

**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 ПРОГРАМНОГО  
ЗАБЕЗПЕЧЕННЯ ТА ІНТЕГРАЦІЯ GOOGLE CALENDAR**

Виконав студент групи КП-41мн  
Слободзян Максим Вікторович

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 ПРОГРАМНОГО**  
**ЗАБЕЗПЕЧЕННЯ ТА ІНТЕГРАЦІЯ GOOGLE CALENDAR**

**Пояснювальна записка**

**ПЗКС.045440-03-81**

Виконавець:

Слободзян М. В.

## ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ.....	3
ВСТУП.....	5
1. АНАЛІЗ ІНСТРУМЕНТІВ ПРОГРАМУВАННЯ ТА ТЕХНОЛОГІЙ.....	6
1.1. Мова програмування Python.....	6
1.2. СУБД PostgreSQL.....	7
1.3. Фреймворк Flask.....	8
1.4. Сервіс Google Calendar.....	9
2. РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	10
2.1. Структура бази даних.....	10
2.2. Модуль авторизації.....	13
2.3. Модуль роботи з нотатками.....	15
2.4. Модуль роботи з подіями.....	16
3. АНАЛІЗ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	19
3.1. Особливості реалізації системи.....	19
3.2. Тестування розробленої системи.....	23
3.3. Рекомендації щодо подальшого вдосконалення.....	24
ВИСНОВКИ.....	26
СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ.....	27

## СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

ПЗ – програмне забезпечення.

СУБД – система управління базами даних (програма для доступу до бази даних);

SQL – Structured Query Language (діалогова мова програмування для здійснення запиту і внесення змін до бази даних, а також керування базами даних);

Фреймворк – програмна платформа, що визначає структуру програмної системи;

JSON – JavaScript Object Notation, текстовий формат обміну даними, що використовує нотацію мови програмування JavaScript.

Cookies – невеликі текстові файли, які веб-сервер зберігає на пристрої користувача і використовуються для збереження даних, що дозволяють веб-сайту ідентифікувати користувача, зберігати його налаштування або стан сесії під час взаємодії з веб-застосунком.

ACID – Atomicity, Consistency, Isolation, Durability, набір властивостей, які забезпечують надійність операцій у базах даних, зокрема у транзакційних системах.

API – Application Programming Interface, інтерфейс для взаємодії між програмними компонентами, який визначає правила та способи обміну даними між ними.

REST – Representational State Transfer, архітектурний стиль для створення веб-сервісів, який базується на принципах роботи протоколу HTTP

HTTP – Hypertext Transfer Protocol, це протокол прикладного рівня, який використовується для передачі даних у веб-застосунках.

UTC – Coordinated Universal Time, всесвітній координований час, який є міжнародним стандартом для вимірювання часу.

SMTP – Simple Mail Transfer Protocol, стандартний мережевий протокол для передачі електронної пошти, що використовується для відправлення та пересилання листів між поштовими серверами, а також для передачі повідомлень від клієнтських застосунків до серверів електронної пошти.

GCP – Google Cloud Platform, набір хмарних обчислювальних сервісів, які надаються компанією Google. Ці сервіси дозволяють розробникам створювати, розгортати та масштабувати застосунки, використовуючи інфраструктуру Google.

CORS – Cross Origin Resource Sharing, механізм безпеки веб-браузерів, який дозволяє обмежувати або надавати доступ до ресурсів веб-сайту з іншого домену.

Ендпоїнт – (Endpoint) кінцева точка доступу до певного ресурсу або функціональності веб-сервісу через мережу.

## ВСТУП

У сучасному світі ефективне управління часом та інформацією стало важливою складовою успіху, як у професійній, так і у повсякденній діяльності. Люди стикаються з дедалі більшим обсягом завдань, подій та іншої інформації, що вимагає організованості та швидкого доступу до даних. Використання цифрових помічників, які можуть допомогти з плануванням, створенням нотаток і нагадувань, стає все більш актуальним.

Розроблення персонального асистента для ведення нотаток та планування подій спрямоване на вирішення цих завдань шляхом створення інтерактивного програмного забезпечення, яке поєднує в собі простоту використання, інтуїтивність та функціональність. Інтерактивний інтерфейс, зокрема підтримка голосових команд та 3D-анімованого персонажа, забезпечує зручну взаємодію користувача із системою. Голосове керування дозволяє користувачам отримати доступ до функціональності без необхідності використання клавіатури чи миші, що особливо важливо для людей з обмеженими можливостями або тих, хто прагне зекономити час. Проєкт сприятиме покращенню організації часу та інформації, забезпечуючи користувачам надійний і сучасний інструмент для вирішення повсякденних завдань.

Ця пояснювальна записка має на меті описати архітектуру і логіку програмного забезпечення, а саме: структуру бази даних системи, модулі авторизації користувача, роботи з нотатками, подіями а також особливості інтеграції з Google Calendar. Вона охоплює ключові аспекти, які забезпечують ефективну роботу персонального асистента, від базових функціональних можливостей до складних механізмів взаємодії з зовнішніми сервісами.

# 1. АНАЛІЗ ІНСТРУМЕНТІВ ПРОГРАМУВАННЯ ТА ТЕХНОЛОГІЙ

## 1.1. Мова програмування Python

Python – це універсальна, високорівнева, інтерпретована мова програмування, що визначається своїм зручним синтаксисом, простотою використання та гнучкістю [1]. Мова вже довгий час користується широкою популярністю у різних галузях, таких як веброзробка, аналіз даних, штучний інтелект та інші.

Однією з ключових переваг Python є його акцент на читабельності коду та зрозумілому синтаксисі. Використання відступів для визначення блоків коду спрощує структуру програм, роблячи їх зрозумілими навіть для початківців.

Python підтримує різні парадигми програмування, включаючи процедурне, об'єктно-орієнтоване та функціональне програмування, що дозволяє розробникам вибирати найбільш підходящий підхід для своїх завдань. Велика стандартна бібліотека мови надає різноманітні модулі для вирішення різноманітних задач, від роботи з файлами до веброзробки та мережевого програмування.

Python володіє живою та активною спільнотою, що призводить до появи численних сторонніх бібліотек і фреймворків, таких як Django для веброзробки, Flask для легких вебзастосунків, NumPy та pandas для роботи з даними, а також TensorFlow з PyTorch для машинного навчання. Ще однією важливою особливістю Python є його кросплатформенність та можливість використання інтерпретатора для інтерактивного виконання коду, що робить мову привабливою для широкого спектру завдань, від простих скриптів до великих і складних проектів у різних областях.

## 1.2. СУБД PostgreSQL

PostgreSQL — це потужна, відкрито кодована система керування базами даних (СУБД), яка підтримує реляційну модель даних, але також пропонує можливості для роботи з нереляційними даними [2]. PostgreSQL відома своєю надійністю, масштабованістю, та гнучкістю. Вона широко використовується для розробки додатків, де важливо зберігати великі обсяги даних та забезпечити їхню цілісність і консистентність.

Однією з головних особливостей PostgreSQL є підтримка розширених типів даних, включаючи JSON, XML, HSTORE, а також можливість створювати власні типи даних. Це дає змогу зберігати не лише структуровані дані, а й більш складні об'єкти, що дозволяє використовувати базу даних для широкого спектра застосувань — від класичних реляційних задач до зберігання та обробки документів чи геопросторових даних.

PostgreSQL підтримує транзакції, що забезпечують атомарність, консистентність, ізоляцію і довговічність (ACID-принципи) [3], що є критичним для підтримки високої надійності у фінансових, медичних та інших системах, де важливою є точність і цілісність даних.

Особливістю PostgreSQL є також її підтримка складних запитів, включаючи агрегацію, вкладені запити, комбінування даних з різних таблиць, а також можливості створення індексів. PostgreSQL підтримує зберігання процедур та функцій на таких мовах, як SQL, PL/pgSQL, Python, що дозволяє розширювати функціональність бази даних.

Завдяки своїй відкритій ліцензії та великій спільноті розробників, PostgreSQL продовжує активно розвиватися, забезпечуючи стабільність і потужність для бізнесових та наукових застосувань. Вона використовується як у малих, так і у великих системах, завдяки своїй здатності масштабуватися під високі навантаження.



### 1.3. Фреймворк Flask

Flask — це популярний мікрофреймворк для мови програмування Python, який забезпечує розробників зручними інструментами для створення веб-додатків [4]. Його гнучкість і мінімалістичність дозволяють легко налаштовувати функціональність залежно від потреб проєкту, уникаючи зайвої складності.

Основними перевагами фреймворку є:

- мінімалізм: Flask не нав'язує жодних строгих правил або структури, що робить його ідеальним для створення прототипів або проєктів із нестандартними вимогами.
- гнучкість: завдяки відсутності вбудованих компонентів, таких як ORM чи система автентифікації, розробники можуть вибирати відповідні інструменти для своїх потреб.
- розширюваність: Flask легко інтегрується з іншими бібліотеками та має велику кількість розширень, таких як Flask-SQLAlchemy, Flask-Login, Flask-Migrate тощо. Багато розширень буде використано у подальшій розробці проєкту.
- простота у навчанні: Flask підходить для початківців, оскільки дозволяє зрозуміти основи веб-розробки, не заглиблюючись у складні концепції.
- спільнота та документація: Велика кількість навчальних матеріалів та активна спільнота підтримують розробників на всіх етапах створення проєкту.

Серед недоліків фреймворку можна виділити обмежений набір вбудованих функцій. Flask не пропонує готових рішень для роботи з базами даних, автентифікації чи адміністративного інтерфейсу, як це є у повноцінних фреймворках на кшталт Django. Також через свою

мінімалістичність Flask часто потребує додаткових налаштувань та інтеграцій.

Проте, Flask надає більше свободи, тоді як Django нав'язує структуру проєкту. Django краще підходить для великих і комплексних проєктів. Тому саме для проєкту невеликого за масштабом доцільно буде обрати саме Flask як головний фреймворк розробки вебзастосунку.

#### **1.4. Сервіс Google Calendar**

Google Calendar — це безкоштовний онлайн-сервіс для управління часом та планування, розроблений компанією Google [5]. Він дозволяє користувачам створювати та редагувати події, встановлювати нагадування, додавати місця проведення та запрошувати інших учасників. Інтеграція з іншими сервісами Google, такими як Gmail, спрощує додавання подій безпосередньо з електронних листів.

Користувачі можуть створювати кілька календарів для різних аспектів свого життя та ділитися ними з іншими, що полегшує координацію та спільне планування. Сервіс доступний через веб-інтерфейс та мобільні додатки для Android та iOS, забезпечуючи синхронізацію даних між пристроями.

Google Calendar також підтримує функцію автоматичного додавання подій з Gmail, коли користувач отримує підтвердження бронювання або запрошення. Крім того, сервіс надає можливість налаштування сповіщень та нагадувань, що допомагає користувачам залишатися організованими та вчасно виконувати заплановані завдання.

Для розробників Google надає API календаря, який дозволяє інтегрувати функціональність календаря у власні додатки. Цей API підтримує створення, оновлення, видалення та перегляд подій, а також управління календарями та їхніми налаштуваннями.

## 2. РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

### 2.1. Структура бази даних

Структура бази даних розроблюваної системи персонального асистенту для ведення нотаток та планування подій зображена на рис. 2.1

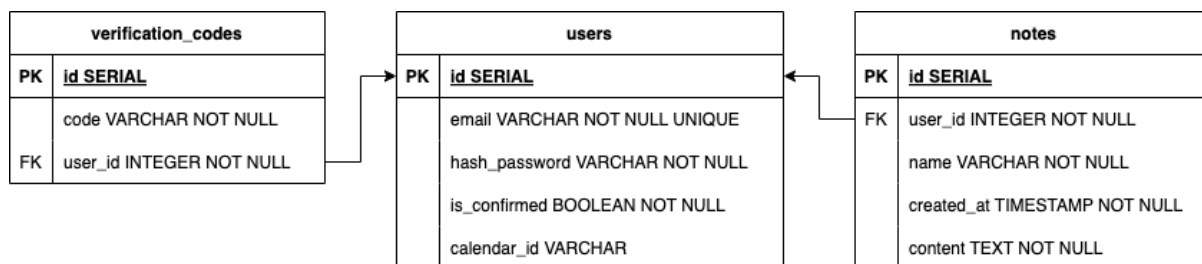


Рис. 2.1. Структура бази даних системи

Як можемо побачити, усього база даних налічує три основні таблиці. Детальніше опишемо кожну з них.

#### 2.1.1. Таблиця users

Таблиця зберігає основну інформацію про користувачів системи. Присутні наступні поля:

- id: SERIAL — первинний ключ, який автоматично генерується для кожного нового запису. Забезпечує унікальну ідентифікацію користувача.
- email: VARCHAR NOT NULL UNIQUE — електронна пошта користувача. Поле є унікальним і обов'язковим для заповнення. Використовується як логін для авторизації.
- hash\_password: VARCHAR NOT NULL — хеш пароля користувача. Зберігається у зашифрованому вигляді для забезпечення безпеки.

- `is_confirmed`: `BOOLEAN NOT NULL` — статус підтвердження користувача. Визначає, чи підтверджена електронна пошта користувача.
- `calendar_id`: `VARCHAR` — ідентифікатор календаря користувача у зовнішній системі (Google Calendar).

### 2.1.2. Таблиця `notes`

Таблиця зберігає нотатки, створені користувачами. Присутні наступні поля:

- `id`: `SERIAL` — первинний ключ, що забезпечує унікальну ідентифікацію кожної нотатки.
- `user_id`: `INTEGER NOT NULL` — зовнішній ключ, що посилається на поле `id` у таблиці `users`. Вказує на користувача, якому належить нотатка.
- `name`: `VARCHAR NOT NULL` — назва нотатки. Визначає короткий опис або заголовок нотатки.
- `created_at`: `TIMESTAMP NOT NULL` — дата та час створення нотатки. Використовується для сортування та відображення в хронологічному порядку.
- `content`: `TEXT NOT NULL` — вміст нотатки. Містить детальний текст, що був збережений користувачем.

### 2.1.3. Таблиця `notes`

Таблиця зберігає коди підтвердження, які використовуються для верифікації користувачів. Присутні наступні поля:

- `id`: `SERIAL` — первинний ключ, що забезпечує унікальну ідентифікацію кожного коду.

- `code: VARCHAR NOT NULL` — сам код підтвердження, що надсилається користувачеві для верифікації.
- `user_id: INTEGER NOT NULL` — зовнішній ключ, що посилається на поле `id` у таблиці `users`. Вказує на користувача, для якого створено код підтвердження.

Розглянемо основні відношення між таблицями у розробленій базі даних. Поле `user_id` у таблицях `notes` та `verification_codes` є зовнішнім ключем і забезпечує зв'язок із полем `id` у таблиці `users`. Це створює відношення один до багатьох між таблицею `users` та дочірніми таблицями `notes` і `verification_codes`, де один користувач може мати кілька нотаток та кодів підтвердження.

Міграція таблиць виконувалася за допомогою розширення `Flask-Migrate`. `Flask-Migrate` — це розширення для `Flask`, яке забезпечує інтеграцію `Alembic` (інструменту для міграцій баз даних) [6] з проєктами на `Flask` [7]. Воно дозволяє легко створювати, виконувати та керувати змінами у схемі бази даних під час розробки.

Серед основних переваг `Flask-Migrate` можна виділити автоматичне створення міграцій на основі змін у моделях `SQLAlchemy`, застосування та відкат міграцій через команди з терміналу та зручна робота з `Flask-SQLAlchemy`, що дозволяє підтримувати зміни в структурі бази даних без втрати даних.

Як можемо побачити, таблиці для подій у базі даних не передбачено. Це зумовлено особливістю інтеграції з сервісом `Google Calendar`, що буде розглянуто у наступних розділах.

## 2.2. Модуль авторизації

Модуль авторизації користувачів застосунку був розроблений на основі фреймворку Flask з використанням розширення Flask-Login.

Flask-Login є розширенням для Flask, яке спрощує реалізацію аутентифікації користувачів у веб-застосунках [8]. Воно дозволяє зберігати інформацію про користувача у сесії, а також надає інструменти для входу, виходу та перевірки статусу користувача. Під час аутентифікації функція `login_user()` додає користувача до поточної сесії, використовуючи його ідентифікатор, що зберігається у cookies. Для перевірки аутентифікації використовується об'єкт `current_user`, який автоматично надається Flask-Login і містить усі дані про активного користувача.

Захист маршрутів здійснюється за допомогою декоратора `@login_required`, який обмежує доступ лише для авторизованих користувачів, перенаправляючи неаутентифікованих на сторінку входу. Вихід із системи реалізується за допомогою функції `logout_user()`, що очищує дані про користувача із сесії. Розширення підтримує обробку "пам'ятати мене" під час входу, що дозволяє зберігати аутентифікацію навіть після закриття браузера.

Використовуючи вищеописане розширення, розроблений модуль авторизації забезпечує реєстрацію, верифікацію, логін, вихід із системи та доступ до захищених ресурсів користувача. Зокрема, було:

- Розширено стандартний декоратор `@login_required` – `login_verify_required`. Цей декоратор обмежує доступ до маршрутів для неаутентифікованих користувачів або для користувачів, які не пройшли верифікацію. Перевіряється статус підтвердження електронної пошти користувача. У випадку невідповідності виконується перенаправлення на сторінку логіну.

- Ініційовано додаткову схему REST API інтерфейсу додатку, що відповідає за маршрути авторизації.
- Створено маршрут `/signup`, що обробляє реєстрацію нового користувача. Пароль користувача хешується за допомогою `generate_password_hash`, а дані зберігаються у базі даних. Також генерується код верифікації, який надсилається користувачу електронною поштою.
- Додано маршрут `/verify`, що обробляє верифікацію користувача за допомогою коду підтвердження. Якщо код вірний, користувача позначають як підтвердженого та автоматично авторизують у систему.
- Додано маршрут `/login`, що забезпечує аутентифікацію користувачів. Перевіряє користувача за вказаною поштою, виконуючи порівняння пароля з хешем, що зберігається в базі даних. У разі успіху користувача аутентифіковано.
- Додано маршрут `/logout` – вихід із системи користувача.

Авторизація була розроблена за допомогою авторизаційних кукі (Cookies) і є одним із ключових механізмів, який використовується у Flask-Login для підтримки стану аутентифікації користувача під час роботи з веб-застосунком. Після успішного входу користувача в систему функція `login_user()` створює спеціальні сесійні кукі, які зберігаються на стороні клієнта у браузері. Ці кукі містять ідентифікатор користувача, який дозволяє серверу розпізнавати аутентифікованого користувача при кожному новому запиті.

При кожному запиті браузер автоматично надсилає авторизаційні кукі на сервер, що дозволяє Flask-Login ідентифікувати користувача через функцію `current_user`. Ця функціональність дозволяє зберігати стан аутентифікації між запитами, без необхідності повторно вводити облікові

дані. Кукі є безпечними та автоматично створюються на основі сесії, яку керує Flask.

У результаті розроблений модуль авторизації є гнучким та безпечним рішенням, що підтримує реєстрацію користувачів, їхню верифікацію, аутентифікацію та вихід із системи. Використання перевірених методів хешування паролів і декораторів для обмеження доступу дозволяє забезпечити високий рівень безпеки та надійності модулю.

### **2.3. Модуль роботи з нотатками**

Модуль роботи з нотатками забезпечує функціональність для створення, отримання та пагінації нотаток, які зберігаються в базі даних. Він інтегрується з ORM (SQLAlchemy) для взаємодії з моделлю Note і базою даних системи.

Створена модель Note, що відповідає сутності таблиці notes, наслідується від базової моделі db.Model, що дозволяє зручно та ефективно виконувати запити до бази даних. Вона містить усі необхідні для роботи поля, яким було призначено тип відповідно до типів таблиці. Також було вказано на залежність від моделі User.

Основну логіку роботи з нотатками було винесено до відповідного репозиторію Notes, що реалізує наступні методи:

- `get_user_notes(user_id)` – повертає всі нотатки, які належать користувачеві із заданим `user_id`. Вона виконує SQL-запит, фільтруючи записи таблиці Note за полем `user_id` та сортує нотатки в порядку зменшення дати створення (`created_at.desc()`). Повертається список об'єктів Note.
- `get_notes_page(user_id, page, limit)` – реалізує пагінацію для отримання частини нотаток користувача. Спочатку перевіряється, чи значення параметра `page` більше або дорівнює 1. Якщо ні,



воно примусово встановлюється на 1. Розраховується зсув (offset) для визначення, з якої нотатки починати вибірку. SQL-запит фільтрує нотатки за user\_id, сортує їх за created\_at і застосовує обмеження (limit) та зсув (offset). повертається список об'єктів Note для відповідної сторінки.

- insert\_note(user\_id, name, content) – додає нову нотатку в базу даних. Створюється новий екземпляр моделі Note з полями user\_id, name та content. Нова нотатка додається до сесії бази даних.

Розроблений модуль реалізує основні операції і є простим, гнучким та ефективним інструментом для роботи з нотатками у контексті бази даних.

## **2.4. Модуль роботи з подіями**

Модуль для роботи з подіями було розроблено з використанням клієнтської бібліотеки google-api-python-client. Вона реалізує підключення до основних API сервісів Google, зокрема і Google Calendar.

Робота з користувацькими даними дещо відрізняється від принципу модуля обробки нотаток. Замість збереження даних у таблиці бази даних, модуль напряму працює з сервісами Google, де зберігаються усі дані користувача. Створення, перегляд, видалення подій здійснюється напряму через взаємодію з відповідним маршрутом API. Це дозволяє не тільки зменшити обсяг бази даних та зусилля на її утримання, а й враховувати додаткові події, які користувач може створювати у власному календарі за рамками розробленого ПЗ.

Першим чином було створено модель події Event, яка містить основні поля, такі як назва, час початку та кінця. Також вона містить додаткові методи to\_calendar\_body та to\_dict, що допомагають перетворити

її у відповідно Google-сутність для операцій з API Google та сутність для роботи на клієнтській частині ПЗ.

Основний розроблений модуль управління подіями можна умовно розбити на три підмодулі – робота з календарем користувача, управління безпосередньо подіями та фільтрація подій за періодом.

До підмодулю роботи з календарем відносяться наступні методи:

- `list_available_calendars`: Повертає список доступних календарів, отримуючи їх через API Google Calendar. Кожен календар ідентифікується своїм `summary`.
- `add_shared_calendar`: Додає спільний календар за його `calendar_id`. Якщо виникає помилка (наприклад, недостатньо прав або календар не знайдено), генерує відповідне виключення.
- `valid_calendar_access`: Перевіряє права доступу до календаря. Повертає `True`, якщо користувач має права `owner` або `writer`, що дозволяють змінювати події.

До підмодулю управління безпосередньо подіями відносяться наступні методи:

- `create_event`: Додає нову подію до зазначеного календаря. Використовує структуру `Event`, що перетворюється у відповідний формат Google Calendar.
- `list_events`: Повертає список усіх подій у календарі. Якщо подій немає або доступ обмежений, викликає відповідні виключення.
- `delete_event`: Видаляє подію за її `event_id` у конкретному календарі. Якщо подію або календар не знайдено, генерує виключення.
- `delete_events_by_name`: Видаляє всі події, які відповідають заданій назві (`event_name`). Повертає кількість успішно видалених подій.

До підмодулю фільтрацій подій за періодом відносяться:

- `list_events_period`: Повертає події за певний період, наприклад:
  - `today`: події на сьогодні.
  - `this_week`: події на поточний тиждень.
  - `next_week`: події на наступний тиждень.
- `list_events_in_time_range`: Отримує події у зазначеному часовому діапазоні, сортує їх за часом початку та повертає у вигляді списку.

Розрахунок часових меж, дат, часу створення сутностей відбувається з урахуванням другого часового поясу (UTC+2). На рисунку 2.2 зображено приклад створених подій у календарі користувача.

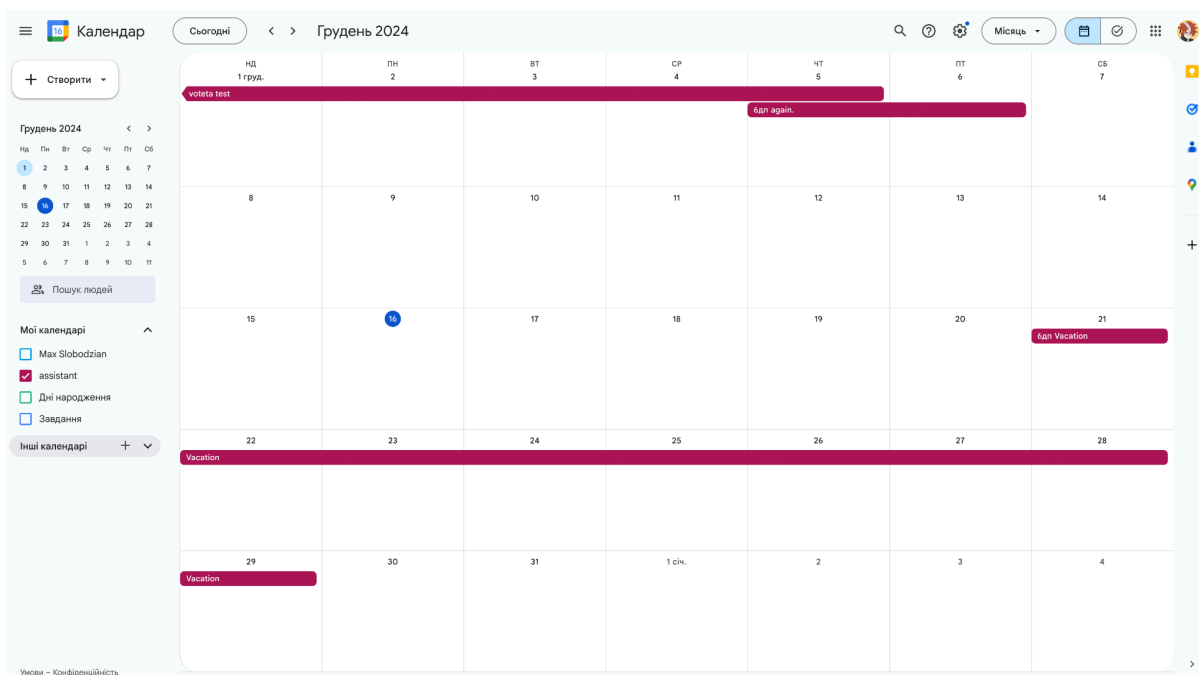


Рис. 2.2. Створені події в календарі користувача

Розроблений модуль взаємодії реалізує усі необхідні методи для керування подіями користувача у його календарі. Він є гнучким і може бути адаптований для роботи з іншими сервісами, що використовують подібні API.

### 3. АНАЛІЗ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

#### 3.1. Особливості реалізації системи

##### 3.1.1. Інтеграція Mailjet

Однією з ключових функцій системи є надсилання верифікаційних кодів користувачам під час реєстрації. Для реалізації цієї задачі використовується інтеграція із сервісом Mailjet, який забезпечує надійне та швидке відправлення електронних листів.

Mailjet — це хмарний сервіс для управління електронною поштою, який дозволяє надсилати транзакційні та маркетингові повідомлення через SMTP або REST API [9]. Сервіс підтримує відправку як окремих листів, так і масових розсилок.

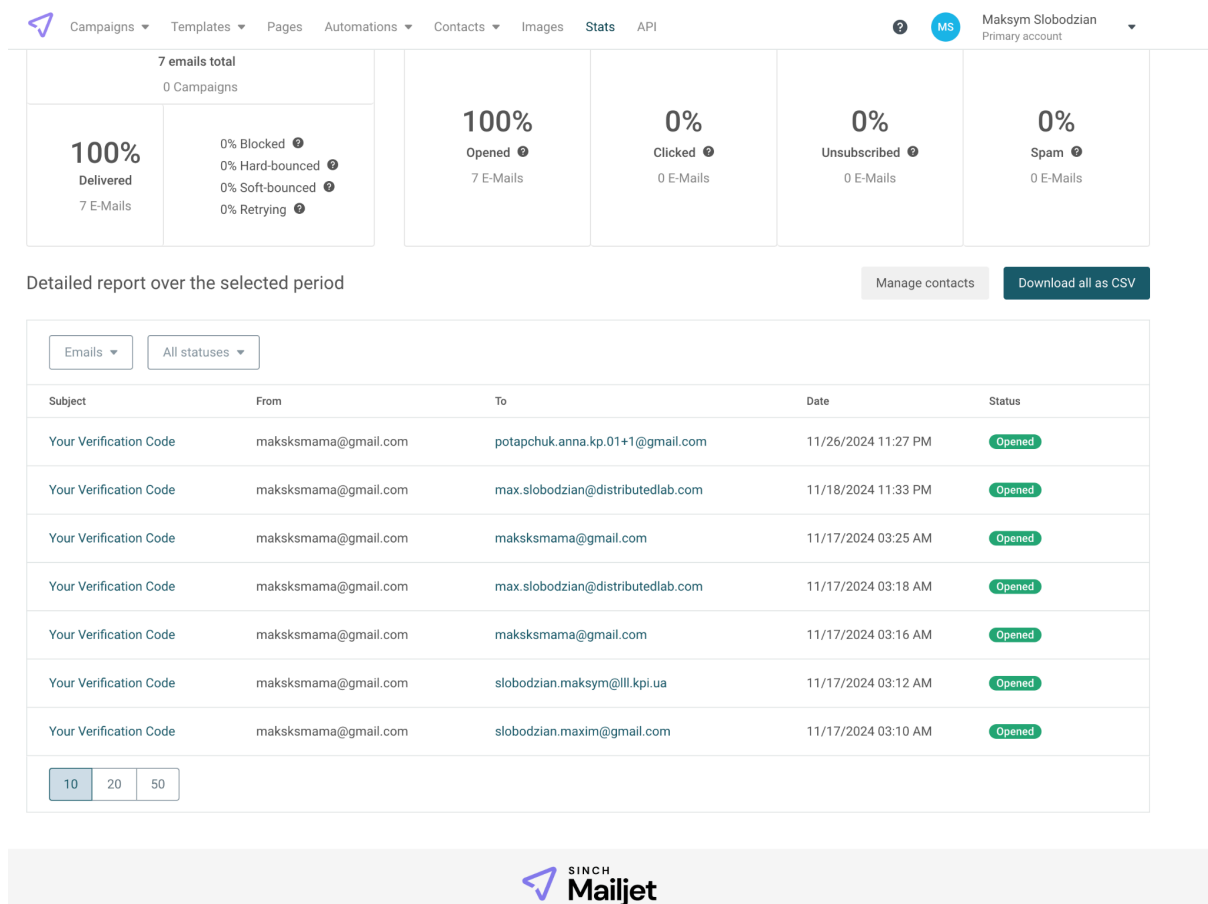


Рис. 3.1. Панель адміністратора Mailjet

Його перевагами є висока швидкість доставки, підтримка шаблонів для персоналізації та можливість відстеження статусу відправлення. Сервіс забезпечує високий рівень безпеки, пропонуючи API-ключі для аутентифікації, а також дозволяє детально відстежувати статус кожного відправленого листа (доставлений, відкритий, відхилений тощо), що зображено на рис. 3.1.

Для інтеграції Mailjet використовується його REST API, що дозволяє програмно створювати та надсилати листи. Процес відправлення верифікаційного коду реалізований таким чином:

- генерація верифікаційного коду: Після реєстрації нового користувача система створює унікальний код підтвердження, який зберігається в базі даних.
- формування листа: Вміст листа генерується за допомогою шаблону, який містить текст повідомлення та динамічне поле для верифікаційного коду.
- відправка через Mailjet API: Викликається функція API Mailjet, яка передає лист із зазначеними параметрами (адреса отримувача, тема, тіло повідомлення).
- обробка відповіді: Система перевіряє статус відправлення, щоб гарантувати доставку листа. У разі помилок (некоректна адреса, обмеження доступу) відповідна інформація записується у лог.

Сервіс Mailjet також дозволяє самостійно налаштовувати тіло повідомлення, його розмітку, дизайн тощо. Приклад авторизаційного листа наведено на рис. 3.2

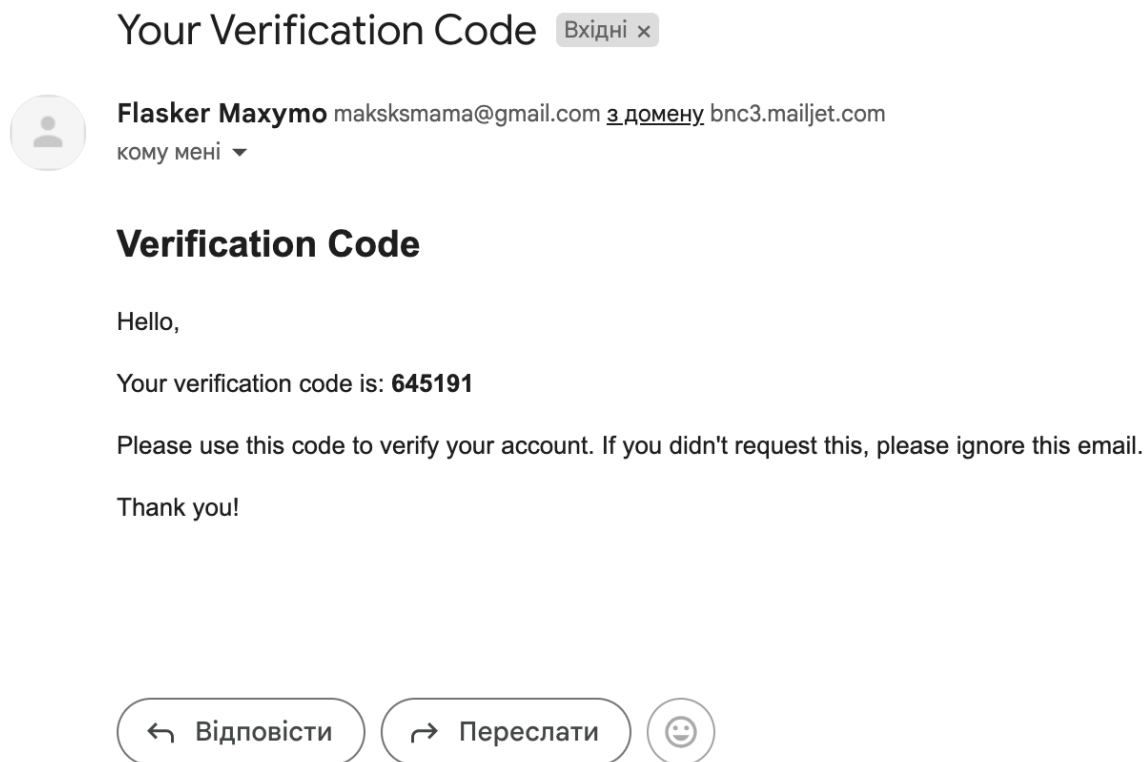


Рис. 3.2. Авторизаційний лист

Інтеграція із Mailjet дозволяє автоматизувати процес надсилання верифікаційних кодів, роблячи його надійним і зручним як для розробників, так і для користувачів. Це важливий елемент системи, що забезпечує безпеку облікових записів і підвищує рівень довіри до платформи. Завдяки можливостям Mailjet система легко масштабується та може обробляти великий обсяг реєстрацій одночасно.

### 3.1.2. Створення Google-застосунку для роботи з календарями

Для інтеграції з Google Calendar API спершу необхідно створити Google-застосунок у Google Cloud Platform (GCP), який надає можливість отримати доступ до API календаря. Щоб почати, потрібно мати обліковий запис Google і доступ до Google Cloud Console.

Спочатку на головному екрані необхідно натиснути кнопку "Створити проект", вибрати його ім'я та погодитись. В новоствореному проєкті необхідно обрати пункт APIs & Services, у полі пошуку ввести Google Calendar API та натиснути кнопку "Застосувати".

Далі необхідно налаштувати авторизацію у застосунку через OAuth 2.0. Налаштовуємо екран згоди та форму. У подальшому вона буде пояснювати користувачам які функції та сервіси зможуть використовуватись нашим додатком. У вкладці "Credentials" створюємо облікові дані доступу до API. Приклад застосунку показано на рис. 3.3:

OAuth consent screen management is changing. This page has been replaced with a new, easier-to-use experience. The current pages will only be available for a few more days. [GO TO NEW EXPERIENCE](#)

Name \*  
Desktop client 1  
The name of your OAuth 2.0 client. This name is only used to identify the client in the console and will not be shown to end users.

Note: It may take 5 minutes to a few hours for settings to take effect

[SAVE](#) [CANCEL](#)

**Additional information**

Client ID	756179061613-kang55ju3ffpkkgadnk4019odu3dk3u.apps.googleusercontent.com
Creation date	November 19, 2024 at 12:41:30 AM GMT+2

**Client secrets**

If you are in the process of changing client secrets, you can manually rotate them without downtime. [Learn more](#)

Having more than one secret increases security risks. Disable and delete the old secret once you have verified that your app is using the new secret instead of the old secret.

Client secret	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	<a href="#">Copy</a> <a href="#">Download</a> <a href="#">Delete</a>
Creation date	November 19, 2024 at 12:41:30 AM GMT+2	
Status	Enabled	<a href="#">DISABLE</a>
Client secret	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	<a href="#">Copy</a> <a href="#">Download</a> <a href="#">Delete</a>
Creation date	December 14, 2024 at 5:50:55 PM GMT+2	
Status	Enabled	<a href="#">DISABLE</a>

+ ADD SECRET

Рис. 3.3. Google-застосунок для роботи з календарями

Для подальшої роботи програми необхідно авторизуватися від імені акаунту, що буде виконувати взаємодію з користувацькими календарями. Для цього може бути обрано як і сервісний акаунт, так і будь-який користувацький обліковий запис. Програмне забезпечення підключення до API Google побудоване таким чином, що при першому запуску програми буде викликано функцію авторизації, яка поверне API токен з доступом до відповідного облікового запису. Таким чином, ПЗ буде повністю налаштоване до коректної роботи.

### 3.1.3. Налаштування користувацького календаря для роботи із застосунком

Щоб налаштувати обробку подій, спершу необхідно поділитися ідентифікатором календаря з акаунтом системи. Для цього потрібно обрати бажаний календар Google та перейти в його налаштування. У графі “Спільне використання” необхідно додати акаунт системи за вказаною адресою та надати йому доступ до змін та перегляду подій, як зображено на рис. 3.4.

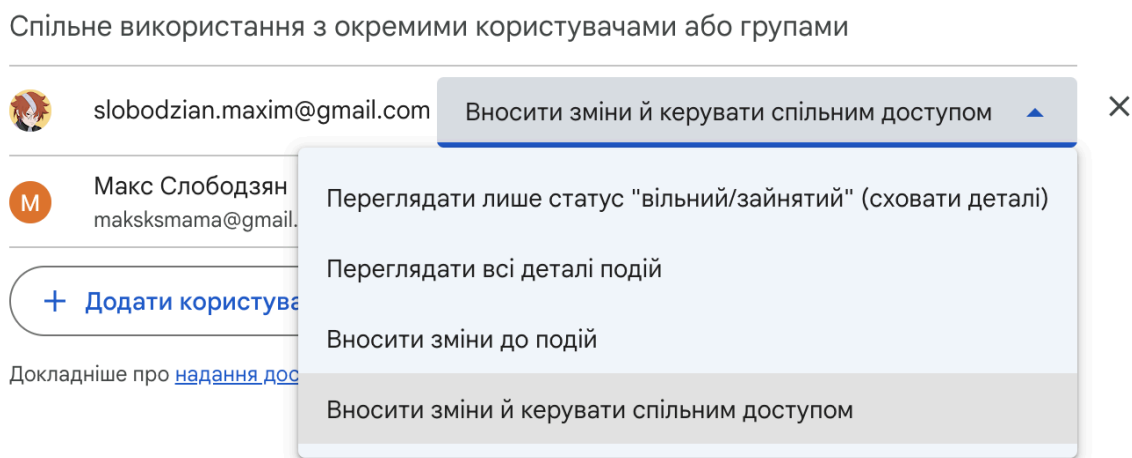


Рис. 3.4. Налаштування користувацького календаря

Після цього графою нижче, у розділі “Інтеграція календаря” можна знайти його загальнодоступний ідентифікатор. Ним і потрібно поділитися з системою.

Якщо не надати авторизованого користувача, або ж наділити його неповними правами, система повідомить про помилку доступу, а отже такий календар не буде підтримуватись застосунком.

## 3.2. Тестування розробленої системи

Тестування розроблюваної системи відбувалося мануально, оскільки цей підхід дозволяє детально перевірити функціонування кожного з



компонентів системи в реальних умовах. Мануальне тестування було особливо важливим для перевірки запитів до бази даних, налаштувань авторизаційної сесії, взаємодією з API Google Календаря, що не було б можливо описати модульним чи інтеграційним тестуванням.

Під час мануального тестування було перевірено функціональність розроблених модулів системи. Воно дозволило виявити проблеми, які могли бути не помічені під час автоматизованих тестів, наприклад, проблеми з CORS-доступами при взаємодії з API-ендпоїнтами

Завдяки такому підходу до тестування вдалося виявити і виправити потенційні помилки в роботі системи до її запуску в реальних умовах. Це допомогло забезпечити високий рівень надійності та стабільності роботи системи, гарантуючи, що всі функції працюють так, як очікується, при будь-яких умовах.

### **3.3. Рекомендації щодо подальшого вдосконалення**

У розділі, присвяченому потенційним новим функціям для вдосконалення системи розпізнавання номерних знаків для управління доступом до паркування, можна розглянути кілька цікавих напрямків для подальшого розвитку та покращення системи. Вони включають як додаткові можливості для користувачів, так і нові функціональні можливості, які сприятимуть покращенню ефективності та масштабованості системи.

Для модуля роботи з нотатками можна додати функцію пошуку за ключовими словами або категоріями, що дозволить користувачам швидко знаходити необхідну інформацію. Також варто розглянути можливість реалізації нагадувань для певних нотаток із інтеграцією з календарем, що покращить управління завданнями.

Оскільки система є масштабованою і може підтримувати сумісні з Google Calendar інтерфейси, є можливість інтеграції з іншими календарними сервісами, такими як Microsoft Outlook, Apple Calendar, або iCalendar. Така функціональність допоможе користувачам обрати зручний для них тип застосунку без необхідності мігрувати на чітко визначені системою, що значно покращить користувацький досвід.

Для пришвидшення роботи розроблених модулів можна застосувати декілька технік. Наприклад, запити, які виконуються часто і мають відносно статичний результат (наприклад, список подій або нотаток), можна кешувати за допомогою таких інструментів, як Redis або Memcached. Це дозволить скоротити навантаження на сервер і базу даних. Оптимізація роботи із зовнішніми API, наприклад, Google Calendar API також пришвидшить час відклику програмного модулю. Для зменшення затримок можна використовувати асинхронні запити, які не блокують основний потік виконання програми. Крім того, варто кешувати відповіді від зовнішніх API, щоб зменшити кількість повторних викликів.

Впровадження нових функцій та технік дозволить не тільки покращити функціональність системи, а й зробити її більш гнучкою та масштабованою, що сприятиме кращому користувацькому досвіду та підвищенню ефективності системи.

## ВИСНОВКИ

Розробка модулів для управління нотатками, подіями та інтеграції з Google Calendar є важливим етапом у створенні ефективного та зручного інструменту для організації часу та інформації. Система, заснована на сучасних веб-технологіях, забезпечує функції створення, редагування та перегляду нотаток, а також управління подіями через інтеграцію з календарями. Використання REST API для взаємодії з Google Calendar дозволяє автоматизувати планування та синхронізацію подій між пристроями, що забезпечує високий рівень зручності для користувачів.

Під час тестування були перевірені ключові аспекти роботи модулів, включаючи коректність виконання CRUD-операцій із нотатками та подіями, стабільність інтеграції з Google Calendar API, а також швидкість обробки запитів. Особливу увагу було приділено безпеці роботи з даними користувачів, зокрема хешуванню паролів, обробці токенів доступу та верифікації електронних листів через інтеграцію з сервісом Mailjet.

У результаті розроблені модулі забезпечують стабільну та надійну роботу системи для автоматизації організації подій та управління інформацією. Система відповідає основним вимогам технічного завдання, включаючи підтримку кросплатформності, масштабованість і безпеку. Також були сформовані рекомендації щодо подальшого вдосконалення застосунку, що дозволить створити ще більш універсальне та потужне рішення для сучасних потреб управління часом та інформацією.

## СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Python: The most popular language [Електронний ресурс]. – Режим доступу: <https://datascientest.com/en/python-the-most-popular-language> – (8.12.2024).
2. PostgreSQL [Електронний ресурс]. – Режим доступу: <https://www.postgresql.org/> – (8.12.2024).
3. ACID Transactions [Електронний ресурс]. – Режим доступу: <https://www.databricks.com/glossary/acid-transactions> – (8.12.2024).
4. Welcome to Flask [Електронний ресурс]. – Режим доступу: <https://flask.palletsprojects.com/en/stable/> – (8.12.2024).
5. Google Calendar [Електронний ресурс]. – Режим доступу: <https://developers.google.com/calendar> – (8.12.2024).
6. Alembic Docs [Електронний ресурс]. – Режим доступу: <https://alembic.sqlalchemy.org/en/latest/> – (8.12.2024).
7. How to Perform Flask-SQLAlchemy Migrations Using Flask-Migrate [Електронний ресурс]. – Режим доступу: <https://www.digitalocean.com/community/tutorials/how-to-perform-flask-sqlalchemy-migrations-using-flask-migrate> – (9.10.2024).
8. How To Add Authentication to Your App with Flask-Login [Електронний ресурс]. – Режим доступу: <https://www.digitalocean.com/community/tutorials/how-to-add-authentication-to-your-app-with-flask-login> – (8.12.2024).
9. Mailjet [Електронний ресурс]. – Режим доступу: <https://www.mailjet.com/> – (10.12.2024).

**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:

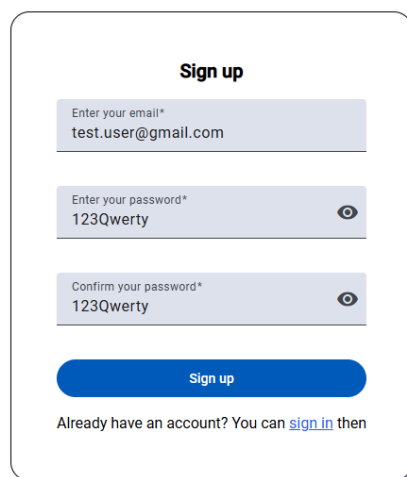
- створення нотатки
- відображення нотаток
- читання нотатки
- створення події
- відображення події на сьогодні/на тиждень/на наступний тиждень
- видалення події

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 within a rounded rectangular container. 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 positioned below the fields. At the bottom, a link reads 'Already have an account? You can [sign in](#) then'.

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.

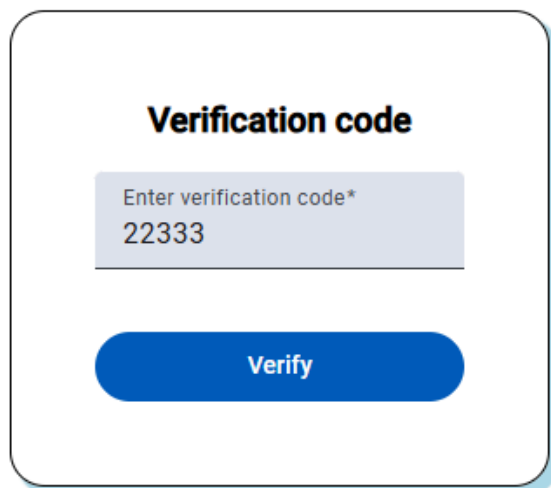
A rectangular form with rounded corners and a light blue border. At the top, the text "Verification code" is centered in bold. Below it is a light blue input field containing the placeholder text "Enter verification code\*" and the value "22333". At the bottom of the form is a blue button with the text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

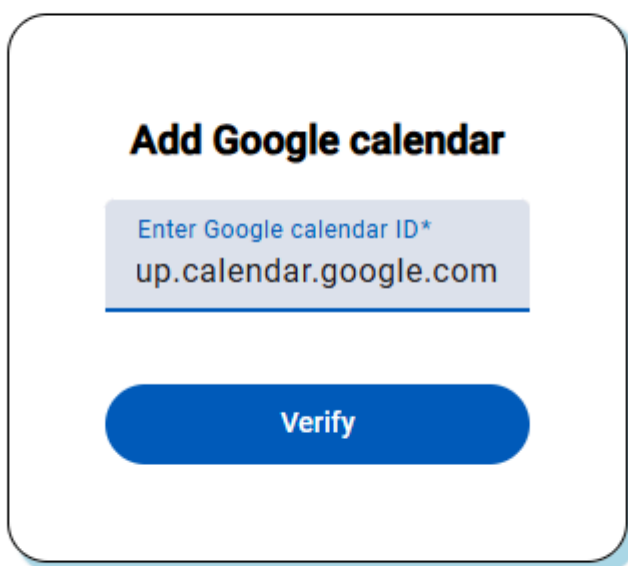
A rectangular form with rounded corners and a light blue border. At the top, the text "Add Google calendar" is centered in bold. Below it is a light blue input field containing the placeholder text "Enter Google calendar ID\*" and the value "up.calendar.google.com". At the bottom of the form is a blue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

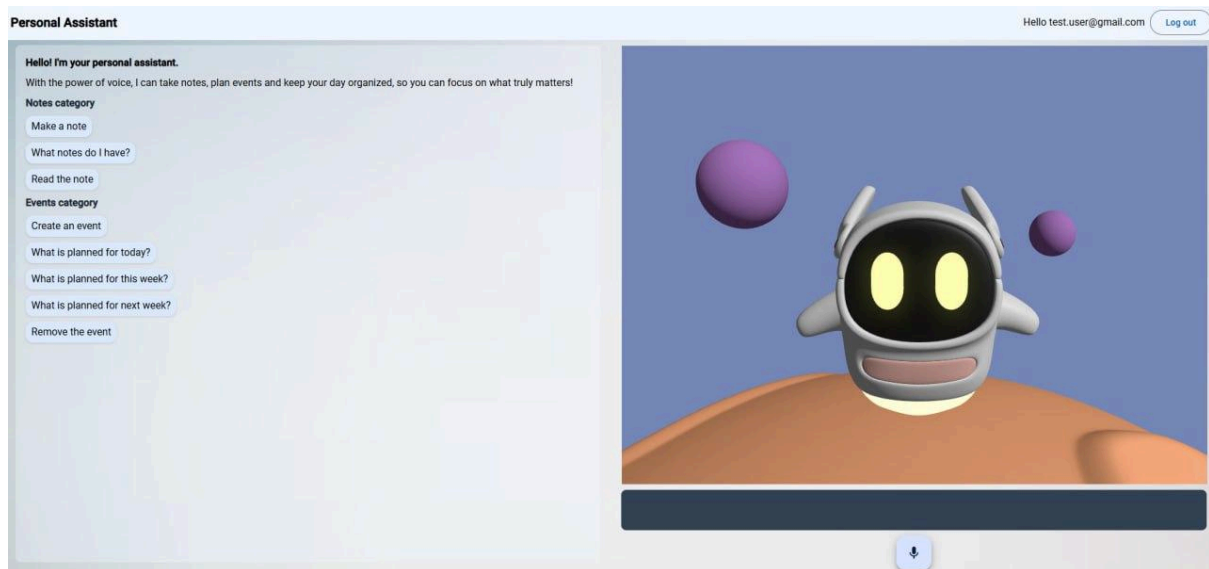


Рисунок 4 – Головна сторінка системи

### 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.

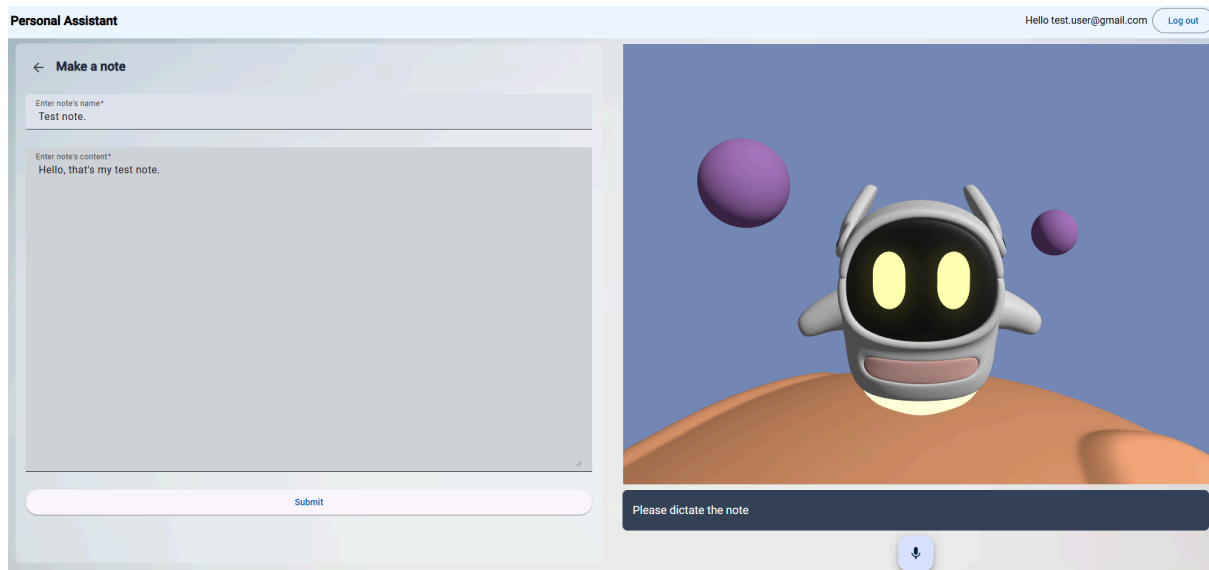


Рисунок 5 – Форма створення нотатки

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

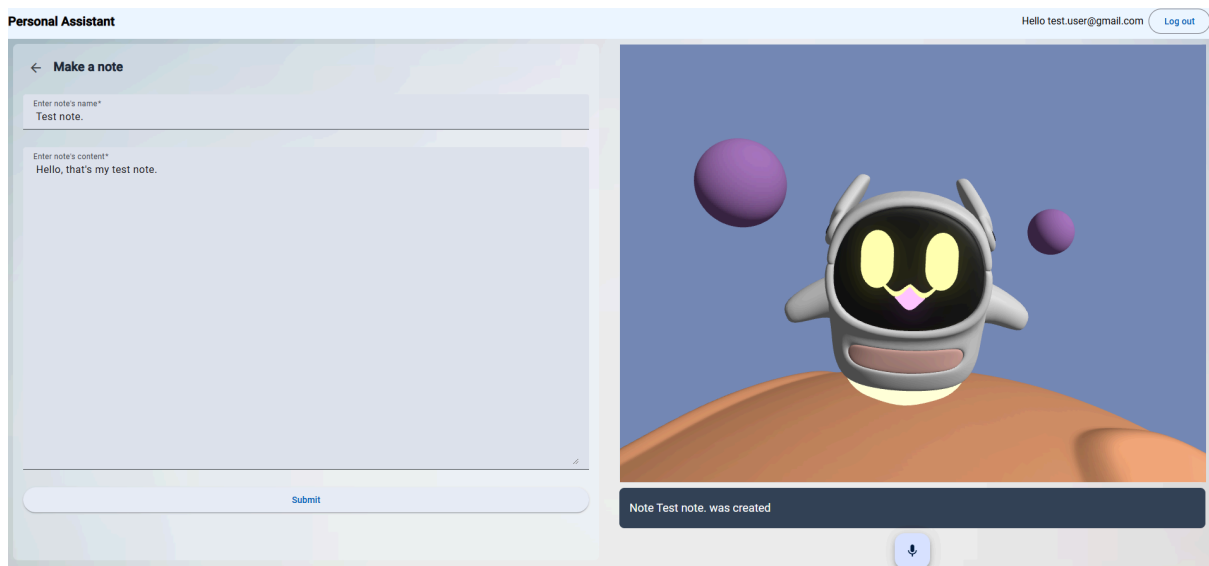


Рисунок 6 – Сповіщення аватара про успішне створення нотатки

### 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.

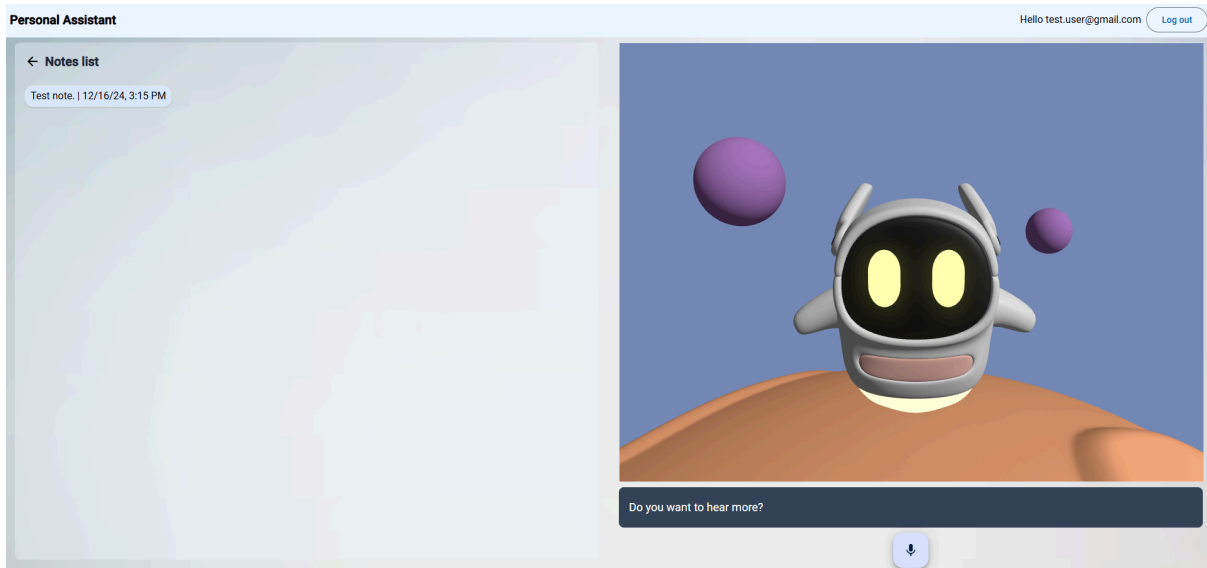


Рисунок 7 – Меню перегляду нотаток

### 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).

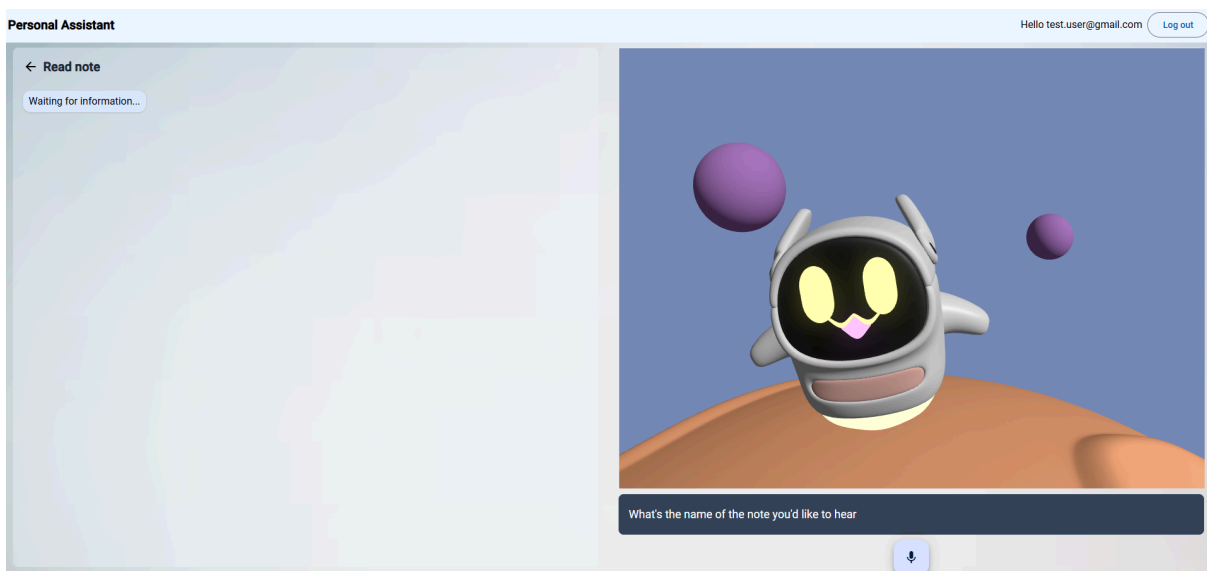


Рисунок 8 – Меню очікування назви нотатки

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

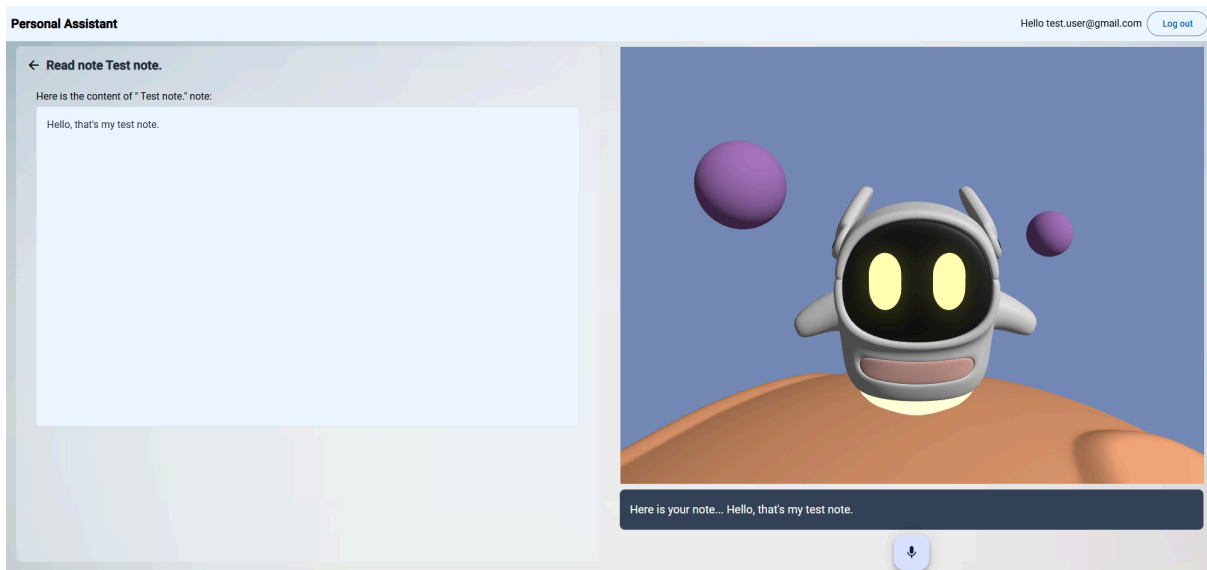


Рисунок 9 – Вікно з текстом нотатки

## 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).

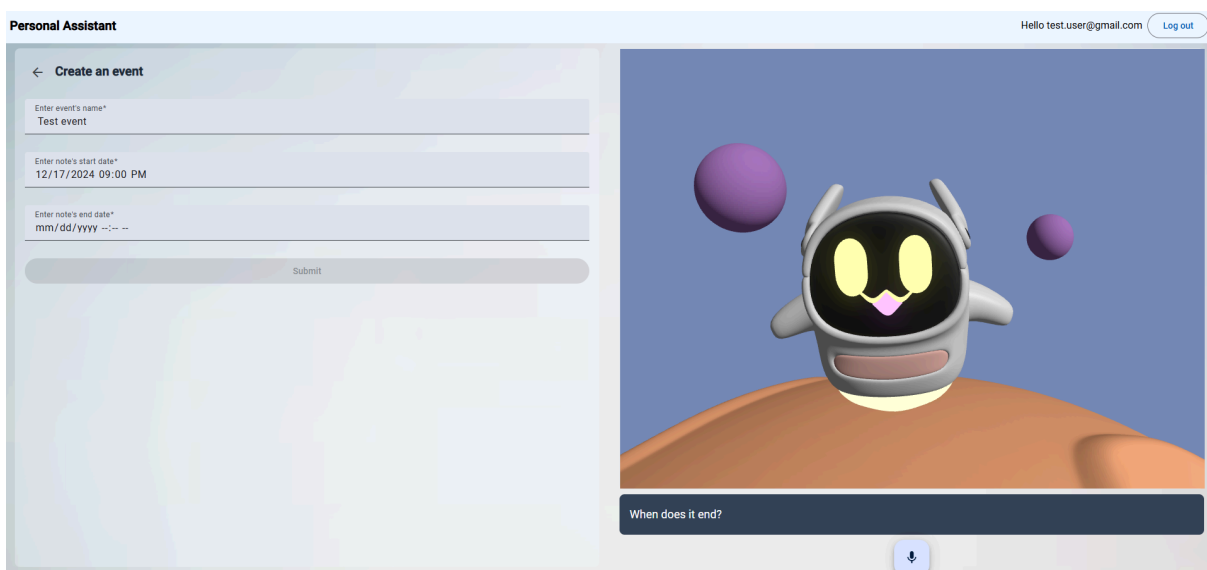


Рисунок 10 – Вікно створення події

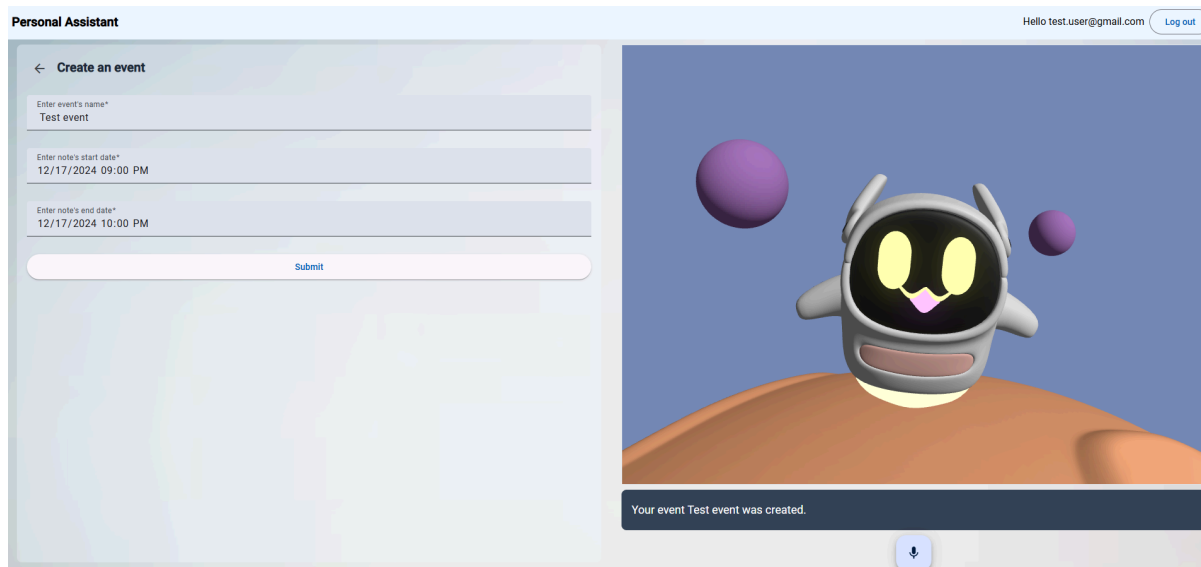


Рисунок 11 – Активна кнопка “Submit”

#### 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.

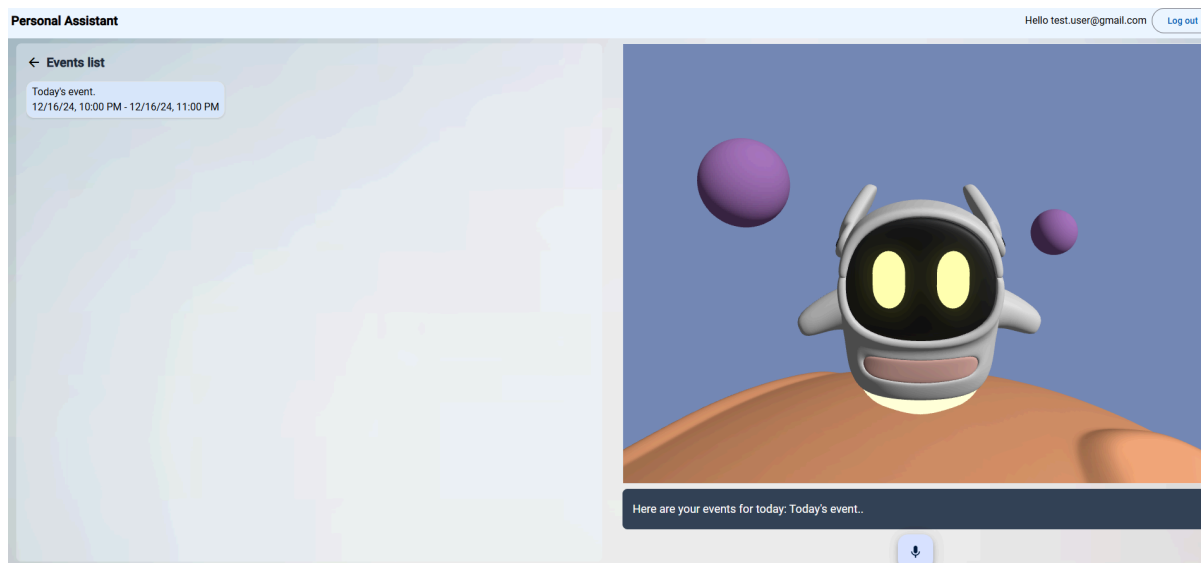


Рисунок 12 – Події, якщо в команді вказати “today”

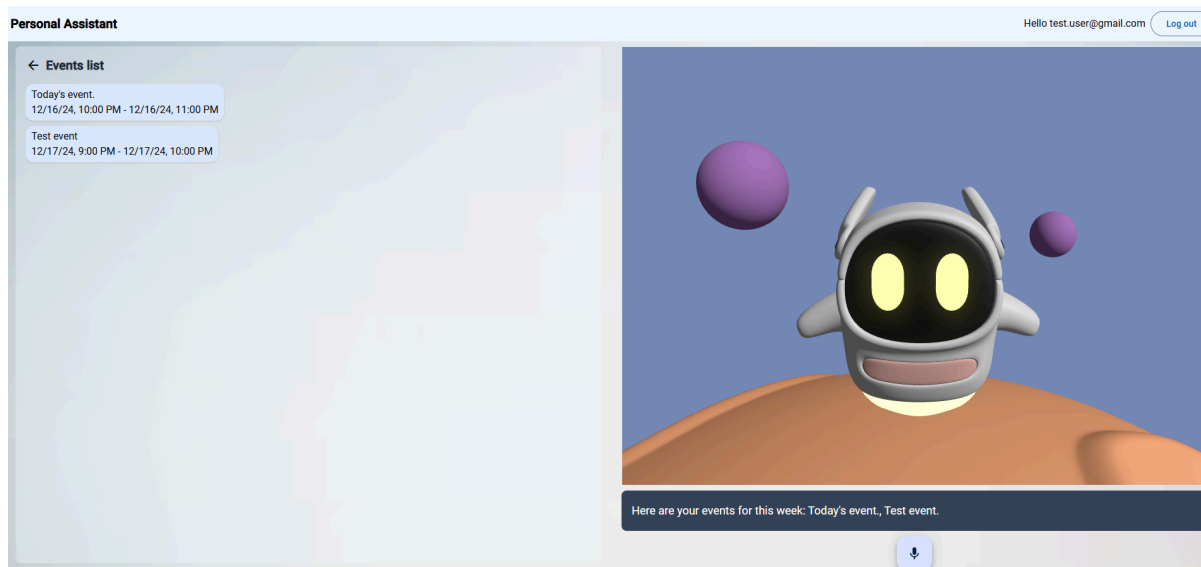


Рисунок 13 – Події, якщо в команді вказати “this week”

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

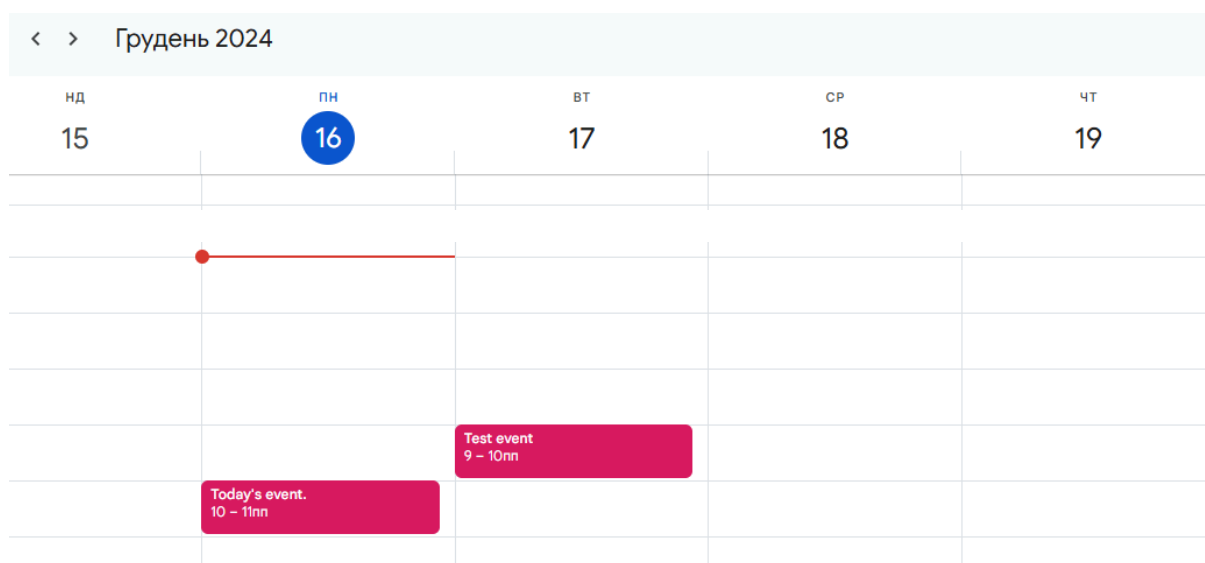


Рисунок 14 – Відображення подій в Google Calendar

#### 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.

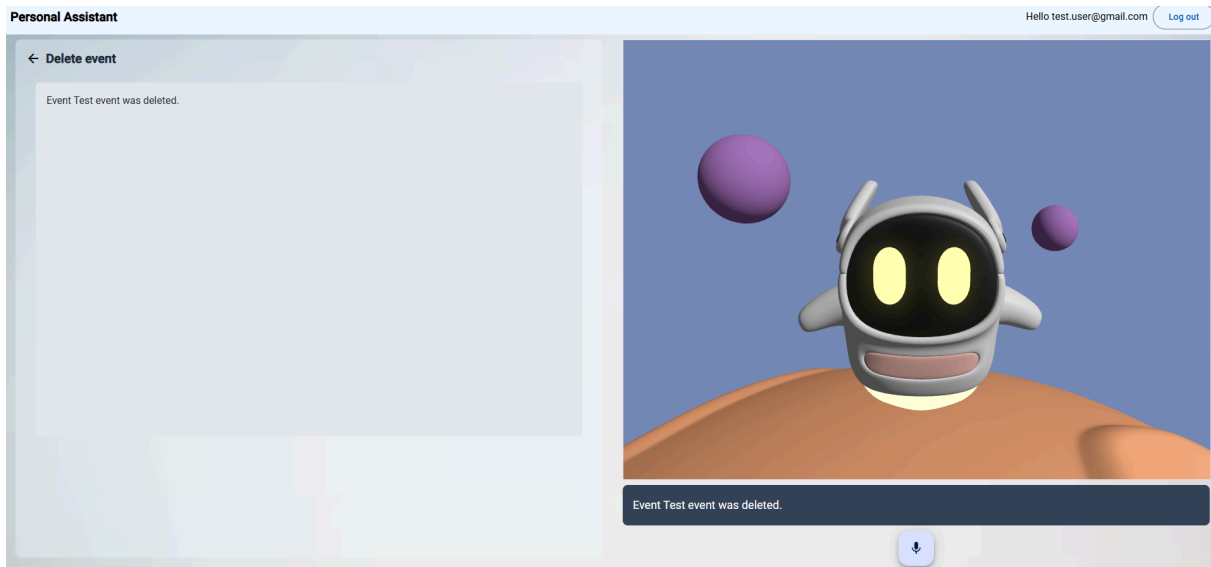


Рисунок 15 – Відображення видалення події

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

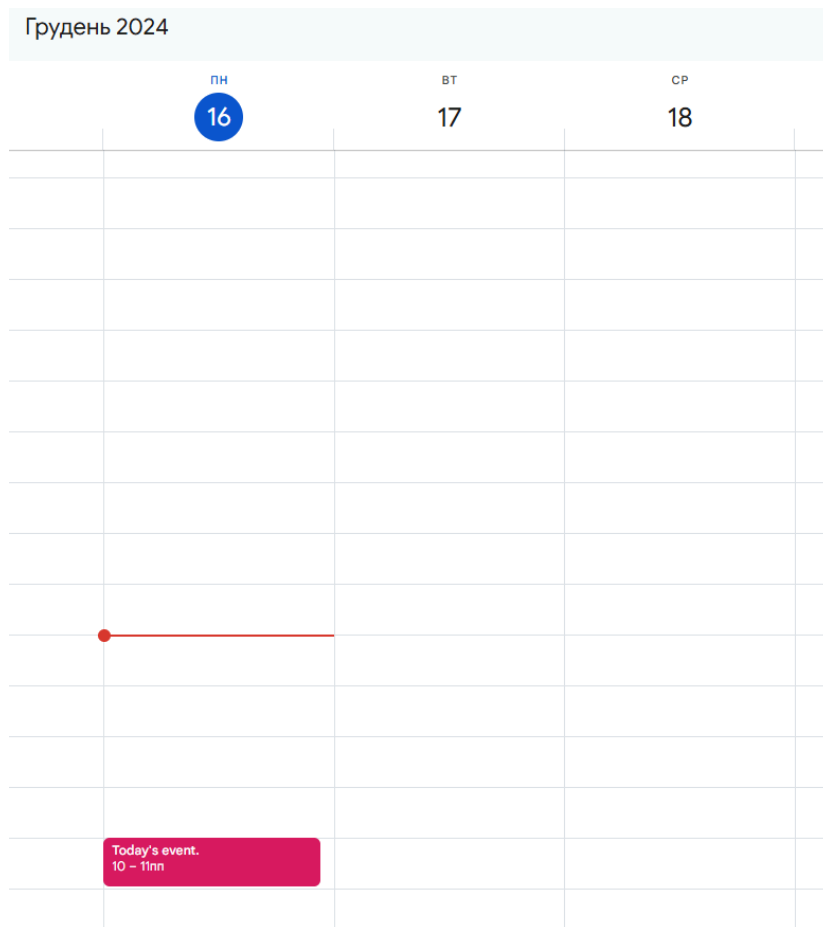


Рисунок 15 – Відображення подій в Google Calendar