Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

магістра
(назва освітньо-кваліфікаційного рівня)

студента	*Лисицького Олега Костянтиновича*
	(ПІБ)
академічної групи	*121М-22-1*
	(шифр)
спеціальності	*121 Інженерія програмного забезпечення*
	(код і назва спеціальності)
освітньої програми	*«121 Інженерія програмного забезпечення»*
	(назва освітньої програми)
на тему:	*Розробка інноваційних гейміфікаційних підходів та дослідження ефективності їх впровадження*

_____*О.К. Лисицький*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
розділів кваліфікаційної роботи				
спеціальний	*доц. Приходченко С.Д.*			
Рецензент				
Нормоконтролер	*проф. Лактіонов І.С.*			

Дніпро
2023

Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
Завідувач кафедри
Програмного забезпечення комп'ютерних систем
(повна назва)
_____ М.О. Алексєєв
(підпис) (прізвище, ініціали)
« » _____ 20 23 Року

ЗАВДАННЯ
на виконання кваліфікаційної роботи магістра

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

студенту *121м-22-1* *Лисицькому Олегу Костянтиновичу*
(група) (прізвище та ініціали)

Тема кваліфікаційної роботи *Розробка інноваційних гейміфікаційних підходів та дослідження ефективності їх впровадження.*

1 ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Наказ ректора НТУ «Дніпровська політехніка» від 09.10.2023 р. № 1227-с

2 МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об'єкт досліджень – гейміфікація як новий підхід до навчання та мотивації студентів.

Предмет досліджень – додаток у вигляді телеграм-боту, який впроваджується у процес навчання студентів.

Мета роботи – підвищення рівня мотивації, поліпшення процесу навчання та вплив на аудиторію за допомогою гейміфікації. Необхідно провести дослідження наявних методів гейміфікації та розробити інноваційні гейміфікаційні стратегії.

Вихідні дані для проведення роботи – теоретичні та експериментальні дослідження, статистика з існуючих гейміфікаційних застосунків, дані щодо популярності використання месенджерів підліткам.

3 ОЧІКУВАНІ НАУКОВІ РЕЗУЛЬТАТИ

Наукова новизна полягає в розробленні гейміфікованих підходів, які можуть використовуватися на різних платформах (десктопи, мобільні пристрої, веб-додатки)

для максимальної доступності, а також у інтеграції повсякденних застосунків у процес навчання.

Практична цінність полягає в інтеграції елементів співпраці та конкуренції між учасниками, що може сприяти активнішій взаємодії та обміну досвідом.

4 ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Результати досліджень мають робити внесок у наукове розуміння галузі, в якій проводилося дослідження. Вони мають становити інтерес для наукового співтовариства і розширювати наявні знання щодо існуючих методів викладання та навчання. Отриманий застосунок має продемонструвати позитивні результати, щоб стати унікальною та невід'ємною частиною освітнього процесу для студентів сьогодення.

5 ЕТАПИ ВИКОНАННЯ РОБІТ

Найменування етапів робіт	**Строки виконання робіт (початок – кінець**
Аналіз теми та постановка задачі.	12.09.2023-30.09.2023
Проведення збору та аналізу інформації щодо існуючих методів навчання, розробка застосунку для впровадження гейміфікаційних технологій у освітній процес.	01.10.2023-10.12.2023
Використання програми та аналіз отриманих результатів.	10.12.2023-17.12.2023

6 РЕАЛІЗАЦІЯ РЕЗУЛЬТАТІВ ТА ЕФЕКТИВНІСТЬ

Підвищення мотивації та учбової активності: в освітніх сферах гейміфікація може сприяти активнішій участі студентів, що може поліпшити їхні навчальні результати та підготовку до майбутньої кар'єри.

Рівень залучення: у контексті месенджерів і онлайн-платформ, де гейміфікацію можна застосовувати для стимулювання активності користувачів, результати роботи можуть призвести до збільшення кількості користувачів і залученості до нового підходу.

Завдання видав _____ *Приходченко С. Д.*
(підпис) (прізвище, ініціали)

Завдання прийняв до виконання _____ *Лисицький О.К.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *12.09.2023 р.*

Термін подання дипломного проекту до ЕК *19.12.2023*

РЕФЕРАТ

Пояснювальна записка: 88 сторінок, 64 рисунки, 2 додатки, 40 джерел.

Об’єкт дослідження: гейміфікація як новий підхід до навчання та мотивації студентів.

Предмет дослідження: додаток у вигляді телеграм-боту, який впроваджується у процес навчання студентів.

Мета магістерської роботи: дослідити наявні методи гейміфікації, розробити й випробувати інноваційні гейміфікаційні підходи в конкретній сфері з метою зрозуміти й оцінити їхню ефективність у підвищенні мотивації, навчання або впливі на цільову аудиторію.

Методи дослідження: поставлені задачі розв’язувалися за допомогою пошуку інформації з міжнародних конференцій та наявного викладацького досвіду.

Наукова новизна даної роботи полягає в розробленні гейміфікованих підходів, які можуть використовуватися на різних платформах (десктопи, мобільні пристрої, веб-додатки) для максимальної доступності, а також у інтеграції повсякденних застосунків у процес навчання.

Практична цінність полягає в інтеграції елементів співпраці та конкуренції між учасниками, що може сприяти активнішій взаємодії та обміну досвідом.

Область застосування: результат кваліфікаційної роботи зосереджено на вдосконаленні навчального процесу серед закладів будь-якого типу та підвищенні мотивації до навчання для молодих поколінь.

Значення роботи та висновки: дослідження виявило, що гейміфікація має значущий потенціал у покращенні навчання. Впровадження гейміфікованих елементів стимулює активну участь студентів, сприяє глибшому розумінню та ефективному засвоєнню матеріалу. Застосування телеграм-боту як зручного інструменту для проведення гейміфікованих вікторин підвищує доступність навчання та забезпечує зручний зв'язок між учасниками освітнього процесу.

Прогнози щодо розвитку досліджень: прогнозується подальше вдосконалення гейміфікованих підходів для забезпечення максимальної ефективності та взаємодії з різними типами навчальних закладів. Очікується, що технологічний прогрес дозволить ще більше розширити функціонал телеграм-боту, забезпечивши користувачам нові можливості для зручного та ефективного навчання. Перехід до концепції "школи в смартфоні" може стати реальністю, де навчальний процес стане максимально доступним та комфортним для всіх учасників. Такий розвиток досліджень сприятиме активнішій інтеграції гейміфікації в освітній простір та підвищить якість навчання молодих поколінь.

Ключові слова: ОСВІТНІЙ ПРОЦЕС, ГЕЙМІФІКАЦІЯ, ТЕЛЕГРАМ-БОТ, ІННОВАЦІЇ В ОСВІТІ, СТУДЕНТСЬКА МОТИВАЦІЯ, ПЕДАГОГІЧНІ ТЕХНОЛОГІЇ, ІНТЕРАКТИВНИЙ ПІДХІД.

ABSTRACT

Explanatory note: 88 pages, 64 figures, 2 appendices, 40 sources.

Object of research: gamification as a new approach to teaching and motivating students.

Subject of research: an application in the form of a telegram bot that is implemented in the process of teaching students.

The purpose of the master's thesis: to study existing gamification methods, develop and test innovative gamification approaches in a specific area in order to understand and evaluate their effectiveness in increasing motivation, learning, or influencing the target audience.

Research methods: the tasks were solved by searching for information from international conferences and existing teaching experience.

The scientific novelty of this work lies in the development of gamified approaches that can be used on different platforms (desktops, mobile devices, web applications) to maximize accessibility, as well as in the integration of everyday applications into the learning process.

The practical value lies in the integration of elements of cooperation and competition between participants, which can promote more active interaction and exchange of experience.

Scope: the result of the qualification work is focused on improving the educational process among institutions of any type and increasing the motivation to learn for younger generations.

Significance of the work and conclusions: the study found that gamification has significant potential to improve learning. The introduction of gamified elements stimulates active student participation, promotes deeper understanding and effective learning. The use of a telegram bot as a convenient tool for conducting gamified quizzes increases the accessibility of learning and provides convenient communication between participants in the educational process.

Research outlook: further improvement of gamified approaches is expected to ensure maximum efficiency and interaction with different types of educational institutions. Technological progress is expected to further expand the functionality of the Telegram bot, providing users with new opportunities for convenient and effective learning. The transition to the concept of a "school in a smartphone" may become a reality, where the educational process will become as accessible and comfortable as possible for all participants. This development of research will facilitate the active integration of gamification into the educational space and improve the quality of education for younger generations.

Key words: EDUCATIONAL PROCESS, GAMIFICATION, TELEGRAM BOT, INNOVATIONS IN EDUCATION, STUDENT MOTIVATION, PEDAGOGICAL TECHNOLOGIES, INTERACTIVE APPROACH.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

XP – Experience Points;

MVP – Model-View-Presenter;

UI – User Interface;

Bot – Бот;

SS – Soft-skills;

API – Application Programming Interface;

URL – Uniform Resource Locator;

FAQ – Frequently Asked Questions;

JSON – JavaScript Object Notation;

SDK – Software Development Kit;

VSC – Visual Studio Code;

DB – Data Base;

LMS – Learning Management System;

eLearning – Electronic Learning;

MOOC – Massive Open Online Course;

AR – Augmented Reality;

MCQ – Multiple Choice Question;

T/F – True/False.

ЗМІСТ

ВСТУП

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Задача дослідження гейміфікації та її впливу на освітній процес

Наразі всесвітньо відомі вищі навчальні заклади реалізують ініціативи, спрямовані на створення необхідних умов для розвитку, впровадження та використання цифрових технологій в освіті, що має підвищити якість освітніх послуг, а також інтерес здобувачів освіти, що є представниками покоління сучасних технологій.

Щоб виконати низку завдань цих ініціатив, необхідно модернізувати систему освіти, привести освітні програми у відповідність до потреб цифрової економіки, широко впровадити цифрові інструменти в навчальній діяльності та інтегрувати їх в єдине інформаційне середовище [38-40]. Також має бути забезпечена така індивідуальна навчальна траєкторія, за якою студент матиме можливість навчатися протягом усього життя, у будь-який час і в будь-якому місці.

Для досягнення цієї мети обирається шлях широкого запровадження онлайн-навчання, зокрема курсів з елементами гейміфікації. Ігри та ігрові елементи активно включаються в освітній процес, забезпечують інтерес учнів, а ігрова механіка стає його ядром. Ігрові механіки в бізнесі використовуються з початку XX століття, а сам термін "гейміфікація" з'явився в 1980-х рр.

Спочатку ігрові механіки залучення використовували в маркетингу і продажах. Потім технологія поширилася і на інші сфери діяльності. Ігрові механіки проникають в освіту в найрізноманітніших формах. У Lingualeo, наприклад, користувачі своїми успіхами у вивченні англійської мови годують віртуального левеня, і чим більший прогрес гравця - тим щасливіший персонаж.

Як відносний неологізм термін "гейміфікація" використовується в багатьох формах у різних контекстах. Можна надати визначення гейміфікації як

введення або застосування елементів ігрових онлайн-технологій з метою створення такої системи, в якій успішність гри учасника залежить від його навичок і знань, які можна перенести в реальний світ.

Ігри є універсальною частиною людського досвіду та існували в усіх культурах. Типологія ігор різноманітна. Проте, у кожної гри можуть бути визначені такі характеристики:

- правила, тобто дії, які відрізняються від повсякденних, що визначають сферу вибору гравця протягом усієї гри;
- система зворотного зв'язку, адже більша частина інтерактивності гри спирається на свою систему зворотного зв'язку, яка часто є миттєвою;
- цілі, або умови перемоги, адже часто ігри мають кілька міні-цілей, які приносять бали для досягнення кінцевої мети.

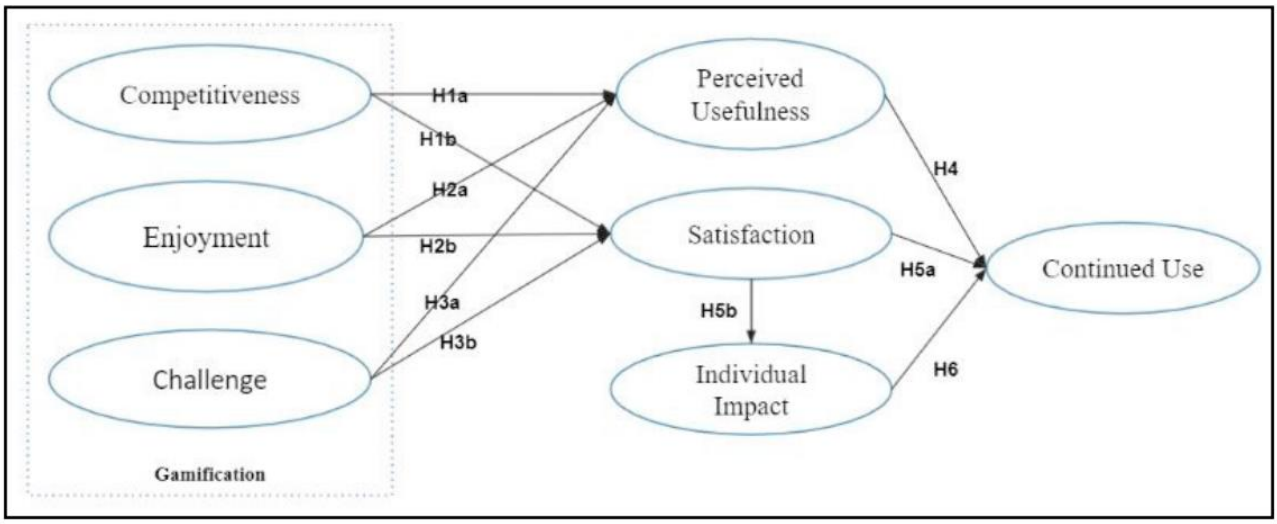

Рис. 1.1. Покроковий вплив гейміфікації на процес продовження навчання

Майже у всіх випадках шлях до перемоги зрозумілий і відомий усім гравцям.

Опис елементів гейміфікації необхідний для цілісного розуміння цієї технології та впровадження її в освітній процес. Завдання це складне насамперед через недостатню вивченість матеріалу. Виявлення таких елементів допоможе викладачеві створити алгоритм реалізації гейміфікованого процесу навчання з

усвідомленням усіх переваг і недоліків. Дослідниками виокремлюються різні елементи гейміфікації та її застосування в освітньому процесі:

- механічні (реєстрація, прогрес і миттєвий зворотний зв'язок);
- особистісні (статус і видимість, колективна відповідальність, списки лідерів або ранжування);
- емоційні (психологічний стан гравців).

Механічний елемент. Реєстрація є першою взаємодією гравця з грою. У більшості ігор, особливо у відеоіграх, є навчальні посібники, мета яких - допомогти гравцям пройти перші кілька хвилин гри. Якщо немає подібних посібників, то перші рівні мають відігравати їхню роль; такі рівні, як правило, легко пройти, і вони слугують для представлення концепцій, що складають гру. В іграх використовується знайомство з механізмами гри і її цілями. В ігровому освітньому середовищі механічний елемент служить двом цілям: по-перше, допомагає збільшити розуміння, як виконати завдання, дозволяючи більш повно брати участь у роботі; по-друге, економить час викладача.

Система прогресії - це особлива послідовність створення гри, за якої спочатку реалізується основний проєкт (базис гри), потім на його основі створюються нові функції, які називаються "інкрементами": цілі, завдання та квести. У більшості ігор, особливо у відеоіграх, міні-нагорода за розв'язання одного з рівнів має бути представлена і бути частиною найскладнішого завдання гри. Під-цілі гри (місії, рівні, квести) нашаровані таким чином, щоб представити додаткові завдання для гравця. Вони чітко визначені та сегментовані, і в більшості за кожну з них пропонуються окремі нагороди [31-33].

Ґрунтуючись на заохоченні поступового просування, ігри містять видимі символи досягнень, часто звані знаками або "level-up". Гравець не тільки зосереджений на позачерговому завданні, яке може призвести до відсутності інтересу після завершення, а й на кінцевій меті курсу, яка може виявитися недосяжною, що позначиться на мотивації. Це забезпечує гарантію виконання проміжних цілей гри.

Зворотний зв'язок - необхідний елемент гри, оскільки результат вибору або дій гравця під час гри мають бути очевидними: коли він приймає рішення, результат видно одразу. Гравці можуть використовувати підказки і паузи для оцінки можливостей і загроз у міру просування до мети. У багатьох іграх використовуються досягнення: одноразові нагороди гравцям за виконання певних завдань або виконання завдань певними способами, які є вторинними щодо основної мети гри. Досягнення публічно відображаються і можуть бути продемонстровані іншим гравцям.

Особистісний елемент. Це саме той елемент, який може допомогти педагогам подальше залучення студентів у навчальний процес. Статус та ідентифікацію за допомогою аватара (представлення себе в грі) видно іншим гравцям, і вони майже завжди розроблені таким чином, щоб дати можливість самовиразитися. Ім'я гравця всередині гри і теги - ще один елемент соціальної адаптації. Вигадуючи собі нові імена й образи, гравці приміряють нові особистості або ролі й ухвалюють значущі рішення в грі з незнайомої точки зору. У контексті освітньої гейміфікації елемент такого типу може дати змогу студентам проєктувати себе в профілі, у комплексі зі своїми науковими досягненнями і представляється для інших студентів не таким, як у зовнішньому світі.

Колективна відповідальність як форма гейміфікації призначена для використання групових занять, щоб стимулювати учнів продовжувати навчання. У багатьох комп'ютерних іграх, як у командних видах спорту, гравці не хочуть підвести своїх товаришів по команді та виконують усі завдання якісно. Це є ключовим мотивувальним фактором [4, 5, 10, 12, 13].

Списки або рейтинги. Усі змагальні ігри ранжують своїх гравців у порядку їхніх досягнень. Таблиця лідерів, де гравці або команди часто відображаються з використанням бальної системи, що демонструє накопичені результати, є широко застосовуваним ранжиром. З етичних причин публічні списки лідерів успішності учнів на базовому освітньому рівні зустрічаються рідко. У тих, хто внизу, може знизитися впевненість, що може призвести до скорочення мотивації

та емоційного спаду. Замість цього списки лідерів, як правило, обмежуються університетськими курсами, орієнтованими на підвищення кваліфікації. Крім того, всі студенти можуть не відображатися в таблиці лідерів.

Емоційний елемент. Один із ключових принципів ігор полягає в тому, що вони приводять гравців у психічний стан, званий "потоком", тобто повної зосередженості на поставленому завданню, ця ідея була вперше запропонована Г. Цихерманом. Чітка мета або набір цілей, чіткий зворотний зв'язок, баланс між викликом і навичками є необхідними умовами для досягнення потоку.

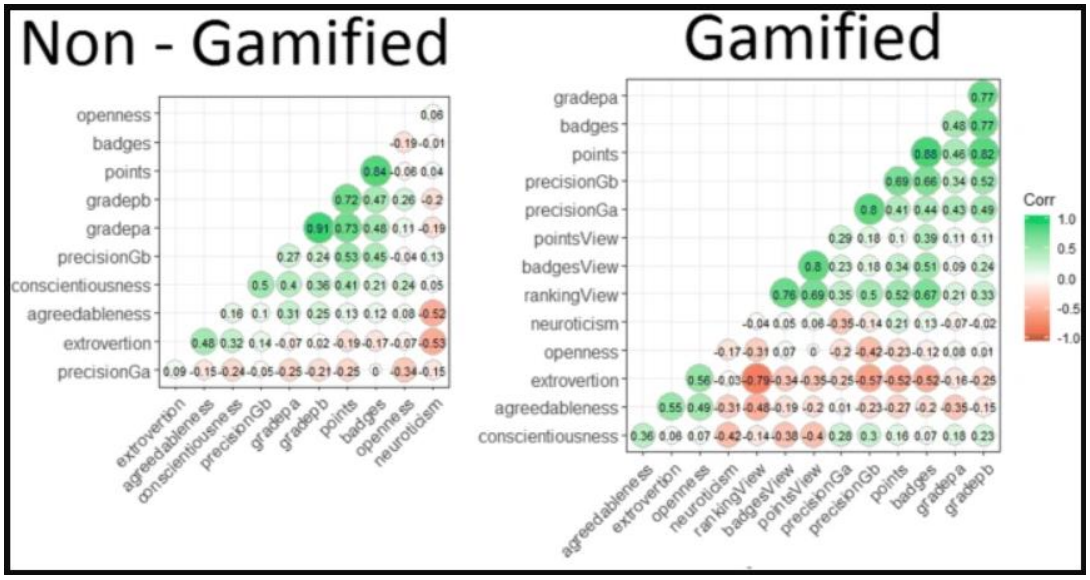

Рис. 1.2. Вплив гейміфікованого підходу на індивідуальні характеристики студентів

Таким чином, гейміфікація - це передова інновація в системі освіти, що постійно змінюється. Освітні ігри мають свої особливості, і під час вибору цього методу навчання необхідно правильно розробити стратегію та методику оцінювання бажаних результатів. Змістовною частиною педагогічної моделі освітнього процесу із застосуванням гейміфікації є запропонований універсальний алгоритм, завданнями якого є: визначити мету, описати гравців, визначити цільову поведінку, позначити шляхи гравців, додати елементи розваги, визначити інструменти, опрацювати зворотний зв'язок і редагувати систему гейміфікації [9, 14, 34-36].

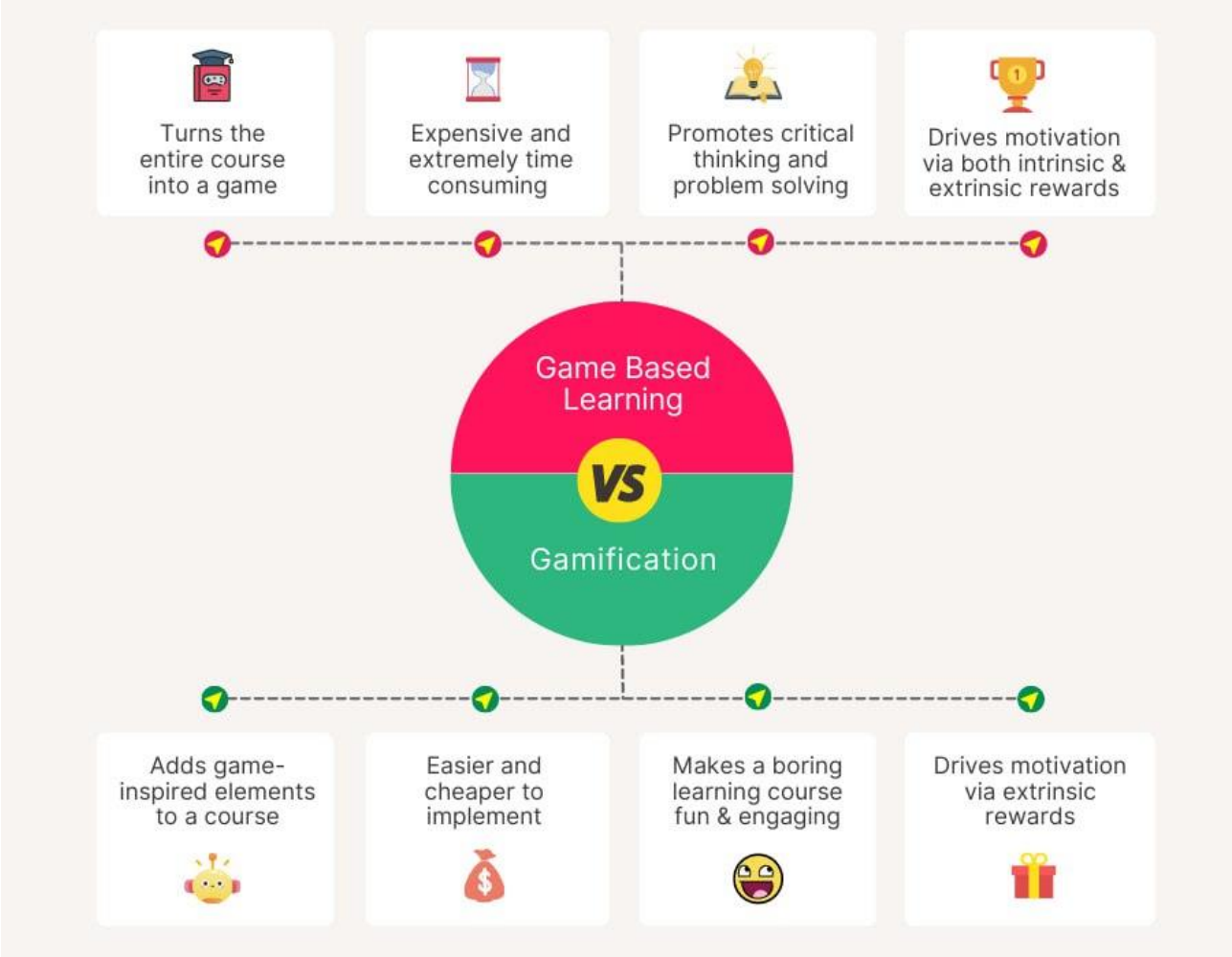

Рис. 1.3. Переваги використання гейміфікаційних методів у існуючому процесі навчання

1.2. Огляд існуючих сервісів, що впроваджують гейміфікацію у освітній процес

Зробити навчання цікавим і захоплюючим завжди було складним завданням для вчителів у всьому світі. Один з найефективніших способів зробити це - створити ігри для учнів. Як би не було важко постійно вигадувати нові ігри для класу, ще більшим викликом є те, щоб учні були повністю залучені в них. І найгірше, що може зробити вчитель, - це змусити дітей брати участь у грі на його умовах: це зводить нанівець усю мету.

Постійний розвиток технологій безпосередньо додає до цієї дилеми, адже учні стають більш технічно підкованими, навіть у ранньому віці, ніж будь-коли.

Як і будь-що інше, сучасні проблеми вимагають сучасних рішень. І одним з найяскравіших прикладів вдалого поєднання гри та освіти є платформа Kahoot.

По суті, Kahoot - це ігрова онлайн-платформа для вчителів та учнів. Вона була створена, щоб зробити навчання легким і цікавим - програма, до якої можна легко отримати доступ через будь-який пристрій, підключений до Інтернету. Майстер гри - вчитель - може зареєструватися і створити гру для гравців, які братимуть у ній участь.

Зазвичай формат гри - це серія запитань, які ставить вчитель, а учні відповідають на них індивідуально або командою, обираючи з запропонованих варіантів відповідей. Особливістю процесу є те, що вищі бали отримують гравці, які найшвидше дають правильні відповіді на кожне запитання. Спосіб, у який розроблено Kahoot, забезпечує повну залученість студентів, адже має під собою змагальну мету.

Вчителі можуть використовувати дані з гри, щоб оцінити успішність своїх учнів. Вона також створена досить простою, щоб зробити навчання легким, і досить складною, щоб зробити навчання цікавим.

Відомо, що такі компанії, як Nike та Starbucks, впроваджують гейміфікацію у своєму бізнесі, і навіть армія США визнала, що використовує її для своїх стратегій рекрутингу [1, 6, 8, 11]. Насправді, ігри, як і Kahoot, є частиною того, що ми називаємо великою EdTech революцією.

Вікторини - це найпопулярніша функція Kahoot. Вона складається з серії запитань, створених вчителем, з варіантами відповідей, що мають певну форму та колір. Маючи понад 70 мільйонів користувачів щомісяця, вчителі також можуть ділитися, використовувати та редагувати вікторини Kahoot, створені іншими користувачами на платформі.

Після створення вікторини Kahoot вчителі можуть розпочати гру, і на екрані з'являється спеціальний ігровий PIN-код. Гравці можуть отримати доступ до цієї гри, ввівши ігровий PIN-код на сайті. Після того, як всі в класі підключені, ведучий може розпочати гру. Процес є швидким і простим. Більше того, результати одразу ж відображаються після завершення гри.

Вчителі можуть використовувати текст або зображення, які відображатимуться на екрані разом із запитанням. Варіанти відповідей - зазвичай від двох до чотирьох - представлені кольорами та формами, що полегшує їх сприйняття студентами. Для того, щоб відповісти, потрібно просто натиснути на потрібний варіант відповіді. Нарешті, майстер гри може також вибрати рандомізацію змісту кожної вікторини, що означає, що якщо ваші учні будуть проходити її повторно, вони не матимуть можливості просто запам'ятовувати порядок відповідей.

Система підрахунку балів у вікторині Kahoot винагороджує гравців, які відповідають на запитання найшвидше (і найправильніше). У таблиці лідерів відображаються 5 найкращих гравців у загальному рейтингу. Всі дані - від рейтингу до швидкості, балів та індивідуальних результатів - доступні для завантаження у вигляді файлу Excel після завершення вікторини.

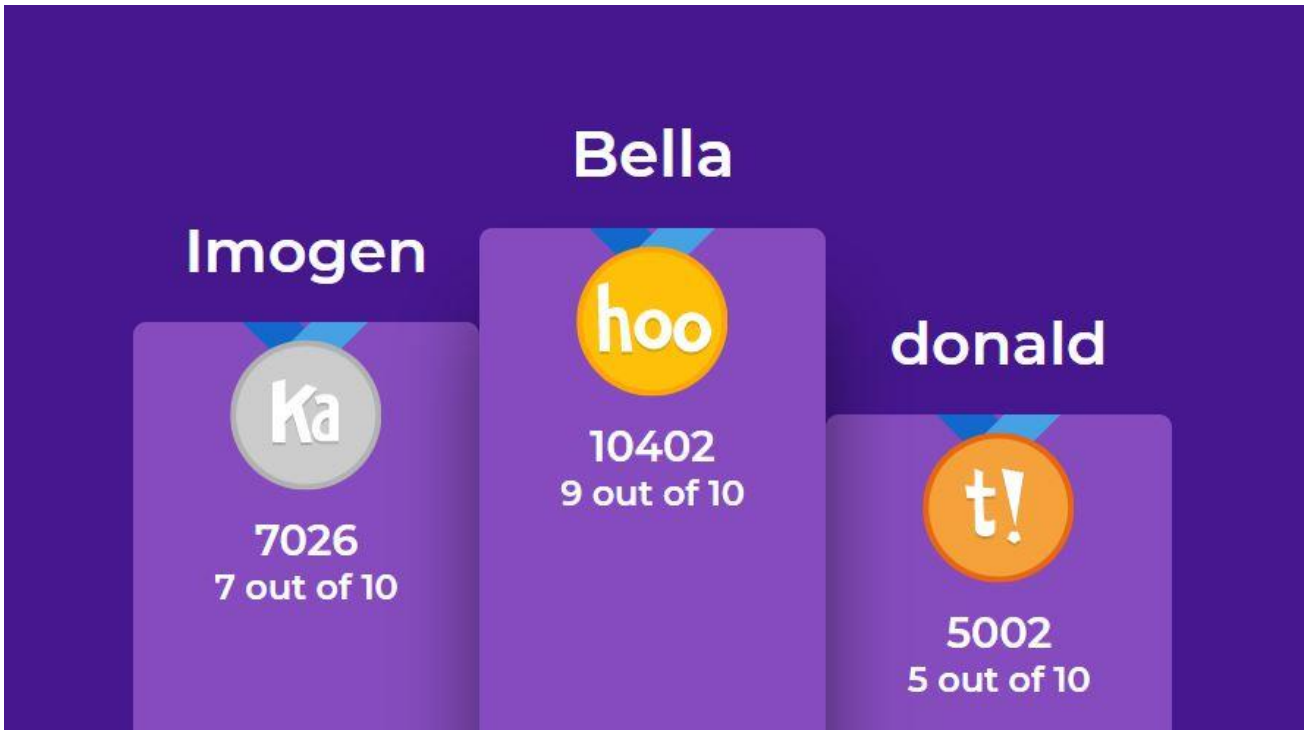

Рис. 1.4. Приклад таблиці лідерів як мотивація до прогресу

1.3. Постановка задачі

Створення застосунку Kahoot всередині телеграм-бота може мати безліч переваг, особливо якщо врахувати популярність і доступність Телеграму, його безкоштовність і зручність використання. Ось кілька ключових переваг:

Універсальність Телеграма. Телеграм є одним із найпопулярніших месенджерів у світі та доступний практично на всіх платформах, включно з мобільними пристроями, планшетами та десктопами. Це означає, що студенти та батьки можуть легко отримувати доступ до освітніх матеріалів у будь-який час і з будь-якого пристрою.

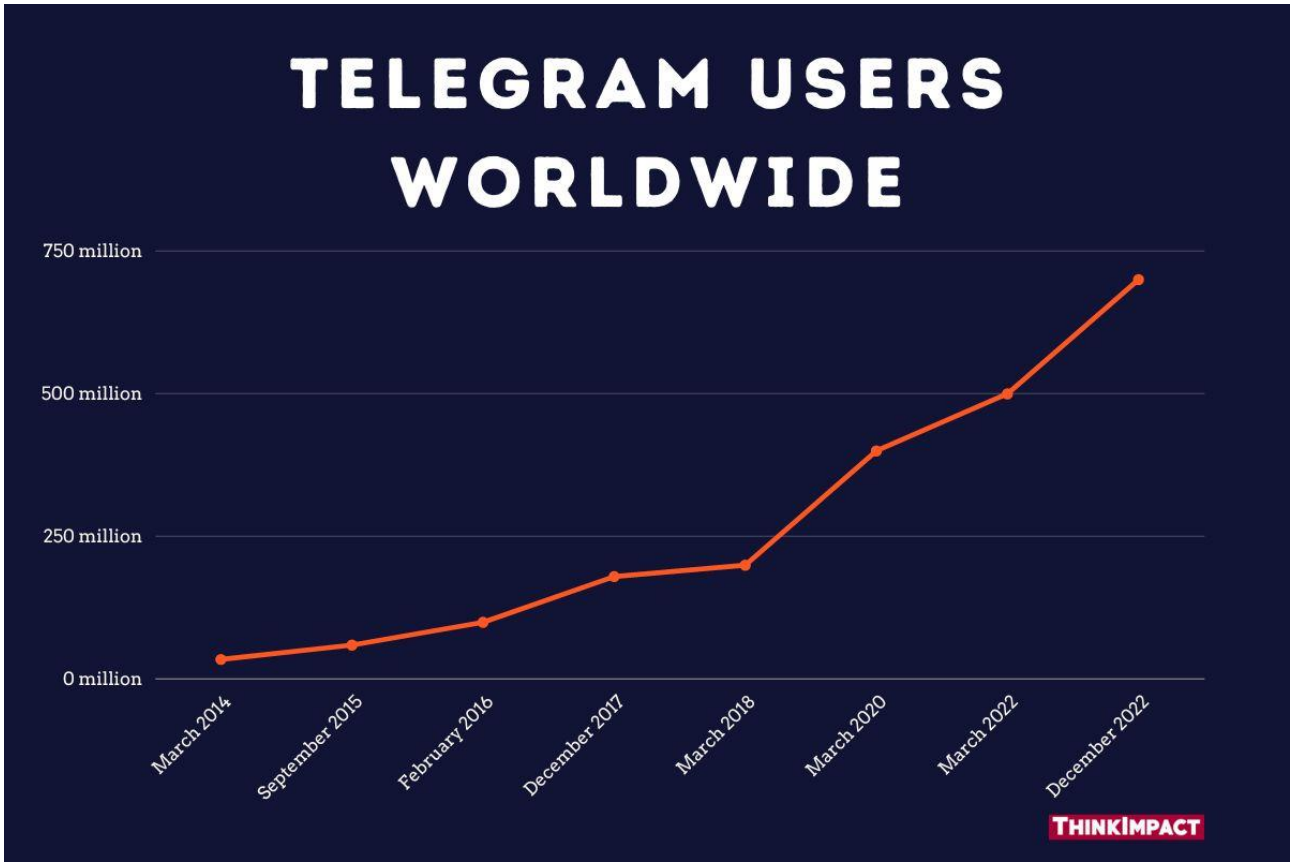

Рис. 1.5. Зростаюча популярність месенджеру Телеграм

Безоплатність. Інтеграція Kahoot у Телеграм-бота не потребує додаткових витрат на створення та підтримку окремого застосунку. Телеграм-боти, як правило, безкоштовні у використанні, і це робить освіту доступнішою для всіх.

Низькі вимоги до обладнання. Телеграм-боти можуть працювати на широкому спектрі пристроїв, навіть на пристроях з обмеженими технічними характеристиками. Це означає, що навіть студенти зі старішими або менш потужними пристроями можуть брати участь в освітньому процесі.

Легкість у використанні. Телеграм-боти зазвичай прості у використанні та не потребують встановлення додаткових додатків. Це знижує бар'єри для вступу в освітній процес.

Миттєве поширення результатів. Одна з головних переваг інтеграції Kahoot у Телеграм-бота полягає в тому, що результати опитувань і вікторин можуть миттєво поширюватися через чати з учнями та батьками. Це сприяє активнішому залученню та дає змогу оперативно оцінювати успіхи.

Зручне управління. Викладачі та адміністратори можуть керувати Quiz-контентом і запитаннями прямо з Телеграм-бота, що робить процес створення та адміністрування вікторин зручнішим.

Безпека та контроль. Телеграм надає засоби для забезпечення безпеки та конфіденційності даних, що важливо в освітньому контексті.

Інтеграція Kahoot у Телеграм-бота надає можливість зробити освіту доступнішою, інтерактивнішою та зручнішою для студентів, батьків і викладачів.

1.4. Висновки

Підводячи підсумки, можна зазначити, що гейміфікація в освіті може підвищити мотивацію та навчальні результати студентів. Це є важливим кроком до покращення освітніх практик.

Огляд популярної платформи Kahoot може допомогти викладачам та закладам освіти краще зрозуміти, які можливості існують при використанні цього інструменту.

Інтеграція Kahoot у Телеграм-бота є інноваційним підходом до вдосконалення освітнього процесу. Це об'єднання популярного месенджера та

інтерактивних вікторин може зробити навчання доступнішим, зручнішим та ефективнішим.

Створення Телеграм-бота з Kahoot дасть змогу учням та їхнім батькам брати участь в освітньому процесі більш активно, миттєво обмінюватися результатами та покращувати взаємодію між вчителями, студентами та батьками.

Ця наукова робота надає основу для подальших досліджень у галузі освіти, гейміфікації та інтеграції технологій. Дослідження можуть бути спрямовані на глибше розуміння впливу інновацій на освіту та навчальні результати. Перспективи використання Kahoot в освіті через Телеграм-бота, надають новий спосіб підвищити якість навчання та залученість студентів. Це може стати важливим кроком у сучасній освіті, враховуючи популярність месенджера та потенціал гейміфікації.

РОЗДІЛ 2. ОПИС МЕТОДУ ВИРІШЕННЯ ЗАДАЧІ

2.1. Дослідження та застосування функціональних можливостей гейміфікації

Спираючись на наявний досвід з проведення занять для дітей та підлітків з використанням гейміфікаційних підходів, було визначено такі функціональні можливості, які необхідно перенести у формат зручного та сучасного бота:

- побудова швидкої та зручної реєстрація;
- дизайн інтуїтивно зрозумілого меню для усіх типів користувачів;
- надання можливості брати участь у вікторині будь-де та будь-коли;
- налаштування доступу до перегляду прогресу;
- збір усього необхідного контенту в єдиному місці;
- синхронізація подій між усіма учасниками процесу;
- розвинена система інформування учасників процесу;
- готовність системи до неочікуваних змін під час освітнього процесу.

Варто відмітити, що Телеграм – це сучасна платформа для взаємодії з користувачем. Вона надає можливості надсилати текстові повідомлення, приймати введення інформації, ділитися зображеннями, створювати інтерактивні клавіатури декількох типів, програвати звукові файли та повідомлення, використовувати вбудовані, або ж додавати власні емодзі. Поєднання цих можливостей надає змогу не лише полегшити досвід використання існуючих гейміфікаційних платформ, а й розширити їхню функціональність шляхом додавання більш сучасних технологій.

Реєстрація – є кроком під номером один, якщо мова заходить про сучасні технології у освітньому процесі. Саме завдяки їй буде отримано можливість групувати та розділяти користувачів за різними критеріями, а також ідентифікувати гравців для використання системи нагород і покарань. Телеграм надає можливість здійснювати реєстрацію лише один раз, а далі користувачу

буде навіть непотрібно входити в обліковий запис, адже бот самостійно буде ідентифікувати користувача будь-якого типу. Окрім входу в акаунт, з'являється зручна можливість запрошень. Студенти вже на першому кроці будуть розподілені між групами, викладачами та батьками, а отже використання паролів чи кодів допуску (як в існуючих гейміфікаційних системах) є зайвим.

Дизайн платформи є також дуже сучасним, мінімалістичним, а отже й інтуїтивно зрозумілим. Розмова з ботом виглядає як звичайний чат з друзями, батьками, де є можливість введення інформації та натискання на кнопки, що прикріплені до усього чату або до окремого повідомлення. В залежності від тієї чи іншої ситуації бот буде пропонувати наступні кроки (наприклад, використання клавіатури). А це означає, що інтерфейс застосунку не буде перевантажений зайвими можливостями та налаштуваннями. Перегляд профілю та власних успіхів завжди під рукою. Більш того, інші типи користувачів, а саме викладачі та батьки матимуть доступ до сторінки студента, що буде додатковою мотивацією для більш кропіткої та уважної роботи під час навчального процесу.

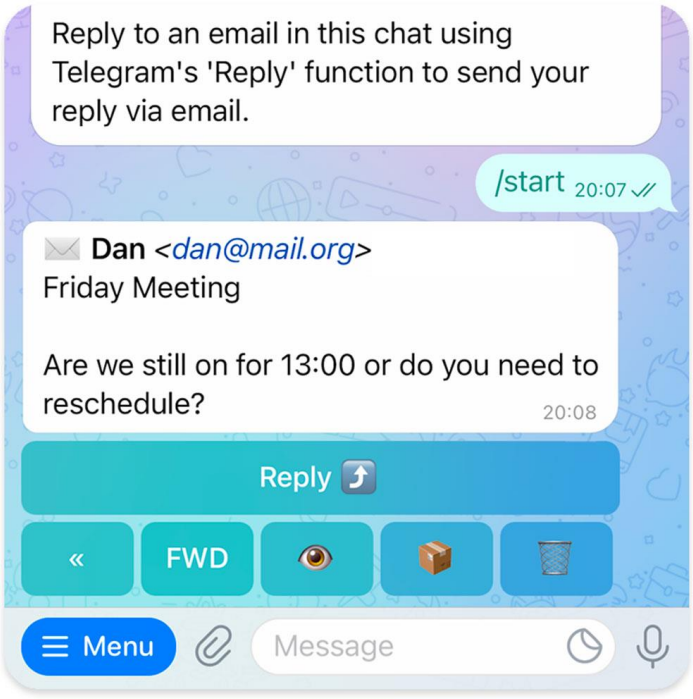

Рис. 2.1. Мінімалістичність та зручність використання Телеграму

Надання можливості брати участь у навчальному процесі будь-де і будь-коли є важливим кроком для сучасного світу. Постійні переміщення сім'ї з тих чи інших причин стали нормою навколишнього середовища, а отже важливою є можливість залучення студента до навчання, навіть якщо він не знаходиться у кабінеті, або ж у онлайн-кімнаті для проведення занять. Повідомлення щодо старту вікторини надійде скрізь, де є інтернет-покриття, навіть на телефон під час знаходження у транспорті. Також відкривається можливість нового формату домашніх завдань, де студент, замість виконання вправ, може пройти вікторину, не домовляючись з іншими учасниками навчального процесу. Як тільки викладач надсилає автоматизоване повідомлення, можна починати тестування.

Перегляд прогресу та статистики є важливим та звичним кроком під час проходження будь-якої сучасної гри. Наприклад, у грі Minecraft надається можливість переглянути кількість пройдених кілометрів, добутих ресурсів, переможених ворогів. Це надає додатковий стимул студенту покращувати власні знання та навички задля швидшого та успішнішого розвитку. Дивлячись на успіхи однодумців, можна з легкістю знаходити власні помилки та переймати досвід для майбутніх випробувань.

Рис. 2.2. Перегляд детальної статистики (досягнутих успіхів) у грі

Викладачу також корисно передивлятися успіхи групи в цілому задля розуміння складності тієї чи іншої теми для їхнього додаткового перегляду разом зі студентами. Звертаючи увагу на подібну статистику, можна зробити висновок, що теми циклу while та ООП стають найважчими для студентів, адже результати усієї групи нижчі ніж зазвичай. Якщо ж лише декілька студентів мають результати гірші за 50% - це може означати необхідність у зміні підходу до викладання теми саме для цієї групи здобувачів освіти або ж у проведенні додаткових занять.

Проведення вікторин у Телеграм боті є зручним, оскільки тут можна одночасно задавати питання, приймати відповіді, а також отримувати корисні матеріали у вигляді тексту, аудіо чи відео. Більше не буде необхідності переключатись між 4-5 платформами під час заняття, все може бути зібране в одному місці з рівними можливостями для кожного користувача (студента групи).

Важливим кроком є впровадження системи повідомлень для батьків та викладачів під час проходження вікторини. Батьки будуть одразу розуміти, що дитина доєдналася до гри, або ж відмовилася від неї. Також вони будуть мати змогу напряму слідкувати за результатами, а не сподіватися на відповідальність дитини чи викладача. Усе, що потрібно, це провести швидку реєстрацію батьків з синхронізацією їхнього профілю з акаунтом дитини. А отже, це гарний інструмент для полегшення роботи викладача та додатковий стимул студентам готуватися активніше.

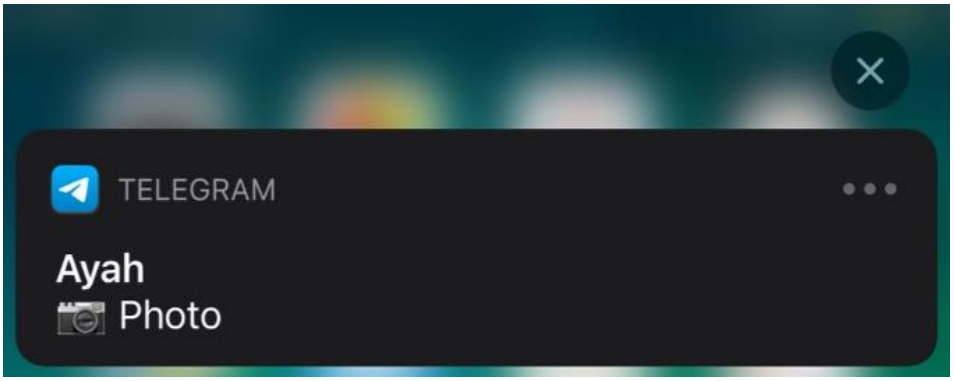

Рис. 2.3. Приклад Телеграм повідомлення для нагадування

Якщо ж дитина не на зайняті с тих чи інших причин, система також буде готова до цього. При відмові від участі буде надіслане відповідне повідомлення викладачу та батькам задля перевірки достовірності відсутності. А під час перегляду статистики можна чітко зрозуміти, якому з студентів ще необхідно пройти гру-вікторину. Тобто, навіть за відсутності студента, група продовжує навчання і питання щодо пропуску буде закрите шляхом того, що усі учасники навчального процесу повідомлені про існуючу вікторину.

2.2. Перегляд системи нагород та покарань

Для побудови цікавого та конкурентного шляху під час проходження вікторини необхідно створити зручну систему підрахунку очок, яка буде мотивувати не лише знаходити правильні відповіді на запитання, а й робити це швидше за своїх колег. Такий підхід надає змогу навчитися знаходити баланс між можливістю засвоювати знання та економити час, адже це найважливіші soft-skills людини сьогодення [7, 37]. Часовий показник також має бути впливати на загальний бал, а отже спонукати студентів до швидшої реакції на події.

Тому для підрахунку набраних очок у кінці вікторини використовується формула, яка надає більшу вагу саме правильним відповідям, однак враховує швидкість проходження усього шляху. Студент, який відповів правильно на всі запитання, може боротися за потрапляння до списку лідерів якщо його витрачений час на проходження вікторини був меншим за час конкурентів. Якщо ж студент допустив помилку під час проходження, він все ще може змагатися за таблицю лідерів, адже інші питання він міг подолати значно швидше. Виходячи з цих тверджень маємо наступну формулу з коефіцієнтами:

$$R = \left(\sum_{i=1}^{n} V_i\right) * k - \frac{\sum_{i=1}^{n} T_i}{k}, \qquad (2.1)$$

де R – це кількість очок, яку буде нараховано гравцеві,
V_i – кількість даних відповідей на поставлені запитання,
T_i – кількість секунд витрачених на вирішення поставлених запитань,
k – коефіцієнт для надання ваги правильним відповідям у порівнянні з витраченим часом.

Ця формула підраховуватиметься одразу після завершення відповідей студента на усі запитання, а її результати будуть порівнюватися один з одним для формування таблиці лідерів. Фінальна таблиця буде демонструвати трьох найуспішніших учасників як винагороду за їхні старання, а ті студенти, які опинилися поза списком не будуть опубліковані на огляд, однак вони матимуть мотивацію рости і розвиватись, щоб наздогнати лідерів групи. Студент, який є переможцем вікторини отримує додатковий бал до власної статистики і у будь-який час зможе перевірити кількість своїх пройдених вікторин, а також кількість власних перемог під час них.

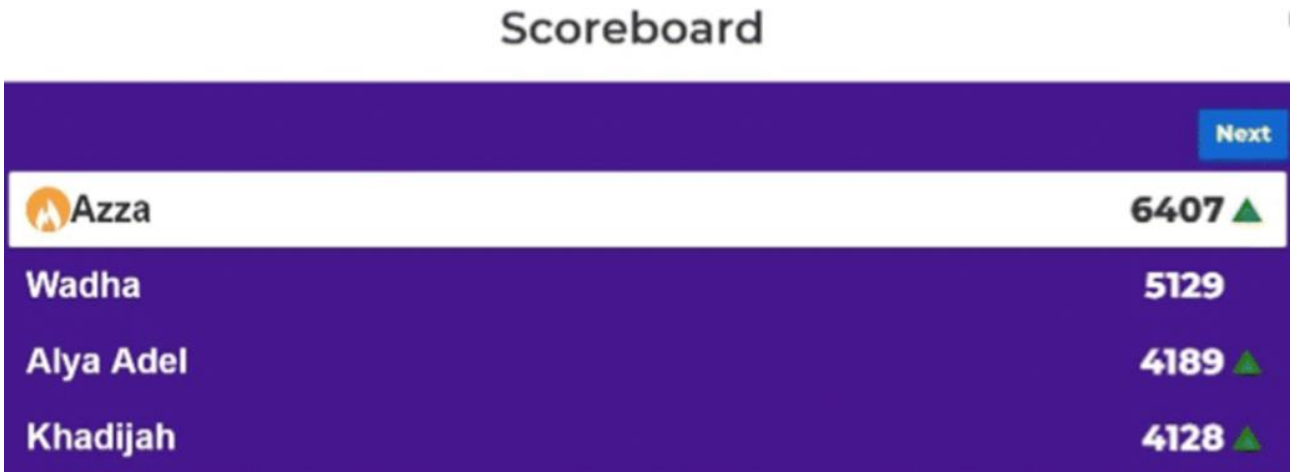

Рис. 2.4. Приклад підрахунку очок після проходження вікторини

Викладач також матиме змогу бачити результати вікторини задля розуміння того, хто повністю зрозумів вивчену тему, а хто має ще попрацювати для закріплення результату.

Окрім системи нагород також присутня система покарань. Вона доступна для викладачів та батьків, однак направлена не на низький бал під час

проходження, а на небажання студента приймати участь у вікторині. Тому при спробі відмовитися від змагань, викладачі та батьки отримають відповідне повідомлення з попередженням щодо цієї ситуації.

2.3. Висновки

Підсумовуючи усе вищесказане, досвід проведення занять для дітей та підлітків привів до визначення ключових функціональних можливостей для перенесення їх у формат бота. Зокрема, наголошено на важливості зручної реєстрації, інтуїтивно зрозумілому дизайні, можливості участі вікторини у будь-якому місці, налаштуванні доступу до прогресу, зборі контенту в одному місці, синхронізації подій між учасниками, інформуванні учасників та готовності системи до неочікуваних змін. Телеграм, як платформа, надає широкі можливості для зручної взаємодії з користувачем. Розглянуті аспекти, наводяться як важливі для покращення досвіду використання гейміфікаційних платформ у сучасному освітньому процесі.

Важливо відзначити, що побудова цікавого та конкурентного шляху викладання з використанням гейміфікації вимагає не лише правильної системи підрахунку очок, а й ретельного врахування факторів швидкості та точності відповідей. Розроблена формула з коефіцієнтами дозволяє створити баланс між здобуттям знань та ефективним використанням часу, що сприяє розвитку важливих soft-skills. Впровадження системи нагород та покарань допомагає стимулювати учасників до активної участі та розвитку, створюючи здоровий конкурентний дух у групі.

Враховуючи високий рівень мотивації та можливість стимулювання розвитку учасників, такий підхід не лише поліпшує якість освіти, а й сприяє формуванню ефективної освітньої спільноти. Така інноваційна система взаємодії може служити прикладом для інших освітніх платформ, що прагнуть поєднати сучасні технології, гейміфікацію та активний навчальний процес.

РОЗДІЛ 3. РОЗРОБКА ЗАСТОСУНКУ ГЕЙМІФІКОВАНОГО ТЕЛЕГРАМ БОТУ

3.1. Опис структури бази даних

Для виконання поставленого завдання було обрано хмарну нереляційну базу даних MongoDB Atlas та побудовано наступну структуру для взаємодії документів формату «ключ-значення» з можливістю синхронізації документів в тому числі між різними колекціями бази даних [3, 21].

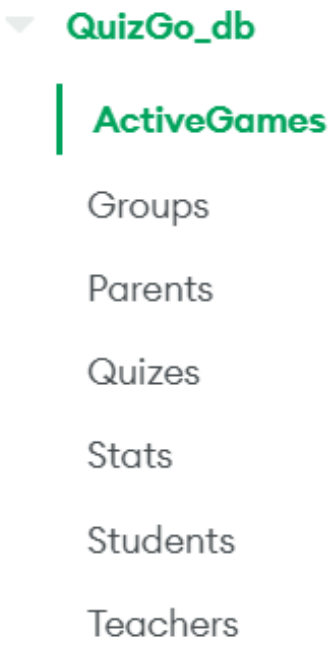

Рис. 3.1. Перелік існуючих колекцій для зберігання документів

Як можна побачити, хмарна база даних складається з семи колекцій, де кожна відповідає за власну унікальну сутність:

- ActiveGames – відповідає за вікторини, які на даний момент активні та проводяться викладачами для студентів;
- Groups – відповідає за групи для студентів, що необхідні для побудови відокремленого навчального процесу школи чи університету;

- Parents – відповідає за профілі батьків, які мають бути пов’язаними зі своїми дітьми;
- Quizes – відповідає за існуючі вікторини, саме тут зберігається бібліотека з питань, які викладачі можуть обирати для проведення у своїй групі;
- Stats – відповідає за збереження ігрової статистики, щоб мати доступ переглядати історію ігор для викладачів з детальною інформацією по кожному студенту та його відповідям;
- Students – відповідає за профілі студентів, які реєструються у системі та синхронізуються з існуючими групами викладача;
- Teachers – відповідає за профілі викладачів, саме вони мають найбільше можливостей для роботи з вікторинами, групами та студентами.

Далі колекції буде розглянуто детальніше задля розуміння того, які ключі формують відповідні документи та які відповідають за синхронізацію даних між різними колекціями бази.

Рис. 3.2. Документ з профілем студента

Профіль студента складається з 10 полів, які необхідно вводити самостійно при реєстрації, обирати, натискаючи кнопки вбудованих клавіатур, або ж отримувати під час участі у вікторинах з використанням власного акаунту.

- _id – є стандартним полем-ключем для роботи з документами в середовищі MongoDB, воно дозволяє ідентифікувати документ незалежно від його місце-знаходження;

– tg_id – є спеціальним ключем Телеграму, який дозволяє ідентифікувати користувача у системі після проходження єдиної одноразової реєстрації;
– firstname – ім’я користувача, заповнюється під час реєстрації;
– lastname – прізвище користувача, заповнюється під час реєстрації;
– email – поштова скринька користувача, заповнюється під час реєстрації;
– phone – телефон користувача, заповнюється під час реєстрації;
– is_control – ключ, який демонструє батьківський контроль над цим профілем, тобто батьки зможуть отримувати повідомлення щодо успіхів дитини;
– group_name – назва групи, яка обирається за допомогою інтерактивної клавіатури та співвідносить студента до тієї чи іншої групи освітнього процесу;
– games_total – ключ, що відповідає за кількість зіграних вікторин, що створюється та інкрементується автоматично;
– wins_total – ключ, що відповідає за кількість переможних вікторин, що створюється та інкрементується автоматично.

Рис. 3.3. Документ з профілем викладача

Профіль викладача складається з 6 полів, які необхідно вводити самостійно при реєстрації.

– _id – є стандартним полем-ключем для роботи з документами в середовищі MongoDB, воно дозволяє ідентифікувати документ незалежно від його місце-знаходження;
– tg_id – є спеціальним ключем Телеграму, який дозволяє ідентифікувати користувача у системі після проходження єдиної одноразової реєстрації;

– firstname – ім’я користувача, заповнюється під час реєстрації;
– lastname – прізвище користувача, заповнюється під час реєстрації;
– email – поштова скринька користувача, заповнюється під час реєстрації;
– phone – телефон користувача, заповнюється під час реєстрації.

Рис. 3.4. Документ з батьківським профілем

Профіль для батьків складається з 7 полів, які необхідно вводити самостійно при реєстрації або обирати, натискаючи кнопки вбудованих клавіатур.

– _id – є стандартним полем-ключем для роботи з документами в середовищі MongoDB, воно дозволяє ідентифікувати документ незалежно від його місце-знаходження;
– tg_id – є спеціальним ключем Телеграму, який дозволяє ідентифікувати користувача у системі після проходження єдиної одноразової реєстрації;
– firstname – ім’я користувача, заповнюється під час реєстрації;
– lastname – прізвище користувача, заповнюється під час реєстрації;
– email – поштова скринька користувача, заповнюється під час реєстрації;
– phone – телефон користувача, заповнюється під час реєстрації;
– student_tg_id – ключ, що відповідає за синхронізацію батьківського акаунта до студентського, обирається шляхом натискання на кнопку інтерактивної клавіатури.

Рис. 3.5. Документ з інформацією про групу

Документ з інформацією про групу складається з 7 полів та створюється адміністратором або методистом навчального закладу.

– _id – є стандартним полем-ключем для роботи з документами в середовищі MongoDB, воно дозволяє ідентифікувати документ незалежно від його місце-знаходження;
– name – назва групи, що необхідна для пошуку студентами та викладачем, є важливою складовою для проведення вікторини та підрахунку результатів для усієї групи;
– language – мова програмування, яку вивчатимуть студенти, що пов’язані з цією групою;
– course – номер курсу, що залежить від обраної мови, тобто маємо можливість переходити на наступний рік навчання мови програмування та її фреймворків;
– day – день тижня, у який проходить заняття з даною групою;
– time – час дня, у який проходить заняття з даною групою;
– teacher_tg_id – ключ, що відповідає за синхронізацію групи з викладачем, який проводить зайняття студентам.

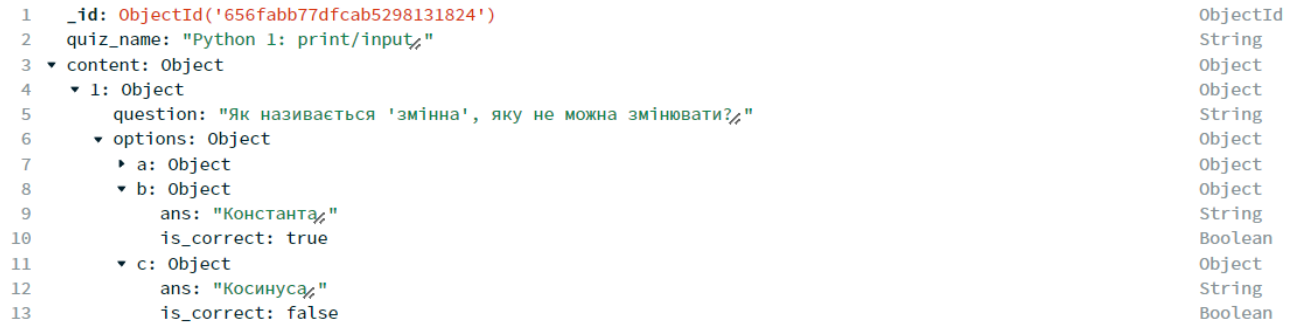

Рис. 3.6. Документ з інформацією про вікторину

Документ з інформацією про вікторину складається з 3 основних полів, які створюються адміністратором або методистом навчального закладу. Ключ content може містити будь-яку кількість запитань для вікторини з будь-якою кількість варіантів для відповіді, де лише одна є правильною.

– _id – є стандартним полем-ключем для роботи з документами в середовищі MongoDB, воно дозволяє ідентифікувати документ незалежно від його місце-знаходження;
– quiz_name – назва вікторини, що використовується при створені гри та запрошенні учасників до проходження її;
– content – об’єкт, що зберігає будь-яку кількість питань та відповідей, промаркованих ключами для ідентифікації правильності чи помилковості.

У свою чергу об’єкт content ділиться на наступні ключі:

– question – одне з питань, на яке необхідно буде відповідати студентам;
– options – можливі варіанти відповіді з маркуванням правильності.

У свою чергу об’єкт options ділиться на варіанти відповіді з наступними ключами:

– ans – одна з можливих відповідей на питання;
– is_correct – ключ-ідентифікатор правильності, є логічним значенням (T/F).

Рис. 3.7. Документ з інформацією про активну гру

Документ з інформацією про активну гру складається з 10 основних полів, які створюються автоматично, якщо викладач розпочинає вікторину для своєї групи.

– _id – є стандартним полем-ключем для роботи з документами в середовищі MongoDB, воно дозволяє ідентифікувати документ незалежно від його місце-знаходження;

– teacher_tg_id – є ключем, що пов’язує активну гру з викладачем, який її створив та проводить для своєї групи;
– teacher_firstname – ім’я викладача, що використовується для майбутнього збереження статистики щодо ігор, які вже відбулися;
– teacher_lastname – прізвище викладача, що використовується для майбутнього збереження статистики щодо ігор, які вже відбулися;
– group_name – назва групи, яка братиме участь у грі та для якої будуть надіслані запрошення;
– group_size – кількість студентів групи, щоб викладач міг розуміти, коли йому розпочинати гру (тобто після того, як усі студенти відреагують на запрошення до гри);
– quiz_name – назва вікторини, що використовується для майбутнього збереження статистики щодо ігор, які вже відбулися;
– quiz_open_time – час у секундах, що відповідає за час початку вікторини та використовується для майбутнього збереження статистики щодо ігор, які вже відбулися;
– quiz_started – ключ, який сигналізує, що гра вже почалася (тобто приєднатися до неї вже не можна);
– quiz_players – масив, який зберігає дані щодо участі студентів у цій грі зі збереженням їхніх відповідей на запрошення та відповідей на запитання.

У свою чергу, масив quiz_players має наступну структуру об’єктів:

Рис. 3.8. Масив quiz_players та структура його об’єктів

Кожен об’єкт є демонстрацією участі студента в обраній вікторині-грі.

– student_tg_id – ключ, який синхронізує профіль студента з профілем учасника гри-вікторини;
– student_firstname – ім’я студента, що бере участь у вікторині;
– student_lastname – прізвище студента, що бере участь у вікторині;
– connection_time – час у секундах, що відповідає за час приєднання студента до гри-вікторини;
– finish_time – час у секундах, що відповідає за час завершення гри-вікторини студентом;
– total_score – загальна кількість правильних відповідей протягом участі у грі-вікторині;
– total_time – загальна кількість часу, витраченого студентом для відповіді на усі запитання гри-вікторини;
– total_points – загальна кількість очок, які рахуються по формулі згідно з кількістю правильних відповідей та витраченого часу на гру-вікторину;
– finish_game – ключ, що демонструє, чи завершив студент проходження гри-вікторини, чи ще у процесі;
– ready_to_play – ключ, що демонструє готовність студента до гри (тобто він прийняв запрошення до конкретної гри-вікторини);
– results – об’єкт з результатами конкретного студента після відповіді на те чи інше запитання.

У свою чергу, об’єкт results має наступну структуру полів:

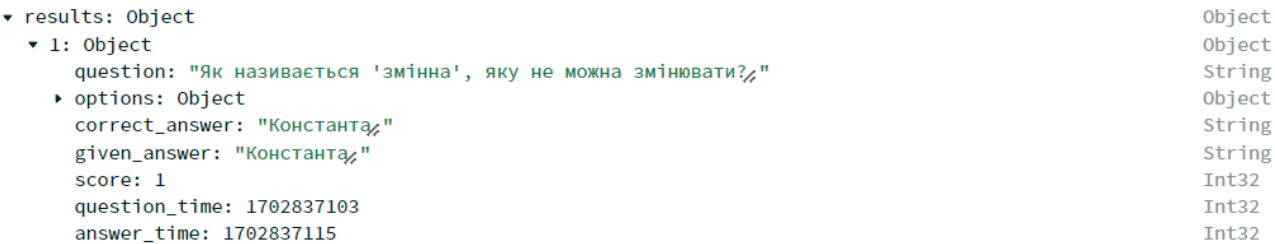

Рис. 3.9. Об’єкт results та структура його полів

Кожен об’єкт є демонстрацією відповіді студента на питання гри-вікторини.

- question – збережене питання, що було поставлене студенту під час проходження гри-вікторини;
- options – збережені варіанти відповіді, що були запропоновані студенту під час проходження гри-вікторини;
- correct_answer – правильна відповідь на питання, яку має надати студент під час проходження гри-вікторини;
- given_answer – надана відповідь на питання під час проходження гри-вікторини;
- score – кількість балів, набраних за це питання під час проходження гри-вікторини;
- question_time – час у секундах, що відповідає за час, коли питання з’явилося у інтерфейсі студента;
- answer_time – час у секундах, що відповідає за час, коли студент відповів на питання у власному інтерфейсі.

Саме з використанням об’єкту results система буде підраховувати результати від час формування таблиці лідерів.

Що стосується колекції Stats, її формат документу є ідентичним до документів колекції ActiveGames, адже після того, як студенти надають відповідь на останнє запитання, документ переміщується з стадії активної гри у «архів» Stats для майбутнього перегляду статистики. Важливо зазначити, що колекція ActiveGames зберігає у собі гру тільки на час проведення, коли ж Stats володіє інформацією протягом усього циклу існування програмного забезпечення. ActiveGames, як і Stats надають можливість маніпулювати з даними гри у будь-який час, отримуючи нові дані та розрахунки з використанням існуючих полів.

3.2. Застосовані програмні засоби та технології для реалізації програмного забезпечення

Для виконання поставленого завдання був обраний набір програмних засобів та технологій, які є доволі сучасними та дозволяють швидко та надійно співпрацювати з Телеграм ботом. Першим кроком обрано базу даних MongoDB та її додатковий сервіс Atlas, що дозволяє безкоштовно зберігати дані у хмарному середовищі.

MongoDB Atlas – це сучасний хмарний сервіс для керування та розгортання баз даних MongoDB, який володіє кількома ключовими перевагами.

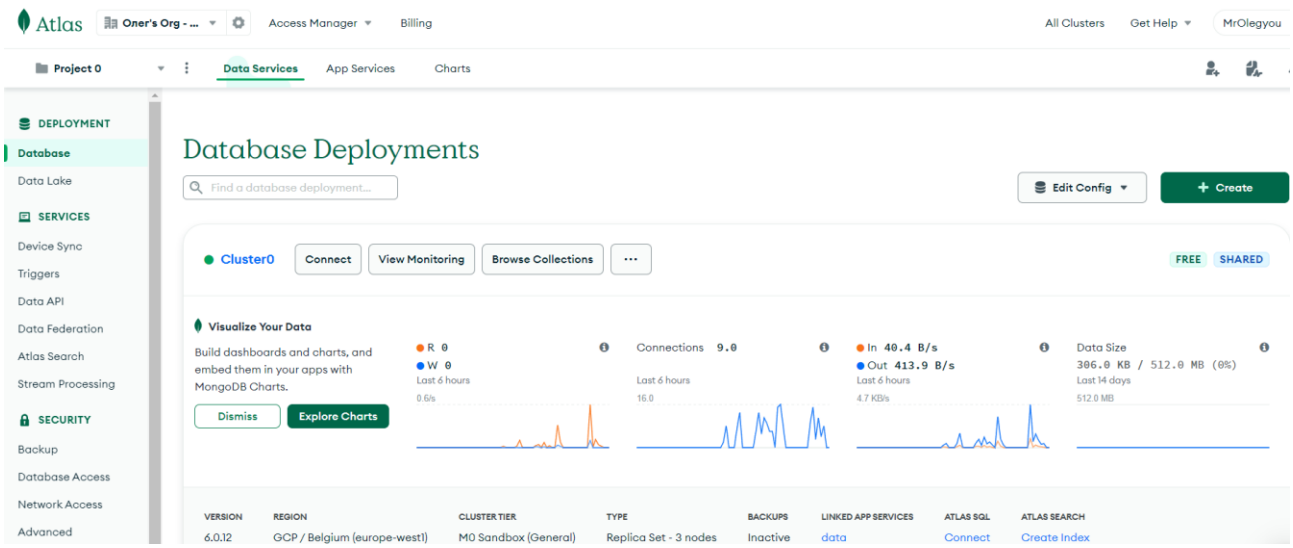

Рис. 3.10. Інтерфейс хмарного середовища MongoDB Atlas

По-перше, він забезпечує простоту та зручність у розгортанні та управлінні базами даних MongoDB в хмарних середовищах, таких як AWS, Azure та Google Cloud Platform (GCP). Не потрібно витрачати час на складні налаштування чи адміністрування інфраструктури.

По-друге, MongoDB Atlas забезпечує високий рівень безпеки для даних. Можна використовувати автентифікацію, авторизацію та шифрування для захисту інформації від несанкціонованого доступу.

По-третє, цей сервіс пропонує гнучкі можливості масштабування, які дозволяють легко збільшувати або зменшувати ресурси в залежності від потреб проєкту. Це забезпечує ефективне використання ресурсів і оптимізує продуктивність.

Завдяки автоматизованим процесам резервного копіювання та відновлення даних, можна бути впевненим в надійності та доступності інформації в разі непередбачуваних ситуацій.

Нарешті, MongoDB Atlas відзначається широкою підтримкою для різних мов програмування та інтеграцією з популярними інструментами розробки, що робить його зручним та універсальним рішенням для розробників та адміністраторів баз даних MongoDB.

У якості мови програмування було обрано сучасний інструмент для будь-якого розробника – Python.

Python – це високорівнева мова програмування, що вирізняється своєю простотою та легкістю вивчення. Її синтаксис є читабельним і лаконічним, нагадуючи англійську мову. Це дозволяє програмістам швидко освоювати мову та писати зрозумілий код [23-30].

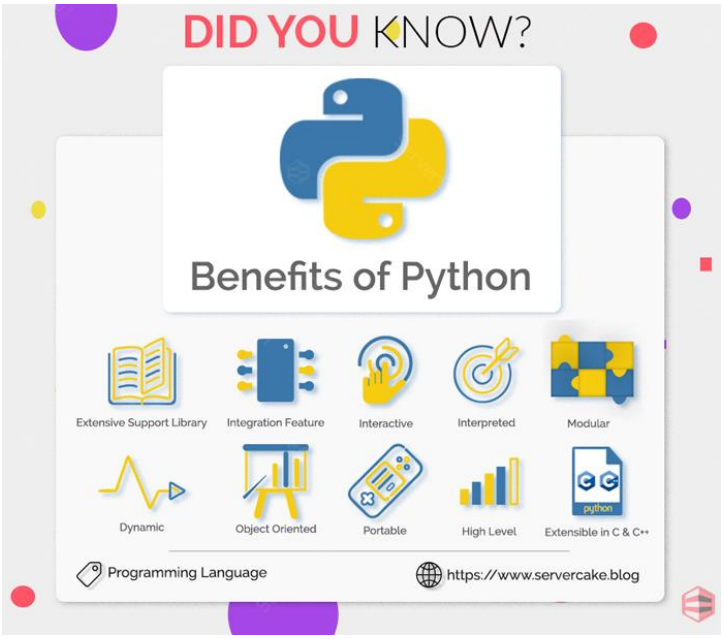

Рис. 3.11. Переваги використання мови програмування Python

Однією з вагомих переваг Python є його універсальність. Він використовується для розробки різноманітних додатків, від веб-сайтів і наукових досліджень до штучного інтелекту та обробки даних. Така гнучкість робить Python популярним в середовищі розробників та дослідників.

Ще однією значущою перевагою є широкий екосистем Python. Багатство бібліотек і фреймворків спрощують розробку та розширення функціональності програм. Наприклад, бібліотека NumPy дозволяє ефективно працювати з масивами даних, а Django полегшує створення веб-додатків.

Python також відомий своєю спільнотою. Велика кількість розробників сприяє активному обміну досвідом і створенню відкритих проєктів. Це робить Python ідеальним вибором для тих, хто цінує підтримку та ресурси спільноти.

Зручний та ефективний відладчик Python робить процес розробки більш простим і продуктивним. Його інтерактивна оболонка, або REPL (Read-Eval-Print Loop), дозволяє виконувати код почергово, спрощуючи вивчення та експериментування з функціями.

Загалом, Python володіє комбінацією простоти, гнучкості та сильної спільноти, що робить його популярним серед програмістів і відмінним вибором для різних завдань у сфері розробки програмного забезпечення.

Рис. 3.12. Демонстрація офіційної взаємодії Python та MongoDB

Для взаємодії з різними типами баз даних, включаючи NoSQL-рішення, існують потужні бібліотеки та фреймворки. Для роботи з MongoDB, наприклад, використовують PyMongo - офіційний драйвер для Python, який забезпечує зручний інтерфейс для взаємодії з цією базою даних. Це дозволяє створювати комплексні та масштабовані рішення, інтегруючи бази даних MongoDB як елемент екосистеми.

Крім того, Python має ефективні засоби для асинхронного програмування, що дозволяє розробникам створювати високопродуктивні та масштабовані додатки, які легко взаємодіють з базами даних у режимі реального часу.

Загалом, використання Python у сфері баз даних, зокрема MongoDB, принесе не лише простоту та зручність, але й потужний інструментарій для розробки високоякісних програмних рішень.

Для написання Python коду та його взаємодії із сервісами MongoDB було обрано програмне забезпечення від компанії Microsoft – Visual Studio Code (VS Code). Це високоефективний текстовий редактор, який набув великої популярності серед розробників завдяки своїм численним перевагам.

Завдяки легкій інсталяції та низьким системним вимогам, VS Code забезпечує швидку і стабільну роботу на різних платформах. Його інтегроване середовище розробки підтримує різноманітні мови програмування, що дозволяє розробникам працювати з різними технологіями в одному редакторі.

Однією з ключових переваг є розширюваність. VS Code має широкий спектр розширень, які полегшують роботу з конкретними мовами програмування, фреймворками чи іншими інструментами. Це робить його універсальним для різних типів проєктів та завдань.

Важливою рисою є вбудована система керування версіями, яка полегшує спільну роботу над проектами та забезпечує відстеження змін в коді. Інтерфейс користувача VS Code простий та інтуїтивно зрозумілий, що дозволяє новачкам швидко впроваджуватись у розробку.

Рис. 3.13. Visual Studio Code його взаємодія з мовою програмування Python

VS Code володіє потужною системою автодоповнення, що значно підвищує продуктивність розробників. Зручні функції відлагодження та вбудовані інструменти для роботи з Git роблять редактор ще більш привабливим для команд розробників.

Усі ці переваги роблять Visual Studio Code одним з найбільш популярних та широко використовуваних текстових редакторів у сфері програмування.

Для взаємодії Python з API Телеграм було використано відому бібліотеку PyTelegramBotAPI. Це потужний інструмент для роботи з Telegram API, що надає численні переваги для розробників ботів [2, 15-20].

По-перше, вона відзначається простотою використання, що дозволяє розробникам швидко реалізувати та налаштувати свого бота. До цього додається зручна документація, яка полегшує освоєння функціоналу бібліотеки.

По-друге, PyTelegramBotAPI має велику кількість функцій, які спрощують взаємодію з Telegram API. Це включає в себе обробку подій, реакцію на повідомлення користувачів, а також можливість взаємодії з різними типами повідомлень та клавіатурами [22].

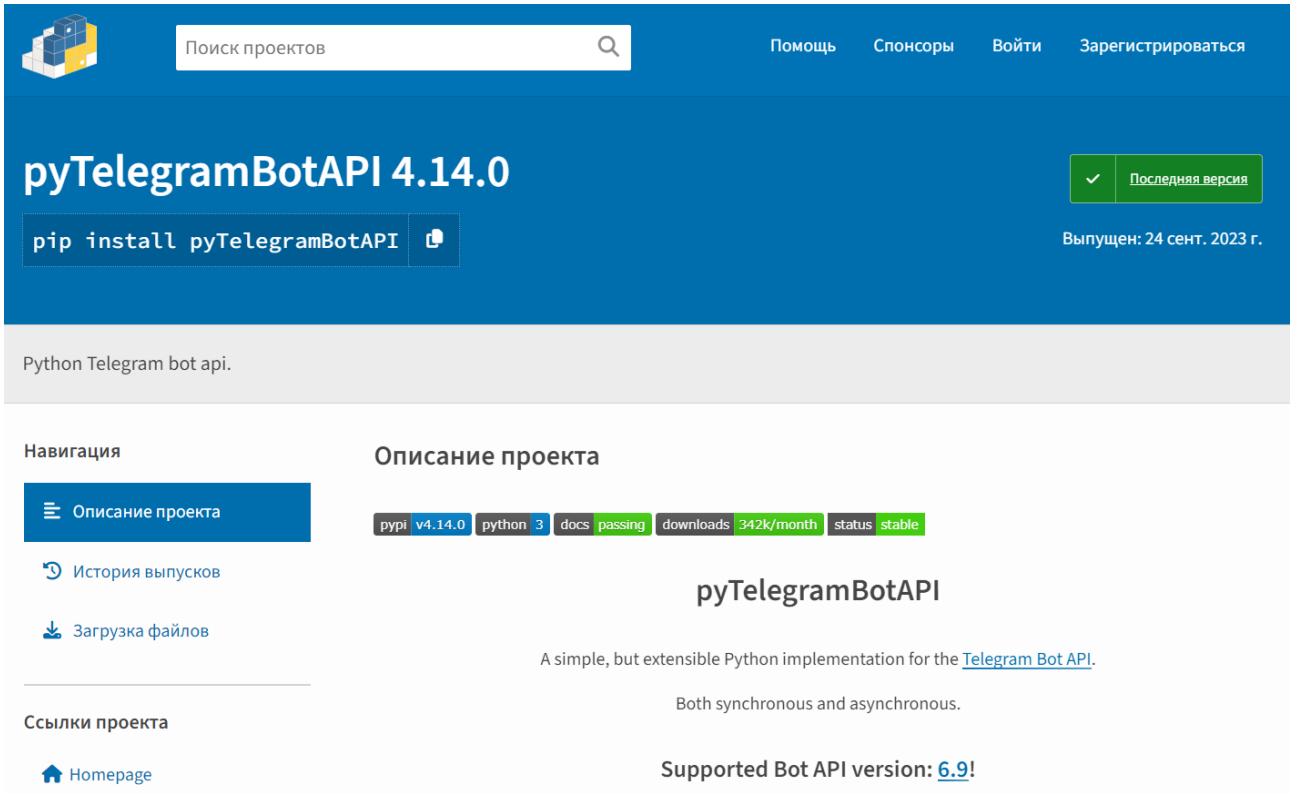

Рис. 3.14. Офіційний сайт з документацією до pyTelegramBotAPI

По-третє, бібліотека активно підтримується та оновлюється, що робить її надійним інструментом для розробки ботів. Регулярні виправлення помилок та вдосконалення забезпечують стабільну роботу ботів та забезпечують їх сумісність з останніми версіями Telegram API.

Зрештою, PyTelegramBotAPI підтримує використання асинхронного програмування, що робить її ефективною для великих та потужних ботів, які повинні обробляти великий потік запитань одночасно. Це робить бібліотеку привабливою для розробників, які прагнуть створювати високопродуктивні та швидкі боти для платформи Telegram.

3.3. Опис структури програмного забезпечення

Програмне забезпечення, створене за допомогою мови програмування Python має вигляд Телеграм боту, тобто машини-співрозмовника з форматом чату месенджера. Воно складається з декількох файлів з розширенням .py, які

відповідають за обробку програмного коду та зберігання токенів та ключів для підключення до Телеграму та бази даних MongoDB.

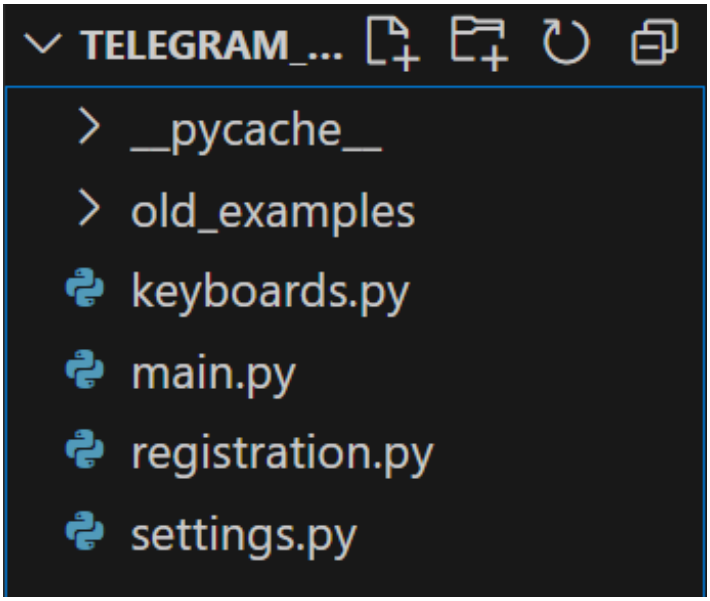

Рис. 3.15. Файли .py необхідні для роботи застосунку Телеграм боту

Файли програми відповідають за 4 окремих процеси:

- main.py – основна логіка програми, тут зберігаються відповіді бота, які він надає будь-якому користувачу;
- settings.py – налаштування системи, тут зберігаються підключення Python коду до Телеграм боту та до бази даних MongoDB;
- registration.py – реєстрація користувачів, тут обирається тип доступу: викладач, студент, батьки, а отримані реєстраційні дані залишаються у пам’яті системи;
- keyboards.py – вбудовані клавіатури для інтерактивної поведінки Телеграм боту, тут вони не лише зберігаються, а й генеруються для відображення динамічних даних.

Нижче будуть наведені рисунки, що демонструють основні функції та алгоритми, що необхідні для роботи програмного застосунку Телеграм боту.

settings.py

```python
import pymongo


#Основні налаштування бота та бази
data = {
    "bot" : {
        "tg_token" : "67    5218:AAH9-        Y9TvyWSpLqqH019vcJVw3E",
        "bot_name" : "QuizGo"
    },
    "db" : {
        "mongo_token" : "mongodb+srv://NTU_       go:       023@cluster0.n77qd.mongodb
        "mongo_db" : "QuizGo_db",
        "mongo_col_teachers" : "Teachers",
        "mongo_col_students" : "Students",
        "mongo_col_parents" : "Parents",
        "mongo_col_groups": "Groups",
        "mongo_col_active_games": "ActiveGames",
        "mongo_col_quizes" : "Quizes",
        "mongo_col_stats" : "Stats"
    }
}
```

Рис. 3.16. Файл settings.py та його словник з токенами та ключами

Файл settings.py зберігає інформацію у вигляді словника data окремо для Телеграм боту та бази даних MongoDB. Ключ tg_token є обов'язковим ідентифікатором для зв'язку коду з API Телеграм. Ключ mongo_token є обов'язковим для зв'язку коду з базою даних MongoDB. Обидва ключі видаються віддаленими сервісами для приватного використання у коді застосунку.

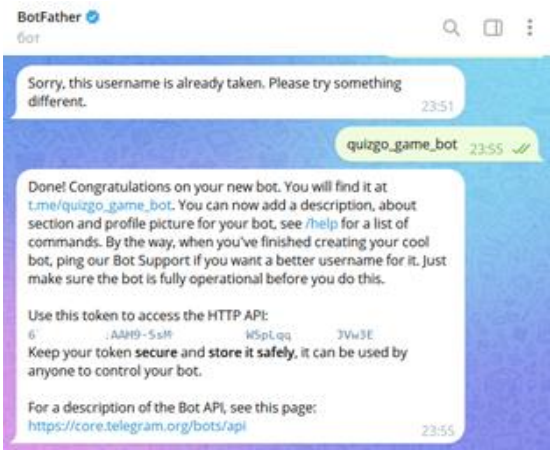

Рис. 3.17. Головний бот Телеграму BotFather згенерував приватний ключ

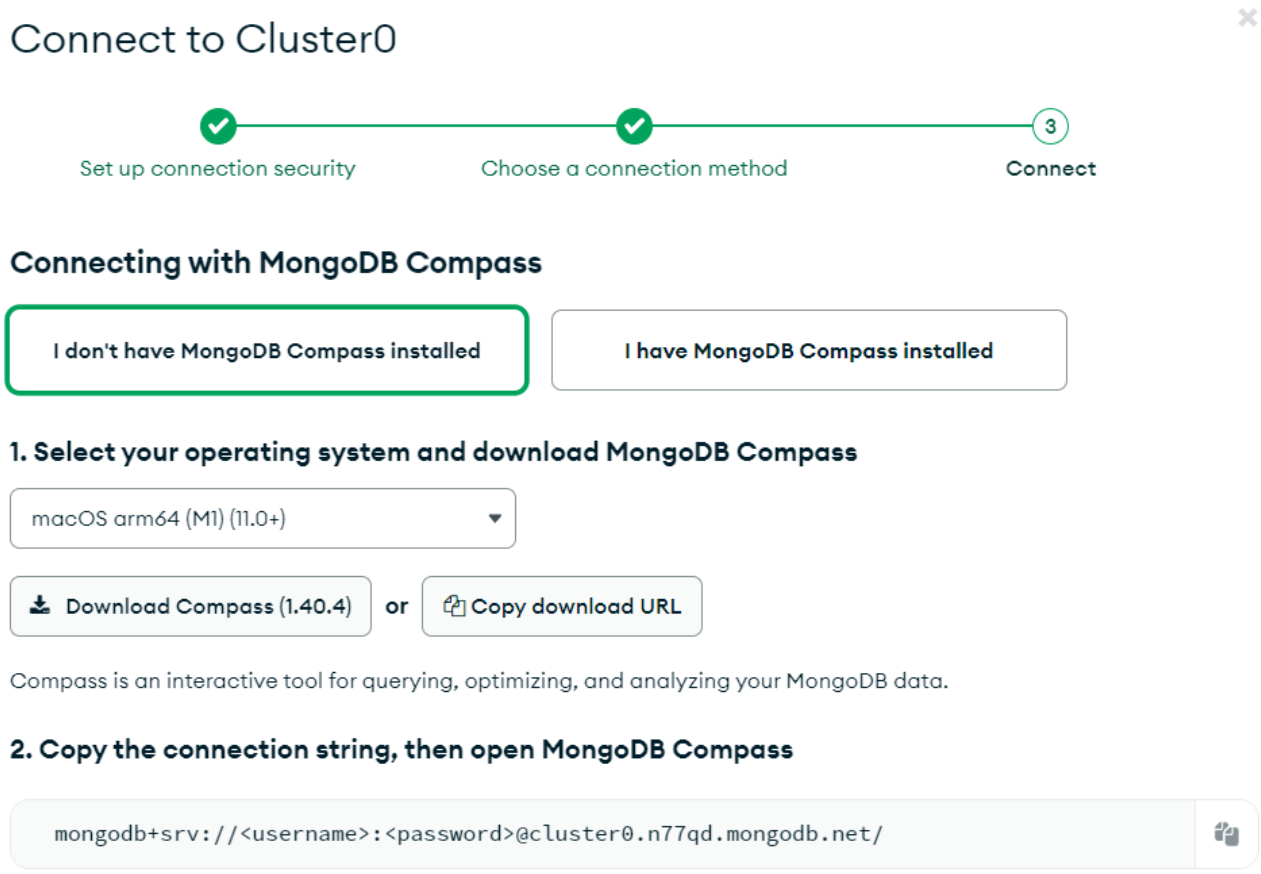

Рис. 3.18. MongoDB Atlas надає посилання, завдяки якому здійснюється підключення до коду

Програмний Python код під'єднано до віддалених сервісів, а завдяки модульності мови Python, ці підключення будуть активні навіть при роботі з іншими файлами програми. Імпортування файлів та функціоналу здійснено за допомогою оператора import.

Завдяки вбудованому функціоналу pyTelegramBotAPI, розробники можуть використовувати «хендлери», тобто спеціальні декоратори для реагування на повідомлення користувачів. Будь-яка взаємодія з ботом відслідковується «хендлером», а функція нижче оброблює запит згідно необхідної логіки програмного забезпечення. Система схожа на усім відомі веб-хуки, однак вони є вбудованою частиною існуючої бібліотеки. Завдяки «хендлерам» Телеграм бот перевіряє дані від callback клавіатури або ж перевіряє текст, що надійшов у вигляді текстового повідомлення.

```python
#Початок роботи бота
bot = telebot.TeleBot(settings.data["bot"]["tg_token"])


@bot.message_handler(commands=["start"])
def get_start(message):
    #Початок реєстрації
    registration.choose_role(bot, message)


@bot.message_handler(content_types=["text"])
def get_text(message):
    #Вибір ролі
    if message.text == "👨‍🎓 Студенти":
        registration.student_register(bot, message)
    elif message.text == "👨‍🏫 Викладачі":
        registration.teacher_register(bot, message)
    elif message.text == "👨 Батьки":
        registration.parent_register(bot, message)
```

Рис. 3.19. Файл main.py та його система «хендлерів»

Що стосується файлу для реєстрації, він також працює за системою «хендлерів». Спочатку він очікує дані для введення від користувача, а потім записує їх у базу даних та передає очікування наступному «хендлеру». Така система має назву покрокового «хендлеру» (next_step_handler). Саме завдяки такій системі, бот буде очікувати дані нескінчену кількість часу, а отже у людини буде достатньо часу на введення персональних даних.

```python
def get_user_lastname(user_lastname):
    user_data["lastname"] = user_lastname.text
    user_email = bot.send_message(message.chat.id, "Введіть поштову скриньку:")
    bot.register_next_step_handler(user_email, get_user_email)

def get_user_firstname(user_name):
    user_data["firstname"] = user_name.text
    user_lastname = bot.send_message(message.chat.id, "Введіть прізвище:")
    bot.register_next_step_handler(user_lastname, get_user_lastname)
```

Рис. 3.20. Використання покрокового «хендлеру» для отримання даних користувача

Останній програмний файл відповідає за побудову клавіатур, при чому він не тільки зберігає вже готові набори кнопок, а й генерує динамічні клавіатури там, де це необхідно. Прикладом є генерація клавіатур для відповідей на запитання вікторини, адже кожна відповідь є унікальною, як і їхня кількість. Тому важливо вміти правильно генерувати кнопки та обирати тип клавіатури. У цьому програмному застосунку використовуються одразу обидва типи. Перший працює на системі кол-беків, тобто спеціальних повідомлень, які є схованими від звичайного користувача (відбуваються на бек-енді програми), другий же працює як швидке текстове повідомлення, тобто функціонал такий же, я і у звичайного повідомлення, надрукованого власноруч. Використання емоджі у таких клавіатурах є дуже важливим не лише для естетичного вигляду боту, а й для легшої і безпечнішої ідентифікації запитів від користувача.

```
#Клавіатура для початку Quiz вікторини
start_game_keyboard = telebot.types.ReplyKeyboardMarkup(resize_keyboard=True, row_width=2)

start_game_button1 = telebot.types.KeyboardButton("📢 Розпочати гру")

start_game_keyboard.add(start_game_button1, back_button)


#Клавіатура для участі у Quiz вікторині
join_game_keyboard = telebot.types.ReplyKeyboardMarkup(resize_keyboard=True, row_width=2)

join_game_button1 = telebot.types.KeyboardButton("▶️ Приєднатися")
join_game_button2 = telebot.types.KeyboardButton("❌ Відмовитися")

join_game_keyboard.add(join_game_button1, join_game_button2)
```

Рис. 3.21. Створення заготовлених клавіатур з використанням емоджі

Як бачимо, функціонал Телеграм боту є дуже різноманітним і водночас інтуїтивно зрозумілим. Усі компоненти працюють дуже злагоджено, адже мають офіційні інтеграції один з одним. Також програма є дуже гнучкою, внесення змін не визиває дискомфорту, а усі додаткові профілі чи вікторини можна створювати та завантажувати у звичному JSON форматі «ключ-значення». Код боту є доволі зрозумілим та відкритим, а отже його також можна брати як приклад для навчання студентів на різних курсах з програмування.

3.4. Опис роботи програмного забезпечення

Робота з Телеграм ботом починається на етапі реєстрації користувача. Тому при першому звертанні бот буде з’ясовувати, чи зареєстрований цей користувач, чи йому необхідно запропонувати клавіатуру для реєстрації.

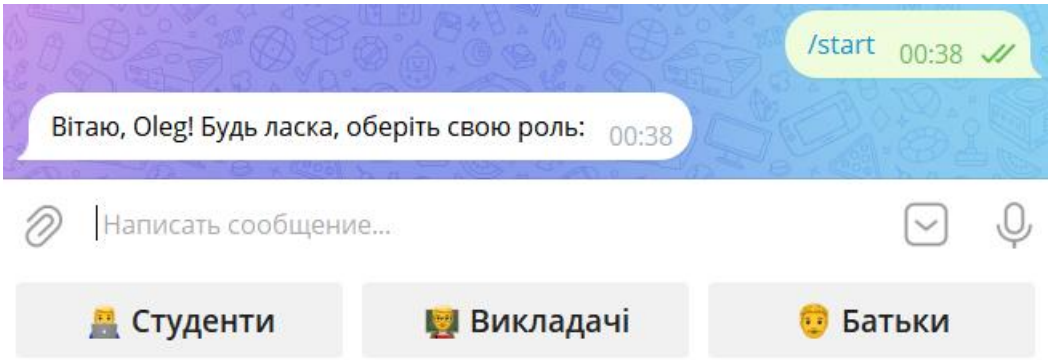

Рис. 3.22. Бот пропонує користувачу обрати роль

Після того, як роль було обрано, тиснемо на необхідну кнопку та продовжуємо введення даних, які необхідні для роботи бота. Першим користувачем буде викладач, він знайомиться з системою раніше за студентів.

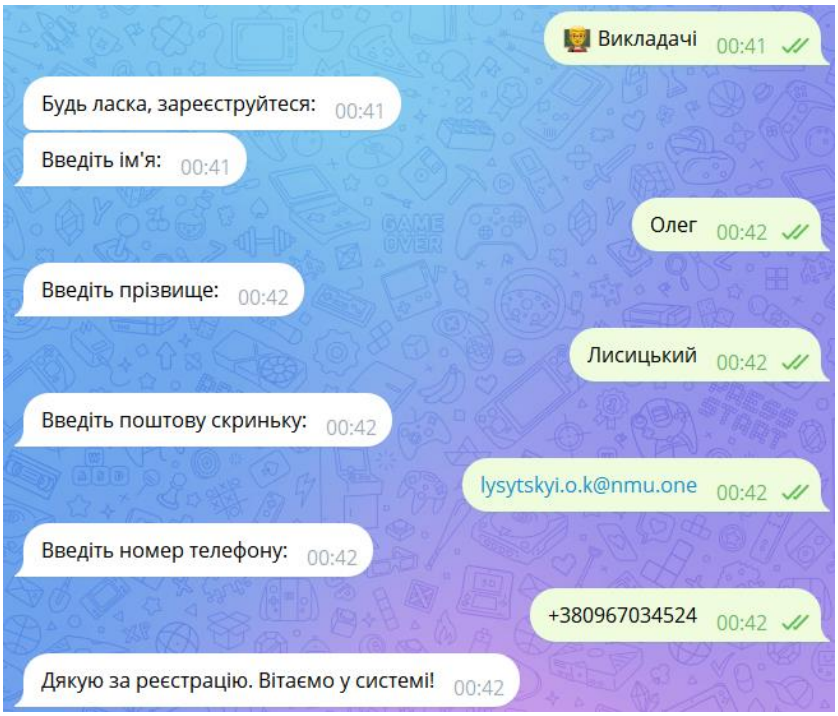

Рис. 3.23. Покрокове введення персональної інформації

Усі наступні спроби зайти у бота не будуть активувати процес реєстрації, адже система, спираючись на Телеграм ID, буде ідентифікувати користувача як існуючого.

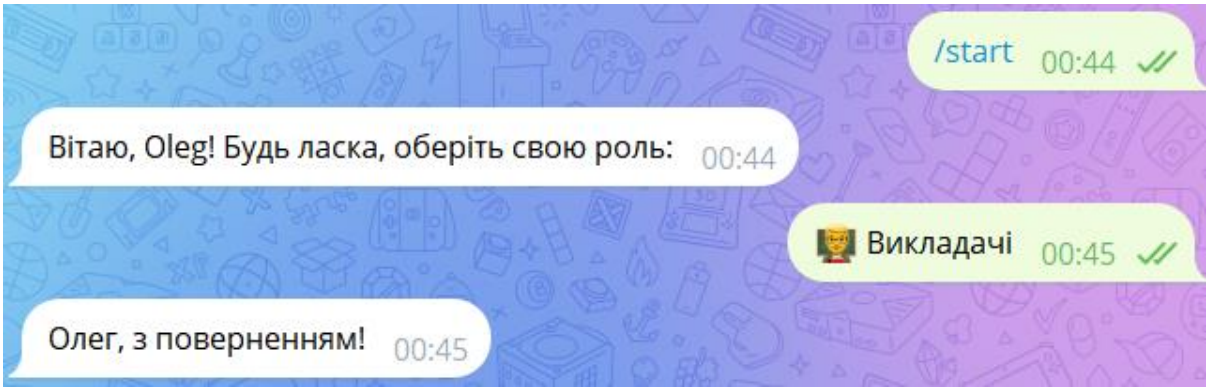

Рис. 3.24. Ідентифікація користувача, який повернувся до системи

Після того, як користувач-викладач зареєструвався, або ж повернувся до системи, йому пропонується меню з 4 кнопками. Кожна з кнопок робить запит до бази даних з метою пошуку необхідної інформації.

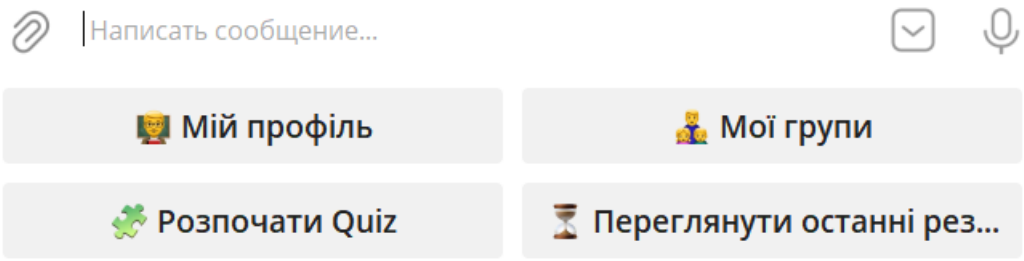

Рис. 3.25. Меню викладача

Обираємо кнопку «Мій профіль», вона має продемонструвати дані, вказані при реєстрації, а також відобразити групи, які були призначені адміністратором чи методистом навчального закладу.

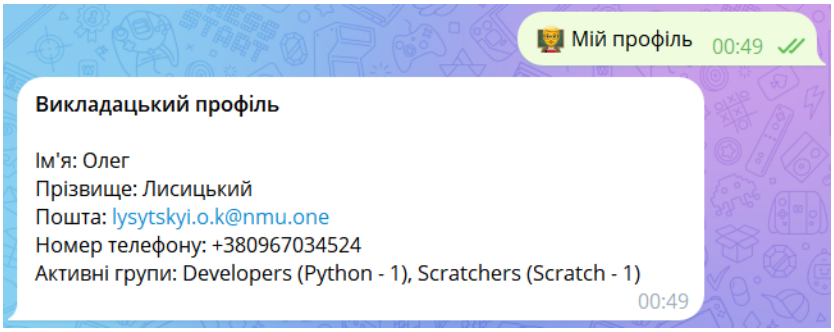

Рис. 3.26. Профіль викладача

Кнопка «Мої групи» відобразить детальну інформацію щодо груп, які прив’язані до викладача, незалежно від їхньої кількості. Система буде використовувати Телеграм ID для пошуку всередині системи.

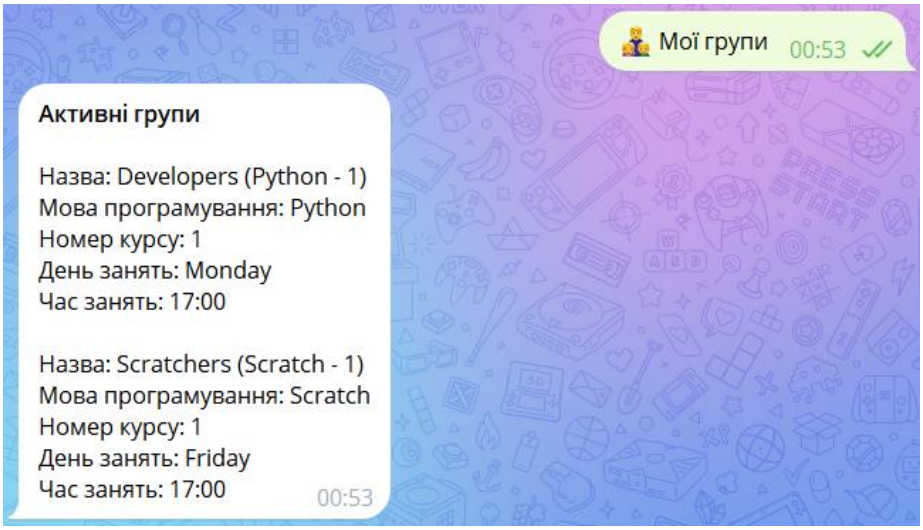

Рис. 3.27. Перелік активних груп викладача

Кнопка «Переглянути останні результати» продемонструє останні результати проходження вікторин серед своїх груп. Оскільки цей викладач ще не проводив вікторини, то й історія проведення наразі пуста.

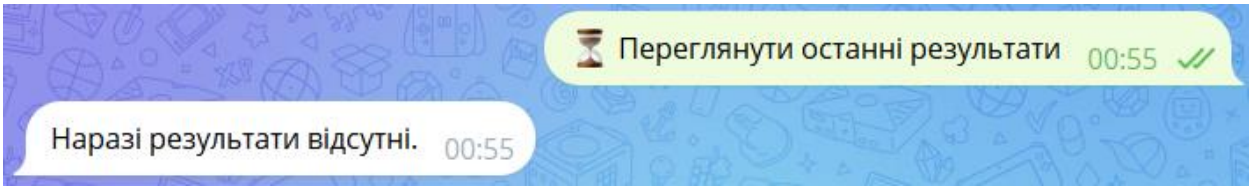

Рис. 3.28. Останні результати наразі пусті

Для того щоб отримати останні результати з вікторин, необхідно провести хоча б одну для будь-якої зі своїх груп. Однак система ще не має студентів, а отже їх необхідно зареєструвати. Реєстрація студента потребує не лише введення персональних даних, а й вибору існуючої групи за допомогою інтерактивної клавіатури.

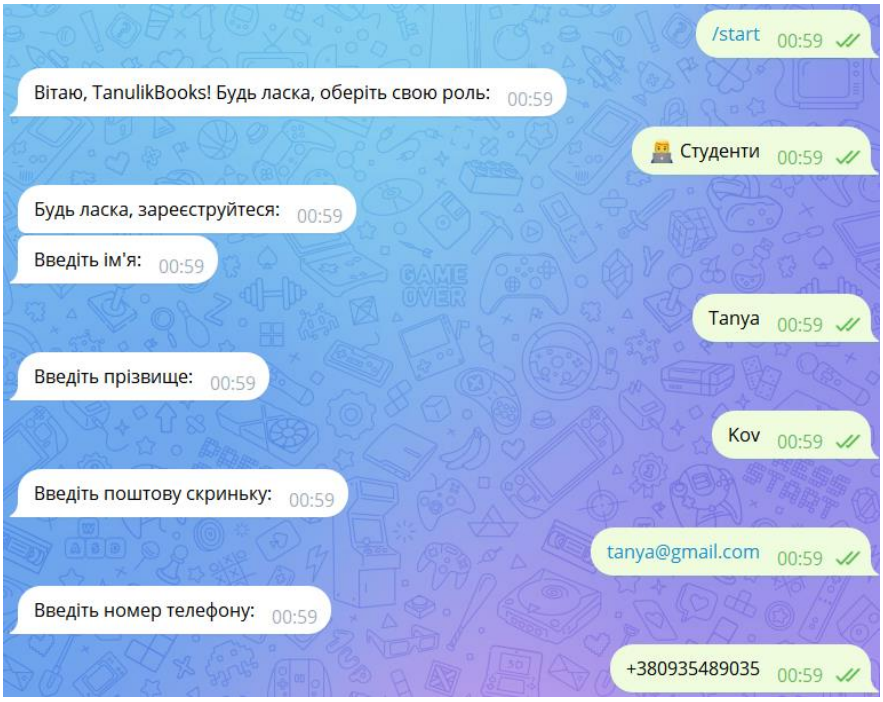

Рис. 3.29. Процес реєстрації користувача-студента

Після того, як усі поля були заповнені, необхідно обрати групу, з якою будуть синхронізовані усі майбутні події, в тому числі запрошення до вікторини.

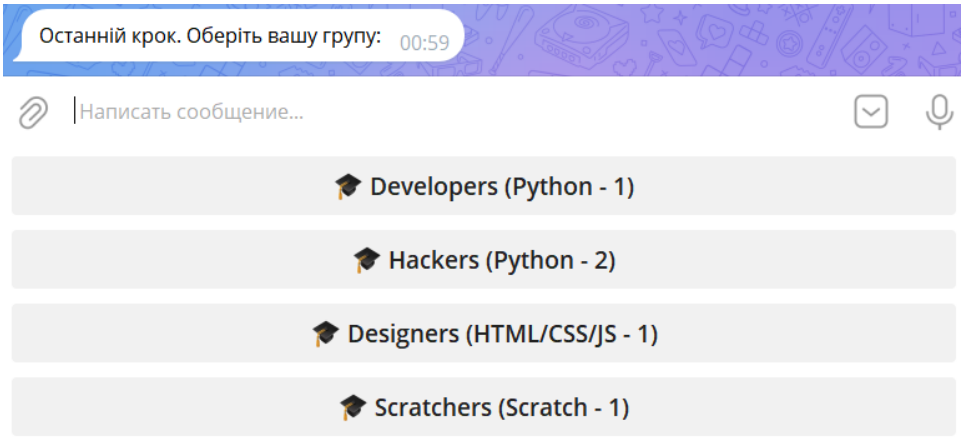

Рис. 3.30. Процес вибору групи студентом

Якщо група успішно обрана, система вітається зі студентом та пропонує переглянути його профіль.

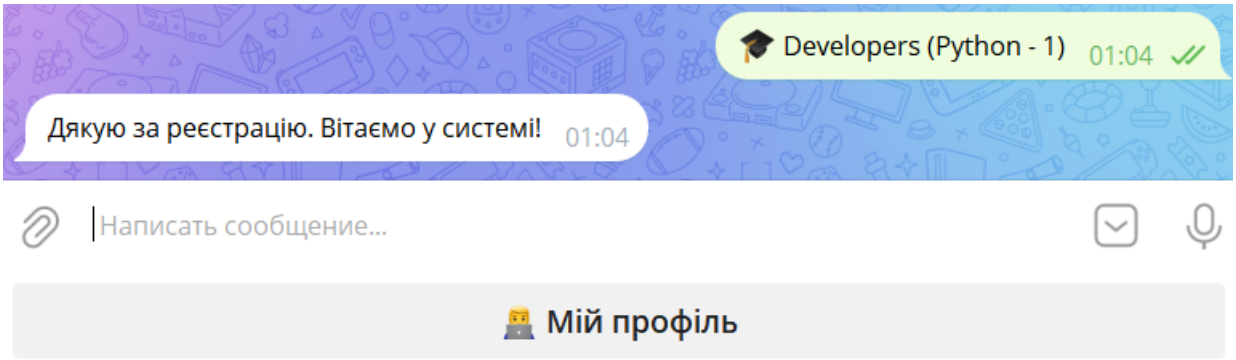

Рис. 3.31. Вітальне меню для студента

Переходимо до меню користувача-студента, щоб ознайомитися з його профілем.

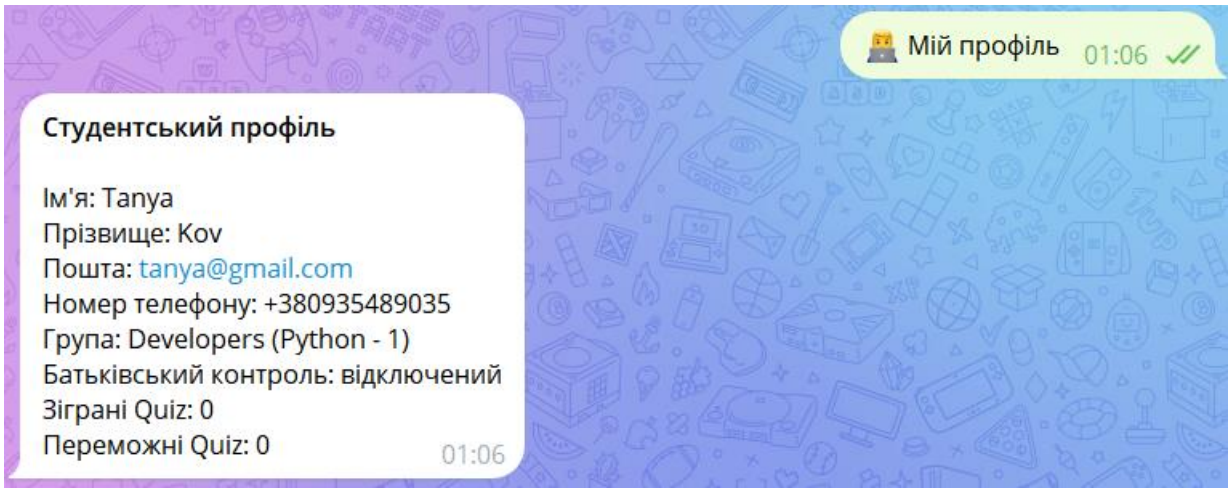

Рис. 3.32. Перегляд профілю користувача-студента

Окрім полів, які було надано студентом, бачимо системно згенеровані поля, що відповідають за батьківський контроль та за кількість зіграних та переможних вікторин. Ці три поля можуть змінюватися в залежності від інших факторів. Наприклад, якщо викладач розпочне вікторину для цієї групи, або ж хтось з батьків підключить студенту функцію батьківського контролю. Для цього батькам необхідно також провести реєстрацію та, після введення необхідних

контактних даних, обрати свою дитину у вигляді кнопки на інтерактивній клавіатурі.

Рис. 3.33. Процес реєстрації для батьків

Після того, як зареєстровані батьки підключили батьківський контроль, вони мають дві доступні функції у меню. Це перегляд власного профілю та перегляд профілю дитини.

Рис. 3.34. Меню для користувачів-батьків

При виборі тієї чи іншої кнопки, система зробить запит до бази даних та знайде усі необхідні дані для відображення у чаті.

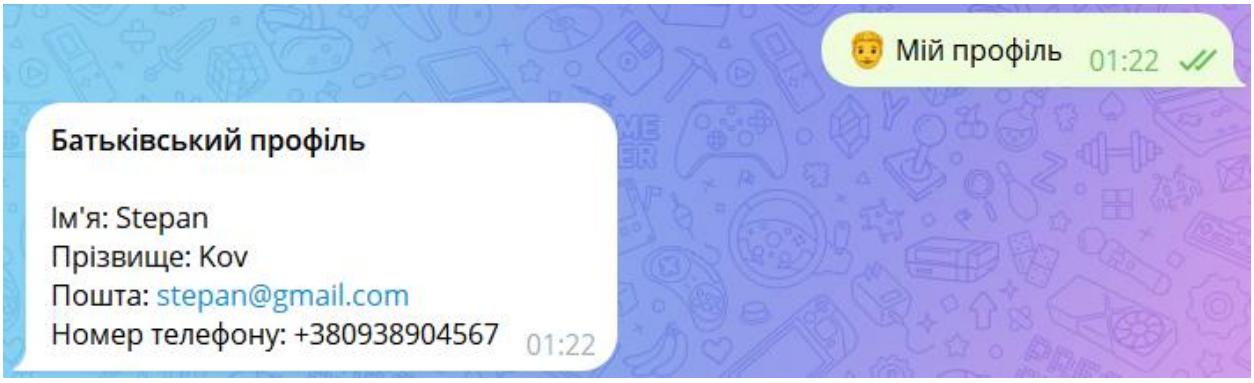

Рис. 3.35. Перегляд батьківського профілю

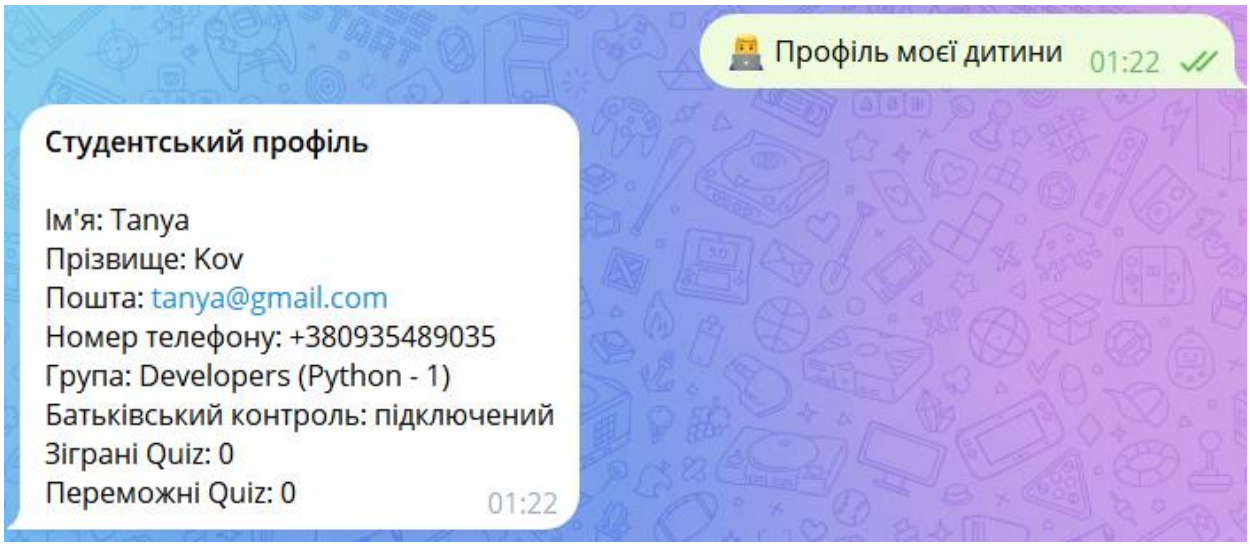

Рис. 3.36. Перегляд профілю дитини під батьківським акаунтом

Як бачимо, батьківський контроль автоматично активовано, а отже батьки зможуть отримувати повідомлення щодо поведінки дитини під час проходження вікторини.

Оскільки викладач, студенти, батьки зареєстровані, спробуємо розпочати одну з існуючих вікторин. Для цього викладач має провести взаємодію з меню, використовуючи власний викладацький профіль.

Переходимо до викладацького меню та обираємо пункт «Розпочати Quiz». Саме цей пункт відповідає за налаштування вікторини перед її початком.

Викладачу будуть запропоновані можливі вікторини, а також перелік його груп для проведення ігор.

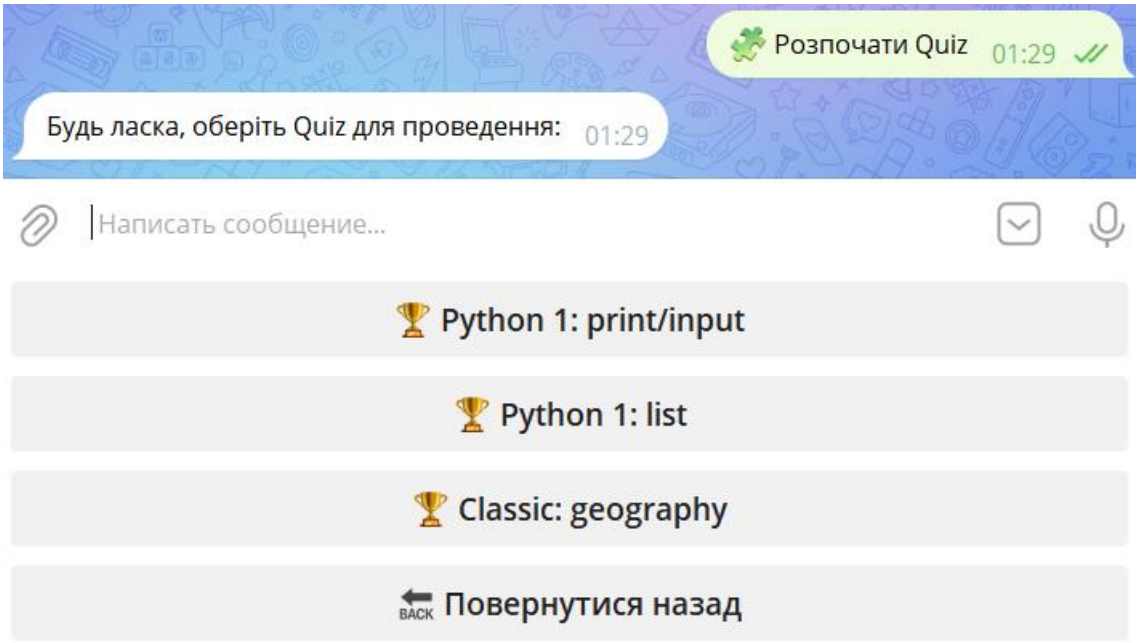

Рис. 3.37. Налаштування майбутньої вікторини викладачем

Як бачимо, система автоматично формує клавіатуру з вибором однієї з трьох існуючих вікторин. Обираємо першу та перейдемо до вибору групи.

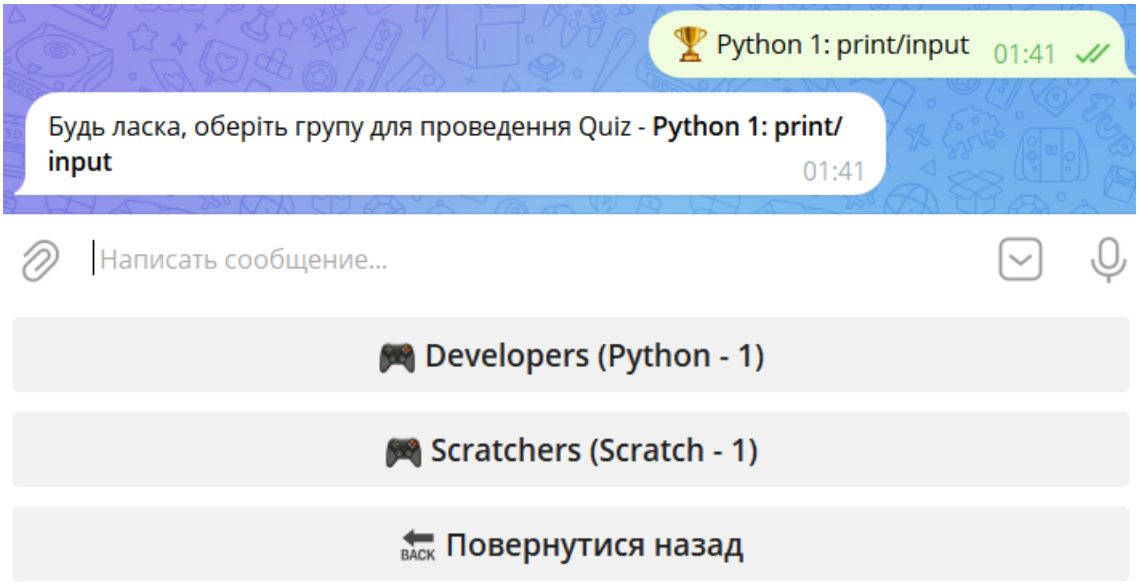

Рис. 3.38. Вибір групи для проведення вікторини

Система пропонує на вибір дві групи, які були прив’язані до викладача адміністратором чи методистом навчального закладу. При виборі будь-якої з них студентам надійде повідомлення щодо участі, на яке вони матимуть відреагувати погодженням або відмовою.

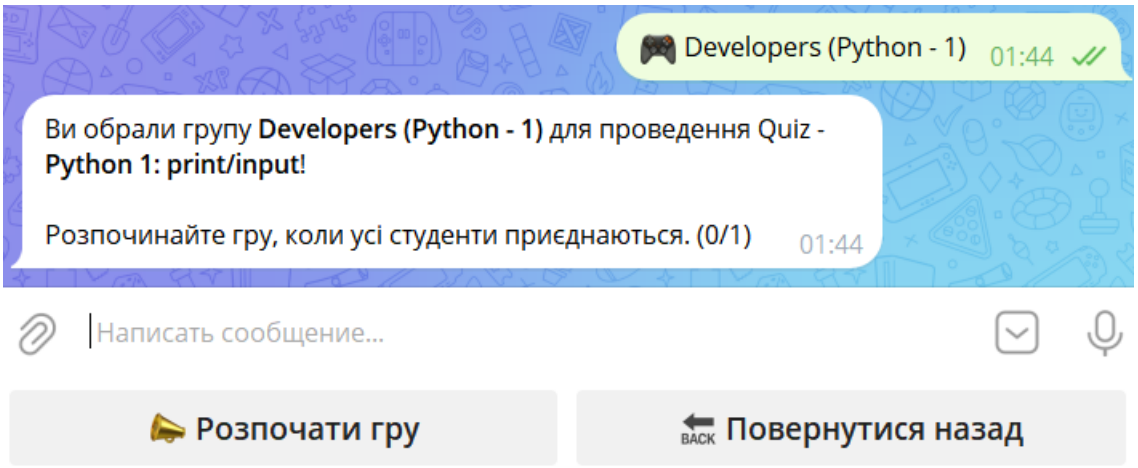

Рис. 3.39. Вибір групи з надсиланням запрошень для участі

На цьому етапі викладач очікує студентів для приєднання. Якщо ж усі відмовляться брати участь – вікторину буде завершено. Якщо ж усі погодяться грати, викладач зможе розпочати її натисканням однієї кнопки.

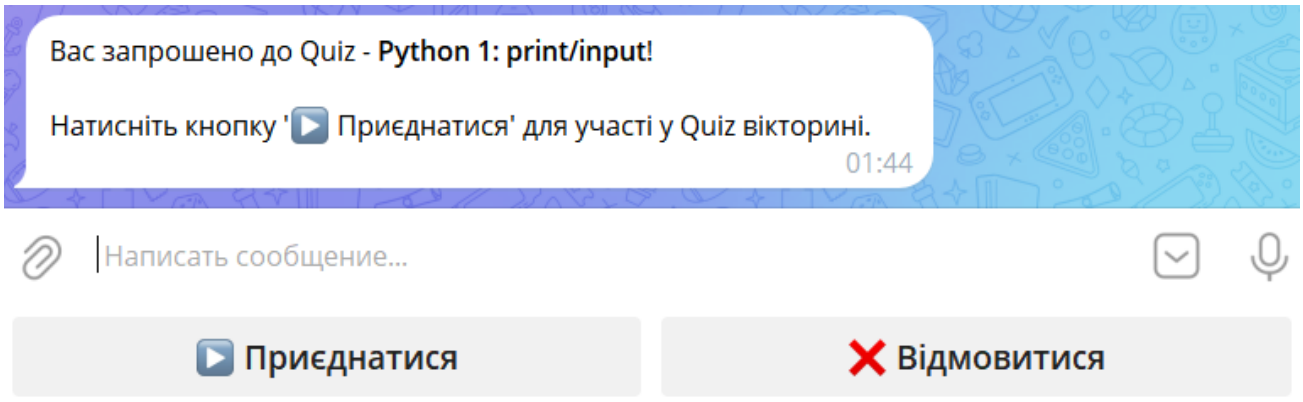

Рис. 3.40. Зовнішній вигляд запрошення для участь у вікторині

Спробуємо ситуацію, де студент відмовляється від участі. Це має призвести до завершення вікторини, а також повідомити батьків про відмову від участі їхньої дитини.

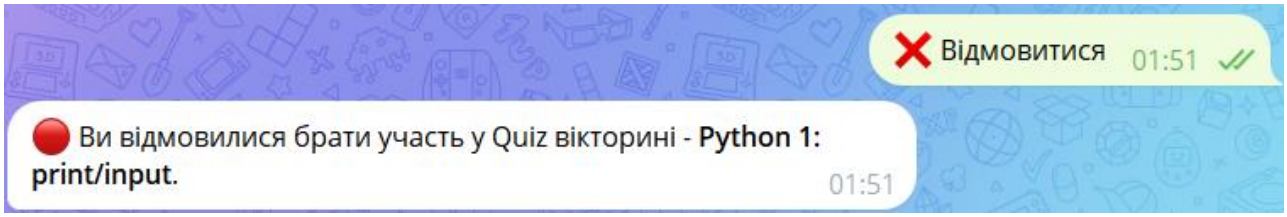

Рис. 3.41. Відмова від участі у вікторині з боку студента

Подібна поведінка надсилає повідомлення викладачу та батькам дитини. Викладач має змогу завершити вікторину, якщо вважатиме за потрібне, а батьки будуть проінформовані щодо вибору дитини.

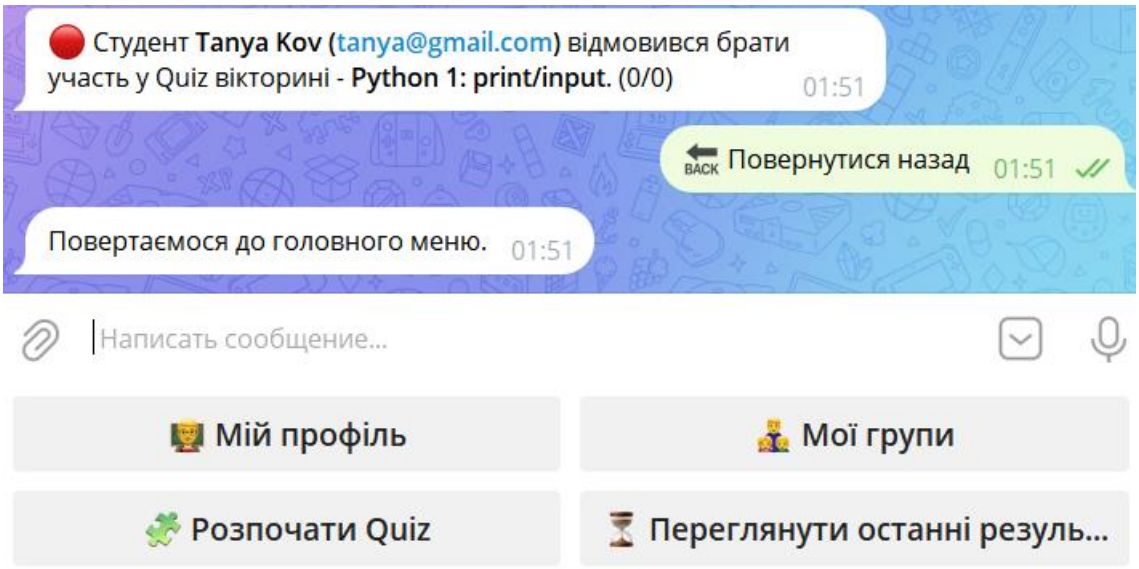

Рис. 3.42. Процес завершення вікторини збоку викладача

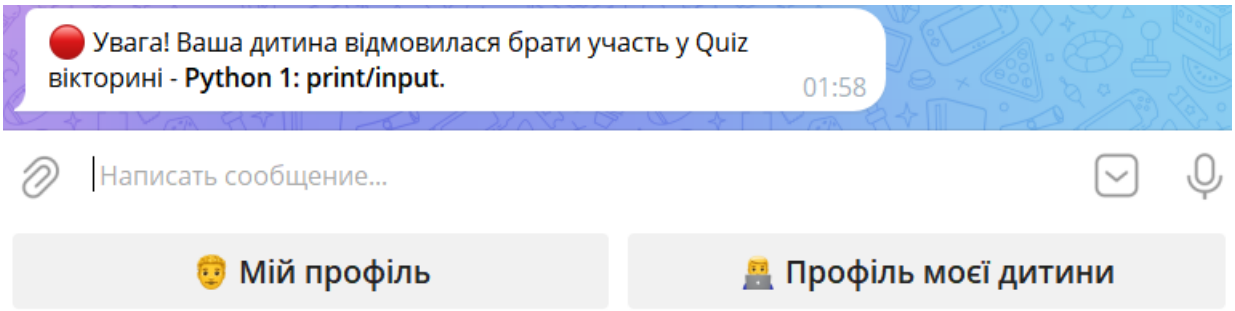

Рис. 3.43. Інформування батьків щодо вибору дитини

Тепер можна переглянути інший варіант розвитку подій, коли студент погоджується взяти участь у вікторині. В цьому випадку усі зацікавлені користувачі отримають позитивні повідомлення.

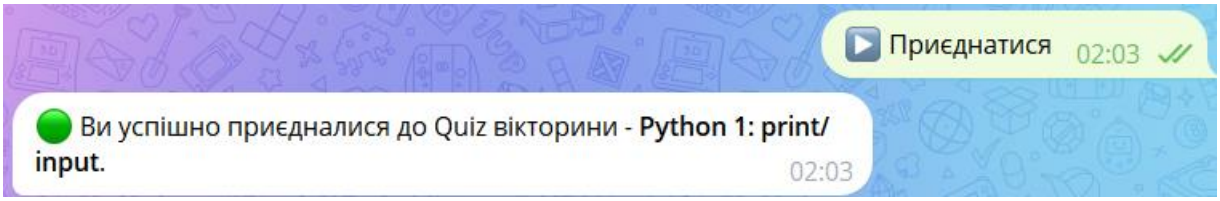

Рис. 3.44. Студент успішно приєднався до участі у вікторині

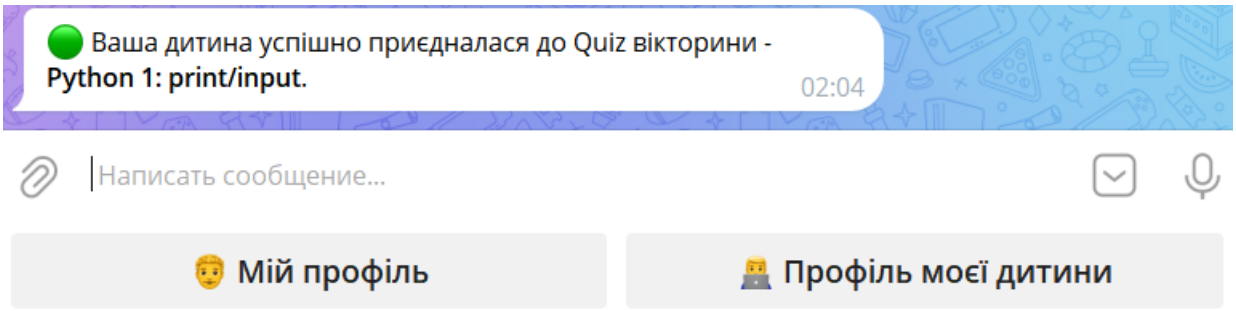

Рис. 3.45. Повідомлення щодо успішного приєднання для батьків

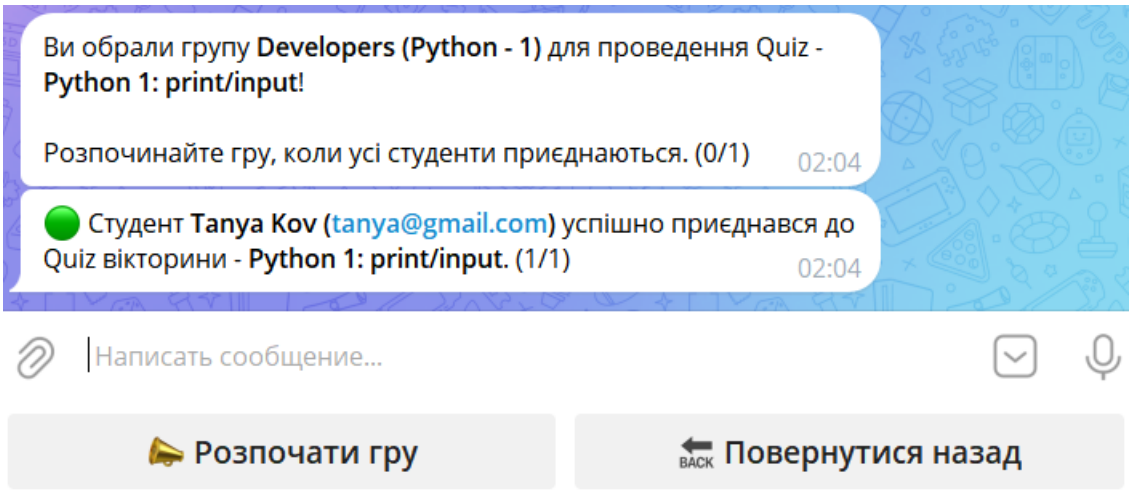

Рис. 3.46. Викладач готовий розпочинати вікторину

Якщо натиснути кнопку «Розпочати гру», усі студенти, які є готовими до вікторини, отримають перші питання та можливість відповідати на них. Система

почне підраховувати час їхнього проходження та нараховуватиме бали за правильні відповіді.

Рис. 3.47. Викладач розпочинає вікторину

Викладач отримує повідомлення про початок, а студенти, у свою чергу, отримують перше запитання на обирають відповідь на нього за допомогою автоматично сформованої клавіатури.

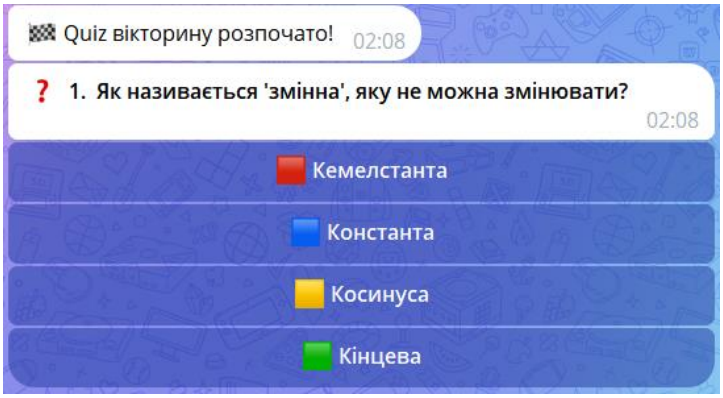

Рис. 3.48. Поява першого питання з автоматично згенерованою клавіатурою

Якщо студент обирає вірний варіант, система відмічає питання зеленим кольором та підтверджує відповідь. При виборі помилкового варіанту, система відмічає питання червоним та демонструє ту відповідь, яка була правильною.

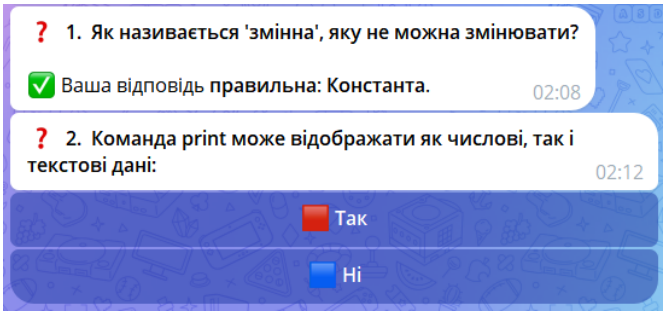

Рис. 3.49. Зовнішній вигляд правильної відповіді на запитання

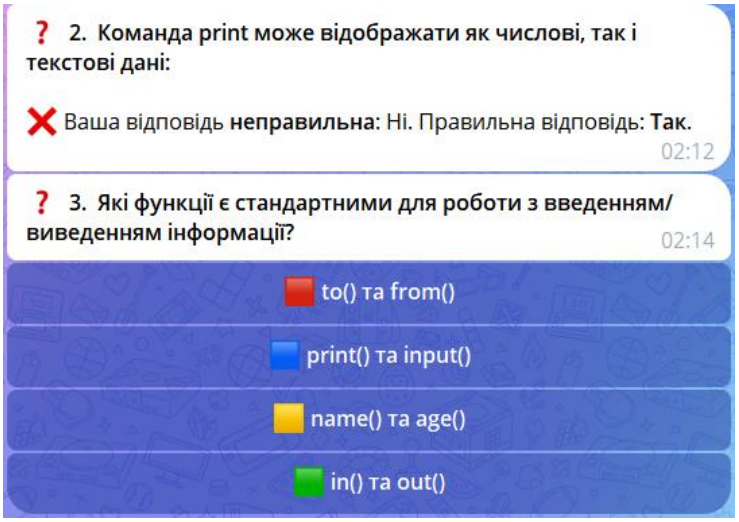

Рис. 3.50. Зовнішній вигляд неправильної відповіді на запитання

Студент продовжує відповідати на запитання, крок за кроком, оновлюючи власні показники набраних балів та витраченого часу. У кінці вікторини система матиме змогу зробити підрахунок очок за вже згаданою формулою та розподілити місця на дошці лідерів.

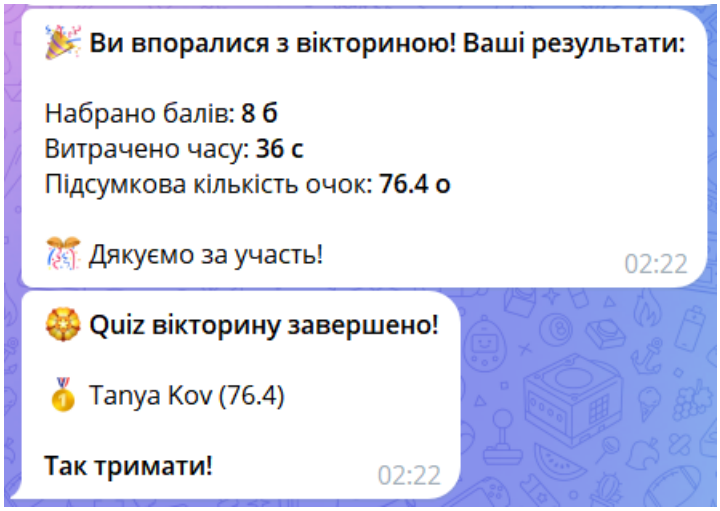

Рис. 3.51. Підрахунок результатів після завершення вікторини

Як бачимо, студент самостійно брав участь у вікторині, відповів вірно на 8 запитань та витратив 36 секунд часу. Згідно зі згаданою формулою отримує 76.4 очка. Система також адаптується під більшу кількість гравців. Саме той, хто

останнім закінчить вікторину, буде слугувати сигналом для підрахунку таблиці лідерів.

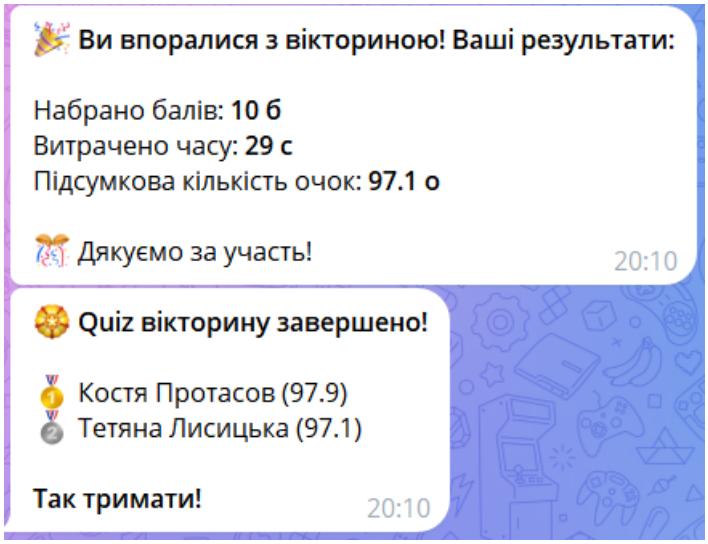

Рис. 3.52. Підрахунок таблиці лідерів з більшою кількістю гравців

Викладач також отримує повідомлення щодо завершення вікторини та може переглянути таблицю лідерів.

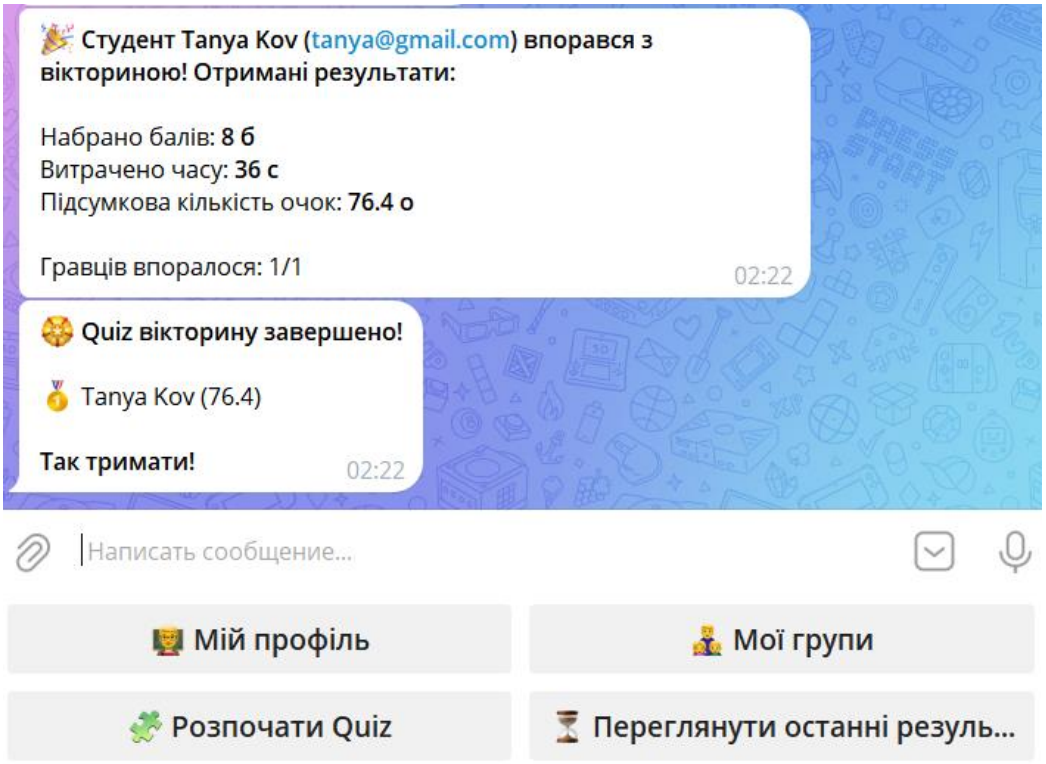

Рис. 3.53. Завершення вікторини для викладача

Вікторини успішно завершено. Студенти отримали відмітки у власних профілях, а викладач отримує можливість переглянути останні результати його груп, що брали участь у вікторинах.

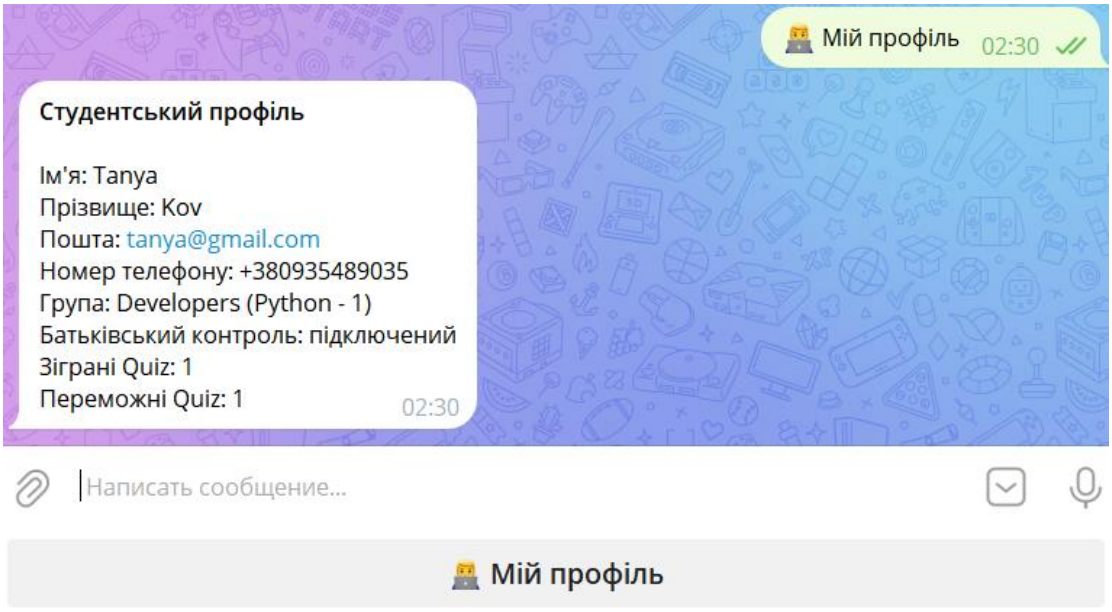

Рис. 3.54. Змінений профіль студента після перемоги у вікторині

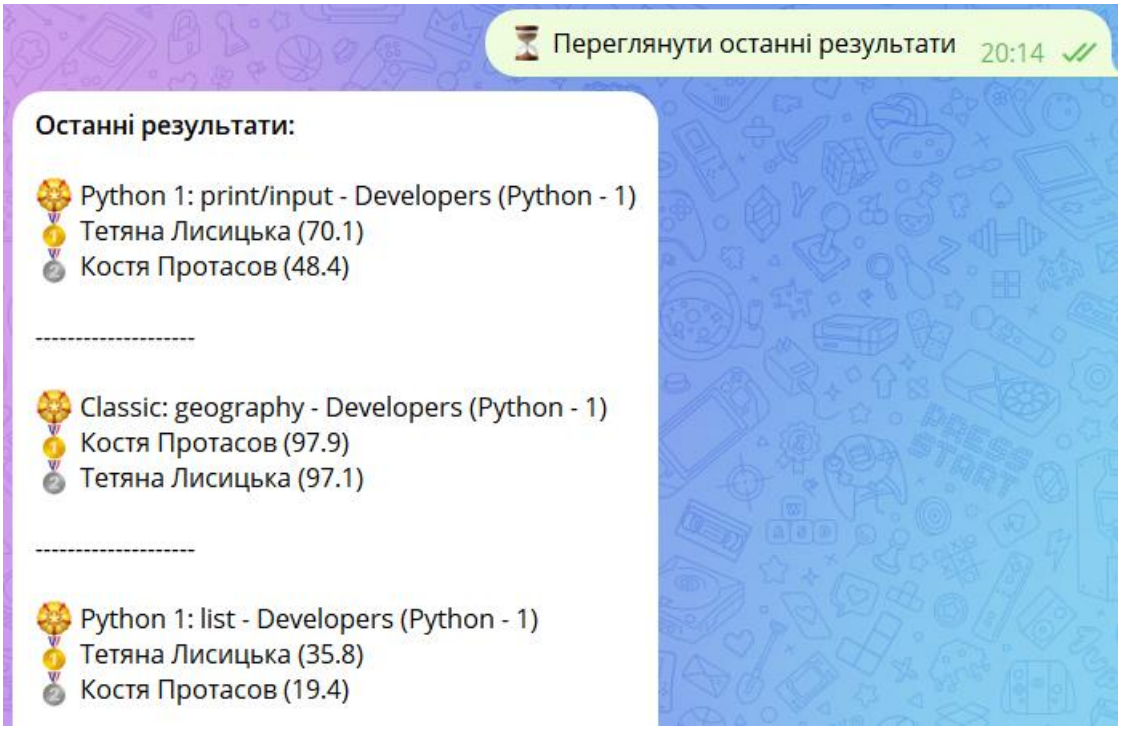

Рис. 3.55. Останні результати вікторин, проведених викладачем

3.5. Висновки

У цьому розділі було розглянуто основні аспекти, які дозволили програмному застосунку працювати як єдине ціле. Були описані переваги використання хмарної нереляційної бази даних MongoDB, мови програмування Python, редактору коду Visual Studio Code та бібліотеки для взаємодії з Телеграм API – pyTelegramBotAPI. Кожен з цих засобів відіграє важливу роль для побудови екосистеми Телеграм боту, який має змогу проводити вікторини для студентів різних навчальних закладів.

База даних містить спеціальні ключі для синхронізації документів та їхньої взаємодії. Python пропонує функції і алгоритми, що займаються як внутрішніми підрахунками, так і зовнішнім виглядом боту.

Окремо було розглянуто взаємодію трьох різних типів користувачів із системою Телеграм боту. Студенти, викладачі та батьки крок за кроком пройшли реєстрацію, внесли особисті дані, обрали необхідні події за допомогою інтерактивних клавіатур. Були протестовані різноманітні повідомлення та різні варіанти розвитку подій всередині системи. Навіть при неочікуваній поведінці, Телеграм бот був готовий допомогти користувачеві отримати необхідну інформацію та перейти до потрібного меню. Результати ігор збережені, а робота з «архівом» є можливою навіть через велику кількість часу у майбутньому.

Таким чином, можна стверджувати, що завдання зі створенням гейміфікованого Телеграм застосунку виконано повністю, адже він покриває усі кейси, що потрібні кінцевому користувачу.

ВИСНОВКИ

У даній роботі було проведено дослідження та розроблено телеграм-бот для проведення гейміфікованих вікторин прямо у месенджері. Для реалізації проєкту використовувалися сучасні технології, такі як мова програмування Python, бібліотека pyTelegramBotAPI, хмарне середовище та база даних MongoDB. Головна ідея роботи полягає в інтеграції гейміфікованого підходу до навчання безпосередньо в месенджері, що дозволяє вчителям, студентам та батькам зручно брати участь у навчальному процесі.

Гейміфікація, використовуючи ідею платформи Kahoot та телеграм-бот, виявилася вельми ефективною стратегією навчання та мотивації студентів. Переваги цього підходу включають залучення студентів, створення мотивації до активного навчання, а також ефективне засвоєння матеріалу через інтерактивний контент.

Статистичні дані підтверджують позитивний вплив гейміфікації на навчання, де 80% студентів вважають, що цей метод збільшує їхню мотивацію, а 63% вчителів повідомляють про покращення результатів навчання. Такий підхід дозволяє зробити навчання цікавим, поліпшити розуміння та засвоєння навчального матеріалу.

Телеграм-бот, розроблений у рамках роботи, відкриває перспективи для подальшого розвитку у напрямку "школи в смартфоні", забезпечуючи зручний доступ до навчання через відомий месенджер. Його швидкість, доступність та інтеграція всіх необхідних функцій надають новий рівень комфорту та ефективності навчання для всіх учасників.

У майбутньому, можливий розвиток телеграм-боту передбачає розширення функціоналу та вдосконалення системи для надання користувачам ще більше можливостей для ефективного навчання. Такий інноваційний підхід до навчання демонструє актуальність та ефективність використання сучасних технологій в освітньому процесі, відзначаючи важливий крок у напрямку сучасної освіти.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. "Гейміфікація в бізнесі: збільште продуктивність за допомогою ігрових стратегій" / / В.П. Коваленко, І.М. Семенов, Ю.О. Лисенко; Видавництво "Бізнес Гейміфікація", Київ, 2022. – 120 с.
2. "Телеграм боти: від ідеї до успішного запуску" / / О.Г. Петренко, Н.А. Мельникова, В.І. Григоренко; Видавництво "БотСтудія", Харків, 2023. – 88 с.
3. "Нереляційні бази даних в сучасному програмуванні: практичні аспекти використання" / / С.В. Шевченко, І.О. Козлов, О.М. Литвиненко; Нац. ун-т інформаційних технологій та управління, Київ, 2022. – 104 с.
4. "Ігровий дизайн та гейміфікація в освіті: нові можливості для вчителів" / / Л.І. Кравченко, В.М. Соловйов, О.В. Білецька; Видавництво "Освітні Ігри", Львів, 2023. – 72 с.
5. "Технології майбутнього: інтеграція гейміфікації та штучного інтелекту в інтерфейсах користувача" / / А.М. Іванов, Є.С. Литвиненко, С.П. Гребінкін; Видавництво "TechInnovate", Одеса, 2022. – 96 с.
6. "Гейміфікація в маркетингу: як залучити та утримати клієнтів за допомогою ігрових стратегій" / / О.І. Сидоренко, К.О. Марченко, Л.М. Гончаренко; Видавництво "МаркетГейм", Херсон, 2023. – 110 с.
7. "Розробка та впровадження інтерактивних навчальних ботів для онлайн-освіти" / / І.О. Кравченко, А.В. Петров, Н.С. Сергієнко; Видавництво "EduTech", Київ, 2022. – 84 с.
8. "Новітні технології управління проектами: роль гейміфікації в досягненні цілей" / / В.В. Козлов, О.П. Іванова, С.М. Григоренко; Видавництво "ПроектМайстер", Львів, 2023. – 92 с.
9. "Гейміфікація в медицині: ігри для покращення фізичного та психічного здоров'я" / / М.А. Зайцев, І.С. Кулик, О.М. Чернишева; Медичне видавництво "HealthyGames", Київ, 2022. – 78 с.

10. Sebastian Deterding "Mattering: Toward a New Theory of Gamification" / MIT Press – Cambridge, 2011 – 224 с.
11. Kevin Werbach, Dan Hunter "For the Win: How Game Thinking Can Revolutionize Your Business" / Wharton Digital Press – Philadelphia, 2012 – 272 с.
12. Amy Jo Kim "Game Thinking: Innovate Smarter & Drive Deep Engagement with Design Techniques from Hit Games" / Ideapress Publishing – California, 2018 – 280 с.
13. Brian Burke "Gamify: How Gamification Motivates People to Do Extraordinary Things" / Bibliomotion – Massachusetts, 2014 – 224 с.
14. Jane McGonigal "SuperBetter: A Revolutionary Approach to Getting Stronger, Happier, Braver and More Resilient" / Penguin Books – New York, 2015 – 480 с. 5.
15. "Developing Telegram Bots with Python: A Step-by-Step Guide" / / Режим доступу: https://www.examplearticle1.com/developing-telegram-bots-python (дата звернення: 15.09.2023).
16. "Pythonic Strategies for Creating Powerful Telegram Bots" / / Режим доступу: https://www.examplearticle2.com/pythonic-strategies-telegram-bots (дата звернення: 26.09.2023).
17. "Mastering the Art of Telegram Bot Development with Python" / / Режим доступу: https://www.examplearticle3.com/mastering-telegram-bot-development-python (дата звернення: 13.10.2023).
18. "Best Practices and Tips for Effective Telegram Bot Development using Python" / / Режим доступу: https://www.examplearticle4.com/best-practices-tips-telegram-bot-python (дата звернення: 27.10.2023).
19. "Unleashing Pythonic Power: Building Innovative Telegram Bots" / / Режим доступу: https://www.examplearticle5.com/unleashing-pythonic-power-telegram-bots (дата звернення: 14.11.2023).

20. "Python-Powered Telegram Bots: Enhancing User Engagement and Functionality" / / Режим доступу: https://www.examplearticle6.com/python-powered-telegram-bots-enhancement (дата звернення: 17.09.2023).
21. "Python for Data Science: A Hands-On Guide to Analyzing and Visualizing Data" / / Дж. Р. МакГлоклен, О. В. Дж. Груммонд; Видавництво "DataInsights", Лондон, 2021. – 240 с.
22. "Web Development with Python and Django: Building Scalable Web Applications" / / С. Мітчелл; Технічне видавництво "WebMasters", Сан-Франциско, 2019. – 320 с.
23. "Python Automation: Streamlining Repetitive Tasks with Python Scripts" / / А. Волкан; Видавництво "ScriptMasters", Нью-Йорк, 2020. – 180 с.
24. "Machine Learning with Python: Practical Approaches for Predictive Analysis" / / К. Сміт, Р. Дж. Картер; Видавництво "MLPublishers", Сіетл, 2018. – 280 с.
25. "Python for Artificial Intelligence: A Comprehensive Guide to Implementing AI Algorithms" / / Е. Харріс, М. Томсон; Видавництво "AIPython", Торонто, 2022. – 340 с.
26. "Python Security: Protecting Your Code from Vulnerabilities and Attacks" / / Б. Андерсон, Г. І. Сміт; Видавництво "SecurePython", Дублін, 2019. – 200 с.
27. "Quantitative Finance with Python: Strategies for Financial Modeling and Analysis" / / Д. Мартін, С. Р. Шарма; Видавництво "QuantFinancePress", Париж, 2021. – 260 с.
28. "Python Robotics: Building Intelligent Robots with Python Programming" / / І. Лі, Л. Чжан; Видавництво "RobotTech", Сеул, 2017. – 310 с.
29. "Python Game Development: Creating Interactive Games with Python and Pygame" / / Ф. Гарсія, А. Лопез; Видавництво "GameMasters", Барселона, 2016. – 230 с.
30. "Natural Language Processing with Python: Unlocking the Power of Text Analysis" / / В. Сміт, К. Джонсон; Видавництво "NLPublishers", Сідней, 2020. – 290 с.

31. "The Science of Learning: How Cognitive Neuroscience Can Enhance Educational Practices" / / Д. Кларк, А. Сміт; Видавництво "CognitiveEd", Лос-Анджелес, 2019. – 320 с.
32. "Mindset Mastery: Unleashing the Power of Growth Mindset in Education" / / К. Двейк, Р. Мартінес; Видавництво "MindfulLearn", Сінгапур, 2021. – 280 с.
33. "The Psychology of Effective Teaching: Strategies for Engaging and Motivating Students" / / Е. Девіс, К. Тейлор; Видавництво "TeachPsych", Торонто, 2018. – 250 с.
34. "Learning Styles and Instructional Design: Tailoring Educational Approaches to Individual Differences" / / Л. Вонг, С. МакКензі; Видавництво "AdaptEd", Стокгольм, 2020. – 310 с.
35. "Emotional Intelligence in Education: Enhancing Academic and Social Success" / / З. Карім, А. Джонсон; Видавництво "EmoLearn", Лондон, 2017. – 270 с.
36. "Effective Feedback: A Guide for Educators on Providing Constructive Input" / / А. Родрігез, К. Кларк; Видавництво "FeedbackPress", Сідней, 2019. – 230 с.
37. "Motivating Students: The Psychology of Student Engagement and Academic Success" / / Г. Сміт, Р. Джонс; Видавництво "MotivateLearn", Нью-Йорк, 2022. – 290 с.
38. "Cultivating Curiosity: Strategies for Fostering Lifelong Learning in the Classroom" / / М. Гарсія, С. Лопес; Видавництво "CuriousEd", Барселона, 2016. – 240 с.
39. "Neuroscience and Education: Bridging the Gap for Optimal Learning" / / Д. Браун, А. Вейнберг; Видавництво "NeuroLearn", Сан-Франциско, 2018. – 300 с.
40. "The Art and Science of Teaching: Strategies for Effective Instruction" / / Л. Мартін, Р. Девіс; Видавництво "TeachArt", Париж, 2021. – 260 с.

ДОДАТОК А

КОД ПРОГРАМИ

Лістинг 1: main.py

```
import telebot
import settings
import registration
import keyboards
import time

#Початок роботи бота
bot = telebot.TeleBot(settings.data["bot"]["tg_token"])

@bot.message_handler(commands=["start"])
def get_start(message):
    #Початок реєстрації
    registration.choose_role(bot, message)

@bot.message_handler(content_types=["text"])
def get_text(message):
    #Вибір ролі
    if message.text == "👨‍💻 Студенти":
        registration.student_register(bot, message)
    elif message.text == "👨‍🏫 Викладачі":
        registration.teacher_register(bot, message)
    elif message.text == "👨‍👦 Батьки":
        registration.parent_register(bot, message)

    #Вибір групи для студентів
    if message.text.startswith("🎓"):
        new_group = message.text[1:].strip()
        settings.col_students.update_one({"tg_id": message.chat.id}, {'$set':{"group_name":
new_group}})
        bot.send_message(message.chat.id, "Дякую за реєстрацію. Вітаємо у системі!",
reply_markup=keyboards.student_menu_keyboard)
```

```
    #Вибір студента для батьків

    if message.text.startswith("👨"):

        new_student_email = message.text[1:].strip().split(" ")[2].replace("(", "").replace(")",
"")

        new_student = settings.col_students.find_one({"email": new_student_email})

        settings.col_parents.update_one({"tg_id": message.chat.id}, {'$set':{"student_tg_id":
new_student["tg_id"]}})

        settings.col_students.update_one({"tg_id": new_student["tg_id"]}, {'$set':{"is_control":
True}})

        bot.send_message(message.chat.id, "Дякую за реєстрацію. Вітаємо у системі!",
reply_markup=keyboards.parent_menu_keyboard)


    #Вибір профілю

    if message.text == "👨💻 Мій профіль":

        student = settings.col_students.find_one({"tg_id": message.chat.id})

        str0 = f"*Студентський профіль*\n\n"

        str1 = f"Ім'я: {student['firstname']}\n"

        str2 = f"Прізвище: {student['lastname']}\n"

        str3 = f"Пошта: {student['email']}\n"

        str4 = f"Номер телефону: {student['phone']}\n"

        str5 = f"Група: {student['group_name']}\n"

        str6 = f"Батьківський контроль: відключений\n"

        str7 = f"Зіграні Quiz: {student['games_total']}\n"

        str8 = f"Переможні Quiz: {student['wins_total']}\n"

        if student['is_control'] == True:

            str6 = f"Батьківський контроль: підключений\n"

        student_info = str0 + str1 + str2 + str3 + str4 + str5 + str6 + str7 + str8

        bot.send_message(message.chat.id, student_info, parse_mode="Markdown",
reply_markup=keyboards.student_menu_keyboard)


    elif message.text == "👨‍🏫 Мій профіль":

        teacher = settings.col_teachers.find_one({"tg_id": message.chat.id})

        str0 = f"*Викладацький профіль*\n\n"

        str1 = f"Ім'я: {teacher['firstname']}\n"

        str2 = f"Прізвище: {teacher['lastname']}\n"

        str3 = f"Пошта: {teacher['email']}\n"

        str4 = f"Номер телефону: {teacher['phone']}\n"

        str5 = f"Активні групи: відсутні\n"
```

```
        teacher_groups = list(settings.col_groups.find({"teacher_tg_id": message.chat.id}))
        teacher_group_names = [group["name"] for group in teacher_groups]
        if len(teacher_groups) >= 1:
            str5 = f"Активні групи: {', '.join(teacher_group_names)}\n"
        teacher_info = str0 + str1 + str2 + str3 + str4 + str5
        bot.send_message(message.chat.id, teacher_info, parse_mode="Markdown",
reply_markup=keyboards.teacher_menu_keyboard)


    elif message.text == "👨‍👦 Мій профіль":
        parent = settings.col_parents.find_one({"tg_id": message.chat.id})
        str0 = f"*Батьківський профіль*\n\n"
        str1 = f"Ім'я: {parent['firstname']}\n"
        str2 = f"Прізвище: {parent['lastname']}\n"
        str3 = f"Пошта: {parent['email']}\n"
        str4 = f"Номер телефону: {parent['phone']}\n"
        parent_info = str0 + str1 + str2 + str3 + str4
        bot.send_message(message.chat.id, parent_info, parse_mode="Markdown",
reply_markup=keyboards.parent_menu_keyboard)


    elif message.text == "👨‍💻 Профіль моєї дитини":
        parent = settings.col_parents.find_one({"tg_id": message.chat.id})
        child = settings.col_students.find_one({"tg_id": parent["student_tg_id"]})
        str0 = f"*Студентський профіль*\n\n"
        str1 = f"Ім'я: {child['firstname']}\n"
        str2 = f"Прізвище: {child['lastname']}\n"
        str3 = f"Пошта: {child['email']}\n"
        str4 = f"Номер телефону: {child['phone']}\n"
        str5 = f"Група: {child['group_name']}\n"
        str6 = f"Батьківський контроль: відключений\n"
        str7 = f"Зіграні Quiz: {child['games_total']}\n"
        str8 = f"Переможні Quiz: {child['wins_total']}\n"
        if child['is_control'] == True:
            str6 = f"Батьківський контроль: підключений\n"
        student_info = str0 + str1 + str2 + str3 + str4 + str5 + str6 + str7 + str8
        bot.send_message(message.chat.id, student_info, parse_mode="Markdown",
reply_markup=keyboards.parent_menu_keyboard)


    #Вибір доступних груп викладача
```

```
    if message.text == "👨‍👨‍👦 Мої групи":
        teacher = settings.col_teachers.find_one({"tg_id": message.chat.id})
        groups = list(settings.col_groups.find({"teacher_tg_id": message.chat.id}))
        group_info = f"*Активні групи*\n"
        if len(groups) >= 1:
            for group in groups:
                str0 = f"\nНазва: {group['name']}\n"
                str1 = f"Мова програмування: {group['language']}\n"
                str2 = f"Номер курсу: {group['course']}\n"
                str3 = f"День занять: {group['day']}\n"
                str4 = f"Час занять: {group['time']}\n"
                group_info += str0 + str1 + str2 + str3 + str4
        else:
            group_info += "\nНаразі активні групи відсутні."
        bot.send_message(message.chat.id, group_info, parse_mode="Markdown",
reply_markup=keyboards.teacher_menu_keyboard)


    #Вибір існуючих вікторин
    if message.text == "🧩 Розпочати Quiz":
        quizes = list(settings.col_quizes.find())
        if len(quizes) >= 1:
            quiz_keyboard = keyboards.choose_quiz(quizes)
            bot.send_message(message.chat.id, "Будь ласка, оберіть Quiz для проведення:",
reply_markup=quiz_keyboard)
        else:
            bot.send_message(message.chat.id, "Наразі Quiz відсутні.",
reply_markup=keyboards.teacher_menu_keyboard)


    #Вибір останніх результатів
    if message.text == "⏳ Переглянути останні результати":
        stats = list(settings.col_stats.find({"teacher_tg_id": message.chat.id}))
        if len(stats) >= 1:
            stats_info = "*⏳ Останні результати:*\n\n"
            places = ["🥇", "🥈", "🥉"]
            for res in stats:
                stats_info += f"🏵 {res['quiz_name']} - {res['group_name']}\n"
                quiz_players = res['quiz_players']
```

```
                top_players = [player for player in quiz_players if player["finish_game"] ==
True]

                top_players.sort(key=lambda x: x.get('total_points', 0), reverse=True)

                for place in range(3):

                    if place < len(top_players):

                        stats_info += f"{places[place]} {top_players[place]['student_firstname']}
{top_players[place]['student_lastname']} ({top_players[place]['total_points']})\n"

                stats_info += "\n--------------------\n\n"

            bot.send_message(message.chat.id, stats_info, parse_mode="Markdown",
reply_markup=keyboards.teacher_menu_keyboard)

        else:

            bot.send_message(message.chat.id, "Наразі результати відсутні.",
reply_markup=keyboards.teacher_menu_keyboard)


    #Повернення до меню для викладача

    if message.text == "🔙 Повернутися назад":

        delete_games = settings.col_games.delete_many({"teacher_tg_id": message.chat.id})

        bot.send_message(message.chat.id, "Повертаємося до головного меню.",
reply_markup=keyboards.teacher_menu_keyboard)


    #Вибір вікторини для викладача

    if message.text.startswith("🏆"):

        quiz = settings.col_quizes.find_one({'quiz_name': message.text[1:].strip()})

        teacher = settings.col_teachers.find_one({"tg_id": message.chat.id})

        groups = list(settings.col_groups.find({"teacher_tg_id": teacher["tg_id"]}))

        if len(groups) >= 1:

            active_game = {

                "teacher_tg_id": teacher["tg_id"],

                "teacher_firstname": teacher["firstname"],

                "teacher_lastname": teacher["lastname"],

                "group_name": '',

                "group_size": 0,

                "quiz_name": quiz["quiz_name"],

                "quiz_open_time": int(time.time()),

                "quiz_started": False,

                "quiz_players": []

            }

            settings.col_games.insert_one(active_game)

            group_keyboard = keyboards.choose_group_for_quiz(groups)
```

```
            bot.send_message(message.chat.id, f"Будь ласка, оберіть групу для проведення Quiz - *{quiz['quiz_name']}*", parse_mode="Markdown", reply_markup=group_keyboard)

        else:

            bot.send_message(message.chat.id, "Наразі активні групи відсутні.", reply_markup=keyboards.teacher_menu_keyboard)


    #Вибір групи для гри у Quiz

    if message.text.startswith("🎮"):

        group_name = message.text[1:].strip()

        students = list(settings.col_students.find({"group_name": group_name}))

        quiz = settings.col_games.find_one({"teacher_tg_id": message.chat.id})

        students_amount = len(students)

        if students_amount >= 1:

            bot.send_message(message.chat.id, f"Ви обрали групу *{group_name}* для проведення Quiz - *{quiz['quiz_name']}*!\n\nРозпочинайте гру, коли усі студенти приєднаються. (0/{students_amount})", parse_mode="Markdown", reply_markup=keyboards.start_game_keyboard)

            settings.col_games.update_one({"teacher_tg_id": message.chat.id}, {'$set':{"group_name": group_name, "group_size": students_amount}})

            for student in students:

                bot.send_message(student["tg_id"], f"Вас запрошено до Quiz - *{quiz['quiz_name']}*!\n\nНатисніть кнопку '▶ Приєднатися' для участі у Quiz вікторині.", parse_mode="Markdown", reply_markup=keyboards.join_game_keyboard)

        else:

            delete_games = settings.col_games.delete_many({"teacher_tg_id": message.chat.id})

            bot.send_message(message.chat.id, "Наразі студенти у групі відсутні.", reply_markup=keyboards.teacher_menu_keyboard)


    #Приєднання до гри у Quiz

    if message.text == "▶ Приєднатися":

        student = settings.col_students.find_one({"tg_id": message.chat.id})

        game = settings.col_games.find_one({"group_name": student["group_name"]})

        if game["quiz_started"] == False:

            quiz = settings.col_quizes.find_one({"quiz_name": game["quiz_name"]})

            parent = settings.col_parents.find_one({"student_tg_id": message.chat.id})

            time_now = int(time.time())

            predefined_results = {}

            for question in quiz["content"]:

                question_number = quiz["content"][question]

                result = {

                    question: {
```

```
                        "question": question_number["question"],
                        "options": question_number["options"],
                        "correct_answer": next(option['ans'] for option in
question_number["options"].values() if option['is_correct']),
                        "given_answer": "N/A",
                        "score": 0,
                        "question_time": time_now,
                        "answer_time": time_now
                    }
                }
                predefined_results.update(result)
            new_player = {
                "student_tg_id": student["tg_id"],
                "student_firstname": student["firstname"],
                "student_lastname": student["lastname"],
                "connection_time": time_now,
                "finish_time": time_now,
                "total_score": 0,
                "total_time": 0,
                "total_points": 0,
                "finish_game": False,
                "ready_to_play": True,
                "results": predefined_results
            }
            settings.col_games.update_one({"group_name": student["group_name"]},
{'$push':{"quiz_players": new_player}})
            game_updated = settings.col_games.find_one({"group_name": student["group_name"]})
            bot.send_message(message.chat.id, f"🟢 Ви успішно приєдналися до Quiz вікторини -
*{game['quiz_name']}*.", parse_mode="Markdown", reply_markup=telebot.types.ReplyKeyboardRemove())
            bot.send_message(game["teacher_tg_id"], f"🟢 Студент *{student['firstname']}
{student['lastname']} ({student['email']})* успішно приєднався до Quiz вікторини -
*{game['quiz_name']}*. ({len([player for player in game_updated['quiz_players'] if
player['ready_to_play'] == True])}/{game_updated['group_size'] - len([player for player in
game_updated['quiz_players'] if player['ready_to_play'] == False])})", parse_mode="Markdown")
            if parent != None:
                bot.send_message(parent["tg_id"], f"🟢 Ваша дитина успішно приєдналася до Quiz
вікторини - *{game['quiz_name']}*.", parse_mode="Markdown",
reply_markup=keyboards.parent_menu_keyboard)
        else:
```

```
            bot.send_message(message.chat.id, f"🟡 Вибачте. Quiz вікторина -
*{game['quiz_name']}* вже розпочалася.", parse_mode="Markdown",
reply_markup=keyboards.student_menu_keyboard)


    #Відмова від гри у Quiz
    if message.text == "❌ Відмовитися":
        student = settings.col_students.find_one({"tg_id": message.chat.id})
        game = settings.col_games.find_one({"group_name": student["group_name"]})
        if game["quiz_started"] == False:
            quiz = settings.col_quizes.find_one({"quiz_name": game["quiz_name"]})
            parent = settings.col_parents.find_one({"student_tg_id": message.chat.id})
            time_now = int(time.time())
            predefined_results = {}
            for question in quiz["content"]:
                question_number = quiz["content"][question]
                result = {
                    question: {
                        "question": question_number["question"],
                        "options": question_number["options"],
                        "correct_answer": next(option['ans'] for option in
question_number["options"].values() if option['is_correct']),
                        "given_answer": "N/A",
                        "score": 0,
                        "question_time": time_now,
                        "answer_time": time_now
                    }
                }
                predefined_results.update(result)
            new_player = {
                "student_tg_id": student["tg_id"],
                "student_firstname": student["firstname"],
                "student_lastname": student["lastname"],
                "connection_time": time_now,
                "finish_time": time_now,
                "total_score": 0,
                "total_time": 0,
                "total_points": 0,
                "finish_game": False,
```

```
                "ready_to_play": False,

                "results": predefined_results

            }

            settings.col_games.update_one({"group_name": student["group_name"]},
{'$push':{"quiz_players": new_player}})

            game_updated = settings.col_games.find_one({"group_name": student["group_name"]})

            bot.send_message(message.chat.id, f"● Ви відмовилися брати участь у Quiz вікторині -
*{game['quiz_name']}*.", parse_mode="Markdown", reply_markup=keyboards.student_menu_keyboard)

            bot.send_message(game["teacher_tg_id"], f"● Студент *{student['firstname']}
{student['lastname']} ({student['email']})* відмовився брати участь у Quiz вікторині -
*{game['quiz_name']}*. ({len([player for player in game_updated['quiz_players'] if
player['ready_to_play'] == True])}/{game_updated['group_size'] - len([player for player in
game_updated['quiz_players'] if player['ready_to_play'] == False])})", parse_mode="Markdown")

            if parent != None:

                bot.send_message(parent["tg_id"], f"● Увага! Ваша дитина відмовилася брати
участь у Quiz вікторині - *{game['quiz_name']}*.", parse_mode="Markdown",
reply_markup=keyboards.parent_menu_keyboard)

        else:

            bot.send_message(message.chat.id, f"🟡 Вибачте. Quiz вікторина -
*{game['quiz_name']}* вже розпочалася.", parse_mode="Markdown",
reply_markup=keyboards.student_menu_keyboard)



    #Початок гри

    if message.text == "📣 Розпочати гру":

        settings.col_games.update_one({"teacher_tg_id": message.chat.id},
{'$set':{"quiz_started": True}})

        game = settings.col_games.find_one({"teacher_tg_id": message.chat.id})

        quiz = settings.col_quizes.find_one({"quiz_name" : game['quiz_name']})

        players = [player for player in game['quiz_players'] if player['ready_to_play'] == True]

        answers_keyboard = keyboards.choose_answers(quiz['content']['1']['options'])

        bot.send_message(message.chat.id, "🏁 Quiz вікторину розпочато!",
reply_markup=telebot.types.ReplyKeyboardRemove())

        for player in players:

            bot.send_message(player["student_tg_id"], "🏁 Quiz вікторину розпочато!")

            bot.send_message(player["student_tg_id"],
f"*❓ 1. {quiz['content']['1']['question']}*", parse_mode="Markdown",
reply_markup=answers_keyboard)



@bot.callback_query_handler(func = lambda call: True)

def get_callback(call):

    #Пошук відповіді на питиння

    if "answer" in call.data:
```

```
        answer_time = int(time.time())

        user_tg_id = call.from_user.id

        question_text = call.message.text

        question = question_text.split("  ")[-1]

        question_number = question_text.split("  ")[1].replace(".", "")

        answer_letter = call.data.split("_")[-1]

        is_correct = call.data.split("_")[0]

        student = settings.col_students.find_one({"tg_id": user_tg_id})

        game = settings.col_games.find_one({"group_name": student["group_name"]})

        quiz = settings.col_quizes.find_one({"quiz_name": game["quiz_name"]})

        full_quiz_question = next(q_data for q_data in quiz['content'].values() if 'question' in
q_data and q_data['question'] == question)

        correct_answer = next(value for key, value in full_quiz_question['options'].items() if
value['is_correct'])

        answer_score = 0

        if is_correct == "correct":

            answer_score = 1

            new_text = f"*{question_text}*\n\n✅ Ваша відповідь *правильна*:
*{full_quiz_question['options'][answer_letter]['ans']}*."

        else:

            new_text = f"*{question_text}*\n\n❌ Ваша відповідь *неправильна*:
{full_quiz_question['options'][answer_letter]['ans']}. Правильна відповідь:
*{correct_answer['ans']}*."

        bot.edit_message_text(chat_id=user_tg_id, text=new_text, parse_mode="Markdown",
message_id=call.message.id)

        settings.col_games.update_one({"quiz_name": game["quiz_name"],
"quiz_players.student_tg_id": user_tg_id},
{"$set":{f"quiz_players.$.results.{question_number}.score": answer_score,
f"quiz_players.$.results.{question_number}.given_answer":
full_quiz_question['options'][answer_letter]['ans'],
f"quiz_players.$.results.{question_number}.question_time": call.message.date,
f"quiz_players.$.results.{question_number}.answer_time": answer_time}})

        game_updated = settings.col_games.find_one({"group_name": student["group_name"]})

        next_question_number = str(int(question_number) + 1)

        if next_question_number in quiz["content"]:

            answers_keyboard =
keyboards.choose_answers(quiz['content'][next_question_number]['options'])

            bot.send_message(user_tg_id,
f"*❓ {next_question_number}. {quiz['content'][next_question_number]['question']}*",
parse_mode="Markdown", reply_markup=answers_keyboard)

        else:

            student_final_results = next((player for player in game_updated["quiz_players"] if
player["student_tg_id"] == user_tg_id), None).get("results", {})
```

```
            student_total_score = 0

            student_total_time_spent = 0

            for question_id, quest in student_final_results.items():

                student_total_score += quest["score"]

                question_start_time = quest["question_time"]

                question_end_time = quest["answer_time"]

                question_time_spent = question_end_time - question_start_time

                student_total_time_spent += question_time_spent

            student_total_points = round((student_total_score * 10) - (student_total_time_spent /
10), 2)

            settings.col_games.update_one({"quiz_name": game_updated["quiz_name"],
"quiz_players.student_tg_id": user_tg_id}, {"$set":{"quiz_players.$.finish_time": answer_time,
"quiz_players.$.total_score": student_total_score, "quiz_players.$.total_time":
student_total_time_spent, "quiz_players.$.total_points": student_total_points,
"quiz_players.$.finish_game": True}})

            game_last_updated = settings.col_games.find_one({"group_name":
student["group_name"]})

            total_players_amount = len([player for player in game_last_updated['quiz_players'] if
player['ready_to_play'] == True])

            total_finished_amount = len([player for player in game_last_updated['quiz_players']
if player['finish_game'] == True])

            bot.send_message(user_tg_id, f"*🚀 Ви впоралися з вікториною! Ваші
результати:*\n\nНабрано балів: *{student_total_score} б*\nВитрачено часу:
*{student_total_time_spent} с*\nПідсумкова кількість очок: *{student_total_points} о*\n\n🎉
Дякуємо за участь!", parse_mode="Markdown", reply_markup=keyboards.student_menu_keyboard)

            bot.send_message(game["teacher_tg_id"], f"*🚀 Студент {student['firstname']}
{student['lastname']} ({student['email']}) впорався з вікториною! Отримані
результати:*\n\nНабрано балів: *{student_total_score} б*\nВитрачено часу:
*{student_total_time_spent} с*\nПідсумкова кількість очок: *{student_total_points} о*\n\nГравців
впоралося: {total_finished_amount}/{total_players_amount}", parse_mode="Markdown")

            parent = settings.col_parents.find_one({"student_tg_id": user_tg_id})

            if parent != None:

                bot.send_message(parent["tg_id"], f"*🚀 Ваша дитина впоралася з вікториною!
Отримані результати:*\n\nНабрано балів: *{student_total_score} б*\nВитрачено часу:
*{student_total_time_spent} с*\nПідсумкова кількість очок: *{student_total_points} о*",
parse_mode="Markdown")

            if total_finished_amount >= total_players_amount:

                places = ["🥇", "🥈", "🥉"]

                quiz_players = game_last_updated['quiz_players']

                top_players = [player for player in quiz_players if player["finish_game"] ==
True]

                top_players.sort(key=lambda x: x.get('total_points', 0), reverse=True)

                finish_info = "*🏁 Quiz вікторину завершено!*\n\n"

                for place in range(3):
```

```
                    if place < len(top_players):

                        finish_info += f"{places[place]}
{top_players[place]['student_firstname']} {top_players[place]['student_lastname']}
({top_players[place]['total_points']})\n"

                finish_info += "\n*Так тримати!*"

                is_top_player = True

                for player in top_players:

                    if is_top_player:

                        settings.col_students.update_one({"tg_id": player["student_tg_id"]},
{'$inc': {'wins_total': 1}})

                        is_top_player = False

                    settings.col_students.update_one({"tg_id": player["student_tg_id"]}, {'$inc':
{'games_total': 1}})

                    bot.send_message(player["student_tg_id"], finish_info, parse_mode="Markdown")

                bot.send_message(game["teacher_tg_id"], finish_info, parse_mode="Markdown",
reply_markup=keyboards.teacher_menu_keyboard)

                settings.col_stats.insert_one(game_last_updated)

                delete_games = settings.col_games.delete_many({"teacher_tg_id":
game_last_updated['teacher_tg_id']})


while True:

    try:

        bot.infinity_polling()

    except:

        pass
```

Лістинг 2: settings.py

```
import pymongo


#Основні налаштування бота та бази

data = {

    "bot" : {

        "tg_token" : "6784245218:AAH9-5sMp5PQaY9TvyWSpLqqH019vcJVw3E",

        "bot_name" : "QuizGo"

    },

    "db" : {

        "mongo_token" : "mongodb+srv://NTU_DP_quizgo:politeh2023@cluster0.n77qd.mongodb.net",

        "mongo_db" : "QuizGo_db",

        "mongo_col_teachers" : "Teachers",
```

```
        "mongo_col_students" : "Students",
        "mongo_col_parents" : "Parents",
        "mongo_col_groups": "Groups",
        "mongo_col_active_games": "ActiveGames",
        "mongo_col_quizes" : "Quizes",
        "mongo_col_stats" : "Stats"
    }
}

#Підключення до БД та колекцій
client = pymongo.MongoClient(data["db"]["mongo_token"])
db = client[data["db"]["mongo_db"]]

col_students = db[data["db"]["mongo_col_students"]]
col_teachers = db[data["db"]["mongo_col_teachers"]]
col_parents = db[data["db"]["mongo_col_parents"]]

col_groups = db[data["db"]["mongo_col_groups"]]

col_games = db[data["db"]["mongo_col_active_games"]]
col_quizes = db[data["db"]["mongo_col_quizes"]]
col_stats = db[data["db"]["mongo_col_stats"]]
```

Лістинг 3: registration.py

```
import settings
import keyboards
import telebot

#Функція для вибору ролі (студенти/викладачі/батьки)
def choose_role(bot, message):
    #Привітання для будь-якого користувача
    bot.send_message(message.chat.id, f"Вітаю, {message.from_user.first_name}! Будь ласка,
оберіть свою роль:", reply_markup=keyboards.role_keyboard)

#Функція для реєстрації студентів
def student_register(bot, message):
    #Блок реєстрації-------------------------------------------------------------------------
-
```

```
    def get_user_phone(user_phone):
        user_data["phone"] = user_phone.text
        settings.col_students.insert_one(user_data)
        group_keyboard = keyboards.propose_groups()
        bot.send_message(message.chat.id, f"Останній крок. Оберіть вашу групу:",
reply_markup=group_keyboard)


    def get_user_email(user_email):
        user_data["email"] = user_email.text
        user_phone = bot.send_message(message.chat.id, "Введіть номер телефону:")
        bot.register_next_step_handler(user_phone, get_user_phone)


    def get_user_lastname(user_lastname):
        user_data["lastname"] = user_lastname.text
        user_email = bot.send_message(message.chat.id, "Введіть поштову скриньку:")
        bot.register_next_step_handler(user_email, get_user_email)


    def get_user_firstname(user_name):
        user_data["firstname"] = user_name.text
        user_lastname = bot.send_message(message.chat.id, "Введіть прізвище:")
        bot.register_next_step_handler(user_lastname, get_user_lastname)
    #Блок реєстрації-------------------------------------------------------------------------
-


    #Перевірка наявності користувача у базі
    user_exists = settings.col_students.find_one(
        {"tg_id": message.chat.id}
    )


    #Якщо користувача не існує, проводимо реєстрацію
    if user_exists == None:
        bot.send_message(message.chat.id, "Будь ласка, зареєструйтеся:")


        user_data =  {
            "tg_id": message.chat.id,
            "firstname" : "",
            "lastname" : "",
            "email" : "",
```

```
            "phone" : "",
            "is_control": False,
            "group_name" : "",
            "games_total": 0,
            "wins_total": 0
        }

        user_name = bot.send_message(message.chat.id, "Введіть ім'я:",
reply_markup=telebot.types.ReplyKeyboardRemove())
        bot.register_next_step_handler(user_name, get_user_firstname)
    else:
        #Привітання для користувача, що увійшов в акаунт
        bot.send_message(message.chat.id, f"{user_exists['firstname']}, з поверненням!",
reply_markup=keyboards.student_menu_keyboard)

#Функція для реєстрації викладачів
def teacher_register(bot, message):
    #Блок реєстрації-------------------------------------------------------------------------
-
    def get_user_phone(user_phone):
        user_data["phone"] = user_phone.text
        settings.col_teachers.insert_one(user_data)
        bot.send_message(message.chat.id, "Дякую за реєстрацію. Вітаємо у системі!",
reply_markup=keyboards.teacher_menu_keyboard)

    def get_user_email(user_email):
        user_data["email"] = user_email.text
        user_phone = bot.send_message(message.chat.id, "Введіть номер телефону:")
        bot.register_next_step_handler(user_phone, get_user_phone)

    def get_user_lastname(user_lastname):
        user_data["lastname"] = user_lastname.text
        user_email = bot.send_message(message.chat.id, "Введіть поштову скриньку:")
        bot.register_next_step_handler(user_email, get_user_email)

    def get_user_firstname(user_name):
        user_data["firstname"] = user_name.text
        user_lastname = bot.send_message(message.chat.id, "Введіть прізвище:")
        bot.register_next_step_handler(user_lastname, get_user_lastname)
```

```
    #Блок реєстрації-------------------------------------------------------------------------------
-


    #Перевірка наявності користувача у базі
    user_exists = settings.col_teachers.find_one(
        {"tg_id": message.chat.id}
    )


    #Якщо користувача не існує, проводимо реєстрацію
    if user_exists == None:
        bot.send_message(message.chat.id, "Будь ласка, зареєструйтеся:")


        user_data =  {
            "tg_id": message.chat.id,
            "firstname" : "",
            "lastname" : "",
            "email" : "",
            "phone" : ""
        }


        user_name = bot.send_message(message.chat.id, "Введіть ім'я:",
reply_markup=telebot.types.ReplyKeyboardRemove())
        bot.register_next_step_handler(user_name, get_user_firstname)
    else:
        #Привітання для користувача, що увійшов в акаунт
        bot.send_message(message.chat.id, f"{user_exists['firstname']}, з поверненням!",
reply_markup=keyboards.teacher_menu_keyboard)


#Функція для реєстрації батьків
def parent_register(bot, message):
    #Блок реєстрації-------------------------------------------------------------------------------
-
    def get_user_phone(user_phone):
        user_data["phone"] = user_phone.text
        settings.col_parents.insert_one(user_data)
        student_keyboard = keyboards.propose_students()
        bot.send_message(message.chat.id, f"Останній крок. Оберіть вашу дитину:",
reply_markup=student_keyboard)
```

```
    def get_user_email(user_email):
        user_data["email"] = user_email.text
        user_phone = bot.send_message(message.chat.id, "Введіть номер телефону:")
        bot.register_next_step_handler(user_phone, get_user_phone)

    def get_user_lastname(user_lastname):
        user_data["lastname"] = user_lastname.text
        user_email = bot.send_message(message.chat.id, "Введіть поштову скриньку:")
        bot.register_next_step_handler(user_email, get_user_email)

    def get_user_firstname(user_name):
        user_data["firstname"] = user_name.text
        user_lastname = bot.send_message(message.chat.id, "Введіть прізвище:")
        bot.register_next_step_handler(user_lastname, get_user_lastname)
    #Блок реєстрації-------------------------------------------------------------------------
-

    #Перевірка наявності користувача у базі
    user_exists = settings.col_parents.find_one(
        {"tg_id": message.chat.id}
    )

    #Якщо користувача не існує, проводимо реєстрацію
    if user_exists == None:
        bot.send_message(message.chat.id, "Будь ласка, зареєструйтеся:")

        user_data =  {
            "tg_id": message.chat.id,
            "firstname" : "",
            "lastname" : "",
            "email" : "",
            "phone" : "",
            "student_tg_id": 0
        }

        user_name = bot.send_message(message.chat.id, "Введіть ім'я:",
reply_markup=telebot.types.ReplyKeyboardRemove())
        bot.register_next_step_handler(user_name, get_user_firstname)
```

```
    else:
        #Привітання для користувача, що увійшов в акаунт
        bot.send_message(message.chat.id, f"{user_exists['firstname']}, з поверненням!",
reply_markup=keyboards.parent_menu_keyboard)
```

Лістинг 4: keyboards.py

```
import telebot
import settings

#Кнопка для повернення до меню
back_button = telebot.types.KeyboardButton("🔙 Повернутися назад")

#Клавіатура для вибору ролі-------------------------------------------------------------
role_keyboard = telebot.types.ReplyKeyboardMarkup(resize_keyboard=True, row_width=3)

role_button1 = telebot.types.KeyboardButton("👨‍💻 Студенти")
role_button2 = telebot.types.KeyboardButton("👨‍🏫 Викладачі")
role_button3 = telebot.types.KeyboardButton("👨‍👦 Батьки")

role_keyboard.add(role_button1, role_button2, role_button3)
#-----------------------------------------------------------------------------------

#Клавіатура для меню Студенти-----------------------------------------------------------
student_menu_keyboard = telebot.types.ReplyKeyboardMarkup(resize_keyboard=True, row_width=2)

student_menu_button1 = telebot.types.KeyboardButton("👨‍💻 Мій профіль")

student_menu_keyboard.add(student_menu_button1)

#Клавіатура для меню Викладачі----------------------------------------------------------
teacher_menu_keyboard = telebot.types.ReplyKeyboardMarkup(resize_keyboard=True, row_width=2)

teacher_menu_button1 = telebot.types.KeyboardButton("👨‍🏫 Мій профіль")
teacher_menu_button2 = telebot.types.KeyboardButton("👨‍👨‍👦 Мої групи")
```

```
teacher_menu_button3 = telebot.types.KeyboardButton("🧩 Розпочати Quiz")

teacher_menu_button4 = telebot.types.KeyboardButton("⏳ Переглянути останні результати")


teacher_menu_keyboard.add(teacher_menu_button1, teacher_menu_button2, teacher_menu_button3,
teacher_menu_button4)


#Клавіатура для меню Батьки-----------------------------------------------------------

parent_menu_keyboard = telebot.types.ReplyKeyboardMarkup(resize_keyboard=True, row_width=2)


parent_menu_button1 = telebot.types.KeyboardButton("👨‍👦 Мій профіль")

parent_menu_button2 = telebot.types.KeyboardButton("👨‍💻 Профіль моєї дитини")


parent_menu_keyboard.add(parent_menu_button1, parent_menu_button2)

#-------------------------------------------------------------------------------


#Клавіатура з вибором групи навчання (формується автоматично)

def propose_groups():

    group_keyboard = telebot.types.ReplyKeyboardMarkup(resize_keyboard=True, row_width=1)

    all_groups = list(settings.col_groups.find())

    if len(all_groups) >= 1:

        for group in all_groups:

            group_button = telebot.types.KeyboardButton(f"🎓 {group['name']}")

            group_keyboard.add(group_button)

    else:

        group_button = telebot.types.KeyboardButton("🎓 Індивідуальний формат навчання")

        group_keyboard.add(group_button)

    return group_keyboard


#Клавіатура з вибором студента (формується автоматично)

def propose_students():

    student_keyboard = telebot.types.ReplyKeyboardMarkup(resize_keyboard=True, row_width=1)

    all_students = list(settings.col_students.find({"is_control": False}))

    if len(all_students) >= 1:

        for student in all_students:

            student_button = telebot.types.KeyboardButton(f"👤 {student['firstname']}
{student['lastname']} ({student['email']})")

            student_keyboard.add(student_button)
```

```
    else:
        student_button = telebot.types.KeyboardButton("👨‍🎓 Майбутній студент")
        student_keyboard.add(student_button)
    return student_keyboard

#Клавіатура з вибором вікторини (формується автоматично)
def choose_quiz(quizes):
    quiz_choice_keyboard = telebot.types.ReplyKeyboardMarkup(resize_keyboard=True, row_width=1)
    for quiz in quizes:
        quiz_choice_button = telebot.types.KeyboardButton(f"🏆 {quiz['quiz_name']}")
        quiz_choice_keyboard.add(quiz_choice_button)
    quiz_choice_keyboard.add(back_button)
    return quiz_choice_keyboard

#Клавіатура з вибором групи для вікторини (формується автоматично)
def choose_group_for_quiz(groups):
    group_choice_keyboard = telebot.types.ReplyKeyboardMarkup(resize_keyboard=True, row_width=1)
    for group in groups:
        group_choice_button = telebot.types.KeyboardButton(f"🎮 {group['name']}")
        group_choice_keyboard.add(group_choice_button)
    group_choice_keyboard.add(back_button)
    return group_choice_keyboard

#Клавіатура для початку Quiz вікторини
start_game_keyboard = telebot.types.ReplyKeyboardMarkup(resize_keyboard=True, row_width=2)

start_game_button1 = telebot.types.KeyboardButton("📣 Розпочати гру")

start_game_keyboard.add(start_game_button1, back_button)

#Клавіатура для участі у Quiz вікторині
join_game_keyboard = telebot.types.ReplyKeyboardMarkup(resize_keyboard=True, row_width=2)

join_game_button1 = telebot.types.KeyboardButton("▶ Приєднатися")
join_game_button2 = telebot.types.KeyboardButton("❌ Відмовитися")

join_game_keyboard.add(join_game_button1, join_game_button2)
```

```
#Клавіатура з вибором правильної відповіді

def choose_answers(options):

    answers_markers = ["🟥", "🟦", "🟨", "🟩"]

    answers_letters = ["a", "b", "c", "d"]

    answers_keyboard = telebot.types.InlineKeyboardMarkup(row_width = 1)

    for number in range(len(options)):

        marker = answers_markers[number]

        text = options[answers_letters[number]]['ans']

        is_correct = options[answers_letters[number]]['is_correct']

        if is_correct == True:

            answers_button = telebot.types.InlineKeyboardButton(f"{marker} {text}", callback_data
= f"correct_answer_{answers_letters[number]}")

        else:

            answers_button = telebot.types.InlineKeyboardButton(f"{marker} {text}", callback_data
= f"incorrect_answer_{answers_letters[number]}")

        answers_keyboard.add(answers_button)

    return answers_keyboard
```

ДОДАТОК Б

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	Опис
Пояснювальні документи	
Кваліфікаційна робота_Лисицький.docx	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Кваліфікаційна робота_Лисицький.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF.
Програма	
Програма_Лисицький.rar	Архів. Містить коди програми і скомпільовану програму.
Презентація	
Презентація_Лисицький.pptx	Презентація кваліфікаційної роботи.