

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет прикладної математики
Кафедра програмного забезпечення комп'ютерних систем

КУРСОВА РОБОТА

З дисципліни “Мультимедійні інтерфейси та 3D-візуалізація”

на тему:

**ПЕРСОНАЛЬНИЙ АСИСТЕНТ ДЛЯ ВЕДЕННЯ НОТАТОК ТА
ПЛАНУВАННЯ ПОДІЙ. АРХІТЕКТУРА І ЛОГІКА МОДУЛЕЙ
РОЗПІЗНАВАННЯ, ОЗВУЧЕННЯ ТА ВИКОНАННЯ КОМАНД**

Виконав студент групи КП-41мн
Пецеля Артем Володимирович

Керівник роботи: д.т.н. Сулема

Є.С. До захисту допущено

(дата, підпис)

Захищено з оцінкою

(дата, підпис)

Київ 2024

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет прикладної математики
Кафедра програмного забезпечення комп’ютерних систем

“ЗАТВЕРДЖЕНО”

Керівник роботи

_____ Є.С. Сулема

“ ____ ” _____ 2024 р.

ПЕРСОНАЛЬНИЙ АСИСТЕНТ ДЛЯ ВЕДЕННЯ НОТАТОК ТА
ПЛАНУВАННЯ ПОДІЙ

Технічне завдання

ПЗКС.045440-02-91

Виконавці:

Беліцький О. С.

Пецеля А. В.

Потапчук А. А.

Слободзян М. В.

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ	3
2. ПІДСТАВА ДЛЯ РОЗРОБЛЕННЯ	3
3. ПРИЗНАЧЕННЯ РОЗРОБКИ	3
4. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ	4
5. ВИМОГИ ДО ПРОЄКТНОЇ ДОКУМЕНТАЦІЇ	7
6. ЕТАПИ ПРОЄКТУВАННЯ	8

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Персональний асистент для ведення нотаток та планування подій.

Галузь застосування: мультимедійні технології.

2. ПІДСТАВА ДЛЯ РОЗРОБЛЕННЯ

Підставою для розроблення є завдання на курсовий проєкт з дисципліни “Мультимедійні інтерфейси та 3D-візуалізація”.

3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Персональний асистент для ведення нотаток та планування подій призначений для автоматизації процесів управління часом та інформацією користувача. Він забезпечує ефективну організацію щоденних завдань, подій та нотаток, інтегруючись з сучасними інструментами календаря та пропонуючи інтуїтивний інтерфейс з підтримкою голосових команд. Система спрямована на полегшення управління щоденними задачами для індивідуальних користувачів, які потребують швидкого доступу до планування подій, нагадувань, а також керування своїми нотатками. Голосове керування дозволяє користувачу взаємодіяти з календарем і нотатками без використання клавіатури чи миші, що робить застосунок особливо корисним для людей з обмеженими можливостями, а також для користувачів, які шукають більш зручний спосіб управління інформацією. Наявність мультимедійного помічника допоможе спростити та зробити процес взаємодії з системою більш інтуїтивним та доступним. Інтерактивний 3D-аватар забезпечить користувачу візуальні підказки під час голосових команд, надаючи відчуття реального спілкування з асистентом.

4. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

4.1. Функціональні вимоги до програмного забезпечення

Інтерактивна система для ведення нотаток та планування подій повинна містити такі основні функції:

- 1) забезпечувати реєстрацію обов'язкового персонального облікового запису користувача для доступу до функціональності системи;
- 2) інформувати користувача про наявні команди при вході в систему;
- 3) забезпечувати можливість виконання наступних команд для управління нотатками:

а) Створення нотатки:

- команда: "Make a note";
- питання для уточнення: "How note should be named?", "Please dictate the note";
- результат: створена нотатка з вказаним ім'ям та вмістом.

б) Перегляд нотаток:

- команда: "What notes do I have?";
- питання для уточнення: "Do you want to hear more?"
- результат: анімований персонаж озвучує п'ять останніх назв нотаток та питає у користувача чи потрібно озвучити більше. У випадку ствердної відповіді анімований персонаж озвучує наступні п'ять імен нотаток.

с) Відтворення нотаток:

- команда: "Read the note";
- питання для уточнення: "What's the name of the note you'd like to hear";
- результат: анімований персонаж озвучує нотатку за вказаною назвою.

- 4) взаємодіяти з електронним календарем користувача для виконання наступних команд для управління подіями:

a) Створення події:

- команда: “Create an event”;
- питання для уточнення: “What is the name of the event?”, “When does it start?”, “When does it end?”;
- результат: створена подія з вказаною назвою, часом початку та кінця.

b) Розклад:

- команди: “What is planned for today?”, “What is planned for this week?”, “What is planned for the next week?”;
- результат: анімований персонаж озвучує користувачу список подій запланованих на сьогодні, цей тиждень або наступний тиждень.

c) Видалення події:

- команда: “Remove the event”;
- питання для уточнення: “What is the event name to remove?”;
- результат: вказана подія видалена або користувачу повідомляється про те, що такої події не існує.

- 5) надавати можливість вводу команд за допомогою голосу або вибору команди у відповідному меню;
- 6) надавати відповідь користувачу за допомогою відтворення згенерованого аудіо та дублювати відповідь у вигляді субтитрів;
- 7) відображати тривимірну анімовану модель під час використання застосунку;
- 8) надавати можливість перегляду записів про нотатки та події користувача із застосуванням інструментів пагінації, сортування та пошуку;
- 9) реагувати на неіснуючі голосові команди та незрозуміле мовлення користувача.

4.2. Нефункціональні вимоги до програмного забезпечення

Інтерактивна система для ведення нотаток та планування подій повинна забезпечувати такі нефункціональні можливості:

- 1) бути кросплатформною:
 - a) вебзастосунок повинен працювати на більшості сучасних браузерів (Chrome, Mozilla, Edge);
 - b) сервер повинен запускатися на різних ОС (Linux, Windows);
- 2) витримувати навантаження в 100 одночасних користувачів та 50 тисяч активних користувачів в місяць;
- 3) дані для автентифікації мають бути захищені, паролі не повинні зберігатися у відкритому вигляді;
- 4) мова інтерфейсу користувача – англійська.

4.3. Функціональні вимоги до моделі та анімації персонажу асистента

Функціональні вимоги моделі та анімації персонажа:

- 1) модель повинна бути багатополігональною, із накладеними матеріалами;
- 2) модель повинна бути виконана у вигляді футуристичного робота, наприклад, EVE із мультфільму “WALL-E”;
- 3) створений скелет моделі;
- 4) модель повинна виражати різні емоції та стан за допомогою анімації й зміни обличчя;
- 5) для кожної визначеної дії персонажа повинні бути реалізовані відповідні анімації:
 - "Привітання" - анімація, де модель махає рукою;
 - "Слухає" - модель робить невеликі рухи, на обличчі з'являється анімація прослуховування у вигляді голосової доріжки.
 - "Думає" - невеликі рухи вгору-вниз або з боку в бік, як ніби робот розмірковує, та вираз обличчя, що зображає завантаження.

- "Відповідає" - більш активні рухи тіла, а також щасливий вираз обличчя.
 - "Не зрозумів команду" - стримана анімація з жестом збентеження та виразом обличчя "помилка".
 - "Прощання" - анімація з прощальним жестом руки або нахилом корпусу.
- 6) персонаж не повинен бути повністю статичним, навіть коли не взаємодіє з користувачем. Необхідно створити просту анімацію, наприклад, невеликі випадкові погойдування або мимовільні рухи антен або інших дрібних деталей;
 - 7) вирази обличчя повинні точно відповідати стану помічника (радість, сум, збентеження);
 - 8) персонаж із анімаціями повинні бути експортовані у форматі glTF.

4.4. Нефункціональні вимоги до моделі та анімації персонажу асистента

Нефункціональні вимоги до моделі та анімації персонажа:

- 1) емоції мають бути чіткими та виразними, щоб користувачі могли легко їх розпізнати.
- 2) модель і анімації повинні бути плавними, без затримок. Потрібно уникати різких, незграбних рухів.
- 3) якщо помічник розмовляє або дає відповідь, анімації повинні бути синхронізовані з голосом.

5. ВИМОГИ ДО ПРОЄКТНОЇ ДОКУМЕНТАЦІЇ

У процесі виконання проєкту повинна бути розроблена наступна документація:

- 1) 4 пояснювальні записки;
- 2) керівництво користувача.

6. ЕТАПИ ПРОЄКТУВАННЯ

Аналіз вимог до програмної системи	23.09.2024
Розроблення та узгодження технічного завдання	07.10.2024
Розроблення архітектури системи	21.10.2024
Розроблення основної логіки системи	01.11.2024
Розроблення компонент голосового інтерфейсу	12.11.2024
Розроблення моделей та анімацій	15.11.2024
Інтеграція компонент програмного продукту	23.11.2024
Тестування системи	03.12.2024
Оформлення технічної документації проєкту	09.12.2024

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет прикладної математики
Кафедра програмного забезпечення комп’ютерних систем

“ЗАТВЕРДЖЕНО”

Керівник роботи

_____ Євгенія Сулема

“ ____ ” _____ 2024 р.

**ПЕРСОНАЛЬНИЙ АСИСТЕНТ ДЛЯ ВЕДЕННЯ НОТАТОК ТА
ПЛАНУВАННЯ ПОДІЙ. АРХІТЕКТУРА І ЛОГІКА МОДУЛЕЙ
РОЗПІЗНАВАННЯ, ОЗВУЧЕННЯ ТА ВИКОНАННЯ КОМАНД**

Пояснювальна записка

ПЗКС.045440-03-81

Виконавець:

Пецеля А. В.

2024

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ.....	3
ВСТУП.....	4
1. АНАЛІЗ ІНТСРУМЕНТІВ ПРОГРАМУВАННЯ ТА ТЕХНОЛОГІЙ РОЗРОБЛЕННЯ СЕРВЕРНОЇ ЧАСТИНИ.....	5
1.1. Мова програмування Python.....	5
1.2. Аналіз веб-фреймворку Flask.....	5
1.3. Аналіз використаних бібліотек.....	6
1.3.1. Бібліотека openai-whisper.....	6
1.3.2. Бібліотека gTTS.....	6
1.3.3. Бібліотека Levenshtein.....	7
1.3.4. Бібліотека dateparser.....	7
2. РОЗРОБЛЕННЯ АРХІТЕКТУРИ І ЛОГІКИ ПРОГРАМИ.....	8
2.1. Аналіз та опис команд та діалогів.....	8
2.2. Архітектура системи.....	10
2.3. Структура серверної частини.....	10
3. АНАЛІЗ РОЗРОБЛЕНИХ МОДУЛІВ.....	11
3.1. Особливості реалізації модулю TTS.....	11
3.2. Особливості реалізації модулю розпізнавання.....	11
3.3. Особливості реалізації модулю нотаток.....	14
3.4. Особливості реалізації модулю подій.....	15
3.5. Тестування розроблених компонентів.....	17
3.6. Рекомендації щодо подальшого вдосконалення.....	17
ВИСНОВКИ.....	18
СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ.....	19

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

API – application programming interface (прикладний програмний інтерфейс)

TTS – text-to-speech

gTTS – Google TTS

Відстань Левенштейна – метрика , що вимірює по модулю різниця між двома послідовностями символів.

ВСТУП

На сьогоднішній день програмне забезпечення відіграє важливу роль у щоденному житті людини, забезпечуючи автоматизацію задач та підвищення продуктивності в різних сферах діяльності. Однією з ключових тенденцій сучасного програмування є створення застосунків, орієнтованих на зручність кінцевого користувача. Очевидно, що успіх програмного продукту залежить не лише від його функціональності, а й від інтуїтивності інтерфейсу та мінімальної кількості взаємодій, необхідних для досягнення потрібного результату. У зв'язку з цим особливої популярності набувають персональні асистенти, які дозволяють швидко вирішувати повсякденні завдання через єдину «точку входу», забезпечуючи користувачам максимальну ефективність та комфорт.

Особливої уваги заслуговують асистенти, призначені для ведення нотаток та планування подій, оскільки організація інформації та часу є однією з основних потреб сучасної людини. Завдяки таким асистентам користувачі можуть структурувати власні думки, керувати завданнями та планувати зустрічі чи події, використовуючи зручний і зрозумілий інтерфейс. Якщо ж цей асистент поєднує текстові, голосові та візуальні можливості, він стає незамінним інструментом для підвищення особистої продуктивності.

Цей проєкт присвячений розробленню архітектури та логіки програмної системи, що реалізує функціональність персонального асистента для ведення нотаток та планування подій. Основна мета полягає у створенні продукту, який спрощуватиме організацію завдань і часу користувачів, а також демонструватиме переваги сучасних мультимедійних технологій для підвищення ефективності та зручності взаємодії.

1. АНАЛІЗ ІНСТРУМЕНТІВ ПРОГРАМУВАННЯ ТА ТЕХНОЛОГІЙ РОЗРОБЛЕННЯ СЕРВЕРНОЇ ЧАСТИНИ

1.1. Мова програмування Python

Python – це високорівнева мова програмування загального призначення, створена Гвідо ван Россумом у 1991 році.

Python є інтерпретованою мовою, що не потребує попередньої компіляції, а його динамічна типізація дозволяє автоматично визначати тип змінної під час виконання програми. Лаконічний і зрозумілий синтаксис робить код легким для написання та читання. Мова підтримує кілька парадигм програмування: об'єктно-орієнтоване, процедурне та функціональне програмування. Крім того, Python має автоматичне керування пам'яттю завдяки вбудованому збирачу сміття. Його велика стандартна бібліотека включає інструменти для роботи з файлами, мережею, обчисленнями та іншими завданнями. Python є кросплатформеним і працює на всіх популярних операційних системах.

Серед переваг мови можна виділити:

- простий синтаксис та легкість у вивченні;
- широке застосування: веб-розробка, автоматизація, наука про дані, AI тощо;
- велика спільнота та багато готових бібліотек;
- гнучка інтеграція з іншими мовами;
- підходить для швидкої розробки прототипів та складних систем.

1.2. Аналіз веб-фреймворку Flask

Flask — це легковаговий веб-фреймворк для мови програмування Python. Побудований на основі стандарту WSGI (Web Server Gateway Interface) та системи шаблонізації Jinja2, Flask надає мінімальний набір функціональних можливостей для створення веб-застосунків. Завдяки своїй гнучкості та відсутності нав'язливих структур, фреймворк дозволяє розробникам самостійно обирати необхідні компоненти для побудови

застосунку, що робить його особливо зручним для швидкої розробки як невеликих проєктів, так і масштабованих систем.

Переваги Flask:

- мінімалістичність та висока гнучкість у побудові архітектури;
- можливість легкого розширення за допомогою сторонніх бібліотек та модулів;
- вбудовані інструменти для тестування та налагодження застосунків;
- зручна інтеграція з базами даних та іншими технологіями;
- оптимальний вибір для створення RESTful API та мікросервісних архітектур.

1.3. Аналіз використаних бібліотек

1.3.1. Бібліотека openai-whisper

OpenAI Whisper — це нейромережева бібліотека для автоматичного розпізнавання мовлення (ASR — Automatic Speech Recognition), розроблена компанією OpenAI. Whisper є потужною системою, що здатна конвертувати аудіо в текст з високою точністю та підтримує понад 90 мов. Бібліотека базується на сучасних архітектурах трансформерів та навчена на великому обсязі багатомовних даних, що дозволяє їй ефективно обробляти як чітке мовлення, так і низькоякісні записи з шумом чи акцентами.

Whisper не лише виконує пряме перетворення аудіо в текст, а й здатна вирішувати додаткові завдання, як-от розпізнавання мов в аудіофайлі, транскрибування та переклад мовлення на англійську мову.

1.3.2. Бібліотека gTTS

Бібліотека gTTS — це бібліотека для мови Python для перетворення тексту в мовлення. Бібліотека сама не виконує власне перетворення, а використовує API Google Translate сервісу для озвучення тексту. Вона має гнучкі налаштування швидкості читання тексту, мови та акценту.

1.3.3. Бібліотека Levenshtein

Бібліотека Levenshtein – це бібліотека для мови Python, яка надає можливість оцінити відстань між двома словами. Бібліотека використовує відстань Левенштейна для оцінки різниці між словами.

1.3.4. Бібліотека dateparser

Бібліотека dateparser – це бібліотека для мови Python, яка надає можливість перетворити розмовний варіант дати та часу в зрозумілий для програми тип даних datetime.

2. РОЗРОБЛЕННЯ АРХІТЕКТУРИ І ЛОГІКИ ПРОГРАМИ

2.1. Аналіз та опис команд та діалогів

Згідно розробленого технічного завдання було визначено список команд, які користувач може використовувати для управління асистентом:

- `create_note` – створити нотатку;
- `list_notes` – отримати список всіх нотаток;
- `get_note` – зачитати певну нотатку;
- `create_event` – створити подію;
- `list_events_today` – отримати список подій на сьогодні;
- `list_events_this_week` – отримати список подій на поточний тиждень;
- `list_events_next_week` – отримати список подій на наступний тиждень;
- `remove_event` – видалити подію.

Серед цих команд є такі, які потребують від користувача додаткової інформації. Для збору додаткової інформації асистенту необхідно ставити питання користувачу. Такі питання мають на меті отримати відповідь у формі голосу та, згодом, бути перетвореними на потрібний тип даних (`text`, `datetime`, `boolean`). Таким чином між асистентом та користувачем виникає діалог, в результаті якого асистент виконує поставлену задачу.

В табл. 1 описано всі можливі варіанти діалогів між користувачем та асистентом. Колонка “Тригер” вказує на фразу-тригер запуску команди, “Питання” на питання, які асистент буде задавати, “Тип даних” на очікуваний тип даних після обробки відповіді користувача, “Результат” на результат виконання команди.

Таблиця 1

Тригер	Питання	Тип даних	Результат
Команда: <code>create_note</code>			
Make a note	How note should be named?	text	Створена нотатка з вказаною назвою та змістом.
	Please dictate the note	text	

Команда: list_notes			
What notes do I have?	Do you want to hear more?	boolean	Представлений список 5 перших нотаток користувача, та на вимогу демонструється ще 5 (циклічно).
Команда: get_note			
Read the note	What's the name of the note you'd like to hear?	text	Знайдена нотатка озвучується.
Команда: create_event			
Create an event	What is the name of the event?	text	Створена подія з вказаною назвою та часом початку і кінця.
	When does it start?	datetime	
	When does it end?	datetime	
Команда: list_events_today			
What is planned for today?	—	—	Представлений список подій на сьогодні.
Команда: list_events_this_week			
What is planned for this week?	—	—	Представлений список подій на поточний тиждень.
Команда: list_events_next_week			
What is planned for next week?	—	—	Представлений список подій на наступний тиждень.
Команда: remove_event			
Remove the event	What is the event name to remove?	text	Вказана подія видалена.

Також було розроблено список можливих помилок, які можуть траплятися у ході діалогу. Серед таких можуть бути: не розпізнана команда, не розпізнаний час та дата, не знайдені нотатка або подія. Такі сценарії також враховані та оброблені.

2.2. Архітектура системи

Описаний застосунок є системою з клієнт-серверною архітектурою. Він складається з клієнтської частини та серверної частини.

Клієнтська частина відповідальна за наочну ілюстрацію асистента у вигляді аватару, виведення аудіо, запис мовлення з мікрофону. Також клієнтська частина відображає процес виконання команд.

Серверна частина відповідальна за розпізнавання команд, відповідей на питання та виконання команд. Також до задач серверної частини належить автентифікація/авторизація користувачів, управління нотатками та подіями, інтеграція з Google Calendar, розпізнавання мовлення.

2.3. Структура серверної частини

Серверна частина складається з таких модулів:

- модуль TTS,
- модуль розпізнавання,
- модуль автентифікації/авторизації,
- модуль нотаток,
- модуль подій.

3. АНАЛІЗ РОЗРОБЛЕНИХ МОДУЛІВ.

3.1. Особливості реалізації модулю TTS

Модуль TTS був розроблений для забезпечення озвучування асистента. Цей модуль використовує бібліотку gTTS для перетворення тексту в аудіо дані. Ці аудіо дані є масивом байтів, який перетворюється в текст за допомогою base64 кодування. В такому вигляді аудіо дані передаються на клієнтську частину. Нижче наведений лістнінг коду цього модуля:

```
import base64
from typing import Generator

from gtts import gTTS

def text_to_speech(text: str) -> Generator[bytes, any, any]:
    return gTTS(text=text, lang='en', slow=True).stream()

def text_to_base64_speech(text: str) -> str:
    combined_bytes = b''.join(text_to_speech(text))
    return base64.b64encode(combined_bytes).decode()
```

3.2. Особливості реалізації модулю розпізнавання

Модуль розпізнавання включає в себе задачі розпізнавання команд, тексту, дати і часу та булевого типу даних. Цей модуль використовує бібліотеку openai-whisper для розпізнавання мовлення та перетворення його в текст. Нижче наведений лістнінг коду розпізнавання мовлення:

```
import whisper
from decouple import config

model_size = config("SPEECH_RECOGNITION_MODEL_SIZE")

model = whisper.load_model(f"{model_size}.en")

def recognize_speech(speech_file: str):
    return whisper.transcribe(model, speech_file)["text"]
```

Оскільки функція розпізнавання мовлення не є детерміністичною, а також мовлення людини може відрізнятися, то при операціях пошуку необхідно враховувати різницю між щойно сказаним та колись сказаним

(збереженим). Була використана відстань Левенштейна для оцінки різниці між словами. Нижче наведений лістнінг з кодом, який знаходить серед масиву тексту той варіант, який найближчий до шуканого:

```
from Levenshtein import distance

def find_nearest(value: str, array: list[str]) -> (str, int):
    distances = list(map(lambda x: distance(x, value), array))
    min_distance = min(distances)
    return array[distances.index(min_distance)], min_distance
```

Задача розпізнавання команд зводиться до задачі пошуку серед можливих фраз-тригерів найближчої. При цьому враховується певний поріг, коли вважається, що команда не є розпізнаною. Нижче наведено лістнінг коду розпізнавання команд:

```
def recognize_command():
    file = request.files.getlist('audio')
    temp_file_path = f"./file_data/{uuid4()}"
    save_file(file[0], temp_file_path)
    recognized_speech = recognize_speech(temp_file_path)
    os.remove(temp_file_path)
    try:
        result = define_nearest_command(recognized_speech)
        if result:
            return jsonify({
                "command": result[0].value,
                "scenario": result[1].serialize()
            })
        else:
            return jsonify('Not found'), 400
    except Exception as e:
        print(e)
        phrase = "I can't recognize command, please try again"
        return jsonify({
            "b64_phrase": text_to_base64_speech(phrase),
            "phrase": phrase
        }), 400

def save_file(file: FileStorage, path: str):
    with open(path, 'wb') as f:
        file.save(f)

def define_nearest_command(input_command: str) -> (cmds.Command, cmds.Scenario):
    commands = list(map(lambda x: x.trigger.lower(), cmds.scenarios.values()))
    input_command = input_command.lower().strip()
    nearest_trigger, distance = find_nearest(input_command, commands)
    print(f"Recognized input: {input_command}\n",
          f"Nearest command: {nearest_trigger}\n",
          f"Distance: {distance}")
    if distance > 5:
```

```

        raise Exception("Could not recognize command")

    for command, scenario in cmds.scenarios.items():
        if scenario.trigger.lower() == nearest_trigger.lower():
            return command, scenario

    raise Exception("Could not recognize command")

```

Задача розпізнавання тексту зводиться до простого перетворення мовлення в текст.

Задача розпізнавання дати полягає в розпізнаванні мовлення та перетворення текстового типу даних в тип даних `datetime`. Можливість такого перетворення надає бібліотека `dateparser`. Нижче наведено лістинг коду з розпізнавання дати і часу:

```

@recognizer.route('/recognize_date', methods=['POST'])
def recognize_date():
    file = request.files.getlist('audio')
    temp_file_path = f"./file_data/{uuid4()}"
    save_file(file[0], temp_file_path)
    recognized_speech = recognize_speech(temp_file_path)
    os.remove(temp_file_path)

    date = dateparser.parse(recognized_speech, settings={'TIMEZONE':
'Europe/Kyiv'})
    print(f"Recognized speech: {recognized_speech}\nParsed date: {date}")
    if date is None:
        phrase = "I can't understand what you said, please repeat"
        return jsonify({
            "b64_phrase": text_to_base64_speech(phrase),
            "phrase": phrase
        }), 400
    return jsonify({"result": date.isoformat()})

```

Задача розпізнавання булевого типу даних полягає в пошуку серед варіантів “yes” чи “no” найближчого до сказаного. При цьому задається поріг відстані між сказаним і опціями в 2 літери. Нижче наведений лістинг:

```

@recognizer.route('/recognize_boolean', methods=['POST'])
def recognize_boolean():
    file = request.files.getlist('audio')
    temp_file_path = f"./file_data/{uuid4()}"
    save_file(file[0], temp_file_path)
    recognized_speech = recognize_speech(temp_file_path)
    os.remove(temp_file_path)

    options = ['yes', 'no']
    nearest_option, distance = find_nearest(recognized_speech.strip().lower(),
options)

```

```

if distance > 2:
    phrase = "I can't understand what you said, please repeat"
    return jsonify({
        "b64_phrase": text_to_base64_speech(phrase),
        "phrase": phrase
    }), 400

if nearest_option == 'yes':
    return jsonify({"result": True})
else:
    return jsonify({"result": False})

```

3.3. Особливості реалізації модулю нотаток

Модуль нотаток відповідальний за виконання команд пов'язаних з нотатками. Серед них: `create_note`, `list_notes`, `get_note`. На кожну команду створений окремий endpoint. Також цей модуль відповідальний за обробку помилок при виконання цих команд, наприклад коли не існує запитаної нотатки. Нижче наведений лістнінг коду цього модуля:

```

@notes.route('/notes', methods=['GET'])
@login_verify_required
def get_notes():
    page = request.args.get('page', 1, type=int)
    limit = request.args.get('limit', 5, type=int)

    notes_page = get_notes_page(current_user.id, page, limit)

    if len(notes_page) == 0:
        phrase = "No notes found"
        return jsonify({
            "phrase": phrase,
            "b64_phrase": text_to_base64_speech(phrase),
            "body": []
        })

    phrase = "Here are some notes that I found"
    return jsonify({
        "phrase": phrase,
        "b64_phrase": text_to_base64_speech(phrase),
        "body": [note.to_dict() for note in notes_page]
    })

@notes.route('/notes', methods=['POST'])
@login_verify_required
def create_note():
    name = request.json['name']
    content = request.json['content']

    insert_note(current_user.id, name, content)

    phrase = f"Note {name} was created"

```

```

    return jsonify({
        "phrase": phrase,
        "b64_phrase": text_to_base64_speech(phrase),
    }), 201

@notes.route('/notes/<name>', methods=['GET'])
@login_verify_required
def get_note(name):
    all_notes = get_user_notes(current_user.id)

    nearest_note_name, distance = find_nearest(name, [note.name for note in
all_notes])

    if distance > 5:
        phrase = "I can't find specified note, please try again"
        return jsonify({
            "b64_phrase": text_to_base64_speech(phrase),
            "phrase": phrase
        }), 404

    nearest_note = next(note for note in all_notes if note.name ==
nearest_note_name)
    phrase = f"Here is your note... {nearest_note.content}"
    return jsonify({
        "b64_phrase": text_to_base64_speech(phrase),
        "phrase": phrase,
        "body": {
            "name": nearest_note_name,
            "content": nearest_note.content
        },
    })

```

3.4. Особливості реалізації модулю подій

Модуль подій відповідальний за виконання команд пов'язаних з подіями. Серед таких: `create_event`, `list_events_today`, `list_events_this_week`, `list_events_next_week`, `remove_event`. Також цей модуль відповідальний за обробку помилок при виконання цих команд, наприклад коли при видаленні події користувач вказує на не існуючу подію. Нижче наведений лістніг цього модулю:

```

@events.route('/events/<period>', methods=['GET'])
@calendar_track_required
@login_verify_required
def get_events(period: str):
    available_periods = ['today', 'this_week', 'next_week']

    if period not in available_periods:
        return 'Bad period', 400

```



```

events_calendar = list_events_period(current_user.calendar_id, period)
if len(events_calendar) == 0:
    phrase = f"There are no events for {" ".join(period.split('_'))}"
    return jsonify({
        "b64_phrase": text_to_base64_speech(phrase),
        "phrase": phrase,
        "body": []
    }), 200

    phrase = f"Here are your events for {" ".join(period.split('_'))}: {"",
".join([event.name for event in events_calendar])}."

    return jsonify({
        "b64_phrase": text_to_base64_speech(phrase),
        "phrase": phrase,
        "body": [event.to_dict() for event in events_calendar]
    }), 200

@events.route('/events', methods=['POST'])
@calendar_track_required
@login_verify_required
def create_event():
    name = request.json['name']
    start = request.json['start']
    end = request.json['end']

    try:
        start_time = datetime.fromisoformat(start)
        end_time = datetime.fromisoformat(end)

        if start_time < datetime.now(kyiv_tz) - timedelta(days=7):
            raise ValueError("Start time is too old")

        if start_time > end_time:
            raise ValueError("Start time is after end time")

        event = Event(name=name, start_time=start_time, end_time=end_time)
        create_event_calendar(current_user.calendar_id, event)

        phrase = f"Your event {name} was created."
        return jsonify({
            "b64_phrase": text_to_base64_speech(phrase),
            "phrase": phrase,
            "body": event.to_dict()
        }), 201

    except ValueError:
        return 'Invalid time format', 400
    except Exception as e:
        if str(e) == "Calendar not found":
            return 'Calendar not found', 404
        if str(e) == "Insufficient permissions":
            return 'Insufficient permissions', 403
        if str(e) == "can't compare offset-naive and offset-aware datetimes":
            return 'Invalid time format', 400
        print(e)
        return 'Failed to create event', 500

```

```

@events.route('/events/<name>', methods=['DELETE'])
@calendar_track_required
@login_verify_required
def delete_event(name: str):
    events_calendar = list_events(current_user.calendar_id)
    event_names = [event.name for event in events_calendar]
    nearest_event, distance = find_nearest(name, event_names)

    if distance > 5:
        phrase = "I can't find specified event, please try again"
        return jsonify({
            "b64_phrase": text_to_base64_speech(phrase),
            "phrase": phrase
        }), 404

    event = next(event for event in events_calendar if event.name ==
nearest_event)
    delete_event_calendar(current_user.calendar_id, event.id)

    phrase = f"Event {event.name} was deleted."
    return jsonify({
        "b64_phrase": text_to_base64_speech(phrase),
        "phrase": phrase,
    }), 200

```

3.5. Тестування розроблених компонентів

Тестування системи відбувалося мануально, оскільки не є можливим провести тестування автоматично. Система не є детерміністичною, спирається на людське мовлення, тому було вирішено прослухати тестування мануально.

Для тестування було створено тестову клієнтську частину, де було реалізовано запис з мікрофону та відтворення аудіо. Кожна з задач розпізнавання, а також усі команди були відтестовані на правильність виконання сценарію і на правильність обробки помилок.

3.6. Рекомендації щодо подальшого вдосконалення

Систему можна покращити імплементувавши більшу частину команд, таких як видалення нотатки, перегляд конкретної події. Також можливе розширення списку фраз-тригерів таким чином, щоб одну команду можна було викликати декількома фразами.

ВИСНОВКИ

Метою цієї курсової роботи було розроблення персонального асистенту з ведення нотаток та планування подій з використанням мультимедійного помічника та голосового асистента. Зокрема в цій частині було розглянуто розроблення архітектури і логіки модулів розпізнавання, озвучення та виконання команд.

Аналіз засобів розроблення мультимедійних помічників, показав доцільність створення веб-серверів на мові програмування TypeScript та Python та застосування відкритих API для задач озвучення та TTS.

Розроблений персональний помічник може:

- створювати нотатки;
- використовувати голосовий інтерфейс для пошуку нотаток;
- озвучувати потрібну нотатку;
- створювати подію;
- використовувати голосовий інтерфейс для пошуку подій на різні періоди;
- видаляти подію.

Розробка виконана у повному обсязі, всі вимоги, викладені у технічному завданні враховані, тестування продукту виконано.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Python: The most popular language [Електронний ресурс]. – Режим доступу: <https://datascientest.com/en/python-the-most-popular-language> – (9.12.2024).
2. Welcome to Flask [Електронний ресурс]. – Режим доступу: <https://flask.palletsprojects.com/en/stable/> – (9.12.2024).
3. Introduction to dateparser [Електронний ресурс]. – Режим доступу: <https://dateparser.readthedocs.io/en/latest/> – (9.12.2024).
4. OpenAI/whisper [Електронний ресурс]. – Режим доступу: <https://github.com/openai/whisper> – (9.12.2024).
5. GTTS Documentation [Електронний ресурс]. – Режим доступу: <https://gtts.readthedocs.io/en/latest/> – (9.12.2024).

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет прикладної математики

“ЗАТВЕРДЖЕНО”

Керівник роботи

_____ Є.С.Сулема

“ ____ ” _____ 2024 р.

ПЕРСОНАЛЬНИЙ АСИСТЕНТ ДЛЯ ВЕДЕННЯ НОТАТОК ТА
ПЛАНУВАННЯ ПОДІЙ
Керівництво користувача
ПЗКС.045440-05-34

Виконавці:

_____ Потапчук А.А.

_____ Беліцький О.С.

_____ Слободзян М.В.

_____ Пецеля А.В.

ЗМІСТ

1. Опис структури системи.....	3
2. Опис реєстрації та авторизації.....	3
3. Опис взаємодії з нотатками.....	5
4. Опис взаємодії з подіями.....	8

1. Опис структури системи

Система складається з наступних сторінок:

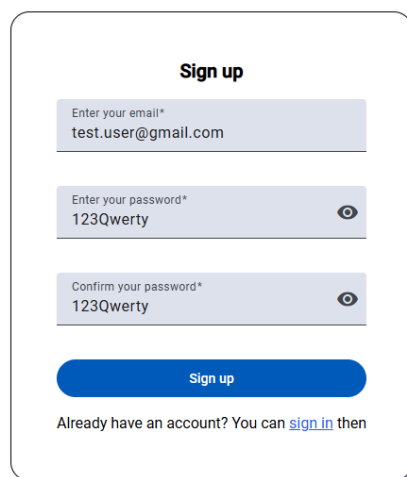
- створення нотатки
- відображення нотаток
- читання нотатки
- створення події
- відображення події на сьогодні/на тиждень/на наступний тиждень
- видалення події

Кожна сторінка містить зверху хедер, де відображені найменування поточної сторінки з навігацією на головну сторінку.

2. Опис реєстрації та авторизації

Для користування програмою необхідно зареєструватись у системі. Реєстрація складається з трьох етапів: введення інформації для входу в систему, підтвердження електронної адреси та інтеграції з Google calendar.

Для проходження першого етапу реєстрації необхідно ввести електронну адресу, пароль та підтвердження паролю.



The image shows a 'Sign up' form with a light blue border. At the top, the text 'Sign up' is centered. Below it are three input fields: 'Enter your email*' with the value 'test.user@gmail.com', 'Enter your password*' with the value '123Qwerty' and an eye icon, and 'Confirm your password*' with the value '123Qwerty' and an eye icon. A blue 'Sign up' button is at the bottom. Below the button, the text 'Already have an account? You can [sign in](#) then' is displayed.

Рисунок 1 – Форма реєстрації

Після введення усіх даних та натискання на кнопку “Sign up” користувачу на вказану поштову адресу має надійти лист з кодом верифікації. Даний код необхідно ввести у поле, зображене на рис. 2.

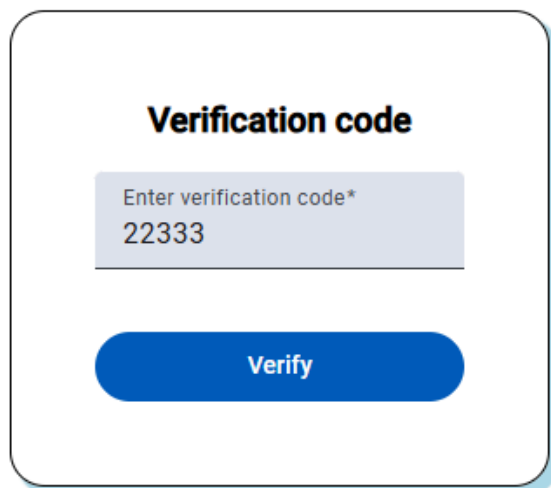
A rectangular form with rounded corners and a light blue border. At the top, the text "Verification code" is centered in bold. Below it is a light blue input field containing the placeholder text "Enter verification code*" and the value "22333". At the bottom of the form is a blue button with the text "Verify" in white.

Рисунок 2 – Форма коду підтвердження

Після успішної верифікації, користувачу необхідно додати Google Calendar ID. Ідентифікатор календаря можна знайти у налаштуваннях календаря. Даний ідентифікатор необхідно ввести у поле, що зображено на рис. 3. В разі успішного додавання користувача буде зареєстровано у системі.

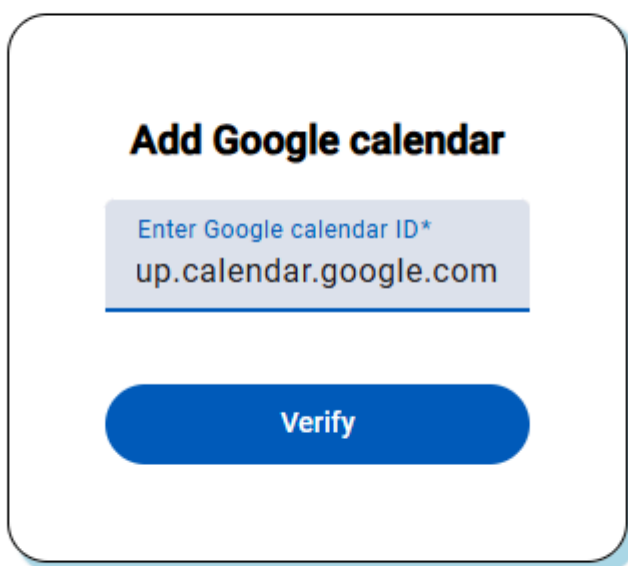
A rectangular form with rounded corners and a light blue border. At the top, the text "Add Google calendar" is centered in bold. Below it is a light blue input field containing the placeholder text "Enter Google calendar ID*" and the value "up.calendar.google.com". At the bottom of the form is a blue button with the text "Verify" in white.

Рисунок 3 – Форма вводу Google Calendar ID

Після того, як користувача було успішно авторизовано у системі, його переносить на головний екран системи, де його очікує персональний асистент. Також на головній сторінці, що зображена на рис. 4, для ознайомлення присутні основні команди, які допомагають здійснювати керування помічником.

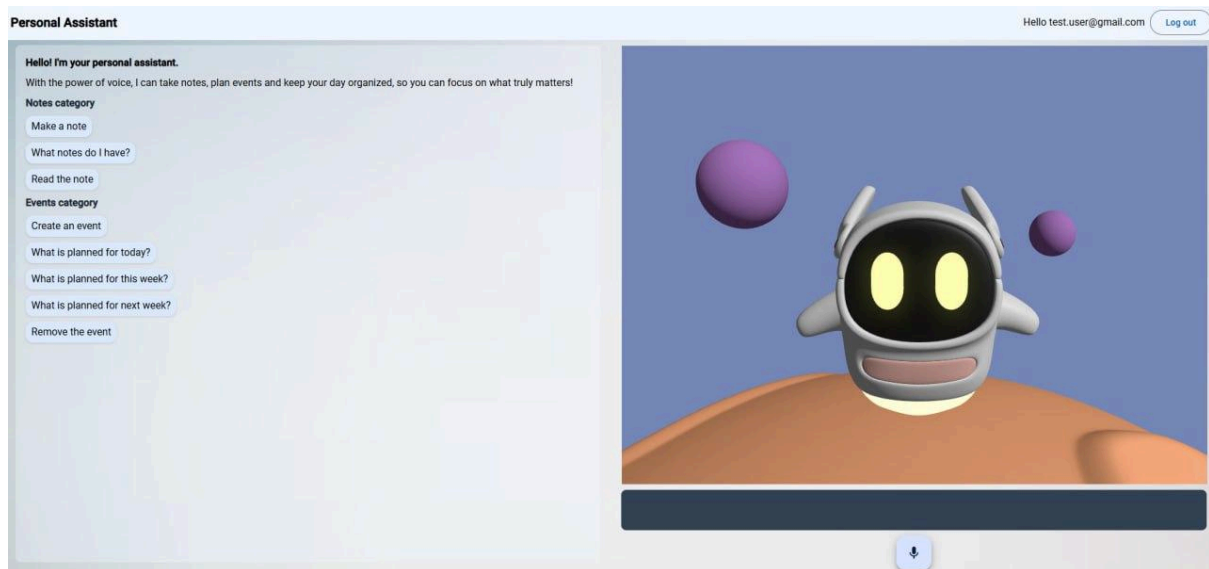


Рисунок 4 – Головна сторінка системи

3. Опис взаємодії з нотатками

3.1. Опис створення нотатки

На головній сторінці користувачу необхідно увімкнути мікрофон та сказати команду “Make a note”. Після розпізнавання команди буде виведено форму для створення нотатки. Спочатку необхідно проговорити назву нотатки. Після розпізнавання назви нотатки система автоматично заповнить поле “Name”. Після цього користувач матиме можливість озвучити текст нотатки. Аналогічно до назви, текст, сказаний користувачем, буде розпізнано та записано в поле “Description”.

Після заповнення форми користувач матиме змогу відредагувати нотатку за необхідності. Далі необхідно натиснути на кнопку “Submit” для збереження нотатки. Дану сторінку можна побачити на рис. 5.

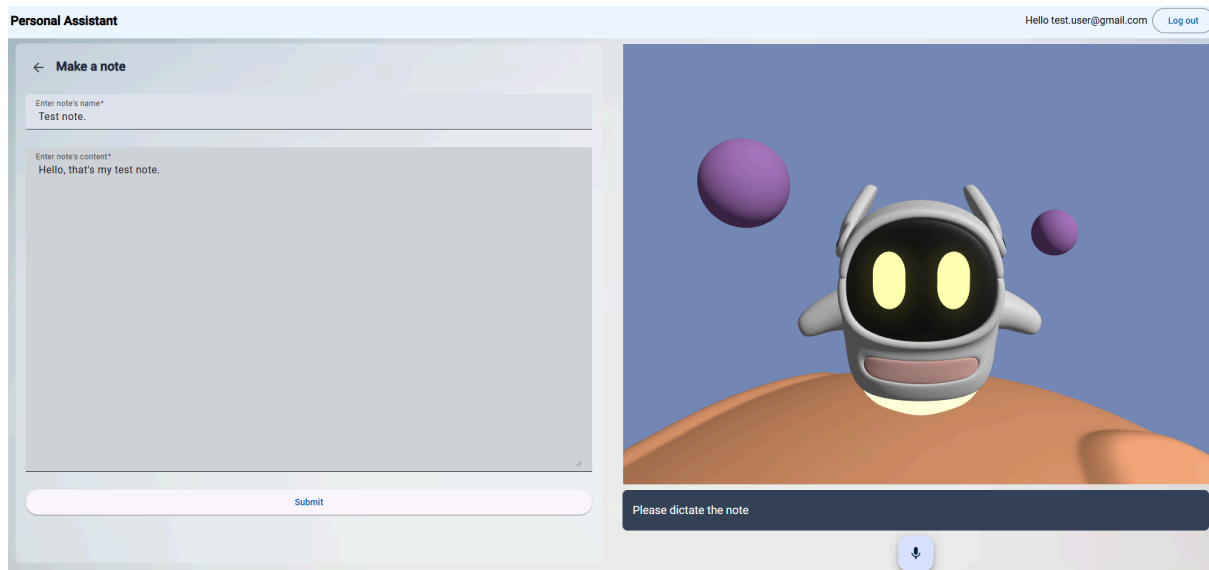


Рисунок 5 – Форма створення нотатки

У разі успішного збереження нотатки асистент сповістить користувача про успішне збереження нотатки (рис. 6).

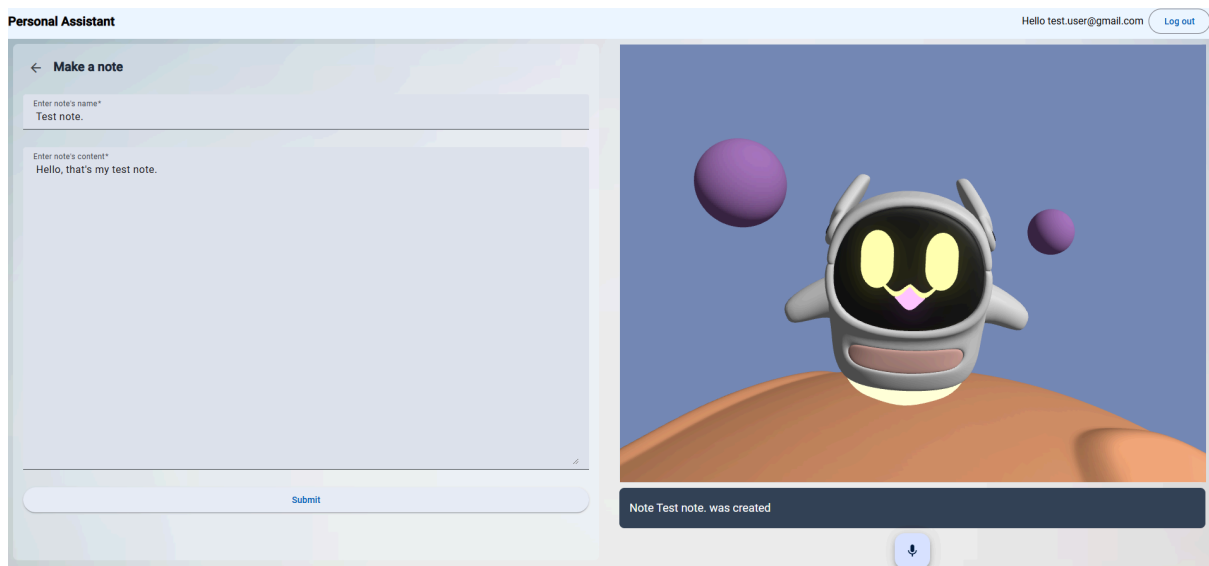


Рисунок 6 – Сповіщення аватара про успішне створення нотатки

3.2. Опис відображення нотаток

На головній сторінці користувачу необхідно увімкнути мікрофон та сказати “What notes do I have?”. Після цього користувача переведе в окреме меню, де він зможе переглянути свої нотатки (рис. 7). Якщо нотаток багато, то користувач може попросити показати більше нотаток.

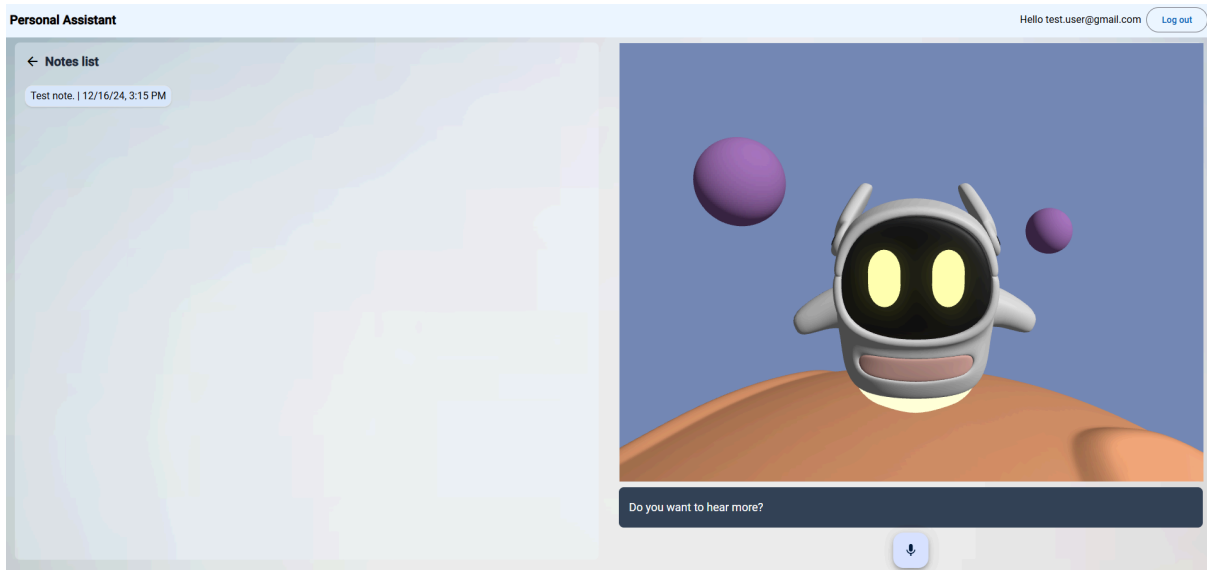


Рисунок 7 – Меню перегляду нотаток

3.3. Опис читання нотатки

На головній сторінці користувачу необхідно увімкнути мікрофон та сказати “Read the note”. Після цього користувача переведе в окреме меню, де асистент запитає користувача про ім’я нотатки, яку він хоче почути (рис. 8).

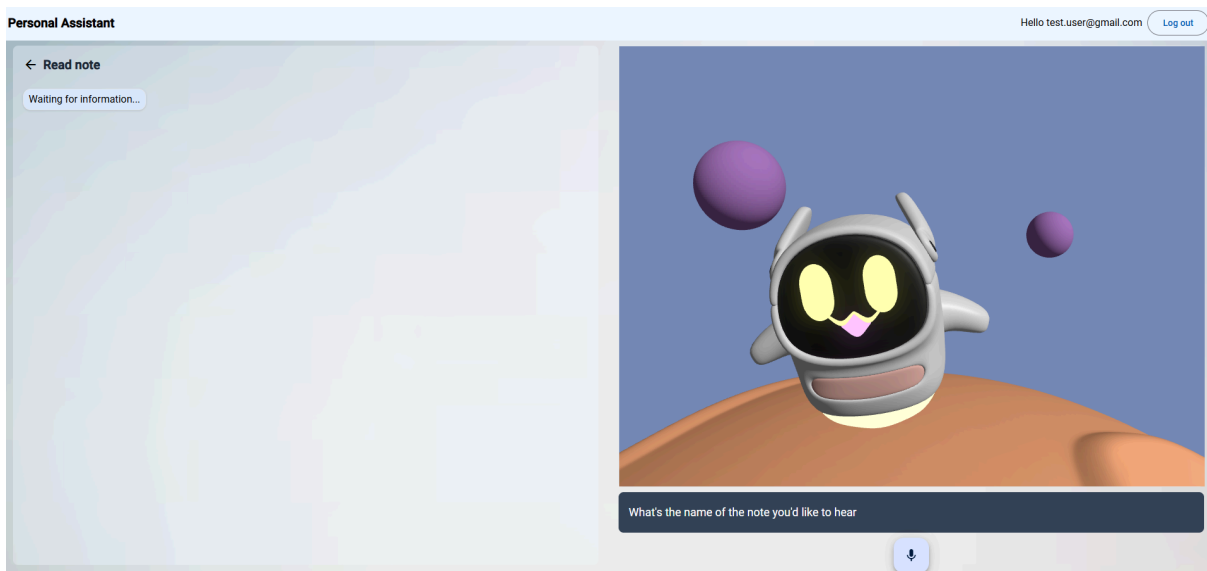


Рисунок 8 – Меню очікування назви нотатки

Після введення назви нотатки з’явиться вікно (рис. 9), у якому буде текст нотатки та асистент почне її озвучувати.

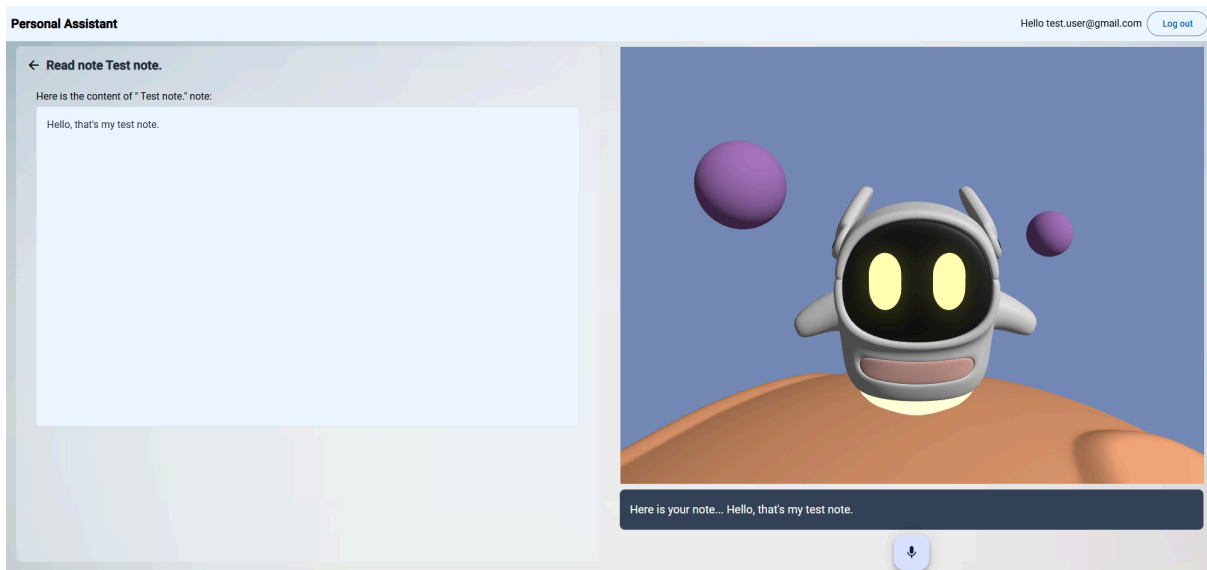


Рисунок 9 – Вікно з текстом нотатки

4. Опис взаємодії з подіями

4.1. Опис створення події

На головній сторінці користувачу необхідно увімкнути мікрофон та сказати “Create an event”. Після цього користувача переведе в окреме меню (рис. 10), де асистент запитає користувача про ім’я події, час початку та час кінця цієї події. Після заповнення відповідних полів необхідно натиснути кнопку “Submit” (рис. 11).

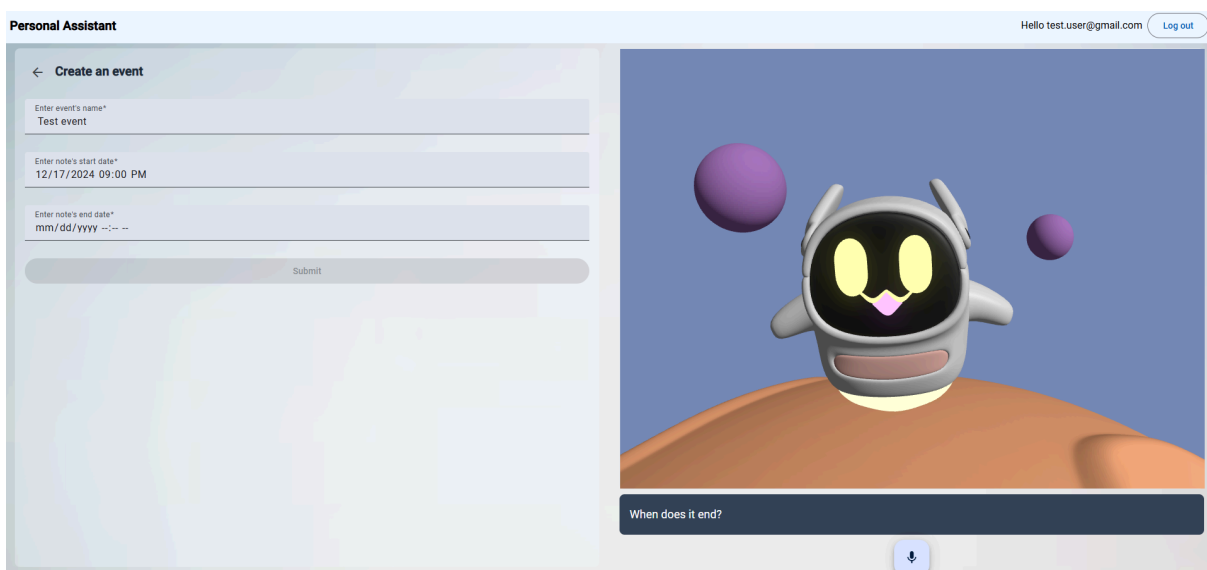


Рисунок 10 – Вікно створення події

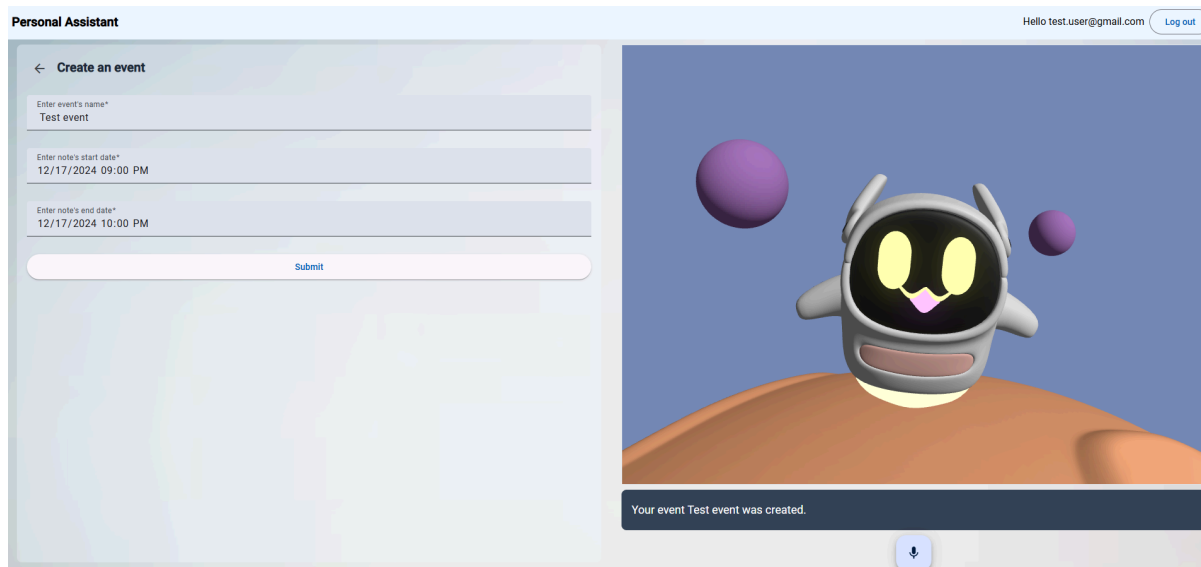


Рисунок 11 – Активна кнопка “Submit”

4.2. Відображення події

На головній сторінці користувачу необхідно увімкнути мікрофон та сказати “What is planned for *time*”. Замість “*time*” необхідно сказати часовий проміжок, який цікавить користувача, наприклад, “today”, “this week” або “next week”. Після цього користувача переведе в окреме меню (рис. 12-13), де користувач зможе переглянути події заплановані на той час, який вказав.

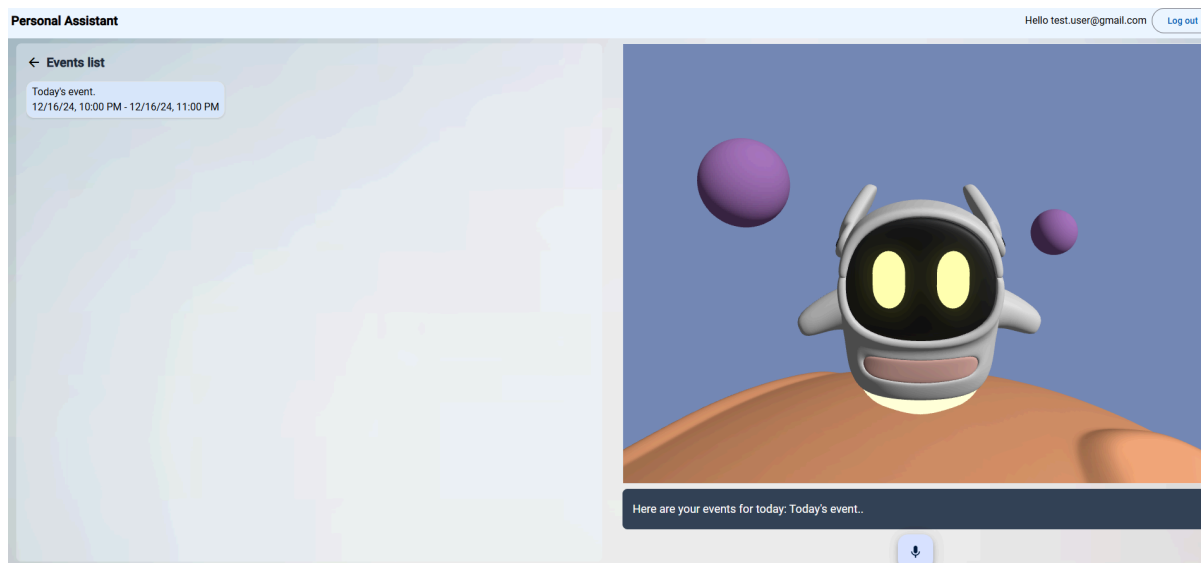


Рисунок 12 – Події, якщо в команді вказати “today”

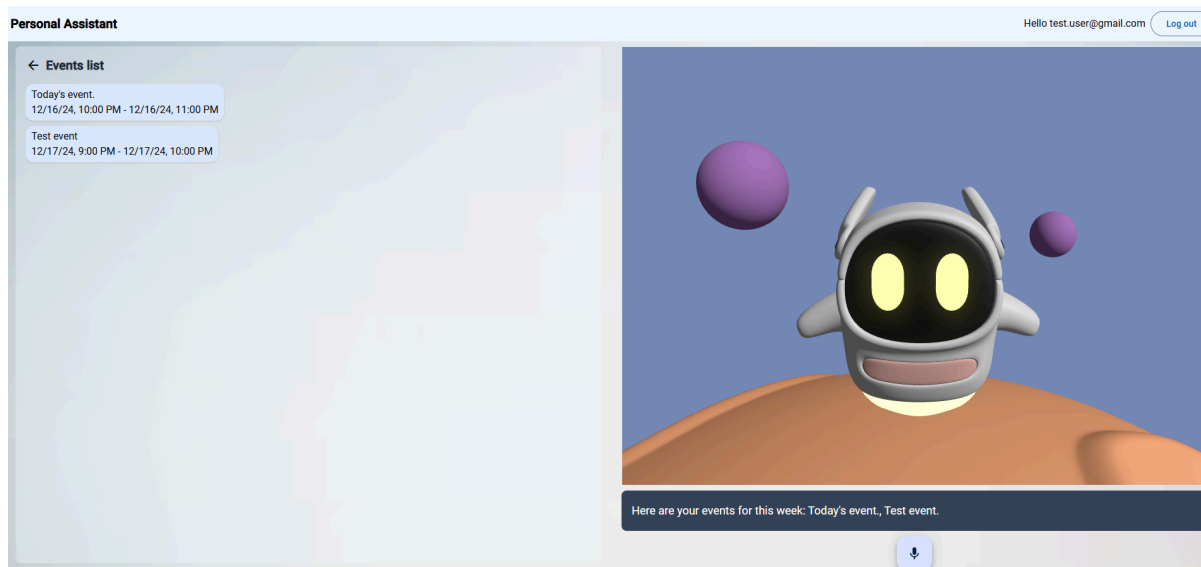


Рисунок 13 – Події, якщо в команді вказати “this week”

Всі події відмічені у власному Google Calendar користувача (рис. 14).

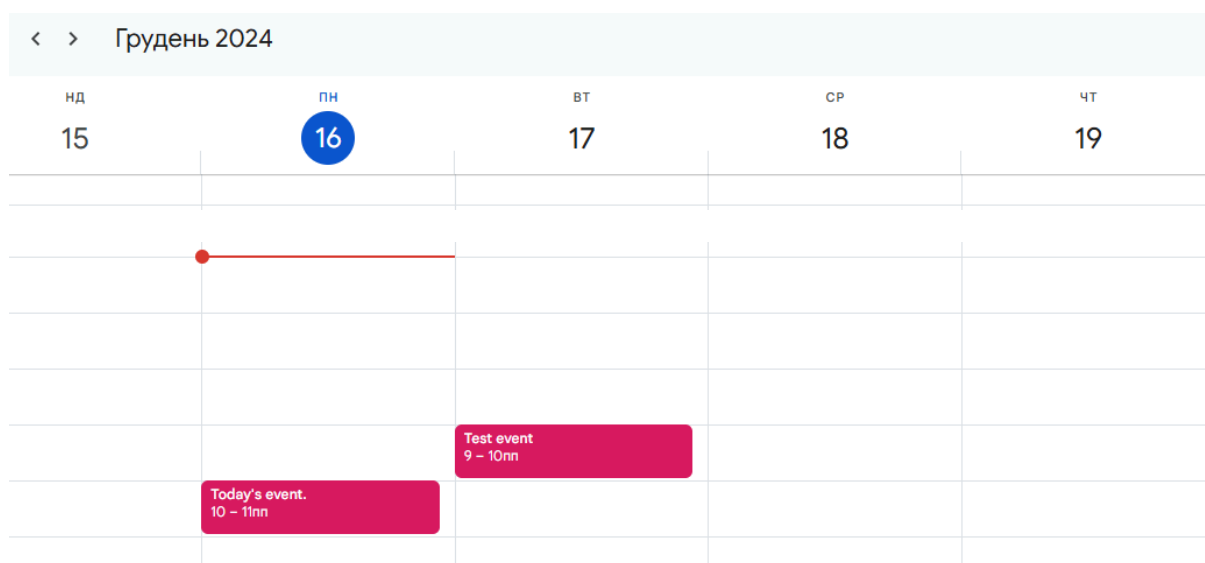


Рисунок 14 – Відображення подій в Google Calendar

4.3. Видалення події

На головній сторінці користувачу необхідно увімкнути мікрофон та сказати “Remove the event”. Після цього користувача переведе в окреме меню (рис. 15), де асистент запитає користувача про ім'я події, яку необхідно видалити. Після цього подію буде видалено.

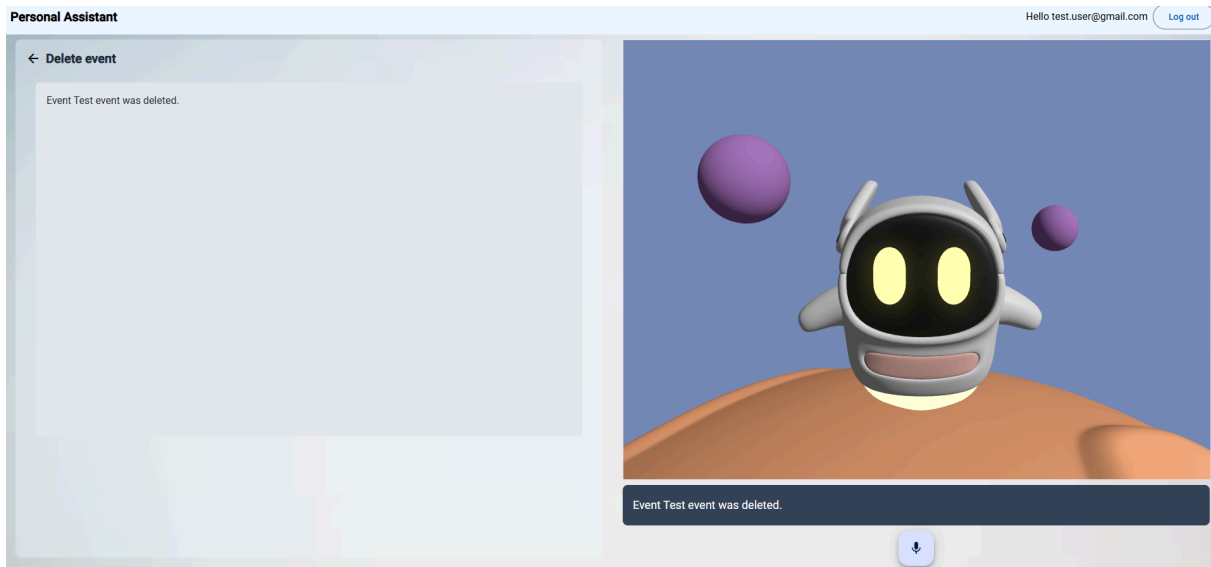


Рисунок 15 – Відображення видалення події

Видалення події також відбувається і в Google Calendar користувача (рис. 15).

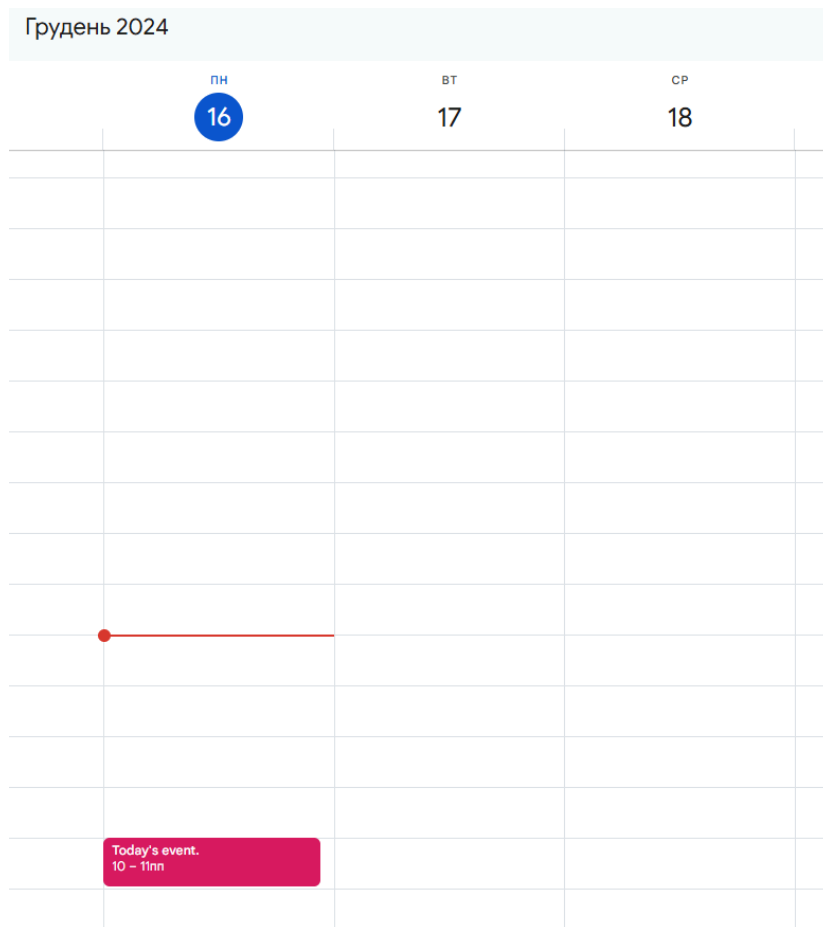


Рисунок 15 – Відображення подій в Google Calendar