, використовуючи голосовий асистент або введення тексту, залежно від потреб користувача.

Monday — це потужна платформа для організації знань, управління проектами та координації роботи команд. Вона об'єднує можливості баз даних, календарів та таск-менеджменту в одному інтерфейсі. Використовуючи Monday, можна ефективно створювати і організовувати завдання, відслідковувати виконання проектів, створювати спільні робочі простори та зберігати різноманітні типи інформації (тексти, таблиці, зображення, посилання тощо).

У цій системі інтеграція з Monday дозволяє автоматизувати управління завданнями, проектами та базами даних через голосові або текстові команди. Ось розширене пояснення, як система може працювати з Monday і чому це корисно.

Для створення інтелектуального асистента для роботи з Monday необхідно розробити систему, яка дозволить користувачам взаємодіяти з платформою за допомогою голосових або текстових команд. Цей асистент повинен стати інструментом для автоматизації управління завданнями, що зберігаються у Monday. Основна ідея полягає у створенні механізму, який дозволить швидко і ефективно виконувати рутинні дії, такі як створення записів, редагування існуючих даних, отримання інформації, пошук необхідних документів, а також інтеграцію з іншими сервісами для розширення функціоналу.

Функціонування системи повинно **бути засноване на використанні API Monday**, яке дозволяє програмно взаємодіяти з базами даних, сторінками та іншими елементами. Завдяки цьому можна реалізувати процеси створення нових записів у відповідних категоріях, їх зміну та перегляд. Асистент повинен підтримувати ідентифікацію користувачів, що дозволить кожному працювати зі своїми персональними даними в безпечному середовищі. Важливим аспектом є можливість налаштування логіки запитів, щоб користувач міг отримувати саме ті дані, які йому потрібні у конкретний момент часу.

Окрім базових функцій, **асистент повинен володіти можливістю розпізнавання голосу та аналізу природної мови.** Це дозволить користувачам спілкуватися з системою так, як вони звикли, використовуючи звичайні фрази для виконання команд. Наприклад, користувач може сказати «Додай нову задачу про зустріч» або «Які у є мене завдання?», і система повинна правильно інтерпретувати запит та відповідно взаємодіяти з Monday. Для цього необхідно використовувати технології обробки природної мови, такі як моделі машинного навчання або готові API для розпізнавання мовлення.

Важливим компонентом є **система обробки відповідей.** Асистент повинен не тільки виконувати команди, а й надавати корисну зворотну інформацію. Якщо користувач хоче змінити опис певного завдання, асистент повинен спочатку знайти це завдання, підтвердити його зміст і лише потім внести зміни. Також повинна бути передбачена можливість повідомлення про помилки

або некоректні запити, щоб користувач розумів, що саме відбулося, якщо система не змогла виконати його команду.

Для забезпечення швидкої та стабільної роботи асистента необхідно оптимізувати механізм обробки запитів. Це може включати використання кешування, щоб зменшити кількість звернень до API Monday, та налаштування багатопотокової обробки запитів для покращення продуктивності. Також слід враховувати можливі обмеження API та знаходити способи їх ефективного обходу.

З точки зору користувацького досвіду, важливо зробити асистента максимально зручним і простим у використанні. Він повинен бути доступним як через текстовий інтерфейс, так і через голосові команди. Також варто передбачити можливість використання веб-додатку або мобільного додатку, де користувач зможе переглядати свої завдання, отримувати аналітику про виконані справи та керувати своїм робочим простором у Monday.

Реалізація такого асистента дозволить значно підвищити ефективність роботи з Monday, автоматизувати багато рутинних процесів та спростити управління інформацією. Він стане корисним інструментом для всіх, хто активно використовує Monday своїй роботі чи особистому житті, допомагаючи краще організовувати завдання, проекти та важливі події.

3.2 Розробка структурної схеми

Структурна схема – це графічне зображення взаємозв’язків між основними компонентами системи. Вона допомагає зрозуміти архітектуру програмного забезпечення, визначити основні модулі та спосіб їхньої взаємодії [26].

Програма побудована для взаємодії з користувачем через голосові команди або текстові введення, дозволяючи створювати події в Google Calendar. Вона використовує кілька основних компонентів, що забезпечують функціональність асистента.

Для інтеграції з Google Calendar програма використовує API Google Calendar. Після того, як користувач авторизується через OAuth 2.0, програма отримує доступ до його календаря і може додавати події. Це відбувається шляхом створення події з зазначеними деталями, такими як заголовок, опис, час початку і кінця. Авторизація відбувається один раз, і збереження токена доступу в файлі дозволяє програмі працювати без повторної авторизації при наступних запусках.

Основна особливість програми полягає в тому, що вона використовує систему голосового розпізнавання, зокрема бібліотеку speech_recognition. Це дозволяє користувачу взаємодіяти з програмою, використовуючи голосові команди, такі як "створити подію". Коли програма чує таку команду, вона запитує додаткові деталі для події, наприклад, заголовок, опис, час початку та кінця. За допомогою сервісу Google Speech-to-Text відбувається перетворення голосу в текст, і програма сприймає ці дані для подальшої обробки.

Що стосується голосових повідомлень, програма використовує бібліотеку pyttsx3 для озвучування тексту. Кожен запит або підтвердження, а також повідомлення про помилки або успішне створення події програма озвучує голосом, що робить взаємодію з нею зручною і природною. Водночас, програма не озвучує зайвих деталей, таких як посилання на події або час, що дозволяє зберегти простоту і зручність.

Програма також дозволяє користувачеві вводити дату і час події вручну, якщо голосове розпізнавання не вдалося або користувач просто вважає це зручнішим. Після введення цих даних програма використовує бібліотеку dateutil.parser для перетворення дат у правильний формат, який потрібен для додавання події в календар.

Крім того, програма має можливість обробляти помилки, наприклад, якщо введена некоректна дата або сталася проблема при створенні події. У разі таких помилок програма повідомить користувача про проблему і надасть відповідне голосове повідомлення. Усі ці функції об’єднані в інтерактивний процес, що дозволяє користувачеві легко створювати події через голосові або текстові

команди. Якщо користувач хоче завершити роботу з асистентом, він може сказати "стоп", і програма завершить свою роботу.

Початок програми –програма запускається.

Визначення параметрів:

– Встановлюється ідентифікатор **BOARD_ID**, який визначає, з якої дошки брати завдання.

– Формується заголовок HEADERS, який містить API-токен та версію API (API-Version: 2023-10).

Формування GraphQL-запиту – створюється запит для отримання всіх потрібних елементів і з дошки для роботи з запитом користувача.

Відправлення запиту через requests.post()

– Запит відправляється до https://api.monday.com/v2.

– Якщо сервер повертає відповідь, перевіряється статус-код (response.status_code).

– У разі не повернення відповіді, асистент видає, що була помилка у відправці даних від сервера до клієнта з відповідним кодом помилки.

Перевірка статусу відповіді

– Якщо статус-код **200 (OK)**, запит пройшов успішно.

– Якщо інший код (наприклад, 400, 401, 500), це означає, що сталася помилка, і програма виводить повідомлення про проблему.

Обробка отриманих даних

– Перевіряється, чи є у відповіді необхідні дані (boards і items).

– Якщо дані є, формується відповідь на запит користувача.

– Якщо завдання поставлене не коректно – асистент просить назвати завдання.

Обробка помилок

– Використовується try-except, щоб відловлювати можливі помилки під час запиту.

– Якщо запит не вдається через помилку мережі або неправильний запит, у консоль виводиться текст помилки (RequestException).

Завершення роботи програми – якщо всі завдання були виведені або оброблено помилку, функція завершує роботу.

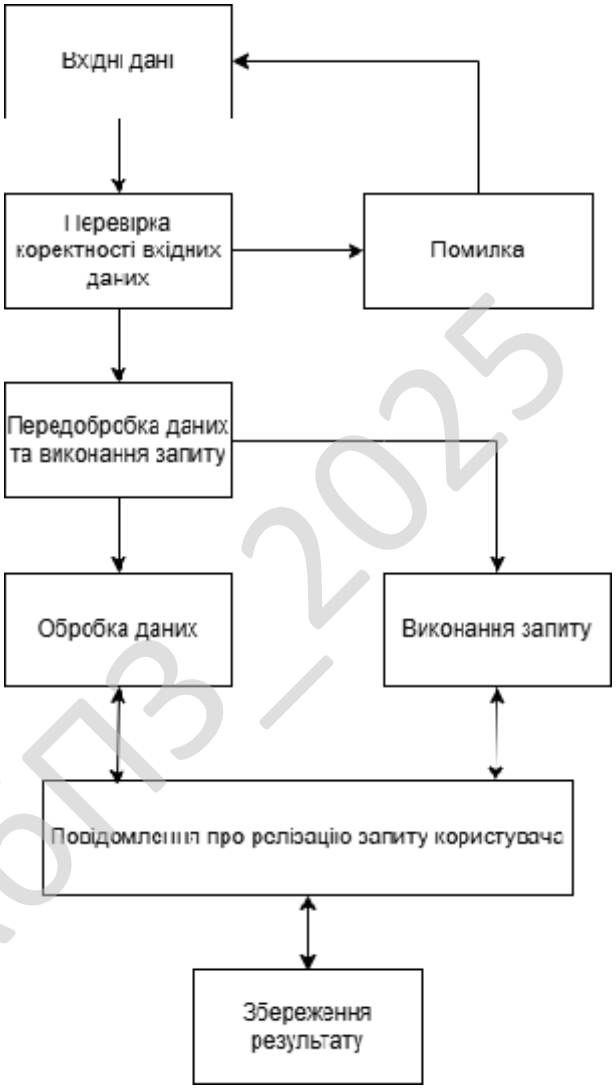

Рисунок 3.1 – Структурна схема систем (загальна для обох асистентів)

3.3 Розробка функціональної схеми

Функціональна схема програми описує основні етапи її роботи, від ініціалізації до створення події в Google Calendar. Кожен етап взаємопов'язаний і

реалізується через окремі компоненти програми, що забезпечує плавне та зручне використання.

Ініціалізація програми

Програма починає свою роботу з ініціалізації необхідних бібліотек і налаштувань. Для голосового озвучування та розпізнавання мови використовуються бібліотеки **pyttsx3** та **speech_recognition**. Перша забезпечує можливість проговорювання тексту, а друга — розпізнавання голосових команд користувача. Крім того, програма налаштовує параметри голосу для озвучування, вибираючи голос за замовчуванням.

Авторизація в Google API

Далі програма здійснює авторизацію через **OAuth 2.0** для доступу до Google Calendar API. Якщо авторизація вже виконана раніше і токен доступу збережено в локальному файлі (token.pickle), програма завантажує його для подальшого використання. Якщо токена немає або він застарів, програма запускає процес авторизації, запитуючи у користувача дозвіл на доступ до календаря.

Голосове розпізнавання

Після успішної ініціалізації програма слухає голосові команди користувача. Це здійснюється за допомогою **speech_recognition**, яка перетворює аудіо в текст. Якщо користувач не може або не хоче використовувати голос, програма надає можливість ввести команду або дані вручну.

Запит на введення даних для події

Після того, як програма отримала команду про створення події, вона запитує у користувача необхідну інформацію. Спочатку це заголовок події, потім — опис. Якщо користувач хоче додати конкретний час початку та завершення події, програма запитує ці дані в текстовому форматі (наприклад, ISO формат). Якщо розпізнавання голосу не спрацювало, користувач може ввести ці дані вручну.

Обробка дат та часу

Всі введені дати й часи обробляються за допомогою **dateutil.parser**, яка перетворює їх у стандартний формат ISO (наприклад, "2025-03-07T10:00:00"). Якщо введена дата некоректна, програма видасть повідомлення про помилку та надасть користувачеві можливість виправити введені дані.

Створення події в Google Calendar

Коли всі необхідні дані зібрані, програма формує запит на створення події в Google Calendar через API. Усі дані про подію (заголовок, опис, час початку і завершення) надсилаються на сервер Google, і якщо все правильно, подія додається до календаря.

Обробка помилок

Програма перевіряє всі введені дані на наявність помилок і виводить відповідні повідомлення, якщо виникають проблеми. Наприклад, якщо введена дата або час не можуть бути розпізнані або створення події не вдалося, користувач отримує повідомлення про помилку і може виправити ситуацію.

Завершення роботи програми

Коли користувач більше не хоче створювати події або хоче завершити роботу програми, він може сказати команду "стоп", і програма завершить свою роботу. Також передбачено завершення роботи програми у разі помилки чи за іншою причиною.

Функціональна схема програми побудована таким чином, щоб забезпечити простоту використання через голосові команди, а також альтернативний спосіб введення даних вручну. Це дозволяє користувачеві швидко та зручно додавати події до свого календаря, навіть якщо він не може використовувати голосову команду через технічні або інші обставини.

Асистент роботи з **Monday** виконує такі функції:

– Отримання списку завдань із **Monday.com**.

– Додавання нових завдань.

– Видалення завдань.

– Голосове керування для взаємодії з платформою. Виведення інформації як у голосовому форматі, так і в текстовому (консоль або UI).

Функціональні модулі:

Модуль взаємодії з користувачем:

– Голосове керування (Speech Recognition + Text-to-Speech).

– Введення команд текстом (через консоль або чат-інтерфейс).

– Виведення результатів голосом та текстом.

API-запити до Monday:

Отримання списку завдань:

– Додавання нового завдання.

– Видалення завдання.

Логіка обробки команд:

– Аналіз тексту або голосового вводу.

– Визначення, яку саме дію потрібно виконати.

– Виклик відповідного запиту до Monday.com API.

Обробка відповіді від API:

– Отримання та форматування інформації.

– Виведення у вигляді списку, таблиці або голосового повідомлення.

– Обробка помилок (неправильні дані, API-недоступне тощо).

Рисунок 3.2 – Функціональна схема для обох систем

3.4 Розробка діаграми процесів

Діаграма процесів системи штучного інтелекту для роботи з аудіоданими описує послідовність дій, які виконуються від моменту отримання голосової команди користувача до моменту обробки та виконання відповідного запиту.

Процес починається з того, що користувач звертається до системи, подаючи голосову команду, наприклад, створення нового завдання у Monday або додавання події в Google Calendar. Система приймає аудіосигнал і передає його до модуля розпізнавання мовлення, який використовує спеціальні алгоритми, наприклад, Google Speech-to-Text або власну нейромережеву модель, для конвертації звукового сигналу в текст.

Отриманий текст аналізується модулем обробки природної мови (NLP[43, 44]), який розпізнає намір користувача, визначає ключові слова та уточнює контекст команди.

Далі система звертається до модуля інтеграції з зовнішніми сервісами, який відповідає за взаємодію з API Google Calendar або Monday. Якщо команда передбачає отримання інформації, наприклад, список завдань або майбутніх

зустрічей, система надсилає запит до відповідного API, отримує відповідь та передає її у форматі, зручному для користувача. Якщо ж необхідно створити або змінити дані, система генерує відповідний запит, валідує отримані дані та передає їх до API.

Після успішного виконання запиту система повертає користувачеві відповідь. Це може бути текстове повідомлення у вигляді голосового сповіщення або відображення інформації у візуальному інтерфейсі, якщо такий передбачений. Якщо в процесі виконання виникла помилка, наприклад, неправильний формат введених даних або проблеми з доступом до сервісу, система повідомляє про це користувача та пропонує альтернативні варіанти дій.

Важливим етапом у процесі є забезпечення безпеки та збереження конфіденційності даних. Усі API-запити автентифікуються за допомогою збережених ключів, які зберігаються у файлі dotenv.env, щоб уникнути компрометації доступів. Додатково система може включати механізми контролю доступу, щоб користувачі з різними правами могли виконувати лише дозволені операції.

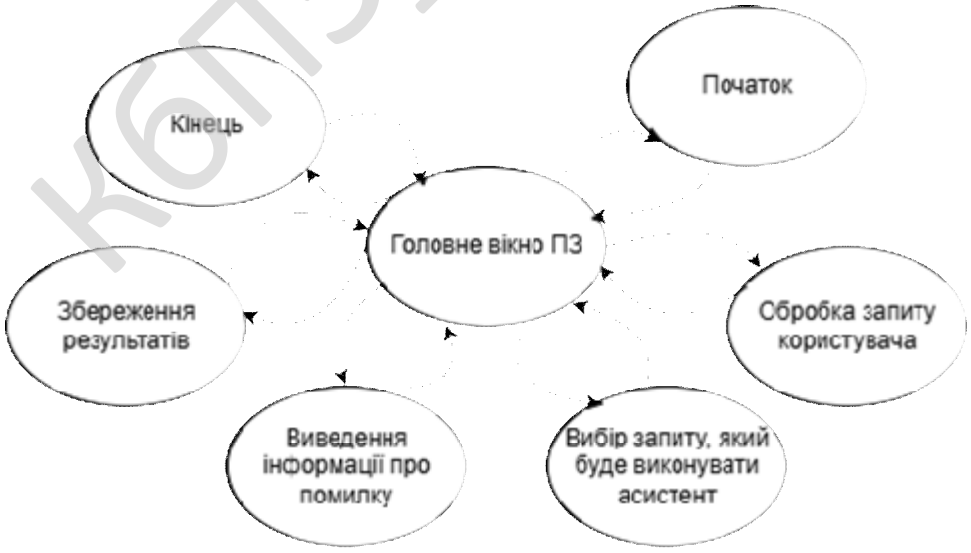

Рисунок 3.3 – Діаграма процесів для обох систем

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Алгоритм функціонування AI-асистента для Google Calendar:

Отримання команди від користувача

– Користувач вводить команду голосом або текстом.

– Якщо голосовий ввід, асистент використовує модуль розпізнавання мови (Speech-to-Text) для перетворення голосу в текст.

– Якщо текст не розпізнано, асистент просить користувача ввести команду вручну.

Аналіз команди

Визначається, яка дія має бути виконана:

– Створення події.

– Оновлення події.

– Видалення події.

– Отримання списку подій.

– Якщо необхідні додаткові параметри (дата, час, назва події), але їх немає, асистент запитує уточнення.

Аутентифікація в Google Calendar API

– Перевіряється наявність токена доступу.

– Якщо токен відсутній або недійсний, асистент ініціює процедуру авторизації через OAuth 2.0.

Формування API-запиту

– Генерується відповідний запит у форматі JSON.

– Відправляється HTTPS-запит через бібліотеку google-api-python-client.

Обробка відповіді від API

– Якщо запит успішний, витягується потрібна інформація (список подій, підтвердження створення/оновлення/видалення).

– Якщо є помилки (400, 401, 500), асистент повідомляє про проблему та пропонує повторити запит або оновити авторизацію.

Вивід інформації користувачеві

– Озвучення результатів через Text-to-Speech.

– Вивід текстової інформації в консоль.

Обробка повторних запитів або завершення роботи

– Якщо користувач дає нову команду, цикл повторюється.

– Якщо користувач каже "Exit", асистент завершує роботу.

Цей алгоритм забезпечує взаємодію AI-асистента з Google Calendar, автоматизуючи керування подіями.

Алгоритм функціонування AI-асистента для Monday.com

– Отримання команди від користувача

Вибір способу введення

Користувач взаємодіє з асистентом через:

– **Голосовий ввід** (використовується модуль розпізнавання мови, наприклад, SpeechRecognition).

– **Текстовий ввід** (через консоль або чат-інтерфейс).

Розпізнавання голосової команди

Якщо користувач використовує голосовий ввід:

1. Запускається модуль **Speech-to-Text (STT)**.

2. Програма записує голосове повідомлення.

3. Голос перетворюється на текст.

4. Виконується базова перевірка розбірливості тексту.

5. Якщо текст не розпізнано або він порожній, асистент просить повторити або ввести команду текстом.

Попередня перевірка введеної команди

– Перевіряється, чи не порожня команда.

– Видаляються зайві пробіли, спеціальні символи, приводиться до нижнього регістру.

– Виконується перевірка на відомі ключові слова (наприклад, *tasks, add, update, delete*).

2. Аналіз команди та визначення дії

Визначення типу запиту

На основі ключових слів асистент визначає, яку операцію потрібно виконати:

1. Отримання списку завдань (наприклад: *"Show my tasks"*, *"List tasks"*)

2. Додавання нового завдання (*"Add task 'Meet client'"*)

3. Видалення завдання (*"Delete task 'Call supplier'"*)

Перевірка наявності додаткових параметрів

Якщо команда потребує додаткової інформації (наприклад, ID завдання або його назву), асистент перевіряє:

– Чи вказав користувач необхідні деталі.

– Якщо ні – запитує уточнення.

3. Формування та виконання API-запиту до Monday.com

Створення GraphQL-запиту

– На основі визначеної дії генерується відповідний **GraphQL-запит**.

– Формується структура JSON із параметрами.

– У заголовку додається API-ключ та версія API ("API-Version": "2023-10").

Надсилання запиту до Monday.com API та обробка відповіді від API

1. Використовується **requests.post()** для відправки HTTP-запиту.

2. Запит передається через **HTTPS** для безпечного з'єднання.

Перевіряється **статус відповіді**:

– **200 OK** – запит успішний, отримані дані обробляються.

– **400 Bad Request** – помилка у форматі запиту.

– **401 Unauthorized** – проблема з авторизацією (невірний API-ключ).

– **500 Internal Server Error** – проблема на сервері Monday.com.

4. Обробка відповіді та виведення інформації

Форматування отриманих даних

Якщо відповідь успішна, дані обробляються:

– Отримується список завдань.

– Витягується **ID**, **назва**, **статус**.

– Якщо список завдань порожній – повідомляється про відсутність завдань.

Вивід інформації користувачу

Відповідь може бути представлена у двох форматах:

– Голосовий вивід (Text-to-Speech, озвучування тексту).

– Текстовий вивід у консоль**.**

5. Обробка помилок та повторення запиту

Стандартна обробка помилок

– Якщо відповідь **400 або 500**, повідомляється про помилку і запитується повторна спроба.

– Якщо **401 Unauthorized**, перевіряється правильність API-ключа.

Повторний запит:

– Якщо сервер тимчасово недоступний – програма повторює запит через 5 секунд.

– Якщо після 3 спроб помилка не зникає – асистент припиняє спроби.

6. Завершення роботи або обробка наступної команди:

– Програма переходить у **режим очікування наступної команди**.

– Якщо користувач каже *"Exit"*, асистент завершує роботу.

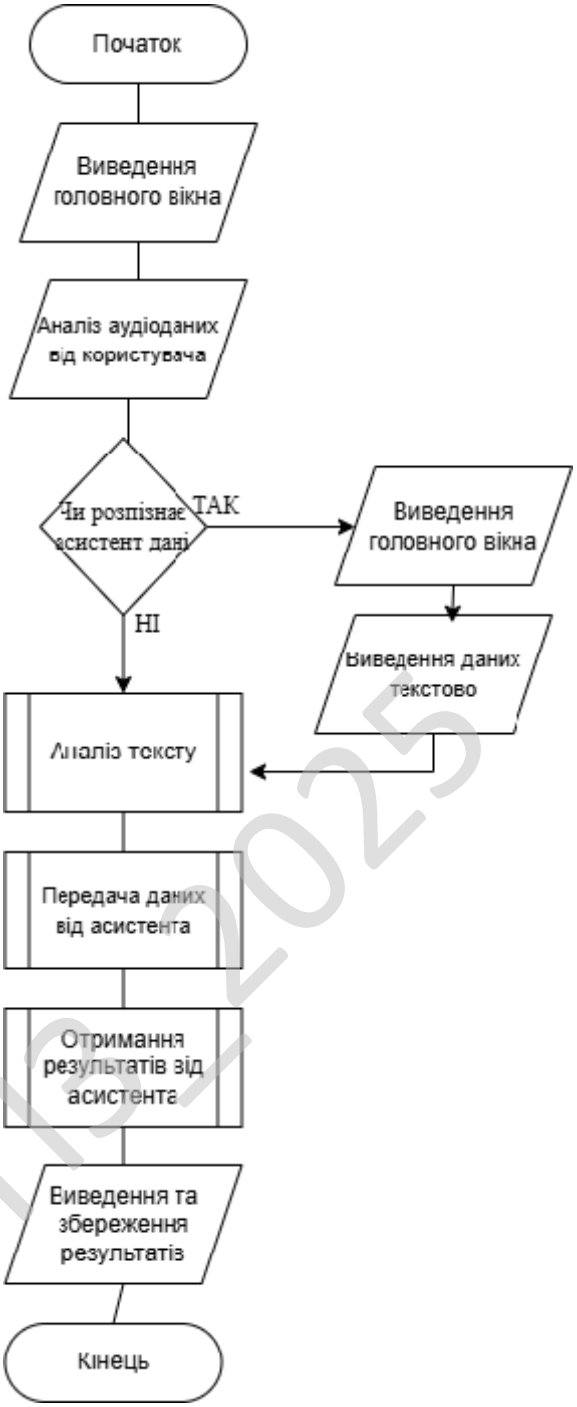

Рисунок 4.1 – Блок-схема роботи основної програми

Лістинг основного файлу програми (calendar_main.py):

```
import speech
import pyttsx3
from google_auth import authenticate_google_api
from speech_recogn import recognize_speech
from calendar_operations import create_event, get_week_schedule, edit_event,
ask_event
from delete_event import delete_event

'''Головний файл програми - тут викликається асистент та посилається запит на
виконання команд, які асистент має виконати.'''

# Text-to-speech settings
engine = pyttsx3.init()
engine.setProperty('rate', 150)  # Set speech rate
engine.setProperty('volume', 1)  # Set volume (from 0 to 1)

def assistant_create_event():
    service = authenticate_google_api()
    speech.speak("Hello! I am your voice assistant.")
    while True:
        speech.speak("Say 'create event', 'check schedule', 'ask about',
'delete event' , 'edit' or 'stop' to exit.")
        user_input = recognize_speech() or input("Enter the action: ")
        if not user_input:
            continue
        if user_input.lower() in ["exit", "quit", "stop"]:
            speech.speak("Goodbye!")
            break
        if "create event" in user_input.lower():
            create_event(user_input)
        elif "check schedule" in user_input.lower():
            get_week_schedule(service)
        elif "delete event" in user_input.lower():
            speech.speak("Which event would you like to delete?")
            event_title = recognize_speech()
            delete_event(service, event_title)
        elif "edit" in user_input.lower():
            edit_event(service)
        elif "ask about" in user_input.lower():
```

```
            ask_event(service)
        else:
            speech.speak("Goodbye!")
            break

if __name__ == "__main__":
assistant_create_event()
```

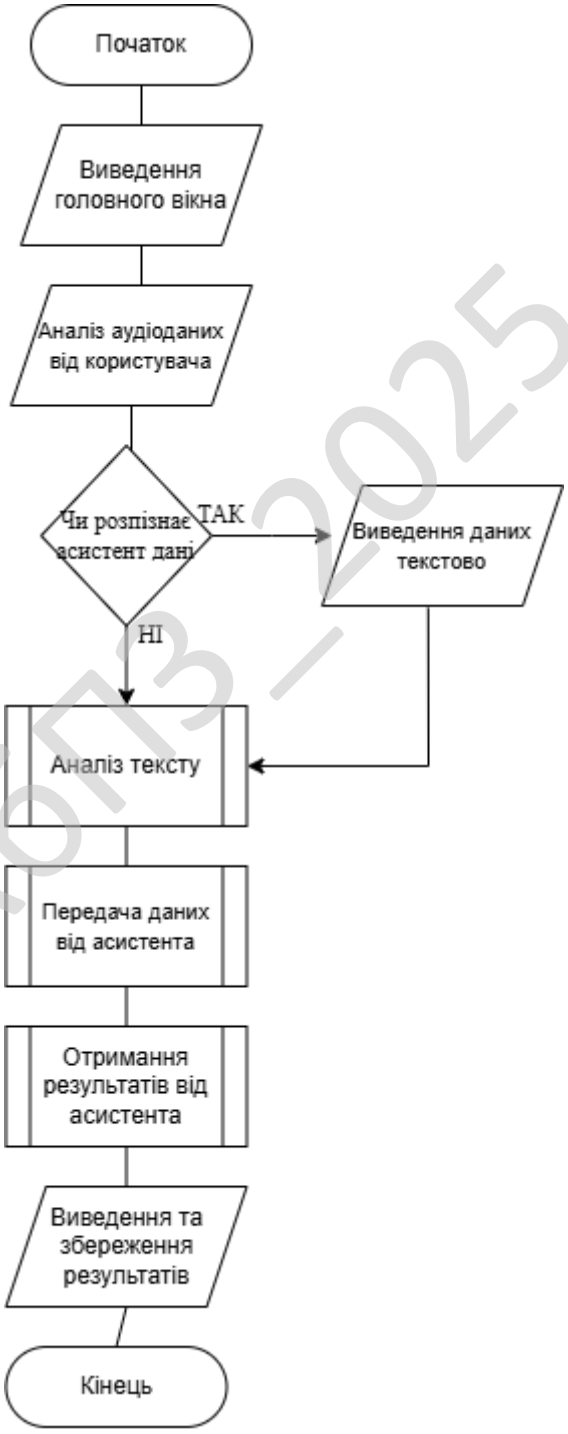

Рисунок 4.2 – Блок-схема роботи підпрограми

Лістинг функції для створення події в календарі(асистента для роботи з Google Calendar)

```
        def create_event(service, summary, description, start_time, end_time):
            event = {
                'summary': summary,
                'description': description,
                'start': {'dateTime': start_time, 'timeZone': 'Europe/Kiev'},
                'end': {'dateTime': end_time, 'timeZone': 'Europe/Kiev'},
            }
            service.events().insert(calendarId='primary', body=event).execute()
            speech.speak("Please say the event title...")
            title = recognize_speech() or input("Enter the event title: ")
            speech.speak("Please say the event description...")
            description = recognize_speech() or input("Enter the event description:
")
            start_time_str = input("Enter start time (YYYY-MM-DDTHH:MM:SS): ")
            end_time_str = input("Enter end time (YYYY-MM-DDTHH:MM:SS): ")
            start_time = parse_date(start_time_str)
            end_time = parse_date(end_time_str)
            if start_time and end_time:
                try:
                    create_event(service, title, description, start_time, end_time)
                    speech.speak("Event created successfully!")
                except Exception as e:
                    speech.speak(f"Error creating event: {e}")
            else:
                speech.speak("Failed to create event due to invalid time input.")
```

Лістинг функції для демонстрації всього розкладу на тиждень в календарі(асистента для роботи з Google Calendar)

```
        def get_event_details_by_title(service, title):
            events = service.events().list(calendarId='primary', q=title,
singleEvents=True, orderBy='startTime').execute().get('items', [])
            if not events:
                print(f"No event found with the title {title}.")
                return
            for event in events:
```

```
            event_summary = event.get('summary', 'No title available')
            event_description = event.get('description', 'No description
available')
            start = event.get('start', {}).get('dateTime', event.get('start',
{}).get('date', 'No start time available'))
            print(f"Event: {event_summary} at {start} with description:
{event_description}")
```

Лістинг функції для створення завдання в Monday(асистента для роботи з Monday)

```
from speak_help import speak
from dotenv import load_dotenv
import requests
import os

'''Файл для створення тасок в Monday'''

load_dotenv("dotenv.env")
MONDAY_API_KEY = os.getenv("MONDAY_API_KEY")
BOARD_ID = os.getenv("BOARD_ID")
HEADERS = {
    "Authorization": MONDAY_API_KEY,
    "Content-Type": "application/json"
}


def create_task(task_name):
    query = """
    mutation createTask($board_id: ID!, $item_name: String!) {
        create_item (board_id: $board_id, item_name: $item_name) {
            id
        }
    }
    """
    variables = {
        "board_id": BOARD_ID,
        "item_name": task_name
    }
    response = requests.post("https://api.monday.com/v2", json={"query":
```

```
query, "variables": variables},
                                    headers=HEADERS)

            if response.status_code == 200:
                speak(f"Task '{task_name}' has been created successfully.")
            else:
                speak(f"Failed to create the task. Error: {response.text}")
```

Лістинг функції видалення завдання в Monday

```
import requests
import os
from dotenv import load_dotenv
import speak_help as speak

'''Файл створений для видалення тасок з таск-менеджера'''

# Завантаження змінних середовища
load_dotenv("dotenv.env")
MONDAY_API_KEY = os.getenv("MONDAY_API_KEY")

# Заголовки для авторизації
HEADERS = {
    "Authorization": MONDAY_API_KEY,
    "Content-Type": "application/json",
    "API-Version": "2023-10"  # Оновлена версія API
}

def delete_task(task_id):
    if not MONDAY_API_KEY:
        speak.speak("Error: API key is missing.")
        return

    query = """
    mutation deleteTask($item_id: ID!) {
      delete_item(item_id: $item_id) {
        id
      }
    }
    """
    variables = {"item_id": str(task_id)}
```

```
            response = requests.post(
                "https://api.monday.com/v2",
                json={"query": query, "variables": variables},
                headers=HEADERS
            )

            if response.status_code == 200:
                data = response.json()
                if data.get("data") and data["data"].get("delete_item"):
                    speak.speak(f"Task {task_id} has been deleted successfully.")
                else:
                    error_message = data.get("errors", [{}])[0].get("message",
"Unknown error")
                    speak.speak(f"Failed to delete the task. Error:
{error_message}")
            else:
                speak.speak(f"Error: {response.status_code}, Response:
{response.text}")
```

Лістинг функції, яка демонструє роботу асистента з користувачем

```
import speak_help as speak
from dotenv import load_dotenv
from create_task import create_task
from get_task import get_task
from delete_task import delete_task
import recognize_speech
import os

'''Цей файл створений для того, щоб зібрати всі таски, які може виконувати
асистент та сформувати їх в один файл'''

load_dotenv("dotenv.env")
MONDAY_API_KEY = os.getenv("MONDAY_API_KEY")
BOARD_ID = os.getenv("BOARD_ID")

HEADERS = {
    "Authorization": MONDAY_API_KEY,
    "Content-Type": "application/json"
}
```

```
        def assistant():
            speak.speak("Hello! I am your Monday assistant. How can I help you?")
            while True:
                speak.speak("Please, say your action (create task, get tasks,
delete task or exit from this app)")
                command = recognize_speech.recognize_speech() or input("Enter the
action: ").lower()
                if not command:
                    continue
                if "create task" in command:
                    speak.speak("What is the task name?")
                    task_name = recognize_speech.recognize_speech() or input("Enter
the name of the task: ")
                    if task_name:
                        create_task(task_name)
                elif "get task" in command:
                    tasks = get_task(BOARD_ID)  # Pass the BOARD_ID here
                    if isinstance(tasks, list):
                        for task in tasks:
                            speak.speak(f"Task ID: {task[0]}, Task Name:
{task[1]}")
                    else:
                        speak.speak(tasks)  # Speak the error message if any
                elif "delete task" in command:
                    tasks = get_task(BOARD_ID)  # Отримуємо список завдань
                    speak.speak(f"Here is your Task: ")
                    if isinstance(tasks, list):
                        for task in tasks:
                            speak.speak(f"Task ID: {task[0]}, Task Name:
{task[1]}")
                    task_id = input("Enter the Task ID to delete: ").strip()  #
Отримуємо ID першого завдання
                    speak.speak(f"Your Task ID is {task_id}")
                    delete_task(task_id)

                elif "exit" in command:
                    speak.speak("Goodbye!")
                    break
                else:
                    speak.speak("I didn't understand. Please try again.")
```

4.2 Захист розробленого програмного забезпечення

Захист системи для роботи з аудіоданими має враховувати кілька ключових аспектів, включаючи безпеку зберігання, передачі, обробки та доступу до даних. Оскільки аудіофайли можуть містити конфіденційну інформацію, необхідно застосовувати методи шифрування як під час їхньої передачі мережею, так і при зберіганні. Використання протоколу HTTPS[27] для захищеного передавання файлів та алгоритмів шифрування, таких як AES-256[28], дозволить унеможливити перехоплення та несанкціонований доступ до аудіоданих.

Для зберігання даних слід використовувати захищені сховища, такі як хмарні сервіси з підтримкою шифрування, наприклад, AWS S3 з активованою функцією server-side encryption. Якщо аудіофайли зберігаються локально, необхідно впровадити систему контролю доступу, яка обмежуватиме можливість їх перегляду або модифікації тільки для авторизованих користувачів. Використання технології цифрових підписів допоможе виявити будь-які зміни файлів і підтвердити їхню автентичність.

Обробка аудіоданих повинна здійснюватися у захищеному середовищі, яке не допускає виконання шкідливого коду або маніпуляцій із вхідними файлами. Використання контейнеризації за допомогою Docker[30] дозволяє ізолювати процеси обробки та мінімізувати ризики компрометації системи через вразливості в коді. Доступ до процесів розпізнавання або модифікації аудіо слід контролювати, обмежуючи його лише для певних користувачів або сервісів за допомогою токенів доступу або двофакторної автентифікації.

Для виявлення несанкціонованого доступу або підозрілої активності необхідно вести журнал подій, який включатиме інформацію про всі операції з аудіофайлами. Це допоможе аналізувати та ідентифікувати потенційні загрози, такі як несанкціоновані спроби завантаження або зміни файлів. Додатково варто застосовувати методи аналізу аномальної активності, які дозволяють виявляти нетипові запити до аудіосистеми або підозрілі дії користувачів.

Обмеження частоти запитів запобігає атакам типу brute-force[31, 32], які можуть бути використані для отримання доступу до захищених файлів або сервісів розпізнавання мовлення. Впровадження механізму rate limiting допоможе уникнути перевантаження системи та забезпечить захист від ботів або автоматизованих атак. Додатково можна використовувати CAPTCHA[33] для підтвердження дій користувача під час взаємодії з критично важливими функціями.

Якщо система використовує машинне навчання для аналізу аудіо, важливо захистити модель від атак, які можуть маніпулювати вхідними даними з метою отримання некоректних результатів. Використання перевірки цілісності моделей, обмеження доступу до їхнього навчання та оновлення, а також впровадження механізмів виявлення аномальних аудіовхідних даних допоможе зменшити ризики експлуатації вразливостей.

Захист системи для роботи з аудіоданими має бути багаторівневим і передбачати комплексний підхід, що охоплює всі етапи взаємодії з файлами – від їхнього збереження та передачі до обробки та доступу. Регулярне оновлення безпекових механізмів, перевірка на наявність вразливостей, аудит логів і застосування сучасних методів автентифікації та шифрування забезпечать високий рівень захищеності та запобіжать несанкціонованому доступу або компрометації даних.

У своїй програмі я використала файл dotenv.env, в якому прописала важливі паролі та доступи до програми, щоб у разі чого користувач не бачив цього в основному файлі коду та не міг використати у своїх цілях. Це дозволяє уникнути ризику витоку конфіденційних даних, особливо якщо код потрібно викладати у відкритий доступ або працювати в команді. Крім цього, важливо впровадити контроль доступу до аудіоданих, обмежуючи можливість їхнього прослуховування або редагування лише для авторизованих користувачів.

Лістинг файлу dotenv.env:

```
MONDAY_API_KEY="                                                    "
BOARD_ID=""
```

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

При впровадженні системи штучного інтелекту для роботи з аудіоданими в промислову експлуатацію важливо не лише забезпечити технічну реалізацію та безпеку, а й адаптувати систему до реальних умов використання. Це передбачає створення механізму безперервного моніторингу її роботи, аналізу продуктивності та виявлення потенційних вузьких місць. Наприклад, якщо система інтегрована з Google Calendar та Monday, необхідно стежити за швидкістю обробки запитів, коректністю оновлення подій та узгодженістю даних між сервісами. У разі збоїв важливо передбачити резервні сценарії роботи, наприклад, локальне кешування даних, щоб користувач міг продовжувати роботу навіть при тимчасовій відсутності зв'язку з API сторонніх сервісів.

Одним із ключових викликів є адаптація системи до різноманітних сценаріїв використання. Розпізнавання мови має працювати ефективно незалежно від рівня шуму, особливостей вимови чи можливих акцентів користувачів. Тому важливо не лише початкове навчання моделі, але й її постійне вдосконалення на основі реальних даних. Для цього можна впровадити систему активного навчання, коли система зберігає помилкові випадки та використовує їх для покращення якості розпізнавання. Впровадження механізму користувацького зворотного зв'язку дозволить користувачам повідомляти про неправильне розпізнавання та коригувати результати в режимі реального часу.

Ще одним важливим аспектом є питання масштабованості. Якщо система використовується великою кількістю користувачів, навантаження на сервери може різко зрости, що вплине на швидкість обробки запитів. Для вирішення цієї проблеми варто застосовувати балансування навантаження та горизонтальне масштабування, що дозволить рівномірно розподіляти запити між кількома серверами або інстансами в хмарі. Використання контейнеризації, наприклад,

Docker або Kubernetes[34], дозволяє легко керувати ресурсами та розгортати систему у різних середовищах без значних витрат часу.

Впровадження системи штучного інтелекту для роботи з аудіоданими у промислову експлуатацію передбачає кілька ключових етапів, кожен із яких вимагає ретельного планування та тестування. Спочатку необхідно визначити технічні вимоги до системи, включаючи апаратне забезпечення, програмні компоненти, а також очікуване навантаження. Це дозволяє вибрати оптимальну інфраструктуру: чи буде система розгорнута на локальних серверах компанії, чи використовуватиметься хмарне рішення, наприклад, AWS, Google Cloud або Azure[35]. На цьому етапі також приймаються рішення щодо використання технологій обробки аудіо, таких як Google Speech-to-Text, Mozilla DeepSpeech або власні нейромережеві моделі для розпізнавання мови.

Наступним кроком є розробка програмного забезпечення та налаштування моделей машинного навчання. Якщо використовується власна модель, то її необхідно навчити на великому масиві аудіоданих, що включає різні акценти, рівні шуму та способи мовлення. Важливо забезпечити достатню точність розпізнавання мови, а також передбачити можливості корекції та самонавчання системи на основі взаємодії з користувачем. У моїй моделі вже налаштовано рівень шуму та спосіб мовлення, що забезпечує високу якість розпізнавання аудіо в різних умовах. Проте наразі система не сприймає українську мову через специфічні налаштування мого індивідуального інструмента розробки. Це питання потребує додаткової адаптації або навчання моделі з урахуванням українських мовних даних. Інтеграція асистента з Google Calendar та Monday вимагає налаштування API-запитів[41], які дозволять отримувати, створювати та змінювати події або завдання у відповідних сервісах.

Перед запуском у промислову експлуатацію необхідно провести ретельне тестування системи. Це включає перевірку коректності розпізнавання мовлення в різних умовах, ефективності взаємодії з календарем та системою управління завданнями, а також перевірку стійкості до можливих збоїв. Тестування повинно

здійснюватися як у лабораторних умовах, так і в реальному середовищі із залученням кінцевих користувачів. Важливо перевірити продуктивність системи при одночасному використанні великою кількістю людей, оскільки це може суттєво вплинути на швидкість обробки запитів.

Одним із ключових аспектів впровадження є безпека. Оскільки система працює з аудіоданими, які можуть містити конфіденційну інформацію, необхідно забезпечити її шифрування як у процесі передачі, так і при зберіганні.

У своїй програмі я використала файл dotenv.env, в якому прописала важливі паролі та доступи до програми, щоб у разі чого користувач не бачив цього в основному файлі коду та не міг використати у своїх цілях [39, 40].

Лістинг коду dotenv.env (асистент для роботи з Monday):

```
MONDAY_API_KEY="  "
BOARD_ID=" "
```

Крім того, слід впровадити систему автентифікації користувачів, що дозволить обмежити доступ до сервісу лише авторизованим особам. Дотримання норм GDPR[36, 37, 38] або інших регуляторних вимог також є важливим фактором при впровадженні системи на комерційному рівні.

Фінальний етап – навчання користувачів та впровадження у повсякденну діяльність. Це передбачає розробку інструкцій, проведення навчальних сесій та забезпечення технічної підтримки у разі виникнення питань або проблем. Крім того, необхідно налагодити процес зворотного зв’язку з користувачами, щоб можна було виявляти та виправляти недоліки системи в реальному часі. Постійне оновлення алгоритмів і моделі розпізнавання дозволить покращувати точність та ефективність роботи асистента, а також адаптувати його під змінювані потреби компанії.

Таким чином, впровадження системи штучного інтелекту для роботи з аудіоданими є складним, але необхідним процесом, що включає проєктування інфраструктури, розробку та навчання моделей, тестування, забезпечення безпеки

та підтримку кінцевих користувачів. Тільки за умови комплексного підходу система зможе ефективно працювати у промисловому середовищі та приносити користь для автоматизації управління завданнями та подіями.

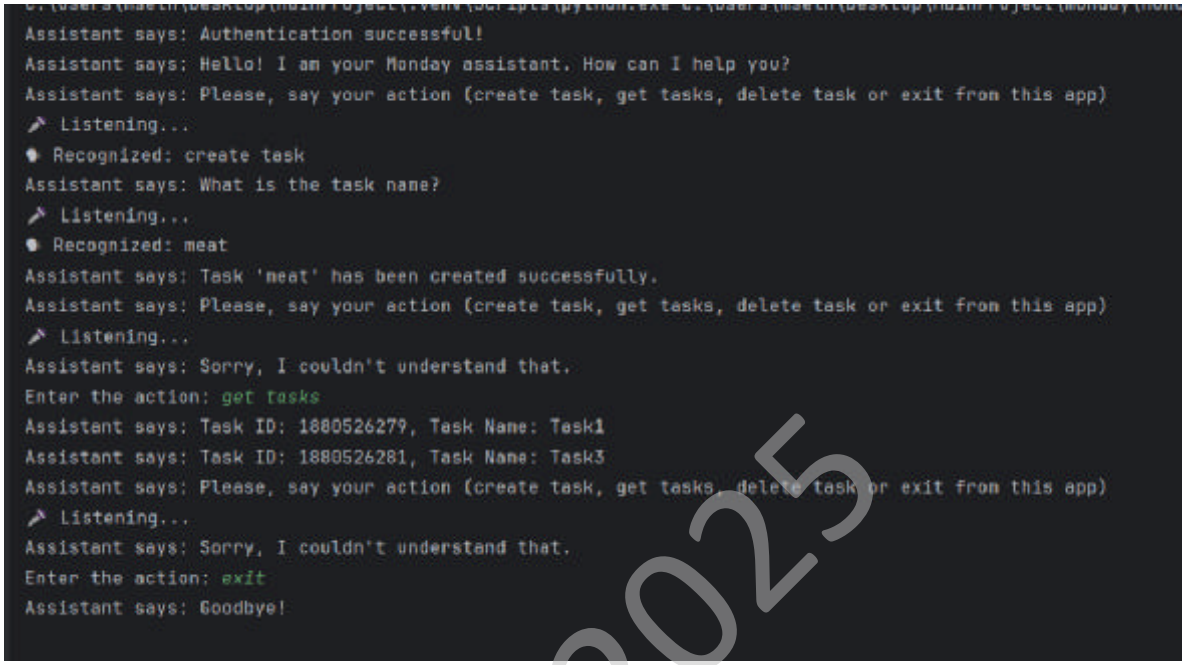

Рисунок 5.1 – Демонстрація роботи асистента Monday

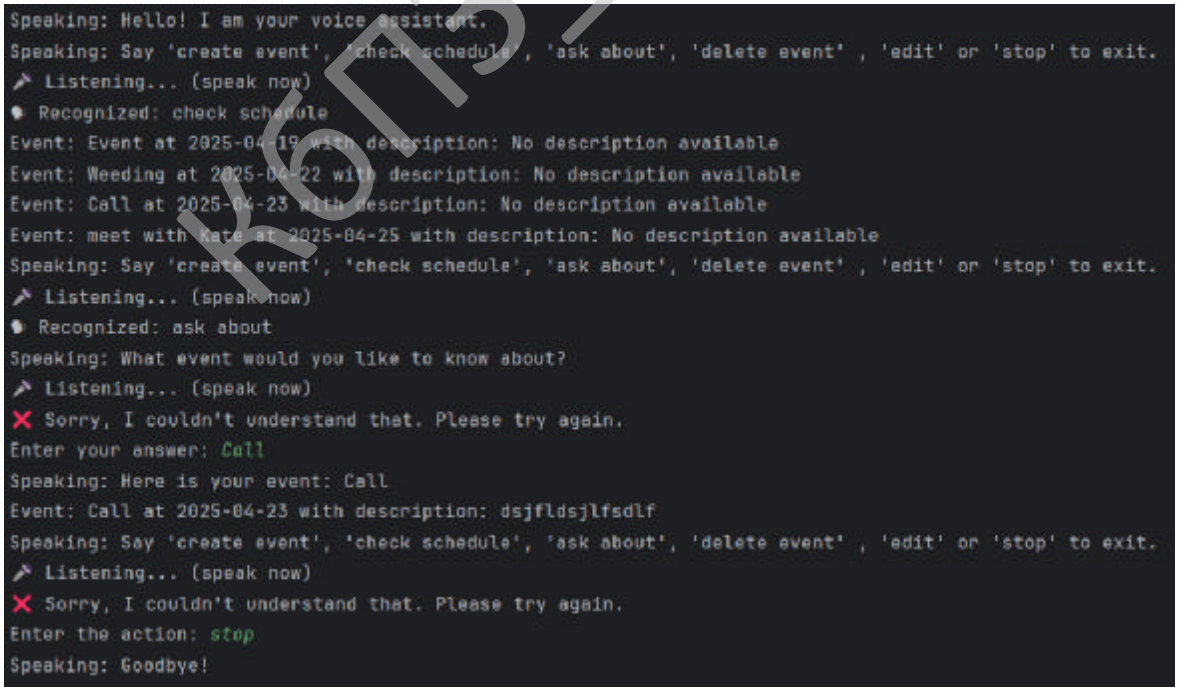

Рисунок 5.2 – Демонстрація роботи асистента Google Calendar

6 ОСНОВНІ ВИСНОВКИ

Розробка системи штучного інтелекту для роботи з аудіоданими є складним процесом, що охоплює кілька важливих етапів: від вибору технологій до їх інтеграції в робочі процеси та тестування в реальних умовах. Вона має потенціал для автоматизації великої кількості рутинних завдань, зокрема управління календарями, розподілу завдань між учасниками команди, нагадувань та організації комунікації. Впровадження голосового асистента дає можливість спростити ці процеси, забезпечивши швидкий і зручний доступ до даних за допомогою голосових команд.

У процесі розробки було розглянуто та реалізовано кілька ключових компонентів. По-перше, це модуль розпізнавання мовлення, який дозволяє переводити голосові команди у текстовий формат для подальшої обробки. Були використані сучасні технології, такі як Google Speech-to-Text та інші відкриті бібліотеки для розпізнавання мовлення. Важливим аспектом стало навчання системи сприймати різні акценти та працювати в умовах навколишнього шуму, що могло б забезпечити точність розпізнавання. Однак ця проблема залишається відкритою, оскільки сучасні алгоритми не завжди здатні правильно відфільтрувати сторонні шуми, особливо в середовищі з високим рівнем звукових перешкод.

Другим ключовим компонентом є модуль обробки запитів, який аналізує розпізнану мову та визначає, яку саме дію потрібно виконати. У системі були реалізовані сценарії для створення, отримання та видалення завдань, що інтегруються з Google Calendar та Monday. Обробка голосових команд включає аналіз контексту, щоб система могла визначати, які конкретно дії потрібно здійснити, а також можливість уточнення у разі невпевненості в запиті.

Третім важливим елементом є інтеграція з API сторонніх сервісів, що дозволяє взаємодіяти з платформами управління завданнями та календарями. У

рамках проєкту були налаштовані запити до API Monday.com[45] та Google Calendar [46] для отримання інформації, створення нових завдань, їх редагування або видалення. Однак у процесі тестування з'ясувалося, що для видалення завдань на Monday.com необхідні додаткові права доступу, що може обмежувати можливості користувача, якщо вони не були правильно налаштовані.

Безпека стала ще одним важливим питанням у розробці, оскільки система працює з конфіденційними даними, такими як особисті розклади, завдання та голосові записи користувачів. Для мінімізації ризиків було прийнято рішення зберігати ключові API-токени у файлі dotenv.env, щоб запобігти витоку інформації. Крім того, передбачено механізми шифрування даних та обмеження доступу лише для авторизованих користувачів. У майбутньому планується впровадження додаткових рівнів автентифікації, щоб підвищити рівень захисту.

Проте під час практичного тестування було виявлено кілька значних викликів, які можуть впливати на ефективність використання системи у реальному середовищі. Перш за все, це стосується швидкості виконання команд. Оскільки робота голосового асистента включає кілька послідовних етапів — розпізнавання мовлення, аналіз запиту, взаємодія з API та генерація відповіді — це створює певні затримки у роботі. У повсякденних завданнях, де потрібне миттєве реагування, такі затримки можуть уповільнювати користувача, що є критичним для сфери event-менеджменту, де необхідно швидко координувати робочі процеси.

Ще одна важлива проблема — рівень точності розпізнавання мовлення в шумному середовищі. Для event-менеджера характерною є робота на заходах, де можуть звучати гучні розмови, музика, сторонні звуки, що значно знижує ефективність голосового управління. Навіть найсучасніші алгоритми обробки аудіо не можуть повністю нівелювати такі перешкоди, що може призводити до помилкового розпізнавання команд або необхідності повторного введення інформації вручну. Це робить використання голосового асистента менш зручним у таких умовах.

З огляду на виявлені виклики, можна зробити висновок, що для підвищення ефективності системи необхідно вдосконалити її технічні характеристики. По-перше, варто інтегрувати більш потужні алгоритми шумоподавлення, щоб підвищити точність розпізнавання мовлення навіть у складних акустичних умовах. По-друге, варто оптимізувати процес обробки запитів, щоб зменшити затримку між голосовою командою та виконанням дії. По-третє, можна додати можливість гібридного управління, де користувач зможе швидко перемикатися між голосовими командами та текстовими запитами, що підвищить гнучкість у використанні.

Таким чином, розроблена система штучного інтелекту для роботи з аудіоданими є перспективним рішенням для автоматизації завдань, однак її ефективність значною мірою залежить від умов використання. У сфері event-менеджменту поточний рівень розвитку технологій розпізнавання мовлення все ще є обмеженням, через що система поки що не може повністю замінити традиційні методи керування завданнями. Проте з подальшим удосконаленням алгоритмів та оптимізацією роботи з API ця технологія може стати ефективним інструментом для організації робочих процесів та підвищення продуктивності в різних галузях. Тому в роботі з календарем та Monday я рекомендую створити чат-бот, який забезпечить швидшу обробку запитів та їх виконання без затримок, характерних для голосових асистентів. Використання текстового бота дозволить уникнути проблем із розпізнаванням мовлення в шумному середовищі та зменшить час очікування відповіді, оскільки текстові запити можна обробляти набагато швидше, ніж голосові.

Чат-бот може працювати як у месенджерах (наприклад, Telegram, Slack), так і у вигляді веб- або мобільного додатка з інтеграцією API Google Calendar та Monday. Це дозволить користувачам легко створювати, редагувати та видаляти події чи завдання через короткі текстові команди. Наприклад, користувач зможе просто написати свій запит або натиснути на необхідну кнопку, і система миттєво виконає ці дії без необхідності розпізнавання голосу.

Крім того, чат-бот може мати вбудовані кнопки для швидкого вибору дій, що ще більше спростить взаємодію з системою. Це дозволить скоротити час на введення команд та зробить управління завданнями більш зручним навіть у динамічному робочому середовищі, що особливо важливо для івент-менеджера.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Тартачний О., "ШІ-генератори голосу: що вони можуть і як працюють," Speka, 30 серпня 2024. [Онлайн]. Available: https://speka.media/si-generatori-golosu-shho-voni-mozut-i-yak-pracyuyut-v45gyk.

2. Андрєєв А., "7 найкращих генераторів голосу з підтримкою ШІ," ApiX-Drive, 26 липня 2023. [Онлайн]. Available: https://apix-drive.com/ua/blog/reviews/7-krashih-generatoriv-golosu-z-shi.

3. Сайт www.gdpr.org.ua [Електронний ресурс]. – Режим доступу до ресурсу: https://www.gdpr.org.ua/

4. Сайт www.ukraine.com.ua [Електронний ресурс]. – Режим доступу до ресурсу: https://www.ukraine.com.ua/wiki/hosting/files/ftp/connection/

5. Сайт cloud.google.com [Електронний ресурс]. – Режим доступу до ресурсу: https://cloud.google.com/speech-to-text

6. Сайт soundguys.com [Електронний ресурс]. – Режим доступу до ресурсу: https://www.soundguys.com/how-to-use-spotify-ai-dj-100465/

7. Сайт aiva.ai [Електронний ресурс]. – Режим доступу до ресурсу: https://www.aiva.ai/

8. Сайт assistant.google.com [Електронний ресурс]. – Режим доступу до ресурсу: https://assistant.google.com/

9. Сайт nuance.com [Електронний ресурс]. – Режим доступу до ресурсу: https://www.nuance.com/dragon/business-solutions/dragon-professional.html

10. Сайт pindrop.com [Електронний ресурс]. – Режим доступу до ресурсу: https://www.pindrop.com/

11. Сайт uk.wikipedia.org [Електронний ресурс]. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/PyCharm

12. Сайт foxminded.ua [Електронний ресурс]. – Режим доступу до ресурсу: https://foxminded.ua/systema-upravlinnia-bazamy-danykh/

13. Сайт.mysql.com [Електронний ресурс]. – Режим доступу до ресурсу: https://www.mysql.com/

14. Сайт thehost.ua [Електронний ресурс]. – Режим доступу до ресурсу: https://thehost.ua/ua/wiki/administration/database/postgresql-install

15. Сайт sqlite.org [Електронний ресурс]. – Режим доступу до ресурсу: https://www.sqlite.org/

16. Сайт uk.wikipedia.org [Електронний ресурс]. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/TensorFlow

17. Сайт github.com [Електронний ресурс]. – Режим доступу до ресурсу: https://github.com/pytorch/pytorch

18. Сайт scikit-learn.org [Електронний ресурс]. – Режим доступу до ресурсу: https://scikit-learn.org/stable/

19. Сайт docker.com [Електронний ресурс]. – Режим доступу до ресурсу: https://www.docker.com/

20. Сайт thehost.ua [Електронний ресурс]. – Режим доступу до ресурсу: https://thehost.ua/ua/wiki/administration/cloud-storage/connect

21. Сайт uk.wikipedia.org [Електронний ресурс]. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/SSH

22. Сайт hi-news.pp.ua [Електронний ресурс]. – Режим доступу до ресурсу: https://developers.google.com/apis-explorer

23. Сайт oauth.net [Електронний ресурс]. – Режим доступу до ресурсу: https://oauth.net/2/

24. Сайт dateutil.readthedocs.io [Електронний ресурс]. – Режим доступу до ресурсу: https://dateutil.readthedocs.io/en/stable/parser.html

25. Сайт https://api.monday.com/v2 [Електронний ресурс]. – Режим доступу до ресурсу: https://api.monday.com/v2

26. Сайт dbn.co.ua [Електронний ресурс]. – Режим доступу до ресурсу: https://dbn.co.ua/blog/strukturna_skhema/2016-12-06-11708

27. Сайт uk.wikipedia.org 333[Електронний ресурс]. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/HTTPS

28. Сайт uk.wikipedia.org [Електронний ресурс]. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Advanced_Encryption_Standard

29. Сайт docs.netapp.com [Електронний ресурс]. – Режим доступу до ресурсу:https://docs.netapp.com/us-en/storagegrid/s3/using-server-side-encryption.html

30. Сайт en.wikipedia.org [Електронний ресурс]. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Docker_(software)

31. Сайт foxminded.ua [Електронний ресурс]. – Режим доступу до ресурсу: https://foxminded.ua/brute-force/

32. Сайт uk.wikipedia.org [Електронний ресурс]. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/CAPTCHA

33. Сайт termin.in.ua [Електронний ресурс]. – Режим доступу до ресурсу: https://termin.in.ua/kapcha-captcha/

34. Сайт kubernetes.io [Електронний ресурс]. – Режим доступу до ресурсу: https://kubernetes.io/

35. Сайт uk.wikipedia.org [Електронний ресурс]. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Microsoft_Azure

36. Сайт legalaid.ua [Електронний ресурс]. – Режим доступу до ресурсу: https://legalaid.ua/ua/shho-take-gdpr/

37. Сайт www.gdpr.org.ua [Електронний ресурс]. – Режим доступу до ресурсу: https://www.gdpr.org.ua/

38. Сайт legalitgroup.com [Електронний ресурс]. – Режим доступу до ресурсу: https://legalitgroup.com/gdpr-novi-eu-tendentsii/

39. Сайт yur-gazeta.com [Електронний ресурс]. – Режим доступу до ресурсу: https://yur-gazeta.com/publications/practice/zahist-intelektualnoyi-vlasnosti-avtorske-pravo/zahist-programnogo-zabezpechennya-z-chogo-pochati-ta-yaki-isnuyut-sposobi-zahistu.html

40. Сайт www.kingston.com [Електронний ресурс]. – Режим доступу до ресурсу:https://www.kingston.com/ua/blog/data-security/what-is-data-security-software

41. Сайт corefy.com [Електронний ресурс]. – Режим доступу до ресурсу: https://corefy.com/uk/glossary/api

42. Сайт pmb.com.ua [Електронний ресурс]. – Режим доступу до ресурсу: https://pmb.com.ua/uk/slovar-terminov/diagrama-protsesu-process-diagram/

43. Сайт metinvest.digital [Електронний ресурс]. – Режим доступу до ресурсу: https://metinvest.digital/ua/page/1052

44. Сайт uk.shaip.com [Електронний ресурс]. – Режим доступу до ресурсу: https://uk.shaip.com/blog/what-is-nlp-how-it-works-benefits-challenges-examples/

45. Сайт developer.monday.com [Електронний ресурс]. – Режим доступу до ресурсу: https://developer.monday.com/api-reference/

46. Сайт cloud.google.com [Електронний ресурс]. – Режим доступу до ресурсу:https://cloud.google.com/integrationconnectors/docs/connectors/gsc_google_calendar/configure

47. Сайт www.ibm.com [Електронний ресурс]. – Режим доступу до ресурсу: https://www.ibm.com/think/topics/api-integration

48. Сайт builtin.com [Електронний ресурс]. – Режим доступу до ресурсу: https://builtin.com/artificial-intelligence/ai-assistant

49. Сайт play.ht [Електронний ресурс]. – Режим доступу до ресурсу: https://play.ht/

50. Сайт voice.ai [Електронний ресурс]. – Режим доступу до ресурсу: https://voice.ai/hub/voices/ai-character-voice-generator/

Додаток А

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи штучного інтелекту для роботи з аудіоданими.

2 Підстава для розробки

Підставою для розробки служить завдання на кваліфікаційну бакалаврську роботу №47-02 від 17.01.2025 року, видане на кафедрі кібербезпеки та програмного забезпечення.

3 Мета та призначення розробки

Метою кваліфікаційної бакалаврської роботи є розробка програмного забезпечення системи кібербезпеки для штучного інтелекту для роботи з аудіоданими.

4 Джерела розробки

Джерелом цієї кваліфікаційної бакалаврської роботи є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи штучного інтелекту з користувачем;

– розробка програми, що реалізує роботу з аудіоданими.

5.2 Показники призначення

Система повинна забезпечувати:

– взаємодію системи штучного інтелекту з користувачем;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ архітектури IBM PC, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпечення.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows 10/11.

5.8.1 Обладнання

Комп’ютер Intel Core i7/8 ГБ /1 Tb/ GeForce GT 1030 2GB або сумісні з ним.

5.8.2 Мова програмування

Програму розроблено на мові програмування Python.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД.

7 Перелік документів, що розробляються

– Структурна схема системи.
– Функціональна схема системи.
– Діаграма процесів.
– Блок-схема алгоритму роботи програми.
– Пояснювальна записка.

8 Етапи розробки

8.1 Збір і обробка інформації по темі кваліфікаційної бакалаврської роботи. Постановка задачі на виконання кваліфікаційної бакалаврської роботи (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень кваліфікаційної бакалаврської роботи.

8.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання кваліфікаційної бакалаврської роботи на попередній захист __.__.2025 р.

11.2 Подання кваліфікаційної бакалаврської роботи на захист __.__.2025 р.

Додаток Б

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи

за першим (бакалаврським) рівнем вищої освіти

_____ Є.В. Мелешко

Програмне забезпечення системи штучного інтелекту для роботи з аудіоданими

Лістинг програми

Код документу 12

Носій: CD/DVD-диск

Загальна кількість аркушів: 15

Літера: РП

Кропивницький – 2025 року

// Calendar_main.py – Головний файл програми-асистента для роботи з Google Calendar

```
import speech
import pyttsx3
from google_auth import authenticate_google_api
from speech_recogn import recognize_speech
from calendar_operations import create_event, get_week_schedule, edit_event,
ask_event
from delete_event import delete_event

'''Головний файл програми - тут викликається асистент та посилається запит на
виконання команд, які асистент має виконати.'''

# Text-to-speech settings
engine = pyttsx3.init()
engine.setProperty('rate', 150)  # Set speech rate
engine.setProperty('volume', 1)  # Set volume (from 0 to 1)

def assistant_create_event():
    service = authenticate_google_api()
    speech.speak("Hello! I am your voice assistant.")
    while True:
        speech.speak("Say 'create event', 'check schedule', 'ask about', 'delete
event' , 'edit' or 'stop' to exit.")
        user_input = recognize_speech() or input("Enter the action: ")
        if not user_input:
            continue
        if user_input.lower() in ["exit", "quit", "stop"]:
            speech.speak("Goodbye!")
            break
        if "create event" in user_input.lower():
            create_event(user_input)
        elif "check schedule" in user_input.lower():
            get_week_schedule(service)
        elif "delete event" in user_input.lower():
            speech.speak("Which event would you like to delete?")
            event_title = recognize_speech()
            delete_event(service, event_title)
        elif "edit" in user_input.lower():
            edit_event(service)
        elif "ask about" in user_input.lower():
            ask_event(service)
        else:
            speech.speak("Goodbye!")
            break


if __name__ == "__main__":
    assistant_create_event()
```

// calendar_operations.py – Розміщені головні таски для роботи з Google Calendar

```
import speech
from speech_recogn import recognize_speech
from utils import parse_date

def create_event(service, summary, description, start_time, end_time):
    event = {
        'summary': summary,
        'description': description,
        'start': {'dateTime': start_time, 'timeZone': 'Europe/Kiev'},
        'end': {'dateTime': end_time, 'timeZone': 'Europe/Kiev'},
    }
    service.events().insert(calendarId='primary', body=event).execute()
    speech.speak("Please say the event title...")
    title = recognize_speech() or input("Enter the event title: ")
    speech.speak("Please say the event description...")
    description = recognize_speech() or input("Enter the event description: ")
    start_time_str = input("Enter start time (YYYY-MM-DDTHH:MM:SS): ")
    end_time_str = input("Enter end time (YYYY-MM-DDTHH:MM:SS): ")
    start_time = parse_date(start_time_str)
    end_time = parse_date(end_time_str)
    if start_time and end_time:
        try:
            create_event(service, title, description, start_time, end_time)
            speech.speak("Event created successfully!")
        except Exception as e:
            speech.speak(f"Error creating event: {e}")
    else:
        speech.speak("Failed to create event due to invalid time input.")

def get_event_details_by_title(service, title):
    events = service.events().list(calendarId='primary', q=title,
singleEvents=True, orderBy='startTime').execute().get('items', [])
    if not events:
        print(f"No event found with the title {title}.")
        return
    for event in events:
        event_summary = event.get('summary', 'No title available')
        event_description = event.get('description', 'No description available')
        start = event.get('start', {}).get('dateTime', event.get('start',
{}).get('date', 'No start time available'))
        print(f"Event: {event_summary} at {start} with description:
{event_description}")


def get_week_schedule(service):
    import datetime
    now = datetime.datetime.utcnow()
    week_end = now + datetime.timedelta(days=7)
    events = service.events().list(
        calendarId='primary',
        timeMin=now.isoformat() + 'Z',
        timeMax=week_end.isoformat() + 'Z',
        singleEvents=True,
        orderBy='startTime'
    ).execute().get('items', [])
    if not events:
        print("No events found for this week.")
        return
    for event in events:
        event_summary = event.get('summary', 'No title available')
        event_description = event.get('description', 'No description available')
        start = event.get('start', {}).get('dateTime', event.get('start',
{}).get('date', 'No start time available'))
        print(f"Event: {event_summary} at {start} with description:
{event_description}")

def edit_event(service):
```

```
    speech.speak("Which event would you like to edit?")
    event_title = recognize_speech() or input("Enter the event title: ")
    speech.speak(f"Here is your event: {event_title}")
    get_event_details_by_title(service, event_title)  # Отримуємо деталі події
    speech.speak(f"Do you want to edit this event '{event_title}'?")
    confirmation = recognize_speech() or input("Enter 'yes' to confirm: ")
    if confirmation.lower() == "yes":
        speech.speak("Please say the event description...")
        description = recognize_speech() or input("Enter the event description:
")
        start_time_str = input("Enter start time (YYYY-MM-DDTHH:MM:SS): ")
        end_time_str = input("Enter end time (YYYY-MM-DDTHH:MM:SS): ")
        start_time = parse_date(start_time_str)
        end_time = parse_date(end_time_str)
        if start_time and end_time:
            speech.speak("Event created successfully!")
        else:
            speech.speak(f"Error editing event: {event_title}")


def delete_event(service, event_title):
    while True:
        if event_title:
            speech.speak(f"Here is your event: {event_title}")
            get_event_details_by_title(service, event_title)
            speech.speak(f"Do you want to delete the event '{event_title}'? Say
'yes' to confirm.")
            speech.wait_until_done()

            confirmation = recognize_speech()

            if confirmation and "yes" in confirmation.lower():
                delete_event(service, event_title)
                speech.speak(f"The event titled '{event_title}' has been
deleted.")
                break
            else:
                speech.speak("Event deletion cancelled.")
                break
        else:
            speech.speak("Sorry, I didn't catch the event title. Please try
again.")

def ask_event(service):
    speech.speak("What event would you like to know about?")
    event_title = recognize_speech() or input("Enter your answer: ")
    if event_title:
        speech.speak(f"Here is your event: {event_title}")
        get_event_details_by_title(service, event_title)
    else:
        speech.speak("Sorry, I didn't catch the event title. Please try again.")
```

// credentials.json – Описані важливі дані для підключення асистента до Google Calendar

{"web":{"client_id":"693439472452-46mstpfuapenf573ui46ucb68nrj4pkm.apps.googleusercontent.com","project_id":"proj-453002","auth_uri":"https://accounts.google.com/o/oauth2/auth","token_uri":"https://oauth2.googleapis.com/token","auth_provider_x509_cert_url":"https://www.googleapis.com/oauth2/v1/certs","client_secret":"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"}}

// google_auth.py – Описано аутентифікацію в Google Calendar

```
import os
import pickle
from google.auth.transport.requests import Request
from google_auth_oauthlib.flow import InstalledAppFlow
from googleapiclient.discovery import build

'''Файл створений для аутентифікації в гугл календар'''

SCOPES = ['https://www.googleapis.com/auth/calendar']

def authenticate_google_api():
    creds = None
    if os.path.exists('../token.pickle'):
        with open('../token.pickle', 'rb') as token:
            creds = pickle.load(token)
    if not creds or not creds.valid:
        if creds and creds.expired and creds.refresh_token:
            creds.refresh(Request())
        else:
            flow = InstalledAppFlow.from_client_secrets_file('credentials.json',
SCOPES)
            creds = flow.run_local_server(port=0)
        with open('../token.pickle', 'wb') as token:
            pickle.dump(creds, token)
    return build('calendar', 'v3', credentials=creds)
```

// speech.py – Описує озвучення інформації асистентом. При натисканні пробілу – програма перестає озвучувати текст тапереходить до наступної дії.

```
import threading
import keyboard
import pyttsx3

engine = pyttsx3.init()

def check_skip():
    """Перевіряє натискання пробілу та зупиняє озвучення."""
    while engine.isBusy():  # Перевіряємо, чи ще триває озвучення
        if keyboard.is_pressed("space"):  # Якщо натиснуто пробіл
            engine.stop()  # Зупиняємо озвучення
            break

def speak(text):
    """Голосове озвучування тексту з можливістю пропуску пробілом."""
    print(f"Speaking: {text}")

    # Запускаємо окремий потік для перевірки натискання пробілу (як демон)
    thread = threading.Thread(target=check_skip, daemon=True)
    thread.start()

    engine.say(text)
    engine.runAndWait()  # Запускаємо озвучення

def wait_until_done():
    """Очікує завершення озвучення перед початком розпізнавання мови."""
    while engine.isBusy():
        pass  # Чекаємо, поки engine закінчить говорити
```

// speech_recognition.py – Файл для розпізнавання голосу і виведення інформації у консольне вікно.

```
import speech_recognition as sr

"""Файл для розпізнавання голосу і виведення інформації у консольне вікно."""

def recognize_speech():
    """Розпізнає голос і повертає текст."""
    recognizer = sr.Recognizer()
    with sr.Microphone() as source:
        print("🎤 Listening... (speak now)")
        recognizer.adjust_for_ambient_noise(source)  # Автоналаштування під навколишній шум

        try:
            audio = recognizer.listen(source, timeout=10, phrase_time_limit=5)  # Збільшено timeout
            text = recognizer.recognize_google(audio)
            print(f"🗣️ Recognized: {text}")
            return text
        except sr.WaitTimeoutError:
            print("⏳ No speech detected. Please try again.")
            return None  # Повертаємо None, щоб цикл міг повторити запит
        except sr.UnknownValueError:
            print("❌ Sorry, I couldn't understand that. Please try again.")
            return None
        except sr.RequestError:
            print("🚫 API request error. Check your internet connection.")
            return None
```

Файли для асистента, який працює з Monday

// assistant.py – Описані підключення всіх завдань до одного файлу. Саме тут відбувається «збірка» всіх дій, які може робити асистент з Monday.

```
import speak_help as speak
from dotenv import load_dotenv
from create_task import create_task
from get_task import get_task
from delete_task import delete_task
import recognize_speech
import os

'''Цей файл створений для того, щоб зібрати всі таски, які може виконувати
асистент та сформувати їх в один файл'''

load_dotenv("dotenv.env")
MONDAY_API_KEY = os.getenv("MONDAY_API_KEY")
BOARD_ID = os.getenv("BOARD_ID")

HEADERS = {
    "Authorization": MONDAY_API_KEY,
    "Content-Type": "application/json"
}


def assistant():
    speak.speak("Hello! I am your Monday assistant. How can I help you?")
    while True:
        speak.speak("Please, say your action (create task, get tasks, delete
task or exit from this app)")
        command = recognize_speech.recognize_speech() or input("Enter the
action: ").lower()
        if not command:
            continue
        if "create task" in command:
            speak.speak("What is the task name?")
            task_name = recognize_speech.recognize_speech() or input("Enter the
name of the task: ")
            if task_name:
                create_task(task_name)
        elif "get task" in command:
            tasks = get_task(BOARD_ID)  # Pass the BOARD_ID here
            if isinstance(tasks, list):
                for task in tasks:
                    speak.speak(f"Task ID: {task[0]}, Task Name: {task[1]}")
            else:
                speak.speak(tasks)  # Speak the error message if any
        elif "delete task" in command:
            tasks = get_task(BOARD_ID)  # Отримуємо список завдань
            speak.speak(f"Here is your Task: ")
            if isinstance(tasks, list):
                for task in tasks:
                    speak.speak(f"Task ID: {task[0]}, Task Name: {task[1]}")
            task_id = input("Enter the Task ID to delete: ").strip()  #
Отримуємо ID першого завдання
            speak.speak(f"Your Task ID is {task_id}")
            delete_task(task_id)

        elif "exit" in command:
            speak.speak("Goodbye!")
            break
        else:
            speak.speak("I didn't understand. Please try again.")
```

// check_authentication.py – Перевірка підключення до Monday, та виведення інформації, що повертає сервер.

```
import requests
from speak_help import speak
from dotenv import load_dotenv
import os

'''Цей файл створений для того, що перевірити чи пройшов користувач
аутентифікацію'''

load_dotenv("dotenv.env")
MONDAY_API_KEY = os.getenv("MONDAY_API_KEY")
BOARD_ID = os.getenv("BOARD_ID")

HEADERS = {
    "Authorization": MONDAY_API_KEY,
    "Content-Type": "application/json"
}

def check_authentication():
    query = """{ me { name } }"""
    response = requests.post("https://api.monday.com/v2", json={"query": query},
headers=HEADERS)
    # print(response.text)  # Перевірка того, що повертає сервер
    if response.status_code == 200:
        speak("Authentication successful!")
    else:
        speak(f"Authentication failed! Error: {response.text}")
```

// create_task.py – Описано, яким чином створюються завдання для роботи з Monday.

```
from speak_help import speak
from dotenv import load_dotenv
import requests
import os

'''Файл для створення тасок в Monday'''

load_dotenv("dotenv.env")
MONDAY_API_KEY = os.getenv("MONDAY_API_KEY")
BOARD_ID = os.getenv("BOARD_ID")
HEADERS = {
    "Authorization": MONDAY_API_KEY,
    "Content-Type": "application/json"
}


def create_task(task_name):
    query = """
    mutation createTask($board_id: ID!, $item_name: String!) {
        create_item (board_id: $board_id, item_name: $item_name) {
            id
        }
    }
    """
    variables = {
        "board_id": BOARD_ID,
        "item_name": task_name
    }
    response = requests.post("https://api.monday.com/v2", json={"query": query,
"variables": variables},
                             headers=HEADERS)

    if response.status_code == 200:
        speak(f"Task '{task_name}' has been created successfully.")
    else:
        speak(f"Failed to create the task. Error: {response.text}")
```

// delete_task.py – Описано, яким чином асистент видаляє завдання, створені користувачем в Monday.

```
import requests
import os
from dotenv import load_dotenv
import speak_help as speak

'''Файл створений для видалення тасок з таск-менеджера'''

# Завантаження змінних середовища
load_dotenv("dotenv.env")
MONDAY_API_KEY = os.getenv("MONDAY_API_KEY")

# Заголовки для авторизації
HEADERS = {
    "Authorization": MONDAY_API_KEY,
    "Content-Type": "application/json",
    "API-Version": "2023-10"  # Оновлена версія API
}

def delete_task(task_id):
    if not MONDAY_API_KEY:
        speak.speak("Error: API key is missing.")
        return

    query = """
    mutation deleteTask($item_id: ID!) {
      delete_item(item_id: $item_id) {
        id
      }
    }
    """
    variables = {"item_id": str(task_id)}

    response = requests.post(
        "https://api.monday.com/v2",
        json={"query": query, "variables": variables},
        headers=HEADERS
    )

    if response.status_code == 200:
        data = response.json()
        if data.get("data") and data["data"].get("delete_item"):
            speak.speak(f"Task {task_id} has been deleted successfully.")
        else:
            error_message = data.get("errors", [{}])[0].get("message", "Unknown
error")
            speak.speak(f"Failed to delete the task. Error: {error_message}")
    else:
        speak.speak(f"Error: {response.status_code}, Response: {response.text}")
```

// get_task.py – Описано,яким чином асистент може отримати інформацію стосовно всіх завдань, створених користувачем в Monday.

```
import requests
import os
from dotenv import load_dotenv

'''Цей файл створений для того, щоб отримати всі завдання з таск менеджера'''

load_dotenv("dotenv.env")
MONDAY_API_KEY = os.getenv("MONDAY_API_KEY")
BOARD_ID = os.getenv("BOARD_ID")

HEADERS = {
    "Authorization": MONDAY_API_KEY,
    "Content-Type": "application/json"
}

def get_task(board_id):
    query = """
    query getTasks($board_id: [ID!]!) {
      boards(ids: $board_id) {
        name
        items_page {
          items {
            id
            name
          }
        }
      }
    }
    """

    variables = {"board_id": [board_id]}  # ID передаємо як список

    response = requests.post(
        "https://api.monday.com/v2",
        json={"query": query, "variables": variables},
        headers=HEADERS
    )

    data = response.json()

    if response.status_code == 200 and "data" in data:
        try:
            tasks = data["data"]["boards"][0]["items_page"]["items"]
            if tasks:
                return [(task["id"], task["name"]) for task in tasks]
            else:
                return "No tasks found."
        except (KeyError, IndexError):
            return "Unexpected response format."
    else:
        return f"Error: {data}"
```

// Monday.py – Описані підключення до таск-менеджеру Monday.

```
import os
import json
import speech_recognition as sr
import pyttsx3
import requests
import assistant
from dotenv import load_dotenv
import check_authentication

# Завантаження змінних середовища
load_dotenv("dotenv.env")
MONDAY_API_KEY = os.getenv("MONDAY_API_KEY")
BOARD_ID = os.getenv("BOARD_ID")

HEADERS = {
    "Authorization": MONDAY_API_KEY,
    "Content-Type": "application/json"
}

# Голосовий синтезатор
engine = pyttsx3.init()


check_authentication.check_authentication()

if __name__ == "__main__":
    assistant.assistant()
```

// recognize_speach.py – Описано, яким чином асистент розпізнає мову та виводить у консольне вікно.

```
from speak_help import speak
import speech_recognition as sr

'''Цей файл створений для того, щоб асистент розпізнавав, що говорить користувач
і виводив це в консольне вікно,
 якщо асистент не розпізнає мову - користувач може написати у консольному
вікні'''

def recognize_speech():
    recognizer = sr.Recognizer()
    with sr.Microphone() as source:
        print("🎤 Listening...")
        recognizer.adjust_for_ambient_noise(source)
        audio = recognizer.listen(source)
    try:
        text = recognizer.recognize_google(audio).lower()
        print(f"🗣️ Recognized: {text}")
        return text
    except sr.UnknownValueError:
        speak("Sorry, I couldn't understand that.")
        return None
    except sr.RequestError:
        speak("There was an issue with the recognition service.")
        return None
```

// speak_help.py – Описані що саме говоритть асистент, інформація описується в консолі.

```
import  pyttsx3

'''Файл створений для того, щоб в консолі було описано, що саме говоритть
асистент'''

# Голосовий синтезатор
engine = pyttsx3.init()

def speak(text):
    print(f"Assistant says: {text}")
    engine.say(text)
    engine.runAndWait()
```

// dotenv.env – Описані дані, для підключення асистента до API Monday.

```
MONDAY_API_KEY="  "
BOARD_ID=" "
```