ДЕРЖАВНИЙ ТОРГОВЕЛЬНО-ЕКОНОМІЧНИЙ

УНІВЕРСИТЕТ

Кафедра комп'ютерних наук та інформаційних систем

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка чат-боту для тестування знань здобувачів освіти»

Студента 4 курсу, 10 групи,

спеціальності
122 «Комп'ютерні науки» _____ *підпис студента* Черненко Дмитро Валерійович

Науковий керівник
Доктор педагогічних наук, доцент _____ *підпис керівника* Підгорна Тетяна Володимирівна

Гарант освітньої програми
кандидат технічних наук, доцент _____ *підпис керівника* Демідов Павло Георгійович

Київ 2023

Державний торговельно-економічний університет

Факультет інформаційних технологій
Кафедра комп'ютерних наук та інформаційних систем
Спеціальність 122 «Комп'ютерні науки»

Затверджую
Зав. кафедри _____Пурський О.І.
«12» грудня 2022р.

Завдання
на випускну кваліфікаційну роботу (проект) студенту
Черненку Дмитру Валерійовичу
(прізвище, ім'я, по батькові)

1. Тема випускної кваліфікаційної роботи (проекту)
«Розробка чат-боту для тестування знань здобувачів освіти»
Затверджена наказом ректора від *«09 » грудня 2022 р. № 3332*
2. Строк здачі студентом закінченої роботи *30 травня2023 року*
3. Цільова установка та вихідні дані до роботи
Мета роботи: розробка чат-боту для тестування знань здобувачів освіти.
Об'єкт дослідження: процес розробки чат-боту для тестування знань здобувачів освіти.
Предмет дослідження: чат-бот тестування знань здобувачів освіти.
4. Перелік графічного матеріалу
_____
_____

Консультанти по роботі із зазначенням розділів, за якими здійснюється консультування:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Підгорна Т.В.	15.12.2022 р.	15.12.2022 р.
2	Підгорна Т.В.	15.12.2022 р.	15.12.2022 р.
3	Підгорна Т.В.	15.12.2022 р.	15.12.2022 р.

5. Зміст випускної кваліфікаційної роботи (перелік питань за кожним розділом)

ВСТУП

РОЗДІЛ 1. СПЕЦИФІКА ІНСТУРМЕНТІВ ДЛЯ ТЕСТУВАННЯ ЗНАНЬ ЗДОБУВАЧІВ ОСВІТИ

1.1. Особливості процесу тестування знань здобувачів освіти

1.2. Огляд існуючих інструментів для тестування знань здобувачів освіти

1.3. Чат-бот як інструмент для тестування знань здобувачів освіти

РОЗДІЛ 2. ПРОЦЕС РОЗРОБКИ ЧАТ-БОТА ДЛЯ ТЕСТУВАННЯ ЗНАНЬ ЗДОБУВАЧІВ ОСВІТИ

2.1. Вибір засобів програмної реалізації чат-бота

2.2. Створення та налагодження програми

РОЗДІЛ 3. ПРАКТИЧНА РЕАЛІЗАЦІЯ ЧАТ-БОТУ ДЛЯ ТЕСТУВАННЯ ЗНАНЬ

3.1. Архітектура чат-боту

3.2. Програмна реалізація чат-боту для тестування

3.3. Функціональні можливості чат-боту для тестування знань здобувачів освіти

ВИСНОВКИ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ДОДАТОК

6. Календарний план виконання роботи

№ Пор.	Назва етапів випускної кваліфікаційної роботи	Строк виконання етапів роботи	
		За планом	фактично
1	2	3	4
1	*Вибір теми випускної кваліфікаційної роботи*	*04.10.2022*	*04.10.2022*
2	*Розробка та затвердження завдання на випускну кваліфікаційну роботу*	*15.12.2022*	*15.12.2022*
3	*Вступ*	*03.02.2023*	*03.02.2023*
4	*РОЗДІЛ 1. Специфіка інстурментів для тестування знань здобувачів освіти*	*28.02.2023*	*28.02.2023*
5	*РОЗДІЛ 2. Процес розробки чат-бота для тестування знань здобувачів освіти*	*06.04.2023*	*06.04.2023*
6	*РОЗДІЛ 3. Практична реалізація чат-боту для тестування знань*	*12.05.2023*	*12.05.2023*
7	*Висновки*	*15.05.2023*	*15.05.2023*
8	*Здача випускної кваліфікаційної роботи на кафедру науковому керівнику*	*30.05.2023*	*30.05.2023*
9	*Попередній захист випускної кваліфікаційної роботи*	*31.05.2023 - 01.06.2023*	*31.05.2023 - 01.06.2023*
11	*Виправлення зауважень, зовнішнє рецензування випускної кваліфікаційної роботи*	*02.06.2023*	*02.06.2023*
12	*Представлення готової зшитої випускної кваліфікаційної роботи на кафедру*	*05.06.2023*	*05.06.2023*
13	*Публічний захист випускної кваліфікаційної роботи*	*За розкладом роботи ЕК*	

7. Дата видачі завдання «15» грудня 2022 р.

Керівник випускної кваліфікаційної роботи (проекту) Підгорна Т.В.
(прізвище, ініціали, підпис)

Гарант освітньої програми Демідов П.Г.
(прізвище, ініціали, підпис)

Завдання прийняв студент-дипломник Черненко Д. В.
(прізвище, ініціали, підпис)

9. Відгук керівника випускної кваліфікаційної роботи (проекту)

Керівник випускної кваліфікаційної роботи (проекту)
_____ 30.05.2023 р.
(підпис, дата)

10. Висновок про випускну кваліфікаційну роботу

Випускна кваліфікаційна робота студента Черненка Д. В.
(прізвище, ініціали)
може бути допущена до захисту в екзаменаційній комісії.

Гарант освітньої програми Демідов П.Г.
(підпис, прізвище, ініціали)

Завідувач кафедри Пурський О.І.
(підпис, прізвище, ініціали)

«____»_____2023 р.

Анотація

Черненко Д. В. Розробка чат-боту для тестування знань здобувачів освіти. У випускній кваліфікаційній роботі проведено аналіз актуальності використання чат-боту для тестування знань здобувачів. Визначено переваги та недоліки використання тестування для оцінювання. Проведено аналіз існуючих програмних продуктів для тестування здобувачів. І в результаті розроблено Telegram чат-бота, який дає змогу викладачам створювати, редагувати та видаляти тести, а здобувачам – проходити та отримувати результати з тестів.

Ключові слова: чат-бот, тестування знань, розробка, модель, інтерфейс.

Annotation

Chernenko D. V. Development of a chatbot for testing the knowledge of education seekers. In the graduation thesis, an analysis of the relevance of using a chatbot for testing the knowledge of applicants was carried out. Additionally, the advantages and disadvantages of using testing for assessment are identified. The thesis also features an analysis of existing software products for testing education seekers. As a result, a Telegram chatbot was developed, enabling teachers to create, edit, and delete tests, while applicants can take tests and receive their results.

Keywords: chatbot, knowledge testing, development, model, interface.

ЗМІСТ

ВСТУП

Актуальність роботи. Сфера освіти, як і велика кількість інших сфер, розвивається та діджиталізуєтья, оскільки комп'ютерні системи є набагато більш швидкісним та простим способом вирішення проблем. Однією з таких проблем є оцінювання знань здобувачів освіти. Проблемі тестування знань присвячено багато науково-педагогічних досліджень. Однак все ширший і глибший процес цифровізації освіти, сприяє вирішенню цієї проблеми на іншому рівні.

Одним з найпопулярніших способів використання комп'ютерних систем для вирішення задач – є чат-боти. Дивлячись на те, наскільки в сучасному світі легко та ефективно використовувати їх, дивно що на практиці є лише одиниці систем оцінювання які використовують дану технологію. Причиною цьому може бути достатня новизна в використанні такої методики навчання, тестування та оцінювання в нашій країні.

Мета і завдання дослідження. Метою дослідження є розробка чат-боту для тестування знань здобувачів освіти. Для досягнення поставленої мети треба вирішити такі **завдання**:

1. На основі аналізу науково-педагогічної літератури визначити засоби для здійснення тестування знань здобувачів освіти.
2. Розглянути використання чат-бота в освітньому процесі.
3. Створити діаграми та визначити з функції чат-боту.
4. Розробити чат-бота з чіткими інструкціями для використання.

Система повинна відповідати стандартам та використовувати рішення, актуальні на момент написання випускної кваліфікаційної роботи.

Об'єкт дослідження: процес розробки чат-боту для тестування знань здобувачів освіти.

Предмет дослідження: чат-бот для тестування знань здобувачів освіти.

Методи дослідження: Теоретичною основою дослідження є загальнонауковий аналітичний метод і дослідження аналітичних джерел оцінки ролі чат-ботів. Інформаційну базу дослідження становлять документація та офіційні джерела, онлайн ресурси та форуми для опису функціональності, синтаксису та елементів чат-ботів. Для практичного вирішення поставлених задач використовувалися методи:

- загальнонауковий аналітичний метод (розділ 1);
- методи математичного моделювання для комплексної оцінки структури чат-ботів (питання 2.1, 2.2);
- методи теорії БД для формування інформаційно-логічної моделі предметної області та БД;
- методи алгоритмічного програмування, для програмної реалізації чат-боту для оцінювання знань здобувачів освіти.

Практичне значення. Отримані результати можуть бути використані в навчальному процесі для здійснення тестування знань здобувачів освіти.

Структура та обсяг випускної кваліфікаційної роботи. Випускна кваліфікаційна робота складається із вступу, трьох розділів, висновків, списку використаних джерел із 22 найменувань, з програмний код поданий в додатках А – Є, 38 сторінок основного тексту, 27 рисунків і 16 таблиці.

РОЗДІЛ 1. СПЕЦИФІКА ІНСТУРМЕНТІВ ДЛЯ ТЕСТУВАННЯ ЗНАНЬ ЗДОБУВАЧІВ ОСВІТИ

1.1. Особливості процесу тестування знань здобувачів освіти

Тестування – це методика оцінювання знань здобувачів освіти шляхом представлення запитань з варіантами можливих відповідей. Головна ціль такої методики полягає в контролі знань з пройдених тем для виявлення прогалин, або проблематичних моментів. Це надає можливість здобувачам освіти оцінити свої можливості з засвоєння вивченого матеріалу, і практичного застосувати здобуті навички та умінь. Також тестування використовують не тільки метою контролю вивченого матеріалу, а ще з метою вибудовування знань, умінь та навичок для засвоєння наступних, нових тем та матеріалів[1].

Тестування як форма об'єктивного контролю та діагностики набуває все більшого поширення в освітньому процесі та інших сферах суспільного життя. Нині воно використовується з метою оцінки результатів навчання на певному освітньому рів-ні або на окремих його етапах, під час іспитів на професійну придатність, при відборі кандидатів у Національну поліцію тощо. Тестування широко використовується у країнах Європи. Воно має певні переваги над іншими формами контролю знань, але має й недоліки. Доцільність використання тестування повинна бути обґрунтована в кожній конкретній ситуації, адже це може мати вплив на об'єктивність оцінювання знань і вмінь суб'єкта контроль-ного заходу [18].

Як відомо, за таксономією Блума, розумові процеси людини умовно можна поділити на низького рівня, до них відносяться знання, розуміння, використання, і високого рівня (оцінювання, синтез, аналіз). В педагогічній

літературі визначено[2,3], що для перевірки відповідних розумових процесів доцільно застосовувати такі типи тестових завдань:

Таблиця 1.1

Типи тестових завдань які перевіряють розумові процеси

Розумові процеси	Типи тестових завдань
Оцінювання	- встановлення правильної послідовності; - вибір кількох правильних відповідей;
Синтез	- доповнити схему (вибір правильної відповіді);
Аналіз	- встановлення правильної послідовності і відповідності;
Використання	- вибір кількох правильних відповідей із запропонованих; - встановлення правильної відповідності і послідовності;
Розуміння	- встановлення правильної послідовності і відповідності; - вибір кількох правильних відповідей із запропонованих;
Знання	- вибір однієї або кількох правильних відповідей.

Тести можуть набувати різних форм. Серед головних форм трестування можна виділити дві ключові: тести відкритої форми та тести закритої форми[4]. Тести відкритої форми передбачають собою питання, які спрямовані на відтворення вивченого здобувачами матеріалу у відкритому форматі. Прикладом такого тестового завдання можуть слугувати питання які потребують від суб'єкта висловити свою думку по темі, завершити твердження у письмовому варіанті, зазначити ключові моменти з пройденого матеріалу. Тести закритої форми передбачають собою питання

з множиною відповідей, серед яких є правильна. Закриті тести можуть пропонувати лише одну правильну відповідь, декілька правильних відповідей, завдання на відновлення та встановлення відповідності, завдання на відтворення правильної послідовності.

Перевірка знань у формі тестових завдань показала себе як дуже ефективна методика, як серед великої кількості досліджень, так і серед використання викладачами її на практиці[1,4-7]. Серед факторів які впливають на такий результат можна відзначити:

- Простота та зрозумілість завдання у вигляді тесту;
- наявність об’єктивно правильної відповіді;
- ефективне визначення прогалини в знаннях;
- миттєве отримання результату;
- схожість на гру з системою балів.

Перший фактор, простота та зрозумілість, дозволяє здобувачам відчути спокій, що призводить до більш сфокусованого та коректного виконання роботи, та отримання позитивного результату. Другий фактор, наявність правильної відповіді, надає об’єктивності в оцінені знань, і дає здобувачам впевненість та мотивацію знайти правильну відповідь, навіть якщо з першого разу не вийшло. Третій фактор, ефективність в визначені прогалин, дозволяє здобувачам побачити над чим їм потрібно попрацювати, для повного засвоєння пройденого матеріалу. Четвертий фактор, миттєве отримання результату, дозволяє швидко та ефективно зрозуміти здобувачу, що потрібно доопрацювати. П’ятий фактор, схожість на гру, дозволяє здобувачам зацікавитися в процесі навчання та мотивує досягати максимального результату.

Серед негативних сторін використання тестів треба відзначити, що їх ефективність залежить від навичок та вміння дотримуватися методологічних вимог людиною котра складає завдання[4].

Здобування знань та оцінка рівня освіти є важливими етапами навчального процесу. Особливості процесу тестування знань здобувачів освіти вимагають постійного вдосконалення методів та інструментів для забезпечення точності, об'єктивності та ефективності оцінювання. Одним із напрямків вдосконалення є використання сучасних інструментів, таких як чат-боти, які забезпечують інтерактивну та зручну форму взаємодії зі студентами.

1.2. Огляд існуючих інструментів для тестування знань здобувачів освіти

Google Форми - один із самих доступних та широко вживаних інструментів для побудови опитувань та тестувань. Він являє собою безкоштовний онлайн сервіс з простим та зрозумілим drag-and-drop інтерфейсом, до якого можна отримати доступ з веб-браузеру на персональному комп’ютері або на мобільних девайсах(Рис. 1.1).

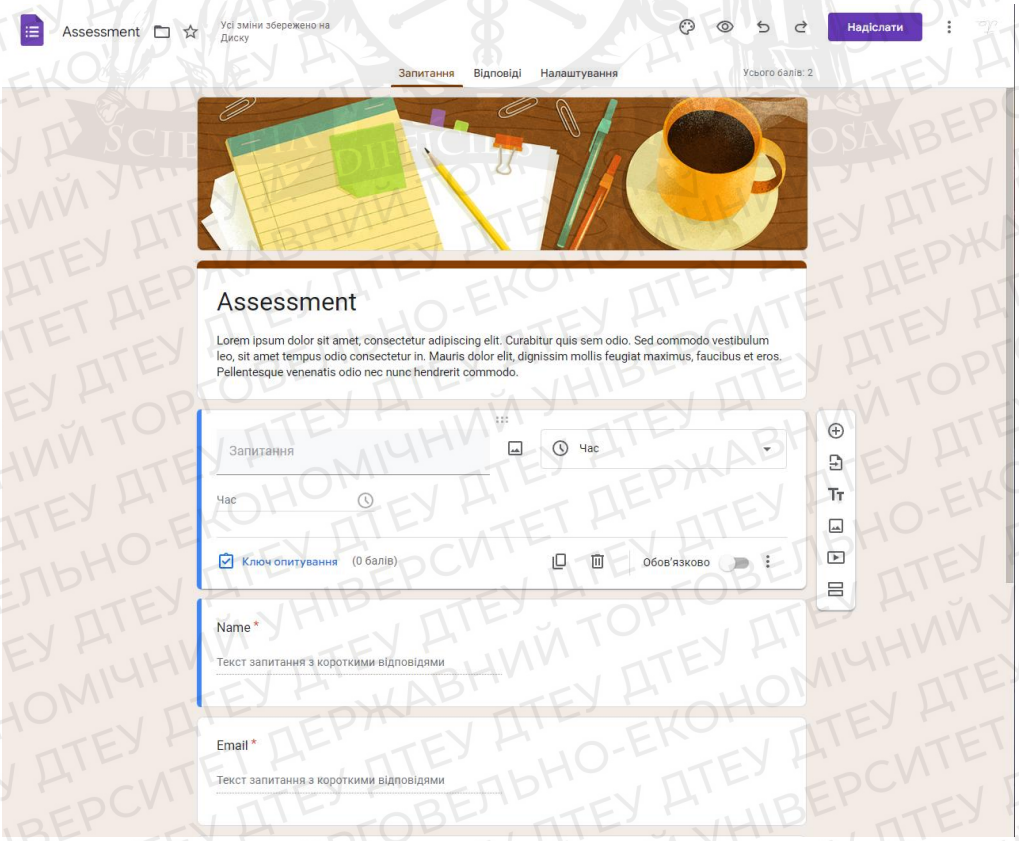

Рис. 1.1. Веб інтерфейс для створення опитування в Google Форми.

Цей сервіс використовується для вирішення широкого спектру проблем, серед яких можна виділити:

- Створення тестів для здобувачів освіти
- Планування конференцій або зустрічей
- Проведення опитувань для подальшого аналізу
- Отримання зворотного зв’язку

Google Форми пропонує для створення велику кількість питань з різноманітними варіантами відповідей[8]. Серед типів відповідей присутні: коротка відповідь, абзац, з варіантами відповіді, прапорці, спадний список, завантаження файлу, лінійна шкала, таблиця з варіантами відповіді, сітка прапорців, дата, час.

Також присутня можливість додавати зображення та відео до запитань, перетворення форми для опитувань в форму для тестів, створювати різні секції в опитуванні, та тематизувати форму вибираючи бажаний розмір та тип шрифту, вибирати зображення або колір для бекграунду. Після створення форми її можна відправити через електрону пошту, або відправити створене посилання.

Головним недоліком даної системи це доступ до тесту може отримати будь хто має посилання, що може призвести до не доброчесно проходження.

Quizlet – система навчання яка базується на вивчені матеріалу за допомогою карточок з питаннями та правильними відповідями. Сервіс доступний в веб-браузері на персональних комп’ютерах та мобільних девайсах, і як додаток на мобільних девайсах (Рис. 1.2).

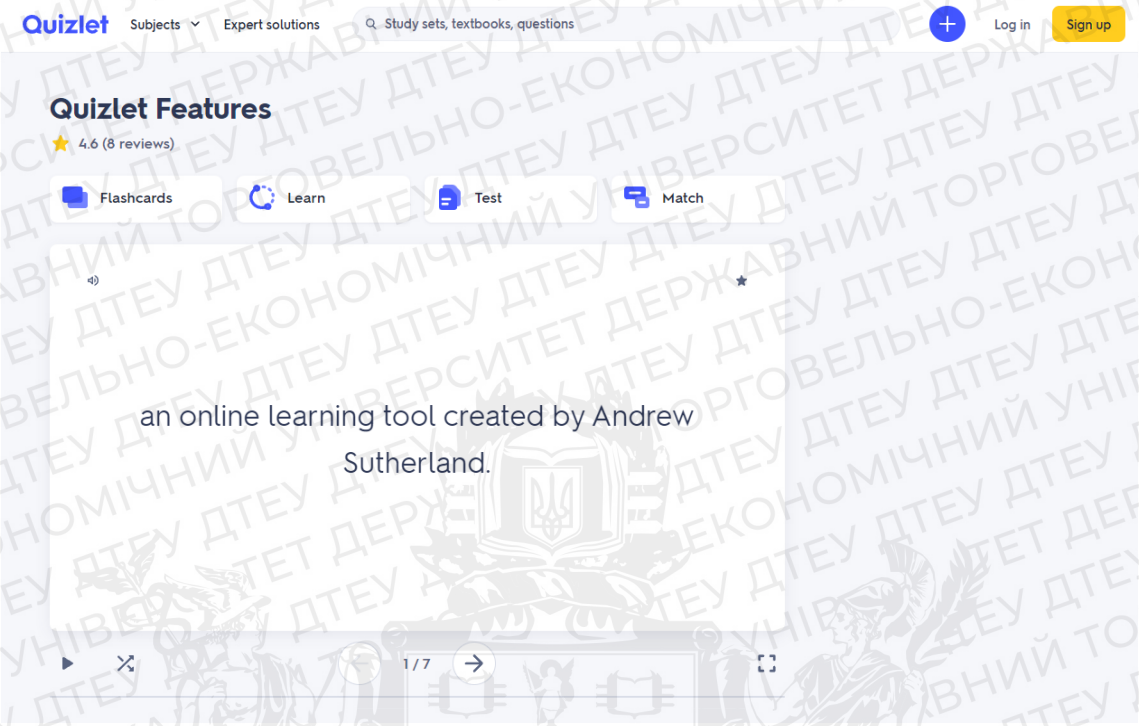

Рис. 1.2. Веб інтерфейс для навчання в Quizlet.

Ця система дозволяє створювати класи з різних тем для навчання та закріплення матеріалу[9]. Quizlet пропонує наступні режими вивчення матеріалу:

- Картки – В цьому режимі користувачу демонструються карти. На одній стороні картки написано питання на яке треба запам’ятати відповідь, а на іншій стороні – відповідь.
- Заучування – В цьому режимі користувачеві демонструються запитаннями з можливими варіантами відповідей. Якщо користувач не зміг відповісти на запитання правильно з першого разу, то йому буде продемонстровано правильну відповідь. Після чого запитання пізніше буде повторюватися, до тих пір поки користувач не зможе з впевненістю відповідати на нього.
- Тест – В цьому режимі користувачеві буде представлено тестове завдання з наступними варіантами відповідей: правильно/неправильно, одна правильна відповідь, підбір відповіді на

відповідність, письмова відповідь. Після проходження тесту буде продемонстрований результат з кількістю правильних та не правильних відповідей.

- Підбір – В цьому режимі на екрані демонструються блоки з питаннями та відповідями які розкидані на ігровому полі. Ці блоки можна перемішувати та перетягувати по ігровому полю. Користувачеві необхідно перетягувати правильні блоки відповідей на блоки запитань, або навпаки – блоки запитань на блоки відповідей.
- В живу – це режим в якому вчитель розподіляє клас на команди для проходження тесту в якості гри. Вчитель самостійно вибирає на що студенти повинні дати відповідь. Кожна команда повинна надати свою відповідь, і та команда в котрої буде більше всього правильних відповідей перемагає.

Всі тестові завдання Quizlet генерує автоматично залежно від введених вчителем запитань та відповідей.

ProProfs – система з великим набором інструментів для навчання в професійному середовищі. Сервіс для створення тестів доступний у веб-браузері на персональних комп’ютерах та мобільних девайсах.

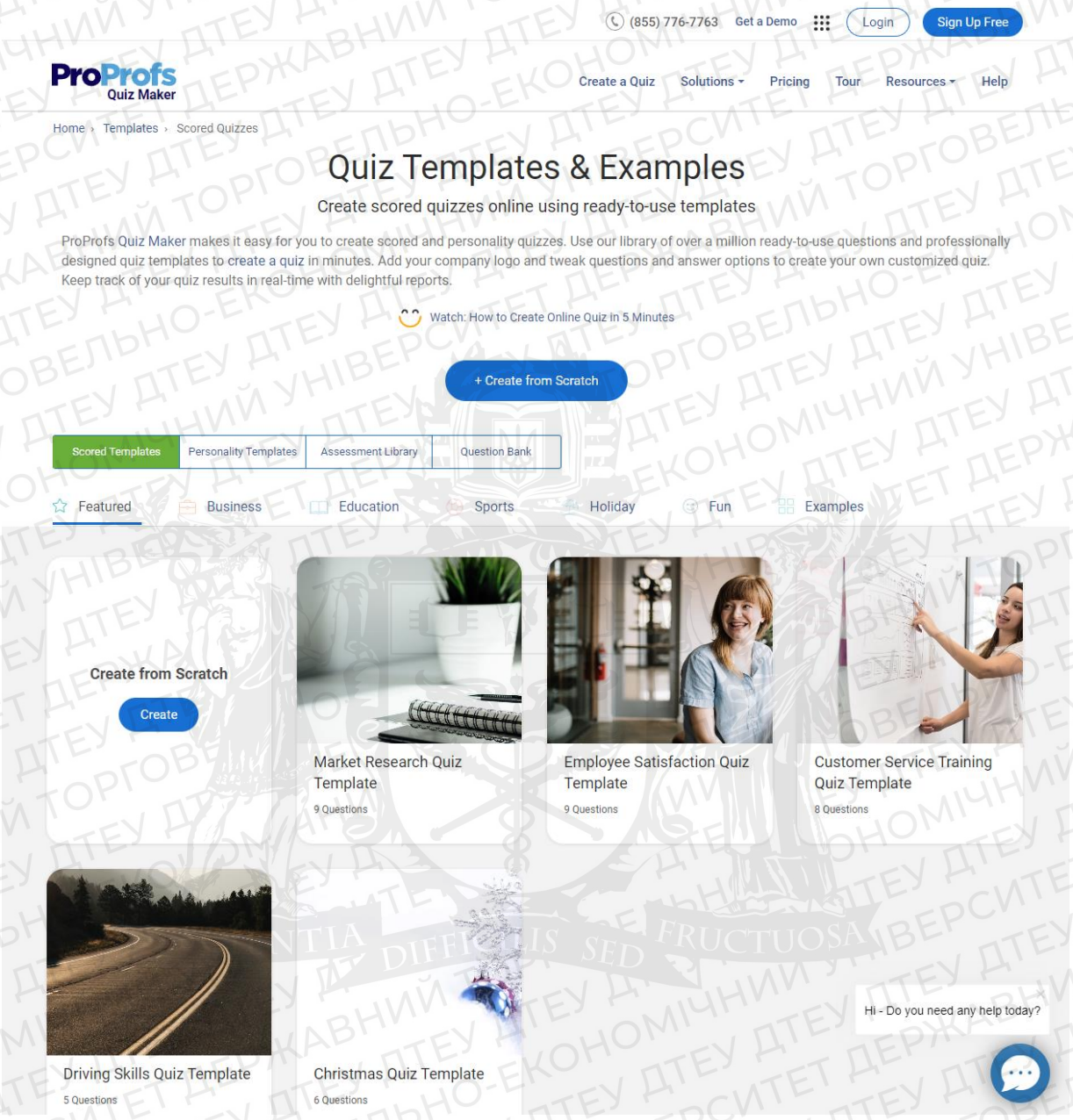

Рис. 1.3. Веб інтерфейс ресурсу ProProfs

ProProfs пропонує великий спектр інструментів та шаблонів для створення тестових завдань та опитувань[10]. Серед можливих варіантів питань налічуються: питання на одну відповідь, на декілька правильних відповідей, вірно/невірно, вписати пропущене, абзац, встановити відповідність, вибрати точку на картинці, спадний список, вписати правильну відповідь, встановити послідовність, перетягнути правильну відповідь, дати відповідь після прослуховування відео або аудіо,

завантажити файл. Також в системі присутня можливість налаштовувати зовнішній вигляд тесту, назву та положення запитань.

Головним недоліком цієї системи є те, що в базовій версії відкриті не всі можливості для створення тестів та є обмеження по кількості створенню тестів в місяць. А також створений тест буде публічним, що може дозволити здобувачам не доброчесно проходити його.

Чат-бот «Цифрограм. Твоя кібергігієна» - чат-бот який направлений поширення та навчання цифровій грамотності (Рис. 1.4). Цей чат-бот був розроблений спеціалістами з Міністерства цифрової трансформації, Vodafone Україна, Міжнародної фундації виборчих систем (IFES) та Координатор проектів ОБСЄ[11].

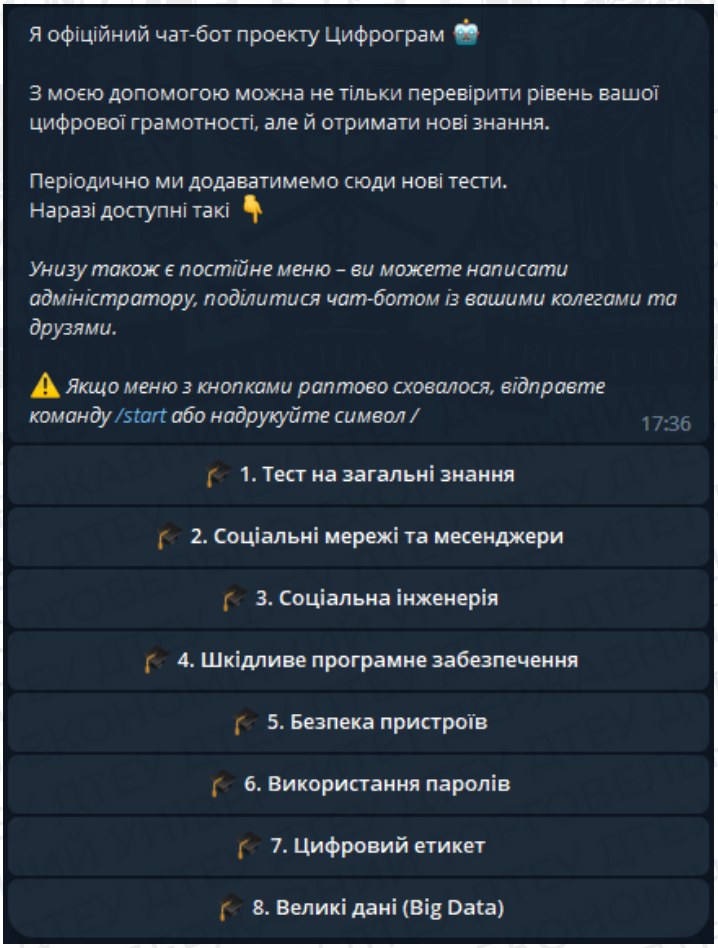

Рис. 1.4. Інтерфейс чат-боту «Цифрограм»

В чат-боті присутні можливості для проходження та ознайомлення тестів з наступних тем: тест на загальні знання, соціальні мережі та месенджери, соціальна інженерія, шкідливе програмне забезпечення, безпека пристроїв, використання паролів, цифровий етикет, великі дані.

Головним недоліком даного чат-боту є обмеження лише одним типом тестів - вибором правильної відповіді.

Огляд існуючих інструментів для тестування знань здобувачів освіти показав, що незважаючи на наявність різноманітних підходів та програмних рішень, існують виклики та обмеження, які потребують подальшого розвитку та вдосконалення. У цьому контексті, чат-боти виступають як потенційно перспективний інструмент для тестування знань здобувачів освіти.

1.3. Чат-бот як інструмент для тестування знань здобувачів освіти

Пріоритетним завданням сучасної системи вищої освіти є створення ефективних електронних освітніх ресурсів, інноваційний характер яких дав би змогу максимально використовувати дидактичний потенціал у навчальному процесі. Головними напрямами створення електронних освітніх ресурсів є мультимедійний супровід пояснення нового матеріалу, проведення віртуальних лабораторних робіт, обробка експериментальних даних. Одним із напрямів створення електронних освітніх ресурсів (разом з вищезазначеними) є контроль рівня знань із використанням тестових завдань [20].

Чат-бот – це система, яка використовує інтерфейс чат/месенджера, для взаємодії з користувачем для досягнення поставлених задач. Технологія використання чат-боту у повсякденному житті не є новинкою для багатьох людей, і так само для здобувачів освіт. Вона використовується багатьма бізнесами та установами для взаємодії з клієнтами для автоматизації та пришвидшення процесу.

Використання чат-боту в навчанні є достатньо новою ідею та практикою в навчальних установах. Але вже існують дослідження та досвіди використання данної технології[12,13].

- Вони допомагають студентам навчатися, обмінюючись інформацією за допомогою повідомлень, схожих на звичайні чати, але згрупованих разом у форматі лекції, створюючи враження присутності та спілкування з іншими в «реальному часі».
- Вони можуть надавати навчальні матеріали в різних форматах, включаючи текстові повідомлення, зображення, відео, аудіо та документи.
- Вони можуть оцінити рівень розуміння кожного студента та продовжити лекції на основі їх успіху.
- Тестування навчальних матеріалів можна проводити в різних режимах, наприклад, на іспитах або на тренажерах.
- Вони доступні 24/7, що означає, що студенти можуть використовувати їх у будь-який час.

Ці досвіди використання чат-боту для створення та проведення тестування демонструють:

- Низьку ціну використання;
- кращу взаємодію з користувачами;
- підвищену ефективність створення та проходження тестів;
- створення більш комфортної атмосфери для здобувачів.

За оцінками Global Market Insights, до 2024 року ринок чат-ботів становитиме понад 1.3 мільярда доларів. Таким чином, чат-боти будуть домінувати в бізнес-комунікаціях. NLP-боти використовують аналіз настроїв і прогнозну аналітику для розумної інтерпретації розмов і намірів запиту. Незважаючи на впровадження технологією штучного інтелекту на підприємствах, найкращі практики чат-ботів все ще важко застосувати.

Завдяки машинному навчанню (ML), штучному інтелекту (AI) і обробці природної мови (NLP) компанії розробляють чат-ботів, які неможливо відрізнити від людей, щоб відповідати очікуванням споживачів і усунути недоліки (NLP) [19].

Статистика найбільш часто вирішуваних завдань чат-ботами наведена на рисунку 1.5:

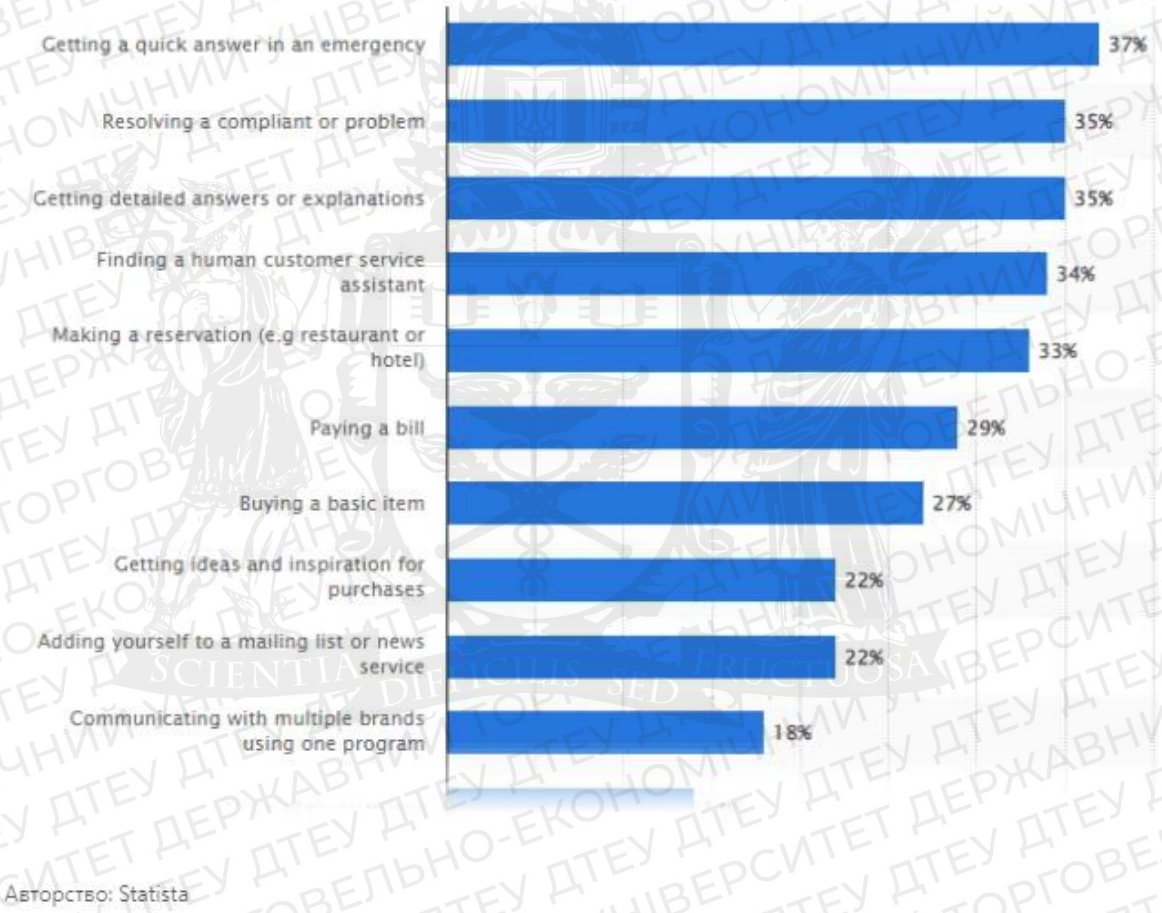

Рис. 1.5. Рейтинг найбільш часто вирішуваних завдань чат-ботами

РОЗДІЛ 2. ПРОЦЕС РОЗРОБКИ ЧАТ-БОТА ДЛЯ ТЕСТУВАННЯ ЗНАНЬ ЗДОБУВАЧІВ ОСВІТИ

2.1. Вибір засобів програмної реалізації чат-бота

Приблизний вік здобувача варіюється в залежності від рівня освіти та типу навчального закладу. Кожен заклад в свою чергу надає освітні послуги і намагається створювати для цього найбільш комфортні умови та варіанти. Для підвищення ефективності та мобільності в оцінюванні знань здобувачів освіти, чат-боти стануть новим і, в перспективі, широковживаним способом для взаємодії між викладачами та учнями.

Серед найпопулярніших засобів програмної реалізації чат-боту, які використовуються на момент написання випускної кваліфікаційної роботи, є:

- Платформи розробки чат-ботів: деякі платформи спрощують розробку чат-ботів, надаючи інструменти та функції для створення, налаштування та розгортання чат-ботів. Наприклад, Dialogflow, Botpress, Chatfuel, ManyChat. Зазвичай вони мають вбудовану інтеграцію з такими популярними месенджерами, як Facebook Messenger, Telegram, Slack тощо.
- Мови програмування та бібліотеки: використання мов програмування, такі як Python, JavaScript, Java або PHP, і відповідні бібліотеки для створення власних чат-ботів.
- API Telegram Bot: API Telegram Bot надає набір методів для взаємодії з чат-ботами Telegram, включаючи отримання повідомлень, надсилання відповідей, керування клавіатурою, обробку медіафайлів тощо.
- Фреймворк для розробки чат-ботів

Telegram є одним із найбільших месенджером у світі з 400 мільйонами активних користувачів щомісяця, і 1,5 мільйона нових користувачів, що реєструється щоденно. Він швидко розвивається і це означає, що багато людей уже користуються Telegram і що кількість користувачів швидко зростає[14].

Як зазначають медіа ресурси[15], люди частіше обирають Telegram, оскільки він являє собою платформа з функціями і месенджера, і соціальної мережі. В умовному рейтингу довіри від Київського міжнародний інститут соціології, Telegram-канали обирають 35% опитаних. Менше довіряють телебаченню (33%), YouTube-каналам (25%), Viber (20%) та Facebook (20%).

Оскільки молоді люди як мінімум декілька сотень разів на день користуються месенджером Telegram для спілкування і вже знають принцип його функціонування, то використовувати цей месенджер в освітніх цілях дозволить зручно, доступно, інтерактивно, без витрати великих зусиль та з миттєвим фідбеком підвищувати рівень знань учнів, та полегшувати роботу для викладачів [21, 22].

Telegram – одна з перших систем, яка відкрила свій API для створення ботів. Користувачеві Telegram зрозуміло, коли спілкування буде вести бот, тому що є вимога системи – ім'я бота, має закінчуватися словом «bot». Управляти ботами в Telegram можна через спеціальний бот, батька всіх ботів BotFather – асистент, який служить для створення ботів, управління ними, і їх описом. Основні функції бота: відповіді на питання, переклад, коментування, пошук інформації, відтворювати відео і аудіо, вбудовування в бота ігор, та багато іншого. Боти в Telegram можна «Додати в групу» (Add To Group), або ними «Поділитися» (Share), налаштувати зворотній зв'язок з користувачами у формі емодзі, кнопочок, лайків, ще один із способів спілкування – текст, рідше голосові команди [20].

Telegram пропонує гнучкий , легкий і безкоштовний фреймворк для

створення та розповсюдження чат-ботів в месенджері. Таким чином це надає можливість користувачам, які вже мають аккаунт в телеграмі, одразу почати взаємодію[16].

Окрім того, що Telegram користується великою популярністю через зручність його використовування, він також надає широкий спектр функцій, які дозволяють створювати чат-ботів, яким можна призначати найбільш об'ємні та корисні завдання:

- Заощадження часу і грошей на обслуговуванні користувачів;
- за допомогою чат-бота можна відповісти за секунди, а не за дні;
- чат-бот доступний 24/7 у всьому світі;
- люди відкриті до використання чат-ботів, якщо вони отримують хорошу серсів.

У випадку оцінювання знань, можна створити чат-бот, який швидко надасть бланк з завданням і так само швидко покаже правильні відповіді.

Для написання чат-боту була обрана мова програмування Python, оскільки вона гнучка, проста в синтаксисі та є наявні бібліотеки в Telegram API для написання чат-ботів на цій мові.

Для зберігання та контролю інформації в базі даних, було обрано використовувати систему керування базами даних MongoDB. MongoDB — це документо-орієнтована база даних, яка була створена для спрощення розробки систем, додатків та масштабування. Ця система дозволяє створювати схеми даних, які підтримують шаблони частого доступу, що дозволяє пришвидши працездатність. Також присутня можливість іпортувати дані з CSV та JSON фалів, і проводити операції з створення, читання, оновлення та видалення інформації з бази даних.[17].

Після вивчення різних програмних реалізацій чат-ботів важливо перейти до процесу створення та налагодження самої програми. Розглянемо деталі розробки чат-бота, починаючи з правильного вибору технології та

платформи та розглядаючи важливі етапи реалізації програми. Це дозволить вам перейти від теоретичних міркувань до практичної реалізації та пройти процес створення чат-бота.

2.2. Створення та налагодження програми

Для створення чат-боту для оцінювання знань здобувачів освіти, було вирішено використовувати деякі вбудовані функції, які є в месенджері Telegram, що дали змогу полегшити розробку чат-боту. Окрім вбудованих функцій також використовувалися сторонні програмні засоби та методи. До них відносяться:

- API Telegram Bot. API Telegram Bot є основним інструментом для взаємодії з чат-ботами Telegram. Він надає можливість отримувати повідомлення від користувачів, надсилати відповіді та виконувати різні операції.
- Бібліотеки або фреймворки програмування: для реалізації логіки тестування, обробки повідомлень і взаємодії з базою даних.
- База даних для зберігання такої інформації, як запитання, відповіді, результати тестів тощо. Популярними базами даних, які можна використовувати, є MySQL, PostgreSQL або SQLite.
- Алгоритми обробки запитань і відповідей: Для кожного типу тесту (тест на відповідність, 1 правильна відповідь, кілька правильних відповідей, правильний порядок) розробляється відповідний алгоритм обробки відповідей користувачів і визначення правильності відповідей.
- Автентифікація та авторизація: механізми автентифікації та авторизації використовуватимуться для забезпечення безпеки та обмеження доступу до функцій чат-бота. У даному випадку, буде реалізовано поділ на викладачів та учнів.

- Взаємодія із зовнішніми службами. залежно від потреб проекту можуть знадобитися засоби взаємодії із зовнішніми службами, наприклад системи зберігання результатів тестів, системи сповіщень або інші додаткові інтеграції.
- Розгортання та розміщення на сервері хостингу або використати платформу для розміщення бота.
- Тестування та налагодження. Важливим етапом є тестування та налагодження чат-бота для виявлення та виправлення можливих помилок і недоліків. Для цього можна використовувати різні методи тестування, включаючи модульне тестування, інтеграційне тестування та тестування за допомогою тестових сценаріїв.

Система чат-боту повинна мати наступні сценарії роботи та взаємодії:

Таблиця 2.1

Сценарій авторизації викладача

Дійові особи	Викладач, Система
Ціль	Авторизуватися
Успішний сценарій 1. Система відображає інтерфейс для вибору типу користувача. 2. Викладач вибирає пункт авторизуватися. 3. Система відображає інтерфейс для авторизації. 4. Викладач вводить свої данні правильно. 5. Система демонструє повідомлення про успішне створення тесту.	
Результат	Викладач успішно авторизується в системі.
Розширення	
а.	Неправильно введенні дані для авторизації. Видається повідомлення про некоректні дані.

Таблиця 2.2

Сценарій авторизації здобувача

Дійові особи	Здобувач, Система
Ціль	Авторизуватися
Успішний сценарій 1. Система відображає інтерфейс для вибору типу користувача. 2. Здобувач вибирає пункт авторизуватися. 3. Система відображає інтерфейс для авторизації.	

4. Здобувач вводить свої данні правильно. 5. Система демонструє повідомлення про успішне створення тесту.	
Результат	Здобувач успішно авторизується в системі.
Розширення	
а.	Неправильно введенні дані для авторизації. Видається повідомлення про некоректні дані.

Таблиця 2.3

Сценарій створення нового тесту

Дійові особи	Викладач, Система
Ціль	Створити тестове завдання
Передумова	Викладач повинен авторизуватися в системі
Успішний сценарій 1. Викладач вибирає пункт створити тестове завдання. 2. Система приймає запит та відображає інтерфейс для створення нового тесту. 3. Викладач заповнює тестові питання. 4. Система зберігає створенні питання і відображає інтерфейс для призначення тесту групам з датою та часом доступу до тесту та стоком здачі. 5. Викладач заповнює відповідну інформацію. 6. Система демонструє повідомлення про успішне створення тесту.	
Результат	Викладач успішно створив нове тестове завдання.

Таблиця 2.4

Сценарій проходження тесту

Дійові особи	Здобувач, Система
Ціль	Пройти тест
Передумова	Здобувач повинен авторизуватися, викладач повинен створити тест
Успішний сценарій 1. Здобувач вибирає дисципліну з якої бажає пройти тест. 2. Система приймає запит та відображає інтерфейс з вибором тестів для проходження. 3. Здобувач обирає потрібний тест для проходження. 4. Система успішно відображає питання до вибраного тесту. 5. Здобувач успішно проходить тест. 6. Система демонструє результат виконаного завдання.	
Результат	Здобувач успішно переглянув оцінки здобувачів освіти.
Розширення	
а.	Здобувач пройшов тест не успішно. Система відобразила питання на які здобувач відповів неправильно.

Також було розроблено UML діаграми концепції чат-бот із взаємодією викладача і студента (Рис. 2.1 – 2.3).

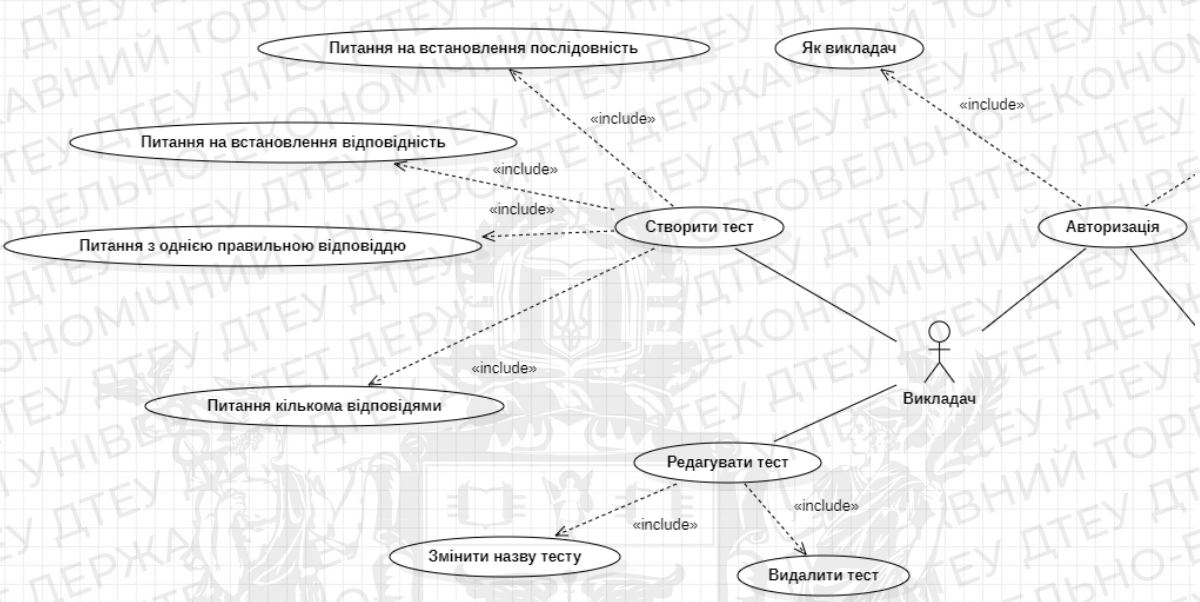

Рис. 2.1. Use-case діаграма взаємодії викладача та системи

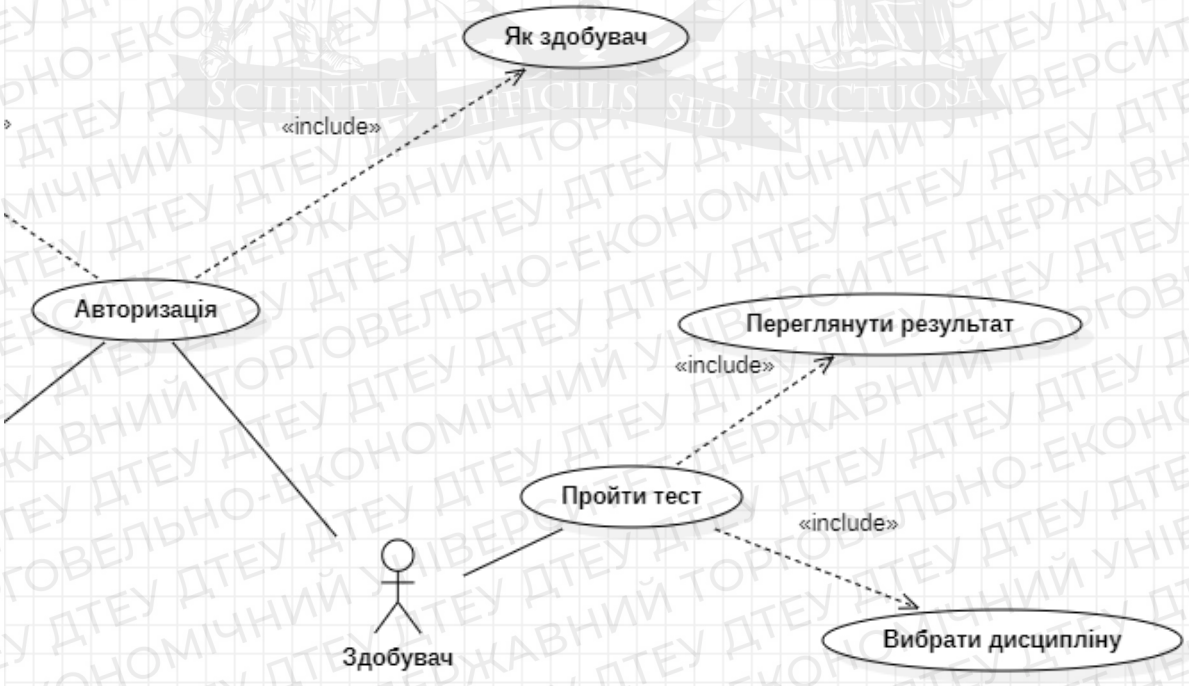

Рис. 2.2. Use-case діаграма взаємодії здобувача та системи

На даній діаграмі можна побачити взаємодію викладача, здобувач з системою чат-боту. Викладач має наступні можливості: авторизуватися в системі, створити новий тест та редагувати вже створений тест. При створенні тесту, викладач може вибирати серед наступних типів запитань: питання з однією правильною відповіддю, питання кількома відповідями, питання на встановлення послідовність, питання на встановлення відповідність. При редагуванні тесту, виклад може змінити назву створеного тесту, або видалити тест.

Серед можливих дій здобувачу належать наступні: авторизуватися в системі, пройти створений тест по вибраній дисципліні, переглянути результат з пройденого тесту.

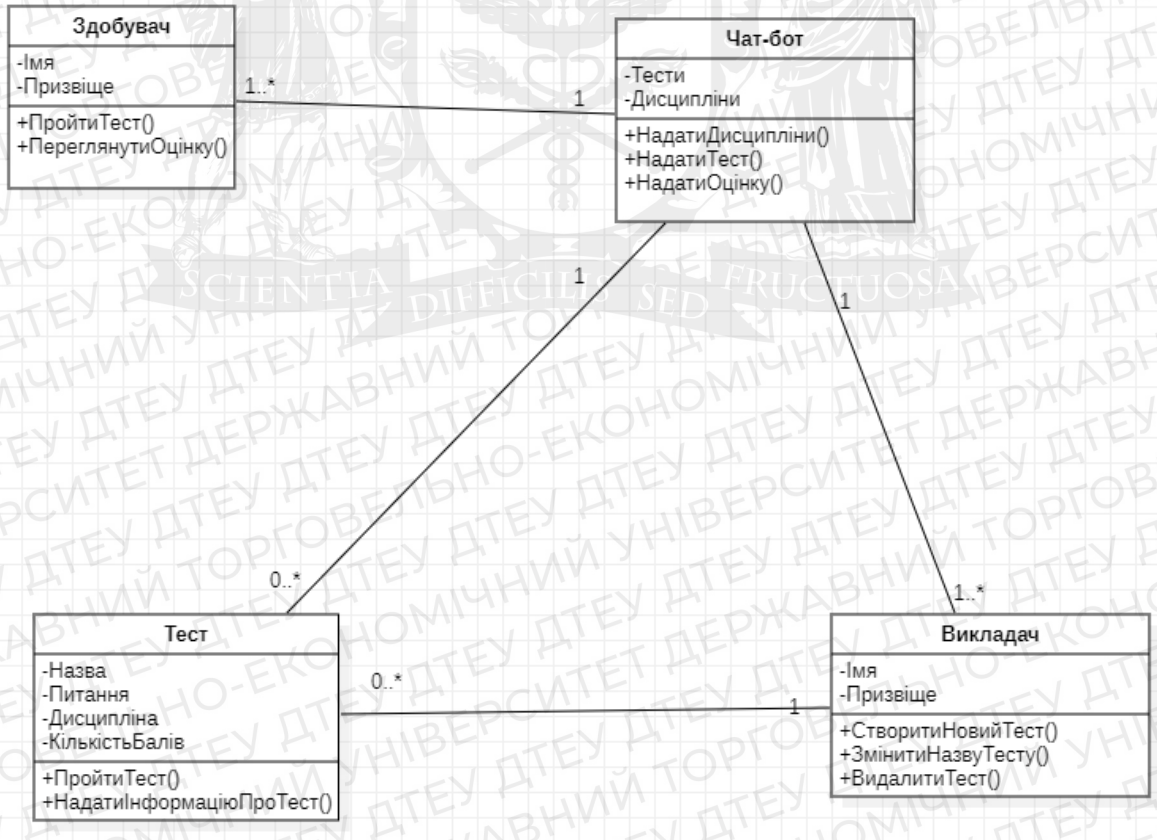

Рис. 2.3. Діаграма класів чат-бота для тестування знань

Як видно з цієї діаграм клас здобувач взаємодіє лише з класом чат-боту, і має функції: пройти тест та переглянути свою оцінку. Клас викладач може взаємодіяти з класами чат-боту та тесту, і має функції: створити тест з питаннями чотирьох типів, змінити назву створеного тесту та видалити.

РОЗДІЛ 3. ПРАКТИЧНА РЕАЛІЗАЦІЯ ЧАТ-БОТУ ДЛЯ ТЕСТУВАННЯ ЗНАНЬ

3.1. Архітектура чат-боту

Перед тим як починати розробку чат-боту в Телеграмі, необхідно ознайомитися архітектурою. Архітектура чат-боту в Телеграмі має декілька компонентів. Серед основних можна виділити:

- Telegram API. Це основний інтерфейс, який дозволяє боту взаємодіяти з користувачем, через оболонку Telegram. Для реалізації цього компоненту існує безліч бібліотек, вибір яких залежить від потрібного функціоналу чат-боту та мови програмування логіки.
- Сервер або хмарне середовище. Цей компонент використовується для обробки вхідних запитів від користувачів і надсилання відповідей. Для цього компонента можна використовувати будь-яку серверну технологію, або хмарний сервіс, щоб обробляти вхідні запити та відправляти відповіді від бота.
- Логіка роботи чат-бота. Цей компонент являє собою програмне код або набір команд, за допомогою якого чат-бот оброблює вхідні повідомлення від користувачів та надсилає відповіді. Для написання логіки чат-боту можна використовувати такі мови програмування, як: PHP, Go, Python і подібні.
- Сховище або База даних. Цей компонент необхідний якщо при використанні чат бота потребується зберігати дані, наприклад користувацькі налаштування, стан розмови і подібне. Для даних потреб можна використовувати MySQL, PostgreSQL, MongoDB.

Telegram API є ключовим компонентом архітектури чат-бота в Телеграмі. Він забезпечує зв'язок між вашим ботом і месенджером Telegram,

дозволяючи боту отримувати оновлення повідомлень від користувачів та надсилати відповіді.

Основні пункти, які варто знати про Telegram API, включають:

- Отримання токену;
- використання методів API;
- обробка оновлень;
- робота з повідомленнями;
- відправлення відповідей;
- робота з клавіатурою.

Використовуючи Telegram API, можливо побудувати потужного чат-бота для тестування знань здобувачів освіти, забезпечуючи зручну взаємодію з користувачами і обробку їх відповідей.

Як сховище даних в даній архітектурі чат-боту для тестування знань здобувачів освіти було використано СКБД MongoDB. MongoDB є нереляційною базою даних, яка зберігає дані у форматі документів JSON-подібного типу, відомих як BSON.

Основні аспекти використання MongoDB в архітектурі чат-боту включають:

- Збереження питань та варіантів відповідей;
- індексування та швидкий доступ;
- масштабованість;
- запити до бази даних для отримання доступу до тестів.

Після детального розгляду архітектури чат-боту, переходимо до його програмної реалізації для здійснення процесу тестування. Розглянемо основні етапи програмування чат-боту, включаючи розробку логіки взаємодії з користувачем, обробку запитів, збереження даних та інтеграцію з необхідними інструментами.

3.2. Програмна реалізація чат-боту для тестування

Даний проєкт був реалізований на мові Python в середовищі розробки Visual Studio Code. Для реалізації взаємодії з чат-ботом було використано API бібліотеки: telegram, telegram.ext, telebot – які відповідно були створені на мові Python. Для зберігання даних використовувалося система управління базами даних MongoDB. Також для реалізації чат-боту для оцінки знань було використано модульних підхід програмування, що допомогло зменшити складність системи та надало гнучкість елементів системи.

Чат-бот для тестування знань здобувачів освіти складається із наступних модулів:

main – головний модуль, ціль якого полягає у зв'язані всіх інших модулів та бути вхідною точкою для взаємодії користувача з чат-ботом. Методи цього модулю записані в табл. 3.1.

Таблиця 3.1

Методи модуля main

Назва методу	Дія методу
help_command(update, context)	Метод який викликається при написанні команди в чат /help. Демонструє всі доступні команди для користувача.
log_in()	Метод початкових команд /start та /exit
exit(update,_)	Метод який визивається при написанні команди в чат /exit. Розлогінює поточного коистувача.
error(update, context)	Метод управління помилками при роботі чат-бот.
start(update, context)	Метод який викликається при написанні команди в чат /start. Запускає роботу боту.

user_type_select(update, context)	Метод який надає доступ до команд залежно від вибраного типу користувача.
teacher()	Метод для роботи з командами викладача.
student()	Метод для роботи з командами здобувача.

Модуль main взаємодії з наступними модулями задля забезпечення можливостей створювати тести, редагувати тести та проходити тести:

- create_test;
- edit_test;
- attempt_test.

create_test – модуль для створення тестів. Налічує команди вибору типів запитань та створення правильних та неправильних відповідей. Методи цього модулю записані в табл. 3.2.

Таблиця 3.2

Методи модуля create_test

Назва методу	Дія методу
start(update, _)	Метод початкового повідомлення при створенні тесту.
Cancel(update, _)	Метод відміни створення тесту.
Enter_discipline(update, _)	Метод вибору дисципліни з якої буде створений тест.
Enter_type(update, _)	Метод вибору типу запитання.
Enter_question(update, _)	Метод опрацювання запитань.
Enter_compliance(update, _)	Метод опрацювання питань на відповідність.
Enter_answer(update, _)	Метод внесення відповіді на запитання.

Enter_possible_answer(update, _)	Метод внесення неправильних відповідей.
Enter_randomness_question(update, _)	Метод запиту на перемішання відповідей для запитання.
Enter_randomness_test(update, _)	Метод запиту на перемішання питань в тесті.
Enter_result_after_question(update, _)	Метод запиту на показ результату після дання відповіді на питання.
Enter_result_after_test(update, _)	Метод запиту на показ результату після закінчення тесту.
Enter_test_name(update, context)	Метод внесення назви тесту.

Також додатково цей модуль взаємодіє з модулем test, для створення об’єктів класу Test, які зберігають в собі інформацію про тест, та питання. Для зв’язку з базою даних та внесення нового тесту було використано код який зображений на Рис. 3.1.

```
user_col = db[userDict[user_id]['discipline']]
user_col.insert_one(
    {'testname': testname, 'testinstance': pickle.dumps(userDict[user_id]['test'])})
```

Рис. 3.1. Програмний код внесення тесту в БД

Після додавання нового тесту до бази даних він буде зберігатися у відповідному розділі дисципліни(Рис. 3.2).

Collection Name	Documents	Logical Data Size	Avg Document Size	Storage Size	Indexes	Index Size	Avg Index Size
tests.Економіка	1	393B	393B	24KB	1	24KB	24KB
tests.Математика	3	1.61KB	551B	36KB	1	36KB	36KB
tests.Програмування	4	1.59KB	408B	36KB	1	36KB	36KB
tests.Фізика	1	463B	463B	20KB	1	20KB	20KB

Рис. 3.2 Структура розділів в СКБД MongoDB

Після створення тесту та додання його до бази даних, його можна видалити або змінити назву завдяки модулю edit_test.

edit_test – модуль редагування існуючих тестів. Цей модуль надає можливість змінювати назву існую чого тесту, або видали існуючий тест. Методи цього модулю записані в табл. 3.3.

Таблиця 3.3

Методи модуля edit_test

Назва методу	Дія методу
start_remove(update, _)	Метод початку видалення тесту.
enter_discipline_remove(update, _)	Метод вибору дисципліни з якої треба видалити тест.
enter_name_remove(update, context)	Метод внесення імені тесту на видалення.
start_rename(update, _)	Метод початку перейменування тесту.
enter_discipline_rename(update, _)	Метод вибору дисципліни з якої треба перейменувати тест.
enter_old_name(update, context)	Метод пошуку тесту для перейменуванню.
enter_new_name(update, context)	Метод внесення нового іменні для тесту.
cancel_edit(update, _)	Метод відміни редагування тесту.

Після створення нового тесту, до нього може отримати доступ здобувач завдяки модулю attempt_test.

attempt_test – модуль проходження тесту. В цьому модулі присутні методи для отримання питанння, надання відповіді, та оброки відповіді від користувача. Методи цього модулю записані в табл. 3.4.

Таблиця 3.4

Методи модуля attempt_test

Назва методу	Дія методу
start(update, _)	Метод початку проходження тесту.
cancel(update, _)	Метод відміни проходження тесту.
enter_discipline(update, _)	Метод вибори дисципліни з якої необхідно пройти тест.
enter_test(update, context)	Метод вибору тесту для проходження.
enter_answer(update, _)	Метод внесення відповіді на питання.
ask_question(update)	Метод відображення запитань.

test – модуль класу Test. Об’єкти цього класу являють собою тест зі списком запитань. Методи цього модулю записані в табл. 3.5.

Таблиця 3.5

Методи модуля test

Назва методу	Дія методу
__init__(self, author="")	Конструктор класу Test.
add_question(self, new_question: Question)	Метод додавання запитання з відповіддю до тесту.
get_questions(self)	Метод отримання списку запитань.

Як можна побачити з програмної реалізації класу на Рис. 3.3, клас тест має параметри:

- question;
- is_random;
- show_results_after_quiz;
- show_results_after_question.

Параметр is_random відповідає за те, чи потрібно відображати питання в самому тесті випадковим чином, параметр show_results_after_quiz – за відображення правильних відповідей після тесту, show_results_after_question – за відображення правильної відповіді після запитання. І останній параметр question являє собою масив об'єктів класу Question, опис яких зберігається у наступному модулі, questions.

```python
from classes.questions import Question

class Test:
    def __init__(self):
        self.questions = []
        self.is_random = False
        self.show_results_after_quiz = True
        self.show_results_after_question = True

    def add_question(self, new_question: Question):
        self.questions.append(new_question)

    def get_questions(self):
        return self.questions.copy()
```

Рис. 3.3. Програмна реалізація класу

questions – модуль класів типів запитань. В цьому модулі налічуються класи Question, MultipleChoices, SingleChoice, SequenceChoice, ComplianceCoice.

Клас Question – це базовий клас запитання, який має параметри запитання, правильної відповіді та відповіді користуача. Методи цього класу записані в табл. 3.6.

Таблиця 3.6

Методи класу Question

Назва методу	Дія методу
__init__(self, question, correct_answer)	Конструктор класу Question
check_solution(self)	Метод перевірки правильної відповіді.
enter_solution(self, answer)	Метод внесення відповіді.

Клас MultiChoises – це клас запитань з декількома правильних відповідей, який наслілує клас Question. Методи цього класу записані в табл. 3.7.

Таблиця 3.7

Методи класу MultiChoise

Назва методу	Дія методу
__init__(self, question, correct_answer)	Конструктор класу MultiChoises.
add_possible_answer(self, new_answer)	Метод додавання додаткої правильної відповіді.
check_solution(self)	Метод перевірки правильної відповіді.

Клас SingleChoice – це клас запитань з однією правильною відповіддю, який наслідує MultiChoises. Методи цього класу записані в табл. 3.8.

Таблиця 3.8

Методи класу SingleChoice

Назва методу	Дія методу
__init__(self, question="", correct_answer="")	Конструктор класу SingleChoice.
enter_solution(self, answer)	Метод внесення правильної відповіді.

Клас SequenceChoice – це клас запитань на послідовність, який наслідує клас MultiChoises. Методи цього класу записані в табл. 3.9.

Таблиця 3.9

Методи класу SequenceChoice

Назва методу	Дія методу
check_solution(self)	Метод перевірки правильної послідовності.

Клас ComplianceChoice – це клас запитань на відповідність, який наслідує клас SequenceChoice. Методи цього класу записані в табл. 3.10.

Таблиця 3.10

Методи класуComplianceChoice

Назва методу	Дія методу
__init__(self, question, correct_answer)	Конструктор класу ComplianceCoice.
add_complaince(self, new_compl)	Метод внесення елементів відповідності.

attempt – це модуль класу Attempt. Об’єкти цього класу являють собою спробу проходження тесту. Методи цього модулю записані в табл. 3.11.

Табилця 3.11

Методи модуля attempt

Назва методу	Дія методу
__init__(self, test: Test)	Конструктор класу Attempt.
has_next_question(self)	Метод перевірки чи є наступне запитання.
act_question(self)	Метод показу поточного запитання.
current_question(self)	Метод отримання поточного запитання.
input_answer(self, user_answer)	Метод внесення відповіді на питання.
enter_answer(self)	Метод перевірки відповіді.

3.3. Функціональні можливості чат-боту для тестування знань здобувачів

Для початку взаємодії з чат-ботом треба написати команду /start, що активує чат бота, та виведе наступне повідомлення (Рис. 3.4).

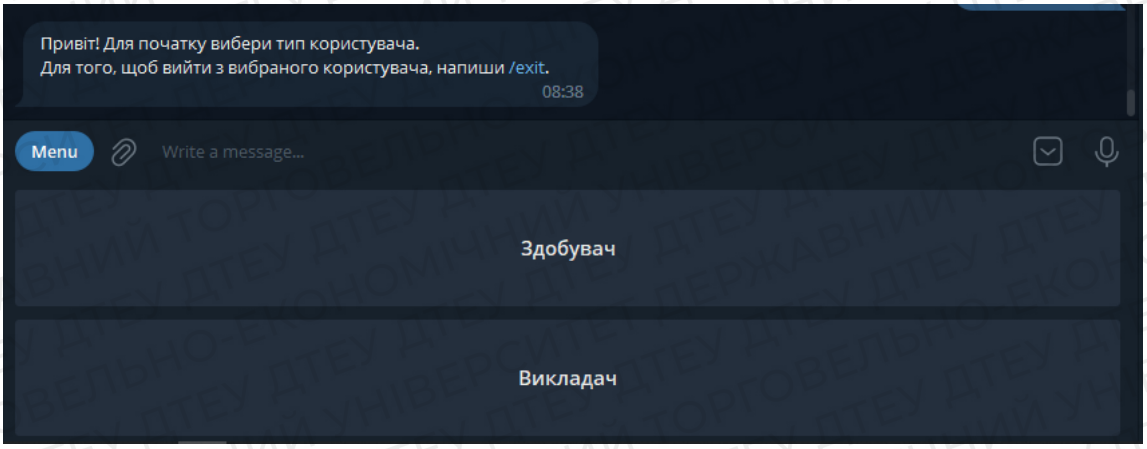

Рис. 3.4. Результат команди /start

Після чого постає вибір за кого авторизуватися за здобувача або за викладача. Якщо вибрати варіант авторизуватися за викладача, то нам стане доступні команди для створення, редагування та видалення тестів (Рис. 3.5).

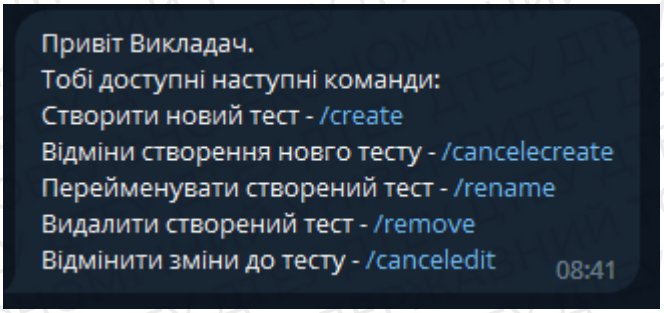

Рис. 3.5. Команди викладача.

Серед можливостей створення тестів доступно: вибір дисципліни з якої бажаємо створити тест, тип запитання, текст запитання, введення правильної відповідь, неправильні відповіді, вибір чи потрібно перемішувати відповіді на запитання, чи потрібно перемішувати питання в тесті, чи демонструвати правильно відповідь після питання, чи демонструвати правильні відповіді після проходження тесту та додавання назви тесту (Рис.3.6 – 3.15).

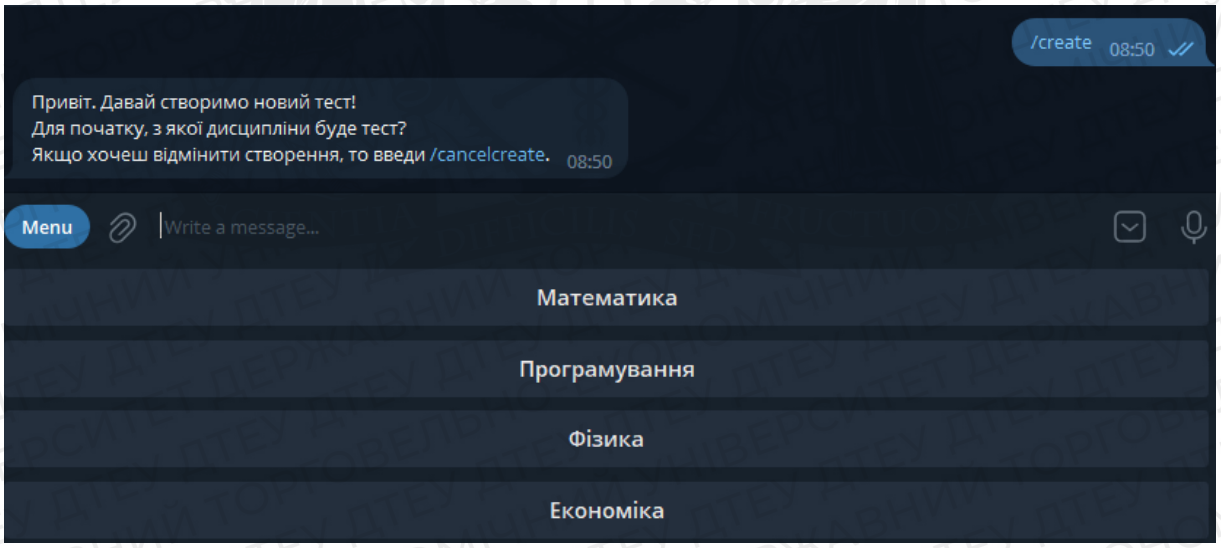

Рис. 3.6. Вибір дисципліни.

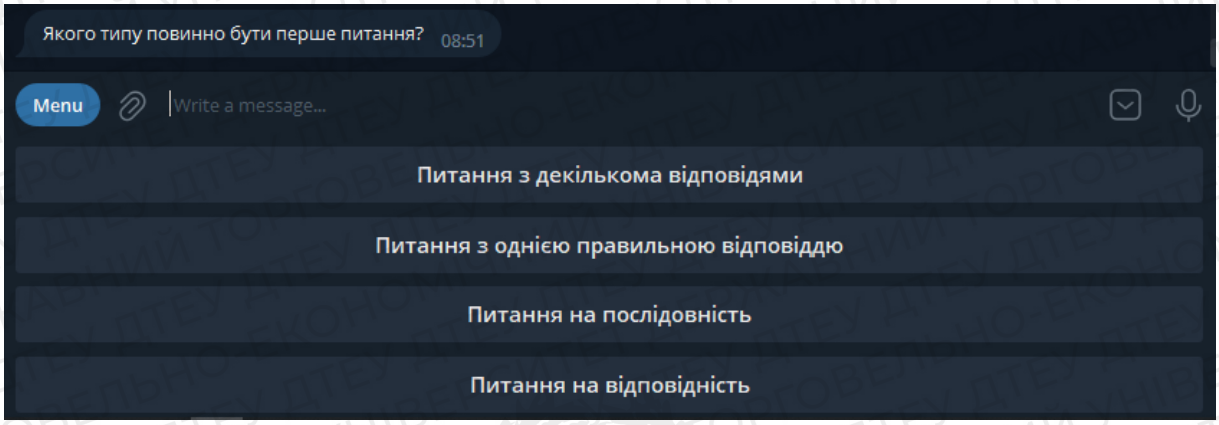

Рис. 3.7. Вибір типу запитання.

Рис. 3.8. Внесення тексту запитання.

Рис. 3.9. Внесення правильної відповіді.

Рис. 3.10. Внесення неправильних відповідей.

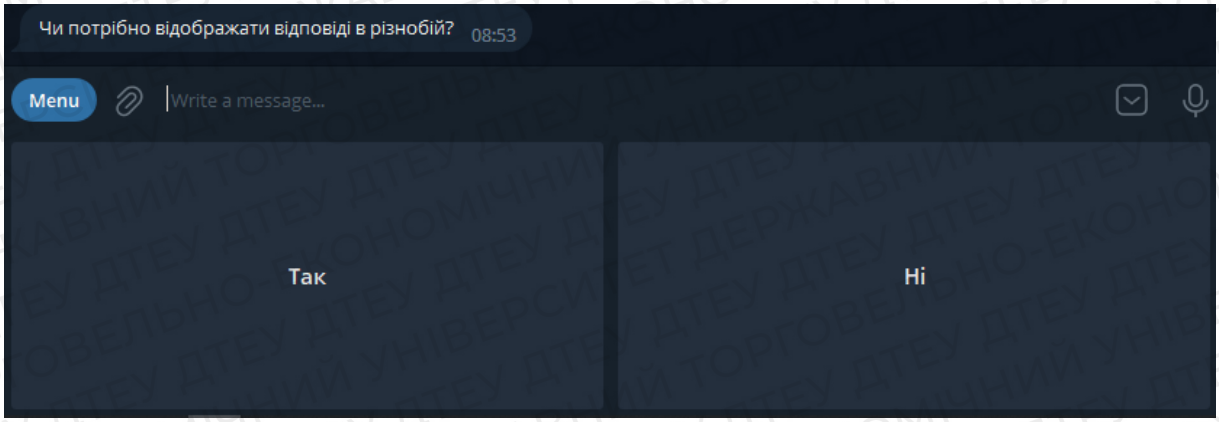

Рис. 3.11. Запит на відображання відповідей в різнобій.

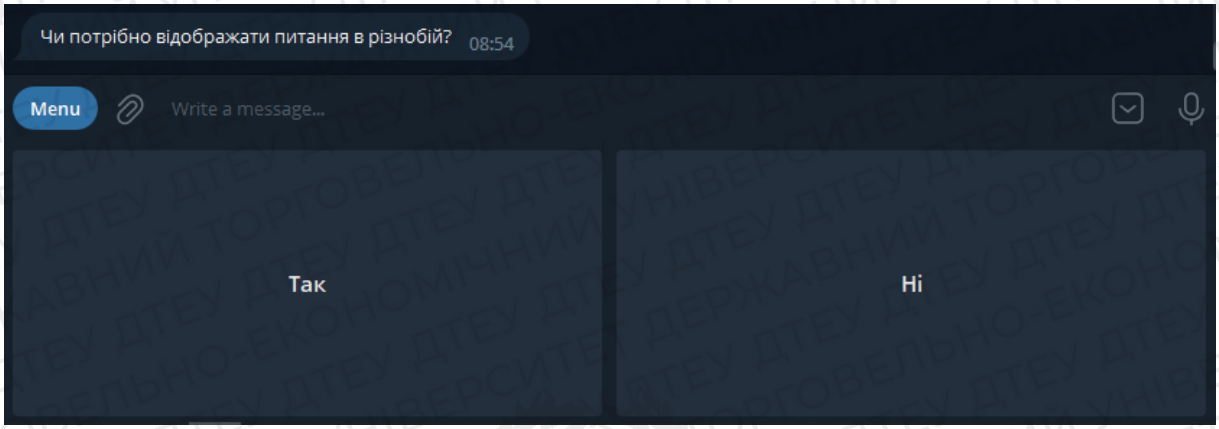

Рис. 3.12. Запит на відображення питань в різнобій.

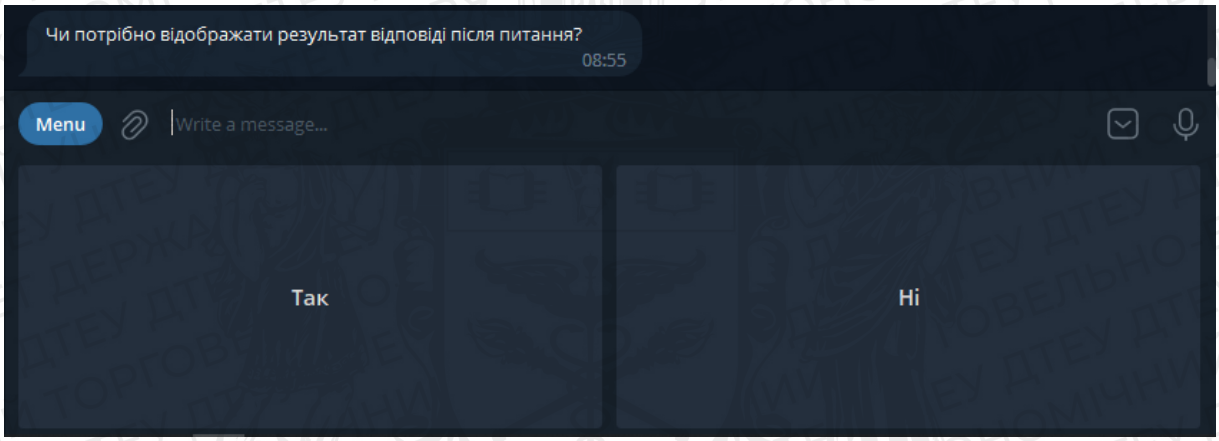

Рис. 3.13. Запит на відображення відповідей після питань.

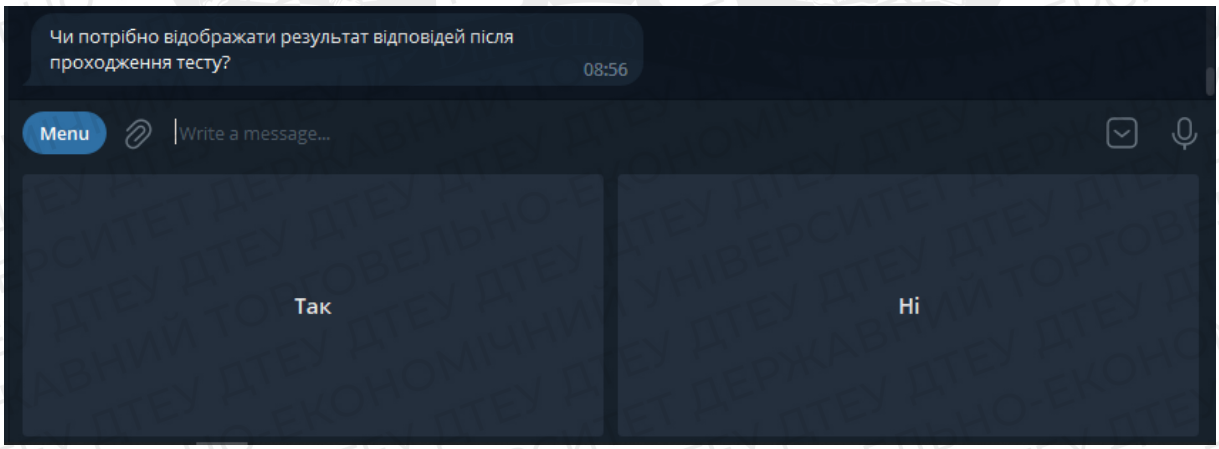

Рис. 3.14. Запит на відображення відповідей в кінці тесту.

Рис. 3.15. Внесення назви тесту.

Серед можливостей для користувача присутні команди проходження тесту (Рис. 3.16).

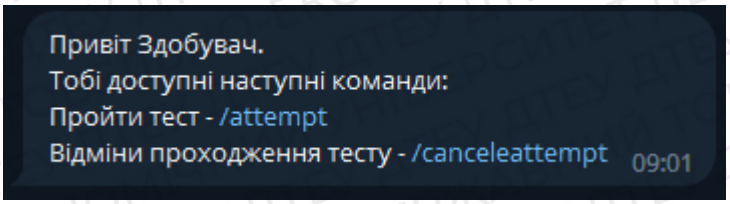

Рис. 3.16. Команди для здобувача.

Серед можливостей проходження тесту є наступні: вибір дисципліни, внесення назви тесту, проходження тесту (Рис. 3.17 -3.19).

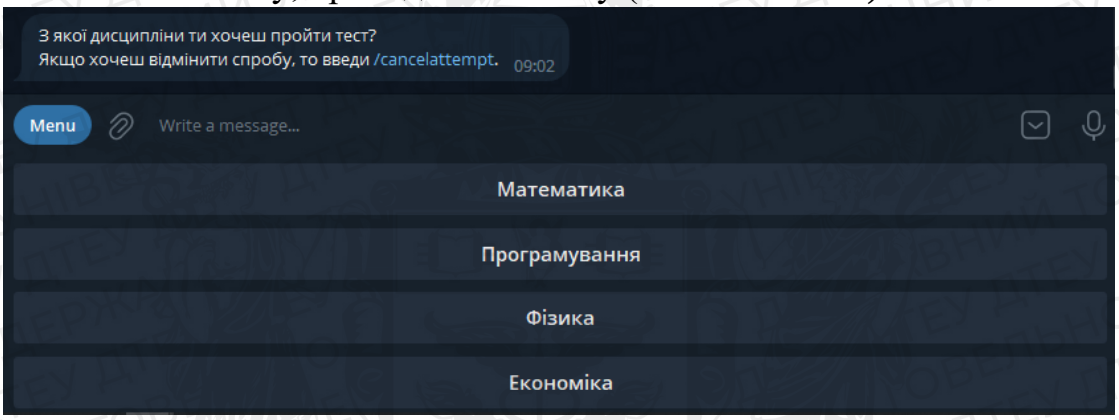

Рис. 3.17. Вибір дисципліни.

Рис. 3.18. Внесення назви тесту.

Рис. 3.19.Проходження тесту.

ВИСНОВКИ

Здійснюючи аналіз науково-педагогічної літератури, були визначені різні засоби для здійснення тестування здобувачів освіти. Ці засоби включають традиційні письмові тести, комп'ютерні тести, онлайн-платформи тестування та чат-ботів. Чат-боти виявилися перспективним інструментом для проведення тестування, забезпечуючи інтерактивний та зручний спосіб спілкування зі студентами.

Використання чат-ботів у освітньому процесі виявилося ефективним інструментом. Вони можуть забезпечувати навчання та тестування здобувачів освіти у діалоговій формі, надаючи індивідуальний підхід та забезпечуючи доступність 24/7. Чат-боти також дозволяють студентам отримувати миттєвий зворотний зв'язок та надають можливість вивчати матеріал у зручний для них час та темп.

У процесі роботи були розроблені діаграми та визначено функціонал чат-боту. Функціонал чат-боту включає можливість проведення тестування з різних предметів, відображення результатів, надання пояснень до правильних відповідей, збір статистики про відповіді студентів та інші корисні функції.

Розроблений чат-бот має чіткі інструкції для використання. Інструкції надаються студентам, щоб забезпечити їхнє зручне користування чат-ботом і ефективне використання його можливостей для тестування знань.

Під час розробки чат-бота було забезпечено його доступність на різних типах пристроїв, таких як комп'ютери, смартфони та планшети. Це дозволяє студентам мати доступ до чат-бота з будь-якого зручного для них пристрою і забезпечує гнучкість використання.

СПИСОК ВИКОРИСТАНИЇ ДЖЕРЕЛ

1. Брітавська О. П. Особливості комп'ютерних засобів тестування знань / Олена Павлівна Брітавська, Іван Іванович Дончев, Наталія Володимирівна Бондаренко, Віталій Вікторович Горохов, Анатолій Володимирович Опарін // Новітні комп'ютерні технології. – Кривий Ріг, 2018. – Том XVI. – С. 127-133.

2. Пометун О. Таксономія Б. Блума і розвиток критичного мислення школярів на уроках історії / О. Пометун Г. Нестор // Український педагогічний журнал. – Вер 2019. – С. 50–58.

3. Дичка Н.І., Павленко О.В. Таксономія Б. Блума в навчанні англійської мови професійного спрямування в XXI столітті / Н.І. Дичка, О.В. Павленко // Науковий журнал "Молодий вчений" №12(52) грудень, 2017. – Херсон: ТОВ. "Видавничий дім "Гельветика", 2017. – С. 370 – 373.

4. Мороховець Г. Ю. Тестування як форма контролю та діагностики знань здобувачів вищої освіти / Г. Ю. Мороховець // Освіта та розвиток обдарованої особистості. - 2018. - № 3. - С. 11-15.

5. Тутова Н. В. Тестові технології оцінювання знань здобувачів вищої освіти та їх застосування / Н. В. Тутова / Шлях успіху і перспективи розвитку (до 26 річниці заснування Харківського національного університету внутрішніх справ) : матеріали міжнар. наук.-практ. конф. (м. Харків, 20 листоп. 2020 р.) / редкол.: Д. В. Швець (голова), О. М. Бандурка, С. М. Гусаров та ін.; МВС України, Харків. нац. ун-т внутр. справ. - Харків : ХНУВС, 2020.- С. 492-494.

6. Мороховець Г. Ю. Тестування як форма контролю знань здобувачів вищої освіти / Г. Ю. Мороховець, Ю. В. Лисанець // Сучасна медична освіта: методологія, теорія, практика : матеріали Всеукр. навч.-наук. конф. з

міжнар. участю, м. Полтава, 19 березня 2020 р. – Полтава, 2020. – С. 144–145.

7. Опарін А. В. Проблеми комп'ютерного тестування знань у сучасній освіті / А. В. Опарін, О. П. Брітавська, Л. Ю. Куценко // Науковий вісник Південноукраїнського національного педагогічного університету ім. К. Д. Ушинського. Педагогічні науки. - 2017. - № 1. - С. 68-74.

8. Google Forms: Online form Creator | Google Workspace [Електроний ресурс]. – Режим доступу: https://www.google.com/forms/about/.

9. Навчальні інструменти, картки та рішення з підручників | Quizlet [Електроний ресурс]. – Режим доступу: https://quizlet.com/uk.

10. Quiz Maker Software | Create Quizzes, Tests & Assessments Online [Електроний ресурс]. – Режим доступу: https://www.proprofs.com/quiz-school/.

11. Розроблено Telegram-бот «Цифрограм. Твоя кібергігієна», який проводить експрес-тестування цифрових знань | Кабінет Міністрів України [Електроний ресурс]. – Режим доступу: https://www.kmu.gov.ua/news/rozrobleno-telegram-bot-cifrogram-tvoya-kibergigiyena-yakij-provodit-ekspres-testuvannya-cifrovih-znan.

12. Vanichvasin P. Chatbot Development as a Digital Learning Tool to Increase Students' Research Knowledge / Patchara Vanichvasin. // International Education Studies. – 2021. – №2. – С. 44–53.

13. Dutta D. Developing an Intelligent Chat-bot Tool to Assist High School Students for Learning General Knowledge Subjects / Debasatwa Dutta. // Georgia Institute of Technology Atlanta, Georgia, USA. – 2017. – С. 1–13.

14. Як створити свого чат-бота у телеграм? Налаштування чат-боту для телеграма на конструкторі. | Gerabot [Електроний ресурс] – Режим доступу: https://gerabot.com/article/yak_stvoriti_svogo_chatbota_u_telegram

15. Популярність Telegram в Україні перевищує рейтинг марафону "Єдині новини", - результат дослідження | Informator [Електроний ресурс] – Режим доступу: https://informator.ua/uk/populyarnist-telegram-v-ukrajini-perevishchuye-reyting-marafonu-yedini-novini-rezultat-doslidzhennya

16. Telegram Bot Features | Telegram [Електроний ресурс] – Режим доступу: https://core.telegram.org/bots/features

17. What is MongoDB? – MongoDB Manual | MongoDB [Електронний ресурс] – Режим доступу до ресурсу: https://www.mongodb.com/docs/manual/

18. Гавриленко Н. І. Тестування як форма об'єктивного контролю та діагностики знань здобувачів вищої освіти / Н. І. Гавриленко // Імідж сучасного педагога. – 2018. – С. 38-40

19. Тенденції та статистика чат-ботів у 2023 році | Digiexe [Електроний ресурс]. – Режим доступу: https://digiexe.com/uk/blog/chatbots-trends-and-statistics-to-follow/

20. Бровко Ю. В. Чат-боти для організації дистанційного навчання | [Електронний ресурс] – Режим доступу до ресурсу: http://dspace.pnpu.edu.ua/bitstream/123456789/20518/1/3.pdf

21. Буткевич А. Підготовка здобувачів освіти до ЗНО засобами тестування / А. Буткевич, Л. Яременко, О. Буткевич // Наукові записки молодих учених. – 2022. – № 10. – С. 1-12

22. Чат-бот для бізнесу в Україні — сценарії та добірка сервісів | Helpcrunch [Електронний ресурс] – Режим доступу до ресурсу: https://helpcrunch.com/blog/uk/chat-bot-dla-biznesu/

ДОДАТОК

ДОДАТОК А

Програмний код чат-боту(модуль main)

```
import telebot
import logging
from typing import Final
from telegram import Update, ReplyKeyboardMarkup, ReplyKeyboardRemove
from telegram.ext import Application, CommandHandler, MessageHandler, filters, ContextTypes,
    ConversationHandler
import create_test as createTest
import edit_test as editTest
import attempt_test as attemptTest
TOKEN: Final = '6017771439:AAHOdLVziHWX2_WzRny5dERoginym4IAbNY'
BOT_USERNAME: Final = '@OtsinyuvachBot'
logging.basicConfig(format='%(asctime)s - %(name)s - %(levelname)s - %(message)s',
                    level=logging.INFO)
logger = logging.getLogger(__name__)
bot = telebot.TeleBot(TOKEN)
app = Application.builder().token(TOKEN).build()
userDict = dict()
type_user = {
   'Викладач',
   'Здобувач'
}
# Commands
async def help_command(update: Update, context: ContextTypes.DEFAULT_TYPE):
    user_id = update.message.from_user.id
    if userDict[user_id] == 'Викладач':
        reply_text = (
            "Створити новий тест - /create\n"
            "Відміни створення новго тесту - /cancelecreate\n"
            "Перейменувати створений тест - /rename\n"
```

```
            "Видалити створений тест - /remove\n"
            "Відмінити зміни до тесту - /canceledit\n"
            "Вийти з профілю - /exit"
        )
    elif userDict[user_id] == 'Здобувач':
        reply_text = (
            "Пройти тест - /attempt\n"
            "Відміни проходження тесту - /canceleattempt\n"
            "Вийти з профілю - /exit"
        )
    else:
        reply_text = (
            "Почати роботу з ботом - /start"
        )

    await update.message.reply_text(reply_text)
def log_in():
    app.handlers.clear()
    log_in_states = {
        'ENTER_USER_TYPE':[ MessageHandler(filters.TEXT & ~filters.COMMAND,
     user_type_select)],
    }
    start_handler = ConversationHandler(
        entry_points=[CommandHandler('start', start)],
        states=log_in_states,
        fallbacks=[CommandHandler('exit', exit)]
    )
    app.add_handler(start_handler)
    app.add_handler(CommandHandler('exit', exit))
     # help command
    app.add_handler(CommandHandler('help', help_command))
```

```
async def exit(update, _):
    user_id = update.message.from_user.id
    await update.message.reply_text(
        "Бувайте. \n"
        'Для того, щоб заново почати роботу, напишіть /start',
        reply_markup=ReplyKeyboardRemove()
    )
    userDict[user_id] = None
    app.handlers.clear()
    log_in()

async def error(update: Update, context: ContextTypes.DEFAULT_TYPE):
    print(f'Update {update} caused error {context.error}')
async def start(update: Update, context: ContextTypes.DEFAULT_TYPE):
    list_users = [[el] for el in list(type_user)]
    await update.message.reply_text(
        "Вітаю! Для початку виберіть тип користувача.\n"
        'Для того, щоб вийти з вибраного профілю, напишіть /exit.',
        reply_markup=ReplyKeyboardMarkup(
            list_users, one_time_keyboard=True)
    )
    return 'ENTER_USER_TYPE'
async def user_type_select(update: Update, context: ContextTypes.DEFAULT_TYPE):

    user_id = update.message.from_user.id
    userDict[user_id] = update.message.text
    if    update.message.text == 'Викладач':
        await update.message.reply_text(
            "Привіт Викладач.\n"
            "Тобі доступні наступні команди:\n"
            "Створити новий тест - /create\n"
            "Відміни створення нового тесту - /cancelecreate\n"
```

```
        "Перейменувати створений тест - /rename\n"
        "Видалити створений тест - /remove\n"
        "Відмінити зміни до тесту - /canceledit"
      )
      teacher()
      logger.info('[%s] enter as Teacher', update.message.from_user.username)
    elif    update.message.text == 'Здобувач':
      await update.message.reply_text(
        "Привіт Здобувач.\n"
        "Тобі доступні наступні команди:\n"
        "Пройти тест - /attempt\n"
        "Відміни проходження тесту - /canceleattempt"
      )
      student()
      logger.info('[%s] enter as Student', update.message.from_user.username)
    return ConversationHandler.END
  def teacher():
    # Conversation if the user wants to create a test
    create_states = {
      'ENTER_DISCIPLINE':[          MessageHandler(filters.TEXT & ~filters.COMMAND,
    createTest.enter_discipline)],
      'ENTER_TYPE': [               MessageHandler(filters.TEXT & ~filters.COMMAND,
    createTest.enter_type)],
      'ENTER_QUESTION': [           MessageHandler(filters.TEXT & ~filters.COMMAND,
    createTest.enter_question)],
      'ENTER_COMPLIANCE': [         MessageHandler(filters.TEXT & ~filters.COMMAND,
    createTest.enter_compliance)],
      'ENTER_ANSWER': [             MessageHandler(filters.TEXT & ~filters.COMMAND,
    createTest.enter_answer)],
      'ENTER_POSSIBLE_ANSWER': [    MessageHandler(filters.TEXT & ~filters.COMMAND,
    createTest.enter_possible_answer)],
      'ENTER_RANDOMNESS_QUESTION': [  MessageHandler(filters.TEXT &
    ~filters.COMMAND, createTest.enter_randomness_question)],
      'ENTER_RANDOMNESS_TEST': [    MessageHandler(filters.TEXT & ~filters.COMMAND,
    createTest.enter_randomness_test)],
```

```
    'ENTER_RESULT_AFTER_QUESTION': [MessageHandler(filters.TEXT &
  ~filters.COMMAND, createTest.enter_result_after_question)],
    'ENTER_RESULT_AFTER_TEST': [    MessageHandler(filters.TEXT & ~filters.COMMAND,
  createTest.enter_result_after_test)],
    'ENTER_TEST_NAME': [          MessageHandler(filters.TEXT & ~filters.COMMAND,
  createTest.enter_test_name)],
  }
  create_handler = ConversationHandler(
    entry_points=[CommandHandler('create', createTest.start)],
    states=create_states,
    fallbacks=[CommandHandler('cancelcreate', createTest.cancel)],
  )
  app.add_handler(create_handler)
  # Conversation about remove or renaming exisiting test
  edit_states = {
    'ENTER_DISCIPLINE_REMOVE': [MessageHandler(filters.TEXT & ~filters.COMMAND,
  editTest.enter_discipline_remove)],
    'ENTER_NAME': [MessageHandler(filters.TEXT & ~filters.COMMAND,
  editTest.enter_name_remove)],
    'ENTER_DISCIPLINE_RENAME': [MessageHandler(filters.TEXT & ~filters.COMMAND,
  editTest.enter_discipline_rename)],
    'ENTER_OLD_NAME': [MessageHandler(filters.TEXT & ~filters.COMMAND,
  editTest.enter_old_name)],
    'ENTER_NEW_NAME': [MessageHandler(filters.TEXT & ~filters.COMMAND,
  editTest.enter_new_name)]
  }
  edit_handler = ConversationHandler(
    entry_points=[CommandHandler('rename', editTest.start_rename), CommandHandler(
      'remove', editTest.start_remove)],
    states=edit_states,
    fallbacks=[CommandHandler('canceledit', editTest.cancel_edit)],
  )
  app.add_handler(edit_handler)
def student():
  # Conversation if the user wants to attempt a test
```

```
    attempt_states = {
        'ENTER_DISCIPLINE': [   MessageHandler(filters.TEXT & ~filters.COMMAND,
    attemptTest.enter_discipline)],
        'ENTER_TEST': [         MessageHandler(filters.TEXT & ~filters.COMMAND,
    attemptTest.enter_test)],
        'ENTER_ANSWER': [       MessageHandler(filters.TEXT & ~filters.COMMAND,
    attemptTest.enter_answer)]
    }
    attempt_handler = ConversationHandler(
        entry_points=[CommandHandler('attempt', attemptTest.start)],
        states=attempt_states,
        fallbacks=[CommandHandler('cancelattempt', attemptTest.cancel)],
    )
    app.add_handler(attempt_handler)
if __name__ == '__main__':
    log_in()
    # log all errors
    app.add_error_handler(error)
    # Polls the bot
    app.run_polling()
```

ДОДАТОК Б

Програмний код чат-боту(модуль create_test)

```
import logging
import pymongo
import pickle
from telegram import ReplyKeyboardRemove, ReplyKeyboardMarkup, constants
from telegram.ext import ConversationHandler
from classes.questions import MultipleChoices, SingleChoice, SequenceChoice, ComplianceCoice
from classes.test import Test
logging.basicConfig(format='%(asctime)s - %(name)s - %(levelname)s - %(message)s',
                    level=logging.INFO)
logger = logging.getLogger(__name__)
MONGODB =
"mongodb+srv://Devi:qwerty123456@cluster0.sq1kkvb.mongodb.net/Otsinyuvach?retryWrites=true
&w=majority"
db = pymongo.MongoClient(MONGODB).Otsinyuvach.tests
list_of_disciplines = {
   'Програмування',
   'Математика',
   'Економіка',
   'Фізика'
}
# Dict with user data like a test instance
userDict = dict()
# Dict with string and associated question class
dict_question_types = {
   'Питання з кількома відповідями':            MultipleChoices,
   'Питання з однією правильною відповіддю':     SingleChoice,
   'Питання на встановлення послідовність':      SequenceChoice,
   'Питання на встановлення відповідність':      ComplianceCoice
}
async def start(update, _):
```

```
    logger.info('[%s] Creation initialized', update.message.from_user.username)
    if update.message.from_user.id in userDict:
        # user is in the middle of a test and cant attempt to a second one
        logger.info('[%s] Creation canceled, because the user is in the middle of a creation.',
                    update.message.from_user.username)
        await update.message.reply_text(
            "Ви вже в процесі створення тесту."
            "Ви не можеш створювати два тести одночасно\n"
            'Якщо ви хочеш відмінити створення, то введіть /cancelcreate.',
            reply_markup=ReplyKeyboardRemove()
        )
        return ConversationHandler.END
    # Init Test for user
    userDict[update.message.from_user.id] = {
        'test': Test()}
    # Asks for type of discipline
    list_disciplines = [[el] for el in list(list_of_disciplines)]
    await update.message.reply_text(
        "Почнемо створювати новий тест!\n"
        "Для початку, з якої дисципліни буде тест?\n"
        'Якщо бажаєте відмінити створення, то введіть /cancelcreate.',
        reply_markup=ReplyKeyboardMarkup(
            list_disciplines, one_time_keyboard=True)
    )
    return 'ENTER_DISCIPLINE'
async def cancel(update, _):
    logger.info('[%s] Creation canceled by user',
                update.message.from_user.username)
    # Delete user data
    userDict.pop(update.message.from_user.id)
    await update.message.reply_text(
        "Створення відмінено. Побачимося наступним разом.",
```

```
        reply_markup=ReplyKeyboardRemove())
    return ConversationHandler.END
async def enter_discipline(update, _):
    user_id = update.message.from_user.id
    userDict[user_id]['discipline'] = update.message.text
    # Asks for type of first question
    list_question = [[el] for el in list(dict_question_types.keys())]
    await update.message.reply_text(
      "Якого типу повинно бути перше питання?\n",
      reply_markup=ReplyKeyboardMarkup(
        list_question, one_time_keyboard=True)
    )
    return 'ENTER_TYPE'
async def enter_type(update, _):
    if update.message.text == "Завершити":
      # User dont want to add more questions
      # Asks for randomness
      await update.message.reply_text(
        "Чи потрібно відображати питання випадковим чином?",
        reply_markup=ReplyKeyboardMarkup(
          [['Так', 'Ні']], one_time_keyboard=True)
      )
      logger.info('[%s] Completed question creation',
                  update.message.from_user.username)
      return 'ENTER_RANDOMNESS_TEST'
    # Save question type
    user_id = update.message.from_user.id
    userDict[user_id]['questtype'] = dict_question_types[update.message.text]
    await update.message.reply_text("Як звучить питання?")
    return 'ENTER_QUESTION'
async def enter_question(update, _):
```

```
    # Save question in userdict
    user_id = update.message.from_user.id
    userDict[user_id]['question'] = update.message.text
    logger.info('[%s] Entered new question type "%s"',
            update.message.from_user.username, update.message.text)
    if userDict[user_id]['questtype'] == ComplianceCoice:
      reply_text = "Введіть значення розділені ', ' до яких будуть будуватися відповідності" ##
      await update.message.reply_text(reply_text)
      return 'ENTER_COMPLIANCE'
    # Ask for correct answer in different ways
    if      userDict[user_id]['questtype'] == SingleChoice:
      reply_text = "Введіть одну правильну відповідь"
    elif    userDict[user_id]['questtype'] == MultipleChoices:
      reply_text = "Введіть правильні відповіді, які розділені ', '"
    else:
      reply_text = ""
    await update.message.reply_text(reply_text)
    return 'ENTER_ANSWER'
  async def enter_compliance(update, _):
    user_id = update.message.from_user.id
    userDict[user_id]['compliance'] = update.message.text
    logger.info('[%s] Entered compliances "%s"',
            update.message.from_user.username, update.message.text)
    reply_text = "Введіть правильну послідовність, значення якої розділені ', '"
    await update.message.reply_text(reply_text)
    return 'ENTER_ANSWER'
  async def enter_answer(update, _):
    user_id = update.message.from_user.id
    # Save correct answer in userDict
    userDict[user_id]['answer'] = update.message.text
    # Try to init question instance
    QuestionType = userDict[user_id]['questtype']
```

```
try:
    userDict[user_id]['questionInstance'] = QuestionType(userDict[user_id]['question'],
                                                         userDict[user_id]['answer'])
except AssertionError:
    await update.message.reply_text(
        "Щось пішло не так. Спробуйте ще раз")
    logger.info('[%s] Entering correct answer "%s" failed',
                update.message.from_user.username, update.message.text)
    return 'ENTER_ANSWER'
logger.info('[%s] Entering correct answer "%s" accepted',
            update.message.from_user.username, update.message.text)

if isinstance(userDict[user_id]['questionInstance'], ComplianceCoice):
    userDict[user_id]['questionInstance'].add_complaince( userDict[user_id]['compliance'])

if isinstance(userDict[user_id]['questionInstance'], MultipleChoices):
    # If MultipleChoices instance, ask for additional possible answers
    await update.message.reply_text(
        "Введіть неправильні відповіді розділені', '")
    return 'ENTER_POSSIBLE_ANSWER'

# Add question to test
userDict[user_id]['test'].add_question(
    userDict[user_id]['questionInstance'])

# Asks for type of next question
list_question = [[el] for el in list(dict_question_types.keys())]
await update.message.reply_text(
    "Якого типу повинно бути наступне питання? "
    "Якщо більне питання не потрібно, натисніть 'Завершити'.",
    reply_markup=ReplyKeyboardMarkup(
        list_question + [['Завершити']], one_time_keyboard=True)
```

```
    )
    return 'ENTER_TYPE'


async def enter_possible_answer(update, _):
    user_id = update.message.from_user.id
    list_possible_answers = update.message.text.split(', ')
    # Add possible answers to question
    for answer in list_possible_answers:
        userDict[user_id]['questionInstance'].add_possible_answer(answer)

    logger.info('[%s] Entered additional possible answers',
                update.message.from_user.username)

    # Ask for
    await update.message.reply_text(
        "Чи потрібно відображати відповіді випадковим чином?",
        reply_markup=ReplyKeyboardMarkup(
            [['Так', 'Ні']], one_time_keyboard=True)
    )

    return 'ENTER_RANDOMNESS_QUESTION'


async def enter_randomness_question(update, _):
    user_id = update.message.from_user.id

    # Check for correct input
    if not update.message.text in ('Так', 'Ні'):
        await update.message.reply_text(
            "Будь ласка, виберіть 'Так' або 'Ні'"
            "Чи потрібно відображати питання випадковим чином?",
```

```
        reply_markup=ReplyKeyboardMarkup(
            [['Так', 'Ні']], one_time_keyboard=True)
    )
    return 'ENTER_RANDOMNESS_QUESTION'

userDict[user_id]['questionInstance'].is_random = update.message.text == 'Так'
logger.info('[%s] Entered randomness of the order of possible answers',
            update.message.from_user.username)

# Add question to test
userDict[user_id]['test'].add_question(
    userDict[user_id]['questionInstance'])
logger.info('[%s] Added the question to the test',
            update.message.from_user.username)

# Asks for type of next question
list_question = [[el] for el in list(dict_question_types.keys())]
await update.message.reply_text(
    "Якого типу повинно бути наступне питання? "
    "Якщо більне питання не потрібно, натисніть 'Завершити'.",
    reply_markup=ReplyKeyboardMarkup(
        list_question + [['Завершити']], one_time_keyboard=True)
)
return 'ENTER_TYPE'


async def enter_randomness_test(update, _):
    user_id = update.message.from_user.id

    # Check for correct input
    if not update.message.text in ('Так', 'Ні'):
        await update.message.reply_text(
```

```
        "Будь ласка, виберіть 'Так' або 'Ні'
        "Чи потрібно відображати питання випадковим чином?",
        reply_markup=ReplyKeyboardMarkup(
            [['Так', 'Ні']], one_time_keyboard=True)
    )
    return 'ENTER_RANDOMNESS_TEST'

    # Process input
    userDict[user_id]['test'].is_random = update.message.text == 'Так'

    # Ask for displaying result after question
    await update.message.reply_text(
        "Чи потрібно відображати результат відповіді після питання?",
        reply_markup=ReplyKeyboardMarkup(
            [['Так', 'Ні']], one_time_keyboard=True)
    )

    return 'ENTER_RESULT_AFTER_QUESTION'


async def enter_result_after_question(update, _):
    user_id = update.message.from_user.id

    # Check for correct input
    if not update.message.text in ('Так', 'Ні'):
        await update.message.reply_text(
            "Будь ласка, виберіть 'Так' або 'Ні'
            "Чи потрібно відображати результат відповіді після питання?",
            reply_markup=ReplyKeyboardMarkup(
                [['Так', 'Ні']], one_time_keyboard=True)
        )
        return 'ENTER_RESULT_AFTER_QUESTION'
```

```
    # Process input
    userDict[user_id]['test'].show_results_after_question = update.message.text == 'Так'

    # Ask for displaying result of every question after test
    await update.message.reply_text(
      "Чи потрібно відображати результат відповідей після проходження тесту?",
      reply_markup=ReplyKeyboardMarkup(
        [['Так', 'Ні']], one_time_keyboard=True)
    )

    return 'ENTER_RESULT_AFTER_TEST'


async def enter_result_after_test(update, _):
    user_id = update.message.from_user.id

    # Check for correct input
    if not update.message.text in ('Так', 'Ні'):
      await update.message.reply_text(
        "Будь ласка, виберіть'Так' або 'Ні"
        "Чи потрібно відображати результат відповідей після проходження тесту?",
        reply_markup=ReplyKeyboardMarkup(
          [['Так', 'Ні']], one_time_keyboard=True)
      )
      return 'ENTER_RESULT_AFTER_TEST'

    # Process input
    userDict[user_id]['test'].show_results_after_test = update.message.text == 'Так'

    # Ask for name of test
    await update.message.reply_text(
```

```
        "Тест створено! \nЯк його назвати?"
    )

    return 'ENTER_TEST_NAME'


async def enter_test_name(update, context):
    logger.info('[%s] Completed test creation',
                update.message.from_user.username)
    user_id = update.message.from_user.id
    testname = update.message.text

    # Bot is typing during database query
    await context.bot.send_chat_action(
        chat_id=update.effective_message.chat_id, action=constants.ChatAction.TYPING)

    # Query for question with input name
    user_col = db[userDict[user_id]['discipline']]
    if not user_col.find_one({'testname': testname}) is None:
        # Test with testname already exists
        await update.message.reply_text(
            "Ця назва {} вже зайнята \nСпробуйте іншу".format(
                testname)
        )
        logger.info('[%s] Test with name "%s" already exists',
                    update.message.from_user.username, update.message.text)

        return 'ENTER_TEST_NAME'

    # Insert Test with testname in database
    user_col.insert_one(
        {'testname': testname, 'testinstance': pickle.dumps(userDict[user_id]['test'])})
```

```
    await update.message.reply_text(
        "Чудово! Новий тест збережено."
        "Тепер здобувачи можу пройти його за назвою {}.".format(testname),
        reply_markup=ReplyKeyboardRemove()
    )
    logger.info('[%s] Test saved as "%s"',
                update.message.from_user.username, update.message.text)
    # Delete user data
    userDict.pop(update.message.from_user.id)

                                                    return ConversationHandler.END
```

ДОДАТОК В

Програмний код чат-боту(модуль edit_test)

```
"""
Module with methods to rename and remove a test with a telegram bot
"""
import logging
import pymongo
from telegram import constants, ReplyKeyboardMarkup
from telegram.ext import ConversationHandler

# user data
user_dict = dict()

logging.basicConfig(format='%(asctime)s - %(name)s - %(levelname)s - %(message)s',
                    level=logging.INFO)
logger = logging.getLogger(__name__)

MONGODB =
    "mongodb+srv://Devi:qwerty123456@cluster0.sq1kkvb.mongodb.net/Otsinyuvach?retryWrites=t
    rue&w=majority"

db = pymongo.MongoClient(MONGODB).Otsinyuvach.tests

list_of_disciplines = {
    'Програмування',
    'Математика',
    'Економіка',
    'Фізика'
}

async def start_remove(update, _):
    """Start a process to remove a test."""
```

```
    logger.info('[%s] Removing process initialized',
                update.message.from_user.username)

    list_disciplines = [[el] for el in list(list_of_disciplines)]
    await update.message.reply_text(
       "З якої дисципліни ти хочеш видалити тест?\n",
       reply_markup=ReplyKeyboardMarkup(
          list_disciplines, one_time_keyboard=True)
    )

    return 'ENTER_DISCIPLINE_REMOVE'


async def enter_discipline_remove(update, _):
    user_dict['discipline'] = update.message.text

    await update.message.reply_text(
        "Який тест ти хочеш видалити? 🙂"
       )
    return 'ENTER_NAME'

async def enter_name_remove(update, context):
    """Deltes a test after entering its' name."""

    discipline =  user_dict['discipline']
    test_name = update.message.text

    await context.bot.send_chat_action(
       chat_id=update.effective_message.chat_id, action=constants.ChatAction.TYPING)
    user_col = db[discipline]
```

```
    # Checks if the test exists
    if user_col.find_one({'testname': test_name}) is None:
        logger.info('[%s] Entered test %s doesn\'t exist',
                    update.message.from_user.username, test_name)
        await update.message.reply_text(
            "Тест '{}' не існує. \nСпробуй ще раз, або відміни видалення вписавши
        /cancelEdit".format(
                test_name)
        )
        return 'ENTER_NAME'

    # Deletes the test
    user_col.delete_one({'testname': test_name})
    logger.info('[%s] Removed %s',
                update.message.from_user.username, test_name)
    await update.message.reply_text(
        "Тест '{}' видалено".format(test_name)
    )
    return ConversationHandler.END


async def start_rename(update, _):
    """Starts a process to rename a test."""
    logger.info('[%s] Renaming process initialized',
                update.message.from_user.username)

    list_disciplines = [[el] for el in list(list_of_disciplines)]
    await update.message.reply_text(
        "З якої дисципліни ти хочеш перейменувати тест?\n",
        reply_markup=ReplyKeyboardMarkup(
            list_disciplines, one_time_keyboard=True)
    )
```

```
    return 'ENTER_DISCIPLINE_RENAME'


async def enter_discipline_rename(update, _):
    user_dict['discipline'] = update.message.text

    await update.message.reply_text(
      "Який тест ти хочеш перейменувати?"
    )
    return 'ENTER_OLD_NAME'

async def enter_old_name(update, context):
    """After entering the old test name, it asks for the new one."""

    discipline =  user_dict['discipline']
    old_test_name = update.message.text

    await context.bot.send_chat_action(
      chat_id=update.effective_message.chat_id, action=constants.ChatAction.TYPING)
    user_col = db[discipline]

    # Checks if a test with this name exists
    if user_col.find_one({'testname': old_test_name}) is None:

      logger.info("[%s] Entered old test '%s' doesn\'t exist",
            update.message.from_user.username, old_test_name)
      await update.message.reply_text(
        "Тест '{}' не існує. \nСпробуй ще раз, або відміни перейменування вписавши
 /cancelEdit".format(
          old_test_name)
      )
```

```
        return 'ENTER_OLD_NAME'

    logger.info("[%s] Entered old test name '%s'",
                update.message.from_user.username, old_test_name)
    # Saves the new test name
    user_dict[discipline] = old_test_name
    await update.message.reply_text(
        "Яка буде нова назва?"
    )
    return 'ENTER_NEW_NAME'


async def enter_new_name(update, context):
    """After entering the new name of the test, it renames it."""

    discipline = user_dict['discipline']
    new_test_name = update.message.text

    await context.bot.send_chat_action(
        chat_id=update.effective_message.chat_id, action=constants.ChatAction.TYPING)
    user_col = db[discipline]

    # Check if a test with the name already exists
    if not user_col.find_one({'testname': new_test_name}) is None:

        logger.info("[%s] Entered new test '%s' already exists",
                    update.message.from_user.username, new_test_name)
        await update.message.reply_text(
            "Тест '{}' вже існує \nСпробуй ще раз, або відміни перейменування вписавши
 /cancelEdit".format(
                new_test_name)
        )
```

```
        return 'ENTER_NEW_NAME'

    # Get old testname and update database
    old_test_name = user_dict[discipline]
    user_col.update_one({'testname': old_test_name}, {
                "$set": {"testname": new_test_name}})
    await update.message.reply_text(
        "Тест '{}' перейменовано на '{}'".format(old_test_name, new_test_name)
    )
    logger.info("[%s] Updated test '%s' to '%s'",
            update.message.from_user.username, new_test_name, old_test_name)

    # delete user data
    user_dict.pop(discipline)
    return ConversationHandler.END


async def cancel_edit(update, _):
    """Cancels the process of deletion or renaming."""
    await update.message.reply_text(
        "Зміни відмінено."
    )
    logger.info("[%s] Canceled editing process by user",
            update.message.from_user.username)

    # delete user data
    user_dict.pop(update.message.from_user.username, None)
    return ConversationHandler.END
```

ДОДАТОК Г

Програмний код чат-боту(модуль attempt_test)

```
import logging
import random
import pickle
import pymongo
from telegram.ext import ConversationHandler
from telegram import ReplyKeyboardMarkup, ReplyKeyboardRemove, constants
from classes.questions import SingleChoice, MultipleChoices, SequenceChoice, ComplianceCoice
from classes.attempt import Attempt

logging.basicConfig(format='%(asctime)s - %(name)s - %(levelname)s - %(message)s',
                    level=logging.INFO)
logger = logging.getLogger(__name__)

MONGODB =
"mongodb+srv://Devi:qwerty123456@cluster0.sq1kkvb.mongodb.net/Otsinyuvach?retryWrites=true
&w=majority"

db = pymongo.MongoClient(MONGODB).Otsinyuvach.tests

# Dict to store user data like an attempt instance
userDict = dict()

list_of_disciplines = {
   'Програмування',
   'Математика',
   'Економіка',
   'Фізика'
}
```

```
async def start(update, _):
    logger.info('[%s] Attempt initialized', update.message.from_user.username)

    if update.message.from_user.id in userDict:
        # user is in the middle of a test and can't attempt a second one
        logger.info('[%s] Attempt canceled because the user is in the middle of a test.',
                    update.message.from_user.username)
        await update.message.reply_text(
            "Ви посеред тесту. Ви не можеш почати інший тест\n"
            'Якщо бажаєте відмінити проходження, напишіть /cancelattempt.'
        )
        return ConversationHandler.END

     # Asks for type of discipline
    list_disciplines = [[el] for el in list(list_of_disciplines)]
    await update.message.reply_text(
        "З якої дисципліни ви бажаєте пройти тест?\n",
        reply_markup=ReplyKeyboardMarkup(
            list_disciplines, one_time_keyboard=True)
    )

    return 'ENTER_DISCIPLINE'


async def cancel(update, _):

    logger.info('[%s] Attempt canceled by user',
                update.message.from_user.username)

    # Remove all user data
    if update.message.from_user.id in userDict:
        userDict.pop(update.message.from_user.id)
```

```
    await update.message.reply_text(
        "Я відмінив вашу спробу. Побачимося пізніше.",
        reply_markup=ReplyKeyboardRemove())
    return ConversationHandler.END


async def enter_discipline(update, _):

    logger.info('[%s] Discipline entered', update.message.from_user.username)

    user_id = update.message.from_user.id
    userDict[user_id] = {"asd":0}
    userDict[user_id]['discipline'] = update.message.text

    await update.message.reply_text(
        'Який тест ви бажаєте пройти?\n'
        'Введіть назву тесту.',
        reply_markup=ReplyKeyboardRemove())

    return 'ENTER_TEST'

async def enter_test(update, context):

    logger.info('[%s] test "%s" entered',
                update.message.from_user.username, update.message.text)

    user_id = update.message.from_user.id

    # name of the test is the first word, the creator the from_user by default
    testname = update.message.text
    discipline = userDict[user_id]['discipline']
```

```
# Bot is typing during database query
await context.bot.send_chat_action(
  chat_id=update.effective_message.chat_id, action=constants.ChatAction.TYPING)

# tests created by the entered user
user_col = db[discipline]
# Looking for testname in the database
test_dict = user_col.find_one({'testname': testname})
if test_dict is None:
  # couldnt find the test
  await update.message.reply_text(
    "Я не зміг знайти тест '{}' Спробуйте іншу назву.".format(
      testname)
  )
  logger.info('[%s] Couldnt find test %s',
        update.message.from_user.username, testname)
  return 'ENTER_TEST'

logger.info('[%s] Found test %s',
      update.message.from_user.username, testname)
# if a test was found, load it and creates an attempt
loaded_test = pickle.loads(test_dict['testinstance'])
userDict[user_id] = Attempt(loaded_test)
await update.message.reply_text(
  "Давайте почнемо! Ось перше питання з тесту '{}'!\n\
   Ви можете вийти тест написавши /cancelattempt.".format(testname)
)

# Asks first question
await ask_question(update)
return 'ENTER_ANSWER'
```

```
async def enter_answer(update, _):

    user_id = update.message.from_user.id
    user_message = update.message.text
    act_question = userDict[user_id].act_question()


    if user_message == 'Завершити':
        logger.info('[%s] Stop entering answers', update.message.from_user.username)

    elif (type(act_question) is MultipleChoices) or (type(act_question) is SequenceChoice) and
user_message != 'Завершити':
        # the current question is a multiple-choice question and not ready to enter

        logger.info('[%s] Insert Answer "%s", Looking for additional answers',
                    update.message.from_user.username, user_message)

        userDict[user_id].input_answer(user_message)
        # wait for next answer
        return 'ENTER_ANSWER'

    elif not type(act_question) is MultipleChoices or SequenceChoice:

        logger.info('[%s] Insert Answer "%s"',
                    update.message.from_user.username, user_message)

        # add answer to list of users' answers
        userDict[user_id].input_answer(user_message)


    # enter the answer of user
```

```
try:
    is_correct, correct_answer = userDict[user_id].enter_answer()
except AssertionError:
    userDict[user_id].user_answers.clear()
    logger.info("[%s] Something went wrong by entering the answer.",
                update.message.from_user.username)
    await update.message.reply_text(
        "Щось пішло не так при вводі відповіді. Сробуйте ще раз.")
    return 'ENTER_ANSWER'

logger.info('[%s] Entered Answer', update.message.from_user.username)

if userDict[user_id].test.show_results_after_question:
    # If creator of the test wants the user to see him/her results after the question
    if is_correct:
        await update.message.reply_text("Відповідь правильна.")
    else:
        await update.message.reply_text(
            "Відповідь не правильна. \nПравильна відповідь була: {}".format(correct_answer))

if userDict[user_id].has_next_question():
    # check for next question
    await ask_question(update)
    return 'ENTER_ANSWER'

# no question left
await update.message.reply_text(
    "Тест пройдено!", reply_markup=ReplyKeyboardRemove())
if userDict[user_id].test.show_results_after_test:
    # If creator of the test wants the user to see him/her results after the test
    count = 1
    for is_correct, question in userDict[user_id].user_points:
```

```
            await update.message.reply_text(
                "Питання {}:\n".format(count)
                + question.question + "\n"
                "Ваша відповідь " +
                ("правильна" if is_correct else "не правильна \nПравильна відповідь була: {}".format(
                    question.correct_answer)),
                reply_markup=ReplyKeyboardRemove())
            count = count + 1

    # Deletes the users entries to closes the attempt
    del userDict[update.message.from_user.id]
    logger.info('[%s] Quitting test', update.message.from_user.username)
    return ConversationHandler.END


async def ask_question(update):
    """
    Formats the keyboard and prints the current question.
    """
    user_id = update.message.from_user.id
    act_question = userDict[user_id].act_question()

    if isinstance(act_question, SingleChoice):
        logger.info('SingleChoice')
        # Single choice question: Choose between possible answers buttons
        list_of_answers = [[el] for el in act_question.possible_answers]
        if act_question.is_random:
            # Shuffle if necessary
            random.shuffle(list_of_answers)
        reply_markup = ReplyKeyboardMarkup(
            list_of_answers, one_time_keyboard=True)
    else:
```

```
    # Single choice question: Choose between possible answers buttons
    list_of_answers = [[el] for el in act_question.possible_answers]
    if act_question.is_random:
        # Shuffle if necessary
        random.shuffle(list_of_answers)
    # add termination button
    list_of_answers.append(['Завершити'])
    reply_markup = ReplyKeyboardMarkup(
        list_of_answers, one_time_keyboard=False)

# print question
if isinstance(act_question, ComplianceCoice):
    reply_text = act_question.question
    for el in act_question.complainces:
        reply_text += "\n" + str(el)

    await update.message.reply_text(
        reply_text,
        reply_markup=reply_markup
    )
else:
    await update.message.reply_text(
        act_question.question,
        reply_markup=reply_markup
    )

logger.info('[%s] Printed new question', update.message.from_user.username)
```

ДОДАТОК Д

Програмний код чат-боту(модуль test)

```
from classes.questions import Question

class Test:
    def __init__(self, author=""):
        self.questions = []
        self.is_random = False
        self.author = author
        self.show_results_after_quiz = True
        self.show_results_after_question = True

    def add_question(self, new_question: Question):
        self.questions.append(new_question)

    def get_questions(self):
    return self.questions.copy()
```

ДОДАТОК Е

Програмний код чат-боту(модуль questions)

```
class Question:

    def __init__(self, question, correct_answer):
        assert question and correct_answer
        self.question = question
        self.correct_answer = correct_answer
        self.user_answer = str()

    def check_solution(self):
        assert self.user_answer

    def enter_solution(self, answer):
        self.user_answer = answer

class MultipleChoices(Question):

    def __init__(self, question, correct_answer):
        assert correct_answer
        super().__init__(question, correct_answer)
        self.is_random = False
        self.possible_answers = correct_answer.split(', ')

    def add_possible_answer(self, new_answer):
        assert not new_answer in self.possible_answers
        self.possible_answers.append(new_answer)

    def check_solution(self):
        super().check_solution()
        return set(self.user_answer.split(", ")) == set(self.correct_answer.split(", "))
```

```
class SingleChoice(MultipleChoices):

    def __init__(self, question="", correct_answer=""):
        assert len(correct_answer.split(", ")) == 1
        super().__init__(question, correct_answer)

    def enter_solution(self, answer):
        assert len(answer.split(", ")) == 1
        return super().enter_solution(answer)


class SequenceChoice(MultipleChoices):

    def check_solution(self):
        super().check_solution()
        return list(self.user_answer.split(", ")) == list(self.correct_answer.split(", "))

class ComplianceCoice(SequenceChoice):
    def __init__(self, question, correct_answer):
        self.complainces = list()
        super().__init__(question, correct_answer)

    def add_complaince(self, new_compl):
        for el in list(new_compl.split(', ')):
            assert not el in self.complainces
            self.complainces.append(el)
```

ДОДАТОК Є

Програмний код чат-боту(модуль attempt)

```
import random
from classes.test import Test

class Attempt:
    def __init__(self, test: Test) -> None:
        self.test = test
        self.questions = test.get_questions()
        self.user_points = list()
        self.user_answers = list()
        if test.is_random:
            random.shuffle(self.questions)

    def has_next_question(self):
        return not (len(self.questions)) == 0

    def act_question(self):
        return self.questions[0]

    def current_question(self):
        return self.questions[0]

    def input_answer(self, user_answer):
        self.user_answers.append(user_answer)
    def enter_answer(self):
        self.current_question().enter_solution(', '.join(self.user_answers))
        self.user_answers.clear()
        return_value = (self.current_question().check_solution(),
                        self.current_question().correct_answer)
        self.user_points.append((return_value[0], self.questions.pop(0)))
       return return_value
```