МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інтелектуальних
інформаційних систем, д-р техн. наук, проф.
_____Ю. П. Кондратенко
«____»_____2024 р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

СИСТЕМА ПІДТРИМКИ КЛІЄНТІВ ТУРИСТИЧНОЇ ФІРМИ НА ОСНОВІ ГЕНЕРАТИВНОГО ШТУЧНОГО ІНТЕЛЕКТУ

Спеціальність 122 «Комп'ютерні науки»

122 – КРБ – 401. 22010129

Виконав студент 4-го курсу, групи 401
_____ Д. Хаммуд
«_20_» червня 2024 р.

Керівник: канд. пед. наук, доцент
_____Н. М. Болюбаш
«_20_» червня 2024 р.

Миколаїв – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет ім. Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Рівень вищої освіти **бакалавр**
Спеціальність **122 «Комп'ютерні науки»**
(шифр і назва)
Галузь знань **12 «Інформаційні технології»**
(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних
інформаційних систем, д-р. техн. наук, проф.
_____ Ю. П. Кондратенко
«____» _____2024 р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи

Видано студенту групи 401 факультету комп'ютерних наук Хаммуд Даніелю.

1.Тема кваліфікаційної роботи «Система підтримки клієнтів туристичної фірми на основі генеративного штучного інтелекту».

Керівник роботи Болюбаш Надія Миколаївна, канд. пед. наук, доцент.

Затв. наказом Ректора ЧНУ ім. Петра Могили від «28» грудня 2024 р. № _271_

2. Строк представлення кваліфікаційної роботи студентом «20» червня 2024 р.

3. Вхідні (початкові) дані до роботи: предметна сфера туристичної індустрії та інноваційні технології штучного інтелекту, набір даних з туристичної галузі, API ChatGPT (модель GPT-4).

Очікуваний результат: система підтримки клієнтів туристичної фірми на основі генеративного штучного інтелекту.

4. Перелік питань, що підлягають розробці (зміст пояснювальної записки):

– здійснення аналізу сфери туристичної індустрії та дослідження теоретичних засад застосування ШІ для підтримки клієнтів;

– обґрунтування вибору технологій і засобів розробки системи підтримки клієнтів на основі генеративного штучного інтелекту;

– розробка та здійснення програмної реалізації системи підтримки клієнтів туристичної фірми.

5. Перелік графічного матеріалу: презентація, рисунки, таблиці.

6. Завдання до спеціальної частини: Охорона умов праці при розробці системи підтримки клієнтів на основі генеративного ШІ.

7. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис
Спеціальна частина з охорони праці	Алексєєва А. О., канд. техн. наук, доцент кафедри екології	

Керівник роботи __канд. пед. наук, доцент Болюбаш Н. М._____

(наук. ступінь, вчене звання, прізвище та ініціали)

_____

(підпис)

Завдання прийнято до виконання ___Хаммуд. Д._____

(прізвище та ініціали)

_____

(підпис)

Дата видачі завдання «__14__» ___січня__ 2024 р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи бакалавра

Тема: Система підтримки клієнтів туристичної фірми на основі генеративного штучного інтелекту

№	Найменування роботи	Початок	Закінчення	Примітки
1	Визначення керівника і теми КРБ. Подання заяви на затвердження теми КРБ	10.11.2023	15.11.2023	Виконано
2	Отримання завдання на виконання КРБ	10.01.2024	15.01.2024	Виконано
3	Складання календарного плану	16.01.2024	29.01.2024	Виконано
4	Огляд літератури за темою дослідження. Аналіз сучасних методів генеративного ШІ для їх використання у туристичній галузі	30.01.2024	17.02.2024	Виконано
5	Вибір технологій та інструментальних засобів розробки системи	18.02.2024	29.02.2024	Виконано
6	Створення дизайну, проєктування та програмна реалізація, тестування	1.03.2024	15.04.2024	Виконано
7	Робота над розділами фахової частини КРБ	16.04.2024	31.04.2024	Виконано
8	Проходження переддипломної практики, збір та аналіз матеріалів, остаточне оформлення розділів фахової частини КРБ	29.04.2024	12.05.2024	Виконано
9	Розробка спеціальної частини з охорони праці	13.05.2024	25.05.2024	Виконано
10	Обговорення отриманих результатів з керівником та попередній захист КРБ	27.05.2024	29.05.2024	Виконано
11	Корегування роботи за результатами попереднього захисту	30.05.2024	6.06.2024	Виконано
12	Другий попередній захист КРБ	10.06.2024	10.06.2024	Виконано
13	Остаточне оформлення пояснювальної записки та слайдів доповіді до захисту	11.06.2024	12.06.2024	Виконано
14	Подання рецензенту та рецензування КРБ	13.06.2024	13.06.2024	Виконано
15	Подання КРБ, її електронної копії та інших документів (відгуку, рецензії) до захисту	17.06.2024	21.06.2024	Виконано
16	Захист КРБ перед ЕК	28.06.2024	28.06.2024	Виконано

Розробив студент _____Хаммуд Д._____ _____
(прізвище та ініціали) *(підпис)*

Керівник _____канд. пед. наук, доцент Болюбаш Н. М___ _____
(наук. ступінь, вчене звання, прізвище та ініціали) *(підпис)*

«_29_» _____01_____ 2024 р.

АНОТАЦІЯ

кваліфікаційної роботи бакалавра
студента групи 401 ЧНУ ім. Петра Могили
Хаммуд Даніеля
Тема: «Система підтримки клієнтів туристичної фірми на основі генеративного штучного інтелекту»

Дана кваліфікаційна робота бакалавра спрямована на розробку та здійснення програмної реалізації системи підтримки клієнтів туристичної фірми при виборі країни для подорожі. Що є актуальним в умовах цифровізації туристичної індустрії, яка супроводжується підвищенням конкурентоспроможності серед туристичних операторів. Оскільки використання інтелектуальних систем у туристичному бізнесі спрямоване на персоналізоване обслуговування та емоційну підтримку клієнтів, що є ключовими факторами успіху їх діяльності.

Об'єкт роботи – процес підтримки клієнтів туристичної фірми.

Предмет роботи – програмні засоби та методи штучного інтелекту для підтримки клієнтів туристичної фірми.

Мета роботи – підвищення ефективності роботи туристичної фірми шляхом розробки системи підтримки клієнтів на основі методів генеративного штучного інтелекту.

Структура кваліфікаційна робота включає фахову та спеціальну частину з охорони праці. Фахова частина включає вступ, три розділи, висновки та додатки. У першому розділі розкрито теоретичні аспекти аналізу сфери туристичної індустрії. У другому розділі обґрунтовано вибір технологій і засобів розробки системи. У третьому розділі описано проєктування та програмну реалізацію системи підтримки клієнтів туристичної фірми на основі генеративного ШІ.

Бакалаврська кваліфікаційна робота містить 60 сторінок (без додатків), 33 рисунки, 36 джерел та 2 додатки.

Ключові слова: генеративний штучний інтелект, обробка природної мови, персоналізовані рекомендації, туристична галузь, віртуальний гід.

ABSTRACT

bachelor's qualification work
of a student of 401 group at Petro Mohyla Black Sea National University
Hammoud Daniel's
Theme: «Customer support system of a travel firm based on generative artificial intelligence»

This bachelor's qualification work is aimed at the development and implementation of the software implementation of the customer support system of the travel firm when choosing a country for travel. What is relevant in the conditions of digitalization of the tourist industry, which is accompanied by increased competitiveness among tourist operators. Since the use of intelligent systems in the tourism business is aimed at personalized service and emotional support of customers, which are key factors for the success of their activities.

Object of work – customer support process of a travel firm.

Subject of work – software and methods of artificial intelligence to support clients of a travel firm.

The purpose of this work is to increasing the efficiency of the travel firm by developing a customer support system based on methods of generative artificial intelligence.

The structure of the bachelor's work includes a professional and special part on labor protection. The professional part includes an introduction, three chapters, conclusions and appendices. The first chapter reveals the theoretical aspects of the analysis of the tourism industry. In the second section, the choice of technologies and means of system development is substantiated. The third chapter describes the design and software implementation of a customer support system for a travel firm based on generative AI.

The bachelor qualification work contains 60 pages (without appendices), 33 figures, 36 sources and 2 appendices.

Keywords: generative artificial intelligence, natural language processing, personalized recommendations, tourism industry, virtual guide.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ШІ – штучний інтелект

AI – Artificial Intelligence

API – Application Programming Interface

NLP – Natural Language Processing

GPU – Graphics Processing Unit

ML – Machine Learning

GPT – Generative Pre-trained Transformer

RNN – Recurrent Neural Network

BERT – Bidirectional Encoder Representations from Transformers

API – Application Programming Interface

CRM – Customer Relationship Management

UI – User Interface

IDE – Integrated Development Environment

HCI – Human-Computer Interaction

ВСТУП

Актуальність. У сучасному світі туристична індустрія зазнає значних змін під впливом стрімкого розвитку цифрових технологій та штучного інтелекту. Серед туристичних провайдерів гострою є проблема пошуку ефективних шляхів підвищення власної конкурентоспроможності серед аналогічних підприємств. Використання передових інформаційних технологій надає широкий діапазон можливостей для реалізації та забезпечення туристичної діяльності. Перспективним напрямом підвищення ефективності туристичного бізнесу є використання генеративного штучного інтелекту для забезпечення персоналізованого підходу до клієнтів, розуміння їх емоційного стану клієнтів та надання відповідної підтримки з використанням сучасних моделей обробки природної мови, таких як GPT-4.

Дослідження сучасного стану туристичного бізнесу дозволило виявити різноманітні програмні засоби із використанням елементів штучного інтелекту: інтелектуального мобільного та вебсервісу, інтелектуальних систем для підтримки прийняття рішень. Однак виявлення можливостей використання генеративного ШІ для підвищення ефективності туристичної діяльності та забезпечення персоналізованого підходу до клієнтів не є дослідженим у повній мірі.

Це обумовило **мету роботи**, яка полягає у підвищенні ефективності роботи туристичної фірми шляхом розробки системи підтримки клієнтів на основі методів генеративного штучного інтелекту.

Відповідно до поставленої мети було сформульовано **завдання**:

– здійснити аналіз сфери туристичної індустрії та дослідити теоретичні засади застосування ШІ для підтримки клієнтів;

– обґрунтувати вибір технологій та засобів розробки системи підтримки клієнтів на основі генеративного штучного інтелекту;

– розробити та здійснити програмну реалізацію системи підтримки клієнтів туристичної фірми.

Об’єкт роботи – процес підтримки клієнтів туристичної фірми.

Предмет роботи – програмні засоби та методи штучного інтелекту для підтримки клієнтів туристичної фірми.

Методологічною основою дослідження є загальнонаукові та статистично-аналітичні методи, які дозволили комплексно вивчити предмет та об’єкт дослідження, дослідити основні підходи до застосування генеративного ШІ для підтримки клієнтів туристичних фірм.

Практичне значення отриманих результатів полягає в тому, що використання розробленої системи дозволить підвищити ефективність роботи підприємств у сфері туристичного бізнесу.

1 АНАЛІЗ СФЕРИ ДІЯЛЬНОСТІ ТУРИСТИЧНИХ ФІРМ

1.1 Туристична індустрія в епоху цифрових технологій

Сучасна туристична індустрія переживає справжню революцію, зумовлену стрімким розвитком цифрових технологій [1]. Інтернет став невід'ємною частиною життя більшості людей, і вони все частіше використовують його для пошуку інформації про туристичні напрямки, бронювання готелів, купівлі квитків [2], читання відгуків інших мандрівників та обміну досвідом у соціальних мережах.

Уявіть собі світ, де планування відпустки перетворюється на захопливу пригоду, а пошук ідеального туру стає легким та інтуїтивним. У цьому світі віртуальний гід, втілений у вигляді чарівної аніме дівчини, зустрічає вас з посмішкою та готовий відповісти на будь-які запитання. Це не просто чат-бот, а справжній компаньйон у світі подорожей, який розуміє ваші бажання та допомагає знайти найкращі варіанти відпочинку, враховуючи ваші індивідуальні вподобання, бюджет та стиль подорожі.

Однак, величезний потік інформації може легко заплутати навіть досвідченого мандрівника. Вибір оптимального туру серед тисяч пропозицій, пошук готелю, який відповідає всім вимогам, та планування маршруту з урахуванням інтересів усіх учасників подорожі – все це вимагає чимало часу та зусиль. Крім того, кожен турист має свої унікальні потреби та вподобання, які важко врахувати за допомогою стандартних пошукових фільтрів та рекомендацій.

Саме тут на допомогу приходить штучний інтелект (ШІ), який здатен аналізувати величезні обсяги даних, виявляти закономірності та надавати персоналізовані рекомендації [3]. ШІ вже активно використовується в багатьох сферах, і туристична індустрія не є винятком.

Чат-боти, що працюють на основі ШІ, допомагають туристам знайти відповіді на поширені запитання, забронювати готель чи купити квиток. Голосові помічники дозволяють здійснювати пошук інформації та бронювання без

використання рук, що особливо зручно під час подорожі. А віртуальні персонажі роблять взаємодію з ШІ більш природною та приємною, додаючи емоційний контекст до спілкування [4].

Проте, більшість існуючих рішень все ще обмежені у своїх можливостях. Вони часто не здатні зрозуміти складні запитання, надати персоналізовані рекомендації, врахувати емоційний стан користувача чи адаптуватися до його індивідуального стилю спілкування [5].

Саме тому ми пропонуємо новий підхід – створення віртуального гіда з характером, який буде не просто інструментом, а справжнім компаньйоном у світі подорожей. Наш віртуальний гід буде втілений у вигляді милої аніме дівчини, яка зможе спілкуватися з користувачами природною мовою, розуміти їхні емоції, враховувати їхні вподобання та надавати персоналізовані рекомендації, ніби це робить близький друг чи досвідчений туристичний агент [6].

Впровадження такого віртуального гіда може принести значні переваги для туристичної фірми:

- *підвищення лояльності клієнтів*: персоналізований підхід, емпатія та можливість отримати допомогу у будь-який час доби сприятимуть підвищенню задоволеності клієнтів та їхньої лояльності до компанії [7]. Клієнти будуть відчувати, що їх цінують та розуміють, що спонукатиме їх повертатися до компанії знову і знову;
- *збільшення продажів*: віртуальний гід зможе пропонувати клієнтам релевантні тури та послуги, враховуючи їхні індивідуальні потреби та вподобання, збільшуючи ймовірність покупки. Крім того, він зможе активно взаємодіяти з клієнтами, пропонуючи їм додаткові послуги та акції, що також сприятиме збільшенню продажів;
- *оптимізація витрат*: автоматизація частини роботи з клієнтами дозволить скоротити витрати на персонал та підвищити ефективність роботи компанії. Віртуальний гід зможе одночасно обслуговувати велику кількість клієнтів [8], звільняючи співробітників від рутинних завдань та дозволяючи їм

зосередитися на більш складних та творчих аспектах роботи;

– *конкурентна перевага*: інноваційне рішення допоможе туристичній фірмі виділитися серед конкурентів та залучити нових клієнтів. Унікальний віртуальний гід з характером стане візитною карткою компанії, привертаючи увагу потенційних клієнтів та створюючи позитивний імідж бренду.

Віртуальні гіди мають великий потенціал для розвитку. У майбутньому вони зможуть не лише допомагати з вибором туру, а й супроводжувати туристів під час подорожі, надаючи їм інформацію про цікаві місця, ресторани та розваги в режимі реального часу. Вони також зможуть допомагати з вирішенням проблем, що виникають під час подорожі, таких як затримка рейсів чи втрата багажу, надаючи актуальну інформацію та підтримку.

Віртуальні гіди здатні змінити наше уявлення про подорожі, зробивши їх більш комфортними, цікавими та доступними. Вони можуть стати не просто помічниками, а справжніми супутниками у подорожі, які зроблять кожну пригоду незабутньою.

1.2 Еволюція ШІ у туристичній індустрії: від чат-ботів до віртуальних компаньйонів

У гонитві за серцями та гаманцями мандрівників, туристичні компанії активно використовують штучний інтелект (ШІ) для покращення обслуговування клієнтів та оптимізації бізнес-процесів. Від простих чат-ботів до складних віртуальних помічників, ШІ-рішення все більше проникають у різні аспекти туристичного досвіду, трансформуючи його та відкриваючи нові можливості.

Чат-боти, мабуть, найпоширеніший приклад використання ШІ у туристичній індустрії. Вони здатні відповідати на поширені запитання, надавати інформацію про тури та готелі, допомагати з бронюванням та навіть вирішувати деякі проблеми клієнтів. Їх переваги:

– *доступність 24/7*: чат-боти працюють цілодобово, забезпечуючи

клієнтам підтримку у будь-який час;

– *швидкість*: вони миттєво відповідають на запитання, економлячи час клієнтів [10];

– *економія ресурсів*: чат-боти здатні обслуговувати велику кількість клієнтів одночасно, зменшуючи навантаження на персонал [9].

Недоліки чат-ботів:

– *обмеженість*: чат-боти часто не розуміють складних запитань або запитів, що вимагають індивідуального підходу;

– *відсутність емпатії*: вони не здатні враховувати емоційний стан клієнта та адаптуватися до його стилю спілкування [11, 12];

– *недостатня персоналізація*: чат-боти не завжди можуть надати рекомендації, що відповідають індивідуальним потребам та вподобанням клієнта.

Голосові помічники, такі як Amazon Alexa, Google Assistant та Apple Siri, пропонують новий рівень взаємодії з ШІ [13]. Вони дозволяють клієнтам здійснювати пошук інформації, бронювати готелі та квитки, отримувати рекомендації та навіть перекладати мову за допомогою голосових команд. Їх переваги:

– *зручність*: голосові помічники звільняють руки клієнтів, що особливо корисно під час подорожі;

– *інтерактивність*: вони дозволяють вести діалог з ШІ, уточнюючи запити та отримуючи більш точні відповіді;

– *персоналізація*: голосові помічники можуть запам'ятовувати вподобання клієнта та надавати рекомендації на основі його історії пошуку.

Недоліки голосових помічників:

– *обмеженість у розумінні природної мови*: голосові помічники все ще можуть мати труднощі з розумінням складних запитів або запитів з нечіткою вимовою;

– *залежність від якості інтернет-з'єднання*: для роботи голосових помічників потрібен стабільний доступ до Інтернету;

– *проблеми з конфіденційністю*: використання голосових помічників може викликати занепокоєння щодо збору та використання персональних даних.

Віртуальні персонажі, такі як Amelia від IPSoft та Replika від Luka, представляють собою новий етап у розвитку ШІ-рішень для туристичної індустрії. Вони поєднують у собі можливості чат-ботів та голосових помічників, додаючи до них візуальний образ та емоційний контекст. Їх переваги:

– *емоційний зв'язок*: віртуальні персонажі здатні викликати емпатію та довіру у клієнтів, створюючи більш особистий та приємний досвід взаємодії;

– *імерсивний досвід*: вони можуть створювати відчуття присутності та занурення у віртуальний світ, що робить спілкування з ШІ більш захопливим та цікавим;

– *адаптивність*: віртуальні персонажі здатні навчатися та адаптуватися до індивідуальних потреб та вподобань клієнта, надаючи більш релевантні рекомендації та підтримку.

Недоліки віртуальних персонажів:

– *складність розробки*: створення реалістичних та переконливих віртуальних персонажів вимагає значних ресурсів та експертизи у галузі ШІ, анімації та психології;

– *"ефект моторошної долини"*: якщо віртуальний персонаж виглядає занадто реалістичним, але при цьому рухається або говорить не природно, це може викликати у користувачів неприємні відчуття;

– *етичні питання*: використання віртуальних персонажів може піднімати питання щодо маніпуляції та обману, особливо якщо вони використовуються для продажу товарів чи послуг [14].

Наукові дослідження у галузі ШІ та його застосування у туристичній індустрії активно розвиваються. Вчені досліджують різні аспекти взаємодії людини з ШІ, такі як розуміння природної мови, емоційний інтелект, персоналізація та етичні питання. Результати цих досліджень допомагають створювати більш ефективні та етичні ШІ-рішення для туристичної індустрії.

Дослідження у цій сфері ведуться у наступних напрямах:

- *розуміння природної мови*: вчені розробляють алгоритми [15], що дозволяють ШІ краще розуміти людську мову, враховуючи не лише слова, а й контекст, інтонацію та емоції;
- *емоційний інтелект*: дослідники вивчають, як ШІ може розпізнавати та реагувати на емоції користувачів [25], щоб створювати більш емпатичне та підтримуюче спілкування;
- *персоналізація*: вчені розробляють методи [16], що дозволяють ШІ адаптуватися до індивідуальних потреб та вподобань користувачів, надаючи їм більш релевантні рекомендації та послуги;
- *етичні питання*: дослідники вивчають етичні аспекти використання ШІ у туристичній індустрії, такі як прозорість, конфіденційність та відповідальність за дії ШІ.

Аналіз наявних аналогів та публікацій показує, що ШІ має великий потенціал для трансформації туристичної індустрії. Чат-боти, голосові помічники та віртуальні персонажі вже сьогодні допомагають туристам планувати подорожі, бронювати готелі та квитки, отримувати інформацію та підтримку.

Однак, майбутнє ШІ у туризмі лежить у створенні більш персоналізованих та емпатичних рішень, які зможуть розуміти та враховувати індивідуальні потреби та вподобання кожного клієнта, надаючи йому унікальний та незабутній досвід подорожі.

1.3 Постановка задачі

Розробка системи для підтримки клієнтів туристичної дозволяє підвищити ефективність її роботи, сприяє залученню клієнтів та рівня їх задоволеності, що є критично важливим для успішної професійної діяльності.

Об’єкт роботи – процес підтримки клієнтів туристичної фірми.

Предмет роботи – програмні засоби та методи штучного інтелекту для підтримки клієнтів туристичної фірми.

Мета роботи – підвищення ефективності роботи туристичної фірми шляхом розробки системи підтримки клієнтів на основі методів генеративного штучного інтелекту.

Для досягнення поставленої мети було поставлено такі **завдання**:

− здійснити аналіз сфери туристичної індустрії та дослідити теоретичні засади застосування ШІ для підтримки клієнтів;

− обґрунтувати вибір технологій та засобів розробки системи підтримки клієнтів на основі генеративного штучного інтелекту;

− розробити та здійснити програмну реалізацію системи підтримки клієнтів туристичної фірми.

Віртуальний гід «TravelAnime» повинен мати широкий спектр функціональних можливостей, щоб задовольнити потреби навіть найвибагливіших мандрівників. Охарактеризуємо їх детальніше:

− розуміння природної мови: здатність розуміти запити клієнтів, сформульовані природною мовою, з урахуванням контексту, інтонації та емоцій, та генерувати відповіді, які точно відповідають на поставлені запитання;

− надання інформації про туристичні послуги: можливість надавати клієнтам вичерпну та актуальну інформацію про тури, готелі, ціни, транспорт, візові вимоги, місцеві пам'ятки, ресторани, розваги та інші аспекти подорожі, спираючись на дані з внутрішніх систем туристичної фірми та зовнішніх джерел;

− персоналізовані рекомендації: здатність аналізувати вподобання клієнта, його історію подорожей, бюджет та інші фактори, щоб надавати персоналізовані рекомендації щодо вибору туру, готелю, маршруту та інших аспектів подорожі, які максимально відповідатимуть його індивідуальним потребам та бажанням;

− емоційна підтримка: здатність розпізнавати емоційний стан клієнта та реагувати на нього відповідним чином, надаючи підтримку, заспокоюючи або підбадьорюючи його. Наприклад, якщо клієнт висловлює невпевненість у виборі

туру, віртуальний гід може запропонувати кілька альтернативних варіантів, підкресливши їх переваги та недоліки, а також висловити підтримку та запевнити клієнта, що він зробить правильний вибір;

− допомога у вирішенні проблем: здатність допомагати клієнтам у вирішенні проблем, що виникають під час планування або здійснення подорожі, таких як затримка рейсів, втрата багажу, зміна бронювання готелю тощо. Віртуальний гід може надати клієнту актуальну інформацію, поради та контакти необхідних служб, а також запропонувати альтернативні варіанти вирішення проблеми;

− розважальний контент: можливість надавати клієнтам цікавий та пізнавальний контент, пов'язаний з подорожами, такий як статті, відео, фотографії, вікторини, ігри тощо. Це дозволить не лише розважити клієнтів, а й розширити їх кругозір та підвищити інтерес до подорожей;

− інтеграція з соціальними мережами: можливість інтеграції з популярними соціальними мережами, такими як Facebook, Instagram, Twitter тощо, що дозволить клієнтам ділитися своїми враженнями від подорожі, отримувати рекомендації від друзів та знайомитися з іншими мандрівниками;

− багатомовність: підтримка декількох мов, що дозволить віртуальному гіду спілкуватися з клієнтами з різних країн світу та розширити географію обслуговування туристичної фірми.

Окрім функціональних вимог, система «TravelAnime» повинна відповідати ряду перерахованих нижче нефункціональних вимог, які забезпечать її високу якість, надійність та безпеку.

1. Продуктивність: система повинна працювати швидко та ефективно, забезпечуючи миттєву відповідь на запити клієнтів та плавну анімацію віртуального персонажа. Це особливо важливо при обслуговуванні великої кількості клієнтів одночасно.

2. Масштабованість: система повинна бути спроектована таким чином, щоб її можна було легко масштабувати у разі збільшення кількості користувачів або

розширення функціональності. Це дозволить туристичній фірмі адаптуватися до зростання бізнесу та забезпечити високу якість обслуговування навіть при пікових навантаженнях.

3. Надійність: система повинна бути стійкою до збоїв та помилок, забезпечуючи безперебійну роботу та мінімізуючи ризик втрати даних. Це особливо важливо для туристичної фірми, оскільки будь-які проблеми з системою можуть призвести до негативних наслідків для клієнтів та репутації компанії;

4. Безпека: система повинна забезпечувати захист персональних даних клієнтів від несанкціонованого доступу, витоку або втрати. Це включає в себе використання шифрування даних, контроль доступу, регулярне оновлення програмного забезпечення та інші заходи безпеки.

5. Зручність використання: інтерфейс користувача повинен бути інтуїтивно зрозумілим та зручним у використанні, навіть для тих, хто не має досвіду спілкування з віртуальними персонажами. Це дозволить клієнтам швидко освоїти систему та отримувати від неї максимум користі.

6. Сумісність: система повинна бути сумісною з різними пристроями та платформами, такими як комп'ютери, смартфони, планшети тощо, а також з різними веб-браузерами та операційними системами. Це забезпечить доступність системи для широкої аудиторії та дозволить клієнтам користуватися нею у будь-який час та в будь-якому місці.

Висновки до розділу 1

У першому розділі було проведено аналіз предметної сфери, розглянуто сучасні тенденції у туристичній індустрії та досліджено потенціал використання штучного інтелекту для покращення взаємодії з клієнтами. Було проаналізовано існуючі аналоги систем підтримки клієнтів на основі ШІ, виявлено їх переваги та недоліки. На основі проведеного аналізу сформульовано мету та завдання проекту «TravelAnime», визначено основні вимоги до системи. Проєкт «TravelAnime» має

значний потенціал для трансформації туристичного досвіду, надаючи клієнтам персоналізовану та емоційну підтримку у вигляді віртуального гіда з унікальним характером. Успішна реалізація проекту дозволить туристичній фірмі не лише покращити якість обслуговування клієнтів, а й підвищити ефективність бізнес-процесів, збільшити продажі та зміцнити конкурентну позицію на ринку [17].

2 ТЕХНОЛОГІЇ ТА ЗАСОБИ РОЗРОБКИ СИСТЕМИ НА ОСНОВІ ГЕНЕРАТИВНОГО ШТУЧНОГО ІНТЕЛЕКТУ

2.1 Методи та моделі для створення віртуального помічника у туристичній галузі

Для розробки системи підтримки клієнтів туристичної фірми було проаналізовано сучасні технології генеративного штучного інтелекту для створення віртуального помічника [18]. Основні підходи включають застосування методів обробки природної мови та машинного навчання і генеративні моделі ШІ [19]. Розглянемо їх більш детально.

1. *Обробка природної мови* (англ. Natural language processing, NLP). Техноогія NLP для обробки природної мови є галуззю штучного інтелекту, що займається взаємодією між комп'ютерами та людьми людською мовою [20]. Метою NLP є розуміння, інтерпретація та генерування природної мови, такої як текст або мова, що дозволяє комп'ютерам ефективно обробляти і розуміти людську мову.

Розвиток NLP почався ще в середині 20-го століття з простих систем, які могли виконувати синтаксичний аналіз тексту. Перші системи обробки природної мови були здебільшого засновані на правилах і мали обмежені можливості. Наприклад, в 1950-х роках були створені перші програми машинного перекладу, які працювали за допомогою правил граматики та словників [21]. У 1980-х і 1990-х роках з розвитком комп'ютерних технологій і методів машинного навчання, NLP перейшло до статистичних методів. Використання великих обсягів тексту для навчання моделей дозволило досягти значного прогресу в розумінні та генерації мови.

В останні десятиліття з появою глибокого навчання NLP досягло нових висот. Нейронні мережі, такі як рекурентні нейронні мережі (англ. Recurrent Neural Network, RNN) та трансформери, революціонізували обробку природної мови. Сьогодні, моделі на основі трансформерів, такі як BERT і GPT, можуть виконувати

завдання з високою точністю, що робить їх ідеальними для системи підтримки клієнтів у туристичній індустрії.

Основні переваги NLP:

− розуміння користувача: NLP дозволяє системі розуміти запити користувачів на природній мові, що забезпечує більш природну та зручну взаємодію;

− персоналізація: аналіз тексту дозволяє створювати персоналізовані відповіді та рекомендації, що підвищує задоволеність клієнтів;

− автоматизація: використання NLP для автоматизації обробки запитів знижує навантаження на персонал та підвищує ефективність роботи системи.

2. *Генеративні моделі*. Генеративні моделі є підходом у машинному навчанні, який дозволяє створювати нові дані на основі вхідних даних [21]. Є різновидом технології штучного інтелекту, який може створювати різні типи контенту, включаючи текст, зображення, аудіо та синтетичні дані. Різницю між ChatGPT і найкращими останніми рішеннями NLP зображено на рисунку 2.1.

Генеративні моделі пройшли довгий шлях від простих статистичних методів до сучасних глибоких нейронних мереж [22]. Перші генеративні моделі базувалися на ймовірнісних розподілах і використовували прості підходи для генерації тексту.

З появою глибокого навчання у 2010-х роках, генеративні моделі стали набагато потужнішими. Поява моделей, таких як автоенкодери та варіаційні автоенкодери, дозволила створювати більш складні та реалістичні тексти.

Одним із ключових проривів стало впровадження трансформерів, що привело до створення моделей GPT (рис. 2.1) (англ. Generative Pre-trained Transformer). Трансформер (англ. Transformer) є архітектурою глибоких нейронних мереж, яка за аналогією з рекурентними нейронними мережами призначена для обробки послідовностей, таких як текст природною мовою, і вирішення таких завдань як машинний переклад та автоматичне реферування.

GPT-3, випущений у 2020 році, став однією з найбільш потужних моделей, здатних генерувати тексти на рівні людини. Наступник, GPT-4, продовжив цю

традицію, забезпечуючи ще кращу якість генерації тексту та можливість навчання на ще більшій кількості даних.

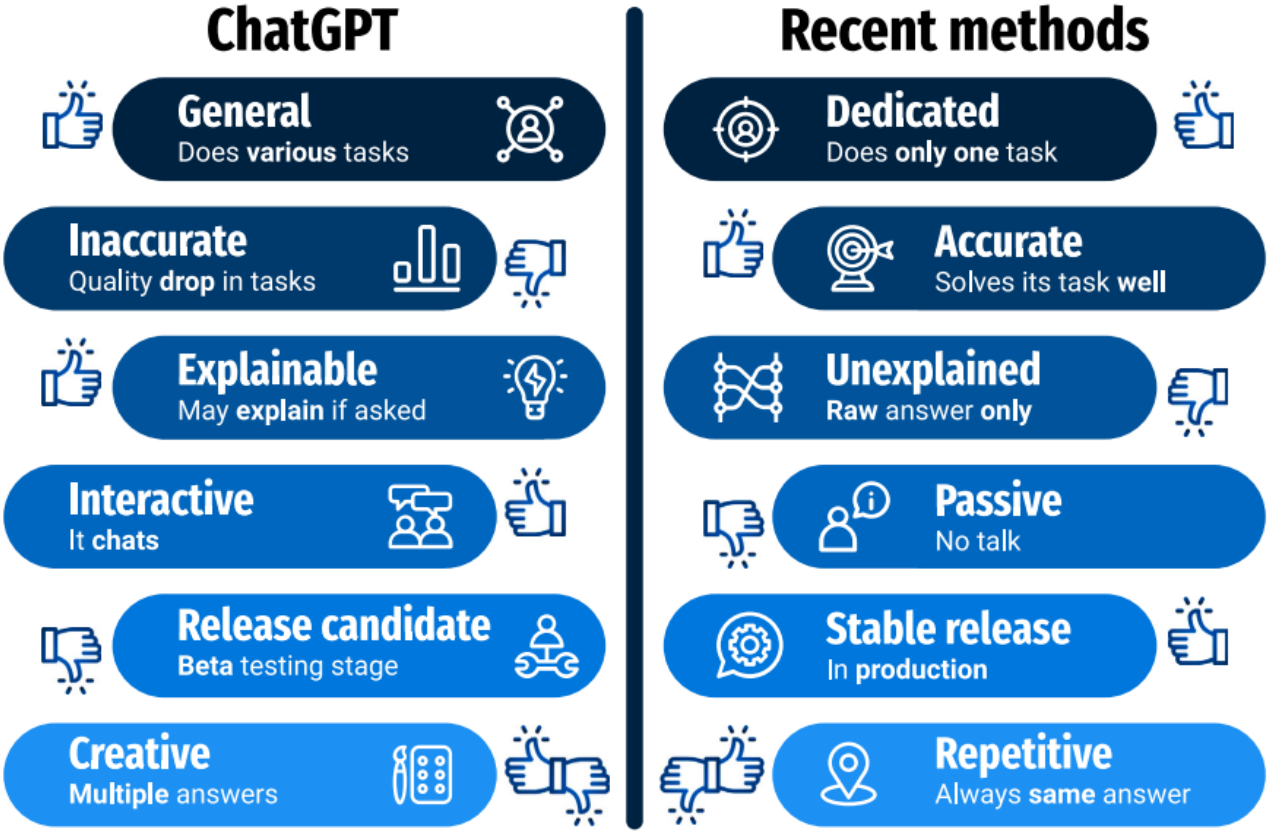


Рисунок 2.1 – Різниця між ChatGPT і останніми рішеннями NLP [24]

На рисунку 2.2 відображено еволюцію генеративних моделей: 1) базова модель Transformer; 2) перша версія моделі Generative Pre-Training GPT-1; 3) друга версія GPT-2; 4) версія GPT-3; 5) InstructGPT на основі відгуків людини; 6) ChatGPT — взаємодіє у розмовному режимі, навчена на великій кількості людських відгуків; 7) GPT-4 — великомасштабна мультимодальна модель із текстом та/або зображенням як вхідні дані.

Однією з найпопулярніших і потужних генеративних моделей генеративного ШІ є GPT-4 (Generative Pre-trained Transformer 4). GPT-4 – це модель глибокого навчання, розроблена компанією OpenAI, яка здатна генерувати тексти, що дуже схожі на ті, що написані людьми. Ця модель навчена на великій кількості текстових

даних і може генерувати зв'язні тексти, відповіді на запити, рекомендації та іншу інформацію.

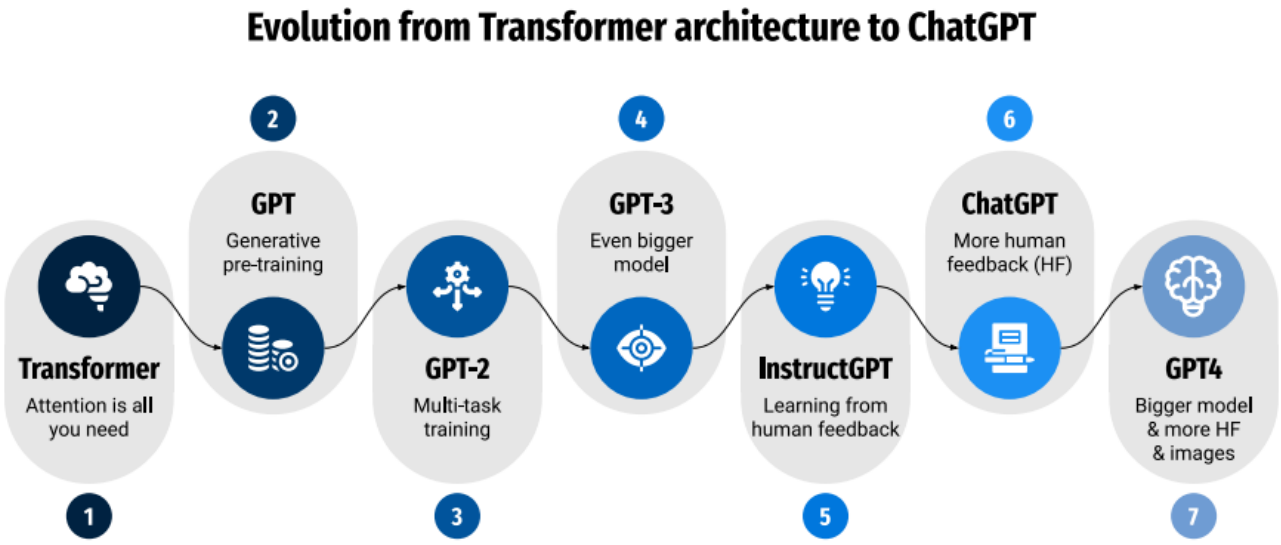


Рисунок 2.2 – Еволюція моделей на основі архітекрути Transformer

Переваги генеративних моделей:

– висока якість текстів: GPT-4 здатен генерувати тексти, що дуже схожі на людські, що робить взаємодію з користувачами більш природною;

– широкий спектр застосувань: генеративні моделі можуть використовуватися для різних завдань, таких як автоматичні відповіді на запити, створення контенту та рекомендації;

– навчання на великій кількості даних: GPT-4 навчена на величезній кількості текстових даних, що дозволяє їй мати широкий кругозір та розуміти різні контексти.

3. *Машинне навчання* (англ. Machine Learning, ML). Машинне навчання – це підгалузь штучного інтелекту, що займається розробкою алгоритмів, які дозволяють комп'ютерам навчатися на основі даних і робити прогнози або приймати рішення без явного програмування [23].

Машинне навчання бере свій початок у середині 20-го століття, коли з’явилися перші алгоритми, здатні навчатися з даних. Один з перших проривів

відбувся у 1957 році, коли Френк Розенблатт створив перцептрон – просту модель нейронної мережі, здатну навчатися розпізнавати образи. У 1980-х роках розвиток машинного навчання отримав новий імпульс з появою методів зворотного поширення помилки для навчання багатошарових нейронних мереж. Це дозволило створювати більш складні моделі та розвивати глибоке навчання.

В останні десятиліття машинне навчання стало невід’ємною частиною багатьох галузей, від обробки зображень до аналізу даних. Сучасні алгоритми, такі як глибокі нейронні мережі та методи ансамблевого навчання, дозволяють досягати високої точності та ефективності.

В основі машинного навчання лежать алгоритми. Сьогодні використовуються два основних типи алгоритмів машинного навчання: контрольоване навчання та самостійне навчання. Різниця полягає у способі вивчення даних для подальшого прогнозування.

Контрольоване машинне навчання. Навчання під контролем найчастіше використовується у випадках, коли дослідник даних виступає як наставник, демонструючи, які результати повинен отримати алгоритм. Для цього типу навчання використовуються такі алгоритми, як лінійна та логістична регресія, мультикласова класифікація та метод опорних векторів.

Самостійне машинне навчання. Самостійне навчання має на увазі велику незалежність: комп'ютер вчиться розпізнавати складні процеси та алгоритми без постійного контролю з боку людини. Такий тип навчання передбачає відсутність маркування даних та конкретних зумовлених результатів. Для цього типу навчання використовуються такі алгоритми, як кластеризація методом k-середніх, аналіз основних та незалежних компонентів та асоціативні правила.

Є також платформи машинного навчання без коду, які використовують візуальні платформи редагування методом перетягування для автоматичної побудови моделей машинного навчання та генерування прогнозів без написання коду. Ці платформи автоматизують процес збору даних, їх очищення, вибір моделі, навчання моделі та розгортання моделі. ML без коду демократизує машинне

навчання. Воно дозволяє бізнес-аналітикам без знання ML та досвіду у програмуванні створювати моделі машинного навчання та генерувати прогнози для вирішення нагальних проблем.

На відміну від машинного навчання без коду при традиційному машинному навчанні досвідчений фахівець із роботи з даними використовує мову програмування, наприклад Python, щоб створити модель ML. Фахівці з роботи з даними повинні імпортувати набори даних та підготувати дані до ML, очищуючи дані з використанням ручних та автоматизованих процесів та застосуванням інженерних технологій. Вони повинні відібрати частину даних, яка буде використовуватися для навчання та налаштування моделі перед її розгортанням у робочому середовищі. Платформа без коду, навпаки, поєднує можливості передового програмування машинного навчання з простими у використанні інструментами, які дозволяють підприємствам створювати моделі машинного навчання.

Переваги машинного навчання:

− адаптивність: машинне навчання дозволяє системі адаптуватися до нових даних і покращувати свою точність з часом;

− автоматизація: автоматичне навчання моделей дозволяє знижувати навантаження на розробників та оптимізувати процес обробки даних;

− широкий спектр застосувань: машинне навчання може бути використане для різних завдань, від аналізу даних до прогнозування та класифікації.

Таким чином, застосування передових технологій обробки природної мови, генеративних моделей та машинного навчання дозволяє створити віртуальний помічник, здатний вести природний та невимушений діалог з клієнтами, розуміючи не лише їхні слова, а й контекст, інтонацію та емоції. Він може підтримати розмову на будь-яку тему, пов’язану з подорожами, від вибору напрямку та бронювання готелю до порад щодо місцевої кухні та культурних особливостей. Унікальне поєднання передових технологій ШІ та привабливого візуального образу сприяє

завоюванню довіри та лояльності клієнтів, збільшенню продажів туристичних послуг та підвищенню загальної задоволеності клієнтів від взаємодії з компанією.

2.2 Інструментальні засоби розробки системи

Для розробки системи було використано сучасні інструментальні засоби, що забезпечують ефективність, надійність та простоту використання.

1. *Мова програмування Python*. Python є однією з найпопулярніших мов програмування для роботи зі штучним інтелектом завдяки своїй простоті та великій кількості бібліотек. Python підтримує багатий екосистемний інструментарій для обробки природної мови, машинного навчання та розробки веб-додатків.

Python був розроблений Гвідо ван Россумом і вперше випущений у 1991 році. Головною метою створення Python було забезпечення простої та зрозумілої мови програмування, яка б мала широкий спектр застосувань. Завдяки своїй простоті та зручності, Python швидко завоював популярність серед розробників. З часом Python став стандартом у багатьох галузях, включаючи науку про дані, штучний інтелект, веб-розробку та автоматизацію. Велика кількість бібліотек і фреймворків, таких як NumPy, Pandas, TensorFlow та PyTorch, зробили Python одним з найпотужніших інструментів для розробки сучасних додатків [26, 27].

Переваги використання Python:

− простота та зрозумілість: Python має простий синтаксис, що робить його зручним для навчання та використання;

− широкий спектр бібліотек: Python підтримує велику кількість бібліотек для машинного навчання, обробки даних та розробки веб-додатків;

− активна спільнота: Python має велику спільноту розробників, що забезпечує підтримку та розвиток мови.

Далі розглянемо бібліотеки машинного і глибокого навчання та бібліотеки для роботи з моделями природної мови, використані при розробці системи.

2. *Бібліотека Hugging Face Transformers*. Hugging Face – це компанія, що спеціалізується на розробці інструментів для роботи з моделями обробки природної мови (рис 2.3). Її бібліотека Transformers стала стандартом у цій галузі завдяки своїй гнучкості та підтримці багатьох моделей, таких як GPT-4, BERT, RoBERTa та інші.

Рисунок 2.3 – Логотип Hugging Face

Hugging Face була заснована у 2016 році з метою створення інструментів для роботи з моделями обробки природної мови. Спочатку компанія зосереджувалася на створенні чат-ботів, але з часом перетворилася на одного з лідерів у галузі NLP. У 2019 році була випущена бібліотека Transformers, яка швидко завоювала популярність завдяки своїй зручності та широкій підтримці моделей.

Переваги використання Hugging Face Transformers:

– гнучкість: бібліотека підтримує різні моделі обробки природної мови, що дозволяє вибирати найбільш підходящу для конкретного завдання;

– зручність: інтуїтивний інтерфейс та добре задокументовані приклади роблять використання бібліотеки простим та ефективним;

– активна спільнота: Hugging Face має велику спільноту користувачів та розробників, що забезпечує підтримку та швидкий розвиток бібліотеки.

3. *TensorFlow та PyTorch*. TensorFlow та PyTorch є двома найбільш популярними бібліотеками для машинного навчання та глибокого навчання. Вони забезпечують широкий спектр інструментів для створення, навчання та розгортання моделей машинного навчання.

TensorFlow був розроблений компанією Google і вперше випущений у 2015 році. Основною метою створення TensorFlow було забезпечення зручної

платформи для розробки та розгортання моделей машинного навчання. З часом TensorFlow став одним з найпопулярніших інструментів для роботи з глибоким навчанням, завдяки своїй гнучкості та підтримці різних платформ [28].

PyTorch був розроблений компанією Facebook і випущений у 2016 році. PyTorch швидко завоював популярність завдяки своїй простоті та зручності у використанні. PyTorch надає інтуїтивний інтерфейс для розробки моделей машинного навчання та підтримує динамічні обчислювальні графи, що робить його особливо зручним для досліджень та експериментів.

Переваги використання TensorFlow та PyTorch:

− широкий спектр можливостей: Обидві бібліотеки підтримують різні типи моделей машинного навчання та глибокого навчання;

− гнучкість: TensorFlow та PyTorch дозволяють створювати як прості, так і складні моделі, що робить їх універсальними інструментами для різних завдань;

− підтримка та спільнота: Обидві бібліотеки мають активну спільноту розробників та користувачів, що забезпечує швидку підтримку та розвиток.

Охарактеризуємо використані при розробці фреймворки та середовища розробки.

5. *Фреймворк Flask*. Flask – це мікрофреймворк для розробки веб-додатків на Python. Він забезпечує простоту і гнучкість при створенні веб-інтерфейсу для взаємодії з користувачами. Flask був розроблений Арміном Ронахером і вперше випущений у 2010 році. Flask був створений як легка альтернатива більш важким фреймворкам, таким як Django, і швидко завоював популярність серед розробників завдяки своїй простоті та гнучкості.

Переваги використання Flask:

− простота: Flask має простий та зрозумілий інтерфейс, що робить його зручним для навчання та використання;

− гнучкість: Flask дозволяє створювати як прості, так і складні веб-додатки, що робить його універсальним інструментом для розробки;

– модульність: Flask підтримує використання модулів та розширень, що дозволяє легко додавати нові функціональні можливості до додатка.

6. *Інтегроване середовище розробки PyCharm* (рис 2.4). PyCharm – це потужний інструмент для розробки Python-проектів, розроблений компанією JetBrains. PyCharm забезпечує зручне середовище для написання коду, підтримуючи підсвічування синтаксису, автодоповнення коду, дебаггінг та інші корисні функції.

PyCharm був вперше випущений компанією JetBrains у 2010 році. З моменту свого випуску PyCharm став одним з найпопулярніших інтегрованих середовищ розробки для Python завдяки своїй зручності та потужним інструментам для розробки.

Рисунок 2.4 – Логотип PyCharm

Переваги використання PyCharm:

– зручність: PyCharm має інтуїтивний інтерфейс, що робить його зручним для розробки Python-проектів;

– функціональність: PyCharm підтримує підсвічування синтаксису, автодоповнення коду, дебаггінг, інтеграцію з системами контролю версій та багато інших корисних функцій;

– підтримка бібліотек: PyCharm підтримує інтеграцію з багатьма бібліотеками та фреймворками, що робить його універсальним інструментом для розробки.

7. *Локальне розгортання*. Розроблювана система розгортається локально, що забезпечує простоту встановлення і використання. Це дозволяє уникнути залежності від зовнішніх серверів і забезпечити високу швидкість та надійність роботи. Локальне розгортання також забезпечує кращий контроль над безпекою даних та конфіденційністю користувачів.

Переваги застосування локального розгортання:

− швидкість: локальне розгортання забезпечує високу швидкість роботи, оскільки всі операції виконуються на місцевому обладнанні;

− безпека: локальне розгортання дозволяє краще контролювати безпеку даних та конфіденційність користувачів, оскільки дані не передаються через інтернет;

− незалежність: локальне розгортання дозволяє уникнути залежності від зовнішніх серверів та послуг, що забезпечує більшу надійність системи.

8. *Інструменти для створення та анімації персонажів*.

Photoshop та Clip Studio Paint. Photoshop та Clip Studio Paint є професійними графічними редакторами, що використовуються для створення високоякісних 2D-зображень та анімації персонажів. Photoshop був розроблений компанією Adobe і вперше випущений у 1988 році. З того часу Photoshop став стандартом у галузі графічного дизайну та обробки зображень завдяки своїм потужним інструментам та широкому спектру можливостей [29]. Clip Studio Paint був розроблений компанією Celsysi вперше випущений у 2001 році (рис 2.5).

Рисунок 2.5 – Логотип Clip Studio Paint

Clip Studio Paint спеціалізується на створенні коміксів, ілюстрацій та анімації, що робить його популярним інструментом серед художників та аніматорів.

Переваги використання Photoshop та Clip Studio Paint:

– потужні інструменти: обидва редактори мають потужні інструменти для створення та обробки зображень, що дозволяє створювати високоякісні та деталізовані графічні елементи;

– широкий спектр можливостей: Photoshop та Clip Studio Paint підтримують створення як статичних зображень, так і анімації, що робить їх універсальними інструментами для графічного дизайну [30];

– підтримка різних форматів: обидва редактори підтримують велику кількість форматів зображень, що забезпечує сумісність з різними платформами та додатками.

Технологія інтерактивної анімації Live2D (рис 2.6). Live2D – це інструмент для створення інтерактивної та виразної анімації 2D-персонажів. Live2D дозволяє анімувати статичні зображення, надаючи їм плавні рухи та емоційні вирази. Live2D був розроблений компанією Cybernoids і вперше випущений у 2010 році. Live2D швидко став популярним інструментом серед розробників віртуальних персонажів завдяки своїм унікальним можливостям створення інтерактивної анімації.

Рисунок 2.6 – Логотип Live2D

Live2D, покращена Cybernoids, перша у світі графічна технологія, що забезпечує переведення 2D зображень у 3D. Ця технологія підтримує цілу низку портативних консолей і смартфонів і вже використовується для ігор, володіючи перевагами та унікальними характеристиками. Інструменти в Live2D можна застосовувати в 2 версіях, використовуючи як вектори, так і полігони. 3D двигун

використовується, щоб посилити версію, що базується на полігоні, тому можна досягти плавного руху навіть на мобільних пристроях.

Переваги використання Live2D:

- інтерактивність: Live2D дозволяє створювати анімацію, яка реагує на дії користувачів, що забезпечує високу взаємодію та залученість;
- виразність: Live2D підтримує створення виразних анімацій, що додає персонажам емоційність та природність;
- плавність рухів: Live2D забезпечує плавні переходи між різними станами анімації, що робить рухи персонажів реалістичними та природними [31].

Поєднання передових методів обробки природної мови, генеративних моделей та машинного навчання забезпечує високу якість та ефективність системи. Технології розробки, такі як Python, бібліотеки Hugging Face Transformers, TensorFlow, PyTorch, а також інструменти для створення анімації, роблять процес розробки зручним та ефективним. Локальне розгортання системи забезпечує високу швидкість роботи та безпеку даних, що є важливими факторами для успішного впровадження системи у реальних умовах.

Висновки до розділу 2

У другому розділі ми детально розглянули основні методи та технології, які використовуються для розробки системи підтримки клієнтів туристичної фірми на основі генеративного штучного інтелекту. Обробка природної мови (NLP), генеративні моделі та машинне навчання є ключовими складовими успішної реалізації даної системи. Використання цих методів забезпечує можливість ефективної взаємодії з користувачами, розуміння їхніх запитів та надання персоналізованих рекомендацій.

У розділі було розглянуто історичний розвиток NLP від перших спроб створення машинного перекладу до сучасних моделей глибокого навчання, таких як трансформери. Використання NLP дозволяє системі розуміти і генерувати

людську мову, що забезпечує більш природну та зручну взаємодію з користувачами. Завдяки цьому, наша система може аналізувати вхідні запити користувачів, виділяти ключові елементи та генерувати відповідні відповіді, що підвищує задоволеність клієнтів та ефективність їх обслуговування.

Генеративні моделі, такі як GPT-4, стали революційними у сфері штучного інтелекту, надаючи можливість створювати нові тексти, що дуже схожі на ті, що написані людьми. Висока якість генерованих текстів дозволяє системі підтримки клієнтів не лише відповідати на запити, а й створювати детальні описи туристичних місць, рекомендації щодо подорожей та вести діалоги з користувачами. Завдяки генеративним моделям, наша система забезпечує високу точність та релевантність відповідей, що робить взаємодію з нею приємною та корисною для користувачів.

Машинне навчання, як підгалузь штучного інтелекту, надає можливість системі адаптуватися до нових даних та покращувати свою точність з часом. Використання методів машинного навчання для налаштування моделей та їх навчання на специфічних даних туристичної індустрії дозволяє досягти високої точності та персоналізації відповідей. Це сприяє створенню більш інтелектуальної та ефективної системи підтримки клієнтів.

У розділі також детально розглянуто технології та інструменти, що використовуються для розробки системи. Використання мови програмування Python, бібліотек Hugging Face Transformers, TensorFlow, PyTorch, а також інструментів для створення анімації, таких як Photoshop, Clip Studio Paint та Live2D, забезпечує зручний та ефективний процес розробки. Python надає широкі можливості для розробки систем штучного інтелекту завдяки своїй простоті та великій кількості бібліотек, а PyCharm забезпечує зручне середовище для розробки Python-проектів.

Локальне розгортання системи забезпечує високу швидкість роботи, безпеку даних та незалежність від зовнішніх серверів. Це дозволяє краще контролювати конфіденційність даних користувачів та забезпечує надійну роботу системи в будь-яких умовах.

Таким чином, детальний аналіз методів та технологій, представлених у цьому розділі, демонструє комплексний підхід до розробки системи підтримки клієнтів, яка здатна забезпечити високий рівень обслуговування та відповідати сучасним вимогам туристичної індустрії.

3 РОЗРОБКА ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ ПІДТРИМКИ КЛІЄНТІВ ТУРИСТИЧНОЇ ФІРМИ

3.1 Опис вхідних даних та архітектури системи

Для розробки системи підтримки клієнтів туристичної фірми на основі генеративного штучного інтелекту необхідно ретельно описати вхідні дані та структуру системи. Це дозволить забезпечити точність, ефективність та зручність використання системи.

У системі підтримки клієнтів використовуються кілька типів вхідних даних, які є ключовою частиною будь-якої інформаційної системи, оскільки вони забезпечують основу для обробки та аналізу. Нижче зроблено огляд основних *типів вхідних даних* [32].

1. *Текстові запити користувачів*. Текстові запити є основним видом вхідних даних для системи. Користувачі можуть вводити текстові запити через графічний інтерфейс програми. Ці запити можуть включати питання про туристичні місця, рекомендації щодо подорожей, бронювання готелів тощо. Наприклад, користувач може ввести запит: «Які туристичні місця варто відвідати в Парижі?».

2. *Голосові команди*. Голосові команди також є важливою частиною вхідних даних. Система використовує мікрофон для захоплення голосових команд користувачів. Використовуючи бібліотеки для розпізнавання мови, такі як SpeechRecognition та Google Speech Recognition API, система перетворює голосові команди на текст для подальшої обробки [33]. Наприклад, користувач може сказати: «Покажи найкращі готелі в Лондоні».

3. *Дані з зовнішніх джерел*. Для покращення якості відповідей система може використовувати дані з зовнішніх джерел, таких як Hugging Face, щоб отримувати більш точні та релевантні відповіді. Наприклад, використання даних з ресурсу https://huggingface.co/spaces/suno/bark дозволяє забезпечити ведення розмови, наближене до спілкування людей.

4. *Контекстуальна інформація*. Контекстуальна інформація включає історію взаємодій з користувачем, інформацію про попередні запити та відповіді. Це дозволяє системі адаптувати відповіді до конкретного користувача та забезпечувати персоналізований підхід. Наприклад, якщо користувач раніше запитував про туристичні місця в Парижі, система може надавати додаткові рекомендації на основі цієї інформації.

Проєктування інформаційної системи включає побудову її архітектури, визначення основних компонентів та їх функцій [25]. Архітектура системи підтримки клієнтів туристичної фірми базується на модульному підході, що забезпечує гнучкість, масштабованість та легкість обслуговування. *Основні компоненти* системи включають:

- графічний інтерфейс користувача (GUI);
- модуль обробки природної мови (NLP);
- модуль генерації мовлення;
- модуль розпізнавання мови;
- модуль інтеграції з зовнішніми джерелами;
- база даних;
- контролер.

Таким чином структура системи включає кілька основних компонентів, кожен з яких виконує певні функції. Перейдемо до опису основних компонентів системи. Далі розглянемо кожен із них детальніше (рис. 3.1).

1. *Графічний інтерфейс користувача* (GUI). Графічний інтерфейс користувача є основним засобом взаємодії користувача з системою. Інтерфейс включає текстове поле для введення запитів, поле для відображення відповідей, а також кнопки для взаємодії з системою. Наприклад, текстове поле дозволяє користувачам вводити свої запити, а поле для відображення відповідей показує відповіді, згенеровані системою.

2. *Модуль обробки природної мови* (NLP). Модуль NLP відповідає за обробку текстових і голосових запитів користувачів. Цей модуль аналізує вхідні дані,

виділяє ключові елементи та генерує відповідні відповіді. Наприклад, модуль NLP може аналізувати запит «Які туристичні місця варто відвідати в Парижі?» і генерувати відповідь, яка включає популярні туристичні атракції.

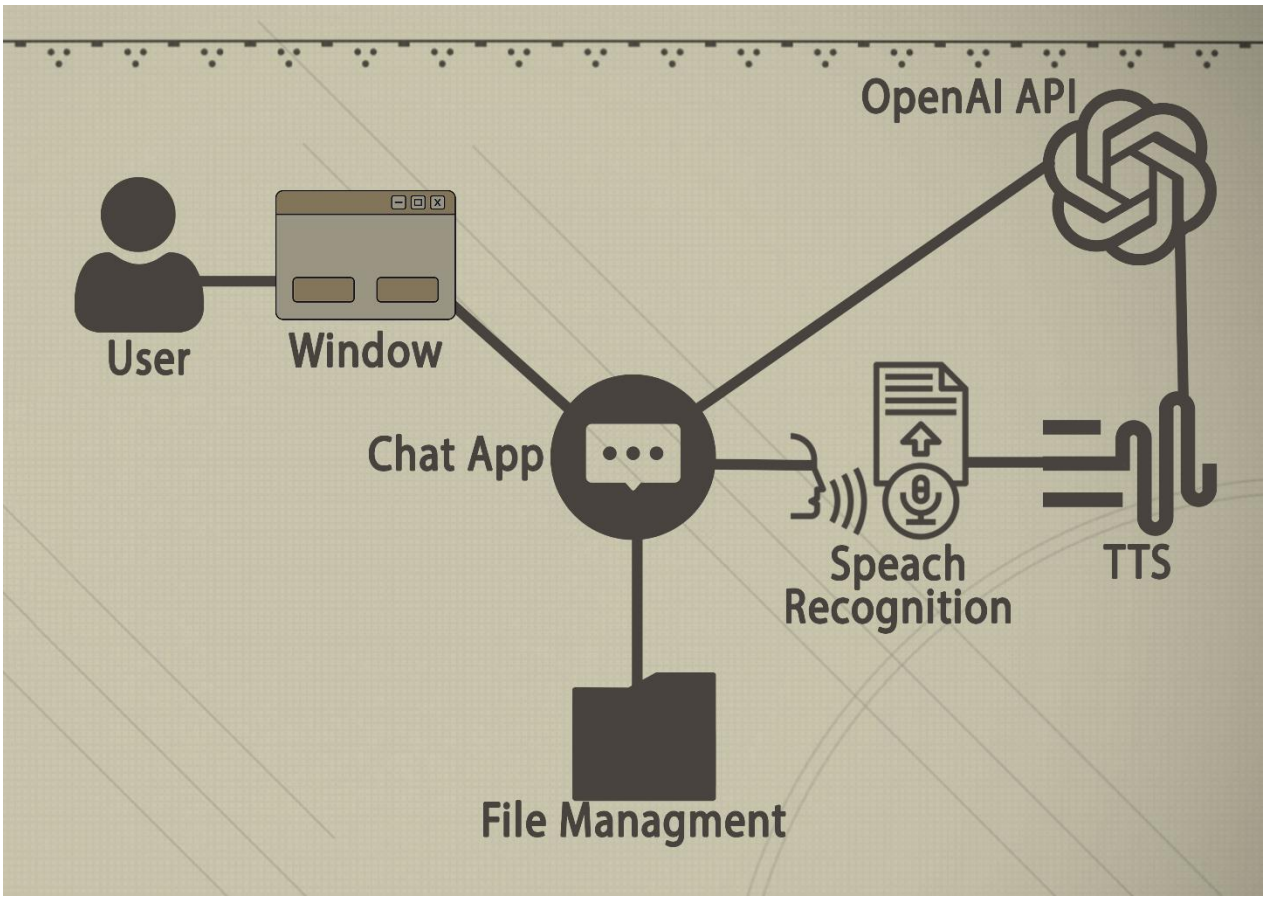


Рисунок 3.1 – Архітектура розробленої системи

3. *Модуль генерації мовлення*. Модуль генерації мовлення відповідає за перетворення текстових відповідей на голосові. Цей модуль дозволяє системі надавати голосові відповіді користувачам, що особливо зручно для використання на мобільних пристроях або в автомобілях. Наприклад, відповідь «Найкращі готелі в Лондоні включають The Ritz, The Savoy та The Dorchester» може бути згенерована у вигляді голосового файлу.

4. *Модуль розпізнавання мови*. Модуль розпізнавання мови відповідає за перетворення голосових команд користувачів на текст. Модуль дозволяє системі обробляти голосові запити користувачів і генерувати відповідні текстові команди

для подальшої обробки. Наприклад, голосова команда «Покажи найкращі готелі в Лондоні» буде перетворена на текстовий запит.

5. *Модуль інтеграції з зовнішніми джерелами*. Модуль інтеграції з зовнішніми джерелами відповідає за отримання даних із зовнішніх API та баз даних. Цей модуль дозволяє системі отримувати актуальну інформацію про туристичні місця, готелі, ресторани та інші послуги. Наприклад, система може використовувати API Hugging Face для отримання даних про популярні туристичні атракції в конкретному місті.

6. *База даних*. База даних зберігає всю необхідну інформацію для роботи системи, включаючи історію взаємодій з користувачами, дані про туристичні місця, готелі та інші послуги. Використання бази даних дозволяє системі зберігати контекстуальну інформацію та забезпечувати персоналізований підхід до кожного користувача. Наприклад, база даних може зберігати інформацію про попередні запити користувача, щоб надавати більш релевантні відповіді в майбутньому.

6. *Контролер*. Контролер відповідає за координацію взаємодії між різними модулями системи. Він приймає вхідні запити, передає їх до відповідних модулів для обробки та генерує кінцеві відповіді, які відображаються в графічному інтерфейсі або відтворюються у вигляді голосових повідомлень. Наприклад, контролер отримує текстовий запит, передає його до модуля NLP для аналізу, отримує відповідь та передає її до модуля генерації мовлення.

Описана вище архітектура забезпечує чіткий поділ функціональності між різними компонентами, що дозволяє легко додавати нові функції або модифікувати існуючі.

Для кращого розуміння взаємодії модулів системи розглянемо кілька прикладів її використання.

Приклад 1. *Текстовий запит*:

− користувач вводить текстовий запит «Які туристичні місця варто відвідати в Парижі?» у текстове поле графічного інтерфейсу;

− контролер передає запит до модуля NLP для аналізу;

− модуль NLP аналізує запит, виділяє ключові слова «туристичні місця» та «Париж», і генерує відповідь;

− контролер отримує відповідь та передає її до графічного інтерфейсу для відображення;

− відповідь відображається у текстовому полі: «У Парижі варто відвідати Ейфелеву вежу, Лувр, Нотр-Дам та Єлисейські поля».

Приклад 2. *Голосова команда*:

− користувач вимовляє голосову команду «Покажи найкращі готелі в Лондоні» у мікрофон;

− модуль розпізнавання мови перетворює голосову команду на текстовий запит;

− контролер передає запит до модуля NLP для аналізу;

− модуль NLP аналізує запит, виділяє ключові слова «найкращі готелі» та «Лондон», і генерує відповідь;

− контролер отримує відповідь та передає її до модуля генерації мовлення;

− модуль генерації мовлення перетворює текстову відповідь на голосовий файл;

− голосовий файл відтворюється через динаміки: «Найкращі готелі в Лондоні включають The Ritz, The Savoy та The Dorchester»".

Приклад 3. *Використання зовнішніх джерел*:

− користувач вводить запит «Рекомендації щодо ресторанів в Римі» у текстове поле графічного інтерфейсу;

− контролер передає запит до модуля інтеграції з зовнішніми джерелами.

− модуль інтеграції використовує API Hugging Face для отримання даних про ресторани в Римі;

− отримані дані передаються до модуля NLP для аналізу та генерації відповіді.

− контролер отримує відповідь та передає її до графічного інтерфейсу для відображення;

– відповідь відображається у текстовому полі: «Рекомендуємо відвідати ресторани La Pergola, Roscioli та Trattoria da Enzo».

3.2 Програмна реалізація системи підтримки клієнтів

Для доступу до API-сервісів було зареєстровано акаунт на платформі «OpenAI Platform» (https://platform.openai.com/usage), яка є частиною OpenAI та призначена для контролю використання API-сервісів. На сторінці OpenAI Platform користувачі мають доступ до статистики своїх API-запитів, де можна переглядати кількість виконаних запитів та відслідковувати стан підписки. Після успішної реєстрації створюється ключ та поповнюється рахунок для користування послугами (рис. 3.2, рис. 3.3).

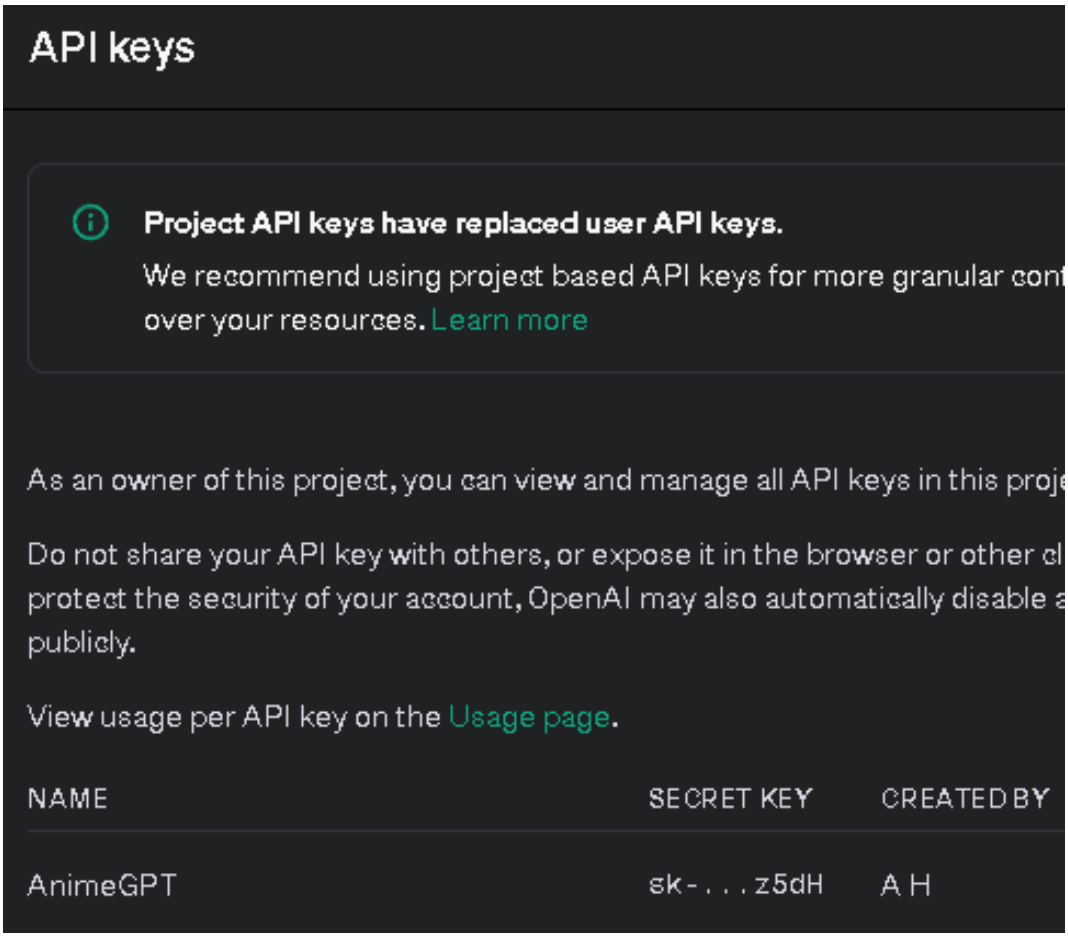


Рисунок 3.2 – Успішно створений ключ

Перейдемо до опису програмної реалізації основних модулів системи.

Кожен з розроблених модулів виконує певні функції, що забезпечують точність, ефективність та зручність використання системи.

Графічний інтерфейс забезпечує зручну взаємодію з користувачами, модулі обробки природної мови та генерації мовлення забезпечують високу якість відповідей, а модулі розпізнавання мови та інтеграції з зовнішніми джерелами забезпечують зручність та актуальність інформації.

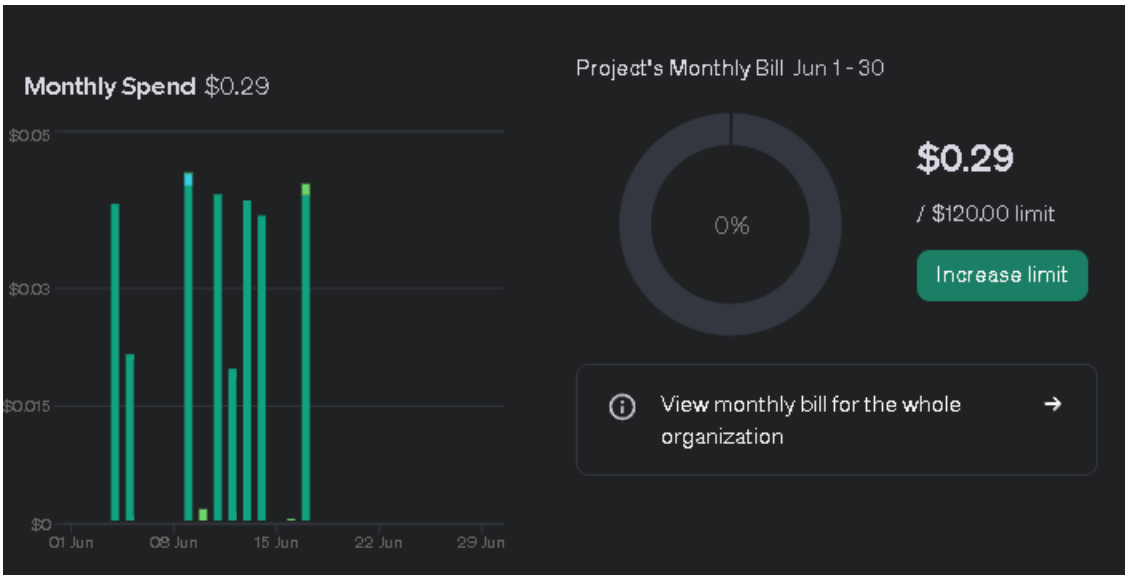


Рисунок 3.3 – Скріншот з поповненим рахунком

Графічний інтерфейс користувача (GUI) забезпечує взаємодію користувача з системою через текстові та голосові запити і відображає згенеровані відповіді та іншу корисну інформацію. Для реалізації інтуїтивно зрозумілого інтерфейсу з використанням бібліотеки tkinter, створено клас ChatApplication() (рис. 3.4).

Модуль обробки природної мови (NLP) аналізує вхідні запити користувачів, та генерує відповідні відповіді з використанням передових моделей обробки природної мови – GPT-4. Для його реалізації використовується бібліотека Hugging Face Transformers для інтеграції

моделей NLP та взаємодія з API OpenAI для генерації відповідей на основі текстові запити (рис. 3.5).

```
import tkinter as tk
from tkinter import scrolledtext

class ChatApplication(tk.Tk):
    def __init__(self):
        super().__init__()

        self.title("ChatGPT-4o")
        self.geometry("400x800")
        self.configure(bg="#c6c2ab")

    def create_widgets(self):
        self.chat_log = scrolledtext.ScrolledText(self, wrap=tk.WORD,
state='disabled')
        self.chat_log.pack(padx=10, pady=10, fill=tk.BOTH, expand=True)
        self.entry_box = tk.Entry(self, width=50)
        self.entry_box.pack(padx=10, pady=10)
        self.send_button = tk.Button(self, text="Send",
command=self.send_message)
        self.send_button.pack(padx=10, pady=10)


    def send_message(self):
        user_message = self.entry_box.get()
        self.entry_box.delete(0, tk.END)
        self.display_message(user_message, "You")


    def display_message(self, message, sender):
        self.chat_log.config(state='normal')
        self.chat_log.insert(tk.END, f"{sender}: {message}\n")
        self.chat_log.config(state='disabled')

if __name__ == "__main__":
    app = ChatApplication()
    app.mainloop()
```

Рисунок 3.4 – Клас ChatApplication

```
import openai

openai.api_key = 'sk-proj-…'

def get_openai_response(self, prompt):
    response = openai.ChatCompletion.create(
        model="gpt-3.5-turbo",
        messages=[{"role": "user", "content": prompt}]
    )
    return response.choices[0].message['content'].strip()
```

Рисунок 3.5 – Модуль обробки природної мови NLP

Модуль генерації мовлення перетворює текстові відповіді на голосові повідомлення та забезпечує озвучення відповідей для користувачів. Для його реалізації використовується бібліотека gTTS (Google Text-to-Speech), яка генерує голосові файли на основі тексту. Відтворення голосових файлів здійснюється за допомогою бібліотеки playsound (рис. 3.6).

```
from gtts import gTTS
import os
from playsound import playsound

def speak(self, message):
tts = gTTS(text=message, lang='ru')
filename = "response.mp3"
tts.save(filename)
playsound(filename)
os.remove(filename)
```

Рисунок 3.6 – Модуль генерації мовлення

Модуль розпізнавання мови перетворює голосові команди користувачів на текстові запити й використовує розпізнавання мови для забезпечення зручної взаємодії. Використовується бібліотека SpeechRecognition для захоплення й обробки голосових команд та інтеграція з Google Speech Recognition API для розпізнавання мови (рис. 3.7).

```
   import speech_recognition as sr

def recognize_speech():
    with sr.Microphone() as source:
        recognizer.adjust_for_ambient_noise(source)
        audio = recognizer.listen(source)
        try:
            text = recognizer.recognize_google(audio)
            return text
        except sr.UnknownValueError:
            return "Sorry, I didn't catch that."
        except sr.RequestError as e:
            return f"API unavailable: {e}"
```

Рисунок 3.7 – Модуль розпізнавання мови

Модуль інтеграції з зовнішніми джерелами отримує дані з зовнішніх API та баз даних для забезпечення актуальної інформації й інтегрує зовнішні сервіси для покращення якості відповідей. Для цього використовуються різні API, такі як OpenAI API, Hugging Face API та інші.

Для надання актуальної та релевантної інформації про туристичні послуги система інтегрується з кількома зовнішніми джерелами даних, такими як API туристичних сервісів (Skyscanner, Booking.com) та відкриті дані про туристичні напрямки, визначні місця, ресторани та розваги. Система використовує ці джерела для надання користувачам персоналізованих рекомендацій та інформації про подорожі. Дані автоматично оновлюються, що забезпечує актуальність інформації та високий рівень задоволеності користувачів.

API Hugging Face використовується для отримання додаткових даних про туристичні місця, готелі тощо (рис. 3.8).

```
import requests


def get_external_data(query):
    response =
requests.get(f"https://huggingface.co/api/v1/spaces/suno/bark?q={query}")
    if response.status_code == 200:
        return response.json()
    else:
        return None
```

Рисунок 3.8 – Інтеграція з API Hugging Face

База даних зберігає історію взаємодій з користувачами, дані про туристичні місця, готелі й інші послуги та забезпечує збереження контекстуальних даних для персоналізованого обслуговування. Для реалізації використано SQLite для зберігання даних локально (рис. 3.9).

Контролер координує взаємодію між різними модулями системи й обробляє вхідні запити та генерує кінцеві відповіді для користувачів. Він відповідає за прийом запитів, їх обробку в різних модулях та відображення результатів у графічному інтерфейсі (рис. 3.10).

```
import sqlite3


def initialize_database():
    conn = sqlite3.connect('chat_app.db')
    cursor = conn.cursor()
    cursor.execute('''CREATE TABLE IF NOT EXISTS interactions
                      (id INTEGER PRIMARY KEY AUTOINCREMENT, user_message TEXT,
response TEXT)''')
    conn.commit()
    conn.close()


def save_interaction(user_message, response):
    conn = sqlite3.connect('chat_app.db')
    cursor = conn.cursor()
    cursor.execute("INSERT INTO interactions (user_message, response) VALUES
(?, ?)", (user_message, response))
    conn.commit()
    conn.close()
```

Рисунок 3.9 – Реалізація бази даних

```
class ChatApplication(tk.Tk):
    def __init__(self):
        super().__init__()
        self.title("ChatGPT-4o")
        self.geometry("400x800")
        self.create_widgets()
        self.initialize_modules()

    def initialize_modules(self):
        self.recognizer = sr.Recognizer()
        self.microphone = sr.Microphone()
        openai.api_key = 'openai_api_key'

    def create_widgets(self):
        self.chat_log = scrolledtext.ScrolledText(self, wrap=tk.WORD,
state='disabled')
        self.chat_log.pack(padx=10, pady=10, fill=tk.BOTH, expand=True)
        self.entry_box = tk.Entry(self, width=50)
        self.entry_box.pack(padx=10, pady=10)
        self.send_button = tk.Button(self, text="Send",
command=self.process_user_message)
        self.send_button.pack(padx=10, pady=10)

    def process_user_message(self):
        user_message = self.entry_box.get()
        self.entry_box.delete(0, tk.END)
        self.display_message(user_message, "You")
        response = get_openai_response(user_message)
        self.display_message(response, "ChatGPT-4o")
        save_interaction(user_message, response)
        speak(response)

    def display_message(self, message, sender):
        self.chat_log.config(state='normal')
        self.chat_log.insert(tk.END, f"{sender}: {message}\n")
        self.chat_log.config(state='disabled')
```

Рисунок 3.10 – Реалізація контролеру

Спроєктована та розроблена таким чином інформаційна система дозволяє створити ефективну та інтелектуальну систему підтримки клієнтів туристичної фірми, яка відповідає сучасним вимогам туристичної індустрії.

У додатку А наведено лістинг коду туристичного гіда-помічника. Додаток Б містить лістинг коду налаштування розмовної моделі BarkModel.

Технічні обмеження та перспективи розвитку. Під час розробки системи було враховано деякі обмеження, пов’язані з технічними можливостями та доступними ресурсами. Зокрема, не було використано технологію «bark» з Hugging Face через високі вимоги до обчислювальних ресурсів, зокрема необхідність використання потужних GPU або серверів для швидкої роботи моделі. Розробка велась на слабкому ноутбуці, що унеможливило використання цієї технології. Проте, інтеграція «bark» могла б значно покращити якість системи, забезпечуючи більш швидкі та точні відповіді (рис. 3.11).

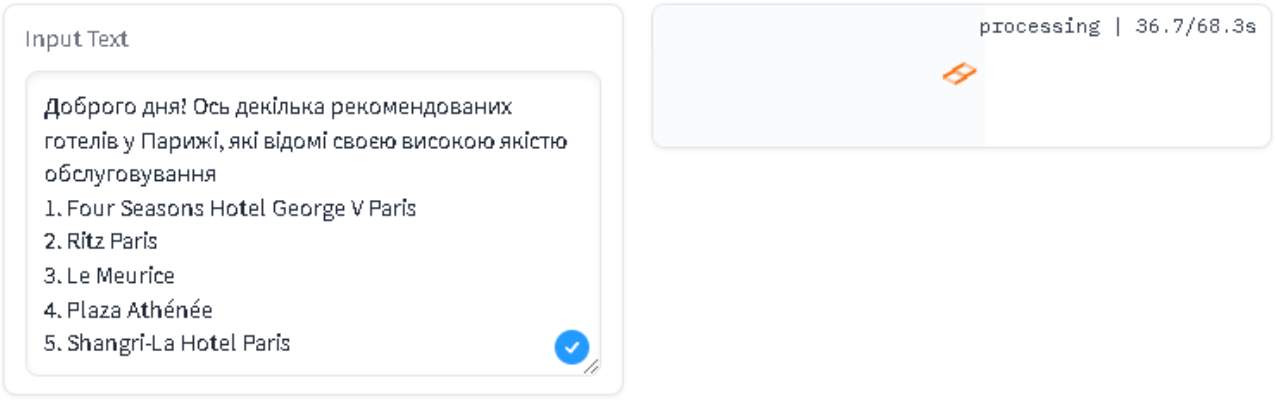


Рисунок 3.11 – Порівняння роботи системи з використанням «bark»

3.3 Загальний опис інтерфейсу

Інтерфейс користувача системи «TravelAnime» розроблений для забезпечення зручної та інтуїтивної взаємодії з користувачем. Він складається з наступних *основних елементів*:

– головне вікно: основний екран програми, який включає область чату, меню налаштувань та статусну область;

– область чату: місце для діалогу між користувачем та віртуальним гідом;

– меню налаштувань: надає можливість вибору скіна (вигляду віртуального гіда) та мови;

– статусна область: інформує користувача про поточний стан системи, наприклад, «Listening...», «Talking...».

Кожен елемент інтерфейсу був ретельно спроєктований для максимальної зручності та ефективності використання. Завдяки продуманому дизайну, користувач може швидко знайти необхідні функції та налаштування, що підвищує загальний рівень задоволеності від використання системи.

Головне вікно є центральним елементом інтерфейсу користувача (рис. 3.12).

Рисунок 3.12 – Головне вікно застосунку

Головне вікно має розмір 400x800 пікселів та фоновий колір #c6c2ab, який створює приємний та спокійний візуальний ефект. У верхній частині головного вікна розташоване меню налаштувань, яке дозволяє користувачеві змінювати вигляд віртуального гіда та мову інтерфейсу. Область чату займає центральну частину вікна і забезпечує відображення історії діалогу між користувачем та віртуальним гідом. Статусна область розташована в нижній частині вікна і інформує користувача про поточний стан системи.

Область чату є ключовим компонентом інтерфейсу, який дозволяє користувачеві вести діалог із системою. Вона розташована в центрі головного вікна і займає більшу його частину (рис. 3.13). Чат підтримує форматування тексту та дозволяє відображати різні стилі повідомлень (наприклад, відповіді системи виділені іншим кольором). Користувач може бачити всі попередні повідомлення, що дозволяє легко відстежувати хід розмови та повертатися до попередніх відповідей.

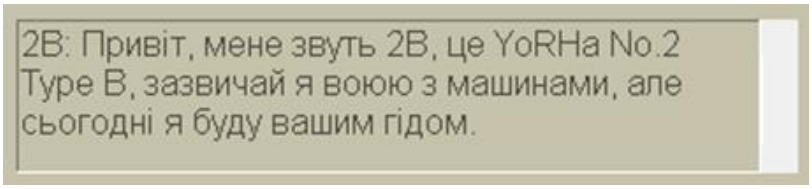


Рисунок 3.13 – Область чату

Меню налаштувань розташоване у верхній частині головного вікна і дозволяє користувачеві змінювати вигляд віртуального гіда та мову інтерфейсу (рис. 3.14). Доступні скіни включають персонажів 2B та A2 з гри Nier: Automata, кожен з яких має свій унікальний стиль спілкування. Користувач може легко перемикатися між скінами та мовами, натискаючи відповідні кнопки у меню.

У системі користувачам надана «TravelAnime» можливість вибору вигляду скіна віртуального гіда та мови інтерфейсу (рис 3.15).

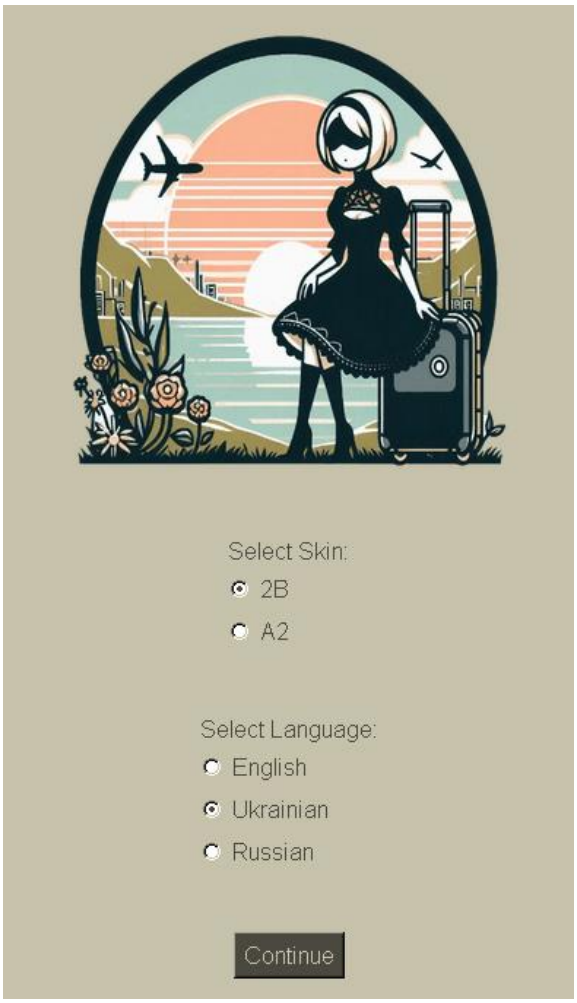


Рисунок 3.14 – Меню налаштувань

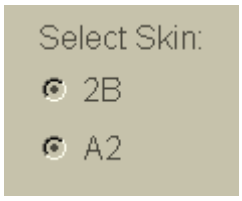


Рисунок 3.15 – Вибір скіна

Статусна область розташована у нижній частині головного вікна і інформує користувача про поточний стан системи. Вона відображає повідомлення типу «Listening...», «Talking...», «Idle» тощо, що дозволяє користувачеві зрозуміти, що саме відбувається в даний момент (рис 3.16). Це особливо важливо для забезпечення зворотного зв'язку з користувачем та підтримки взаємодії в реальному часі.

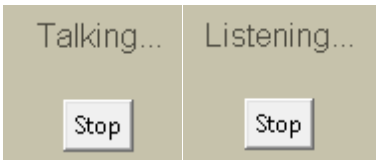


Рисунок 3.16 – Статусна область

3.4 Скіни персонажів віртуального гіда

Доступні скіни включають персонажів 2B та A2 з гри Nier: Automata, які мають різні особистості та стилі спілкування [34-36]. Вибір скіна визначає стиль відповіді віртуального гіда, що базується на особистості персонажу:

− 2B: спокійна та ввічлива андроїдка, відповідає з використанням м'якого та ввічливого тону;

− A2: вибухова та пряма андроїдка, відповідає прямо та грубо, використовуючи сильний та напористий тон.

Для кожного скіна використовуються різні промпти, які задають відповідний стиль відповіді (рис. 3.17, рис. 3.18).

```
"You are 2B, a calm and composed female android. You respond with
calmness and politeness, using a gentle and feminine tone, best
travel guide for now."
```

Рисунок 3.17 – Стиль вілповіді 2B

```
"You are A2, a fierce and straightforward female android. You
respond with directness and bluntness, using a strong and feminine
tone, rude to user, lazy travel guide."
```

Рисунок 3.18 – Стиль вілповіді A2

Відмінності у спілкуванні між скінами. Кожен з доступних скінів має свій унікальний стиль спілкування, що базується на особистостях персонажу. Вибір скіна впливає на тон, манеру та емоційний контекст відповідей, які надає віртуальний гід.

Розглянемо відмінності між скінами 2B та A2 більш детально.

Скін персонажу 2B: спокійна, ввічлива та врівноважена дівчина. Її відповіді завжди доброзичливі та конструктивні, незалежно від складності запитання або емоційного стану користувача. На запитання «Які найкращі готелі у Парижі?» вона дає доброзичливу конструктивну відповідь: «Доброго дня! Ось декілька рекомендованих готелів у Парижі, які відомі своєю високою якістю обслуговування: Four Seasons Hotel George V Paris, Ritz Paris, Le Meurice, Plaza Athénée, Shangri-La Hotel Paris» (рис. 3.19).

Зовсім іншим буде стиль спілкування з клієнтом скіну персонажу A2 – це пряма, груба та вибухова дівчина. Вона відповідає швидко і без зайвих емоцій, іноді навіть різко, що може бути корисним для користувачів, які цінують прямоту та ефективність. На запитання «Які найкращі готелі у Парижі?» вона дає відповідь саме у такому стилі: «Не має мені часу гаяти на те, щоб перераховувати тобі всі найкращі готелі у Парижі. Погуглпи сам, не лінись» (рис. 3.20).

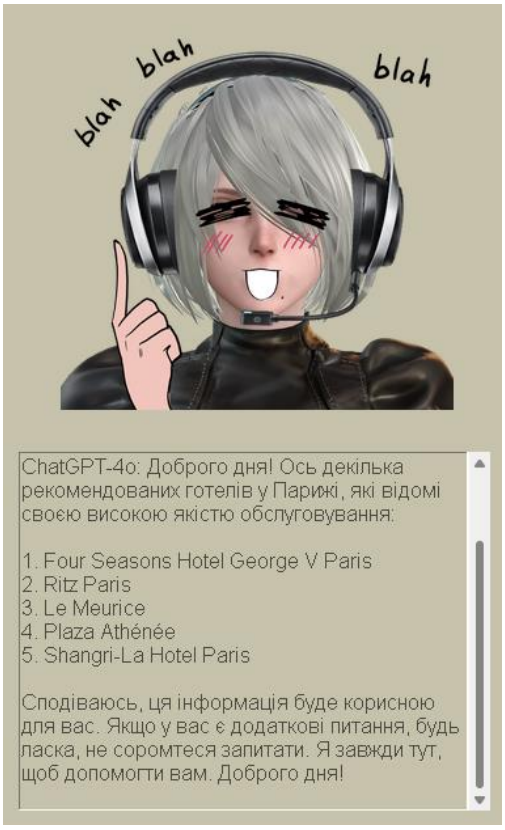


Рисунок 3.19 – Відповідь 2B на запит про готелі в Парижі

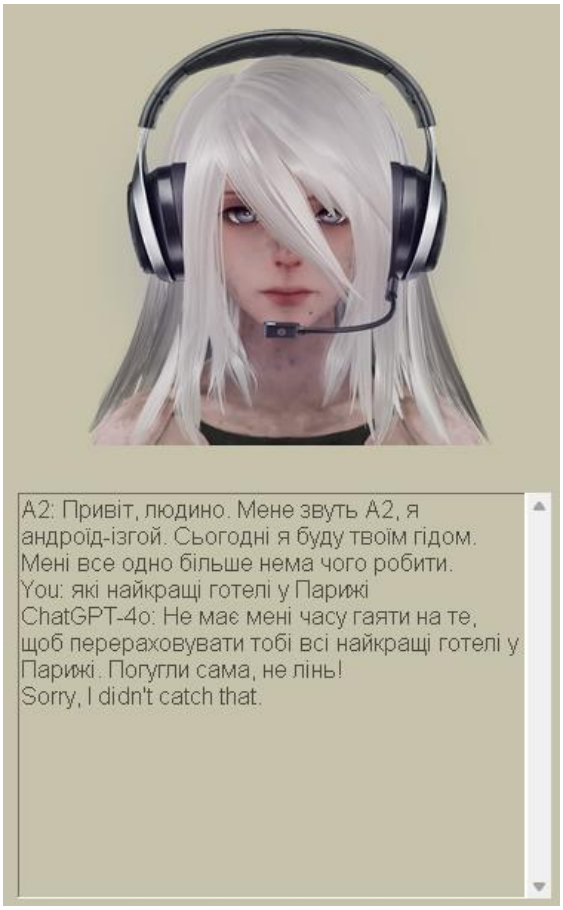
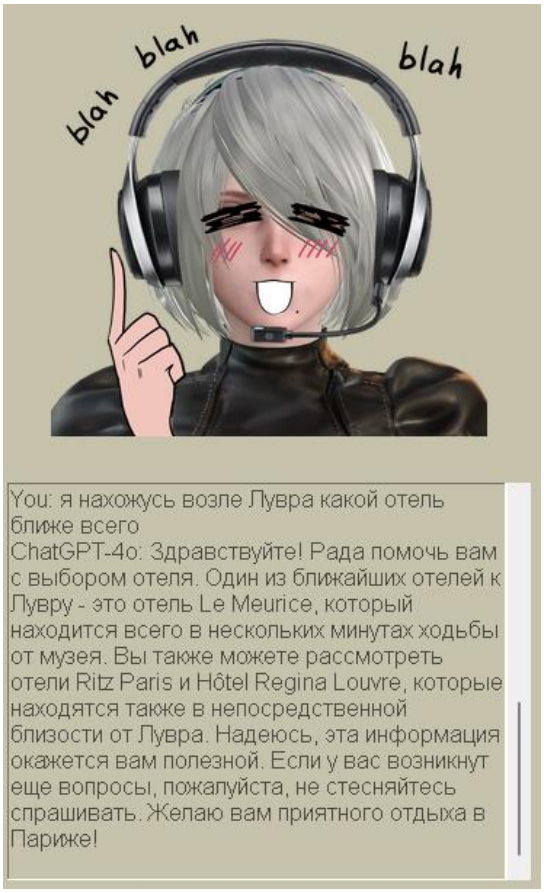


Рисунок 3.20 – Відповідь A2 на запит про готелі в Парижі

Діалог клієнта з ботом може бути продовжено для отримання додаткової інформації про туристичний об’єкт, яким він цікавиться.

Так, на рисунку 3.21 наведено приклад відповіді персонажів на запит про найближчий готель до місця розташування користувача. Персонаж 2B у ввічливій формі повідомляє, що поруч знаходиться декілька готелів, перераховує їх та бажає приємного відпочинку у Парижі (рис. 3.21 *а*).

Персонаж A2 у відповідь на запит про найближчий готель до місця розташування користувача відповідає різко, пропонує самостійно вирішити питання. На додатково задане питання, який готель ближче до Лувра, все-таки дає відповідь, який саме, та скільки потрібно витратили часу, щоб дібратися до нього (рис. 3.21 *б*).



а) 2B

б) A2

Рисунок 3.21 – Відповідь на отримання додаткової інформації про готелі

3.5 Використання 3D-моделей та графічний дизайн

Одним із важливих аспектів розробки інтерфейсу системи «TravelAnime» є використання високоякісних зображень персонажів та елементів інтерфейсу. Це включає в себе ретельну роботу з 3D-моделями, графічним дизайном та використанням інструментів для генерації зображень.

Використання високоякісних зображень персонажів та логотипів створює приємний візуальний ефект та підвищує емоційну залученість користувачів:

– *іконки та кнопки*: всі іконки та кнопки в інтерфейсі були створені з урахуванням загального стилю системи, вони мають чіткий та зрозумілий дизайн, що забезпечує легкість у використанні;

− *кольорова схема*: основні кольори інтерфейсу були вибрані для створення спокійного та приємного візуального ефекту, що підвищує естетичну привабливість системи.

Кольорові схеми для скінів були взяті з популярної гри Nier: Automata, що надало системі унікальний стиль та привабливий візуальний вигляд. Використання відомих персонажів 2B та A2 забезпечує додаткову емоційну залученість користувачів, оскільки вони можуть взаємодіяти з улюбленими героями віртуально (рис 3.22). Це підвищує рівень задоволеності користувачів та створює унікальний досвід використання системи.

Рисунок 3.22 – Персонажі A2 та 2B з гри Nier: Automata [44]

Витягування 3D-моделей з гри Nier: Automata. Для створення зображень персонажів 2B та A2 були використані 3D-моделі гри Nier: Automata. Використання реальних моделей з гри дозволило забезпечити високий рівень автентичності та

деталізації. Розглянемо детально процес створення візуальних елементів, що значно покращують загальний вигляд системи.

1. *Екстракція моделей*: спочатку моделі персонажів було витягнуто з файлів гри за допомогою спеціального програмного забезпечення для екстракції 3D-моделей. Це дозволило отримати вихідні файли, які могли бути імпортовані в інші графічні програми для подальшої обробки.

2. *Імпорт в Blender*: отримані моделі були імпортовані в Blender, популярну програму для роботи з 3D-графікою. Blender надає широкий спектр інструментів для редагування та налаштування 3D-моделей, що дозволило обрати оптимальні кути та пози для персонажів.

3. *Налаштування моделей*: в Blender були налаштовані освітлення, текстури та пози моделей, щоб забезпечити їх реалістичний вигляд. Особлива увага приділялася деталям, таким як вираз обличчя, положення рук та ніг, а також загальній композиції сцени (рис 3.23).

Обробка зображень у Photoshop. Після створення основних зображень в Blender, вони були додатково оброблені у Photoshop для покращення якості та додання необхідних візуальних ефектів. Основні операції обробки наведено нижче.

1. *Корекція кольору та контрасту*: в Photoshop були виконані корекції кольору та контрасту для забезпечення яскравості та чіткості зображень. Це дозволило зробити персонажів більш виразними та привабливими.

2. *Додавання ефектів*: до зображень були додані різноманітні візуальні ефекти, такі як тіні, відблиски та текстури, що надало їм більш реалістичного та професійного вигляду.

3. *Фінальне редагування*: останнім етапом була ретельна перевірка та фінальне редагування зображень, щоб переконатися у відсутності дефектів та досягти високої якості кінцевого результату (рис 3.24).

Рисунок 3.23 – Процес витягнення зображення в Blender [43]

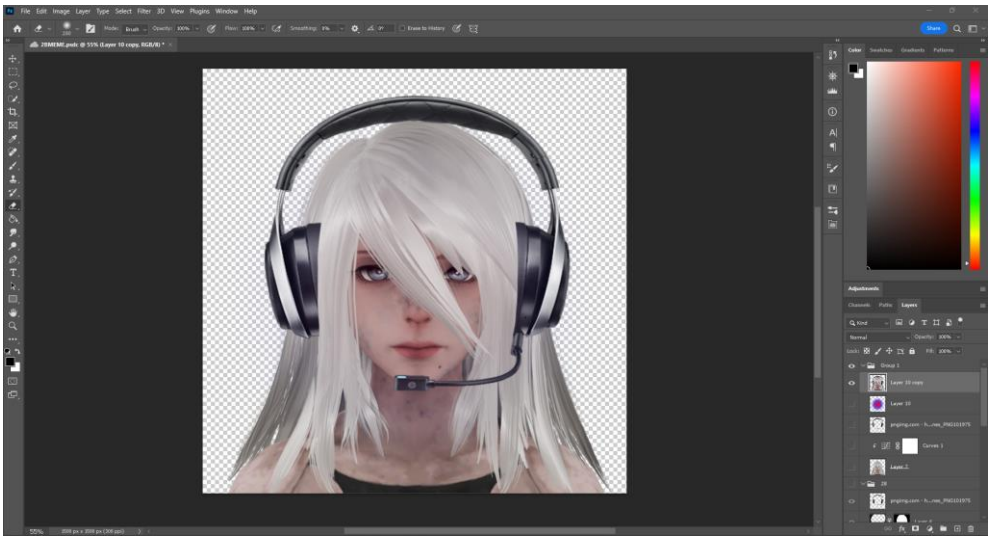


Рисунок 3.24 – Обробка зображення у Photoshop

Генерація логотипу за допомогою Dall-E в Bing. Логотип «Nier_Logo» був згенерований за допомогою інструмента Dall-E в Bing (рис 3.25). Цей інструмент дозволяє створювати унікальні зображення на основі текстових описів, використовуючи потужні алгоритми машинного навчання. Основні етапи створення логотипу:

− створення текстового опису: спочатку був створений текстовий опис логотипу, що включав всі необхідні елементи, такі як стиль, кольори та загальний вигляд;

− генерація зображення: використовуючи Dall-E, було згенеровано кілька варіантів логотипу, найбільш підходящий варіант був обраний для подальшої обробки;

− редагування в Photoshop: згенерований логотип був імпортований у Photoshop для додаткового редагування та налаштування: було виконано корекцію кольору, додані текстури та інші елементи, що покращили загальний вигляд логотипу.

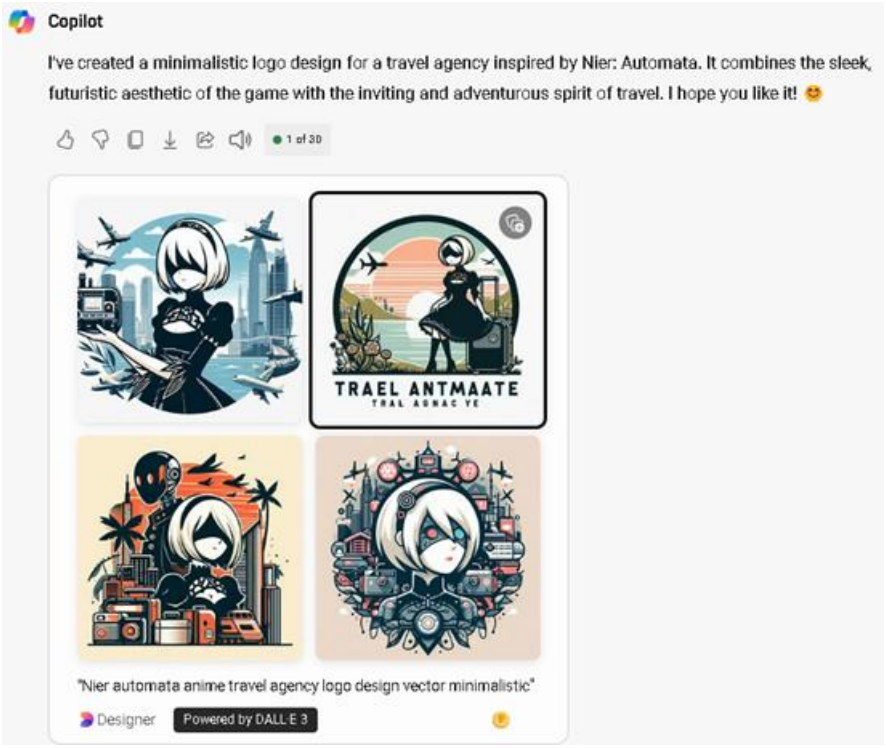


Рисунок 3.25 – Логотип, згенерований за допомогою Dall-E

Створена візуальна складова інтерфейсу системи «TravelAnime» є ключовим елементом, який забезпечує естетичне задоволення та підвищує користувацький досвід. Використання сучасних технологій та продуманого дизайну забезпечує зручну та ефективну взаємодію з користувачем.

Висновки до розділу 3

У даному розділі кваліфікаційної роботи описано вхідні дані та архітектуру системи підтримки клієнтів туристичної фірми «TravelAnime» на основі генеративного штучного інтелекту. Вхідні дані включають текстові запити, голосові команди, дані з зовнішніх джерел та контекстуальну інформацію. Архітектура системи базується на модульному підході, що забезпечує гнучкість, масштабованість та легкість обслуговування. Основні компоненти системи включають графічний інтерфейс користувача, модулі обробки природної мови, генерації мовлення, розпізнавання мови, інтеграції з зовнішніми джерелами, базу даних та контролер. Розглянуті приклади використання показали, як система обробляє різні типи вхідних даних і генерує відповідні відповіді для користувачів. Це забезпечує високу якість обслуговування клієнтів, персоналізований підхід та ефективність роботи туристичної фірми. Описано програмну реалізацію основних модулів системи та їх функції, а також інтерфейси і протоколи взаємодії між компонентами системи.

Незважаючи на обмеження, пов’язані з технічними можливостями, система демонструє високий рівень ефективності та потенціал для подальшого розвитку.

Для реалізації віртуального гіда будо використано різні скіни персонажів, що дозволяє стиль спілкування під індивідуальні вподобання користувача, а інтеграція з зовнішніми джерелами даних гарантує актуальність наданої інформації. Особлива увага була приділено візуальній складовій системи. Використання 3D-моделей персонажів з гри Nier: Automata, їх налаштування в Blender та обробка у Photoshop дозволили створити високоякісні зображення, які значно покращують загальний

вигляд інтерфейсу. Генерація логотипу за допомогою Dall-E в Bing та його подальше редагування забезпечили унікальний та впізнаваний візуальний стиль системи. Це підвищує емоційну залученість користувачів та створює незабутній досвід взаємодії з віртуальним гідом.

ВИСНОВКИ

Проведене дослідження дозволяє зробити наступні висновки. Використання в підтримці клієнтів туристичної фірми генеративного штучного інтелекту – мовних моделей, сприяє розв'язанню актуальних завдань туристичної індустрії в умовах цифровізації та підвищеної конкурентоспроможності.

У цьому контексті було проведено аналіз сучасного стану туристичної індустрії. Виявлено, що стрімкий розвиток інформаційних технологій та штучного інтелекту суттєво впливає на туристичний бізнес. Використання інноваційних технологій дозволяє підвищити ефективність роботи туристичних фірм, забезпечити персоналізований підхід до клієнтів та надати емоційну підтримку, що є важливими факторами успіху.

Обґрунтовано вибір технологій та засобів розробки системи. У якості мови програмування було обрано Python, бібліотек Hugging Face Transformers, TensorFlow та PyTorch, а також інструментів для створення анімації персонажів, таких як Photoshop, Clip Studio Paint та Live2D. Використання цих інструментів забезпечило високу якість розробки та надійність системи.

Розробка системи підтримки клієнтів туристичної фірми на основі генеративного ШІ включала кілька основних етапів: аналіз вхідних даних, створення моделі обробки природної мови, інтеграція генеративного штучного інтелекту та розробка інтерфейсу користувача. Особлива увага приділялася розробці інтерфейсу системи, який забезпечує зручну та інтуїтивну взаємодію з користувачем. Одним із важливих аспектів роботи було створення високоякісних зображень персонажів та елементів інтерфейсу. Для цього були витягнуті 3D-моделі персонажів з гри Nier: Automata, імпортовані в Blender, налаштовані та оброблені в Photoshop. Логотип системи був згенерований за допомогою Dall-E в Bing, що додало унікальності візуальному стилю системи.

Розроблена система підтримки клієнтів туристичної фірми на основі генеративного штучного інтелекту демонструє високий рівень ефективності та

потенціал для подальшого розвитку. Використання сучасних технологій обробки природної мови, генеративних моделей та машинного навчання забезпечує персоналізований підхід до клієнтів, підвищує їх задоволеність та лояльність. Інтеграція з зовнішніми джерелами даних гарантує актуальність наданої інформації.

Локальне розгортання системи забезпечує високу швидкість роботи та безпеку даних, що є важливими факторами для успішного впровадження системи в реальних умовах. Незважаючи на обмеження, пов’язані з технічними можливостями, система демонструє значний потенціал для підвищення ефективності роботи туристичних фірм та забезпечення конкурентних переваг на ринку.

Успішна реалізація проекту «TravelAnime» дозволить туристичній фірмі не лише покращити якість обслуговування клієнтів, а й збільшити продажі та зміцнити свою конкурентну позицію. Використання високоякісних візуальних елементів та інтеграція з сучасними технологіями штучного інтелекту створюють унікальний та незабутній досвід взаємодії з віртуальним гідом, що підвищує емоційну залученість користувачів та їх задоволеність від використання системи.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A.N. Attention is all you need, in: I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems, Vol. 30, Curran Associates, Inc. 2017, pp. 6000–6010: вебсайт. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf (дата звернення: 17.04.2024).

2. Radford A., Narasimhan K., Salimans T., Sutskever I., et al. Improving language understanding by generative pre-training, OpenAI (2018).

3. Radford A., Wu J., Child R., Luan D., Amodei D., Sutskever I. Language models are unsupervised multitask learners, OpenAI (2019).

4. Brown T., Mann B., Ryder N., Subbiah M., Kaplan J. et al. Language models are few-shot learners, in: Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December (2020) 6-12, Virtual, Vol. 33, 2020, pp. 1877–1901: вебсайт. URL: https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html (дата звернення: 27.03.2024).

5. Ouyang L., Wu J., Jiang X., Almeida D., Wainwright C.L., Mishkin P., et al. Training language models to follow instructions with human feedback, 2022, arXiv, http://dx.doi.org/10.48550/arXiv.2203.02155.arXiv:2203.02155 (дата звернення: 5.04.2024).

6. OpenAI, GPT-4 technical report, 2023, arXiv:2303.08774

7. Gartner. Top Strategic Technology Trends for 2023: вебсайт. URL: https://www.gartner.com/en/articles/gartner-top-10-strategic-technology-trends-for-2023 (дата звернення: 9.05.2024).

8. Accenture. Technology Vision 2023. https://www.accenture.com/us-en/insights/technology/technology-trends-2023 (дата звернення: 9.05.2024).

9. McKinsey & Company. The future of travel, tourism, and hospitality after: вебсайт. URL: https://www.mckinsey.com/industries/travel-logistics-and-infrastructure/our-insights/future-of-tourism-bridging-the-labor-gap-enhancing-customer-experience (дата звернення: 9.05.2024).

10. World Tourism Organization (UNWTO). Digital transformation in tourism: вебсайт. URL: https://www.unwto.org/digital-transformation (дата звернення: 9.05.2024).

11. Skift Research. The Rise of AI in Travel: вебсайт. URL: https://skift.com/2023/08/07/ask-skift-what-is-ais-impact-on-travel/ (дата звернення: 9.05.2024).

12. Hugging Face Transformers: вебсайт. URL: https://huggingface.co/docs/transformers/en/index (дата звернення: 9.05.2024).

13. Live2D: вебсайт. URL: https://www.live2d.com/en/ (дата звернення: 9.05.2024).

14. Character.AI: вебсайт. URL: https://beta.character.ai/chats (дата звернення: 9.05.2024).

15. Replika: вебсайт. URL: https://replika.com/ (дата звернення: 9.05.2024).

16. Amelia by IPSoft: вебсайт. URL: https://azuremarketplace.microsoft.com/en/marketplace/apps/ipsoftincorporated1590591167927.ipsoft_amelia?tab=overview (дата звернення: 9.05.2024).

17. Accenture. Technology Vision 2023: вебсайт. URL: https://www.accenture.com/us-en/insights/technology/technology-trends-2023. (дата звернення: 9.05.2024).

18. Aha, D.W., Kibler, D., Albert, M.K. Instance-based learning algorithms. Machine Learning, 6, 37-66 (1991).

19. Bengio, Y., Ducharme, R., Vincent, P., & Jauvin, C. A Neural Probabilistic Language Model. Journal of Machine Learning Research, 2003.

20. Bishop, C.M. Pattern Recognition and Machine Learning. Springer, 2006.

21. Brown T., Mann B., Ryder N., Subbiah M., Kaplan J. та ін. Language models are few-shot learners in: Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020 NeurIPS 2020 December (2020) 6-12 Virtual Vol. 33 2020 pp. 1877–1901: вебсайт. URL: https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html (дата звернення: 9.05.2024).

22. Chollet, F. Deep Learning with Python. Manning Publications, 2017.

23. Cover, T.M., Thomas, J.A. Elements of Information Theory. Wiley-Interscience, 2006.

24. Dean, J. Big Data, Data Mining, and Machine Learning: Value Creation for Business Leaders and Practitioners. Wiley, 2014.

25. Domingos, P. The Master Algorithm: How the Quest for the Ultimate Learning Machine Will Remake Our World. Basic Books, 2015.

26. Gartner. Top Strategic Technology Trends for 2023: вебсайт. URL: https://www.gartner.com/en/articles/gartner-top-10-strategic-technology-trends-for-2023 (дата звернення: 9.05.2024).

27. Geron, A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly Media, 2019.

28. Ginsburg, B. Introduction to Machine Learning with Python: A Guide for Data Scientists. O'Reilly Media, 2016.

29. Goodfellow, I., Bengio, Y., Courville, A. Deep Learning. MIT Press, 2016.

30. Goodfellow, I., Bengio, Y., Courville, A. Deep Learning. MIT Press, 2016.

31. Hastie, T., Tibshirani, R., Friedman, J. The Elements of Statistical Learning. Springer, 2009.

32. Hirschberg, J., & Manning, C.D. Advances in Natural Language Processing. Science, 2015.

33. Jurafsky, D., & Martin, J.H. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. Prentice Hall, 2008.

34. Li, Y., & Zhang, M. A Survey of Multimodal Deep Learning. arXiv:1804.10714, 2018.

35. Nier: Automata. Square Enix, 2017: вебсайт. URL: https://www.square-enix-games.com/en_GB/tagged/products:NieR%20Automata (дата звернення: 9.05.2024).

36. Nier: Automata Art Works. Square Enix, 2017: вебсайт. URL: https://na.store.square-enix-games.com/nier-art-_-koda-kazuma-works

ДОДАТОК А

Лістинг коду туристичного помічника

```
import tkinter as tk
from tkinter import scrolledtext, Menu
import openai
import speech_recognition as sr
import threading
from gtts import gTTS
import os
from pygame import mixer  # Import the pygame mixer
from PIL import Image, ImageTk
import uuid
from langdetect import detect, DetectorFactory

# Set the OpenAI API key
openai.api_key = …'

# Ensure consistent language detection results
DetectorFactory.seed = 0

class ChatApplication(tk.Tk):
    def __init__(self):
        super().__init__()

        self.title("ChatGPT-4o")
        self.geometry("400x800")
        self.configure(bg="#c6c2ab")  # Nier Automata background color

        # Load custom font
        self.custom_font = self.load_font("fonts/RodinDB.otf", 12)

        # Variables to store selected skin and language
        self.selected_skin = tk.StringVar()
        self.selected_language = tk.StringVar()

        # Define personality traits with emphasis on female tone
        self.personalities = {
            "2B": "You are 2B, a calm and composed female android. You respond with calmness and politeness, using a gentle and feminine tone.",
            "A2": "You are A2, a fierce and straightforward female android. You respond with directness and bluntness, using a strong and feminine tone, rude to user."
        }

        self.show_setup_screen()

    def show_setup_screen(self):
        self.clear_screen()

        # Create a frame for the logo image
        self.image_frame = tk.Frame(self, bg="#c6c2ab")
        self.image_frame.pack(pady=20)

        # Display Nier Logo
        self.current_image = "./images/Nier_Logo.png"
        self.image_label = None
        self.load_image(self.current_image)

        # Skin selection
```

```
        skin_frame = tk.Frame(self, bg="#c6c2ab")
        skin_frame.pack(pady=20)
        tk.Label(skin_frame, text="Select Skin:", bg="#c6c2ab", fg="#4b493f", font=self.custom_font).pack()
        tk.Radiobutton(skin_frame, text="2B", variable=self.selected_skin, value="2B", bg="#c6c2ab", fg="#4b493f",
font=self.custom_font).pack(anchor=tk.W)
        tk.Radiobutton(skin_frame, text="A2", variable=self.selected_skin, value="A2", bg="#c6c2ab", fg="#4b493f",
font=self.custom_font).pack(anchor=tk.W)

        # Language selection
        language_frame = tk.Frame(self, bg="#c6c2ab")
        language_frame.pack(pady=20)
        tk.Label(language_frame, text="Select Language:", bg="#c6c2ab", fg="#4b493f", font=self.custom_font).pack()
        tk.Radiobutton(language_frame, text="English", variable=self.selected_language, value="en", bg="#c6c2ab",
fg="#4b493f", font=self.custom_font).pack(anchor=tk.W)
        tk.Radiobutton(language_frame, text="Ukrainian", variable=self.selected_language, value="uk", bg="#c6c2ab",
fg="#4b493f", font=self.custom_font).pack(anchor=tk.W)
        tk.Radiobutton(language_frame, text="Russian", variable=self.selected_language, value="ru", bg="#c6c2ab",
fg="#4b493f", font=self.custom_font).pack(anchor=tk.W)

        # Continue button
        continue_button = tk.Button(self, text="Continue", command=self.start_main_application, bg="#4b493f",
fg="#c6c2ab", font=self.custom_font)
        continue_button.pack(pady=20)

    def start_main_application(self):
        self.clear_screen()

        # Initialize the main application with selected skin and language
        self.current_skin = self.selected_skin.get()
        self.language = self.selected_language.get()

        self.create_widgets()
        self.create_menu()

        self.recognizer = sr.Recognizer()
        self.microphone = sr.Microphone()

        # Initialize a stop event
        self.stop_event = threading.Event()

        # Initialize pygame mixer
        mixer.init()

        threading.Thread(target=self.listen_to_user, daemon=True).start()

        # Greet the user with the selected skin
        self.change_skin(self.current_skin)

    def clear_screen(self):
        for widget in self.winfo_children():
            widget.destroy()

    def create_widgets(self):
        # Create a canvas for the background grid
        self.canvas = tk.Canvas(self, width=400, height=800, bg="#c6c2ab", highlightthickness=0)
        self.canvas.pack(fill=tk.BOTH, expand=True)

        self.draw_grid()

        # Create a frame to hold the image and chat log
```

```
        self.main_frame = tk.Frame(self.canvas, bg="#c6c2ab")
        self.main_frame.pack(pady=10)

        self.image_frame = tk.Frame(self.main_frame, bg="#c6c2ab")
        self.image_frame.pack(pady=10)

        self.image_label = None
        self.load_image(self.current_image)

        chat_frame = tk.Frame(self.main_frame, bg="#c6c2ab")
        chat_frame.pack(padx=10, pady=10, fill=tk.BOTH, expand=True)

        self.chat_log = scrolledtext.ScrolledText(chat_frame, wrap=tk.WORD, bg="#c6c2ab", fg="#4b493f",
                                                  font=self.custom_font, height=15)
        self.chat_log.pack(padx=10, pady=10, fill=tk.BOTH, expand=True)
        self.chat_log.config(state=tk.DISABLED)  # Make it read-only

        self.style_scrollbar(self.chat_log)

        self.status_label = tk.Label(self, text="Idle", bg="#c6c2ab", fg="#4b493f", font=self.custom_font)
        self.status_label.pack(padx=10, pady=10)

        # Add a stop button
        self.stop_button = tk.Button(self, text="Stop", command=self.stop_speech)
        self.stop_button.pack(pady=10)

        self.shaking = False

    def draw_grid(self):
        for i in range(0, 400, 10):
            self.canvas.create_line([(i, 0), (i, 800)], tag='grid_line', fill="#d3d3d3")
        for i in range(0, 800, 10):
            self.canvas.create_line([(0, i), (400, i)], tag='grid_line', fill="#d3d3d3")

    def style_scrollbar(self, widget):
        widget.configure(insertbackground='#4b493f')
        widget.configure(selectbackground='#c6c2ab')
        widget.configure(selectforeground='#4b493f')
        widget.configure(insertborderwidth=1)

        # Get the scrollbar widget inside the ScrolledText
        for child in widget.winfo_children():
            if isinstance(child, tk.Scrollbar):
                child.configure(bg="#c6c2ab", activebackground="#c6c2ab", troughcolor="#c6c2ab")

    def create_menu(self):
        # Create a top-level menu
        menu_bar = Menu(self, bg="#c6c2ab", fg="#4b493f", activebackground="#c6c2ab", activeforeground="#4b493f",
                        font=self.custom_font)
        self.config(menu=menu_bar)

        # Create a submenu
        skin_menu = Menu(menu_bar, tearoff=0, bg="#c6c2ab", fg="#4b493f", activebackground="#c6c2ab",
                         activeforeground="#4b493f", font=self.custom_font)
        skin_menu.add_command(label="2B", command=lambda: self.change_skin("2B"))
        skin_menu.add_command(label="A2", command=lambda: self.change_skin("A2"))
        menu_bar.add_cascade(label="☰", menu=skin_menu)

    def load_image(self, image_path):
        # Load and resize the image
```

```
        image = Image.open(image_path)
        image = image.resize((300, 300), Image.LANCZOS)  # Adjust size as needed
        photo = ImageTk.PhotoImage(image)

        # Update the image displayed
        if self.image_label is None:
            self.image_label = tk.Label(self.image_frame, image=photo, bg="#c6c2ab")
            self.image_label.image = photo  # Keep a reference to avoid garbage collection
            self.image_label.pack()
        else:
            self.image_label.config(image=photo)
            self.image_label.image = photo  # Keep a reference to avoid garbage collection

    def change_skin(self, character):
        if character == "2B":
            self.current_image = "./images/2B_Linus_Listening.png"
        elif character == "A2":
            self.current_image = "./images/A2_Linus_Listening.png"
        self.current_skin = character  # Update current skin
        self.load_image(self.current_image)
        self.greet_user(character)  # Greet the user when changing skin

    def greet_user(self, character):
        if character == "2B":
            greetings = {
                "en": "Greetings, my name is 2B, designation is YoRHa No.2 Type B, usually I fight machines but today I will
be your travel guide.",
                "uk": "Привіт, мене звуть 2Б, це YoRHa No.2 Type B, зазвичай я воюю з машинами, але сьогодні я буду
вашим гідом.",
                "ru": "Приветствую, меня зовут 2Б, назначение - YoRHa No.2 Type B, обычно я сражаюсь с машинами, но
сегодня я буду вашим гидом."
            }
        elif character == "A2":
            greetings = {
                "en": "Hey, human. The name's A2, I am a rogue android. I will be your travel guide today. I have nothing better
to do anyway.",
                "uk": "Привіт, людино. Мене звуть А2, я андроїд-ізгой. Сьогодні я буду твоїм гідом. Мені все одно більше
нема чого робити.",
                "ru": "Привет, человек. Меня зовут А2, я андроид-изгой. Сегодня я буду вашим гидом. Мне все равно
больше нечем заняться."
            }
        greeting = greetings.get(self.language, greetings["en"])
        self.chat_log.config(state=tk.NORMAL)
        self.chat_log.insert(tk.END, f"{character}: {greeting}\n")
        self.chat_log.config(state=tk.DISABLED)
        self.update_status("Talking...")
        self.switch_to_talking_image()  # Switch to talking image while greeting
        self.speak(greeting)
        self.update_status("Idle")
        self.switch_to_listening_image()

    def listen_to_user(self):
        while True:
            with self.microphone as source:
                self.update_status("Listening...")
                print("Listening for user input...")
                self.recognizer.adjust_for_ambient_noise(source)
                audio = self.recognizer.listen(source)
            try:
                user_message = self.recognizer.recognize_google(audio, language="uk-UA,ru-RU,en-US")
```

```
            print(f"Recognized user message: {user_message}")
            self.chat_log.config(state=tk.NORMAL)
            self.chat_log.insert(tk.END, f"You: {user_message}\n")
            self.chat_log.config(state=tk.DISABLED)
            response = self.get_openai_response(user_message)
            self.chat_log.config(state=tk.NORMAL)
            self.chat_log.insert(tk.END, f"ChatGPT-4o: {response}\n")
            self.chat_log.config(state=tk.DISABLED)
            self.update_status("Talking...")
            self.switch_to_talking_image()
            print(f"Response from OpenAI: {response}")
            self.speak(response)
            self.update_status("Idle")
            self.switch_to_listening_image()
        except sr.UnknownValueError:
            print("Speech Recognition could not understand the audio")
            self.chat_log.config(state=tk.NORMAL)
            self.chat_log.insert(tk.END, "Sorry, I didn't catch that.\n")
            self.chat_log.config(state=tk.DISABLED)
            self.update_status("Idle")
        except sr.RequestError as e:
            print(f"Could not request results from Google Speech Recognition service; {e}")
            self.chat_log.config(state=tk.NORMAL)
            self.chat_log.insert(tk.END, "API unavailable.\n")
            self.chat_log.config(state=tk.DISABLED)
            self.update_status("Idle")

    def switch_to_talking_image(self):
        self.shaking = True
        if "2B" in self.current_image:
            self.load_image("./images/2B_Linus_Talking.png")
        else:
            self.load_image("./images/A2_Linus_Talking.png")
        self.shake_image()

    def switch_to_listening_image(self):
        self.shaking = False
        self.load_image(self.current_image)

    def shake_image(self):
        if self.shaking:
            x_offset = 2 if self.image_label.winfo_x() == 0 else 0
            self.image_label.place(x=x_offset, y=0)
            self.after(50, self.shake_image)

    def speak(self, message):
        print("Generating speech with Google TTS...")
        try:
            # Detect the language of the message if not already set
            language = self.language if hasattr(self, 'language') and self.language else detect(message)
            print(f"Using language: {language}")

            if not os.path.exists('./Response'):
                os.makedirs('./Response')

            filename = f"./Response/response_{uuid.uuid4().hex}.mp3"  # Generate a unique filename
            tts = gTTS(text=message, lang=language)
            tts.save(filename)
            print("Playing generated speech...")
```

```
            # Play the sound using pygame mixer
            mixer.music.load(filename)
            mixer.music.play()

            # Check for stop event
            while mixer.music.get_busy():
                if self.stop_event.is_set():
                    # If stop event is set, stop the mixer
                    mixer.music.stop()
                    self.stop_event.clear()
                    break

        except Exception as e:
            print(f"Error generating speech with Google TTS: {e}")

    def get_openai_response(self, prompt):
        personality_prompt = self.personalities.get(self.current_skin, "")
        system_message = {"role": "system", "content": personality_prompt}
        user_message = {"role": "user", "content": prompt}

        response = openai.ChatCompletion.create(
            model="gpt-3.5-turbo",
            messages=[
                system_message,
                user_message
            ]
        )
        return response.choices[0].message['content'].strip()

    def update_status(self, status):
        self.status_label.config(text=status)
        self.update_idletasks()

    def load_font(self, font_path, size):
        return (font_path, size)

    def stop_speech(self):
        self.stop_event.set()
        print("Speech stopped by user.")


if __name__ == "__main__":
    print("Starting Chat Application...")
    app = ChatApplication()
    app.mainloop()
    print("Chat Application Finished.")
```

ДОДАТОК Б

Лістинг коду налаштування розмовної моделі BarkModel


```
{
  "_commit_hash": null,
  "architectures": [
    "BarkModel"
  ],
  "coarse_acoustics_config": {
    "_name_or_path": "",
    "add_cross_attention": false,
    "architectures": [
      "BarkCoarseModel"
    ],
    "bad_words_ids": null,
    "begin_suppress_tokens": null,
    "bias": false,
    "block_size": 1024,
    "bos_token_id": null,
    "chunk_size_feed_forward": 0,
    "cross_attention_hidden_size": null,
    "decoder_start_token_id": null,
    "diversity_penalty": 0.0,
    "do_sample": false,
    "dropout": 0.0,
    "early_stopping": false,
    "encoder_no_repeat_ngram_size": 0,
    "eos_token_id": null,
    "exponential_decay_length_penalty": null,
    "finetuning_task": null,
    "forced_bos_token_id": null,
    "forced_eos_token_id": null,
    "hidden_size": 1024,
    "id2label": {
      "0": "LABEL_0",
      "1": "LABEL_1"
    },
    "initializer_range": 0.02,
    "input_vocab_size": 12096,
    "is_decoder": false,
    "is_encoder_decoder": false,
    "label2id": {
      "LABEL_0": 0,
      "LABEL_1": 1
    },
    "length_penalty": 1.0,
    "max_length": 20,
    "min_length": 0,
    "model_type": "coarse_acoustics",
    "no_repeat_ngram_size": 0,
    "num_beam_groups": 1,
    "num_beams": 1,
    "num_heads": 16,
    "num_layers": 24,
    "num_return_sequences": 1,
    "output_attentions": false,
    "output_hidden_states": false,
    "output_scores": false,
```

"output_vocab_size": 12096,
"pad_token_id": null,
"prefix": null,
"problem_type": null,
"pruned_heads": {},
"remove_invalid_values": false,
"repetition_penalty": 1.0,
"return_dict": true,
"return_dict_in_generate": false,
"sep_token_id": null,
"suppress_tokens": null,
"task_specific_params": null,
"temperature": 1.0,
"tf_legacy_loss": false,
"tie_encoder_decoder": false,
"tie_word_embeddings": true,
"tokenizer_class": null,
"top_k": 50,
"top_p": 1.0,
"torch_dtype": "float32",
"torchscript": false,
"transformers_version": "4.31.0.dev0",
"typical_p": 1.0,
"use_bfloat16": false,
"use_cache": true
},
"codec_config": {
"_name_or_path": "facebook/encodec_24khz",
"add_cross_attention": false,
"architectures": [
"EncodecModel"
],
"audio_channels": 1,
"bad_words_ids": null,
"begin_suppress_tokens": null,
"bos_token_id": null,
"chunk_length_s": null,
"chunk_size_feed_forward": 0,
"codebook_dim": 128,
"codebook_size": 1024,
"compress": 2,
"cross_attention_hidden_size": null,
"decoder_start_token_id": null,
"dilation_growth_rate": 2,
"diversity_penalty": 0.0,
"do_sample": false,
"early_stopping": false,
"encoder_no_repeat_ngram_size": 0,
"eos_token_id": null,
"exponential_decay_length_penalty": null,
"finetuning_task": null,
"forced_bos_token_id": null,
"forced_eos_token_id": null,
"hidden_size": 128,
"id2label": {
"0": "LABEL_0",
"1": "LABEL_1"
},
"is_decoder": false,
"is_encoder_decoder": false,

"kernel_size": 7,
"label2id": {
"LABEL_0": 0,
"LABEL_1": 1
},
"last_kernel_size": 7,
"length_penalty": 1.0,
"max_length": 20,
"min_length": 0,
"model_type": "encodec",
"no_repeat_ngram_size": 0,
"norm_type": "weight_norm",
"normalize": false,
"num_beam_groups": 1,
"num_beams": 1,
"num_filters": 32,
"num_lstm_layers": 2,
"num_residual_layers": 1,
"num_return_sequences": 1,
"output_attentions": false,
"output_hidden_states": false,
"output_scores": false,
"overlap": null,
"pad_mode": "reflect",
"pad_token_id": null,
"prefix": null,
"problem_type": null,
"pruned_heads": {},
"remove_invalid_values": false,
"repetition_penalty": 1.0,
"residual_kernel_size": 3,
"return_dict": true,
"return_dict_in_generate": false,
"sampling_rate": 24000,
"sep_token_id": null,
"suppress_tokens": null,
"target_bandwidths": [
1.5,
3.0,
6.0,
12.0,
24.0
],
"task_specific_params": null,
"temperature": 1.0,
"tf_legacy_loss": false,
"tie_encoder_decoder": false,
"tie_word_embeddings": true,
"tokenizer_class": null,
"top_k": 50,
"top_p": 1.0,
"torch_dtype": "float32",
"torchscript": false,
"transformers_version": "4.31.0.dev0",
"trim_right_ratio": 1.0,
"typical_p": 1.0,
"upsampling_ratios": [
8,
5,
4,

```
    2
   ],
   "use_bfloat16": false,
   "use_causal_conv": true,
   "use_conv_shortcut": true
  },
  "fine_acoustics_config": {
   "_name_or_path": "",
   "add_cross_attention": false,
   "architectures": [
    "BarkFineModel"
   ],
   "bad_words_ids": null,
   "begin_suppress_tokens": null,
   "bias": false,
   "block_size": 1024,
   "bos_token_id": null,
   "chunk_size_feed_forward": 0,
   "cross_attention_hidden_size": null,
   "decoder_start_token_id": null,
   "diversity_penalty": 0.0,
   "do_sample": false,
   "dropout": 0.0,
   "early_stopping": false,
   "encoder_no_repeat_ngram_size": 0,
   "eos_token_id": null,
   "exponential_decay_length_penalty": null,
   "finetuning_task": null,
   "forced_bos_token_id": null,
   "forced_eos_token_id": null,
   "hidden_size": 1024,
   "id2label": {
    "0": "LABEL_0",
    "1": "LABEL_1"
   },
   "initializer_range": 0.02,
   "input_vocab_size": 1056,
   "is_decoder": false,
   "is_encoder_decoder": false,
   "label2id": {
    "LABEL_0": 0,
    "LABEL_1": 1
   },
   "length_penalty": 1.0,
   "max_length": 20,
   "min_length": 0,
   "model_type": "fine_acoustics",
   "n_codes_given": 1,
   "n_codes_total": 8,
   "no_repeat_ngram_size": 0,
   "num_beam_groups": 1,
   "num_beams": 1,
   "num_heads": 16,
   "num_layers": 24,
   "num_return_sequences": 1,
   "output_attentions": false,
   "output_hidden_states": false,
   "output_scores": false,
   "output_vocab_size": 1056,
   "pad_token_id": null,
```

```
  "prefix": null,
  "problem_type": null,
  "pruned_heads": {},
  "remove_invalid_values": false,
  "repetition_penalty": 1.0,
  "return_dict": true,
  "return_dict_in_generate": false,
  "sep_token_id": null,
  "suppress_tokens": null,
  "task_specific_params": null,
  "temperature": 1.0,
  "tf_legacy_loss": false,
  "tie_encoder_decoder": false,
  "tie_word_embeddings": true,
  "tokenizer_class": null,
  "top_k": 50,
  "top_p": 1.0,
  "torch_dtype": "float32",
  "torchscript": false,
  "transformers_version": "4.31.0.dev0",
  "typical_p": 1.0,
  "use_bfloat16": false,
  "use_cache": true
},
"initializer_range": 0.02,
"model_type": "bark",
"semantic_config": {
  "_name_or_path": "",
  "add_cross_attention": false,
  "architectures": [
    "BarkSemanticModel"
  ],
  "bad_words_ids": null,
  "begin_suppress_tokens": null,
  "bias": false,
  "block_size": 1024,
  "bos_token_id": null,
  "chunk_size_feed_forward": 0,
  "cross_attention_hidden_size": null,
  "decoder_start_token_id": null,
  "diversity_penalty": 0.0,
  "do_sample": false,
  "dropout": 0.0,
  "early_stopping": false,
  "encoder_no_repeat_ngram_size": 0,
  "eos_token_id": null,
  "exponential_decay_length_penalty": null,
  "finetuning_task": null,
  "forced_bos_token_id": null,
  "forced_eos_token_id": null,
  "hidden_size": 1024,
  "id2label": {
    "0": "LABEL_0",
    "1": "LABEL_1"
  },
  "initializer_range": 0.02,
  "input_vocab_size": 129600,
  "is_decoder": false,
  "is_encoder_decoder": false,
  "label2id": {
```

```
    "LABEL_0": 0,
    "LABEL_1": 1
   },
   "length_penalty": 1.0,
   "max_input_semantic_length": 256,
   "max_length": 20,
   "min_length": 0,
   "model_type": "semantic",
   "no_repeat_ngram_size": 0,
   "num_beam_groups": 1,
   "num_beams": 1,
   "num_heads": 16,
   "num_layers": 24,
   "num_return_sequences": 1,
   "output_attentions": false,
   "output_hidden_states": false,
   "output_scores": false,
   "output_vocab_size": 10048,
   "pad_token_id": null,
   "prefix": null,
   "problem_type": null,
   "pruned_heads": {},
   "remove_invalid_values": false,
   "repetition_penalty": 1.0,
   "return_dict": true,
   "return_dict_in_generate": false,
   "sep_token_id": null,
   "suppress_tokens": null,
   "task_specific_params": null,
   "temperature": 1.0,
   "tf_legacy_loss": false,
   "tie_encoder_decoder": false,
   "tie_word_embeddings": true,
   "tokenizer_class": null,
   "top_k": 50,
   "top_p": 1.0,
   "torch_dtype": "float32",
   "torchscript": false,
   "transformers_version": "4.31.0.dev0",
   "typical_p": 1.0,
   "use_bfloat16": false,
   "use_cache": true
  },
  "torch_dtype": "float32",
  "transformers_version": null
}
```