Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук

**Бакалаврська кваліфікаційна робота на тему:**

# РОЗРОБКА ЧАТ-БОТУ ДЛЯ ДОГЛЯДУ ЗА ДОМАШНІМ АКВАРІУМОМ ЧЕРЕЗ ПЛАТФОРМУ TELEGRAM

Виконав: студент 4 курсу, групи 2КН-21б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Романовський К.В.
(прізвище та ініціали)
Керівник: доктор філософії, асистент каф. КН
Перепелиця В.І.
(прізвище та ініціали)
«04» червня 2025 р.
Рецензент: к.т.н., доцент каф. САІТ
Варчук І. В.
(прізвище та ініціали)
«05» червня 2025 р.

Допущено до захисту
Зав. кафедри Яровий А.А.
«06» червня 2025 р.

Вінниця ВНТУ - 2025 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти перший (бакалаврський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 122 Комп'ютерні науки
Освітньо - професійна програма – Комп'ютерні науки

**ЗАТВЕРДЖУЮ**
Завідувач кафедри КН
проф., д.т.н. Яровий А.А.
«05» травня 2025 р.

# ЗАВДАННЯ
# НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Романовському Кирилу Володимировичу

1. Тема роботи: Розробка чат-боту для догляду за акваріумом через платформу Telegram.
керівник роботи: Перепелиця В'ячеслав Ігорович, асистент.
затверджені наказом вищого навчального закладу "20" березня 2025 року №97

2. Термін подання студентом роботи 30 травня 2025 р.

3. Вихідні дані до роботи: мова програмування – об'єктно-орієнтована, функціональна; кількість вкладеності тем – без обмежень; система має підтримувати обробку запитів користувача, збереження даних, планування нагадувань, графічне відображення параметрів, просту взаємодію через меню; інтерфейс – текстовий, структурований, інтуїтивно зрозумілий.

4. Зміст текстової частини (перелік питань, які потрібно розробити):
вступ; огляд і аналіз існуючих рішень; проектування чат-боту для догляду за домашнім акваріумом; розробка чат-боту для догляду за акваріумом; висновки; список використаних джерел; додатки.

5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень): схема роботи чат-боту; схема послідовності чат-боту; UML діаграма класів програми; Загальний алгоритм взаємодії з чат-ботом; приклад роботи програми.

6. Консультанти розділів роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| 1-3 | Перепелиця В.І., асистент | | |

7. Дата видачі завдання 05 травня 2025 року

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назви робіт | Термін Виконання | | Примітка |
|---|---|---|---|---|
| | | Початок | Закінчення | |
| 1 | Вступ. Огляд і аналіз існуючих рішень. | 05.05.25 | 07.05.25 | |
| 2 | Розробка чат-бота для для догляду за акваріумом через платформу Telegram | 08.05.25 | 11.05.25 | |
| 3 | Програмна реалізація чат-бота для для догляду за акваріумом через платформу Telegram | 12.05.25 | 15.05.25 | |
| 4 | Висновки та аналіз отриманих результатів | 16.05.25 | 26.05.25 | |
| 5 | Оформлення додатків | 27.05.25 | 29.05.25 | |
| 6 | Розробка інструкції користувача | 30.05.25 | 01.06.25 | |
| 7 | Оформлення матеріалів до захисту БКР | 02.06.25 | 04.06.25 | |

**Студент** _____ **Романовський К.В.**
(підпис) (прізвище та ініціали)

**Керівник роботи** _____ **Перепелиця В.І.**
(підпис) (прізвище та ініціали)

# АНОТАЦІЯ

Бакалаврська кваліфікаційна робота складається з 91 сторінки формату А4, на яких наведено 26 рисунків, 4 таблиці, список використаних джерел містить 23 найменувань.

Метою роботи є є розширення функціональних можливостей чат-ботів для догляду за акваріумом

У роботі обґрунтовано доцільність застосування Telegram-бота як зручного інструменту взаємодії користувача з сервісом догляду за акваріумом. Спроєктовано структуру та архітектуру системи з урахуванням вимог до функціональності, зокрема: ведення параметрів води з побудовою графіків, перевірка сумісності риб, надання відповідей на часті запитання, створення нагадувань, базова діагностика захворювань риб за описом симптомів.

Чат-бот реалізований мовою програмування Python із використанням бібліотеки aiogram та планувальника APScheduler, графіки побудовано з використанням matplotlib. Для діагностики хвороб риб інтегровано зовнішнє API. Тестування показало стабільну роботу бота, відповідність логіці обробки повідомлень і зручність взаємодії з користувачем.

Ключові слова: Telegram-бот, акваріум, Python, aiogram, акваріумістика, параметри води.

# ABSTRACT

The bachelor's qualification thesis consists of 91 A4 pages and includes 26 figures, 4 tables, and a list of 23 references.

The aim of the work is to expand the functional capabilities of chatbots for aquarium care.The thesis justifies the feasibility of using a Telegram bot as a convenient tool for user interaction with an aquarium maintenance service. The structure and architecture of the system have been designed according to the functional requirements, including: tracking water parameters with graph visualization, checking fish compatibility, providing answers to frequently asked questions, creating maintenance reminders, and performing basic diagnostics of fish diseases based on symptom descriptions.

The chatbot was implemented in Python using the aiogram library and the APScheduler scheduler; graphs were generated using matplotlib. An external API was integrated for fish disease diagnostics. Testing confirmed stable bot performance, proper response handling, and user-friendly interaction.



Keywords: Telegram bot, aquarium, Python, aiogram, aquaristics, water parameters.

**ЗМІСТ**

# ВСТУП

У сучасному світі цифрові технології стрімко змінюють підходи до вирішення повсякденних задач, проникаючи в різні сфери людської діяльності. Зростання кількості користувачів інтернету та мобільних пристроїв створює нові можливості для автоматизації процесів, підвищення зручності та ефективності. Однією з таких сфер є акваріумістика – галузь, що вимагає від власників акваріумів регулярного догляду, спеціалізованих знань і своєчасного виконання рутинних операцій. Чат-боти, як сучасний інструмент інтерактивної комунікації, стають дедалі популярнішими завдяки своїй простоті використання, швидкості реагування та можливості інтеграції з популярними платформами, такими як месенджери. Вони дозволяють автоматизувати задачі, надавати миттєвий доступ до інформації та покращувати досвід користувачів, що робить їх ідеальним рішенням для підтримки акваріумістів [1].

**Актуальність** створення чат-бота для догляду за домашнім акваріумом зумовлена високою потребою в ефективних інструментах, які полегшують догляд за акваріумними екосистемами. Сьогодні акваріумістика набуває популярності як хобі, однак вона пов’язана з низкою викликів: необхідністю регулярного моніторингу параметрів води, виконання процедур догляду та пошуку достовірної інформації про утримання риб. Чат-боти здатні вирішувати ці проблеми, надаючи користувачам цілодобовий доступ до порад, нагадувань і рекомендацій. Платформа Telegram, яка налічує сотні мільйонів активних користувачів, вирізняється зручним інтерфейсом, підтримкою ботів і кросплатформністю, що робить її оптимальним вибором для реалізації такого рішення. Розробка чат-бота для акваріумістики не лише спрощує догляд за акваріумами, але й сприяє популяризації цифрових технологій у цій сфері, підвищуючи рівень обізнаності та комфорту користувачів.

**Метою бакалаврської кваліфікаційної роботи** є розширення функціоналинх можливостей чат-боту для догляду за домашнім акваріумом, що забезпечить зручний доступ до інформації, автоматизацію рутинних задач і персоналізовану взаємодію для акваріумістів різного рівня досвіду.

**Об'єктом дослідження** є процеси автоматизації взаємодії з користувачами в контексті догляду за акваріумами.

**Предметом дослідження** є методи та засоби створення чат-боту на платформі Telegram для надання інформації та автоматизації задач, пов'язаних з акваріумістикою.

Для досягнення цієї мети визначено такі **задачі дослідження**:

1. Обґрунтувати доцільність розробки чат-боту для догляду за домашнім акваріумом.

2. Спроектувати чат-бот для догляду за домашнім акваріумом.

3. Програмно реалізувати  чат-бот для догляду за домашнім акваріумом через платформу Telegram.

4. Виконати тестування програми та провести аналіз отриманих результатів.

5. Розробити інструкцію користувача.

**Практична цінність** роботи полягає в створенні чат-бота, який полегшить догляд за акваріумами, надаючи користувачам зручний інструмент для отримання інформації, планування задач і підтримки здоров'я акваріумних екосистем. Розроблене рішення сприятиме підвищенню ефективності догляду, зменшенню помилок і популяризації акваріумістики серед широкої аудиторії.

**Апробація роботи.**

Результати досліджень було апробовано на LIV Всеукраїнській науково-технічній конференції підрозділів Вінницького національного технічного університету (НТКП ВНТУ) м. Вінниці у 2025 р. [1].

**Публікації:** за основними результатами досліджень опубліковано тези доповіді на науково-технічній конференції [1].

# 1 ОГЛЯД І АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

## 1.1 Огляд існуючих чат-ботів для догляду за тваринами та рослинами

Цифрові технології відіграють дедалі більшу роль у спрощенні повсякденних завдань, включаючи догляд за тваринами, рослинами та акваріумами. Чат-боти, як інтерактивні віртуальні помічники, стали популярним інструментом завдяки здатності надавати миттєвий доступ до інформації, автоматизувати рутинні завдання та пропонувати персоналізовані рекомендації. У сфері акваріумістики чат-боти можуть допомагати підтримувати оптимальні умови для риб, рослин і обладнання, забезпечуючи своєчасні нагадування про годування, заміну води чи чищення фільтрів, а також надаючи поради щодо сумісності видів чи діагностики проблем. Огляд наявних рішень дозволяє оцінити їхні можливості, виявити сильні та слабкі сторони, а також визначити прогалини, які може заповнити чат-бот, розроблений для догляду за акваріумами [1].

Чат-боти для догляду за тваринами та рослинами різноманітні за функціоналом і платформами. Деякі з них зосереджені на наданні інформаційних довідок, наприклад, як вибрати корм для кота чи які умови потрібні для вирощування орхідеї. Інші автоматизують рутинні задачі, надсилаючи нагадування про полив чи чищення акваріума. Є також боти, що пропонують діагностичні функції, аналізуючи симптоми хвороб на основі описів чи фотографій. Наприклад, акваріуміст може написати боту: "Моя риба плаває боком", і отримати пораду перевірити рівень аміаку у воді. Такі можливості роблять чат-ботів незамінними для тих, хто прагне спростити догляд за живими організмами, але не завжди має достатньо знань чи часу. Популярність цих інструментів зростає: за даними дослідження, опублікованого в Journal of Veterinary Behavior (2020), близько 60% власників домашніх тварин у США використовували цифрові помічники для догляду за

улюбленцями, і ця тенденція поширюється на інші країни, включаючи Україну [2, 3].

Серед чат-ботів для догляду за тваринами виділяється PetCoach. Цей бот пропонує консультації з ветеринарами, відповідаючи на запитання про здоров’я собак, котів чи птахів. Наприклад, на запит "Чому мій пес чухається?" бот може відповісти, що це може бути алергія, порадивши звернутися до лікаря або змінити шампунь. PetCoach також надсилає нагадування про щеплення чи візити до клініки, що особливо корисно для зайнятих власників. Його сильна сторона – інтерактивність і можливість підключення до реальних фахівців через чат, що забезпечує високий рівень довіри. Проте бот не підтримує українську мову, а його функціонал обмежений домашніми тваринами, ігноруючи рослини чи акваріуми. За оцінками, PetCoach має десятки тисяч користувачів, але точна кількість залежить від платформи, і в Україні його використання обмежене через мовний бар’єр. Головна сторінка проекту наведена на рисунку 1.1.

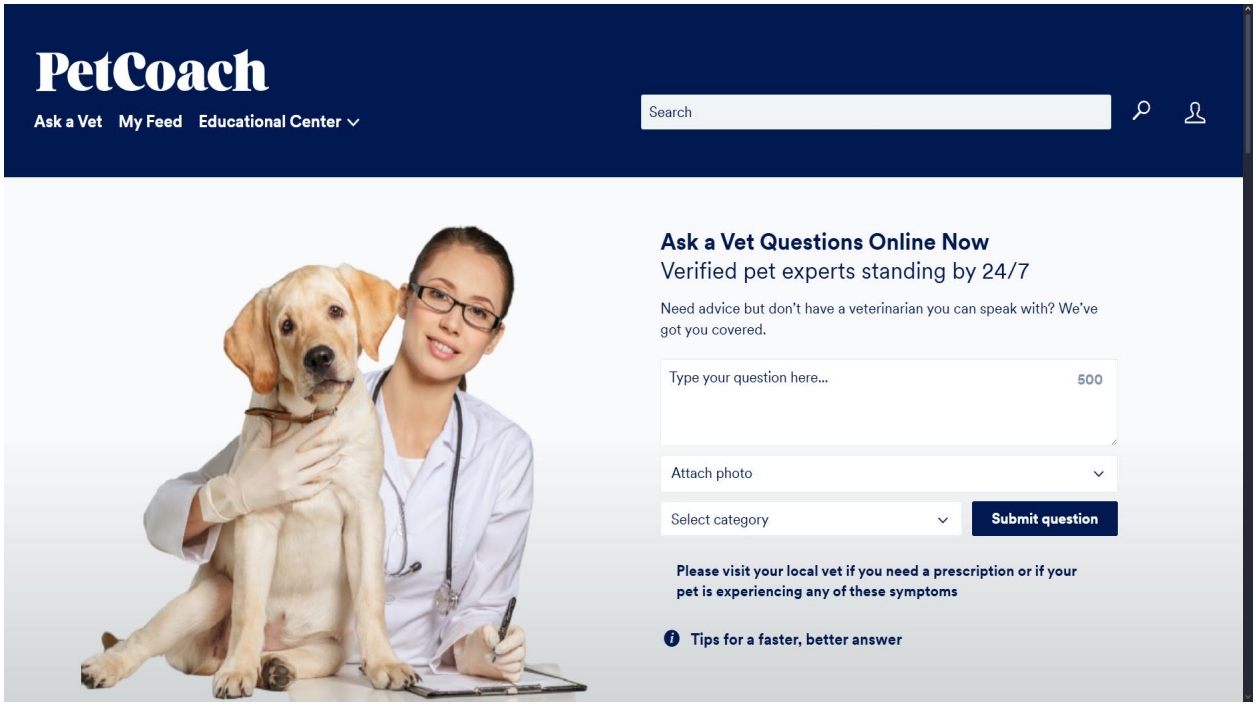


Рисунок 1.1 – Сторінка проекту PetCoach

Іншим прикладом є VetBot — чат-бот, орієнтований на діагностику симптомів у домашніх тварин. Користувач вводить опис стану, наприклад: «У собаки кашель і температура», після чого бот аналізує повідомлення, пропонує можливі причини, як-от респіраторна інфекція, і надає базові рекомендації щодо подальших дій.

VetBot використовує технології обробки природної мови (NLP) і спирається на базу з ветеринарними порадами. Він має доступ до структурованої бази даних із перевіреними ветеринарними порадами та стандартами лікування. Серед його переваг — відносно висока точність попередньої діагностики та інтеграція з месенджером WhatsApp, який у 2024 році мав понад 2 мільярди користувачів.

Водночас VetBot має й обмеження: він не підтримує українську мову, не охоплює акваріуми чи рослини, а для використання через WhatsApp Business API потрібна платна підписка, що ускладнює доступ для невеликих проєктів.

У сфері догляду за рослинами популярним є Plant Doctor, бот у Telegram, а також додаток, який допомагає діагностувати проблеми з рослинами за фотографіями. Користувач може надіслати фото листя з плямами чи пожовтінням, і бот, використовуючи технології комп’ютерного зору, запропонує рішення, наприклад, обробку фунгіцидом при грибковій інфекції. Plant Doctor має базу даних із сотнями видів рослин, що робить його корисним для садівників. Наприклад, на запит про пожовтіння листя фікуса бот може вказати на надмірний полив і порадити скоротити його до одного разу на тиждень. Проте ефективність бота залежить від якості фотографій, а відсутність функцій нагадувань чи підтримки інших сфер, як-от догляд за тваринами чи акваріумами, обмежує його універсальність. Plant Doctor має тисячі користувачів у Telegram, але точна статистика недоступна через закритий характер даних. Інтерфейс даної програми наведений на рисунку 1.2.


Рисунок 1.2 – Інтерфейс програми від проекту Plant Doctor

Ще одне рішення – Garden Helper, мобільний додаток для iOS і Android, який дозволяє створювати графіки поливу та підгодівлі рослин. Користувачі можуть налаштувати частоту поливу, наприклад, кожні 3 дні для фікуса, і отримувати push-сповіщення. Додаток містить базу даних із характеристиками

різних видів рослин, наприклад, вимоги до освітлення для орхідей (8–10 годин розсіяного світла). Garden Helper має близько 50 тисяч завантажень, за даними Google Play, і є популярним серед садівників-початківців. Однак він не інтегрований із месенджерами, що знижує його доступність порівняно з чат-ботами, а також не підтримує функції для догляду за тваринами чи акваріумами.

У сфері акваріумістики чат-боти менш поширені, але деякі рішення вже існують. Fish Care Bot у Telegram пропонує базові поради з догляду за акваріумами, наприклад, рекомендації щодо температури води для гупі (24–26°C) чи частоти заміни води (30% щотижня). Бот також надсилає нагадування про чищення фільтрів, що допомагає уникнути накопичення шкідливих речовин, як-от нітрати (<40 мг/л). Запущений у 2020 році, Fish Care Bot має близько 12 тисяч активних користувачів, за оцінками розробників. Його переваги – безкоштовність і простота використання через Telegram (900 млн користувачів у 2024 році), але база знань обмежена 50 популярними видами риб, а функції діагностики чи аналізу параметрів води відсутні. Наприклад, на запит "Чому риба плаває боком?" бот може порадити перевірити рівень аміаку, але не здатен обробляти складніші описи чи фото.

Інший приклад – AquaBot у Viber, який зосереджений на інформаційних довідках про сумісність риб і акваріумні рослини. На запит "Чи можна тримати гупі з барбусами?" бот відповідає, що це небажано через агресивність барбусів, і рекомендує pH 6.8–7.8 та температуру 24–26°C. AquaBot містить базу даних із 100 видів риб і 30 типів рослин, що робить його корисним для планування акваріума. Запущений у 2021 році, бот має ~6 тисяч користувачів, але не підтримує нагадувань чи діагностики, а платний API Viber (~€4500/рік) ускладнює оновлення. Крім того, Viber менш популярний серед акваріумістів порівняно з Telegram.

Порівняння цих рішень показує їхні сильні сторони. PetCoach вирізняється інтерактивністю та можливістю консультацій із фахівцями, що

забезпечує високий рівень довіри серед власників тварин. Plant Doctor демонструє потенціал технологій комп'ютерного зору, що особливо корисно для діагностики рослин. Fish Care Bot і AquaBot пропонують спеціалізовану підтримку для акваріумістів, що є рідкістю серед чат-ботів. Garden Helper, хоч і не чат-бот, забезпечує зручні нагадування для садівників. Проте всі ці рішення мають спільні недоліки, які обмежують їхню універсальність. По-перше, вони зосереджені на вузьких сферах: PetCoach – на тваринах, Plant Doctor – на рослинах, Fish Care Bot і AquaBot – на акваріумах. По-друге, мовна підтримка часто обмежена англійською, що створює бар'єри для україномовних користувачів. По-третє, навіть просунуті боти, як Plant Doctor, залежать від зовнішніх факторів, таких як якість інтернет-з'єднання чи чіткість фотографій, що може знижувати їхню ефективність. Нарешті, жоден із цих ботів не пропонує комплексного підходу, який би поєднував інформаційні довідки, нагадування та діагностику в одному рішенні, особливо для акваріумістики, де точність і своєчасність догляду критично важливі.

Вплив чат-ботів на досвід користувачів є значним. Власники тварин і рослин цінують можливість швидко отримати відповідь, не гортаючи книги чи форуми. Наприклад, акваріуміст, який помічає млявість риби, може звернутися до бота, описати симптоми й отримати пораду за лічені секунди, що економить час і зменшує стрес. Дослідження, опубліковане в Computers in Human Behavior (2021), показує, що 70% користувачів чат-ботів для догляду за тваринами відзначають підвищення впевненості у своїх діях завдяки швидким і зрозумілим рекомендаціям. У рослинництві боти, як Garden Helper, зменшують загибель рослин на 35% завдяки нагадуванням про полив. Для акваріумістики такі показники ще більш значущі, адже затримка в заміні води чи чищенні фільтра може призвести до отруєння риб аміаком чи нітратами [4].

Чат-боти мають значні переваги завдяки інтеграції з популярними месенджерами, які стали невід'ємною частиною повсякденного цифрового життя. Одним із найпоширеніших є Telegram, який у 2024 році налічував

близько 900 мільйонів активних користувачів щомісяця. Ця платформа пропонує зручне середовище для створення ботів із підтримкою інтерактивних елементів: текстових повідомлень, інлайн-кнопок, меню, а також мультимедійного контенту.

Завдяки такій інтеграції чат-боти стають доступними широкій аудиторії без необхідності встановлення окремого додатку, що особливо зручно для початківців. Водночас, існуючі рішення мають низку обмежень: більшість ботів не підтримують кілька мов або локалізацію, не мають адаптації під індивідуальні потреби користувача.

У контексті акваріумістики ці обмеження стають ще помітнішими. Такі боти, як Fish Care Bot або AquaBot, хоча й корисні на базовому рівні, не здатні забезпечити всебічний підхід до догляду за акваріумом. Вони не охоплюють усі важливі аспекти — від моніторингу параметрів води. Таким чином, залишається запит на більш комплексні, інтелектуальні рішення, здатні виконувати роль персонального помічника акваріуміста.

Очікується, що розвиток чат-ботів у сфері догляду за тваринами, рослинами та акваріумами продовжиться. Інтеграція з апаратними пристроями, як-от сенсорами для моніторингу параметрів води (pH, аміак), може забезпечити автоматичне отримання даних і надання рекомендацій. Наприклад, сенсор Seneye Home міг би передавати дані про pH 7.8, а бот пропонував би додати торф для зниження показника. Використання штучного інтелекту, як-от моделі GPT-4, дозволить чат-ботам обробляти складні запити, наприклад, пояснювати причини млявості риб (перенаселення, низький рівень кисню) чи пропонувати методи боротьби з водоростями, як-от скорочення світлового дня до 8 годин. Комплексний чат-бот, який поєднує інформаційні довідки, нагадування, діагностику та підтримку української мови, може заповнити прогалини наявних рішень, підвищуючи ефективність догляду за акваріумами та популяризуючи це хобі [3]. Графік використання чат ботів наведений на рисунку 1.3.


Рисунок 1.3 – Графік використання чат-ботів для догляду за тваринами, рослинами та акваріумами.

Таблиця 1.1 – Порівняльна таблиця чат-ботів для догляду за тваринами, рослинами та акваріумами.

| **Назва бота** | **Платформа** | **Основні функції** | **Обмеження** |
|---|---|---|---|
| PetCoach | Telegram, Facebook Messenger | Консультації з ветеринарами, нагадування | Немає української мови, лише тварини |
| VetBot | WhatsApp | Діагностика симптомів, нагадування | Платний API, без акваріумів |
| Plant Doctor | Telegram | Діагностика за фото, довідки | Залежність від якості фото, без нагадувань |
| Garden Helper | iOS, Android | Графіки поливу, база рослин | Без месенджерів, лише рослини |
| Fish Care Bot | Telegram | Поради, нагадування | Обмежена база, без діагностики |
| AquaBot | Viber | Сумісність риб, довідки | Платний API, без нагадувань |

Огляд наявних чат-ботів та їхнє порівняння у таблиці 1.1. показує, що хоча інструменти для догляду за тваринами та рослинами активно розвиваються, спеціалізовані рішення для акваріумістики ще не досягли

повного потенціалу. Комплексний чат-бот, який поєднує інформаційні довідки, нагадування, діагностику та підтримку української мови, може стати значним кроком уперед, пропонуючи всебічну підтримку для акваріумістів і сприяючи популяризації цього хобі.

### 1.2 Платформи для розробки чат-ботів

Чат-боти є важливим інструментом для автоматизації завдань, зокрема у сфері догляду за акваріумами, де вони можуть надавати інформаційні довідки, нагадування та рекомендації для підтримання здоров’я акваріумної екосистеми. Успіх чат-бота залежить від платформи, яка визначає його доступність, зручність і охоплення аудиторії. Найпоширенішими платформами є месенджери, такі як Telegram, WhatsApp, Viber, Facebook Messenger і WeChat, а також конструктори чат-ботів, як SendPulse, ManyChat і Chatfuel. Ці платформи дозволяють створювати боти з різним рівнем складності, від простих інформаційних до тих, що використовують штучний інтелект для обробки запитів. Для чат-бота, призначеного для догляду за акваріумами, важливо обрати платформу, яка забезпечує інтерактивність, підтримку української мови та можливість інтеграції мультимедійного контенту, як-от фото чи відеоінструкції з чищення акваріума.

Месенджери є основними платформами для впровадження чат-ботів завдяки їхній популярності та інтеграції в повсякденне життя. Зростання мобільного трафіку на 55% у 2023 році порівняно з 2018-м, за даними звіту State of Mobile 2024 від Data.ai, і середній час, проведений у месенджерах (2 години 15 хвилин на день), роблять їх ідеальними для швидкого доступу до інформації. Telegram, із 900 мільйонами активних користувачів у 2024 році, за даними Statista, вирізняється простотою створення чат-ботів через Bot API, який є безкоштовним і дозволяє навіть розробникам-початківцям створювати функціональні програми. Наприклад, чат-бот для акваріумістики може

надсилати нагадування про заміну 30% води щотижня, відповідати на запитання типу "Який pH потрібен для гупі?" (6.8–7.8) або демонструвати відео з правильного чищення фільтра. Telegram підтримує мультимедійний контент, включаючи фото, відео та документи, що ідеально для надання інструкцій із догляду за акваріумами. Його популярність в Україні (~20 мільйонів користувачів) і підтримка української мови роблять Telegram оптимальним вибором для локальних проєктів.

WhatsApp, із 2 мільярдами користувачів у 2024 році, є лідером за аудиторією і підтримує бізнес-боти через WhatsApp Business API. Ця платформа дозволяє створювати автоматизовані відповіді, наприклад, для консультацій про догляд за акваріумами чи замовлення кормів у зоомагазинах. Однак висока вартість API (від $50 на місяць залежно від регіону) і суворі правила модерації роблять WhatsApp менш доступним для невеликих проєктів. WeChat, із 1.3 мільярда користувачів, переважно в Китаї, є універсальною платформою, яка поєднує месенджер, соціальну мережу та платіжну систему. Чат-боти в WeChat можуть інтегруватися з базами даних, наприклад, для підбору акваріумних рослин, але їхня розробка вимагає знання китайського ринку та підтримки китайської мови, що обмежує використання для україномовної аудиторії. Facebook Messenger, із 988 мільйонами користувачів, пропонує гнучкий API для створення ботів із кнопками, каруселями зображень чи відеоінструкціями, наприклад, про годування риб. Проте в Україні Messenger менш популярний порівняно з Telegram чи Viber, що знижує його привабливість для локальних проєктів. Viber, із 400 мільйонами користувачів, також популярний в Україні, але його API для чат-ботів став платним із 2024 року (від €4500 на рік для бізнес-акаунтів), а менша гнучкість порівняно з Telegram (обмежена підтримка складних інтерактивних елементів, як каруселі) робить його менш привабливим [5]. Огляд популярності платформ наведений на рисунку 1.4.


Рисунок 1.4 – Графік популярності месенджерів для чат-ботів у 2024 році.

Окрім месенджерів, важливу роль відіграють конструктори чат-ботів, які дозволяють створювати боти без глибоких знань програмування. SendPulse є однією з найпопулярніших платформ, підтримуючи створення ботів для Telegram, WhatsApp, Viber і Facebook Messenger. За допомогою SendPulse можна налаштувати чат-бот для акваріумістики, який надсилатиме нагадування про чищення фільтрів кожні два тижні чи відповідатиме на запитання, як-от "Чому вода в акваріумі каламутна?" (можлива причина - надлишок корму). Безкоштовний тариф SendPulse дозволяє надсилати до 10 тисяч повідомлень на місяць, що достатньо для тестування чи невеликого проєкту, але розширені функції, як-от інтеграція з платіжними системами чи AI-моделями, вимагають підписки від $10 на місяць. Платформа вирізняється простотою: візуальний конструктор дозволяє створювати сценарії взаємодії, перетягуючи блоки, що ідеально для розробників без досвіду програмування. ManyChat, орієнтований на маркетинг, підходить для створення простих

інформаційних ботів, які надають довідки про сумісність акваріумних риб чи нагадують про внесення добрив для рослин.

ManyChat підтримує Telegram і Facebook Messenger, але його функціонал обмежений для складних завдань, як-от діагностика проблем в акваріумі. Вартість починається від $15 на місяць, хоча безкоштовний тариф дозволяє протестувати базові функції. Chatfuel – ще один конструктор, який підтримує Telegram і Messenger, дозволяючи створювати боти з інтерактивними меню для вибору категорій, наприклад, "Догляд за рибами" чи "Параметри води". Безкоштовний тариф Chatfuel обмежений 50 користувачами, а платний коштує від $14.99 на місяць, що може бути бар'єром для невеликих проєктів.

Для створення чат-бота з розширеним функціоналом, як-от обробка складних запитів чи інтеграція з базами даних, доцільніше використовувати програмування на Python із бібліотекою python-telegram-bot. Ця бібліотека є безкоштовною, підтримує асинхронну обробку запитів і дозволяє створювати боти з гнучкими сценаріями, наприклад, відповідати на запит "Яка температура для неонів?" (22–26°C) чи надсилати нагадування про перевірку рівня нітратів (<40 мг/л). Python також дозволяє інтегрувати базу даних, як SQLite, для зберігання інформації про 100+ видів риб, їхні вимоги до pH, жорсткості води чи сумісності. Наприклад, бот може повідомити, що гупі не варто тримати з барбусами через їхню агресивність. Крім того, Python забезпечує можливість підключення до API штучного інтелекту, як-от OpenRouter API, для обробки складних запитів, наприклад, аналізу причин каламутності води (надлишок корму чи забруднення фільтра). Такий підхід вимагає хостингу, наприклад, на Heroku чи AWS, і базових навичок програмування, але надає повний контроль над функціоналом бота, що є перевагою порівняно з конструкторами [6].

Порівняння платформ показує, що Telegram є оптимальним вибором для чат-бота з догляду за акваріумами завдяки великій аудиторії в Україні,

безкоштовному API та підтримці мультимедійного контенту, як-от відеоінструкції з чищення акваріума. Конструктори, як SendPulse, спрощують створення бота для початківців, але обмежені у складних функціях, таких як діагностика проблем в акваріумі. Python із python-telegram-bot забезпечує гнучкість і масштабованість, дозволяючи реалізувати інформаційні довідки, нагадування та інтеграцію з AI. Обмеження всіх платформ включають залежність від стабільного інтернет-з'єднання та потребу в регулярному оновленні контенту, наприклад, бази знань про акваріумні рослини чи риб. Для україномовного проєкту важлива підтримка української мови, яку забезпечують Telegram і SendPulse, тоді як WhatsApp і WeChat менш гнучкі через мовні бар'єри чи платний доступ.

Огляд платформ показує, що Telegram у поєднанні з Python і python-telegram-bot є найкращим вибором для створення чат-бота для догляду за акваріумами завдяки доступності, підтримці української мови та гнучкості. Конструктори, як SendPulse, підходять для швидкого створення, але для складних функцій, як-от діагностика чи інтеграція з AI, Python забезпечує більше можливостей, що робить його оптимальним для реалізації універсального інструменту для акваріумістів.

### 1.3 Аналіз рішень у сфері догляду за акваріумами

У сучасному світі цифрові технології дедалі глибше проникають у повсякденне життя, і навіть такі сфери, як догляд за домашніми акваріумами, не залишаються осторонь цієї тенденції. Якщо раніше утримання акваріума було повністю вручну контрольованим процесом, що вимагав уважності, досвіду та значного часу, то сьогодні акваріумісти, як початківці, так і професіонали, мають змогу значно полегшити цей процес за допомогою програмних рішень. Застосування мобільних додатків і чат-ботів дозволяє автоматизувати рутинні процедури, зберігати історію обслуговування, отримувати підказки, а також своєчасно реагувати на зміни у стані водного

середовища. Це особливо актуально у випадках, коли власник має кілька акваріумів, зайнятий графік або недостатньо досвіду в обслуговуванні.

Цифрові інструменти стають невід'ємною частиною так званих «розумних» екосистем, і акваріумістика не є винятком. Відповідно до даних галузевих досліджень, глобальний ринок інтелектуальних акваріумних технологій продовжує зростати: у 2024 році він оцінювався приблизно в 1,6 мільярда доларів США, і згідно з прогнозами, має досягти 4 мільярдів до 2033 року. Це свідчить про інтенсивний розвиток попиту на технологічні рішення у сфері догляду за декоративними водними екосистемами – як для приватних домогосподарств, так і для комерційних закладів, що використовують акваріуми у своїй діяльності.

Однією з найпоширеніших категорій програм є так звані «менеджери акваріума» – спеціалізовані мобільні застосунки, що дозволяють користувачеві вести журнал усіх важливих подій, пов'язаних із доглядом за акваріумом. Наприклад, додаток Aquarium Log, розроблений для платформи Android, є зразком інтуїтивно зрозумілого та зручного інструмента для ведення календаря завдань, щоденника параметрів та нагадувань. Завдяки можливості реєстрації параметрів води – таких як pH, температура, рівень нітратів тощо – користувач отримує змогу візуалізувати ці дані у вигляді графіків. Такий підхід дозволяє не лише фіксувати зміни, а й виявляти закономірності, що можуть бути ознаками прихованих проблем в акваріумі [7]. Крім того, Aquarium Log підтримує одночасне обслуговування декількох акваріумів, що особливо зручно для досвідчених акваріумістів або комерційних власників (наприклад, у зоомагазинах). Дані синхронізуються з хмарним сховищем, тож користувач не ризикує втратити важливу інформацію навіть у разі зміни пристрою. Варто зазначити, що цей додаток регулярно оновлюється, а висока оцінка користувачів у магазині Google Play (4.8 з 5) свідчить про його якість і затребуваність. Інтерфейс програми Aquatic Log наведений на рисунку 1.5.

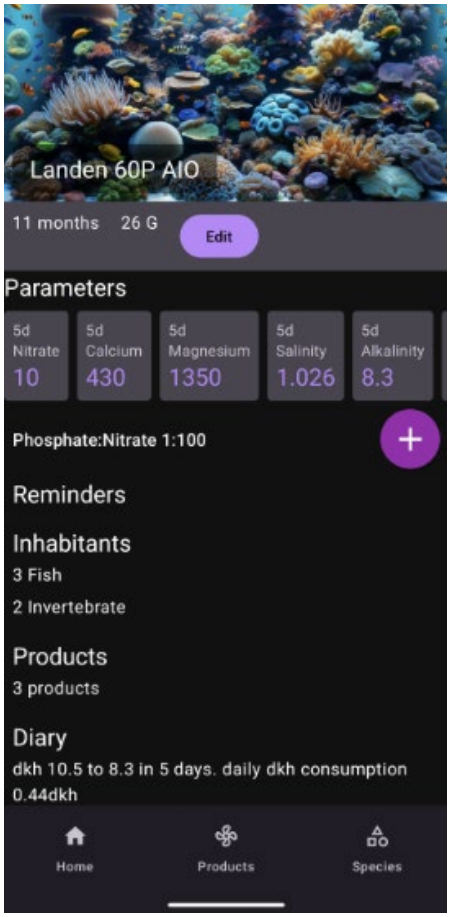


Рисунок 1.5 – Інтерфейс програми Aquatic Log

Ще більш функціонально насиченим є додаток Aquarimate, доступний для обох основних мобільних платформ – iOS і Android. Його функціонал значно розширений порівняно зі стандартними журналами: окрім фіксації параметрів, програма включає об’ємну енциклопедію риб, коралів і водних рослин із вказаними умовами утримання, рівнями агресивності, сумісності між видами тощо. У додатку також передбачено корисні інструменти – такі як конвертери одиниць виміру, розрахунки концентрацій та інтеграція з хмарним сервісом для збереження інформації. Aquarimate дозволяє аналізувати дані у вигляді графіків, що корисно як для рутинного спостереження, так і в разі виникнення проблем. Однак, попри велику кількість функцій, програма вимагає одноразової покупки або оформлення підписки, що може бути бар’єром для деяких користувачів. Ще одним недоліком є обмеженість мовної

підтримки – зокрема, відсутність українського інтерфейсу, що актуально для локального ринку [8, 9].

Інший представник цієї категорії – Aqua-Info, додаток для iOS, який, попри свою простоту, містить значну офлайн-базу даних про різноманітні водні організми та хвороби риб. Aqua-Info дозволяє планувати регулярне обслуговування, включаючи підміни води, дозування добрив та фільтрацію. Для новачків програма може стати справжнім помічником, оскільки містить описові рекомендації та попередження щодо потенційних ризиків, пов’язаних із недотриманням графіка обслуговування. Однак, як і попередні програми, Aqua-Info наразі не має підтримки української мови, що може ускладнити її використання деякими категоріями користувачів.

Цікавою нішевою розробкою у сфері акваріумістики є AI Reef Cam — мобільний додаток, орієнтований на фотоаналіз стану акваріумів. Програма розроблена для пристроїв на платформі iOS і використовує штучний інтелект, зокрема технології Apple Core ML, для розпізнавання понад 350 видів морських риб, що дозволяє акваріумістам швидко ідентифікувати мешканців свого акваріума за допомогою камери.

Окрім цього, AI Reef Cam може аналізувати зображення для вимірювання інтенсивності освітлення в одиницях PAR (photosynthetically active radiation), що є важливим параметром для росту коралів. Також додаток має функцію корекції кольорової температури знімків, що робить його корисним не лише з естетичної точки зору (наприклад, для публікацій у соціальних мережах), а й для точного налаштування світлового середовища в акваріумі.

Проте, незважаючи на інноваційні функції, AI Reef Cam залишається доступним лише для користувачів iOS і орієнтований насамперед на власників морських (рифових) акваріумів, які потребують точного контролю умов для утримання коралів та специфічних морських видів.

Окрему увагу серед сучасних рішень заслуговують додатки з елементами соціальної взаємодії. Яскравим прикладом є Aquabuildr — багатофункціональна платформа, яка поєднує менеджер акваріуму, базу даних риб, соціальну мережу для акваріумістів та інтеграцію з чат-ботом, що працює на основі штучного інтелекту.

Завдяки цьому додатку користувачі можуть не лише вести облік мешканців акваріуму та параметрів води, а й ділитися фотографіями, ставити запитання спільноті, отримувати поради та рекомендації. Однією з ключових функцій є автоматичний підбір сумісних риб за допомогою запатентованого алгоритму, що враховує видову сумісність, розмір, поведінку та інші фактори.

Особливо корисною є функція вбудованого чат-асистента — інтелектуального помічника, здатного відповідати на запити про якість води, сумісність мешканців, підбір обладнання чи планування нових акваріумів. Проте варто враховувати, що безкоштовна версія має низку обмежень, зокрема на довжину або кількість запитів до AI-помічника. Для розблокування повного функціоналу користувачеві пропонується оформити підписку. Інтерфейс на рисунку 1.6.

З розвитком нейронних мереж, чат-боти стали ще одним важливим напрямом у сфері цифрової підтримки акваріумістів. Поява чат-ботів, створених на базі великих мовних моделей (таких як GPT-4), зробила можливим створення віртуальних помічників, які можуть надавати вичерпні відповіді на широкий спектр запитів. Наприклад, такі боти, як Aquarium Advisor та Aquarium Buddy, дозволяють у зручному форматі діалогу отримувати рекомендації щодо налаштування нового акваріума, вибору фільтрів, систем освітлення, визначення симптомів хвороб у риб та їх лікування, а також оптимальних параметрів води.

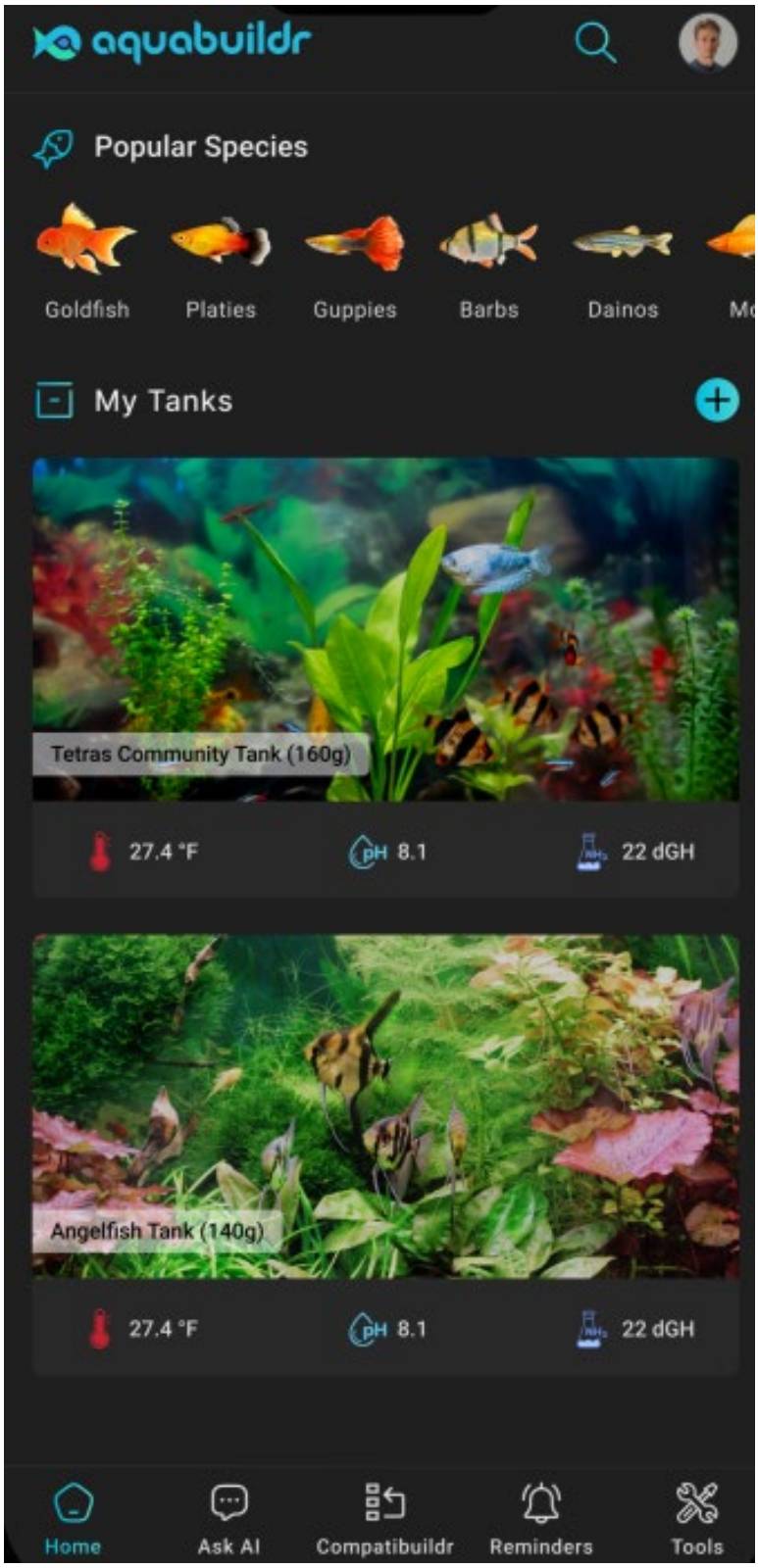


Рисунок 1.6 – Інтерфейс програми Aquabuildr

Основна перевага таких рішень – їхня інтерактивність: користувач не просто читає статтю, а ставить запитання та отримує персоналізовану

відповідь, адаптовану до його ситуації. Порівняння функціоналу готових рішень наведено у таблиці 1.2.

Таблиця 1.2 – порівняння функціональних можливостей кількох популярних акваріумних програм

| **Назва застосунку** | **Платформи** | **Основні функції** | **Наявність штучного інтелекту** | **Вартість** | **Підтримка української мови** |
|---|---|---|---|---|---|
| **AquaPlanner Pro** | iOS, Android | Облік параметрів води, нагадування, калькулятор дозування | Немає | Безкоштовно / $4.99 | Немає |
| **Aquarimate** | iOS, Android | Графіки, журнали, база риб і рослин, хмарна синхронізація | Немає | Безкоштовно / $9.99 | Немає |
| **Fishkeeper App** | Android | Планувальник, сповіщення, база даних хвороб, фото-альбоми | Частково (пошук по фото) | Безкоштовно | Немає |
| **Aquarium Note** | Android | Повний облік, мультиакваріуми, планування годування | Немає | Безкоштовно / наявність реклами | Немає |
| **MyAquarium Assistant** | iOS | Статистика параметрів, поради з утримання, нагадування | Обмежене (вбудований FAQ) | Безкоштовно | Немає |

Проте, незважаючи на очевидні переваги, існують і певні обмеження: більшість таких сервісів наразі орієнтовані на англомовну аудиторію, тож українським користувачам може бути складно взаємодіяти з ними без належного рівня володіння мовою. Локалізація, зокрема підтримка української, залишається обмеженою або відсутньою, що знижує доступність і зручність використання. Крім того, чат-боти здебільшого вимагають постійного доступу до інтернету, а функції безкоштовного доступу часто мають обмеження у кількості звернень або функціоналі. Це може створювати бар’єри для користувачів у регіонах із нестабільним з’єднанням або для тих, хто шукає повноцінний функціонал без додаткових витрат.

## 1.4 Постановка задачі для розробки чат-боту для догляду за акваріумом

Запропонована задача передбачає створення чат-бота в месенджері Telegram, який надаватиме користувачам підтримку в догляді за акваріумами. Необхідно забезпечити зручне зберігання та видачу інформації про риб, рослини, параметри води та обладнання, а також автоматизацію рутинних завдань, таких як нагадування про заміну води чи чищення фільтрів. Для реалізації слід оцінити наявні ресурси, розробити систему та інтегрувати чат-бота в Telegram.

Чат-бот повинен відповідати наступним вимогам:

- Інтеграція з месенджером Telegram для зручного доступу.
- Обробка текстових запитів для інтуїтивної взаємодії.
- Можливість розширення бази знань для включення нових даних про акваріуми.

Для створення чат-бота буде використано мову програмування Python із бібліотекою python-telegram-bot, яка забезпечує інтеграцію з Telegram через Bot API. Ця бібліотека дозволяє обробляти текстові повідомлення, наприклад, відповідати на запити про оптимальну температуру для гупі чи надсилати нагадування про перевірку рівня нітратів, і підтримує розширення функціоналу, наприклад, додавання бази даних про акваріумні види.

Проведений аналіз показав, що чат-боти можуть бути універсальними інструментами, інтегрованими в різні сервіси, зокрема месенджери, які є популярними через зростання мобільного трафіку. Telegram вирізняється великою аудиторією в Україні та технічною гнучкістю, що робить його оптимальним для реалізації чат-бота. Наявні рішення для догляду за акваріумами, такі як апаратні пристрої чи додатки, мають обмеження, зокрема високу вартість або відсутність інтерактивних функцій. Розробка чат-бота на базі Python і Telegram дозволить створити доступний інструмент, який поєднує

інформаційну підтримку та автоматизацію. Очікується, що чат-бот сприятиме спрощенню догляду за акваріумами, надаючи користувачам можливість отримувати довідки, наприклад, про сумісність риб, та нагадування про чищення акваріума, що підвищить ефективність утримання акваріумів і популяризуватиме акваріумістику.

### 1.5 Висновок до розділу 1

У цьому розділі було обгрунтовано доцільність розробки чат-боту для догляду за домашнім акваріумом, розглянуто сучасні підходи до створення чат-ботів, зокрема порівняно можливості різних платформ, таких як месенджери, конструктори ботів і рішення на основі програмування. Аналіз показав, що попри зручність конструкторів і вбудованих інструментів для месенджерів, саме програмна реалізація чат-бота надає найбільшу гнучкість та можливість реалізації складних функцій, що є критично важливим для вузькоспеціалізованих задач.

Також було здійснено огляд існуючих цифрових рішень у сфері акваріумістики, зокрема мобільних додатків, сайтів і інструментів із функціями планування, моніторингу та взаємодії з користувачем. Було виявлено, що більшість доступних інструментів орієнтовані на широке коло користувачів і не забезпечують високого рівня персоналізації, інтерактивності чи автоматизації. Крім того, сегмент чат-ботів для акваріумістів залишається майже не заповненим.

Отже, на основі проведеного аналізу можна зробити висновок про актуальність створення спеціалізованого чат-бота для акваріумістики, який поєднає переваги програмної гнучкості з практичними потребами користувачів. Такий підхід дозволить заповнити наявну нішу, підвищити зручність догляду за акваріумом та сприяти розвитку цифрових сервісів у цій сфері.

# 2 ПРОЕКТУВАННЯ ЧАТ-БОТУ ДЛЯ ДОГЛЯДУ ЗА ДОМАШНІМ АКВАРІУМОМ

## 2.1 Огляд вимог до функціональності чат-боту

Розробка чат-боту для догляду за домашнім акваріумом потребує чіткого визначення його функціональності, щоб забезпечити зручність і ефективність для користувачів. Чат-бот має надавати інформаційні довідки про риб, рослини, параметри води, а також автоматизувати завдання, такі як нагадування про заміну води чи чищення фільтрів. Для створення чат-ботів існує два основних підходи: програмування з використанням мов, таких як Python, або використання конструкторів на спеціалізованих сервісах. Чат-бот, розроблений на Python із бібліотекою python-telegram-bot, є серверним додатком, який взаємодіє з Telegram через Bot API. Такий підхід вимагає інфраструктури, включаючи хостинг і базу даних, але забезпечує гнучкість у реалізації функцій, обмежену лише можливостями платформи Telegram. Натомість чат-боти, створені через конструктори, такі як SendPulse, обмежені функціоналом сервісу, що може не дозволити реалізувати складні запити, наприклад, діагностику проблем в акваріумі [10].

Функціональність чат-бота визначається його здатністю обробляти запити користувачів і надавати релевантні відповіді. Користувач може надсилати текстові команди, наприклад, "Яка температура потрібна для неонів?" або "Нагадати про чищення фільтра". Ці запити передаються через Telegram до сервера, де програмне забезпечення, написане на Python, обробляє їх і формує відповідь. Процес взаємодії включає такі етапи:

- Користувач надсилає команду чи запит чат-боту.
- Бот пересилає запит на сервер через Bot API Telegram.
- Програма на сервері аналізує запит і генерує відповідь.
- Сервер надсилає відповідь назад чат-боту.
- Бот відображає відповідь у чаті користувача.

Цей цикл повторюється для кожної взаємодії, забезпечуючи швидке та безперебійне спілкування через HTTPS-інтерфейс Telegram, який є спрощеним API для ботів. Чат-бот має підтримувати інтерактивність, наприклад, відповідати на запити про сумісність риб, як-от уникнення поєднання гупі з агресивними барбусами, або надсилати нагадування про заміну 25% води щотижня. Крім того, чат-бот повинен мати розширювану базу знань, щоб додавати інформацію про нові види риб чи методи догляду, наприклад, боротьбу з водоростями. Такий функціонал забезпечить універсальність, дозволяючи користувачам із різним досвідом отримувати підтримку в утриманні акваріумів.

## 2.2 Структура системи чат-боту

Для розуміння роботи чат-боту для догляду за домашнім акваріумом необхідно розглянути структуру системи, яка забезпечує його функціонування. Система включає месенджер Telegram, встановлений на пристрої користувача, і серверну програму, написану на Python із використанням бібліотеки python-telegram-bot. Ця програма взаємодіє з Telegram через Bot API, обробляючи запити та надаючи відповіді. Структура системи чат-бота зображена на схемі на рисунку 2.1.

Система працює за простим принципом. Користувач надсилає запит, наприклад, "Який pH потрібен для гупі?", через Telegram. Запит передається через Bot API на сервер, де Python-програма обробляє його, звертаючись до бази даних із інформацією про акваріумні види чи параметри води. Після аналізу програма формує відповідь, наприклад, "Оптимальний pH для гупі – 6.8–7.8", і надсилає її назад через Telegram для відображення в чаті.

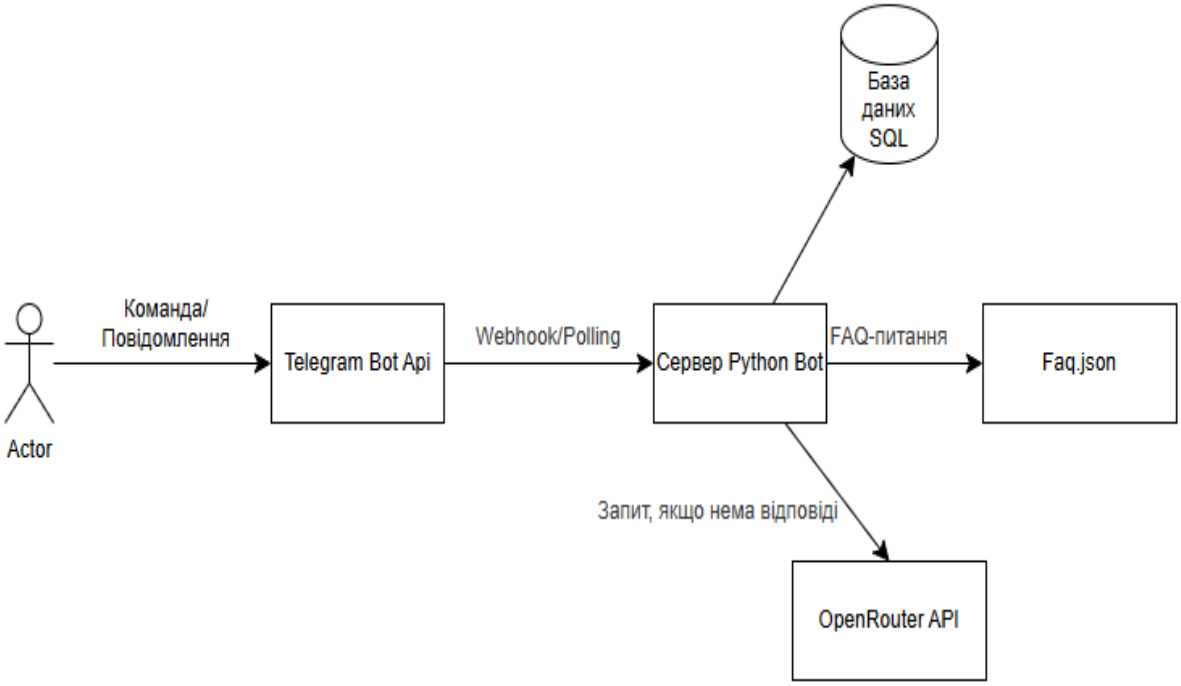


Рисунок 2.1 – Схема роботи чат-бота

Логічні компоненти системи включають модуль обробки текстових повідомлень, базу даних для зберігання довідкової інформації та інтерфейс для надсилання нагадувань, наприклад, про чищення фільтра. Така структура забезпечує гнучкість, дозволяючи розширювати функціонал, наприклад, додаванням модуля для аналізу запитів про хвороби риб [11].

Для детальнішого розуміння взаємодії між користувачем, Telegram, сервером та базою даних у процесі обробки запиту, доцільним є аналіз динаміки системи. Взаємодія основних компонентів під час типового запиту користувача представлена на діаграмі послідовності (рис. 2.2), яка ілюструє кроки передачі повідомлень, викликів функцій і відповідей між ключовими учасниками процесу.

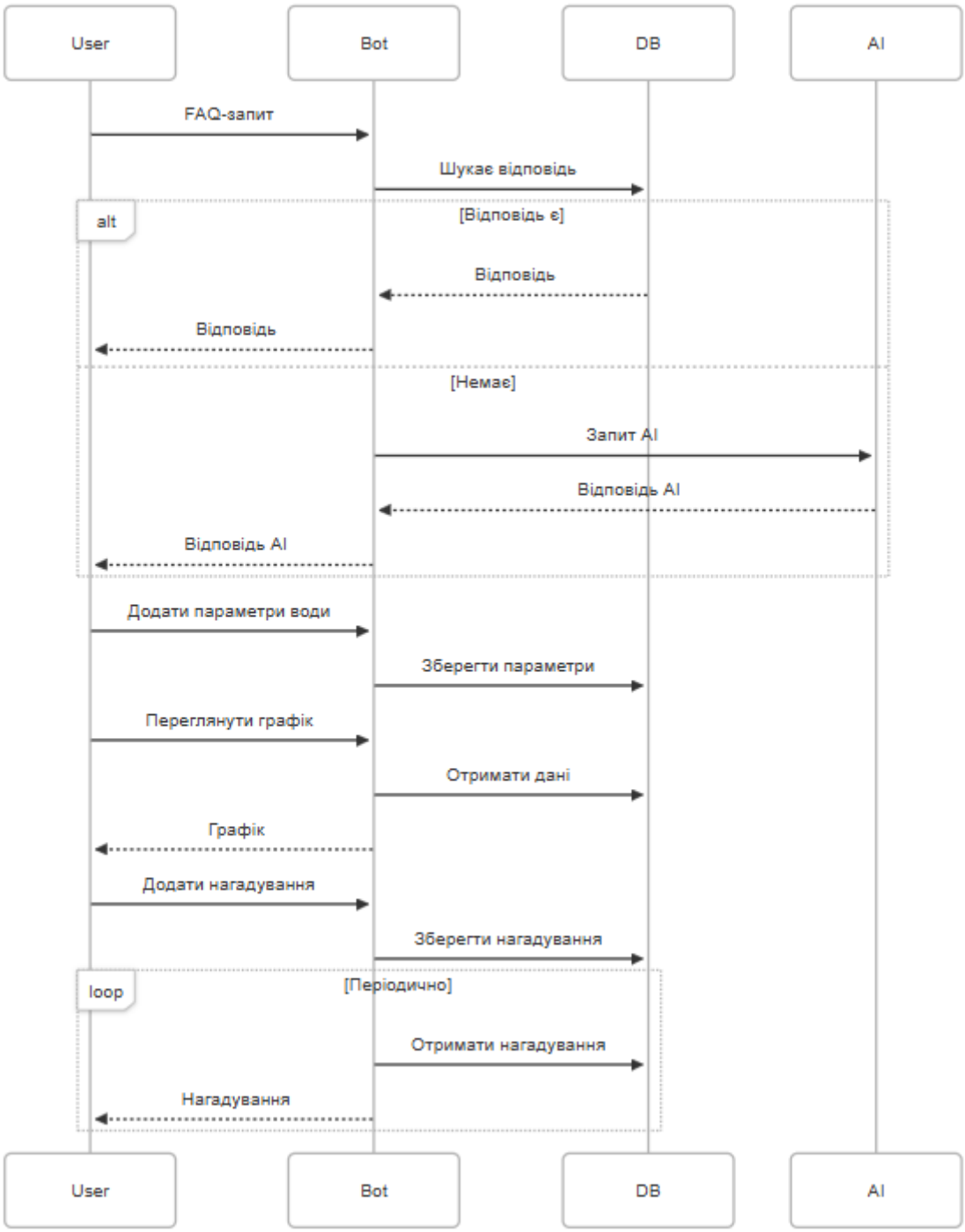


Рисунок 2.2 – Діаграма послідовності чат-боту

## 2.3 Розробка архітектури чат-бота

Архітектура чат-бота для догляду за домашнім акваріумом розроблена для забезпечення зручної та ефективної взаємодії користувачів із системою через месенджер Telegram. Чат-бот призначений для надання інформаційних

довідок про риб, рослини, параметри води, надсилання нагадувань про рутинні завдання, такі як чищення фільтрів чи заміна води, а також обробки текстових запитів для вирішення проблем, пов’язаних із утриманням акваріума. Архітектура включає кілька ключових компонентів, які забезпечують стабільну роботу та гнучкість системи:

- Модуль обробки текстових повідомлень аналізує запити користувачів, наприклад, "Яка температура потрібна для неонів?" чи "Чому вода каламутна?".
- База даних зберігає структуровану інформацію про акваріумні види, параметри води (pH, нітрати), а також графіки догляду.
- Модуль нагадувань надсилає автоматичні сповіщення, наприклад, про заміну 25–30% води щотижня чи перевірку рівня аміаку.
- Інтерфейс Telegram Bot API забезпечує зв’язок між сервером і месенджером, дозволяючи передавати текстові повідомлення, фото чи відео.
- Модуль інтеграції з OpenRouter API обробляє складні текстові запити, використовуючи моделі штучного інтелекту для генерації релевантних відповідей.
- Модуль адміністрування дозволяє оновлювати базу знань, додаючи дані про нові види риб, рослини чи методи боротьби з водоростями.

Чат-бот розроблено на мові Python із використанням бібліотеки python-telegram-bot, яка забезпечує просту та надійну інтеграцію з Telegram через Bot API. Серверна програма, розгорнута на хмарній платформі, наприклад, Heroku, обробляє запити користувачів, звертаючись до бази даних, реалізованої через SQLite, для отримання інформації, наприклад, про оптимальну температуру 24–26°C для неонів або pH 6.8–7.8 для гупі. Для обробки складних запитів, які потребують аналізу природної мови, використовується OpenRouter API, що дозволяє підключати моделі штучного інтелекту, такі як LLaMA чи Mixtral, для генерації детальних відповідей, наприклад, рекомендацій щодо зменшення каламутності води шляхом

скорочення годування чи чищення фільтра. Користувач взаємодіє з ботом через текстові повідомлення, надсилаючи запити чи команди, які передаються на сервер через HTTPS-інтерфейс Telegram. Сервер аналізує запит, звертається до бази даних або OpenRouter API, формує відповідь і повертає її в чат, наприклад, пораду уникати поєднання гупі з агресивними барбусами через їхню поведінку [11].

Функціональні можливості чат-бота охоплюють широкий спектр завдань, які полегшують догляд за акваріумами:

- Надання інформаційних довідок: користувачі можуть отримувати дані про види риб, рослини, параметри води, наприклад, вимоги до жорсткості води 5–15 dGH для дискусів.
- Автоматизація нагадувань: користувачі отримують сповіщення про завдання, такі як чищення фільтра кожні два тижні, перевірка рівня нітратів (<40 мг/л) або годування риб двічі на день (0.5–1 г корму на рибу).
- Обробка текстових запитів: користувачі можуть ставити питання, наприклад, "Чому риба ховається?", і отримувати рекомендації, такі як перевірка рівня аміаку чи забезпечення укриттів в акваріумі.
- Розширення бази знань: адміністратори можуть додавати інформацію про нові види, наприклад, екзотичних креветок, або методи догляду, такі як використання торфу для зниження pH.

Архітектура розроблена з урахуванням зручності, гнучкості та масштабованості. Модуль обробки повідомлень використовує алгоритми розпізнавання ключових слів і передає складні запити до OpenRouter API, який забезпечує аналіз природної мови. Наприклад, на запит "Як боротися з водоростями?" API може згенерувати детальну відповідь, включаючи рекомендації щодо скорочення світлового дня до 8 годин або додавання рослин, таких як анубіас. База даних SQLite зберігає структуровану інформацію, наприклад, таблиці з даними про 100 видів риб, їхні вимоги до температури, pH, жорсткості води, а також графіки догляду, такі як заміна 25%

води щотижня. Модуль нагадувань працює асинхронно, використовуючи бібліотеку python-telegram-bot для надсилання повідомлень у заданий час, наприклад, щопонеділка о 9:00 для перевірки фільтра. Telegram Bot API забезпечує стабільний зв'язок, підтримуючи мультимедійний контент, як-от фото здорового акваріума чи відеоінструкції з чищення обладнання [13].

Для забезпечення масштабованості архітектура підтримує додавання нових функцій. Наприклад, у майбутньому можна інтегрувати модуль аналізу зображень, який дозволить користувачам завантажувати фото акваріума для діагностики проблем, таких як розпізнавання водоростей або хворих риб. Інший напрям розширення – інтеграція з апаратними пристроями, наприклад, сенсорами Seneye Home, для автоматичного отримання даних про pH чи аміак і надання рекомендацій на їх основі. Хостинг на хмарних платформах, таких як Heroku чи AWS, забезпечує стабільну роботу навіть при зростанні кількості користувачів. Для безпеки всі запити через Telegram Bot API шифруються, а доступ до бази даних обмежується лише серверною програмою, що захищає дані користувачів, наприклад, графіки їхніх нагадувань.

Архітектура також враховує потреби різних груп користувачів. Новачки можуть отримувати базові довідки, наприклад, про догляд за гупі (температура 24–26°C, pH 6.8–7.8), тоді як досвідчені акваріумісти можуть запитувати складнішу інформацію, наприклад, про сумісність екзотичних видів, таких як дискус і скалярії, або методи боротьби з чорною бородою. Модуль адміністрування дозволяє оновлювати базу знань через адміністративний інтерфейс, наприклад, додавати нові види риб чи рекомендації щодо використання хімічних засобів для корекції параметрів води. Для підвищення інтерактивності чат-бот може надсилати не лише текстові відповіді, а й мультимедійний контент, наприклад, відео про правильне годування риб або фото ідеального акваріума з рослинами.

Для наочного подання структури основних компонентів чат-бота та взаємозв'язків між ними було створено UML-діаграму класів (рис. 2.3). Вона

ілюструє внутрішню організацію програмного забезпечення, демонструючи ключові класи системи, їхні атрибути, методи та взаємодію між модулями, такими як обробка повідомлень, робота з базою даних, нагадування та інтеграція зі сторонніми API. Такий підхід дає змогу структуровано представити архітектуру застосунку, полегшує аналіз коду, сприяє його підтримці та подальшому розширенню функціональності. UML-діаграма також є зручним інструментом для спілкування між учасниками команди розробки, оскільки дозволяє швидко зрозуміти загальну логіку побудови системи [12].

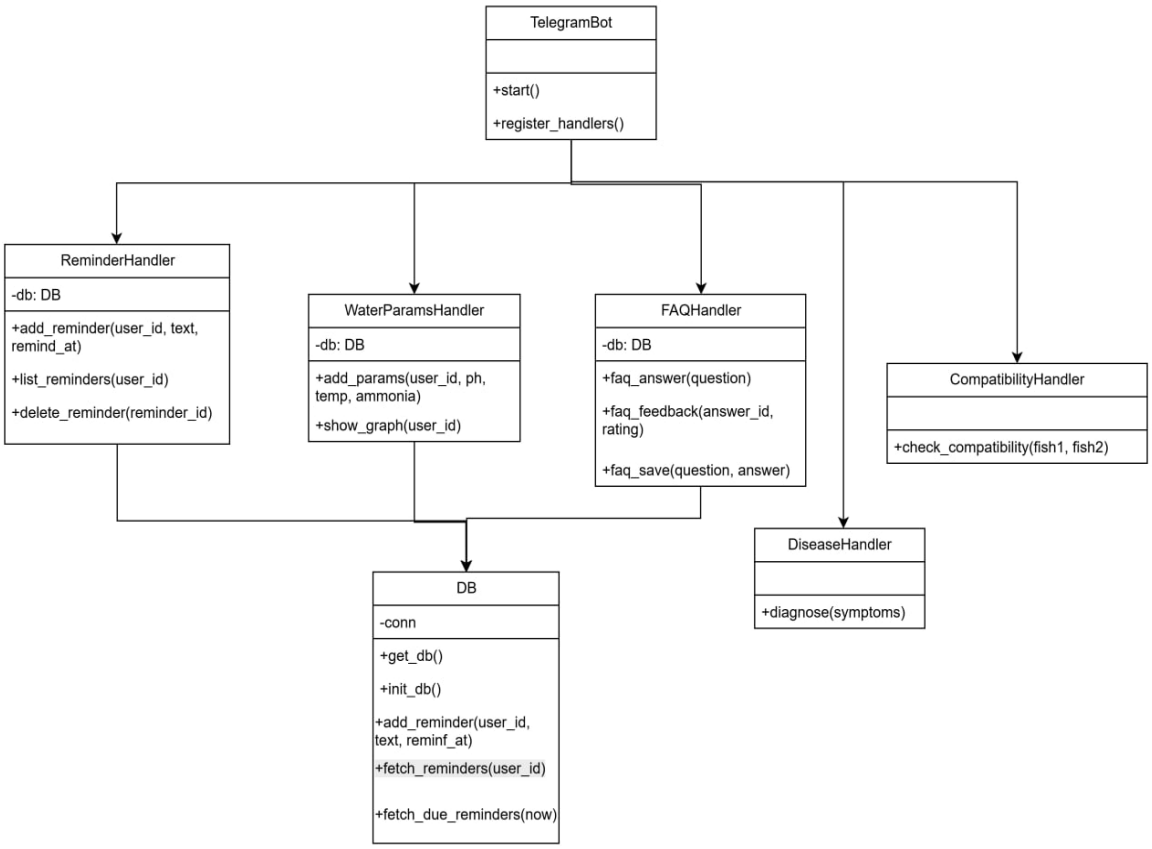


Рисунок 2.3 – UML діаграма класів програми

Така архітектура забезпечує ефективну роботу чат-бота, поєднуючи простоту використання, гнучкість і потенціал для розширення. Завдяки інтеграції з OpenRouter API чат-бот може обробляти складні запити, які виходять за межі статичної бази знань, наприклад, надавати детальні

пояснення причин млявості риб (можлива нестача кисню чи перенаселення). Система є адаптивною до потреб користувачів, дозволяючи як автоматизувати рутинні завдання, так і надавати консультаційну підтримку, що сприяє популяризації акваріумістики та підвищенню якості догляду за акваріумами.

## 2.4 Розробка алгоритмів функціонування

Проєктування алгоритмів функціонування Telegram-бота є критичним етапом, який визначає його реакцію на дії користувача, обробку даних, а також побудову відповідей. Чат-бот, що розробляється включає набір функціональних модулів (FAQ, нагадування, якість води, діагностика хвороб), кожен з яких реалізований у вигляді окремого сценарію обробки повідомлень. Для наочного представлення логіки функціонування було використано блок-схеми, які дозволяють візуалізувати послідовність дій у кожному окремому випадку використання бота. Загальна логіка алгоритму викладена на рисунку 2.4.

**І сценарій**

Загальний алгоритм навігації користувача складається з таких кроків (рис 2.4):

Крок 1. Користувач надсилає команду /start або відкриває чат із ботом.

Крок 2. Бот відображає головне меню з кнопками: "FAQ", "Нагадування", "Якість води", "Діагностика хвороб", "Сумісність риб".

Крок 3. Залежно від обраної опції, бот передає управління відповідному хендлеру.

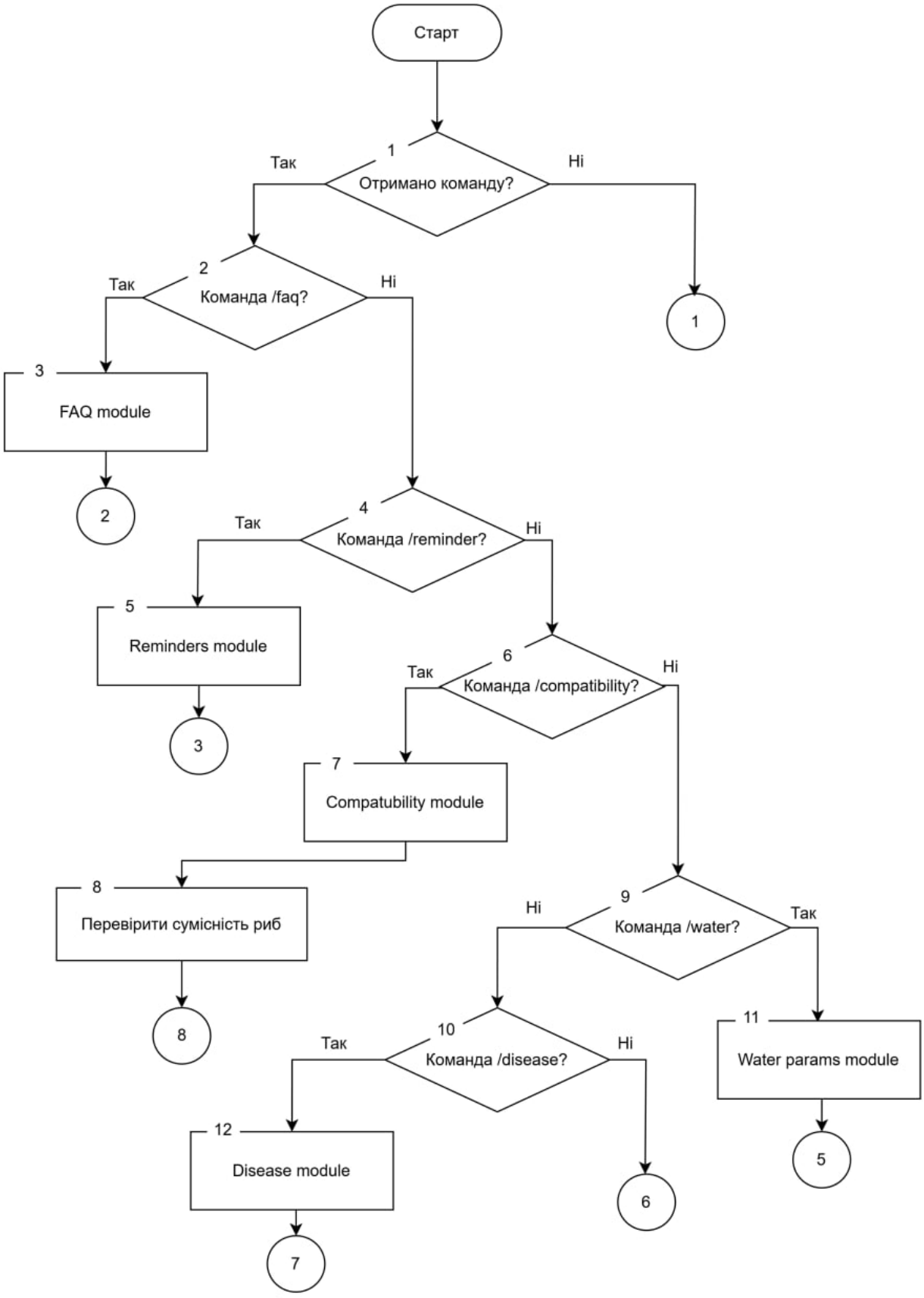


Рисунок 2.4 – Загальний алгоритм взаємодії з чат-ботом

**ІІ сценарій**

Алгоритм блоку довідки складається з таких кроків (рис. 2.4 – аркуш 2): він описує типовий сценарій взаємодії користувача з ботом у випадку, коли потрібна відповідь на запитання щодо догляду за акваріумом.

Крок 4. Користувач натискає кнопку "FAQ".

Крок 5. Бот просить ввести запитання у вільній формі.

Крок 6. Після отримання запиту бот перевіряє наявність релевантної відповіді у базі JSON.

Крок 7. Якщо збіг знайдено, то відповідь повертається користувачу.

Крок 8. Якщо відповідь неповна або відсутня, бот надає уточнююче запитання або fallback-відповідь (через OpenRouter).

Крок 9. Користувач оцінює корисність відповіді. У разі позитивної оцінки – відповідь зберігається до бази знань.

**ІІІ сценарій**

Алгоритм обробки нагадувань складається з таких кроків (рис. 2.4 – аркуш 3): він демонструє, як користувач створює або керує нагадуваннями через інтерфейс чат-бота для регулярного обслуговування акваріума.

Крок 10. Користувач обирає пункт "Нагадування".

Крок 11. Бот пропонує додати нове нагадування або переглянути/видалити наявні.

Крок 12. У разі створення нагадування, бот послідовно просить вказати тип дії (заміна води, годівля, тощо), дату та періодичність.

Крок 13. Бот перевіряє валідність введених даних.

Крок 14. Якщо дані коректні, бот додає запис до БД і запускає відповідний job через APScheduler.

Крок 15. Користувачу підтверджується успішне створення нагадування.

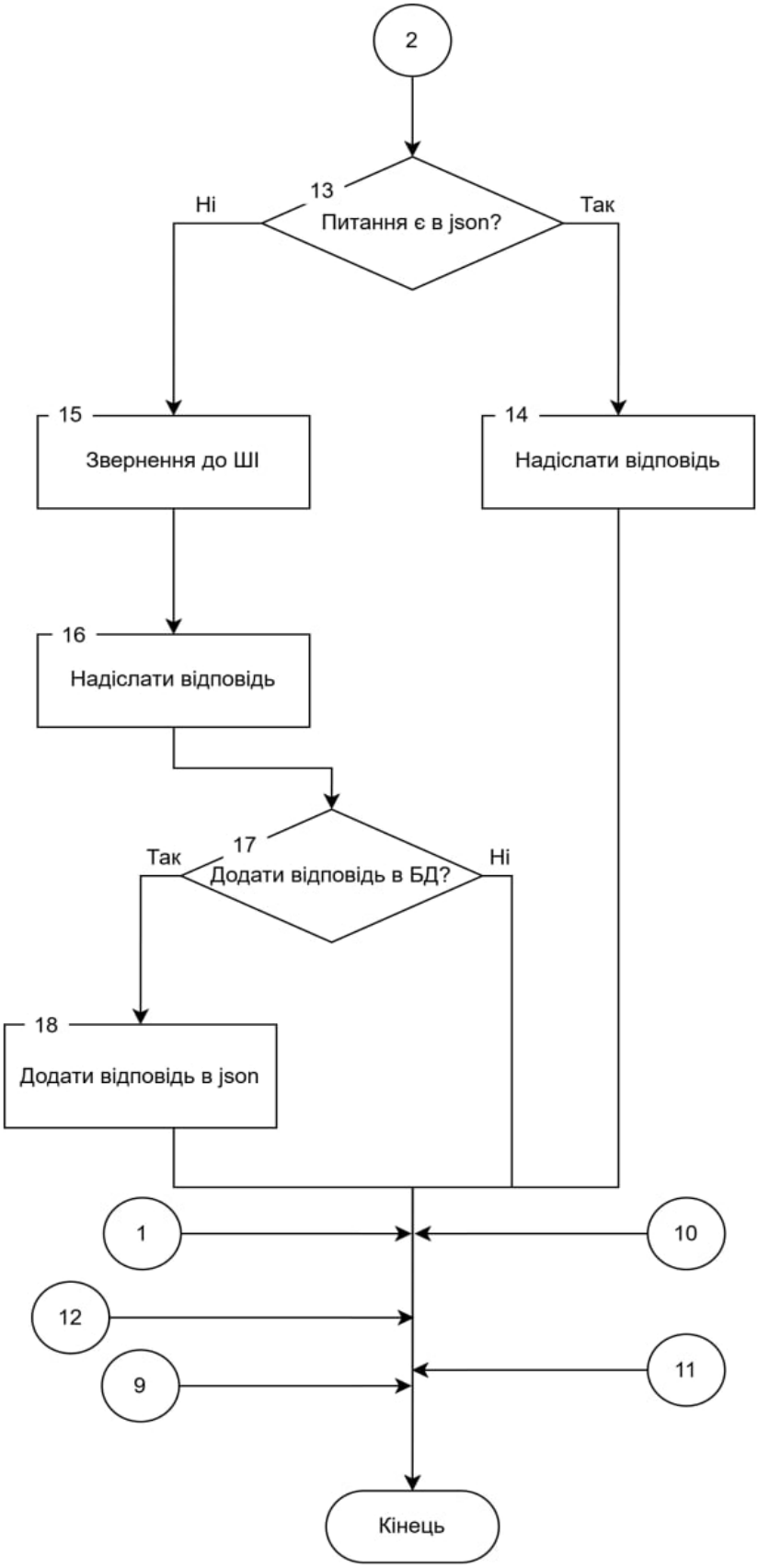


Рисунок 2.4 – аркуш 2

**IV сценарій**

Алгоритм модуля обробки параметрів води складається з таких кроків (рис 2.4):

Крок 16. Користувач натискає "Якість води".

Крок 17. Бот надає два варіанти: "Додати показники" або "Переглянути графік".

Крок 18. При додаванні бот послідовно запитує значення температури, pH, GH тощо.

Крок 19. Якщо всі значення введені коректно, бот записує їх у базу даних.

Крок 20. У разі запиту графіку бот будує графік змін (наприклад, за останні 7 днів) та надсилає користувачу з поясненням.

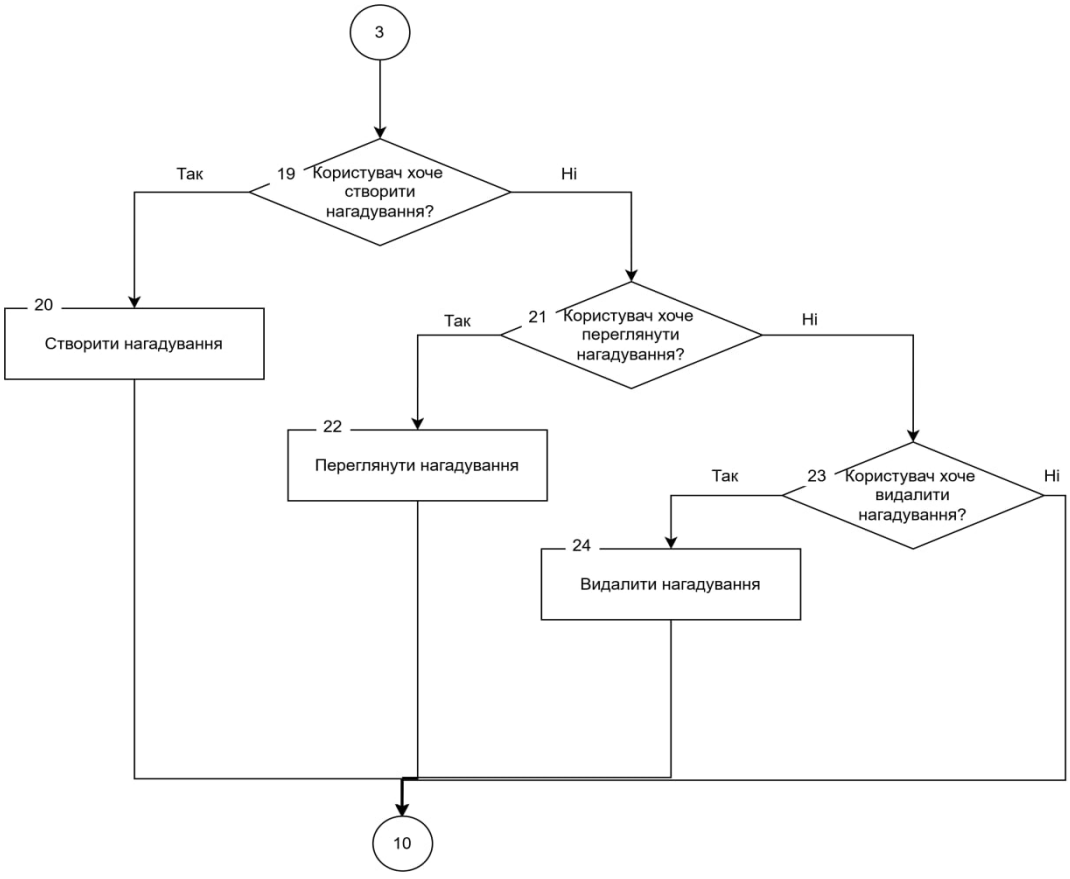


Рисунок 2.4 – аркуш 3

**V сценарій**

Алгоритм модуля діагностики хвороб складається з таких кроків (рис 2.4 – аркуш 4):

Крок 21. Користувач обирає "Діагностика хвороб".

Крок 22. Бот просить вказати помітні симптоми або описати ситуацію.

Крок 23. Дані надсилаються до API (OpenRouter), де здійснюється попередній аналіз.

Крок 24. Отримана відповідь містить можливу хворобу, супутні симптоми та поради.

Крок 25. Користувач отримує інструкцію з лікування, за потреби може повторити запит.

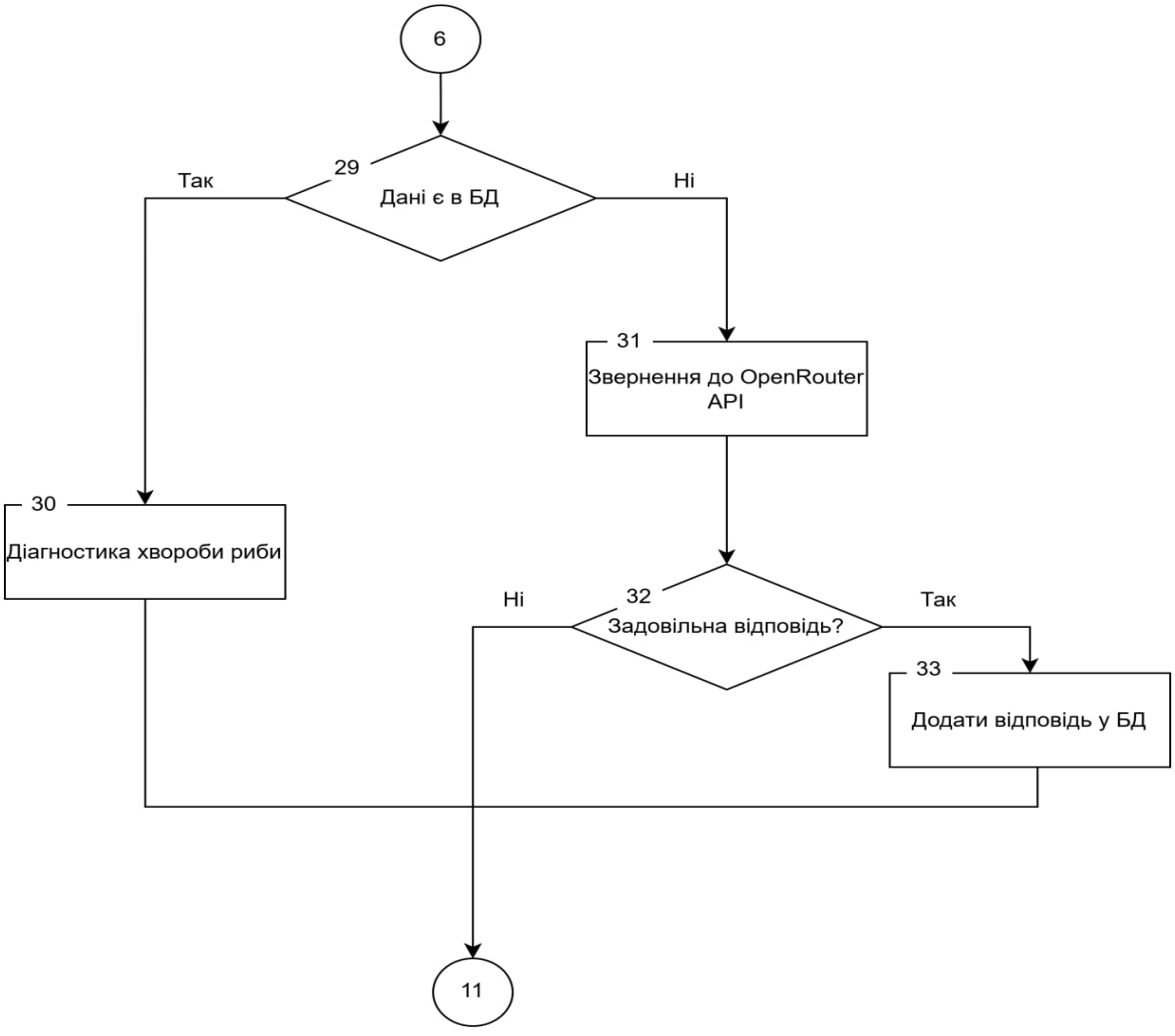


Рисунок 2.4 – аркуш 4

**VI сценарій**

Алгоритм модуля перевірки сумісності складається з таких кроків (рис 2.4 – аркуш 5):

Крок 26. Користувач вводить назви риб

Крок 27. Перевіряється валідність даних, при неправильному синтаксисі виводиться можливий варіант того, що хотів користувач.

Крок 28. Якщо варіант неправильний, користувач повторно вводить дані.

Крок 29. Користувач отримує відповідь про сумісність риб.

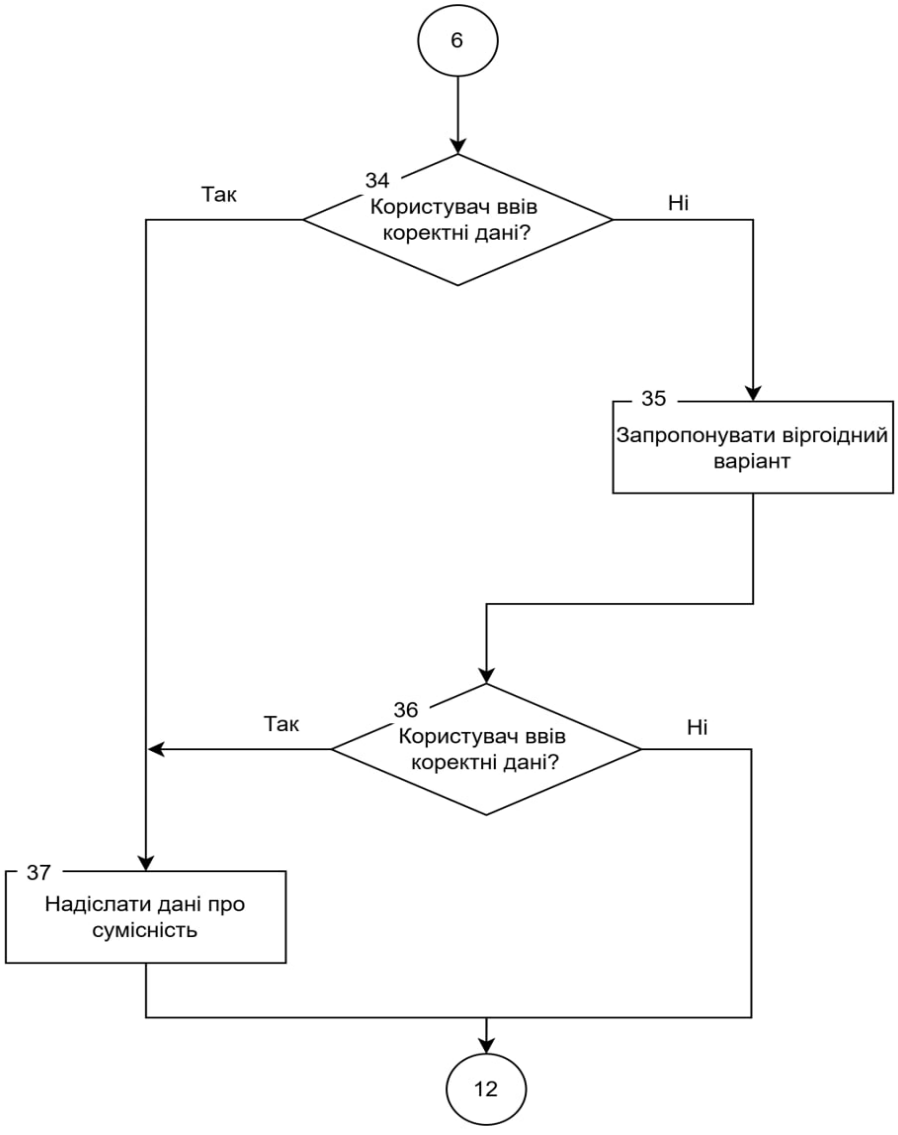


Рисунок 2.4 – аркуш 5

## 2.5 Висновок до розділу 2

У цьому розділі було здійснено проєктування архітектури Telegram-бота для догляду за домашнім акваріумом, що включає визначення його основних функціональних компонентів та їх взаємозв’язків. Запропонована структура базується на модульному підході, що дозволяє забезпечити зручну організацію коду, спрощує розробку, тестування та подальшу підтримку системи.

Описано загальну архітектуру чат-бота, яка передбачає поділ на окремі логічні модулі: обробники повідомлень користувача, модулі взаємодії з базою даних, модуль планування задач, а також блоки, відповідальні за конкретні функції, такі як обробка частих запитань, аналіз параметрів води, нагадування тощо. Такий підхід забезпечує чітке розмежування відповідальності між компонентами системи та полегшує її масштабування.

Розроблено структурну схему бота, яка демонструє взаємодію між модулями. Також наведено діаграму класів, що описує структуру даних і дозволяє краще зрозуміти логіку зберігання та обробки інформації. Передбачено окремі дані для зберігання налаштувань, параметрів користувача та шаблонів запитань, що забезпечує гнучкість системи.

Проєктування також включає опис організації файлів і каталогів у рамках проєкту, що забезпечує зручну навігацію та підтримку чистої архітектури при подальшому розширенні функціоналу.

Запропонована структура Telegram-бота створює надійний фундамент для його подальшої реалізації, відповідає принципам модульності, масштабованості та зрозумілості, що дозволяє ефективно розвивати систему в межах обраної тематики.

# 3 РОЗРОБКА ЧАТ-БОТУ ДЛЯ ДОГЛЯДУ ЗА АКВАРІУМОМ

## 3.1 Вибір технологій та інструментів для розробки чат-боту

Для створення ефективного Telegram-бота, який би виконував завдання з догляду за акваріумом, надсилав нагадування, аналізував параметри води та забезпечував взаємодію з користувачем через інтерфейс месенджера, необхідно було обрати відповідні технології. Основним критерієм вибору були простота у використанні, активна спільнота, наявність відповідних бібліотек, підтримка API Telegram та можливість швидкої розробки MVP.

У якості основної мови програмування було обрано Python. Такий вибір обумовлений рядом переваг:

- Простота синтаксису, що дозволяє швидко створювати функціональність без зайвого коду.
- Велика кількість готових бібліотек, зокрема для роботи з API (aiogram, requests, telethon), обробки даних (pandas, matplotlib) та інтеграції з базами даних (SQLAlchemy, asyncpg).
- Активна спільнота, що значно спрощує вирішення потенційних проблем.
- Підтримка асинхронного програмування, що дозволяє обробляти запити користувачів без затримок.

Хоча Python поступається за продуктивністю мовам низького рівня, наприклад C++ або Java, у контексті розробки Telegram-бота, де акцент робиться на логіці та зручності інтеграцій, ця вада є незначною.

Для порівняння також було розглянуто мову JavaScript, яка може використовуватись як на сервері (через Node.js), так і на клієнті (табл. 3.1). Проте у випадку Telegram-бота, де немає графічного інтерфейсу у браузері, переваги універсальності JavaScript виявились менш суттєвими. Також для

Python існує більш зручна типізація (з використанням pydantic) та інтеграція з Telegram API через спеціалізовані бібліотеки [14].

Окрім того, було розглянуто можливість використання мови Go (Golang), яка відзначається високою продуктивністю, простотою синтаксису та зручною побудовою асинхронного коду. Go є чудовим вибором для створення високонавантажених сервісів, однак у порівнянні з Python має обмеженішу екосистему для швидкої розробки чат-ботів, зокрема через меншу кількість готових Telegram-бібліотек і складнішу інтеграцію з інструментами штучного інтелекту [14].

Таблиця 3.1 – Порівняння мов програмування для розробки чат-бота

| **Характеристика** | **Python** | **JavaScript** | **Go** |
|---|---|---|---|
| Простота вивчення | Так | Так | Так |
| Велика кількість бібліотек | Так | Так | Частково |
| Підтримка Telegram API | Так | Так | Так |
| Асинхронність | Так | Так | Так |
| Продуктивність | Ні | Так | Так |
| Крос-платформеність | Так | Так | Так |
| Наявність типізації | Так | Частково | Так |
| Підтримка інструментів ML/AI | Так | Обмежено | Обмежено |
| Швидкість розробки MVP | Висока | Середня | Середня |

Враховуючи вищенаведене, було остаточно вирішено зупинитись на Python.

У якості середовища розробки було обрано PyCharm. Це професійне IDE для Python, що має вбудовані засоби для автодоповнення коду, налагодження, підтримку віртуальних середовищ та інтеграцію з системами контролю версій. PyCharm також зручно працює з файлами конфігурації (.env, pyproject.toml, requirements.txt) та дозволяє створювати окремі середовища для проєкту (venv, pipenv) [15].

Для реалізації логіки чат-бота було використано бібліотеку aiogram – сучасну асинхронну обгортку над Telegram Bot API, яка підтримує FSM (машину станів), кастомні фільтри, модульну структуру та просту інтеграцію з планувальниками (наприклад, APScheduler). Це дозволило реалізувати як обробку команд користувача, так і розсилку автоматичних повідомлень [16].

Додатково було використано такі бібліотеки:

- matplotlib – для побудови графіків змін параметрів води;
- pandas – для збереження та обробки таблиць з вимірами;
- SQLAlchemy – ORM для зручної роботи з базою даних PostgreSQL;
- apscheduler – для реалізації планувальника нагадувань.

Оскільки бот працює з персональними даними користувачів, було обрано SQLite як основну систему управління базами даних. Вона забезпечує надійність, транзакційність та можливість масштабування.

У якості структури проєкту було обрано модульний підхід з поділом на директорії: handlers, database, data, services, що дозволяє зберігати логіку окремих функцій у незалежних файлах. Це спрощує розширення функціоналу в майбутньому та полегшує налагодження [17].

Отже, поєднання мови Python, середовища PyCharm та бібліотеки aiogram дозволило ефективно реалізувати функціональність Telegram-бота для акваріумістів з гнучкою архітектурою та високою швидкістю розробки.

### 3.2 Реалізація основних функцій чат-боту

Telegram-бот Aquaris створений з метою забезпечити зручний цифровий інструмент для щоденного догляду за акваріумом, що дозволяє акваріумістам автоматизувати рутинні завдання, перевіряти параметри води, слідкувати за здоров’ям риб та отримувати рекомендації в інтерактивному режимі. Основою архітектури бота стала бібліотека aiogram, яка забезпечує ефективну обробку

запитів, підтримку finite-state machines (FSM) для діалогів з користувачем і асинхронну роботу з базою даних.

Після запуску бот надсилає вітальне повідомлення та виводить головне меню у вигляді клавіатури Telegram. Меню містить ключові функції, зокрема: "Якість води", "Сумісність риб", "Нагадування", "Діагностика хвороб" та "Поширені питання". Основна логіка управління ботом побудована у файлі bot.py, де задаються обробники команд і викликаються функції з відповідних модулів [18]. Наприклад, ось фрагмент коду для стартової команди:

```
@dp.message(CommandStart())
async def cmd_start(message: Message):
    await message.answer("Вітаю в боті Aquaris! Оберіть дію:", reply_markup=main_menu_keyboard)
```

Однією з найбільш використовуваних функцій є секція "Поширені запитання". Вона працює на основі локального JSON-файлу з питаннями та відповідями. Коли користувач ставить запитання, бот виконує пошук за ключовими словами. Для цього використовується просте порівняння за допомогою методу схожості:

```
from difflib import get_close_matches

def search_faq(query, faq_data):
    questions = [entry['question'] for entry in faq_data]
    matches = get_close_matches(query.lower(), questions, n=3, cutoff=0.4)
    return matches
```

Якщо відповідь не знайдена, бот звертається до API мовної моделі через зовнішній запит, наприклад OpenRouter чи OpenAI. Користувачу надається відповідь з позначкою, що вона згенерована автоматично [21]. Якщо користувач вважає відповідь корисною, вона зберігається до бази, підкріплюючи локальний датасет:

```
def add_to_faq(question, answer, filename='faq_data.json'):
```

```
    with open(filename, 'r+', encoding='utf-8') as f:
        data = json.load(f)
        data.append({"question": question, "answer": answer})
        f.seek(0)
        json.dump(data, f, indent=2, ensure_ascii=False)
```

Функція контролю якості води реалізує як введення параметрів, так і побудову графіків. Користувач вводить значення pH, температури, вмісту аміаку, нітритів та нітратів, після чого дані записуються до бази (SQLite через aiosqlite) із позначкою дати. Для перегляду змін у динаміці бот будує графік за допомогою бібліотеки matplotlib:

```
import matplotlib.pyplot as plt

def create_water_quality_plot(data):
    dates = [row['date'] for row in data]
    ph_values = [row['ph'] for row in data]
    temp_values = [row['temperature'] for row in data]

    plt.figure(figsize=(10, 5))
    plt.plot(dates, ph_values, label="pH", marker="o")
    plt.plot(dates, temp_values, label="Температура", marker="x")
    plt.legend()
    plt.title("Динаміка параметрів води")
    plt.xlabel("Дата")
    plt.ylabel("Значення")
    plt.grid(True)
    plt.tight_layout()
    plt.savefig("plot.png")
```

Згенерований графік надсилається користувачеві в чат, що значно підвищує зручність моніторингу. Це дозволяє вчасно реагувати на коливання

умов середовища, що можуть бути небезпечними для живих організмів акваріума [20].

Не менш важливою функцією є перевірка сумісності риб. Користувач обирає один або кілька видів риб з інтерактивного меню, після чого бот проводить аналіз сумісності на основі попередньо підготовленої таблиці сумісності у JSON-форматі. Система враховує агресивність, територіальність, темперамент, умови утримання:

```
def check_compatibility(selected_fish, compatibility_data):
    results = []
    for i in range(len(selected_fish)):
        for j in range(i+1, len(selected_fish)):
            pair = sorted([selected_fish[i], selected_fish[j]])
            key = f"{pair[0]}_{pair[1]}"
            result = compatibility_data.get(key, "невідомо")
            results.append(f"{pair[0]} + {pair[1]}: {result}")
    return "\n".join(results)
```

Результати виводяться користувачу у вигляді пояснень, наприклад:

*Гупі + Скаляра: часткова сумісність*

*Скалярія + Барбус: несумісні*

Це дозволяє уникати конфліктів, канібалізму або стресу серед риб.

Функціонал нагадувань реалізований через бібліотеку APScheduler. Користувач задає тип нагадування (наприклад, "Підміна води"), періодичність (кожні 7 днів) і бажаний час доби. Бот додає завдання до розкладу та регулярно надсилає відповідні повідомлення:

```
scheduler.add_job(
    send_reminder,
    trigger='interval',
    days=7,
    start_date=datetime.now() + timedelta(days=1),
```

*args=[user_id, "Пора підмінити воду!"]*

*)*

Це особливо важливо для користувачів, які доглядають за великим або біотопним акваріумом, де регулярність має критичне значення [19].

Функція діагностики хвороб є найінтелектуальнішою частиною бота. Вона використовує зовнішні API на базі LLM для інтерпретації опису симптомів риби, які вводить користувач. Наприклад: *“*У риби тьмяні плавці, вона тримається біля дна, з’явились білі плями*”*. Бот формує запит до API, який повертає опис можливої хвороби (наприклад, іхтіофтіріоз) і пропонує базовий план лікування:

*prompt = f"Ознаки: {symptoms}. Яка це хвороба акваріумної риби та як її лікувати?"*

*response = openrouter.chat(prompt)*

У результаті користувач отримує поради, адаптовані до його опису, що дозволяє швидше зорієнтуватися у ситуації без потреби шукати інформацію вручну.

Важливим елементом реалізації всієї логіки є використання FSM (StatesGroup) з бібліотеки aiogram. Це дозволяє зберігати контекст при багатокроковій взаємодії, наприклад під час заповнення параметрів води чи створення нагадувань. Наприклад:

*class WaterInputStates(StatesGroup):*

*ph = State()*

*temperature = State()*

*ammonia = State()*

Завдяки FSM користувач може вводити значення поетапно, а бот – перевіряти їх коректність та зберігати у базу.

Загальна структура коду організована модульно: в каталозі handlers/ зберігаються обробники кожного розділу, database/ відповідає за збереження інформації у SQLite, а data/ містить JSON-файли з довідковими даними. Це

дозволяє легко масштабувати систему, додавати нові функції або замінювати окремі компоненти.

Таким чином, реалізація функцій Telegram-бота Aquaris забезпечує користувача широким інструментарієм для догляду за акваріумом – від аналізу води до лікування хвороб. Поєднання асинхронного програмування, структурованого підходу до обробки діалогів і використання сучасних API забезпечує ефективну, надійну та зручну взаємодію з ботом, яка може бути легко адаптована до різних рівнів досвіду користувачів – як початківців, так і досвідчених акваріумістів.

### 3.3 Реєстрація та налаштування чат-боту на платформі Telegram

Для створення нового Telegram-бота найзручніше скористатися спеціальним сервісом під назвою BotFather. Це офіційний бот від Telegram, який дозволяє швидко створити, налаштувати та керувати ботами без необхідності володіння спеціальними технічними знаннями. Завдяки простому інтерфейсу, BotFather підходить як початківцям, так і досвідченим користувачам.

BotFather дозволяє зареєструвати необмежену кількість ботів, при цьому єдина обов’язкова вимога – наявність унікального імені користувача (username), яке повинно закінчуватись на bot.

Telegram-боти – це особливий тип облікових записів, які не потребують окремого номера телефону. Взаємодія з ними може відбуватись двома основними способами:

1. Через пряме спілкування в особистому чаті або при додаванні бота до групи.
2. Через вбудований режим: введення @username бота безпосередньо в полі введення повідомлення у будь-якому чаті.

Повідомлення, які надсилають користувачі, потрапляють на сервери, де працює програмне забезпечення бота. Сам Telegram виконує роль посередника, відповідаючи за шифрування та взаємодію з Telegram Bot API. Розробник взаємодіє із сервісом через простий HTTPS-інтерфейс – так зване Bot API.

BotFather підтримує розширені функції, зокрема:

- налаштування інтерактивного меню для спрощення навігації;
- встановлення вебхуків, що дозволяє миттєво отримувати оновлення;
- підключення до сторонніх сервісів (CRM, платіжні системи тощо).

BotFather – це універсальний інструмент для створення ефективних та зручних ботів, що можуть автоматизувати багато завдань(рис 3.1).

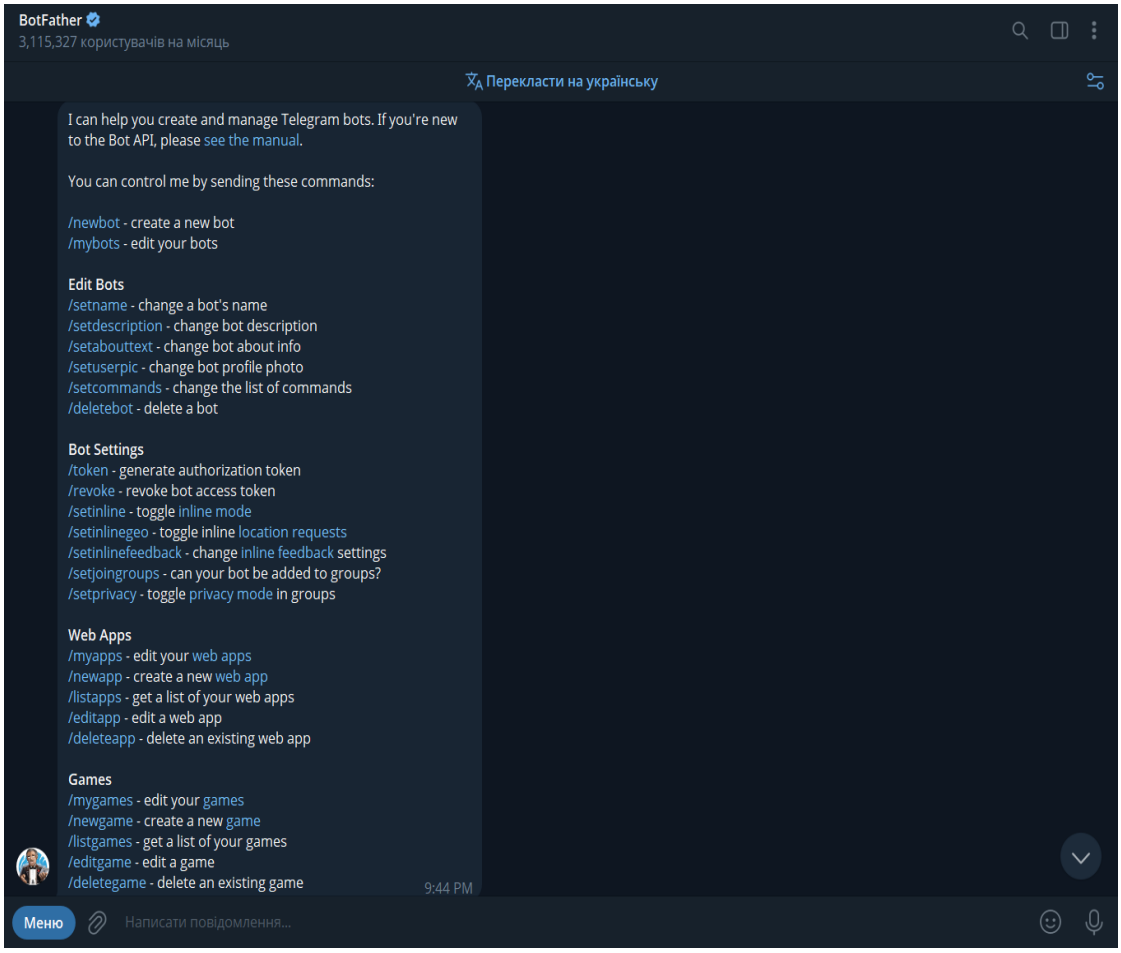


Рисунок 3.1 – Вікно команд чат-бота «BotFather»

Щоб створити нового Telegram-бота, достатньо виконати декілька послідовних дій (рис 3.2):

1. Придумати ім’я бота – воно буде показуватись у чатах. Ім’я може бути довільним, головне – зрозумілим і доречним.

2. Вибрати username – має бути унікальним, латиницею, без пробілів. Обов’язково має завершуватись на bot. Наприклад: MyAquariumBot.

3. Отримати токен доступу – після створення бота, BotFather надішле спеціальний токен. Це ключ для взаємодії з API. Його потрібно зберігати у безпечному місці.

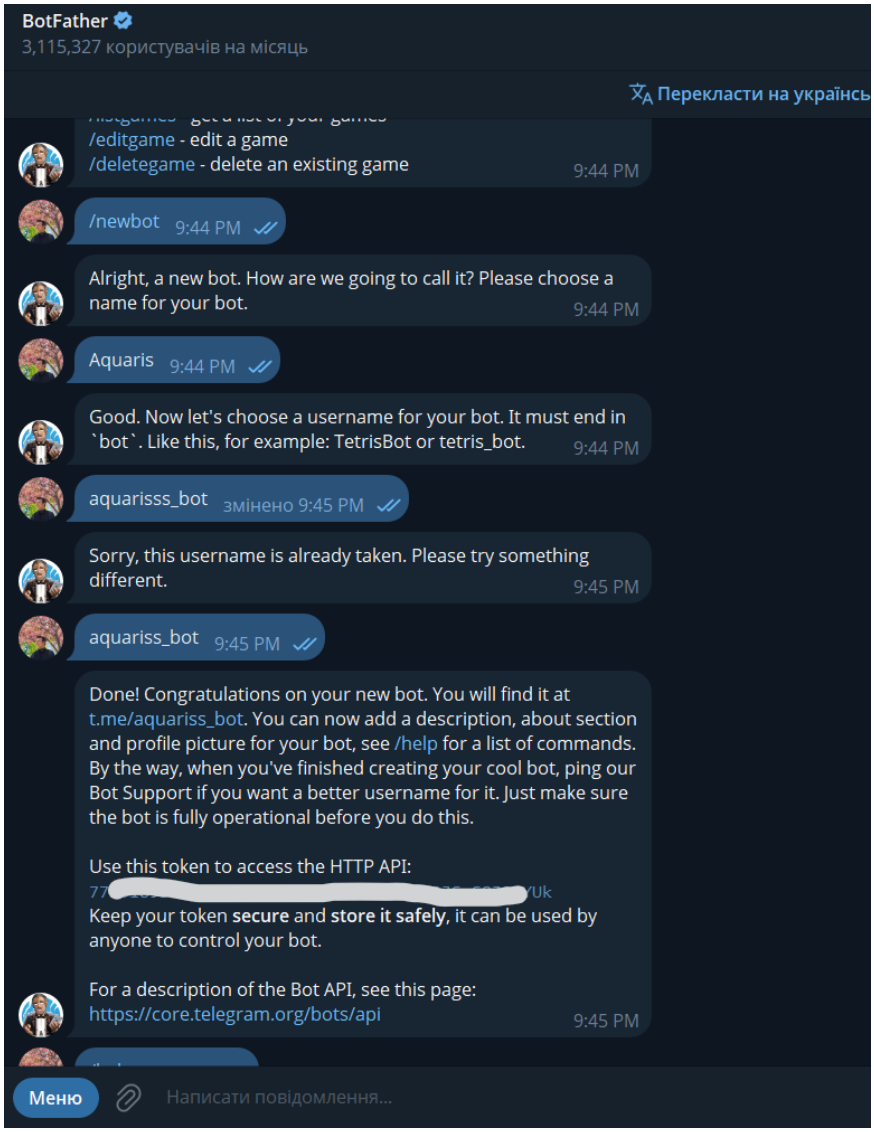


Рисунок 3.2 – Отримане повідомлення з токеном

Після створення бота можна виконати базове налаштування (рис 3.3):

- встановити аватар профілю;
- додати короткий опис;
- оновити інформацію «Про бота»;
- налаштувати список доступних команд.

Ці дії виконуються за допомогою відповідних команд у BotFather.

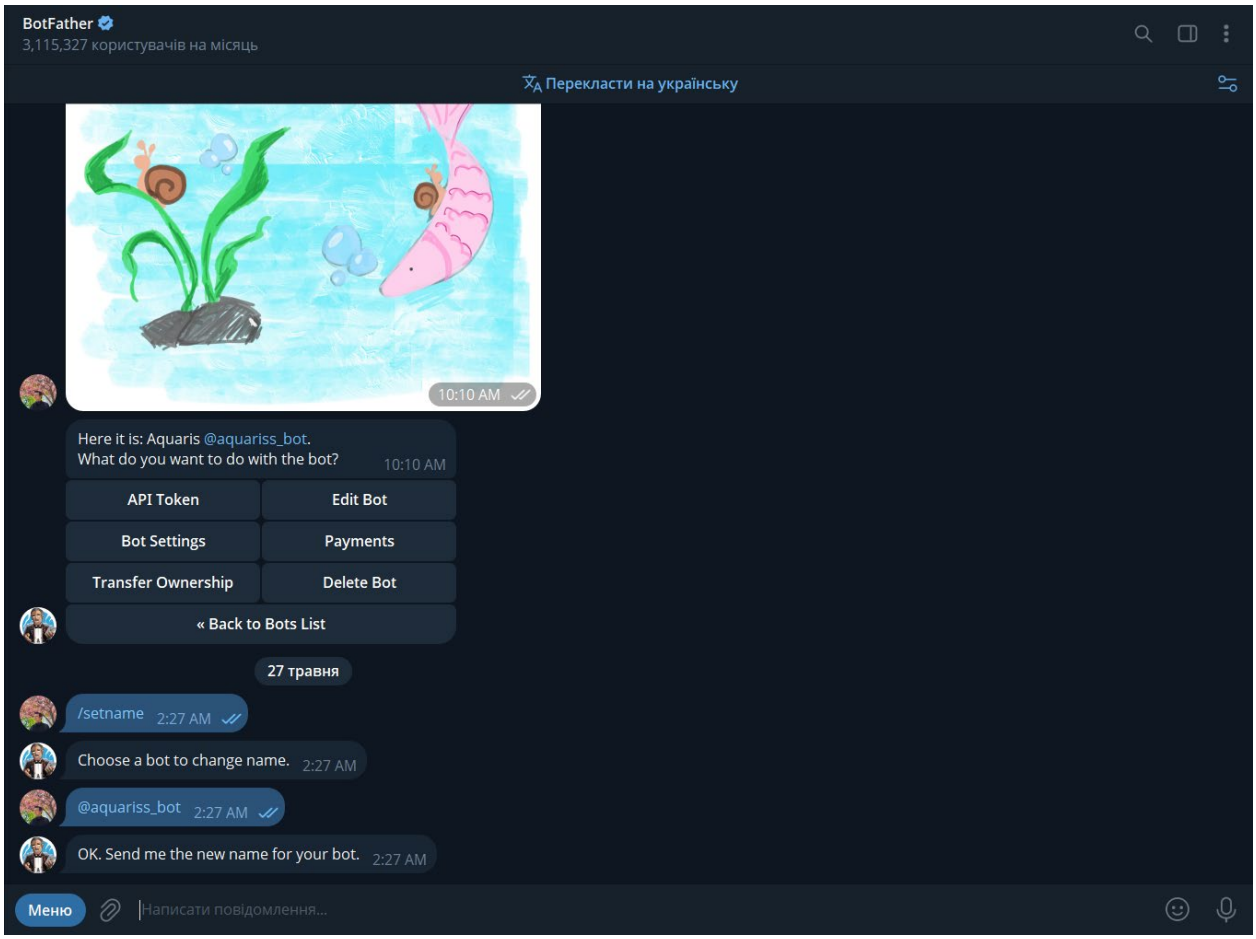


Рисунок 3.3 – Перейменування бота за допомогою BotFather

BotFather підтримує набір команд для управління ботами:

- /newbot – створення нового бота;
- /mybots – список ваших ботів;
- /setname – зміна імені;
- /setdescription – редагування опису;
- /setabouttext – оновлення інформації «Про бота»;

- /setuserpic – встановлення фотографії;
- /setcommands – налаштування списку команд;
- /token – повторне отримання токена;
- /revoke – відкликання поточного токена;
- /deletebot – видалення бота;
- /setinline – увімкнення вбудованого режиму;
- /setinlinegeo – дозволити надсилання геолокації в inline-режимі.

Налаштувавши вигляд бота, можна переходити до налаштування його функціональності. Через команду /setcommands ви можете визначити перелік основних дій, які буде виконувати ваш бот. Це можуть бути, наприклад, команди на зразок:

water – додати параметри води

compatibility – перевірка сумісності риб

reminder – створити нагадування

faq – пошук відповіді

Важливо пам’ятати, що BotFather – це лише перший етап у створенні бота. Після налаштувань починається етап програмування логіки взаємодії з користувачем, обробки повідомлень та підключення зовнішніх сервісів [22].

### 3.4 Тестування функціональності чат-боту

Після завершення етапу реалізації основних функцій чат-бота важливим кроком є тестування його функціональності. Для даного проєкту було обрано мануальне (ручне) тестування, що дозволяє перевірити логіку роботи, коректність обробки повідомлень, наявність помилок і збоїв у взаємодії з користувачем.

Тестування здійснювалося безпосередньо в середовищі Telegram, з використанням реального облікового запису бота. Усі функціональні модулі перевірялись послідовно: меню, додавання параметрів води, перегляд

графіків, розділ FAQ, перевірка сумісності риб, створення нагадувань і базова діагностика хвороб (табл 3.2).

Мета тестування:

- Перевірити доступність і реакцію на команди та меню.
- Переконатись у правильності логіки обробки користувацьких повідомлень.
- Виявити некоректні сценарії використання.
- Зібрати візуальні підтвердження правильної роботи кожного модуля.

Таблиця 3.2 – Чек-лист для тестування чат-бота

| **№** | **Функція** | **Дія користувача** | **Очікуваний результат** |
|---|---|---|---|
| 1 | Запуск бота | Відкрити чат і натиснути кнопку «Почати» | З’являється головне меню |
| 2 | FAQ | Обрати пункт меню «FAQ» і ввести питання | Бот повертає відповідь з бази або уточнює питання |
| 3 | Якість води → Додати параметри | Обрати «Якість води» → «Додати параметри» | Бот запитує значення параметрів по черзі |
| 4 | Якість води → Перегляд графіку | Обрати «Якість води» → «Переглянути графік» | Бот надсилає графік за попередньо введеними даними |
| 5 | Сумісність риб | Вибрати дві риби в діалозі перевірки сумісності | Бот виводить результат аналізу сумісності |
| 6 | Нагадування | Обрати функцію створення нагадування | Бот пропонує час і підтвердження |
| 7 | Діагностика хвороб | Ввести симптоми у відповідному розділі | Бот надсилає припущення та рекомендації |
| 8 | Помилкове введення | Ввести неочікувану команду або повідомлення | Бот виводить повідомлення про невірний формат |

Опис кроків мануального тестування:

Крок 1. Запуск і доступність меню

Відкрийте Telegram, знайдіть бота за username, натисніть «Start». Перевірте, чи відображається головне меню (рис. 3.4).

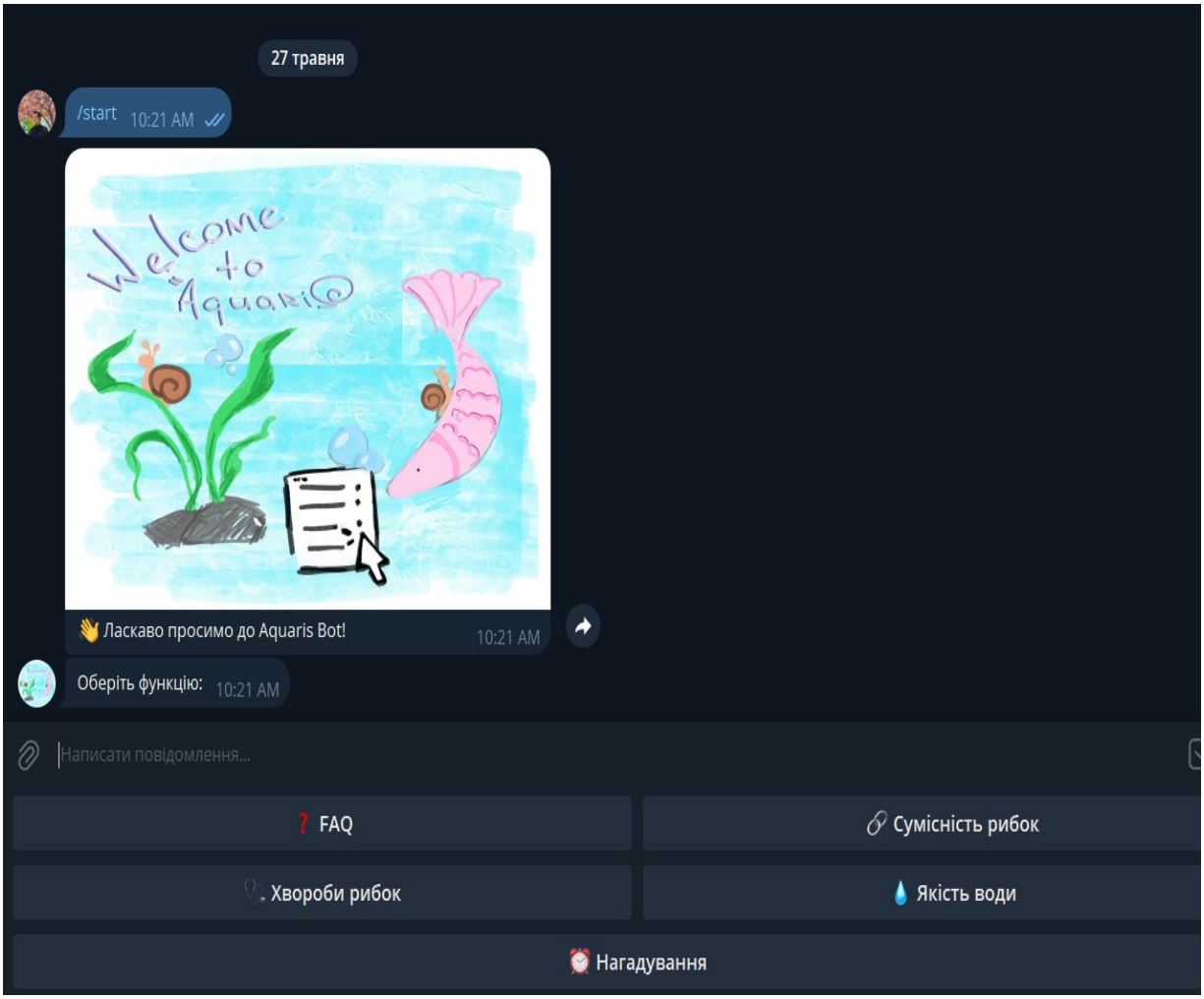


Рисунок 3.4 – Головне меню бота після старту

Крок 2. Перевірка FAQ

Виберіть пункт «FAQ», введіть питання (наприклад, «Як часто змінювати воду?») (рис 3.5). Бот має знайти відповідь у базі або поставити уточнювальне питання.

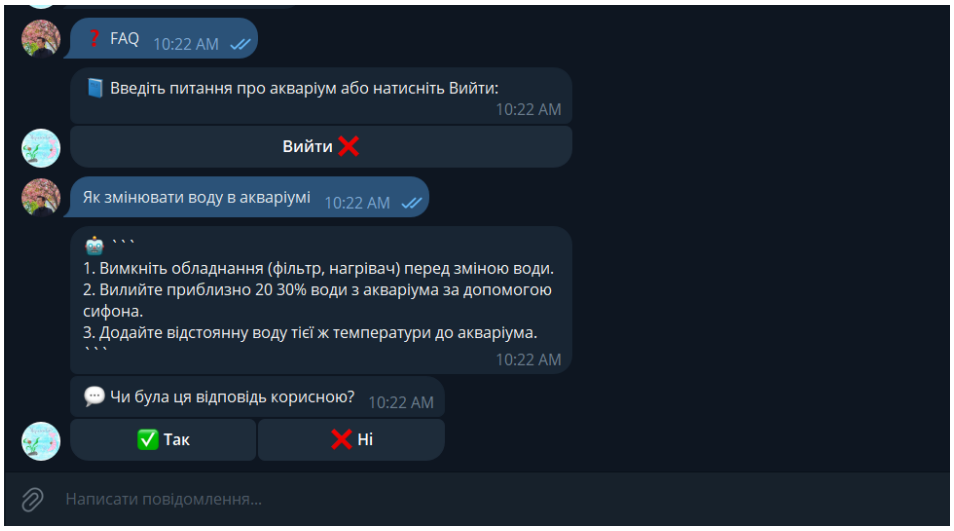


Рисунок 3.5 – Взаємодія з розділом FAQ

Крок 3. Додавання параметрів води

У головному меню оберіть «Якість води», далі натисніть «Додати параметри» (рис 3.6). Введіть запропоновані значення (pH, температура, нітрати тощо).

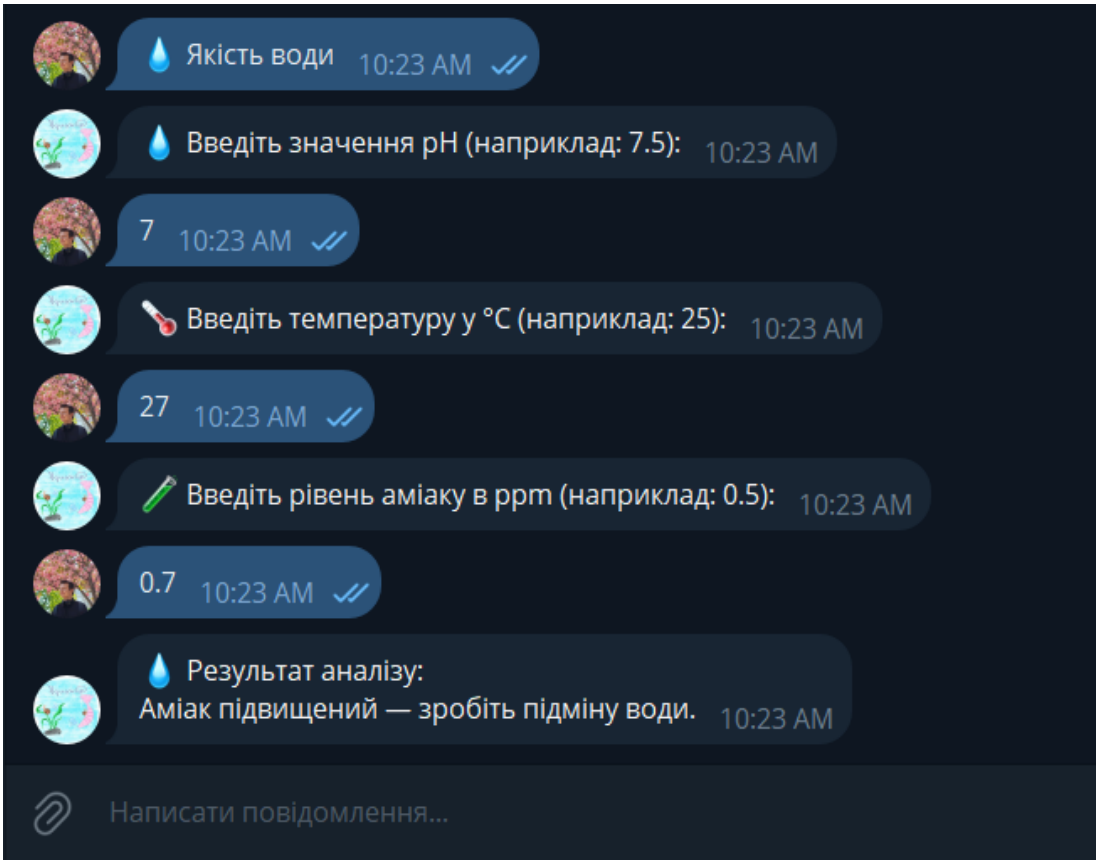


Рисунок 3.6 – Введення параметрів води

Крок 4. Перегляд графіку

Після додавання даних поверніться до розділу «Якість води» і оберіть «Переглянути графік» (рис 3.7). Бот надішле графік з matplotlib.

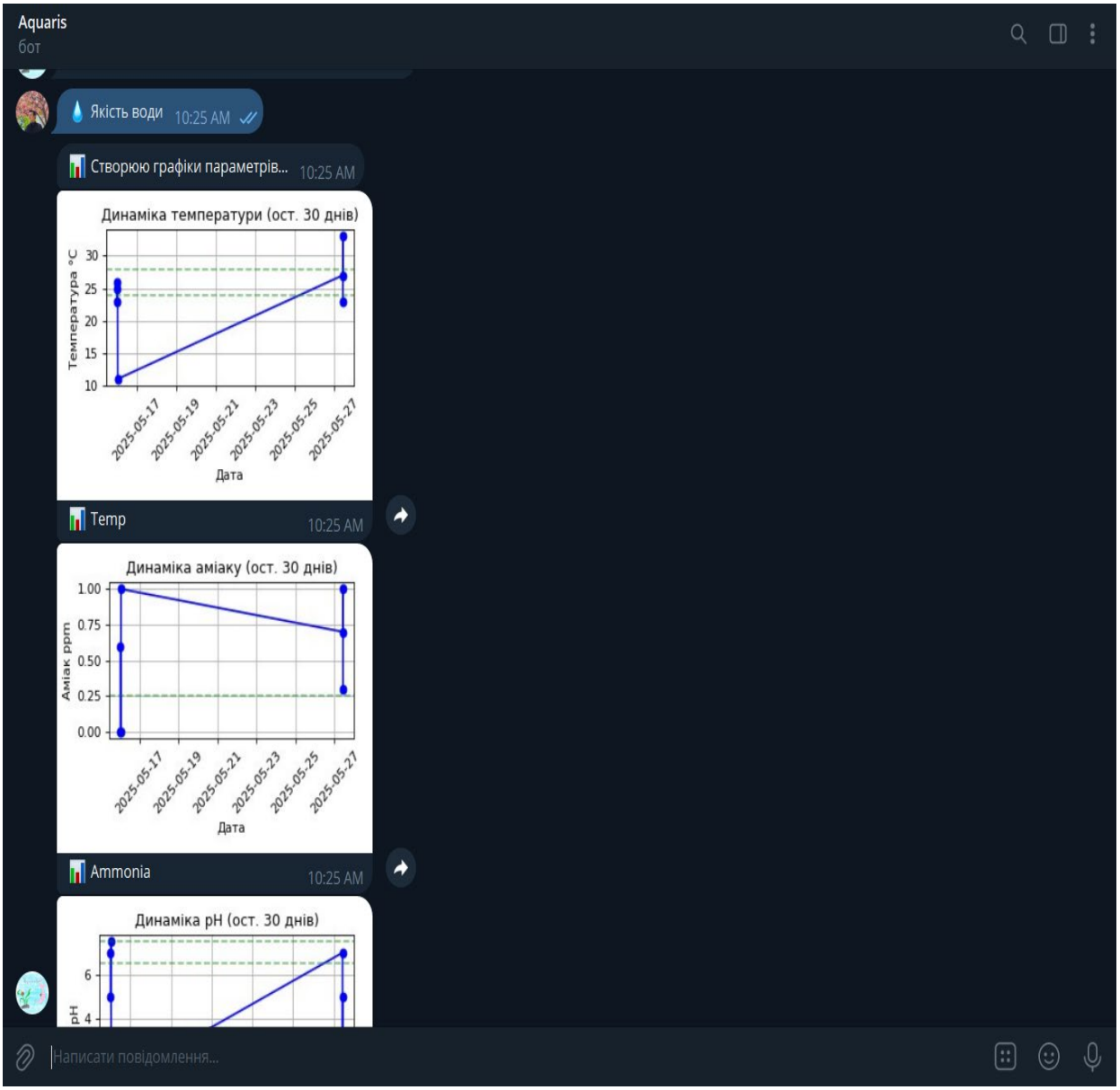


Рисунок 3.7 – Надісланий ботом графік параметрів води

Крок 5. Перевірка сумісності риб

Запустіть розділ «Сумісність риб», введіть пару (наприклад, неон + скалярія) (рис 3.8). Перевірте, чи бот видає правильне попередження або позитивний результат.

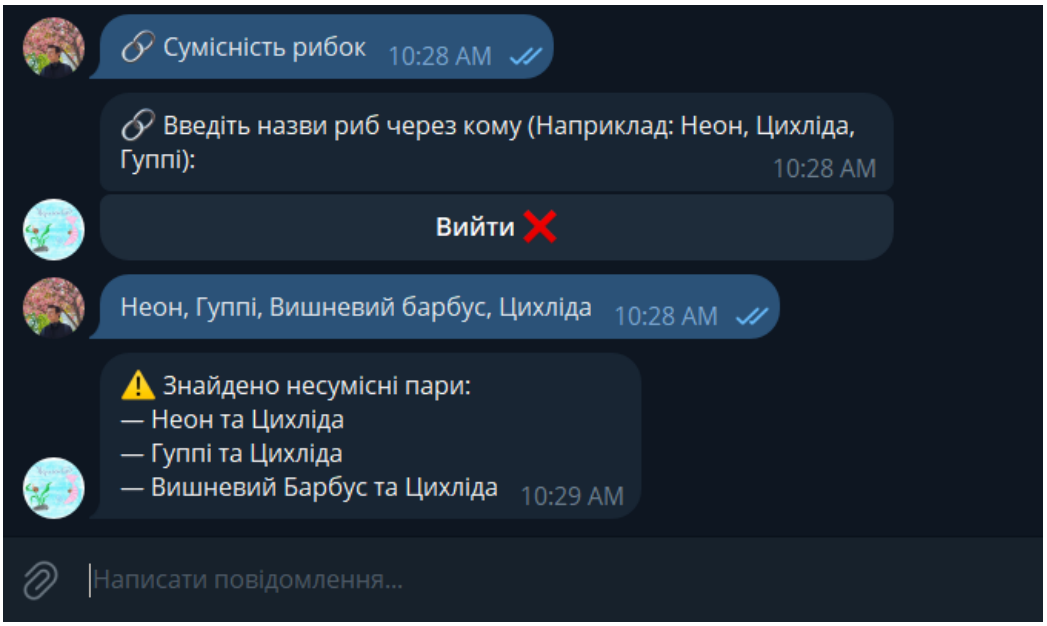


Рисунок 3.8 – Результат перевірки сумісності риб

Крок 6. Створення нагадування

У меню оберіть «Нагадування», вкажіть подію (наприклад, заміна води) і час (рис 3.9). Перевірте, чи бот створює нагадування з підтвердженням та чи надсилає його в заданий час (рис 3.10).

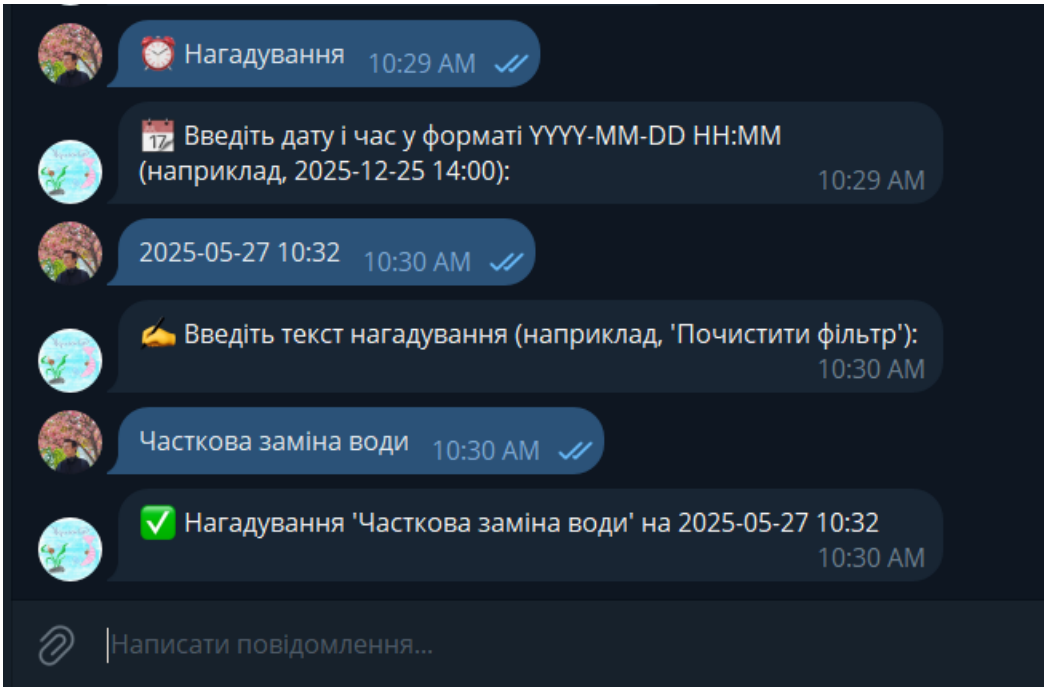


Рисунок 3.9 – Створене нагадування у боті


Рисунок 3.10 – Надіслане ботом нагадування

Крок 7. Діагностика хвороб

Запустіть діагностику, введіть симптом (наприклад, «плавники злиплися») (рис 3.11). Перевірте, чи бот пропонує відповідь (через OpenRouter API).

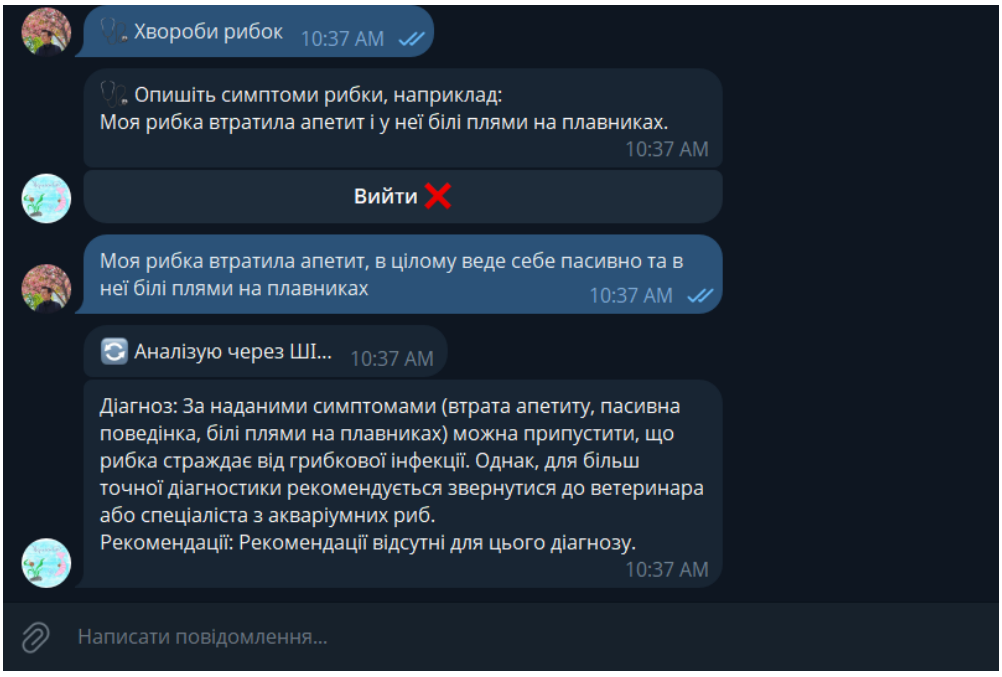


Рисунок 3.11 – Відповідь діагностики за введеним симптомом

Крок 8. Тест на помилкове введення

Спробуйте надіслати повідомлення, яке бот не обробляє (наприклад, «123456»). Переконайтесь, що бот відповідає з інструкцією або повідомленням про помилку (рис 3.12).

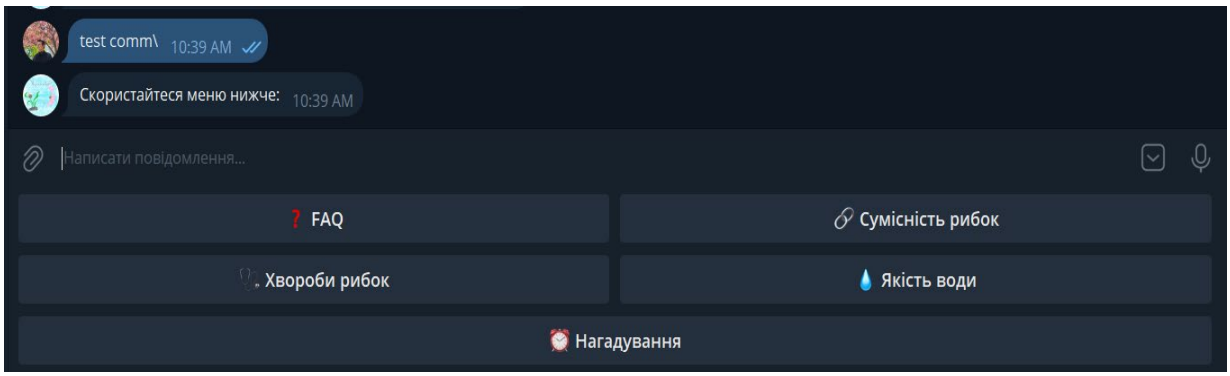


Рисунок 3.12 – Обробка невірної команди ботом

Під час ручного тестування було перевірено повний набір реалізованих функцій бота. Кожен модуль працює відповідно до очікуваної логіки. Користувачі отримують зрозумілу взаємодію та інформативні відповіді. У разі виявлення помилок чи аномалій, вони можуть бути локалізовані завдяки чіткій структурі коду.

### 3.5 Висновок до розділу 3

У цьому розділі було розглянуто процес реалізації основних функцій Telegram-бота для догляду за домашнім акваріумом. Детально описано підходи до реалізації ключових можливостей системи, зокрема функціоналу частих запитань (FAQ), перевірки сумісності риб, аналізу параметрів води, побудови графіків, нагадувань про обслуговування та первинної діагностики захворювань риб.

Для розробки було обрано мову програмування Python разом із бібліотекою aiogram, що забезпечує асинхронну обробку подій, гнучкість

структурування коду та підтримку інтеграції з Telegram API. Структуру застосунку реалізовано у вигляді модулів із чітким розділенням відповідальностей, що покращує підтримуваність і масштабованість проєкту. Для збереження даних використано SQLite як легку вбудовану базу даних, а для планування задач – APScheduler, що дозволило реалізувати персоналізовані нагадування.

Для аналізу якості води й відображення динаміки параметрів застосовано бібліотеку matplotlib, яка забезпечує візуалізацію даних у вигляді графіків. Додатково реалізовано обробку природної мови за допомогою інтеграції з AI-моделлю через API, що дозволяє покращити відповіді у випадках, коли база FAQ є недостатньою.

Проведено тестування бот-системи, що підтвердило її відповідність функціональним вимогам, зручність у користуванні та потенціал для подальшого розширення. Порівняння з існуючими аналогами показало конкурентоспроможність бота як доступного, адаптивного та практичного інструменту для акваріумістів-початківців і досвідчених користувачів.

Отже, реалізований Telegram-бот має три ключові переваги: інтуїтивно зрозумілий інтерфейс у популярному месенджері, модульну архітектуру, що забезпечує гнучке розширення функцій, та використання сучасного Python-стеку з можливостями AI-аналітики, що дозволяє ефективно підтримувати користувача в щоденному догляді за акваріумом.

# ВИСНОВКИ

У процесі виконання бакалаврської кваліфікаційної роботи було досліджено сучасні рішення у сфері догляду за домашніми тваринами та акваріумами, проаналізовано наявні платформи для створення чат-ботів і виявлено відсутність спеціалізованих рішень для комплексного догляду за акваріумом у формі Telegram-бота. На основі цього сформульовано вимоги до функціональності, структури й архітектури майбутньої системи.

В результаті дослідження, проєктування та реалізації було створено Telegram-бота для догляду за домашнім акваріумом, який поєднує декілька ключових функцій:

- надання відповідей на часті запитання за допомогою локальної бази знань із можливістю уточнення запитів;
- ведення обліку параметрів води та побудова графіків для їх візуального аналізу;
- перевірка сумісності риб на основі заздалегідь сформованих правил;
- створення нагадувань щодо регулярного обслуговування акваріума;
- базова діагностика хвороб риб за описом симптомів з використанням зовнішнього API.

Для реалізації проєкту було обрано мову програмування Python та платформу Telegram у зв’язку з її відкритим API, популярністю серед користувачів і зручністю реалізації ботів. Архітектура бота побудована на модульній основі, що забезпечує гнучкість і можливість масштабування проєкту. Застосовано бібліотеки aiogram, APScheduler, matplotlib та інші.

Мануальне тестування довело коректність роботи основних функцій: бот реагує на команди та повідомлення згідно із закладеною логікою, забезпечує зручну взаємодію з користувачем і демонструє стабільну роботу.

Усі поставлені на початку дослідження завдання були успішно виконані:

1. Обґрунтовано доцільність створення чат-бота для догляду за акваріумом.
2. Спроєктовано архітектуру та функціональну структуру системи.
3. Реалізовано чат-бота засобами Python та Telegram API.
4. Проведено тестування та проаналізовано результати роботи.
5. Розроблено інструкцію користувача для ефективної взаємодії з ботом.

Таким чином, поставлену мету – було досягнуто. Створений бот вирішує низку практичних задач і може бути основою для подальшого розвитку, зокрема шляхом додавання машинного навчання для покращення діагностики хвороб, інтеграції з хмарними сервісами або мобільними застосунками.

Бакалаврську кваліфікаційну роботу було оформлено відповідно до вимог і методичних вказівок [23], що гарантує її відповідність академічним стандартам.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Романовський К.В., Перепелиця В.І. Перспективи розробки чат-боту для догляду за домашнім акваріумом. *LIV Всеукраїнська науково-технічна конференція факультету інтелектуальних інформаційних технологій та автоматизації.* URL: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2025/paper/view/24045/19870 (дата звернення: 12.05.2025).

2. Enhancing Pet Care with AI [Електронний ресурс] – звернення: Режим доступу: https://fastbots.ai/blog/enhancing-pet-care-with-ai-the-role-of-veterinary-chatbots (дата звернення: 12.05.2025).

3. AI Chatbots in Pet Healthcare: Benefits and Challenges for Owners [Електронний ресурс] – звернення: Режим доступу:https://surl.li/hoqjbz (дата звернення: 11.05.2025).

4. Computer in Human Behavior (2021) [Електронний ресурс] – звернення: Режим доступу: https://www.sciencedirect.com/journal/computers-in-human behavior/vol/114/suppl/C (дата звернення: 10.05.2025).

5. 2024 State of Mobile [Електронний ресурс] – звернення: Режим доступу:

https://sensortower.com/blog/state-of-mobile-2024?utm_source=chatgpt.com (дата звернення: 14.05.2025).

6. The best AI chatbots in 2025 [Електронний ресурс] – звернення: Режим доступу: https://zapier.com/blog/best-ai-chatbot/ (дата звернення: 12.05.2025).

7. Artificial Intelligence in Ecology [Електронний ресурс] – звернення: Режим доступу: https://esajournals.onlinelibrary.wiley.com/doi/full/10.1002/bes2.2097 (дата звернення: 9.05.2025).

8. Global smart aquarium devices market report overview [Електронний ресурс] – звернення: Режим доступу: https://surl.li/xzofez (дата звернення: 12.05.2025).

9. The Future of Pet Care: 8 Ways a Chatbot is Reshaping the Pet Industry [Електронний ресурс] – звернення: Режим доступу: https://surl.li/odmyfi (дата звернення: 14.05.2025).

10. How to make a bot [Електронний ресурс] – звернення: Режим доступу: https://www.mindk.com/blog/how-to-develop-a-chat-bot/ (дата звернення: 14.05.2025).

11. Designing a contextual chatbot in Telegram with Python [Електронний ресурс] – звернення: Режим доступу: https://surl.li/tcbngf (дата звернення: 15.05.2025).

12. Chatbot System Architecture, Moataz Mohammed, Mostafa M. Aref [Електронний ресурс] – звернення: Режим доступу: https://surl.li/zjmgai (дата звернення: 15.05.2025).

13. What do you want to automate with OpenRouter and ChatBot? [Електронний ресурс] – звернення: Режим доступу:https://pipedream.com/ apps/openrouter/integrations/ chatbot?utm_source=chatgpt.com (дата звернення: 17.05.2025).

14. Різня між Javascript та Python. [Електронний ресурс] – звернення: Режим доступу: https://wezom.com.ua/ua/blog/nodejs-protiv-python-chto-vybrat (дата звернення: 18.05.2025).

15. Pycharm [Електронний ресурс] – звернення: Режим доступу: https://www.jetbrains.com/pycharm/features/ (дата звернення: 16.05.2025).

16. 5 Tips for Building Chatbots with Python and Aiogram 3.x [Електронний ресурс] – звернення: https://python.plainenglish.io/5-tips-for-building-chatbots-with-python-and-aiogram-3-x-d34feeb18d2f (дата звернення: 16.05.2025).

17. Як створити Telegram бота на Python [Електронний ресурс] – звернення: Режим доступу: https://foxminded.ua/telehram-bot-na-python/ (дата звернення: 13.05.2025).

18. How to create a Python Telegram bot that sends scheduled messages [Електронний ресурс] – звернення: Режим доступу: https://community .latenode.com/t/how-to-create-a-python-telegram-bot-that-sends-scheduled-messages/14040 (дата звернення: 13.05.2025).

19. Creating a Telegram Reminder Bot with Python [Електронний ресурс] – звернення: https://medium.com/%40thedevp/creating-a-telegram-reminder-bot-with-python-a280958b574b (дата звернення: 14.05.2025).

20. Automatically send all Matplotlib plots to your phone via telegram-send [Електронний ресурс] – звернення: https://blog.llandsmeer.com/tech/2021/04/06/matplotlib-telegram.html?utm_source =chatgpt.com (дата звернення: 12.05.2025).

21. difflib – Helpers for computing deltas [Електронний ресурс] – звернення: https://docs.python.org/3/library/difflib.html?utm_source=chatgpt.com (дата звернення: 18.05.2025).

22. BotFather. Можливості, команди та функціонал. [Електронний ресурс]: https://gerabot.com/article/botfather_mozhlivosti_ta_funkcional (дата звернення: 17.05.2025).

23. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів спеціальності 122 «Комп'ютерні науки» [Електронний ресурс] / уклад.: А. А. Яровий, О. В. Сілагін, І. В. Богач. – Вінниця : ВНТУ, 2023. 54 с.

# ДОДАТКИ

## ДОДАТОК А (обов'язковий)
## ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Розробка чат-боту для догляду за акваріумом через платформу Telegram

Тип роботи: бакалаврська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ кафедра комп'ютерних наук
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 0,44 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

- ✓ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

- ☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

- ☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

Яровий А.А., зав. каф. КН _____ _____
(прізвище, ініціали, посада) (підпис)

Колесницький О.К., проф. каф КН _____ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Озеранський В.С.
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Перепелиця В.І.
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Романовський К.В.
(підпис) (прізвище, ініціали)

# ДОДАТОК Б (обов’язковий)

## Лістинг програми

```
import logging
import datetime
import asyncio
from telegram import Update, ReplyKeyboardMarkup
from telegram.ext import (
    ApplicationBuilder, CommandHandler,
    MessageHandler, filters, ConversationHandler
)
from apscheduler.schedulers.background import BackgroundScheduler

from config import BOT_TOKEN
from handlers.faq import get_faq_conversation_handler
from handlers.compatibility import get_compat_conversation_handler
from handlers.disease import get_disease_conversation_handler
from handlers.water_params import get_water_conversation_handler
from handlers.reminders import get_reminders_conversation_handler, send_due_reminders
from database.db import init_db

# Налаштування логування
logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)

# Ініціалізуємо БД
init_db()

# Головне меню
main_menu = ReplyKeyboardMarkup(
    [
        ["❓ FAQ", "🔗 Сумісність рибок"],
        ["🩺 Хвороби рибок", "💧 Якість води"],
        ["⏰ Нагадування"]
    ],
    resize_keyboard=True
)

async def start(update: Update, context):
    await context.bot.send_photo(
        chat_id=update.effective_chat.id,
        photo=open('data/welcome.jpg', 'rb'),
        caption="👋 Ласкаво просимо до Aquaris Bot!"
    )
    await update.message.reply_text("Оберіть функцію:", reply_markup=main_menu)

async def menu_fallback(update: Update, context):
    logger.info(f"Fallback triggered for user {update.effective_user.id}: {update.message.text}")
    if context.user_data.get('conversation_state'):
        await update.message.reply_text(
            "Будь ласка, завершіть поточну дію або виберіть 'Вийти' у меню."
        )
        return
    await update.message.reply_text("Скористайтеся меню нижче:", reply_markup=main_menu)

async def cancel_conversation(update: Update, context):
    logger.info(f"Cancel conversation for user {update.effective_user.id}")
    context.user_data.clear()
    await update.message.reply_text(
        "❎ Розмову завершено. Оберіть функцію:", reply_markup=main_menu
    )
    return ConversationHandler.END
```

```
async def run_async_reminders(bot):
    try:
        await send_due_reminders(bot)
    except Exception as e:
        logger.error(f"Error running reminders: {e}")

def schedule_reminders(bot, loop):
    # Функція для виклику асинхронної задачі через BackgroundScheduler
    asyncio.run_coroutine_threadsafe(run_async_reminders(bot), loop)

if __name__ == '__main__':
    app = ApplicationBuilder().token(BOT_TOKEN).build()

    # Команда /start
    app.add_handler(CommandHandler("start", start))

    # Команда /cancel
    app.add_handler(CommandHandler("cancel", cancel_conversation))

    # Меню-функції
    app.add_handler(get_faq_conversation_handler())
    app.add_handler(get_compat_conversation_handler())
    app.add_handler(get_disease_conversation_handler())
    app.add_handler(get_water_conversation_handler())
    app.add_handler(get_reminders_conversation_handler())

    # Загальний fallback
    app.add_handler(MessageHandler(filters.TEXT & ~filters.COMMAND, menu_fallback))

    # Планувальник
    scheduler = BackgroundScheduler(timezone="Europe/Kyiv")
    scheduler.add_job(
        schedule_reminders,
        trigger='interval',
        seconds=60,  # Перевіряти щохвилини
        next_run_time=datetime.datetime.now() + datetime.timedelta(seconds=10),
        args=[app.bot, asyncio.get_event_loop()]
    )
    scheduler.start()
    logger.info("✅ Scheduler запущений")

    logger.info("✅ Бот запущено")
    app.run_polling()
import json
import aiohttp
import os
import re
from telegram import Update, InlineKeyboardButton, InlineKeyboardMarkup
from telegram.ext import (
    ContextTypes, ConversationHandler, MessageHandler,
    CallbackQueryHandler, filters
)
from config import OPENROUTER_API_KEY, OPENROUTER_MODEL, OPENROUTER_ENDPOINT

ASK_QUESTION, CONFIRM_FAQ_MATCH, HANDLE_FEEDBACK = range(3)

faq_path = os.path.join(os.path.dirname(os.path.dirname(os.path.abspath(__file__))), 'data', 'faq.json')


def load_faq():
    with open(faq_path, 'r', encoding='utf-8') as f:
        return json.load(f)


def save_faq(data):
```

```
    with open(faq_path, 'w', encoding='utf-8') as f:
        json.dump(data, f, ensure_ascii=False, indent=4)


FAQ_DATA = load_faq()


async def faq_menu(update: Update, context: ContextTypes.DEFAULT_TYPE):
    context.user_data.clear()
    keyboard = [[InlineKeyboardButton("Вийти ✖", callback_data="CANCEL")]]
    await update.message.reply_text(
        "📋 Введіть питання про акваріум або натисніть Вийти:",
        reply_markup=InlineKeyboardMarkup(keyboard)
    )
    return ASK_QUESTION


async def faq_answer(update: Update, context: ContextTypes.DEFAULT_TYPE):
    query = update.message.text.lower()
    context.user_data['user_question'] = update.message.text

    for question, answer in FAQ_DATA.items():
        if question.lower() in query or query in question.lower():
            context.user_data['matched_question'] = question
            context.user_data['matched_answer'] = answer

            keyboard = InlineKeyboardMarkup([
                [InlineKeyboardButton("Так", callback_data="YES_MATCH"),
                 InlineKeyboardButton("Hi", callback_data="NO_MATCH")]
            ])
            await update.message.reply_text(
                f"❓ Ви мали на увазі:\n<b>{question}</b>?",
                reply_markup=keyboard,
                parse_mode='HTML'
            )
            return CONFIRM_FAQ_MATCH

    return await ask_ai(update, context)


async def handle_faq_confirmation(update: Update, context: ContextTypes.DEFAULT_TYPE):
    query = update.callback_query
    await query.answer()

    if query.data == "YES_MATCH":
        await query.edit_message_text(f"📋 {context.user_data['matched_answer']}")
        return ConversationHandler.END
    else:
        await query.edit_message_text("🔄 Запитуємо у асистента...")
        return await ask_ai(update, context)


async def ask_ai(update: Update, context: ContextTypes.DEFAULT_TYPE):
    question = context.user_data.get('user_question')
    # Відправляємо повідомлення перед запитом
    msg = await (update.message or update.callback_query.message).reply_text("🔄 Запитуємо у асистента...")

    headers = {"Authorization": f"Bearer {OPENROUTER_API_KEY}"}
    payload = {
        "model": OPENROUTER_MODEL,
        "messages": [
            {
                "role": "system",
                "content": "Ти асистент з акваріумного догляду. Відповідай українською, коротко (2-3 речення), чітко, без
списків, заголовків чи Markdown."
```

```
            },
            {"role": "user", "content": question}
        ],
        "max_tokens": 300
    }

    async with aiohttp.ClientSession() as session:
        try:
            resp = await session.post(OPENROUTER_ENDPOINT, json=payload, headers=headers)
            data = await resp.json()
            answer = data.get('choices', [{}])[0].get('message', {}).get('content', 'Вибач, не зміг знайти відповідь.')
            # Очищаємо залишкове форматування
            answer = re.sub(r'[#*-]+|\n\s*\n', ' ', answer).strip()
        except Exception as e:
            answer = f"Вибач, сталася помилка при зверненні до ШІ: {str(e)}"

    context.user_data['ai_answer'] = answer
    # Видаляємо повідомлення "Запитуємо у асистента"
    await msg.delete()
    await (update.message or update.callback_query.message).reply_text(f"🤖 {answer}")

    keyboard = InlineKeyboardMarkup([
        [InlineKeyboardButton("✅ Так", callback_data="FEEDBACK_YES"),
         InlineKeyboardButton("❌ Ні", callback_data="FEEDBACK_NO")]
    ])
    await (update.message or update.callback_query.message).reply_text("💬 Чи була ця відповідь корисною?",
                                                                       reply_markup=keyboard)
    return HANDLE_FEEDBACK


async def handle_feedback(update: Update, context: ContextTypes.DEFAULT_TYPE):
    query = update.callback_query
    await query.answer()

    if query.data == "FEEDBACK_YES":
        q = context.user_data.get("user_question")
        a = context.user_data.get("ai_answer")

        # Записати в json, якщо ще не існує
        if q and a and q not in FAQ_DATA:
            FAQ_DATA[q] = a
            save_faq(FAQ_DATA)
            await query.edit_message_text("✅ Дякуємо! Ми зберегли цю відповідь у базі.")
        else:
            await query.edit_message_text("✅ Дякуємо за ваш відгук!")

    elif query.data == "FEEDBACK_NO":
        await query.edit_message_text("❎ Добре, ми постараємось дати кращу відповідь наступного разу.")

    return ConversationHandler.END


async def faq_cancel(update: Update, context: ContextTypes.DEFAULT_TYPE):
    await update.callback_query.answer()
    await update.callback_query.edit_message_text("❎ Вихід з FAQ.")
    return ConversationHandler.END


def get_faq_conversation_handler():
    return ConversationHandler(
        entry_points=[
            MessageHandler(filters.Regex("^❓ FAQ$"), faq_menu)
        ],
        states={
            ASK_QUESTION: [
```

```
            MessageHandler(filters.TEXT & ~filters.COMMAND, faq_answer)
        ],
        CONFIRM_FAQ_MATCH: [
            CallbackQueryHandler(handle_faq_confirmation, pattern="^(YES_MATCH|NO_MATCH)$")
        ],
        HANDLE_FEEDBACK: [
            CallbackQueryHandler(handle_feedback, pattern="^(FEEDBACK_YES|FEEDBACK_NO)$")
        ],
    },
    fallbacks=[
        CallbackQueryHandler(faq_cancel, pattern="^CANCEL$")
    ]
)



import csv
import difflib
from telegram import Update, InlineKeyboardMarkup, InlineKeyboardButton
from telegram.ext import (
    ContextTypes, ConversationHandler, MessageHandler,
    CallbackQueryHandler, filters
)

# Стан конверсейшена
CHOOSING = 0

# Завантажуємо таблицю сумісності
FISH_COMPAT = {}
with open('data/fish.csv', 'r', encoding='utf-8') as f:
    reader = csv.reader(f)
    header = next(reader)[1:]
    for row in reader:
        fish = row[0].lower()
        FISH_COMPAT[fish] = dict(zip(
            [h.lower() for h in header], row[1:]
        ))
# Список усіх видів для підказок
ALL_FISH = list(FISH_COMPAT.keys())

async def compat_menu(update: Update, context: ContextTypes.DEFAULT_TYPE):
    """
    Показує діалог з кнопкою виходу і чекає на введення списку риб через кому.
    """
    keyboard = [[InlineKeyboardButton("Вийти ✖", callback_data="CANCEL")]]
    await update.message.reply_text(
        "🖇 Введіть назви риб через кому (Наприклад: Неон, Цихліда, Гуппі):",
        reply_markup=InlineKeyboardMarkup(keyboard)
    )
    return CHOOSING

async def compat_answer(update: Update, context: ContextTypes.DEFAULT_TYPE):
    """
    Обробляє введення користувачем. Виконує fuzzy matching для невідомих
    видів і перевіряє пари на несумісність.
    """
    text = update.message.text
    fishes = [s.strip().lower() for s in text.split(',') if s.strip()]

    known, unknown, suggestions = [], [], {}
    for fish in fishes:
        if fish in FISH_COMPAT:
            known.append(fish)
        else:
            # Пропозиція найближчих варіантів
            close = difflib.get_close_matches(fish, ALL_FISH, n=3, cutoff=0.6)
```

```
            if close:
                suggestions[fish] = close
            unknown.append(fish)

    if unknown:
        # Показуємо невідомі та пропозиції
        msg = "❌ Не знайдено: "
        for f in unknown:
            if f in suggestions:
                msg += f"— «{f}», можливо: {', '.join(suggestions[f])} "
            else:
                msg += f" — «{f}» "
        msg += "Спробуйте ще раз з правильною назвою."
        await update.message.reply_text(msg)
        return CHOOSING
        await update.message.reply_text(msg)
        return CHOOSING

    # Перевіряємо всі пари
    incompatible = []
    for i in range(len(known)):
        for j in range(i+1, len(known)):
            f1, f2 = known[i], known[j]
            status = FISH_COMPAT[f1].get(f2)
            if status == '❌':
                incompatible.append((f1.title(), f2.title()))

    # Формуємо відповідь
    if incompatible:
        msg = " ⚠️ Знайдено несумісні пари:\n"
        msg += '\n'.join(f"— {a} та {b}" for a, b in incompatible)
    else:
        msg = "✅ Усі введені рибки сумісні між собою."

    await update.message.reply_text(msg)
    return ConversationHandler.END

async def compat_cancel(update: Update, context: ContextTypes.DEFAULT_TYPE):
    """
    Вихід з діалогу сумісності.
    """
    await update.callback_query.answer()
    await update.callback_query.edit_message_text("❎ Вихід з Сумісності.")
    return ConversationHandler.END

from telegram.ext import CallbackQueryHandler as CQH, MessageHandler as MH


def get_compat_conversation_handler():
    return ConversationHandler(
        entry_points=[MH(filters.Regex(r"^🔗 Сумісність рибок$"), compat_menu)],
        states={CHOOSING: [MH(filters.TEXT & ~filters.COMMAND, compat_answer)]},
        fallbacks=[CQH(compat_cancel, pattern="^CANCEL$")]
    )
```

**ДОДАТОК В (обов'язковий)**

# ГРАФІЧНА ЧАСТИНА

# «РОЗРОБКА ЧАТ-БОТ ДЛЯ ДОГЛЯДУ ЗА ДОМАШНІМ АКВАРІУМОМ ЧЕРЕЗ ПЛАТФОРМУ TELEGRAM»

Виконав: студент 4 курсу, групи 2КН-21б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Романовський К.В.
(прізвище та ініціали)

Керівник: доктор філософії, асистент кафедри КН
Перепелиця В.І.
(прізвище та ініціали)
«03» червня 2025 р.


Рисунок В.1 – Графік використання чат-ботів для догляду за тваринами, рослинами та акваріумами.


Рисунок В.2 – Графік популярності месенджерів для чат-ботів у 2024 році.

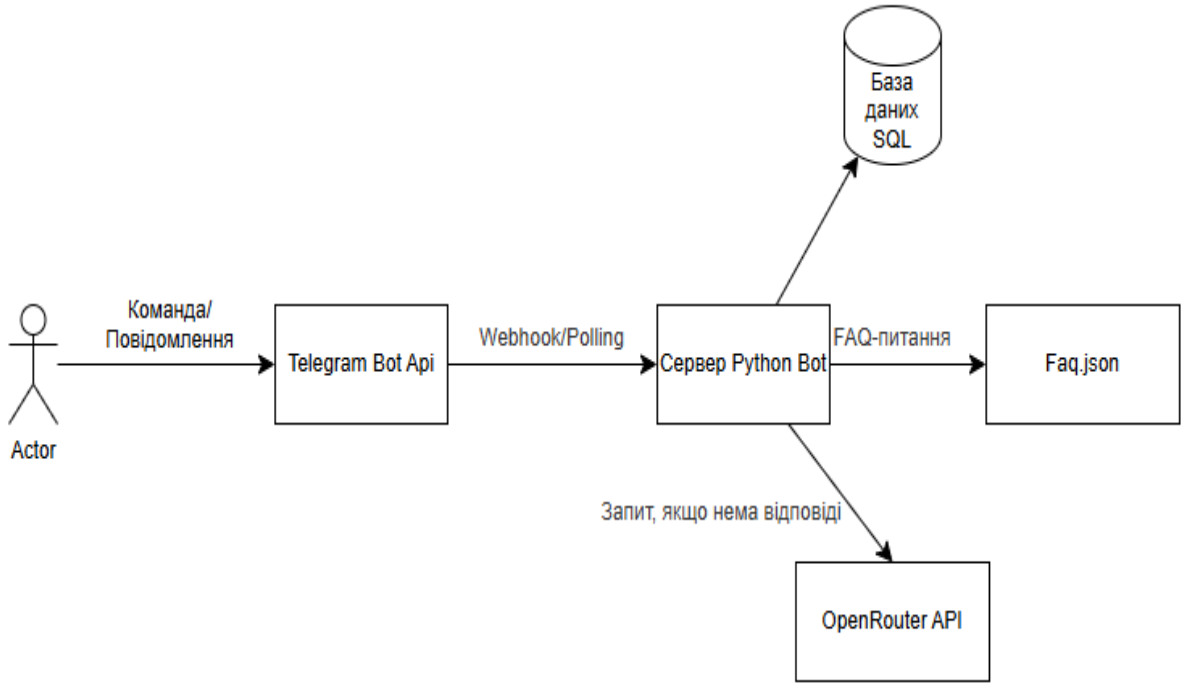


Рисунок В.3 – Схема роботи чат-бота

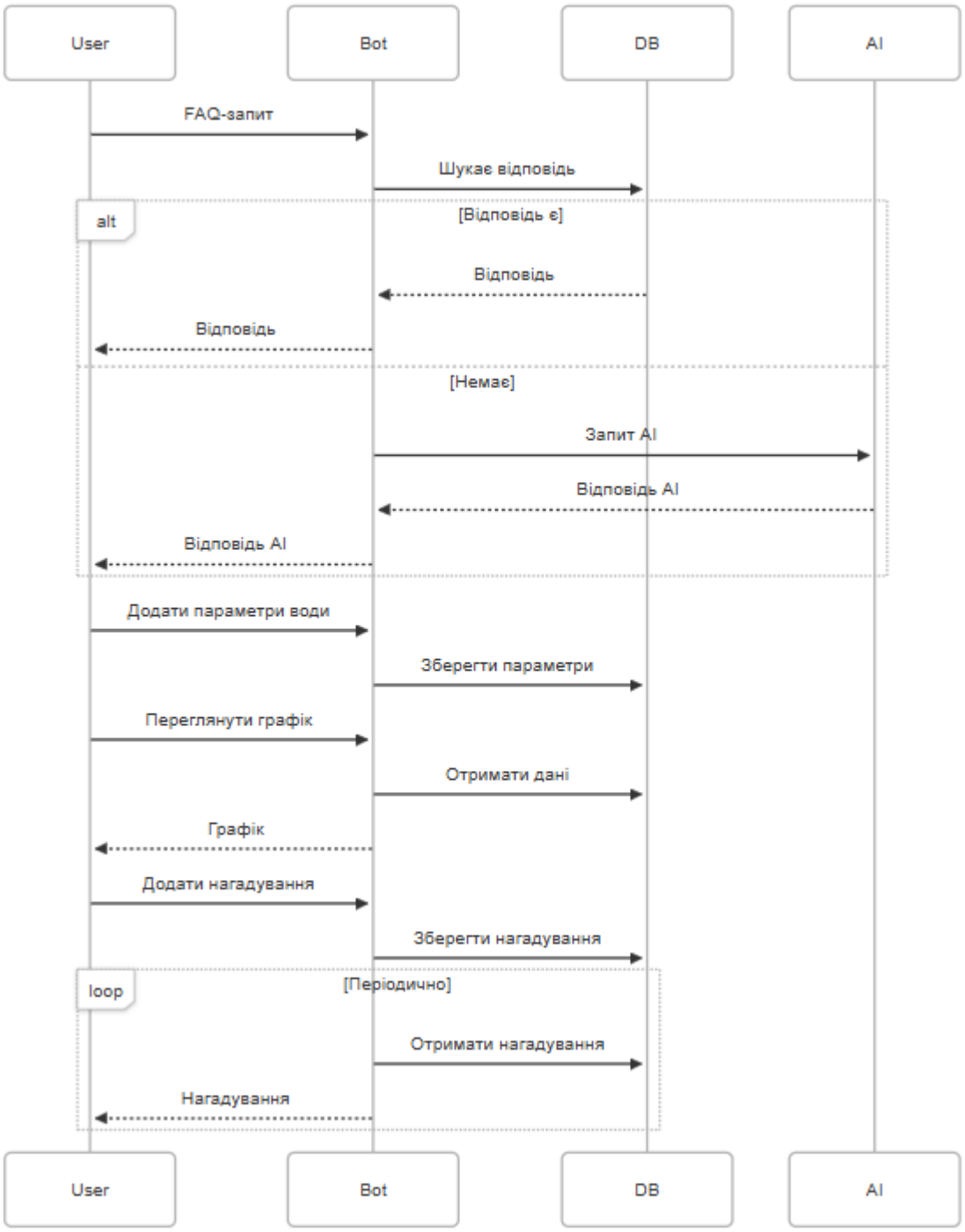


Рисунок В.4 – Діаграма послідовності чат-боту

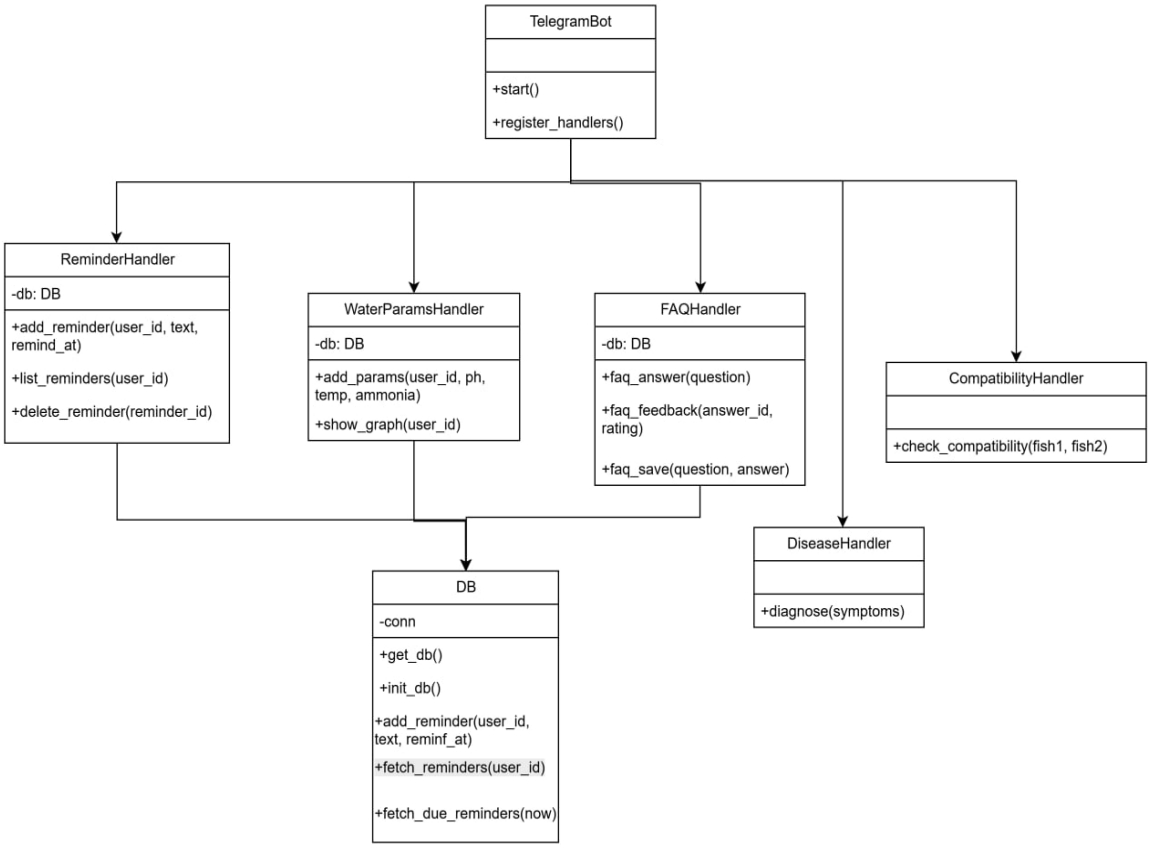


Рисунок В.5 – UML діаграма класів програми

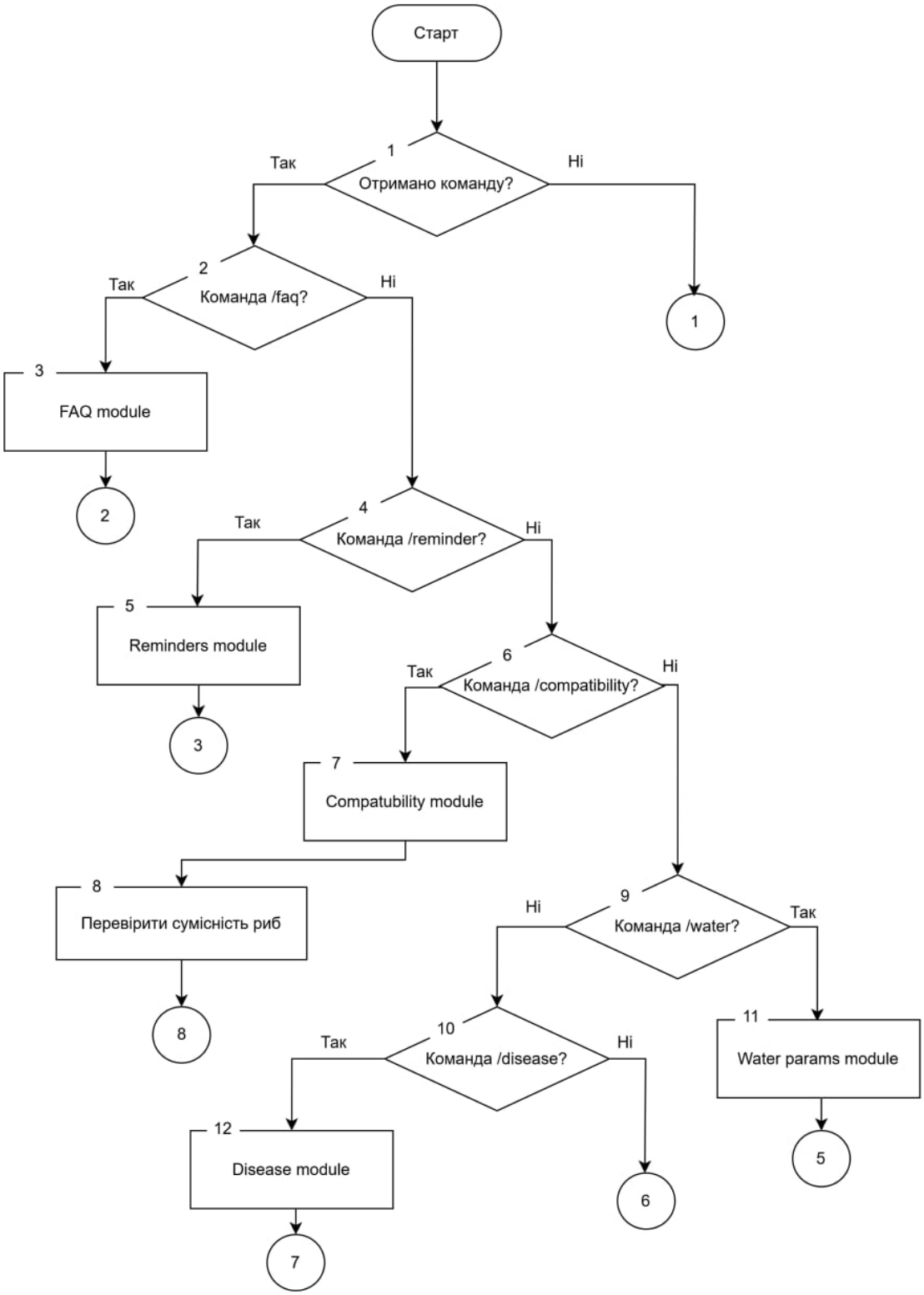


Рисунок В.6 – Загальний алгоритм взаємодії з чат-ботом

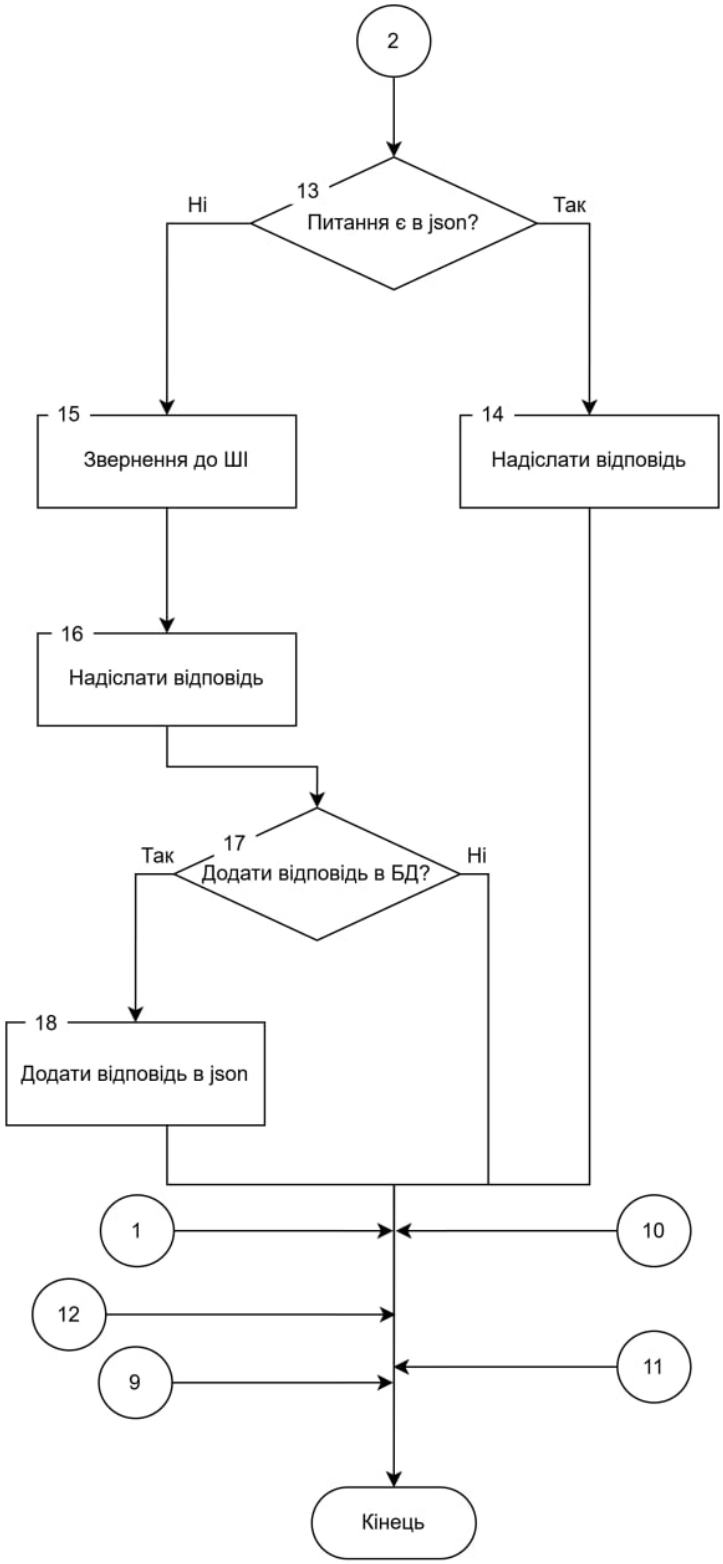


Рисунок В.6 – аркуш 2

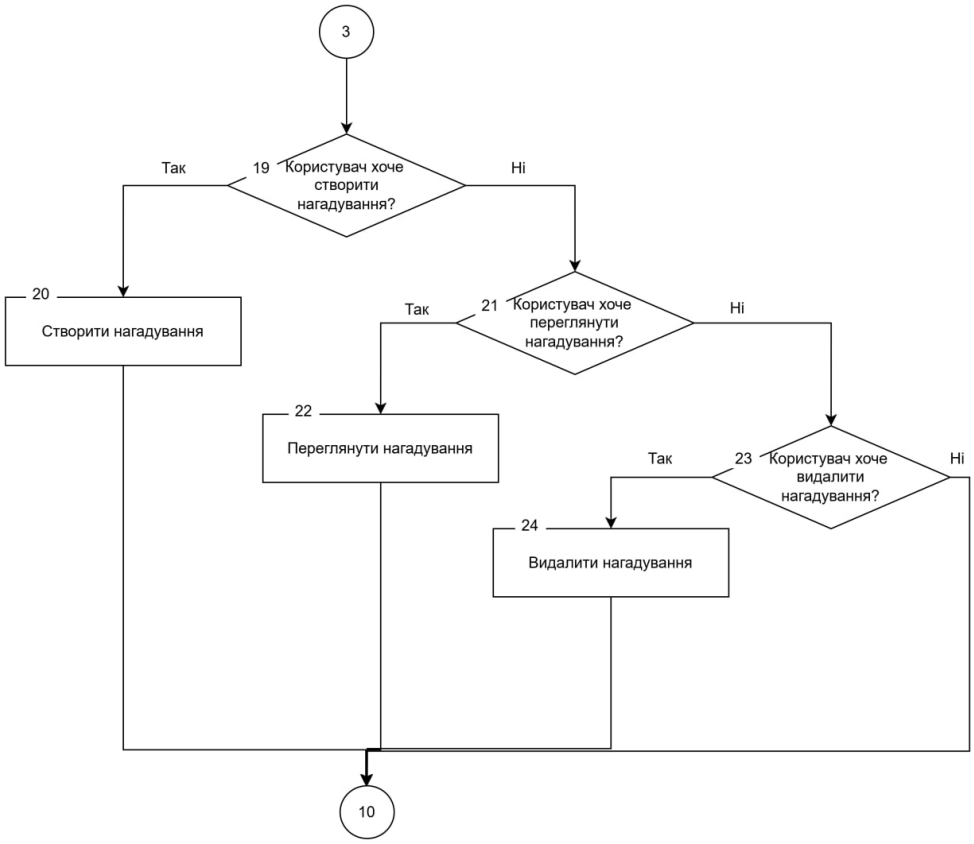


Рисунок В.6 – аркуш 3

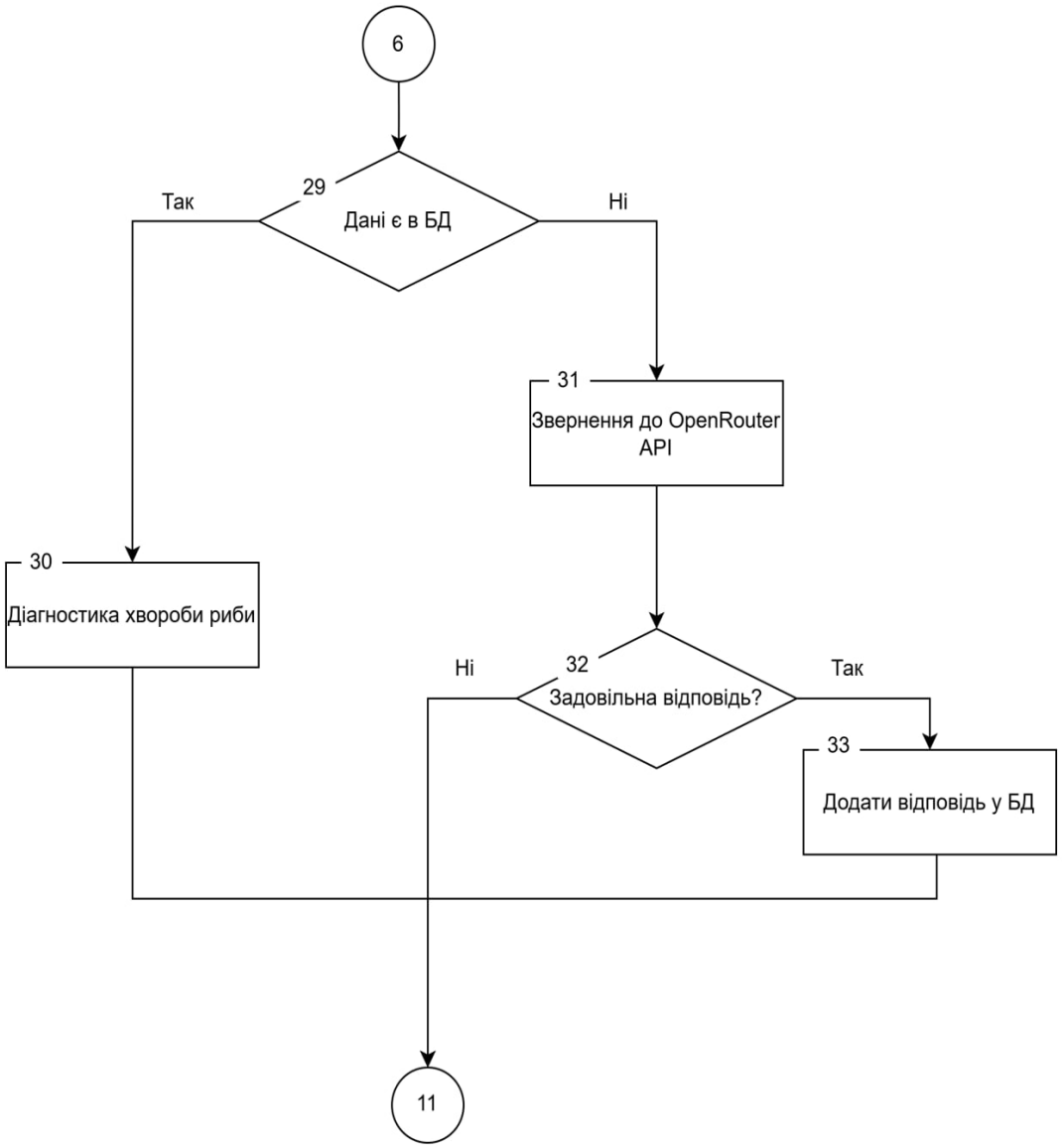


Рисунок В.6 – аркуш 4

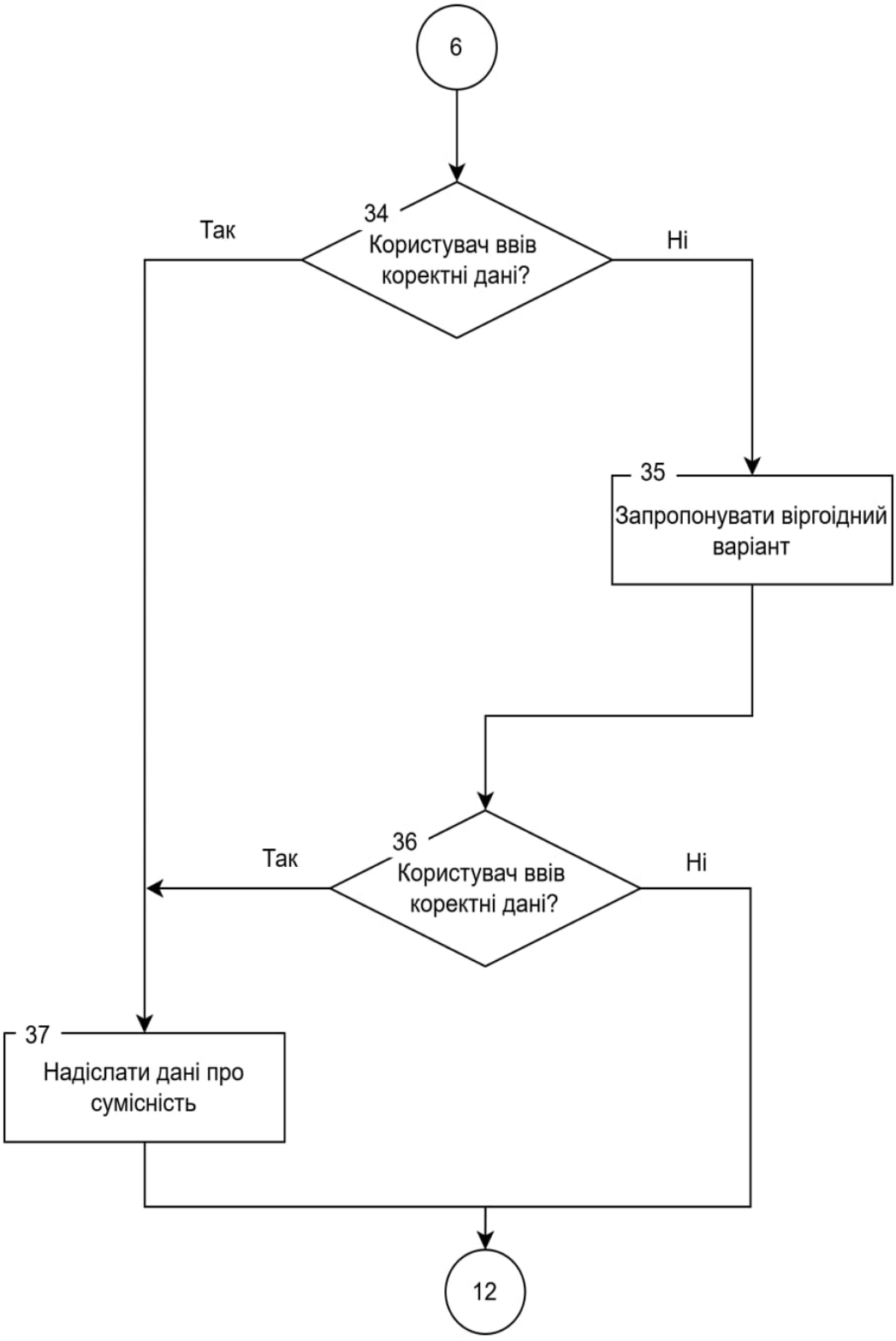


Рисунок В.6 – аркуш 5

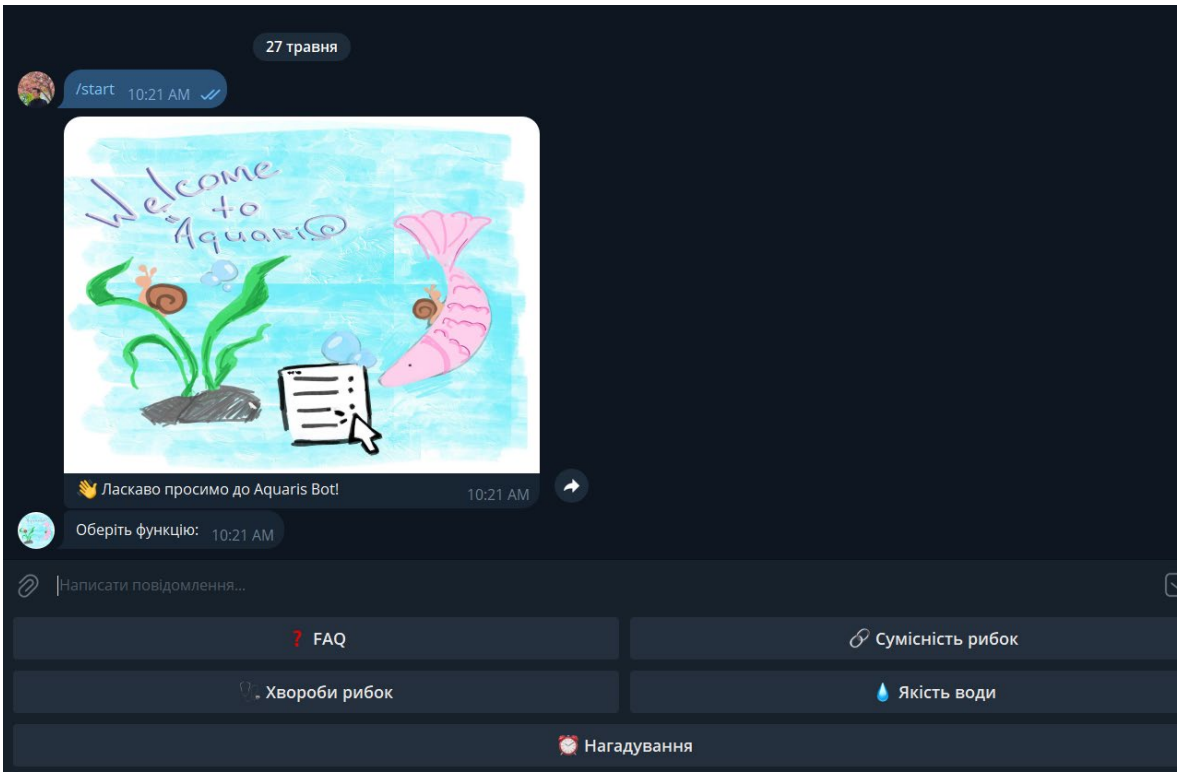


Рисунок В.7 – Головне меню бота після старту

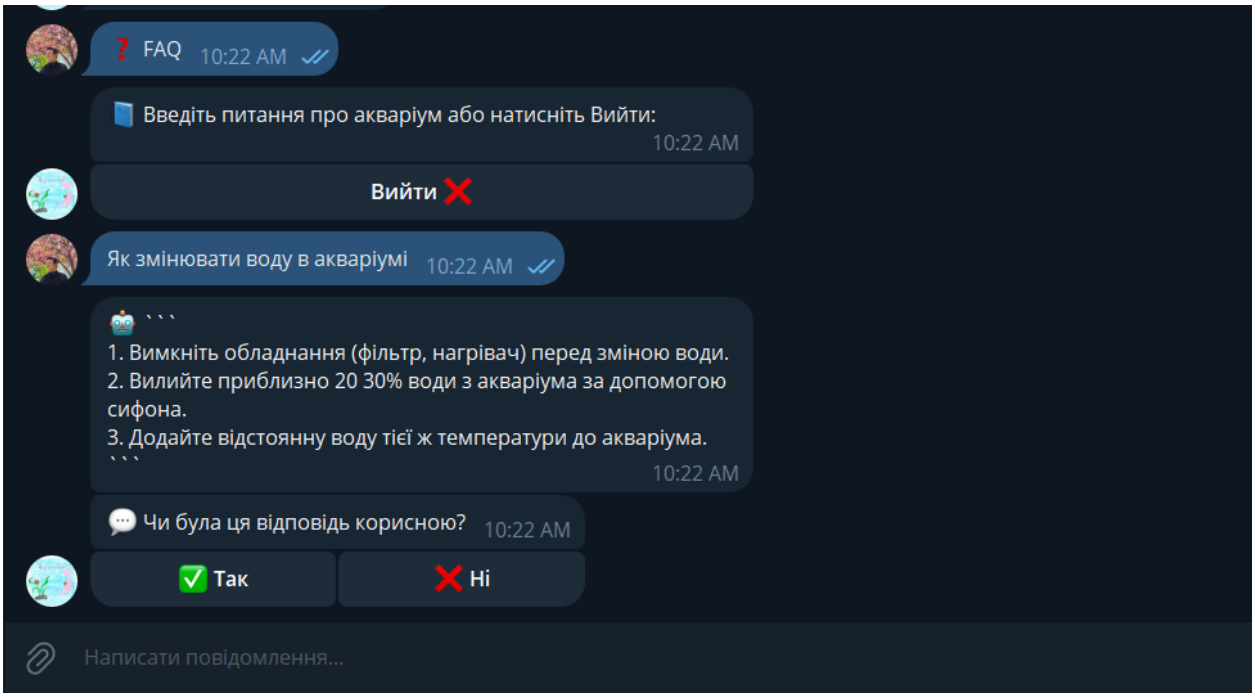


Рисунок В.8 – Взаємодія з розділом FAQ

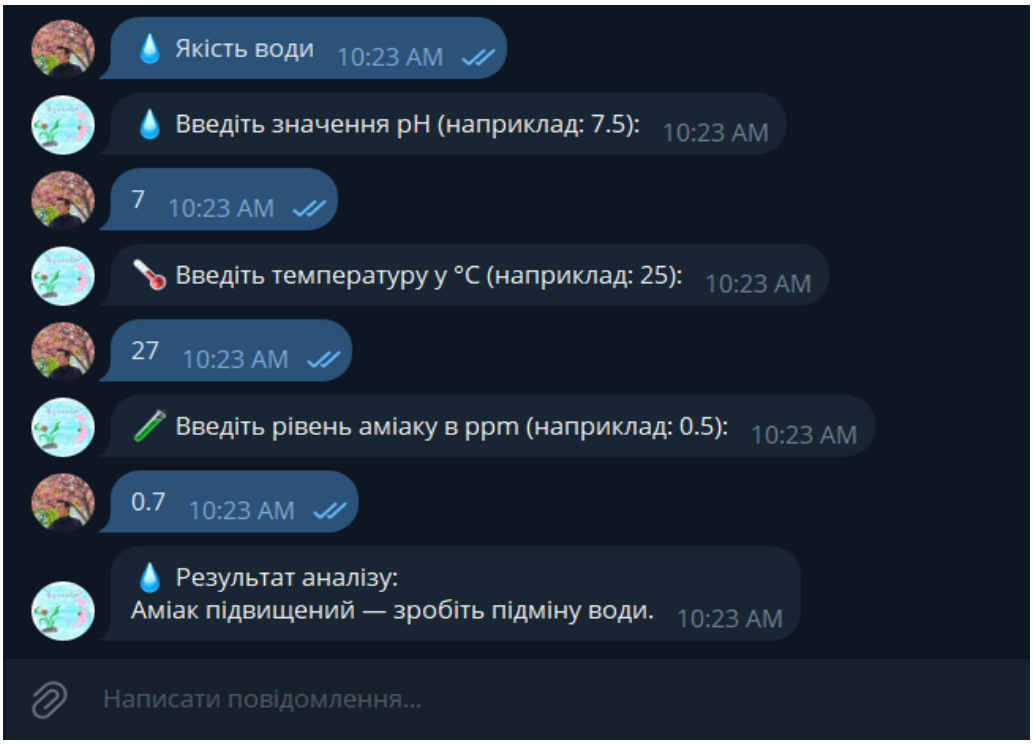


Рисунок В.9 – Введення параметрів води


Рисунок В.10 – Надісланий ботом графік параметрів води

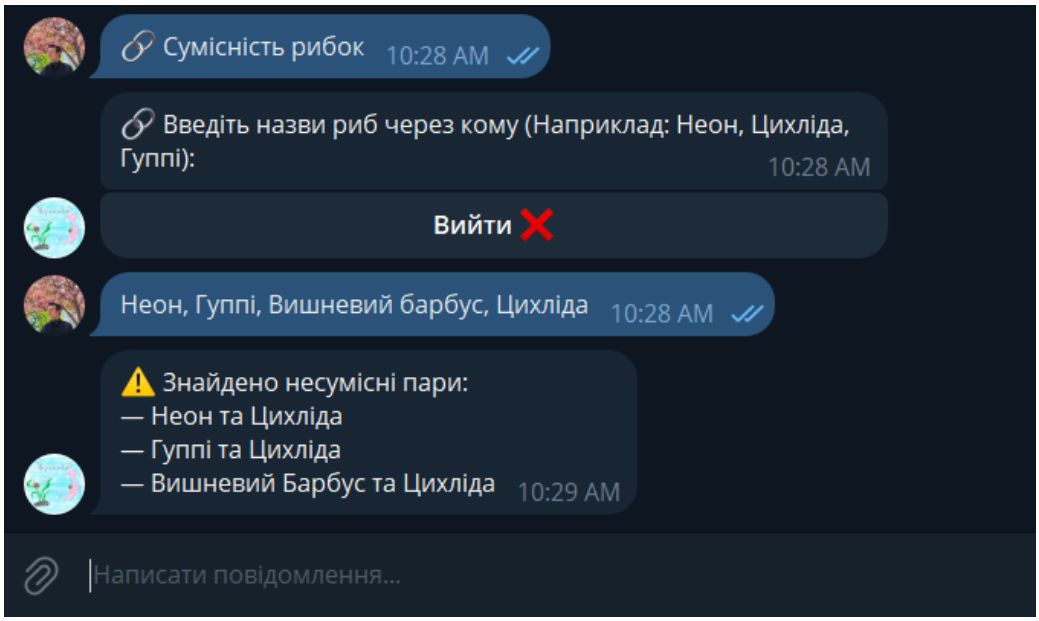


Рисунок В.11 – Результат перевірки сумісності риб

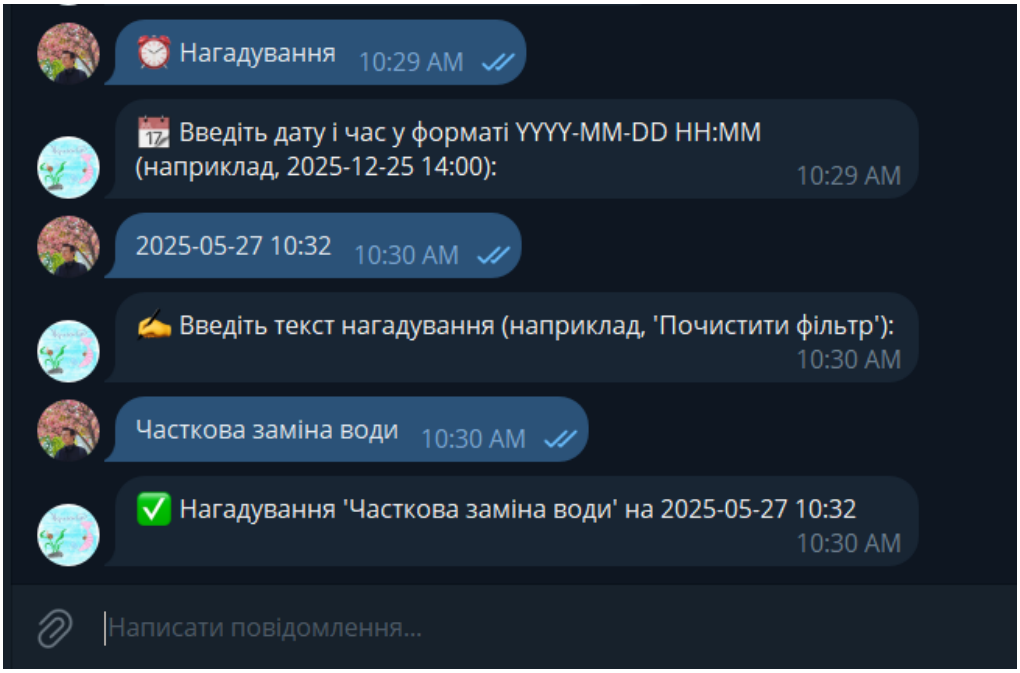


Рисунок В.12 – Створене нагадування у боті


Рисунок В.13 – Надіслане ботом нагадування

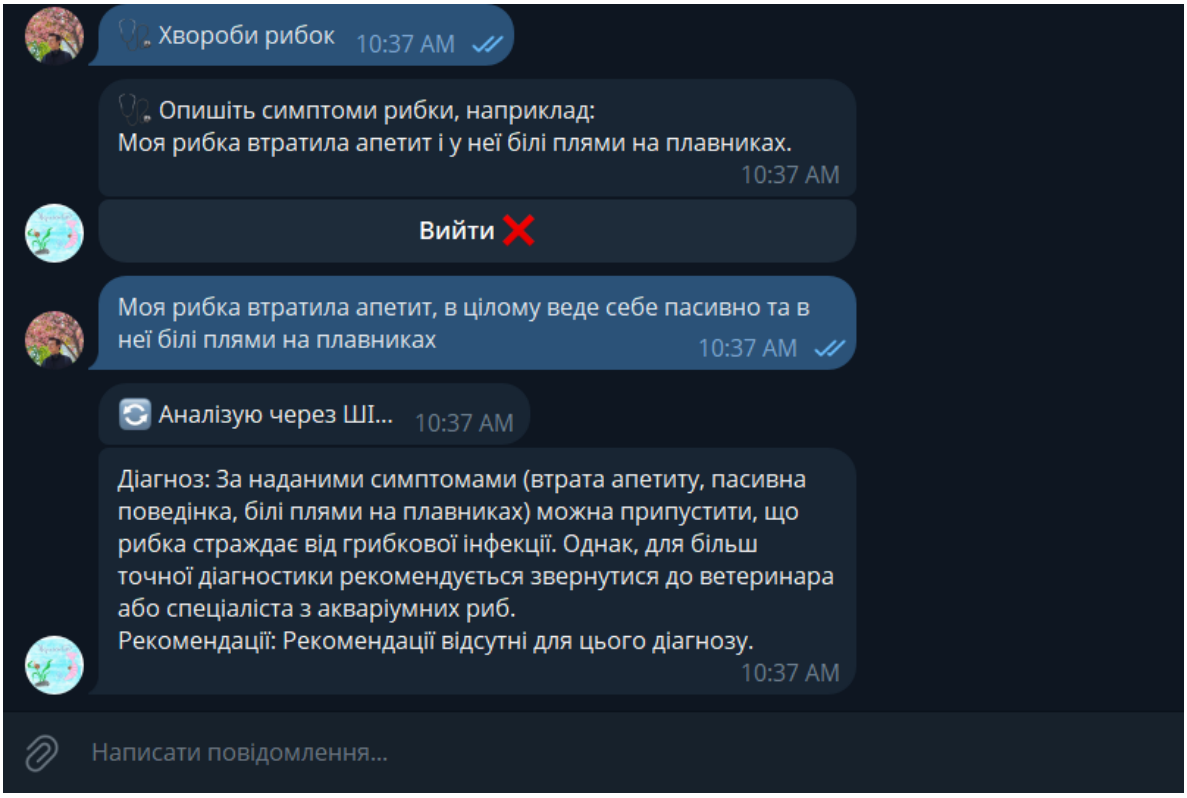


Рисунок В.14 – Відповідь діагностики за введеним симптомом

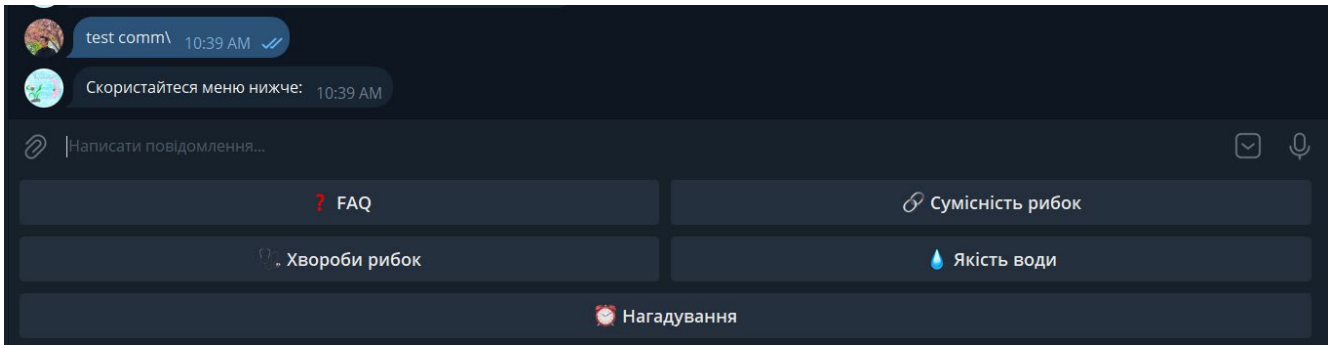


Рисунок В.15 – Обробка невірної команди ботом

# ДОДАТОК Г (довідниковий)

## Інструкція користувача

Після запуску чат-бота в Telegram користувачу відкривається головне меню, що містить перелік доступних функцій, зокрема:

- FAQ (Поширені запитання)
- Якість води
- Сумісність риб
- Нагадування
- Діагностика хвороб

### Доступ до функціональних можливостей

Щоб скористатися будь-якою функцією, користувачу потрібно натиснути на відповідну кнопку в меню або ввести команду вручну.

### Використання розділу «FAQ»

Користувач може поставити питання про утримання акваріума. Бот надає відповідь із локальної бази знань. Якщо відповідь недостатньо точна, бот запропонує уточнити запит або звернутися до розширеного джерела.

### Використання розділу «Якість води»

У цьому розділі користувачу доступні два варіанти:

- Додати параметри – бот послідовно запитує значення таких параметрів води, як pH, температура, нітрати тощо.
- Переглянути графік – бот надсилає графік змін параметрів води за останній період, якщо дані раніше були введені.

### Використання розділу «Сумісність риб»

Користувач вводить назви двох або більше видів риб, після чого бот перевіряє їхню сумісність на основі попередньо закладеної бази правил. У

результаті користувач отримує повідомлення з висновком про можливість спільного утримання риб.

**Використання розділу «Нагадування»**

Бот дозволяє створити нагадування про обслуговування акваріума, зокрема про заміну води, чистку фільтра, перевірку температури тощо. Для створення нагадування потрібно:

1. Вибрати тип нагадування.
2. Вказати періодичність (наприклад, раз на 7 днів).
3. Підтвердити створення.

Коли настає час обслуговування, бот надсилає повідомлення у вигляді нагадування.

**Використання розділу «Діагностика хвороб»**

Користувач описує симптоми, які він спостерігає у риб. Бот надсилає опис до зовнішнього API та повертає найбільш імовірну причину та рекомендації щодо лікування.

Система не є медичною, тому результати носять інформативний характер.