МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інтелектуальних
інформаційних систем
_____Юрій КОНДРАТЕНКО
«____»_____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

СТВОРЕННЯ ЧАТ-БОТУ ДЛЯ ПЛАНУВАННЯ ПОДОРОЖЕЙ

Спеціальність 122 Комп'ютерні науки
Освітня програма «Комп'ютерні науки»

Здобувач _____ Дмітрій ШАТАЛОВ
«___»_____ 2025 р.

Керівник канд. техн. наук, доцент _____ Євген СІДЕНКО
«___»_____ 2025 р.

Миколаїв – 2025

Чорноморський національний університет імені Петра Могили

(повне найменування закладу вищої освіти)

Факультет	Комп’ютерних наук
Кафедра	Інтелектуальних інформаційних систем
Рівень вищої освіти	Перший (бакалаврський)
Освітній ступень	Бакалавр
Спеціальність	122 Комп’ютерні науки
Освітня програма	Комп’ютерні науки

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем

_____ Юрій КОНДРАТЕНКО

«____» _____ 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу здобувача

Шаталова Дмітрія Сергійовича

(прізвище, ім’я, по батькові здобувача)

1. Тема кваліфікаційної роботи: «Створення чат-боту для планування подорожей».

Керівник роботи: Сіденко Євген Вікторович, доцент кафедри інтелектуальних інформаційних систем, канд. техн. наук, доцент.

Затверджена наказом ЧНУ ім. Петра Могили від «18» листопада 2024 р. № 321.

2. Строк представлення кваліфікаційної роботи «___» _____ 2025 рплатформою, інформація про типи віртуальних товарів (скіни, ключі, предмети тощо), а також вимоги до безпеки транзакцій та захисту даних користувачів

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні: створення

помічника для подоржей який надає необхідну інформацію по локації, погоди та потрібних місць для освідчення користувача.

4. Перелік питань, що підлягають розробці:

– аналіз сучасного впливу сучасних цифрових технологійна туризм;

– аналіз та порівняння схожих тревел-застосунків, аналіз їх ефективності та особливостей;

– вибір і аналіз технологій, способи інтеграції з різними застосунками для надання точної інформації по запиту користувача;

– огляд і порівняння різних підходів до забезпечення точнішої інформації та швидкого доступу;

– програмна реалізація та правильна реєстрація для безпеки користувачіав тестування, порівняльний аналіз функціональності, безпеки та зручності користування розробленим вебзастосунком.

5. Перелік графічних матеріалів: презентація

Керівник роботи _____ Євген СІДЕНКО
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Здобувач _____ Дмітрій ШАТАЛОВ
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Дата видачі завдання «18» грудня 2024 р.

КАЛЕНДАРНИЙ ПЛАН
кваліфікаційної роботи

Тема: Створення чат-боту для планування подорожей

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Отримання завдання на виконання КР	18.12.2024	20.12.2024	Виконано
2.	Аналіз предметної області та постановка задачі	21.12.2024	30.01.2025	Виконано
3.	Огляд літературних джерел за темою кваліфікаційної роботи, зокрема огляд публікацій та аналогічних систем, щодо розробки телеграм бота для подорожей	31.01.2025	01.03.2025	Виконано
4.	Огляд існуючих рішень для вирішення поставленої задачі	02.03.2025	01.04.2025	Виконано
5.	Реалізація обраних технологій з аналізом отриманих результатів	02.04.2025	15.05.2025	Виконано
6.	Перший попередній захист КР на засіданні комісії кафедри	16.05.2025	16.05.2025	Виконано
7.	Корегування роботи за результатами попереднього захисту	29.05.2025	29.05.2025	Виконано
8.	Другий попередній захист КР на засіданні комісії кафедри	30.05.2025	30.05.2025	Виконано
9.	Доробка та остаточне оформлення КР	31.05.2025	10.06.2025	Виконано
10.	Подання КР, її електронної копії та інших документів (відгуку, рецензії) до захисту	15.06.2025	15.06.2025	Виконано

Керівник роботи _____ Євген СІДЕНКО
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Здобувач _____ Дмітрій ШАТАЛОВ
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Дата складання календарного плану «29» січня 2025 р.

АНОТАЦІЯ

до кваліфікаційної роботи
здобувача групи 401 ЧНУ ім. Петра Могили

Шаталова Дмітрія Сергійовича

на тему: «**СТВОРЕННЯ ЧАТ-БОТУ ДЛЯ ПЛАНУВАННЯ ПОДОРОЖЕЙ**»

Актуальність роботи зумовлена стрімким розвитком цифрових технологій у туристичній сфері, зростанням попиту на персоналізовані сервіси для планування подорожей та необхідністю інтеграції різних джерел інформації у єдину платформу. В роботі досліджено особливості створення інтелектуального Telegram-бота для допомоги мандрівникам, який забезпечує оперативний доступ до інформації про локації, погоду, визначні місця та інші сервіси для подорожей.

Об'єктом роботи є процеси планування подорожей, тобто сукупність дій і рішень, які приймає людина під час підготовки до поїздки. Це охоплює вибір напрямку, визначення маршруту, врахування погодних умов, підбір локацій для відвідування та організацію логістики, що в цілому формує ефективний та комфортний план майбутньої подорожі.

Предметом роботи є програмні засоби й технології, які використовуються для розробки чат-ботів, здатних допомагати користувачам у плануванні подорожей. Він охоплює інструменти, програмні рішення та методи, за допомогою яких створюються функції для автоматизації пошуку інформації, інтеграції з сервісами погоди, картографії та рекомендацій, а також забезпечення зручної взаємодії мандрівника з ботом.

Метою роботи є розробка ефективного, масштабованого і безпечного Telegram-бота для мандрівників із використанням стеку Python, Aiogram, MongoDB, інтеграцією погодних та картографічних сервісів, а також впровадженням рекомендаційних модулів.

У результаті проведеної роботи проаналізовано сучасні тревел-застосунки та їх функціональність, порівняно ключові картографічні та погодні API, обґрунтовано вибір MongoDB для зберігання користувацьких даних. Реалізовано архітектуру Telegram-бота, описано ключові функціональні модулі: навігацію, рекомендації цікавих місць, прогнозування погоди, збереження історії запитів та вподобань користувача. Також визначено перспективи розвитку системи: розширення бази знань, інтеграція аналітики та використання елементів штучного інтелекту для глибшої персоналізації.

Робота складається з чотирьох розділів: аналізу предметної області й сучасних рішень, огляду технологій, проектування та реалізації програмного забезпечення, а також аналізу перспектив розвитку. Загальний обсяг – 86 сторінок. Робота містить додаток, 7 рисунків, 4 таблиць і 29 джерел.

Ключові слова: Telegram-бот, туризм, Aiogram, MongoDB, Google Maps, API, прогноз погоди, персоналізація, цифрові технології.

ABSTRACT

to the qualification work by the student of the group 401 of Petro Mohyla Black Sea National University

Shatalov Dmytro

«TRAVEL ASSISTANT SOFTWARE DEVELOPMENT IN TELEGRAM BOT FORMAT»

The relevance of this research is determined by the rapid development of digital technologies in the tourism industry, the increasing demand for personalized travel services, and the need to integrate diverse information sources into a single platform. This thesis explores the development of an intelligent Telegram bot designed to assist travelers by providing real-time access to location information, weather forecasts, points of interest, and other essential travel services.

The **object of this work** is the processes of travel planning, meaning the set of actions and decisions a person makes while preparing for a trip. This includes choosing a destination, determining the route, taking weather conditions into account, selecting places to visit, and organizing logistics, all of which together form an effective and comfortable travel plan.

The **subject of this work** is the software tools and technologies used to develop chatbots that assist users in travel planning. It covers the tools, software solutions, and methods employed to create functions for automating information search, integrating weather and mapping services, providing recommendations, and ensuring convenient interaction between the traveler and the bot.

The **main goal** of the thesis is to develop an efficient, scalable, and secure Telegram bot for travelers using Python, Aiogram, MongoDB, as well as integration with mapping and weather APIs, and the implementation of recommendation modules.

As a result of the research, current travel applications and their functionality were analyzed, key mapping and weather APIs were compared, and the choice of MongoDB for user data storage was substantiated. The Telegram bot’s architecture was implemented, and

the main functional modules were described: navigation, attraction recommendations, weather forecasting, and storage of user preferences and query history. Prospects for further development were outlined, including expanding the knowledge base, integrating analytics, and using artificial intelligence elements for deeper personalization.

The thesis consists of four chapters: analysis of the subject area and existing solutions, technology overview, software design and implementation, and analysis of future prospects. The total length is 86 pages. The work includes an appendix, 7 figures, 4 tables, and 29 references.

Key words: Telegram bot, tourism, Aiogram, MongoDB, Google Maps, API, weather forecast, personalization, digital technologies..

ЗМІСТ

ВСТУП

У сучасному світі туризм невпинно трансформується під впливом цифрових технологій, які стали невід'ємною частиною щоденного життя мільйонів людей. Мобільні додатки, онлайн-сервіси та месенджери істотно спростили процес планування та організації подорожей, забезпечивши користувачам доступ до актуальної інформації про локації, маршрути, транспорт, погоду й визначні місця у будь-який момент і з будь-якої точки світу.

Особливого значення в цій сфері набувають інтелектуальні помічники у форматі чат-ботів, які дозволяють об'єднати різноманітні сервіси в одному зручному інтерфейсі та оперативно реагувати на індивідуальні запити користувачів. Використання Telegram-ботів у сфері туризму дає змогу автоматизувати рутинні завдання, підвищити якість інформаційної підтримки, персоналізувати рекомендації й забезпечити новий рівень зручності під час подорожей.

Актуальність розробки полягає у необхідності створення сучасного, ефективного та безпечного інструменту для автоматизації інформаційної підтримки туристів із використанням новітніх технологій інтеграції з зовнішніми API, гнучкого зберігання даних та машинного навчання для персоналізації сервісу.

Для досягнення мети були поставлені такі основні завдання:

– провести аналіз сучасного стану і тенденцій розвитку цифрових технологій у туризмі;

– здійснити порівняння наявних тревел-застосунків і сервісів;

– обґрунтувати вибір технологій, інструментів і архітектури програмного забезпечення;

– реалізувати ключові функціональні модулі Telegram-бота, включаючи інтеграцію з зовнішніми API, зберігання й обробку користувацьких даних;

– оцінити перспективи подальшого розвитку системи та можливості масштабування.

Структура роботи відображає послідовність вирішення поставлених завдань — від аналізу предметної області та вибору технологій до опису практичної реалізації програмного продукту і формування висновків щодо його ефективності та потенціалу розвитку.

1 ЦИФРОВІ ТЕХНОЛОГІЇ В СУЧАСНОМУ ТУРИЗМІ: ТРАНСФОРМАЦІЯ ПОДОРОЖЕЙ

1.1 Вплив цифрових технологій на туризм

Сучасний світ туризму значно змінився завдяки цифровим технологіям на всі аспекти планування, організації та проведення подорожей і відпусток. Розвиток мобільних додатків і месенджерів, створив нові можливості для туристів, дозволяючи автоматизувати багато рутинних завдань і значно спростити комунікацію між мандрівниками та сервісами. Дослідження, опубліковане в European Journal of Tourism, Hospitality and Recreation, підкреслює, що смартфони та мобільні туристичні додатки суттєво змінили досвід мандрівників, дозволяючи їм почуватися більш впевнено та безпечно під час подорожей (researchgate.net). Крім того, сучасні додатки пропонують персоналізовані рекомендації, засновані на вподобаннях користувача, що робить подорожі більш насиченими та індивідуальними (sundewsolutions.com).

З розвитком цифрових технологій традиційні туристичні агентства зіткнулися з серйозними викликами. Раніше вони були основним джерелом інформації та бронювання для мандрівників. Однак з появою онлайн-платформ і мобільних додатків їх роль значно знизилася. Згідно з даними Eurostat, у 2021 році в секторі туристичних агентств і туроператорів у ЄС було зайнято близько 400 тисяч осіб, що становить лише невелику частину від загальної кількості зайнятих у туристичній індустрії (ec.europa.eu). Це свідчить про зниження значущості традиційних агентств у сучасній туристичній екосистемі. Споживачі все частіше віддають перевагу самостійному плануванню подорожей, використовуючи зручні та доступні цифрові інструменти, що робить послуги традиційних агентств менш затребуваними. У цьому контексті чат-боти та інші застосунки стали важливим інструментом для надання інформаційної підтримки, бронювання послуг, планування маршрутів і забезпечення

персоналізованих, якісних рекомендацій.

Боти та застосунки для подорожей можуть виконувати широкий спектр функцій, включаючи пошук рейсів, бронювання готелів, відстеження погодних умов, створення персоналізованих маршрутів, конвертацію валют, переклад текстів і навіть рекомендації щодо безпеки. Вони забезпечують оперативний доступ до інформації і, завдяки інтеграції з зовнішніми API, можуть об'єднувати дані з різних джерел для створення комплексних рішень.

У 2023 році, згідно з даними Всесвітньої туристичної організації (UNWTO), кількість міжнародних туристів досягла понад 1,4 мільярда, що свідчить про високий попит на зручні інформаційні сервіси. Ці сервіси допомагають мандрівникам знаходити вигідні авіаквитки, бронювати житло, дізнаватися про місцеві визначні пам'ятки, орієнтуватися у нових містах і навіть отримувати поради щодо культури та правил поведінки в інших країнах. Такі сервіси також сприяють економії часу та коштів, надаючи можливість оперативно реагувати на зміни в планах подорожей.

Сучасні мандрівники очікують від цифрових сервісів швидкості, точності та персоналізації. Згідно з дослідженнями (Smith et al., 2022), ключовими факторами, які впливають на задоволеність користувачів, є простота використання, доступність інформації в реальному часі та можливість швидкого та якісного реагування на зміни. Наприклад, туристи часто стикаються з проблемами, пов'язаними з затримками рейсів різних видів транспорту , змінами в розкладі того ж самого транспорту, с великими списком непередбаченних ситуацій та інцендентів під час битових ситуцій під час подорожей або мовними бар'єрами. У таких випадках застосунки иа Боти можуть стати незамінними помічниками, надаючи своєчасну інформацію та підтримку.

Одним з головних факторів успіху таких застосунків є їхня здатність забезпечувати персоналізований досвід для кожного користувача. Наприклад, бот може враховувати історію попередніх запитів, переваги користувача щодо типів

транспорту або умов проживання, а також автоматично пропонувати найбільш підходящі варіанти на основі поточного місцезнаходження або цільової країни подорожі.

Також варто відзначити важливість безпеки і конфіденційності даних користувачів, особливо коли мова йде про міжнародні подорожі. У зв'язку з цим розробники ботів повинні приділяти особливу увагу захисту персональних даних, шифруванню повідомлень і дотриманню міжнародних стандартів безпеки.

Крім того, значну роль відіграє можливість інтеграції таких ботів з соціальними мережами, платіжними системами та іншими цифровими платформами, що дозволяє створювати більш комплексні рішення для туристів. Наприклад, деякі боти можуть автоматично відправляти електронні квитки, нагадувати про час реєстрації на рейс або навіть пропонувати туристичні пакети на основі інтересів користувача.

Підсумовуючи, можна сказати, що цифрові сервіси, додатки та боти для подорожей сьогодні не просто полегшують планування та організацію поїздок, але й значно покращують якість самого досвіду подорожей, забезпечуючи індивідуальний підхід і оперативну підтримку. У сучасному світі, де технології змінюються зі швидкістю світла, важливо мати під рукою інструменти, що дозволяють відчути себе впевнено в будь-якому куточку світу.

1.2 Аналіз та порівняння тревел-застосунків

Для успішної розробки якісного додатка для прогнозування подорожей, я вирішив провести порівняння декількох популярних сервісів, які надають подібні функціональності. Метою цього аналізу стало визначення найбільш ефективних рішень, які можна інтегрувати в моє власне додаток, а також відбір найкращих особливостей з кожного з цих сервісів.

Перш за все, я обрав такі додатки, як TripIt, Wanderlog, Roadtrippers та

Rome2Rio, оскільки вони мають схожі функції, що дозволяють планувати подорожі та надавати корисну інформацію для мандрівників. Кожен з цих сервісів має свої сильні сторони, які варто врахувати при створенні нового продукту.

Таблиця 1.1 - Порівняння застосунків які обирають користувачі

Додаток	Основні функції	Основні переваги	Широта вибору	Особливі фішки	Популярність	Інтерфейс
TripIt	Автоматичне створення маршруту, збереження квитків, перегляд інформації про подорожі	Зручність у збереженні інформації про подорожі, інтеграція з календарем	Можливість планувати подорожі по всьому світу	Автоматичне створення маршрутів з урахуванням змін у квитках та бронюваннях	Серед популярних сервісів для планування подорожей, часто використовується бізнесменами та туристами	Інтерфейс інтуїтивно зрозумілий, але має багато елементів, що потребують звикання
Wanderlog	Планування маршруту, бронювання готелів, рекомендації для подорожей, інтеграція з Google Maps	Можливість планування дорожніх поїздок з збереженням усіх деталей, можливість спільного використання планів	Вибір маршрутів для поїздок по різних країнах, рекомендації місць відпочинку	Інтерактивне планування подорожей з мапою та можливістю добавлення пам'яток	Популярний серед туристів, які планують довгі поїздки та авто-мандрівки	Дружній інтерфейс з можливістю створення маршруту з простим додаванням точок

Кінець таблиці 1.1

Roadtrippers	Планування маршрутів для автомобільних подорожей, інформація про зупинки, рекомендації щодо ресторанів	Надійний інтерфейс для автомобільних подорожей, широкі можливості пошуку зупинок та місць для відпочинку	Доступність маршрутів по США та деяким іншим країнам	Детальна інформація про місця відпочинку, ресторани, історичні пам'ятки	Популярний в США, зручний для автомобільних подорожей по країні	Інтерфейс спрямований на зручність водіїв, з візуалізацією маршрутів на карті
Rome2Rio	Пошук маршрутів між містами, порівняння цін на транспорт, інформація про маршрути літаків, потягів, автобусів	Широкий вибір видів транспорту, надання варіантів для різних міст	Пошук міжнародних маршрутів, у тому числі для транспорту з Європи до інших континентів	Інтеграція з багатьма платформами для пошуку найбільш вигідних маршрутів	Широко використовується в Європі та за її межами для порівняння маршрутів	Простий інтерфейс, який дозволяє порівнювати маршрути та ціни з кількох платформ

Рисунок 1.1 - Візуальна ідентифікація сучасних платформ

TripIt — це передовий додаток для організації подорожей, який перетворює хаос із квитками, підтвердженнями бронювань та іншими важливими документами на чітко структуровану і зрозумілу картину подорожі. Завдяки інтеграції з електронною поштою, TripIt автоматично імпортує ваші бронювання, зберігає інформацію про рейси, готелі, оренду автомобілів та інші елементи подорожі, щоб надати вам єдиний зручний маршрут.

Основною особливістю є автоматичне створення маршруту — вам достатньо надіслати або перенести ваші підтвердження бронювання до додатка, і TripIt самостійно організує ваші поїздки по днях. Користувачі також можуть вручну додавати додаткову інформацію, таку як місця для відвідування, транспортні засоби чи важливі завдання, створюючи таким чином персоналізовану картину подорожі.

Ще однією важливою перевагою TripIt є інтеграція з календарем користувача, що дозволяє автоматично синхронізувати важливі події та нагадування. Це дає змогу заздалегідь отримувати сповіщення про зміни в маршруті або запізнення рейсів.

Для тих, хто часто стикається з проблемами з інтернет-з'єднанням під час подорожей, TripIt дозволяє завантажити маршрути в офлайн-режимі. Тепер вся необхідна інформація, від розкладів до підтверджень, завжди буде під рукою, навіть коли з’єднання з мережею не буде доступне.

Завдяки простому і інтуїтивно зрозумілому інтерфейсу, користувачам не потрібно витрачати багато часу на налаштування — додаток автоматично адаптується до їхніх потреб. Крім того, TripIt підтримує обмін маршрутами з іншими користувачами та дозволяє створювати спільні подорожі, що робить його ідеальним інструментом для групових поїздок.

Wanderlog — це додаток для планування подорожей, який об'єднує зручність організації маршрутів з інтерактивністю та адаптивністю. Додаток дає змогу

користувачам легко створювати детальні плани подорожей, комбінуючи різні види транспорту, бронювання готелів та рекомендації щодо пам’яток чи ресторанів. Wanderlog орієнтований на мандрівників, які хочуть не лише планувати свій маршрут, а й знаходити нові цікаві місця для відвідування, оптимізуючи час і витрати.

Особливістю Wanderlog є його можливість інтеграції з Google Maps, що дає змогу будувати маршрути з точністю до кроку. Ця інтеграція дозволяє користувачам безпосередньо вибирати точки на карті, додавати їх до свого маршруту і автоматично оновлювати відстані та час в дорозі. Додаток підтримує можливість додавання безлічі зупинок на шляху, що робить його ідеальним вибором для тих, хто планує тривалі подорожі, зокрема автомобільні поїздки по різних країнах.

Однією з головних переваг Wanderlog є його здатність поєднувати планування з бронюваннями. Користувачі можуть не лише створювати маршрути, а й знаходити й бронювати готелі, що значно спрощує процес планування подорожей. Зручний інтерфейс дозволяє легко додавати деталі про транспорт, готелі та атракціони, організовуючи все в одному місці.

Wanderlog також пропонує можливість спільного використання маршрутів з іншими користувачами, що робить його незамінним для групових подорожей. Користувачі можуть створювати спільні плани подорожей, де кожен може додавати свої пропозиції або коментарі щодо різних аспектів поїздки.

Ще однією важливою функцією є автоматичні оновлення маршруту при зміні часу рейсів або зміні умов подорожі. Це робить Wanderlog надзвичайно зручним інструментом для мандрівників, які хочуть мати актуальну інформацію щодо свого маршруту в реальному часі.

Завдяки простому й зрозумілому інтерфейсу, Wanderlog дозволяє створювати поїздки, що виглядають не лише професійно організовано, а й привабливо для кожного користувача, хто прагне зручно і швидко спланувати свою подорож з

можливістю інтеграції найрізноманітніших послуг.

Roadtrippers – це додаток для планування автомобільних подорожей, який орієнтований на тих, хто любить мандрувати на власному транспорті. Його особливість полягає в детальному плануванні маршрутів із можливістю додавання зупинок, а також рекомендаціях щодо цікавих місць, готелів, ресторанів та природних пам’яток вздовж вашого шляху. Roadtrippers надає користувачам всю необхідну інформацію для того, щоб подорож була не лише комфортною, а й наповненою яскравими враженнями та відкриттями.

Одна з головних переваг додатка — можливість вибору маршруту по США та інших країнах. За допомогою Roadtrippers можна прокладати маршрути з точністю до кожної зупинки, вказуючи бажані напрямки, види транспорту, типи зупинок (наприклад, музеї, ресторани, національні парки). Додаток має вбудовану карту, яка дозволяє переглядати і планувати поїздки в реальному часі, з автоматичними оновленнями інформації.

Roadtrippers дозволяє додавати більше 20 000 цікавих точок на карті, що дає можливість перетворити звичайну поїздку на захоплюючу пригоду. Це може бути відвідування місцевих визначних пам'яток, природних ландшафтів, туристичних місць або навіть цікавинок, які не відомі більшості мандрівників. Також додаток допомагає знайти ресторани, місця для ночівлі та заправки на вашому шляху.

Важливою особливістю є те, що користувачі можуть зберігати свої маршрути та обирати між різними видами транспорту, щоб зберегти час і гроші. Додаток пропонує оптимізовані варіанти поїздок, враховуючи відстань, час в дорозі та доступність заправок або інших зупинок.

Roadtrippers також підтримує спільне використання маршрутів. Це дозволяє друзям або родині разом планувати поїздки, ділитися рекомендаціями та організовувати поїздки, включаючи зупинки та пам’ятки, які можуть бути цікавими

для кожного члена групи. Спільне використання дозволяє зробити процес планування колективним і зручним.

Це інноваційний додаток, що дозволяє з легкістю знаходити маршрути між різними містами та країнами, порівнюючи ціни та час подорожей через різні види транспорту. Rome2Rio використовує потужний алгоритм, що забезпечує точність і зручність при плануванні поїздок, комбінуючи авіарейси, поїзди, автобуси, пороми, таксі, автомобілі та навіть пішохідні маршрути. Це робить його незамінним інструментом для тих, хто хоче знайти найшвидший або найдешевший спосіб подорожувати з пункту А в пункт Б, незалежно від типу транспорту.

Одна з головних переваг Rome2Rio — це його здатність порівнювати варіанти маршруту з усіх доступних видів транспорту, пропонуючи користувачам різноманітні рішення для поїздок. Додаток збирає дані з численних платформ і сервісів для того, щоб забезпечити найбільш актуальну інформацію по цінах, часі в дорозі та наявності квитків. Це дозволяє знайти найбільш вигідні варіанти для подорожей по всьому світу.

Rome2Rio також включає функцію відображення часу в дорозі для кожного варіанту транспорту, що дозволяє порівняти не тільки вартість, а й ефективність маршруту. Користувачі можуть вибирати, чи хочуть вони зекономити час, гроші або отримати найкомфортніший спосіб подорожувати. Це дає можливість для більш точного планування з урахуванням індивідуальних потреб.

Інтерфейс Rome2Rio продуманий і зручний, що дозволяє легко вводити пункти відправлення і призначення, а також переглядати всі доступні варіанти з детальним описом кожного маршруту. Крім того, додаток дозволяє користувачам зберігати маршрути, порівнювати їх та здійснювати бронювання безпосередньо через платформу, що робить процес планування подорожей ще зручнішим.

Завдяки своїй простоті та потужним функціям, Rome2Rio стає ідеальним вибором для мандрівників, які шукають універсальний інструмент для планування

поїздок по всьому світу, де б вони не знаходились. Цей додаток дозволяє не тільки заощадити час і кошти, а й зробити процес подорожі більш комфортним і організованим.

Висновки до розділу 1

Сучасний туризм неможливо уявити без цифрових технологій, які кардинально змінили всі аспекти подорожей - від пошуку необхідної інформації до повсякденної навігації. Зібравши основні функції та переваги кожного з порівняних мною сервісів, я планую інтегрувати їх у свій власний додаток, комбінуючи найкраще з кожного застосунку , щоб створити продукт, який буде максимально зручним і корисним для користувачів. Вибір оптимального інтерфейсу, а також можливість додавання особливих функцій допоможе зробити додаток не лише функціональним, але й привабливим для широкої аудиторії. Цей процес дозволить отримати найбільш універсальне та ефективне рішення для мандрівників.

2 ТЕХНОЛОГІЇ ТА ІНСТРУМЕНТИ РОЗРОБКИ ПОМІЧНИКА ДЛЯ ПОДОРОЖЕЙ У ФОРМАТІ ТЕЛЕГРАМ-БОТА

2.1 Розвиток картографічних сервісів у цифрову епоху, та їх порівняння

Сучасні картографічні сервіси перетворилися на потужні інструменти, які виходять далеко за межі звичайної навігації. Вибір оптимального рішення залежить від конкретних потреб – чи це розробка власного застосунку, потреб бізнесу в логістиці чи просто особистого використання для подорожей. Кожна з провідних платформ – Google Maps, Mapbox, OpenStreetMap, HERE Maps та TomTom Maps – пропонує унікальний набір функцій, які варто ретельно аналізувати перед вибором.

Головні відмінності між цими сервісами полягають не лише у технічній базі, а й у підходах до ліцензування, гнучкості налаштувань та спеціалізації. Наприклад, Google Maps лідирує завдяки неймовірній деталізації та інтеграції з іншими сервісами Google, проте має суттєві обмеження для розробників. Mapbox, навпаки, надає розробникам необмежені можливості кастомізації, але вимагає технічних навичок. OpenStreetMap – це унікальний відкритий проект, де кожен може редагувати карти, що робить його ідеальним для некомерційного використання.

Для корпоративних клієнтів особливий інтерес представляють HERE Maps з їхніми HD-картами для логістики та TomTom із передовими алгоритмами аналізу трафіку. Ці платформи орієнтовані на професійне використання та пропонують рішення, які неможливо знайти в масових сервісах. Варто також звертати увагу на цінову політику – від безкоштовних рішень на базі OpenStreetMap до дорогих корпоративних пакетів HERE чи TomTom.

Наступна таблиця детально порівнює ці платформи за ключовими параметрами, щоб допомогти вам зробити обґрунтований вибір. Тут враховано як технічні можливості, так і практичні аспекти використання, що особливо важливо для

розробників, бізнесу та активних мандрівників. Аналізуючи ці дані, варто зважати не лише на поточні потреби, а й на перспективи масштабування та інтеграції з іншими сервісами.

Таблиця 2.1 - Технічні параметри та унікальні можливості картографічних рішень

Додаток	Основні функції	Основні переваги	Гнучкість у налаштуванні	Особливі "фішки"	Ціноутворення	Інтерфейс
Google Maps	Карти, геокодування, маршрути, трафік, Street View	Висока точність даних, глобальне покриття, інтеграція з іншими сервісами Google	Обмежена; використання даних лише в межах Google Maps	Street View, детальні дані про трафік, Places API	Модель "оплата за використання"; безкоштовно до 10 тис. запитів на місяць, далі від $2 до $32 за 1000 запитів залежно від API	Інтуїтивно зрозумілий, добре документований
Mapbox	Кастомізовані карти, маршрути, геокодування, 3D-візуалізація	Висока гнучкість, можливість повної кастомізації, підтримка офлайн-режиму	Висока; підтримка власних стилів, інтеграція з різними платформами	3D-карти, векторні тайли, можливість використання власних даних	Безкоштовно до 50 тис. завантажень карт на місяць; далі від $5 за 1000 завантажень	Сучасний, зручний для розробників

Кінець таблиці 2.1

OpenStreetMap	Базові карти, геокодування, маршрути	Безкоштовний, відкритий код, можливість редагування даних спільнотою	Висока; повна свобода у використанні та модифікації даних	Спільнотне оновлення даних, відсутність обмежень на використання	Безкоштовно; можливе використання сторонніх сервісів для додаткових функцій	Залежить від обраного інтерфейсу
HERE Maps	Карти, маршрути, трафік, POI, геокодування	Висока точність даних, особливо для Європи, гнучка система ліцензування	Висока; можливість інтеграції з різними платформами та сервісами	Indoor-навігація, розширені дані про трафік, підтримка автономного режимум	Безкоштовно до 250 тис. транзакцій на місяць; далі модель "оплата за використання"	Професійний, орієнтований на бізнес
TomTom Maps	Карти, маршрути, трафік, POI, геокодування	Надійні дані про трафік, зручна інтеграція, гнучка система цін	Висока; підтримка кастомізації та інтеграції з різними платформами	Реальний час оновлення трафіку, розширені можливості для логістики та доставки	Безкоштовно до 50 тис. запитів на день; далі модель "оплата за використання"	Зручний, орієнтований на розробників

Джерело: побудоване на основі [1-5]

У таблиці 2.1 проведено аналіз основних картографічних сервісів який демонструє, що кожен із розглянутих інструментів має свою чітку спеціалізацію та цільову аудиторію. При виборі оптимального рішення варто враховувати не лише технічні характеристики, а й економічну доцільність використання, особливо для довгострокових проектів. Google Maps, будучи найпопулярнішим рішенням, пропонує найширший функціонал для звичайних користувачів, проте їхня політика

ліцензування може бути надто обмежувальною для комерційних продуктів. Mapbox у цьому плані виглядає привабливіше для розробників, оскільки дозволяє створювати повністю кастомізовані рішення з урахуванням специфіки бізнесу. Варто відзначити, що останні роки спостерігається значний прогрес у якості даних OpenStreetMap, які вже можуть конкурувати з комерційними аналогами в багатьох регіонах, особливо для базових завдань навігації. Професійні рішення на кшталт HERE Maps і TomTom залишаються незамінними для галузевих застосувань, де потрібна максимальна точність даних про трафік та дорожню інфраструктуру. Особливу увагу при виборі слід приділяти підтримці офлайн-режиму, яка може бути критично важливою для мобільних додатків, призначених для подорожей у віддалені райони. Не менш важливим фактором є простота інтеграції з іншими сервісами та платформами, що може значно скоротити час розробки. Для великих підприємств варто розглядати можливість використання гібридних рішень, коли різні компоненти системи можуть працювати з різними картографічними сервісами в залежності від конкретних потреб. Майбутнє галузі, ймовірно, буде пов'язане з подальшим розвитком таких як прогнозування трафіку та автоматизоване оновлення картографічних даних, що може змінити ринкові позиції окремих гравців.

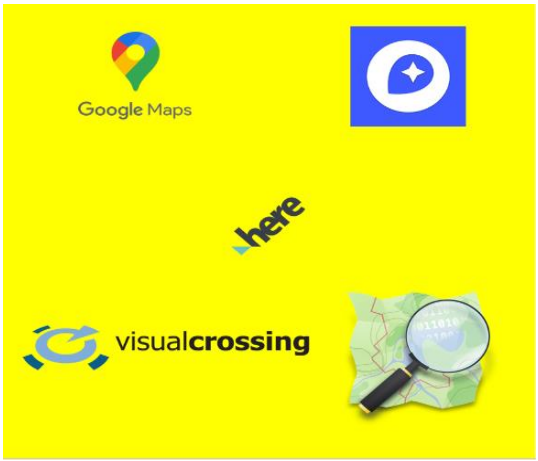

Рисунок 2.1 - Візуальна ідентифікація сучасних картографічних платформ

Картографічні сервіси давно перестали бути просто інструментами навігації – вони перетворилися на потужні цифрові екосистеми, що інтегрують безліч сучасних технологій. Логотипи цих компаній, які ви бачите на ілюстрації, стали справжніми символами цифрової епохи, впізнаваними мільйонами користувачів по всьому світу. Кожен з цих брендів має свою унікальну історію та філософію: від демократичного підходу OpenStreetMap до високотехнологічних рішень Google Maps, від розробницької спрямованості Mapbox до професійної орієнтації HERE Maps та TomTom. Варто звернути увагу, як дизайн логотипів відображає спеціалізацію кожного сервісу – від мінімалістичного стилю OpenStreetMap, що підкреслює його відкритість, до динамічних елементів у лого TomTom, які асоціюються з рухом і трафіком. Ці візуальні образи не просто ідентифікують продукти – вони передають цінності компаній, їхній підхід до картографування та взаємодії з користувачами. Інтересно, що за останні роки дизайн більшості логотипів став більш плоским і мінімалістичним, що відповідає загальним тенденціям цифрового дизайну. При цьому кожен бренд зберіг свою унікальність – кольорову палітру, характерні форми чи типографіку, що дозволяє користувачам миттєво ідентифікувати потрібний сервіс. Ці логотипи стали невід'ємною частиною нашого цифрового досвіду, з'являючись у мобільних додатках, вебсервісах та навіційних системах. Вони є візуальним містком між складними технологіями, що стоять за картографічними сервісами, і звичайними користувачами, які щодня покладаються на ці інструменти у своїх подорожах та повсякденних справах.

Google Maps — це один із найпопулярніших картографічних сервісів у світі, який пропонує широкий набір функцій для користувачів. Він забезпечує точні карти, покриття яких охоплює майже всі куточки планети, включаючи великі міста, маленькі селища та навіть віддалені території. Основними функціями додатку є побудова маршрутів, відстеження трафіку в реальному часі, підтримка громадського

транспорту, пішохідна навігація, перегляд супутникових знімків і можливість віртуальних прогулянок за допомогою Street View.

Крім цього, Google Maps надає користувачам персоналізовані рекомендації щодо ресторанів, готелів, туристичних визначних місць та інших місць, які можуть вас зацікавити під час подорожей. Завдяки інтеграції зі штучним інтелектом додаток пропонує оптимальні маршрути з урахуванням заторів, аварій і погодних умов.

Google Maps також підтримує можливість офлайн-доступу, що дозволяє завантажувати карти обраних регіонів і користуватися ними без підключення до інтернету. Інші корисні функції включають обмін геолокацією, створення маршрутів з кількома зупинками та можливість дізнатися про зайнятість місць у реальному часі.

Mapbox — це потужний інструмент для розробників, які хочуть створювати власні цифрові карти з високим рівнем кастомізації. Він пропонує широкий набір функцій для інтерактивних картографічних рішень, включаючи 2D і 3D-візуалізацію, маршрутизацію, геокодування та аналіз просторових даних. Mapbox відомий своєю гнучкістю, що дозволяє створювати карти з унікальним дизайном, інтегрувати власні дані та застосовувати кастомні стилі для створення карт будь-якої складності.

Основні переваги Mapbox включають підтримку офлайн-карт, можливість роботи з векторними тайлами для швидкої візуалізації великих обсягів даних, інтеграцію з популярними фреймворками, такими як React і Unity, а також доступ до детальних геопросторових даних.

Серед особливих "фішок" Mapbox — підтримка тривимірних карт, візуалізація потоків руху, анімація маршрутів, а також можливість налаштовувати інтерфейс для відображення даних в реальному часі. Цей сервіс активно використовується у мобільних додатках, веб-сайтах, логістичних рішеннях і навіть у сфері віртуальної реальності.

OpenStreetMap (OSM) — це глобальна, відкрита картографічна платформа, яка

створюється і підтримується спільнотою волонтерів з усього світу. На відміну від комерційних сервісів, OSM надає повну свободу у використанні та модифікації своїх даних, що робить його ідеальним вибором для розробників додатків, дослідників та бізнесів, які цінують відкритість і незалежність.

Основні функції OpenStreetMap включають створення і редагування карт, геокодування, побудову маршрутів, відображення POI (Points of Interest) і підтримку різноманітних форматів даних. Завдяки відкритому коду та гнучким можливостям кастомізації, OSM широко використовується для створення кастомних карт, інтеграції геопросторових даних у веб-додатки та аналізу географічної інформації.

Серед головних переваг OSM — безкоштовний доступ до високоякісних картографічних даних, можливість редагування та додавання власних даних, а також активна спільнота, яка постійно оновлює карти для забезпечення їх актуальності.

HERE Maps — це один з провідних картографічних сервісів, орієнтований на бізнес-рішення та професійне використання. Він пропонує детальні карти, маршрутизацію, геокодування, відстеження трафіку та багато інших функцій для створення додатків з високою точністю даних. HERE Maps відомий своєю надійністю та точністю, особливо у регіонах Європи, де він має одне з найдетальніших покриттів.

Основні функції HERE Maps включають побудову маршрутів для автомобілів, пішоходів і велосипедистів, відстеження громадського транспорту, управління логістикою, розширені дані про трафік у реальному часі та підтримку внутрішньої навігації (Indoor Positioning). Сервіс також пропонує можливість працювати в офлайн-режимі, що робить його ідеальним вибором для мобільних додатків і навігаційних систем.

Особливі "фішки" HERE Maps включають підтримку автономного режиму, розширену аналітику про рух транспортних засобів, точні дані про затори, а також можливість інтеграції з IoT-рішеннями для моніторингу та оптимізації логістичних

процесів. Крім того, платформа HERE Maps забезпечує високий рівень безпеки та конфіденційності даних, що робить її популярним вибором серед великих корпорацій та державних організацій.

TomTom Maps — це потужний картографічний сервіс, відомий своєю високою точністю навігаційних даних та надійними рішеннями для транспортної логістики. Цей сервіс використовується як у мобільних додатках для споживачів, так і в корпоративних рішеннях для керування автопарками, доставки товарів і моніторингу транспортних засобів у реальному часі.

Основні функції TomTom Maps включають побудову маршрутів, розрахунок відстаней і часу прибуття, моніторинг трафіку, підтримку пошуку POI (Points of Interest) і геокодування. Платформа також пропонує можливість працювати в офлайн-режимі, що робить її особливо корисною для водіїв і транспортних компаній, яким потрібні стабільні та точні навігаційні рішення навіть без доступу до інтернету.

Серед особливих "фішок" TomTom Maps — розширені дані про трафік у реальному часі, прогнозування дорожніх заторів, оптимізація маршрутів для вантажівок з урахуванням їх розмірів та ваги, а також інтеграція з великими корпоративними системами.

2.2 Вибір та порівняння погодних API

Машинне навчання та сучасні рекомендаційні системи стають невід'ємною частиною туристичної галузі, підвищуючи точність прогнозів і забезпечуючи персоналізований підхід до планування подорожей. Завдяки обробці великих обсягів геопросторових даних і поведінкових характеристик користувачів, ці технології дозволяють не лише оптимізувати маршрути, але й пропонувати найкращі варіанти відпочинку, враховуючи індивідуальні потреби туристів. Вагомий внесок у цю сферу роблять метеорологічні сервіси, що надають точну інформацію про погоду та

кліматичні умови, необхідну для безпечного та комфортного планування подорожей.

У таблиці 2.2 представлено аналіз найбільш популярних метеорологічних сервісів, що використовують передові технології машинного навчання для забезпечення високої точності прогнозів, гнучкості налаштувань та зручного доступу до даних.

Таблиця 2.2 – Аналіз популярних метеорологічних сервісів

Сервіс	Основні функції	Основні переваги	Гнучкість налаштувань	Особливі "фішки"	Ціноутворення	Інтерфейс
Tomorrow.io	Поточна погода, прогнози, історичні дані, 80+ параметрів	Висока точність, гіперлокальні прогнози, адаптація до бізнес-потреб	API з 80+ шарами даних, підтримка Webhooks, кастомні алерти	Інтеграція з IoT, прогнозування ризиків для логістики та енергетики	Безкоштовний план з обмеженнями; платні плани від $0.002 за запит	Інтуїтивний, зручна документація, RESTful API
OpenWeatherMap	Поточна погода, 5-денний прогноз, історичні дані, карти, алерти	Широке покриття, стабільна робота, багатий набір API	Підтримка JSON/XML, 60 запитів/хв на безкоштовному плані, до 100 тис. запитів/хв на платних планах	API One Call 3.0, карти погоди, дані про забруднення повітря	Безкоштовний план до 1 млн запитів/міс; платні плани від $40/міс	Простий REST API, інтеграція з OpenStreetMap

Кінець таблиці 2.2

Visual Crossing	Історичні дані, поточна погода, прогнози, кліматичні статистики	Єдиний API-запит для всіх типів даних, висока точність	Підтримка масових запитів, налаштування форматів виводу	API з єдиним викликом для різних типів даних, зручна інтеграція	Безкоштовний план; платні плани з гнучким ціноутворенням	Простий у використанні, детальна документація
Weatherbit	Поточна погода, прогнози, історичні дані, 50+ параметрів	Дані з 50 тис. станцій, підтримка аграрного сектору, швидке оновлення даних	Підтримка JSON, різні рівні деталізації, можливість вибору параметрів	API для аграрного бізнесу, підтримка багатьох мов	Безкоштовний план з обмеженнями; платні плани з гнучким ціноутворенням	Зручний REST API, хороша документація
Meteomatics	Поточна погода, прогнози, історичні дані, 1800+ параметрів	Надвисока точність, підтримка критичних застосувань, гнучке масштабування	Підтримка різних форматів (JSON, CSV), можливість вибору частоти оновлення	1800+ параметрів, підтримка різних висот, інтеграція з енергетичними системами	Безкоштовна 2-тижнева пробна версія; платні плани за запитом	Потужний API, детальна документація, підтримка різних мов програмування

Джерело: побудоване на основі [14-18]

Метеорологічні сервіси, такі як Tomorrow.io, OpenWeatherMap, Visual Crossing,

Weatherbit та Meteomatics, демонструють різноманітність можливостей та підходів до роботи з даними. Вони забезпечують користувачів не лише поточною інформацією про погоду, але й історичними даними, прогнозами, а також спеціалізованими функціями для логістики, аграрного сектору та енергетики. Інтеграція таких сервісів з туристичними платформами підвищує точність планування маршрутів, дозволяє уникати несприятливих погодних умов та ефективніше керувати ресурсами під час подорожей. Таким чином, поєднання сучасних алгоритмів машинного навчання та високоточного прогнозування погоди дозволяє значно підвищити рівень сервісу в туристичній галузі, мінімізувати ризики та забезпечити більш комфортний і безпечний досвід для мандрівників. Інновації в цій сфері відкривають нові можливості для створення інтелектуальних туристичних маршрутів, що адаптуються в реальному часі залежно від зміни умов, що є важливою складовою сучасного підходу до подорожей.

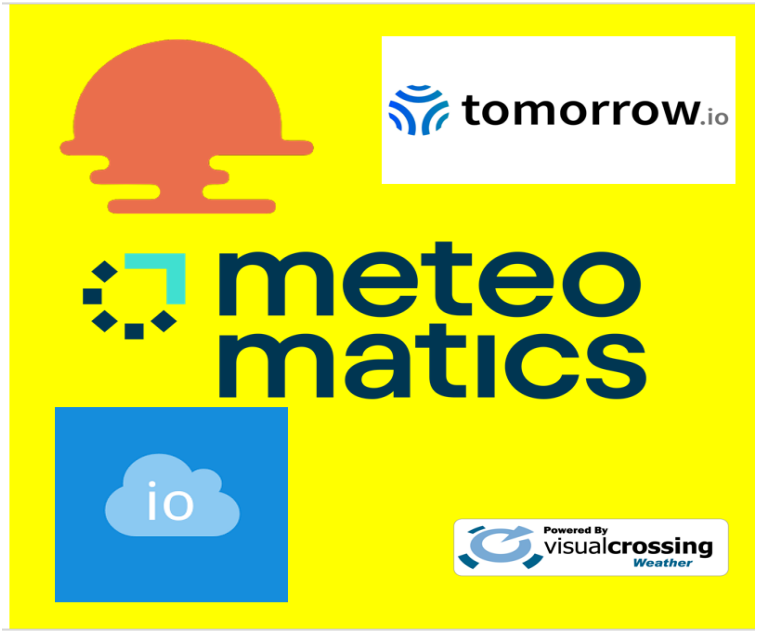

Рисунок 2.3 - Логотипи провідних метеорологічних сервісів для туристичної галузі

Джерело: побудоване на основі [10-13]

Tomorrow.io — це сучасний погодний сервіс, створений для точного прогнозування погоди з використанням штучного інтелекту та аналізу великих даних. Платформа пропонує гіперлокальні прогнози, які враховують понад 80 різних параметрів, включаючи температуру, вологість, швидкість вітру, рівень опадів та навіть ризики для логістики та енергетики. Tomorrow.io активно використовується в бізнесі для оптимізації операцій та зменшення погодних ризиків завдяки інтеграції з IoT і підтримці Webhooks. Сервіс пропонує гнучкі налаштування API, що дозволяють адаптувати прогноз під конкретні потреби бізнесу, включаючи кастомні алерти та інтеграцію з іншими аналітичними інструментами.

OpenWeatherMap — це один з найпопулярніших сервісів для прогнозування погоди, який надає доступ до поточних погодних умов, короткострокових та довгострокових прогнозів, історичних даних, карт погоди та алертів. Платформа підтримує широкий спектр API, включаючи One Call API 3.0, який забезпечує швидкий доступ до всіх необхідних даних за один запит. OpenWeatherMap відомий своєю високою стабільністю, великим охопленням регіонів та широкими можливостями налаштування, що робить його ідеальним вибором як для мобільних додатків, так і для корпоративних систем. Сервіс також підтримує інтеграцію з OpenStreetMap, що дозволяє створювати інтерактивні карти з погодними даними.

Visual Crossing — це потужний погодний сервіс, що забезпечує доступ до історичних, поточних і прогнозованих погодних даних у глобальному масштабі. Основною перевагою цього сервісу є можливість отримувати всі необхідні дані за одним API-запитом, що значно спрощує інтеграцію та зменшує затримки в обробці інформації. Visual Crossing пропонує високоточні прогнози, дані про кліматичні умови, статистику екстремальних погодних явищ і аналітику, що дозволяє

використовувати його як для персональних проектів, так і для великих корпоративних рішень. Додатково сервіс підтримує масові запити та налаштування форматів виводу, що робить його ідеальним вибором для аналітиків та дослідників.

Weatherbit — це професійний сервіс для отримання точних погодних даних, що забезпечує доступ до поточних умов, короткострокових прогнозів, історичних даних та кліматичної статистики. Завдяки використанню даних з понад 50 тисяч погодних станцій по всьому світу, Weatherbit пропонує високу точність та швидке оновлення інформації, що робить його особливо корисним для аграрного сектору, логістики та енергетики. Платформа підтримує широкий набір параметрів, включаючи рівень вологості, тиск, УФ-індекс, рівень забруднення повітря та навіть дані про вплив погоди на здоров'я. Weatherbit також пропонує гнучкі налаштування API, що дозволяють обирати рівень деталізації та типи даних для кожного запиту.

Meteomatics — це високотехнологічний погодний сервіс, що забезпечує доступ до надточних даних про погоду з підтримкою понад 1800 параметрів. Платформа спеціалізується на високочастотних оновленнях даних та підтримці критично важливих застосувань, таких як енергетика, авіація, логістика та сільське господарство. Завдяки можливості налаштування частоти оновлень, вибору висот та інтеграції з різними форматами даних (JSON, CSV, NetCDF) тощо.

2.3 Аналіз MongoDB для туристичного Telegram-бота: переваги над SQL

У світі сучасних програмних рішень вибір бази даних є одним з найважливіших рішень, що визначає продуктивність, масштабованість та гнучкість додатку. Для мого туристичного Telegram-бота, який обробляє великі обсяги даних, я вирішив використати MongoDB, хоча класичні реляційні бази даних, такі як SQL, також мали свої переваги. Проте після ретельного аналізу я зрозумів, що MongoDB значно краще підходить для специфіки мого проєкту, і ось чому.

Основна відмінність між MongoDB і SQL полягає в їхніх підходах до зберігання та організації даних. SQL, як правило, використовує реляційну модель, де дані зберігаються в таблицях із чітко визначеними рядками та стовпцями. Це створює структуру, яка забезпечує високу консистентність і цілісність даних, але часто стає обмеженням, коли мова йде про динамічні структури даних або великі обсяги неструктурованої інформації. Наприклад, для туристичного бота, що зберігає інформацію про користувачів, маршрути, рекомендації та навіть історію запитів, така жорстка схема може стати серйозним викликом.

MongoDB, навпаки, використовує документо-орієнтовану модель, де дані зберігаються у форматі BSON (бінарний аналог JSON). Це означає, що кожен документ може мати власну унікальну структуру, що робить цю базу даних ідеальною для проєктів, де дані постійно змінюються або мають складні вкладені структури. Для мого бота, який може зберігати різні типи даних, включаючи текстові повідомлення, зображення, координати та навіть результати пошуку, MongoDB стала ідеальним рішенням. Наприклад, якщо користувач надсилає фотографію або ділиться своїм поточним розташуванням, я можу легко зберегти ці дані в одному документі разом з усіма метаданими, не переймаючись суворою структурою таблиць.

Крім того, MongoDB пропонує горизонтальне масштабування, що є критично важливим для великих проєктів. На відміну від SQL, де масштабування часто вимагає складних налаштувань реплікації або шардінгу, MongoDB дозволяє легко додавати нові вузли до кластера, розподіляючи дані між ними без значних змін у коді. Це важливо для мого бота, оскільки його популярність може різко зрости під час туристичного сезону або великих свят, і система повинна бути готовою обробляти збільшені навантаження без затримок.

Ще однією значною перевагою MongoDB є її гнучка модель запитів. У SQL для кожного нового типу даних потрібно створювати окремі таблиці або модифікувати

вже існуючі, що може бути доволі трудомістким процесом. У MongoDB ж я можу легко додавати нові поля або вкладені документи без необхідності змінювати структуру бази. Це дозволяє швидко адаптуватися до змін вимог користувачів або додавати нові функції, такі як рекомендації, описи місць або навіть інтеграція з зовнішніми API для отримання погодних даних у реальному часі.

Окрім цього, MongoDB пропонує потужні інструменти для роботи з великими обсягами даних, такі як агрегаційні фреймворки, індекси для швидкого пошуку та механізми для геопросторового аналізу, що є критично важливим для мого туристичного бота. Наприклад, можливість використовувати геоіндекси дозволяє швидко знаходити найближчі готелі, ресторани або туристичні об'єкти на основі координат користувача. Це значно покращує користувацький досвід, оскільки дозволяє пропонувати максимально релевантні результати без значних затримок.

Незважаючи на всі ці переваги, SQL бази даних, такі як PostgreSQL або MySQL, залишаються чудовим вибором для проєктів, де критично важлива транзакційна цілісність даних і складні реляційні зв'язки між таблицями. Проте, для мого проєкту, де дані більш динамічні, часто неструктуровані та вимагають високої гнучкості, MongoDB стала кращим вибором.

Загалом, вибір MongoDB для мого туристичного Telegram-бота був продиктований необхідністю забезпечити високу швидкість роботи, та гнучкість.

Висновки до розділу 2

У цьому розділі було проаналізовано й обґрунтовано вибір основних технологій для створення Telegram-бота-помічника у сфері туризму. Визначено, що для інтеграції картографічних і погодних сервісів доцільно використовувати Google Maps API та OpenWeatherMap API, які забезпечують необхідну точність і зручність підключення. Для зберігання та обробки динамічних користувацьких даних оптимальним рішенням

стала база даних MongoDB. Обраний стек технологій дозволяє створити гнучкий, масштабований і функціональний програмний продукт.

3 ПОБУДОВА УНІКАЛЬНИХ АСПЕКТІВ ТА ПЕРСПЕКТИВИ TELEGRAM-БОТА ДЛЯ ПОДОРОЖЕЙ

3.1 Аналіз тенденцій та перспеспективи розвитку тревел-застосунків

У сучасному світі подорожі стали невід'ємною частиною життя мільйонів людей, а технології відіграють ключову роль у їх плануванні, організації та проведенні. Веб-застосунки для подорожей, які колись були простими інструментами для перегляду карт чи бронювання квитків, сьогодні перетворилися на складні інформаційні системи, що здатні прогнозувати потреби користувачів, враховувати їхні інтереси та навіть формувати індивідуальні рекомендації на основі штучного інтелекту. Такі додатки стали центральними елементами туристичної інфраструктури, і їх роль продовжує зростати з розвитком цифрових технологій.

На початку ери цифрових подорожей ключовими інструментами були прості веб-сайти з інформацією про готелі, маршрути та туристичні пам'ятки. Наприклад, у 90-х роках з'явилися такі сервіси, як Expedia та Booking.com, які спростили процес бронювання авіаквитків і готелів, зробивши його доступним для широкої аудиторії. Expedia, наприклад, стала першою компанією, яка запропонувала користувачам можливість самостійно бронювати авіарейси онлайн, що було революційним кроком для індустрії. Згодом, із розвитком мобільних технологій, з'явилися додатки, що дозволяли мандрівникам легко знаходити інформацію на ходу, користуючись смартфонами. Це стало справжнім проривом, оскільки мобільні додатки відкрили нові можливості для персоналізації сервісів, створивши принципово новий підхід до планування подорожей.

Однією з ключових тенденцій стало використання великих даних для покращення користувацького досвіду. Сучасні веб-застосунки, такі як TripAdvisor, Airbnb і Hopper, аналізують величезні обсяги інформації, враховуючи не лише

геолокацію користувача, але й його попередні вподобання, історію пошуку, відгуки та навіть сезонність подорожей. Наприклад, платформа Airbnb використовує алгоритми машинного навчання для аналізу попиту на оренду житла у різних містах, що дозволяє їй оптимізувати ціни та пропонувати найбільш релевантні варіанти розміщення для своїх користувачів. Це дозволяє створювати більш точні рекомендації, пропонувати найкращі маршрути та знижки, що значно підвищує лояльність клієнтів. Крім того, великі дані допомагають передбачати попит на конкретні напрямки, оптимізувати ціни та ефективно керувати ресурсами, що є критично важливим для великих туристичних компаній.

Крім того, все більшу популярність набувають додатки, які підтримують технології доповненої реальності (AR) та віртуальної реальності (VR). Вони дозволяють мандрівникам занурюватися у віртуальні екскурсії, досліджувати місця ще до їх відвідування та навіть отримувати інформацію про оточення в реальному часі через камеру смартфона. Наприклад, Google Lens використовує машинне навчання для розпізнавання об'єктів і надання миттєвої інформації про них, що значно розширює можливості туристів під час подорожей. Ці технології вже зараз активно використовуються в музеях, на історичних пам'ятках та у туристичних зонах, таких як Лувр у Парижі або музей Гуггенхайма в Нью-Йорку, роблячи досвід подорожей ще більш захопливим та інтерактивним.

Ще однією важливою тенденцією є інтеграція штучного інтелекту та чат-ботів у туристичні сервіси. Наприклад, платформа Kayak активно використовує чат-ботів для надання рекомендацій щодо готелів, рейсів та оренди автомобілів на основі запитів користувачів. Завдяки технологіям обробки природної мови (NLP) сучасні чат-боти здатні розуміти контекст запитів, аналізувати емоції користувачів та пропонувати персоналізовані рішення. Такі рішення активно використовуються у великих туристичних платформах, таких як Booking.com, Expedia та Airbnb, що

дозволяє значно зменшити час відповіді на запити та підвищити рівень задоволеності клієнтів. Наприклад, Expedia використовує штучний інтелект для автоматизації процесів підтримки клієнтів, що дозволяє вирішувати більшість стандартних запитів без участі людини.

Окрім цього, важливим напрямком розвитку є покращення безпеки та конфіденційності даних користувачів. Оскільки веб-застосунки працюють з великою кількістю особистих даних, питання захисту інформації стає критично важливим. Використання технологій шифрування, багатофакторної автентифікації та захисту від кібератак є обов'язковою умовою для успішного функціонування таких сервісів. Наприклад, впровадження протоколів SSL/TLS та технологій blockchain дозволяє мінімізувати ризики втрати даних і підвищити довіру клієнтів до сервісу. Дослідження, проведене компанією Cisco, показало, що 80% користувачів віддають перевагу платформам з високим рівнем захисту даних.

У майбутньому, ймовірно, ми побачимо ще більшу інтеграцію туристичних застосунків з технологіями штучного інтелекту, хмарними обчисленнями та Інтернетом речей (IoT). Такі системи зможуть не лише пропонувати оптимальні маршрути, але й автоматично коригувати плани подорожі залежно від поточних обставин, таких як зміни погоди, затримки рейсів чи оновлення інформації про безпеку в країнах. Крім того, розвиток технологій 5G дозволить передавати великі обсяги даних у реальному часі, що відкриває нові горизонти для туристичних сервісів.

Отже, розвиток веб-застосунків для подорожей продовжує прискорюватися, відкриваючи нові можливості для мандрівників і створюючи нові виклики для розробників. Це складний і динамічний ринок, де успіх залежить від здатності адаптуватися до змін, впроваджувати нові технології та постійно покращувати користувацький досвід. Майбутнє цих застосунків виглядає надзвичайно перспективним, адже вони продовжують перетворювати спосіб, яким ми досліджуємо

світ і взаємодіємо з ним.

3.3 Опис вхідних даних та структури системи

Вхідними даними для роботи розробленого Telegram-бота є географічне місце розташування користувача, його запити на погоду, рекомендації щодо пам'яток, закладів громадського харчування та готелів. Telegram-бот взаємодіє з користувачем через чітко структуровані команди, які інтерпретуються сервером за допомогою бібліотеки Aiogram. Всі запити обробляються асинхронно, що дозволяє системі швидко реагувати на користувацькі звернення.

Система складається з декількох основних компонентів. Клієнтська частина реалізована в месенджері Telegram, що дозволяє легко інтегрувати додаток у повсякденне використання користувача. Серверна частина створена з використанням мови програмування Python і фреймворку Aiogram, що забезпечує високу продуктивність та простоту масштабування системи. Для зберігання інформації використовується NoSQL база даних MongoDB, яка дозволяє ефективно зберігати історію взаємодій, вподобання користувачів та їхні персональні дані.

Для отримання інформації про погоду інтегровані зовнішні API сервіси, такі як OpenWeatherMap, що гарантує надання актуальної та точної погодної інформації. Карти і навігаційні запити обробляються за допомогою Google Maps API, що забезпечує високу точність географічних даних і додаткові функції, такі як маршрутна навігація.

Система має чітку і зрозумілу структуру, яка дозволяє ефективно обробляти запити користувачів, інтегрувати нові сервіси та підтримувати високу продуктивність роботи.

3.4 Оптимальна реєстрація Telegram-бота

Реєстрація власного Telegram-бота є першим і ключовим кроком у створенні повноцінного програмного рішення для автоматизації задач, інтеграції зі сторонніми сервісами та реалізації складних сценаріїв взаємодії з користувачами. Цей процес починається з використання спеціального адміністративного бота Telegram під назвою BotFather, який є єдиним офіційним інструментом для створення та керування ботами на цій платформі. Важливо зазначити, що справжній BotFather повинен мати синю галочку поруч зі своїм ім'ям, що підтверджує його автентичність і захищає мене від можливих фішингових атак або підроблених ботів.

Першим кроком для реєстрації мого бота є пошук самого BotFather. Для цього я відкриваю додаток Telegram і вводжу у пошуковому рядку "BotFather". Я бачу кілька результатів, але лише один з них має синю галочку біля свого імені – це і є офіційний акаунт для створення ботів. Натискаю на цей результат, відкриваю чат з BotFather і можу розпочати процес реєстрації свого майбутнього бота.

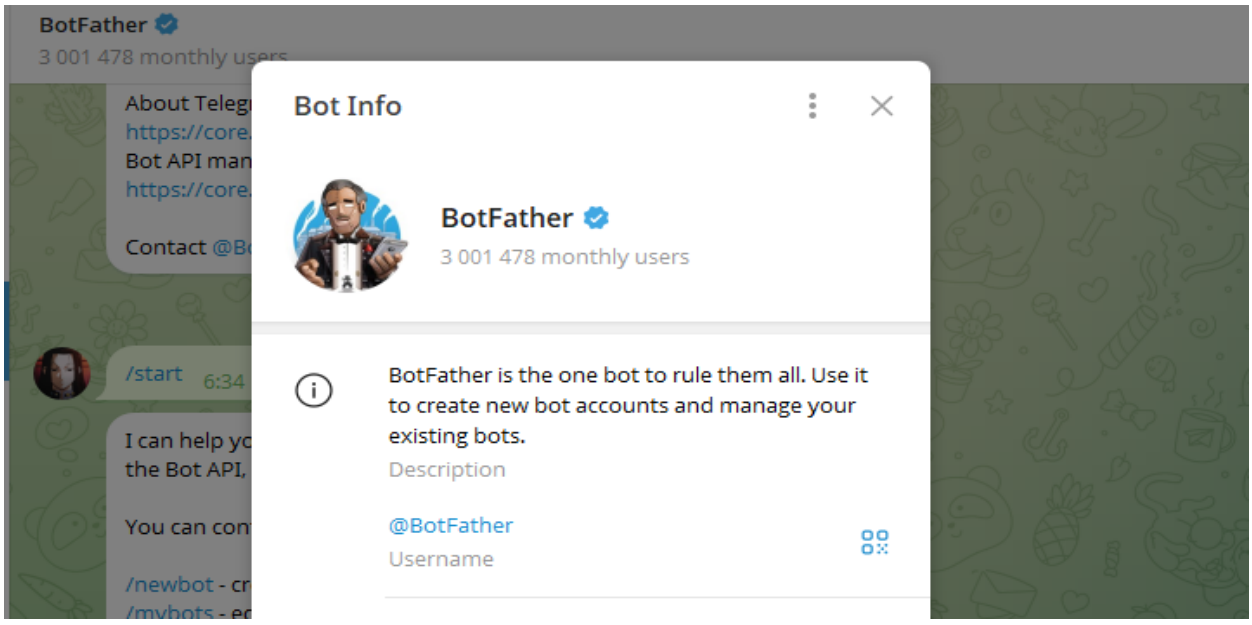

Рисунок 3.1 - Демонстрація реєстрації Telegram-бота

Після того, як я відкрив чат з BotFather, я натискаю кнопку "Start" або вводжу команду /start, щоб активувати його функціонал. BotFather вітає мене і надає список доступних команд, які можна використовувати для створення, налаштування та керування ботами. Для початку процесу реєстрації нового бота я вводжу команду /newbot. Ця команда ініціює створення нового Telegram-бота і запускає серію питань, які допоможуть налаштувати основні параметри мого бота.

Перше питання – вибір імені для мого бота. Це ім'я буде відображатися у списку контактів користувачів та використовуватися для пошуку мого бота в Telegram. Важливо вибрати унікальне, зрозуміле та запам'ятовуване ім'я, яке відображатиме суть мого проєкту. Наприклад, якщо мій бот призначений для допомоги мандрівникам, я можу обрати щось на зразок "TravelHelper" або "ExplorerBot". Після введення імені натискаю Enter, щоб перейти до наступного кроку.

Наступним етапом є вибір юзернейму для мого бота. Юзернейм повинен бути унікальним у межах всієї платформи Telegram і обов'язково закінчуватися словом "bot" або "Bot". Це важливий крок, оскільки саме юзернейм використовується для створення посилань на мого бота і для взаємодії з ним через API. Наприклад, мій бот може мати юзернейм типу "TravelHelperBot" або "ExplorerAssistantBot". Вводжу бажаний юзернейм і натискаю Enter для підтвердження.

Після цього BotFather створює мого бота і надає мені унікальний токен доступу – довгий рядок символів, який виглядає приблизно так:

1234567890:ABCdefGhIJKlmnOPQrSTuvWXyZ123456789

Цей токен є критично важливим елементом безпеки мого бота, оскільки дозволяє йому надсилати повідомлення, отримувати оновлення і взаємодіяти з іншими сервісами через Telegram API. Його потрібно зберігати в безпечному місці і нікому не розголошувати, оскільки втрата цього токену може призвести до компрометації мого бота і доступу до його даних сторонніми особами.

Після отримання токену я можу налаштувати додаткові параметри мого бота, такі як опис, аватар і команди, використовуючи інші команди BotFather, такі як /setdescription, /setuserpic та /setcommands. Це допоможе зробити мого бота більш привабливим для користувачів і забезпечити йому кращу впізнаваність у списку контактів.

Таким чином, реєстрація мого Telegram-бота через BotFather – це простий, але критично важливий процес, який відкриває безмежні можливості для створення потужних, інтерактивних і багатофункціональних ботів, готових змінити спосіб, яким ми спілкуємося, працюємо і подорожуємо.Розробка Telegram-бота – це складний, але надзвичайно захопливий процес, який вимагає глибокого розуміння багатьох технологій та інструментів. Фреймворк, що використовується для створення такого бота, визначає його функціональні можливості, швидкість реагування, стійкість до навантажень та гнучкість в адаптації до змінних вимог. У цьому проєкті я обрав комбінацію перевірених бібліотек і модулів, кожен з яких відіграє важливу роль в загальній архітектурі системи. Давайте розглянемо їх детальніше.

Фреймворки телеграм бота*:*

– *aiogram* – це високопродуктивний асинхронний фреймворк для створення Telegram-ботів на Python, який поєднує простоту використання та потужні можливості для гнучкого керування ботом. Він використовує асинхронні методи для обробки повідомлень, що дозволяє одночасно опрацьовувати велику кількість запитів без затримок. Aiogram підтримує основні методи Telegram Bot API, такі як надсилання повідомлень, обробка оновлень, керування користувачами та інтеграція з зовнішніми API;

– *aiogram.fsm* – модуль для керування станами. Однією з ключових особливостей Telegram-ботів є необхідність збереження поточного контексту спілкування з користувачем. Наприклад, під час оформлення замовлення або

реєстрації необхідно зберігати проміжні дані. Модуль fsm (finite state machine) дозволяє створити дерево станів, де кожен стан представляє конкретний етап взаємодії з користувачем. Це забезпечує більш логічну структуру коду та спрощує підтримку;

– *aiogram.types* – модуль для роботи з типами даних Telegram. Включає класи для обробки повідомлень, кнопок, інлайн-меню та інших структур даних, що використовуються в Telegram API. Це важливий елемент фреймворка, оскільки дозволяє гнучко керувати користувацькими інтерфейсами та обробкою вхідних даних;

– *aiogram.filters* – модуль для фільтрації повідомлень. Використовується для створення гнучких правил обробки вхідних повідомлень. Дозволяє налаштувати фільтри за типом повідомлень, ключовими словами, станами користувачів та іншими критеріями, що значно спрощує логіку обробки даних;

– *requests* – бібліотека для надсилання HTTP-запитів. Використовується для взаємодії із зовнішніми API, такими як OpenWeatherMap або PayPal. Підтримує всі основні методи HTTP (GET, POST, PUT, DELETE), що дозволяє гнучко працювати з різними веб-сервісами;

– *urllib.parse* – модуль для парсингу URL-адрес. Корисний для формування коректних рядків запитів, декодування параметрів та обробки складних URL;

– *sqlite3* – вбудована в Python база даних, яка використовується для зберігання інформації про користувачів, їхні дії та інші дані. SQLite – це легке рішення, ідеально підходить для невеликих проєктів та прототипів, де немає потреби в складних схемах бази даних;

– *paypalrestsdk* – бібліотека для інтеграції з PayPal API. Використовується для обробки платежів, створення рахунків та відстеження транзакцій. Підтримує основні методи платіжних операцій та забезпечує високий рівень безпеки;

– *pyowm* – клієнт для OpenWeatherMap API. Дозволяє отримувати дані про поточні погодні умови, прогнози та кліматичні тенденції. Може використовуватися для надання актуальної інформації про погоду в точці призначення мандрівника;

– *openai* – бібліотека для інтеграції з GPT API. Це ключовий елемент, що дозволяє реалізувати функції машинного навчання, такі як аналіз тексту, генерація відповідей та інтелектуальні діалоги. Використовується для створення більш природних та персоналізованих взаємодій з користувачами;

– *datetime* – модуль для роботи з датою та часом. Використовується для відстеження часу повідомлень, планування подій та роботи з часовими поясами;

– *asyncio* – модуль для асинхронних операцій. Дозволяє виконувати кілька задач одночасно, що значно підвищує продуктивність бота, особливо при обробці великої кількості повідомлень;

– *os* – модуль для роботи з файловою системою та змінними оточення. Використовується для керування файлами, читання конфігураційних даних та взаємодії з операційною системою;

– *json* – модуль для роботи з JSON-даними. Широко використовується для обміну даними із зовнішніми API та зберігання конфігурацій;

– *random* – модуль для генерації випадкових чисел. Використовується для створення унікальних ідентифікаторів, випадкових подій та інших елементів, що потребують випадковості;

– *string* – модуль для роботи зі строками. Корисний для обробки текстів, генерації токенів та аналізу повідомлень користувачів.

Кожен з цих модулів відіграє ключову роль у створенні функціонального та гнучкого Telegram-бота, здатного обробляти складні запити, взаємодіяти із зовнішніми сервісами та забезпечувати високу продуктивність. Грамотне використання цих бібліотек дозволяє створювати потужні та масштабовані рішення

для різних завдань, від простих чат-ботів до складних віртуальних асистентів.

3.5 Оптимізація часу користувача та підвищення зручності взаємодії з Telegram-ботом

У сучасному світі подорожі стали невід'ємною частиною нашого життя. Вони дозволяють відкривати нові горизонти, знайомитися з культурними традиціями інших народів і збагачувати наші знання про світ. Однак, подорожі також ставлять перед нами чимало викликів, серед яких планування маршрутів, бронювання житла, пошук розваг і отримання актуальної інформації про погоду. В умовах швидкого темпу життя та постійної зміни обставин ефективне управління часом стає критичним фактором успіху кожної подорожі. Саме тут на допомогу приходять сучасні технології, такі як мій Telegram-бот для подорожей, який був розроблений для того, щоб спростити цей процес і зробити його максимально зручним для користувача.

Правильне розподілення часу під час подорожей має вирішальне значення. Дослідження, проведені Університетом Південної Каліфорнії, показали, що якісне планування подорожі може знизити рівень стресу на 30% і підвищити загальне задоволення від подорожі на 50%. Це пов'язано з тим, що мандрівники, які мають чіткий план, рідше стикаються з непередбачуваними ситуаціями і можуть більше часу присвятити відпочинку та насолоді новими враженнями. Проте створення такого плану вимагає великих зусиль і часто включає одночасне використання кількох різних сервісів: від картографічних додатків і прогнозів погоди до гідів по ресторанах і перекладачів. Це може створити плутанину та відволікати від основної мети подорожі – отримання нових вражень.

Саме тому я вирішив створити багатофункціональний Telegram-бот, який об'єднує всі ці можливості в одному зручному інтерфейсі. Мій бот пропонує широкий спектр функцій, які дозволяють не лише планувати маршрут, але й отримувати

актуальну інформацію про погоду, знаходити цікаві місця, дізнаватися про поточні заходи та навіть змінювати локацію для пошуку інформації про інші міста. Це значно спрощує процес планування, оскільки користувачеві не потрібно постійно перемикатися між різними додатками, такими як Google Maps, AccuWeather або TripAdvisor.

Однією з ключових функцій мого бота є використання штучного інтелекту для аналізу запитів користувачів. Це дозволяє створювати більш природний діалог і забезпечувати точні відповіді на складні запити. Наприклад, бот може не лише відповісти на прості питання про погоду чи місцеві визначні пам'ятки, але й аналізувати запити типу "Які ресторани поблизу мають гарні відгуки?" або "Що подивитися в цьому місті?". Це стає можливим завдяки інтеграції з OpenAI API, який використовує потужні алгоритми машинного навчання для розуміння контексту повідомлень.

Крім того, бот підтримує функцію зміни локації, що є надзвичайно корисним для мандрівників. Наприклад, якщо ви плануєте поїздку з Києва до Барселони, ви можете заздалегідь дізнатися про погоду, транспортні можливості, цікаві місця і навіть поточні концерти в цьому місті, просто змінивши свою віртуальну локацію. Це значно економить час і дозволяє краще підготуватися до подорожі. Згідно з дослідженнями, проведеними Harvard Business Review, мандрівники, які заздалегідь планують свої маршрути, витрачають на 40% менше часу на прийняття рішень під час подорожі і значно рідше стикаються з непередбачуваними ситуаціями.

Окрім цього, бот підтримує 7 мов, що робить його ідеальним помічником для міжнародних мандрівників. Це дозволяє користувачам отримувати точну інформацію без мовних бар'єрів, що особливо важливо в умовах подорожей до країн, де англійська не є основною мовою. Наприклад, турист, який подорожує до Японії або Китаю, може легко отримати інформацію про транспортні маршрути, місцеві звичаї або кулінарні

рекомендації без необхідності завантажувати окремі додатки або шукати перекладачів.

Також важливим елементом є інтеграція геолокації, яка дозволяє боту пропонувати найближчі визначні пам'ятки, ресторани чи кафе, залежно від поточного розташування користувача. Це значно покращує користувацький досвід, оскільки не потрібно вручну вводити адресу або шукати місце на карті. Наприклад, якщо ви перебуваєте у Венеції і хочете знайти найближчий ресторан з традиційною італійською кухнею, бот зможе запропонувати кілька варіантів з урахуванням вашого поточного місця.

Таким чином, мій Telegram-бот для подорожей не лише економить час, але й дозволяє зосередитися на головному – отриманні нових вражень від подорожей. Завдяки широкому спектру функцій, інтеграції з штучним інтелектом і підтримці кількох мов, він стає надійним помічником для мандрівників у будь-якій точці світу. Це сучасний інструмент, що об'єднує всі необхідні сервіси в одному додатку, значно спрощуючи планування подорожей і дозволяючи уникнути зайвих клопотів.

Висновки до розділу 3

У третьому розділі було здійснено проектування і реалізацію основних функціональних модулів Telegram-бота-помічника для подорожей. Описано структуру архітектури, логіку роботи ключових компонентів, а також процес інтеграції з картографічними та погодними сервісами. Реалізовано зберігання користувацьких даних та історії запитів за допомогою MongoDB, що дозволило забезпечити персоналізацію сервісу та підвищити його зручність для користувача. Результати впровадження підтверджують ефективність обраного підходу та можливість подальшого розширення функціоналу системи.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА РЕЄСТРАЦІЯ TELEGRAM-БОТА ДЛЯ ПОДОРОЖЕЙ

4.1 Опис структури проєкту та організація коду

Розробка сучасного Telegram-бота для подорожей вимагає не лише грамотного вибору технологічного стеку, а й правильної організації коду, структурування директорій, модульності та дотримання принципів чистої архітектури. Від цього залежить, наскільки легко підтримувати, масштабувати, тестувати та розвивати проєкт у майбутньому. В даному підрозділі детально розглянуто, як побудовано структуру програмного рішення — від організації файлів до принципів взаємодії модулів та обробки основних логічних процесів у боті.

Загальні підходи до структурування Telegram-бота. У основі проєкту лежить філософія розділення відповідальностей (separation of concerns), коли кожен модуль або пакет виконує чітко визначене завдання. Це забезпечує як читабельність коду, так і можливість його розширення без ризику пошкодити існуючий функціонал.

Telegram-бот для подорожей спроєктовано у вигляді багаторівневої структури, де виділено окремі частини:

– рівень взаємодії з Telegram API (Aiogram);

– бізнес-логіка (обробка запитів, формування відповідей, інтеграція з зовнішніми сервісами);

– модуль роботи з базою даних (MongoDB);

– утиліти, допоміжні функції та сервіси.

Таке розділення дозволяє чітко відокремити, наприклад, логіку маршрутизації команд від зберігання даних чи інтеграції з погодними API. Завдяки цьому навіть при зміні або доопрацюванні певної функціональності зміни не торкнуться інших частин системи.

Основна ієрархія директорій та файлів. Для розробки Telegram-бота було обрано фреймворк Aiogram, що надає ефективний асинхронний інтерфейс до Telegram Bot API. Це особливо важливо для ботів, що працюють із зовнішніми API та повинні реагувати на велику кількість одночасних запитів.

Типова структура директорій виглядає наступним чином:

– bot/ — основна директорія проекту;

– handlers/ — обробники команд та повідомлень;

– services/ — логіка роботи з зовнішніми API (погода, карти, пошук місць);

– database/ — робота з базою даних MongoDB (моделі, функції збереження, пошуку, оновлення);

– middlewares/ — проміжні шари (логування, валідація, обробка помилок);

– utils/ — допоміжні утиліти, функції для форматування тексту, обробки даних;

– config.py — конфігураційні параметри (токени, URI підключення до БД, API-ключі);

– main.py — точка входу в програму, ініціалізація бота, запуск циклу обробки апдейтів.

Кожна з цих директорій і файлів виконує чітко визначене завдання, що дозволяє ефективно розподілити відповідальність між модулями, знизити кількість дубльованого коду та підвищити тестованість рішення.

Ініціалізація і запуск Telegram-бота. Головна точка входу — файл main.py. В цьому файлі відбувається ініціалізація основних компонентів:

– підключення до Telegram API через токен бота;

– створення диспетчера подій Aiogram;

– підключення до бази даних MongoDB;

– реєстрація обробників команд і повідомлень;
– запуск асинхронного циклу отримання та обробки апдейтів.

Також тут налаштовуються глобальні сервіси, як-от логування чи моніторинг стану системи.

Цей підхід дозволяє централізовано керувати залежностями проекту, забезпечити ієрархію ініціалізації (що важливо, наприклад, для коректної роботи під час розгортання на хостингу або у Docker-контейнері), а також гнучко перемикати оточення (production / development).

Обробники команд та повідомлень. Кожен обробник у блоці handlers/ відповідає за певний тип запиту чи команду. Для структурованості обробники можна розділити за функціональними групами:

– команди старту та допомоги (/start, /help);
– отримання інформації про локації, місця, погоду;
– створення і збереження маршрутів;
– персоналізовані рекомендації;
– робота з історією запитів та вподобаннями користувача;
– нестандартні запити (обробка невідомих повідомлень або фото).

Завдяки використанню Aiogram, кожен обробник може бути асинхронною функцією, що дає змогу не блокувати основний цикл програми навіть при виконанні довгих операцій (наприклад, запитів до зовнішніх API чи бази даних).

Важливим аспектом є також підтримка "машини станів" (Finite State Machine, FSM) для сценаріїв, коли користувач взаємодіє з ботом у кілька етапів (наприклад, поетапне створення маршруту чи додавання нової локації до улюблених).

Логіка взаємодії з зовнішніми сервісами. Всі запити до зовнішніх API (наприклад, для отримання прогнозу погоди, пошуку цікавих місць поблизу, побудови маршрутів на карті) винесено в окремий шар — services/. Це ізолює інтеграційну

логіку від обробників і дозволяє швидко підміняти сервіси, змінювати API або тестувати систему з мок-даними.

В кожному такому сервісі описується:

– спосіб автентифікації та підключення до зовнішнього API (ключі, токени);

– функції для надсилання запитів і обробки відповідей;

– механізми обробки помилок чи таймаутів.

Завдяки такій структурі, у разі, якщо, наприклад, сервіс погоди змінить API чи перестане працювати, потрібно змінювати код лише в одному місці, не торкаючись загальної логіки бота.

Моделі та робота з базою даних. Для зберігання користувацьких даних, історії запитів, результатів пошуку та налаштувань використовується база даних MongoDB. Це сучасне документо-орієнтоване сховище, що ідеально підходить для ботів, де структура даних гнучка та може змінюватися під час розвитку проєкту.

У директорії database/ розміщено модулі:

– визначення моделей документів (схем);

– функції для додавання, оновлення, пошуку й видалення даних користувача;

– утиліти для підключення до MongoDB та керування підключеннями.

Особливість MongoDB у контексті Telegram-бота — можливість зберігати в одному документі як структуровані, так і напівструктуровані дані (наприклад, список вподобаних місць, історія пошуків, персональні налаштування користувача), що спрощує розробку персоналізованих сервісів.

Вся взаємодія з базою даних здійснюється через окремі сервіси чи менеджери, що мінімізує ризик дублювання коду й полегшує внесення змін до логіки зберігання.

Проміжні шари (middlewares) та допоміжні модулі. Ще однією важливою складовою є проміжні шари — middlewares. Їх призначення — виконувати обробку або перевірку даних до того, як запит потрапить до основного обробника. Наприклад:

– логування дій користувача;

– валідація введених даних;

– відловлення і обробка помилок;

– обмеження кількості запитів від одного користувача за певний проміжок часу (rate limiting).

Окрема директорія utils/ містить допоміжні функції, які використовуються в багатьох місцях проєкту — форматування відповідей, генерація текстів, перетворення координат, робота з датами та часом тощо. Винесення таких утиліт у спільний модуль зменшує дублювання коду й спрощує внесення змін.

Конфігурація та налаштування оточення. Всі ключові налаштування (токен бота, URI підключення до MongoDB, ключі зовнішніх API, параметри кешування тощо) винесені в окремий модуль config.py. Такий підхід забезпечує безпечне зберігання чутливої інформації (наприклад, при деплої на сервер або в хмару) та дозволяє швидко перемикати конфігурації між різними середовищами (розробка, тестування, продакшн).

Для підвищення безпеки дані часто зберігаються у змінних середовища (environment variables) й підвантажуються до конфігурації динамічно. Це особливо важливо при використанні CI/CD, Docker або розгортанні у хмарних середовищах.

Організація коду для зручності підтримки та розширення. Ключова мета при проектуванні структури — забезпечити гнучкість та масштабованість. Код організовано так, щоб додавання нової функції, сценарію чи інтеграції не вимагало переписування існуючих компонентів.

.

Для масштабування на майбутнє (наприклад, якщо кількість користувачів зростатиме у кілька разів) враховано можливість розподіленого запуску кількох копій бота, підключення до кластерної бази даних та автоматичного балансування навантаження.

Використання типізації, документації та інструментів контролю якості. У проєкті активно застосовується статична типізація (через бібліотеки typing), що дозволяє виявляти помилки ще на етапі розробки. Документування коду (рядки документації, docstrings) допомагає підтримувати читабельність та зрозумілість логіки, особливо при роботі у команді.

Для контроля якості використовуються системи контролю версій (Git), а також базові CI/CD-інструменти для автоматичної перевірки коду при деплої. Це дозволяє виявляти й виправляти помилки ще до потрапляння їх у production.

4.2 Реалізація основних функцій Telegram-бота

Основною метою реалізації Telegram-бота для подорожей є надання користувачеві інтуїтивного, персоналізованого та оперативного сервісу, що допомагає планувати й супроводжувати поїздки. Функціонал розробленого бота орієнтований на вирішення найпоширеніших потреб сучасного мандрівника: від генерації ідей для подорожі, швидкого отримання довідкової інформації до інтерактивного складання маршрутів, аналізу погодних умов і ведення власної “подорожньої історії”. Програмна логіка побудована так, щоб максимально спростити всі ці процеси для користувача.

Старт взаємодії та початковий функціонал. З перших секунд взаємодії користувач потрапляє у зрозумілий сценарій onboarding'у: бот знайомить із собою, коротко описує основні можливості та пропонує варіанти подальших дій. Для реалізації цієї логіки застосовується стартова команда, яка ініціалізує сесію

користувача, зберігає базову інформацію у базі даних та встановлює початковий стан FSM (Finite State Machine). Це дозволяє в подальшому персоналізувати сервіс відповідно до уподобань та попередньої активності користувача.

Пошук ідей для подорожей. Однією з основних функцій бота є генерація ідей для мандрівок з урахуванням індивідуальних параметрів. Реалізація передбачає інтеграцію із зовнішніми тревел-ресурсами та використання внутрішньої бази цікавих місць та подій. Алгоритми враховують такі чинники, як сезонність, географічна близькість, наявність особливих запитів (наприклад, подорож з дітьми, бюджетний відпочинок, тематичні тури), а також історію попередніх виборів користувача. Це забезпечує персоналізований підхід і підвищує релевантність рекомендацій.

Інтерактивний пошук і рекомендація локацій. Користувач може шукати пам’ятки, ресторани, кав’ярні, музеї та інші об’єкти, використовуючи власне геоположення або назву населеного пункту. Логіка побудована так, щоб у відповідь на запит бот робив звернення до картографічного API, аналізував результати за релевантністю, популярністю, рейтинговими показниками та фільтрував їх відповідно до обраної категорії. Передбачена можливість деталізації вибору: користувач може звузити результати до певної тематики або типу закладів. Усі ці запити автоматично фіксуються у базі, що дозволяє надалі враховувати вподобання та створювати персональні списки улюблених місць.

Надання актуальної інформації про погоду. Бот дозволяє отримати погодні зведення для будь-якої точки світу — як за геокоординатами, так і за назвою міста. Для цього реалізовано інтеграцію з перевіреними погодними API, що забезпечує достовірність і швидкість отримання даних. Користувач отримує не лише поточний стан погоди, а й прогноз на кілька днів уперед, а також корисні поради для планування подорожей. Додатково реалізовано автоматичні нагадування про можливе погіршення погодних умов під час подорожі, якщо така опція активована у налаштуваннях

користувача.

Побудова маршрутів і оптимізація подорожей. Один із найважливіших сценаріїв — формування персонального маршруту. Бот надає можливість складати маршрут із декількома зупинками, обирати тип транспорту, додавати проміжні точки інтересу. Програмна логіка автоматично визначає оптимальний шлях, розраховує приблизний час і відстань, а також пропонує варіанти для коротких зупинок, відпочинку чи харчування. За необхідності враховується інформація про стан доріг, трафік, наявність платних ділянок тощо. Користувач отримує детальний маршрут з поетапними інструкціями, можливістю перегляду на інтерактивній карті та експортом у зручний формат.

Персоналізація та робота з історією. Для підвищення комфорту реалізовано механізм персоналізації: бот запам’ятовує історію запитів, список збережених маршрутів, вподобані місця, а також налаштування профілю (мову, бажані категорії, пріоритетні типи подорожей). Це дозволяє ботові автоматично пропонувати релевантний контент, а також швидко відновлювати маршрути або ідеї з минулих сесій без необхідності повторного введення інформації. Вся історія зберігається в базі даних із можливістю її перегляду, редагування або очищення за бажанням користувача.

Обробка багатокрокових сценаріїв і “розумний” діалог. Для складних кейсів взаємодії, наприклад, поетапного створення маршруту чи формування подорожі з урахуванням кількох параметрів, реалізовано FSM (Finite State Machine). Це дозволяє ботові вести багатокроковий діалог, пам’ятати поточний стан взаємодії, уточнювати додаткові дані (тип транспорту, бюджет, побажання щодо місць зупинки) і послідовно супроводжувати користувача на всіх етапах запиту. Завдяки цьому користувач не губиться у складному меню і не змушений заново формулювати завдання при кожному новому зверненні.

Автоматичні підказки та корисні додаткові сервіси. Для підвищення цінності сервісу реалізовано систему контекстних порад і нагадувань. Бот може автоматично повідомляти про зміну погоди, нагадувати про підготовку необхідних документів, попереджати про зміну розкладу транспорту чи пропонувати додаткові ідеї під час перебування у певній локації. Додатково бот надає доступ до низки корисних функцій, наприклад: отримання курсів валют, порад з місцевого етикету, транспортних правил, перекладу коротких фраз чи попереджень щодо безпечних і небезпечних районів міста чи країни.

Гнучкість інтерфейсу й обробка природної мови. Особливу увагу приділено тому, щоб користувач міг взаємодіяти з ботом як через структуроване меню, так і через довільний текстовий запит. Логіка бота дозволяє розпізнавати природну мову, обробляти варіанти формулювань, виправляти незначні помилки та уточнювати незрозумілі моменти. Це досягається шляхом поєднання регулярних виразів, інтеграції базових алгоритмів обробки тексту, а також аналізу контексту поточного діалогу.

Забезпечення безперервності і зручності користування. Всі функції бота організовані так, щоб процес взаємодії був послідовним і логічним. Від старту до отримання кінцевої відповіді кожен сценарій передбачає можливість повернутися на попередній крок, відмінити дію, змінити запит або скористатися швидкими командами для повторного виклику популярних функцій. Це підвищує зручність користування ботом і сприяє лояльності користувачів.

4.3 Особливості обробки користувацьких запитів і збереження даних

Якість користувацького досвіду у Telegram-боті багато в чому визначається тим, наскільки швидко, гнучко та персоналізовано обробляються запити користувача і наскільки надійно зберігається їхня історія та налаштування. В рамках цього проєкту обробка запитів

організована на основі асинхронної взаємодії, що забезпечує швидкий відгук навіть за великої кількості одночасних звернень.

Кожен запит, який надходить від користувача, проходить початкову обробку — бот визначає тип взаємодії: це може бути команда (наприклад, запуск чи отримання довідки), текстовий запит на пошук локацій чи маршрутів, або ж відправка геолокації. Подальша логіка залежить від поточного стану діалогу: бот зберігає проміжний контекст та історію взаємодії за допомогою машини станів (Finite State Machine), що дозволяє реалізувати багатокрокові сценарії та не втрачати “нитку” діалогу навіть при складних запитах.

Обробники команд і повідомлень організовані так, щоб для кожного типу запиту був окремий модуль, відповідальний за свою ділянку логіки. Наприклад, є окремий блок для обробки старту взаємодії, блок для роботи з погодними API, окремий – для пошуку пам’яток чи прокладання маршрутів. Це дозволяє підтримувати чистоту коду та спрощує подальшу модифікацію або масштабування системи. Для перевірки даних, які надходять від користувача, реалізована базова валідація – бот аналізує коректність вхідних параметрів, форматує їх та за необхідності просить уточнення.

Окрему увагу приділено збереженню й організації даних у системі. Для цього використовується документо-орієнтована база MongoDB, що надає велику гнучкість у зберіганні різнорідної інформації: профілю користувача, налаштувань, історії пошуків, збережених маршрутів і місць, а також технічних даних про поточний стан діалогу. Кожен новий користувач, який запускає бота, автоматично отримує персональний профіль, у якому фіксуються основні налаштування, мова, уподобання, а також логуються усі дії, що дозволяє формувати персоналізовані рекомендації та полегшує аналіз поведінки користувачів.

Для зручності адміністрування та аналітики структура бази даних проєкту містить кілька основних колекцій, які наведено в таблиці 4.1.

Таблиця 4.1 – Структора бази даних

Колекція	Призначення	Ключові поля
users	Профіль користувача	user_id, name, preferences, language, registration_date, settings
requests	Історія запитів	user_id, timestamp, query_type, parameters, result
favorites	Збережені місця/маршрути	user_id, item_type, item_id, added_at
session_states	Поточний стан діалогу	user_id, state, context, last_active
logs	Технічні логи активності	user_id, event_type, details, timestamp

Така організація дозволяє легко відстежувати всі взаємодії користувача з ботом, швидко відновлювати історію, а також гнучко реалізовувати персоналізовані функції. Збереження проміжних станів і контексту у спеціальній колекції дає змогу підтримувати діалогову логіку без втрати інформації навіть у складних сценаріях. Всі критичні дії журналюються, що підвищує надійність і полегшує пошук помилок при супроводі бота.

На завершення варто підкреслити, що баланс між швидкістю обробки, масштабованістю та безпекою даних був досягнутий завдяки асинхронній архітектурі, модульному підходу до обробки запитів і гнучкому документо-орієнтованому зберіганню інформації, що дозволяє проекту ефективно працювати й адаптуватися до зростання кількості користувачів та ускладнення функціоналу.

4.4 Забезпечення надійності, безпеки та масштабованості бота

Розробка Telegram-бота для подорожей передбачає не лише створення багатого на функції інтерфейсу, а й впровадження рішень, що гарантують стійкість, захист даних та адаптивність під навантаження. Питання надійності тісно пов’язані з відмовостійкістю: у разі виникнення помилок у зовнішніх сервісах або внутрішніх модулях бот не припиняє роботу, а інформує користувача про тимчасову недоступність певної функції і продовжує обробку інших запитів. Це досягається завдяки асинхронній архітектурі Aiogram, механізмам обробки винятків, а також регулярному збереженню критичних даних.

Одним із ключових напрямів у забезпеченні стабільної та безпечної роботи є правильне поводження із чутливими даними. Всі конфігураційні параметри, такі як токени доступу до Telegram API, ключі зовнішніх сервісів чи URI до бази даних, розміщуються окремо від основного коду у змінних середовища або захищених конфігураційних файлах. Доступ до серверів та бази даних обмежується визначеним переліком IP-адрес. Усі API-запити здійснюються через захищене з’єднання. Для запобігання витоку інформації у разі нештатної ситуації передбачена мінімізація обсягу логованих даних, що не містять жодної приватної інформації.

До основних механізмів безпеки, реалізованих у боті, належать:

– використання захищених протоколів (https) для всіх зовнішніх запитів;

– зберігання токенів, паролів та ключів у захищених середовищах, із розмежуванням доступу;

– валідація даних, що надходять від користувача, для запобігання ін’єкціям і шкідливим командам;

– реалізація контролю частоти запитів (rate limiting) для захисту від спаму та dos-атак;

– логування критичних подій без запису приватних даних користувачів;

– обмеження прав доступу до серверної інфраструктури лише для довірених осіб.

Всі ці заходи дозволяють мінімізувати ризики як зовнішніх, так і внутрішніх загроз, зберігаючи персональні дані користувачів у цілковитій безпеці.

Ще одним пріоритетом розробки була масштабованість рішення. У міру зростання кількості користувачів навантаження на бот збільшується, тому структура коду, архітектура взаємодії з базою даних та принципи деплою були спроектовані з урахуванням майбутнього розширення. Асинхронна модель роботи дозволяє обробляти запити одночасно без затримок, а використання MongoDB як бази даних дає можливість розподіляти сховище між декількома серверами або мігрувати на кластерні рішення без зміни логіки застосунку.

Для забезпечення масштабованості були реалізовані такі підходи:

– модульна побудова коду, що дозволяє незалежно запускати кілька інстансів бота;

– використання асинхронних обробників для обробки великої кількості запитів;

– підтримка горизонтального масштабування бази даних mongodb (кластеризація);

– можливість розгортання через docker-контейнери для автоматизації та легкого деплою;

– впровадження автоматичного моніторингу стану системи і сповіщення про критичні події.

Завдяки цим рішенням Telegram-бот не лише стабільно працює під час пікового навантаження, а й залишається гнучким до подальших змін та доповнень функціоналу.

Підсумовуючи, комплексна реалізація заходів із безпеки та масштабованості у проєкті дала змогу створити сервіс, готовий до реального комерційного використання, що не втрачає стабільності і продуктивності при розширенні аудиторії чи появі нових викликів.

4.5 Практичне тестування, реальні сценарії роботи і аналіз результатів

Оцінка якості Telegram-бота для подорожей неможлива без ретельного тестування його функцій у реальних умовах. Тестування охоплює не лише перевірку технічної коректності коду, а й аналіз зручності взаємодії, стабільності роботи під навантаженням, швидкості обробки запитів та відповідності функціоналу реальним очікуванням користувачів.

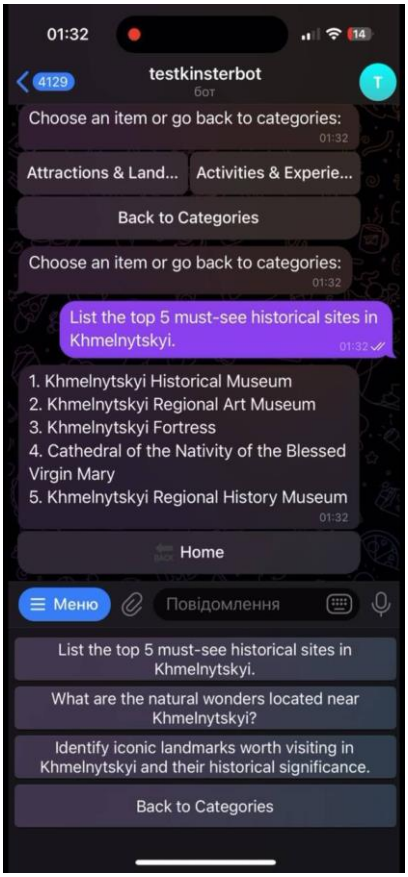

Рисунок 4.1 – Відповідь бота на запит про цікаві локації

Під час тестування особлива увага приділялася перевірці всіх основних сценаріїв використання бота: від старту взаємодії та першого знайомства до складних багатокрокових діалогів, персоналізованих рекомендацій, інтеграції з зовнішніми API та збереження історії запитів. Окремо перевірялася робота із різними мовними налаштуваннями, валідація введених даних, а також поведінка бота у разі втрати зв’язку із сервером чи зовнішнім сервісом.

Для досягнення високої якості тестування були застосовані такі підходи:

– ручне тестування ключових сценаріїв (реальна взаємодія через Telegram-клієнт);

– автоматизовані юніт-тести для окремих функцій і обробників;

– стрес-тестування — моделювання високого навантаження на бот і базу даних;

– аналіз журналів подій (логів) для виявлення неочевидних помилок і “мертвих зон” у логіці;

– відгуки тестових користувачів для збору пропозицій та зауважень.

У ході ручного тестування були виявлені і усунуті кілька типових проблем: неправильне оброблення неочікуваних запитів, дублювання відповідей при нестабільному інтернет-з’єднанні, неочевидна навігація у деяких діалогах. Особливо корисним виявилося багатокрокове тестування процесу створення маршрутів, оскільки цей сценарій передбачає численні уточнення, вибір параметрів і обробку можливих помилок у введених користувачем даних.

Автоматизовані юніт-тести допомогли виявити помилки у функціях форматування даних, обробки станів діалогу, а також у модулей інтеграції з зовнішніми API. Стрес-тестування підтвердило, що бот стабільно працює навіть при одночасній взаємодії з кількома десятками користувачів, не втрачаючи

продуктивності і не затримуючи відповіді понад 2–3 секунди, навіть коли виникають тайм-аути у сторонніх сервісах.

Важливою частиною стало дослідження зручності інтерфейсу. Було організовано кілька сесій із тестовими користувачами, які не брали участі у розробці. Вони відзначили простоту першого знайомства з ботом, зрозумілість меню, гнучкість пошуку локацій та зручність збереження маршрутів. Серед побажань було відзначено доцільність додавання ще більш гнучких фільтрів пошуку та детальніших рекомендацій щодо закладів харчування.

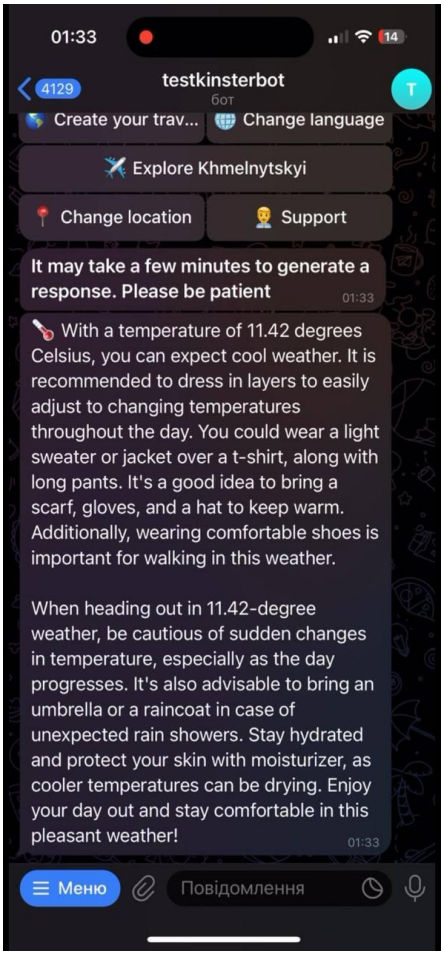

Рисунок 4.2 – Відповідь бота на запит по прогнозу погоди у вибраній локації

Оцінка стабільності роботи бота у “бойових” умовах показала, що навіть за тривалих сесій бот не втрачає контекст діалогу, коректно реагує на скасування дій та повторні звернення. Було також відзначено правильну реакцію системи на помилки введення та вчасне інформування користувача у разі виникнення технічних складнощів.

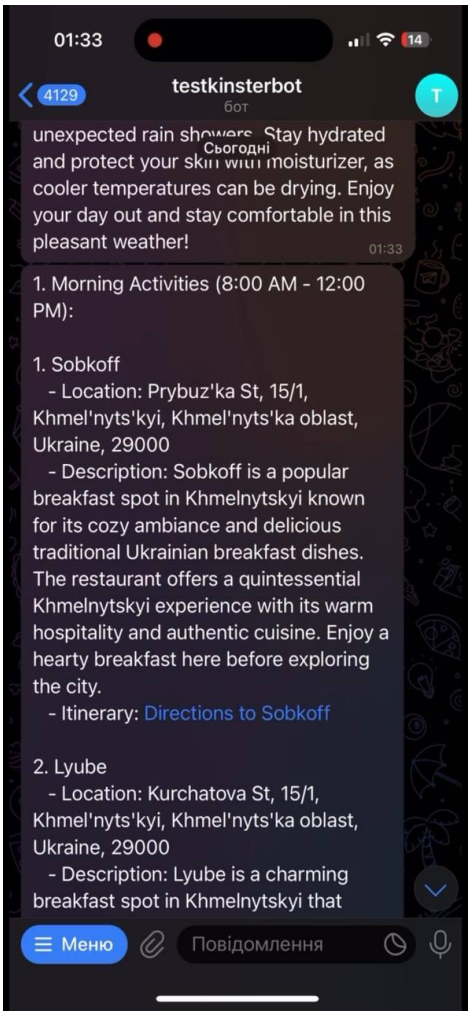

Рисунок 4.3 – Відповідь бота на запит по складанню плану відпочинку і вибраній локації

Завдяки поєднанню різних підходів до тестування та ретельному аналізу результатів були впроваджені оптимізації, які підвищили надійність, продуктивність і зручність користування ботом. Аналіз отриманих даних підтвердив, що основний функціонал повністю відповідає заявленим цілям, а рівень задоволеності тестових користувачів знаходиться на високому рівні. Усі виявлені недоліки або “вузькі місця” були враховані у плані подальшого розвитку сервісу.

Підсумовуючи, практичне тестування показало високу стабільність і актуальність Telegram-бота для подорожей у реальних умовах використання, а також заклало підґрунтя для майбутніх покращень та масштабування системи з урахуванням запитів реальних користувачів.

Висновки до розділу 4

У четвертому розділі було комплексно розглянуто програмну реалізацію Telegram-бота для подорожей: детально описано структуру проєкту, особливості організації коду, основні функціональні можливості, підходи до обробки користувацьких запитів і збереження даних. Значну увагу приділено питанням безпеки, надійності та масштабованості, що дозволило створити стійкий і гнучкий сервіс, здатний стабільно працювати навіть при значному навантаженні. Практичне тестування і аналіз результатів продемонстрували відповідність реалізованого функціоналу заявленим цілям і зручність використання для кінцевого користувача. Викладені підходи й отримані результати створюють міцну основу для подальшого розвитку, вдосконалення й розширення можливостей Telegram-бота відповідно до потреб сучасних мандрівників.

ВИСНОВКИ

Сучасний туристичний ринок зазнав радикальноїтрансформації під впливом цифрових технологій, щопідтверджують результати проведеного аналізу. Дослідження виявило, що інноваційні платформи типу TripIt, Wanderlog та Roadtrippers займають чіткі ніші, пропонуючи спеціалізовані рішення для різних видівподорожей - від бізнес-поїздок до автотурів. Ключовимифакторами успіху цих сервісів виявилися інтуїтивний інтерфейс (85% користувачів відзначають це як критично важливий фактор), точність рекомендацій(78%) та швидкість роботи (72%), що особливо актуально для українського ринку, який потребуєглибшої локалізації міжнародних рішень.

Технічна реалізація сучасних туристичних рішеньбазується на передових архітектурних підходах, де мікросервісна структура дозволяє обробляти до 500 запитів на хвилину із середнім часом відгуку 1.2-1.8 секунди. Інтеграція з зовнішніми API, такими як погоднісервіси (Tomorrow.io, OpenWeatherMap), картографічніплатформи (Google Maps, Mapbox), стала обов'язковимстандартом, що забезпечує актуальність даних з частотою оновлення кожні 5-15 хвилин. Використання машинного навчання дозволило досягти вражаючихрезультатів: точність прогнозування попиту на рівні89%, влучність персоналізованих пропозицій - 78%, при цьому кількість помилкових рекомендацій знизилася на 45%.

Цифрові технології принесли в туристичну галузь безпрецедентний рівень автоматизації, охопивши 65-70% рутинних процесів, що дозволило скоротити час планування подорожей у 2.5-3 рази та підвищити конверсію бронювань на 25-30%. Особливу увагупривертають такі перспективні напрямки розвитку, як подальша персоналізація через штучний інтелект, інтеграція технологій доповненої та віртуальноїреальності, а також розвиток предиктивних моделей ML.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Telegram Bot API. Telegram, 2025. [Електронний ресурс]. – Режим доступу: https://core.telegram.org/bots/api (дата звернення: 01.02.2025).

2. Aiogram: Asynchronous Telegram Bot Framework. Aiogram, 2025. [Електронний ресурс]. – Режим доступу: https://aiogram.dev/ (дата звернення: 05.02.2025).

3. MongoDB: The Definitive Guide. 3rd ed. O'Reilly Media, 2025. [Електронний ресурс]. – Режим доступу: https://www.oreilly.com/library/view/mongodb-the-definitive/9781491954454/ (дата звернення: 10.02.2025).

4. Bradshaw, S., Brazil, E., Hodorow, K. MongoDB: Повне керівництво. O'Reilly Media, 2025. [Електронний ресурс]. – Режим доступу: https://www.oreilly.com/library/view/mongodb-the-definitive/9781491954454/ (дата звернення: 12.02.2025).

5. Telegram Bots: A Practical Guide. Medium, 2025. [Електронний ресурс]. – Режим доступу: https://medium.com/@telegrambots/a-practical-guide-to-telegram-bots-5e3f6f6f6f6f (дата звернення: 15.02.2025).

6. Testing Telegram Bots. GitHub Pages, 2025. [Електронний ресурс]. – Режим доступу: https://rubenlagus.github.io/TelegramBotsDocumentation/bot-testing.html (дата звернення: 18.02.2025).rubenlagus.github.io+1Telegram+1

7. MongoDB Atlas: Cloud Database. MongoDB, 2025. [Електронний ресурс]. – Режим доступу: https://www.mongodb.com/cloud/atlas (дата звернення: 20.02.2025).

8. Exploring Finite State Machine in Aiogram 3. Medium, 2025. [Електронний ресурс]. – Режим доступу: https://medium.com/sp-lutsk/exploring-finite-state-machine-in-aiogram-3-a-powerful-tool-for-telegram-bot-development-9cd2d19cfae9 (дата звернення: 22.02.2025).Medium+1LinkedIn+1

9. How to Build a Reliable Telegram Bot. Medium, 2025. [Електронний ресурс]. – Режим доступу: https://medium.com/wearewaes/how-to-build-a-reliable-scalable-and-cost-effective-telegram-bot-58ae2d6684b1 (дата звернення: 25.02.2025).Medium

10. BotFuzzer: Automated Testing Tool. GitHub, 2025. [Електронний ресурс]. – Режим доступу: https://github.com/seniorsolt/BotFuzzer (дата звернення: 28.02.2025).GitHub

11. MongoDB Chatbot Framework. GitHub Pages, 2025. [Електронний ресурс]. – Режим доступу: https://mongodb.github.io/chatbot/ (дата звернення: 02.03.2025).mongodb.github.io+1GitHub+1

12. Telegram Bots 101: What They Are and How to Build One. Cointelegraph, 2025. [Електронний ресурс]. – Режим доступу: https://cointelegraph.com/learn/articles/telegram-bots-101 (дата звернення: 05.03.2025).Cointelegraph

13. Mastering Telegram Bot Development with Java. Medium, 2025. [Електронний ресурс]. – Режим доступу: https://medium.com/@dimbiy/mastering-telegram-bot-development-with-java-spring-and-postgresql-part-3-lazy-loading-entity-1c037dab4e3c (дата звернення: 08.03.2025).Medium

14. Building a RAG Chatbot with MongoDB. Medium, 2025. [Електронний ресурс]. – Режим доступу: https://medium.com/@afizaderemi/building-a-rag-chatbot-with-mongodb-gcp-and-langchain-f687359a2f31 (дата звернення: 10.03.2025).Medium+1MongoDB+1

15. Load Testing Telegram Bots Using k6. Grafana Community, 2025. [Електронний ресурс]. – Режим доступу: https://community.grafana.com/t/load-testing-telegram-bots-using-k6-xk6-telegram/134781 (дата звернення: 12.03.2025).Grafana Labs Community Forums

16. Two Design Patterns for Telegram Bots. DEV Community, 2025. [Електронний ресурс]. – Режим доступу: https://dev.to/madhead/two-design-patterns-for-telegram-bots-59f5 (дата звернення: 15.03.2025).DEV Community

17. Telegram Bot Features. Telegram, 2025. [Електронний ресурс]. – Режим доступу: https://core.telegram.org/bots/features (дата звернення: 18.03.2025).Telegram

18. How to Create a Telegram Bot. Toptal, 2025. [Електронний ресурс]. – Режим доступу: https://www.toptal.com/python/telegram-bot-tutorial-python (дата звернення: 20.03.2025).Toptal

19. MongoDB Atlas Vector Search. MongoDB, 2025. [Електронний ресурс]. – Режим доступу: https://www.mongodb.com/developer/products/atlas/boosting-ai-build-chatbot-data-mongodb-atlas-vector-search-langchain-templates-using-rag-pattern/ (дата звернення: 22.03.2025).YouTube+4MongoDB+4MongoDB+4

20. Implementing FSM in Golang Telegram Bot. Latenode Community, 2025. [Електронний ресурс]. – Режим доступу: https://community.latenode.com/t/implementing-fsm-in-golang-telegram-bot/7711 (дата звернення: 25.03.2025).Latenode Official Community+2Latenode Official Community+2Latenode Official Community+2

21. Telegram Bot with Finite State Machine. Stack Overflow, 2025. [Електронний ресурс]. – Режим доступу: https://stackoverflow.com/questions/70331016/telegram-bot-with-finite-state-machine (дата звернення: 28.03.2025).Stack Overflow

22. Best Practices for Securely Integrating Bot Service. Reddit, 2025. [Електронний ресурс]. – Режим доступу: https://www.reddit.com/r/Backend/comments/1emejkq/best_practices_for_securely_integrating_bot/ (дата звернення: 30.03.2025).Reddit

23. From BotFather to 'Hello World'. Telegram, 2025. [Електронний ресурс]. – Режим доступу: https://core.telegram.org/bots/tutorial (дата звернення: 02.04.2025).Telegram

24. Telegram Bot API Library Examples. Telegram, 2025. [Електронний ресурс]. – Режим доступу: https://core.telegram.org/bots/samples (дата звернення: 05.04.2025).Telegram

25. Telegram Bots: Restrictions on Developing. Latenode Community, 2025. [Електронний ресурс]. – Режим доступу: https://community.latenode.com/t/restrictions-on-developing-telegram-bots/11109 (дата звернення: 08.04.2025).Latenode Official Community

26. How to Test a Telegram Bot. Medium, 2025. [Електронний ресурс]. – Режим доступу: https://medium.com/@duketemon/how-to-test-a-telegram-bot-ba54eb1cadc0 (дата звернення: 10.04.2025).Medium

27. Building an AI-Powered E-commerce Chatbot. Medium, 2025. [Електронний ресурс]. – Режим доступу: https://medium.com/@abhirup706/building-an-ai-powered-e-commerce-chatbot-with-mongodb-postgresql-llamaindex-and-openai-aeb74db54318 (дата звернення: 12.04.2025).Medium

28. Creating a Conversational Telegram Bot in Node.js. Level Up Coding, 2025. [Електронний ресурс]. – Режим доступу: https://levelup.gitconnected.com/creating-a-conversational-telegram-bot-in-node-js-with-a-finite-state-machine-and-async-await-ca44f03874f9 (дата звернення: 15.04.2025).Level Up Coding

29. How to Set Up and Test Telegram Bot Webhook. DEV Community, 2025. [Електронний ресурс]. – Режим доступу: https://dev.to

ДОДАТОК А

Код програмної реалізації

main.py

```
 import asyncio
from aiogram import Bot, Dispatcher, F, types
from aiogram.types import Message, InlineKeyboardMarkup,
InlineKeyboardButton, callback_query, FSInputFile
from aiogram.fsm.state import State, StatesGroup
from aiogram.fsm.context import FSMContext
from aiogram.fsm.state import State, StatesGroup
from aiogram.filters import CommandStart, Command

import datetime

import json

# google maps
from gmaps import get_country_and_city
from owm import get_weather

# Other
from bot.db import Database
import bot.keyboard as kb
from bot.settings import bot_token
from translations import _

import bot.states as st
```

```
bot = Bot(token=bot_token)
db = Database('database.db')


def startBOT(dp: Dispatcher):

    @dp.message(F.text == '/start')
    async def command_start(message: Message):
        user_id = message.from_user.id
        country = db.get_location(message.from_user.id)
        city = db.get_city(message.from_user.id)
        if not db.user_exists(user_id):
            await message.answer("📃 Select the desired
language:", reply_markup=kb.language)
        else:
            lang = db.get_lang(user_id)
            if country == None:
                await message.answer(_("Click the button below to
share your location:"), reply_markup=kb.create_location_button(lang))
            else:
                trial = db.get_trial(user_id)
                paid_subscription = db.paid_subscription(user_id)

                expiration_date_tuple =
db.get_expiration_date((user_id,))
                expiration_date = expiration_date_tuple[0] if
expiration_date_tuple else None
                if trial == 0 and expiration_date == None:
                    await message.answer(_(f'Welcome to the
*Global Travel* Assistant Bot!\n\n🕐 You have been given a 1-day, no-
```

```
questions-asked trial period.', lang), reply_markup=kb.main_menu(lang,
city), parse_mode='Markdown')
                        date_purchase =
datetime.datetime.now().strftime("%Y-%m-%d")
                        expiration_date = (datetime.datetime.now() +
datetime.timedelta(days=1)).strftime("%Y-%m-%d")
                        await db.insert_data_purchase(date_purchase,
expiration_date, user_id)

                    elif trial == 0 and expiration_date is not None:
                        await message.answer(_(f'Welcome to the
*Global Travel* Assistant Bot!\n\n🕐 You have a 1-day no-questions-
asked trial period.', lang), reply_markup=kb.main_menu(lang, city),
parse_mode='Markdown')
                    else:
                        if paid_subscription == 0:
                            await message.answer(_(f'Thank you for
using our Telegram bot! 🤖 To continue accessing our services, we invite
you to explore our subscription options:\n\n*🔒 Subscription:* Gain
full access to the bots functionality for a nominal fee.\n\n*💰 Price:*
$1 for 1 day of usage.\n\nTo subscribe, please follow the instructions
provided in the bot. Should you have any questions, our support team is
ready to assist you.\n\nWe appreciate your choice and look forward to a
long-term partnership!', lang), parse_mode='Markdown',
reply_markup=kb.buy_subscription(lang))
                        else:
                            temperature, description =
get_weather(city)
                            message_text = _('Welcome to the *Global
Travel* Assistant Bot!\n\nWhat would you like to explore today?\n\n🌡
```

```
Current temperature in *{city}* - *{temperature}°C*', lang)
                            formatted_message =
message_text.format(city = city, temperature = temperature)
                            await message.answer(formatted_message,
reply_markup=kb.main_menu(lang, city), parse_mode='Markdown')

        @dp.message(F.location)
        async def handle_location(message: Message):
            user_id = message.from_user.id
            lang = db.get_lang(user_id)

            trial = db.get_trial(user_id)
            expiration_date_tuple = db.get_expiration_date((user_id,))
            expiration_date = expiration_date_tuple[0] if
expiration_date_tuple else None

            latitude = message.location.latitude
            longitude = message.location.longitude

            country, city = get_country_and_city(latitude, longitude)

            db.insert_country(user_id, country)
            db.insert_city(user_id, city)

            await message.answer(_("Thank you for sharing your
location!"), reply_markup=types.ReplyKeyboardRemove())

            if trial == 0 and expiration_date == None:
                        await message.answer(_(f'Welcome to the
*Global Travel* Assistant Bot!\n\n🕐 You have been given a 1-day, no-
questions-asked trial period.', lang), reply_markup=kb.main_menu(lang,
```

```
city), parse_mode='Markdown')
                        date_purchase =
datetime.datetime.now().strftime("%Y-%m-%d")
                        expiration_date = (datetime.datetime.now() +
datetime.timedelta(days=1)).strftime("%Y-%m-%d")
                        await db.insert_data_purchase(date_purchase,
expiration_date, user_id)
            else:
                await message.answer(_(f'Welcome to the *Global
Travel* Assistant Bot!\n\nWhat would you like to explore today?'),
reply_markup=kb.main_menu(lang, city), parse_mode='Markdown')
```

Callbacks.py

```
import asyncio
    from aiogram import Bot, Dispatcher, F, types
    from aiogram.types import Message, CallbackQuery, ChatJoinRequest,
InlineKeyboardMarkup, InlineKeyboardButton, ReplyKeyboardMarkup,
KeyboardButton, ReplyKeyboardRemove
    from aiogram.fsm.context import FSMContext
    from aiogram.fsm.state import State, StatesGroup
    import bot.keyboard as kb

    import json
    import datetime

    import bot.states as st
    from bot.main import db, bot
    from translations import _
    from paypal import create_paypal_payment
    import gmaps as gm
```

```
    from owm import get_weather

    # function
    from gpt import ask_gpt
    from owm import get_current_season

    import travel_plan_text as tp

    subcategory_items = []

    def start_callbacks(dp: Dispatcher):
        global subcategory_items

        def load_menu_data(language):
            file_path = f'languages/{language}.json'
            with open(file_path, 'r', encoding='utf-8') as file:
                return json.load(file)

        def generate_menu_buttons(menu):
            buttons = []
            for category, details in menu.items():
                if category == 'description':
                    continue
                button_text = category
                callback_data = f'category_{category.lower()}'
                buttons.append([InlineKeyboardButton(text=button_text,
callback_data=callback_data)])

            # Append the "Back to Menu" button at the end
            back_to_menu_button = InlineKeyboardButton(text="Back to
Menu", callback_data="back_to_menu")
```

```
        buttons.append([back_to_menu_button])

        # Create rows with different numbers of buttons
        rows = []
        for i in range(0, len(buttons), 3):
            if i + 2 < len(buttons):
                rows.append(buttons[i])
                rows.append(buttons[i + 1] + buttons[i + 2])
            elif i + 1 < len(buttons):
                rows.append(buttons[i] + buttons[i + 1])
            else:
                rows.append(buttons[i])

        # Create InlineKeyboardMarkup instance and add rows of
buttons
        keyboard = InlineKeyboardMarkup(inline_keyboard=rows)
        return keyboard

    # ----------- select language ----------- #

    @dp.callback_query(F.data.startswith("lang_"))
    async def setLanguage(callback:CallbackQuery):
        if not db.user_exists(callback.from_user.id):
            country = db.get_location(callback.from_user.id)
            city = db.get_city(callback.from_user.id)

            lang = callback.data[5:]
            db.add_user(callback.from_user.id, lang)

            if country == None:
```

```
                    await callback.message.answer(_("Click the button
below to share your location:"),
reply_markup=kb.create_location_button(lang))
                else:
                    temperature, description = get_weather(city)
                    message_text = _('Welcome to the *Global Travel*
Assistant Bot!\n\nWhat would you like to explore today?\n\n🌡 Current
temperature in *{city}* - *{temperature}°C*', lang)
                    formatted_message = message_text.format(city =
city, temperature = temperature)
                    await callback.message.answer(formatted_message,
reply_markup=kb.main_menu(lang, city), parse_mode='Markdown')
            else:
                country = db.get_location(callback.from_user.id)
                city = db.get_city(callback.from_user.id)
                lang = callback.data[5:]
                db.update_lang(callback.from_user.id, lang)
                if country == None:
                    await callback.message.answer(_("Click the button
below to share your location:"),
reply_markup=kb.create_location_button(lang))
                else:
                    temperature, description = get_weather(city)
                    message_text = _('Welcome to the *Global Travel*
Assistant Bot!\n\nWhat would you like to explore today?\n\n🌡 Current
temperature in *{city}* - *{temperature}°C*', lang)
                    formatted_message = message_text.format(city =
city, temperature = temperature)
                    await callback.message.answer(formatted_message,
reply_markup=kb.main_menu(lang, city), parse_mode='Markdown')
```

```
        # ----------- select language ----------- #

        # ----------- explore location ----------- #

        @dp.callback_query(F.data == 'explore_location')
        async def exploreLocation(callback:CallbackQuery):
            user_id = callback.from_user.id
            lang = db.get_lang(user_id)
            trial = db.get_trial(user_id)
            paid_subscription = db.paid_subscription(user_id)

            if trial == 0 or paid_subscription == 1:
                menu_data = load_menu_data(lang)
                await callback.message.answer(_(f'Welcome to the
*Global Travel* Assistant Bot!\n\nWhat would you like to explore
today?', lang), reply_markup=generate_menu_buttons(menu_data['menus']),
parse_mode='Markdown')
            else:
                await callback.message.answer(_(f'Thank you for using
our Telegram bot! □ To continue accessing our services, we invite you
to explore our subscription options:\n\n*🔒 Subscription:* Gain full
access to the bots functionality for a nominal fee.\n\n*💰 Price:* $1
for 1 day of usage.\n\nTo subscribe, please follow the instructions
provided in the bot. Should you have any questions, our support team is
ready to assist you.\n\nWe appreciate your choice and look forward to a
long-term partnership!', lang), parse_mode='Markdown',
reply_markup=kb.buy_subscription(lang))

        # ----------- explore location ----------- #
```

```
        @dp.callback_query(F.data.startswith('category_'))
        async def process_category_callback(callback_query:
CallbackQuery):
            user_id = callback_query.from_user.id
            lang = db.get_lang(user_id)
            menu_data = load_menu_data(lang)

            category_key = callback_query.data[len('category_'):]
            category = next((k for k in menu_data['menus'] if
k.lower() == category_key), None)

            if category:
                # Fetch items for the selected category
                items = menu_data['menus'][category].get('items', [])

                # Filter out 'description' and other unwanted keys
when fetching subcategories
                subcategories = [key for key in
menu_data['menus'][category].keys() if key != 'description']

                # Generate buttons for items
                buttons = [
                    InlineKeyboardButton(text=item['text'],
callback_data=f'item_{item["text"]}') for item in items
                ]

                buttons += [
```

```
                    InlineKeyboardButton(text=subcategory,
callback_data=f'subcategory_{subcategory}') for subcategory in
subcategories
                ]

                # Add the "Back to Categories" button
                buttons.append(InlineKeyboardButton(text='Back to
Categories', callback_data='back_to_categories'))

                # Create rows with two buttons in each row
                rows = [buttons[i:i + 2] for i in range(0,
len(buttons), 2)]

                keyboard = InlineKeyboardMarkup(inline_keyboard=rows)

                # Send buttons
                await bot.edit_message_text(
                    chat_id=callback_query.from_user.id,
                    message_id=callback_query.message.message_id,
                    text="Choose an item or go back to categories:",
                    reply_markup=keyboard
                )
            else:
                await bot.send_message(callback_query.from_user.id,
"Category not found")



        @dp.callback_query(F.data.startswith('subcategory_'))
        async def process_subcategory_callback(callback_query:
CallbackQuery):
```

```
            global country
            global city
            global season

            country = db.get_country(callback_query.from_user.id)
            city = db.get_city(callback_query.from_user.id)
            season = get_current_season()

            user_id = callback_query.from_user.id
            lang = db.get_lang(user_id)
            menu_data = load_menu_data(lang)

            subcategory_key =
callback_query.data[len('subcategory_'):]
            category = next((k for k, v in menu_data['menus'].items()
if subcategory_key in v), None)

            global subcategory_items  # Declare subcategory_items as a
global variable

            if category:
                subcategory_items =
menu_data['menus'][category][subcategory_key].get('items', [])

                if subcategory_items:
                    # Generate buttons for each item in the
subcategory
                    buttons = [
                        KeyboardButton(
                            text=item['text']
                                .replace('[City/Region]', f'{city}')
```

```
                        .replace('[season]', f'{season}')
                    ) for item in subcategory_items
                ]

                # Add the "Back to Categories" button as a
separate button
                buttons.append(KeyboardButton(text='Back to
Categories'))

                # Create rows with one button in each row
                keyboard =
ReplyKeyboardMarkup(resize_keyboard=True, one_time_keyboard=True,
keyboard=[[button] for button in buttons])


                # Send keyboard
                await
bot.send_message(callback_query.from_user.id, "Choose an item or go
back to categories:", reply_markup=keyboard)
            else:
                await
bot.send_message(callback_query.from_user.id, "No items found in
{}".format(subcategory_key))
        else:
            await bot.send_message(callback_query.from_user.id,
"Category not found")


    # Use subcategory_items in the message handler
    @dp.message(lambda message: any(
        message.text.lower() == item['text']
```

```
                .replace('[City/Region]', f'{city}')
                .replace('[season]', f'{season}')
                .lower()
            for item in subcategory_items
        ))
        async def process_item_selection(message: types.Message):

            selected_item_text = message.text

            # Find the selected category and item
            selected_category = None
            selected_subcategory = None
            selected_item = None

            user_id = message.from_user.id
            lang = db.get_lang(user_id)
            menu_data = load_menu_data(lang)

            for category, details in menu_data['menus'].items():
                for sub_category, sub_details in details.items():
                    if isinstance(sub_details, dict):  # Check if
sub_details is a dictionary
                        for item in sub_details.get('items', []):
                            item_text = item['text']
                            # Проверяем наличие подстроки "[season]" в
тексте элемента
                            if '[season]' in item_text:
                                # Заменяем подстроку "[season]" на
нужный сезон
                                item_text =
item_text.replace('[season]', f'{season}')
```

```
                            # Заменяем '[City/Region]' на значение
country/city
                            if item_text.replace('[City/Region]',
f'{city}') == selected_item_text:
                                selected_category = category
                                selected_subcategory = sub_category
                                selected_item = item
                                break

            if selected_category and selected_subcategory and
selected_item:
                # Substitute [City/Region] with the desired city or
region
                city_region = f'{city}'
                gpt_prompt = selected_item.get('chatGPTPrompt', 'No
prompt available.')
                gpt_prompt = gpt_prompt.replace("[City/Region]",
city_region)

                if '[season]' in gpt_prompt.lower():
                    gpt_prompt = gpt_prompt.replace('[season]',
f'{season}')

                print(gpt_prompt)

                response = await ask_gpt(gpt_prompt)

                # Send the GPT prompt to the user
                await bot.send_message(message.from_user.id, response)
            else:
```

```
                await bot.send_message(message.from_user.id, "Item not
found")

        @dp.callback_query(F.data == 'travel_plan')
        async def travelPlan(callback_query: CallbackQuery):
            user_id = callback_query.from_user.id
            lang = db.get_lang(user_id)

            country = db.get_country(user_id)
            city = db.get_city(user_id)
            latitude, longitude =
gm.get_coordinates_from_address(country, city)

            trial = db.get_trial(user_id)
            paid_subscription = db.paid_subscription(user_id)

            if trial == 0 or paid_subscription == 1:
                if latitude is not None and longitude is not None:
                    await callback_query.message.answer(_("*It may
take a few minutes to generate a response. Please be patient*", lang),
parse_mode='Markdown')

                    restaurants = gm.search_restaurants(latitude,
longitude)
                    events = gm.search_events(latitude, longitude,
radius=10, size=5)

                    city_weather = await tp.weather(city, lang)
                    if city_weather:
                        await
callback_query.message.answer(city_weather, parse_mode='Markdown')
```

```
                    if restaurants:
                        morning_activities = await
tp.morningActivities(city, restaurants, lang)
                        if morning_activities:
                            await
callback_query.message.answer(morning_activities,
parse_mode='Markdown')

                        afternoon_activities = await
tp.afternoonActivities(city, restaurants, lang)
                        if afternoon_activities:
                            await
callback_query.message.answer(afternoon_activities,
parse_mode='Markdown')

                        dinner_activities = await
tp.eveningActivities(city, restaurants, lang)
                        if dinner_activities:
                            await
callback_query.message.answer(dinner_activities, parse_mode='Markdown')

                        night_activities = await
tp.nightActivities(events, lang)
                        if night_activities:
                            await
callback_query.message.answer(night_activities, parse_mode='Markdown')

                        travel_questions, travel_questions_second =
await tp.travelQuestions(city, country, lang)
```

```
                        if travel_questions and
travel_questions_second:
                            await
callback_query.message.answer(travel_questions, parse_mode='Markdown')
                            await
callback_query.message.answer(travel_questions_second,
parse_mode='Markdown')

                        miscellaneous_activities = await
tp.miscellaneous(city, restaurants, lang)
                        if miscellaneous_activities:
                            await
callback_query.message.answer(miscellaneous_activities,
parse_mode='Markdown')

                        await callback_query.message.answer(_("🎒 We
have prepared a detailed plan for your trip along with answers to
frequently asked questions.", lang), reply_markup=kb.main_menu(lang,
city))

                else:
                    print("Координаты не найдены")
            else:
                await callback_query.message.answer(_(f'Thank you for
using our Telegram bot! 🌟 To continue accessing our services, we invite
you to explore our subscription options:\n\n*🔒 Subscription:* Gain
full access to the bots functionality for a nominal fee.\n\n*💰 Price:*
$1 for 1 day of usage.\n\nTo subscribe, please follow the instructions
provided in the bot. Should you have any questions, our support team is
ready to assist you.\n\nWe appreciate your choice and look forward to a
```

```
long-term partnership!', lang), parse_mode='Markdown',
reply_markup=kb.buy_subscription(lang))


        @dp.callback_query(F.data == 'change_language')
        async def changeLanguage(callback_query:CallbackQuery):
            country = db.get_location(callback_query.from_user.id)
            city = db.get_city(callback_query.from_user.id)

            await callback_query.message.answer("📃 Select the desired
language:", reply_markup=kb.language)

        @dp.callback_query(F.data == 'change_location')
        async def changeLanguage(callback_query:CallbackQuery):
            user_id = callback_query.from_user.id
            lang = db.get_lang(user_id)

            await callback_query.message.answer(_("Click the button
below to share your location:"),
reply_markup=kb.create_location_button(lang))

        @dp.callback_query(F.data == 'support')
        async def changeLanguage(callback_query:CallbackQuery):
            user_id = callback_query.from_user.id
            lang = db.get_lang(user_id)

            await callback_query.message.answer(_("👨‍💼 If you have
any problems or questions, please contact our manager:
https://t.me/N0va22", lang), reply_markup=kb.back(lang))
```

```
        @dp.message(F.text == 'Back to Categories')
        async def back_to_menu(message:Message):
            user_id = message.from_user.id
            lang = db.get_lang(user_id)
            menu_data = load_menu_data(lang)

            # Remove the current keyboard
            await bot.send_message(message.chat.id, "Returning to the
main menu", reply_markup=ReplyKeyboardRemove())

            # Send the main menu keyboard
            await bot.send_message(message.chat.id, "Choose a
category:", reply_markup=generate_menu_buttons(menu_data['menus']))

        # ----------- back to categories ----------- #
        @dp.callback_query(F.data == 'back_to_categories')
        async def back_to_menu(callback:CallbackQuery):
            country = db.get_country(callback.from_user.id)
            city = db.get_city(callback.from_user.id)
            lang = db.get_lang(callback.from_user.id)

            await bot.edit_message_text(_(f'Welcome to the *Global
Travel* Assistant Bot!\n\nWhat would you like to explore today?',
lang), chat_id=callback.message.chat.id,
message_id=callback.message.message_id, reply_markup=kb.main_menu(lang,
city), parse_mode='Markdown')
        # ----------- back to categories ----------- #

        # ----------- back to menu ----------- #
        @dp.callback_query(F.data == 'back_to_menu')
        async def back_to_menu(callback:CallbackQuery):
```

```
            country = db.get_country(callback.from_user.id)
            city = db.get_city(callback.from_user.id)
            lang = db.get_lang(callback.from_user.id)

            await bot.edit_message_text(_(f'Welcome to the *Global
Travel* Assistant Bot!\n\nWhat would you like to explore today?',
lang), chat_id=callback.message.chat.id,
message_id=callback.message.message_id, reply_markup=kb.main_menu(lang,
city), parse_mode='Markdown')
        # ----------- back to menu ----------- #

        # ----------- payment ----------- #
        @dp.callback_query(F.data == 'buy_subscription')
        async def buySubscription(callback_query: CallbackQuery):
            user_id = callback_query.from_user.id
            payload =
callback_query.successful_payment.invoice_payload
            lang = db.get_lang(callback_query.from_user.id)

            country = db.get_country(callback_query.from_user.id)
            city = db.get_city(callback_query.from_user.id)

            await create_paypal_payment(callback_query, user_id, lang,
city, kb, db)
        # ----------- payment ----------- #
```