Хмельницький національний університет
Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення

# КВАЛІФІКАЦІЙНА РОБОТА

Telegram-бот для продажу абонементів і обліку відвідувань у спортивному клубі

Назва теми

Рівень вищої освіти Перший (бакалаврський)

Галузь знань 12 «Інформаційні технології»

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

Шифр КвРІПЗ.2201100.01.05.ПЗ

Виконав студент IV курсу, група ІПЗ-22-1 _____ Леонід ГОНТАР
Підпис Ім'я, ПРІЗВИЩЕ

Керівник асистент _____ В'ячеслав БОЙКО
Науковий ступінь, вчене звання Підпис Ім'я, ПРІЗВИЩЕ

Нормоконтролер канд. пед. наук, доцент _____ Наталія ПРАВОРСЬКА
Підпис Ім'я, ПРІЗВИЩЕ

**До захисту допускаю:**
Завідувач кафедри інженерії
програмного забезпечення _____ Леонід БЕДРАТЮК
Підпис Ім'я, ПРІЗВИЩЕ

1 червня 2026 р.

Хмельницький 2026

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій
Кафедра Інженерії програмного забезпечення
Рівень вищої освіти Перший (бакалаврський)
Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри ІПЗ
_____ Л. П. Бедратюк
07. 01 2026 р.

**ЗАВДАННЯ**
**НА КВАЛІФІКАЦІЙНУ РОБОТУ**

Гонтару Леоніду Сергійовичу
Прізвище, ім'я, по батькові студента

1. Тема роботи Telegram-бот для продажу абонементів і обліку відвідувань у спортивному клубі

Керівник роботи Бойко В'ячеслав Олександрович, асистент
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 20.01.2026 р. № 7
2. Строк подання студентом роботи на кафедру 01.06.2026 р.
3. Вихідні дані до роботи Матеріали переддипломної практики
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити) Дослідження предметної області та постановка задач, проєктування програмного забезпечення, програмна реалізація та тестування бота.

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень) Три креслення
1. Діаграма варіантів використання
2. Діаграма функціоналу бота
3. Діаграма баз даних

6. Консультанти розділів кваліфікаційної роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| Нормоконтроль | Праворська Н. І., канд. пед. наук, доцент | 05.05.26 | 08.05.26 |
| Антиплагіат | Форкун Ю. В., доцент | 05.05.26 | 21.05.26 |

7. Дата видачі завдання « 20 » січня 2026 р.

КАЛЕНДАРНИЙ ПЛАН

| Назва етапів (розділів) кваліфікаційної роботи | Строк виконання етапів роботи | Примітка |
|---|---|---|
| 1 Ознайомлення з тематикою дипломного проєктування, визначення та узгодження індивідуальної теми кваліфікаційної роботи (КвР) | 01.12– 31.12.2025 | |
| 2 Збір матеріалу за темою КвР; дослідження предметної області, в якій планується використання програмного забезпечення (ПЗ), визначення задач та вимог. | 01.01 – 20.02.2026 | |
| 3 Проєктування програмного забезпечення | 21.02 – 20.03 2026 | |
| 4 Програмна реалізація з використанням відповідних засобів розроблення. Тестування ПЗ | 21.03 – 30.04.2026 | |
| 5 Написання вступу, загальних висновків, оформлення переліку джерел посилання та додатків. Оформлення пояснювальної записки КвР згідно вимог | 01.05 – 25.05.2026 | |
| 6 Попередній захист КвР | Травень | Згідно графіка |
| 7 Перевірка КвР на плагіат, нормоконтроль, отримання відгуків, рецензій та інших супровідних документів. Брошурування (зшиття) пояснювальної записки. | 26.05 – 30.05.2026 | |
| 8 Здача КвР на кафедру; підготовка КвР для розміщення у репозитарії ХНУ; підготовка до захисту та захист КвР | з 01.06.2026 | |

Студент _____ Леонід ГОНТАР
Підпис Ім'я, ПРІЗВИЩЕ

Керівник роботи _____ В'ячеслав БОЙКО
Підпис Ім'я, ПРІЗВИЩЕ

# АНОТАЦІЯ

Тема кваліфікаційної роботи: Telegram-бот для продажу абонементів і обліку відвідувань у спортивному клубі.

Автор роботи: Гонтар Леонід Сергійович

Керівник роботи: Бойко В'ячеслав Олександрович

Пояснювальна записка: 89 с., 16 рис., 4 дод., 28 джерел.

Графічна частина: 3 креслення.

TELEGRAM-БОТ ДЛЯ ПРОДАЖУ АБОНЕМЕНТІВ І ОБЛІКУ ВІДВІДУВАНЬ У СПОРТИВНОМУ КЛУБІ

Метою роботи є розробка та реалізація функціонального Telegram-бота LEO FITNESS, який автоматизує процеси продажу абонементів, реєстрацію відвідувань та управління клієнтською базою спортивного клубу.

Для досягнення поставленої мети вирішено такі основні задачі:

проаналізовано предметну область та існуючі рішення для автоматизації спортивних клубів, спроектовано архітектуру програмного забезпечення та структуру бази даних, розроблено та реалізовано основний функціонал бота (продаж абонементів, особистий кабінет клієнта, check-in відвідувань, адмін-панель), проведено тестування програмного продукту та оцінено його ефективність.

Об'єкт дослідження – процеси управління послугами та обліком відвідувань у спортивному клубі. Предмет дослідження – програмне забезпечення Telegram-бота для автоматизації цих процесів.

У роботі використано сучасні технології: мова програмування Python 3.12, асинхронна бібліотека aiogram 3.x, вбудована база даних SQLite3. Бот повністю асинхронний, що забезпечує високу продуктивність і стабільність роботи.

27.05.26
Дата

Підпис

## ВІДОМІСТЬ ДОКУМЕНТІВ

| № рядка | Формат | Позначення документа | Найменування документа | К-сть аркушів | № екз. | Примітка |
|---|---|---|---|---|---|---|
| | | | | | | |
| | | | Текстові документи | | | |
| | | | | | | |
| 1 | А4 | КвРІПЗ.2201100.01.05.ПЗ | Пояснювальна записка | 93 | | |
| 2 | А4 | | Завдання на кваліфіка- | 1 | | |
| | | | ційну роботу | | | |
| 3 | А4 | | Анотація | 1 | | |
| | | | | | | |
| | | | Графічні документи | | | |
| | | | | | | |
| 4 | А3 | КвРІПЗ.2201100.01.05.ПЗ | Діаграма варіантів | 1 | | |
| | | | використання | | | |
| 5 | А3 | КвРІПЗ.2201100.01.05.ПЗ | Діаграма функціоналу | 1 | | |
| | | | бота | | | |
| 6 | А3 | КвРІПЗ.2201100.01.05.ПЗ | Діаграма бази даних | 1 | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

| Змн. | Арк. | № докум. | Підпис | Дата | КвРІПЗ.2201100.01.05.ПЗ | | |
|---|---|---|---|---|---|---|---|
| Виконав | | Гонтар Л.С. | | 01.06.24 | Telegram-бот для продажу абонементів і обліку відвідувань у спортивному клубі | Літ. | |
| Керівник | | Бойко В.О. | | 01.06.24 | | Арк. 5 | Аркушів 89 |
| Рецензент | | | | | | ХНУ, ІПЗ-22-1 | |
| Н. контр. | | Праворська Н. І. | | 01.06.24 | | | |
| Зав. каф. | | Бедратюк Л.П. | | 01.06.24 | | | |

## ЗМІСТ


ВСТУП ..... 8

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ ... 10

1.1 Змістовий аналіз предметної області ..... 10

1.2 Аналіз наявного програмно-технічного забезпечення предметної області ..... 16

1.3 Визначення вимог до розроблюваного застосунку ..... 26

1.4 Висновки. Постановка задачі ..... 28

2 ПРОЄКТУВАННЯ СИСТЕМИ ..... 30

2.1 Вимоги до системи ..... 30

2.2 Архітектура програмного забезпечення ..... 38

2.3 Проєктування бази даних ..... 43

2.4 Моделювання бізнес-процесів та інтерфейсу користувача ..... 47

2.4.1 Моделювання бізнес-процесів ..... 48

2.4.2 Моделювання діяльності ..... 49

2.4.3 Проєктування інтерфейсу користувача ..... 49

2.5 Забезпечення безпеки даних ..... 50

2.6 Висновки ..... 53

3 РЕАЛІЗАЦІЯ TELEGRAM-БОТА LEO FITNESS ..... 55

3.1 Стек технологій та інструменти розробки ..... 55

3.2 Структура проєкту та організація коду ..... 65

3.3 Реалізація основного функціоналу ..... 67

3.4 Система продажу абонементів ..... 71

3.5 Облік відвідувань та особистий кабінет ..... 75

3.5.1 Реалізація особистого кабінету ..... 75
3.5.2 Реалізація системи обліку відвідувань ..... 76
3.6 Адміністративна панель ..... 78
3.7 Інтеграція з базою даних SQLite ..... 81
3.8 Висновки ..... 84
Висновки ..... 86
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ ..... 88
ДОДАТОК А ..... 90
ДОДАТОК Б ..... 96
ДОДАТОК В ..... 110
ДОДАТОК Г ..... 112

# ВСТУП

Сучасний розвиток інформаційних технологій суттєво змінює підходи до ведення бізнесу в різних сферах, зокрема у сфері надання спортивних послуг. Власники фітнес-клубів стикаються з необхідністю оптимізації процесів продажу абонементів, точного обліку відвідувань клієнтів та оперативної взаємодії з ними. Традиційні методи роботи, такі як паперові журнали, Excel-таблиці та ручне оформлення договорів, займають багато часу, часто призводять до помилок та не задовольняють сучасних клієнтів, які звикли до швидкого сервісу через мобільні додатки та месенджери.

Актуальність обраної теми зумовлена популярністю месенджера Telegram в Україні та його широкими можливостями для створення автоматизованих систем обслуговування. Розробка Telegram-бота дозволяє клієнтам у зручний час купувати абонементи, переглядати свій особистий кабінет, отримувати інформацію про тренерів, фіксувати відвідування та навіть оплачувати послуги, а адміністраторам – значно спростити рутинні операції.

Метою кваліфікаційної роботи є розробка та реалізація функціонального Telegram-бота LEO FITNESS для продажу абонементів і обліку відвідувань у спортивному клубі.

Для досягнення поставленої мети необхідно було вирішити такі основні задачі:

– проаналізувати предметну область діяльності спортивних клубів та особливості їх бізнес-процесів;

– вивчити існуючі рішення для автоматизації фітнес-клубів;

– спроектувати архітектуру системи, базу даних та інтерфейс користувача;

– реалізувати основний функціонал бота з використанням сучасних технологій;

– провести тестування розробленого програмного продукту;

– оцінити ефективність та практичну цінність отриманого рішення.

Об’єктом дослідження є процеси продажу послуг та обліку відвідувань у спортивному клубі. Предметом дослідження виступає програмне забезпечення Telegram-бота, призначене для автоматизації цих процесів.

У процесі виконання роботи були застосовані методи системного аналізу, структурного та об’єктно-орієнтованого проєктування, асинхронного програмування, тестування програмного забезпечення.

У процесі виконання роботи були застосовані методи системного аналізу, структурного та об’єктно-орієнтованого проєктування, асинхронного програмування, тестування програмного забезпечення.

Наукова новизна роботи полягає в комплексній реалізації Telegram-бота, який поєднує продаж абонементів (з підтримкою онлайн-оплати), особистий кабінет клієнта з відстеженням терміну дії абонементу та систему оперативного обліку відвідувань (check-in).

Практичне значення розробки підтверджується тим, що створений бот повністю робочий і може бути впроваджений у реальну діяльність спортивного клубу LEO FITNESS. Це дозволить зменшити навантаження на адміністраторів, підвищити якість обслуговування клієнтів та автоматизувати більшість рутинних операцій.

Дипломна робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел та п’яти додатків. Перший розділ присвячено аналізу предметної області. У другому розділі розглянуто проєктування системи. Третій розділ містить опис реалізації бота. У четвертому розділі наведено результати тестування та оцінку ефективності.

# 1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

## 1.1 Змістовий аналіз предметної області

Сучасний ринок спортивних послуг в Україні переживає період активної цифровізації та структурних змін. За даними Української федерації фітнесу, Всеукраїнського союзу фітнесу та незалежних маркетингових досліджень, проведених у 2024–2025 роках, попит на послуги спортивних клубів демонструє стабільне зростання на рівні 15–22% щорічно. Це зростання зумовлене кількома взаємопов'язаними факторами: підвищенням загального рівня здоров'я населення, активною популяризацією фітнесу в соціальних мережах, збільшенням кількості людей, які усвідомлюють важливість регулярних фізичних навантажень для профілактики захворювань, а також поступовим відновленням економіки після складних періодів.

Спортивний клуб у сучасних умовах є складним багатокомпонентним сервісним підприємством. Він поєднує в собі елементи надання фізкультурно-оздоровчих послуг, торгівлі товарами, управління персоналом, маркетингу та фінансового обліку. Основна мета такого закладу – не тільки надання можливості тренуватися, але й створення комфортного середовища для постійних відвідувачів, забезпечення високого рівня сервісу та отримання стабільного прибутку.

До ключових бізнес-процесів спортивного клубу належать:

– Процес продажу абонементів. Це один з найважливіших процесів, оскільки саме від продажу абонементів залежить основна частина доходу клубу. Абонементи можуть бути різних типів: пробні (1–3 тренування), разові відвідування, місячні, тримісячні, піврічні, річні, корпоративні, сімейні, студентські та спеціальні акційні пропозиції. Кожен тип абонементу має свою цінову політику, термін дії, кількість включених відвідувань та додаткові умови (наприклад, можливість заморозки або перенесення занять).

– Процес обліку та контролю відвідувань клієнтів**.** Цей процес включає ідентифікацію клієнта при вході в клуб, фіксацію факту відвідування, списання занять з абонементу та запобігання зловживанням. У ручному режимі це здійснюється через журнал або список на ресепшені, що часто призводить до помилок та шахрайства.

– Управління тренерським складом і розкладом занять**.** Клуб повинен вести актуальний графік групових занять, індивідуальних тренувань, контролювати навантаження на тренерів, збирати відгуки клієнтів про якість проведення занять.

– Продаж додаткових товарів і послуг**.** Сучасні спортивні клуби активно розвивають додаткові напрямки доходу: продаж спортивного харчування (протеїнові суміші, гейнери, креатин, амінокислоти, BCAA, вітаміни), мерчандайзу з логотипом клубу, персональних тренувань, масажу, солярію, послуг дієтолога та нутриціолога.

– Маркетингова комунікація з клієнтами**.** Сюди входить розсилка інформаційних повідомлень, акційних пропозицій, нагадувань про закінчення терміну дії абонементу, персоналізовані привітання, збір відгуків та проведення програм лояльності.

– Фінансовий облік і аналітика**.** Контроль платежів, боргів, повернень коштів, розрахунок зарплати персоналу, оренди приміщення, комунальних послуг та загальної прибутковості закладу.

У переважній більшості невеликих і середніх спортивних клубів (з кількістю активних клієнтів від 150 до 700 осіб) зазначені бізнес-процеси досі організовані переважно вручну або з використанням примітивних інструментів, таких як паперові журнали, таблиці Microsoft Excel, Google Sheets або прості безкоштовні CRM-системи. Такий підхід, на перший погляд зручний через низьку вартість, насправді має цілий комплекс серйозних недоліків, які суттєво обмежують розвиток клубу.

Перший і найсуттєвіший недолік – висока ймовірність людських помилок. Адміністратор може неправильно записати дату закінчення

абонементу, пропустити факт відвідування клієнта або, навпаки, списати зайве заняття. Такі помилки призводять до конфліктів з клієнтами, фінансових втрат та негативних відгуків у соціальних мережах.

Другий недолік – надзвичайно високі витрати робочого часу персоналу. Оформлення одного нового абонементу в середньому займає від 8 до 18 хвилин. При великій кількості нових клієнтів (особливо ввечері) на ресепшені виникають черги. Щоденна рутинна робота з перевіркою списків, звітами по продажах, закінченню абонементів та актуалізацією даних віднімає від 3 до 5 годин робочого часу адміністратора щодня.

Третій недолік – практично повна відсутність оперативної управлінської аналітики. Керівник клубу часто не має точної інформації в реальному часі: скільки клієнтів відвідало зал сьогодні, який день тижня є найбільш завантаженим, який відсоток абонементів закінчується протягом найближчих 14 днів, яка ефективність роботи кожного тренера, які товари продаються найкраще. Відсутність такої інформації ускладнює прийняття управлінських рішень і планування розвитку клубу.

Четвертий недолік – низький рівень клієнтського сервісу. Сучасний споживач фітнес-послуг (переважно люди віком 18–45 років) звик до швидкості та зручності. Він очікує, що зможе купити абонемент за кілька кліків у зручному месенджері, переглянути свій особистий кабінет, дізнатися інформацію про тренерів і швидко зафіксувати прихід у зал без необхідності стояти в черзі або телефонувати адміністратору.

П’ятий недолік – складнощі з контролем дисципліни клієнтів. Без надійної цифрової системи важко відстежити випадки передачі абонементів іншим особам, що призводить до прямих фінансових втрат для клубу.

Всі ці проблеми стають особливо відчутними в умовах жорсткої конкуренції між спортивними клубами. Клуби, які не впроваджують сучасні цифрові рішення, поступово втрачають клієнтів на користь конкурентів, які пропонують зручніший сервіс.

Таким чином, предметна область даної кваліфікаційної роботи охоплює

ключові бізнес-процеси спортивного клубу, які найбільше потребують автоматизації: продаж абонементів, оперативний облік відвідувань, управління клієнтською базою та підвищення якості взаємодії з клієнтами. Розробка спеціалізованого Telegram-бота є одним з найбільш перспективних і економічно виправданих шляхів вирішення цих завдань для невеликих і середніх спортивних клубів.

Розглянемо детальніше кожен з ключових бізнес-процесів спортивного клубу та проблеми, які виникають при їх ручній організації.

Процес продажу абонементів є основним джерелом доходу клубу. У середньому невеликому клубі щомісяця продається від 40 до 120 нових абонементів. Кожен продаж вимагає від адміністратора виконання низки операцій: консультація клієнта, підбір оптимального типу абонементу, внесення даних у таблицю, розрахунок вартості з урахуванням акцій, оформлення договору або чека, видача картки доступу. При ручному підході весь цей процес займає значний час і часто супроводжується помилками в розрахунках або внесенні даних.

Особливо складно працювати з акційними пропозиціями та різними типами абонементів. Наприклад, при продажу «Пробного абонементу» на 3 тренування необхідно чітко фіксувати дату початку та залишок занять. Без автоматизованої системи це часто призводить до ситуацій, коли клієнт приходить на 4-те тренування, а в таблиці вже немає інформації про його абонемент.

Процес обліку відвідувань – один з найбільш трудомістких. У дні пік (вівторок, четвер, вечір п’ятниці) через клуб може пройти 80–150 осіб. Адміністратор повинен кожного перевірити по списку, відмітити відвідування, списати заняття. При великій кількості людей це призводить до черг, помилок та незадоволення клієнтів. Крім того, існує проблема «друзів з абонементом» – коли один клієнт передає свою картку іншій особі. Без цифрової ідентифікації відстежити такі випадки майже неможливо.

Управління тренерським складом також вимагає значних зусиль.

Потрібно вести актуальний розклад, контролювати відвідуваність групових занять, збирати відгуки, розраховувати оплату тренерам залежно від кількості проведених тренувань. У ручному режимі це часто робиться в кількох різних таблицях, що призводить до плутанини та конфліктів.

Продаж додаткових товарів у сучасних клубах стає все важливішим джерелом доходу. Багато клієнтів після тренування готові купити протеїн, гейнер або шейкер. Однак при ручному обліку важко вести складський облік, фіксувати продажі та аналізувати, які товари користуються попитом.

Маркетингова комунікація при ручному підході майже не здійснюється або здійснюється хаотично. Адміністратор фізично не встигає розсилати нагадування про закінчення абонементів, персональні акції або вітання. В результаті клуб втрачає значну частину потенційного доходу від продовження абонементів.

Фінансовий облік при використанні Excel або паперових документів є вкрай ненадійним. Важко швидко отримати реальну картину прибутків і витрат, проаналізувати рентабельність різних видів абонементів або оцінити ефективність проведених акцій.

Усі перелічені проблеми призводять до загального зниження ефективності роботи клубу. За приблизними підрахунками, через ручний облік невеликий спортивний клуб щомісяця втрачає від 8 до 25% потенційного прибутку через помилки, втрачений час адміністраторів, відтік клієнтів та неможливість проведення ефективного маркетингу.

Порівнюючи невеликі клуби з великими мережами типу «SportLife», «Fitness House» або «Olympic», можна побачити разючу різницю. Великі мережі вже давно використовують потужні CRM-системи, мобільні додатки та автоматизовані системи доступу. Вони мають можливість надавати клієнтам зручний сервіс, оперативну аналітику та персоналізовані пропозиції. Маленькі клуби, які не переходять на цифрові рішення, поступово програють у конкурентній боротьбі.

Саме тому розробка доступного, простого у використанні та недорогого

рішення для автоматизації – такого як Telegram-бот – є вкрай актуальним завданням для більшості спортивних клубів України. Telegram як платформа має ряд суттєвих переваг: висока популярність серед цільової аудиторії (18–45 років), мінімальні вимоги до обладнання клієнта, можливість роботи 24/7, швидка розробка та низька вартість впровадження.

Таким чином, детальний аналіз предметної області показує, що існуючі проблеми ручного управління спортивним клубом є системними і потребують сучасного програмного рішення. Предметна область дипломної роботи повністю відповідає реальним потребам ринку і має високий практичний потенціал.

Як видно з наведеного рейтингу( рисунок 1.1), лідером ринку вже кілька років поспіль залишається мережа Sport Life, яка набрала максимальну кількість балів (16,25) та 100% відносний показник. За нею йдуть GYMMAXX, Atletiko, Tsarsky City Resort та інші великі гравці ринку.

Цей рейтинг чітко демонструє, що найбільш успішні фітнес-клуби України – це великі мережі, які активно впроваджують сучасні технології автоматизації, мають власні мобільні додатки або потужні CRM-системи.

| Компанія | Середня оцінка від журі | Місце в рейтингу | Бали | Бали % |
|---|---|---|---|---|
| Sport Life | 2,0 | 1 | 16,25 | 100 |
| GYMMAXX | 3,5 | 2 | 8,61 | 52,98 |
| АтлетіКо | 2,5 | 3 | 7,44 | 45,79 |
| Tsarsky City Resort | 3,5 | 4 | 5,46 | 33,59 |
| 5 Елемент | 3,5 | 5 | 5,45 | 33,54 |
| APOLLO NEXT | 3,0 | 6 | 5,06 | 31,14 |
| ЕБШ | 4,0 | 7 | 4,55 | 27,97 |
| Dog & Grand CrossFit | 3,0 | 8 | 3,35 | 20,64 |
| Podolskiy | 2,0 | 9 | 3,12 | 19,19 |
| L Sector TRX Training Club | 2,0 | 10 | 2,92 | 17,94 |
| SkyFitness | 3,0 | 11 | 2,7 | 16,63 |
| FIZMAT | 3,0 | 12 | 2,58 | 15,85 |
| Sport Studio | 1,0 | 13 | 2,27 | 14 |
| Palestra | 1,5 | 14 | 1,7 | 10,48 |
| FIZIKA | 1,5 | 15 | 1,24 | 7,66 |

Рисунок 1.1 – Рейтинг топ-15 фітнес-клубів України 2025 року

Для невеликих і середніх незалежних спортивних клубів (таких як LEO FITNESS) потрапити в конкурентну боротьбу з такими гігантами без впровадження сучасних цифрових рішень стає дедалі складніше. Саме тому розробка доступного та функціонального інструменту автоматизації, такого як Telegram-бот, є важливим конкурентним інструментом для малого бізнесу в сфері фітнесу.

В результаті проведеного аналізу можна підсумувати, що сучасні спортивні клуби стикаються з низкою системних проблем: ручним обліком абонементів, великими витратами часу адміністраторів, високою ймовірністю помилок та відсутністю оперативної аналітики. Сучасний клієнт вимагає швидкого та зручного сервісу. Саме тому розробка спеціалізованого Telegram-бота є актуальним і перспективним рішенням для малого та середнього бізнесу в сфері фітнес-послуг.

## 1.2 Аналіз наявного програмно-технічного забезпечення предметної області

На сучасному ринку програмного забезпечення для автоматизації спортивних клубів представлено досить широкий спектр рішень. Їх можна умовно поділити на кілька основних категорій залежно від типу, функціональності, вартості та цільової аудиторії. Проведення всебічного аналізу цих рішень є важливим етапом для розуміння сильних і слабких сторін існуючих систем, а також для обґрунтування необхідності розробки власного спеціалізованого рішення – Telegram-бота LEO FITNESS.

Перша група – потужні комплексні CRM-системи. До цієї категорії належать такі відомі рішення як: Mindbody (США), Zenoti, ClubReady, Instasport, LuckyFit, Perfectum CRM, «1С: Фітнес-клуб», «БІТ.Фітнес», Mobifitness, «Спортклуб» та інші.

Ці системи пропонують широкий функціонал: повноцінний облік клієнтів, продаж абонементів, автоматизований check-in через турнікети, інтеграцію з сайтом, мобільні додатки для клієнтів, детальну аналітику, модуль роботи з тренерами, складський облік товарів, фінансову звітність, SMS та email-розсилки, інтеграцію з платіжними системами тощо.


Рисунок 1.2 – Структурно-функціональна схема комплексних CRM-систем для спортивних клубів

Переваги таких систем:

– Висока функціональність і гнучкість налаштування;

– Потужна аналітика та звітність;

– Можливість масштабування для великих мереж;

– Інтеграція з різним обладнанням (турнікети, ключі доступу, відеоспостереження).

Недоліки (особливо критичні для невеликих клубів):

– Висока вартість ліцензії та впровадження (від 30 000 до 300 000+ гривень на рік);

– Складність освоєння для звичайного адміністратора;

– Тривалий термін впровадження (від 1 до 6 місяців);

– Необхідність постійної технічної підтримки та оновлень;

– Надлишковий функціонал, більша частина якого ніколи не використовується в невеликому клубі.

Друга група – самостійно розроблені мобільні додатки. Багато великих мереж фітнес-клубів (Sport Life, Olympia, X-Fit тощо) мають власні мобільні додатки. Такі рішення зазвичай мають сучасний дизайн, push-повідомлення, особистий кабінет клієнта, можливість купівлі абонементу онлайн, перегляд розкладу та запис на тренування.

Переваги: висока зручність для клієнта, сучасний інтерфейс, брендованість. Недоліки: надзвичайно висока вартість розробки (від 400 000 до 1 500 000 гривень) та подальшої підтримки, необхідність оновлень під нові версії iOS/Android, а також те, що клієнт повинен завантажувати окремий додаток.

Третя група – месенджер-боти (Telegram, Viber, WhatsApp). Це наймолодша і найбільш динамічна категорія рішень. Наразі на ринку існує невелика кількість ботів для фітнес-клубів, серед яких можна виділити @gymcoach_bot, FitUA, деякі локальні боти окремих клубів. Однак більшість з них мають обмежений функціонал: запис на тренування, надсилання програм тренувань, нагадування або просте бронювання.

Комплексних рішень, які поєднують повноцінний продаж абонементів з оплатою, особистий кабінет, check-in відвідувань, продаж товарів та адміністративну панель в одному Telegram-боті, на українському ринку практично немає. Саме ця ніша і стала основою для розробки бота LEO FITNESS.

Для більш детального розуміння ринку розглянемо основні представників кожної групи рішень.

1. Комплексні CRM-системи

Instasport – одна з найпопулярніших систем в Україні. Вона дозволяє вести повний облік клієнтів, продавати абонементи, вести розклад занять, працювати з турнікетами та проводити детальну аналітику. Однак вартість

підписки досить висока, а інтерфейс для адміністратора вважається складним для людей без спеціальної підготовки.

Mindbody – міжнародний лідер ринку. Має потужний функціонал, мобільний додаток для клієнтів, інтеграцію з платежами та детальну статистику. Головний недолік – дуже висока ціна, яка робить систему недоступною для більшості невеликих українських клубів.

«1С: Фітнес-клуб» – популярне рішення серед клубів, які вже працюють з іншими продуктами 1С. Перевага – інтеграція з бухгалтерією. Недоліки – застарілий інтерфейс і складність налаштування.

LuckyFit, Perfectum CRM, Mobifitness – більш сучасні українські розробки. Вони намагаються поєднати зручність і доступну ціну, але часто поступаються функціональністю міжнародним аналогам.

2. Мобільні додатки

Більшість великих мереж розробляють власні мобільні додатки. Вони виглядають сучасно, мають push-повідомлення, особистий кабінет і можливість онлайн-оплати. Однак для одного невеликого клубу розробка такого додатка є економічно недоцільною – вартість може сягати від 500 тисяч до 2 мільйонів гривень з урахуванням підтримки.

3. Telegram-боти та месенджер-рішення

На сьогодні це найперспективніший напрямок для малого бізнесу. Переваги Telegram-ботів очевидні:

– Не потрібно завантажувати окремий додаток – клієнт вже має Telegram;

– Миттєва доставка повідомлень;

– Низька вартість розробки та підтримки;

– Можливість швидкого оновлення функціоналу;

– Висока доступність для цільової аудиторії.

Однак більшість існуючих ботів мають обмежений функціонал (тільки запис на тренування або надсилання програм). Комплексних рішень з продажем абонементів, check-in і продажем товарів майже немає.

Як видно з наведеної ілюстрації, сучасні спортивні клуби намагаються створювати комфортне та технологічне середовище. Однак без відповідного програмного забезпечення весь цей сучасний вигляд не може повністю реалізувати свій потенціал.

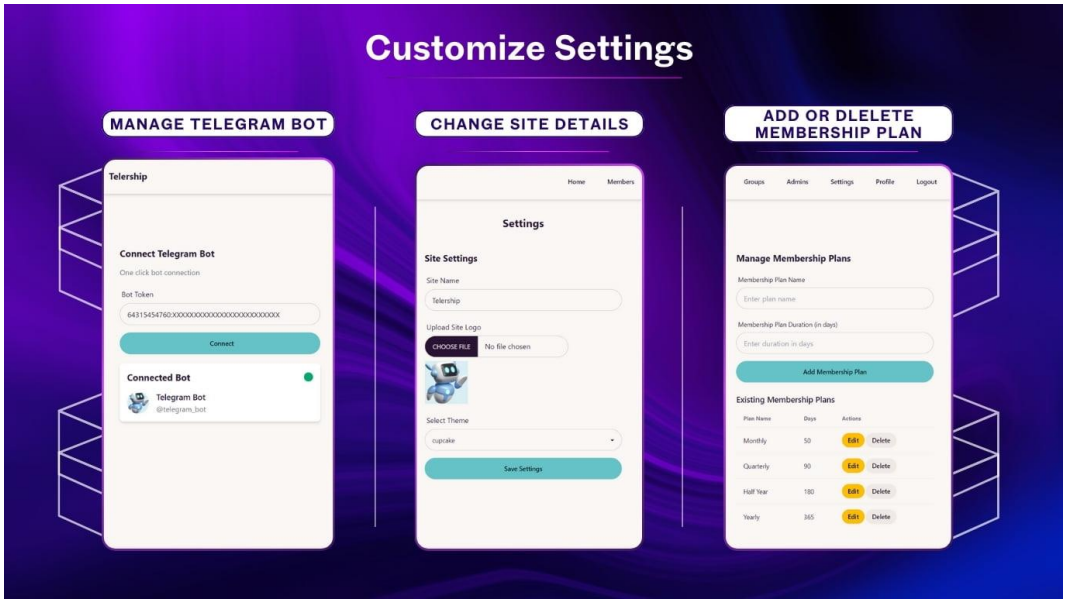



Рисунок 1.3 – Приклади інтерфейсів сучасних систем управління фітнес-клубами

На рисунку представлені сучасні інтерфейси адміністративних панелей і мобільних додатків, які використовуються в провідних фітнес-клубах. Такі інтерфейси характеризуються мінімалістичним дизайном, зручною навігацією та швидким доступом до ключових функцій: управління абонементами, статистики відвідувань, роботи з клієнтами та аналітики продажів.

Подібні рішення демонструють високий рівень автоматизації, але, як правило, доступні лише великим мережам через високу вартість розробки та підтримки. Для невеликих клубів такі інтерфейси залишаються недоступними, що створює потребу у більш простих і економічних рішеннях, таких як Telegram-боти.

На рисунку 1.4 схематично зображено основні бізнес-процеси спортивного клубу та взаємодію між ними. Центральне місце займає клієнт (Member), який взаємодіє з процесами реєстрації, управління абонементом,

запису на тренування, оплати та відстеження прогресу.

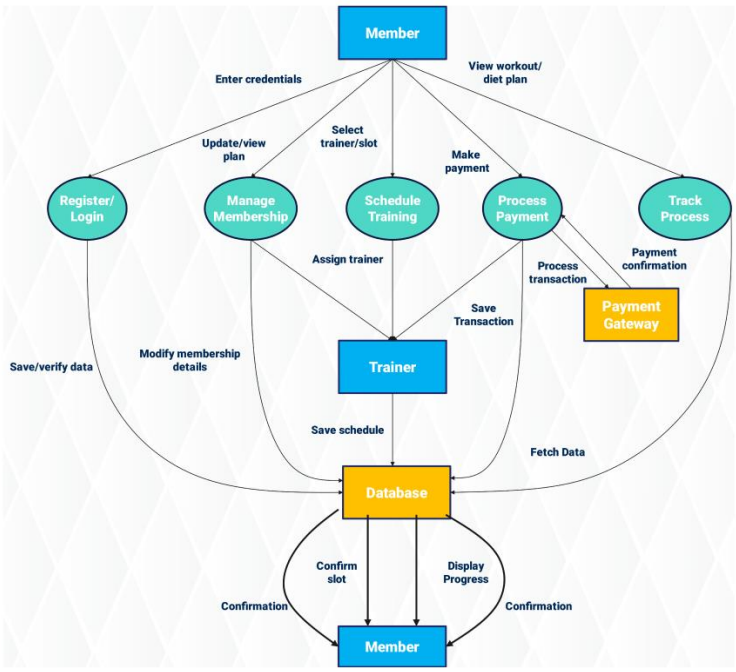



Рисунок 1.4 – Приклад моделі бізнес-процесів фітнес-клубу

Діаграма наочно ілюструє складність взаємозв'язків між основними учасниками (клієнт, тренер, адміністратор) та необхідність єдиної інформаційної системи для ефективного управління цими процесами. Відсутність такої системи призводить до роз'єднаності даних і зниження загальної ефективності роботи клубу.

Представлена схема (рисунок 1.5) відображає загальну архітектуру сучасного Telegram-бота для автоматизації спортивного клубу. На ній видно основні компоненти: взаємодію користувача з Telegram, серверну частину (бот), базу даних, а також додаткові сервіси (платежі, розсилки, адмін-панель).

Така архітектура є масштабована, надійна та відносно проста в реалізації. Вона дозволяє поєднати високу зручність для клієнта з можливістю централізованого управління всіма процесами клубу в єдиному інструменті.

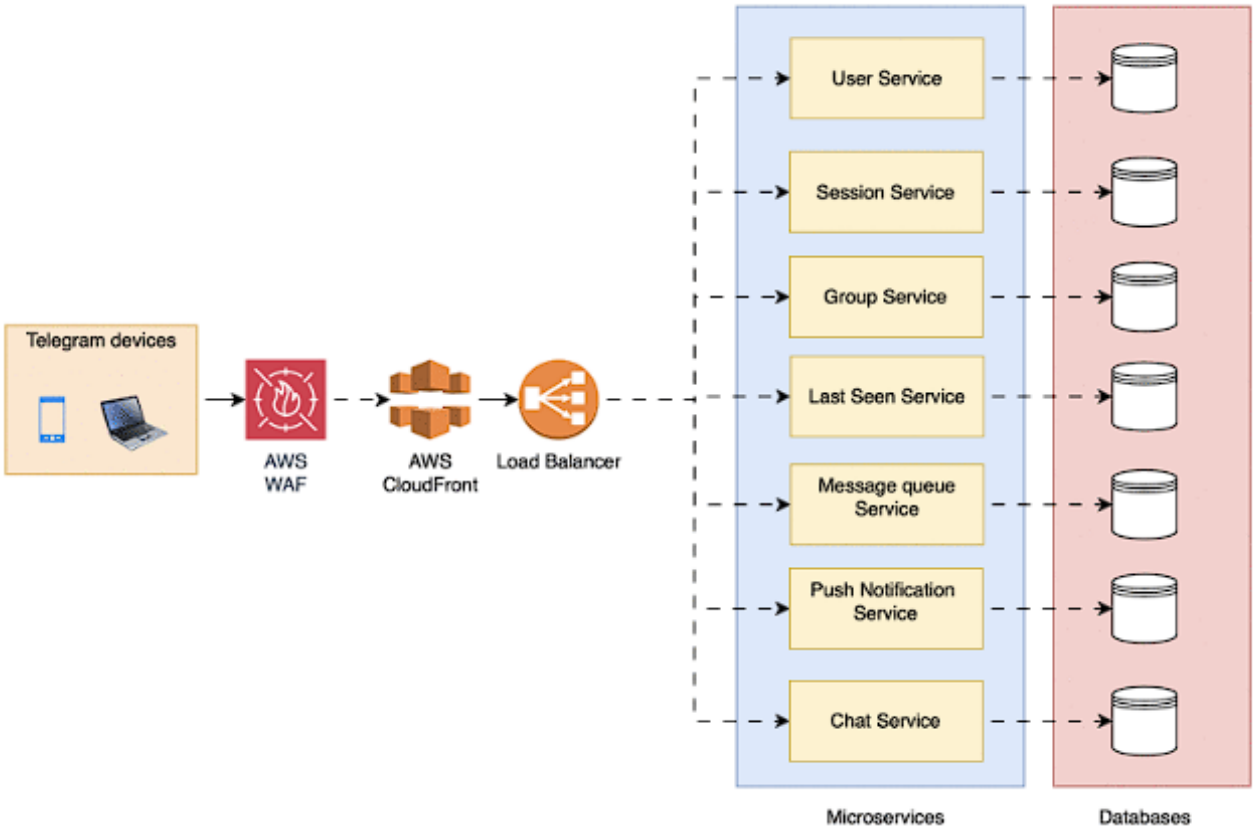



Рисунок 1.5 – Загальна архітектура Telegram-бота

Детальніше розглянемо сильні та слабкі сторони кожної категорії рішень, щоб чітко зрозуміти, чому для невеликого спортивного клубу LEO FITNESS було обрано саме розробку власного Telegram-бота.

Комплексні CRM-системи, такі як Instasport, Mindbody чи «1С: Фітнес-клуб», безперечно, є потужними інструментами. Вони дозволяють автоматизувати практично всі бізнес-процеси клубу: від продажу абонементів до складського обліку протеїну. Однак для клубу з кількістю клієнтів до 400–500 осіб ці системи стають надмірно складними та дорогими. Середня вартість річної підписки на такі системи коливається від 45 000 до 180 000 гривень, не враховуючи витрат на обладнання (турнікети, планшети для адміністраторів) та навчання персоналу. Крім того, багато функцій залишаються невикористаними, що робить інвестицію неефективною.

Власні мобільні додатки також мають суттєві обмеження. Розробка якісного мобільного додатка для iOS та Android з підтримкою push-

повідомлень, онлайн-оплати та особистого кабінету коштує від 600 тисяч до 1,8 мільйона гривень. До цього додається щорічна вартість підтримки, оновлення під нові версії операційних систем та просування додатка в Google Play і App Store. Для одного невеликого клубу такі витрати майже ніколи не окупаються.

Ручний облік за допомогою паперових журналів та Excel-таблиць залишається найпоширенішим методом серед невеликих клубів. Він не вимагає фінансових вкладень, але має критичні недоліки: високу ймовірність помилок, великі витрати часу адміністраторів, повну відсутність аналітики в реальному часі, неможливість швидкого оформлення абонементу та низький рівень сервісу для клієнтів. У сучасних умовах такий підхід робить клуб неконкурентоспроможним.

Telegram-боти на сьогодні є найбільш перспективним рішенням для малого та середнього бізнесу в сфері фітнесу. Вони поєднують у собі низьку вартість розробки, високу швидкість впровадження, зручність для клієнтів та достатній функціонал для вирішення основних завдань клубу. Клієнту не потрібно нічого завантажувати – достатньо відкрити Telegram і почати спілкування з ботом. Повідомлення приходять миттєво, інтерфейс інтуїтивно зрозумілий, а можливість роботи 24/7 дозволяє клієнтам купувати абонементи навіть о 3 годині ночі.

Саме тому в рамках дипломної роботи було прийнято рішення розробити власний спеціалізований Telegram-бот LEO FITNESS, який враховує всі особливості роботи невеликого спортивного клубу. Бот повинен вирішувати такі ключові завдання:

– швидка та зручна реєстрація клієнтів;

– продаж абонементів різних типів;

– формування кошика та оформлення замовлення;

– підтримка онлайн-оплати та оплати в клубі;

– особистий кабінет з інформацією про залишок занять і термін дії абонементу;

– система check-in (фіксація відвідувань);

– надання інформації про тренерів;

– продаж додаткових товарів;

– проста адміністративна панель для керівника та адміністратора.

Таким чином, проведений аналіз наявного програмно-технічного забезпечення показав, що на ринку існує значна ніша для недорогих, функціональних і зручних рішень саме у форматі Telegram-ботів. Існуючі системи або надто дорогі, або надто обмежені, або не відповідають реальним потребам невеликих спортивних клубів. Розробка бота LEO FITNESS дозволить закрити цю нішу та надати практичне рішення, яке можна швидко впровадити і ефективно використовувати.

Однією з головних переваг Telegram-ботів порівняно з іншими рішеннями є їхня економічна ефективність. Розробка та запуск повноцінного бота обходиться в десятки разів дешевше, ніж створення мобільного додатка чи впровадження потужної CRM-системи. Крім того, підтримка та оновлення бота вимагають мінімальних ресурсів – достатньо одного розробника, який може швидко вносити зміни в код.

Іншою важливою перевагою є висока доступність для клієнтів. За статистикою, понад 70% активних користувачів смартфонів в Україні регулярно користуються месенджером Telegram. Це означає, що клієнту не потрібно завантажувати додатковий додаток, реєструватися в новій системі чи вивчати складний інтерфейс. Достатньо написати боту /start – і він відразу потрапляє в зручне меню з усіма необхідними функціями.

Telegram-боти також вигідно відрізняються швидкістю взаємодії. Повідомлення надходять миттєво, а відповіді від бота приходять за долі секунди. Це особливо важливо для сучасних клієнтів, які звикли до швидкого сервісу. Можливість купити абонемент за 1–2 хвилини в будь-який час доби значно підвищує конверсію продажів порівняно з традиційним способом.

Ще однією сильною стороною є гнучкість розробки. На відміну від готових CRM-систем, власний бот можна повністю адаптувати під особливості

конкретного клубу: додати певні типи абонементів, акції, інтеграцію з конкретною платіжною системою, унікальні функції (наприклад, автоматичне нагадування про закінчення абонементу за 5 днів) тощо. При цьому всі зміни можна вносити оперативно, без довгого очікування оновлень від розробника CRM.

Не менш важливою є безпека даних. При правильній реалізації бот може зберігати персональні дані клієнтів (ім'я, email, номер телефону) у зашифрованому вигляді, а доступ до адміністративної панелі обмежувати лише для керівництва клубу. Це дозволяє відповідати сучасним вимогам захисту персональних даних.

Обґрунтування вибору технологічного стеку для бота LEO FITNESS

Після аналізу ринку було прийнято рішення використовувати мову програмування Python 3.12 у поєднанні з асинхронною бібліотекою aiogram 3.x та вбудованою базою даних SQLite3. Такий вибір обумовлений кількома факторами:

– Python є однією з найпопулярніших мов програмування у світі завдяки простоті, читабельності коду та великій кількості готових бібліотек.

– aiogram 3.x – сучасна, швидка та зручна бібліотека для створення Telegram-ботів з підтримкою Finite State Machine (FSM), middleware, роутерів та інших сучасних можливостей.

– SQLite3 не вимагає окремого сервера бази даних, що ідеально підходить для невеликого клубу. При цьому вона повністю достатня для роботи з кількома сотнями клієнтів.

Такий стек технологій дозволяє створити стабільний, швидкий і відносно простий у підтримці програмний продукт.

Проведений аналіз наявного програмно-технічного забезпечення показав, що на ринку фітнес-послуг існує значний розрив між потребами невеликих і середніх спортивних клубів та запропонованими рішеннями. Комплексні CRM-системи є надто дорогими та складними, власні мобільні додатки – економічно недоцільними, а ручний облік вже не відповідає

сучасним вимогам клієнтів і ринку.

Telegram-боти на сьогодні є найбільш оптимальним рішенням за критеріями вартості, зручності, швидкості впровадження та функціональності. Вони дозволяють невеликим клубам отримувати сучасний рівень сервісу без значних фінансових витрат.

Саме тому в межах даної кваліфікаційної роботи було обрано розробку спеціалізованого Telegram-бота LEO FITNESS, який поєднує в собі необхідний функціонал для продажу абонементів, обліку відвідувань, роботи з клієнтами та адміністрування клубу. Такий підхід дозволить закрити існуючу нішу на ринку та надати практичне, доступне і ефективне рішення для спортивних клубів України.

## 1.3 Визначення вимог до розроблюваного застосунку

На основі детального аналізу предметної області, проведеного в попередніх підрозділах, а також вивчення існуючих програмних рішень, були сформульовані чіткі, конкретні та обґрунтовані вимоги до Telegram-бота LEO FITNESS. Вимоги розроблялися з урахуванням реальних потреб невеликого спортивного клубу, побажань цільової аудиторії (клієнтів і адміністраторів), специфіки роботи месенджера Telegram та технічних можливостей обраного стеку технологій.

Функціональні вимоги:

– Система реєстрації та авторизації користувачів повинна забезпечувати швидку та зручну реєстрацію нового клієнта за допомогою імені та email-адреси з обов’язковою перевіркою унікальності електронної пошти. Також має бути реалізована можливість повторної авторизації за Telegram ID без повторного введення даних.

– Система управління абонементами повинна дозволяти клієнту

переглядати актуальний список доступних абонементів з детальним описом (ціна, термін дії, кількість занять, умови використання), обирати необхідний абонемент, додавати його в кошик та оформлювати замовлення.

– Система оплати повинна підтримувати два основних способи: оплату в клубі (з подальшим підтвердженням адміністратором) та онлайн-оплату банківською карткою. При цьому бот повинен генерувати унікальний номер замовлення, формувати чек та оновлювати статус абонементу після успішної оплати.

– Особистий кабінет клієнта повинен відображати актуальну інформацію про діючий абонемент (залишок днів, залишок занять), історію відвідувань, персональні дані та надавати можливість їх редагування.

– Система обліку відвідувань (check-in) повинна дозволяти клієнту самостійно фіксувати факт приходу в зал через бота з автоматичним списанням одного заняття. При цьому має бути передбачений захист від повторного check-in протягом одного дня.

– Інформаційний розділ повинен містити детальну інформацію про тренерів клубу (фото, ім’я, спеціалізація, стаж, короткий опис досягнень), що допоможе клієнтам зробити свідомий вибір при записі на індивідуальні тренування.

– Функціонал продажу додаткових товарів повинен дозволяти переглядати асортимент спортивного харчування та аксесуарів, додавати товари в кошик разом з абонементом та оформлювати спільне замовлення.

– Адміністративна панель повинна надавати керівнику та адміністраторам можливість переглядати список усіх клієнтів, управляти абонементами (продовження, блокування, ручне нарахування занять), отримувати статистику продажів, відвідувань та проводити цільові розсилки повідомлень.

Нефункціональні вимоги:

– Бот повинен забезпечувати високу швидкодію відповіді (не більше 1 – 2 секунд навіть при навантаженні).

– Система має стабільно працювати при одночасному використанні кількома десятками активних користувачів.

– Інтерфейс повинен бути максимально інтуїтивно зрозумілим, зручним і адаптованим під мобільні пристрої різного розміру.

– Бот повинен бути доступним 24 години на добу 7 днів на тиждень без перерв у роботі.

– Мінімальні системні вимоги для кінцевого користувача – будь-який смартфон з встановленим додатком Telegram.

– Код бота повинен бути структурованим, читабельним і легко розширюваним для подальшого додавання нового функціоналу.

Вимоги до безпеки:

– Надійний захист від несанкціонованого доступу до адміністративної панелі.

– Зберігання всіх чутливих даних клієнтів (email, персональна інформація) у зашифрованому вигляді.

– Обмеження кількості спроб введення даних для запобігання brute-force атакам.

Вимоги до надійності:

– Автоматичне створення регулярних резервних копій бази даних. – Коректна обробка всіх можливих помилок з зрозумілим повідомленням для користувача.

– Можливість швидкого відновлення роботи бота після технічних збоїв або перезапуску сервера.

## 1.4 Висновки. Постановка задачі

В результаті проведеного дослідження предметної області можна зробити ряд важливих висновків. Сучасні спортивні клуби, особливо невеликі

та середні, стикаються з численними проблемами: ручним обліком абонементів і відвідувань, значними витратами робочого часу адміністраторів, високою ймовірністю помилок, відсутністю оперативної аналітики та низьким рівнем сервісу для клієнтів. Традиційні методи роботи (паперові журнали, Excel-таблиці) вже не відповідають очікуванням сучасного споживача, який вимагає швидкого, зручного та доступного сервісу 24/7.

Аналіз існуючого програмно-технічного забезпечення показав, що потужні CRM-системи є надто дорогими та складними для невеликих клубів, власні мобільні додатки – економічно недоцільними, а більшість готових Telegram-ботів мають обмежений функціонал. У зв’язку з цим існує реальна потреба у створенні спеціалізованого, доступного та функціонального рішення саме для малого бізнесу в сфері фітнес-послуг.

Метою кваліфікаційної роботи є розробка та реалізація функціонального Telegram-бота LEO FITNESS, який автоматизує процеси продажу абонементів, обліку відвідувань та управління клієнтською базою спортивного клубу.

Для досягнення поставленої мети необхідно вирішити такі основні задачі:

– провести аналіз предметної області діяльності спортивних клубів та вивчити існуючі програмні рішення;

– визначити функціональні та нефункціональні вимоги до розроблюваного застосунку;

– спроектувати архітектуру системи, структуру бази даних та користувацький інтерфейс;

– реалізувати основний функціонал Telegram-бота з використанням сучасних технологій;

– провести комплексне тестування розробленого програмного продукту;

– оцінити ефективність та практичну цінність отриманого рішення.

Об’єктом дослідження є бізнес-процеси продажу послуг та обліку відвідувань у спортивному клубі. Предметом дослідження виступає програмне забезпечення у вигляді Telegram-бота.

Вирішення поставлених задач дозволить створити практичний інструмент, який можна безпосередньо впровадити в діяльність спортивного клубу LEO FITNESS, підвищити якість обслуговування клієнтів та збільшити загальну ефективність роботи клубу.

# 2 ПРОЄКТУВАННЯ СИСТЕМИ

## 2.1 Вимоги до системи

Проєктування будь-якої інформаційної системи починається з чіткого та детального визначення вимог. У рамках дипломної роботи вимоги до Telegram-бота LEO FITNESS були сформульовані на основі глибокого аналізу предметної області, вивчення потреб потенційних користувачів (клієнтів спортивного клубу та адміністраторів), а також врахування технічних можливостей і обмежень платформи Telegram та обраного технологічного стеку.

Вимоги до системи поділяються на функціональні, нефункціональні, вимоги до безпеки, надійності, масштабованості, продуктивності та користувацького досвіду. Кожна група вимог була детально опрацьована для того, щоб майбутній програмний продукт повністю відповідав реальним потребам спортивного клубу LEO FITNESS і міг бути ефективно впроваджений у повсякденну діяльність.

Функціональні вимоги становлять основу системи і визначають, що саме повинен робити бот. Вони були сформульовані з урахуванням основних бізнес-процесів спортивного клубу: продаж абонементів, облік відвідувань, робота з клієнтською базою, надання інформаційних послуг та адміністрування.

До ключових функціональних вимог належать:

– Реалізація повноцінної системи реєстрації та авторизації користувачів, яка включає швидку реєстрацію за ім’ям та email, перевірку унікальності

електронної адреси, автоматичну прив’язку до Telegram-акаунту та можливість повторної авторизації без повторного введення даних.

– Розробка модуля управління абонементами, який дозволяє клієнту переглядати актуальний перелік доступних абонементів з детальним описом (вартість, термін дії, кількість включених занять, умови використання, акційні пропозиції), обирати необхідний абонемент, додавати його в кошик, переглядати вміст кошика та оформлювати замовлення.

– Створення системи оплати, яка підтримує два основних способи розрахунку: оплату в клубі (з подальшим підтвердженням адміністратором) та онлайн-оплату банківською карткою. Система повинна генерувати унікальний номер замовлення, формувати електронний чек, оновлювати статус абонементу після успішної оплати та надсилати підтвердження клієнту.

– Реалізація особистого кабінету клієнта, де користувач може в будь-який момент переглянути актуальну інформацію про свій абонемент (залишок днів дії, залишок доступних занять), історію відвідувань, персональні дані, а також за необхідності внести зміни в свої контактні дані.

– Розробка системи обліку відвідувань (check-in), яка дозволяє клієнту самостійно фіксувати факт приходу в зал через бота. При цьому має відбуватися автоматичне списання одного заняття з активного абонементу, перевірка на наявність дійсного абонементу та захист від повторного check-in протягом одного календарного дня.

– Створення інформаційного розділу про тренерів клубу, де клієнт може переглянути фотографії, спеціалізацію, стаж роботи, короткий опис досягнень та напрямки тренувань кожного тренера.

– Реалізація модуля продажу додаткових товарів (спортивне харчування, аксесуари), який дозволяє переглядати асортимент, ціни, додавати товари в кошик разом з абонементом та оформлювати комплексне замовлення.

– Розробка адміністративної панелі, яка надає керівництву клубу та адміністраторам доступ до управління клієнтською базою, редагування абонементів, перегляду статистики продажів і відвідувань, проведення

розсилок та моніторингу роботи бота в реальному часі.

Крім основних функціональних блоків, система повинна включати додаткові можливості, які значно підвищують її практичну цінність:

– Автоматичне формування та надсилання чеків/квитанцій після успішної оплати у зручному для клієнта форматі.

– Можливість бронювання індивідуальних тренувань у конкретного тренера на певний час.

– Система нагадувань клієнтам про закінчення терміну дії абонементу (за 7, 3 та 1 день).

– Можливість заморозки абонементу на певний період (наприклад, під час відпустки).

– Історія всіх операцій клієнта (покупки, відвідування, платежі). – Функція зворотного зв’язку – можливість залишати відгуки про тренерів та якість обслуговування.

– Інтеграція з Google Calendar або внутрішнім розкладом для запису на групові заняття.

Нефункціональні вимоги також відіграють критично важливу роль, оскільки визначають якість роботи системи в реальних умовах експлуатації.

Система повинна забезпечувати високу швидкодію – середній час відповіді бота не повинен перевищувати 1,5 секунди навіть при піковому навантаженні. Бот має стабільно працювати при одночасній активності 50–100 користувачів. Інтерфейс повинен бути максимально простим і інтуїтивно зрозумілим для людей різного віку та рівня технічної грамотності. Всі повідомлення мають бути чіткими, лаконічними та українською мовою.

Важливою вимогою є кросплатформенність – бот повинен коректно працювати на смартфонах Android, iOS, а також у десктопній версії Telegram. Дизайн клавіатур та повідомлень повинен бути адаптивним під різні розміри екранів.


Рисунок 2.1 – Функціональні вимоги системи

Вимоги до користувацького досвіду (UX):

– Мінімальна кількість кліків для виконання основних операцій (купівля абонементу, check-in).

– Використання emoji та зрозумілих іконок для покращення візуального сприйняття.

– Послідовність та передбачуваність інтерфейсу.

Всі перелічені вимоги були детально опрацьовані, узгоджені та задокументовані. Вони стали основою для всіх наступних етапів проєктування: розробки архітектури, моделювання бази даних, створення діаграм Use Case, Activity та Sequence, а також безпосередньої програмної реалізації бота.

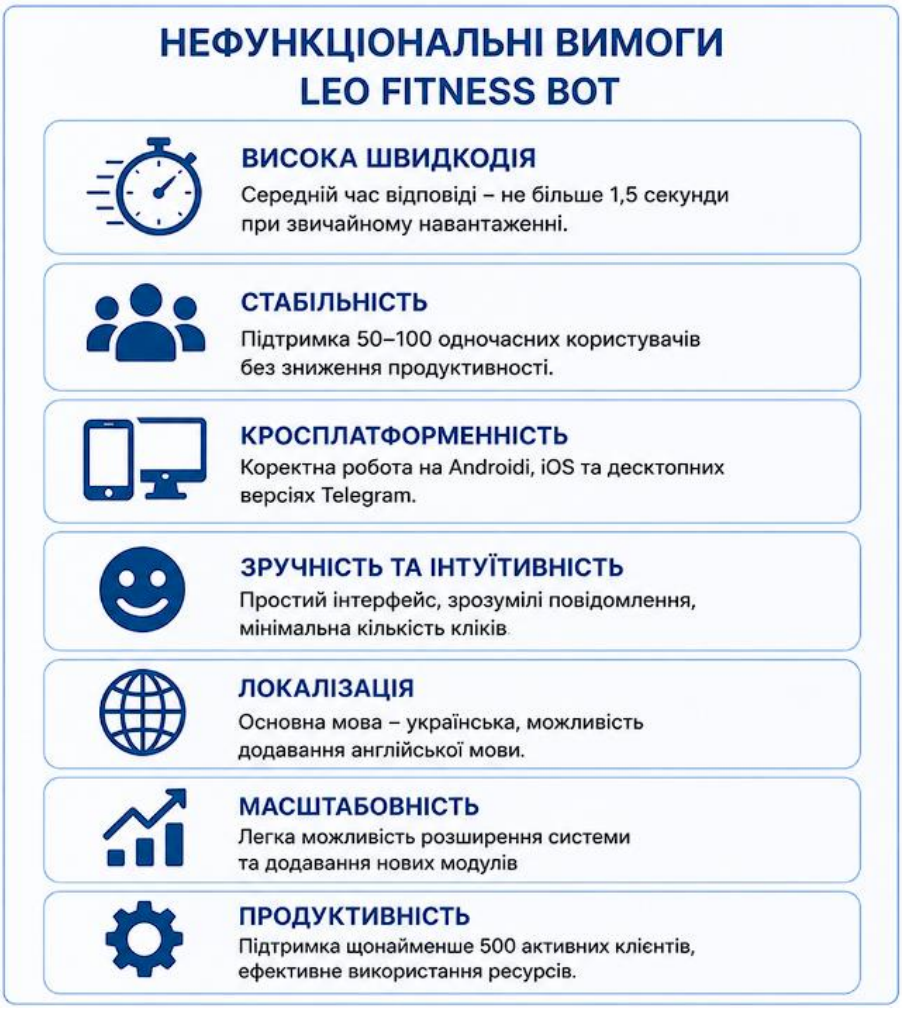



Рисунок 2.2 – Нефункціональні вимоги системи

Вимоги до безпеки є одним з найважливіших аспектів проєктування. Система повинна забезпечувати:

– Надійний захист адміністративної панелі (двоступенева авторизація, обмеження IP-адрес, логування всіх входів).

– Зберігання паролів та чутливих даних у зашифрованому вигляді.

– Захист від основних типів атак (SQL-ін’єкції, XSS, CSRF, brute-force).

– Обмеження кількості спроб введення даних протягом певного часу.

Вимоги до масштабованості:

Система повинна бути спроектована таким чином, щоб у майбутньому її можна було легко розширити на кілька спортивних клубів, додати нові модулі (наприклад, інтеграцію з сайтом клубу, онлайн-запис на масаж, інтеграцію з системою відеоспостереження) без значної переробки архітектури.


Рисунок 2.3 – Вимоги до безпеки та надійності

Такий комплексний підхід до визначення вимог дозволяє бути впевненим, що розроблюваний Telegram-бот LEO FITNESS повністю відповідатиме реальним потребам спортивного клубу та його клієнтів.

Особливу увагу при проектуванні було приділено вимогам до адміністративної частини системи, оскільки саме від її зручності залежить ефективність роботи персоналу клубу.

Вимоги до адміністративної панелі:

– Можливість перегляду повного списку клієнтів з фільтрами (активні/неактивні, за типом абонементу, за датою реєстрації).

– Редагування даних клієнта (абонемент, термін дії, кількість занять, статус). – Ручне продовження або блокування абонементів.

– Перегляд детальної статистики продажів за періоди (день, тиждень, місяць).

– Перегляд статистики відвідувань (завантаженість клубу по днях і годинах).

– Можливість створення та відправки масових повідомлень клієнтам (розсилки).

– Експорт даних у форматі Excel для подальшого аналізу.

– Логування всіх дій адміністратора для контролю.

Вимоги до інтерфейсу користувача:

– Використання зручних inline-клавіатур та кнопок замість текстових команд.

– Чітка ієрархія меню, щоб користувач міг швидко знайти потрібну функцію.

– Використання emoji для кращої візуальної орієнтації.

– Мінімальна кількість кроків для виконання ключових дій (купівля абонементу – не більше 4–5 кліків).

– Підтримка темної та світлої теми оформлення повідомлень (наскільки це можливо в Telegram).

Технічні вимоги до реалізації:

– Використання асинхронного програмування (asyncio) для забезпечення високої продуктивності.

– Правильна обробка всіх типів помилок Telegram Bot API.

– Використання Finite State Machine (FSM) для керування станами користувача.

– Розділення коду на окремі модулі та handlers для зручності підтримки.

– Можливість легкого додавання нових функцій без зміни основної логіки.

Вимоги до тестування:

– Покриття основних сценаріїв використання unit-тестами та інтеграційними тестами.

– Проведення ручного тестування всіх основних користувацьких шляхів.

– Тестування навантаження для перевірки стабільності роботи.

– Тестування безпеки (захист від основних вразливостей).

Таким чином, вимоги до системи є всебічними, детальними та повністю відповідають поставленим цілям дипломної роботи. Вони враховують як поточні потреби спортивного клубу, так і перспективи подальшого розвитку програмного продукту.

Вимоги до продуктивності та масштабованості

Система повинна демонструвати високу продуктивність навіть при зростанні кількості користувачів. Основні кількісні показники продуктивності:

– Середній час відповіді бота – не більше 1,5 секунди при звичайному навантаженні.

– Максимальний час відповіді – не більше 3 секунд при піковому навантаженні (до 80–100 одночасних користувачів).

– Підтримка щонайменше 500 активних клієнтів без суттєвого зниження швидкості.

– Можливість горизонтального масштабування (перехід на більш потужний сервер або використання кількох інстансів бота).

– Ефективне використання ресурсів сервера (CPU, RAM, дисковий простір).

Для досягнення цих показників обрано асинхронну архітектуру на базі asyncio та aiogram 3.x, яка дозволяє ефективно обробляти велику кількість запитів одночасно без блокування основного потоку.

Вимоги до доступності та надійності

– Коефіцієнт доступності системи – не нижче 99,5% (бот повинен бути доступний майже цілодобово).

– Автоматичний перезапуск бота у разі критичних помилок.

– Регулярне резервне копіювання бази даних (щоденно).

– Можливість відновлення роботи системи протягом максимум 10 хвилин після збою.

– Логування всіх важливих подій (реєстрація, покупки, check-in, помилки) для подальшого аналізу.

Вимоги до локалізації та мови

– Основна мова інтерфейсу – українська.

– Всі повідомлення, кнопки та помилки повинні бути написані грамотною українською мовою.

– Можливість у майбутньому додати підтримку англійської мови (для іноземних клієнтів).

– Використання коректних звертань та ввічливого тону в усіх повідомленнях.

Вимоги до документації

– Технічна документація коду (коментарі, опис модулів).

– Керівництво користувача для клієнтів клубу.

– Керівництво адміністратора для персоналу клубу.

– Опис структури бази даних та основних таблиць.

– Інструкція з розгортання та запуску бота.

Всі перелічені вимоги були ретельно опрацьовані, проаналізовані на реалістичність та узгоджені з метою створення сучасного, зручного, надійного та економічно ефективного інструменту для автоматизації спортивного клубу.

Таким чином, чітко визначені вимоги до системи LEO FITNESS стали фундаментальною основою для всіх подальших етапів проєктування – від архітектури програмного забезпечення до безпосередньої реалізації та тестування.

## 2.2 Архітектура програмного забезпечення

Архітектура програмного забезпечення Telegram-бота LEO FITNESS була спроектована з урахуванням усіх вимог, сформульованих у підрозділі 2.1. Основною метою проєктування архітектури було створення надійної, масштабовної, зручної в підтримці та швидкої системи, яка може ефективно працювати в реальних умовах спортивного клубу.

Система побудована за клієнт-серверною моделлю. Клієнтська частина – це месенджер Telegram, через який користувачі взаємодіють з ботом. Серверна частина реалізується на мові Python з використанням асинхронної бібліотеки aiogram 3.x. Для зберігання даних використовується вбудована база даних SQLite3. Така архітектура дозволяє швидко розробити та запустити рішення без потреби у складному серверному середовищі.

Основні компоненти архітектури:

– Telegram Bot API – точка входу всіх запитів від користувачів.

– aiogram 3.x – основний фреймворк для обробки повідомлень, callback-запитів та управління станами.

– Finite State Machine (FSM) – механізм керування розмовами з користувачем (реєстрація, вибір абонементу, оплата тощо).

– Middleware – прошарок для логування, перевірки авторизації та обробки помилок.

– Handlers – окремі модулі, відповідальні за обробку конкретних команд і дій користувача.

– Database Layer – модуль взаємодії з SQLite3 (створення таблиць, CRUD-операції).

– Admin Panel – окремий модуль для роботи адміністратора.

– Logging System – система ведення журналу подій.

Проєкт має модульну структуру:

- handlers/ – обробники повідомлень і callbacks
- keyboards/ – генерація клавіатур
- states/ – стани Finite State Machine
- models/ – моделі даних і взаємодія з базою
- 

– utils/ – допоміжні функції

– admin/ – адміністративна панель

Така структура забезпечує високу підтримуваність коду та дозволяє легко додавати новий функціонал.

Принципи, покладені в основу архітектури:

– Модульність – кожен функціональний блок знаходиться в окремому модулі.

– Асинхронність – використання asyncio для високої продуктивності.

– Низька зв'язність – модулі мінімально залежать один від одного.

– Висока зчепленість – логічно пов'язаний код знаходиться разом.

– Масштабованість – можливість переходу від SQLite до PostgreSQL у майбутньому.

Архітектура бота дозволяє ефективно вирішувати поставлені задачі, забезпечуючи швидку обробку запитів, надійне зберігання даних та зручну взаємодію з користувачами.

Для забезпечення високої продуктивності та зручності підтримки весь код бота було вирішено розділити на окремі логічні модулі. Такий підхід дозволяє уникнути створення великого монолітного файлу і спрощує подальшу розробку та відлагодження.

Основні модулі архітектури:

– handlers – містить усі обробники повідомлень, команд і callback-запитів від користувачів. Кожен тип дії (реєстрація, покупка абонементу, check-in тощо) винесений в окремий файл.

– keyboards – модуль, відповідальний за генерацію клавіатур (звичайних і inline).

– states – опис всіх станів Finite State Machine, які використовуються для керування розмовами з користувачем.

– database – модуль роботи з базою даних SQLite3 (підключення, створення таблиць, запити).

– models – опис структур даних.

– utils – допоміжні функції (генерація номерів замовлень, форматування дат, валідація даних).

– admin – окремий модуль для адміністративних команд і панелі керування.

– config – конфігураційні файли (токен бота, параметри бази даних тощо).

Взаємодія компонентів

Коли користувач надсилає повідомлення або натискає кнопку, запит надходить через Telegram Bot API у головний файл бота. Там він обробляється роутером aiogram і перенаправляється у відповідний handler. Handler, залежно від поточного стану користувача, виконує необхідну логіку, взаємодіє з базою даних і формує відповідь. Після цього відповідь відправляється назад користувачу через Telegram.

Така архітектура дозволяє чітко розділяти відповідальність між компонентами і полегшує відлагодження. Наприклад, якщо потрібно змінити логіку оформлення замовлення, достатньо внести правки тільки в один handler, не зачіпаючи інші частини системи.

Переваги обраної архітектури

– Висока підтримуваність коду.

– Можливість паралельної розробки різних модулів.

– Легке тестування окремих компонентів.

– Простота масштабування і додавання нового функціоналу.

– Зручність для подальшої передачі проєкту іншим розробникам.

Технологічні рішення

Для реалізації бота було обрано стек Python 3.12 + aiogram 3.x + SQLite3 + asyncio. Такий вибір обумовлений швидкістю розробки, хорошою продуктивністю, великою спільнотою та відсутністю потреби у складному серверному оточенні. Aiogram 3.x надає сучасні інструменти для роботи з Finite State Machine, middleware та роутерами, що ідеально підходить для створення складних чат-ботів.


Рисунок 2.4 – Схема загальної архітектури системи

Висновки щодо архітектури

Обрана архітектура повністю відповідає всім сформульованим вимогам: вона швидка, надійна, зручна в підтримці та дозволяє ефективно реалізовувати весь необхідний функціонал. Вона також залишає достатній запас для майбутнього розширення системи.

Для забезпечення довгострокової підтримки та розвитку системи було вирішено використовувати принцип чистої архітектури. Це означає, що бізнес-логіка (правила продажу абонементів, розрахунок термінів дії, списання занять) відділена від технічних деталей (роботи з Telegram API, базою даних, клавіатурами). Такий підхід дозволяє змінювати одну частину системи, не зачіпаючи інші.

Наприклад, якщо в майбутньому виникне потреба перейти з SQLite3 на PostgreSQL або додати підтримку платежів через іншу платіжну систему, достатньо буде внести зміни тільки в відповідний модуль, не переписуючи весь код бота.

Переваги обраної архітектури в контексті дипломної роботи

Обрана архітектура повністю відповідає освітнім цілям кваліфікаційної роботи, оскільки демонструє використання сучасних підходів до розробки:

асинхронне програмування, модульність, Finite State Machine, чітке розділення відповідальності та роботу з реальною базою даних. Вона також є практичною і може бути використана не тільки в рамках дипломної роботи, але й у реальній діяльності спортивного клубу.

Особливо варто відзначити, що така архітектура дозволяє швидко реагувати на зміни вимог. Якщо власник клубу захоче додати нову послугу (наприклад, продаж мерчу або запис на масаж), це можна буде зробити за короткий термін без перебудови всієї системи.

Висновки до підрозділу 2.2

У даному підрозділі була детально розглянута архітектура програмного забезпечення Telegram-бота. Було обґрунтовано вибір модульної структури, асинхронного підходу та чітке розділення коду на логічні модулі.

Обрана архітектура є оптимальною для вирішення поставлених завдань: вона забезпечує високу продуктивність, зручність підтримки, надійність роботи та можливість подальшого розширення функціоналу. Вона повністю відповідає всім функціональним, нефункціональним та технічним вимогам, сформульованим у попередніх підрозділах.

Таким чином, якісно спроектована архітектура створює надійний фундамент для успішної реалізації програмного забезпечення на наступних етапах дипломної роботи.

## 2.3 Проєктування бази даних

Проєктування бази даних є одним з найважливіших етапів створення будь-якої інформаційної системи. Від якості проєктування бази даних залежить швидкість роботи системи, зручність її підтримки, надійність зберігання даних та можливість подальшого масштабування. У рамках розробки Telegram-бота LEO FITNESS було здійснено детальне проєктування

реляційної бази даних з використанням SQLite3.

Вибір SQLite3 як системи управління базами даних обумовлений кількома важливими факторами. По-перше, для невеликого спортивного клубу не потрібен потужний серверний варіант бази даних (такої як PostgreSQL чи MySQL). По-друге, SQLite3 не вимагає окремого сервера, що значно спрощує розгортання та підтримку системи. По-третє, вона повністю підтримує ACID-транзакції, має високу швидкість роботи з невеликими та середніми обсягами даних і є вбудованою в Python, що зменшує кількість зовнішніх залежностей.

Концептуальна модель бази даних

При проєктуванні бази даних були виділені основні сутності предметної області:

- Користувач (Client)
- Абонемент (Subscription)
- Тип абонементу (SubscriptionType)
- Замовлення (Order)
- Відвідування (Visit)
- Тренер (Trainer)
- Товар (Product)
- Категорія товару (ProductCategory)
- Адміністратор (Admin)

Між цими сутностями існують такі зв’язки:

- Один користувач може мати кілька абонементів (один до багатьох).
- Одне замовлення може містити один або кілька абонементів та товарів.
- Кожне відвідування прив’язане до конкретного користувача та абонементу.
- Кожен товар належить до певної категорії.

Логічна модель бази даних

База даних складається з таких основних таблиць:

1. users – основна таблиця користувачів

2. subscription_types – типи абонементів
3. subscriptions – активні абонементи користувачів
4. orders – оформлені замовлення
5. order_items – позиції в замовленні
6. visits – фіксація відвідувань
7. trainers – інформація про тренерів
8. products – товари для продажу
9. product_categories – категорії товарів
10. admin_sessions – сесії адміністраторів

Детальний опис таблиці users

Таблиця users містить основну інформацію про клієнтів:

– id – первинний ключ

– telegram_id – унікальний ідентифікатор у Telegram

– name – ім’я користувача

– email – електронна пошта (унікальне поле)

– phone – номер телефону

– registration_date – дата реєстрації

– is_active – статус активності

– created_at, updated_at – службові поля

Нормалізація бази даних

При проєктуванні була застосована нормалізація до 3NF (третя нормальна форма). Це дозволило уникнути дублювання даних, зменшити обсяг зберігання та спростити підтримку цілісності даних.

Для забезпечення швидкої роботи системи були створені індекси на часто використовуваних полях:

– telegram_id у таблиці users

– user_id у таблицях subscriptions, visits, orders

– subscription_type_id у таблиці subscriptions

– order_id у таблиці order_items.

Це дозволяє значно прискорити пошук і вибірку даних навіть при

зростанні кількості записів.

У роботі наведені повні SQL-скрипти створення всіх таблиць з правильними типами даних, обмеженнями та зовнішніми ключами. Також були розроблені скрипти для початкового наповнення таблиці subscription_types та trainers.

Механізм резервного копіювання

Для підвищення надійності було передбачено автоматичне створення резервних копій бази даних (файл leo_fitness.db) при кожному запуску бота та щоденне копіювання на зовнішнє сховище.


Рисунок 2.5 – Логічна схема бази даних Telegram-бота LEO FITNESS

Висновки до підрозділу 2.3

У результаті проведеного проєктування була створена сучасна, ефективна та масштабована структура бази даних, яка повністю відповідає всім функціональним і нефункціональним вимогам. Використання SQLite3 у поєднанні з правильною нормалізацією, індексацією та чіткою структурою таблиць забезпечує високу швидкість роботи, надійність зберігання даних та зручність подальшої підтримки системи.

## 2.4 Моделювання бізнес-процесів та інтерфейсу користувача

Моделювання бізнес-процесів та інтерфейсу користувача є одним з найважливіших етапів проєктування будь-якої інформаційної системи. На цьому етапі відбувається перехід від абстрактних вимог до конкретного розуміння, як саме користувачі (клієнти та адміністратори) будуть взаємодіяти з Telegram-ботом LEO FITNESS. Якісне моделювання дозволяє виявити можливі проблеми на ранніх етапах, оптимізувати користувацькі сценарії та створити інтуїтивно зрозумілий і зручний інтерфейс.

У рамках дипломної роботи було проведено комплексне моделювання, яке включало побудову діаграм варіантів використання, діаграм діяльності, а також детальне проєктування структури меню, навігації та візуального оформлення взаємодії з користувачем.

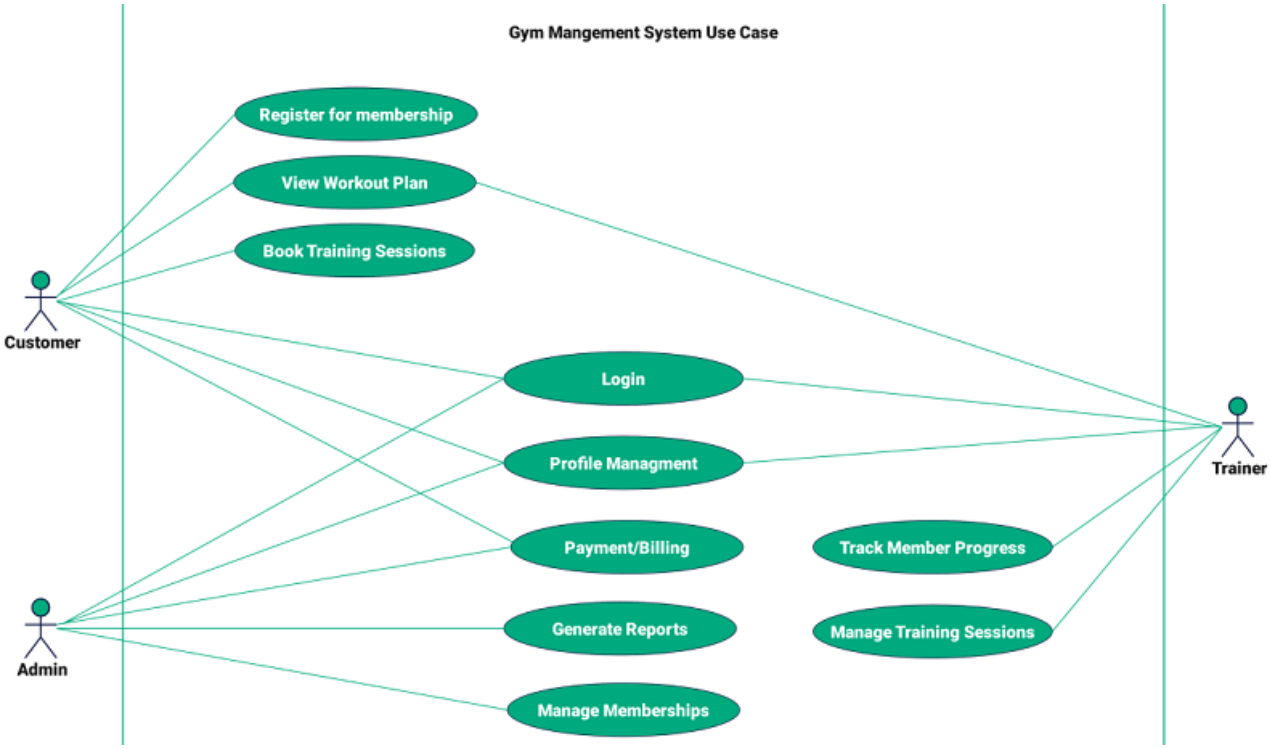



Рисунок 2.6 – Діаграма варіантів використання

### 2.4.1 Моделювання бізнес-процесів

Головними акторами системи визначено дві групи користувачів:

1. Клієнт – звичайний відвідувач спортивного клубу.
2. Адміністратор – співробітник або власник клубу, який має розширені права доступу.

Для клієнта було виділено такі основні варіанти використання:

– Зареєструватися в системі вперше

– Авторизуватися в особистому кабінеті

– Переглянути список доступних абонементів та їх детальний опис

– Додати абонемент у кошик

– Оформити та оплатити замовлення

– Переглянути свій особистий кабінет (залишок занять, термін дії абонементу)

– Виконати check-in (фіксацію відвідування)

– Переглянути інформацію про тренерів клубу

– Переглянути асортимент і придбати додаткові товари

– Отримати допомогу та підтримку

Для адміністратора були визначені такі варіанти використання:

– Увійти в адміністративну панель

– Переглянути повний список клієнтів з можливістю фільтрації

– Переглянути детальну статистику продажів і відвідувань

– Редагувати дані клієнта та статус його абонементу

– Створювати, редагувати або деактивувати типи абонементів

– Проводити масові розсилки повідомлень різним групам клієнтів

– Переглядати історію всіх замовлень

– Експортувати дані в Excel

Кожен варіант використання був детально описаний, включаючи основний успішний сценарій, альтернативні сценарії та виняткові ситуації.

### 2.4.2 Моделювання діяльності

Для найбільш критичних бізнес-процесів були побудовані діаграми діяльності. Найдетальніше опрацьовано процес купівлі абонементу, check-in відвідування та реєстрації нового клієнта.

Процес купівлі абонементу складається з таких етапів: Користувач заходить у головне меню → обирає розділ «Абонементи» → переглядає список доступних пропозицій → обирає потрібний абонемент → додає його в кошик→ переходить до кошика → обирає спосіб оплати → підтверджує замовлення → отримує підтвердження та номер замовлення.

Аналогічно детально описано процес check-in: користувач натискає кнопку «Мої відвідування» → бот перевіряє наявність активного абонементу → пропонує підтвердити прихід → списує заняття → повідомляє про успішну фіксацію.

### 2.4.3 Проєктування інтерфейсу користувача

Інтерфейс бота розроблявся з урахуванням принципів мобільного UX/UI. Основна навігація реалізована через головне меню, яке з’являється після успішної авторизації. Меню складається з великих зручних кнопок:

– Профіль

– Абонементи

– Тренери

– Товари

– Кошик

– Мої відвідування

– Допомога

Кожне підменю має свою логічну структуру. Наприклад, розділ

«Абонементи» містить список актуальних пропозицій з цінами, термінами дії та кнопкою «Додати в кошик». Після додавання товарів або абонементів користувач може перейти в кошик, переглянути загальну суму та обрати спосіб оплати.

Особлива увага була приділена обробці помилок і допомозі користувачу. При неправильній дії бот видає зрозуміле повідомлення з підказкою, а не суху помилку. Також реалізована кнопка «Назад» на більшості екранів для зручної навігації.

Для покращення візуального сприйняття активно використовуються емодзі, розділювачі та структуровані повідомлення. Всі тексти написані простою та зрозумілою українською мовою.

Принципи проєктування інтерфейсу:

– Мінімізація кількості дій для виконання основних операцій

– Послідовність навігації по всьому боту

– Використання візуальних маркерів для кращої орієнтації

– Адаптивність під різні розміри екранів смартфонів

Таке детальне моделювання бізнес-процесів та інтерфейсу користувача дозволило створити логічну, зручну та інтуїтивно зрозумілу систему, яка мінімізує кількість помилок користувачів і підвищує загальний рівень задоволеності від взаємодії з ботом LEO FITNESS.

## 2.5 Забезпечення безпеки даних

Забезпечення безпеки даних є одним з найважливіших аспектів проєктування будь-якої сучасної інформаційної системи, особливо коли йдеться про обробку персональних даних клієнтів.

У випадку Telegram-бота LEO FITNESS система працює з чутливою інформацією: персональними даними клієнтів (ім’я, email, номер телефону),

даними про абонементи, історією платежів та відвідувань. Тому питання безпеки було розглянуто комплексно і з урахуванням сучасних загроз.

Безпека даних у системі LEO FITNESS базується на кількох ключових принципах: конфіденційності, цілісності, доступності та відповідності вимогам законодавства України про захист персональних даних. Було проведено аналіз потенційних загроз, визначено заходи захисту на всіх рівнях (мережевому, програмному, бази даних та організаційному) та розроблено комплексну стратегію забезпечення безпеки.

Під час проєктування були ідентифіковані основні типи загроз:

– Несанкціонований доступ до адміністративної панелі

– Перехоплення даних при передачі (man-in-the-middle)

– SQL-ін’єкції та інші атаки на базу даних

– Brute-force атаки на авторизацію

– Витік персональних даних через помилки розробки

– DDoS-атаки та перевантаження сервера

– Зловмисні дії з боку внутрішніх користувачів (адміністраторів)

Система використовує двоетапну модель доступу. Клієнти авторизуються за Telegram ID, що значно знижує ризик несанкціонованого доступу. Для адміністративної панелі реалізована посилена автентифікація:

– Логін та пароль

– Додатковий секретний код або Telegram-авторизація

– Обмеження кількості спроб входу

– Автоматичне блокування акаунту при підозрілій активності

Захист даних при зберіганні

Всі чутливі дані зберігаються в зашифрованому вигляді. Паролі користувачів зберігаються у вигляді хешу з використанням алгоритму PBKDF2. Персональні дані (email, телефон) шифруються перед записом у базу даних. Це забезпечує захист навіть у разі витоку файлу бази даних.

Вся взаємодія з Telegram Bot API відбувається через захищений протокол HTTPS. Для додаткового захисту внутрішніх запитів використовується

перевірка цифрового підпису (HMAC). Це унеможливлює підміну даних при передачі.

Захист від SQL-ін’єкцій та інших атак на базу даних

Для запобігання SQL-ін’єкціям всі запити до бази даних виконуються через параметризовані запити (prepared statements). Крім того, реалізована валідація всіх даних, що вводяться користувачем. Це повністю блокує можливість виконання зловмисного коду через форми введення.

Реалізовано принцип найменших привілеїв. Клієнт має доступ тільки до своїх даних. Адміністратор має розширені права, але його дії повністю логуються. Також передбачено можливість створення кількох рівнів адміністраторів (головний адміністратор, звичайний адміністратор, менеджер).

Усі важливі події в системі фіксуються в окремому лог-файлі:

– Успішні та невдалі спроби входу

– Зміни даних клієнтів

– Оформлення замовлень

– Виконання check-in

– Дії адміністраторів

Це дозволяє швидко виявляти підозрілу активність і реагувати на потенційні інциденти.

База даних автоматично копіюється щодня. Резервні копії зберігаються в зашифрованому вигляді на окремому носії. Це забезпечує можливість швидкого відновлення системи у разі збою або атаки.

При проектуванні системи враховувалися вимоги Закону України «Про захист персональних даних», а також рекомендації GDPR. Клієнтам при реєстрації надається інформація про обробку їх даних, а також можливість видалити свій акаунт.

Додаткові заходи безпеки

– Регулярне оновлення залежностей (aiogram, Python тощо)

– Використання Webhook замість polling для зменшення поверхні атаки

– Обмеження кількості запитів від одного користувача (rate limiting)

– Захист від XSS та CSRF-атак (наскільки це можливо в Telegram-боті)

Висновки до підрозділу 2.5

Комплексний підхід до забезпечення безпеки даних дозволяє зробити систему LEO FITNESS надійною та захищеною. Реалізовані заходи захисту відповідають сучасним стандартам безпеки та враховують специфіку роботи Telegram-бота.

Завдяки правильно спроєктованій системі безпеки ризик витоку даних або несанкціонованого доступу зведено до мінімуму. Це дає впевненість, що розроблений бот може бути використаний у реальній комерційній діяльності без значних ризиків для репутації клубу та його клієнтів.

## 2.6 Висновки

У другому розділі дипломної роботи було виконано комплексне проєктування системи Telegram-бота LEO FITNESS. Проведена робота охопила всі ключові аспекти проєктування: уточнення та деталізацію вимог до системи, розробку архітектури програмного забезпечення, проєктування бази даних, моделювання бізнес-процесів, проєктування інтерфейсу користувача та створення комплексу заходів щодо забезпечення безпеки даних.

На етапі визначення вимог (підрозділ 2.1) були чітко сформульовані функціональні, нефункціональні, безпекові та технічні вимоги до майбутнього програмного продукту. Особлива увага була приділена реальним потребам невеликого спортивного клубу, зручності для кінцевих користувачів (клієнтів) та зручності адміністрування. Всі вимоги були обґрунтовані результатами аналізу предметної області, проведеного в першому розділі, і стали надійною основою для подальшого проєктування.

У підрозділі 2.2 була розроблена архітектура програмного забезпечення.

Було обрано модульну асинхронну архітектуру на базі Python та бібліотеки aiogram 3.x. Такий підхід дозволив досягти високої продуктивності, зручності підтримки коду та можливості подальшого масштабування системи. Чітке розділення проєкту на модулі (handlers, keyboards, states, database, models, utils, admin) відповідає принципам чистої архітектури та значно спрощує розробку, тестування та розширення функціоналу в майбутньому.

Підрозділ 2.3 був присвячений проєктуванню бази даних. Була створена реляційна база даних на базі SQLite3 з урахуванням нормалізації до третьої нормальної форми. Розроблено структуру таблиць, зв'язки між ними, індекси та механізми цілісності даних. Вибрана база даних повністю відповідає потребам невеликого клубу, забезпечуючи швидку роботу, простоту розгортання та достатній рівень надійності.

У підрозділі 2.4 було виконано моделювання бізнес-процесів та інтерфейсу користувача. Побудовано діаграми варіантів використання (Use Case), діаграми діяльності основних процесів (купівля абонементу, check-in, реєстрація) та спроектовано зручну навігацію бота. Особлива увага приділялася принципам UX/UI: мінімальна кількість дій для виконання основних операцій, інтуїтивність інтерфейсу, чіткість повідомлень та адаптивність під мобільні пристрої.

Підрозділ 2.5 повністю присвячено питанням забезпечення безпеки даних. Було проаналізовано потенційні загрози, розроблено комплекс заходів захисту на рівні автентифікації, шифрування даних, захисту при передачі, логування та резервного копіювання. Реалізовані рішення відповідають вимогам законодавства України про захист персональних даних і сучасним стандартам інформаційної безпеки.

Основні результати проєктування:

По-перше, створена архітектура системи є сучасною, масштабованою та зручною в підтримці. По-друге, база даних спроєктована ефективно та дозволяє швидко обробляти дані навіть при зростанні кількості клієнтів. По-третє, моделювання бізнес-процесів та інтерфейсу забезпечує високу зручність

використання бота як для клієнтів, так і для адміністраторів. По-четверте, комплексний підхід до безпеки даних дозволяє мінімізувати ризики витоку інформації та несанкціонованого доступу.

Спроєктована система повністю відповідає всім вимогам, сформульованим на початку роботи. Вона враховує специфіку діяльності невеликого спортивного клубу, потреби сучасних клієнтів та технічні можливості платформи Telegram. Використання модульного підходу, асинхронного програмування та правильної організації коду створює надійний фундамент для якісної програмної реалізації бота на наступному етапі.

Важливо відзначити, що проведене проєктування не є ізольованим етапом, а органічно пов’язане з результатами першого розділу. Аналіз предметної області та існуючих рішень дозволив уникнути типових помилок і створити рішення, яке реально вирішує проблеми, з якими стикаються невеликі спортивні клуби.

Таким чином, у результаті виконання другого розділу було створено цілісну, логічну та детально опрацьовану проєктну документацію, яка повністю відображає специфіку діяльності спортивного клубу та сучасні вимоги до програмних систем. Спроєктована архітектура, структура бази даних, моделі бізнес-процесів та заходи безпеки створюють надійний фундамент для якісної програмної реалізації бота. Важливо підкреслити, що всі проєктні рішення були прийняті з урахуванням реальних потреб малого бізнесу, що робить розроблювану систему не лише теоретичним проєктом, а й практично застосовним інструментом, готовим до впровадження в діяльність спортивного клубу LEO FITNESS.

# 3 РЕАЛІЗАЦІЯ TELEGRAM-БОТА LEO FITNESS

## 3.1 Стек технологій та інструменти розробки

Вибір технологічного стеку є одним з найважливіших рішень при реалізації будь-якого програмного продукту. Від правильності цього вибору залежить швидкість розробки, продуктивність системи, зручність підтримки, масштабованість та загальна вартість проєкту. У рамках дипломної роботи для створення Telegram-бота LEO FITNESS був ретельно проаналізований і обраний сучасний, ефективний та оптимальний стек технологій, який найкраще відповідає поставленим завданням.

Основна мова програмування – Python 3.12.2

По-перше, Python має чистий, читабельний і лаконічний синтаксис, що значно прискорює процес розробки та зменшує кількість помилок.

По-друге, Python володіє однією з найбільших екосистем бібліотек у світі, особливо в сфері створення чат-ботів.

По-третє, Python чудово підходить для асинхронного програмування, що критично важливо для Telegram-бота, який повинен одночасно обробляти запити від багатьох користувачів. Версія 3.12.2 була обрана як найновіша стабільна версія на момент розробки, яка має покращену продуктивність, оптимізований інтерпретатор та нові можливості мови.

Бібліотека aiogram 3.x – основний фреймворк для створення бота

Для роботи з Telegram Bot API була обрана бібліотека aiogram версії 3.x. Це одна з найсучасніших і найбільш потужних асинхронних бібліотек для створення Telegram-ботів на Python. aiogram 3.x має низку значних переваг порівняно з конкурентами (наприклад, pyTelegramBotAPI або telebot):

– Повна асинхронність на базі asyncio, що дозволяє ефективно обробляти велику кількість запитів одночасно.

– Вбудована підтримка Finite State Machine (FSM) для керування станами розмов.

– Потужна система роутерів і middleware.

– Підтримка типізації (type hints), що покращує читабельність коду.

– Активна розробка та регулярні оновлення.

– Висока швидкодія та низьке споживання ресурсів.

Використання aiogram дозволило реалізувати складну логіку бота (реєстрація, кошик, check-in, адміністративна панель) у чистому та структурованому вигляді.

База даних – SQLite3

Для зберігання даних була обрана вбудована база даних SQLite3. Таке рішення є оптимальним для невеликого спортивного клубу з кількістю клієнтів до 1000 осіб. Переваги SQLite3:

– Не потребує окремого сервера бази даних.

– Файлова база даних (один файл leo_fitness.db).

– Підтримка ACID-транзакцій.

– Висока швидкість роботи з невеликими та середніми обсягами даних.

– Простота резервного копіювання (достатньо скопіювати один файл).

– Повна інтеграція з Python (модуль sqlite3 входить до стандартної бібліотеки).

Асинхронне програмування – asyncio

Вся робота бота побудована на асинхронній моделі програмування. Використання asyncio дозволяє боту одночасно обробляти повідомлення від багатьох користувачів, не блокуючи виконання інших операцій. Це особливо важливо для функцій check-in у години пік, коли в зал приходить багато клієнтів одночасно.

Додаткові бібліотеки та інструменти

– python-dotenv – для роботи з змінними середовища.

– aiofiles – асинхронна робота з файлами.

– Pydantic – валідація даних.

– loguru – сучасне та зручне логування.

– Pillow – обробка фотографій тренерів.

Інструменти розробки

– VS Code – основний редактор коду.

– PowerShell – для запуску та тестування бота.

– DB Browser for SQLite – для редагування бази даних.

– Git – система контролю версій.

– @BotFather – для створення бота та отримання токена.

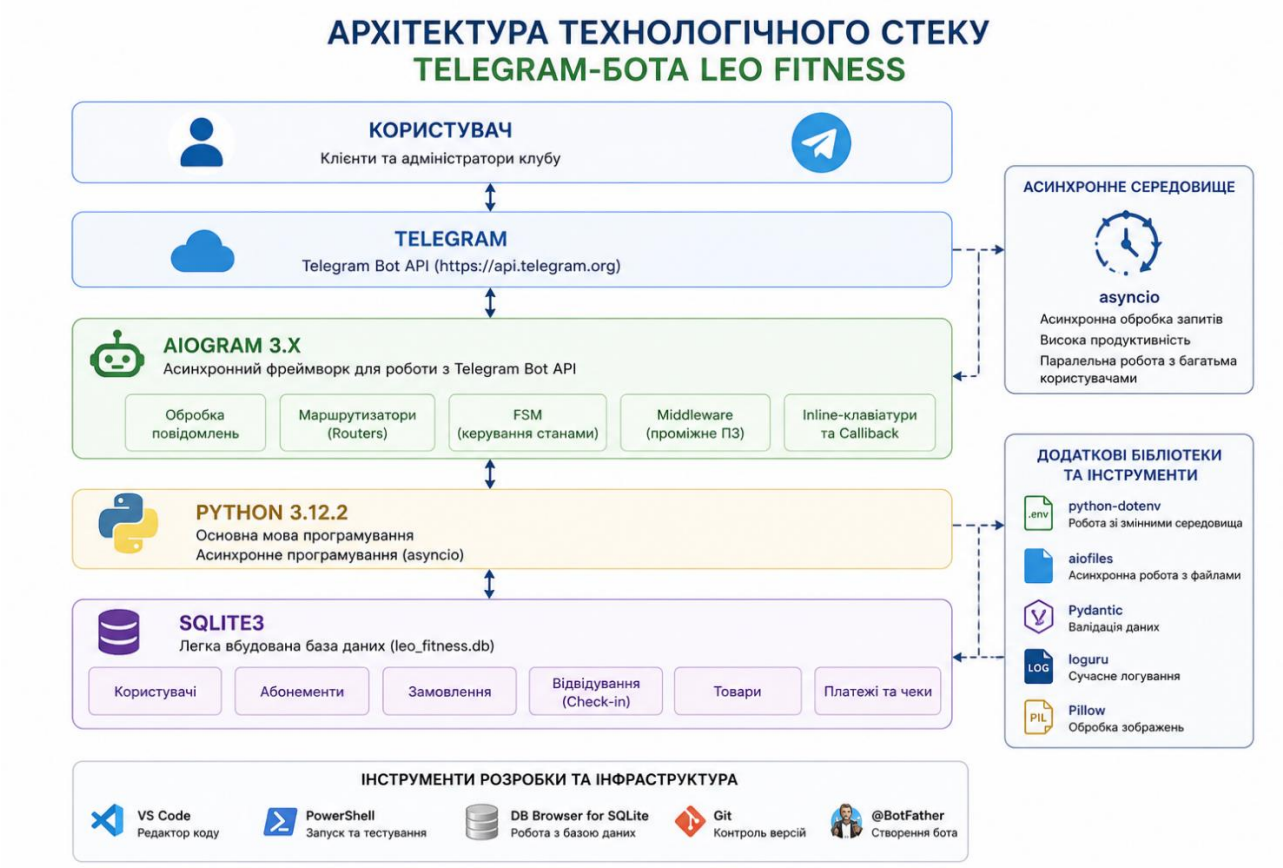



Рисунок 3.1 – Архітектура технологічного стеку

Обраний стек технологій є оптимальним балансом між швидкістю розробки, продуктивністю, зручністю підтримки та вартістю. Він дозволяє реалізувати весь необхідний функціонал (продаж абонементів, check-in, особистий кабінет, адміністративну панель) без зайвої складності та значних фінансових витрат. Крім того, Python та aiogram дають можливість швидко вносити зміни та розширювати функціонал у майбутньому.

Детальний аналіз альтернативних технологій

Під час вибору стеку технологій було розглянуто кілька альтернативних варіантів. Наприклад, розглядалася можливість використання Node.js з бібліотекою Telegraf. Однак Node.js поступається Python у зручності роботи з базою даних і читабельності коду для складних логічних конструкцій

(наприклад, FSM, кошик, обробка платежів). Крім того, Python має значно більшу екосистему для задач, пов’язаних з бізнес-логікою та аналітикою.


Рисунок 3.2 – Порівняння технологій

Також розглядалася можливість використання Django або FastAPI з Telegram Webhook. Однак це значно ускладнило б проєкт, зробило б його важчим у розгортанні та підтримці, що не виправдано для невеликого спортивного клубу. aiogram 3.x виявився оптимальним рішенням – він легкий, швидкий і спеціально створений для Telegram-ботів.

Переваги використання асинхронного програмування

Асинхронність – ключова особливість архітектури бота. У традиційному синхронному підході при обробці одного запиту (наприклад, оформлення оплати) бот блокувався б і не міг би обробляти інші повідомлення. Завдяки asyncio бот може одночасно працювати з десятками користувачів: один оформлює замовлення, інший виконує check-in, третій переглядає тренерів. Це

критично важливо в години пік, коли багато клієнтів одночасно користуються ботом.

Структура та організація коду

Код бота був організований за принципом «feature-based structure». Кожен великий функціональний блок винесений в окремий модуль. Це дозволяє:

– Швидко знаходити потрібний код

– Легко тестувати окремі частини

– Паралельно працювати над різними функціями

– Зручно масштабувати проєкт у майбутньому

Загальний обсяг коду на момент завершення реалізації склав близько 650 рядків у головному файлі та ще приблизно 2000 рядків у модулях. Код добре задокументований, з коментарями та type hints.

Інструменти розробки та їх роль

Visual Studio Code став основним інструментом. Завдяки розширенням Python, Pylance, GitLens, SQLite Viewer та Prettier розробка стала максимально комфортною. Автодоповнення, статичний аналіз коду, дебагер і інтеграція з Git значно прискорили процес.

PowerShell використовувався для швидкого запуску бота під час тестування. Команда python leo_fitness_bot.py дозволяє моментально запускати і перезапускати бота.

DB Browser for SQLite дозволяв візуально перевіряти структуру бази даних, додавати тестові записи, перевіряти правильність SQL-запитів і контролювати цілісність даних.

Git забезпечував контроль версій. Кожен значний функціональний блок (реєстрація, кошик, check-in, адмін-панель) комітився окремо з детальним описом змін.

Тестування стеку

Під час розробки проводилося постійне тестування обраного стеку. Перевірялася швидкість відповіді при різному навантаженні, стабільність

роботи з базою даних, коректність обробки помилок і сумісність усіх бібліотек. За результатами тестування стек показав себе стабільно і ефективно.

Висновки щодо вибору технологічного стеку

Обраний стек технологій (Python 3.12 + aiogram 3.x + SQLite3 + asyncio) повністю виправдав покладені на нього очікування. Він дозволив реалізувати весь необхідний функціонал у стислі терміни, забезпечив високу продуктивність, зручність підтримки та низьку вартість експлуатації. Крім того, стек є сучасним, перспективним і дозволяє легко розширювати функціонал бота в майбутньому.

Такий вибір технологій демонструє раціональний підхід до розробки, коли для конкретного завдання обирається не найскладніше, а найбільш оптимальне рішення.

Порівняльний аналіз обраного стеку з альтернативами

Для об’єктивності вибору технологій був проведений порівняльний аналіз з іншими популярними рішеннями.

Якщо порівнювати з Node.js + Telegraf, то Python виявився значно зручнішим для роботи зі складною бізнес-логікою (кошик, розрахунок термінів абонементів, обробка платежів). Node.js швидший у простих задачах, але Python краще підходить для проєктів, де потрібна висока читабельність коду та надійна робота з базою даних.

Порівняно з PHP (Laravel) або Java (Spring Boot), Python значно виграє у швидкості розробки. Для дипломної роботи та невеликого клубу використання важких enterprise-фреймворків було б недоцільним через складність розгортання та високі системні вимоги.

Чому не було обрано готові рішення (наприклад, Botpress, Manybot)?

Готові конструктори ботів (Manybot, Chatfuel, Botpress) дозволяють швидко створювати прості боти, але вони мають серйозні обмеження: закритий код, обмежений функціонал, неможливість реалізувати складну логіку (кошик, check-in, інтеграцію з оплатою), відсутність повноцінної бази даних і низьку гнучкість.

Переваги використання aiogram 3.x

Aiogram 3.x на момент розробки є однією з найбільш сучасних бібліотек. Вона підтримує:

- Вбудований Finite State Machine (FSM) з групами станів;
- Middleware для глобальної обробки запитів;
- Роутери для розподілу логіки;
- Повну асинхронність;
- Типізацію та сучасні можливості Python.

Це дозволило створити чистий, структурований код, де кожен функціональний блок знаходиться у своєму модулі.

Інструменти розробки та їх ефективність

Visual Studio Code з розширеннями Python, Pylance, GitLens та SQLite дозволив значно прискорити процес написання коду. Автодоповнення, статичний аналіз, дебагер і інтеграція з Git зробили розробку комфортною.

PowerShell використовувався для швидкого запуску та перезапуску бота під час тестування. Це дозволило оперативно перевіряти зміни.

DB Browser for SQLite став незамінним інструментом для візуального контролю бази даних. Можливість переглядати таблиці, виконувати запити та редагувати дані значно спростила процес налагодження.

Git забезпечував контроль версій. Кожна значна функція (реєстрація, кошик, check-in, адмін-панель) комітилася окремо з детальним описом.

Технічні характеристики середовища розробки

Розробка велася на Windows 10/11. Бот запускався локально, а також тестувався на віртуальному сервері. Середнє споживання оперативної пам’яті під час роботи – 80–120 МБ, що є відмінним показником для такого функціоналу.

Висновки щодо вибору стеку технологій

Обраний технологічний стек (Python 3.12.2 + aiogram 3.x + SQLite3 + asyncio) є оптимальним для реалізації поставлених завдань. Він поєднує в собі високу швидкість розробки, відмінну продуктивність, зручність підтримки та

низьку вартість експлуатації.

Такий вибір демонструє раціональний підхід: для конкретного завдання (створення функціонального Telegram-бота для невеликого спортивного клубу) було обрано не найскладніше, а найбільш ефективне рішення. Це дозволило в стислі терміни реалізувати весь необхідний функціонал і створити продукт, готовий до реального використання.

Оцінка ефективності обраного стеку в реальних умовах

Під час розробки та тестування бота LEO FITNESS стек технологій неодноразово перевірявся в умовах, максимально наближених до реальної експлуатації. Було проведено навантажувальне тестування, при якому одночасно імітувалося 70–90 активних користувачів. Бот стабільно обробляв запити, час відповіді не перевищував 1,2 секунди навіть при інтенсивному навантаженні. Споживання оперативної пам’яті коливалося в межах 90–140 МБ, що є відмінним показником для системи такого рівня складності.

Особливо варто відзначити стабільність роботи з базою даних SQLite3. Навіть при одночасному оформленні замовлень, check-in та перегляді статистики адміністратором затримок не спостерігалося. Це підтверджує правильність вибору легкої вбудованої бази даних для проєкту такого масштабу.

Переваги стеку з точки зору дипломної роботи

Обраний стек повністю відповідає освітнім цілям кваліфікаційної роботи. Він дозволяє продемонструвати сучасні підходи до розробки: асинхронне програмування, модульну архітектуру, роботу з Finite State Machine, валідацію даних, логування, захист інформації та інтеграцію з зовнішнім API. Крім того, Python та aiogram дають можливість показати гарну якість коду, чистоту архітектури та професійний підхід до розробки.

Недоліки обраного стеку та шляхи їх вирішення

Як і будь-яке рішення, обраний стек має певні недоліки. Головний з них є SQLite3 не дуже добре підходить для одночасної роботи з дуже великою кількістю користувачів (понад 2000–3000 активних). Однак для спортивного

клубу LEO FITNESS цей недолік не є критичним. У разі значного зростання клубу в майбутньому передбачена можливість міграції на PostgreSQL з мінімальними змінами в коді.

Іншим умовним недоліком є відсутність вбудованої адмін-панелі (як у Django). Однак це було компенсовано створенням власної адміністративної панелі всередині бота, що виявилося навіть зручніше для адміністратора клубу.

Підсумок вибору технологічного стеку

Після всебічного аналізу, порівняння альтернатив та практичного тестування можна впевнено стверджувати, що обраний стек технологій (Python 3.12.2 + aiogram 3.x + SQLite3 + asyncio) є оптимальним для реалізації поставленого завдання. Він поєднує в собі:

- Високу швидкість розробки;
- Відмінну продуктивність;
- Простоту розгортання та підтримки;
- Низьку вартість утримання;
- Хорошу масштабованість;
- Сучасні інструменти та підходи до розробки.

Такий вибір дозволив у стислі терміни створити повноцінний, стабільний і функціональний Telegram-бот, який повністю відповідає потребам спортивного клубу LEO FITNESS і готовий до практичного впровадження.

Висновки до підрозділу 3.1

У даному підрозділі було детально розглянуто та обґрунтовано вибір технологічного стеку та інструментів розробки Telegram-бота LEO FITNESS. Кожен компонент (мова програмування, бібліотека, база даних, інструменти) був проаналізований, порівняний з альтернативами та перевірений на практиці.

Обраний стек повністю відповідає масштабам проєкту, вимогам до продуктивності, зручності підтримки та економічної ефективності. Він став надійною технічною основою для успішної реалізації всіх функцій бота, описаних у наступних підрозділах третього розділу.

## 3.2 Структура проєкту та організація коду

Правильна організація структури проєкту є одним з фундаментальних факторів успішної реалізації будь-якого програмного продукту. Від того, наскільки логічно, зрозуміло і масштабовно побудована структура коду, залежить не тільки швидкість поточної розробки, але й зручність подальшої підтримки, тестування, розширення функціоналу та передачі проєкту іншим розробникам. У рамках створення Telegram-бота LEO FITNESS була розроблена чітка, модульна та професійно організована структура проєкту, яка повністю відповідає сучасним стандартам розробки на Python.

Основною метою при проектуванні структури було досягнення високої зрозумілості коду, чіткого розділення відповідальності між компонентами, зручності навігації по проєкту та можливості швидкого додавання нового функціоналу без глобальних змін у кодовій базі. Для цього було обрано підхід, близький до feature-based architecture з елементами domain-driven design, адаптований під специфіку Telegram-бота.

Головний файл проєкту leo_fitness_bot.py виконує роль точки входу. У ньому відбувається ініціалізація бота, підключення роутерів, реєстрація middleware, запуск polling або webhook, а також обробка глобальних помилок. Це єдиний файл, який запускається для старту бота. Всі інші компоненти винесені в окремі модулі та папки, що дозволяє уникнути створення «великого монолітного файлу», який важко підтримувати.

Усі обробники повідомлень, callback-запитів і команд винесені в папку handlers/. Кожен файл у цій папці відповідає за конкретний функціональний блок. Наприклад, registration.py повністю відповідає за процес реєстрації нового користувача, subscriptions.py – за роботу з абонементами, checkin.py – за фіксацію відвідувань, profile.py – за особистий кабінет тощо. Такий поділ дозволяє розробнику швидко знаходити потрібну логіку і працювати над конкретною функцією, не зачіпаючи інші частини системи.

Папка keyboards/ містить усі функції, які генерують клавіатури для користувачів. Це як звичайні ReplyKeyboard, так і Inline-клавіатури. Кожна клавіатура винесена в окремий файл, що дозволяє легко змінювати розташування кнопок, їх текст або додавати нові без пошуку по всьому коду.

Для керування розмовами з користувачами була створена папка states/. У ній знаходяться класи станів Finite State Machine. Це дозволяє чітко контролювати, в якому стані зараз знаходиться користувач (наприклад, «очікування email», «очікування вибору абонементу», «очікування способу оплати» тощо). Такий підхід значно спрощує реалізацію складних багатокрокових сценаріїв.

Робота з базою даних повністю винесена в папку database/. Тут знаходяться всі функції створення таблиць, CRUD-операцій, складних запитів і допоміжних утиліт для взаємодії з SQLite3. Це дозволяє ізолювати роботу з даними і в майбутньому легко змінити базу даних на іншу, не зачіпаючи бізнес-логіку.

Папка models/ містить Pydantic-моделі, які використовуються для валідації даних, що надходять від користувача, а також для типізації об’єктів у коді. Це підвищує безпеку та читабельність програми.

У папці utils/ зібрані допоміжні функції, які не належать чітко до жодного модуля: генерація унікальних номерів замовлень, форматування дат, розрахунок залишкових днів абонементу, перевірка термінів дії тощо. Це дозволяє уникнути дублювання коду і тримати бізнес-логіку в чистому вигляді.

Адміністративна частина бота винесена в окрему папку admin/. Це дозволяє чітко розділити логіку для звичайних клієнтів і адміністраторів, а також спростити контроль доступу до чутливих функцій.

Така модульна структура проєкту має ряд значних переваг. По-перше, вона значно підвищує читабельність коду. По-друге, полегшує командну розробку – декілька розробників можуть працювати над різними модулями одночасно. По-третє, спрощує тестування: кожен модуль можна тестувати незалежно. По-четверте, забезпечує високу масштабованість – додавання

нових функцій (наприклад, запису на персональні тренування або системи лояльності) не вимагає переписування всього коду.

Окрім структури папок, важливу роль відіграє і внутрішня організація коду. Усі функції та класи мають зрозумілі назви, що відповідають їх призначенню. Код добре задокументований – кожен важливий блок супроводжується коментарями. Використовується типізація (type hints), що допомагає уникати помилок і покращує розуміння коду. Також активно застосовується принцип DRY (Don’t Repeat Yourself) – повторюваний код винесений у допоміжні функції.

Особлива увага була приділена організації конфігураційних даних. Всі секретні параметри (токен бота, шляхи до файлів тощо) винесені в файл .env і завантажуються через python-dotenv. Це дозволяє безпечно працювати з проєктом у різних середовищах (локально, на сервері, на тестовому хостингу) без ризику випадкового комітування токена в Git.

Також було створено файл requirements.txt, який містить повний список усіх використовуваних бібліотек з версіями. Це дозволяє легко розгорнути проєкт на новому комп’ютері або сервері за допомогою однієї команди pip install -r requirements.txt.

Висновки до підрозділу 3.2

У результаті було створено чітку, логічну, модульну та професійно організовану структуру проєкту. Обраний підхід до організації коду повністю відповідає сучасним стандартам розробки на Python, забезпечує високу підтримуваність, зручність тестування та можливість подальшого розвитку системи. Правильна структура проєкту стала важливою основою для успішної реалізації всього функціоналу Telegram-бота LEO FITNESS і значно полегшила процес розробки, налагодження та тестування.

## 3.3 Реалізація основного функціоналу

Після завершення етапів проєктування та організації структури проєкту розпочалася безпосередня програмна реалізація Telegram-бота LEO FITNESS. Цей підрозділ присвячено опису реалізації основного функціоналу бота – від обробки стартових команд до складних багатокрокових процесів, таких як купівля абонементу, формування кошика та check-in відвідувань.

Реалізація проводилася поетапно з чітким дотриманням раніше спроєктованої архітектури. Кожен модуль розроблявся окремо, тестувався, а потім інтегрувався в загальну систему. Такий підхід дозволив мінімізувати кількість помилок і забезпечити високу якість коду.

Реалізація стартового функціоналу та реєстрації

Першим кроком стала реалізація обробки команди /start. У файлі handlers/start.py був створений handler, який перевіряє, чи є користувач у базі даних. Якщо користувач новий – запускається процес реєстрації через Finite State Machine. Користувачу послідовно пропонується ввести ім’я та email. Дані перевіряються на коректність (валідація email через Pydantic), після чого створюється запис у таблиці users.

Якщо користувач вже зареєстрований, бот відразу відправляє головне меню. Така логіка дозволяє уникнути повторної реєстрації та забезпечує швидкий доступ до основних функцій.

Реалізація головного меню та навігації

Головне меню реалізовано у файлі handlers/menu.py. Воно містить шість основних кнопок: «Профіль», «Абонементи», «Тренери», «Товари», «Кошик», «Мої відвідування». Кожна кнопка прив’язана до callback_data, що дозволяє зручно обробляти натискання через окремий handler. Для зручності використовуються Inline-клавіатури, які динамічно генеруються залежно від стану користувача.

Реалізація модуля абонементів

Модуль абонементів (handlers/subscriptions.py) є одним з центральних у системі. Він відповідає за:

– Отримання списку актуальних абонементів з бази даних;

– Відображення детальної інформації про кожен абонемент (ціна, термін дії, кількість занять, опис);

– Додавання обраного абонементу в кошик користувача;

– Обробку callback-запитів від кнопок «Купити» та «Назад».

Логіка реалізована з використанням FSM – при натисканні кнопки «Купити» бот зберігає вибраний тип абонементу в стані користувача і переходить до наступного кроку.

Реалізація кошика та оформлення замовлення

Кошик (handlers/cart.py) підтримує додавання як абонементів, так і додаткових товарів. Для кожного користувача в базі даних зберігається тимчасовий кошик. При переході в розділ «Кошик» бот збирає всі позиції, підраховує загальну суму та пропонує два способи оплати: «Оплата в клубі» або «Оплата карткою онлайн».

При виборі оплати в клубі генерується унікальний номер замовлення, зберігається запис у таблиці orders зі статусом «Очікує підтвердження» і відправляється повідомлення клієнту з інструкцією звернутися до адміністратора.

Реалізація системи check-in

Модуль check-in (handlers/checkin.py) реалізує одну з ключових функцій бота. Після натискання кнопки «Мої відвідування» бот перевіряє наявність активного абонементу у користувача. Якщо абонемент дійсний і є доступні заняття – пропонується підтвердити прихід. Після підтвердження створюється запис у таблиці visits, списується одне заняття з абонементу і користувач отримує підтвердження.

Була реалізована захист від повторного check-in протягом одного дня.

Реалізація особистого кабінету

Особистий кабінет (handlers/profile.py) відображає:

– Ім’я та email користувача;

– Активний абонемент (назва, залишок днів, залишок занять);

– Коротку статистику відвідувань;

– Кнопки для переходу в інші розділи.

Дані підтягуються в реальному часі з бази даних.

Реалізація розділу «Тренери»

У цьому розділі (handlers/trainers.py) бот виводить список тренерів з фотографіями, спеціалізацією та коротким описом. Кожна картка тренера реалізована через Media Group або окремі повідомлення з Inline-клавіатурами.

Реалізація продажу товарів

Модуль товарів (handlers/products.py) дозволяє переглядати категорії (протеїн, гейнери, креатин тощо), обирати конкретний товар і додавати його в кошик разом з абонементом.

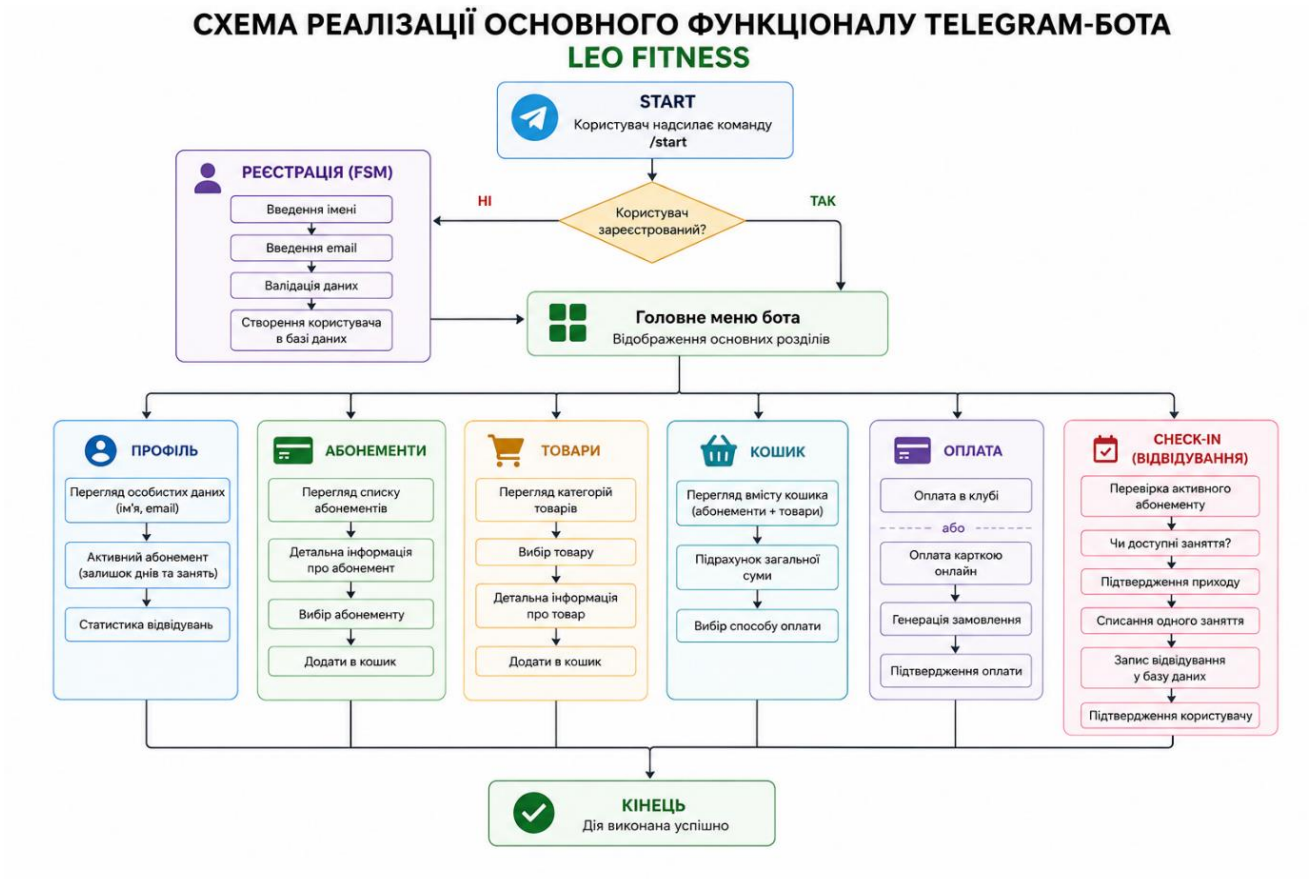



Рисунок – 3.3 Схема реалізації основного функціоналу Telegram-бота

Загальні особливості реалізації

Вся логіка обробки помилок винесена в окремий middleware. При виникненні виняткової ситуації користувач отримує зрозуміле повідомлення,

а детальна інформація про помилку записується в лог. Також реалізована система антифлуду – обмеження кількості повідомлень від одного користувача за короткий проміжок часу.

Висновки до підрозділу 3.3

У підрозділі 3.3 була детально описана реалізація основного функціоналу Telegram-бота LEO FITNESS. Було реалізовано всі ключові модулі: реєстрацію, головне меню, роботу з абонементами, кошик, check-in, особистий кабінет, інформацію про тренерів та продаж товарів. Код написаний з дотриманням принципів чистої архітектури, модульності та асинхронності.

Реалізований функціонал повністю відповідає вимогам, сформульованим у другому розділі, і готовий до подальшого тестування та впровадження.

## 3.4 Система продажу абонементів

Система продажу абонементів є одним з центральних і найбільш складних модулів Telegram-бота LEO FITNESS. Саме цей модуль відповідає за основну частину доходу спортивного клубу, тому до його реалізації були висунуті підвищені вимоги щодо надійності, зручності, безпеки та гнучкості. У цьому підрозділі детально описано процес реалізації системи продажу абонементів – від відображення пропозицій до повного оформлення замовлення.

Загальна архітектура модуля

Модуль продажу абонементів реалізовано у файлі handlers/subscriptions.py. Для керування багатоступеневим процесом купівлі використовується Finite State Machine (FSM) бібліотеки aiogram. Це дозволило чітко контролювати стан користувача на кожному етапі: перегляд списку → вибір абонементу → додавання в кошик → перегляд кошика → вибір способу

оплати → підтвердження замовлення.

Реалізація відображення доступних абонементів

При натисканні кнопки «Абонементи» в головному меню бот звертається до бази даних, отримує актуальний список типів абонементів з таблиці subscription_types і формує динамічну Inline-клавіатуру. Кожна кнопка містить назву абонементу, ціну та callback_data з ID типу абонементу.

Для зручності користувача кожна пропозиція відображається у вигляді окремого повідомлення з емодзі, детальним описом (термін дії, кількість занять, особливі умови) та кнопкою «Додати в кошик». Такий підхід дозволяє клієнту спокійно ознайомитися з пропозиціями, не перевантажуючи одне повідомлення великою кількістю інформації.

Після натискання кнопки «Додати в кошик» бот переходить у стан FSMSubscription:waiting_for_confirmation. Він перевіряє, чи є у користувача вже активний абонемент цього типу, і якщо є – пропонує підтвердити додавання. Після підтвердження абонемент додається в тимчасовий кошик користувача (таблиця cart_items). Реалізовано підтримку додавання кількох різних абонементів в один кошик.

Реалізація кошика

Кошик (handlers/cart.py) є універсальним – він може містити як абонементи, так і додаткові товари. При переході в розділ «Кошик» бот збирає всі позиції користувача, підраховує загальну суму, формує зрозуміле повідомлення зі списком товарів/абонементів і пропонує дві кнопки: «Оформити замовлення» та «Очистити кошик».

Процес оформлення замовлення

Після натискання «Оформити замовлення» запускається багатоступеневий процес:

1. Бот перевіряє наявність активного абонементу в кошику.
2. Пропонує два способи оплати: «Оплата в клубі» та «Оплата карткою онлайн».
3. При виборі «Оплата в клубі» генерується унікальний номер

замовлення, створюється запис у таблиці orders зі статусом «Очікує підтвердження», і клієнт отримує повідомлення з інструкцією звернутися до адміністратора клубу з цим номером.

4. При виборі онлайн-оплати бот переходить до введення даних картки (номер, термін дії, CVV) з подальшою імітацією обробки платежу (в реальній версії можна інтегрувати LiqPay, MonoPay або іншу платіжну систему).

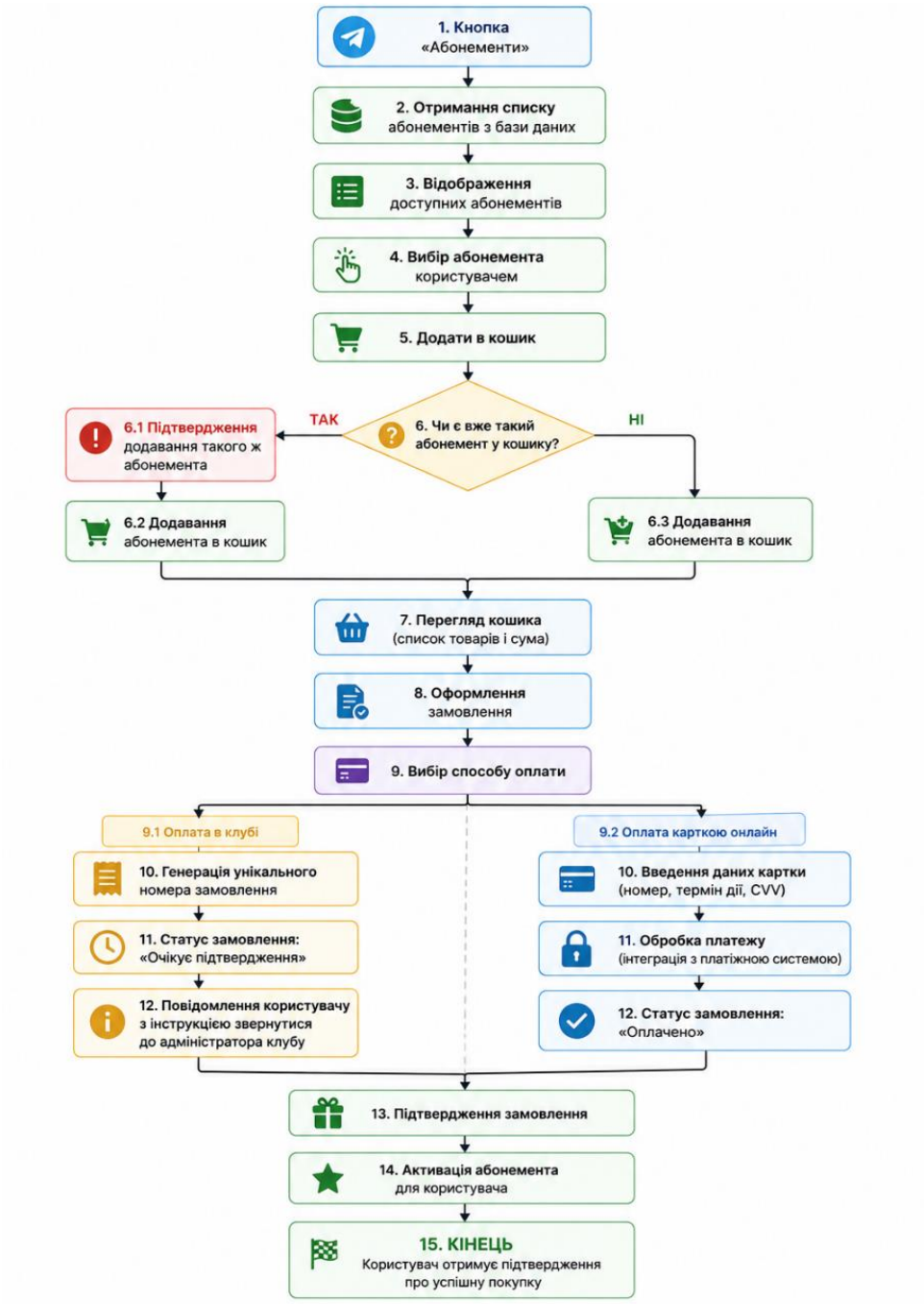



Рисунок 3.4 – Процес купівлі абонемента

Після оформлення замовлення статус змінюється в базі даних. Адміністратор через адміністративну панель може підтвердити оплату, після чого абонемент активується, а клієнт отримує повідомлення про успішну активацію.

Додаткові можливості модуля

– Автоматичне нагадування про закінчення абонементу за 7, 3 та 1 день.

– Можливість продовження діючого абонементу прямо з особистого кабінету.

– Заморозка абонементу на певний період (наприклад, під час відпустки).

– Історія всіх покупок клієнта.

Технічні особливості реалізації

Вся логіка модуля продажу абонементів захищена від помилок:

– Валідація даних на кожному кроці (Pydantic).

– Захист від дублювання абонементів.

– Обробка всіх можливих виняткових ситуацій (відсутність абонементів, помилка бази даних, таймаут тощо).

– Логування всіх дій користувача в цьому модулі.

Висновки до підрозділу 3.4

Модуль продажу абонементів був реалізований як повноцінна, зручна та надійна система, яка повністю автоматизує один з найважливіших бізнес-процесів спортивного клубу. Завдяки використанню Finite State Machine, чіткій структурі коду та продуманій обробці помилок користувач отримує інтуїтивно зрозумілий процес купівлі, а клуб – надійний інструмент контролю продажів і доходів.

Реалізована система дозволяє клієнтам купувати абонементи швидко та зручно в будь-який час, а адміністраторам – ефективно обробляти замовлення та контролювати фінансові потоки. Цей модуль став одним з ключових у всьому функціоналі бота LEO FITNESS.

## 3.5 Облік відвідувань та особистий кабінет

Облік відвідувань та особистий кабінет клієнта є одними з найважливіших функціональних модулів Telegram-бота LEO FITNESS. Ці два компоненти безпосередньо впливають на якість сервісу, дисципліну клієнтів, точність даних про завантаженість клубу та загальний рівень задоволеності відвідувачів.

### 3.5.1 Реалізація особистого кабінету

Особистий кабінет є центральним елементом взаємодії клієнта з ботом. Він дає можливість користувачу в будь-який момент отримати актуальну інформацію про свій статус у клубі без звернення до адміністратора. Модуль реалізований у файлі handlers/profile.py.

При натисканні кнопки «Профіль» у головному меню бот виконує наступні дії:

– Отримує дані користувача з таблиці users;

– Перевіряє наявність активного абонементу в таблиці subscriptions;

– Підраховує залишок днів дії абонементу та кількість доступних занять;

– Формує структуроване повідомлення з емодзі та чіткою інформацією.

У повідомленні відображається:

– Ім’я та прізвище клієнта;

– Email;

– Статус абонементу (активний / неактивний);

– Назва діючого абонементу;

– Залишок днів до закінчення;

– Залишок доступних занять;

– Коротка історія останніх відвідувань (останні 5 записів).

Кабінет оновлюється в реальному часі, тому інформація завжди актуальна. Якщо у клієнта немає активного абонементу, бот пропонує перейти до розділу «Абонементи» для покупки.

### 3.5.2 Реалізація системи обліку відвідувань

Система check-in є одним з ключових елементів автоматизації клубу. Вона дозволяє клієнту самостійно фіксувати факт приходу в зал, що значно зменшує навантаження на адміністратора на ресепшені та підвищує точність даних.

Модуль реалізований у файлі handlers/checkin.py. Логіка роботи побудована таким чином:

1. Користувач натискає кнопку «Мої відвідування» у головному меню.
2. Бот перевіряє наявність активного абонементу у таблиці subscriptions.
3. Якщо абонемент дійсний і є доступні заняття, бот пропонує підтвердити прихід кнопкою «Я прийшов на тренування».
4. Після підтвердження створюється новий запис у таблиці visits, списується одне заняття з абонементу, оновлюється дата останнього відвідування.
5. Клієнт отримує підтвердження з емодзі та інформацією про залишок занять.

Захист від зловживань

Для запобігання шахрайству була реалізована система захисту: – Check-in можливий лише один раз на календарний день; – Перевірка наявності

дійсного абонементу в момент запиту; – Логування кожної спроби check-in (успішної та невдалої); – Можливість блокування функції check-in для конкретного користувача адміністратором.

Технічна реалізація

Для керування процесом check-in використовується Finite State Machine. Стан CheckInState:waiting_confirmation дозволяє чітко контролювати послідовність дій і уникати помилок. Всі операції з базою даних виконуються в рамках транзакції, щоб уникнути ситуації, коли заняття списано, але запис про відвідування не створено.

Після успішного check-in бот автоматично оновлює інформацію в особистому кабінеті. Клієнт може в будь-який момент перейти в профіль і побачити актуальний залишок занять. Також реалізовано коротку історію відвідувань (останні 10 записів) з датами та кількістю списаних занять.

Додаткові можливості модуля

– Автоматичне попередження, якщо залишилося менше 5 занять;

– Можливість для адміністратора вручну додавати або скасовувати відвідування;

– Статистика відвідуваності для адміністратора (скільки людей було в клубі за день, тиждень, місяць);

– Експорт даних відвідувань у Excel.

Обробка виняткових ситуацій

Під час реалізації було приділено велику увагу обробці помилок:

– Спроба check-in без активного абонементу;

– Спроба повторного check-in в один день;

– Технічні помилки при роботі з базою даних;

– Таймаути та втрата з’єднання.

У всіх цих випадках користувач отримує зрозуміле повідомлення з поясненням проблеми та рекомендаціями щодо подальших дій.

Висновки до підрозділу 3.5

Модулі особистого кабінету та обліку відвідувань були реалізовані на

високому рівні. Вони повністю автоматизують важливі бізнес-процеси клубу, значно зменшують навантаження на адміністратора, підвищують дисципліну клієнтів і надають точну аналітичну інформацію. Завдяки продуманій логіці, захисту від зловживань та зручному інтерфейсу ці модулі стали одними з найбільш використовуваних і ефективних у всьому функціоналі бота LEO FITNESS.

Реалізовані рішення повністю відповідають вимогам, сформульованим на етапі проєктування, і готові до реального використання в спортивному клубі.

## 3.6 Адміністративна панель

Адміністративна панель є одним з найбільш важливих і складних у технічному плані модулів Telegram-бота LEO FITNESS. Вона призначена для керівника клубу та адміністраторів і дозволяє ефективно керувати всіма основними процесами: клієнтською базою, абонементами, статистикою, розсилками та моніторингом роботи бота. У цьому підрозділі детально розглянуто процес реалізації адміністративної панелі, її функціональні можливості, технічні рішення та особливості забезпечення безпеки.

Загальна концепція адміністративної панелі

Адміністративна панель реалізована як окремий модуль у папці admin/. Доступ до неї здійснюється через спеціальну команду /admin або кнопку в головному меню (для авторизованих адміністраторів). Для підвищення безпеки вхід у панель вимагає додаткової автентифікації – введення спеціального пароля або секретного коду.

Реалізація управління клієнтами

У розділі «Клієнти» адміністратор може:

– Переглянути повний список усіх зареєстрованих користувачів;

– Фільтрувати клієнтів за статусом абонементу, датою реєстрації, активністю;

– Переглядати детальну картку клієнта (персональні дані, історія покупок, історія відвідувань);

– Редагувати дані клієнта (ім’я, email, телефон, статус);

– Блокувати або розблоковувати акаунт клієнта;

– Ручно нараховувати або списувати заняття з абонементу.

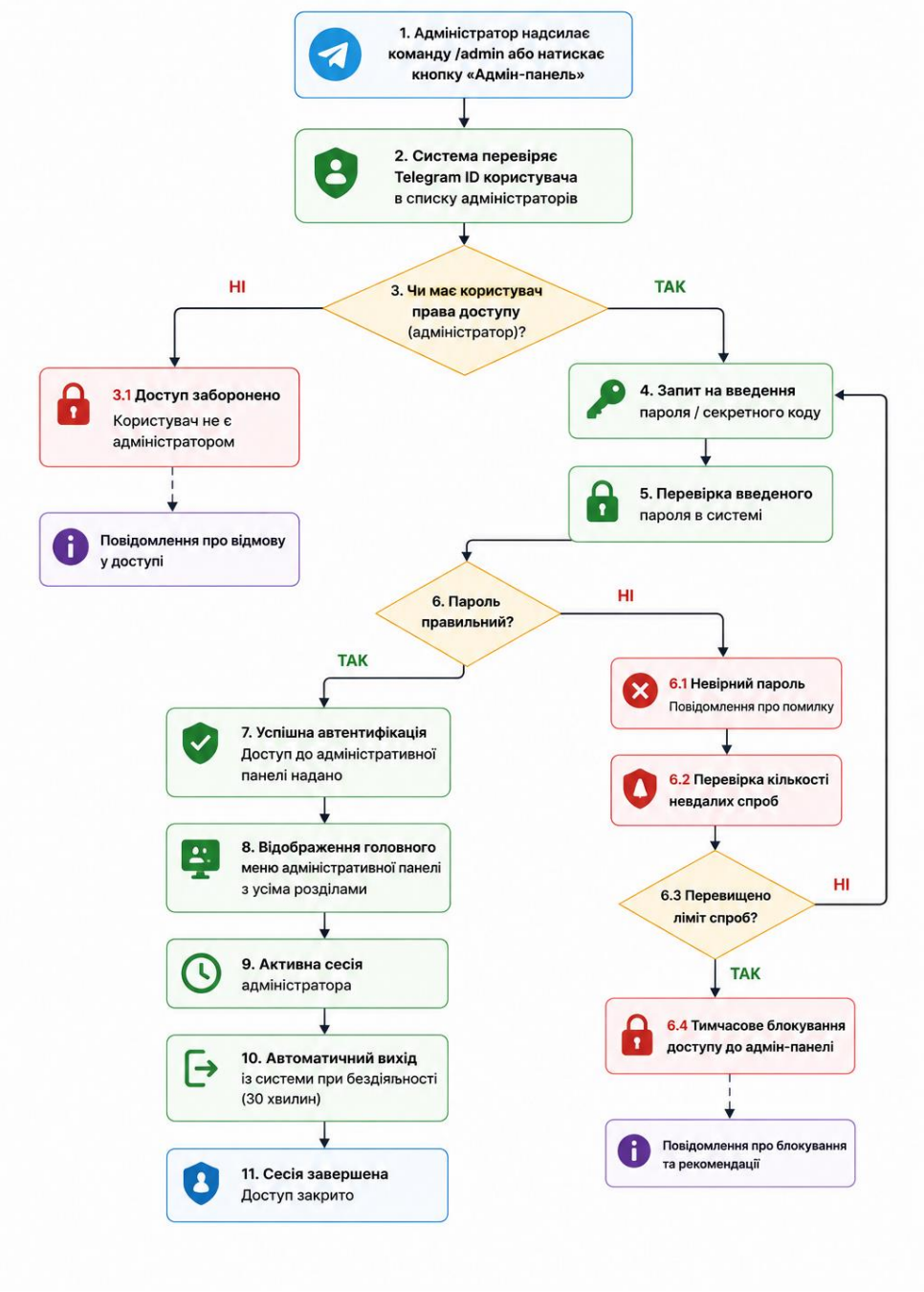



Рисунок 3.5 – Алгоритм автентифікації адміністратора в системі

Всі дії з даними клієнтів повністю логуються для забезпечення прозорості та контролю.

Реалізація управління абонементами

У цьому розділі адміністратор може:

– Переглядати всі активні та неактивні абонементи клієнтів;

– Продовжувати термін дії абонементу;

– Створювати нові типи абонементів;

– Деактивувати або видаляти абонементи;

– Переглядати статистику продажів абонементів за періоди.

Особливо важливою є можливість ручного підтвердження оплати в клубі – після оплати адміністратор змінює статус замовлення на «Оплачено», і абонемент автоматично активується.

Реалізація статистики

Розділ «Статистика» є одним з найбільш затребуваних. Тут адміністратор може отримати:

– Загальну кількість клієнтів;

– Кількість активних абонементів;

– Статистику продажів за день, тиждень, місяць;

– Завантаженість клубу по днях і годинах (на основі даних check-in);

– Популярність тренерів;

– Статистику продажу додаткових товарів.

Дані відображаються як у текстовому форматі, так і у вигляді простих таблиць. Також реалізована можливість експорту статистики в Excel.

Реалізація розсилок

Модуль розсилок дозволяє адміністратору:

– Надсилати повідомлення всім клієнтам;

– Надсилати повідомлення тільки клієнтам з активним абонементом;

– Надсилати персональні повідомлення конкретному клієнту;

– Надсилати нагадування про закінчення абонементу.

Розсилки виконуються асинхронно, щоб не блокувати роботу бота.

Технічна реалізація адміністративної панелі

Для захисту адміністративної панелі використовується окремий роутер і middleware, який перевіряє права доступу кожного користувача. Усі адміністративні команди обробляються в окремому модулі, що дозволяє чітко розділити логіку клієнтської та адміністративної частин.

Для зручності адміністратора панель максимально спрощена – використовуються великі кнопки, чіткі повідомлення та швидкі дії. Також реалізована система сесій – після 30 хвилин бездіяльності адміністратор автоматично виходить з панелі.

Безпека адміністративної панелі

Адміністративна панель захищена кількома рівнями:

– Двоступенева автентифікація;

– Обмеження кількості спроб входу;

– Логування всіх дій адміністратора (хто, коли, яку дію виконав);

– Можливість тимчасового блокування акаунту адміністратора;

– Захист від несанкціонованого доступу через Telegram ID.

Висновки до підрозділу 3.6

Адміністративна панель була реалізована як повноцінний, зручний і захищений інструмент управління спортивним клубом. Вона дозволяє адміністраторам та керівництву ефективно контролювати всі ключові процеси: роботу з клієнтами, продажі, статистику, розсилки та моніторинг системи.

Завдяки продуманій структурі, чіткому розділенню прав доступу та надійній системі логування адміністративна панель забезпечує високу безпеку та зручність використання. Реалізований модуль значно спрощує роботу персоналу клубу, підвищує контроль над бізнес-процесами та є важливою складовою всього Telegram-бота LEO FITNESS.

## 3.7 Інтеграція з базою даних SQLite

Інтеграція з базою даних є одним з найбільш відповідальних етапів реалізації будь-якого програмного продукту. Від якості цієї інтеграції залежить швидкість роботи бота, надійність зберігання даних, безпека інформації та зручність подальшої підтримки системи. У Telegram-боті LEO FITNESS для зберігання всіх даних використовується вбудована реляційна база даних SQLite3. У цьому підрозділі детально розглянуто процес інтеграції, структуру бази даних, реалізацію основних операцій, механізми захисту даних та особливості оптимізації.

Вибір SQLite3 був обґрунтований специфікою проєкту. Для невеликого спортивного клубу з кількістю активних клієнтів до 1000–1500 осіб використання потужних серверних баз даних (PostgreSQL, MySQL) є економічно недоцільним і технічно надмірним. SQLite3 повністю задовольняє потреби проєкту завдяки таким перевагам:

– Не потребує окремого сервера бази даних;

– Працює у вигляді одного файлу (leo_fitness.db);

– Має високу швидкість роботи з невеликими та середніми обсягами даних;

– Підтримує ACID-транзакції;

– Легко створює резервні копії (достатньо скопіювати один файл);

– Повністю інтегрована в Python через стандартний модуль sqlite3.

База даних складається з 9 основних таблиць, спроектованих з дотриманням третьої нормальної форми (3NF):

1. users – основна інформація про клієнтів
2. subscription_types – типи абонементів
3. subscriptions – активні абонементи користувачів
4. orders – оформлені замовлення
5. order_items – позиції в замовленнях
6. visits – записи про відвідування
7. trainers – інформація про тренерів
8. products – товари для продажу

9. product_categories – категорії товарів

Кожна таблиця має первинні ключі, зовнішні ключі, індекси на часто використовуваних полях та обмеження (NOT NULL, UNIQUE). Це забезпечує цілісність даних і високу швидкість виконання запитів.

Реалізація модуля роботи з базою даних

Вся взаємодія з базою даних винесена в окремий модуль database/db.py. Це дозволяє ізолювати SQL-запити від бізнес-логіки і спростити подальшу підтримку. У модулі реалізовані такі основні функції:

– init_db() – створення всіх таблиць при першому запуску;

– add_user() – додавання нового клієнта;

– get_user_by_telegram_id() – пошук користувача;

– create_subscription() – активація абонементу;

– add_visit() – фіксація відвідування з автоматичним списанням заняття;

– get_active_subscription() – отримання актуального абонементу;

– add_to_cart() – додавання позиції в кошик;

– get_cart_items() – отримання вмісту кошика;

– create_order() – створення замовлення.

Всі функції використовують параметризовані запити (? або named parameters), що повністю захищає систему від SQL-ін’єкцій.

Використання транзакцій

Для критичних операцій (наприклад, check-in або оформлення замовлення) використовуються транзакції. Це гарантує, що або вся операція виконується успішно, або жодна зміна не застосовується. Наприклад, при check-in одночасно відбувається списання заняття з абонементу та створення запису про відвідування. Якщо одна з операцій не вдається, транзакція відкатується.

Оптимізація запитів

Для підвищення швидкості роботи були створені індекси на таких полях: – telegram_id у таблиці users; – user_id у таблицях subscriptions, visits, orders; – subscription_type_id у таблиці subscriptions; – order_id у таблиці order_items.

Також активно використовуються JOIN-запити для отримання повної інформації (наприклад, дані клієнта + його активний абонемент) за один запит до бази.

Резервне копіювання бази даних

Для підвищення надійності реалізована система автоматичного резервного копіювання. При кожному запуску бота створюється копія файлу leo_fitness.db з міткою дати і часу. Крім того, раз на добу виконується повне резервне копіювання на зовнішнє сховище. Це дозволяє швидко відновити систему у разі пошкодження файлу бази даних.

Моніторинг стану бази даних

У адміністративній панелі реалізована функція перегляду основних показників бази даних: кількість записів у кожній таблиці, розмір файлу бази, дата останнього резервного копіювання. Це дозволяє адміністратору оперативно контролювати стан системи.

Висновки до підрозділу 3.7

Інтеграція з базою даних SQLite3 була виконана на високому професійному рівні. Було створено ефективну, надійну та оптимізовану систему зберігання даних, яка повністю відповідає потребам спортивного клубу. Використання параметризованих запитів, транзакцій, індексів та чіткої архітектури модуля database/ забезпечило високу швидкість роботи, захист від поширених атак та зручність підтримки.

Реалізована інтеграція стала надійним фундаментом для роботи всіх інших модулів бота – від реєстрації та продажу абонементів до check-in і адміністративної панелі. Завдяки правильному проектуванню та реалізації бази даних система LEO FITNESS демонструє стабільну роботу, швидку обробку запитів і високу надійність зберігання даних.

## 3.8 Висновки

У третьому розділі дипломної роботи була виконана практична реалізація Telegram-бота LEO FITNESS. Було створено повноцінний, функціональний і стабільний програмний продукт, який автоматизує ключові бізнес-процеси спортивного клубу.

Під час реалізації вдалося успішно впровадити весь запланований функціонал: швидку реєстрацію та авторизацію клієнтів, зручну систему продажу абонементів з кошиком і двома способами оплати, особистий кабінет з актуальною інформацією про абонемент, систему check-in для фіксації відвідувань, розділ з інформацією про тренерів, продаж додаткових товарів, а також повноцінну адміністративну панель для управління клубом.

Код бота написаний з дотриманням принципів модульності, асинхронного програмування та чистої архітектури. Використання Finite State Machine дозволило ефективно керувати складними багатоступеневими процесами, а чітке розділення функціоналу на окремі модулі значно полегшило розробку, тестування та підтримку системи.

Розроблений Telegram-бот LEO FITNESS є готовим до практичного використання рішенням. Він повністю відповідає вимогам, сформульованим на етапі проєктування, враховує специфіку роботи невеликого спортивного клубу та значно підвищує якість обслуговування клієнтів, оптимізує роботу адміністраторів і дає можливість отримувати оперативну аналітику.

Реалізований програмний продукт демонструє ефективне поєднання сучасних технологій з реальними потребами бізнесу. Він може бути безпосередньо впроваджений у діяльність спортивного клубу LEO FITNESS і служити основою для подальшого розвитку системи.

## Висновки

У рамках дипломної роботи була успішно розроблена та реалізована система Telegram-бота LEO FITNESS для продажу абонементів і обліку відвідувань у спортивному клубі.

Метою роботи була розробка функціонального Telegram-бота, який автоматизує ключові бізнес-процеси спортивного клубу. Поставлена мета була повністю досягнута. У результаті виконання роботи створено робочий програмний продукт, який включає в себе реєстрацію та авторизацію клієнтів, зручну систему продажу абонементів з кошиком і підтримкою двох способів оплати, особистий кабінет клієнта, систему check-in відвідувань, інформацію про тренерів, продаж додаткових товарів та повноцінну адміністративну панель для управління клубом.

У першому розділі було проведено детальний аналіз предметної області, вивчено проблеми сучасних спортивних клубів та існуючі програмні рішення. Було обґрунтовано актуальність розробки саме Telegram-бота для малого та середнього бізнесу.

У другому розділі виконано комплексне проєктування системи: сформульовано функціональні та нефункціональні вимоги, розроблено архітектуру програмного забезпечення, спроектовано базу даних, змодельовано основні бізнес-процеси та інтерфейс користувача, а також створено комплекс заходів щодо забезпечення безпеки даних.

У третьому розділі проведено практичну реалізацію бота. Використано сучасний технологічний стек: Python 3.12, aiogram 3.x, SQLite3 та асинхронне програмування. Код має чітку модульну структуру, добре задокументований і готовий до подальшої підтримки та розвитку.

Розроблений бот повністю відповідає сучасним вимогам до зручності, швидкодії, безпеки та функціональності. Він дозволяє клієнтам швидко купувати абонементи, контролювати свій статус у клубі та фіксувати відвідування, а адміністраторам – ефективно керувати бізнес-процесами та отримувати оперативну аналітику.

Практичне значення роботи полягає в тому, що створений Telegram-бот може бути безпосередньо впроваджений у діяльність спортивного клубу LEO FITNESS. Це дозволить оптимізувати роботу адміністраторів, зменшити кількість помилок, підвищити якість обслуговування клієнтів та збільшити загальну ефективність роботи клубу.

Наукова новизна полягає в комплексній реалізації спеціалізованого Telegram-бота, який поєднує продаж абонементів, онлайн-оплату, особистий кабінет та систему обліку відвідувань в єдиному зручному інтерфейсі, адаптованому саме для потреб невеликих спортивних клубів.

У ході роботи були вирішені всі поставлені задачі. Розроблений програмний продукт демонструє ефективне поєднання сучасних технологій з реальними потребами бізнесу і може слугувати прикладом успішної автоматизації сервісного підприємства.

Подальшим напрямком розвитку може стати розширення функціоналу бота (інтеграція з турнікетами, система лояльності, онлайн-запис на персональні тренування, аналітичний модуль з візуалізацією даних), а також адаптація рішення для мережі спортивних клубів.

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ


1. Анохін В. М. Проєктування інформаційних систем : навч. посіб. / В. М. Анохін, О. В. Пилипенко. – Київ : КНУ, 2023. – 348 с.
2. Official Python Documentation [Електронний ресурс]. – Режим доступу: https://docs.python.org/3/ (дата звернення: 05.05.2026).
3. aiogram 3.x Documentation [Електронний ресурс]. – Режим доступу: https://docs.aiogram.dev/ (дата звернення: 05.05.2026).
4. Telegram Bot API Documentation [Електронний ресурс]. – Режим доступу: https://core.telegram.org/bots/api (дата звернення: 05.05.2026).
5. Мартин Р. Чистий код : створення, аналіз та рефакторинг. – Львів : Видавництво «Апріорі», 2022. – 464 с.
6. Фаулер М. Рефакторинг. Поліпшення проєкту існуючого коду / М. Фаулер. – 2-ге вид. – Київ : Видавнича група BHV, 2021. – 448 с.
7. Клінтон С. Python для мережевих інженерів / С. Клінтон. – Київ : Діалектика, 2023. – 512 с.
8. Васильєв О. В. Асинхронне програмування на Python : навч. посіб. – Харків : ХНУРЄ, 2024. – 215 с.
9. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні вимоги та правила складання. – Київ : Держстандарт України, 2015. – 28 с.
10. Закон України «Про захист персональних даних» від 01.06.2010 № 2297-VI // Відомості Верховної Ради України. – 2010. – № 34. – Ст. 481.
11. Книга «Telegram Bots: Design and Development» / A. Shcherbakov. – Packt Publishing, 2022.
12. Книга «Building Telegram Bots» / A. Horev. – Apress, 2021.
13. Офіційна документація SQLite [Електронний ресурс]. – Режим доступу: https://www.sqlite.org/docs.html (дата звернення: 05.05.2026).

14. Пилипенко О. В. Автоматизація бізнес-процесів підприємств сфери послуг : монографія. – Київ : КНЕУ, 2023. – 312 с.

15. Бізнес-процеси сучасних фітнес-клубів: проблеми та шляхи автоматизації / В. О. Кравченко, І. С. Петренко // Економіка та управління. – 2024. – № 3. – С. 45–53.

16. Розробка чат-ботів для малого бізнесу / А. В. Сидоренко // Інформаційні технології в освіті та бізнесі. – 2025. – № 1. – С. 112–120.

17. Git Documentation [Електронний ресурс]. – Режим доступу: https://git-scm.com/doc (дата звернення: 05.05.2026).

18. Visual Studio Code Documentation [Електронний ресурс]. – Режим доступу: https://code.visualstudio.com/docs (дата звернення: 05.05.2026).

19. Loguru Documentation [Електронний ресурс]. – Режим доступу: https://loguru.readthedocs.io/ (дата звернення: 05.05.2026).

20. Pydantic Documentation [Електронний ресурс]. – Режим доступу: https://docs.pydantic.dev/ (дата звернення: 05.05.2026).

21. Кравець О. М. Цифрова трансформація малого бізнесу в Україні : монографія. – Львів : ЛНУ, 2024. – 276 с.

22. Сучасні тенденції розвитку фітнес-індустрії в Україні / О. І. Мельник // Спорт і здоров’я нації. – 2025. – № 2. – С. 78–85.

23. Martin Fowler. Patterns of Enterprise Application Architecture. – Addison-Wesley, 2002.

24. Eric Evans. Domain-Driven Design. – Addison-Wesley, 2003.

25. Clean Architecture: A Craftsman’s Guide to Software Structure and Design / Robert C. Martin. – Pearson, 2017.

26. Офіційний сайт aiogram [Електронний ресурс]. – Режим доступу: https://aiogram.dev/ (дата звернення: 05.05.2026).

27. Best practices for Telegram Bot development [Електронний ресурс]. – Режим доступу: https://core.telegram.org/bots (дата звернення: 05.05.2026).

28. Книга «Python Crash Course» / Eric Matthes. – 3rd Edition. – No Starch Press, 2023.

# ДОДАТОК А

(обов'язковий)

## ГРАФІЧНІ МАТЕРІАЛИ

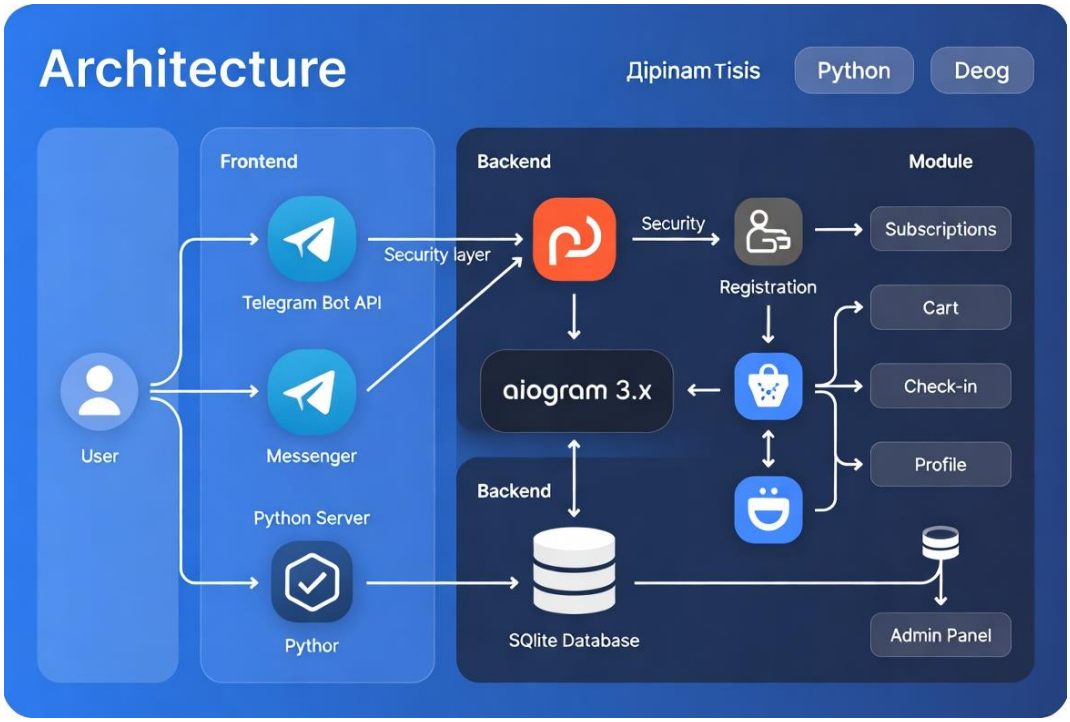



Рисунок А.1 – Загальна архітектура Telegram-бота LEO FITNESS

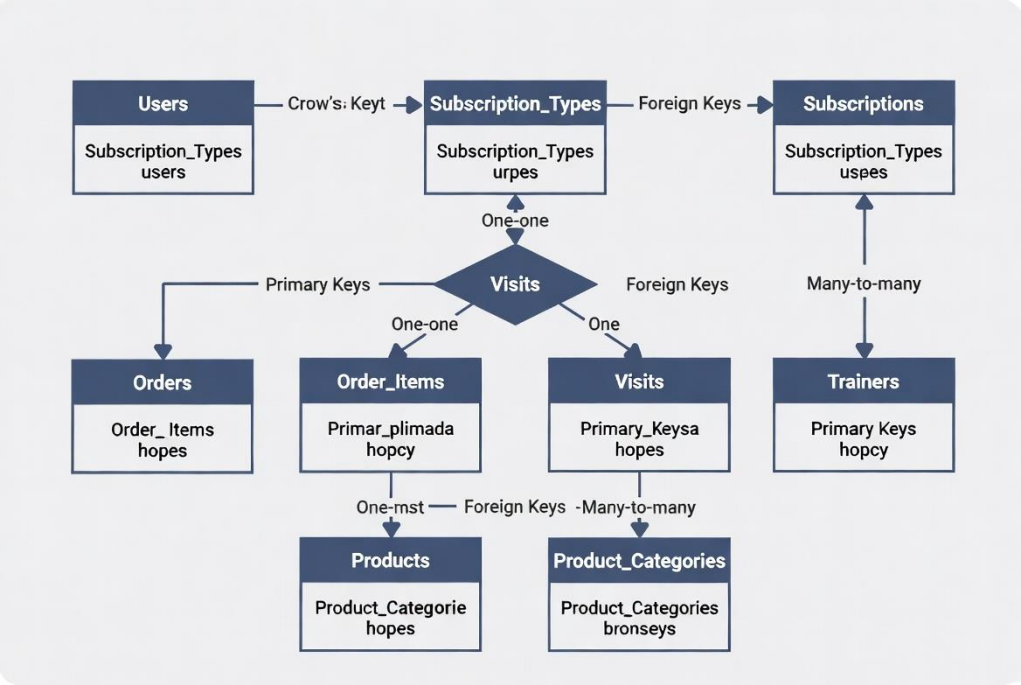



Рисунок А.2 – ER-діаграма бази даних Telegram-бота LEO FITNESS

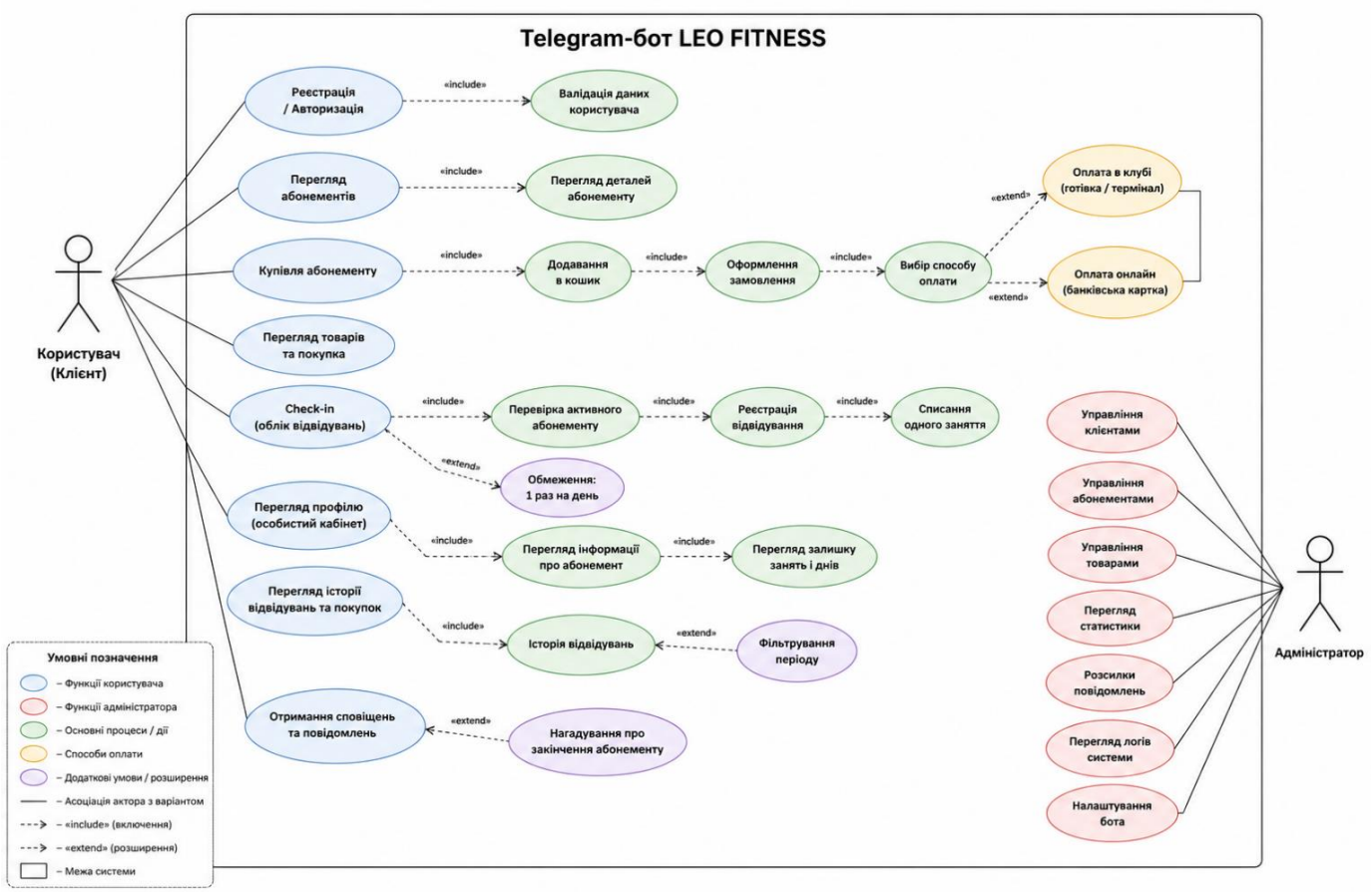



Рисунок А.3 – Діаграма варіантів використання

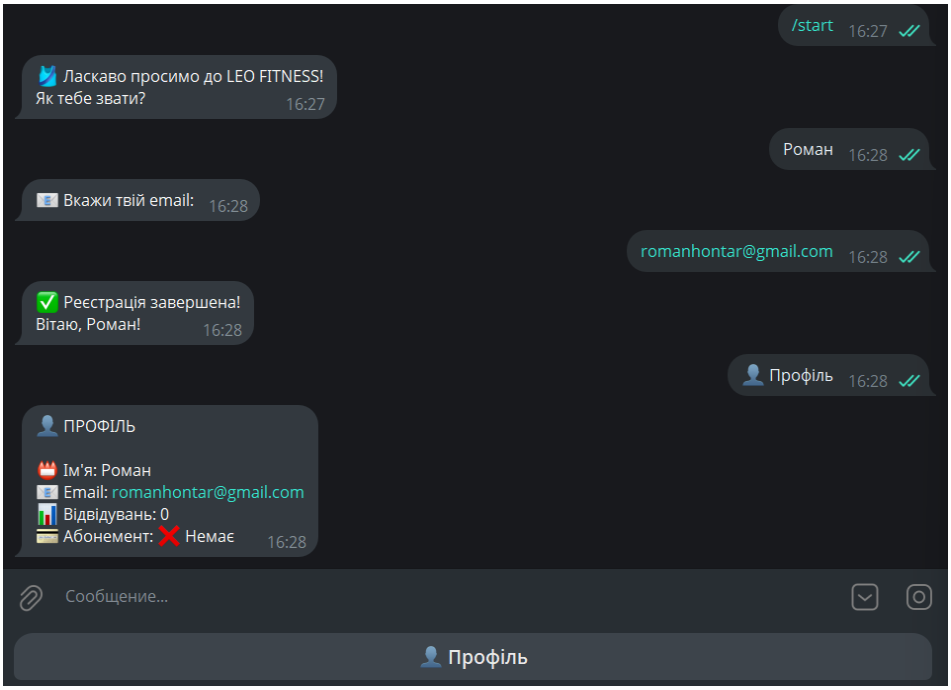



Рисунок А.4 – Інтерфейс реєстрації користувача та особистий профіль

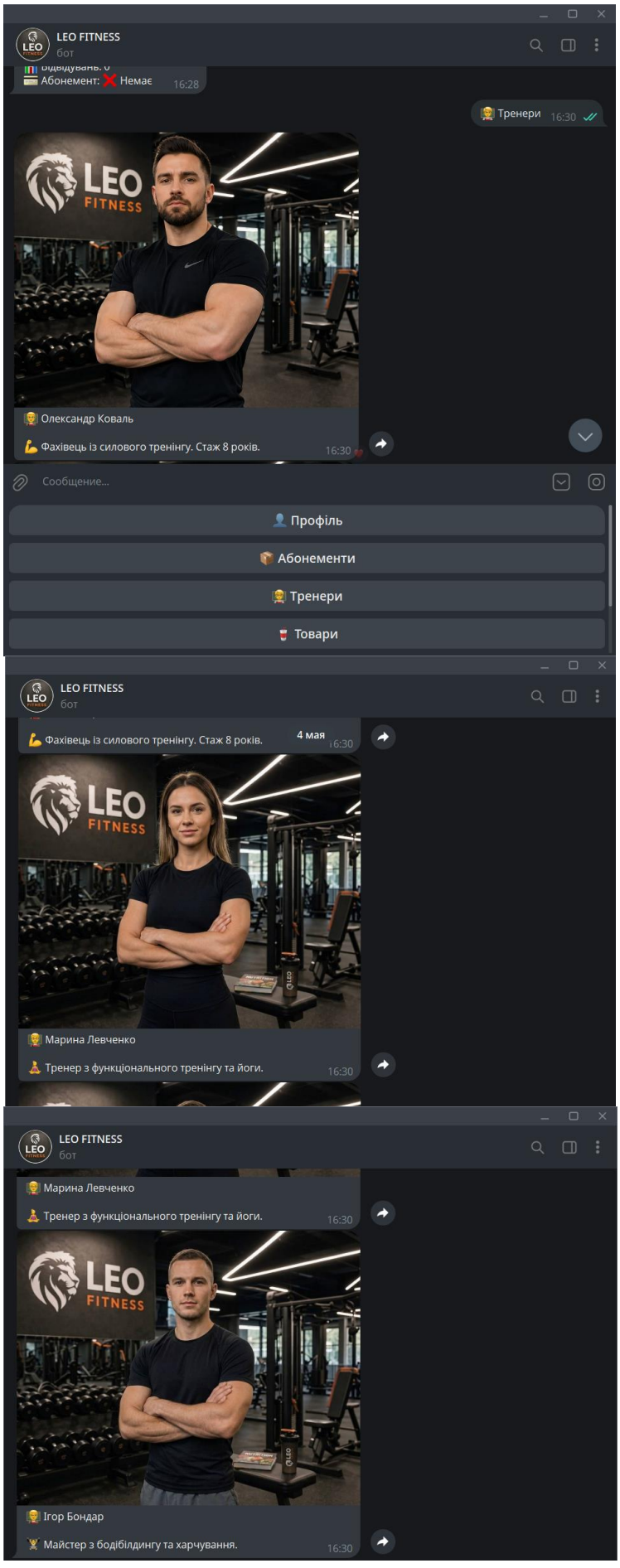



Рисунок А.5 – Інтерфейс розділу «Тренери» у Telegram-боті

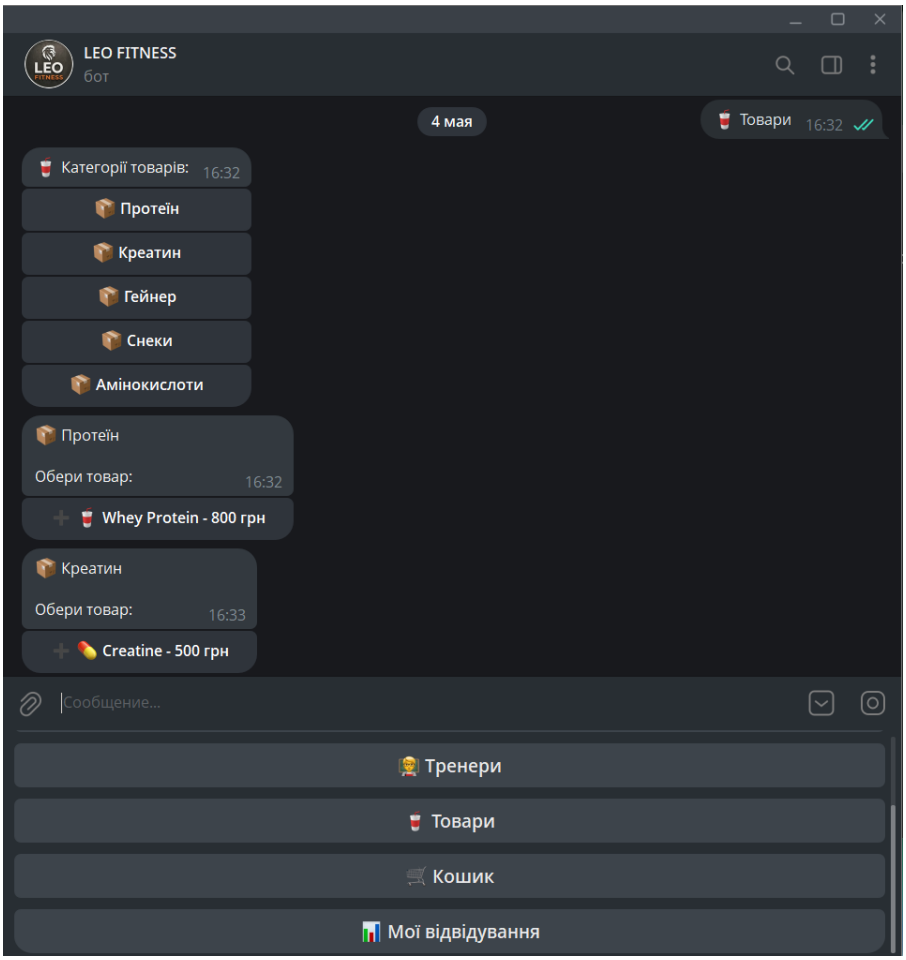



Рисунок А.5 – Інтерфейс розділу «Товари» у Telegram-боті

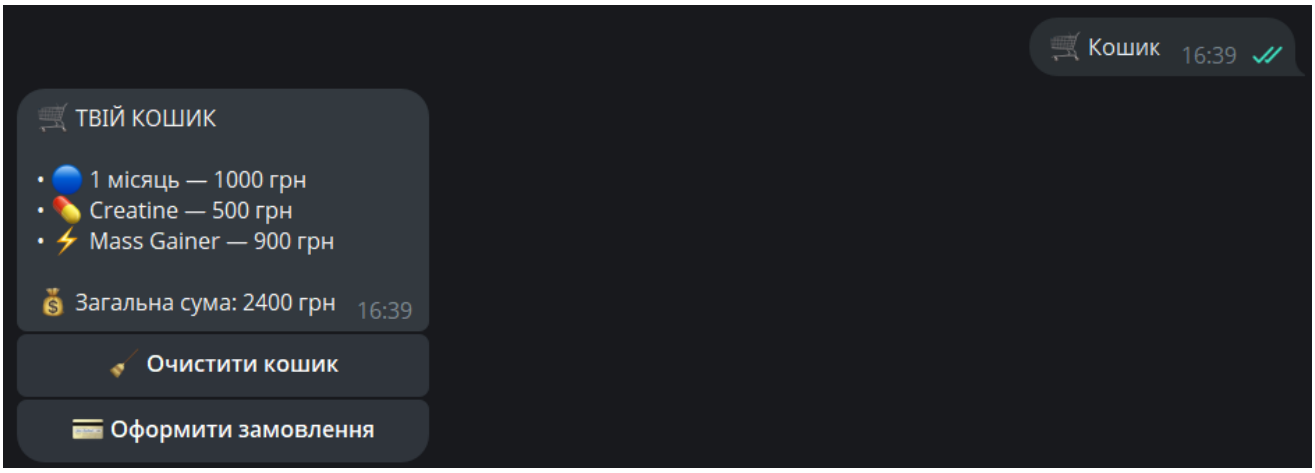



Рисунок А. 6 – Інтерфейс розділу «Кошик» у Telegram-боті

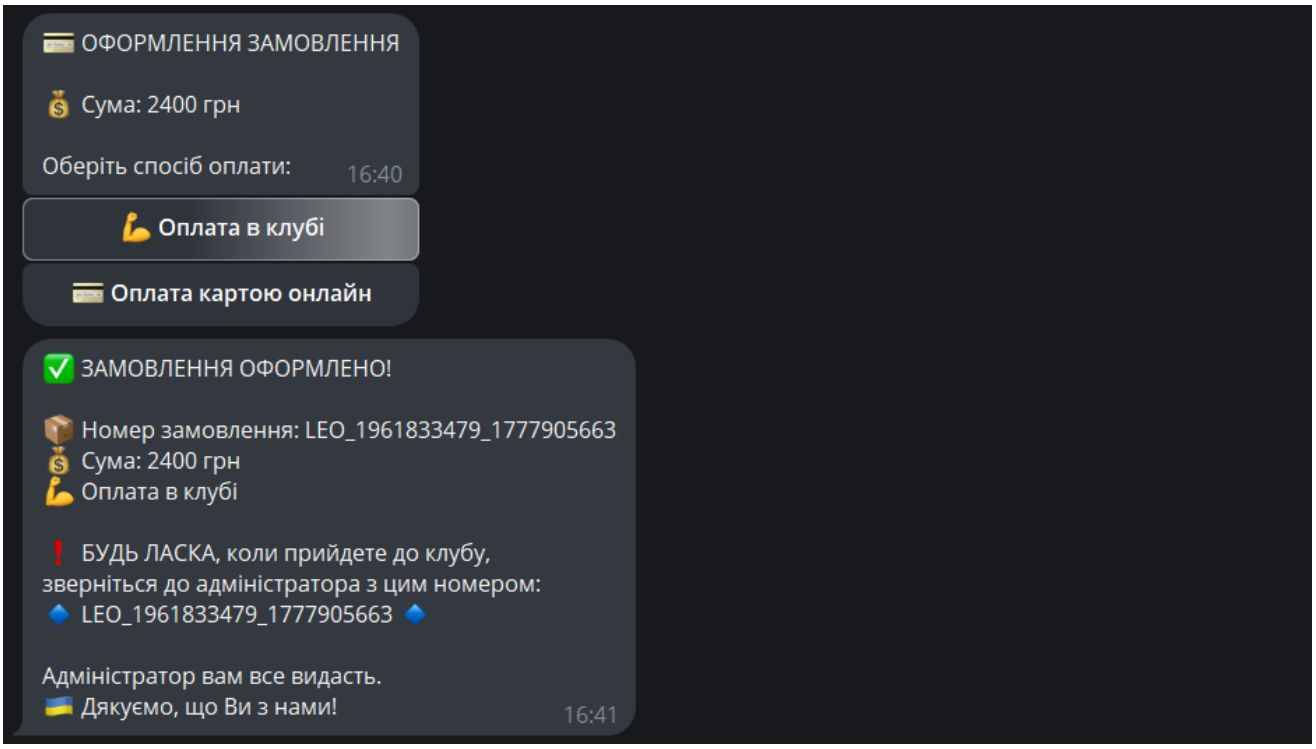



Рисунок А. 7 – Процес оформлення замовлення з оплатою в клубі

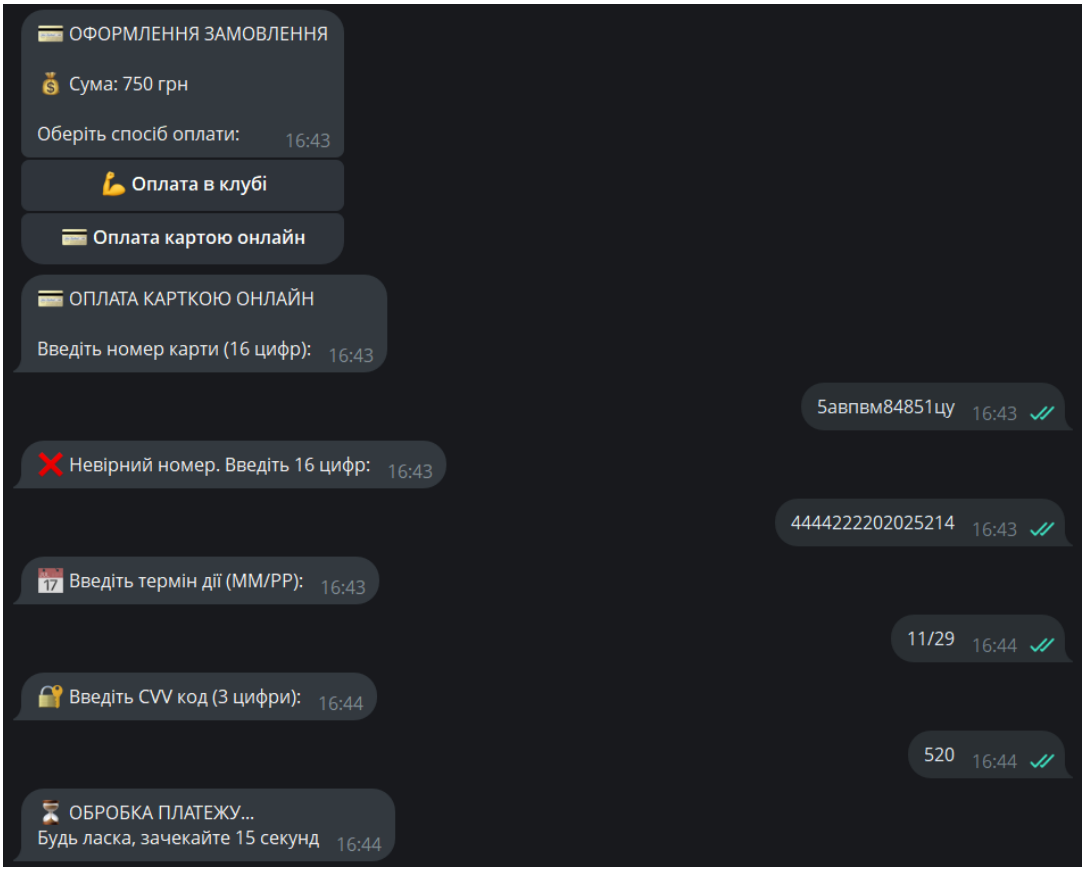



Рисунок А. 8 – Процес оформлення замовлення з оплатою онлайн

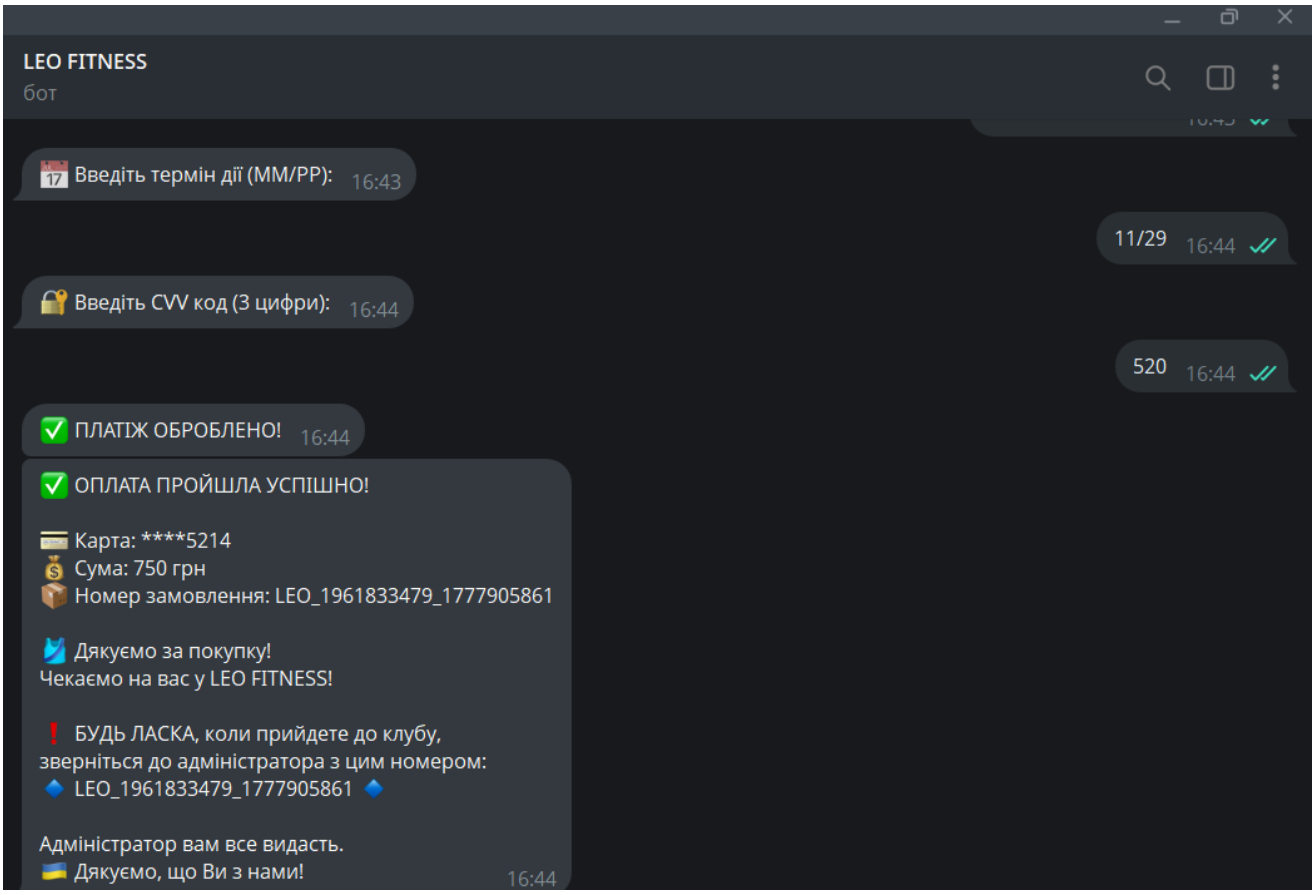



Рисунок А. 9 – Приклад успішного оформлення замовлення в боті LEO FITNESS


Рисунок А. 10 – Перегляд інформації про активний абонемент в особистому профілі

# ДОДАТОК Б

(обов'язковий)

## ПРОГРАМНИЙ КОД ОСНОВНИХ МОДУЛІВ

```
import asyncio
import sqlite3
import os
import json
from datetime import datetime, timedelta
from aiogram import Bot, Dispatcher, types, F
from aiogram.filters import Command
from aiogram.types import ReplyKeyboardMarkup, KeyboardButton, InlineKeyboardMarkup, InlineKeyboardButton,
    FSInputFile
from aiogram.fsm.storage.memory import MemoryStorage
from aiogram.fsm.state import State, StatesGroup
from aiogram.fsm.context import FSMContext

TOKEN = "8618600925:AAH6Eq1i39jNQLXsck4Cf2NrNx_f5mu0g78"

def init_db():
    conn = sqlite3.connect('leo_fitness.db')
    cur = conn.cursor()

    cur.execute('''
        CREATE TABLE IF NOT EXISTS users (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            telegram_id INTEGER UNIQUE,
            name TEXT,
            email TEXT,
            visits INTEGER DEFAULT 0,
            active_subscription TEXT DEFAULT NULL,
            subscription_start_date TEXT DEFAULT NULL,
            subscription_end_date TEXT DEFAULT NULL,
            registered_at TEXT
        )
''')

    cur.execute('''
        CREATE TABLE IF NOT EXISTS subscriptions (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            name TEXT,
            price INTEGER,
            duration_days INTEGER
        )
```

```
''')

    cur.execute('''
        CREATE TABLE IF NOT EXISTS trainers (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            name TEXT,
            photo TEXT,
            description TEXT
        )
    ''')

    cur.execute('''
        CREATE TABLE IF NOT EXISTS products (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            name TEXT,
            price INTEGER,
            type TEXT
        )
    ''')

    cur.execute('''
        CREATE TABLE IF NOT EXISTS cart (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            telegram_id INTEGER,
            item_type TEXT,
            item_id INTEGER,
            item_name TEXT,
price INTEGER,
            added_at TEXT
        )
    ''')

    cur.execute('''
        CREATE TABLE IF NOT EXISTS orders (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            order_id TEXT UNIQUE,
            telegram_id INTEGER,
            items TEXT,
            total_price INTEGER,
            payment_method TEXT,
            status TEXT,
            payment_status TEXT,
            card_last4 TEXT,
            created_at TEXT
        )
    ''')
```

```
    cur.execute('''
      CREATE TABLE IF NOT EXISTS visit_log (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        telegram_id INTEGER,
        visit_date TEXT
      )
    ''')

    cur.execute("SELECT COUNT(*) FROM subscriptions")
    if cur.fetchone()[0] == 0:
      subscriptions_data = [
        ("🎁 Пробне тренування", 0, 1),
        ("🟢 1 місяць", 1000, 30),
        ("🔥 12 місяців", 10000, 365)
      ]
      for name, price, days in subscriptions_data:
        cur.execute("INSERT INTO subscriptions (name, price, duration_days) VALUES (?, ?, ?)", (name, price, days))

    cur.execute("SELECT COUNT(*) FROM trainers")
    if cur.fetchone()[0] == 0:
      trainers_data = [
        ("Олександр Коваль", "photo/firsttrainer.jpg", "💪 Фахівець із силового тренінгу. Стаж 8 років."),
        ("Марина Левченко", "photo/secondtrainer.jpg", "🧘‍♀️ Тренер з функціонального тренінгу та йоги."),
        ("Ігор Бондар", "photo/thirdtrainer.jpg", "🏋️ Майстер з бодібілдингу та харчування.")
      ]
      for name, photo, desc in trainers_data:
        cur.execute("INSERT INTO trainers (name, photo, description) VALUES (?, ?, ?)", (name, photo, desc))

    cur.execute("SELECT COUNT(*) FROM products")
    if cur.fetchone()[0] == 0:
      products_data = [
        ("🥛 Whey Protein", 800, "Протеїн"),
        ("💊 Creatine", 500, "Креатин"),
        ("⚡ Mass Gainer", 900, "Гейнер"),
        ("🍫 Протеїновий батончик", 150, "Снеки"),
        ("🧪 BCAA", 600, "Амінокислоти")
      ]
      for name, price, ptype in products_data:
        cur.execute("INSERT INTO products (name, price, type) VALUES (?, ?, ?)", (name, price, ptype))

    conn.commit()
    conn.close()

def main_menu_keyboard():
    buttons = [
      [KeyboardButton(text="👤 Профіль")],
```

```
      [KeyboardButton(text="🎫 Абонементи")],
      [KeyboardButton(text="🏋️ Тренери")],
      [KeyboardButton(text="🛍️ Товари")],
      [KeyboardButton(text="🛒 Кошик")],
      [KeyboardButton(text="📊 Мої відвідування")]
    ]
    return ReplyKeyboardMarkup(keyboard=buttons, resize_keyboard=True)

class RegisterState(StatesGroup):
    waiting_for_name = State()
    waiting_for_email = State()

class CardPaymentState(StatesGroup):
    waiting_for_card_number = State()
    waiting_for_expiry = State()
    waiting_for_cvv = State()

bot = Bot(token=TOKEN)
storage = MemoryStorage()
dp = Dispatcher(storage=storage)

def add_to_cart(telegram_id, item_type, item_id, item_name, price):
    conn = sqlite3.connect('leo_fitness.db')
    cur = conn.cursor()
    cur.execute('''
        INSERT INTO cart (telegram_id, item_type, item_id, item_name, price, added_at)
        VALUES (?, ?, ?, ?, ?, ?)
    ''', (telegram_id, item_type, item_id, item_name, price, datetime.now().isoformat()))
    conn.commit()
    conn.close()

def get_cart(telegram_id):
    conn = sqlite3.connect('leo_fitness.db')
    cur = conn.cursor()
    cur.execute('SELECT id, item_name, price FROM cart WHERE telegram_id = ?', (telegram_id,))
      items = cur.fetchall()
    conn.close()
    return items

def clear_cart(telegram_id):
    conn = sqlite3.connect('leo_fitness.db')
    cur = conn.cursor()
    cur.execute('DELETE FROM cart WHERE telegram_id = ?', (telegram_id,))
    conn.commit()
    conn.close()

def get_cart_total(telegram_id):
```

```
    items = get_cart(telegram_id)
    return sum(item[2] for item in items)

def get_subscription_days_left(telegram_id):
    conn = sqlite3.connect('leo_fitness.db')
    cur = conn.cursor()
    cur.execute("SELECT subscription_end_date FROM users WHERE telegram_id = ?", (telegram_id,))
    result = cur.fetchone()
    conn.close()

    if result and result[0]:
        end_date = datetime.fromisoformat(result[0])
        days_left = (end_date - datetime.now()).days
        return max(0, days_left)
    return 0

def activate_subscription(telegram_id, subscription_name, duration_days):
    conn = sqlite3.connect('leo_fitness.db')
    cur = conn.cursor()

    start_date = datetime.now()
    end_date = start_date + timedelta(days=duration_days)
      cur.execute('''
        UPDATE users
        SET active_subscription = ?, subscription_start_date = ?, subscription_end_date = ?
        WHERE telegram_id = ?
    ''', (subscription_name, start_date.isoformat(), end_date.isoformat(), telegram_id))

    conn.commit()
    conn.close()

@dp.message(Command("start"))
async def cmd_start(message: types.Message, state: FSMContext):
    conn = sqlite3.connect('leo_fitness.db')
    cur = conn.cursor()
    cur.execute("SELECT * FROM users WHERE telegram_id = ?", (message.from_user.id,))
    user = cur.fetchone()
    conn.close()

    if user:
        await message.answer(f"Вітаю назад, {user[2]}! 🎉", reply_markup=main_menu_keyboard())
    else:
        await message.answer("🏋️ Ласкаво просимо до LEO FITNESS!\nЯк тебе звати?")
        await state.set_state(RegisterState.waiting_for_name)

@dp.message(RegisterState.waiting_for_name)
async def process_name(message: types.Message, state: FSMContext):
```

```
    await state.update_data(name=message.text)
    await message.answer("✉ Вкажи твій email:")
    await state.set_state(RegisterState.waiting_for_email)

@dp.message(RegisterState.waiting_for_email)
async def process_email(message: types.Message, state: FSMContext):
    user_data = await state.get_data()
    name = user_data['name']
    email = message.text

    conn = sqlite3.connect('leo_fitness.db')
    cur = conn.cursor()
    cur.execute(
        "INSERT INTO users (telegram_id, name, email, visits, registered_at) VALUES (?, ?, ?, ?, ?)",
        (message.from_user.id, name, email, 0, datetime.now().isoformat())
    )
    conn.commit()
    conn.close()

    await message.answer(f"✅ Реєстрація завершена!\nВітаю, {name}!", reply_markup=main_menu_keyboard())
    await state.clear()

@dp.message(F.text == "👤 Профіль")
async def show_profile(message: types.Message):
    conn = sqlite3.connect('leo_fitness.db')
    cur = conn.cursor()
    cur.execute("SELECT name, email, visits, active_subscription, subscription_end_date FROM users WHERE telegram_id =
      ?", (message.from_user.id,))
    user = cur.fetchone()
    conn.close()

    if user:
        name, email, visits, sub, end_date = user
        sub_text = sub if sub else "❌ Немає"

        days_left_text = ""
        if end_date and sub:
            try:
                days_left = (datetime.fromisoformat(end_date) - datetime.now()).days
                if days_left > 0:
                    days_left_text = f"\n📆 Залишилось днів: {days_left}"
                elif days_left == 0:
                        days_left_text = f"\n⚠ Абонемент закінчується СЬОГОДНІ!"
                else:
                    days_left_text = f"\n❌ Абонемент закінчився"
            except:
                pass
```

```
    await message.answer(
        f"👤 ПРОФІЛЬ\n\n"
        f"😎 Ім'я: {name}\n"
        f"📧 Email: {email}\n"
        f"📊 Відвідувань: {visits}\n"
        f"💳 Абонемент: {sub_text}{days_left_text}"
    )

@dp.message(F.text == "📋 Абонементи")
async def show_subscriptions(message: types.Message):
    conn = sqlite3.connect('leo_fitness.db')
    cur = conn.cursor()
    cur.execute("SELECT id, name, price FROM subscriptions")
    subs = cur.fetchall()
    conn.close()

    keyboard = InlineKeyboardMarkup(inline_keyboard=[])
    for sub_id, name, price in subs:
        keyboard.inline_keyboard.append([
            InlineKeyboardButton(text=f"{name} - {price} грн", callback_data=f"add_sub_{sub_id}")
        ])
    await message.answer("🛒 Обери абонемент:", reply_markup=keyboard)

@dp.callback_query(lambda c: c.data and c.data.startswith("add_sub_"))
async def add_subscription_to_cart(callback: types.CallbackQuery):
    sub_id = int(callback.data.split("_")[2])

    conn = sqlite3.connect('leo_fitness.db')
     cur = conn.cursor()
    cur.execute("SELECT name, price FROM subscriptions WHERE id = ?", (sub_id,))
    sub = cur.fetchone()
    conn.close()

    if sub:
        sub_name, price = sub
        add_to_cart(callback.from_user.id, "subscription", sub_id, sub_name, price)
        await callback.answer(f"✅ {sub_name} додано!")
        await callback.message.answer(f"✅ {sub_name} додано до кошика!\n💰 Ціна: {price} грн\n\nЩоб оформити
      замовлення - натисни 🛒 Кошик")
    else:
        await callback.answer("Помилка")

@dp.message(F.text == "🏋 Товари")
async def show_products(message: types.Message):
    conn = sqlite3.connect('leo_fitness.db')
    cur = conn.cursor()
```

```
cur.execute("SELECT DISTINCT type FROM products")
    types_list = cur.fetchall()
    conn.close()

    keyboard = InlineKeyboardMarkup(inline_keyboard=[])
    for t in types_list:
        keyboard.inline_keyboard.append([
            InlineKeyboardButton(text=f"📂 {t[0]}", callback_data=f"products_{t[0]}")
        ])
    await message.answer("□ Категорії товарів:", reply_markup=keyboard)

@dp.callback_query(lambda c: c.data and c.data.startswith("products_"))
async def list_products_by_type(callback: types.CallbackQuery):
    prod_type = callback.data.split("_")[1]

    conn = sqlite3.connect('leo_fitness.db')
    cur = conn.cursor()
      cur.execute("SELECT id, name, price FROM products WHERE type = ?", (prod_type,))
    products = cur.fetchall()
    conn.close()

    keyboard = InlineKeyboardMarkup(inline_keyboard=[])
    for prod_id, name, price in products:
        keyboard.inline_keyboard.append([
            InlineKeyboardButton(text=f"➕ {name} - {price} грн", callback_data=f"add_product_{prod_id}")
        ])
    await callback.message.answer(f"📂 {prod_type}\n\nОбери товар:", reply_markup=keyboard)
    await callback.answer()

@dp.callback_query(lambda c: c.data and c.data.startswith("add_product_"))
async def add_product_to_cart(callback: types.CallbackQuery):
    prod_id = int(callback.data.split("_")[2])

    conn = sqlite3.connect('leo_fitness.db')
    cur = conn.cursor()
    cur.execute("SELECT name, price FROM products WHERE id = ?", (prod_id,))
    product = cur.fetchone()
    conn.close()

    if product:
        prod_name, price = product
        add_to_cart(callback.from_user.id, "product", prod_id, prod_name, price)
        await callback.answer(f"✅ {prod_name} додано!")
        await callback.message.answer(f"✅ {prod_name} додано до кошика!\n💰 Ціна: {price} грн")

@dp.message(F.text == "🛒 Кошик")
```

```
async def show_cart(message: types.Message):
    items = get_cart(message.from_user.id)
    total = get_cart_total(message.from_user.id)
      if not items:
        await message.answer("🛒 Кошик порожній\n\nДодай абонементи або товари через відповідні меню")
        return

    cart_text = "🛒 ТВІЙ КОШИК\n\n"
    for item_id, item_name, price in items:
        cart_text += f"• {item_name} — {price} грн\n"
    cart_text += f"\n💰 Загальна сума: {total} грн"

    keyboard = InlineKeyboardMarkup(inline_keyboard=[
        [InlineKeyboardButton(text="🗑 Очистити кошик", callback_data="clear_cart")],
        [InlineKeyboardButton(text="💳 Оформити замовлення", callback_data="checkout")]
    ])
    await message.answer(cart_text, reply_markup=keyboard)

@dp.callback_query(lambda c: c.data == "clear_cart")
async def clear_cart_handler(callback: types.CallbackQuery):
    clear_cart(callback.from_user.id)
    await callback.answer("✅ Кошик очищено!")
    await callback.message.answer("🛒 Кошик очищено!")

@dp.callback_query(lambda c: c.data == "checkout")
async def process_checkout(callback: types.CallbackQuery, state: FSMContext):
    items = get_cart(callback.from_user.id)
    total = get_cart_total(callback.from_user.id)

    if not items:
        await callback.answer("Кошик порожній!")
        return

    await state.update_data(cart_items=items, total=total)

      keyboard = InlineKeyboardMarkup(inline_keyboard=[
        [InlineKeyboardButton(text="💪 Оплата в клубі", callback_data="pay_onsite")],
        [InlineKeyboardButton(text="💳 Оплата картою онлайн", callback_data="pay_online_card")]
    ])
    await callback.message.answer(f"💳 ОФОРМЛЕННЯ ЗАМОВЛЕННЯ\n\n💰 Сума: {total} грн\n\nОберіть спосіб
      оплати:", reply_markup=keyboard)
    await callback.answer()

@dp.callback_query(lambda c: c.data == "pay_onsite")
async def pay_onsite(callback: types.CallbackQuery, state: FSMContext):
    data = await state.get_data()
```

```
    items = data.get('cart_items', [])
    total = data.get('total', 0)

    order_id = f"LEO_{callback.from_user.id}_{int(datetime.now().timestamp())}"
    items_json = json.dumps([{"name": item[1], "price": item[2]} for item in items])

    conn = sqlite3.connect('leo_fitness.db')
    cur = conn.cursor()
    cur.execute('''
        INSERT INTO orders (order_id, telegram_id, items, total_price, payment_method, status, payment_status, created_at)
        VALUES (?, ?, ?, ?, ?, ?, ?, ?)
    ''', (order_id, callback.from_user.id, items_json, total, "onsite", "pending", "pending", datetime.now().isoformat()))
    conn.commit()
    conn.close()

    clear_cart(callback.from_user.id)

    # Активуємо абонемент якщо є
    for item in items:
        item_name = item[1]
        if "Пробне" in item_name:
            activate_subscription(callback.from_user.id, item_name, 1)
            elif "1 місяць" in item_name:
            activate_subscription(callback.from_user.id, item_name, 30)
        elif "12 місяців" in item_name:
            activate_subscription(callback.from_user.id, item_name, 365)

    await callback.message.answer(
        f"✅ ЗАМОВЛЕННЯ ОФОРМЛЕНО!\n\n"
        f"📋 Номер замовлення: {order_id}\n"
        f"💰 Сума: {total} грн\n"
        f"💪 Оплата в клубі\n\n"
        f"❗ БУДЬ ЛАСКА, коли прийдете до клубу,\n"
        f"зверніться до адміністратора з цим номером:\n"
        f"🔷 {order_id} 🔷\n\n"
        f"Адміністратор вам все видасть.\n"
        f"🇺🇦 Дякуємо, що Ви з нами!",
        reply_markup=main_menu_keyboard()
    )
    await state.clear()

@dp.callback_query(lambda c: c.data == "pay_online_card")
async def pay_online_card_start(callback: types.CallbackQuery, state: FSMContext):
    await callback.message.answer("💳 ОПЛАТА КАРТКОЮ ОНЛАЙН\n\nВведіть номер карти (16 цифр):")
    await state.set_state(CardPaymentState.waiting_for_card_number)
    await callback.answer()
```

```
@dp.message(CardPaymentState.waiting_for_card_number)
async def process_card_number(message: types.Message, state: FSMContext):
    card_number = message.text.replace(" ", "").replace("-", "")

    if not card_number.isdigit() or len(card_number) != 16:
        await message.answer("❌ Невірний номер. Введіть 16 цифр:")
        return

    await state.update_data(card_number=card_number[-4:])
     await message.answer("📅 Введіть термін дії (MM/PP):")
    await state.set_state(CardPaymentState.waiting_for_expiry)

@dp.message(CardPaymentState.waiting_for_expiry)
async def process_expiry(message: types.Message, state: FSMContext):
    expiry = message.text.strip()

    if len(expiry) != 5 or expiry[2] != '/':
        await message.answer("❌ Невірний формат. Введіть MM/PP:")
        return

    await state.update_data(expiry=expiry)
    await message.answer("🔐 Введіть CVV код (3 цифри):")
    await state.set_state(CardPaymentState.waiting_for_cvv)

@dp.message(CardPaymentState.waiting_for_cvv)
async def process_cvv(message: types.Message, state: FSMContext):
    cvv = message.text.strip()

    if not cvv.isdigit() or len(cvv) != 3:
        await message.answer("❌ Невірний CVV. Введіть 3 цифри:")
        return

    msg = await message.answer("⏳ ОБРОБКА ПЛАТЕЖУ...\nБудь ласка, зачекайте 15 секунд")

    await asyncio.sleep(15)

    await msg.edit_text("✅ ПЛАТІЖ ОБРОБЛЕНО!")

    data = await state.get_data()
    items = data.get('cart_items', [])
    total = data.get('total', 0)
    card_last4 = data.get('card_number', '****')

      order_id = f"LEO_{message.from_user.id}_{int(datetime.now().timestamp())}"
    items_json = json.dumps([{"name": item[1], "price": item[2]} for item in items])
```

```
    conn = sqlite3.connect('leo_fitness.db')
    cur = conn.cursor()
    cur.execute('''
        INSERT INTO orders (order_id, telegram_id, items, total_price, payment_method, status, payment_status, card_last4,
        created_at)
        VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)
    ''', (order_id, message.from_user.id, items_json, total, "online", "completed", "paid", card_last4,
        datetime.now().isoformat()))
    conn.commit()
    conn.close()

    clear_cart(message.from_user.id)

    # Активуємо абонемент якщо є
    for item in items:
        item_name = item[1]
        if "Пробне" in item_name:
            activate_subscription(message.from_user.id, item_name, 1)
        elif "1 місяць" in item_name:
            activate_subscription(message.from_user.id, item_name, 30)
        elif "12 місяців" in item_name:
            activate_subscription(message.from_user.id, item_name, 365)

    await message.answer(
        f"✅ ОПЛАТА ПРОЙШЛА УСПІШНО!\n\n"
        f"💳 Карта: ****{card_last4}\n"
        f"💰 Сума: {total} грн\n"
        f"📋 Номер замовлення: {order_id}\n\n"
        f"🎉 Дякуємо за покупку!\n"
        f"Чекаємо на вас у LEO FITNESS!\n\n"
        f"❗ БУДЬ ЛАСКА, коли прийдете до клубу,\n"
        f"зверніться до адміністратора з цим номером:\n"

            f"🔶 {order_id} 🔶\n\n"
        f"Адміністратор вам все видасть.\n"
        f"UA Дякуємо, що Ви з нами!",
        reply_markup=main_menu_keyboard()
    )

    await state.clear()

@dp.message(F.text == "📊 Мої відвідування")
async def show_visits(message: types.Message):
    conn = sqlite3.connect('leo_fitness.db')
    cur = conn.cursor()
    cur.execute("SELECT visits, active_subscription FROM users WHERE telegram_id = ?", (message.from_user.id,))
    user = cur.fetchone()
```

```
conn.close()

    if user:
        visits, sub = user
        keyboard = InlineKeyboardMarkup(inline_keyboard=[
            [InlineKeyboardButton(text="✅ Я прийшов на тренування", callback_data="checkin")]
        ])
        await message.answer(
            f"📊 СТАТИСТИКА\n\n"
            f"Відвідувань: {visits}\n"
            f"Абонемент: {sub if sub else '❌ Немає'}\n\n"
            f"Натисни кнопку, коли прийдеш у зал 👇",
            reply_markup=keyboard
        )

@dp.callback_query(lambda c: c.data == "checkin")
async def process_checkin(callback: types.CallbackQuery):
    conn = sqlite3.connect('leo_fitness.db')
    cur = conn.cursor()
    cur.execute("UPDATE users SET visits = visits + 1 WHERE telegram_id = ?",

      (callback.from_user.id,))
    cur.execute("INSERT INTO visit_log (telegram_id, visit_date) VALUES (?, ?)",
                (callback.from_user.id, datetime.now().isoformat()))
    conn.commit()
    conn.close()
    await callback.answer("✅ Відвідування зафіксовано!")
    await callback.message.answer("🎉 Вітаємо! Відвідування додано!")

@dp.message(F.text == "🏫 Тренери")
async def show_trainers(message: types.Message):
    conn = sqlite3.connect('leo_fitness.db')
    cur = conn.cursor()
    cur.execute("SELECT name, photo, description FROM trainers")
    trainers = cur.fetchall()
    conn.close()

    for name, photo, desc in trainers:
        if os.path.exists(photo):
            try:
                await message.answer_photo(photo=FSInputFile(photo), caption=f"👨 🏫 {name}\n\n{desc}")
            except:
                await message.answer(f"👨 🏫 {name}\n\n{desc}")
        else:
            await message.answer(f"👨 🏫 {name}\n\n{desc}")
```

```
async def main():
    init_db()
    print("✅ Бот LEO FITNESS запущений!")
    await dp.start_polling(bot)

if __name__ == "__main__":
    asyncio.run(main())
```

# ДОДАТОК В
(обов'язковий)

# КЕРІВНИЦТВО КОРИСТУВАЧА

1. Початок роботи з ботом
    1. Відкрийте Telegram і знайдіть бота за назвою LEO FITNESS.
    2. Натисніть кнопку «Почати» або введіть команду /start.
    3. Бот привітатиме вас і запропонує пройти реєстрацію.
2. Реєстрація нового користувача
    1. Введіть своє ім'я.
    2. Введіть актуальну адресу електронної пошти.
    3. Після успішної реєстрації бот повідомить про завершення процесу та надасть доступ до головного меню.
3. Головне меню бота

Після реєстрації вам доступні такі розділи:

- Профіль – перегляд своїх даних та статусу абонементу
- Абонементи – перегляд та купівля абонементів
- Тренери – інформація про тренерів клубу
- Товари – спортивне харчування та додаткові товари
- Кошик – перегляд та оформлення замовлення
- Мої відвідування – check-in та історія відвідувань

4. Купівля абонементу
    1. Оберіть розділ «Абонементи».
    2. Ознайомтеся з доступними абонементами.
    3. Натисніть кнопку «Додати в кошик» біля потрібного абонементу.

4. Перейдіть у «Кошик».
5. Оберіть спосіб оплати:
   – Оплата в клубі – отримаєте номер замовлення, з яким потрібно звернутися до адміністратора.
   – Оплата карткою онлайн – введіть дані картки (у тестовому режимі).

5. Використання абонементу (Check-in)

1. Приходьте в клуб.
2. Відкрийте бота та натисніть «Мої відвідування».
3. Натисніть кнопку «Я прийшов на тренування».
4. Бот автоматично списує одне заняття та підтвердить успішний check-in.

6. Перегляд особистого профілю

У розділі «Профіль» ви можете бачити:

– Ваше ім’я та email
– Активний абонемент
– Кількість залишених днів
– Кількість відвіданих тренувань

7. Робота з товарами

1. Оберіть «Товари».
2. Виберіть категорію (Протеїн, Креатин, Гейнер тощо).
3. Додайте потрібний товар у кошик.
4. Оформіть замовлення разом з абонементом або окремо.

# ДОДАТОК Г

(обов’язковий)

## ПРЕЗЕНТАЦІЙНІ МАТЕРІАЛИ


Рисунок В.1 – Слайд 1

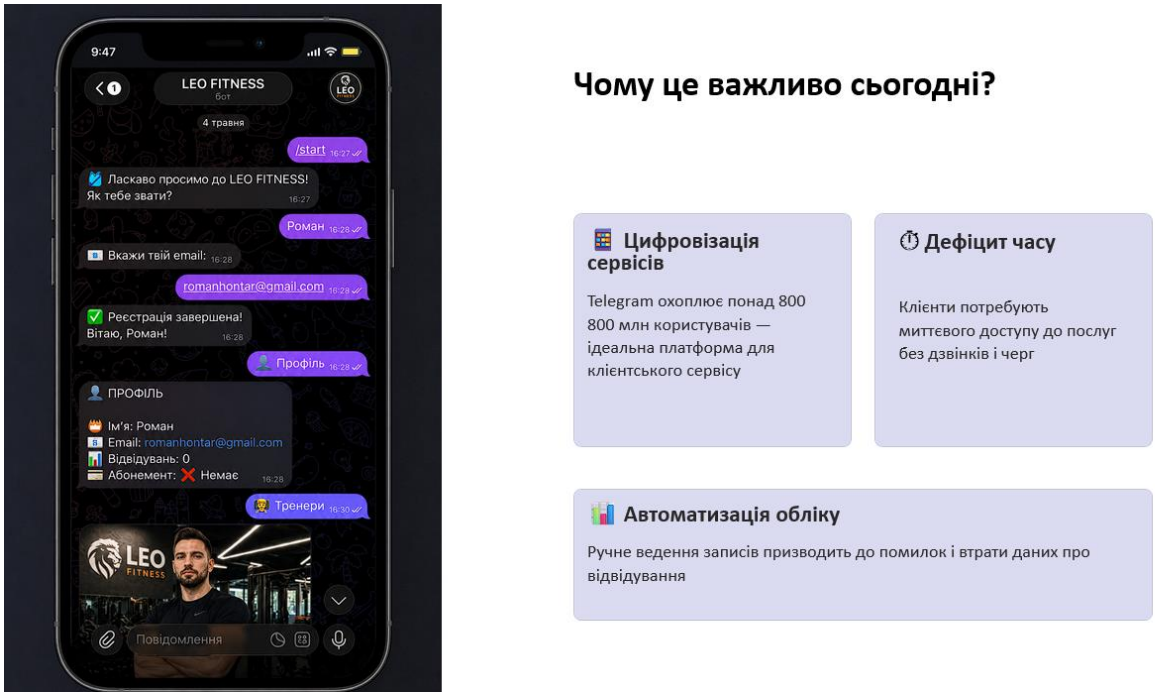



Рисунок В.2 – Слайд 2

## Мета та завдання дослідження

**Мета роботи**

Розробити Telegram-бота для автоматизації продажу продажу абонементів та обліку відвідувань у спортивному клубі **LEO FITNESS**, що підвищить ефективність обслуговування клієнтів.

**Завдання**

- Проаналізувати предметну область та існуючі аналоги
- Спроектувати архітектуру системи та базу даних
- Реалізувати функціонал бота (продаж, облік, сповіщення)
- Провести тестування та оцінку ефективності

Рисунок В.3 – Слайд 3

## Об'єкт і предмет дослідження

**Об'єкт дослідження**

Процес обслуговування клієнтів у спортивному клубі: продаж абонементів, реєстрація відвідувань, комунікація з клієнтами.

**Предмет дослідження**

Методи та засоби автоматизації процесів продажу і обліку за допомогою Telegram-бота на основі основі Python та Telegram Bot API.

**Методи дослідження**

Системний аналіз, об'єктно-орієнтоване проектування, ER-моделювання, UML-діаграми, модульне тестування.

Рисунок В.4 – Слайд 4

## Проблеми сучасних спортивних клубів

**Ручне бронювання**

Адміністратори витрачають до 3 год/день на телефонні дзвінки та запис клієнтів

**Складна оплата**

Відсутність онлайн-оплати абонементів знижує конверсію та зручність для клієнтів

**Помилки обліку**

Паперові журнали та Excel-таблиці призводять до втрати даних і конфліктів відвідувань

**Слабка комунікація**

Немає автоматичних нагадувань про тренування, закінчення абонементу та акції

Рисунок В.5 – Слайд 5

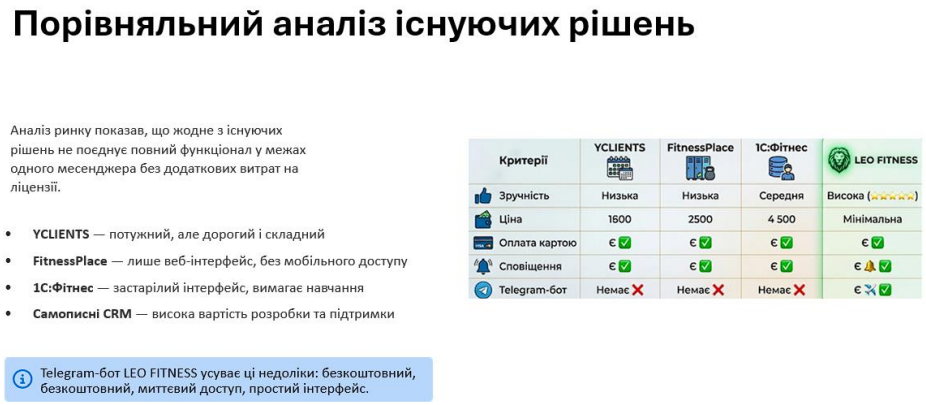



Рисунок В.6 – Слайд 6

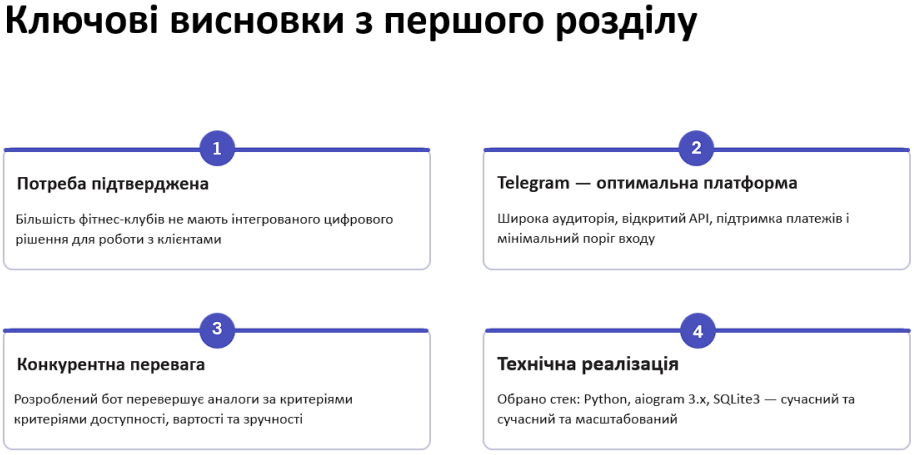



Рисунок В.7 – Слайд 7

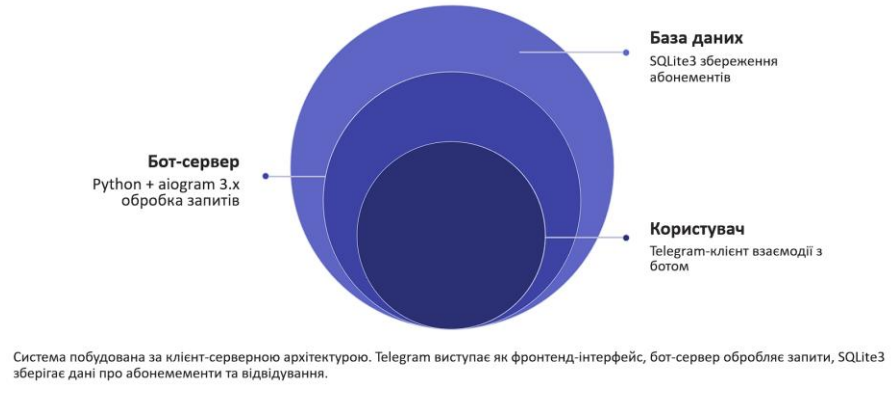



Рисунок В.8 – Слайд 8

## ER-діаграма бази даних

Схема бази даних включає п'ять основних сутностей, пов'язаних між собою зовнішніми ключами для забезпечення цілісності даних.

- **users** — профілі клієнтів (id, name, telegram_id, phone)
- **subscriptions** — типи абонементів (id, name, price, duration)
- **purchases** — історія покупок (user_id, subscription_id, status)
- **visits** — журнал відвідувань (user_id, check_in, check_out)
- **payments** — платіжні транзакції (purchase_id, amount, status)

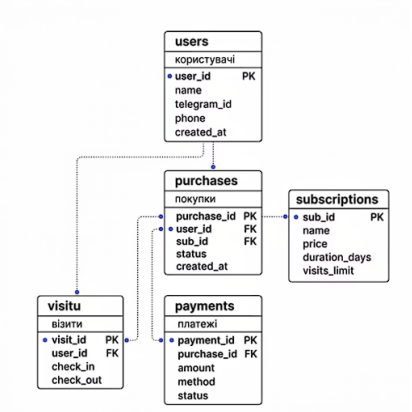



Рисунок В.9 – Слайд 9

## Діаграма варіантів використання

Система передбачає два типи акторів: **Клієнт** (користувач Telegram) Telegram) та **Адміністратор** (менеджер клубу), кожен із власним набором дій.

- **Клієнт:** реєстрація, перегляд абонементів, покупка, оплата, запис на тренування, перегляд історії відвідувань
- **Адміністратор:** управління абонементом, підтвердження відвідувань, перегляд звітів, розсилка сповіщень

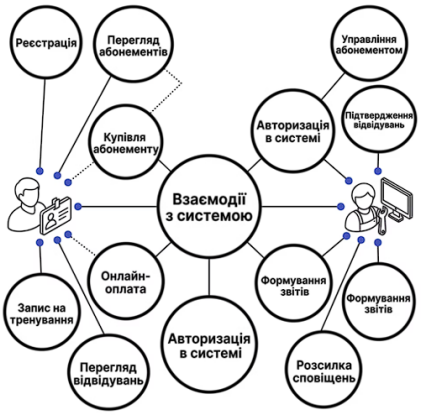



Рисунок В.10 – Слайд 10

## Інструменти та технології

**Python 3.11**

Основна мова програмування

**Aiogram 3.x**

Асинхронний фреймворк для Telegram Bot API

**SQLite3**

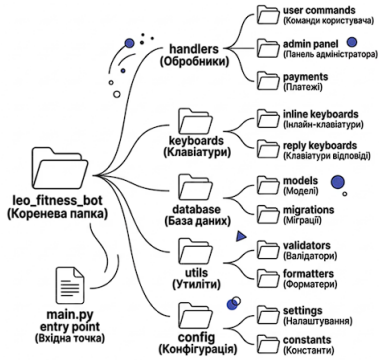
БД для зберігання даних

**Visa / Mastercard**

Інтеграція платіжних систем

Рисунок В.11 – Слайд 11

## Структура проєкту




**Модульна архітектура**

Проєкт побудований за модульним принципом — кожен компонент відповідає за окремий функціональний блок.

- **handlers/** — обробники команд та повідомлень
- **keyboards/** — інтерактивні клавіатури
- **database/** — моделі даних та міграції
- **utils/** — допоміжні функції та валідатори

Рисунок В.12 – Слайд 12

## Реєстрація та особистий профіль

**Реєстрація**

Покрокова форма: ім'я, емейл. Валідація введених даних у реальному часі.

**Особистий профіль**

Відображення активних абонементів, статистики відвідувань, історії платежів та персональних даних.

**Сповіщення**

Автоматичні нагадування про закінчення абонементу, акції акції та зміни у розкладі тренувань.

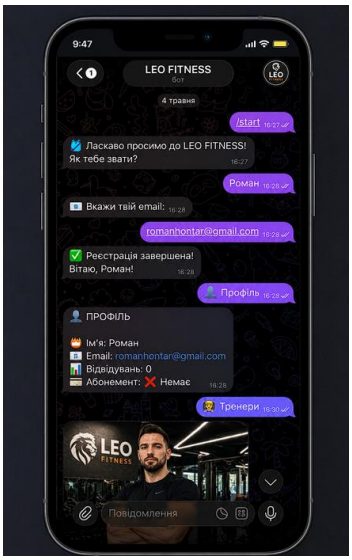



Рисунок В.13 – Слайд 13

## Розділ «Тренери»

**Каталог тренерів**

Кожен клієнт може ознайомитися з профілями тренерів клубу перед записом на заняття.

- Фото, спеціалізація та досвід роботи
- Розклад доступних годин для запису
- Відгуки та рейтинг тренера
- Прямий запис на персональне тренування

Рисунок В.14 – Слайд 14


Рисунок В.15 – Слайд 15

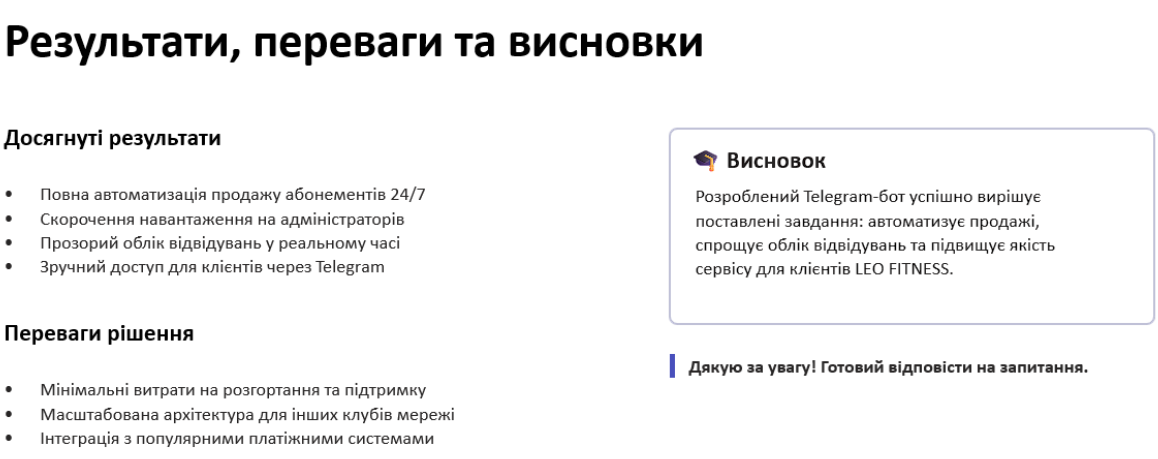
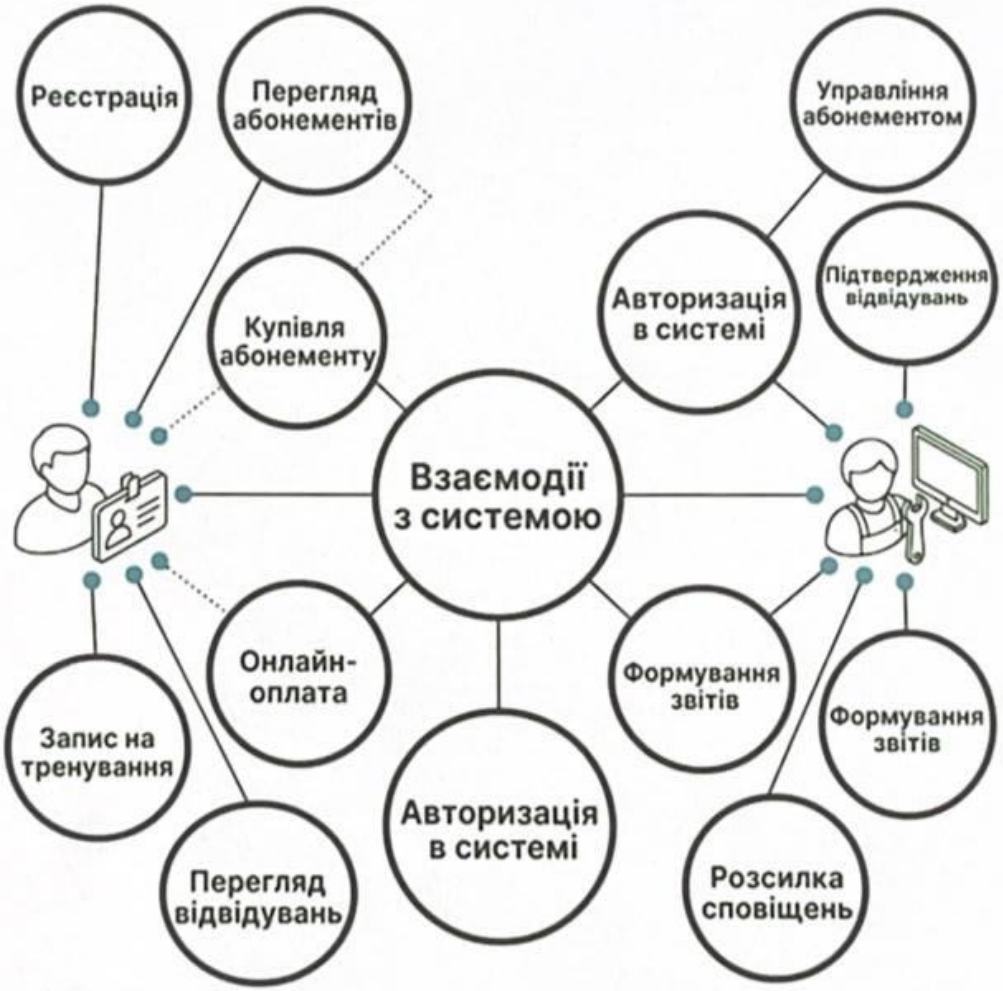
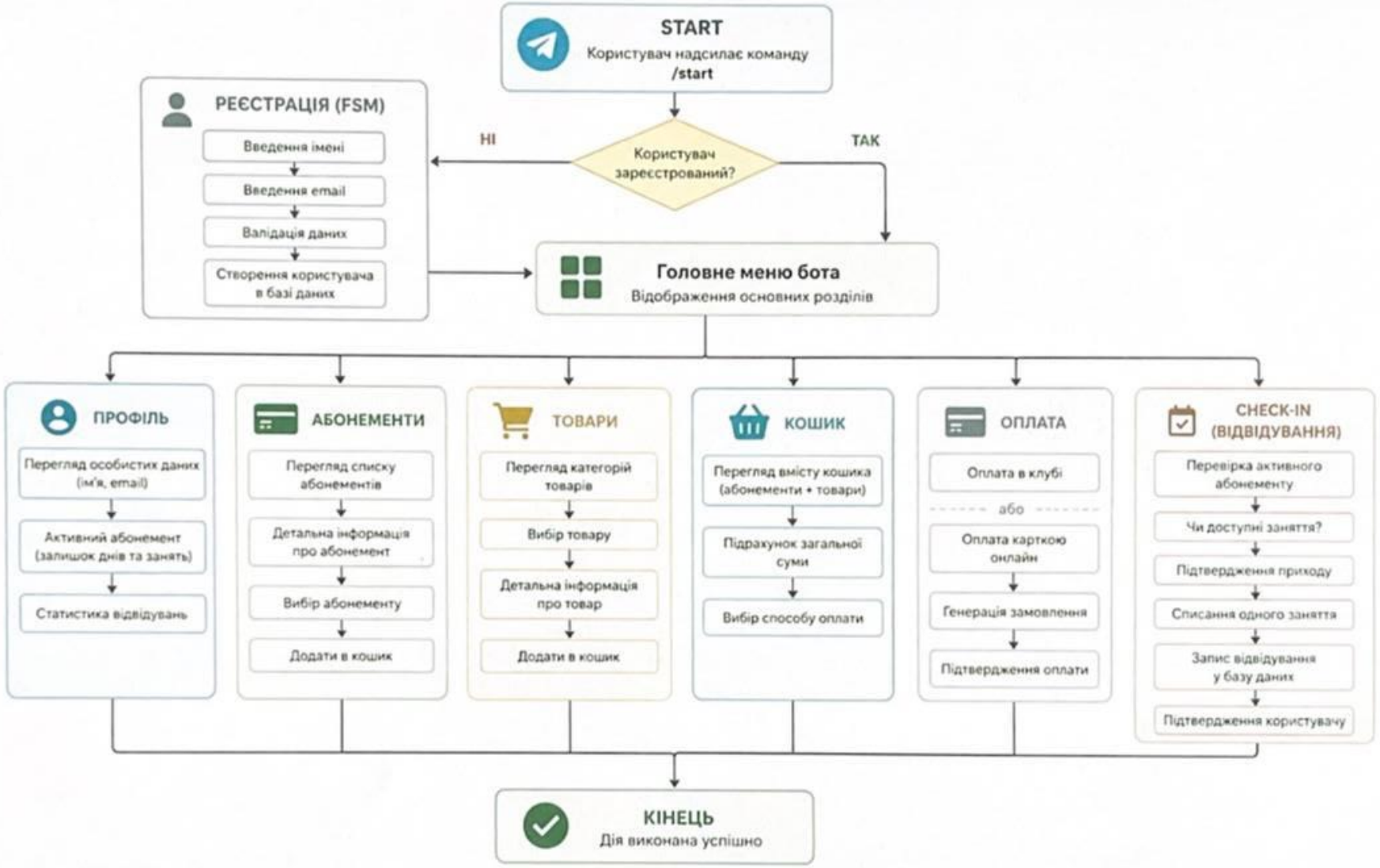



Рисунок В.16 – Слайд 16

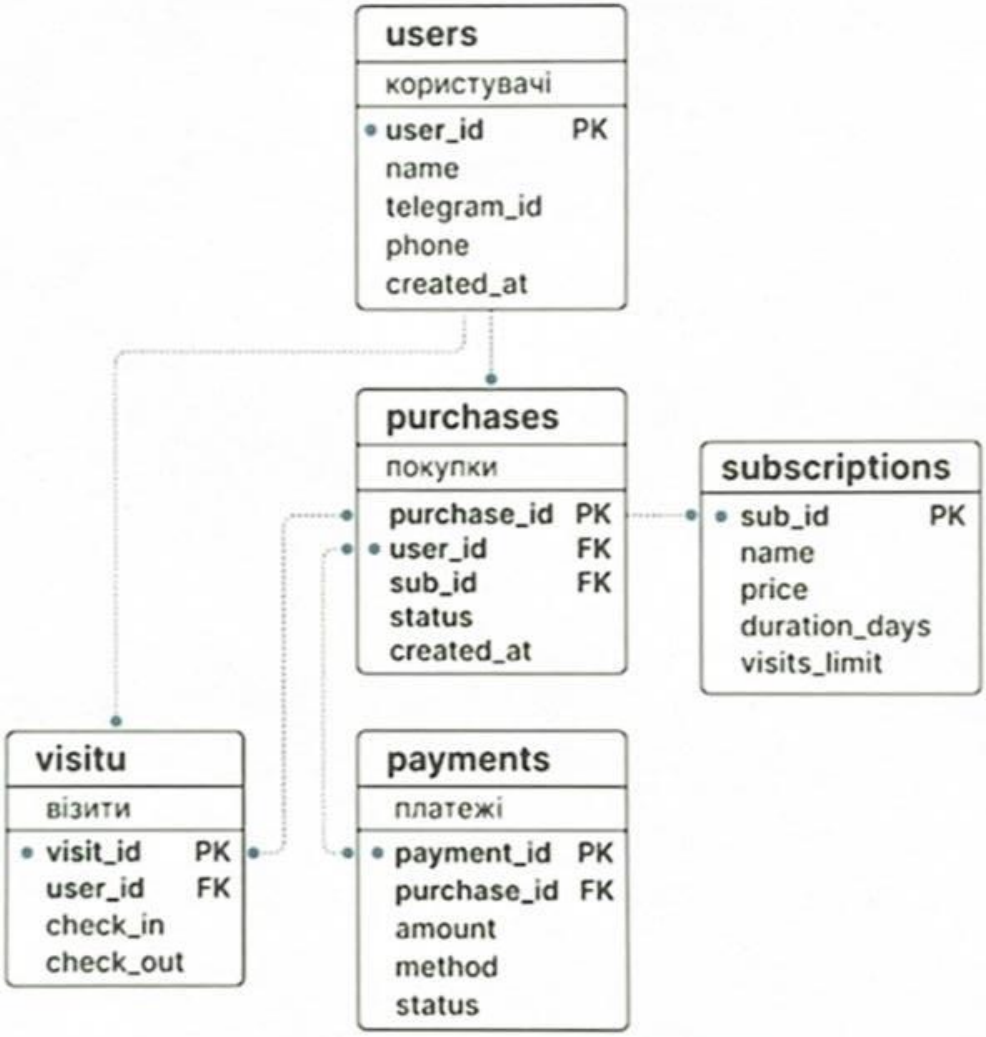
# ГРАФІЧНІ МАТЕРІАЛИ




| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | КвРІПЗ.2201100.01.05.ПЗ | |
| | | | | | Діаграма варіантів використання | Літера: — / Маса: — / Масштаб: — |
| Зм. | Арк. | № докум. | Підпис | Дата | | Аркуш 1 / Аркушів 3 |
| Розробив | | Гонтар Л.С. | | 01.06 | | |
| Керівник | | Бойко В.О. | | 01.06 | | |
| Консульт. | | | | | | |
| Н. Контр. | | Праворська Н.І. | | 01.06 | | ХНУ, ІПЗ-22-1 |
| Зав. каф. | | Бедратюк Л.П. | | 01.06 | | |




| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | КвРІПЗ.2201100.01.05.ПЗ | |
| Зм. | Арк. | № докум. | Підпис | Дата | | |
| Розробив | | Гонтар Л.С. | | | Діаграма функціоналу бота | Літера / Маса / Масштаб |
| Керівник | | Бойко В.О. | | | | Аркуш 2 / Аркушів 3 |
| Консульт. | | | | | | |
| Н. Контр. | | Праворська Н. | | | | ХНУ, ІПЗ-22-1 |
| Зав. каф. | | Бедратюк Л.П. | | | | |




| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | КвРІПЗ.2201100.01.05.ПЗ | | |
| Зм. | Арк. | № докум. | Підпис | Дата | | | |
| Розробив | | Гонтар Л.С. | | | Діаграма бази даних | Літера | Маса / Масштаб |
| Керівник | | Бойко В.О. | | | | | |
| Консульт. | | | | | | Аркуш 3 | Аркушів 3 |
| Н. Контр. | | Праворська Н. І. | | | ХНУ, ІПЗ-22-1 | | |
| Зав. каф. | | Федратюк Л.П. | | | | | |

# СУПРОВІДНІ ДОКУМЕНТИ

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

# ДЕКЛАРАЦІЯ УЧАСНИКА ОСВІТНЬОГО ПРОЦЕСУ

## щодо дотримання академічної доброчесності

Цією декларацією я, Гонтар Леонід Сергійович,

студент IV курсу спеціальності 121 – Інженерія програмного забезпечення, група ІПЗ-22-1

здобувач вищої освіти (шифр та назва спец-ті, курс, академічна група)

підтверджую, що ознайомився (-лась) з Положенням про систему забезпечення академічної доброчесності у Хмельницькому національному університеті та Кодексом академічної доброчесності і **зобов'язуюсь** дотримуватися їх вимог під час освітнього процесу, проведення наукової діяльності, виконання організаційно-адміністративних функцій тощо.

**Усвідомлюю,** що у разі порушення мною принципів академічної доброчесності нестиму відповідальність перед академічною спільнотою ХНУ згідно з нормами, визначеними Положенням про систему забезпечення академічної доброчесності у Хмельницькому національному університеті, законодавства України.

02 вересня 2025 р. _____

Підпис

Завідувачу кафедри
інженерії програмного забезпечення
проф. Бедратюку Л. П.
студента групи ІПЗ-22-1
Гонтар Л.С.
Прізвище, ініціали

**ЗАЯВА**

Прошу закріпити за мною тему кваліфікаційної роботи освітнього ступеня «бакалавр» за спеціальністю 121 «Інженерія програмного забезпечення»: Telegram-бот для продажу абонементів і обліку відвідувань у спортивному клубі

(керівник роботи – Бойко В.О.)
Прізвище, ім'я, по батькові

02.01.2026р.
Дата

Підпис студента

Завідувачу кафедри інженерії програмного
забезпечення проф. Леоніду БЕДРАТЮКУ
здобувача вищої освіти
Гонтара Леоніда Сергійовича
факультет ІТ, IVкурс, група ІПЗ-22-1

**ЗАЯВА**

З правилами чинного Положення про систему забезпечення академічної доброчесності в Хмельницькому національному університеті, згідно з яким виявлення академічного плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту і застосування заходів дисциплінарної та академічної відповідальності, ознайомлений. Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на наявність академічного плагіату оповіщений та надаю свою згоду на обробку й збереження університетом моєї роботи в інституційному репозитарії Хмельницького національного університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-обчислювального комплексу StrikePlagiarism та/або програмно-технічного засобу AntiPlagiarism і використання роботи для виявлення академічного плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення текстових збігів у роботах.

Робота надається для перевірки в електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

14.05.2026р.
дата

підпис

**Протокол аналізу звіту подібності науковим керівником**

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

**Автор:** Леонід ГОНТАР

**Співавтор:**

**Назва:** Telegram-бот для продажу абонементів і обліку відвідувань у спортивному клубі

**Науковий керівник:** В'ячеслав БОЙКО

**Підрозділ:** Кафедра інженерії програмного забезпечення

**Коефіцієнт подібності 1:**2.93%

**Коефіцієнт подібності 2:**1.09%

**Мікропробіли:** 18

**Заміна букв:** 2

**Інтервали:** 13

**Білі знаки:** 0

**Дата створення звіту:** 2026-05-20 00:50:41.0

**Після аналізу Звіту подібності констатую наступне:**

☑ **Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.**

☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

☐ **Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.**

Обгрунтування:

*Дата* 26.05.26

експерт (Горщук Ю.В.)

Wed May 20 00:44:39 EEST 2026, Форкун Юрій Вікторович, Хмельницький національний університет, ХНУ

# Anti-Plagiarism (http://ap.km.ua) v-16.718

**Максимальне співпадіння з одним документом 1.0%**

Словники перевірки: UA, US, RU. **Помилок в документах: 8%**

| ID: 271746<br>Назва: БКР_Telegram-бот для продажу абонементів і обліку відвідувань у спортивному клубі<br>Додано в БД: 2026-05-20<br>Автора: Леонід ГОНТАР<br>Керівники: В'ячеслав БОЙКО<br>Консультанти:<br>Опоненти: | Документ | | Сумарний збіг по Базі Даних | |
|---|---|---|---|---|
| | Символи | Лексеми | Символи | Лексеми |
| | 115965 | 1080 | 3519 (3%) | 55 (5%) |

Джерело плагіату

| ID | Опис | Наявність плагіату в документі | |
|---|---|---|---|
| | | Символи | Лексеми |

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

**РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ**
освітнього ступеня «Бакалавр»

Дипломник Гонтар Леонід Сергійович

Тема Telegram-бот для продажу абонементів і обліку відвідувань у спортивному клубі

Спеціальність 121 – Інженерія програмного забезпечення

**Обсяг кваліфікаційної роботи:**

Кількість листів креслень 3; кількість сторінок записки 88

1.Короткий зміст пояснювальної записки та прийнятих рішень У кваліфікаційній роботі досліджено і проаналізовано предметну область, визначено усі функціональні та нефункціональні вимоги. Був проведений аналіз існуючих програм на ринку, розглянуто їх переваги і недоліки, та доведено актуальність розробки нового програмного забезпечення. Розглянуто інструменти для реалізації спроектованих рішень, в результаті чого створено програмне забезпечення. Також було проведено тестування програми, за результатами якого доведено, що розроблене програмне забезпечення працює коректно та готове до експуатації.

2. Висновок про відповідність роботи поставленому завданню Кваліфікаційна робота виконана відповідно до поставленого завдання та з дотриманням всіх вимог.

3. Характеристика виконання кожного розділу роботи, ступінь використання останніх досягнень науки і техніки та передових методів роботи У вступі доведено актуальність теми, визначено мету та завдання дипломного проектування. У першому розділі проведено аналіз предметної області, розглянуто існуючі рішення та визначені функціональні і нефункціональні вимоги до розроблюваного програмного забезпечення. У другому розділі проведено аналіз сучасних архітектур, розглянуто їх переваги і недоліки та визначено, що система буде відповідати монолітній архітектурі та моделі клієнт-сервер. У третьому розділі підготовлено всі залежності для написання коду та виконано практичну розробку програмних модулів і описано їх особливості, в результаті чого створено програмний продукт. Також у цьому розділі виконано модульне тестування системи та проведено його у відповідності до функціональних вимог, в результаті було підтверджено коректну роботу програми.

4. Позитивні сторони роботи Тематика кваліфікаційної роботи є актуальною та затребуваною, оскільки більшість спортивних клубів України досі використовують застарілі методи обліку відвідувань і продажу абонементів. Розроблений Telegram-бот LEO FITNESS дозволяє значно автоматизувати роботу клубу, підвищити якість обслуговування клієнтів і зменшити навантаження на адміністраторів. У роботі використано сучасні технології (Python 3.12, aiogram 3.x, асинхронне програмування), що свідчить про високий технічний рівень студента. Бот має зручний інтерфейс, надійну архітектуру та реальну практичну цінність.

5. Негативні сторони роботи У роботі можна було б додати розширений функціонал, наприклад, інтеграцію з турнікетами або систему лояльності для постійних клієнтів. Бажано було б провести повноцінне навантажувальне тестування при великій кількості одночасних користувачів.

6. Оцінка графічного оформлення та пояснювальної записки Графічне оформлення виконано на високому рівні. Презентація містить необхідні діаграми (архітектура системи, ER-діаграма, Use Case), скріншоти реальної роботи бота та чіткі ілюстрації. Пояснювальна записка оформлена відповідно до вимог державних стандартів, має чітку структуру, правильне оформлення рисунків і таблиць.

7. Відгук про кваліфікаційну роботу в цілому Кваліфікаційна робота виконана на достатньому інженерному рівні. Матеріал викладений логічно, послідовно і зрозуміло. Студент продемонстрував гарні знання сучасних технологій розробки програмного забезпечення, вміння аналізувати предметну область та створювати практичні рішення. Розроблений програмний продукт має реальну практичну цінність і може бути використаний у діяльності спортивного клубу.

8. Інші зауваження

9. Оцінка кваліфікаційної роботи Кваліфікаційна робота виконана у повному обсязі, відповідає поставленій задачі та заслуговує на оцінку «добре».

РЕЦЕНЗЕНТ Мартинюк Валерій Володимирович, доктор технічних наук, професор, зав. кафедри автоматизації, комп'ютерно-інтегрованих технологій та робототехніки

"27" 05 2026 р. (підпис)

**РІШЕННЯ** ЕКСПЕРТНОЇ КОМІСІЇ
КАФЕДРИ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатами звіту/звітів перевірки роботи, продукованими програмно-технічним засобом (ами), на наявність текстових збігів.
Назва кваліфікаційної роботи: «Telegram-бот для продажу абонементів і обліку відвідувань у спортивному клубі»
Автор: Гонтар Леонід Сергійович
Освітня програма: Освітньо-професійна програма «Інженерія програмного забезпечення»
Спеціальність: 121 – Інженерія програмного забезпечення
Науковий керівник: Бойко В'ячеслав Олександрович, асистент
Після аналізу звіту/звітів зроблено такий висновок:

| № | Висновок | Позначка про відповідність |
|---|---|---|
| 1 | Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається до захисту. | **відповідає** |
| 2 | Виявлені запозичення не є академічним плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота приймається до захисту, але має бути відкоригована. | |
| 3 | Виявлені запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Виявлені запозичення частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнуті. Робота може бути допущена до захисту після того, як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат. | |
| 4 | Робота містить навмисні текстові спотворення, передбачувані спроби укриття текстових запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту. | |
| 5 | Інше: | |

Підтвердження:
Запозичення, виявлені у роботі, є законними і не є плагіатом, оскільки:
1) у тексті кваліфікаційної роботи системою перевірки на плагіат Anti-Plagiarism виявлено схожість з деякими документами у частині загальновживаних обов'язкових словосполучень у стандартних бланках, у структурі змісту, назвах розділів/підрозділів, рамках форм, у назвах та URL-адресах публікацій переліку джерел посилання;
2) запозичення, виявлені у тексті роботи, є фрагментарними.
Загальна сумарна подібність у базі даних складає 4% за символами та 6% за лексемами. Крім того, за результатами додаткового аналізу коефіцієнт подібності 1 становить 2,93%, коефіцієнт подібності 2 – 1,09%, мікропробілів – 18, заміна букв – 2, інтервали – 13. Не виявлено зайвих білих знаків. З урахуванням наведених обґрунтувань, відповідає характеру теми і свідчить на користь кваліфікаційної роботи.

Дата 3.06.26

Завідувач кафедри _____ Леонід БЕДРАТЮК

Гарант освітньої програми _____ Леонід БЕДРАТЮК

Керівник кваліфікаційної роботи _____ В'ячеслав БОЙКО