Полтавський університет економіки і торгівлі

Навчально-науковий інститут денної освіти
Форма навчання денна
Кафедра комп'ютерних наук та інформаційних технологій

Допускається до захисту
Завідувач кафедри
_____Олена ОЛЬХОВСЬКА
(підпис)
«___»_____202_ р.

КВАЛІФІКАЦІЙНА РОБОТА

на тему
«РОЗРОБКА TELEGRAM-БОТА ПО СТВОРЕННЮ КОМПЛЕКСУ ПЕРСОНАЛЬНИХ ФІЗИЧНИХ НАВАНТАЖЕНЬ»

зі спеціальності 122 «Комп'ютерні науки»
освітня програма «Комп'ютерні науки»
ступеня бакалавр

Виконавець роботи Кухлій Ростислав Євгенович
_____«__»_____202_р.
(підпис)
Науковий керівник к.ф.-м.н.,доц.,_____
_____«__»_____202_р.
(підпис)
Рецензент

ПОЛТАВА 2026

ЗАТВЕРДЖУЮ
Завідувач кафедри_____Олена ОЛЬХОВСЬКА
«____» вересня 2025 р.

ЗАВДАННЯ ТА КАЛЕНДАРНИЙ ГРАФІК ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

на тему «Розробка telegram-бота по створенню комплексу персональних фізичних навантажень»
зі спеціальності 122 «Комп'ютерні науки»
освітня програма «Комп'ютерні науки»
ступеня бакалавр
Прізвище, ім'я, по батькові Кухлій Ростислав Євгенович
Затверджена наказом ректора № ____-Н від « » _____ 2026 р.
Термін подання студентом роботи «__ » _____ 2026р.
Вихідні дані до кваліфікаційно роботи: методики персоналізації фізичних навантажень, публікації з розробки чат-ботів, вхідні дані користувача.

ВСТУП
РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ
1.1. Мета та завдання проєкту
1.2 Аргументація необхідності створення чат-бота персональних фізичних навантажень.
1.3 Характеристика предметної області (тренування, навантаження, користувацькі профілі)
РОЗДІЛ 2. ІФОРМАЦІЙНИЙ ОГЛЯД.
2.1 Проблематика створення фітнес-систем та цифрових тренерів.
2.2 Огляд програмних засобів для створення чат-ботів
2.3 Основні поняття штучного інтелекту в контексті персоналізації тренувань
РОЗДІЛ 3. ТЕОРЕТИЧНА ЧАСТИНА
3.1 Розробка структури проєкту
3.2 Обґрунтування вибору технологій для розробки чат-бота
3.3 Обґрунтування вибору середовища розробки
3.4 Алгоритм роботи програмного забезпечення
3.5 Блок-схема роботи ПЗ
РОЗДІЛ 4. ПРАКТИЧНА ЧАСТИНА
4.1 Вибір інструментальних засобі
4.2 Структура програмного забезпечення
4.3 Реалізація логіки обробки повідомлень
4.4 Реалізація функції генерації тренувань

4.5 Тестування роботи чат-бота
4.6 Реалізація функціоналу “Калькулятор калорій”
4.7 Демонстрація роботи програми
4.8 Висновки
ВИСНОВКИ
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ
ДОДАТОК А. Код програми
ДОДАТОК Б. Перелік тестових питань

Перелік графічного матеріалу: 3-4 аркуші графічного матеріалу, інші необхідні ілюстрації.
Консультанти розділів кваліфікаційної роботи

Розділ	ПІП, посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Постанова задачі			
Інформаційний огляд			
Теоретична частина			
Практична реалізація			

Календарний графік виконання кваліфікаційної роботи

Зміст роботи	Термін виконання	Фактичне виконання
1. Вступ		
2. Вивчення методичних рекомендацій та стандартів та звіт керівнику		
3. Постановка задачі		
4. Інформаційний огляд джерел бібліотек та інтернету		
5. Теоретична частина		
6. Практична частина		
7. Закінчення оформлення		
8. Доповідь студента на кафедрі		
9. Доробка (за необхідністю), рецензування		

Дата видачі завдання « __» _____ 202_ р.
Здобувач вищої освіти Кухлій Ростислав Євгенович
Науковий керівник _____.
Результати захисту кваліфікаційної роботи

Кваліфікаційна робота оцінена на _____
(балів, оцінка за національною шкалою, оцінка за ECTS)
Протокол засідання ЕК № ____ від «____» _____ 202_ р.
Секретар ЕК_____ _____
(підпис) (ініціал та прізвище)

Затверджую
Зав. кафедрою _____
к.ф.-м.н. Олена ОЛЬХОВСЬКА
«__» _____ 202_ р.

Погоджено
Науковий керівник_____
_____
«__» _____ 202_ р.

План

кваліфікаційної роботи на тему
«Розробка чат-бота для створення комплексу персональних фізичних навантажень»
зі спеціальності 122 Комп'ютерні науки
освітня програма 122 Комп'ютерні науки
ступеня бакалавр
Прізвище, ім'я, по батькові Кухлій Ростислав Євгенович

ВСТУП
РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ
1.1. Мета та завдання проєкту
1.2 Аргументація необхідності створення чат-бота персональних фізичних навантажень.
1.3 Характеристика предметної області (тренування, навантаження, користувацькі профілі)
РОЗДІЛ 2. ІФОРМАЦІЙНИЙ ОГЛЯД.
2.1 Проблематика створення фітнес-систем та цифрових тренерів.
2.2 Огляд програмних засобів для створення чат-ботів
2.3 Основні поняття штучного інтелекту в контексті персоналізації тренувань
РОЗДІЛ 3. ТЕОРЕТИЧНА ЧАСТИНА
3.1 Розробка структури проєкту
3.2 Обґрунтування вибору технологій для розробки чат-бота
3.3 Обґрунтування вибору середовища розробки
3.4 Алгоритм роботи програмного забезпечення
3.5 Блок-схема роботи ПЗ
РОЗДІЛ 4. ПРАКТИЧНА ЧАСТИНА
4.1 Вибір інструментальних засобі
4.2 Структура програмного забезпечення
4.3 Реалізація логіки обробки повідомлень
4.4 Реалізація функції генерації тренувань
4.5 Тестування роботи чат-бота

Здобувач вищої освіти _____ Ростислав КУХЛІЙ
«___» _____ 202_ р.

РЕФЕРАТ

Записка: 58 ст, 4 розділи, 20 джерел.

ЧАТ-БОТ, ПЕРСОНАЛЬНІ ФІЗИЧНІ НАВАНТАЖЕННЯ, КАЛЬКУЛЯТОР КАЛОРІЙ, C++, VISUAL STUDIO CODE, АЛГОРИТМИ ПЕРСОНАЛІЗАЦІЇ, АВТОМАТИЗАЦІЯ РЕКОМЕНДАЦІЙ.

Об’єкт розробки – створення Telegram-бота для автоматизованого формування персонального тренувального плану та розрахунку добової норми калорій.

Мета роботи – розробити програмний продукт у вигляді чат-бота, який дозволяє користувачеві обрати мету тренувань, рівень підготовки, отримати готовий комплекс вправ, а також розрахувати добову потребу в калоріях залежно від індивідуальних параметрів.

Методи реалізації – програмна реалізація виконана мовою C++ із використанням бібліотеки для роботи з Telegram Bot API. Для розробки використовувалося середовище Visual Studio Code. Логіка роботи побудована на обробці повідомлень користувача та послідовному переході між етапами вибору параметрів.

Практична частина включає розробку алгоритму роботи бота, створення блок-схеми, програмну реалізацію функціоналу формування розкладу тренувань (схуднення або набір маси), вибір рівня підготовки та груп м’язів. Окремо реалізовано модуль калькулятора калорій, який виконує розрахунок базового обміну речовин та визначає рекомендовану добову норму калорій.

У результаті створено працюючий програмний прототип, який можна використовувати для отримання персоналізованих рекомендацій щодо тренувань і харчування через месенджер Telegram.

У роботі розглянуто основні аспекти та етапи створення програмного забезпечення, орієнтованого на автоматизований підбір фізичних навантажень у форматі чат-бота. Проведено огляд сучасних цифрових рішень у сфері фітнес-рекомендацій та інтелектуальних помічників, визначено основні переваги використання персоналізованих тренувальних систем.

Розкрито проблематику формування ефективних фізичних навантажень для користувачів із різними рівнями підготовки, обґрунтовано необхідність автоматизації вибору тренувальних комплексів. Визначено алгоритм функціонування чат-бота, побудовано блок-схему основних процесів взаємодії з користувачем.

Програмна реалізація чат-бота виконувалася у середовищі Visual Studio Code із використанням мови програмування C++, що забезпечило гнучкість розробки, можливість розширення функціоналу та застосування об’єктно-орієнтованої архітектури. Створені програмні модулі дозволяють формувати персоналізовані комплекси фізичних навантажень, які можуть бути доповнені новими функціями та інтегровані у повноцінну систему цифрового фітнес-консультування.

Додатково у програмному продукті реалізовано функціонал калькулятора добової норми калорій. Модуль забезпечує розрахунок базового обміну речовин (BMR) та визначення рекомендованої добової калорійності з урахуванням віку, статі, зросту, маси тіла та рівня фізичної активності користувача.

На основі отриманих результатів формується орієнтовне співвідношення білків, жирів і вуглеводів, що дозволяє користувачу комплексно підходити до планування фізичних навантажень і харчування. Реалізація даного функціоналу підвищує практичну цінність програмного продукту та розширює можливості його застосування у сфері цифрового фітнес-консультування.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Умовні позначення, символи, скорочення, терміни	Пояснення умовних позначень, символів, скорочень
C++	Об’єктно-орієнтована мова програмування, використана для створення чат-бота
VS Code	Visual Studio 2022 — інтегроване середовище розробки, у якому реалізовано чат-бот
Чат-бот	Програмний застосунок, що автоматично взаємодіє з користувачем через текстові повідомлення
ПЗ	Програмне забезпечення
API	Application Programming Interface — інтерфейс взаємодії програмних компонентів
IDE	Інтегроване середовище розробки
Фреймворк	Програмний каркас, що спрощує процес створення ПЗ
Telegram Bot API	Протокол і набір методів для розробки Telegram-ботів
JSON	Формат обміну структурованими даними між системами
HTTP	Протокол передачі даних, який використовується для відправки запитів ботом
База даних (БД)	Засіб зберігання інформації, що використовується чат-ботом (за потреби)
MVC	Model-View-Controller — архітектурний шаблон проєктування ПЗ
AI/ML (за наявності)	Технології штучного інтелекту, що можуть бути використані для обробки відповідей бота

ВСТУП

Актуальність даної роботи полягає в тому, що штучний інтелект, автоматизація та цифрові сервіси активно впроваджуються у всі сфери життя, зокрема й у сферу освіти, сервісної діяльності та взаємодії користувача з інформаційними системами. З кожним роком збільшується кількість студентів та розробників, які використовують чат-ботів для спрощення комунікації, автоматизації побутових завдань, отримання довідкової інформації та підтримки навчального процесу.

Чат-боти сьогодні є невід'ємною частиною сучасних цифрових платформ, вони інтегруються у соціальні мережі, месенджери та веб-сайти, надаючи користувачам можливість миттєвої взаємодії без участі живого оператора. Така тенденція обумовлена потребою у швидкому доступі до інформації, економії часу, автоматизації обслуговування та підвищенні ефективності обміну даними між користувачем і системою.

Активний розвиток технологій, зокрема API-сервісів, C++ та інструментів розробки, дозволяє створювати функціональних чат-ботів навіть у навчальних цілях. Це робить тему даної курсової актуальною, оскільки розробка чат-бота формує у студента практичні навички програмування, роботи з мережевими запитами, застосування структур даних та архітектурних підходів.

Створення чат-бота на мові програмування C++ [2] у середовищі Visual Studio 2022 дозволяє засвоїти сучасні інструменти розробки та отримати досвід у реалізації реального програмного продукту. Крім того, використання Telegram Bot API дає можливість створити застосунок, який може бути інтегрований у повсякденне використання студентами, викладачами або звичайними користувачами.

Відповідно, розроблений чат-бот може слугувати прикладом практичної реалізації знань з програмування, а також допоміжним інструментом для

автоматизації певних задач (інформаційних чи навчальних), що підвищує ефективність взаємодії користувача з інформаційними сервісами.

Метою роботи є розробка програмного забезпечення — чат-бота з використанням мови програмування C++ інтегрованого середовища розробки Visual Studio Code та сервісу Telegram Bot API [4].

Досягнення мети передбачає виконання наступних завдань:

1. Аналіз існуючих рішень та технологій для створення чат-ботів.
2. Вибір інструментів, середовища розробки та API для реалізації проєкту.
3. Розробка структури та архітектури чат-бота.
4. Реалізація програмних модулів чат-бота мовою C++.
5. Тестування роботи чат-бота та перевірка коректності взаємодії з користувачем.
6. Опис отриманих результатів та аналіз роботи створеного програмного забезпечення.

Для розробки програмного забезпечення за темою «Розробка чат-бота» була використана мова програмування C++ у середовищі Visual Studio 2022 з інтеграцією Telegram Bot API. Застосування даного API [4] дає змогу створити простий, інтуїтивно зрозумілий та зручний для користувача інтерфейс взаємодії, який працює через популярний месенджер Telegram. Таким чином, користувачі можуть швидко отримувати інформацію, виконувати необхідні дії та взаємодіяти із системою без складних налаштувань.

Завданням чат-бота є автоматизація отримання інформації, спрощення взаємодії користувача з цифровими сервісами та забезпечення ефективної комунікації без участі оператора. Завдяки простій структурі та зрозумілому інтерфейсу, користувач може легко використовувати чат-бот як інструмент для отримання довідок, відповідей на запитання чи виконання певних функціональних дій відповідно до його призначення.

Методи дослідження: емпіричні, теоретичні, комплексні (узагальнення, порівняння, групування, систематизація).

Об’єкт дослідження: сучасні автоматизовані системи комунікації та цифрові сервіси на основі чат-ботів.

Предмет дослідження: програмне забезпечення чат-бота, розроблене з використанням мови програмування C++ та Telegram Bot API.

Інформаційна база: дослідження становлять наукові статті, монографії, підручники, навчальні посібники, технічна документація Telegram Bot API, аналітичні огляди, ресурси мережі Internet та інші джерела, що висвітлюють проблематику створення та впровадження чат-ботів.

ПОСТАНОВКА ЗАДАЧІ

Опис алгоритму

На даному етапі розроблено детальний опис послідовності кроків, які необхідно виконати для створення працездатного чат-бота, що автоматично формує комплекс персональних фізичних навантажень. У цьому розділі представлено покроковий алгоритм функціонування чат-бота — від моменту запуску користувачем до формування кінцевого індивідуального тренувального плану.

Алгоритм містить усі необхідні дії, починаючи з отримання вхідних даних (ціль тренування, рівень підготовки, вибір м’язових груп), їх обробки та закінчуючи генерацією готового комплексу вправ. Також у цьому етапі алгоритм буде подано у вигляді текстової блок-схеми з використанням графічних елементів для кращого розуміння логіки роботи чат-бота.

Перенесення алгоритму у вигляд блок-схеми

Це завдання полягає у поданні розробленого алгоритму у формі блок-схеми — графічного відображення послідовності дій, які виконує чат-бот. Блок-схема допоможе наочно продемонструвати логіку роботи бота, показати основні етапи взаємодії з користувачем, прийняття рішень та формування тренувальної програми.

У блок-схемі використовуються стандартні позначення:

- Процеси / дії — обробка даних, генерація вправ.
- Рішення / умови — перевірка вибраної цілі, рівня, групи м’язів.
- Ввід / вивід — отримання відповіді від користувача та відправка сформованого комплексу.
- З’єднувачі та стрілки — для позначення послідовності переходу між етапами.

Блок-схема дає змогу візуалізувати основні кроки роботи чат-бота та полегшує розуміння структури алгоритму під час подальшої реалізації.

Розробка елементів чат-бота

Завдання полягає у створенні ключових елементів програмного забезпечення, які забезпечують повноцінну роботу чат-бота з формування персональних тренувальних програм.

Основні елементи чат-бота:

- Модуль взаємодії з користувачем — отримання відповідей: ціль тренування (схуднення, набір маси, рельєф), рівень підготовки, групи м’язів або запит на повне тренування.
- Модуль логіки — обробка введених даних, вибір відповідних вправ залежно від рівня та цілі, розподіл навантаження.
- База даних або набір структурованих списків вправ — містить вправи для окремих груп м’язів (груди, спина, ноги, руки, плечі, прес), а також їх адаптацію до рівня складності.
- Модуль генерації комплексу тренувань — формує індивідуальний план, який може містити кількість підходів, повторень, рекомендації щодо техніки виконання.
- Модуль виведення результатів — відправка користувачу оформленого тренувального комплексу у Telegram.

Результатом роботи чат-бота є персональний комплекс фізичних навантажень, який відповідає цілям та фізичним можливостям користувача та може бути використаний у реальних тренуваннях.

РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ

У сучасних умовах розвитку інформаційних технологій зростає попит на програмні засоби, що дозволяють автоматизувати процес формування індивідуальних програм фізичних навантажень. Багато користувачів не мають достатнього досвіду для самостійного складання ефективного тренувального плану або розрахунку добової норми калорій, що може призводити до помилок та неефективності занять.

У зв’язку з цим виникає необхідність розробки програмного продукту, який забезпечить зручну взаємодію користувача з системою та дозволить автоматично формувати рекомендації щодо тренувань залежно від поставленої мети та рівня фізичної підготовки.

Метою даної роботи є розробка Telegram-бота для формування індивідуального розкладу тренувань та розрахунку добової норми калорій з урахуванням персональних параметрів користувача.

Для досягнення поставленої мети необхідно вирішити такі задачі:

– проаналізувати предметну область та визначити основні функціональні вимоги до програмного забезпечення;

– обрати середовище розробки та інструменти реалізації;

– реалізувати взаємодію користувача з ботом через Telegram Bot API;

– реалізувати функціонал вибору мети тренування (схуднення або набір маси);

– забезпечити можливість вибору рівня підготовки та групи м’язів;

– реалізувати алгоритм формування індивідуального розкладу тренувань;

– реалізувати калькулятор розрахунку добової норми калорій та БЖУ;

– передбачити перевірку коректності введених даних;

– провести тестування програмного забезпечення.

Розроблений програмний продукт повинен забезпечувати коректність обчислень, зручність використання та стабільність роботи під час взаємодії з користувачем.

1.1 Мета та завдання проєкту

Метою даного проєкту є створення програмного застосунку у вигляді чат-бота для Telegram, який дозволяє користувачу автоматично формувати персональний комплекс фізичних навантажень. Розробка спрямована на забезпечення швидкого доступу до готових тренувальних програм, що враховують мету користувача, рівень фізичної підготовки та вибрану групу м’язів.

Створений чат-бот повинен функціонувати як інструмент, який надає рекомендації з тренувань у зручній і зрозумілій формі, що дозволяє користувачу оперативно отримувати готові комплекси вправ без необхідності звертатися до довідників чи спеціалізованої літератури.

Основні завдання проєкту

Щоб досягти визначеної мети, необхідно виконати комплекс взаємопов’язаних задач, до яких належать:

1. Аналіз предметної області

- Вивчення принципів побудови тренувального процесу.
- Ознайомлення з особливостями складання програм для початкового та просунутого рівнів.
- Дослідження вимог до тренувань на різні групи м’язів: груди, спина, руки, ноги, плечі.
- Визначення структури стандартного тренувального заняття та тижневого плану.

2. Формування переліку вихідних даних

Необхідно визначити, які саме дані має надати користувач:

- мета тренування (схуднення або набір маси);

- рівень підготовки (початковий чи просунутий);
- конкретна група м’язів (для набору маси);
- бажаний тип плану (одне тренування чи повний тижневий цикл).

Це дозволяє організувати чітку структуру діалогу між користувачем і ботом.

3. Визначення вимог до чат-бота

Сюди входить:

- опис функціональних можливостей (обробка команд, відповіді, формування програм тренувань);
- опис інтерфейсу взаємодії бот–користувач;
- вимоги до збереження стану діалогу під час роботи;
- вимоги до коректності введених даних та повідомлень про помилки.

4. Проєктування архітектури

На цьому етапі необхідно визначити структуру програми, а саме:

- модуль взаємодії з Telegram Bot API через бібліотеку TgBot;
- модуль обробки текстових повідомлень;
- модуль збереження інформації про вибір користувача (мета, група м’язів, рівень);
- модуль формування текстових тренувальних програм.

Структура повинна забезпечувати стабільну роботу та можливість подальшого розширення.

5. Розробка алгоритму створення тренувальних програм

Необхідно підготувати алгоритм, який:

- приймає дані від користувача;
- здійснює перевірку правильності введених значень;
- обирає відповідний набір тренувальних вправ;
- формує готовий план тренування;
- надсилає результат у вигляді структурованого повідомлення.

Алгоритм повинен бути описаний у вигляді блок-схеми та у вигляді детального текстового опису.

6. Реалізація програмної частини

Це включає:

- написання коду мовою C++;
- використання бібліотеки TgBot для роботи з Telegram Bot API;
- реалізацію функцій обробки команд "/start" та відповідей на повідомлення;
- створення внутрішньої логіки формування тренувань на основі користувацьких даних;
- забезпечення стабільного циклу отримання та відправлення повідомлень.

7. Тестування чат-бота

Потрібно:

- перевірити працездатність усіх команд;
- протестувати реакції на різні варіанти користувацьких відповідей;
- перевірити коректність формування планів для всіх груп м’язів і всіх рівнів;
- перевірити поведінку програми в разі неправильних або неповних даних;
- забезпечити відсутність збоїв у тривалому режимі роботи.

8. Оцінка результатів та можливості подальшого розвитку

Після завершення реалізації необхідно:

- проаналізувати переваги створеного чат-бота;
- визначити обмеження реалізації;
- запропонувати шляхи вдосконалення, наприклад:
  - додавання нових тренувальних програм;
  - введення статистики прогресу;
  - можливість редагування попередніх виборів;
  - додавання бази відео-вправ;
  - інтеграція з іншими сервісами.

1.2 Аргументація необхідності створення чат-бота персональних фізичних навантажень

У сучасних умовах значна частина населення активно цікавиться питаннями фізичного здоров’я, правильної організації тренувального процесу та підтримання оптимальної фізичної форми. Проте для багатьох користувачів характерними є такі проблеми, як недостатність знань про тренувальні методики, відсутність професійної консультації та складність самостійного складання тренувальної програми. Саме тому створення чат-бота, який надає персоналізовані комплекси фізичних навантажень, є актуальним та обґрунтованим рішенням.

По-перше, чат-бот дозволяє зробити тренувальний процес доступним для широкого кола людей. Користувачеві не потрібно звертатися до тренера або витрачати багато часу на пошук інформації в Інтернеті. Достатньо ввести декілька параметрів — і він одразу отримує структурований та зрозумілий план тренувань.

По-друге, чат-бот забезпечує зручність використання у будь-який час і з будь-якого пристрою. Telegram є однією з найпоширеніших платформ, доступних на смартфонах, планшетах і комп’ютерах. Це робить розроблений бот універсальним інструментом, який може працювати без встановлення додаткового програмного забезпечення.

По-третє, чат-бот дозволяє уніфікувати та стандартизувати процес складання тренувальних програм. Усі вправи, повторення та рекомендації формуються відповідно до заздалегідь розроблених правил та алгоритмів. Це забезпечує стабільність і передбачуваність результатів, а також виключає помилки, які можуть виникати при ручному складанні планів.

По-четверте, система допомагає підвищити мотивацію користувачів до занять спортом, адже вони отримують чіткі інструкції, розгалужені тренувальні

плани і можуть самостійно обирати типи навантажень. Наявність готових програм робить процес початку тренувань простішим, а виконання вправ — більш упорядкованим.

По-п’яте, чат-бот значно економить час користувача, адже йому не потрібно опрацьовувати великі обсяги інформації або будувати тренувальний план вручну. Взаємодія з ботом відбувається швидко — відповіді формуються у межах лічених секунд.

По-шосте, розробка чат-бота актуальна також з технічної точки зору. Проєкт дозволяє застосувати на практиці навички програмування мовою C++, роботу з Telegram Bot API, створення алгоритмів, обробку текстових повідомлень та організацію логіки програмного продукту. Це робить чат-бот корисним навчальним прикладом.

По-сьоме, чат-бот може бути розширений у майбутньому, наприклад:

- додавання нових тренувальних програм,
- порад щодо відновлення,
- тематичних комплексів вправ,
- можливості фіксації прогресу або ведення історії тренувань.

Таким чином, створення чат-бота персональних фізичних навантажень є обґрунтованим рішенням, що відповідає реальним потребам користувачів та дозволяє застосувати сучасні технології у сфері здоров’я та спорту. Чат-бот забезпечує зручність, доступність, структурованість тренувального процесу та формує тренувальні комплекси на основі параметрів, які користувач надає самостійно.

1.3 Характеристика предметної області (тренування, навантаження, користувацькі профілі)

Предметна область проєкту пов’язана з організацією фізичних тренувань та формуванням індивідуальних програм фізичних навантажень. Для якісного

складання тренувальних комплексів необхідно розуміти основні поняття, які стосуються тренувального процесу, характеру фізичного навантаження та особливостей користувацьких профілів.

Тренування як основа фізичного розвитку

Тренування являє собою послідовно організовану діяльність, спрямовану на розвиток сили, витривалості, гнучкості або інших фізичних якостей. Тренувальний процес складається з кількох обов'язкових частин:

- підготовча частина (розминка);
- основний комплекс вправ;
- завершення тренування (розтягування або легкі вправи для відновлення).

У практиці фізичної підготовки виділяють різні типи тренувань: силові, кардіо, змішані, функціональні. Вибір конкретного типу залежить від мети, рівня підготовки користувача та бажаного результату.

Фізичні навантаження та їх класифікація

Фізичне навантаження — це робота, яку виконує організм під час вправ. Воно може бути розподілене за такими категоріями:

1. Силові навантаження — спрямовані на розвиток м'язових груп, збільшення м'язової сили та формування рельєфу.

Приклади: жим, тяги, присідання, випади.

2. Кардіонавантаження — спрямовані на покращення роботи серця та легень.

Приклади: біг, ходьба, стрибки, активні рухи високої інтенсивності.

3. Комплексні навантаження — поєднують силові та кардіоелементи, підходять для всебічного розвитку.

4. Локальні навантаження — спрямовані на конкретні м'язові групи:

a. груди,

b. спина,

c. ноги,

d. руки,

e. плечі,

f. прес.

Для кожного виду навантаження визначають кількість повторень, підходів, інтенсивність та тривалість.

Користувацькі профілі

Для формування персональних тренувальних програм важливо враховувати дані користувача. Користувацький профіль включає кілька основних характеристик:

1. Мета тренувань:

a. зменшення ваги;

b. збільшення м’язової маси;

c. розвиток витривалості;

d. підтримання фізичної форми.

2. Рівень фізичної підготовки:

a. початковий (мінімальний рівень досвіду);

b. просунутий (наявний тренувальний досвід).

3. Вибір м’язових груп:

У разі мети “набір маси” користувач часто орієнтується на розвиток окремих м’язових зон:

a. груди,

b. спина,

c. ноги,

d. руки,

e. плечі.

4. Формат тренування:

a. одноразове тренування (план на день);

b. структурований тижневий комплекс.

Кожен з цих параметрів впливає на побудову тренувальної програми та визначає, які вправи доцільно включити в комплекс.

Зв’язок предметної області з роботою чат-бота

Предметна область містить чітко визначені правила, структури та залежності. Саме це дозволяє використовувати її у вигляді звичайних алгоритмічних рішень, які можуть бути реалізовані в програмному коді.

У межах розробки чат-бота предметна область визначає:

- перелік вправ;
- їх розподіл за групами м’язів;
- логіку побудови тренувальних циклів;
- правила комбінування навантажень;
- черговість виконання вправ.

Це дозволяє формувати тренувальні програми відповідно до параметрів профілю користувача без використання будь-яких інтелектуальних технологій чи механізмів прогнозування.

РОЗДІЛ 2. ІНФОРМАЦІЙНИЙ ОГЛЯД

Інформаційний огляд присвячено вивченню матеріалів, які стосуються створення чат-ботів, організації тренувального процесу та побудови фізичних навантажень. Розділ охоплює аналіз джерел, технологій та існуючих рішень, що допомагають сформувати теоретичну основу для подальшої розробки програмного забезпечення.

2.1 Проблематика створення фітнес-систем та цифрових тренерів

Розробка фітнес-систем та цифрових тренерів є складним завданням, яке передбачає глибоке розуміння особливостей тренувального процесу, правил формування фізичних навантажень та специфіки роботи з користувачами. Попри зростаючий інтерес до цифрових рішень у сфері спорту, створення таких систем супроводжується низкою проблем і труднощів, які необхідно враховувати у процесі розробки.

1. Різноманітність фізичних навантажень та тренувальних методик

Однією з основних проблем є різноманітність тренувальних підходів, які застосовуються у спортивній практиці. Існує велика кількість видів навантажень — силові, кардіо, функціональні, статичні, вибухові, змішані. Кожен вид тренування має свої правила:

- кількість повторень;
- кількість підходів;
- рекомендовані перерви між вправами;
- порядок виконання;
- тривалість тренування.

Для цифрової системи потрібно врахувати ці особливості та подати їх у зрозумілій формі.

2. Індивідуальні відмінності між користувачами

Ще однією важливою проблемою є те, що кожен користувач має різні:

- цілі тренувань,
- рівень фізичної підготовки,
- стан здоров’я,
- досвід занять спортом,
- пріоритетні м’язові групи,
- можливості виконувати певні вправи.

Цифрова система повинна врахувати базові параметри користувача, щоб не запропонувати надмірно складне або, навпаки, занадто легке тренування.

3. Складність структуризації тренувального процесу

Фітнес-системи мають перетворювати спортивні знання на чіткі, формалізовані правила. Це непросто, оскільки тренування включає:

- вибір вправ,
- чергування навантажень,
- визначення темпу та черговості,
- побудову денного та тижневого розкладу.

У тренувальних програмах важливо забезпечити логічну послідовність і уникати надмірного навантаження на одну м’язову групу.

4. Відсутність універсального підходу

У сфері фізичної підготовки не існує універсальної програми, яка підійшла б усім. Те, що ефективно для однієї людини, може бути малоефективним або навіть небажаним для іншої. Тому цифровим тренерам необхідно формувати різні варіанти тренувальних планів залежно від параметрів користувача.

5. Вимоги до безпеки тренувань

Цифрові фітнес-системи повинні зважати на безпечність виконання вправ, оскільки неправильний підбір навантажень може призвести до травм. До основних вимог належать:

- уникнення надмірної інтенсивності для новачків;
- включення вправ на розігрів та завершення тренування;
- збалансованість навантажень;
- рекомендації щодо правильної техніки.

Складність полягає в тому, що цифрова система має надати достатньо інформації користувачу без можливості контролю виконання вправ.

6. Лімітації інтерфейсу цифрових тренерів

У чат-ботів та подібних систем є обмеження:

- невеликі текстові повідомлення,
- лінійна структура діалогу,
- неможливість демонструвати складні рухи,
- відсутність можливості оцінювати техніку виконання.

Тому головне завдання — подавати інформацію максимально просто, доступно і структуровано.

7. Підтримання актуальності та повноти тренувальних комплексів

Тренувальні методики постійно змінюються та оновлюються. Для цифрової системи важливо:

- підтримувати актуальність вправ;
- уникати застарілих рекомендацій;
- забезпечувати різноманіття навантажень.

Інша проблема — обмеження обсягу інформації: система повинна містити достатній набір вправ, але при цьому залишатися простою для використання.

8. Узгодження технічної частини із тренувальною логікою

Цифровий тренер повинен одночасно враховувати спортивну методику і технічні вимоги програмування. Це стосується:

- структури програми,

- побудови діалогу з користувачем,
- обробки введених даних,
- формування відповідей,
- уникнення помилок у роботі.

Поєднання цих аспектів становить окрему складність під час розробки.

Проблематика створення фітнес-систем і цифрових тренерів полягає у поєднанні спортивної методики з чіткими правилами, які можуть бути реалізовані у програмному забезпеченні. Необхідно враховувати різні види тренувань, індивідуальні параметри користувачів, обмеження інтерфейсу й вимоги до безпеки. Саме ці фактори формують ключові виклики, які слід подолати під час створення чат-бота для тренувальних програм.

2.2 Огляд програмних засобів для створення чат-ботів

Створення чат-ботів є поширеним напрямом у сучасному програмуванні, оскільки такі системи дозволяють автоматизувати взаємодію з користувачами в різних сферах — від обслуговування клієнтів до освітніх чи спортивних застосунків. Для реалізації чат-бота розробники можуть використовувати різноманітні програмні засоби, бібліотеки та технології. У цьому підрозділі розглядаються найбільш поширені інструменти, які застосовуються для створення чат-ботів у Telegram та інших месенджерах.

1. Telegram Bot API

Telegram Bot API є офіційним засобом взаємодії з платформою Telegram. Він забезпечує можливість:

- приймати текстові повідомлення,
- надсилати відповіді,
- обробляти команди,
- працювати з клавіатурами,
- створювати інтерактивні меню.

Це один із найбільш доступних та простих у використанні інструментів, оскільки Telegram надає детальну документацію та стабільну роботу API. Telegram Bot API не залежить від конкретної мови програмування, що дозволяє використовувати будь-яку з доступних мов.

2. Бібліотеки для створення ботів

Оскільки Telegram Bot API є універсальним, для різних мов програмування існують спеціальні бібліотеки, які спрощують роботу з ним. Найпоширеніші з них:

• TgBot (C++)

Бібліотека, яка дозволяє реалізувати Telegram-бота мовою C++. Вона підтримує:

- обробку команд;
- читання текстових повідомлень;
- надсилання відповідей користувачам;
- довготривалі підключення через Long Polling.

TgBot є оптимальним вибором для розробників, які використовують C++.

• python-telegram-bot (Python)

Популярна бібліотека для мови Python. Забезпечує зручний спосіб створення простих і складних ботів.

• node-telegram-bot-api (JavaScript)

Широко використовувана бібліотека для створення ботів у середовищі Node.js. Має простий синтаксис і підтримує всі можливості API.

• TeleBot / Aiogram для Python

Також надають функції для роботи з Telegram, включаючи маршрутизацію повідомлень та обробку подій.

• .NET Telegram.Bot (C#)

Сумісна з екосистемою .NET, підходить для розробників, які використовують C#.

Хоча для проєкту обрано саме TgBot, огляд інших бібліотек дозволяє оцінити їхні можливості та особливості.

3. Середовища розробки

Для написання та налагодження чат-бота використовують сучасні IDE та текстові редактори:

• Visual Studio 2022

Використовується у даному проєкті. Дозволяє:

- працювати з C++;
- встановлювати розширення для автодоповнення коду;
- запускати компіляцію;
- налагоджувати програму;
- працювати з файловою структурою проєкту.

• Visual Studio 2022

Підходить для розробки на C++, C# та інших мовах. Має розширені можливості налагодження.

• CLion (JetBrains)

Професійне середовище для C++ з розвинутими інструментами аналізу коду.

• Code::Blocks та Dev-C++

Полегшені середовища, які можуть використовуватися для навчальних або невеликих проєктів.

4. Додаткові засоби підтримки

Під час розробки чат-ботів можуть застосовуватися й інші компоненти:

- Менеджери пакетів (vcpkg, Conan) для встановлення бібліотек.
- Git-системи для контролю версій — GitHub, GitLab.
- Формати даних (JSON, TXT) для зберігання текстових шаблонів або службової інформації.
- Методи підключення — Long Polling або Webhook (для серверних реалізацій).

У даному проєкті використовується метод Long Polling, оскільки він не потребує окремого сервера.

Проведений інформаційний огляд дав можливість комплексно дослідити теоретичні та практичні аспекти, пов’язані з розробкою чат-ботів та формуванням тренувальних програм. Було розглянуто основні підходи до побудови фізичних навантажень, класифікацію вправ, принципи організації тренувальних циклів, а також проаналізовано існуючі цифрові рішення у сфері фітнесу.

Окрему увагу приділено програмним засобам, які застосовуються для створення чат-ботів. Проаналізовано Telegram Bot API, бібліотеку TgBot для мови C++ та основні можливості середовища розробки Visual Studio 2022. Це дозволило визначити оптимальний набір інструментів для реалізації поставленої задачі.

У межах розділу також були окреслені проблеми, характерні для створення фітнес-систем та цифрових тренерів: різноманітність тренувальних методик, індивідуальність потреб користувачів, вимоги до безпеки виконання вправ, а також технічні обмеження платформи. Знання цих аспектів створює основу для правильної побудови логіки чат-бота та структури тренувальних програм.

Загалом інформаційний огляд сформував теоретичну базу, необхідну для подальшого проектування та розробки програмного забезпечення, і дозволив визначити ключові вимоги та підходи, які будуть використані у наступних розділах роботи.

РОЗДІЛ 3. ТЕОРЕТИЧНА ЧАСТИНА

Теоретична частина роботи містить основні поняття, моделі та закономірності, які необхідні для коректного проектування та реалізації чат-бота. У розділі розглядаються особливості структури тренувальних програм, принципи побудови фізичних навантажень, алгоритмічні підходи до їх формування, а також основи взаємодії програмного забезпечення з користувачем у форматі діалогових систем.

3.1 Розробка структури проєкту

Розробка структури проєкту є важливим етапом створення чат-бота, оскільки визначає логіку його роботи, організацію програмного коду, взаємодію між компонентами та порядок обробки даних. Чітка структура дозволяє забезпечити зрозумілість коду, стабільність роботи бота та можливість подальшої модифікації або розширення функціональності.

У цьому підрозділі описуються основні елементи проєкту, їх призначення та взаємозв’язки між ними.

1. Основні компоненти програмного забезпечення

Структура чат-бота складається з кількох логічних блоків, кожен з яких виконує свою роль:

1.1. Модуль взаємодії з Telegram

Цей модуль забезпечує:

- підключення до Telegram через токен;
- отримання повідомлень від користувача;
- відправлення відповідей;
- обробку команд (/start та інші).

У проєкті за роботу з цим модулем відповідає бібліотека TgBot, яка містить необхідні класи та методи для обміну повідомленнями.

1.2. Модуль обробки повідомлень

Цей блок визначає, як програма реагує на введення користувача. Основні функції:

- аналіз тексту, який надсилає користувач;
- визначення етапу діалогу;
- збереження тимчасових даних (мета, рівень, група м’язів);
- передача даних у генератор тренувань.

Функціонально цей модуль відповідає за логіку діалогу між користувачем і чат-ботом.

1.3. Модуль генерації тренувальних програм

Окремий блок проєкту відповідає за створення тренувальних планів. Він складається з:

- набору правил;
- функцій формування вправ;
- шаблонів для різних цілей;
- готових тренувальних програм.

Головна функція цього модуля — generateSchedule, яка будує комплекс вправ залежно від параметрів користувача.

1.4. Модуль зберігання тимчасових даних

Чат-бот веде діалог у кілька етапів, тому важливо зберігати проміжні дані, поки користувач не надасть повний набір параметрів.У проєкті для цього застосовуються колекції типу:

- map<int64_t, string> — для збереження мети;
- map<int64_t, string> — для запису вибраної групи м’язів.

Цей модуль дозволяє розрізняти користувачів та підтримувати діалоги з кількома людьми одночасно.

1.5. Модуль запуску та циклу обробки подій

Чат-бот працює безперервно. Для цього використовується цикл Long Polling, який:

- постійно перевіряє наявність нових повідомлень;
- передає їх у модуль обробки;
- підтримує роботу бота в режимі реального часу.

Цей блок є основною «точкою входу» програми.

2. Логічна схема взаємодії модулів

Структура проєкту організована так, щоб усі модулі працювали узгоджено:

1. Telegram API

↓ (отримання повідомлення)

2. Модуль обробки повідомлень

↓ (визначення параметрів)

3. Модуль збереження тимчасових даних

↓ (нагадає, що вже введено)

4. Модуль генерації тренувань

↓ (готовий текст)

5. Відправлення результату користувачу через Telegram

Завдяки такій послідовності програма залишається простою та логічно впорядкованою.

3. Вимоги до структури проєкту

Під час розробки структури були враховані такі вимоги:

- Чіткість та зрозумілість — кожен модуль має окрему функцію.
- Можливість розширення — до системи легко додати нові типи тренувань або команди.
- Стабільність роботи — програма не повинна зависати в процесі діалогу.
- Зручність підтримки — структура повинна бути простою для змін.

4. Переваги розробленої структури

Описана структура має низку переваг:

- дозволяє легко організувати логіку діалогу;
- забезпечує правильний поділ відповідальності між частинами коду;
- спрощує читання та налагодження програми;
- дає можливість швидко додавати нові функції або розширювати існуючі;
- підходить для програмного забезпечення, яке працює з великою кількістю користувачів.

Розроблена структура проєкту дозволяє організувати роботу чат-бота зрозумілим і логічним чином. Кожен модуль виконує свою роль, а взаємодія між ними відбувається послідовно та впорядковано. Такий підхід забезпечує стабільність роботи програми та створює основу для її подальшого розвитку.

3.2 Обґрунтування вибору технологій для розробки чат-бота

Для створення чат-бота було обрано мову програмування C++, оскільки вона забезпечує високу продуктивність, контроль над роботою програми та стабільність під час обробки користувацьких повідомлень. Крім того, мова дозволяє легко інтегрувати зовнішні бібліотеки, необхідні для роботи з мережевими протоколами.

Для взаємодії з Telegram було використано бібліотеку TgBot, яка підтримує отримання і відправлення повідомлень, роботу з командами та обробку текстових відповідей. Ця бібліотека є надійною та достатньо гнучкою для реалізації логіки чат-бота будь-якої складності.

Застосування HTTP-запитів, обробки JSON-структур та стандартних контейнерів C++ (таких як map) забезпечує правильну організацію діалогу з користувачем і можливість зберігати тимчасові дані між кроками.

Обрані технології дозволяють створити ефективний, швидкодіючий та розширюваний чат-бот, який формує персональні тренувальні програми відповідно до введених параметрів.

3.3 Обґрунтування вибору середовища розробки

Для створення чат-бота було обрано середовище розробки Visual Studio 2022. Вибір цього програмного інструменту зумовлений його функціональністю, зручністю використання та відповідністю вимогам, що виникають при розробці застосунків мовою C++. У цьому підрозділі наведено основні причини, які підтверджують доцільність використання саме цього середовища.

1. Легкість встановлення та налаштування

Visual Studio 2022 має простий процес встановлення, не потребує складної конфігурації та підтримує всі необхідні компоненти для роботи з C++. Після встановлення достатньо додати розширення:

- C/C++ Extension Pack,
- C++ Intellisense,
- Debugger for C++,

що дозволяє повноцінно працювати над проєктом.

2. Зручна структура інтерфейсу

Середовище має зрозумілий інтерфейс, який складається з:

- панелі файлів,
- області редагування,
- терміналу,
- налагоджувача,
- панелі розширень.

Така структура дає можливість швидко перемикатися між файлами, переглядати структуру проєкту та керувати процесом розробки.

3. Вбудований термінал

Visual Studio 2022 дозволяє запускати компіляцію та виконання C++ програм безпосередньо вбудованим терміналом. Це значно полегшує тестування коду та забезпечує швидкий доступ до системних команд.

4. Підтримка налагодження (debugging)

Однією з важливих переваг є наявність гнучкого механізму налагодження:

- встановлення точок зупину,
- перегляд змінних,
- покрокове виконання коду,
- аналіз значень під час роботи програми.

Налагоджувач дає змогу швидко знаходити помилки та виявляти неправильну логіку в програмі.

5. Підтримка роботи з Git

Visual Studio Code має вбудовані інструменти для роботи з Git, що дозволяє:

- відстежувати зміни у файлах,
- створювати коміти,
- працювати з гілками,
- синхронізувати проєкт із GitHub або GitLab.

Це особливо зручно для зберігання резервних копій проєкту або командної розробки.

6. Можливість використання розширень

Однією з ключових переваг є підтримка великої кількості розширень. Вони дозволяють:

- покращити автодоповнення,
- форматувати код,
- працювати з різними конфігураціями компіляторів,
- використовувати інструменти аналізу структури коду.

Наявність розширень дає можливість адаптувати середовище під конкретні потреби проєкту.

7. Невисокі вимоги до ресурсів

Visual Studio 2022 працює швидко навіть на комп’ютерах із середніми характеристиками.

Це спрощує його використання в навчальних цілях або в ситуаціях, коли немає можливості працювати на потужних робочих станціях.

8. Портативність та кросплатформеність

Середовище може бути встановлене на:

- Windows,
- Linux,
- macOS.

Це дозволяє працювати над проєктом у будь-якій операційній системі, що робить розробку гнучкою та мобільною.

Visual Studio 2022 було обрано як основне середовище для розробки чат-бота завдяки його функціональності, простоті використання та широким можливостям налаштування. Підтримка роботи з C++, наявність налагоджувача, інтеграція з Git [5], велика кількість розширень і зручність інтерфейсу роблять це середовище оптимальним вибором для розробки програм подібного типу.

3.4 Алгоритм роботи програмного забезпечення

Алгоритм роботи чат-бота визначає послідовність дій, які виконує програма від моменту запуску до формування та відправлення тренувального комплексу користувачу. Чітке структурування алгоритму дозволяє забезпечити стабільну й передбачувану роботу системи. Нижче наведена покрокова схема роботи чат-бота.

1. Ініціалізація програми

Після запуску:

1. програма завантажує необхідні бібліотеки;
2. створюється об’єкт бота з використанням токена доступу;
3. ініціалізуються структури для збереження даних користувачів (map<int64_t, string>);
4. встановлюються обробники команд і повідомлень.

Це дозволяє підготувати бот до роботи з будь-яким користувачем.

2. Отримання команди /start

Перший крок взаємодії — отримання команди: /start
У відповідь бот надсилає вступне повідомлення з описом можливостей та запитом про вибір мети тренування. На цьому етапі програма не переходить до наступних кроків, поки користувач не введе свою мету.

3. Збереження мети користувача

Після того як користувач вводить:

- «схуднення» або
- «набір маси»

бот зберігає це значення у колекції userGoal[chatId].

Далі алгоритм розгалужується:

- якщо мета → «схуднення», бот запитує рівень підготовки;
- якщо мета → «набір маси», перед цим бот запитує групу м’язів.

4. Вибір групи м’язів (для набору маси)

Якщо користувач обрав «набір маси», бот пропонує вибрати:

- груди
- спина
- руки
- ноги
- плечі
- або повний «розклад на тиждень»

Вибір зберігається у userMuscle[chatId].

Після цього користувач переходить до наступного кроку.

5. Отримання рівня підготовки

Бот запитує одне з двох значень:

- початковий
- просунутий

Отримане значення використовується під час формування тренувальної програми.

6. Формування параметрів тренування

Після того як користувач надає:

- мету,
- групу м’язів (за потреби),
- рівень підготовки,

бот передає отримані параметри у функцію: generateSchedule(goal, level, muscleGroup)

Функція підбирає відповідні вправи та створює готовий текст тренувального плану.

7. Генерація готового тренувального комплексу

У функції generateSchedule виконується:

1. перевірка значень параметрів;
2. вибір шаблону тренування, який відповідає меті користувача;
3. формування списку вправ;
4. підготовка структурованого тексту з тренуванням;
5. повернення готового результату у вигляді рядка.

8. Надсилання результату користувачу

Після формування плану бот надсилає його у чат.

Програма форматує повідомлення так, щоб:

- воно було читабельним;
- містило дні тижня (для повних розкладів);

- включало позначення вправ та повторень.

9. Очищення тимчасових даних

Після відправлення відповіді бот видаляє дані про користувача:

userGoal.erase(chatId);
userMuscle.erase(chatId);

Це робиться для того, щоб наступний діалог починався з нуля і не змішувався з попереднім.

10. Продовження роботи у циклі Long Polling

Бот переходить у стан очікування нових повідомлень.

Long Polling забезпечує:

- безперервне отримання нових текстів,
- миттєве реагування на команди,
- стабільність роботи бота без серверної інфраструктури.

Цикл працює без зупинки, поки програма не буде завершена вручну.

Алгоритм роботи чат-бота складається з послідовних етапів: отримання даних, їх збереження, вибір параметрів, формування тренувальної програми та відправлення результату. Завдяки чіткому алгоритму програмне забезпечення працює стабільно, обробляє кілька діалогів одночасно та видає користувачу правильний тренувальний комплекс.

3.5 Блок-схема роботи ПЗ

Блок-схема відображає повну логіку функціонування Telegram-бота (рис.2.3) та послідовність обробки запитів користувача. Алгоритм передбачає дві основні гілки роботи: формування розкладу тренувань та розрахунок добової норми калорій[20].

1. Початок роботи програми.

Після запуску здійснюється ініціалізація бота, підключення до Telegram API та створення структур для збереження тимчасових даних користувача.

2. Очікування повідомлення.

Програма переходить у режим постійного прослуховування вхідних повідомлень.

3. Отримання команди /start.

Після надходження команди бот відображає головне меню з вибором режиму роботи.

4. Вибір режиму роботи.

Користувачу пропонується обрати один із варіантів:

– «Розклад тренувань»

– «Калькулятор калорій»

5. Гілка «Розклад тренувань».

5.1. Вибір мети тренування: «Схуднення» або «Набір маси».

5.2. Якщо обрано «Схуднення», виконується запит рівня підготовки.

5.3. Якщо обрано «Набір маси», додатково здійснюється вибір групи м’язів, після чого — вибір рівня підготовки.

5.4. На основі введених параметрів формується тренувальний план.

5.5. Результат надсилається користувачу у вигляді текстового повідомлення.

5.6. Тимчасові дані очищуються, після чого система повертається в режим очікування.

6. Гілка «Калькулятор калорій».

6.1. Користувач вводить необхідні параметри: вагу, зріст, вік, стать та рівень фізичної активності.

6.2. Виконується перевірка коректності введених даних.

6.3. У разі помилки виводиться повідомлення та відбувається повернення до режиму очікування.

6.4. Якщо дані введено коректно, виконується розрахунок базового обміну речовин (BMR), добової норми калорій та співвідношення білків, жирів і вуглеводів.

6.5. Результат надсилається користувачу, після чого здійснюється очищення стану та повернення до режиму очікування.

Таким чином, блок-схема демонструє повний цикл роботи програми, включаючи обробку помилок, формування результату та повторний перехід у режим очікування нових повідомлень.

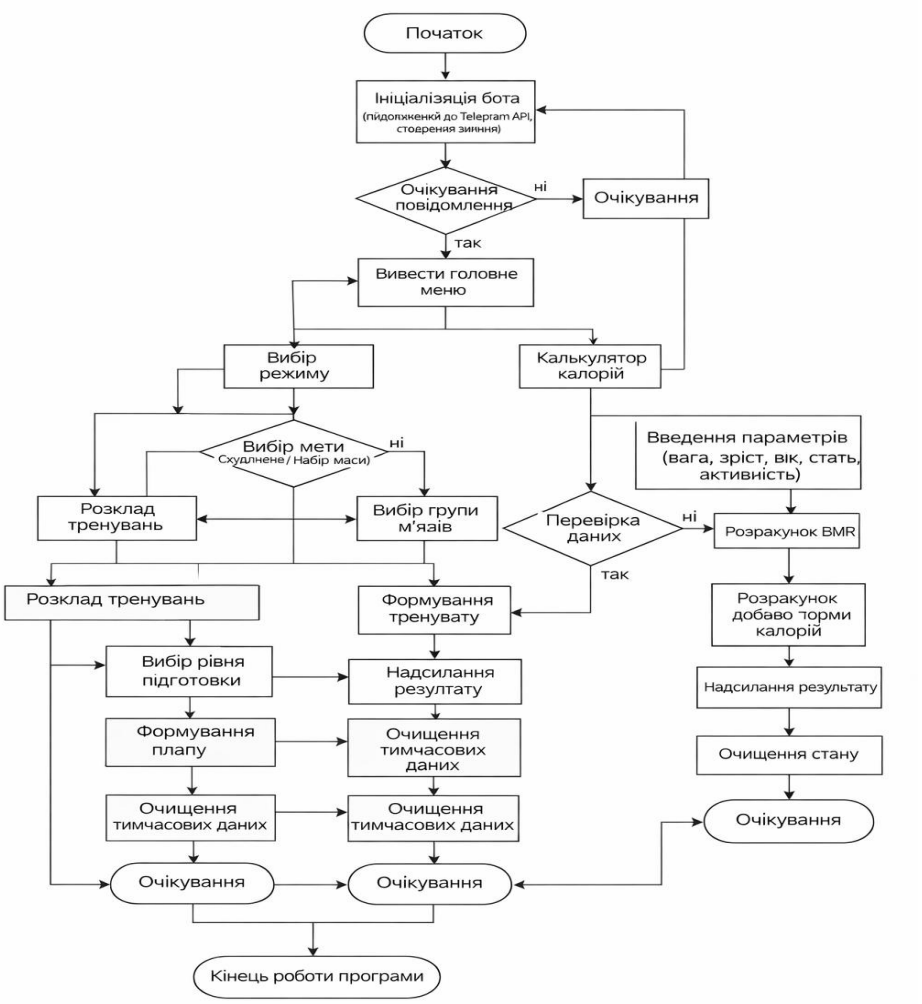

Рисунок 2.3 – Блок-схема алгоритму роботи Telegram-бота

РОЗДІЛ 4. ПРАКТИЧНА ЧАСТИНА

4.1 Вибір інструментальних засобів

Для створення чат-бота було використано мову програмування C++, оскільки вона забезпечує високу швидкодію та дає можливість точно контролювати роботу програми. Також мова має широкий вибір бібліотек для мережевої взаємодії й достатньо гнучка для реалізації логіки обробки користувацьких запитів.

Для взаємодії з Telegram Bot API використано бібліотеку TgBot, яка дозволяє опрацьовувати вхідні повідомлення, надсилати відповіді, зберігати дані користувачів та працювати з командами. Бібліотека підтримує обробку повідомлень у режимі Long Polling, що забезпечує стабільну роботу чат-бота без додаткового серверного обладнання.

Розробка здійснювалася у середовищі Visual Studio 2022, яке було обране через підтримку C++, зручні інструменти для налагодження, можливість роботи з терміналом та розширюваність завдяки плагінам. Використання Git дозволило зберігати проміжні версії проєкту.

4.2 Структура програмного забезпечення

Розроблене ПЗ складається з декількох функціональних модулів:

1. Модуль обробки команд

Відповідає за реагування на команду /start та формування початкових повідомлень.

2. Модуль діалогу з користувачем

Обробляє відповіді користувача та зберігає проміжні параметри:

- мету тренувань,
- групу м'язів (для набору маси),
- рівень підготовки.

Для цього використовуються колекції типу:

```
map<int64_t, string> userGoal;
map<int64_t, string> userMuscle;
```

3. Модуль генерації тренувальних програм

Функція:

```
generateSchedule(const string& goal, const string& level, const string& muscleGroup = ""
```

створює готовий текст тренування на основі переданих параметрів.

4. Модуль взаємодії з Telegram API

Виконує:

- прийом повідомлень,
- відправку відповідей,
- обробку помилок під час роботи.
-

4.3 Реалізація логіки обробки повідомлень

Основна логіка роботи чат-бота побудована на використанні обробника подій:

```
bot.getEvents().onAnyMessage([&bot, &userGoal, &userMuscle](Message::Ptr message)
```

Ця функція дозволяє обробляти повідомлення у визначеній послідовності:

1. Перевірка, чи користувач ще не обрав мету.
2. Якщо мету обрано — очікування вибору групи м’язів.
3. Далі — запит рівня підготовки.
4. Після отримання всіх параметрів викликається функція генерації тренування.

5. Готовий результат надсилається користувачу.

Після завершення діалогу тимчасові дані очищуються:

```
userGoal;
userMuscle;
```

Це дозволяє уникнути змішування інформації різних користувачів.

4.4 Реалізація функції генерації тренувань

Функція generateSchedule аналізує введені параметри та формує один із варіантів тренувальної програми.

Передбачено кілька сценаріїв:

- тренування для схуднення (початковий / просунутий рівень);
- тренування для набору маси з урахуванням обраної групи м’язів;
- повний тижневий розклад.

За кожною гілкою прописано текстові шаблони, які повертаються користувачеві.

Наприклад (фрагмент):

```
if (goal == "схуднення") {
    if (level == "початковий") return
        "🔥 Схуднення — початковий рівень:\n\n"
        "📅 Понеділок:\n"
        "🤸 Кардіо: 20 хв\n"
```

Цей підхід забезпечує керованість та можливість подальшого розширення функціоналу.

4.5 Тестування роботи чат-бота

Після реалізації програмного забезпечення було проведено тестування шляхом взаємодії з ботом у Telegram.

Перевірялося:

- коректність обробки команди /start;

- правильність збереження мети та групи м’язів;
- відповідність тренувальних програм обраним параметрам;
- стабільність роботи у режимі довгого опитування (Long Polling);
- відсутність збоїв під час одночасного використання ботом кількома користувачами.

У результаті тестування встановлено:

- бот формує тренувальні комплекси згідно з введеними параметрами;
- помилкові введення обробляються коректно;
- програма працює стабільно та не потребує додаткових системних ресурсів.

4.6 Реалізація функціоналу “Калькулятор калорій”

Програма працює в режимі «Калькулятор калорій». Користувач вводить такі дані:

- вага
- зріст
- вік
- стать
- рівень фізичної активності

Дані зчитуються через istringstream та розподіляються по змінних.

Далі відбувається розрахунок базального метаболізму (BMR) за формулою Міффліна–Сан Жеора:

- для чоловіків:

$10 \times \text{вага} + 6.25 \times \text{зріст} - 5 \times \text{вік} + 5$

- для жінок:

$10 \times \text{вага} + 6.25 \times \text{зріст} - 5 \times \text{вік} - 161$

Після цього застосовується коефіцієнт фізичної активності, що дозволяє визначити добову норму калорій.

Таким чином, реалізований алгоритм автоматично обробляє введені дані та виконує розрахунок відповідно до сучасної формули визначення енергетичних потреб організму.

4.7 Інструкція роботи з ботом

У практичній частині представлено приклади роботи розробленого Telegram-бота. Програма взаємодіє з користувачем у діалоговому форматі та поетапно пропонує вибір необхідних параметрів.

Після запуску бота (/start) користувач отримує стартове повідомлення та меню з доступними функціями. Далі, залежно від обраного режиму, система або формує план тренування, або виконує розрахунок добової норми калорій.

Алгоритм роботи побудований таким чином, щоб користувач послідовно вводив необхідні дані. На основі отриманої інформації програма обробляє запит та повертає відповідний результат.

Нижче наведено фрагменти роботи програми з прикладами виконання основних функцій.

1. Для початку роботи з програмою користувач запускає бота в середовищі Telegram та вводить команду **/start** (рис. 4.1). Після цього відображається головне меню з вибором режиму роботи: формування розкладу тренувань або калькулятор калорій.

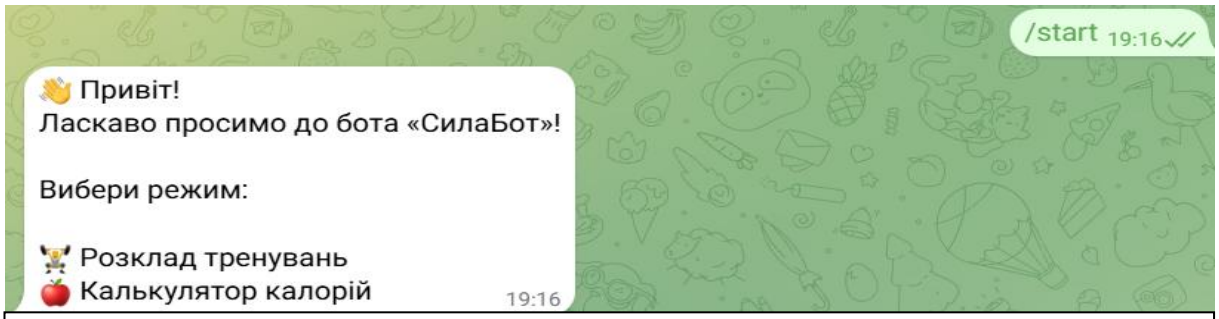

Рисунок 4.1 - Головне меню бота

Після вибору режиму «Розклад тренувань» користувач обирає мету тренування: схуднення або набір маси (рис. 4.2). Далі необхідно вказати рівень підготовки (початковий або просунутий). Якщо обрано режим набору маси, додатково пропонується вибір групи м’язів.

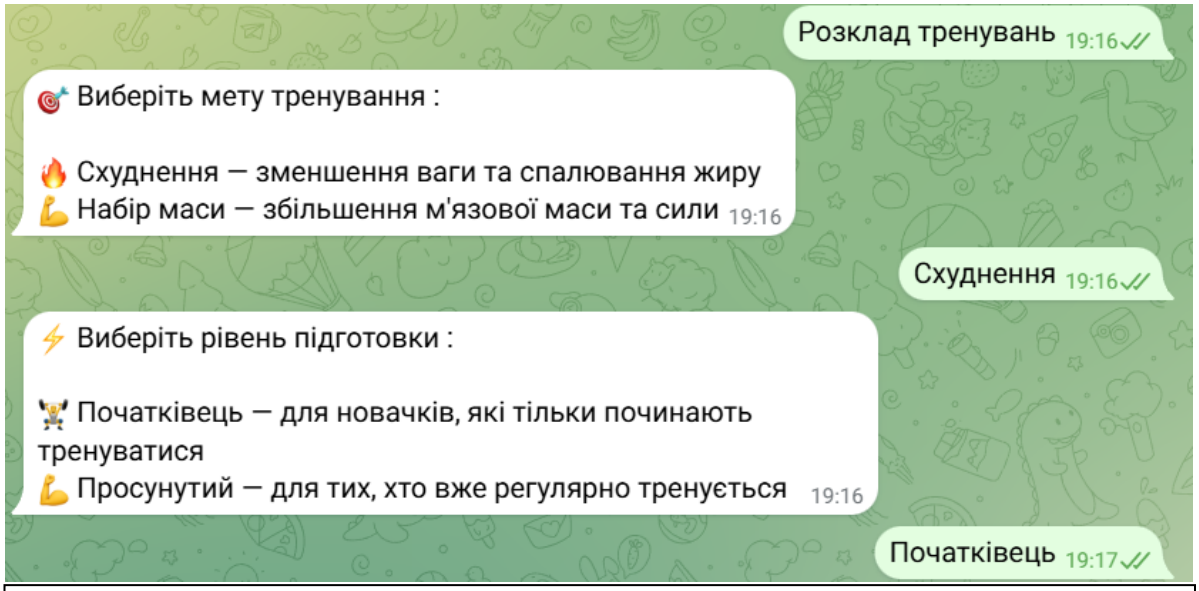

Рисунок 4.2 – Вибір параметрів тренування

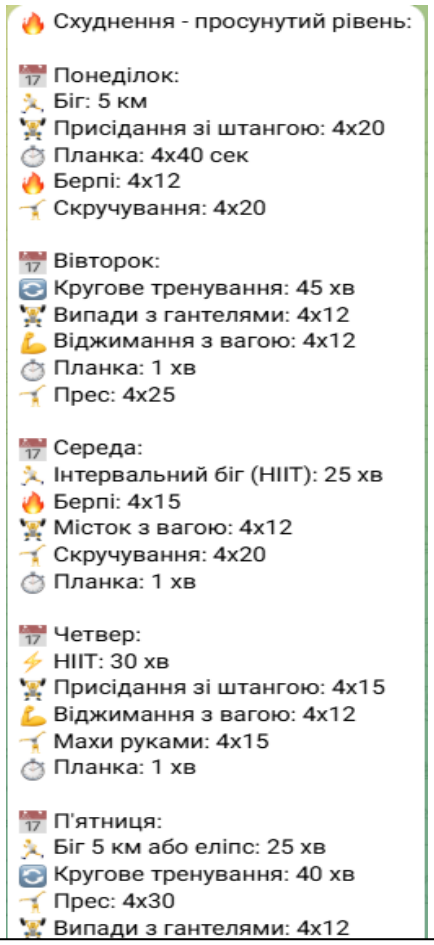

Рисунок 4.3 – Приклад сформованого тренувального плану

У разі вибору режиму «Калькулятор калорій» користувач вводить персональні дані: вагу, зріст, вік, стать та рівень фізичної активності. На основі цих даних виконується розрахунок базового обміну речовин та добової норми калорій. Результат відображається у чаті.

Після завершення роботи з вибраним режимом бот переходить у стан очікування нових команд.

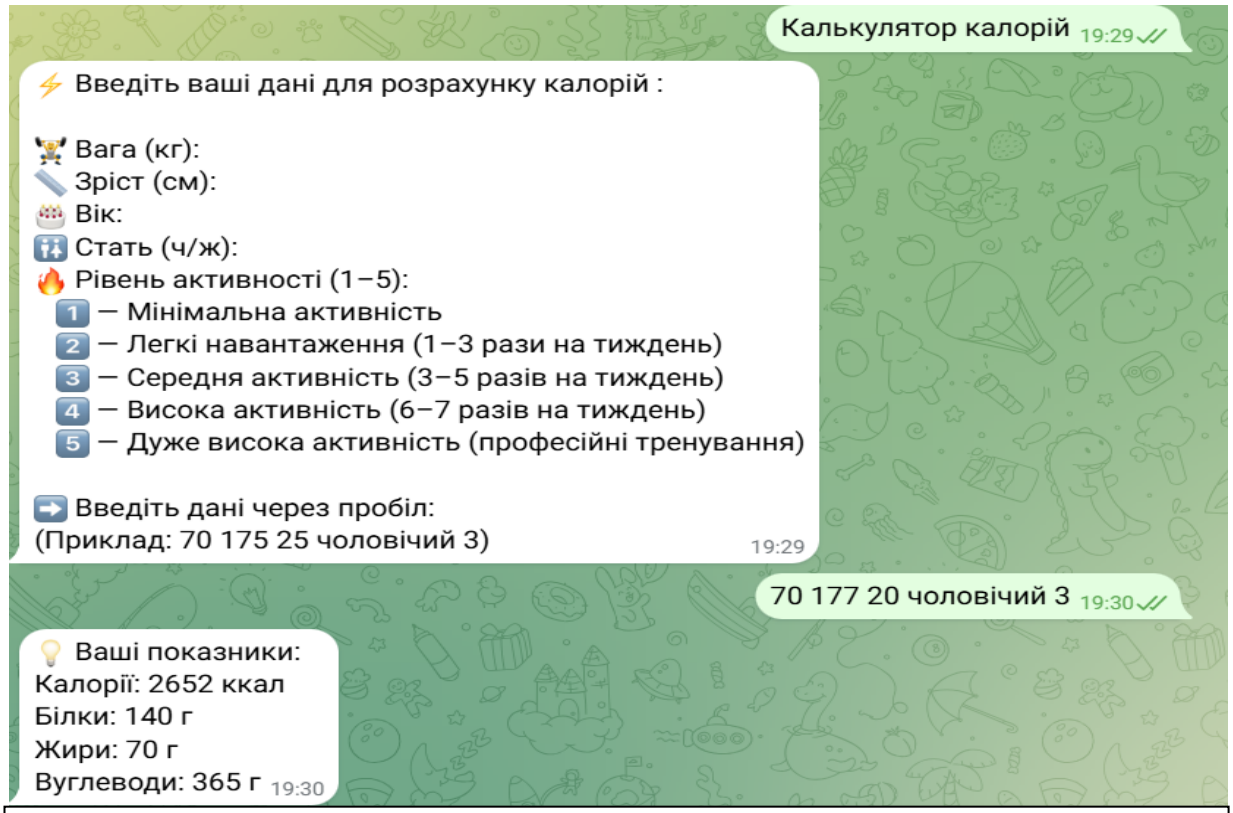

Рисунок 4.4 – Робота калькулятора калорій у Telegram-боті

4.8 Висновки

У процесі практичної реалізації було створено повністю робочий чат-бот, який здатний формувати персональні тренувальні програми відповідно до обраної мети, рівня та групи м’язів.

ПЗ виконане відповідно до поставлених вимог та може бути легко розширене новими функціями.

ВИСНОВОК

У процесі написання дипломної роботи було розроблено бота для Telegram, який створює індивідуальні програми тренувань і розраховує добову калорійність раціону. Під час роботи було проаналізовано тематичну область, визначено основні вимоги до програмного забезпечення та обрано інструменти реалізації.

У практичній частині основна функціональність бота була реалізована на мові програмування C++ з використанням API Telegram Bot. Програма дозволяє користувачеві вибрати мету тренувань, рівень підготовки та отримати готовий графік занять. Окремо було реалізовано калькулятор калорій, який розраховує добові енергетичні потреби організму та співвідношення білків, жирів і вуглеводів на основі введених параметрів.

Під час розробки було реалізовано перевірку введених даних та забезпечено стабільну роботу програми. Тестування показало, що бот правильно реагує на дії користувача та генерує відповідні результати.

Таким чином, мета роботи була досягнута, і розроблений програмний продукт може бути використаний як допоміжний інструмент для планування фізичних навантажень і контролю харчування.У майбутньому можна розширити функціонал, зокрема, додавши можливість зберігати історію тренувань або інтегрувати з іншими сервісами.

Розроблений чат-бот «СилаБот» доцільно рекомендувати для використання як інструмент формування індивідуального розкладу тренувань залежно від мети користувача та рівня його фізичної підготовки. Інтерфейс взаємодії побудований у формі покрокового вибору режиму роботи, що забезпечує зрозумілу та послідовну взаємодію користувача з програмою.

Система передбачає вибір одного з основних режимів: формування розкладу тренувань або розрахунок добової норми калорій. Такий підхід дозволяє розмежувати функціональні можливості програмного продукту та зробити його використання більш структурованим.

Особливістю реалізації є механізм перевірки коректності введених даних. У випадку помилкового або неповного введення бот повідомляє користувача про невірну комбінацію параметрів та пропонує можливі варіанти вибору. Це підвищує зручність використання та зменшує ймовірність неправильного формування тренувального плану.

Рекомендується використовувати даний програмний продукт для попереднього планування фізичних навантажень початківцями, які не мають достатнього досвіду самостійного складання програм тренувань. Наявність варіантів «початківець» та «просунутий» дозволяє адаптувати навантаження до рівня підготовки користувача.

Функціонал калькулятора калорій доцільно застосовувати як допоміжний інструмент при досягненні поставленої мети (схуднення або набір маси), оскільки поєднання контролю фізичної активності та харчування забезпечує більш ефективний результат.

З метою подальшого розвитку програмного забезпечення доцільно:

- розширити перелік можливих комбінацій тренувальних програм;
- додати збереження історії вибору користувача;
- передбачити можливість формування тижневого або місячного плану тренувань;
- реалізувати більш гнучку систему перевірки введених параметрів;
- додати автоматичні рекомендації при виникненні помилок введення.

Таким чином, розроблений чат-бот має практичну спрямованість, зручний алгоритм взаємодії та може бути використаний у реальних умовах для планування фізичної активності.

СПИСОК ЛІТЕРАТУРИ

1. Kernighan B., Ritchie D. *The C Programming Language*. Prentice Hall, 2022.
2. Stroustrup B. *The C++ Programming Language*. Addison-Wesley, 2021.
3. Josuttis N. *The C++ Standard Library: A Tutorial and Reference*. Addison-Wesley, 2019.
4. Офіційна документація Telegram Bot API. — Режим доступу: https://core.telegram.org/bots/api
5. Офіційний репозиторій бібліотеки TgBot для C++. — Режим доступу: https://github.com/reo7sp/tgbot-cpp
6. Visual Studio Code Documentation. — Режим доступу: https://code.visualstudio.com/docs
7. Bompa T., Buzzichelli C. *Periodization Training for Sports*. Human Kinetics, 2018.
8. Delavier F. *Strength Training Anatomy*. Human Kinetics, 2016.
9. Фізичне виховання: Підручник / ред. Атаманюк О. — К.: Академія, 2020.
10. Методичні рекомендації з організації фізичних тренувань / МОН України, 2021.
11. Сучасні підходи до побудови тренувальних програм: науково-методична стаття. — «Фізична культура і спорт», №3, 2020.
12. Стаття: «Особливості розробки програмного забезпечення для чат-ботів». — Журнал «Комп’ютерні системи та технології», №4, 2022.
13. Deitel P. *C++ How to Program*. Pearson, 2020.
14. Sharp J. *C++ Programming for Beginners*. Independently Published, 2021.
15. Довідник програміста C++ / Колектив авторів. — К.: Диасофт, 2019.
16. ISO/IEC 14882:2020. Programming languages — C++.
17. Stroustrup B., 2013. *The C++ Programming Language* (4th ed.). Addison-Wesley.
18. Meyers S., 2014. *Effective Modern C++*. O’Reilly Media.
19. Gamma E., Helm R., Johnson R., Vlissides J., 1994. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.

20. Pressman R.S., Maxim B.R., 2020. *Software Engineering: A Practitioner's Approach*. McGraw-Hill.

ДОДАТОК А
КОД ПРОГРАМИ

```
#include <tgbot/tgbot.h>

#include <iostream>

#include <string>

#include <map>

#include <sstream>

using namespace std;

using namespace TgBot;


string toLowerCase(const string& s) {

    string result = s;

    for (auto& c : result) c = tolower(c);

    return result;

}

string generateSchedule(const string& goal, const string& level, const string& muscleGroup = "") {

    if (goal == "Схуднення") {

        if (level == "Початківець") return

                "□ Схуднення - початковий рівень:\n\n"
            "□ Понеділок:\n"
            "□ Кардіо: 20 хв\n"
            "□□ Присідання: 3x15\n"
            "□ Віджимання: 3x10\n"
            "□ Скручування: 3x15\n"
```

```
        "□ Планка: 30 сек\n"
        "□ Берпі: 2х10\n\n"
        "□ Вівторок:\n"
        "□ Прогулянка: 30 хв\n"
        "□ Скручування: 3х20\n"
        "□ Махи руками: 3х15\n"
        "□□ Випади: 3х12\n"
        "□ Місток: 3х12\n\n"
        "□ Середа:\n"
        "□ Біг: 15 хв\n"
        "□□ Випади: 3х12\n"
        "□ Берпі: 3х10\n"
        "□ Планка: 30 сек\n"
        "□ Прес: 3х20\n\n"
        "□ Четвер:\n"
        "□ Кардіо: 25 хв\n"
        "□ Планка: 40 сек\n"
        "□□ Присідання: 3х20\n"
        "□ Махи руками: 3х15\n"
        "□ Скручування: 3х15\n\n"
        "□ П'ятниця:\n"
        "□ Ходьба: 40 хв\n"
        "□ Прес: 3х25\n"
        "□ Віджимання: 3х12\n"
        "□□ Присідання: 3х15\n"
        "□ Берпі: 2х12";

    if (level == "Просунутий") return
     "□ Схуднення - просунутий рівень:\n\n"
     "□ Понеділок:\n"
     "□ Біг: 5 км\n"
     "□□ Присідання зі штангою: 4х20\n"
     "□ Планка: 4х40 сек\n"
     "□ Берпі: 4х12\n"
     "□ Скручування: 4х20\n\n"
     "□ Вівторок:\n"
     "□ Кругове тренування: 45 хв\n"
     "□□ Випади з гантелями: 4х12\n"
```

```
        "□ Віджимання з вагою: 4x12\n"
        "□ Планка: 1 хв\n"
        "□ Прес: 4x25\n\n"
        "□ Середа:\n"
        "□ Інтервальний біг (HIIT): 25 хв\n"
        "□ Берпі: 4x15\n"
        "□□ Місток з вагою: 4x12\n"
        "□ Скручування: 4x20\n"
        "□ Планка: 1 хв\n\n"
        "□ Четвер:\n"
        "□ HIIT: 30 хв\n"
        "□□ Присідання зі штангою: 4x15\n"
        "□ Віджимання з вагою: 4x12\n"
        "□ Махи руками: 4x15\n"
        "□ Планка: 1 хв\n\n"
        "□ П'ятниця:\n"
        "□ Біг 5 км або еліпс: 25 хв\n"
        "□ Кругове тренування: 40 хв\n"
        "□ Прес: 4x30\n"
        "□□ Випади з гантелями: 4x12\n"
        "□ Берпі: 3x15";
}
if (goal == "Набір маси") {
    // якщо користувач вибрав конкретну групу м'язів
    if (!muscleGroup.empty() && muscleGroup != "Розклад тренувань") {
        if (muscleGroup == "Груди") {
            if (level == "Початківець") return
                "□ Груди (початківець):\n"
                "□□ Жим гантелей на горизонтальній лаві: 3x12\n"
                "□□ Жим гантелей на похилій лаві: 3x12\n"
                "□ Віджимання: 3x15\n"
                "□ Розведення гантелей: 3x12\n"
                "□ Пуловер з гантелями: 3x10\n"
                "□ Планка: 30 сек\n"
                "□ Берпі: 2x10";
            if (level == "Просунутий") return
                "□ Груди (просунутий):\n"
                "□□ Жим штанги на горизонтальній лаві: 4x10\n"
```

```
            "□□ Жим штанги на похилій лаві: 4x10\n"
            "□ Віджимання з вагою: 4x12\n"
            "□ Розведення гантелей: 4x12\n"
            "□ Пуловер з гантелями: 4x12\n"
            "□ Планка: 1 хв\n"
            "□ Берпі: 3x12\n"
            "□ Кардіо: 10 хв";
        }
// користувач вибрав "розклад на тиждень" або нічого не вибрав
if (muscleGroup.empty() || muscleGroup == "Розклад тренувань") {
    if (level == "Початківець") return
        "□ Набір маси - Початковий рівень (Повний спліт на тиждень):\n\n"
        "□ Понеділок (Груди + Трицепс):\n"
        "□□ Жим лежачи 3x10\n"
        "□ Віджимання 3x15\n"
        "□□ Розведення гантелей 3x12\n"
        "□ Французький жим 3x10\n"
        "□ Віджимання вузьким хватом 3x12\n"
        "□□ Планка 30 сек\n"
        "□ Кардіо 15 хв\n\n"
        "□ Вівторок (Спина + Біцепс):\n"
        "□□ Тяга в нахилі 3x10\n"
        "□□ Гіперекстензія 3x12\n"
        "□ Підйом штанги на біцепс 3x10\n"
        "□ Молотки з гантелями 3x12\n"
        "□□ Тяга верхнього блоку 3x10\n"
        "□ Планка 30 сек\n"
        "□ Ходьба 20 хв\n\n"
        "□ Середина (Ноги):\n"
        "□□ Присідання 3x15\n"
        "□□ Випади 3x12\n"
        "□□ Жим ногами 3x10\n"
        "□□ Згинання ніг 3x12\n"
        "□□ Підйом на носки 3x20\n"
        "□ Планка 40 сек\n"
        "□ Кардіо 15 хв\n\n"
        "□ Четвер (Плечі):\n"
        "□□ Жим гантелей сидячи 3x12\n"
```

```
    "□□ Розведення в сторони 3x12\n"
    "□□ Тяга до підборіддя 3x10\n"
    "□□ Підйом гантелей вперед 3x12\n"
    "□ Планка 40 сек\n"
    "□ Ходьба 20 хв\n"
    "□ Розтяжка плечей 5 хв\n\n"
    "□ П'ятниця (Преса + Кардіо):\n"
    "□ Скручування 3x25\n"
    "□ Велосипед 3x30 сек\n"
    "□ Підйом ніг 3x15\n"
    "□ Берпі 3x10\n"
    "□ Кардіо 25 хв\n"
    "□□ Планка 1 хв\n"
    "□ Розтяжка 10 хв";

if (level == "Просунутий") return
    "□ Набір маси - Просунутий рівень (Повний спліт на тиждень):\n\n"
    "□ Понеділок (Груди + Трицепс):\n"
    "□□ Жим лежачи 4x10\n"
    "□□ Розводка на лаві 4x12\n"
    "□□ Віджимання на брусах 4x12\n"
    "□ Французький жим 4x10\n"
    "□ Жим вузьким хватом 4x8\n"
    "□ Планка 1 хв\n"
    "□ Кардіо 20 хв\n\n"
    "□ Вівторок (Спина + Біцепс):\n"
    "□□ Підтягування 4x10\n"
    "□□ Тяга штанги в нахилі 4x10\n"
    "□□ Тяга горизонтального блоку 4x12\n"
    "□ Підйом штанги на біцепс 4x10\n"
    "□ Молотки 4x12\n"
    "□□ Тяга верхнього блоку за голову 4x10\n"
    "□ Планка 1 хв\n\n"
    "□ Середина (Ноги):\n"
    "□□ Присідання зі штангою 4x12\n"
    "□□ Випади 4x10\n"
    "□□ Жим ногами 4x10\n"
    "□□ Станова тяга 4x8\n"
```

```
            "□□ Підйом на носки 4x20\n"
            "□ Берпі 3x15\n"
            "□ Кардіо 25 хв\n\n"
            "□ Четвер (Плечі):\n"
            "□□ Армійський жим 4x10\n"
            "□□ Розведення гантелей 4x12\n"
            "□□ Тяга штанги до підборіддя 4x10\n"
            "□□ Підйом гантелей вперед 4x10\n"
            "□ Віджимання від підлоги вузьким хватом 3x15\n"
            "□ Планка 1 хв\n"
            "□ Розтяжка 10 хв\n\n"
            "□ П'ятниця (Преса + Кардіо):\n"
            "□ Скручування 4x25\n"
            "□ Підйом ніг 4x20\n"
            "□ Планка з підйомом ніг 4x40 сек\n"
            "□ HIIT 25 хв\n"
            "□ Берпі 4x15\n"
            "□ Інтервальний біг 20 хв\n"
            "□ Розтяжка 10 хв";
    }
}
    return "□ Расписание не найдено!\nПопробуй другую комбинацию:\n"
     "• Цель: похудение или набор массы\n"
     "• Группа мышц (для набора массы): грудь, спина, руки, ноги, плечи\n"
     "• Уровень: начинающий или продвинутый\n\n"
     "Примеры:\n"
     "  похудение начинающий\n"
     "  набор массы грудь продвинутый\n"
     "  набор массы расписание на неделю начинающий";
}

int main() {

setlocale(LC_ALL, "");

string token = "YOUR_BOT_TOKEN";

TgBot::Bot bot(token);
```

```
map<int64_t, string> userGoal;
map<int64_t, string> userMuscle;

bot.getEvents().onCommand("start", [&bot](Message::Ptr message) {
   bot.getApi().sendMessage(message->chat->id,
      "□ Привіт!\n"
      "Ласкаво просимо до бота «СилаБот»!\n\n"
      "Вибери режим:\n\n"
      "□□ Розклад тренувань\n"
      "□ Калькулятор калорій");
   return;
   });

bot.getEvents().onAnyMessage([&bot, &userGoal, &userMuscle](Message::Ptr
message) { int64_t chatId = message->chat->id; string text = message->text;

string textLower = text;
 for (auto& c : textLower) c = tolower(c);

 if (text == "/start") return;

  static std::map<int64_t, std::string> userMode;
 if (userMode.find(chatId) == userMode.end()) {
    if (text == "Розклад тренувань" || text == "Калькулятор калорій") {
       userMode[chatId] = text;
       if (text == "Розклад тренувань") {
          bot.getApi().sendMessage(chatId, "□ Виберіть мету тренування : \n\n"
             "□ Схуднення — зменшення ваги та спалювання жиру\n"
             "□ Набір маси — збільшення м'язової маси та сили\n\n"
          );
       }
       else {
          bot.getApi().sendMessage(chatId, "□ Введіть ваші дані для розрахунку
калорій : \n\n"
             "□□ Вага (кг): \n"
             "□ Зріст (см): \n"
             "□ Вік: \n"
             "□ Стать (ч/ж): \n"
             "□ Рівень активності (1–5):\n"
```

```
                "  1□□ — Мінімальна активність\n"
                "  2□□ — Легкі навантаження (1–3 рази на тиждень)\n"
                "  3□□ — Середня активність (3–5 разів на тиждень)\n"
                "  4□□ — Висока активність (6–7 разів на тиждень)\n"
                "  5□□ — Дуже висока активність (професійні тренування)\n\n"
                "□□ Введіть дані через пробіл:\n"
                "(Приклад: 70 175 25 чоловічий 3)\n"
            );
        }
    }
    else {
        bot.getApi().sendMessage(chatId, "□□ Неправильний вибір! \n\n"
            "Будь ласка, виберіть один з режимів:\n\n"
            "□□ Розклад тренувань — підібрати тренування\n"
            "□ Калькулятор калорій — розрахувати БЖУ і калорії\n\n");
        userGoal.erase(chatId);
        userMuscle.erase(chatId);
        userMode.erase(chatId);
    }
    return;
}
if (userMode[chatId] == "Розклад тренувань") {
    // якщо ще не обрана мета
    if (userGoal.find(chatId) == userGoal.end()) {
        if (text == "Схуднення" || text == "Набір маси") {
            userGoal[chatId] = text;
            if (text == "Набір маси") {
                bot.getApi().sendMessage(chatId,
                    "□ Відмінно! Тепер вибери групу м'язів, яку хочеш тренувати, або
напиши «розклад тренувань» для повного спліту:\n"
                    "□□♂□ Груди\n"
                    "□♂□ Спина\n"
                    "□ Руки\n"
                    "□ Ноги\n"
                    "□♂□ Плечі\n"
                    "□ Розклад тренувань");
            }
            else {
```

```
            bot.getApi().sendMessage(chatId, "□ Виберіть рівень підготовки :\n\n"
                "□□ Початківець — для новачків, які тільки починають
тренуватися\n"
                "□ Просунутий — для тих, хто вже регулярно тренується\n\n");
            }
        }
        else {
            bot.getApi().sendMessage(chatId, "□ Невірна мета!\n\n"
                "Спробуйте вибрати одну з доступних цілей:\n"
                "□ Схуднення — для зниження ваги\n"
                "□ Набір маси — для збільшення м'язової маси\n\n");
            userGoal.erase(chatId);
            userMuscle.erase(chatId);
            userMode.erase(chatId);
        }
        return;
    }
}
// якщо обрана мета "набір маси" і ще не вибрана група м'язів
if (userGoal[chatId] == "Набір маси" && userMuscle.find(chatId) ==
userMuscle.end()) {
    if (text == "Груди" || text == "Спина" || text == "Руки" || text == "Ноги" || text ==
"Плечі" || text == "Розклад тренувань") {
        userMuscle[chatId] = text;
        bot.getApi().sendMessage(chatId, "□ Виберіть рівень підготовки : \n\n"
            "□□ Початківець — для новачків, які тільки починають тренуватися\n"
            "□ Просунутий — для тих, хто вже регулярно тренується\n\n");
    }
    else {
        bot.getApi().sendMessage(chatId, "□ Неправильна група м'язів!\n\n"
            "Спробуйте вибрати одну з доступних груп:\n"
            "□□♂□ Груди\n"
            "□♂□ Спина\n"
            "□ Руки\n"
            "□ Ноги\n"
            "□♂□ Плечі\n"
            "□ Розклад на тиждень");
        userGoal.erase(chatId);
```

```
        userMuscle.erase(chatId);
        userMode.erase(chatId);
    }
    return;
}

// користувач ввів рівень string level = text; string goal = userGoal[chatId]; string
muscle = userMuscle.count(chatId) ? userMuscle[chatId] : "";

// ---- Блок для расчета калорий и БЖУ ---- if (userMode[chatId] == "Калькулятор
калорій") { int weight, height, age, activity; string gender;

// распарсим сообщение пользователя
istringstream iss(text);
iss >> weight >> height >> age >> gender >> activity;

// базовый метаболизм по формуле Миффлина-Сан Жеора
double bmr = 0;
if (gender == "чоловічий") {
    bmr = 10 * weight + 6.25 * height - 5 * age + 5;
}
else if (gender == "жіночий") {
    bmr = 10 * weight + 6.25 * height - 5 * age - 161;
}
else {
    bot.getApi().sendMessage(chatId, "□ Некоректна стать!\n\n"
        "Будь ласка, введіть:\n"
        "□ 'чоловічий'\n"
        "□ 'жіночий'");
    userGoal.erase(chatId);
    userMuscle.erase(chatId);
    userMode.erase(chatId);
    return;
}

// коэффициент активности (1-5)
double activityFactor = 1.2;
if (activity == 2) activityFactor = 1.375;
else if (activity == 3) activityFactor = 1.55;
```

```
else if (activity == 4) activityFactor = 1.725;
else if (activity == 5) activityFactor = 1.9;

double calories = bmr * activityFactor;

// расчёт БЖУ
double protein = weight * 2;          // белки г
double fat = weight * 1;              // жиры г
double carbs = (calories - (protein * 4 + fat * 9)) / 4; // углеводы г

string result = "□ Ваші показники:\n";
result += "Калорії: " + to_string((int)calories) + " ккал\n";
result += "Білки: " + to_string((int)protein) + " г\n";
result += "Жири: " + to_string((int)fat) + " г\n";
result += "Вуглеводи: " + to_string((int)carbs) + " г\n";

bot.getApi().sendMessage(chatId, result);

// сброс состояния
userMode.erase(chatId);
return;


}

  // если пользователь ввел данные для расчета БЖУ
   if (text.find(' ') != string::npos && text.find_first_not_of("0123456789 ") ==
string::npos) {
       int b, f, c, w;
       std::istringstream iss(text);
       iss >> b >> f >> c >> w;
       double kcal = b * 4 + f * 9 + c * 4;
       bot.getApi().sendMessage(chatId,
          "□ Розрахунок калорій:\n"
          "Калорії: " + to_string(kcal) + " ккал\n"
          "Білки: " + to_string(b) + " г\n"
          "Жири: " + to_string(f) + " г\n"
          "Вуглеводи: " + to_string(c) + " г");
       return;
```

```
    }
    //Конец блока расчета БЖУ

    string schedule = generateSchedule(goal, level, muscle);
    bot.getApi().sendMessage(chatId, schedule);

    //збросити стан користувача
    userGoal.erase(chatId);
    userMuscle.erase(chatId);
    });

try {
    cout << "Бот запущен..." << endl;
    TgLongPoll longPoll(bot);
    while (true) longPoll.start();
}
catch (TgBot::TgException& e) {
    cerr << "Ошибка: " << e.what() << endl;
}

return 0;


}
```

ДОДАТОК Б. Перелік тестових питань

- Що таке Telegram-бот і для чого він використовується?
- Які основні функції реалізовані у розробленій програмі?
- Яку мову програмування використано для створення застосунку?
- Яку бібліотеку застосовано для роботи з Telegram Bot API?
- Яким чином бот обробляє повідомлення користувача?
- Що таке токен доступу та для чого він потрібний?
- Як реалізовано обчислення індексу маси тіла (ІМТ) у програмі?
- За якою формулою розраховується добова норма калорій?
- Які перевірки введених даних передбачені у програмі?
- Як можна розширити функціонал розробленого бота?
- Які переваги використання C++ для створення такого застосунку?
- Які можливі напрями подальшого вдосконалення програми?