МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет

Факультет електроніки та інформаційних технологій

Кафедра комп'ютерних наук

«До захисту допущено»

В.о. завідувача кафедри

Ігор ШЕЛЕХОВ

(підпис)

грудня 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня магістр

зі спеціальності 122 - Комп'ютерних наук,

освітньо-професійної програми «Інформатика»

на тему: «Інформаційна технологія адаптивної підготовки абітурієнтів до вступних іспитів»

здобувача групи ІН.м-22 Пінчук Софії Михайлівни

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Софія ПІНЧУК

(підпис)

Керівник
ст. викл., к.ф.-м.н. Оксана ШОВКОПЛЯС

(підпис)

Консультант
ст. викл., к.ф.-м.н. Оксана ШОВКОПЛЯС

(підпис)

Суми – 2023

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня магістр

зі спеціальності 122 - Комп'ютерних наук, освітньо- професійної програми «Інформатика»
здобувача групи ІН.м-22 Пінчук Софії Михайлівни

1. Тема роботи: «Інформаційна технологія адаптивної підготовки абітурієнтів до вступних іспитів» затверджую наказом по СумДУ від *«06» грудня 2023 року № 1412-VI*
2. Термін здачі здобувачем кваліфікаційної роботи *до 18 грудня 2023 року*
3. Вхідні дані до кваліфікаційної роботи _____
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1) Аналіз проблеми предметної області, постановка й формування завдань дослідження. 2) Огляд технологій, що використовуються для розробки Telegram-ботів. 3) Розробка схеми механізму для адаптивності боту. 4) Вибір методології, мови програмування та засоби програмної реалізації 5) Розробка бота. 6) Аналіз результатів.*
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)_____
6. Консультанти до проєкту (роботи), із значенням розділів проєкту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання « » р.

Завдання прийняв до виконання _____ (підпис) Керівник _____ (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Аналіз проблеми предметної області, постановка й формування завдань дослідження*		
2	*Огляд технологій, що використовуються для розробки Telegram-ботів*		
3	*Розробка схеми механізму для адаптивності боту.*		
4	*Вибір методології, мови програмування та засоби програмної реалізації*		
5	*Розробка бота.*		
6	*Аналіз результатів.*		
7	*Оформлення пояснювальної записки до кваліфікаційної роботи.*		

Здобувач вищої освіти _____ (підпис) Керівник _____ (підпис)

АНОТАЦІЯ

Записка: 62 стор., 39 рис., 1 додаток, 35 джерел.

Обґрунтування актуальності теми роботи – тема кваліфікаційної роботи є актуальною, оскільки присвячена розв'язанню важливої практичної задачі розроблення моделі та інформаційної технології адаптивної підготовки абітурієнтів до іспитів.

Об'єкт дослідження – процес проєктування адаптивної системи для підготовки абітурієнтів.

Мета роботи – розроблення інформаційної технології, яка є адаптивною до кожного користувача та призначена для підготовки абітурієнтів до іспитів.

Методи дослідження – методи інформаційного аналізу та статистичний аналіз.

Результати – проаналізовано можливі методи розроблення інформаційної технології та інструменти для її створення. Розроблено інформаційну технологію для адаптивної підготовки абітурієнтів до вступних іспитів, яка має персоналізований підхід до кожного користувача, збираючи статистику його відповідей та пропонуючи відповідного рівня питання. Інформаційну технологію реалізовано як Telegram-бот з використанням мови програмування Python та фреймворку aiogram.

АДАПТИВНА ПІДГОТОВКА, ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ, ТЕСТУВАННЯ, AIOGRAM, ASYNCIO, PYTHON, TELEGRAM-BOT.

ЗМІСТ

ВСТУП 6
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 8
1.1 Дослідження актуальності проблеми 8
1.2 Визначення поняття адаптивної підготовки та використання Telegram-ботів в ній 9
1.3 Постановка задачі 11
2 АНАЛІТИЧНА ЧАСТИНА 13
2.1 Створення функціональних вимог 13
2.2 Огляд доступних мов програмування та фреймворків для створення інформаційної технології 14
2.3 Віртуальне середовище для розробки Telegram-бота 25
2.4 Огляд та вибір бази даних 25
3 ПРАКТИЧНА РЕАЛІЗАЦІЯ 29
3.1 Архітектурна модель 29
3.2 Програмна реалізація 30
3.3 Огляд створеної інформаційної технології 41
ВИСНОВОК 54
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 55
ДОДАТОК А 58

ВСТУП

Актуальність. В наш час, коли освіта перейшлла до дистанційного режиму, досить актуальним стало навчання онлайн. Вже було розроблено безліч різних додатків, курсів та ін. для такого типу навчання. Але важливо тут розглянути питання зручності та адаптивності створених технологій.

При вступі до університету, абітурієнти мають складати вступні іспити. В більшості випадків виникає проблема в підготовці, адже матеріалу багато, тестів багато, але осягнути все досить складно і не зрозуміло з чого починати. Для цього необхідний персоналізований підхід до кожного абітурієнта.

Дана кваліфікаційна робота магістра присвячена саме розробці такої адаптивної технології, яка буде індивідуально підходити до кожного користувача та цим допомогати у підготовці до вступу.

Об’єкт дослідження – процес адаптивного тестування.

Предмет дослідження – технологія адаптивного тестування, методи пропонування релевантних тестів на основі статистики відповідей.

Гіпотеза. Високу якість підготовки абітурієнтів до вступних іспитів можна досягти завдяки персоналізованому підходу, який враховує рівень знань індивідуально для кожного та адаптує технологію тестування використовуючи статистичні дані кожного користувача.

Наукова новизна. На відміну від існуючих аналогів інформаційних технологій, дана розробка, що несе в собі персоналізацію для кожного користувача та робить цим себе адаптивною, дозволяє реалізувати ефективний та простий в використанні механізм, а саме Telegram-бот, який буде адаптовуватися під кожного абітурієнта та персоналізовано надсилати йому тести. Саме таке поєднання Telegram-бота та адаптивності робить технологію ефективною та доступною кожному.

Апробація матеріалів роботи. Основні результати роботи оприлюднені та оговорені на VIII Всеукраїнській науково-практичній конференції “Перспективні напрямки сучасної електроніки, інформаційних і комп'ютерних

систем” (MEICS-2023) м.Дніпро (22-24 листопада 2023 р.).

Структура. Дана кваліфікаційна робота магістра складається зі вступу, інформаційно-аналітичного огляду предметної області щодо актуальності проблеми та окреслення поняття адаптивності; аналітичної частини щодо створення функціональних вимог, огляду інструментів для розробки та бази даних; практичної реалізації інформаційної технології; висновків; списку використаних джерел та додатку.

Зв’язок роботи з науковою темою. Кваліфікаційна робота виконана на кафедрі комп’ютерних наук та пов’язана з виконанням науково-дослідної роботи №0120U103407 «Застосування технологій games learning, blended learning, віртуальної та доповненої реальності в навчальному процесі» (2020-2025).

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Дослідження актуальності проблеми

У сучасному світі діджиталізація займає передову позицію. Все людство намагається полегшити та модернізувати своє повсякденне життя за допомогою сучасних технологій. Це стосується найрізноманітніших сфер життя: медицина, освіта, аграрни бізнес, економіка, державна сфера, торгівля, тощо. Дивлячись зі сторони студента, розглянемо детальніше прорив технологій в освіті, адже на даний момент це найближча тематика як для студента.

За часи пандемії [1] та повномасштабного вторгнення освітня сфера в Україні та світі зробила величезний крок у розвитку та впровадженні інформаційних технологій в свій процес. Звичайно і раніше ми мали справу з якимись інноваціями, наприклад, спочатку ще давно це були диски з підручниками, потім це переросло в повноцінні онлайн курси. Але це лише була крапля в морі. На сьогодні ми вийшли на новий рівень, значна частина освіти базується на впровадженні інноваційних технологій, що покращують якість знань студентів та підвищують конкурентоспроможність вищих навчальних закладів.

Як зазначалося вище, часи пандемії та повномасштабного вторгнення внесли свої корективи в освіту. На власному досвіді можу відмітити стрімкий ріст популярності використання Telegram. Досить зручно організовувати своє навчання, створюючи чати групи з різних дисциплін, інформаційні канали, тощо.

Повертаючись до часів вступу до бакалаврату/магістратури, згадується нескінченна кількість матеріалів для повторення та підготовки. Готуючись до ЗНО, були пройдені мабуть всі можливі тести за минулі роки. При підготовці до магістратури, було переглянуто неймовірну кількість матеріалів. Дивлячись на зручність використання Telegram, хотілося б якось полегшити підготовчий процес для абітурієнтів з використанням даного месенджера. Саме цьому і

присвячена дана робота, а саме створенню Telegram-бота, котрий буде допомагати в адаптивній підготовці абітурієнтів до вступних іспитів.

Telegram-бот може стати ефективним інструментом персоналізації навчання, пропонуючи індивідуальний підхід кожному користувачеві. Бот може бути використаний як для неформальної освіти, так і в офіційних навчальних закладах. Дана технологія представляє собою потужний інструмент для покращення якості навчання та забезпечення широкого доступу до освіти.

1.2 Визначення поняття адаптивної підготовки та використання Telegram-ботів в ній

То що ж собою являє ця адаптивна підготовка? Це концепція, яка визначає індивідуальну траєкторію навчання для кожного студента, враховуючи персоналізовані потреби, темп навчання та рівень знань. Такий підхід використовує сильні сторони кожного студента і спрямований на вирішення його проблемних сторін. Дана концепція базується на впровадженні інтелектуальних систем, інформаційних технологій та аналізі даних.

З важливих аспектів адаптивної підготовки є персоналізація, використання технологій, аналіз та звітність. Персоналізація передбачає коригування та/або адаптацію навчального процесу до індивідуальних потреб та особливостей студента. Адаптивна підготовка часто передбачає використання, наприклад, штучного інтелектру, машинного навчання, віртуальних інструментів, інтерактивних платформ, тощо. Вони дозволяють систематично аналізувати прогрес студентів [2], їхні успіхи та труднощі, що допомагає вчителям відстежувати засвоєння матеріалу та індивідуалізувати траєкторію навчання [3].

Перейдемо до визначення бота. Бот визначається як комп'ютерна програма, яка здатна імітувати інтелектуальний діалог з однією або кількома людьми і може розпізнавати як письмову, так і усну мову. Термін "чат-бот" був

отриманий для позначення двох основних атрибутів, а саме "чат" - для розмови і "бот" – для робота. Боти використовуються як навчальний інструмент для підтримки процесу викладання та навчання. Система чат-ботів є однією з найпопулярніших технологій штучного інтелекту, що використовується для підтримки викладання та навчання. Бот також розглядається як віртуальний асистент, здатний правильно відповісти на низку запитань, поставлених людиною [4].

Застосування ботів може бути надзвичайно різноманітним та корисним [5]. Аналізуючи все вище написане, з впевненістю можна сказати, що Telegram-боти можуть впоратися з значною частиною поставлених цілей адаптивної підготовки. Наприклад:

- Персоналізовані навчальні матеріали. Бот може аналізувати відповіді студента, виявляти сильні та слабкі місця. Потім на основі цього може надсилати індивідуалізовані навчальні матеріали, що допоможуть в покращенні знань
- Тестування та зворотній зв’язок. Бот може проводити тести з різних дисциплін на регулярній основі. Потім, спираючись на відповіді, може надсилати їх аналіз, статистику. Таким чином, студенти можуть бачити на які моменти у навчанні необхідно звернути увагу.
- Рекомендації. Бот може надсилати корисні поради та рекомендації щодо конкретних дисциплін, виокремлюючи важливі частини, на які необхідно звернути увагу. Також це можуть бути стратегії, наприклад, щодо відповідей.

Цифровий формат освоєння навчальних програм продовжує свою експансію на всіх рівнях освіти. І хоча масові відкриті онлайн-курси зробили навчання доступним для мільйонів людей по всьому світу, дослідження показують, що тільки 7% слухачів від зарахованих на курс фактично завершують їх, в більшості випадків через відсутність зворотного зв’язку та підтримки [6–8].

Розглянемо кілька варіантів використання адаптивного тестування в відомих застосунках:

- Duolingo - платформа для вивчення різних мов. Платформа використовує персоналізацію навчання. Має різні рівні складності, тоді складність завдань, які отримує користувач залежить від його успіхів. Також система пропонує повторення пройдення матеріалу в тих моментах, де користувачеві складно засвоїти інформацію. З недоліків, в контексті задачі яка стоїть перед нами - простота використання, доступність широкому колу людей, швидкість, немає потреби реєстрації, немає ніяких суттєвих, окрім проходження етапу реєстрації. В Telegram-боті реєстрація не буде проходити, а аутентифікація користувача буде працювати за його айді користувача.
- Khan Academy - освітня онлайн-платформа, що пропонує безкоштовні курси з різних предметів. Система адаптивна. Завдання адаптуються до рівня студента, також надаються додаткові матеріали для закріплення знань. Для використання також необхідна реєстрація.
- edX - платформа, де пропонуються курси від вищих навчальних закладів з усього світу. Не всі курси мають адаптивні методи, що дозволяє навчатися власним темпом. Студенти можуть створювати індивідуалізовані плани навчання.

1.3 Постановка задачі

В ході огляду використання технологій адаптивної підготовки було обрано Telegram-бот. Він може надавати зворотний зв’язок у вигляді аналізу відповідей користувача, виведення статистики його відповідей та пропонувати матеріал по темам, які абітурієнт хоче покращити.

Виходячи з інформаційного огляду, найбільш ефективною технологією адаптивної підготовки в розрізі закріплення та перевірки знань та навичок користувача є та що відповідає наступним критеріям:

- Надає пояснення до відповідей (посилання на матеріали).
- Користувач не має змоги перемикатися між питаннями.
- Список питань в тесті постійно змінюється, беручи їх з великої бази питань.
- При кожній спробі проходження питання, порядок відповідей змінюється, що не дає користувачеві змоги “зазубрити” відповіді.
- Пропонує користувачу релевантні питання з темами які є для нього проблемними.

Telegram-бот, як засіб для індивідуалізації навчального процесу, надає персоналізовані матеріали та завдання, що відповідають потребам абітурієнта. Це сприяє створенню інтерактивного та адаптивного навчального середовища, спрямованого на ефективну підготовку до вступних іспитів. Комбінація зручності, персоналізації й ефективності робить цей підхід дуже привабливим для абітурієнтів. Також Telegram-бот є доступним широкому колу користувачів. Тому це гарна ідея для навчальних закладів використовувати бот в якості підготовки своїх студентів.

2 АНАЛІТИЧНА ЧАСТИНА

2.1 Створення функціональних вимог

На початку необхідно визначитися хто буде цільовою аудиторією та які потреби користувачів потрібно закрити.

Виходячи з завдання бота, адаптивна підготовка абітурієнтів до іспитів, маємо два типи користувачів: абітурієнт та викладач (адміністратор).

Розглянемо їхні очікування (потреби, які необхідно задовольнити) від Telegram-бота.

Таблиця 2.1 Користувачі та їхні вимоги до бота

Користувач	**Вимоги до бота**
Абітурієнт	Користувач, який очікує від бота допомогу в адаптивній підготовці до іспитів. З вимог: можливість проходження тестування, надання рекомендованої літератури для покращення прогалин у знаннях, виведення статистики надання відповідей.
Викладач (адміністратор)	Користувач, який очікує від бота можливість організації дисципліни таким чином, аби абітурієнт мав можливість легше підготуватися до

	вступних іспитів. З вимог: можливість внесення в базу даних інформації про дисципліну, питання та відповіді до дисципліни, відслідковування прогресу відповідей абітурієнтів.

Отже, як бачимо, необхідно створити два інтерфейса: для абітурієнтів та викладачів (адміністраторів).

2.2 Огляд доступних мов програмування та фреймворків для створення інформаційної технології

Написання Telegram-бота може бути виконано на багатьох мовах програмування, навіть на сучасній українській мові програмування - мавка.укр. [9]. Розглянемо кілька популярних мов для розробки Telegram-ботів:

- Python. Останнім часом надзвичайно популярна мова, особливо серед аналітиків. Основне про дану мову:
  - використовується для веб-розробки, розробки програмного забезпечення, аналітиці, тощо;
  - працює на різних платформах (Linux, Windows, Mac, тощо);
  - розглядається як об’єктно-орієнтована, функціональна або процедурна мова;
  - інтуїтивно зрозумілий та простий синтаксис (схожість з англійською мовою)

- завдяки лаконічному синтаксису кількість рядків коду є меншою у порівнянні з деякими іншими мовами програмування, що робить код більш читабельним;
- працює на основі системи інтерпретаторів (код може бути виконаним одразу після написання), що пришвидшує створення прототипів [10].

Для розробки Telegram-ботів найчастіше використовують python-telegram-bot або aiogram.

- Javascript (Node.js). Досить популярна через простоту та кількість бібліотек мова програмування або мова сценаріїв [11]. Із основного про дану мову:
  - інтерпретована мова;
  - створення динамічності та інтерактивності на веб-сторінках;
  - відповідає за взаємодію з користувачем на веб-сторінках;
  - для написання сценаріїв веб-сторінок;
  - програмування на боці сервера за допомогою Node.js;
  - створення веб-застосунків;
  - об’єктно-орієнтований підхід до програмування.

Для розробки Telegram-ботів використовують Telegram Bot API або telegraf.

- Java. Одна з найстаріших мов програмування, на якій працюють мільярди застосунків та пристроїв. З основного про дану мову:
  - використовується для мобільних, десктоп та веб-додатків;
  - використовується для ігор, підключень до баз даних, тощо;

- працює на різних платформах (Linux, Windows, Mac, тощо);
- велике ком'юніті підтримки (велика кількість розробників);
- швидка та потужна [12].

Найпопулярнішими для розробки Telegram-ботів є Telegram Bot Java Library та Telegram Bot API [13].

Аналізуючи мови програмування, для розробки Telegram-бота було обрано мову Python. Вибір впав саме на цю мову, спираючись на:

- її неймовірну популярність для створення Telegram-ботів;
- простоту в використанні;
- читабельність;
- багатий функціонал;
- велику кількість відкритих бібліотек там готових рішень;
- підтримка асинхронності;
- багато хостинг-платформ підтримують Python, що полегшує розгортання та підтримку Telegram-ботів.

З мовою програмування визначилися, тепер перейдемо до фреймворків. Спочатку дамо визначення цьому поняттю. Фреймворк – структура, на основі якої можна створювати програмне забезпечення. Це, так би мовити, основа, тому ви не розпочинаєте свою роботу з нуля [14].

Перейдемо до фреймворків, які використовуються для розробки Telegram-ботів на Python. Розглянемо п'ять найкращих з них:

1. Aiogram – сучасний та повністю асинхронний фреймоврк для Telegram-ботів. Даний фреймворк працює з PyPy, підтримує Telegram Bot

Api, має анотації типів, підтримує мідлвари, має вбудований кінцевий автомат, тощо [15].

Плюси:

- асинхронність;
- багатий функціонал.

Мінуси:

- вищий поріг входу для початківців у порівнянні з іншими фреймворками.

На основі статистики репозитаріїв з GitHub було визначено, що aiogram відмічено 3 996 разів. Вцілому фреймворк завантажують 78 477 разів на тиждень. Кількість завантажень – це середня щотижнева кількість завантажень за останні 6 тижнів (рис. 2.2.1 – 2.2.2) [16]. Підтримує Telegram Bot API 6.9 та отримує швидке оновлення версій Bot API [17].

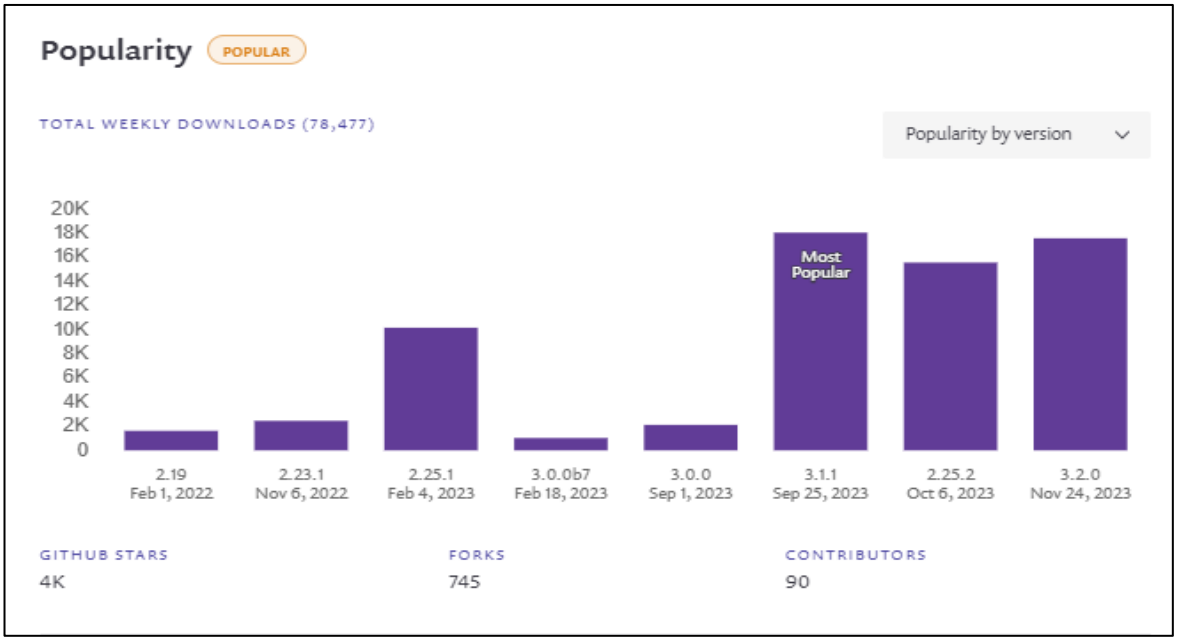

Рис. 2.2.1 – Статистика популярності фреймворка aiogram відносно версії

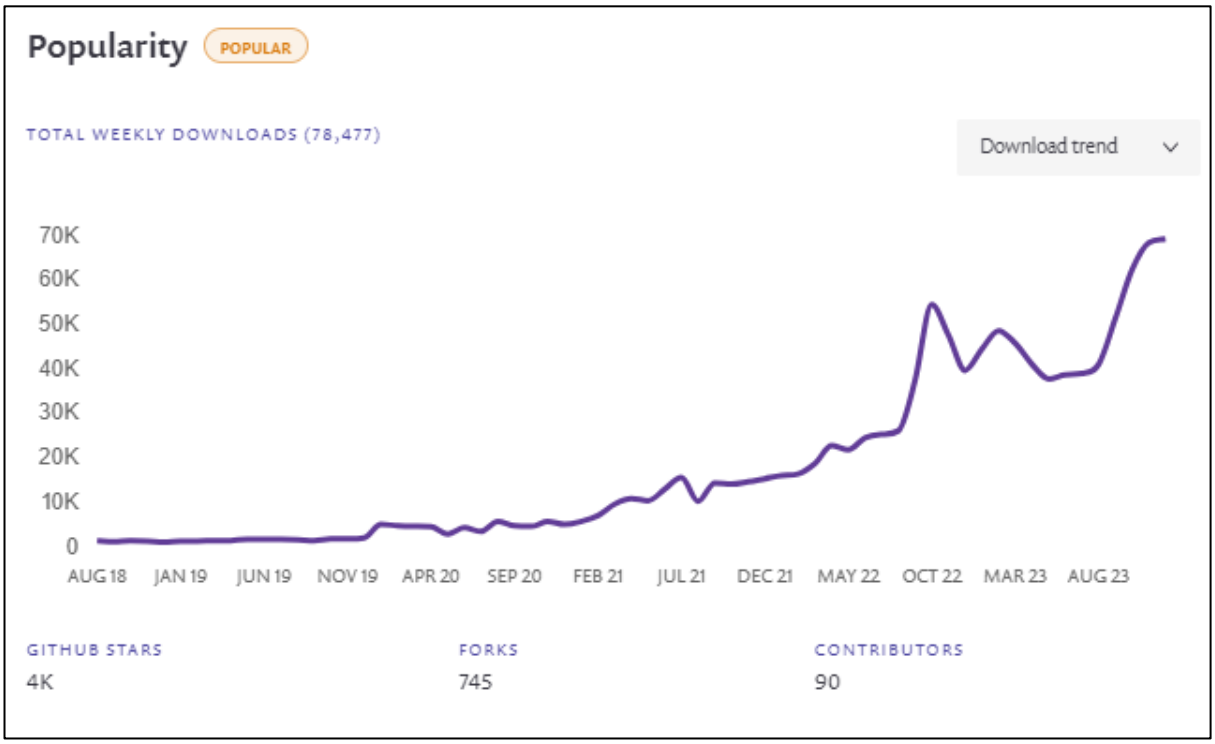

Рис. 2.2.2 – Тренд завантажень фреймворка aiogram

2. pyTelegramBotAPI або telebot – простий, швидкий та зручний в використанні фреймворк для розробки Telegram-ботів, але має обмежений функціонал у порівнянні з aiogram [18]. Підтримує Telegram Bot API 6.9 [19]. Плюси:

- швидко;
- зручно;
- простото в використанні.

Мінуси:

- обмежений функціонал;
- складніше з асинхронністю, адже її підтримка розроблена не так добре в порівнянні з aiogram

На основі статистики репозитаріїв з GitHub було визначено, що pyTelegramBotAPI відмічено 7 340 разів. Вцілому фреймворк завантажують 85 746 разів на тиждень (рис. 2.2.3 – 2.2.4) [20].

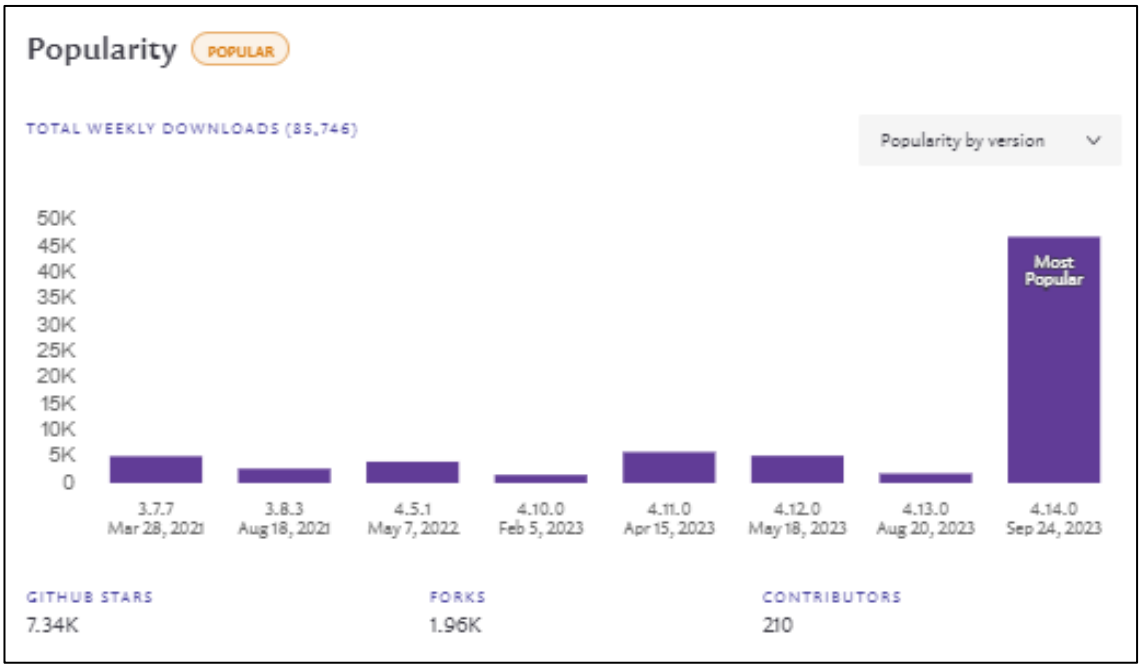

Рис. 2.2.3 – Статистика популярності фреймворка pyTelegramBotAPI відносно версії

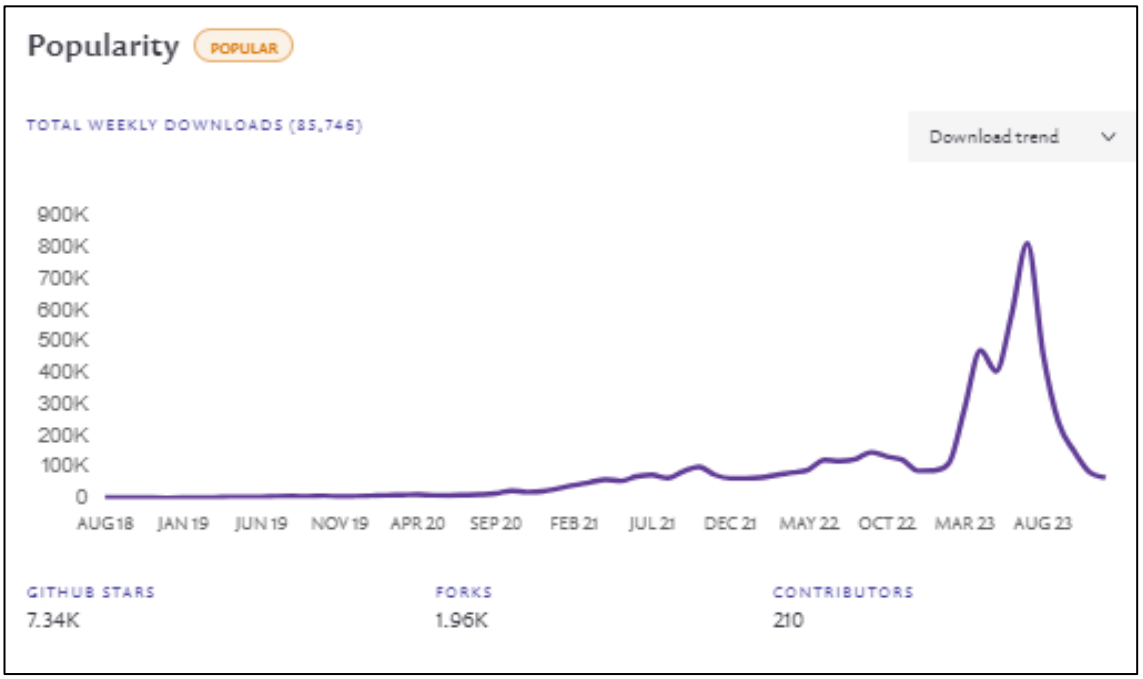

Рис. 2.2.4 – Тренд завантажень фреймворка pyTelegramBotAPI

3. python-telegram-bot – потужна бібліотека за допомогою якої створюють Telegram-ботів. Відсутня асинхронність, але має широкий функціонал. Підтримує всі типи та методи Telegram Bot API 6.9 [21].

Плюси:

- широка активна спільнота розробників;
- широкий функціонал.

Мінуси:

- відсутня асинхронність;
- можуть виникнути проблеми з профілюванням та налагодженнямм у випадку великих проєктів.

На основі статистики репозитаріїв з GitHub було визначено, що python-telegram-bot відмічено 23 703 разів. Вцілому фреймворк завантажують 300 358 разів на тиждень (рис. 2.2.5 – 2.2.6) [22].

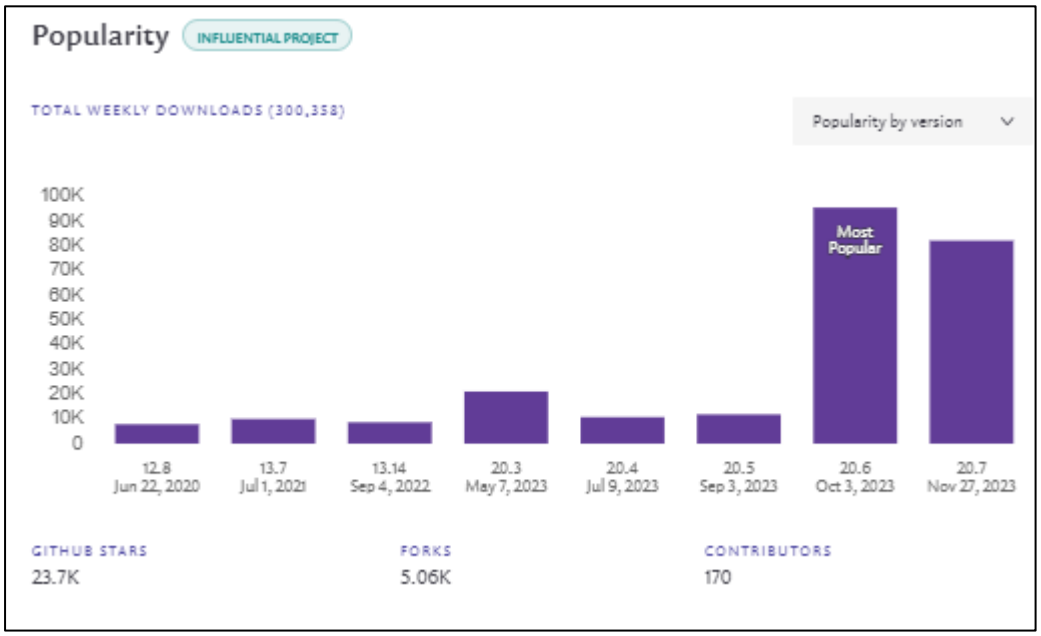

Рис. 2.2.5 – Статистика популярності бібліотеки python-telegram-bot відносно версії

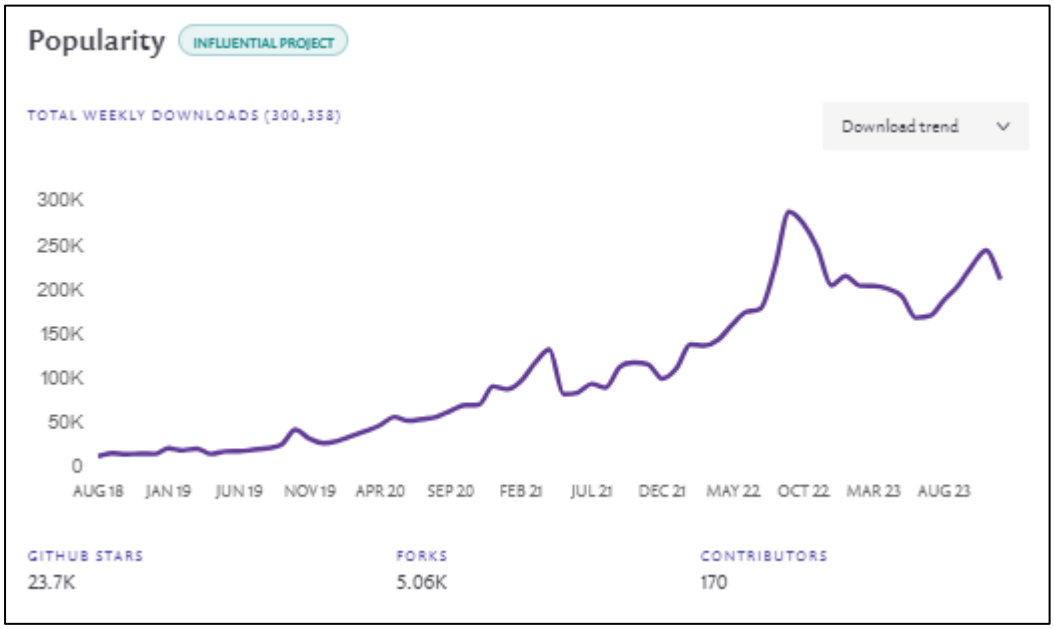

Рис. 2.2.6 – Тренд завантажень бібліотеки python-telegram-bot

4. Telethon – бібліотека для роботи з повним Telegram API. Є асинхронною. За допомогою не можна створювати складні проєкти, але

вимагає вищих знань від розробника та не спеціалізується на роботі з ботами [23].

Плюси:

- асинхронність;
- робота з повним Telegam API;
- має досить активну спільноту користувачів

Мінуси:

- не спеціалізується на роботі з ботами;
- вибір готових функцій менший;
- необхідний вищий поріг знань для розробника.

На основі статистики репозитаріїв з GitHub було визначено, що python-telegram-bot відмічено 8 444 разів. Вцілому фреймворк завантажують 379 753 разів на тиждень (рис. 2.2.7 – 2.2.8) [24].

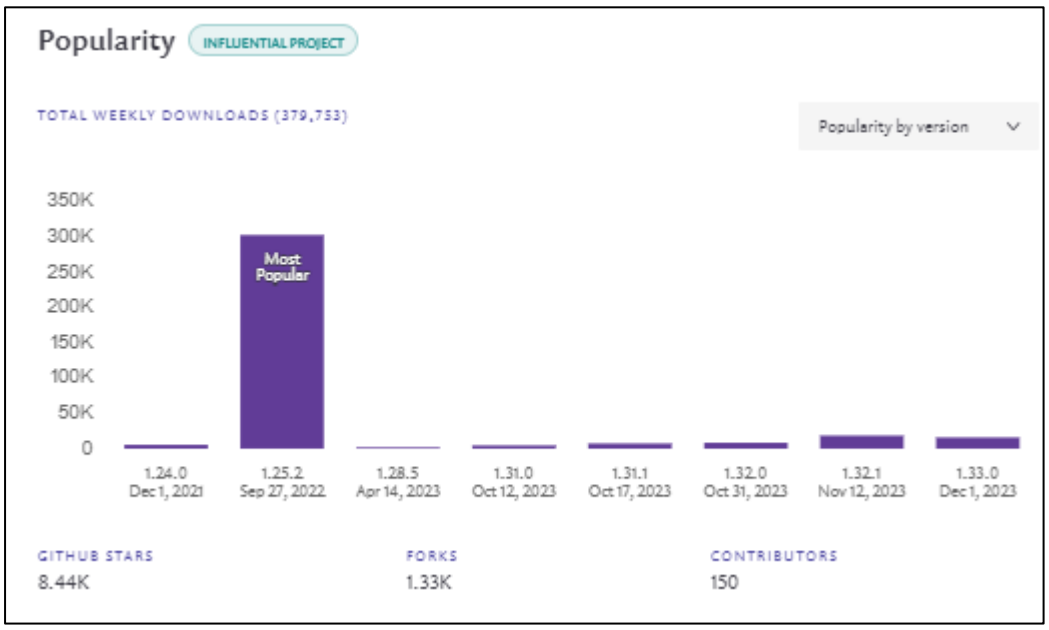

Рис. 2.2.7 – Статистика популярності бібліотеки telethon відносно версії

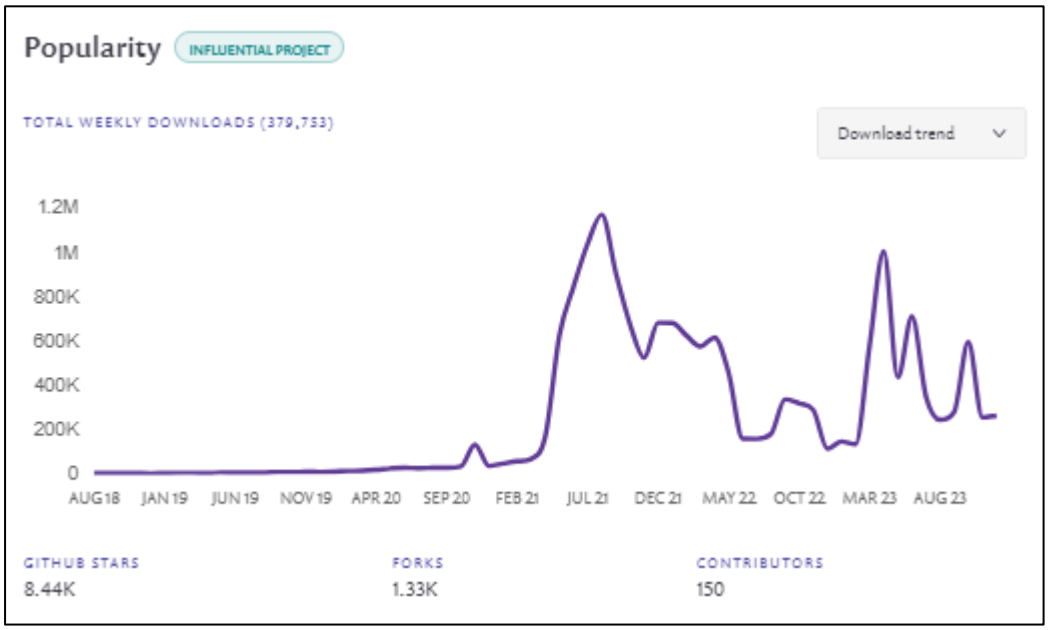

Рис. 2.2.8 – Тренд завантажень бібліотеки telethon

5. Pyrogram – сучасний та асинхронний фреймворк для роботи з повним Telegram API. Схожа ситуація як і з telethon: дає можливість розробляти складні проєкти, але необхідно більше знань та неспеціалізований на роботі з ботами [25].

Плюси:

- асинхронність;
- багатофункціональність;
- активна спільнота;
- робота з повним Telegram API.

Мінуси:

- несумісність з деякими старими весріями Python;
- вищий поріг знань для розробників.

На основі статистики репозитаріїв з GitHub було визначено, що python-telegram-bot відмічено 3 855 разів. Вцілому фреймворк завантажують 45 743 разів на тиждень (рис. 2.2.9 – 2.2.10) [26].

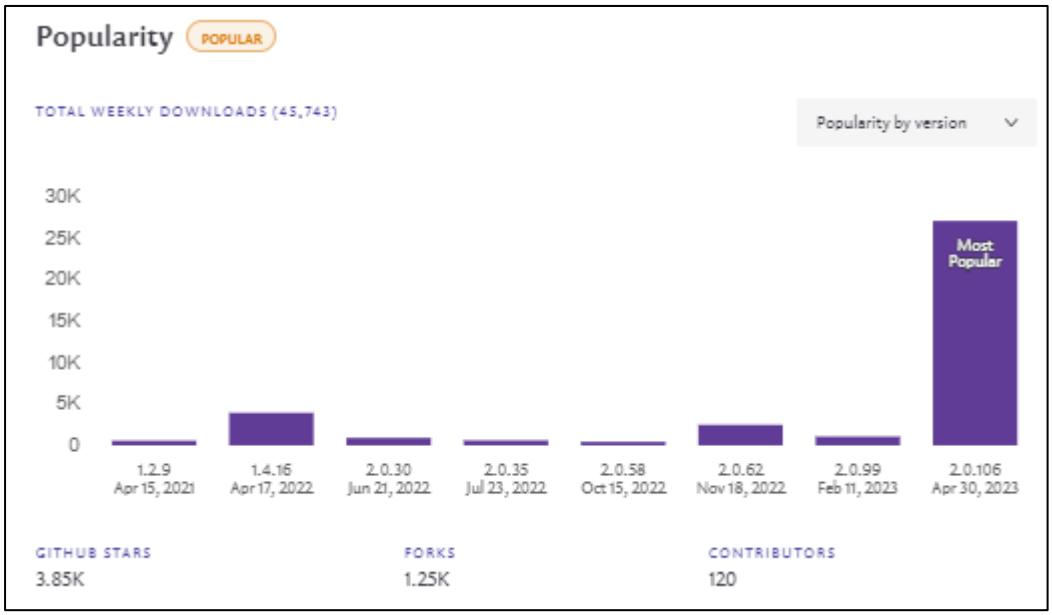

Рис. 2.2.9 – Статистика популярності фреймворка pyrogram відносно версії

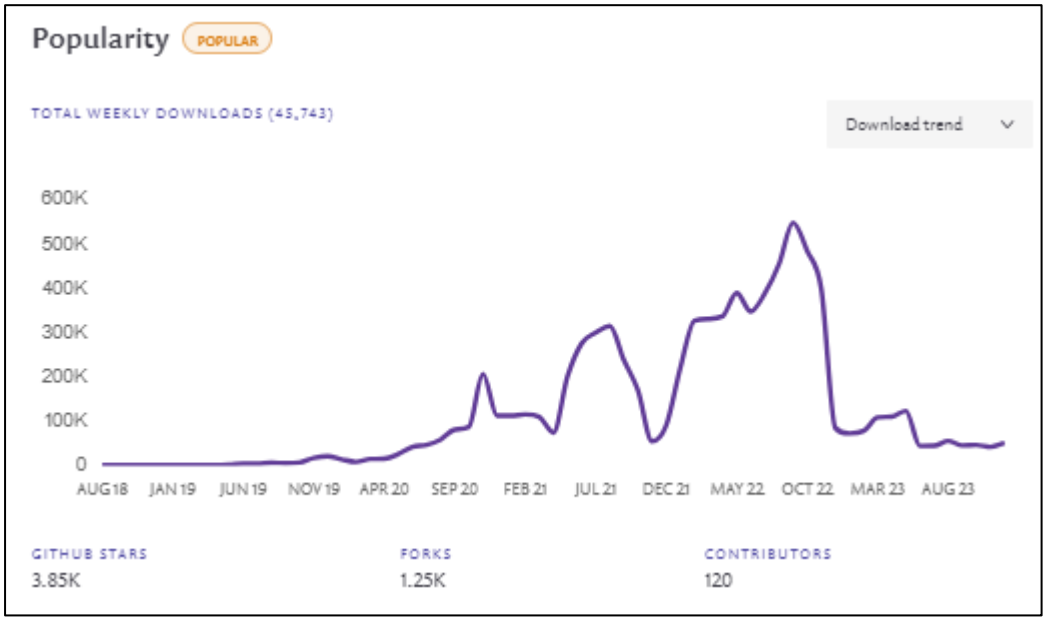

Рис. 2.2.10 – Тренд завантажень фреймворка pyrogram

Аналізуючи наведену вище інформацію, мій вибір впав на фреймворк aiogram. Адже він:

- асинхронний, що дозволить ефективно обробляти велику кількість запитів і відповідей без блокування потоку;
- має зручний та простий синтаксис, тобто робить код читабельним;
- велика функціональність (робота з клавіатурами, inline-режимами, медіа-файлами, тощо);
- детальна документація;
- тощо.

2.3 Віртуальне середовище для розробки Telegram-бота

Спершу окреслимо поняття віртуального середовища. Віртуальне середовище (virtual environment) – це ізольований простір, де можна встановити та управляти залежностями конкретного проєкту. Його використання забезпечує чистоту середовища, запобігає виникненню конфліктів між версіями бібліотек, що використовуються в різних проєктах.

Якщо коротко, то віртуальне середовище – це про ізоляцію проєкта.

З Python3.3 для створення та управління віртуальним середовищем використовується бібліотека «venv».

2.4 Огляд та вибір бази даних

При створенні Telegram-бота буде необхідна база даних для можливості зберігання та управління інформацією. Розглянемо декілька найпопулярніших баз даних для Python проєктів:

- SQLite. Найбільш широко використовувана база даних для Python завдяки легкості та простоті використання [27]. Немає необхідності встановлення окремого сервера баз даних, адже SQLite входить за замовчуванням до встановлення Python. Обмін файлами, де зберігаються дані, не викликає труднощів. Завдяки своїй простоті та великій кількості функцій, вважається однією з найкращих баз даних для розробки на Python [28].
- MySQL – реляційна система управління базами даних. Має відкритий тий вихідний код. Працює за моделлю «клієнт-сервер». Вона була розроблена шведською компанією в 1984 році, потім була викуплена американською компанією, і в кінцевому результаті в 2010-му році потрапила до рук Oracle, котра зараз займається її розробкою та підтримкою [29]. Користувачами MySQL є всесвітньо відомі компанії: YouTube, PayPal, LinkedIn, Facebook, Uber, Twitter, GitHub, Netflix, тощо [30]. Свою популярність дана система отримала через простоту, продуктивність, безпеку та стандартизацію. Щодо використання MySQL з Python, то завдяки коннектору MySQL Python можна отримати доступ до MySQL безпосередньо з коду на Python. Підтримка JSON робить MySQL гнучкою, що призводить до легкості використання бази даних з Python. Фреймворк Django має вбудовану підтримку [28].
- PostgreSQL – це потужна об’єктно-реляційна система баз даних з відкритим вихідним кодом. Розробкою даної системи займаються понад 35 років. Свою популярність та стійку репутацію отримала за надійність, функціональну стабільність та продуктивність [31]. PostgreSQL пропонує більший функціонал, ніж MySQL. Не бажано використовувати для read-only операцій, адже предназначена для read-write. Використовується при управлінні велики датасетами та складними запитами [32]. Щодо використання PostgreSQL з Python, то завдяки psycopg адаптеру PostgreSQL можна отримати доступ до PostgreSQL

безпосередньо з коду на Python. Фреймворк Django поставляється в інтеграції з PostgreSQL [28].

- MongoDB – найвідоміша документоорієнтована NoSQL база даних. Дозволяє розробникам зберігати дані в вигляді об’єктів типу JSON. За замовчуванням дані захищені неперервним шифруванням при зберіганні та передачі, складним управлінням доступом з використанням ролей, постійною аутентифікацією, мережевою ізоляцією, тощо. MongoDB інтегрована з Google Cloud, Microsoft Azure, Postman, AWS, Accenture, Slack, Spring, тощо [33]. Щодо використання MongoDB з Python, то драйвер PyMongo забезпечує ідіоматичний доступ з додатку Python. Фреймворк Flask поставляється в інтеграції з PyMongo [28].
- Redis – сховище даних в пам’яті з відкритим вихідним кодом. Використовується як сховище даних в реальному часі, для кешування та зберігання сеансів, для потокової передачі та обміну повідомленнями. Дозволяє створювати додатки з моделями даних JSON. Redis використання такими відомими компнаіями як Twitter, GitHub, StackOverflow, Snapchat, Airbnb, Slack, Trello, тощо [34]. Щодо використання Redis з Python, то для спрощення використаня Redis з застосунків, що потребують низької затримки та високої пропускної можливості, використовують модуль redis-py [28].

Аналізуючи інформацію вище, за базу даних я обрала PostgreSQL, адже:

- надійно;
- має вбудовану підтримку JSON, то ж може ефективно обробляти та зберігати дані в цьому форматі;
- функціоналість;
- висока продуктивність;
- гнучкість та можливість адаптації під потреби проєкту;
- стійкість до великої кількості користувачів.

Отже, за інформаційну технологію для адаптивної підготовки абітурієнтів до вступних іспитів було обрано Telegram-бот, для реалізації якого було обрано:

- мову програмування Python з версією 3.10.12;
- web-фреймворк aiogram для створення Telegram-ботів;
- venv, як модуль для створення та керування віртуальними середовищами;
- систему контролю баз даних PostrgreSQL, як одну з найпопулярніших, стабільних та швидких.

В ході розробки було обрано до використання наступні інструменти розробника для пришвидшення і полегшення процесу:

- PyCharm - середовище розробки для мови Python;
- DataGrip - середовище розробки і перегляду баз даних.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ

3.1 Архітектурна модель

Розглянемо спрощену структуру, яка береться за основу для Telegram-бота:

- Telegram Bot API (через нього бот буде взаємодіяти з Telegram, відправляти відповіді та отримувати повідомлення);
- асинхронний веб-сервер (aiogram використовує асинхронний підхід, тому необхідно асинхронний веб-сервер, наприклад від бібліотеки aiohttp;
- база даних (для зберігання даних, які буде надсилати бот користувачеві. В нашому випадку для зберігання питань, відповідей та записів про користувачів);
- логування (для відстеження дій та виявлення помилок).

Перейдемо тепер до критеріїв майбутньої технології:

- масштабованість;
- можливість додавати новий функціонал, покращувати та редагувати вже існуючий.

Виходячи з інформаційного огляду, найбільш ефективною технологією адаптивної підготовки в розрізі закріплення та перевірки знань та навичок користувача є та що відповідає наступним критеріям:

- надає пояснення до відповідей (посилання на матеріали);
- користувач не має змоги перемикатися між питаннями;
- список питань в тесті постійно змінюється, беручи їх з великої бази питань;
- при кожній спробі проходження питання, порядок відповідей змінюється, що не дає користувачеві змоги «зазубрити» відповіді;

- пропонує користувачу релевантні питання з темами які є для нього проблемними.

Побудуємо схематично етапи роботи майбутньої технології (рис.3.1):

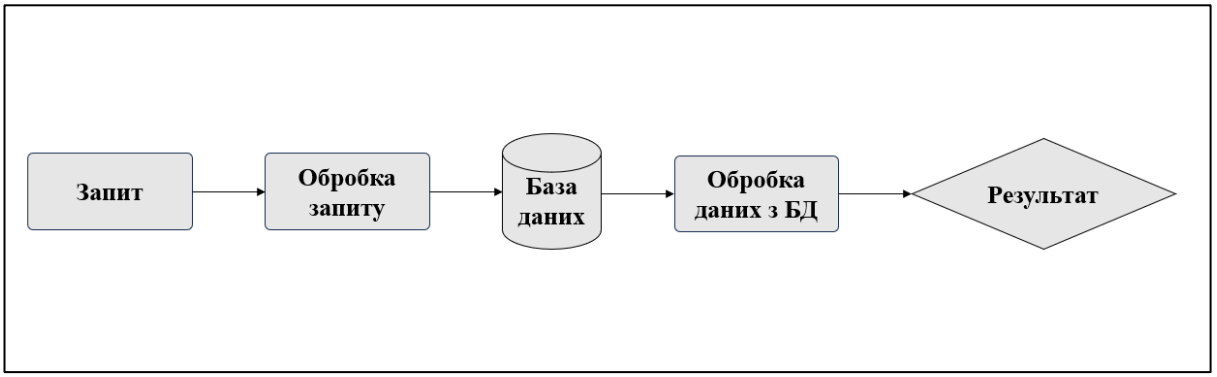

Рис.3.1.1 – Схема роботи майбутньої технології

3.2 Програмна реалізація

Створення та реалізація Telegram-бота можуть бути різними, але основні етапи залишаються стандартними [35].

Найперше що необхідно зробити – це отримати токен для Telegram-бота. Для цього заходимо в Telegram, в пошуку знаходимо бот @BotFather запускаємо його (рис.3.2.1). Він надає можливість створювати та керувати ботами в Telegram. Саме завдяки ньому ми отримуємо токен для нашого бота.

Рис.3.2.1 – Telegram-бот @BotFather

Для реєстрації бота проходимо інтуїтивно зрозумілі кроки. Спочатку вводимо /start для запуску BotFather, далі буде повідомлення з інструкцією що робити. В нашому випадку вводимо /newbot, адже ми створюємо нового бота. Вводимо для нього назву, юзернейм. І вже потім при успішній реєстрації бота отримуємо його токен (рис.3.2.2).

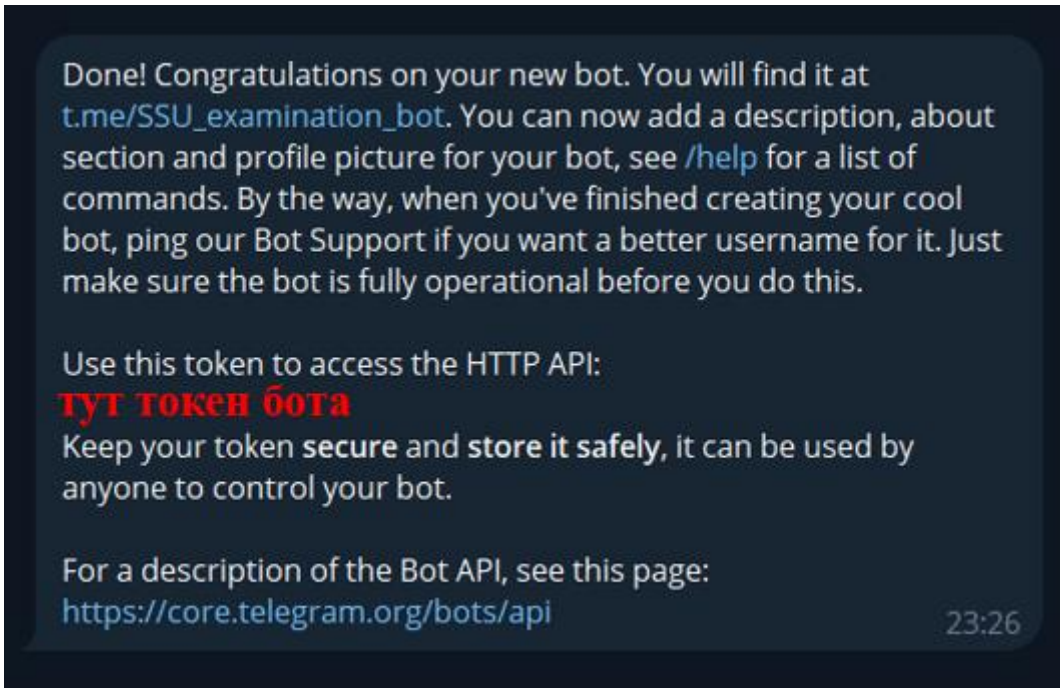

Рис.3.2.2 – Повідомлення про успішну реєстрацію з токеном бота

Далі переходимо до створення програмної частини бота. Окреслимо як має працювати бот. Він має складатися з двох інтерфейсів: користувач (абітурієнт) та адміністратор (викладач).

Розглянемо що потрібно для сторони адміна:

- створення дисциплін, їх перегляд, редагування та видалення;
- створення питань до дисциплін, їх перегляд, редагування та видалення;
- створення відповідей до питань, їх перегляд, редагування та видалення.

Розглянемо що потрібно для сторони юзера:

- можливість обирати дисципліни з яких хоче пройти тестування;
- можливість проходження тестування;
- можливість отримання рекомендаційного матеріалу.

Створюємо проєкт в PyCharm (рис.3.2.3).

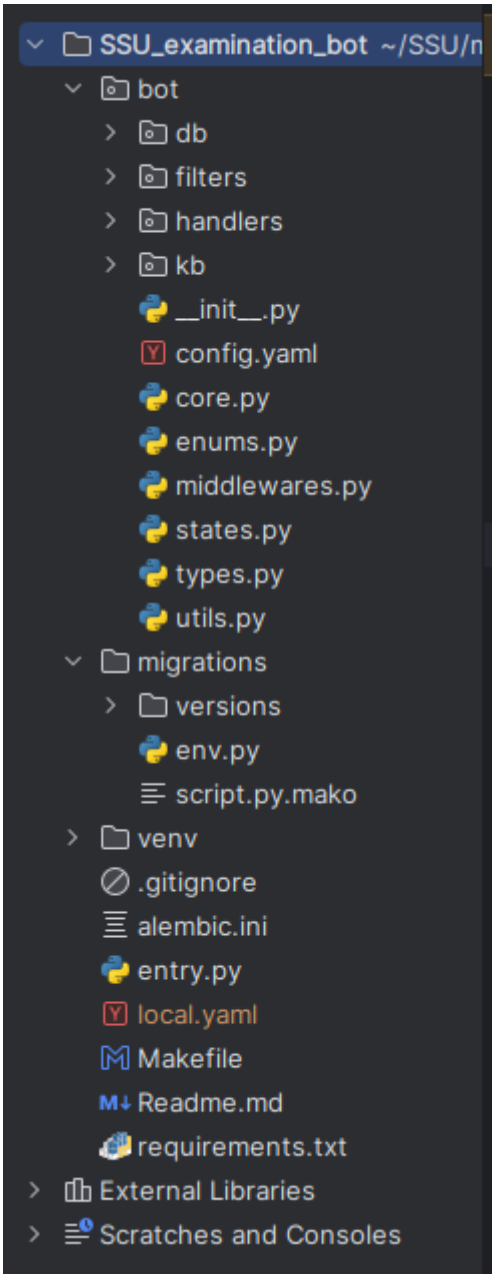

Рис. 3.2.3 - Структура проєкту для Telegram-бота в PyCharm

Розглянемо папку «bot»:

- папка «db» - папка для бази даних. Подивимося що всередині неї (рис. 3.2.4).

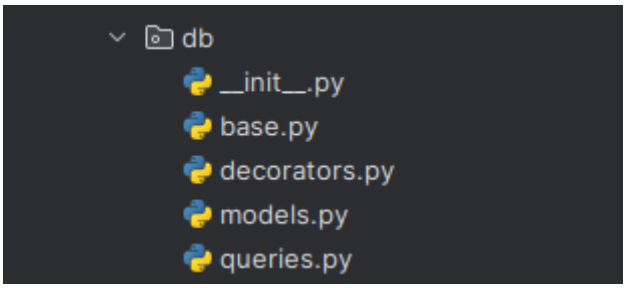

Рис. 3.2.4 - Вміст папки «db»

- base.py - містить класи, наприклад, для отримання атрибутів з поточного контексту, базовий клас для моделей бази даних (містить в собі методи, такі як “to_dict”, “get_or_create”, “get”, ‘get_list”, “create”, “update”, “delete” та “exists”), клас для відстеження часу створення та оновлення запису, клас для створення сесій з базою даних; функції для підключення до бази даних та відключення; екземпляр “SessionProxy” (надає зручний доступ до поточної сесії);
- decorators.py - містить декоратор для роботи з асинхронними сесіями бази даних;
- models.py - містить моделі для представлення об’єктів у базі даних;
- queries.py - містить запити до бази даних;

● папка «filters» - папка для фільтрів. Містить в собі файл admin.py, котрий створений для перевірки чи є користувач адміном.

● папка «handlers» - папка для хендлерів. Подивимося, що всередині папки (рис. 3.2.5).

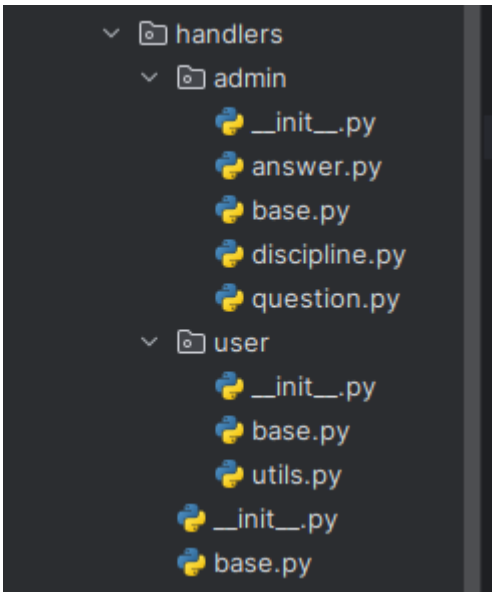

Рис. 3.2.5 - Вміст папки «handlers»

Маємо дві папки - для адміна та юзера та файл base.py.

- base.py - містить хендлер, що опрацьовую “відміну” при вводі /cancel.
- Папка «admin»:
  - answer.py - містить хендлери для відповідей;
  - base.py - містить хендлер для входу в адмінстративне меню;
  - discipline.py - містить хендлери для дисциплін;
  - question.py - містить хендлери для питань.
- Папка «user»:
  - base.py - містить хендлери, які викликаються після входу в режим користувача;
  - utils.py – містить функцію, яка відправляє питання користувачу;

- папка «kb» - папка, що містить клавіатури для адміна та юзера (рис.3.2.6).

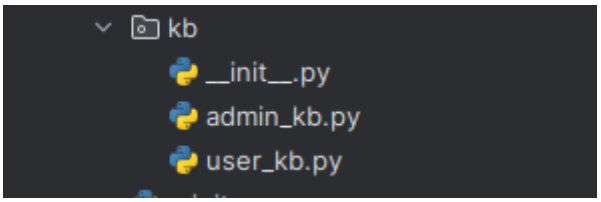

Рис.3.2.6 - Вміст папки «kb»

- admin_kb.py - містить клавіатури, що використовуються для інтерфейсу адміна;
- user_kb.py - містить клавіатури, що використовуються для інтерфейсу юзера.

- config.yaml - основний файл конфігурації системи.
- core.py - файл, де створюються об’єкти бота та диспетчера для роботи з aiogram.
- enums.py - файл, де визначаються декілька класів перерахувань використовуючи модуль “enum”.
- middlewares.py - файл, визначає мідлвар для обробки медіагруп у aiogram.
- states.py - файл, де визначаються стани для кожного об’єкта в боті, котрий може виникати при взаємодії з користувачем.
- types.py- файл, де описується набір даних для статистики.
- utils.py - файл, де визначаються меткалас Singleton та клас конфігурації.

Розглянемо папку «migrations»:

- папка «versions» - папка для зберігання версій міграцій;
- env.py - містить скрипт alembic для запуску міграцій бази даних;
- script.py.mako - шаблон файлу міграції alembic.

Також в структурі проєкту маємо:

- alembic.ini - конфігураційний файл alembic;
- entry.py - файл містить скрипт, що є вхідною точкою для запуску бота;
- Makefile - файл, що містить набір команд, де команда make запускає їх з цього файлу;
- Readme.md - текстовий файл, що містить інформацію про даний проєкт;
- requirements.txt - файл, що містить список всіх пакетів та бібліотек, що необхідні для роботи проєкту.

Базуючись на потребах, створимо базу даних для бота. Для цього в файлі models.py створимо моделі для представлення об’єктів у базі даних. Нам необхідні такі об’єкти:

- discipline - зберігає дані про дисципліни;
- question - зберігає дані про питання дисциплін;
- answer - зберігає відповіді на питання;
- user - зберігає дані про юзерів;
- user_answer - зберігає дані в співвідношенні юзер-відповідь.

Наприклад, об’єкт Question описується таким чином:

```
class Question(Base, CreatedMixin, UpdatedMixin):
  __tablename__ = "question"


discipline_id = Column(Integer, ForeignKey("discipline.id", ondelete="CASCADE"), nullable=False)
        photo = Column(ARRAY(String), nullable=True)
        text = Column(String(100), nullable=False)
        material = Column(String(800), nullable=False)
```

Об’єкт складається з:

- discipline_id - айді дисципліни, що приходить від об’єкту Discipline як зовнішній ключ. Тобто ми знаємо до якої дисципліни відноситься це питання;
- photo - фото до питання. Можливо питання потребує відобразити якийсь графік чи картинку;
- text - текст питання;
- material - посилання на матеріал, що відноситься до цього питання. Виступає в ролі рекомендаційного матеріалу для засвоєння.

Як результат, маємо базу даних, що відображено на діаграмі рис.3.2.7.

Рис. 3.2.7 - Діаграма бази даних

Перейдемо до моменту адаптивності в боті. В чому вона буде заключатися? Якщо просто, то якщо користувач дає три відповіді неправильно, то йому пропонуватиметься простіше питання. Якщо хоча б одна відповідь буде правильна, то пропонуватиметься складніше питання. Це дасть змогу тримати інтерес користувача, аби він “не закинув” тестування.

Ще гарною функцією є те, що відповіді на питання кожного разу будуть міняти свій порядок. Це зроблено аби юзер не “зазубрював” порядок правильних відповідей. Як реалізується ця адаптивність?

Юзер обирає дисципліну з якої хоче пройти тестування. Далі з’являється перше питання. Маємо два випадки:

- навчена система;
- не навчена система.

Якщо система не навчена ще, то складність всіх питань буде вважатися однаковою. І користувачу будуть пропонуватися питання випадково до моменту поки система не пройде навчання.

У випадку, коли система навчена, ми маємо статистику складності питання. Питання розподіляються на дві групи: складні та прості. Статистика визначається як відношення кількості правильних відповідей до загальної кількості відповідей користувача по даному питанню. По цьому параметру можна визначати складність питань дисципліни індивідуально для кожного користувача. Тоді маємо три випадки:

- якщо користувач ще не відповідав на питання даної дисципліни, або має менше трьох відповідей. Тоді пропонуємо йому випадкове питання. Це тому що, статистику для такого користувача ми не маємо;

- якщо користувач відповідає підряд на три питання неправильно. Тоді бот пропонує простіше питання, аби тримати інтерес користувача до тесту та фокусувати його увагу;
- якщо користувач відповів на останні три питання і з них менше трьох неправильних. Тоді бот надсилає питання складнішого рівня.

Описаний принцип роботи зображено на рис. 3.2.8. Код, що виконує даний механізм наведено у Додатку.

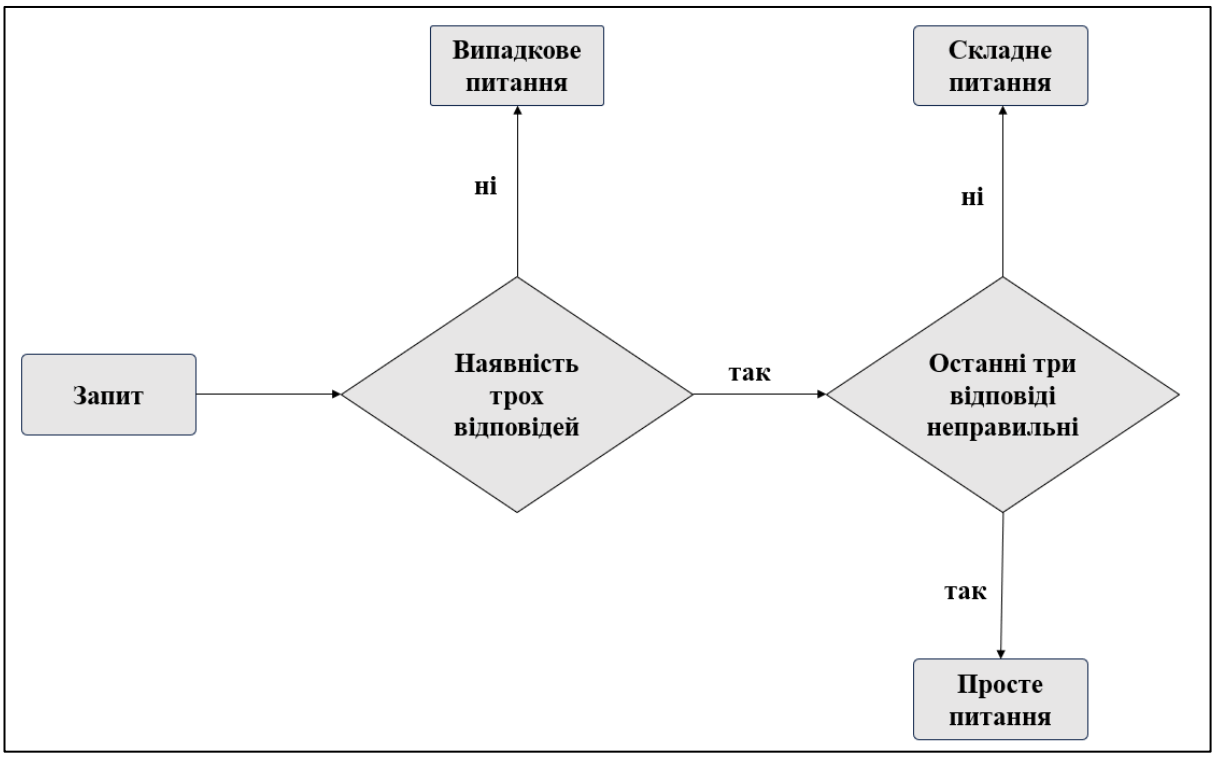

Рис. 3.2.8– Принцип підбору релевантних питань

3.3 Огляд створеної інформаційної технології

Розглянемо спочатку сторону адміністратора. Аби перейти до інтерфейсу адміна, необхідно ввести /admin (рис.3.3.1).

Рис.3.3.1 - Вхід як адміністратора

Далі натискаємо на кнопку “Дисципліни” та бачимо список дисциплін, які вже створено (рис.3.3.2).

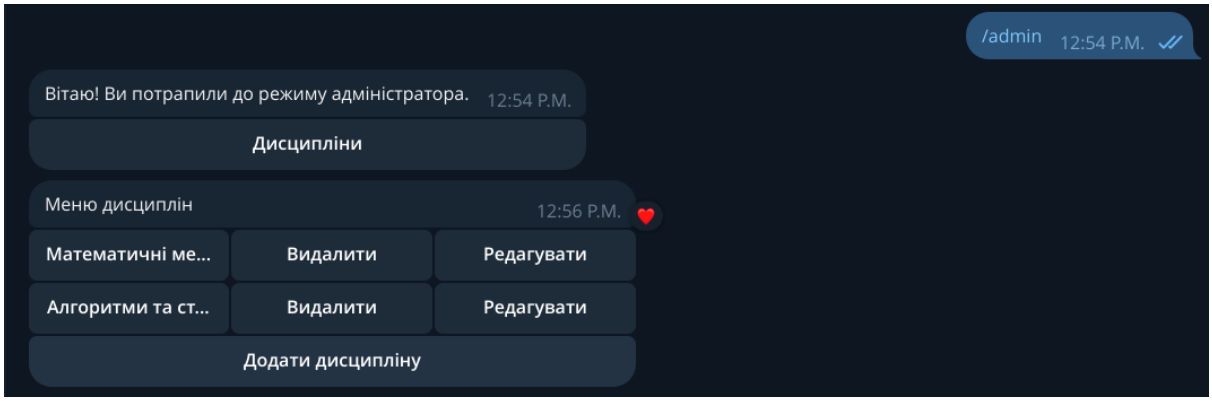

Рис.3.3.2 - Перегляд переліку дисциплін.

Як бачимо, дисципліну можна видалити та редагувати. Перейдомо зараз до створення нової дисципліни. Натискаємо на кнопку “ Додати дисципліну”, вводимо назву та опис дисципліни (рис.3.3.3).

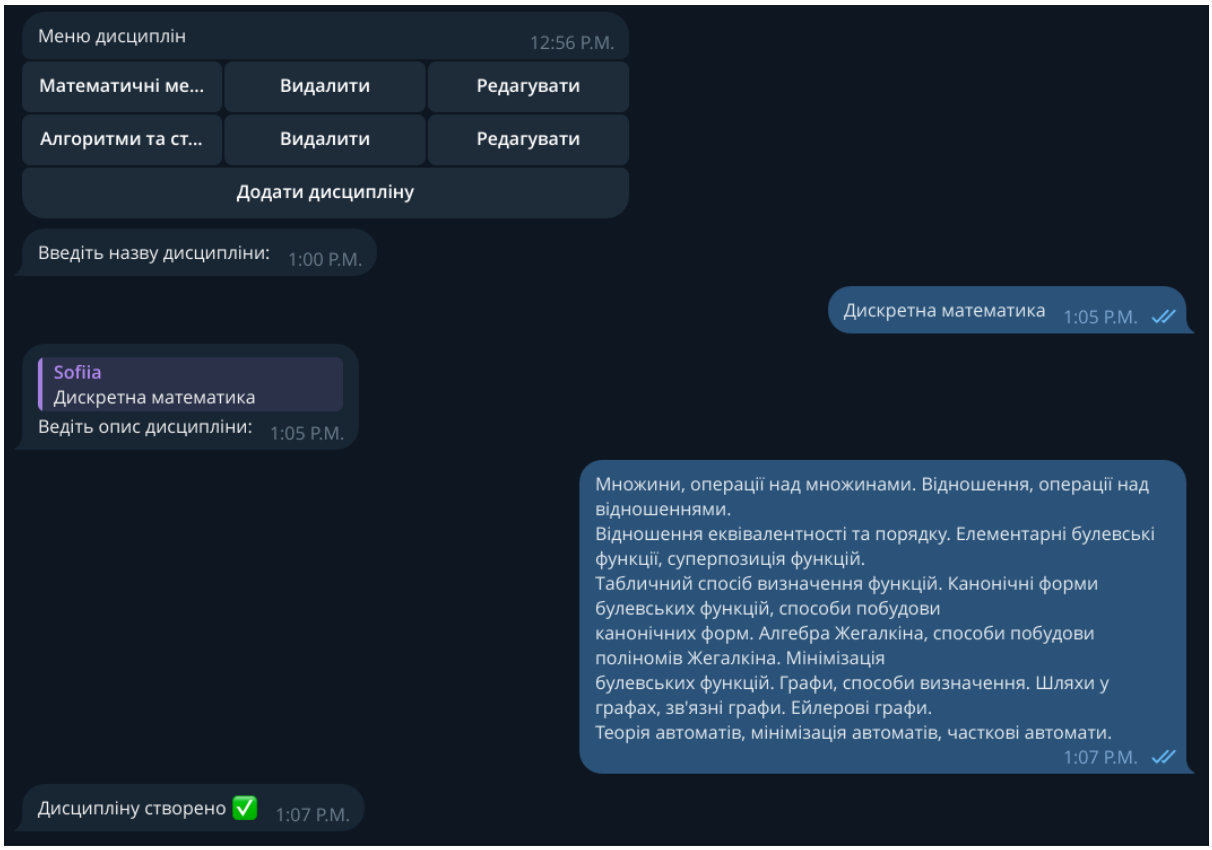

Рис. 3.3.3 - Створення дисципліни “ Дискретна математика”

Тепер новостворена дисципліна відображається в списку дисциплін (рис.3.3.4).

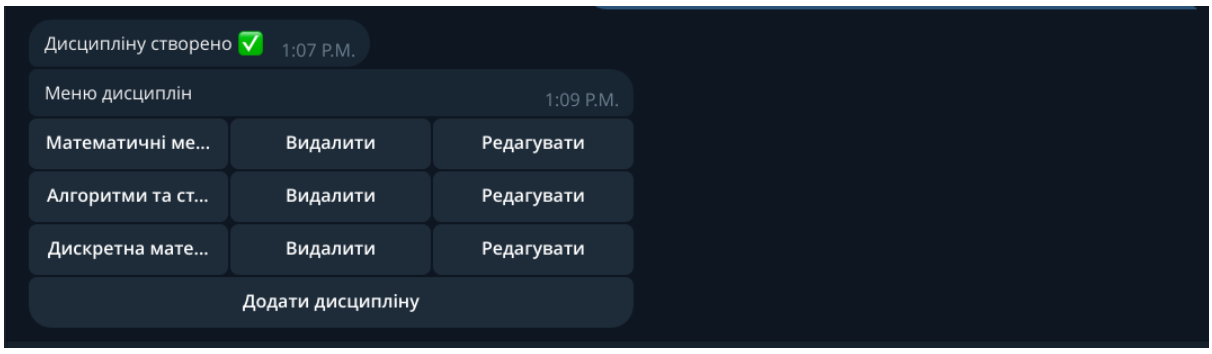

Рис. 3.3.4 - Оновлений список дисциплін

Розглянемо тепер детальніше кнопку “Редагувати” навпроти дисципліни “Дискретна математика”. При натисканні ми отримує дві функції:

- редагувати назву;
- редагувати опис.

Спробуємо відредагувати, наприклад, назву дисципліни (рис.3.3.5).

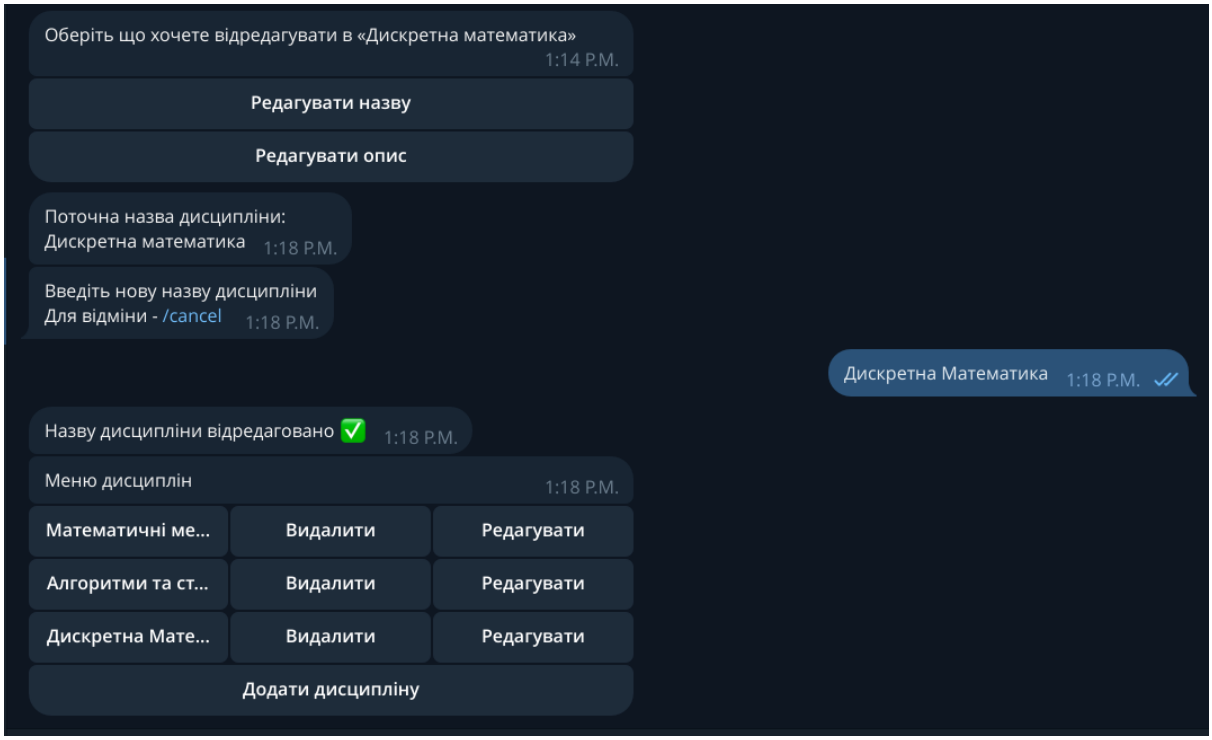

Рис. 3.3.5 - Редагування назви дисципліни “Дискретна математика” на “Дискретна Математика”

Тепер перейдемо до створення питань до даної дисципліни. Натискаємо на саму дисципліну і далі мають відображатися питання до неї (рис.3.3.6).

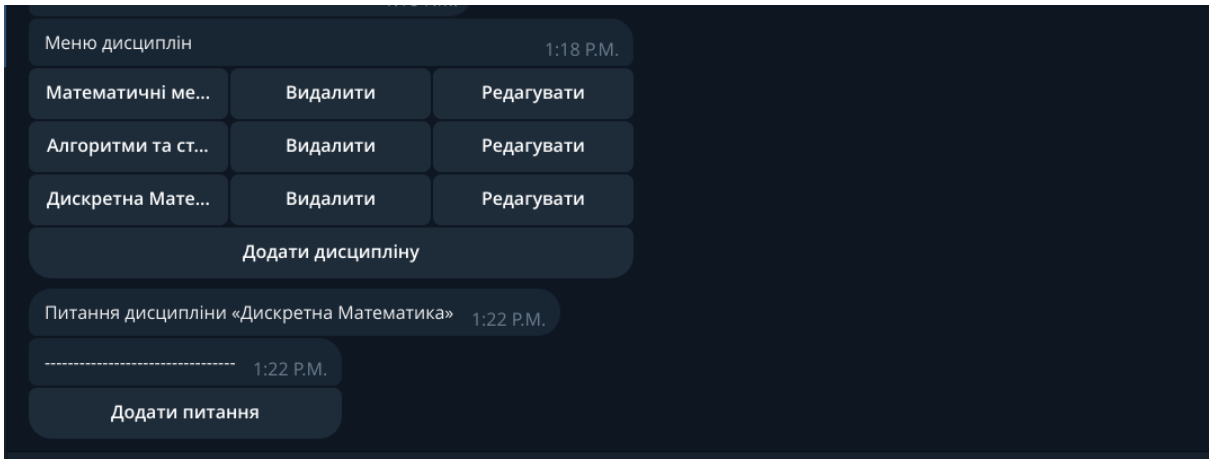

Рис.3.3.6 - Відображення питань до дисципліни “Дискретна Математика”

Так як це новостворена дисципліна, то питань ми ще не маємо в переліку, то необхідно їх створити натиснувши на кнопку “Додати питання” (рис.3.3.7).

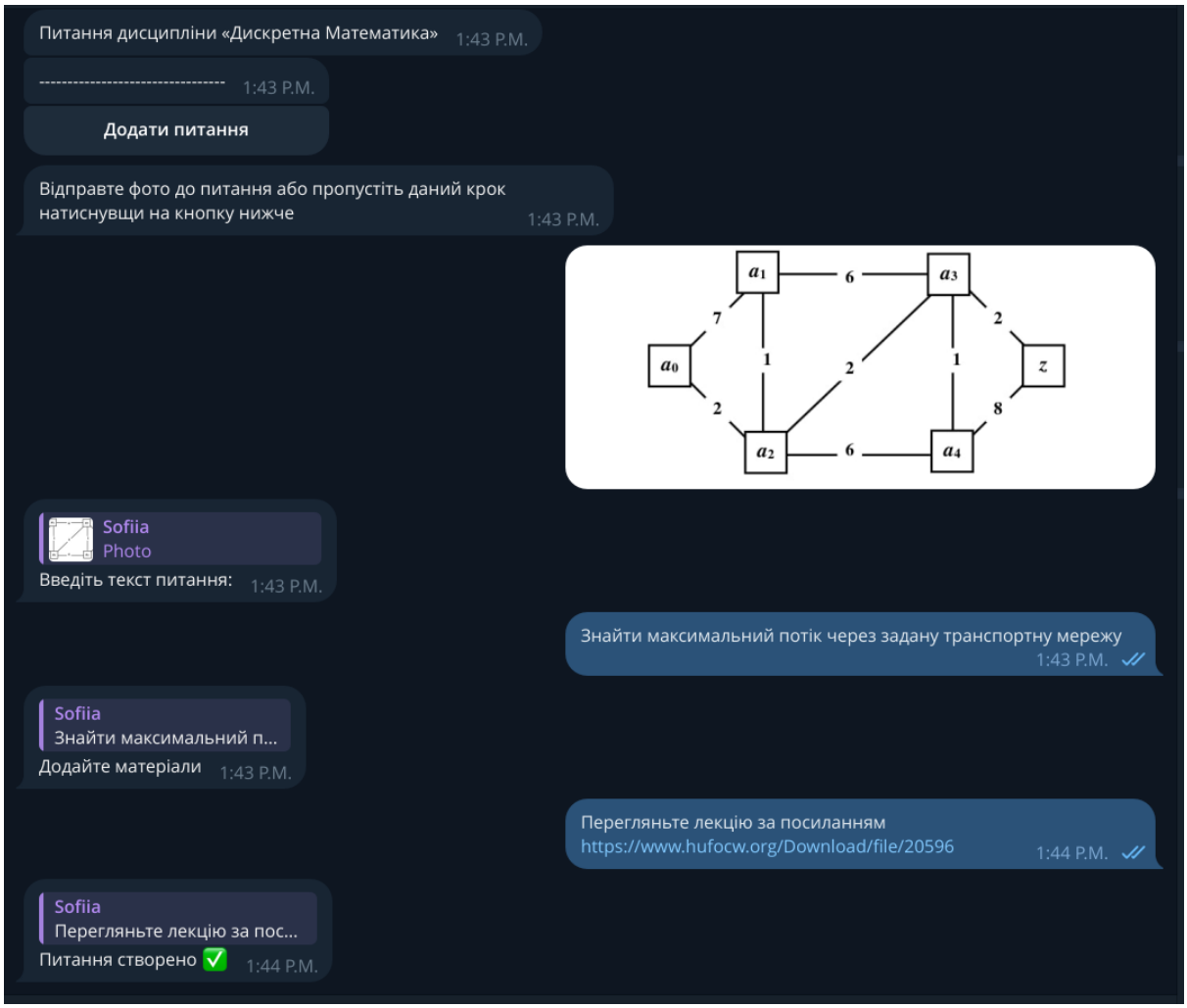

Рис.3.3.7 - Створення питання до дисципліни “Дискретна Математика”

Як бачимо, тепер питання відображається в переліку питань до дисципліни (рис.3.3.8).

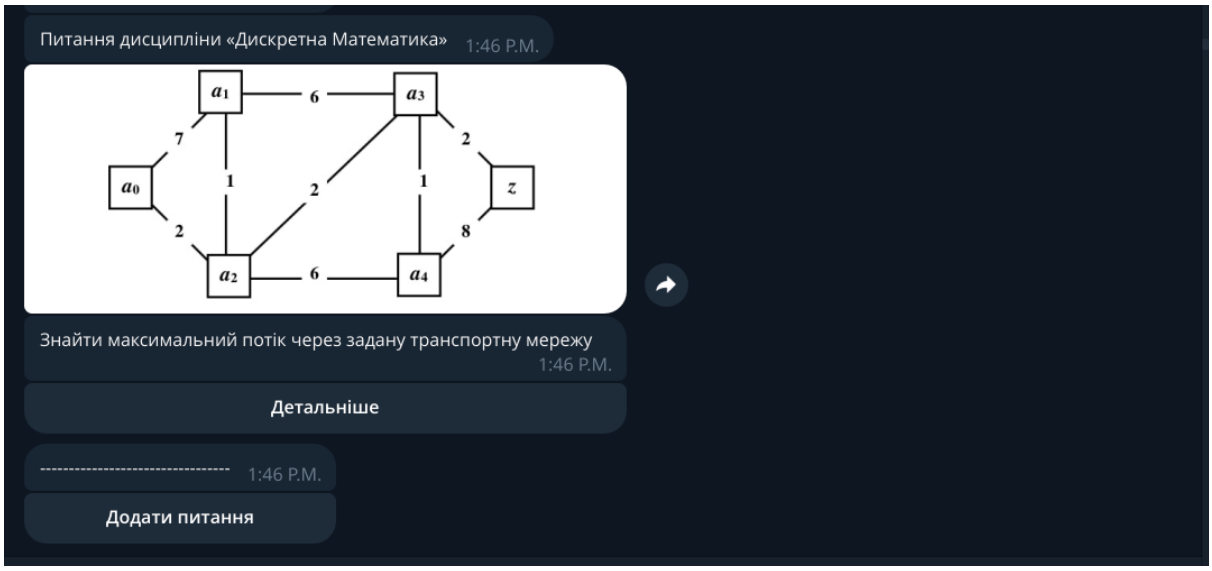

Рис.3.3.8 - Перелік питань до дисципліни “Дискретна Математика”

Тепер перйдемо до створення відповідей до даного питання. Для цього натискаємо на кнопку “Детальніше під питанням” та отримуємо перелік можливих дій з даним питанням (рис.3.3.9).

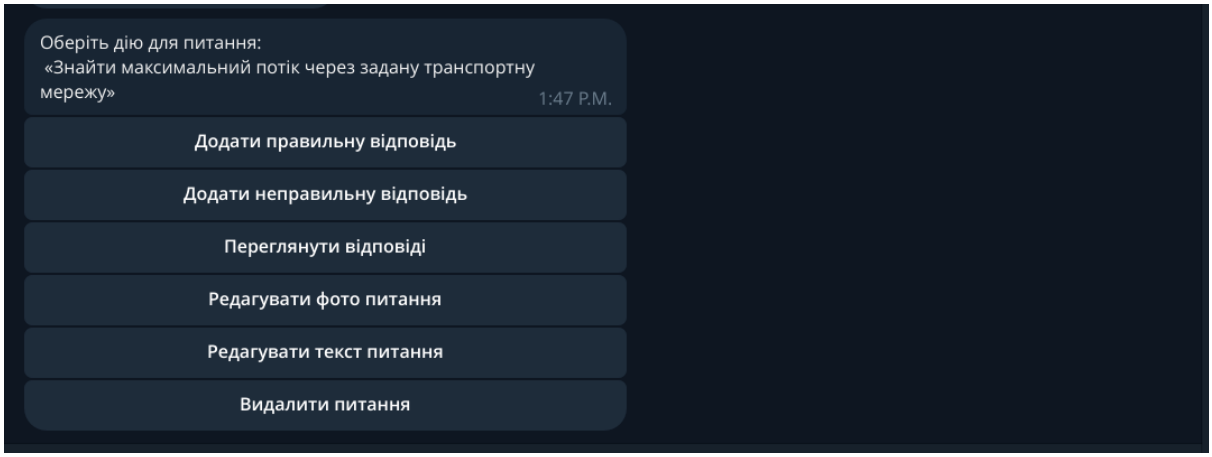

Рис. 3.3.9 - Перелік дій з питанням

Додамо відповіді на питання натискаючи на відповідні кнопки (рис.3.3.10).

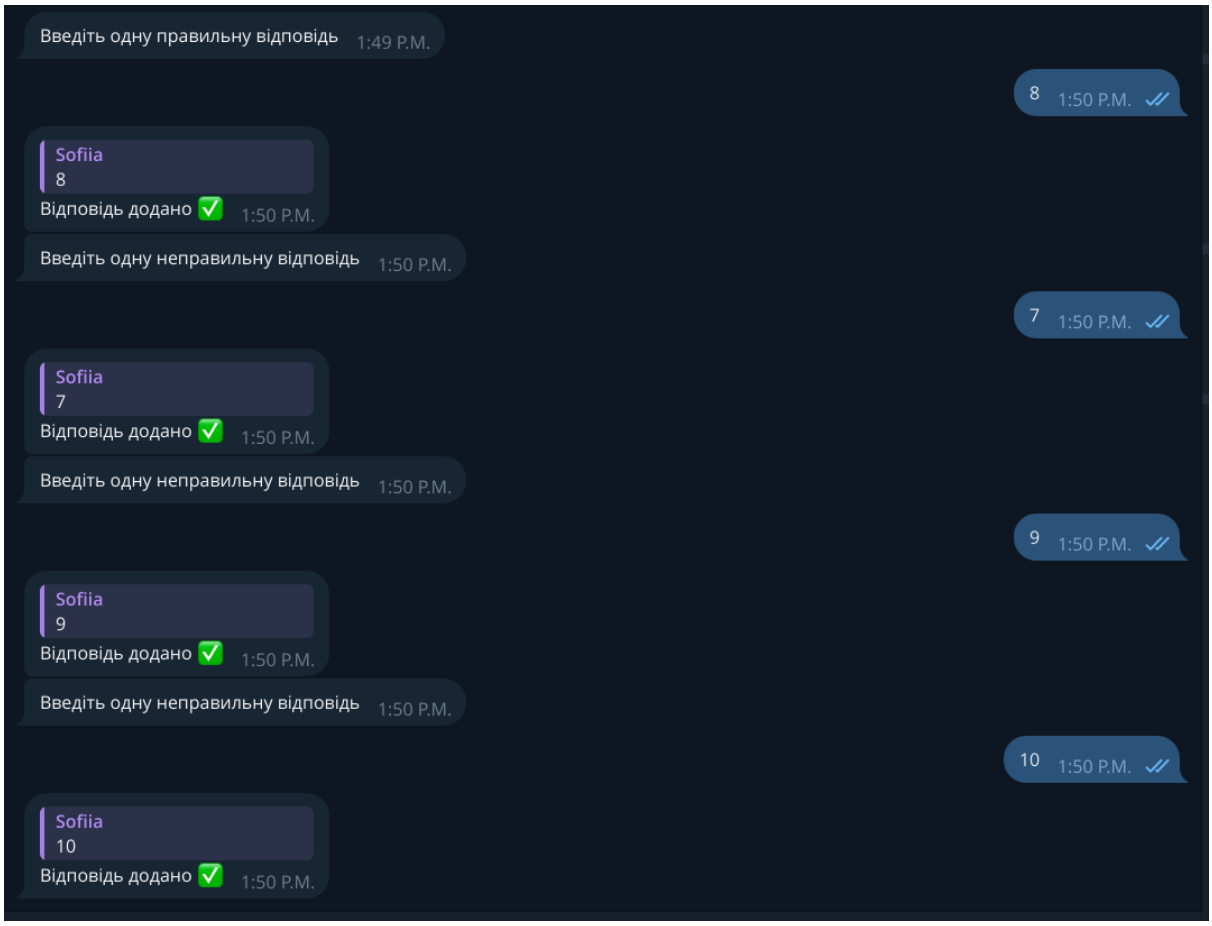

Рис.3.3.10 - Додавання відповідей на питання

Переглянемо тепер відповіді на дане питання натиснувши на кнопку “Переглянути відповіді” (рис.3.3.11).

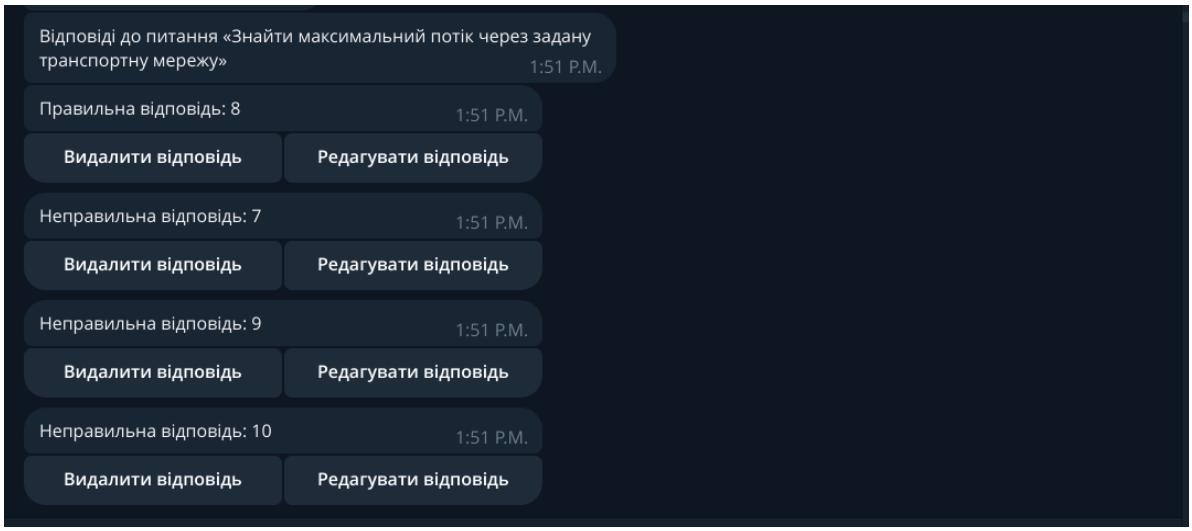

Рис.3.3.11 - Перелік відповідей на питання

Як бачимо, відповіді можна теж редагувати та видаляти.

Додамо ще кілька питань аби створити базу питань, на яких потім буде навчатися бот адаптивності.

Наприклад, коли до питання ми не маємо фото, то можемо пропусти крок його додавання натиснувши на відповідну кнопку (рис.3.3.12).

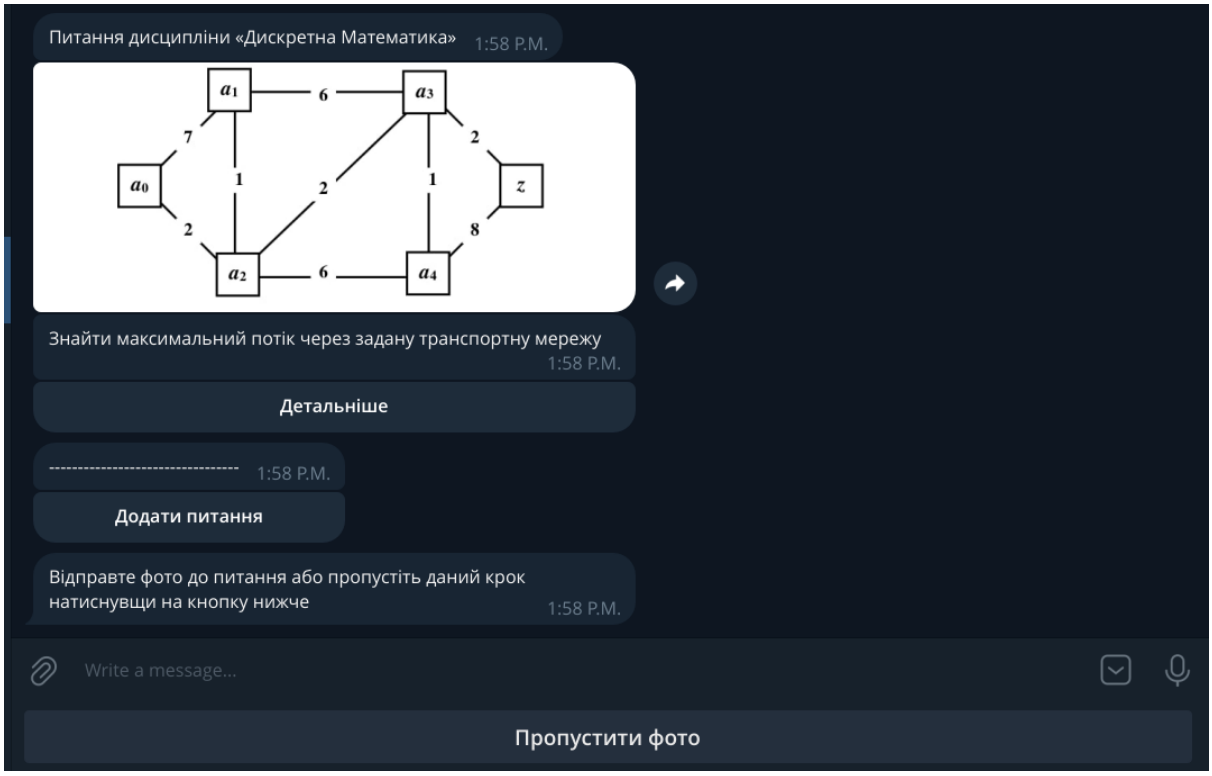

Рис.3.3.12 - Пропуск фото

Було створено п’ять питань до дисципліни “Дискретна математика” (рис.3.3.13).

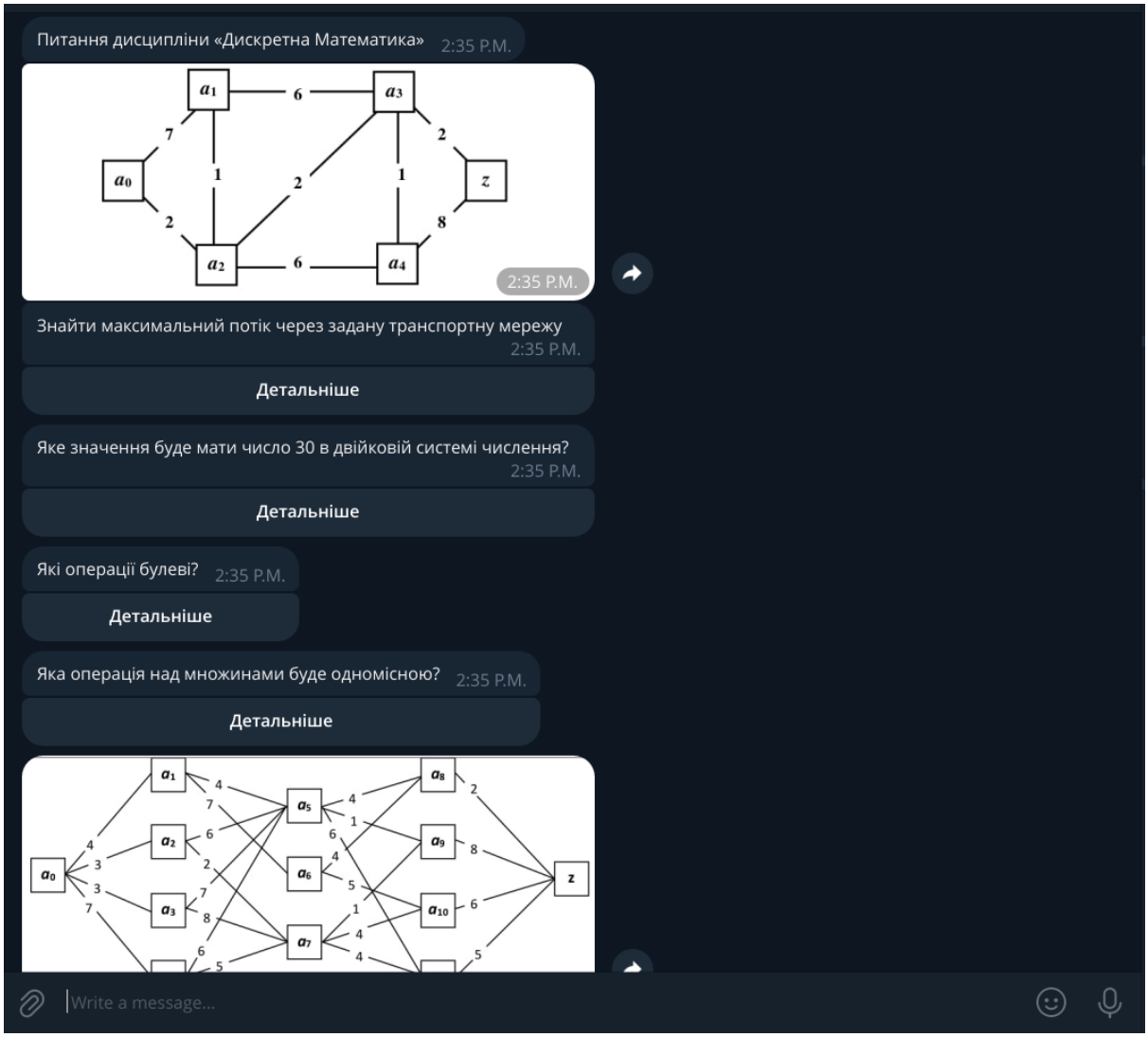

Рис. 3.3.13 - Питання до дисципліни “Дискретна математика”

Перейдемо до інтерфейсу користувача. Для цього вводимо /start (рис. 3.3.14).

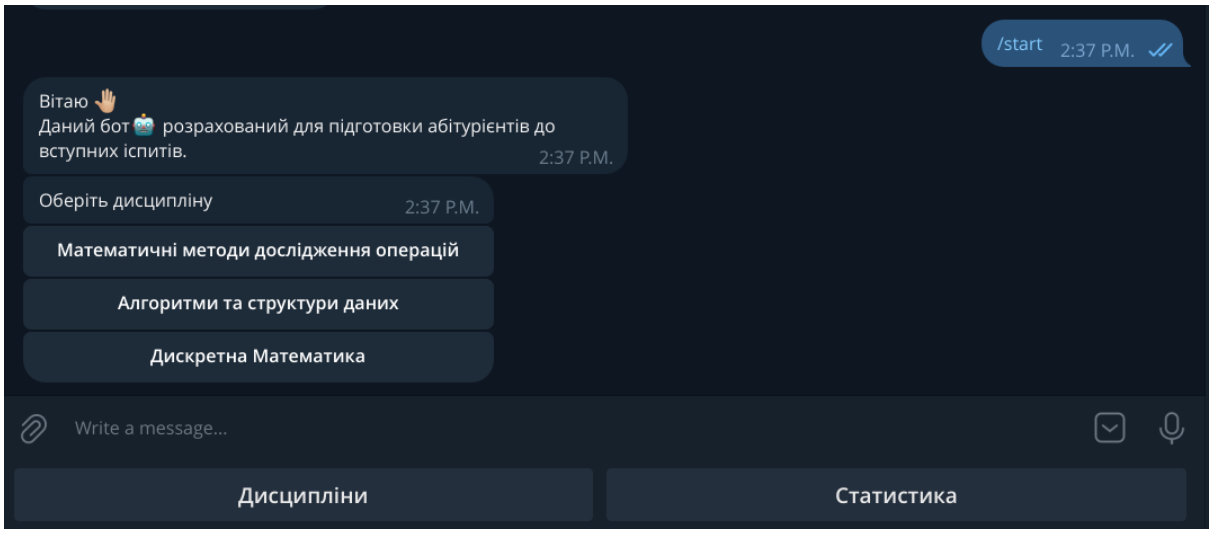

Рис. 3.3.14 - Вхід до інтерфейсу користувача

Ми можемо обрати дисципліну або переглянути статистику. Так як користувач новий, то статистика ще недоступна, адже не було дано ще ніяких відповідей.

Оберемо дисципліну “Дискретна Математика” (рис. 3.3.15).

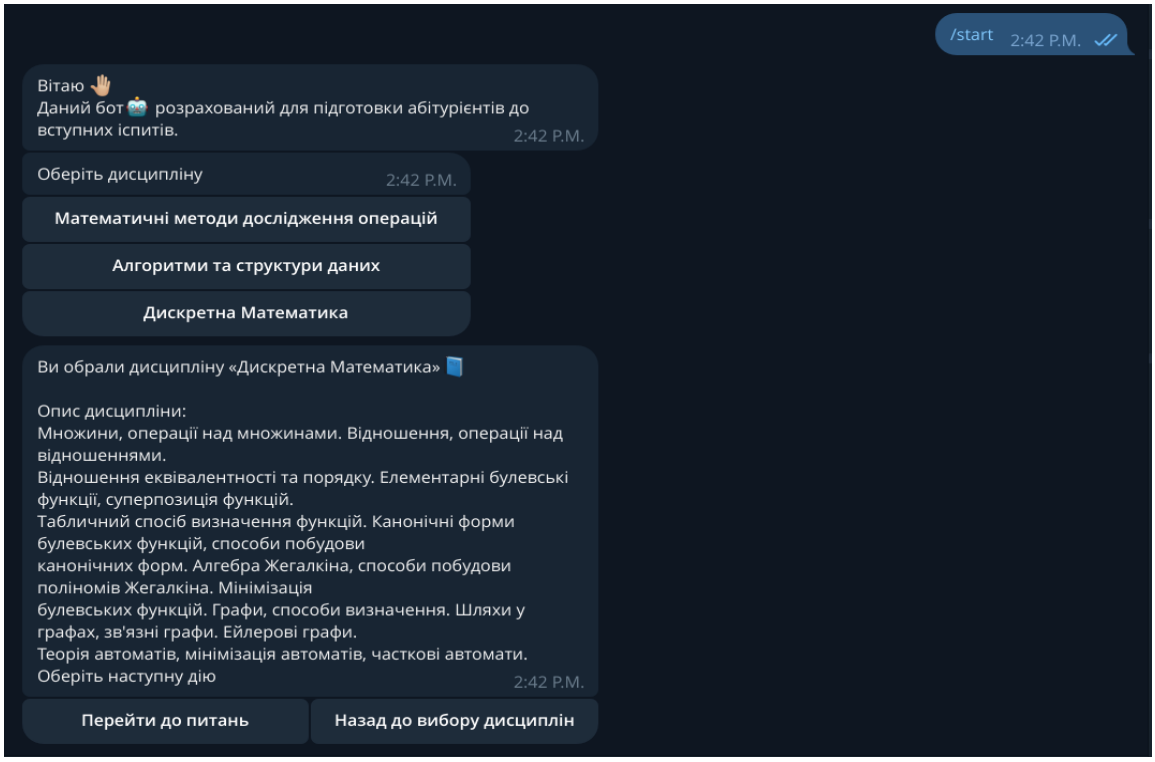

Рис.3.3.15 - Вибір дисципліни “Дискретна Математика”

Ми бачимо назву дисципліни, її опис. Можемо обрати дві дії:

- перейти до питань;
- повернутися назад до вибору дисциплін.

Перейдемо до питань. В нас з’являється перше питання (рис.3.3.16).

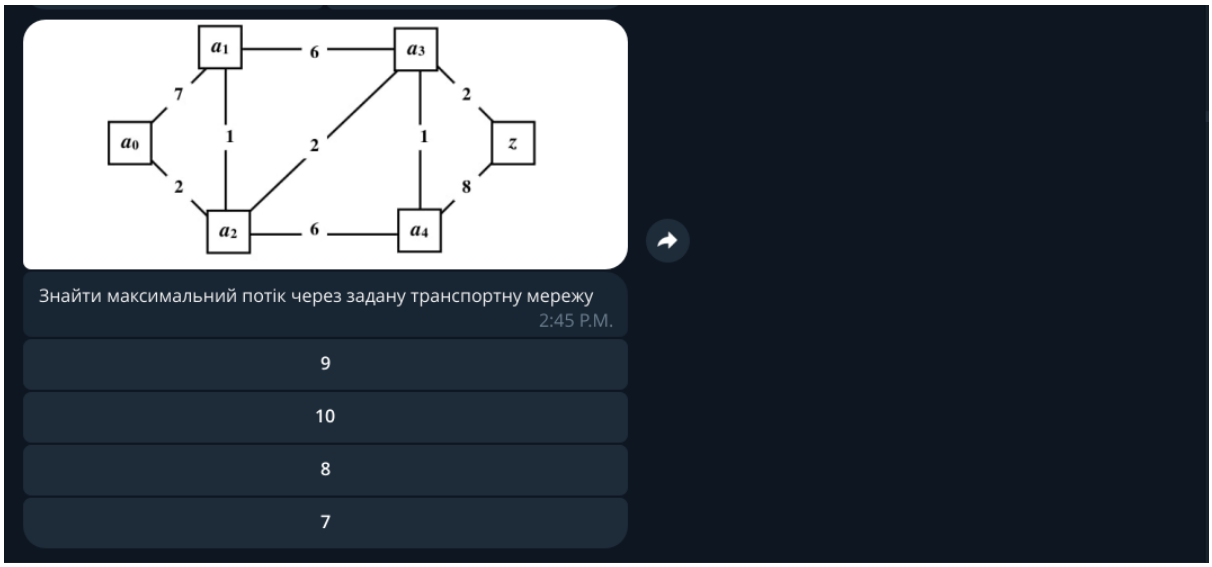

Рис. 3.3.16 - Перше питання дисципліни “Дискретна математика”

Коли маємо нового користувача, для якого система ще не навчена, то спочатку він отримає перші питання рандомно, поки система не навчиться визначати складність питання для даного користувача.

Даємо, наприклад, неправильну відповідь на питання (рис.3.3.17).

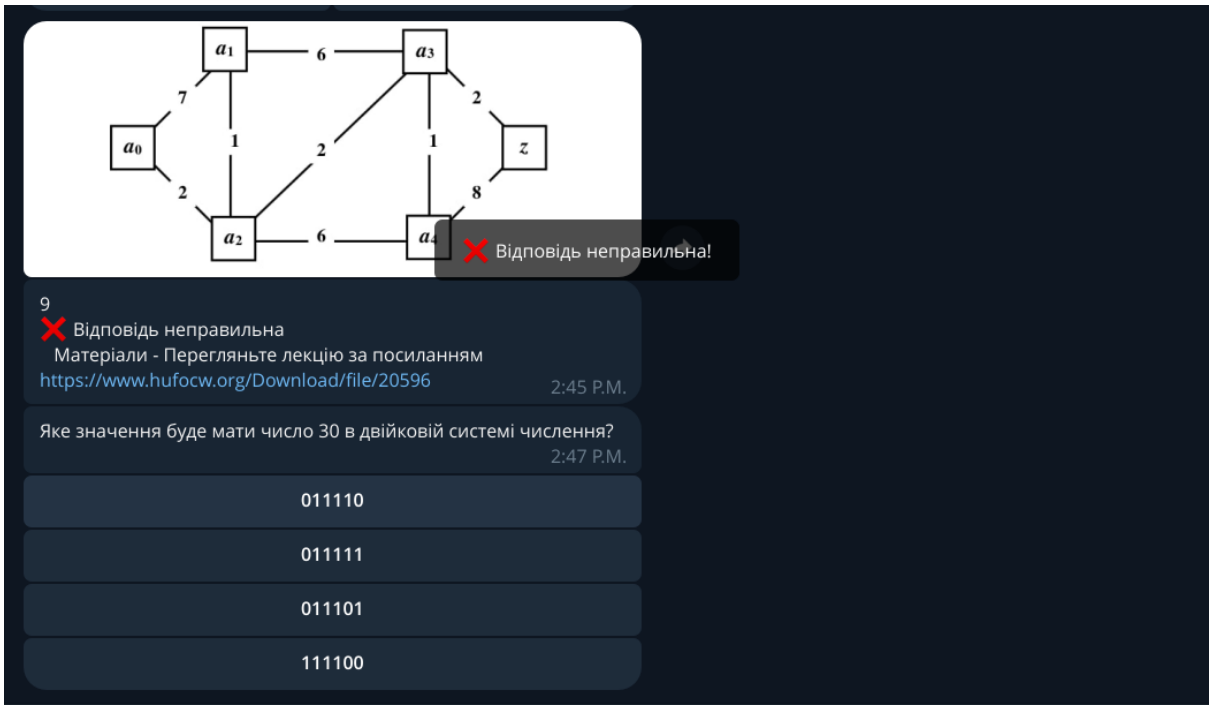

Рис.3.3.18 - Випадок неправильної відповіді

Отримуємо далі поки що питання обрані випадково (рис.3.3.19).

Рис.3.3.19 - Наступне випадково обране питання

Давши кілька правильних відповідей на питання, система починає розуміти які питання є складними та які простішими для даного користувача. І далі починає працювати за розробленою схемою адаптивного тестування.

В будь-який момент можна переглянути статистику по дисципліні натиснувши на кнопку “Статистика” (рис. 3.3.20).

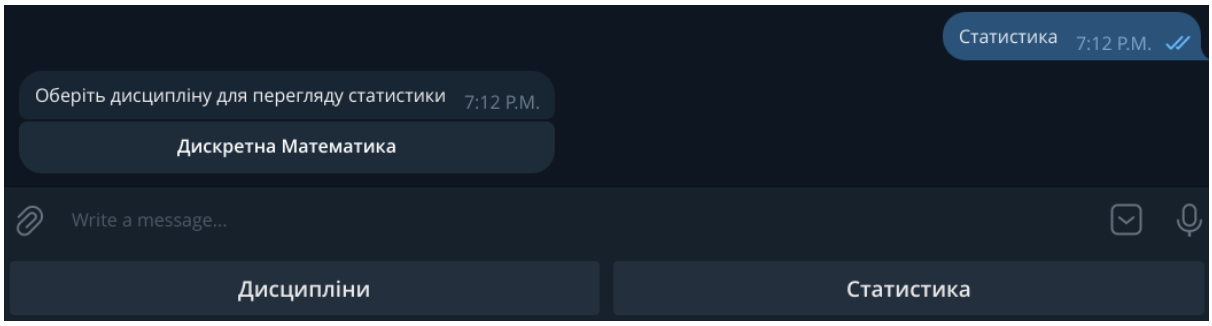

Рис.3.3.20 - Виклик статистики

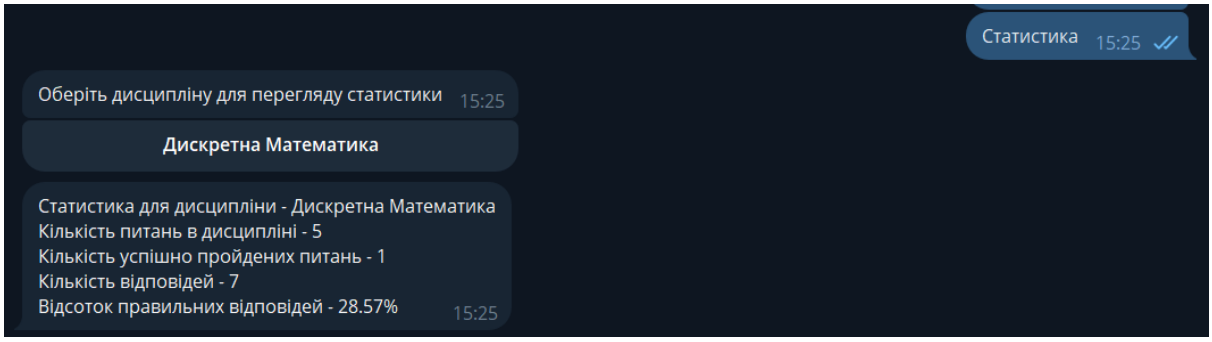

Рис.3.3.21 – Відображення статистики по дисципліни «Дискретна Математика»

Як бачимо в статистиці відображається:

- назва самої дисципліни;
- кількість питань в дисципліні;
- кількість успішно пройдених питань;
- кількість пройдених питань загалом;
- відсоток правильних відповідей від усієї кількості відповідей.

Статистика може слугувати користувачеві як відслідковування своїх результатів та мотивацією аби відсоток правильних відповідей був ближче до 100.

ВИСНОВОК

У ході виконання магістерської кваліфікаційної роботи було розроблено інформаційну технологію для адаптивної підготовки абітурієнтів до вступних іспитів. Крім того, проведено аналіз існуючих аналогів, оцінено їхній функціонал. На основі цього був проведений аналіз того, що відсутнє в існуючих рішеннях, і визначено способи їх поліпшення. Розглянуто доступні інструменти для реалізації ідеї. В результаті цього визначено, що найбільш оптимальним варіантом є створення Telegram-боту, спрямованого на задоволення потреб користувачів.

Однією з ключових переваг розробленого бота є його доступність та простота використання. Інтерфейс користувача відзначається високою інтуїтивністю, адже для взаємодії з ним не потрібно реєструватися – достатньо просто запустити бота. Адміністраторська частина вигідно вирізняється мінімалістичним дизайном та високою зручністю в експлуатації, надаючи повний необхідний функціонал для організації тестування.

У результаті виконаної роботи можна зазначити, що створена технологія є актуальною в контексті сучасних подій в Україні, зокрема в умовах дистанційного навчання. Ця інноваційна система може бути успішно використана не лише для підготовки абітурієнтів до вступних іспитів, але й для проведення тестувань учнями та студентами як ефективного засобу поточної перевірки їхніх знань.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Adaptive learning: Helpful to the flipped classroomin the online environment of COVID? / Renee M. Clark, Autar K. Kaw, Rafael Braga Gomes.
2. Adaptive learning in a numerical methods course for engineers: Evaluation in blended and flipped classrooms / Clark R.M., Kaw A. / Computer Applications in Engineering Education. 2020. Vol. 28, № 1.
3. Research Trends in Adaptive Online Learning: Systematic Literature Review (2011–2020) / Ochukut S.A. / Technology, Knowledge and Learning. 2023. Vol. 28, № 2.
4. Telegram Bot Usage as An Instructional Design Method / Nur T. / International Journal of Accounting, Finance and Business. 2022. Vol. 7, № 42.
5. Evolution of chatbots for smart assistance / Aggarwal V. / International Journal of Innovative Technology and Exploring Engineering. 2019. Vol. 8, № 10.
6. Чат-боти у навчанні: 7 ідей для їх використання [Electronic resource] // https://learnlifelong.net/chat-boty-u-navchanni-7-idej-dlya-yih-vykory/.
7. Modeling of the adaptive system of individualization and personalization of future specialists' professional training in the conditions of blended learning [Electronic resource] // https://journals.indexcopernicus.com/search/article?articleId=3633300.
8. Adaptive and Intelligent Technologies for Web-based Eduction [Electronic resource] // https://www.researchgate.net/publication/220633613_Adaptive_and_Intelligent_Technologies_for_Web-based_Eduction.
9. Українською мовою програмування: перший телеграм бот [Electronic resource] // https://dou.ua/forums/topic/42413/.
10. Python Tutorial [Electronic resource] // https://www.w3schools.com/python/default.asp.
11. JavaScript [Electronic resource] // https://developer.mozilla.org/en-US/docs/Web/JavaScript.
12. Java Introduction [Electronic resource] // https://www.w3schools.com/java/java_intro.asp.
13. 25 Best Libraries for Writing Telegram Chatbots in 2020 [Electronic resource] // https://medium.com/botsclub/25-best-libraries-for-writing-telegram-chatbots-in-2020-4e717a0e4493.

14. What Is a Framework? [Electronic resource] // https://www.codecademy.com/resources/blog/what-is-a-framework/.
15. aiogram 3.2.0 documentation [Electronic resource] // https://docs.aiogram.dev/en/latest/.
16. Aiogram v3.2.0 [Electronic resource] // https://snyk.io/advisor/python/aiogram.
17. Aiogram 3.2.0 [Electronic resource] // https://pypi.org/project/aiogram/.
18. Welcome to pyTelegramBotAPI's documentation! [Electronic resource] // https://pytba.readthedocs.io/en/latest/.
19. pyTelegramBotAPI 4.14.0 [Electronic resource] // https://pypi.org/project/pyTelegramBotAPI/.
20. pyTelegramBotAPI v4.14.0 [Electronic resource] // https://snyk.io/advisor/python/pytelegrambotapi.
21. Python-telegram-bot 20.7 [Electronic resource] // https://pypi.org/project/python-telegram-bot/].
22. Python-telegram-bot v20.7 [Electronic resource] // https://snyk.io/advisor/python/python-telegram-bot.
23. Telethon 1.33.1 [Electronic resource] // https://pypi.org/project/Telethon/.
24. Telethon v1.33.1 [Electronic resource] // https://snyk.io/advisor/python/telethon.
25. Pyrogram 2.0.106 [Electronic resource] // https://pypi.org/project/Pyrogram/.
26. Pyrogram v2.0.106 [Electronic resource] // https://snyk.io/advisor/python/pyrogram.
27. What Is SQLite? [Electronic resource] // https://www.sqlite.org/index.html.
28. Best Databases for Python in 2023 – A Comprehensive Comparison [Electronic resource] // https://www.ropstam.com/best-databases-for-python/.
29. ЩО TAKE MYSQL [Electronic resource] // https://freehost.com.ua/ukr/faq/wiki/chto-takoe-mysql/.
30. MySQL [Electronic resource] // https://www.mysql.com/.
31. PostgreSQL: The World's Most Advanced Open Source Relational Database [Electronic resource] // https://www.postgresql.org/.
32. PostgreSQL vs MySQL: The Critical Differences [Electronic resource] // https://www.integrate.io/blog/postgresql-vs-mysql-which-one-is-better-for-your-use-case/.

33. MongoDB: The Developer Data Platform [Electronic resource] // https://www.mongodb.com/.
34. Introduction to Redis [Electronic resource] // https://redis.io/docs/about/]. .
35. From BotFather to “Hello World” [Electronic resource] // https://core.telegram.org/bots/tutorial.

ДОДАТОК А

```
from sqlalchemy import select, func, and_, literal_column
from sqlalchemy.orm import aliased

from bot.db.models import Question, Discipline, UserAnswer, Answer
from bot.types import UserDisciplineStatistics


async def get_question(session, discipline_id: int, user_id) -> Optional[Question]:
  # check last tries
  last_user_tries_query = (
     select(
        Answer.is_correct
     )
     .select_from(Answer)
     .join(UserAnswer, UserAnswer.answer_id == Answer.id)
     .join(Question, Question.id == Answer.question_id)
     .where(UserAnswer.user_id == user_id, Question.discipline_id == discipline_id)
     .order_by(UserAnswer.created_at.desc())
     .limit(3)
  )
  last_user_tries = (await session.execute(last_user_tries_query)).scalars().all()
  if len(last_user_tries) < 3:
     query = (
        select(Question)
        .select_from(Question)
        .filter_by(discipline_id=discipline_id)
        .order_by(func.random())
```

```
            .limit(1)
        )
        row = await session.execute(query)
        return row.scalars().one_or_none()

    correct_answer: UserAnswer = aliased(UserAnswer, name="correct_answer")
    un_correct_answer: UserAnswer = aliased(UserAnswer, name="un_correct_answer")
    filters = [
        UserAnswer.user_id == user_id,
        Discipline.id == discipline_id,
    ]

    query = (
        select(
            Question,

        ).filter(*filters)
        .select_from(Question)
        .join(Answer, Answer.question_id == Question.id)
        .join(
            correct_answer,
            and_(correct_answer.answer_id == Answer.id, Answer.is_correct.is_(True)),
            isouter=True,
        )
        .join(
            un_correct_answer,
            and_(
                un_correct_answer.answer_id == Answer.id, Answer.is_correct.is_(False)
            ),
```

```
            isouter=True,
        ).group_by(Question.id).limit(1).order_by(func.random())
    )

    correct_answer_of_last_tries = sum(
        [answer for answer in last_user_tries if answer is True]
    )

    if correct_answer_of_last_tries == 0:
        query = query.having(((100 * func.greatest(1, func.count(correct_answer.id))) / func.greatest(1, (
                func.count(correct_answer.id) + func.count(un_correct_answer.id)))) <= 70)
    else:
        query = query.having(((100 * func.greatest(1, func.count(correct_answer.id))) / func.greatest(1, (
                func.count(correct_answer.id) + func.count(un_correct_answer.id)))) > 70)

    question = (await session.execute(query)).scalars().one_or_none()
    if  question is None:
        query = (
            select(Question)
            .select_from(Question)
            .filter_by(discipline_id=discipline_id)
            .order_by(func.random())
            .limit(1)
        )
        row = await session.execute(query)
        return row.scalars().one_or_none()
    return question
```

```
async def get_user_active_disciplines(session, user_id):
    query = (
        select(Discipline)
        .select_from(Discipline)
        .join(Question, Question.discipline_id == Discipline.id)
        .join(Answer, Answer.question_id == Question.id)
        .join(UserAnswer, UserAnswer.answer_id == Answer.id)
        .where(UserAnswer.user_id == user_id)
    ).group_by(
        Discipline.id,
        Discipline.name,
        Discipline.created_at,
        Discipline.short_description,
    )

    row = await session.execute(query)
    return row.scalars().all()



async def get_user_stats_for_discipline(session, user_id: int, discipline_id: int):
    # кількість правильних відповідей всередині дисципліни від юзера (2 однакові правильні відповіді на 1 питання рахуються як 2)
    right_answers_count_query = (
        select(func.count(UserAnswer.id))
        .select_from(UserAnswer)
        .join(Answer, UserAnswer.answer_id == Answer.id)
        .join(Question, Question.id == Answer.question_id)
        .join(Discipline, Discipline.id == Question.discipline_id)
        .where(
```

```
            UserAnswer.user_id == user_id,
            Discipline.id == discipline_id,
            Answer.is_correct.is_(True),
        )
    )
    right_answers_count = (
        (await session.execute(right_answers_count_query)).scalars().one()
    )

    # кількість всіх відповідей юзера всередині дисципліни
    all_answers_count_query = (
        select(func.count(UserAnswer.id))
        .select_from(UserAnswer)
        .join(Answer, UserAnswer.answer_id == Answer.id)
        .join(Question, Question.id == Answer.question_id)
        .join(Discipline, Discipline.id == Question.discipline_id)
        .where(UserAnswer.user_id == user_id, Discipline.id == discipline_id)
    )
    all_answers_count = (await session.execute(all_answers_count_query)).scalars().one()

    # кількість правильно пройдених питань юзера всередині дисципліни (2 однакові правильні відповіді на 1 питання рахуються як 1 )
    distinct_right_answers_count_query = (
        select(func.count(func.distinct(Question.id)))
        .select_from(UserAnswer)
        .join(Answer, UserAnswer.answer_id == Answer.id)
        .join(Question, Question.id == Answer.question_id)
        .join(Discipline, Discipline.id == Question.discipline_id)
        .where(
```

```
        UserAnswer.user_id == user_id,
        Discipline.id == discipline_id,
        Answer.is_correct.is_(True),
    )
    .group_by(Question.id)
)
distinct_right_answers_count = (
    (await session.execute(distinct_right_answers_count_query)).scalars().one()
)

# кількість питань всередині дисципліни
discipline_questions_count_query = (
    select(func.count(Question.id))
    .select_from(Question)
    .join(Discipline, Question.discipline_id == Discipline.id)
    .where(Discipline.id == discipline_id)
)
discipline_questions_count = (
    (await session.execute(discipline_questions_count_query)).scalars().one()
)

return UserDisciplineStatistics(
    discipline_question_count=discipline_questions_count,
    discipline_right_answer_count=distinct_right_answers_count,
    user_all_answers_count=all_answers_count,
    user_right_answers_count=right_answers_count,
)
```