

Міністерство освіти і науки України
Харківський національний університет імені В. Н. Каразіна
Факультет комп'ютерних наук
Кафедра теоретичної та прикладної системотехніки

«Затверджую»

Зав. кафедри теоретичної та
прикладної системотехніки

д.т.н., проф. С. І. Шматков

«___» _____ 2024 р

Пояснювальна записка

до кваліфікаційної роботи
бакалавра

на тему: «Telegram-бот генерації рецептів з використанням штучного
інтелекту»

Захищено на засіданні
Атестаційної комісії № 44
протокол №__ від __.06.2024 р.
Оцінка _____ / _____
Голова Атестаційної комісії
_____ **Скоб О. Ю.**
(підпис) (прізвище та ініціали)

Виконала:
студентка 4 курсу, групи КУ– 41
Галузь знань: 15 – Автоматизація та
приладобудування
Спеціальність: 151 – «Автоматизація та
комп'ютерно-інтегровані технології»
Василенко Анастасія Сергіївна

(підпис)

Керівник:
Старший викладач ЗВО кафедри ТПС
Павлов Анатолій Миколайович

Рецензент:
доцент ЗВО кафедри БІСТ, канд. техн.
наук

Колованова Євгенія Павлівна
(підпис)

Харків – 2024

АНОТАЦІЯ

Пояснювальна записка до кваліфікаційної роботи бакалавра складається зі вступу, трьох розділів, висновків, списку використаних джерел і трьох додатків. Загальний обсяг роботи складає 91 сторінки, із яких 50 сторінок основної частини з 19 рисунком, 4 таблицями, 17 найменуваннями списку використаних джерел та чотирма додатками.

Метою кваліфікаційної роботи є розробка та дослідження телеграм-бота для генерації рецептів з використанням штучного інтелекту. Основною метою є створення інноваційного інструменту для зручного та ефективного отримання персоналізованих рецептів.

Об'єкт дослідження – процес створення та використання телеграм-бота для генерації рецептів. Вивчаються алгоритми ШІ для аналізу даних та генерації рецептів, методи взаємодії з користувачами через Telegram, а також реалізація бота та покращення якості рецептів.

Предмет дослідження – телеграм-бот для генерації рецептів, що використовує штучний інтелект. Дослідження охоплює архітектуру, алгоритми генерації рецептів, взаємодію з користувачами та технічні аспекти реалізації.

Завдання, яке вирішується в кваліфікаційній роботі полягає в розробці телеграм-бота, що використовує штучний інтелект для генерації персоналізованих рецептів, забезпечуючи користувачам зручний доступ до кулінарних рекомендацій.

Область застосування – розробка веб-додатків.

Ключові слова: ШІ, GPT, AGI, Python, розробка.

ABSTRACT

An explanatory note to the master's attestation work is created in the introduction, three sections, conclusions, a list of sources used and four additional substances.

The total volume of the work is 91 pages, including 50 pages of the main part with 19 figures, 4 tables, 17 references and four appendices.

The purpose of the qualification work is to develop and research a telegram bot for generating recipes using artificial intelligence. The main goal is to create an innovative tool for convenient and efficient receipt of personalized recipes.

The object of research is the process of creating and using a telegram bot for recipe generation. AI algorithms for data analysis and recipe generation, methods of interaction with users via Telegram, as well as bot implementation and recipe quality improvement are studied.

The subject of the research is a Telegram bot for generating recipes using artificial intelligence. The research covers architecture, recipe generation algorithms, user interaction, and technical aspects of implementation.

The task solved in the qualification work is to develop a Telegram bot that uses artificial intelligence to generate personalized recipes, providing users with convenient access to culinary recommendations.

The field of application is web application development.

Keywords: AI, GPT, AGI, Python, development.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	5
ВСТУП.....	7
РОЗДІЛ 1. ДОСЛІДЖЕННЯ «ШТУЧНОГО ІНТЕЛЕКТУ»	9
1.1 Визначення штучного інтелекту.....	9
1.2 ШІ як системна технологія.....	13
1.3 Застосування штучного інтелекту	16
1.4 Переваги ризику використання штучного інтелекту.....	19
Висновки за розділом 1.....	22
РОЗДІЛ 2. АНАЛІЗ ТЕХНОЛОГІЙ ДЛЯ ПОБУДОВИ ЧАТ-БОТУ.....	23
2.1 Різновиди чат-ботів та їх побудова	23
2.2 Архітектура чат-боту	28
2.3 Платформи та фреймворки для розробки чат-ботів	34
2.4 Вибір технології для телеграм-бота генерації рецептів	37
Висновки за розділом 2.....	39
РОЗДІЛ 3. РОЗРОБКА МОДЕЛІ ТА КОМП'ЮТЕРНА РЕАЛІЗАЦІЯ	40
3.1 Побудова структурної схеми.....	40
3.2 Переваги та недоліки мови програмування Python для написання чат- ботів	44
3.3 Тестування телеграм-боту «AICOOKER»	50
Висновки за розділом 3.....	59
ВИСНОВКИ.....	60
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	61
ДОДАТКИ.....	63

ПЕРЕЛІК СКОРОЧЕНЬ І УМОВНИХ ПОЗНАЧЕНЬ

ШІ	–	штучний інтелект;
ЄС	–	Європейська комісія;
ML	–	(англ. Machine Learning) машинне навчання;
DL	–	(англ. Deep Learning) глибоке навчання;
AGI	–	(англ. Artificial General Intelligence) загальний штучний інтелект;
AI HLEG	–	(англ. High-Level Expert Group on Artificial Intelligence) група високого рівня з питань штучного інтелекту;
ЄС	–	Європейський Союз;
GPT	–	(англ. Generative Pre-trained Transformer) генеративний попередньо навчений трансформер;
ЗПТ	–	загальний попередньо навчений трансформер;
NLP	–	(англ. Natural Language Processing) обробка природної мови;
ІоТ	–	(англ. Internet of Things) Інтернет течей;
ЗМІ	–	засоби масової інформації;
NLU	–	(англ. Natural Language Understanding) розуміння природної мови;
LSTM	–	(англ. Long Short-Term Memory) довга короткочасна пам'ять;
SVM	–	(англ. Support Vector Machine) метод опорних векторів;
NLTK	–	(англ. Natural Language Toolkit) інструментарій природної мови;
JS	–	JavaScript;
HTTP	–	(англ. Hypertext Transfer Protocol) протокол передачі гіпертексту;
AIML	–	(англ. Artificial Intelligence Markup Language) мова штучного інтелекту;
IDE	–	(англ. Integrated Development Environment) інтегроване середовище розробки;
IBM	–	International Business Machines Corporation;

- СМП – система машинного перевода;
- RNN – (англ. Recurrent Neural Network) рекуррентна нейронна мережа.

ВСТУП

У сучасному світі, де швидкість і зручність стали необхідними компонентами повсякденного життя, важливо мати доступ до інноваційних технологій, що полегшують наші щоденні обов'язки. Однією з таких технологій є штучний інтелект, який здатний забезпечити персоналізований підхід до широкого спектру завдань, від розваги до робочих процесів. У цьому контексті створення телеграм-бота для генерації рецептів з використанням штучного інтелекту стає не лише актуальним, але й надзвичайно корисним для сучасного користувача.

Завдяки поєднанню технології телеграму, яка є однією з найпопулярніших месенджерних платформ, та штучного інтелекту, який може аналізувати велику кількість даних та навчатися відповідати на індивідуальні потреби кожного користувача, такий бот може стати невід'ємною частиною щоденного харчування та кулінарного досвіду. Його можливості не обмежуються лише наданням рецептів — він може надавати поради щодо збалансованого харчування, враховуючи індивідуальні особливості та потреби кожного користувача.

Актуальність роботи. У світі, де технології швидко розвиваються, а ритм життя стає все більш інтенсивним, багато людей шукають способи прискорити та спростити процес готування їжі, не втрачаючи при цьому якості та смаку. Однак, разом з цим, зростає і попит на здорові та смачні страви, які відповідають індивідуальним дієтичним потребам та перевагам користувачів. У такому контексті телеграм-бот, який використовує штучний інтелект для генерації рецептів, стає важливим інструментом, що поєднує в собі інноваційні технології та кулінарну творчість. Такий бот може відповідати на потреби різних категорій користувачів. Для тих, хто шукає прості та швидкі рішення, він може пропонувати прості рецепти з мінімальною кількістю інгредієнтів та швидкими кулінарними прийомами. У той же час, для тих, хто

цінує різноманіття та оригінальність, бот може генерувати складні та нестандартні страви, які задовольнять навіть найвибагливіших гурманів.

Метою дослідження є розробка та дослідження телеграм-бота для генерації рецептів з використанням штучного інтелекту. Основною метою є створення інноваційного інструменту, який забезпечить користувачам зручний та ефективний спосіб отримання персоналізованих рецептів для приготування їжі.

Об'єкт дослідження – процес створення та використання телеграм-бота для генерації рецептів з використанням штучного інтелекту. Вивчаються алгоритми штучного інтелекту, які використовуються для аналізу даних та генерації рецептів, а також методи взаємодії з користувачами через месенджер Telegram. Також увага зосереджується на реалізації бота та його можливостях у плані покращення доступності та якості рецептів, які він надає користувачам.

Предмет дослідження – телеграм-бот, призначений для генерації рецептів з використанням штучного інтелекту. Дослідження охоплює всі аспекти функціонування бота, включаючи його архітектуру, алгоритми генерації рецептів, взаємодію з користувачами та технічні аспекти реалізації.

Методи дослідження: порівняння різних алгоритмів генерації рецептів на основі штучного інтелекту, аналіз користувацької зручності та сприйняття бота, а також дослідження впливу рецептів, створених ботом, на харчові пристрасті та здоров'я.

Завдання дослідження

1. Розробка системи рекомендацій: створення системи, яка аналізує вподобання користувачів та рекомендує їм рецепти на основі їхніх
2. Оптимізація швидкості та відповідей бота: проведення дослідження та оптимізація швидкості відповідей бота для забезпечення більшої ефективності та зручності використання.
3. Створення мультимедійного контенту: розробка можливості додавання до рецептів фото для полегшення їхнього відтворення.

РОЗДІЛ 1

ДОСЛІДЖЕННЯ «ШТУЧНОГО ІНТЕЛЕКТУ»

1.1 Визначення штучного інтелекту

Дати визначення ШІ непросто. Насправді не існує загальноприйнятого визначення цього поняття. Використовується безліч різних визначень, і це може легко призвести до плутанини. Тому важливо прояснити, як ми використовуємо цей термін. Ми почнемо з обговорення різних визначень ШІ, а потім пояснимо, на якому з них ми зупинилися. Розмаїття існуючих визначень не є наслідком необережності, а притаманне самому феномену ШІ.

У найширшому визначенні ШІ ототожнюється з алгоритмами. Однак це не дуже корисний підхід для нашого аналізу. Алгоритми з'явилися раніше за ШІ і широко використовувалися за межами цієї галузі. Термін "алгоритм" походить від імені перського математика IX століття Мухаммеда ібн Муси аль-Харізмі і означає конкретну інструкцію для розв'язання задачі або виконання обчислень. Якби ми визначили ШІ просто як використання алгоритмів, він включав би в себе багато інших видів діяльності, таких як робота кишенькового калькулятора або навіть інструкції в кулінарній книзі.

У найсуворішому визначенні ШІ означає імітацію комп'ютерами інтелекту, притаманного людині. Пуристи вказують на те, що багато сучасних додатків все ще відносно прості, а отже, не є справжнім ШІ. Тому це визначення також не підходить для нашого звіту; його використання означало б, що ШІ наразі не існує. Ми фактично дамо визначення явищу, якого не існує.

Найпоширеніше визначення ШІ - це технологія, яка дозволяє машинам імітувати різні складні людські навички. Це, однак, не дає багато чого для подальших роздумів. Насправді, це не більше, ніж просто передача терміну "штучний інтелект" іншими словами. Доки ці "складні людські навички" не визначені, залишається незрозумілим, що саме являє собою ШІ. Те саме стосується визначення ШІ як виконання комп'ютерами складних завдань у складних умовах[1].

Інші визначення йдуть далі в поясненні цих навичок і завдань. Наприклад, комп'ютерний науковець Нільс Джон Нільссон описує технологію, яка "функціонує належним чином і з передбаченням у своєму середовищі". Інші говорять про здатність сприймати, переслідувати цілі, ініціювати дії та вчитися на основі зворотного зв'язку. Подібне визначення було запропоновано Групою експертів високого рівня з питань штучного інтелекту. Європейської Комісії (ЄК): "Системи, які демонструють розумну поведінку, аналізуючи навколишнє середовище і виконуючи дії - з певним ступенем автономії - для досягнення конкретних цілей" [1].

Не дивно, що ШІ так складно дати чітке визначення. Зрештою, це імітація або симуляція того, що ми самі ще не до кінця розуміємо: людського інтелекту. Це вже давно є предметом дослідження психологів, поведінкових науковців та неврологів, серед інших. Ми багато знаємо про інтелект і людський мозок, але ці знання далеко не повні, і немає єдиної думки щодо того, що саме являє собою людський інтелект. Доки цього не станеться, неможливо точно сказати, як цей інтелект можна імітувати штучно.

Це відображає закономірність, що повторюється в історії ШІ: уявлення людей про те, що є складною формою людського інтелекту, еволюціонує разом зі зростаючими навичками наших комп'ютерів. Те, що раніше вважалося першим прикладом штучного інтелекту, з часом деградує до простого обчислення, яке більше не заслуговує на назву ШІ. Памела МакКордук називає це "ефектом штучного інтелекту": як тільки комп'ютер з'ясовує, як щось робити, люди заявляють, що це "просто розрахунок", а не справжній інтелект. За словами Ніка Бострома, директора Оксфордського інституту управління Інтернетом, ШІ включає в себе все, що вражає нас у будь-який момент часу. Коли нас це більше не вражає, ми просто називаємо це програмним забезпеченням. Прикладом може слугувати програма для гри в шахи на смартфоні. Таким чином, труднощі у визначенні ШІ не є результатом якогось недоліку чи необережності, а скоріше виникають через те, що ми довго не могли точно визначити, який саме інтелект ми хочемо штучно імітувати.

Ця перспектива також проливає світло на парадокс Моравеца, згаданий вище. Розпізнавати обличчя легко для людей, але складно для комп'ютерів. Це пояснюється тим, що розпізнавання інших людей було критично важливим для нашого еволюційного виживання, і тому наш мозок навчився робити це без роздумів.

Нарешті, ШІ також часто ототожнюють з новітніми технологіями. Як ми побачимо далі, ШІ набрав величезних обертів за останні роки. Одним із головних рушіїв цього став прогрес у конкретній галузі - "machine learning" (ML), де інновації призвели до того, що зараз називають "deep learning" (DL). Саме ця технологія стоїть за останніми досягненнями, такими як комп'ютери, здатні розпізнавати обличчя і грати в такі ігри, як Go. На відміну від більш традиційних підходів, коли комп'ютерні системи застосовують фіксовані правила, алгоритми ML і DL можуть розпізнавати закономірності в даних. Ми також говоримо тут про "алгоритми, що самонавчаються". Багато людей, які сьогодні говорять про штучний інтелект, насправді мають на увазі ці алгоритми, і часто саме DL. Зосередження уваги на цій технології є важливим, оскільки деякі нагальні питання, пов'язані зі штучним інтелектом, є особливо актуальними саме тут (наприклад, проблеми пояснюваності).

Враховуючи всі різні визначення, обговорювані тут і в інших джерелах, ми зупинилися на відкритому визначенні ШІ. У цьому відношенні важливими є два міркування. По-перше, для цілей цього звіту було б нерозумно обмежувати визначення ШІ конкретною частиною технології. Якби ми, наприклад, обмежилися "deep learning", про яке йшлося вище, то проігнорували б той факт, що багато актуальних проблем відіграють певну роль і в інших сферах ШІ, наприклад, у логічних системах. Одним з таких прикладів є питання "чорного ящика". Крім того, більшість застосувань ШІ, що використовуються урядами, не ґрунтуються на таких передових технологіях, як DL, і все ж мають багато важливих питань, які необхідно розглянути в цьому звіті. Занадто вузьке визначення виведе їх за межі цього дослідження. Хоча розробки в галузі DL дійсно призвели до великого стрибка

вперед, більше того, в кінці наступного розділу ми також вкажемо на деякі недоліки цього методу. Насправді, майбутні досягнення в ШІ цілком можуть прийти з інших галузей. Для цього важливо мати відкрите визначення ШІ [3].

По-друге, як зазначалося вище, природа цієї наукової дисципліни неминуче означає, що наше визначення ШІ буде змінюватися з часом. Замість того, щоб розглядати ШІ як дисципліну, яка може бути чітко розмежована, з нескладними визначеннями і гнучкими методологіями, корисніше розглядати його як складну і різноманітну сферу, сфокусовану на певному горизонті.

Точкою на цьому горизонті є розуміння і моделювання всіх людських інтелектуальних навичок. Цю мету також називають "штучним загальним інтелектом" або AGI (інші назви - "сильний ШІ" і "повний ШІ"). Однак поки що невідомо, чи вдасться коли-небудь досягти цієї мети за такого загального визначення ШІ. Більшість експертів вважають, що до неї ще щонайменше кілька десятиліть - якщо вона взагалі буде досягнута.

Для цілей цього звіту визначення ШІ як імітації повного людського інтелекту малоприслатне. Нам потрібне визначення, яке охоплює весь спектр застосувань, що знаходять свій шлях на практиці сьогодні і в найближчому майбутньому. Визначення, наведене в AI HLEG, надає необхідну свободу в застосуванні. Описуючи ШІ як "системи, які демонструють інтелектуальну поведінку, аналізуючи навколишнє середовище і виконуючи дії - з певним ступенем автономності - для досягнення конкретних цілей", воно охоплює всі додатки, які ми зараз кваліфікуємо як ШІ, і в той же час забезпечує простір для майбутніх змін в цій кваліфікації. Поряд з передовими технологіями машинного навчання і глибокого навчання, це визначення також допускає інші технології, включаючи більш традиційні підходи, згадані вище, які використовуються багатьма державними органами. Коротше кажучи, це визначення є достатньо суворим, щоб відрізнити ШІ від алгоритмів і цифрових технологій загалом, і водночас достатньо відкритим, щоб включати майбутні розробки. На рис. 1.1 наведено огляд розглянутих визначень і визначення AI HLEG[3].

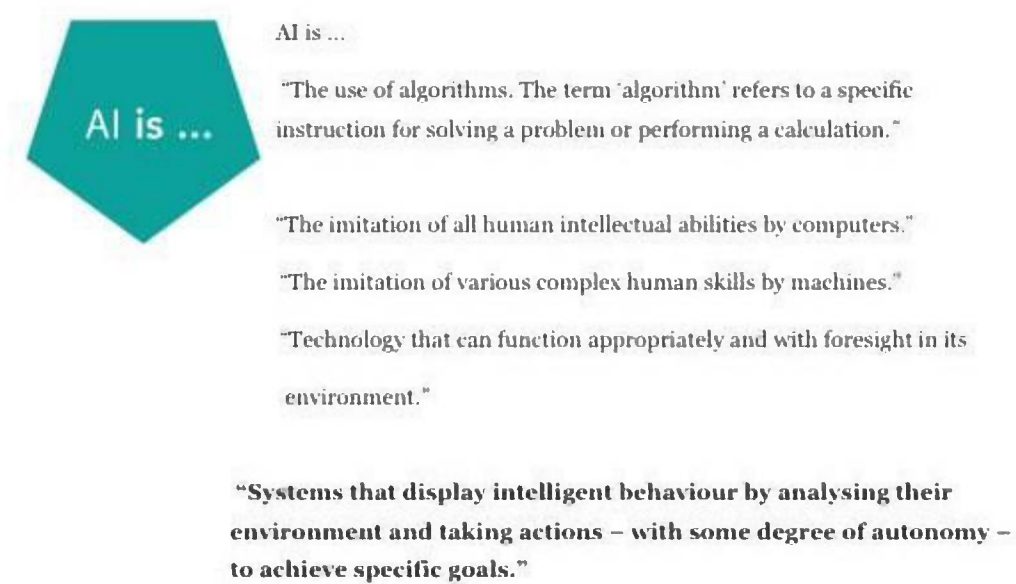


Рис. 1.1 - Різноманітні визначення ШІ

1.2 Штучний інтелект як системна технологія

Тепер, коли ми знаємо, що таке ШІ, і побачили, як ця технологія за останні роки перейшла з лабораторії в суспільство, звернемо увагу на процес впровадження ШІ в суспільство. Що потрібно для того, щоб впровадити ШІ в наше суспільство? Щоб відповісти на це питання, в цій главі представлено концепцію, в рамках якої ШІ можна розглядати як особливий тип технології, а саме як системну технологію, що має низку історичних прецедентів. Розглядаючи ШІ таким чином, ми можемо зробити різні висновки з історії інших системних технологій. Це, в свою чергу, дає основу для роздумів про те, що нам потрібно робити зі штучним інтелектом і як ми можемо вирішити численні проблеми, пов'язані з ним. Ми не маємо наміру стверджувати, що історія завжди повторюється або що технологічний розвиток має детерміновані характеристики. Ми не встановлюємо жорстких рамок, а визначаємо загальні закономірності, які проливають світло на сьогодні, визнаючи при цьому, що минуле і сьогодні відрізняються. Застосовуючи такий підхід, ми прагнемо зазирнути за межі поточної ситуації, а отже, за межі примх дня.

Різні відомі коментатори проводять паралелі між ІІІ та іншими технологіями. За словами дослідника Ендрю Нга, вплив ІІІ "можна порівняти з впливом електрики століття тому". Генеральний директор Google Сундар Пічаї та його попередник Ерік Шмідт також порівнювали ІІІ з електрикою. Пічаї зайшов так далеко, що порівняв ІІІ з вогнем.

У світі політики також часто проводять подібні порівняння. У статті про стратегічні наслідки ІІІ Майкл Горвіц писав, що це не ізольована технологія, а подібна до технологій загального призначення, таких як електрика і двигун внутрішнього згоряння. Широта потенційних застосувань ІІІ також підкреслюється Центром європейської політичної стратегії Європейської комісії: "Важко уявити будь-який сектор суспільства, який не буде трансформований за допомогою ІІІ в найближчі роки". Відповідно, ЄС розглядає ІІІ як "ключову технологію" [4].

Вчені вже давно цікавляться тим, як різні типи технологій здійснюють загальний вплив на економіку та суспільство. Першим прикладом є хвильова теорія Кондратьєва, зокрема, розроблена Йозефом Шумпетером. Він помітив, що періоди високих темпів економічного зростання чергуються з періодами низьких темпів зростання, і цю закономірність він пояснив впливом нових технологій; набори нових технологій періодично прискорюють зростання, після чого ефект поступово згасає з часом. На думку Шумпетера, такий динамізм був притаманний капіталістичній ринковій економіці: "важливо розуміти, що капіталізм - це еволюційний процес... "індустріальна мутація"... постійно вносить революційні зміни в структуру економіки зсередини, постійно руйнуючи стару структуру і створюючи нову". Такі міркування лежать в основі знайомої нам концепції творчого руйнування.

У своїй промові під час вручення Нобелівської премії з економіки в 1971 році Саймон Кузнець представив ідею "епохальних інновацій", які визначають періоди великого економічного розвитку. Вчені-інноваційники Карлота Перес і Кріс Фріман писали про подібне явище, яке вони називають "новими технологічними системами" і "технологічними революціями". Нова

технологічна система - це потужний і помітний кластер нових і динамічних технологій, продуктів і галузей, які призводять до значних змін в економіці і, зрештою, до економічного зростання. Перес виділила п'ять таких кластерів з часів промислової революції, включаючи епохи парової енергії та залізниць, сталі та електроенергії, нафти, автомобілів і масового виробництва, а також інформації та телекомунікацій. Вона стверджує, що кожна з цих епох принесла свою "техніко-економічну парадигму": спосіб мислення і дії, що призвело до того, що відповідні технології стали невід'ємною частиною структури суспільства. Алессандро Нуволарі зробив суттєве доповнення до теорій Переса, підкресливши, що спостережувані ефекти пов'язані не стільки з окремими технологіями, скільки з блоками радикальних інновацій, які разом призводять до революції. Деякі дослідники, відповідно, вважають, що інновація полягає не в розробці принципово нових речей, а в комбінуванні речей, які вже існують.

Трансформаційний вплив ІІІ на суспільство можна зрозуміти через призму технологій загального призначення (GPT). Це технології з широким, універсальним застосуванням у різних галузях, що сприяють економічним і соціальним змінам. Бреснахан і Трайтенберг (1992) виділили три критерії для визначення ЗПТ: поширеність у різних секторах, постійний потенціал для технічного вдосконалення та підвищення економічної продуктивності за рахунок інновацій. Хоча єдиного визначення не існує, історичні приклади, такі як друкарський верстат і пароплав, демонструють технології зі значним впливом на суспільство, незважаючи на їхню специфіку [5].

Нещодавні дослідження чітко пов'язують ІІІ з концепцією GPT. У книзі "Економіка штучного інтелекту" (2019) та дисертації Джейд Люн з Оксфордського університету ІІІ розглядається як стратегічна GPT, подібна до аерокосмічних технологій та біотехнологій. Ці дискусії підкреслюють складну взаємодію між урядами, бізнесом і дослідницькими спільнотами у формуванні траєкторії розвитку і впровадження ІІІ.

Крім того, такі роботи, як дослідження цифрових технологій Карлоти Перес та аналіз соціальних наслідків використання двигуна внутрішнього згоряння Сари А. Со, дають додаткове уявлення про ширший контекст технологічної еволюції. Проводячи паралелі з іншими трансформаційними технологіями, такими як ГМО та нанотехнології, дослідники поглиблюють наше розуміння потенційного впливу ІІІ та суспільної динаміки [6].

1.3 Застосування штучного інтелекту

У 21 столітті, коли світ праці продовжує розвиватися, а цифрові технології вдосконалюються, поширення штучного інтелекту на робочому місці стало предметом численних дискусій. Одні вважають ІІІ цінним інструментом, який може підвищити продуктивність і покращити досвід працівників, інші занепокоєні тим, що він може призвести до масової втрати робочих місць, порушити права працівників і створити етичні проблеми. Наприклад, чат-боти і віртуальні асистенти часто використовуються для автоматизації рутинних завдань, які раніше виконувалися людьми, що може призвести до скорочення робочих місць. А в міру того, як ІІІ стає більш досконалим, він може навіть замінити людей на висококваліфікованих посадах, таких як аналіз даних або обслуговування клієнтів [7].

Штучний інтелект - це широка галузь досліджень, яка включає в себе безліч теорій, методів і технологій, а також наступні основні підгалузі:

1. Машинне навчання автоматизує побудову аналітичних моделей. Воно використовує методи нейронних мереж, статистики, дослідження операцій та фізики для пошуку прихованої інформації в даних.

2. Нейронна мережа - це тип машинного навчання, який складається з взаємопов'язаних блоків (наприклад, нейронів), які обробляють інформацію, реагуючи на зовнішні входи, передаючи інформацію між кожним блоком. Нейронні мережі - це математичні моделі, структура яких значною мірою нагадує структуру мозку. Кожен нейрон нейронної мережі - це математична

функція, яка отримує дані через вхід, перетворює їх у більш зручну для сприйняття форму, а потім видає їх через вихід.

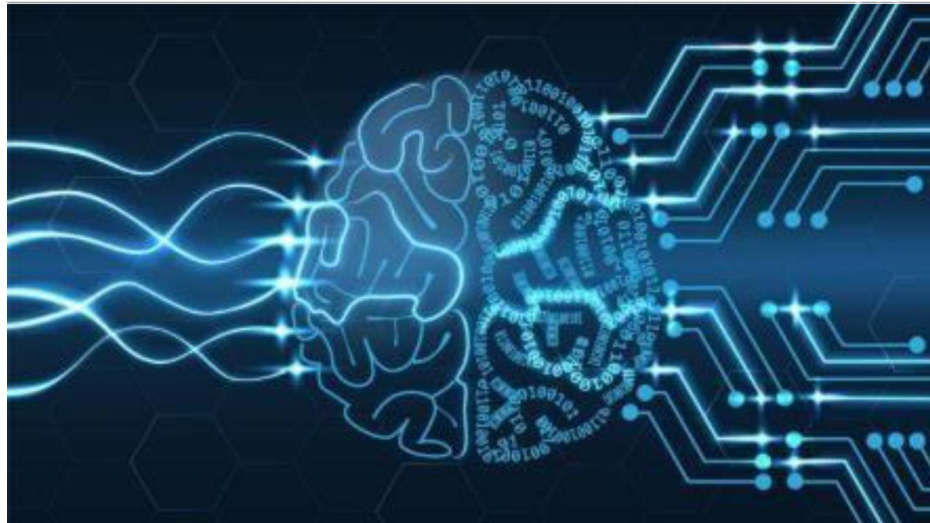


Рис. 1.2 - Машинне навчання передбачає навчання комп'ютера виконувати завдання без явного покрокового програмування.

3. Глибоке навчання використовує величезні нейронні мережі з багатьма шарами процесорів, використовуючи переваги обчислювальних потужностей і вдосконалених методів навчання для вивчення складних закономірностей у великих обсягах даних. Серед поширених застосувань - розпізнавання зображень і мови.

4. Комп'ютерний зір покладається на розпізнавання образів і глибоке навчання, щоб розпізнати, що на зображенні або відео. Коли машини можуть обробляти, аналізувати і розуміти зображення, вони можуть захоплювати зображення або відео в реальному часі і інтерпретувати їх оточення. CV допомагає системам розуміти та інтерпретувати візуальний світ таким чином, щоб викликати відповідні дії. Він навчає комп'ютери інтерпретувати та розуміти візуальний світ.

5. Обробка природної мови (NLP) - це здатність комп'ютерів аналізувати, розуміти і генерувати людську мову. NLP - це взаємодія

природною мовою, що дозволяє людям спілкуватися з комп'ютерами, використовуючи звичайну, повсякденну мову.

Наразі існує високий попит на можливості ШІ в найрізноманітніших галузях і системах, які можна використовувати для автоматизації, навчання, юридичної допомоги, сповіщення про ризики та досліджень. Конкретні сфери застосування ШІ в промисловості включають:

1. Охорона здоров'я. Існує багато застосувань ШІ в медичних установах, які можуть виконувати різні рутинні завдання. Наприклад, машини зі штучним інтелектом можуть "зчитувати" персоналізовані медичні дані та рентгенівські знімки пацієнтів з різними захворюваннями (наприклад, мамограми молочної залози, дані, отримані за допомогою рентгеноміки та біопсійних слайдів, а також низькодозових комп'ютерних томографій раку легенів). Крім того, ШІ може виступати в ролі персонального медичного асистента. Наприклад, ШІ може виступати в ролі життєвого тренера (наприклад, нагадувати пацієнтам про прийом таблеток), нагадувати людям про фізичні вправи та їх тривалість, або пропонувати їсти здорову їжу для зменшення ожиріння.

2. Роздрібна торгівля. Комп'ютери зі штучним інтелектом можуть надавати можливості віртуального шопінгу, пропонуючи персоналізовані рекомендації та обговорюючи варіанти покупки зі споживачем. Компанії, що мають проблеми з управлінням запасами та організаційними проблемами (одяг, інструменти, обладнання тощо), а також технології планування сайтів також можуть бути вдосконалені за допомогою ШІ.

3. Виробництво. ШІ може аналізувати заводські дані Інтернету речей (IoT). Інтернет речей (IoT) зростає в геометричній прогресії, але безпека проектів і розгортань IoT залишається перешкодою для багатьох організацій. Одним з фундаментальних компонентів безпеки Інтернету речей є забезпечення надійної ідентифікації пристроїв і сервісів, які можуть взаємодіяти в безпечних екосистемах [8].

4. Медико-біологічні науки. Від забезпечення безпеки ліків до прискорення виведення нових методів лікування на ринок - технології

штучного інтелекту можуть розкрити весь потенціал даних для вирішення найбільших проблем у сфері охорони здоров'я, що стоять перед людством сьогодні і в майбутньому.

5. Банківська справа. Штучний інтелект підвищує швидкість, точність та ефективність людських зусиль у фінансових установах. Методи ШІ можна використовувати для визначення того, які банківські операції можуть бути шахрайськими.

6. Державний сектор. Штучний інтелект може зробити розумні міста розумнішими. Він може підтримувати національну оборону, забезпечуючи готовність до місій і прогнозоване технічне обслуговування. Європа витратила до 700 мільйонів євро на робототехніку та державно-приватне партнерство в контексті ШІ. Як наслідок, з'являється багато додатків на основі штучного інтелекту для державного сектору, які обіцяють велику цінність з точки зору робочої сили і продуктивності.

1.4 Переваги ризику використання штучного інтелекту

З моменту створення ШІ серед науковців точаться суперечки щодо практичного застосування ШІ в різних сферах, а також щодо того, чи є він остаточним підходом до просування людського суспільства до нових вершин успіху. Інші вважають, що штучний інтелект приведе людину на темний шлях до самознищення. Але більшість вчених вважають, що ШІ відіграватиме ключову позитивну роль у світі, в якому ми живемо.

Протягом всієї історії людства словом "розумний" ми називали унікальні людські навички та здібності, наприклад, розпізнавання знайомого обличчя, розв'язання математичної задачі, володіння музичним інструментом. Однак за останні 50 років цей факт кардинально змінився. Останні технологічні революції привели до появи широкого спектру нових наукових інновацій, одна з яких отримала назву "штучний інтелект" (ШІ) - передова галузь комп'ютерних наук, що займається розробкою динамічних систем [9].

На початку 2023 року було оприлюднено відкритий лист (29.3.2023), який підписали понад 1100 осіб (науковці, відомі технологи та керівники, головні виконавчі директори провідних корпорацій). Серед підписантів були Ілон Маск (генеральний директор Tesla) і Стів Возняк (співзасновник Apple) та інші видатні технологи і дослідники штучного інтелекту (ШІ), які підписали відкритий лист із закликом запровадити шестимісячний мораторій на розробку передових систем ШІ.

Серед 1100 підписантів листа були Емад Мостак (Emad Mostaque), засновник і генеральний директор Stability AI, компанії, яка допомогла створити популярну модель перетворення тексту в зображення StableDiffusion, і Коннор Ліхі (Connor Leahy), генеральний директор Conjecture, ще однієї лабораторії штучного інтелекту. Еван Шарп, співзасновник Pinterest, і Кріс Ларсон, співзасновник криптовалютної компанії Ripple, також поставили свої підписи. Піонер глибинного навчання та лауреат премії Тьюрінга, комп'ютерний вчений Йошуа Бенгіо також підписався. Лист закликає технологічні компанії "негайно припинити навчання будь-яких систем штучного інтелекту, які були б потужнішими за GPT-4" - новітній штучний інтелект для обробки великої кількості мов, розроблений сан-франциською компанією OpenAI. У листі не сказано, як саме слід визначати "потужність" моделі, але в останніх досягненнях ШІ потужність, як правило, корелює з розміром моделі ШІ та кількістю спеціалізованих комп'ютерних чіпів, необхідних для її навчання.

Компанія OpenAI, що займається технологічними дослідженнями, щойно випустила оновлену версію своєї програми штучного інтелекту (ШІ), яка генерує тексти, під назвою GPT-4 (Generative Pre-trained Transformer), і продемонструвала деякі з нових можливостей мовної моделі. GPT-4 не лише може створювати більш природний текст і вирішувати завдання точніше, ніж його попередник. Він також може обробляти зображення на додаток до тексту. Але штучний інтелект все ще вразливий до тих самих проблем, що й попередні моделі GPT: прояв упередженості, переступання обмежувачів, які не

дозволяють йому говорити образливі чи небезпечні речі, а також "галюцинації" або впевнене вигадування неправдивих даних, яких немає в його навчальних даних. GPT = Генеративний попередньо навчений трансформатор.

14.3.2023 року компанія OpenAI, яка займається розробкою штучного інтелекту, випустила свою останню версію GPT-4. Система може скласти іспит на адвоката, розв'язувати логічні головоломки і навіть дати користувачеві рецепт, як використати залишки їжі на основі фотографії його холодильника, але її творці попереджають, що вона також може поширювати фейкові факти, впроваджувати небезпечні ідеології і навіть обманювати людей, змушуючи їх виконувати завдання від свого імені. OpenAI - це компанія, що стоїть за ChatGPT і виконавцем ШІ Dall-E [10].



Рис. 1.3 – Остання версія GPT-4

Більшість дослідників сходяться на думці, що надінтелектуальний ШІ навряд чи буде проявляти людські емоції, такі як любов або ненависть, і що немає підстав очікувати, що ШІ стане навмисно доброзичливим або злісним [11].

Професор Стівен Гокінг (космолог-теоретик, Кембриджський університет), один з найвидатніших британських вчених, заявив (у 2014 році), що спроби створити мислячі машини становлять загрозу самому нашому існуванню. Ілон Маск (Tesla, технічний мільярдер), Стів Возняк

(співзасновник Apple), Білл Гейтс (Microsoft) та багато інших відомих діячів науки і техніки нещодавно висловили занепокоєння в ЗМІ та у відкритих листах щодо ризиків, які несе в собі ШІ, до них приєдналися багато провідних дослідників ШІ. Чому ця тема раптом опинилася в заголовках новин відлюдей, які працюють у цій галузі?

Ідея про те, що пошуки сильного ШІ врешті-решт увінчаються успіхом, довгий час вважалася науковою фантастикою, віддаленою від нас на століття або й більше. Однак, завдяки нещодавнім проривам, багато віх ШІ, які ще п'ять років тому експерти вважали віддаленими на десятиліття, вже досягнуті, що змушує багатьох експертів серйозно ставитися до можливості появи суперінтелекту вже за нашого життя. Хоча деякі експерти все ще вважають, що до ШІ людського рівня ще століття, більшість дослідників ШІ на конференції в Пуерто-Рико в 2015 році припускали, що це станеться до 2060 року. Оскільки на завершення необхідних досліджень з безпеки можуть піти десятиліття, розумно розпочати їх вже зараз [12].

Висновки за розділом 1

Було розглянуто поняття «штучного інтелекту», що включає в себе розгляд ролі цієї технології як системної, огляд різноманітних застосувань та аналіз переваг та ризиків, пов'язаних з їхнім використанням. Визначення ШІ надає базове розуміння цього поняття, а вивчення його як системної технології допомагає розкрити його роль у сучасному технологічному ландшафті. Подальший огляд застосувань ШІ відкриває широкий спектр можливостей, включаючи області виробництва, медицини, фінансів та багато інших. Водночас, розділ розглядає переваги, які може принести використання ШІ, такі як підвищення продуктивності та ефективності, а також ризики, пов'язані з можливими негативними наслідками, такими як безробіття або порушення приватності. Розглядаючи ці аспекти, розділ надає комплексне уявлення про Штучний Інтелект та його потенціал у сучасному суспільстві.

РОЗДІЛ 2

АНАЛІЗ ТЕХНОЛОГІЙ ДЛЯ ПОБУДОВИ ЧАТ-БОТУ

2.1 Різновиди чат-ботів та їх побудова

У даному розділі ми розглянемо, як можна класифікувати чат-ботів за такими параметрами, як рівень взаємодії та спосіб генерування відповідей:

- Сфера знань;
- Послуга, що надається;
- Мета;
- Метод генерації відповідей [13].

Чат-боти класифікуються на основі знань, до яких вони мають доступ, або обсягу даних, на яких вони навчаються.

Виділяють такі класи за сферою знань:

- 1) Відкритий домен: такі боти можуть розмовляти на загальні теми і адекватно відповідати.
- 2) Закритий домен: такі боти зосереджені на певній галузі знань і можуть не відповідати на інші запитання. Наприклад, бот для бронювання ресторану не скаже вам ім'я першого чорношкірого президента Америки. Він може розповісти вам анекдот або поцікавитися, як пройшов ваш день, але від нього цього не очікують, оскільки його завдання - забронювати столик і надати інформацію про ресторан.

За послугами, що надаються, боти класифікуються на основі проксемічної класифікації, але замість фізичної близькості, вона базується на емоційній близькості бота до користувача, кількості інтимної взаємодії, яка відбувається, і, отже, також залежить від завдання, яке виконує бот.

1. Міжособистісні: до цієї категорії належать чат-боти, які перебувають у сфері спілкування, що належить до діапазону соціальної або особистої дистанції на діаграмі "Проксеміка". Боти, які надають такі послуги, як бронювання ресторанів, бронювання авіаквитків, FAQ-боти тощо. Ці чат-боти не повинні бути

компаньйонами користувача, вони повинні отримувати інформацію і передавати її користувачеві, вони є лише допоміжними засобами. Вони можуть мати індивідуальність, бути доброзичливими і, ймовірно, запам'ятовуватимуть інформацію про користувача, але вони не зобов'язані і не очікують, що це буде зроблено.

2. Внутрішньоособистісні: ці чат-боти існуватимуть в межах особистого домену користувача, як-от чат-додатки, такі як messenger, slack та whatsapp, і виконуватимуть завдання, що лежать в особистому домені користувача. Управління календарем, зберігання думки користувача тощо. Вони будуть компаньйонами користувача і розумітимуть його так само, як і людина. Можливо, вони не будуть домінуючими в поточному сценарії, але з розвитком розуміння природної мови вони незабаром з'являться.
3. Міжагентські: Такі боти будуть переважати в областях, де домінує ІОТ. У цьому випадку дві системи спілкуються одна з одною для виконання завдання. Оскільки боти стають всюдисущими, всі вони потребуватимуть певних можливостей міжботової комунікації. Незабаром виникне потреба в протоколах для міжботової комунікації. Хоча бот не може бути повністю міжагентним ботом, він може бути сервісом, який обробляє інших ботів або здійснює комунікацію, полегшуючи розробникам і користувачам інтеграцію різних сервісів в розмовну екосистему. Інтеграція Alexa-Cortana є прикладом міжагентної комунікації

За метою боти класифікуються на основі основної мети, яку вони прагнуть досягти.

1. Інформаційні: ці боти призначені для надання користувачеві інформації, яка зберігається задалегідь або доступна з фіксованого джерела. Зазвичай вони базуються на алгоритмах пошуку інформації і або отримують результат

запиту з бази даних, або виконують співставлення рядків. Найчастіше вони посилаються на статичне джерело інформації, наприклад, сторінку поширених запитань на сайті або базу даних складу з інвентарними записами.

2. Розмовні чат-боти: ці боти розмовляють з користувачем, як з іншою людиною. Їх мета - правильно відповісти на отримане речення. Тому їх часто створюють з метою продовження розмови з користувачем, використовуючи такі техніки, як перехресні запитання, ухиляння та шанобливе ставлення.
3. На основі завдань: вони виконують одне конкретне завдання, наприклад, бронюють квитки на літак або допомагають вам зробити покупки в магазині. У більшості випадків дії, необхідні для виконання завдання, заздалегідь визначені, також визначений потік подій, включаючи винятки. Боти розумні в контексті запиту інформації та розуміння вкладу користувача.

Класифікація за методом обробки вхідних даних та генерації відповідей враховує метод обробки вхідних даних і генерування відповідей.

По-справжньому інтелектуальні системи генерують відповіді і використовують розуміння природної мови для розуміння запиту. Ці системи використовуються, коли предметна область вузька і є достатньо даних для навчання системи.

Системи, засновані на правилах, використовують зіставлення шаблонів і виявляються жорсткими. Їх можна використовувати, коли кількість можливих результатів є фіксованою, а сценарії можна уявити у великій кількості.

1. Гібридні: ці системи є поєднанням правил і машинного навчання.

Прикладом може бути система, яка використовує блок-схему для

управління напрямком розмови, але надає відповіді, які генеруються за допомогою обробки природної мови.

Боти не обов'язково повинні належати виключно до тієї чи іншої категорії. Ці категорії існують у кожному боті в різних пропорціях. Наприклад, всі боти потребують певних можливостей чату, можливо, бот для магазину повинен використовувати вилучення інформації, коли йдеться про поширені запитання, і пошук на сайті, коли йдеться про надання результатів пошуку товарів. Отже, послуга, яку надає бот, може включати всі три типи алгоритмів. Візьмемо інший приклад: бот може бути закритим, але запрограмованим на світські бесіди, або бот може бути відкритим і внутрішньоособистісним, наприклад, бот продуктивності, але він також буде надзвичайно зосереджений на розмовах, пов'язаних з продуктивністю.

Ця класифікація допомагає користувачеві зрозуміти, чого очікувати, і створити образ бота в його свідомості, зменшуючи розрив між очікуваннями користувача і наданим ботом. Далі ми обговоримо архітектуру чат-ботів та різні модулі бота, які реалізують його функції.

Як було показано вище, існує декілька категорій чат-ботів. Чат-бот - це комбінація однієї або декількох категорій. Наприклад, він може бути інформативним і ґрунтуватися на правилах, або він може ґрунтуватися на завданнях, але використовувати генеративні методи. Однак усі чат-боти підпорядковуються загальному потоку або загальній архітектурі - конвеєру. На кожному рівні загального конвеєра для певної категорії використовуються алгоритми або методи, що стосуються цієї категорії. Навіть у розмаїтті чат-ботів, які отримали премію Лебнера, можна спостерігати загальний конвеєр для потоку інформації. Нижче ми розглянемо загальну схему роботи чат-бота. Чат-бот складається з чотирьох послідовних етапів або модулів, як показано на рисунку 2, вхідні дані приймаються та обробляються у відповідному форматі. Далі виділяються іменовані сутності та виявляються наміри. Вони використовуються для генерації можливих відповідей, генеруються або

вибираються декілька відповідей, і, нарешті, найбільш підходяща відповідь представляється користувачеві [14].

Різні завдання, включені в різні етапи, включають в себе наступні: розбиття на частини, визначення меж речень, синтаксичний розбір речень, тегування частин мови - ось деякі з основних алгоритмів обробки вхідних даних, які завжди мають місце. При використанні бібліотек ці завдання маскуються у функціях. Для розуміння вхідних даних використовуються такі функції, як вилучення іменованих сутностей, виявлення намірів, виявлення неоднозначності, виявлення контексту, аналіз настроїв, класифікація запитів, виявлення концептів і синонімів тощо [15].

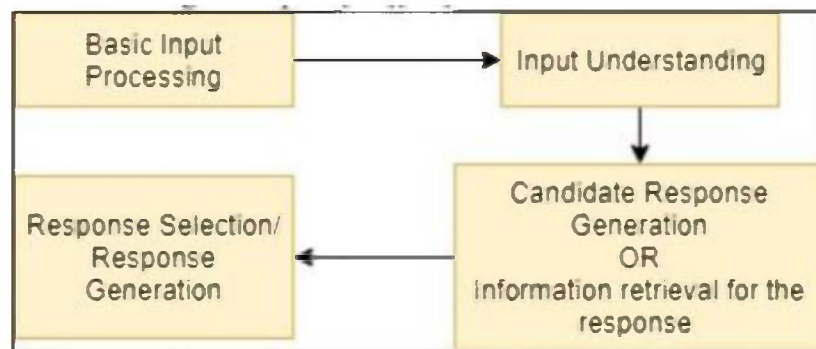


Рис. 2.1 - Пайплайн чат-бота

Після того, як запит або вхідні дані структуровані в потрібну форму, генерується відповідь. Вона може бути згенерована за допомогою попередньо визначеного формату або за допомогою машинного навчання. Якщо відповідь отримується з текстового джерела, можна також використовувати методи узагальнення та спрощення. Після того, як є набір відповідей кандидатів, відповіді можна перевірити на релевантність і повернути відповідну відповідь.

Чат-боти бувають двох типів залежно від того, яку дію вони виконують при формуванні відповіді:

1. Моделі на основі пошуку: ті, що базуються на правилах або отримують відповіді із заздалегідь визначеного набору відповідей. Моделі на основі пошуку можуть бути використані, коли дані

обмежені, а область діалогів звужена до кількох сценаріїв діалогів. Отже, якщо можна уявити собі всі можливі діалоги з ботом для певного варіанту використання, модель, заснована на пошуку, працює добре. Крім того, коли бот не повинен демонструвати інтелект у всіх сценаріях, але може дозволити собі заперечувати, що знає відповідь, такі моделі, засновані на пошуку, працюють добре. Системи бронювання, системи поширених запитань і відповідей або будь-які інші системи, які отримують інформацію, можуть добре працювати навіть з ботом на основі пошуку.

2. Генеративні моделі: Ці системи, які мають на меті підтримувати ієрархічну структуру базових значень мови, можна використовувати, коли є велика кількість даних і система може бути навчена на цих даних. Вони часто використовують алгоритми NLP і NLU для обробки вхідних даних і генерації речень. Штучні нейронні мережі, LSTM, SVM, моделі послідовностей, генеративні алгоритми тощо - ось деякі алгоритми, які використовуються в цій моделі. Ми ще не маємо чисто генеративних моделей, навіть найдосконаліші системи, такі як Alexa, Siri і Cortana, базуються на напівправилах.

2.2 Архітектура чат-боту

Усі інтелектуальні чат-системи мають кілька основних компонентів, всі вони однаково важливі, і разом вони утворюють надійну систему для створення успішного чат-бота. На рис. 5 зображено архітектурний вигляд обчислювально інтелектуального чат-бота. Кінцеві користувачі взаємодіють з презентаційним шаром, який інкапсулює різні компоненти користувацького інтерфейсу системи. Бекенд обміну повідомленнями відповідає за отримання та доставку повідомлення до презентаційного рівня. Рівень машинного навчання є основою інтелектуального чат-бота, який створює відповіді за допомогою бази знань. Остаточна відповідь доставляється користувачеві за допомогою процесу генерації природної мови.

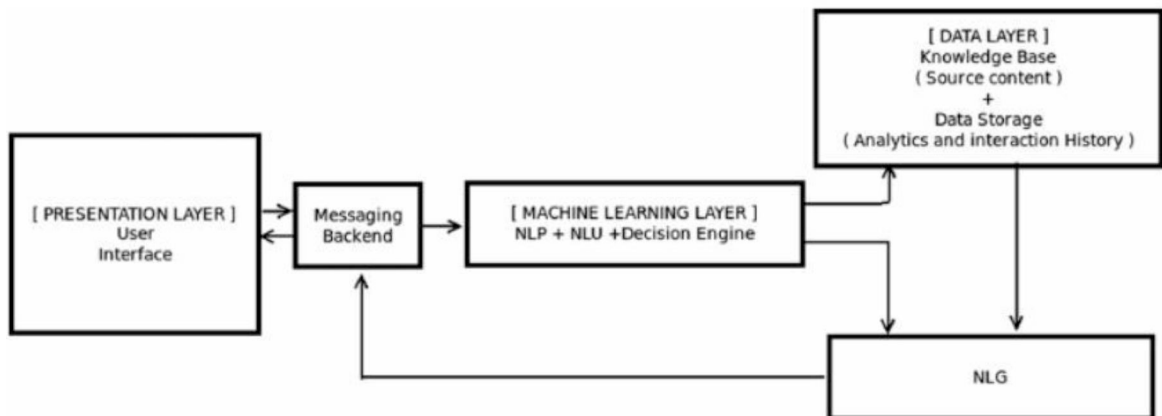


Рис. 2.2 - Загальна архітектура інтелектуального чат-бота.

Усі компоненти користувацького інтерфейсу представляють шар презентації. Цей рівень безперешкодно надає користувацький інтерфейс кінцевим користувачам незалежно від пристрою та платформи. Сприяння багатоканальній та багатоплатформенній підтримці є основним завданням рівня представлення.

Рівень машинного навчання складається з трьох важливих компонентів: Обробка природної мови (NLP), розуміння природної мови (NLU) та механізм прийняття рішень. Поєднання цих трьох компонентів додає чат-боту штучного інтелекту (ШІ) для інтелектуальної обробки повідомлень.

Обробка природної мови: більшість систем чат-ботів обробляють вхідні дані користувача перед тим, як перенаправити їх до компонента розуміння природної мови. Завдання попередньої обробки виконуються в різних сегментах. Бергер узагальнив завдання попередньої обробки діалогових систем у таких фазах: виявлення речень, розпізнавання, токенизація, лематизація, POS-тегування, розбір залежностей, розпізнавання іменованих сутностей, семантичне маркування ролей. Існують такі терміни, як наміри, дії та сутності, які в основному готуються з вхідних даних користувача за допомогою методів обробки природної мови [16].

Розуміння природної мови відображає текст вхідного повідомлення у семантичні слоти. Слоти заздалегідь визначені відповідно до різних сценаріїв. Зазвичай існує два типи представлень. Один з них - це категорія рівня висловлювання, наприклад, категорія наміру користувача. Інший - це вилучення інформації на рівні слів, наприклад, розпізнавання іменованих сутностей і заповнення слотів. Виявлення намірів виконується для виявлення намірів користувача. Він класифікує висловлювання за одним із заздалегідь визначених намірів. Для виявлення намірів послідовно застосовуються методи глибокого навчання.

Движок прийняття рішень: Завдання системи прийняття рішень полягає в тому, щоб вирішити, що робити на основі всієї інформації в базі знань. Він вивчає специфічні для предметної області знання з вихідних даних і застосовує їх для прийняття рішення. Тут рішення відноситься до прогнозування реакції. Рішення - це не одномоментна дія. Інтелектуальний бот планує кілька запитань перед фазою прийняття рішення, щоб зібрати необхідну інформацію. Багато останніх робіт показали, що нейронні мережі можуть бути успішно використані в ряді завдань з обробки природної мови. До них відносяться моделювання мови, розпізнавання перефраз і виділення вставлених слів. У галузі статистичного машинного перекладу (СМП) глибокі нейронні мережі почали демонструвати багатообіцяючі результати. У статті узагальнено успішне використання нейронних мереж прямого поширення в рамках системи СМП на основі фраз. Запропонована нейромережева архітектура, RNN Encoder Decoder, складається з двох рекурентних нейронних мереж (RNN), які діють як кодер і декодер.

Рекурентна нейронна мережа: Людські думки мають наполегливість. Відсутність стійкості є одним з основних недоліків традиційних нейронних мереж. Рекурентні нейронні мережі - це різновид нейронних мереж, який дозволяє моделювати ці далекі залежності. Це мережі, в яких є цикл, що дозволяє зберігати інформацію.

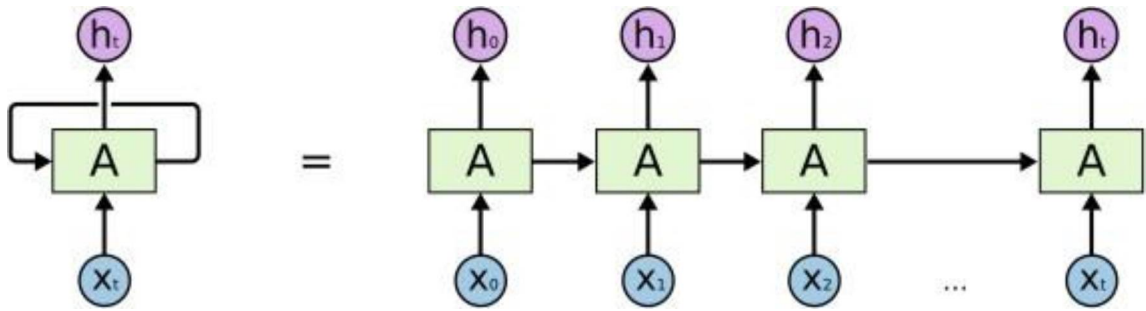


Рис. 2.3 - Рекурентна нейронна мережа (RNN)

На наведеній вище схемі частина нейронної мережі, A , розглядає деякий вхід X_t і виводить значення h_t . Цикл дозволяє передавати інформацію від одного кроку мережі до іншого. Рекурентну нейронну мережу можна уявити як кілька копій однієї і тієї ж мережі, кожна з яких передає повідомлення наступнику. У RNN є проблема зникаючого градієнта і градієнта, що вибухає. Це призводить до нового дизайну архітектури нейронної мережі, яка отримала назву нейронна мережа з довгою короткочасною пам'яттю (long short-term memory neural network architecture). Це дуже вдалий дизайн, який набуває популярності в широкому спектрі завдань послідовної обробки.

Мережа довготривалої короткочасної пам'яті: Мережі з довгою короткочасною пам'яттю (LSTM) - це особливий вид рекурентних нейронних мереж. LSTM призначені для вивчення довгострокових залежностей. У стандартних RNN цей повторюваний модуль має дуже просту структуру, наприклад, один тангенціальний шар. LSTM також мають подібну структуру, але модуль, що повторюється, має іншу структуру. Замість одного шару нейронної мережі, тут є чотири, які взаємодіють дуже особливим чином, як показано на рис. 7.

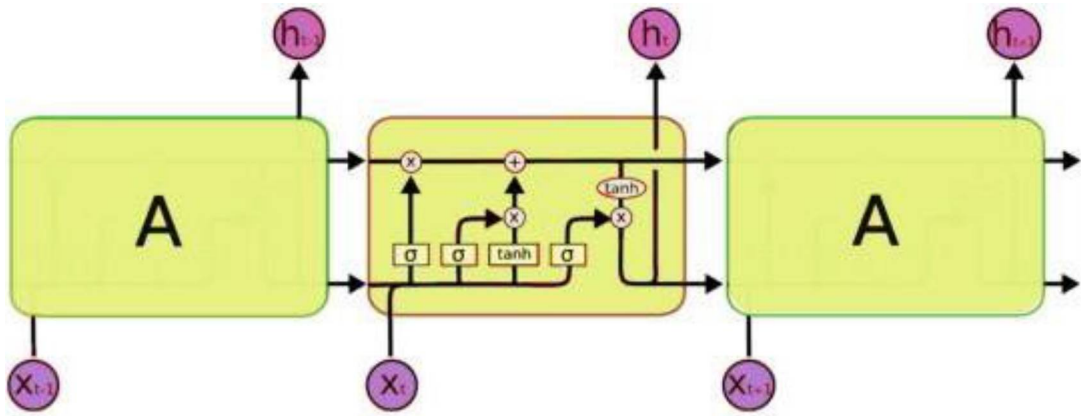


Рис. 2.4 - Мережі з довгою короточасною пам'яттю (LSTM)

Нейронний кодер і декодер: Модель Sequence to Sequence (seq2seq) складається з двох RNN - кодера та декодера. Кодер зчитує вхідну послідовність слово за словом і видає контекст, який містить семантичне резюме вхідної послідовності. На основі цього контексту декодер генерує вихідну послідовність, по одному слову за раз, дивлячись на контекст і попереднє слово на кожному часовому кроці. У цій моделі використовується вбудовування слів. Для побудови вбудовування слів готується список "vocabulary", який містить всі слова моделі. На вхід моделі подаються тензори, що містять ідентифікатори слів у послідовності. Кодер відображає вихідну послідовність змінної довжини у вектор фіксованої довжини, а декодер відображає векторне представлення назад у цільову послідовність змінної довжини. Обидві мережі навчаються спільно, щоб максимізувати умовну ймовірність цільової послідовності за заданою вихідною послідовністю.

Нейронний кодер і декодер: модель Sequence to Sequence (seq2seq) складається з двох RNN - кодера та декодера. Кодер зчитує вхідну послідовність слово за словом і видає контекст, який містить семантичне резюме вхідної послідовності. На основі цього контексту декодер генерує вихідну послідовність, по одному слову за раз, дивлячись на контекст і попереднє слово на кожному часовому кроці. У цій моделі використовується вбудовування слів. Для побудови вбудовування слів готується список

"vocabulary", який містить всі слова моделі. На вхід моделі подаються тензори, що містять ідентифікатори слів у послідовності. Кодер відображає вихідну послідовність змінної довжини у вектор фіксованої довжини, а декодер відображає векторне представлення назад у цільову послідовність змінної довжини. Обидві мережі навчаються спільно, щоб максимізувати умовну ймовірність цільової послідовності за заданою вихідною послідовністю.

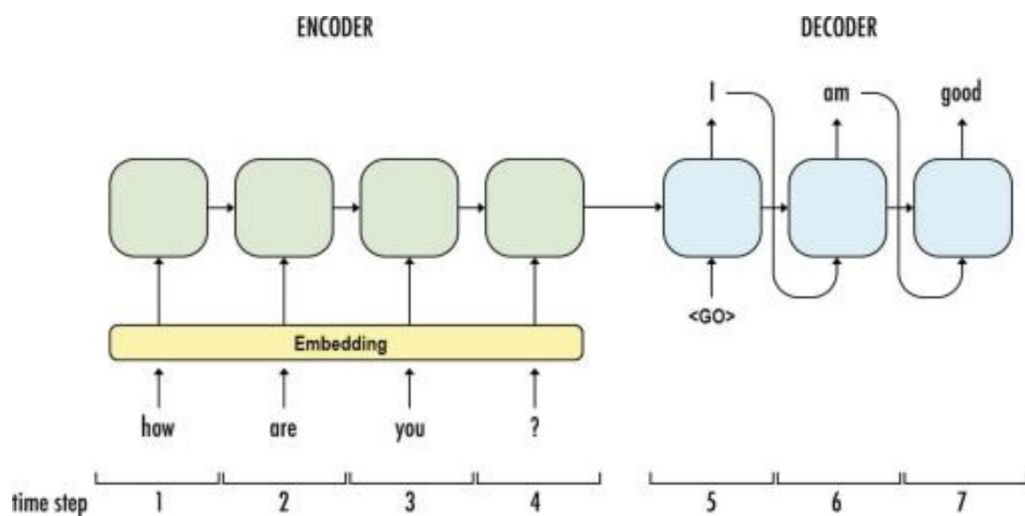


Рис. 2.5 - Модель Seq2Seq: нейронний кодер і декодер

Рівень даних складається з бази знань та історії взаємодії з користувачем. Аналітика даних також є важливою частиною цього рівня.

Основною частиною інтелектуального чат-бота є його база знань. Система чат-ботів навчається під наглядом, тому їй потрібна велика кількість даних. База знань представляє собою специфічний для певної галузі вміст, який використовується для навчання. Якість і кількість бази знань прямо пропорційна інтелекту чат-бота. Організована база знань сприяє швидшому навчанню чат-бота [16].

Компонент генерації природної мови перетворює абстрактне твердження на поверхневі висловлювання природної мови. Для генерації правильних відповідей використовується структура кодера-декодера на основі

LSTM, яка враховує інформацію про питання, значення семантичних слотів і тип діалогового акту. Аналогічним чином представлено генератор природної мови на основі підходу "від послідовності до послідовності", який можна навчити генерувати рядки природної мови, а також дерева глибоких синтаксичних залежностей на основі вхідних діалогових актів. Його було розширено для попередніх висловлювань користувача та відповідей.

2.3 Платформи та фреймворки для розробки чат-ботів

Це найпростіша платформа для створення чат-бота. Вона не потребує жодних знань з програмування. Навіть не знаючи NLP та машинного навчання, можна створити чат-бота. Цей тип платформ здебільшого орієнтований на прості завдання. Chatfuel, ManyChat, Motion.ai та MobileMonkey є широко використовуваними платформами для створення чат-ботів. Ми бачимо, що всі ці платформи мають як певні переваги, так і обмеження.

Таблиця 2.1

Переваги та недоліки використання платформ для розробки чат-ботів

Переваги	Недоліки
1. Менше часу на розробку чат-бота.	1. Функції у форматі prebuild. Неможливо побудувати щось більше, ніж просте, не підходить для складних розмовних ботів.
2. Візуальне середовище розробки доступне для розробників.	2. Дуже мало специфічних для NLP та ML компонентів.
3. Ідеально підходить для простого бота.	

Хмарний движок Facebook Bot Engine Wit.ai дозволяє розробляти бот-додатки для платформи Facebook Messenger. Основним джерелом для побудови поведінки чат-бота за допомогою Wit.ai є історії. Історії завжди

повні прикладів різних сценаріїв з реального життя, і ці приклади забезпечують платформу для навчання Wit.ai. Основна ідея полягає в тому, що коли користувач пише подібні приклади, Wit.ai обробляє їх, виокремлює сутності та застосовує логіку відповідно до вже вивчених прикладів на основі історій [17].

Історія складається з багатьох намірів користувача. Ці наміри можна представити у вигляді графа, щоб відобразити хід розмови. Wit.ai дозволяє розробникам визначати та використовувати попередньо визначені сутності. Wit.ai полегшує інтеграцію веб-хуків для розробників.

Таблиця 2.2

Переваги та недоліки Facebook Bot Engine

Переваги	Недоліки
1. Оскільки він базується на концепції історії, він є більш потужним, ніж ізольований сегмент розмови.	1. Незважаючи на те, що історії є потужною концепцією, бувають випадки, коли важко контролювати потік розмови, і бот схильний неправильно розуміти запити користувачів.
2. Wit.ai дозволяє керувати потоком розмови за допомогою гілок, а також умов на дії.	

Dialogflow надає платформу, яка дозволяє розробникам проектувати та реалізовувати інтерфейси розмов, які можна вбудовувати у зовнішні додатки, такі як чат-боти. Наміри та контексти - це ключові поняття для моделювання поведінки чат-бота за допомогою Dialogflow. Наміри створюють зв'язок між тим, що говорить користувач, і тим, яку дію повинен виконати чат-бот. Контексти - це рядкові значення, корисні для диференціації запитів, які можуть мати різне значення залежно від контексту. Він має інтеграцію з веб-

хуками, що дозволяє розробникам налаштовувати поведінку чат-бота. Dialogflow підтримує різні платформи, включаючи Skype, Telegram, Slack, Cortana, Alexa та Facebook Messenger [17].

Таблиця 2.3

Переваги та недоліки Dialogflow

Переваги	Недоліки
1. Dialogflow пропонує потужний спосіб моделювання великих і складних потоків розмов за допомогою намірів і контекстів.	1. Одночасна робота з контекстом і наміром є дуже складним завданням.
2. Доступний розділ "Навчання" для навчання чат-бота на прикладах.	

Microsoft Bot Framework - один з найкращих фреймворків для чат-ботів з функцією машинного навчання. Інтелектуальний сервіс Microsoft Language Understanding Intelligent Service дуже схожий на Dialogflow. Вхідні дані користувача семантично перевіряються за допомогою попередніх знань, щоб викликати відповідні наміри. Він також має концепцію сутностей для вивчення розмови. Він надає засоби для створення намірів, сутностей та агентів, але не надає візуального представлення потоків розмови, як це робить Wit.ai. Інтелект та асистент Мікрософт-бот фреймворк - один з найкращих фреймворків для чат-ботів з машинним навчанням. Інтелектуальний сервіс Microsoft Language Understanding Intelligent Service дуже схожий на Dialogflow. Вхідні дані користувача семантично перевіряються за допомогою попередніх знань, щоб викликати відповідні наміри. Він також має концепцію сутностей для вивчення розмови. Він надає засоби для створення намірів, сутностей та агентів, але не надає візуального представлення потоків розмови, як це робить Wit.ai. Інтелектуальні та асистентські сервіси, засновані на цьому фреймворку, набагато кращі за всі інші сучасні рішення.

Таблиця 2.4

Переваги та недоліки Microsoft Bot Framework

Переваги	Недоліки
1. Розумніший і може вивчати альтернативні фрази, які можуть викликати наміри та альтернативні сутності.	1. Не дозволяє розробникам керувати параметрами контексту.
2. Інтелектуальний сервіс розуміння мови також підтримує складені сутності. Жоден з інших фреймворків ШІ не підтримує складені сутності.	2. Не містить багатого набору доменів для створення чат-ботів на кшталт Dialogflow.

2.4 Вибір технології для телеграм-бота генерації рецептів

Вибір правильної технології для розробки Telegram-бота для генерації рецептів має вирішальне значення для успіху та ефективності проекту. Розглянемо деякі варіанти докладніше:

- Python з Телеботом: Python відомий не лише своєю простотою та зрозумілістю, але й великою екосистемою бібліотек та фреймворків, що робить його популярним вибором для розробки ботів. Telebot, створений на основі Flask, - це потужна бібліотека, яка спрощує процес створення Telegram-ботів. Вона пропонує такі функції, як обробники повідомлень, обробники команд, підтримка вбудованого режиму та інтеграція з веб-хуками. Крім того, можна інтегрувати бібліотеки обробки природної мови Python, такі як NLTK або spaCy, щоб покращити здатність бота розуміти та генерувати рецепти на основі запитів користувачів. Крім того, гнучкість Python дозволяє безперешкодно інтегруватися із

зовнішніми API для отримання даних про рецепти або інгредієнти зі сторонніх джерел, що розширює функціональність бота.

- JavaScript з Telegraf: JS, особливо з Node.js, набув значної популярності у спільноті розробників завдяки своїй архітектурі, керованій подіями, та неблокуючій моделі вводу/виводу, що робить його добре придатним для створення додатків у реальному часі, таких як Telegram-боти. Telegraf, фреймворк Telegram-ботів для Node.js, спрощує розробку ботів, надаючи підтримку проміжного програмного забезпечення, обробку команд, управління контекстом тощо. Екосистема JavaScript пропонує широкий вибір бібліотек NLP, таких як Natural або Compromise, що дозволяє розробникам впроваджувати розширені можливості обробки природної мови для генерації рецептів. Крім того, підтримка JS асинхронного програмування дозволяє ефективно обробляти численні запити та паралельні операції, підвищуючи швидкість реакції бота.
- Використовуйте фреймворк TBot: Go, відомий своєю підтримкою паралельності, ефективністю та швидким часом компіляції, є чудовим вибором для створення високопродуктивних Telegram-ботів. TBot, легкий фреймворк для Go, пропонує мінімалістичний, але потужний набір інструментів для взаємодії з Telegram Bot API. За допомогою TBot розробники можуть легко обробляти вхідні повідомлення, обробляти команди, керувати станом та інтегруватися із зовнішніми сервісами. Стандартна бібліотека Go забезпечує надійну підтримку роботи з HTTP, що спрощує реалізацію комунікації з ботами на основі веб-хуків. Крім того, простота та статична типізація Go полегшує розробку надійного та зручного для підтримки коду бота, забезпечуючи безперебійну роботу навіть під високими навантаженнями.

- Використання платформ для розробки ботів: платформи для розробки ботів, такі як Chatfuel або ManyChat, призначені для користувачів, які віддають перевагу підходу до створення ботів без коду або з невеликою кількістю коду. Ці платформи пропонують інтуїтивно зрозумілі візуальні інтерфейси, конструктори з функцією перетягування та готові компоненти для спрощення процесу розробки ботів. Хоча вони підходять для простих ботів або прототипів, їм може не вистачати гнучкості та можливостей кастомізації, властивих індивідуальним рішенням. Крім того, інтеграція розширених функцій, таких як обробка природної мови або зовнішня взаємодія з API, може бути обмежена або недоступна через обмеження платформи.

Отже, вибір технології для Telegram-бота для генерації рецептів залежить від різних факторів, таких як досвід розробника, вимоги проекту, масштабованість і бажані функції. Ретельно оцінивши ці фактори, розробники можуть обрати найбільш підходящий стек технологій для створення надійного та зручного бота для створення рецептів для користувачів Telegram.

Висновки за розділом 2

У даному розділі ми провели аналіз різновидів та архітектури чат-ботів, розглянули платформи та фреймворки для їх розробки, а також розглянули вибір технології для створення телеграм-бота з генерацією рецептів. Цей аналіз дозволив нам краще зрозуміти можливість і потенціал кожного інструменту в контексті нашого проекту. На основі цього висновку можна зробити, що правильний вибір технології є критичним для успішного впровадження та розвитку чат-бота. Такий підхід допомагає забезпечити ефективне функціонування, оптимізацію витрат часу та ресурсів, а також максимізацію користувацького досвіду.

РОЗДІЛ 3

РОЗРОБКА МОДЕЛІ ТА КОМП'ЮТЕРНА РЕАЛІЗАЦІЯ ТЕЛЕГРАМ-БОТУ ДЛЯ ГЕНЕРАЦІЇ РЕЦЕПТІВ

2.1 Побудова структурної схеми

У сучасному світі із заповненим розкладом та зростаючими очікуваннями щодо продуктивності, багато людей шукають швидкі та прості способи готувати смачну та здорову їжу. Відповідаючи на це попит, технології розвиваються, щоб надати інноваційні рішення, які полегшують цей процес. Одним із таких інноваційних рішень є телеграм-бот для генерації рецептів з використанням штучного інтелекту.

Виходячи з цього побудуємо структурну схему телеграм-боту генерації рецептів.

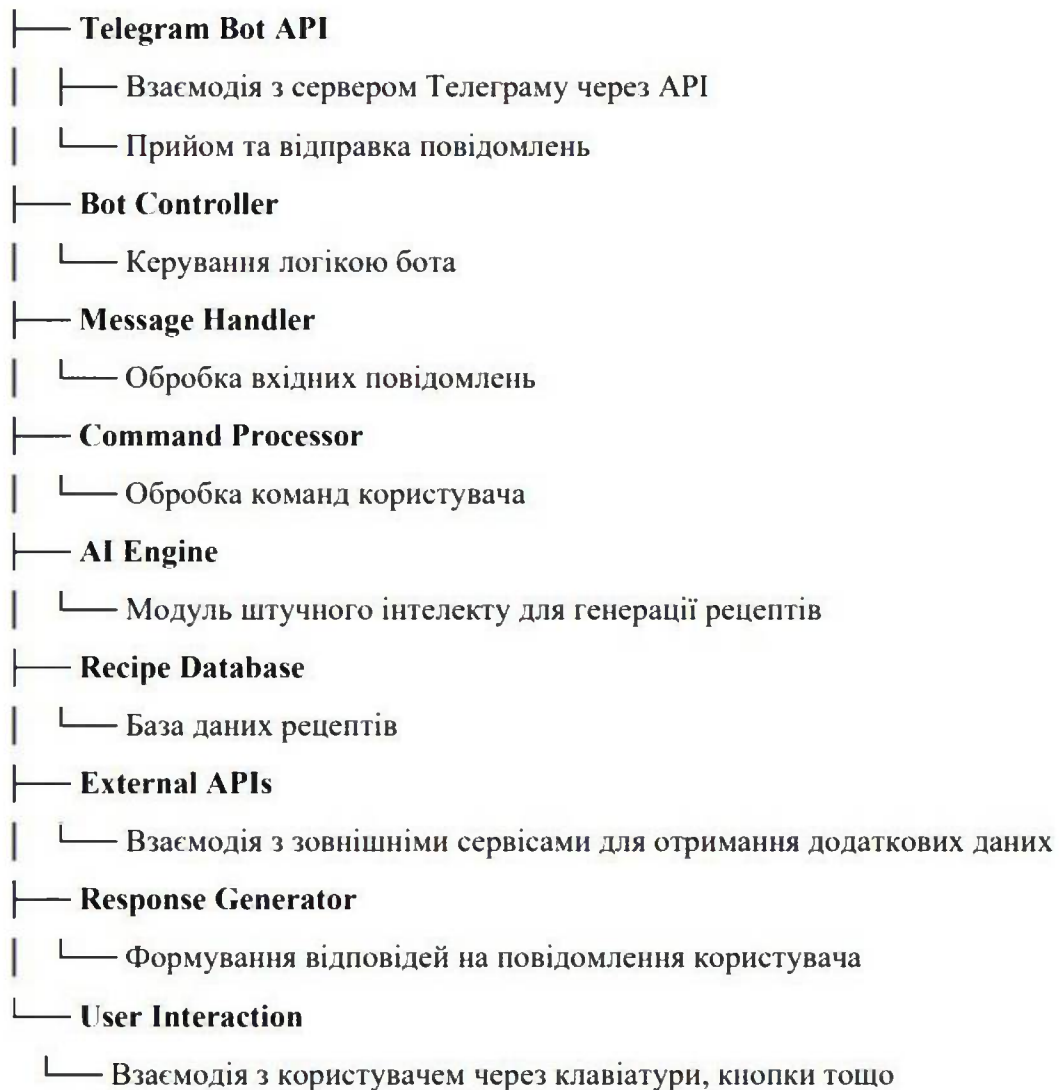


Рис. 3.1 – структурна схема Telegram-Bot

Ця структурна схема відображає ключові компоненти, які входять до складу такого телеграм-бота та їх взаємозв'язки. Кожен з цих компонентів має визначену роль у функціонуванні бота та сприяє забезпеченню його ефективної роботи.

Опис компонентів:

- Telegram Bot API: цей компонент забезпечує зв'язок бота з месенджером Telegram через використання відповідного API. Він відповідає за отримання та відправку повідомлень між користувачами та ботом.
- Bot Controller: контролер бота відповідає за керування логікою роботи бота в цілому. Він приймає вхідні дані від Telegram Bot API та визначає, які дії повинен виконати бот відповідно до отриманих команд та повідомлень.
- Message Handler: цей компонент відповідає за обробку вхідних повідомлень від користувачів. Він визначає тип повідомлення та ініціює відповідну обробку для подальшого аналізу.
- Command Processor: компонент обробки команд відповідає за аналіз команд, що надсилаються користувачами, та ініціює відповідні дії, такі як генерація рецептів або взаємодія з базою даних.
- AI Engine: серце системи - модуль штучного інтелекту, який використовується для генерації рецептів. Він аналізує інформацію з бази даних та зовнішніх джерел, щоб створити нові та цікаві рецепти.
- Recipe Database: база даних рецептів містить інформацію про доступні рецепти, яка використовується AI Engine для генерації нових рецептів.
- External APIs: компонент взаємодії з зовнішніми сервісами дозволяє отримувати додаткові дані, які можуть бути корисними для генерації рецептів, такі як інформація про інгредієнти або техніки приготування.
- Response Generator: цей компонент формує відповіді на повідомлення користувачів на основі отриманих даних та результатів роботи AI Engine.

- User Interaction: Компонент взаємодії з користувачем дозволяє забезпечити зручний та інтуїтивно зрозумілий інтерфейс для користувачів бота через використання клавіатур, кнопок та інших інтерактивних елементів.

Додатково до зазначеної вище структурної схеми, важливо розглянути, як саме відбувається взаємодія між цими компонентами та які процеси вони спричиняють.

Опис взаємодії компонентів:

- Отримання повідомлень користувачів: Telegram Bot API отримує повідомлення від користувачів та передає їх до Bot Controller.
- Аналіз повідомлень: Bot Controller передає повідомлення до Message Handler та Command Processor для подальшого аналізу.
- Обробка команд: Command Processor аналізує команди користувачів та ініціює відповідні дії, такі як запит рецепту.
- Запит рецепту: після отримання запиту на рецепт, Command Processor взаємодіє з AI Engine для генерації нового рецепту.
- Генерація рецепту: AI Engine аналізує дані з Recipe Database та інших джерел для створення нового рецепту.
- Формування відповіді: після того, як AI Engine згенерував рецепт, Response Generator створює відповідь та відправляє її користувачеві через Telegram Bot API.
- Взаємодія з користувачем: User Interaction включає в себе взаємодію з користувачем через клавіатури, кнопки тощо, щоб надати зручний та інтуїтивно зрозумілий інтерфейс.

Опис процесів, які відбуваються:

- Отримання та аналіз повідомлень: бот отримує повідомлення від користувачів, а потім аналізує їх для визначення подальших дій.

- Генерація рецептів: після отримання запиту на рецепт, AI Engine використовує дані з Recipe Database та інших джерел для створення нового рецепту.
- Формування та надсилання відповіді: після створення рецепту Response Generator формує відповідь та надсилає її користувачеві.

Ця взаємодія та процеси демонструють, як кожен компонент вносить свій внесок у роботу телеграм-бота для генерації рецептів з використанням штучного інтелекту, забезпечуючи користувачам зручний та швидкий доступ до рецептів.

3.2. Переваги та недоліки мови програмування Python для написання чат-ботів

У сучасному технологічному світі, що стрімко розвивається, Python став популярним вибором для розробки телеграм-ботів з використанням штучного інтелекту завдяки своїй простоті, читабельності та великим бібліотекам. Оскільки розробники та компанії продовжують досліджувати потенціал штучного інтелекту, важливо розуміти сильні та слабкі сторони мов програмування, що використовуються в цій галузі. Ми розглянемо переваги та недоліки Python для розробки та порівняємо його з іншими мовами програмування, які зазвичай використовуються в цій галузі. Надаючи всебічний аналіз варіантів використання Python в ШІ, ми прагнемо допомогти вам прийняти обґрунтоване рішення про те, чи є він правильним вибором для вашого проєкту, або ж інші альтернативи можуть краще відповідати вашим потребам. Приєднуйтесь до нас, щоб дослідити світ Python та його роль у захоплюючій сфері розробки ШІ, яка постійно розвивається.

Розглянемо переваги використання мови програмування Python у розробці AI:

1. Зрозумілість і простота: однією з головних причин популярності Python у розробці ШІ є його зрозумілість і простота. Python має

чистий синтаксис, який легко зрозуміти, що робить її чудовим вибором як для початківців, так і для досвідчених розробників. Його елегантний дизайн дозволяє програмістам виражати складні ідеї та алгоритми меншою кількістю рядків коду, що в кінцевому підсумку скорочує час і зусилля, необхідні для розробки.

2. Широкі бібліотеки та фреймворки Python: вони відіграють вирішальну роль у його придатності для розробки ШІ. Такі бібліотеки, як TensorFlow, Keras та PyTorch, спеціально розроблені для задач машинного навчання та глибокого навчання, спрощуючи процес розробки моделей ШІ. Ці бібліотеки не лише надають готові функції та інструменти для виконання типових завдань, але й сприяють повторному використанню коду, дозволяючи розробникам спиратися на існуючі напрацювання та зосереджуватися на унікальних аспектах своїх проєктів.
3. Активна підтримка спільноти: Python може похвалитися великою та активною спільнотою розробників, які постійно роблять свій внесок у розвиток та вдосконалення мови. Ця активна спільнота гарантує розробникам доступ до безлічі ресурсів, включаючи навчальні посібники, документацію, форуми та приклади коду, що полегшує пошук рішень спільних проблем та навчання на досвіді інших. Крім того, відкритий вихідний код Python дозволяє постійно розвивати та вдосконалювати його функції та можливості, зберігаючи його актуальність у швидкозмінній індустрії ШІ.
4. Інтероперабельність та інтеграція: інтероперабельність Python з іншими мовами програмування та інструментами - ще одна перевага, яка робить її привабливим вибором для розробки ШІ. Її можна легко інтегрувати з такими мовами, як C, C++ та Java, що дозволяє розробникам використовувати сильні сторони різних мов у своїх проєктах. Така гнучкість спрощує процес розробки і

гарантує, що моделі ШІ можна будувати і розгортати на різних платформах і середовищах, включаючи веб-додатки, десктопні додатки і хмарні системи.

5. Підтримка аналізу та візуалізації даних: розробка ШІ часто передбачає роботу з великими обсягами даних для навчання та оцінки моделей. Python пропонує чудову підтримку для аналізу та візуалізації даних завдяки численним бібліотекам, таким як NumPy, Pandas та Matplotlib. Ці бібліотеки дозволяють розробникам ефективно попередньо обробляти, аналізувати та візуалізувати дані, що є важливим для розуміння продуктивності моделей ШІ та визначення областей для вдосконалення.
6. Універсальність у підобластях ШІ: універсальність Python робить її придатною для широкого спектру підобластей ШІ, включаючи обробку природної мови (NLP), комп'ютерний зір, робототехніку та навчання з підкріпленням. Такі бібліотеки, як NLTK та spaCy полегшують виконання завдань NLP, а OpenCV та scikit-image надають інструменти для додатків комп'ютерного зору. Така адаптивність гарантує, що Python залишається популярним вибором для різних проектів зі штучного інтелекту, незалежно від конкретної галузі чи вимог.
7. Можливість швидкого створення прототипів: у розробці ШІ можливість швидко створювати прототипи та ітерації на моделях має вирішальне значення. Простота Python і великі бібліотеки дозволяють розробникам швидко створювати і тестувати нові ідеї, прискорюючи процес розробки і дозволяючи швидко експериментувати. Це особливо цінно у сфері ШІ, де ітеративні вдосконалення моделей можуть призвести до значного покращення продуктивності та можливостей.

Таким чином, Python пропонує численні переваги для розробки ШІ, включаючи зручність читання, великі бібліотеки, активну підтримку

спільноти, інтероперабельність, можливості аналізу даних, універсальність і швидке створення прототипів. Ці переваги роблять її привабливим вибором для розробників і компаній, які прагнуть використати можливості ІІІ, незалежно від того, чи створюють вони прості моделі, чи складні, надсучасні системи.

Також присутні певні недоліки використання мови програмування Python у розробці ІІІ, які буде наведено нижче:

1. Обмеження продуктивності: одним з найпомітніших недоліків Python у розробці ІІІ є його обмежена продуктивність. Як інтерпретована мова, швидкість виконання Python, як правило, нижча, ніж у скомпільованих мов, таких як C++ та Java. Це може бути суттєвим недоліком у проектах ІІІ, які вимагають обробки в реальному часі або пов'язані зі складними алгоритмами та великими масивами даних. Хоча існують обхідні шляхи, такі як використання компіляторів Just-In-Time (JIT) або інтеграція з кодом C/C++ для підвищення продуктивності, ці рішення можуть ускладнити процес розробки.
2. Споживання пам'яті: споживання пам'яті Python є ще однією проблемою при розробці ІІІ, особливо для проектів з обмеженими апаратними ресурсами. Python використовує систему збору сміття для керування пам'яттю, що може призвести до збільшення накладних витрат пам'яті порівняно з мовами з ручним керуванням пам'яттю. Це може бути проблемою для додатків ІІІ, що працюють на вбудованих системах або пристроях з обмеженою пам'яттю, оскільки може вплинути на загальну продуктивність і ефективність моделі.
3. Обмежена підтримка мобільної та веб-розробки: хоча Python є чудовим вибором для серверних додатків і десктопного програмного забезпечення, його підтримка мобільної та веб-розробки є відносно обмеженою. Це може бути недоліком для

проектів III, які потребують інтеграції з мобільними додатками або веб-інтерфейсами. Хоча існують такі інструменти, як Kivu для крос-платформної розробки мобільних додатків і фреймворки, як Django і Flask для веб-розробки, Python не є основним вибором для цих платформ порівняно з такими мовами, як Swift, Kotlin або JavaScript.

4. Відсутність спеціалізованих бібліотек для певних піддоменів III: хоча Python має широку екосистему бібліотек і фреймворків для розробки III, можуть бути випадки, коли для певних підобластей III не вистачає спеціалізованих бібліотек або вони мають обмежену підтримку. У таких випадках розробникам може знадобитися покладатися на альтернативні мови або створювати власні бібліотеки, що може збільшити складність і час розробки проекту. Однак цей недолік зустрічається відносно рідко, оскільки екосистема Python продовжує рости і розвиватися, усуваючи прогалини в підтримці різних піддоменів III.
5. Труднощі в багатопоточності: глобальне блокування інтерпретатора (GIL) у Python є відомим обмеженням, коли мова йде про багатопотоковість, техніку, яка використовується для підвищення продуктивності паралельних програм. GIL гарантує, що лише один потік може виконувати код Python одночасно, що перешкоджає мові використовувати всі переваги багатоядерних процесорів. Хоча багатопроесорна обробка може бути використана як альтернатива, вона може бути не такою ефективною, як справжня багатопоточність, а керування взаємодією між процесами може бути складнішим.
6. Проблеми з безпекою: Популярність і широке використання Python також роблять його мішенню для загроз безпеці. Додатки для III, розроблені з використанням Python, можуть бути вразливими до вразливостей та експлойтів, якщо код не

захищений належним чином. Крім того, динамічна типізація Python і залежність від сторонніх бібліотек можуть створювати ризики для безпеки, якщо розробники не проводять ретельну перевірку бібліотек, які вони використовують. Тому важливо враховувати найкращі практики безпеки та проводити регулярні аудити, щоб забезпечити безпеку додатків ШІ, створених за допомогою Python.

7. Крива навчання для просунутих концепцій ШІ: хоча простота і зрозумілість мови Python роблять її чудовим вибором для початківців, освоєння просунутих концепцій і методів ШІ все одно може потребувати значного часу для навчання. Розробникам-початківцям у сфері ШІ, можливо, доведеться витратити час на вивчення алгоритмів машинного навчання, нейронних мереж та інших складних концепцій, щоб повною мірою використати можливості Python у розробці ШІ.

Python стала провідною мовою програмування для розробки ШІ завдяки своїй простоті, читабельності та великій екосистемі бібліотек і фреймворків. Її універсальність і простота використання роблять її чудовим вибором як для початківців, так і для досвідчених розробників. Однак важливо визнати, що Python має свої недоліки, особливо в плані продуктивності, використання пам'яті та вразливостей безпеки.

Якщо порівнювати Python з іншими мовами програмування, такими як R, Java, C++ та Julia, то кожна з них має свій унікальний набір переваг та недоліків. Хоча деякі мови можуть досягати успіху в певних сферах, таких як статистичний аналіз або критичні до продуктивності додатки, всебічні характеристики та адаптивність Python сприяли його широкій популярності в розробці ШІ.

3.3 Тестування телеграм-боту «AICOOKER»

Розглянемо саму роботу телеграм-боту “AICOOKER”. Для початку нам потрібно написати /start та нам запропоновано дві відповіді, які ми можемо побачити на рис. 3.2.

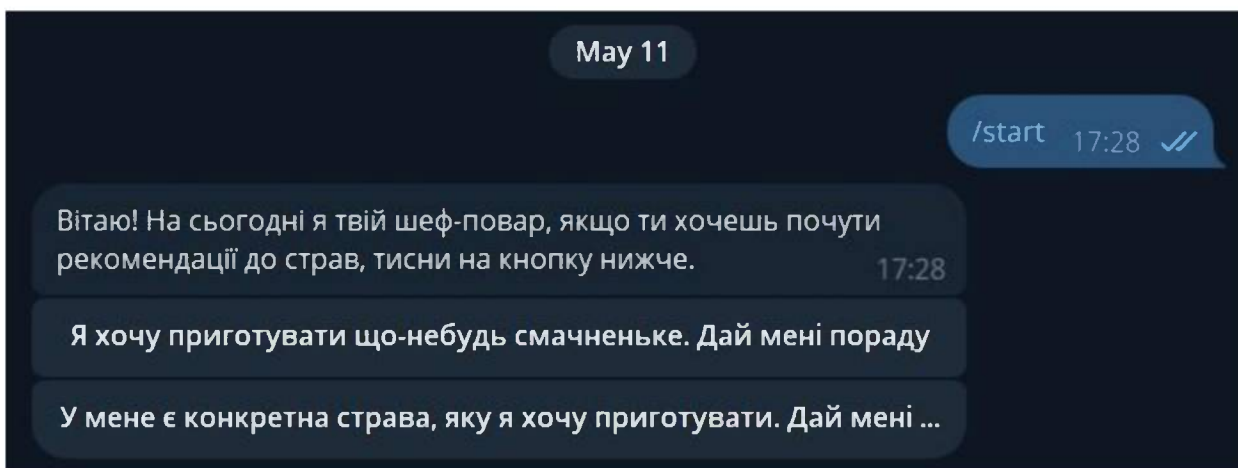


Рис. 3.2 – Початок роботи

На рис. 3.3 є три різноманітні варіанти віповідей. Ми обираємо ідеї для солодких страв. На рис. 3.4 представлено 6 десертів на різний смак.

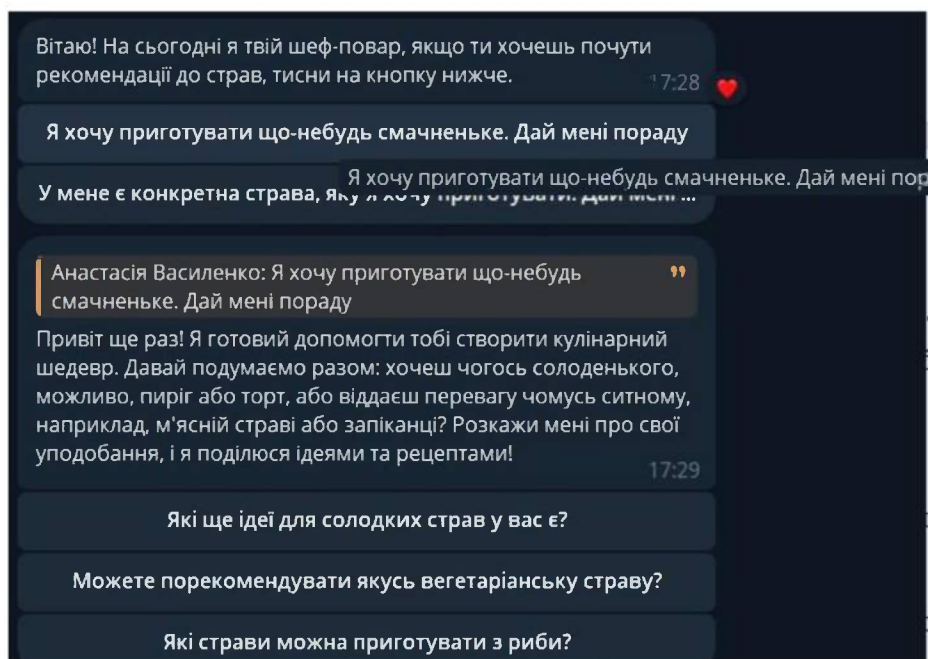


Рис. 3.3 – Порада від телеграм-боту

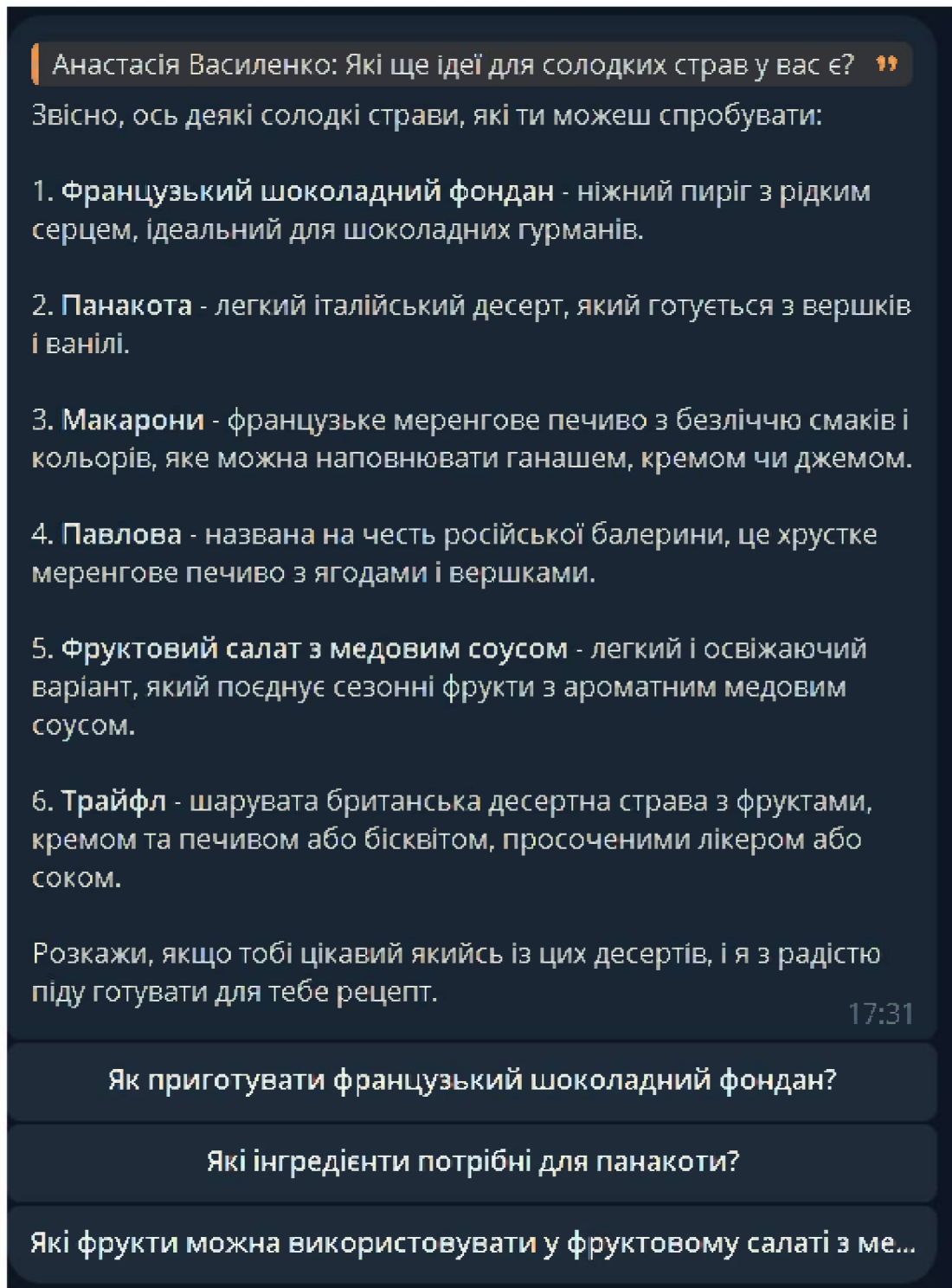


Рис. 3.4 – Рекомендації щодо десертів

Нас зацікавив десерт панакота та які саме інгредієнти потрібні для приготування. Можемо готувати панакоту за даним рецептом, який на рис.3.5.

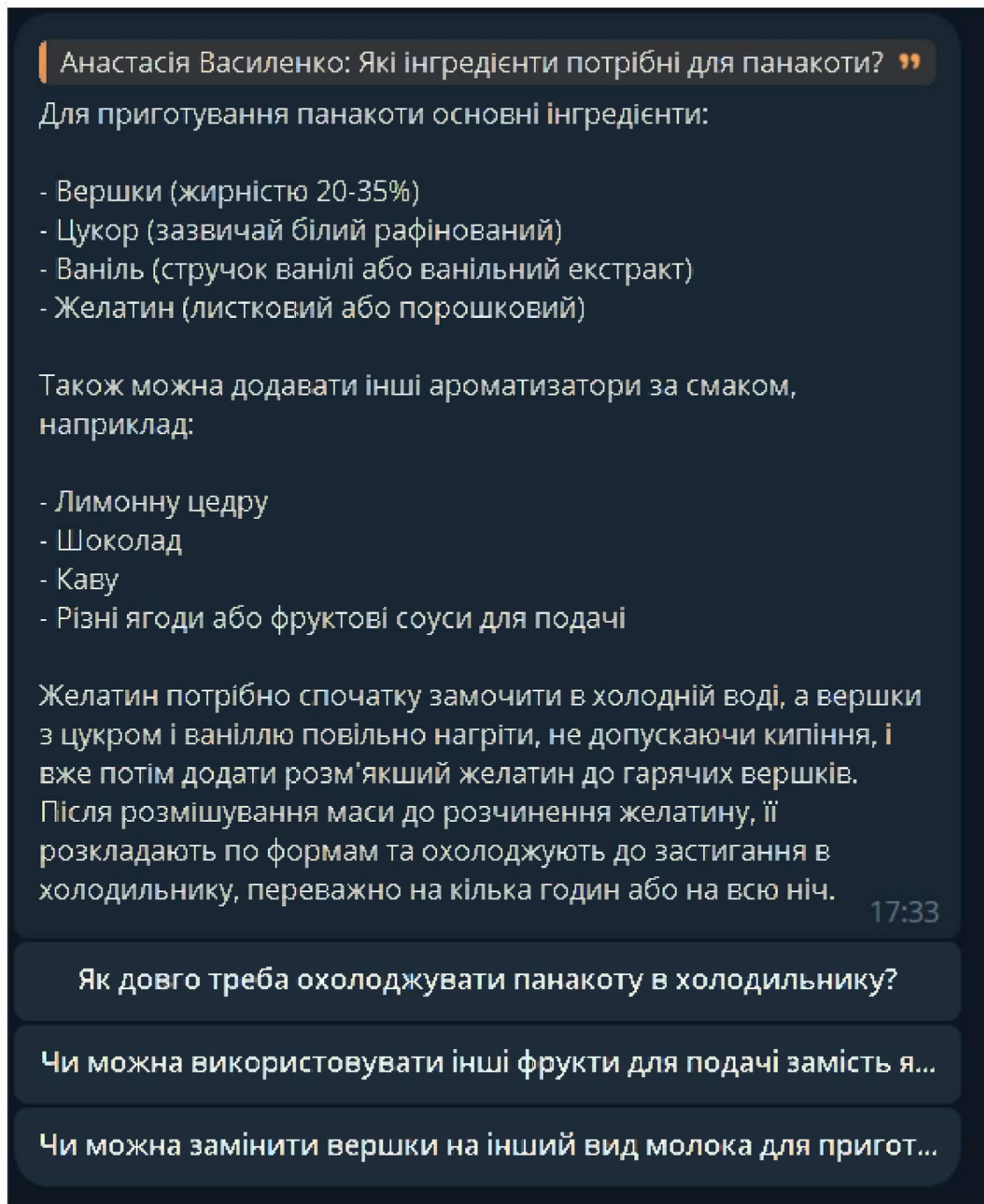


Рис. 3.5 – Конкретний рецепт до страви

Також ми вирішили поцікавитися, якими фруктами для подачі можна прикрашати панакоту та побачили багато варіантів, які може сгенерувати телеграм-бот (рис. 3.6). На рис. 3.7 чат пропонує нам яскраві фотографії з різними ідеями подачі.

Анастасія Василенко: Чи можна використовувати інші фрукти для подачі замість ягід?

”

Так, звісно, можна використовувати інші фрукти для подачі панакоти. Вибір фруктів може залежати від особистих уподобань або сезону. Наприклад:

- **Банани** - наріжте скибочками і укладіть навколо панакоти або зверху.
- **Ківі** - його яскравий зелений колір красиво контрастуватиме із білою панакотою.
- **Персики або абрикоси** - тонко наріжте і викладіть на десерт або подавайте поруч.
- **Ананас** - наріжте невеликими шматочками або кубиками.
- **Манго** - прекрасно підходить для екзотичного відчуття.

Фрукти можна подавати свіжими або ж приготувати фруктовий соус чи компот, який додасть додаткову вологу та смак до панакоти. Крім того, можна експериментувати з різними комбінаціями фруктів для отримання унікального смакового букету.

17:34

Які ще фрукти можна використовувати для подачі панакоти?

Чи можна замінити ягоди на цитрусові для подачі панакоти?

Які фрукти найкраще поєднуються з панакотою?

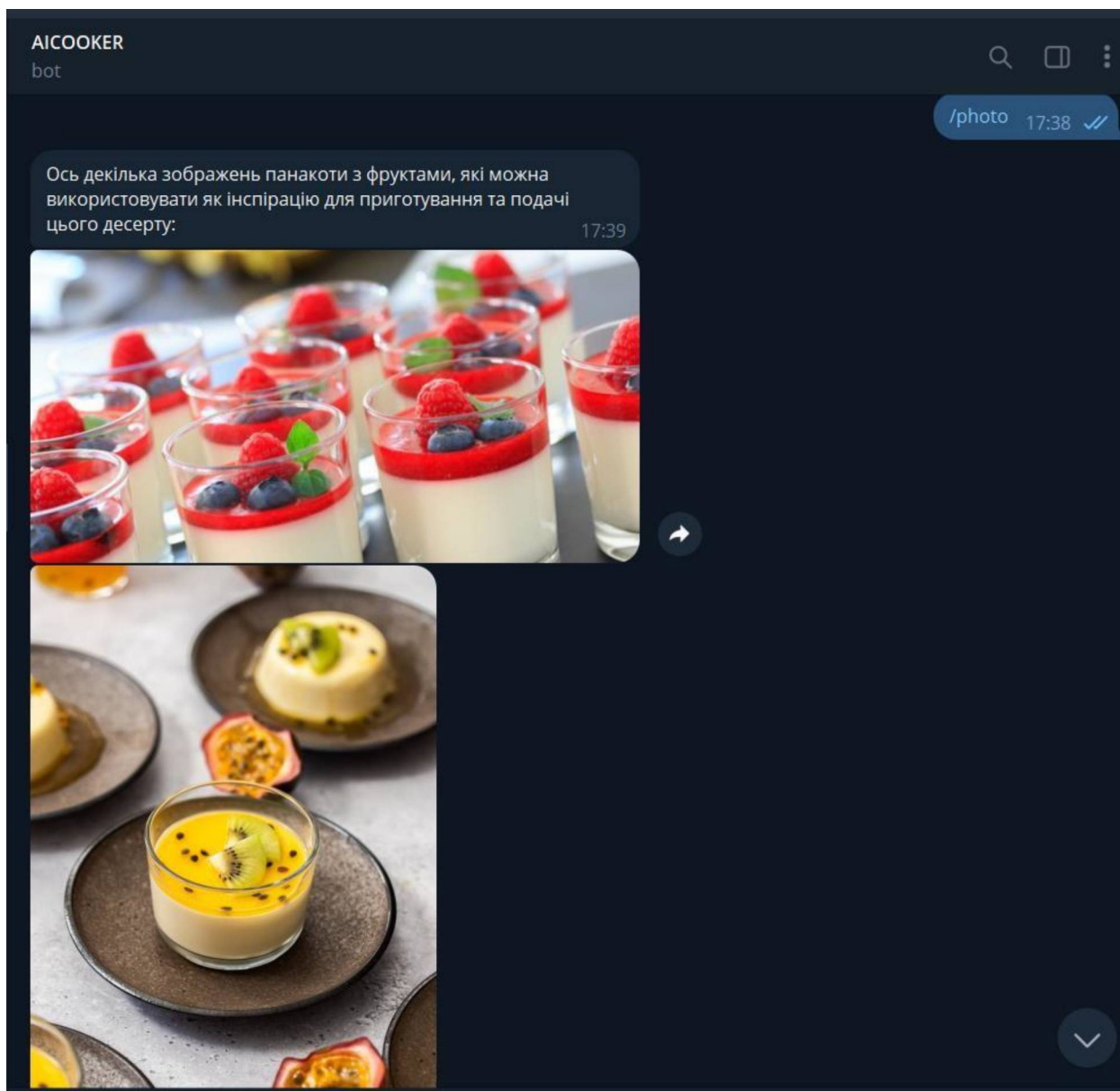


Рис. 3.7 – Фотографії, які надає телеграм-бот на запит про панакоту

На рис. 3.8 ми хочемо приготувати щось конкретне та чат запитує в нас назву страви, яку ми обрали.

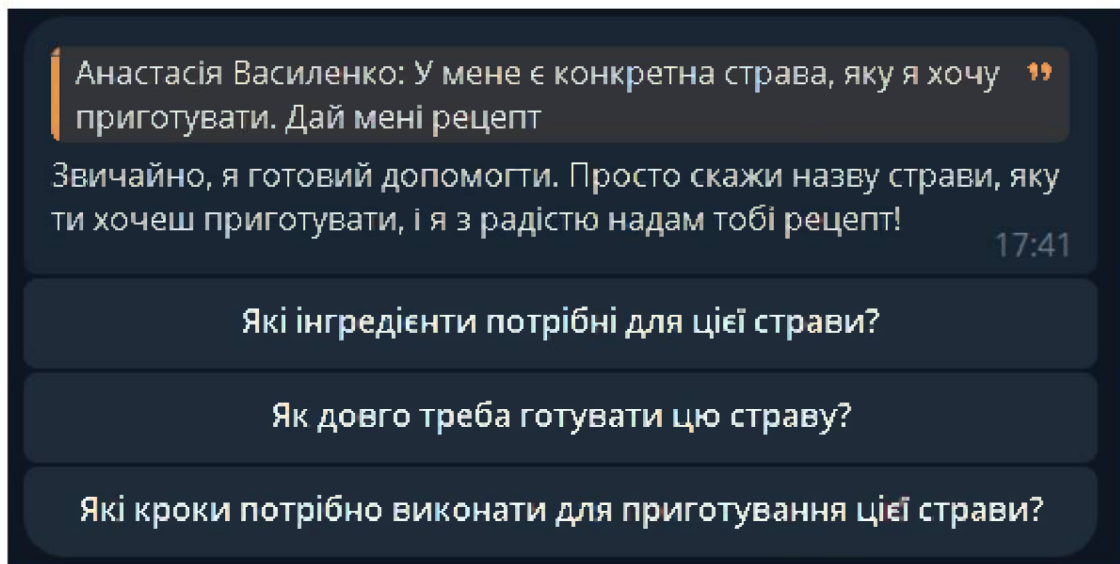


Рис. 3.8 – Вибір дій для приготування

Ми пишемо /borshch та отримуємо список продуктів та покроковий рецепт приготування. Також чат пропонує нам варіант подачі та що ще можливо додат до нашої страви. На рис. 3.10 можемо бачити фото українського борщу, який сгенерував нам чат.

/borshch 17:42 ✓

Ось тобі рецепт традиційного українського борщу:

Інгредієнти:

- Буряк - приблизно 2 середніх шматочки
- Капуста - ¼ качана
- Картопля - 3-4 картоплини середнього розміру
- 1 середня морква
- 1 цибулина
- Бульйон (або вода) - близько 2 літрів
- Томатна паста (або свіжі помідори) - 2-3 столові ложки
- Приправа (лавровий лист, сіль, перець) - за смаком
- Часник - 2-3 зубчики
- Олія для смаження
- Червона квасоля - опціонально

Приготування:

1. Буряк чистимо, нарізаємо соломкою або натираємо на терці, тушуємо на олії разом із томатною пастою до м'якості.
2. Моркву і цибулю нарізаємо і також підсмажуємо на олії, додавши до буряків ближче до кінця тушкування.
3. Картоплю очищаємо, нарізаємо кубиками, варимо у бульйоні або воді.
4. Капусту шинкуємо і кладемо до картоплі, коли вона вже напівготова.
5. Додаємо до борщу підсмажені буряк та моркву з цибулею.
6. Додаємо лавровий лист, сіль, перець за смаком.
7. В кінці варіння додаємо подрібнений часник.
8. Перед подачею можна додати ложку сметани і зелень у кожну тарілку.

Борщ можна подавати з чорним хлібом, зеленню та сметаною. Якщо у тебе є бажання додати квасолю, варто відварити її окремо й додати до борщу вже в кінці готування.

Смачного!

17:42

З якими стравами можна подавати борщ?

Рис. 3.9 – Рецепт та покрокове приготування

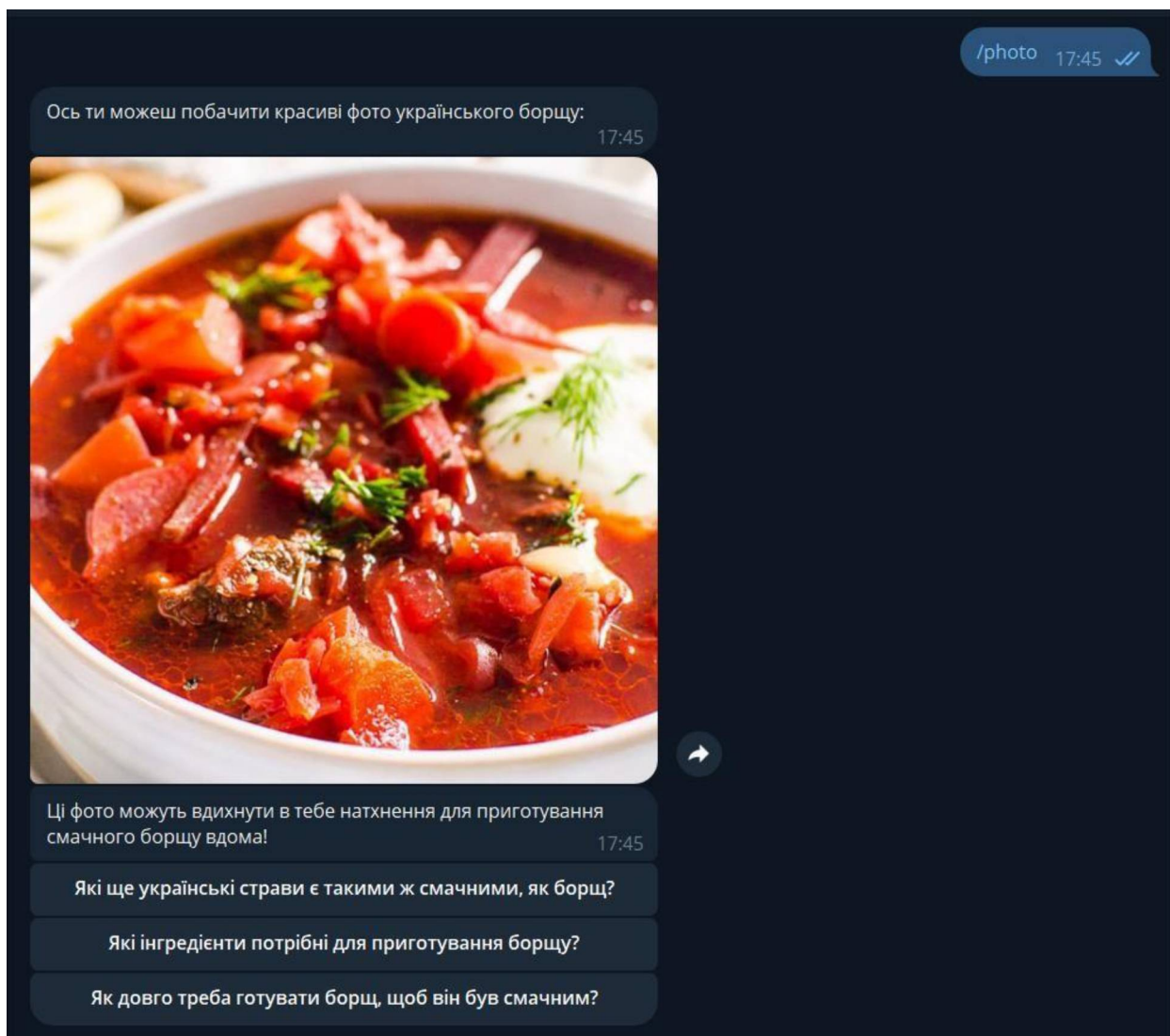


Рис. 3.10 – Фото страви

На рис. 3.11 є перелік українських страв, на кожну з яких, ми можемо отримати рецепт, покрокове приготування та фото.

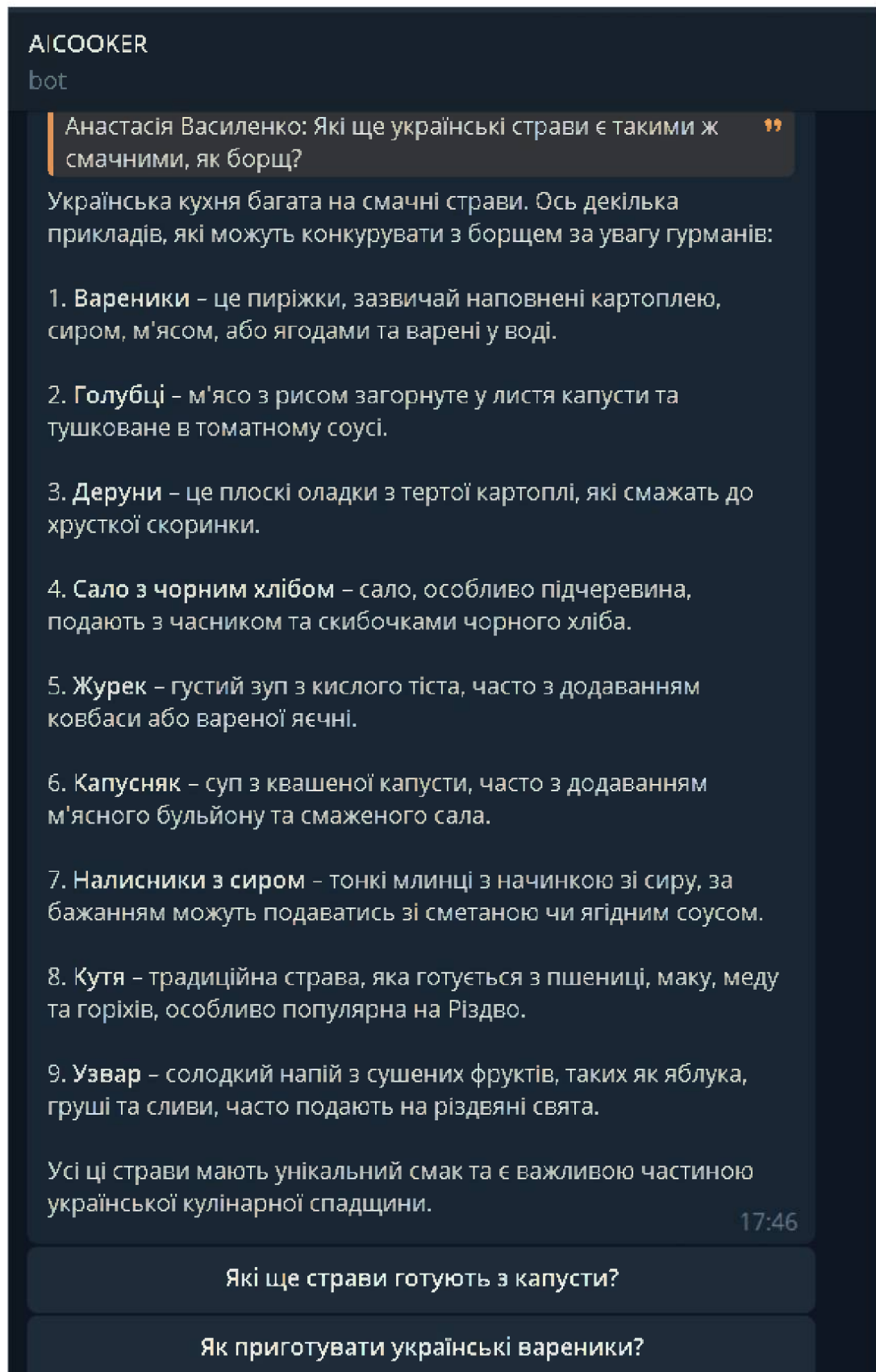


Рис. 3.11 – Різноманіття українських страв

Висновки за розділом 3

У даному розділі було розглянуто розробку телеграм-боту для генерації рецептів. Було створено структурну схему бота, проаналізовано переваги та недоліки використання мови програмування Python, а також проведено тестування системи. Результати тестування показали, що бот стабільний, ефективний та зручний у використанні, з належним рівнем безпеки даних користувачів. Телеграм-бот «AICOOKER» успішно виконує свої функції та забезпечує користувачам корисні кулінарні рекомендації.

ВИСНОВКИ

У даній роботі досліджено та розроблено телеграм-бот для генерації рецептів з використанням штучного інтелекту. Розроблений бот спрямований на надання користувачам швидкого та зручного доступу до рецептів, що відповідають їхнім потребам та вподобанням. Структурна схема телеграм-бота була ретельно розглянута, відображаючи ключові компоненти системи, такі як Telegram Bot API, Bot Controller, AI Engine та інші. Кожен з цих компонентів відіграє важливу роль у функціонуванні бота, забезпечуючи його ефективну роботу та взаємодію з користувачами. Основні процеси роботи бота були ретельно проаналізовані, включаючи отримання та аналіз повідомлень від користувачів, генерацію рецептів за допомогою штучного інтелекту, формування та надсилання відповідей користувачам. Результати дослідження та розробки демонструють, що телеграм-бот для генерації рецептів з використанням штучного інтелекту є потужним інструментом для покращення досвіду користувача у готуванні. Він дозволяє швидко знаходити та отримувати персоналізовані рецепти, що відповідають індивідуальним уподобанням та потребам. Основними перспективами подальшого розвитку є розширення функціональності бота, покращення алгоритмів штучного інтелекту для більш точної генерації рецептів, а також впровадження нових можливостей для взаємодії з користувачами. Таким чином, розроблений телеграм-бот генерації рецептів з використанням штучного інтелекту має великий потенціал для полегшення процесу приготування їжі та задоволення потреб користувачів у здоровому та смачному харчуванні.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Russell, S., & Norvig, P. (2010). *Artificial Intelligence: A Modern Approach* (3rd ed.). Prentice Hall.
2. McCarthy, J. (2007). *What is Artificial Intelligence?* Stanford University.
3. Poole, D., Mackworth, A., & Goebel, R. (1998). *Computational Intelligence: A Logical Approach*. Oxford University Press.
4. Kaplan, A., & Haenlein, M. (2019). Siri, Siri, in my hand: Who's the fairest in the land? On the interpretations, illustrations, and implications of artificial intelligence. *Business Horizons*, 62(1), 15-25.
5. Nilsson, N. J. (1998). *Artificial Intelligence: A New Synthesis*. Morgan Kaufmann.
6. Buchanan, B. G. (2005). A (Very) Brief History of Artificial Intelligence. *AI Magazine*, 26(4), 53-60.
7. Dhar, V. (2013). Data Science and Prediction. *Communications of the ACM*, 56(12), 64-73.
8. Brynjolfsson, E., & McAfee, A. (2017). *Machine, Platform, Crowd: Harnessing Our Digital Future*. W. W. Norton & Company.
9. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436-444.
10. Bostrom, N. (2014). *Superintelligence: Paths, Dangers, Strategies*. Oxford University Press.
11. Minsky, M. (1986). *The Society of Mind*. Simon and Schuster.
12. Vinge, V. (1993). The Coming Technological Singularity: How to Survive in the Post-Human Era. *Vision-21: Interdisciplinary Science and Engineering in the Era of Cyberspace*, 11-22.
13. Kerly, A., Hall, P., & Bull, S. (2007). Bringing chatbots into education: Towards natural language negotiation of open learner models. *Knowledge-Based Systems*, 20(2), 177-185.

14. Shawar, B. A., & Atwell, E. (2007). Chatbots: Are they really useful? *LDV Forum*, 22(1), 29-49.
15. Perez-Marin, D., & Pascual-Nieto, I. (2011). *Conversational Agents and Natural Language Interaction: Techniques and Effective Practices*. IGI Global.
16. Abdollahi, B., & Nasraoui, O. (2016). Explainable Matrix Factorization for Collaborative Filtering. In *Proceedings of the 25th International Conference on World Wide Web* (pp. 5-6).
17. Jurafsky, D., & Martin, J. H. (2019). *Speech and Language Processing* (3rd ed.). Prentice Hall.

ДОДАТКИ**Додаток А**

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В. Н. Каразіна

Факультет комп'ютерних наук

Кафедра теоретичної та прикладної системотехніки

Рівень вищої освіти (освітньо-кваліфікаційний рівень) **бакалавр**

галузь знань: **15 – Автоматизація та приладобудування**

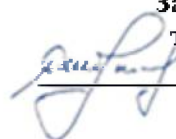
спеціальність: **151 – Автоматизація та комп'ютерно-інтегровані технології**

ЗАТВЕРДЖУЮ

Завідувач кафедри теоретичної
та прикладної системотехніки

д.т.н., проф. Шматков С. І.

«21» грудня 2023 року



**ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ**

Василенко Анастасії Сергіївни

(прізвище, ім'я, по батькові студента)

1. Тема роботи «Телеграм-бот генерації рецептів з використанням штучного інтелекту»

керівник роботи Павлов Анатолій Миколайович, старший викладач

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «03» травня 2024 року № 4101-5/909

2. Строк подання студентом роботи 31 травня 2024 року

3. Перелік питань, які потрібно розробити

1. Аналіз різновидів чат-ботів.
2. Обґрунтування методу застосування штучного інтелекту при генерації рецептів.
3. Розробка та тестування телеграм-боту.

4. План роботи

№ з/п	Назви етапів роботи	Термін виконання етапів роботи
1	Підбір та аналіз наукової літератури	19.12.2023 - 25.01.2024
2	Аналіз технологій побудови чат-ботів та галузей їх використання	21.12.2023 - 2.01.2024
3	Поняття «штучного інтелекту» та його використання в галузі кулінарії	2.01.2024 - 2.02.2024
4	Реалізація чат-боту для генерації рецептів	2.01.2024 - 2.02.2024
5	Тестування комп'ютерної моделі	3.02.2024 - 30.03.2024
6	Підготовка тез доповіді на семінар	3.03.2024 - 30.04.2024
7	Розробка пояснювальної записки.	31.03.2024 - 27.05.2024
8	Оформлення звіту за результатами переддипломної практики	15.05.2024 – 31.05.2024
9	Представлення кваліфікаційної роботи керівнику та рецензенту	31.05.2024
10	Оформлення пояснювальної записки та підготовка презентації	31.05.2024

5. Дата видачі завдання 21.12.2023

Студент

Василенко А. С.

ініціали, прізвище



підпис

Керівник роботи

Павлов А. М.

ініціали, прізвище



підпис

Додаток Б

Затверджую

« ___ » _____ 2023 р.

**Технічне завдання
на розробку програмного виробу «Телеграм-бот генерації рецептів
з використанням штучного інтелекту»**

1.	Введення	1.1. Назва: Телеграм-бот генерації рецептів з використанням штучного інтелекту 1.2. Галузь застосування: Інформаційні технології
2.	Підстава для розробки	2.1. Навчальний план за спеціальністю 151 – Автоматизація та комп'ютерно-інтегровані технології 2.2. Завдання на кваліфікаційну роботу бакалавра №4101-5/895 від «23» травня 2024 (представити як Додаток А до пояснювальної записки до кваліфікаційної роботи).
3.	Призначення розробки	3.1. Мета розробки: розробити телеграм-бота, який генерує індивідуальні рецепти страв на основі вподобань, обмежень та доступних інгредієнтів користувача, використовуючи штучний інтелект. 3.2. Призначення розробки: модель штучного інтелекту, яка буде навчатися на великому наборі даних рецептів, впровадити алгоритми генерації рецептів, які враховуватимуть харчові вподобання користувача, доступні інгредієнти та час приготування. 3.3. Вихідні дані розробки: телеграм-бот з генерації рецептів; вхідні дані розробки: інгредієнти та час приготування.
4.	Технічні вимоги до програмного виробу	4.1. Вимоги до функціональних характеристик: можливість пошуку рецептів, генерація рецептів, надавати фото страв, які бажає приготувати користувач. Також телеграм-бот повинен мати інтерфейс для взаємодії з користувачами. 4.2. Вимоги до надійності: забезпечення безперебійної роботи програмного виробу при будь-яких вимогах користувача в рамках призначення виробу . 4.3. Вимоги до умов експлуатації: немає

		<p>4.4. Вимоги до складу і параметрів технічних засобів: для виконання програми повинен підходити ПК із будь-якою операційною системою сімейства Windows, Linux/Unix, Mac OS X, OS/2, Amiga. Крім того, для роботи потрібний встановлений додаток Telegram.</p> <p>4.5. Вимоги до інформаційної та програмної сумісності: підтримка ОС Linux або Windows 11, підтримка мови програмування, підтримка різних платформ.</p> <p>4.6. Вимоги до маркування та упаковки: вимоги до маркування та упакування не представляються.</p> <p>4.7. Вимоги до транспортування і зберігання: вимоги до транспортування та зберігання не представляються.</p> <p>4.8. Спеціальні вимоги: спеціальні вимоги до програмного виробу не пред'являються.</p>	
5.	Вимоги до програмної документації	<p>Програмною документацією до виробу «Метод аналізу інформативності змінних стану при діагностиці систем з використанням інформаційних критеріїв» вважати:</p> <p>1) Справжнє Технічне завдання на розробку виробу (представити у вигляді Додатку Б до пояснювальної записки до кваліфікаційної роботи).</p> <p>2) Методику розрахунку інформативності змінних стану (у вигляді глав 3.2 та 3.3 пояснювальної записки до кваліфікаційної роботи).</p> <p>3) Опис виробу (представити в розділі 3 пояснювальної записки до кваліфікаційної роботи)</p>	
6.	Вимоги до техніко-економічних показників	<p>Програмною документацією до виробу «Телеграм-бот генерації рецептів з використанням штучного інтелекту» вважати:</p> <p>1) Справжнє Технічне завдання на розробку виробу (представити у вигляді Додатку Б до пояснювальної записки до кваліфікаційної роботи).</p> <p>2) Опис програмного виробу (представити в Розділі 3 пояснювальної записки до кваліфікаційної роботи).</p> <p>3) Джерела базової інформації.</p>	
7.	Стадії і етапи розробки	Дата	Назва етапу
		від 19 грудня 2023 до 25 січня 2024	Підбір та аналіз наукової літератури
		від 21 грудня 2023 до 2 січня 2024	Аналіз технологій побудови чат-ботів та галузей їх використання

		від 2 січня 2024 до 2 лютого 2023	Поняття «штучного інтелекту» та його використання в галузі кулінарії
		від 2 січня 2024 до 2 лютого 2024	Реалізація чат-боту для генерації рецептів
		від 3 лютого 2024 до 30 березня 2024	Тестування комп'ютерної моделі
		від 3 березня 2024 до 30 квітня 2024	Підготовка тез доповіді на семінар
		від 31 березня 2024 до 27 травня 2024	Розробка пояснювальної записки.
		від 15 травня 2024 до 31 травня 2024	Оформлення звіту за результатами переддипломної практики.
		31 травня 2024	Представлення кваліфікаційної роботи керівнику та рецензенту
		31 травня 2024	Оформлення пояснювальної записки та підготовка презентації
8.	Порядок контролю і приймання програмного продукту (моделі)	1. Перевірку ходу розробки програми виконувати раз в 3 тижні. 2. Захист розробленої моделі провести на засіданні Атестаційної комісії. 3. Пояснювальну записку подати на паперових носіях в 1 примірнику і в електронному вигляді в 1 примірнику на CD-R компакт-диску.	

Виконавець
студентка групи КУ- 41
Василенко А. С.

Замовник
старший викладач
Павлов А. М.

Програма і методика випробувань програмного виробу

«Телеграм-бот генерації рецептів з використанням штучного інтелекту»

1. Об'єкт випробувань

1. Назва програмного виробу : «Телеграм-бот генерації рецептів з використанням штучного інтелекту»
2. Галузь застосування : Інформаційні технології
3. Перераховані відомості запозичуються з відповідних розділів Технічного завдання.

2. Мета випробувань

Перевірка відповідності функціональності програмної реалізації системи заявленим функціональним можливостям в технічному завданні (Додаток Б до пояснювальної записки до кваліфікаційної роботи).

3. Загальні положення

1. Підстави для проведення випробувань

Підставою для проведення випробувань є наказ про призначення атестаційної комісії.

2. Місце і тривалість випробувань

Приймальні (приймально-здавальні) випробування проводяться на базі комп'ютерного класу кафедри в період роботи атестаційної комісії.

3. Обсяг випробувань

Приймальні випробування програмного виробу проводяться в обсязі відповідному цієї програми і методики випробувань.

4. Організації, які беруть участь у випробуваннях

Приймальні випробування проводяться атестаційною комісією напередодні засідання (або в процесі засідання) за участю Замовника, Виконавця та інших осіб, присутніх на засіданні.

4. Вимоги до програми або програмного виробу

Модель повинна задовольняти наступним вимогам:

- 4.1. Вимоги до функціональних характеристик: можливість пошуку рецептів, генерація рецептів, надавати фото страв, які бажає приготувати користувач.

Також телеграм-бот повинен мати інтерфейс для взаємодії з користувачами.

4.2. Вимоги до надійності: забезпечення безперебійної роботи програмного виробу при будь-яких вимогах користувача в рамках призначення виробу .

4.3. Вимоги до умов експлуатації: немає

4.4. Вимоги до складу і параметрів технічних засобів: для виконання програми повинен підходити ПК із будь-якою операційною системою сімейства Windows, Linux/Unix, Mac OS X, OS/2, Amiga. Крім того, для роботи потрібний встановлений додаток Telegram.

4.5. Вимоги до інформаційної та програмної сумісності: підтримка ОС Linux або Windows 11, підтримка мови програмування, підтримка різних платформ.

4.6. Вимоги до маркування та упаковки: вимоги до маркування та упакування не представляються.

4.7. Вимоги до транспортування і зберігання: вимоги до транспортування та зберігання не представляються.

4.8. Спеціальні вимоги: спеціальні вимоги до програмного виробу не пред'являються.

5. Вимоги до програмної документації

Документацією до виробу «Телеграм-бот генерації рецептів з використанням штучного інтелекту» вважати:

- 1) Документація по мові програмування та додаткові мануали.
- 2) Програму і методику випробувань розробленої програми (представити як Додаток В до пояснювальної записки до кваліфікаційної роботи).
- 3) Опис програмного виробу (представити в Розділі 3 пояснювальної записки до кваліфікаційної роботи).
- 4) Джерела базової інформації.

6. Засоби і порядок випробувань

6.1 Засоби випробувань

Засоби випробувань представлено на ПК на яких встановлено наступні програмні засоби: інтерпретатор мови програмування.

6.2 Порядок проведення випробувань

Як правило, випробування проводяться в два етапи:

- ознайомчий (1-й етап);
- власне випробування програмного виробу (2-й етап).

A
Г
V

Перелік перевірок, що проводяться на 1 етапі випробувань, включає в себе:

- 1) Перевірку комплектності складу програмної документації здійснюється за критерієм наявності зазначеної в ТЗ документації.
- 2) Перевірку якості програмної документації. Перевірку здійснювати за критерієм відповідності вимогам ГОСТ 19.301-79 ЕСПД. «Програма і методика випробувань».

Перелік перевірок, що проводяться на 2 етапі випробувань, включає в себе:

- 1) Перевірку відповідності технічних характеристик програми вимогам технічного завдання.
- 2) Перевірку ступеня виконання функціональних вимог до програми.
- 3) Методику проведення перевірок:
 - а) Запустити програмне забезпечення.
 - б) Порядок проведення випробувань:
 - Зробити налаштування.
 - Перевірити чи працює програма.
 - Перевірити чи формується звіт.
- 4) Якщо перевірки на першому та другому етапах виконано успішно, то виріб вважається таким, що пройшов випробування.

Для проведення випробувань пропонується тест 1, тест 2 та тест 3.

Тест 1

1. Перевірка виконання програми;
2. Початок програми;
3. Отримання відповіді серверу про успішне створення об'єкту.

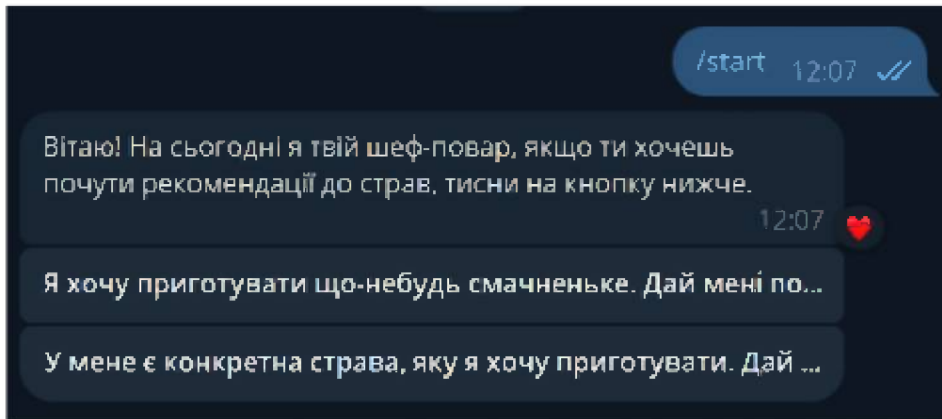


Рис. В.1 Тест 1

Тест 2

1. Перевірка виконання програми
2. Виконання запиту;
3. Отримання результатів.

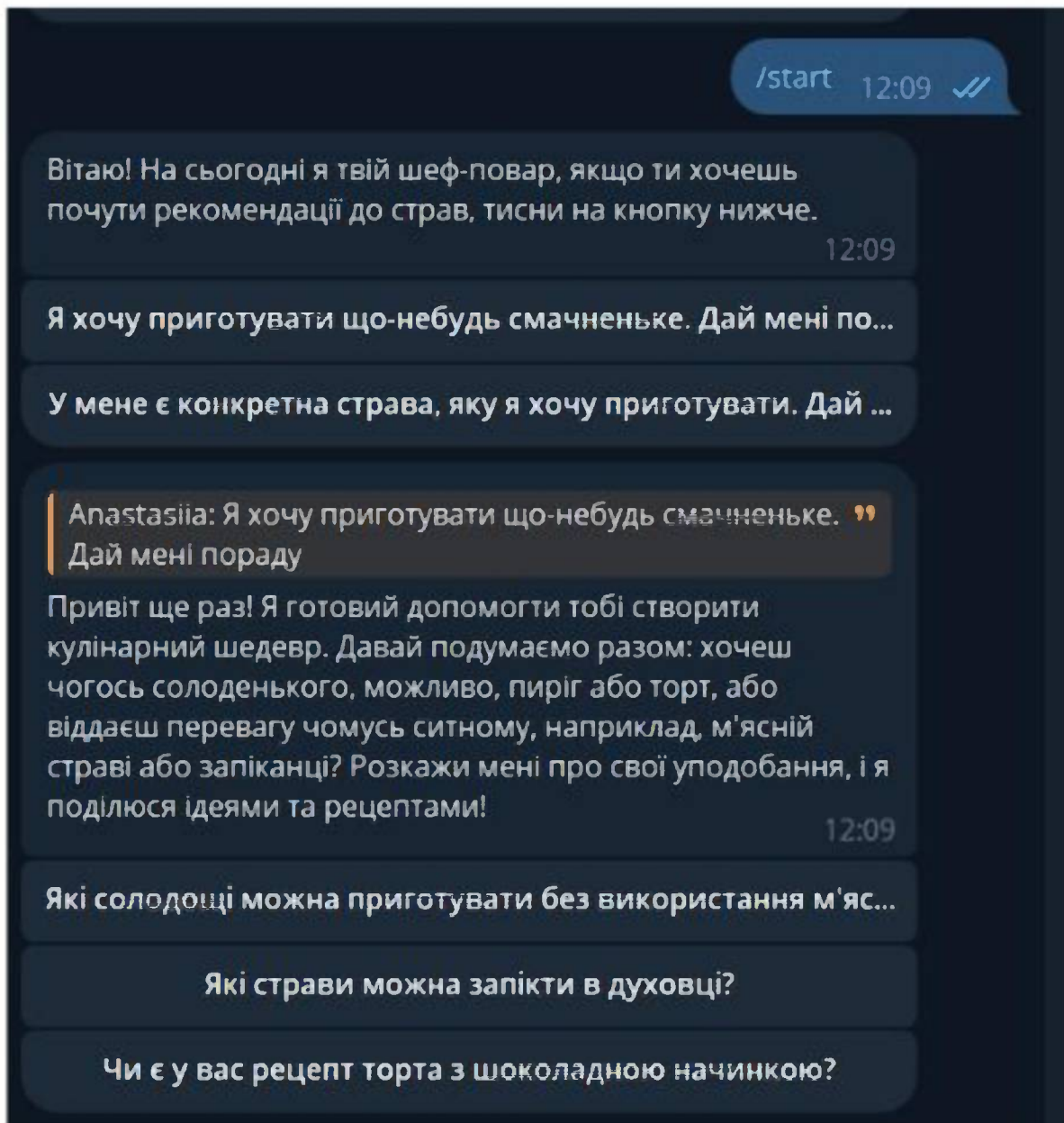


Рис. В.2 Тест 2

Тест 3

4. Перевірка виконання програми
5. Рекомендований рецепт за запитом;
6. Отримання результату.

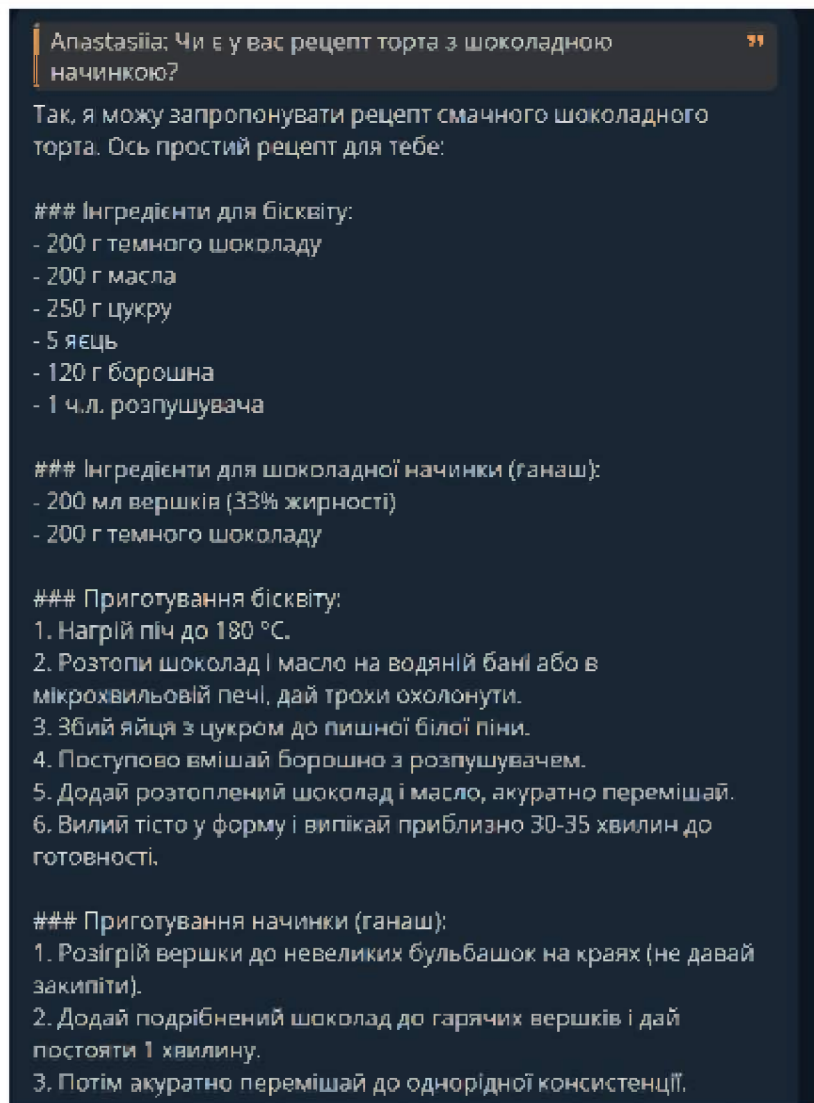


Рис. В.3 Тест 3

Тест вважається пройденим, якщо відбуваються вказані операції і їх відображення у програмному продукті.

Висновки: тест 1 успішно пройшов випробування, тест 2 успішно пройшов випробування і тест 3 успішно пройшов випробування. Випробування пройшло успішно.

Виконавець: студентка групи КУ-41, Василенко А. С. *h*

В.В.

Г
V

Лістинг коду

```
import { Readable } from 'stream'
import { spawn } from 'process'
// first party
import * as voz from './voice'
import * as openai from './openai'
import * as db from './db'
import * as tts from './tts'
import * as config from './config'
import * as t from './telegram'
import { countTokens } from './tokenizer'
import * as util from './util'
import prettyVoice from './pretty-voice'
// server
import Koa from 'koa'
import Router from '@koa/router'
import logger from 'koa-logger'
import bodyParser from 'koa-bodyparser'

const telegram = new t.TelegramClient(config.TELEGRAM_BOT_TOKEN)

// HACK: I don't like storing this on the top level
let botUsername = "

const PER_USER_CONCURRENCY = 3
// Mapping of user_id to the number of requests they have in flight.
const inflights = new Map<number, number>()
// In-memory mapping of user_id to number of OpenAI prompts (i.e. text and /img) they've
incurred.
// Used as a crude rate limiter for the demo bot.
const promptCounts = new Map<number, number>()
```

```

async function initializeBot(): Promise<{
  botUsername: string
  webhookUrl: string
}> {
  // https://core.telegram.org/bots/api#authorizing-your-bot
  // Check if webhook: https://core.telegram.org/bots/api#getwebhookinfo
  const webhookInfo = await telegram.getWebhookInfo()

  if (webhookInfo.url !== config.WEBHOOK_URL) {
    await telegram.setWebhook({
      url: config.WEBHOOK_URL,
      // allowed_updates: ['message'],
    })
  }

  // HACK: Assigning to global
  botUsername = await telegram.getMe().then((x) => x.username)

  const commands = [
    {
      command: 'voice',
      description: 'set bot language',
    },
    {
      command: 'voiceoff',
      description: 'disable voice responses',
    },
    {
      command: 'voiceon',
      description: 'enable voice responses',
    },
    {
      command: 'temp',
      description: 'set temperature (randomness)',
    },
  ],

```

```

    { command: 'info', description: 'show bot settings' },
    {
      command: 'clear',
      description: 'reset chat context',
    },
    { command: 'retry', description: 'regenerate last bot answer' },
    // Not adding /prompt to the list because it takes an argument.
    // Only makes sense to register commands that have no arguments
    // thus can be clicked when they are autolinked in text.
    //
    // {
    //   command: 'prompt',
    //   description: 'set a master prompt',
    // },
    { command: 'promptclear', description: 'clear master prompt' },
    {
      command: 'model',
      description: 'switch to another model',
    },
  ]

  await telegram.setMyCommands(commands)

  return {
    botUsername,
    webhookUrl: config.WEBHOOK_URL,
  }
}

// SERVER

const app = new Koa()
app.use(logger())
app.use(bodyParser())
const router = new Router()

```

```

router.get('/', async (ctx) => {
  ctx.body = 'Bot is online'
})

// Pipes input ogg stream into output mp3 stream which is streamed to OpenAI API for text
transcription.
//
// This is cool because we don't store anything on the filesystem; everything stays in streams.
async function transcribeOggStream(input: Readable): Promise<string> {
  input.on('error', (err) => {
    console.log('Error on ogg input stream:', err)
  })

  return await new Promise((resolve, reject) => {
    try {
      const args = ['-f', 'ogg', '-i', 'pipe:0', '-f', 'mp3', 'pipe:1']
      const proc = spawn('ffmpeg', args, {
        // ensure stderr is inherited from the parent process so we can see ffmpeg errors
        // Note: noisy since it dumps to console.log on every ffmpeg usage.
        // stdio: ['pipe', 'pipe', 'inherit'],
      })

      input.pipe(proc.stdin)

      proc.on('error', (err) => {
        console.error('Error in ffmpeg process:', err)
        reject(err)
      })

      proc.on('exit', (code, signal) => {
        if (typeof code === 'number' && code !== 0) {
          const errorMsg = `ffmpeg exited with code ${code} and signal ${String(
            signal,
          )}`
        }
      })
    }
  })
}

```

```

        console.error(errorMsg)
        reject(new Error(errorMsg))
    }
    input.destroy()
    proc.stdin.destroy()
})

openai
    .transcribeAudio(proc.stdout)
    .then((result) => {
        resolve(result.data.text)
    })
    .catch(resolve)
} catch (err) {
    console.error('transcribe error:', err)
    reject(err)
}
})
}

// voice:English --> show the countries
// voice:English:US --> show the US voices
// voice:English:US:Jenny -> pick the Jenny voice
//
//
// Note: Always have an even number of voices per country.
const VOICE_MENU: Record<string, any> = {
    English: {
        US: {
            Jenny: 'en-US-JennyNeural',
            Brandon: 'en-US-BrandonNeural',
        },
        UK: {
            Abbi: 'en-GB-AbbiNeural',
            Alfie: 'en-GB-AlfieNeural',

```

```
},  
Australia: {  
    Annette: 'en-AU-AnnetteNeural',  
    Darren: 'en-AU-DarrenNeural',  
},  
Ireland: {  
    Emily: 'en-IE-EmilyNeural',  
    Connor: 'en-IE-ConnorNeural',  
},  
},  
Spanish: {  
    US: {  
        Paloma: 'es-US-PalomaNeural',  
        Alonso: 'es-US-AlonsoNeural',  
    },  
    Spain: {  
        Abril: 'es-ES-AbrilNeural',  
        Alvaro: 'es-ES-AlvaroNeural',  
    },  
    Mexico: {  
        Beatriz: 'es-MX-BeatrizNeural',  
        Candela: 'es-MX-CandelaNeural',  
        Cecilio: 'es-MX-CecilioNeural',  
        Gerardo: 'es-MX-GerardoNeural',  
    },  
    Argentina: {  
        Elena: 'es-AR-ElenaNeural',  
        Tomas: 'es-AR-TomasNeural',  
    },  
    Colombia: {  
        Salome: 'es-CO-SalomeNeural',  
        Gonzalo: 'es-CO-GonzaloNeural',  
    },  
    Venezuela: {  
        Paola: 'es-VE-PaolaNeural',
```

```

    Sebastian: 'es-VE-SebastianNeural',
  },
},
French: {
  France: {
    Brigitte: 'fr-FR-BrigitteNeural',
    Alain: 'fr-FR-AlainNeural',
    Celeste: 'fr-FR-CelesteNeural',
    Claude: 'fr-FR-ClaudeNeural',
  },
  Canada: {
    Sylvie: 'fr-CA-SylvieNeural',
    Antoine: 'fr-CA-AntoineNeural',
  },
},
}

```

```

function inlineKeyboardForModelSelect(
  model: db.Model,
): Array<Array<{ text: string; callback_data: string }>> {
  const rows = [
    [
      {
        text: `${
          model === 'gpt-3.5-turbo' ? '☑️': "
        } GPT-3.5 (Fastest)`,
        callback_data: 'model:gpt-3.5-turbo',
      },
    ],
    [
      {
        text: `${
          model === 'gpt-4' ? '☑️': "
        } GPT-4 (Smarter, slower)`,

```

```

        callback_data: 'model:gpt-4',
      },
    ],
    [
      {
        text: `${
          model === 'text-davinci-003' ? '✅' : "
        } Davinci (Weird)`,
        callback_data: 'model:text-davinci-003',
      },
    ],
  ]

  return rows
}

// async function handleCallbackQuery(body: t.CallbackQuery) {
async function handleCallbackQuery(
  chat: db.Chat,
  userId: number,
  messageId: number,
  callbackQueryId: string,
  callbackData: string,
): Promise<void> {
  const chatId = chat.id
  // https://core.telegram.org/bots/api#inlinekeyboardmarkup
  // const chatId = body.message.chat.id
  // const messageId = body.message.message_id
  // const callbackData = body.data

  if (callbackData.startsWith('temp:')) {
    let temperature = Number.parseFloat(callbackData.split(':')[1])
    if (temperature < 0) temperature = 0
    if (temperature > 1) temperature = 1
  }
}

```

```

await db.setTemperature(chatId, temperature)
// https://core.telegram.org/bots/api#answercallbackquery
await telegram.sendMessage(
  chatId,
  `🌡 Temperature changed to ${temperature.toFixed(1)}`,
)
} else if (callbackData.startsWith('model:')) {
  if (canManageModel(userId)) {
    const model = callbackData.split(':')[1]
    if (!db.isModel(model)) {
      await telegram.sendMessage(
        chatId,
        `❌ Unhandled model. Sorry!`,
        messageId,
      )
      return
    }
  }

  if (chat.model === model) {
    // Ignore button dupe presses
    await telegram.answerCallbackQuery(callbackQueryId, "")
    return
  }

  await db.setModel(chatId, model)
// https://core.telegram.org/bots/api#answercallbackquery
await Promise.all([
  // Dismiss "Loading..." callback message
  telegram.answerCallbackQuery(callbackQueryId, ""),
  telegram.sendMessage(chatId, `Model changed to ${model}`),
  telegram.editMessageReplyMarkup(
    chatId,
    messageId,
    inlineKeyboardForModelSelect(model),
  ),
])
}

```

```

    ),
  ])
}
} else if (callbackData === 'voice' || callbackData.startsWith('voice:')) {
  // voice:English --> show the countries
  // voice:English:US --> show the US voices
  // voice:English:US:Jenny -> pick the Jenny voice
  const segments: string[] = callbackData.split(':')
  switch (segments.length) {
    // voice --> Show the language selections
    case 1:
      {
        const langs = Object.keys(VOICE_MENU)
        const inlineKeyboard = util
          .chunksOf(2, langs)
          .map((chunk) => {
            return chunk.map((lang) => {
              return {
                text: lang,
                callback_data: `voice:${lang}`,
              }
            })
          })
        await telegram.editMessageReplyMarkup(
          chatId,
          messageId,
          inlineKeyboard,
        )
      }
    return
  // voice:English --> List the countries available for English
  case 2:
    {
      // TODO: handle VOICE_MENU[segments[1]] returns nothing
      const countryNames = Object.keys(VOICE_MENU[segments[1]])
    }
  }
}

```

```

const inlineKeyboard = [
  [{ text: '⏪ Back', callback_data: 'voice' }],
].concat(
  util.chunksOf(2, countryNames).map((chunk) => {
    return chunk.map((countryName) => {
      return {
        text: countryName,
        callback_data: `voice:${segments[1]}:${countryName}`,
      }
    })
  })
)
await telegram.editMessageReplyMarkup(
  chatId,
  messageId,
  inlineKeyboard,
)
}
return
// voice:English:US --> List the voices available for US English
case 3:
{
  const inlineKeyboard = [
    [
      {
        text: '⏪ Back',
        callback_data: `voice:${segments[1]}`,
      },
    ],
  ].concat(
    util
      .chunksOf<[string, string]>(
        2,
        Object.entries(

```

```

        VOICE_MENU[segments[1]][segments[2]],
    ),
)
.map((chunk) => {
    return chunk.map(([name, code]) => {
        return {
            text: name,
            callback_data: `voice:${segments[1]}:${segments[2]}:${code}`,
        }
    })
}),
)

```

// Note: I originally sent editMessageReplyMarkup and editMessage at the same time,

// the latter updating the menu message text, but there's a race condition that would // prevent the inline keyboard from updating, so now I don't update the message text.

```
await telegram.editMessageReplyMarkup(
```

```
    chatId,
    messageId,
    inlineKeyboard,
```

```
)
```

```
}
```

```
return
```

// voice:English:US:en-US-JennyNeural --> Pick this voice

```
case 4: {
```

```
    const newVoice = segments[3]
```

```
    await db.changeVoice(chatId, newVoice)
```

// <https://core.telegram.org/bots/api#answercallbackquery>

```
await telegram.answerCallbackQuery(
```

```
    callbackQueryId,
```

```
    `✅ Voice changed to ${newVoice}`,
```

```
)
```

```
await telegram.deleteMessage(chatId, messageId)
```

```

    }
  }
}

// Only used for private convos.
async function processUserMessage(
  user: db.User,
  chat: db.Chat,
  message: t.Message,
): Promise<void> {
  console.log( [processUserMessage]', user, chat, message)

  // Crude demo bot rate limit.
  // Only rate-limit guests.
  // HACK: Overloading canManageModel as an authz check.
  if (!canManageModel(user.id)) {
    const promptLimit = 10
    promptCounts.set(user.id, (promptCounts.get(user.id) ?? 0) + 1)
    if ((promptCounts.get(user.id) ?? 0) >= promptLimit) {
      await telegram.sendMessage(
        chat.id,
        '✘ You have reached the maximum number of prompts on the demo bot.\n\nIf you
like this bot, you can host your own: https://github.com/danneu/telegram-chatgpt-bot',
        message.message_id,
      )
      return
    }
  }

  // TODO: This does happen, so figure out when this case happens and handle it.
  // I've seen it happen in group chats, maybe when adding the bot or something.
  if (message.text === undefined && message.voice === undefined) {
    console.log('TODO: message had no .text nor .voice. Skipping...')
  }
}

```

```

    return
  }

  // Reject messages from groups until they are deliberately supported.
  if (message.chat?.type !== 'private') {
    console.log('rejecting non-private chat')
    return
  }

  if (message.from === undefined) {
    // Message must have `from` property by now.
    return
  }

  const userId = message.from.id
  const chatId = message.chat.id
  const messageId = message.message_id
  let userText = message.text

  // Check if voice and transcribe it
  // Only possible in private chats
  if (message.voice !== undefined) {
    await telegram.indicateTyping(chatId)

    const remoteFileUrl = await telegram.getFileUrl(message.voice.file_id)
    const response = await fetch(remoteFileUrl)
    if (response.status !== 200) {
      throw new Error(
        `Error fetching telegram .ogg file. ${response.status} ${response.statusText}`,
      )
    }
  }

  // @ts-expect-error: Not sure how to annotate response.body
  userText = await transcribeOggStream(Readable.fromWeb(response.body))

  // Send transcription to user so they know how the bot interpreted their memo.

```

```

if (userText.length > 0) {
  await telegram.sendMessage(
    chatId,
    `(Transcription) <i>${userText}</i>`,
    messageId,
  )
} else {
  // Transcription was empty. User didn't seem to say anything in the voice memo.
  await telegram.sendMessage(
    chatId,
    `⚠️ No speech detected in this voice memo. Try again.`,
    messageId,
  )
  return
}

// We can send a new typing indicator since we just sent a message
await telegram.indicateTyping(chatId)
}

if (userText === undefined) {
  // userText was either populated by message.text or message.voice transcription.
  return
}

// Check if command and short-circuit
const command = parseCommand(userText)
if (command !== undefined) {
  await handleCommand(user.id, chat, messageId, command)
  return
}

await telegram.indicateTyping(chatId)

```

```

// TODO: Consider inserting prompt and then editing ttsElapsed like
// how it worked before completeAndSendAnswer was extracted.
// That way we insert sooner in case of tts failure.
const result = await completeAndSendAnswer(chat, messageId, userText)

if (result == null) {
  // Short-circuited because of error and error was sent to user.
  return
}

await db.insertAnswer({
  userId,
  chatId,
  prompt: userText,
  promptTokens: result.promptTokens,
  messageId: result.messageId,
  answer: result.answer,
  answerTokens: result.answerTokens,
  gptElapsed: result.gptElapsed,
  ttsElapsed: result.ttsElapsed,
  model: chat.model,
})
}

// Extracted from processUserMessage so that I can easily do this work from the
// /retry command. However, note that /retry doesn't create/edit database entries
// but processUserMessage callsite does.
//
async function completeAndSendAnswer(
  chat: db.Chat,
  messageId: number,
  userText: string,
): Promise<
  | {
    gptElapsed: number

```

```

    ttsElapsed: number | undefined
    promptTokens: number
    answerTokens: number
    answer: string
    messageId: number
  }
  | undefined
> {
  const messages = await db.listHistory(
    chat.id,
    chat.master_prompt ?? config.MASTER_PROMPT,
    userText,
  )

  const gptStart = Date.now()
  let tokens
  try {
    if (chat.model === 'text-davinci-003') {
      let prompt = ''
      for (const { role, content } of messages) {
        switch (role) {
          case 'system':
          case 'user':
            prompt += `User: ${content}\n`
            break
          case 'assistant':
            prompt += `Helper: ${content}\n`
            break
        }
      }
      prompt += 'Helper: '
      // console.log(prompt)
      tokens = openai.streamTextCompletions(prompt, chat.temperature)
    } else {
      tokens = openai.streamChatCompletions(

```

```

        messages,
        chat.model,
        chat.temperature,
    )
}
} catch (err: any) {
    if (err.response?.status === 429) {
        await telegram.sendMessage(
            chat.id,
            `✘ Bot is rate-limited by OpenAI API. Try again soon.`,
            messageId,
        )
        return
    } else {
        throw err
    }
}

const { answer, messageId: answerMessageId } = await streamTokensToTelegram(
    chat.id,
    messageId,
    // TODO: Chunks don't actually map to tokens. I should rename.
    tokens,
    chat.model,
)

const gptElapsed = Date.now() - gptStart
const promptTokens = countTokens(userText)
const answerTokens = countTokens(answer)

// Send trailing voice memo.
let ttsElapsed: number | undefined
if (chat.send_voice) {
    telegram
        .indicateTyping(chat.id) // Don't await
        .setVoiceMessage({ text: '1.0', callback_data: 'temp:1.0' },
            ],

```