Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Чат-бот для тестування знань студента»

08-54.БКР.013.00.000 ПЗ

Виконав: студент 4 курсу, групи 1СП-21б
спеціальності 123 – «Комп'ютерна інженерія»
освітня програма – «Системне програмування»
Селівейстр О.В.
(прізвище та ініціали)
Керівник к.т.н. доц. каф. ОТ
Кожем'яко А. В.
(прізвище та ініціали)
Рецензент к.т.н., доцент, доц. каф.ПЗ
Майданюк В. П..
(прізвище та ініціали)

Допущено до захисту
Завідувач кафедри ОТ
д.т.н., проф. Азаров О. Д.
«19» червня 2025 р.

Вінниця ВНТУ – 2025 рік

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Освітньо-кваліфікаційний рівень бакалавр
Галузь знань – 12 «Інформаційні технології»
Спеціальність – 123 «Комп'ютерна інженерія»
Освітньо-професійна програма «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
д.т.н., проф. Азаров О. Д.
«21» березня 2025 року

ЗАВДАННЯ

НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Селівейстру Олександру Володимировичу

1 Тема роботи: «Чат-бот для тестування знань студента», керівник роботи к.пед.н., доц. каф. ОТ Кожем'яко А.В.. затверджені наказом ВНТУ від «20» березня 2025 року №97.

2 Термін подання студентом роботи: 13.05.2025

3 Вихідні дані до роботи: операційна система – Windows 10,середовище розробки PyCharm 2025.1.1, мова програмування Python.

4 Зміст розрахунково-пояснювальної записки: вступ; аналіз та формулювання задачі; проєктування та розробка систем; розробка структурної схеми; реалізація програмного забезпечення; тестування та верифікація роботи бота; висновок; список використаних джерел; додатки, графічна частина.

5 Перелік ілюстративного матеріалу: схема архітектури Telegram-бота для тестування знань студентів, модель діалогу з користувачем, введення особистих даних та ідентифікації груп, результат завершення тесту та використання /results.

6 Консультанти розділів роботи наведені у таблиці 1

Таблиця 1 — Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
Розділ 1-3	к.т.н., доц. каф. ОТ Кожем'яко А. В.	21.03.2025	13.05.2025
Нормоконтроль	ас.. каф. ОТ Швець С. І.	14.05.2025	30.05.2025

7 Дата видачі завдання: 21.03.2025 р.

8 Календарний план виконання наведений в таблиці 2.

Таблиця 2 — Календарний план

№	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Вибір, узгодження та затвердження теми БКР	21.03.2025	22.03.2025	
2	Аналіз літературних джерел та аналогів у сфері онлайн-тестування	23.03.2025	27.03.2025	
3	Постановка задачі, визначення вимог до функціоналу Telegram-бота	28.03.2025	30.03.2025	
4	Проєктування архітектури програмного забезпечення	31.03.2025	04.04.2025	
5	Розробка основного функціоналу бота (реєстрація, вибір предмету)	05.04.2025	09.04.2025	
6	Реалізація системи тестування та обробки відповідей	10.04.2025	17.04.2025	
7	Додавання збереження результатів і команди для перегляду статистики	18.04.2025	20.04.2025	
8	Проведення тестування Telegram-бота, усунення помилок	21.04.2025	27.04.2025	
9	Оформлення пояснювальної записки та графічних матеріалів	28.04.2025	06.05.2025	
10	Нормоконтроль та рецензування БКР	07.05.2025	12.05.2025	
11	Попередній захист БКР	13.05.2025	13.05.2025	

Студент Олександр СЕЛІВЕЙСТР

Керівник роботи к.т.н., доц., Андрій КОЖЕМ'ЯКО

АНОТАЦІЯ

УДК 004.42

Селівейстр О.В. «Чат-бот для тестування знань студента». Кваліфікаційна робота бакалавра з фаху 123 «Комп'ютерна інженерія», освітня програма «Системне програмування». Вінниця: ВНТУ, 2025. 75 с.

На укр. мові. Бібліогр.: 20 назв; рис.:19; табл. 2.

Бакалаврська робота Telegram-бота для тестування знань студентів. У роботі проаналізовано існуючі підходи до створення чат-ботів, зокрема, для освітніх потреб. Детально створено процес побудови структури тестування, реалізації діалогу з користувачем, обробки команд, реєстрації користувачів та збереження результатів тестування.

Telegram-бот виконано мовою програмування Python із використанням бібліотеки python-telegram-bot. Бот дозволяє студентам проходити тести з вибором варіантів, перевіряти правильність відповідей у режимі реального часу, а також отримувати зведення результатів. Передбачено авторизацію доступу до пароля тестів через введення. Для збереження структури тестів використано формат JSON. Бот підтримує персоналізовану реєстрацію (ПІБ і група), команду виходу з облікового запису та можливість адміністратора переглядати останні 10 результатів.

Ключові слова: Telegram-бот, тестування знань, Python, чат-бот, JSON, дистанційне навчання.

ABSTRACT

Seliveystr O.V. “Chatbot for testing student knowledge.” Bachelor's qualification project in the field of 123 “Computer Engineering,” educational program “System Programming.” Vinnytsia: VNTU, 2025. 75 p.

In Ukrainian. Bibliography: 20 titles; figs.: 19; tables: 2.

Bachelor's thesis Telegram bot for testing student knowledge. The thesis analyzes existing approaches to creating chatbots, in particular for educational purposes. It examines in detail the process of building a testing structure, implementing dialogue with the user, processing commands, registering users, and saving test results.

The developed Telegram bot is written in the Python programming language using the python-telegram-bot library. The bot allows students to take multiple-choice tests, check the correctness of their answers in real time, and receive a summary of their results. Access to tests is authorized by entering a password. The JSON format is used to save the test structure. The bot supports personalized registration (full name and group), a command to log out of the account, and the ability for the administrator to view the last 10 results.

Keywords: Telegram bot, knowledge testing, Python, chatbot, JSON, distance learning.

ЗМІСТ

ВСТУП

Актуальність теми бакалаврської роботи обумовлена необхідністю підвищення ефективності контролю знань у сучасній освітній системі. Проведення контролю знань є невід'ємним елементом освітнього процесу, а тестування дозволяє швидко оцінити рівень засвоєння матеріалу, виявити слабкі місця та надати зворотний зв'язок як студенту, так і викладачеві. Однак традиційні засоби перевірки, зокрема у паперовому вигляді або за допомогою стаціонарних систем, потребують значних витрат часу і ресурсів. Інтеграція Telegram-платформи із сучасними освітніми технологіями відкриває перспективи для створення доступних, оперативних і ефективних засобів тестування студентів у режимі реального часу.

Аналіз останніх досліджень і публікацій показав, що в останні роки активно розвиваються чат-боти для освітніх цілей на базі платформ Telegram, Discord та подібних месенджерів. У численних наукових і технічних джерелах висвітлюються питання створення інтелектуальних освітніх систем, вдосконалення алгоритмів персоналізованого навчання, а також програмних підходів до автоматизації перевірки знань. При цьому напрям інтеграції Telegram-ботів із системами контролю знань студентів у вищих навчальних закладах залишається порівняно новим і недостатньо опрацьованим.

Мета роботи полягає у розробці та впровадженні системи перевірки знань на базі Telegram-бота, яка забезпечує індивідуальне тестування користувачів, облік результатів, підтримку кількох навчальних дисциплін, а також зручну взаємодію між студентом та адміністратором без необхідності використання окремих веб-платформ чи складних реєстраційних систем.

Для досягнення поставленої мети в роботі вирішуються такі **задачі**:

— аналіз сучасних онлайн-систем і платформ для тестування знань, включаючи рішення на базі чат-ботів;

— порівняння функціональних можливостей наявних освітніх ботів для вибору оптимального підходу до реалізації;

— проєктування архітектури Telegram-бота, яка враховує сценарії взаємодії з користувачем, структуру тестів та облік результатів;

— розробка програмного забезпечення бота з використанням мови програмування Python, Telegram Bot API та структури JSON для зберігання тестових даних;

— реалізація механізмів реєстрації студентів, вибору предмету, проходження тестування з виведенням результатів та збереженням статистики;

— проведення тестування працездатності системи та розробка інструкції користувача.

Об'єктом дослідження є процеси взаємодії між компонентами системи контролю знань студентів засобами чат-ботів.

Предметом дослідження є методи реалізації та використання Telegram-ботів для контролю знань студентів, а також підвищення ефективності дистанційного навчання шляхом спрощення оцінювання.

Методи дослідження включають:

— методи порівняльного аналізу при оцінці ефективності різних платформ для створення освітніх чат-ботів;

— методи системного аналізу для формування вимог до функціональності бота та його логічної структури;

— аналітичний підхід при порівнянні існуючих рішень і доборі технологічного стеку;

— методи проєктування програмних систем;

— експериментальні методи налагодження та тестування.

1 АНАЛІЗ ТА ФОРМУЛЮВАННЯ ЗАДАЧІ

1.1 Аналіз сучасних засобів для тестування знань онлайн

Пандемія COVID-19 у 2020 році змусила багато університетів швидко перейти на онлайн-навчання та оцінювання. Для багатьох це призвело до помітних змін у виді оцінювань у спробі протидіяти відсутності контролю за іспитами, що проводяться онлайн. Хоча ця раптова зміна була руйнівною як для викладачів, так і для студентів, вона призвела до необхідного перегляду мети оцінювання. Онлайн-оцінювання має додаткову перевагу, оскільки дозволяє швидко та конкретно отримувати зворотний зв'язок великим когортам студентів щодо їхньої особистої успішності, дозволяючи студентам зосередитися на своїх слабких сторонах для виправлення. Це має наслідки для покращення як педагогіки, так і ефективності оцінювання великих когорт, де за замовчуванням часто оцінюються базові знання з пам'яті в оцінюванні з вибором відповідей. [3]

Сучасний етап розвитку цифрових реалій помітно трансформував методи навчання, способи засвоєння та контролю отриманих знань. зокрема, стрімко розвиваються системи онлайн-тестування, що враховують швидке оцінювання знань студентів, організацію дистанційного отримання знань, розробку адаптивних тестів та накопичення загальних даних про ефективність. У зв’язку з цим активно розвиваються інструменти, які дозволяють створювати, адмініструвати та аналізувати тести в режимі онлайн. Це, зокрема, такі сервіси, як Google Forms, Moodle, Testportal, Kahoot!, Quizizz, а також програмні рішення на базі Telegram-ботів.

Google Forms є одним із найпоширеніших і найпростішим у використанні інструментом для створення тестів. Цей сервіс дозволяє швидко налаштувати форму, додати запитання з вибором варіантів, короткими або довгими відповідями, а також автоматично підраховувати бали. Попри зручність, цей інструмент має обмежений функціонал: відсутність системи ідентифікації

користувача, неможливість реалізації адаптивних тестів, а також слабкий захист від списування обмежують його застосування в академічному середовищі. Практичне застосування Google Forms зображено на рисунку 1.1.

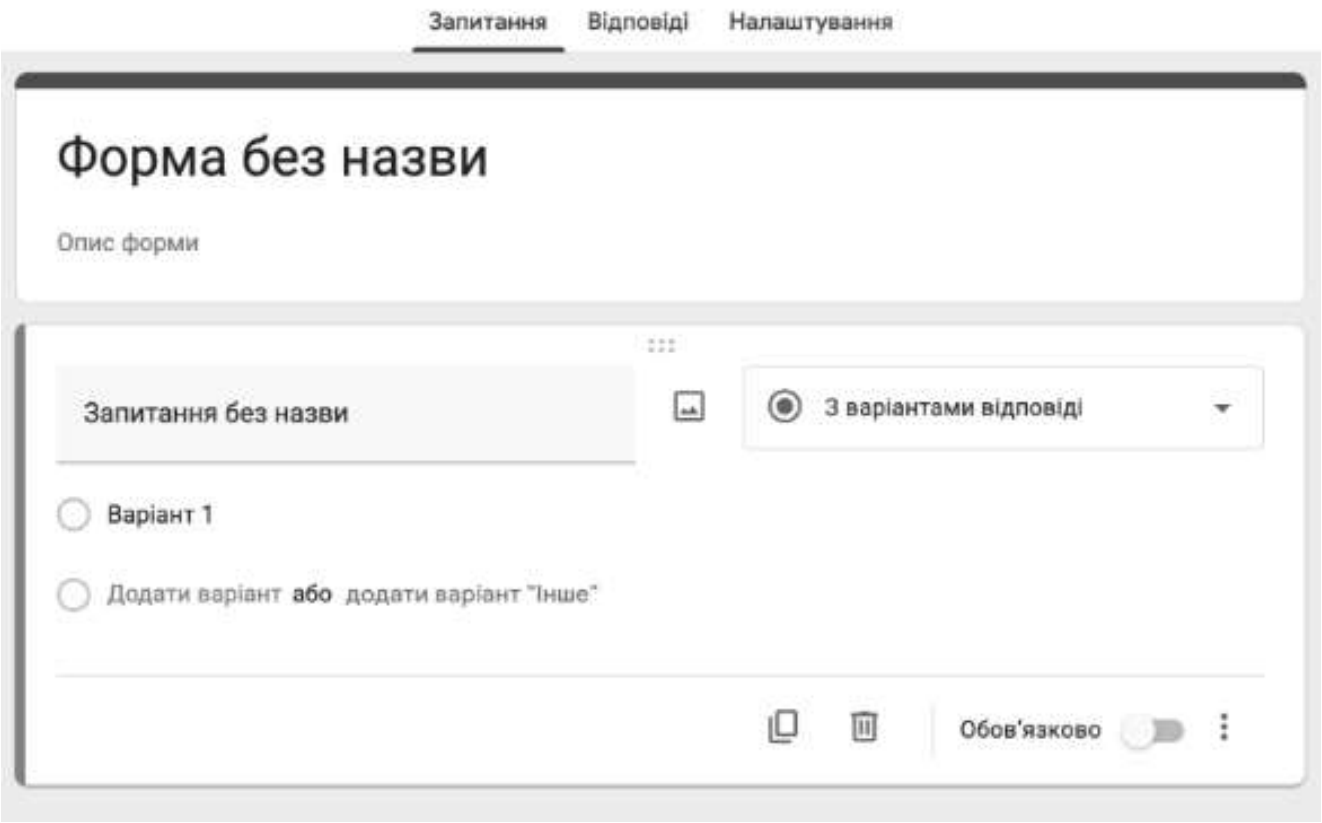

Рисунок 1.1 — Інтерфейс створення тесту в Google Forms

А от Moodle має в собі комплексну систему управління навчальним процесом (LMS), яка забезпечує широкий функціонал для створення курсів, проведення тестів, оцінювання, взаємодії між викладачем і студентом. Moodle підтримує різні типи запитань (множинний вибір, відповідність, числові, відкриті), налаштування рівня складності, таймери, адаптивне тестування, статистику та інтеграцію з базами студентів. Проте її використання залежить від серверного середовища, адміністрування, а інтерфейс для не адаптованих людей може здаватися складним та незрозумілим, який зображений на рисунку 1.2.

Testportal — спеціалізований онлайн-сервіс для перевірки знань, який дозволяє створювати тести з гнучким налаштуванням, обліком часу, переглядом статистики, авторизацією користувачів за email або кодом.

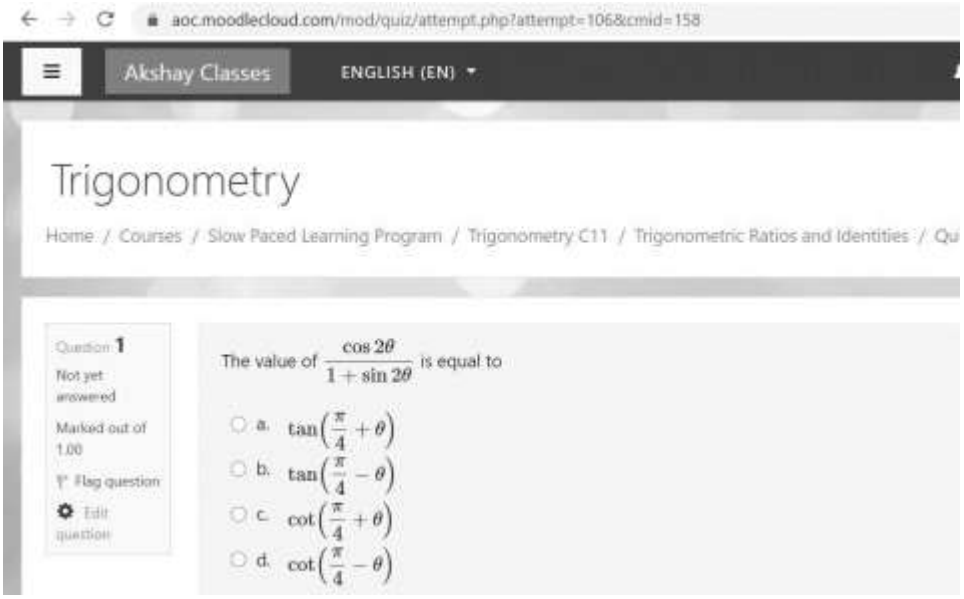

Рисунок 1.2 — Інтерфейс тесту в Moodle

Цей сервіс особливо зручний для навчальних закладів та корпоративного використання. Інтерфейс зображений на рисунку 1.3. Його недоліком є обмежена безкоштовна версія, а також складність у кастомізації логіки тестування.

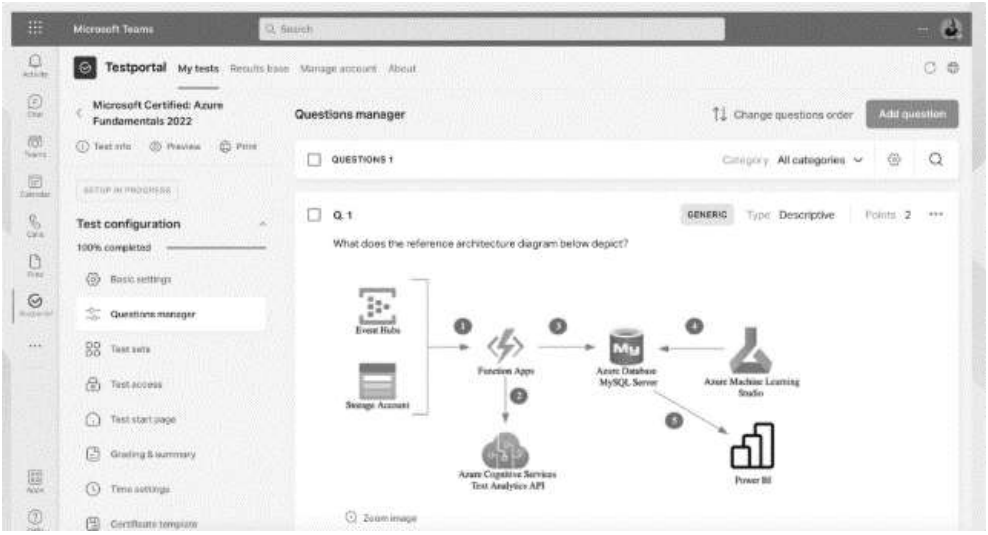

Рисунок 1.3 — Інтерфейс створення тесту в Testportal

Платформи з ігровими елементами, які більше підходять для неформального навчання, шкільної аудиторії або проведення інтерактивних занять називається Kahoot! і Quizizz. Основна ідея — створення вікторин, які користувачі проходять у реальному часі з оцінюванням на швидкість та правильність відповідей. Практичне застосування інтерфейсу тесту та перевірки знань в Kahoot зображено на рисунку 1.4. Такі сервіси добре мотивують учасників, проте не надають можливостей для серйозної аналітики чи тривалого тестування.

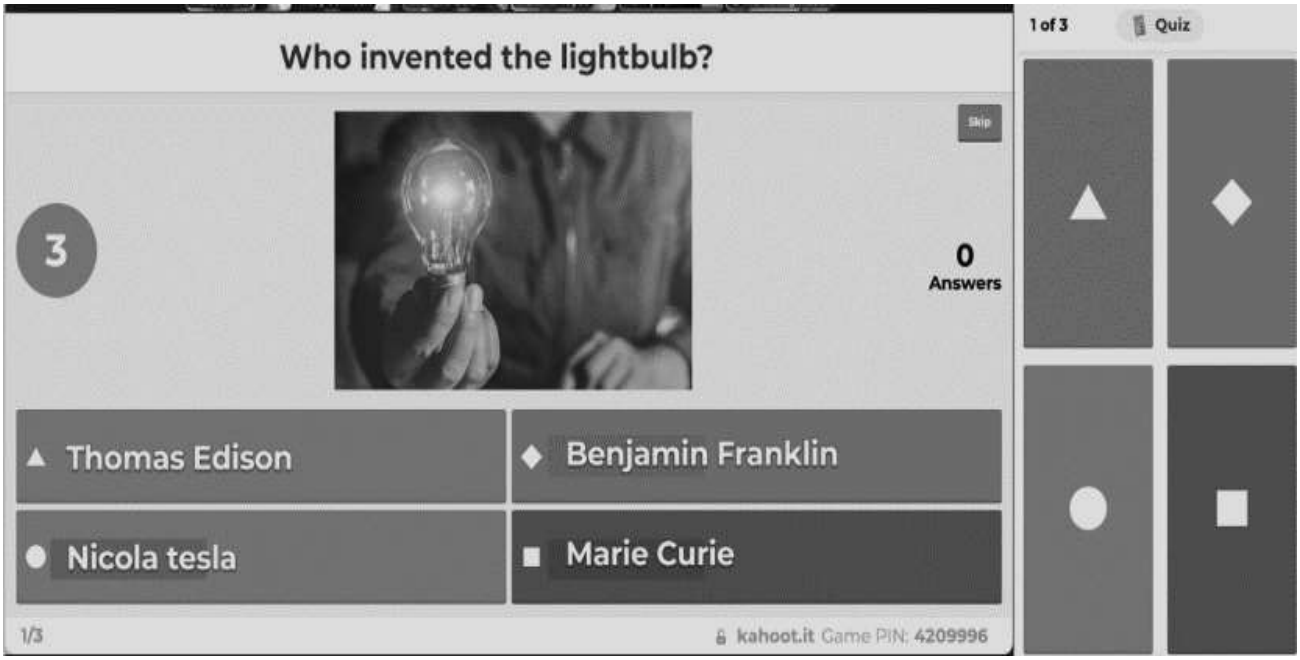

Рисунок 1.4 — Інтерфейс тесту в Kahoot!

Альтернативою традиційним сервісам все частіше стають Telegram-боти — програмні агенти в месенджері Telegram, які дозволяють реалізувати повноцінне тестування знань у зручному чат-інтерфейсі. Такий підхід має низку переваг. По-перше, Telegram-боти доступні на будь-якому пристрої, не потребують встановлення додаткового ПЗ та мають простий інтерфейс спілкування. По-друге, вони можуть забезпечувати персоналізований підхід до користувача, реєстрацію, облік результатів, вибір предметів, введення пароля доступу,

перевірку правильності відповіді в реальному часі, а також зберігання статистики останніх результатів. По-третє, Telegram-боти надають повну гнучкість у побудові логіки тестування: підтримка кількох предметів, покрокове проходження тесту, варіанти відповідей у вигляді кнопок, обробка правильних і неправильних відповідей, формування звітів та підсумкових оцінок.

Для прикладу на рисунку 1.5 представлено Telegram-бота для підготовки до НМТ та ЗНО.

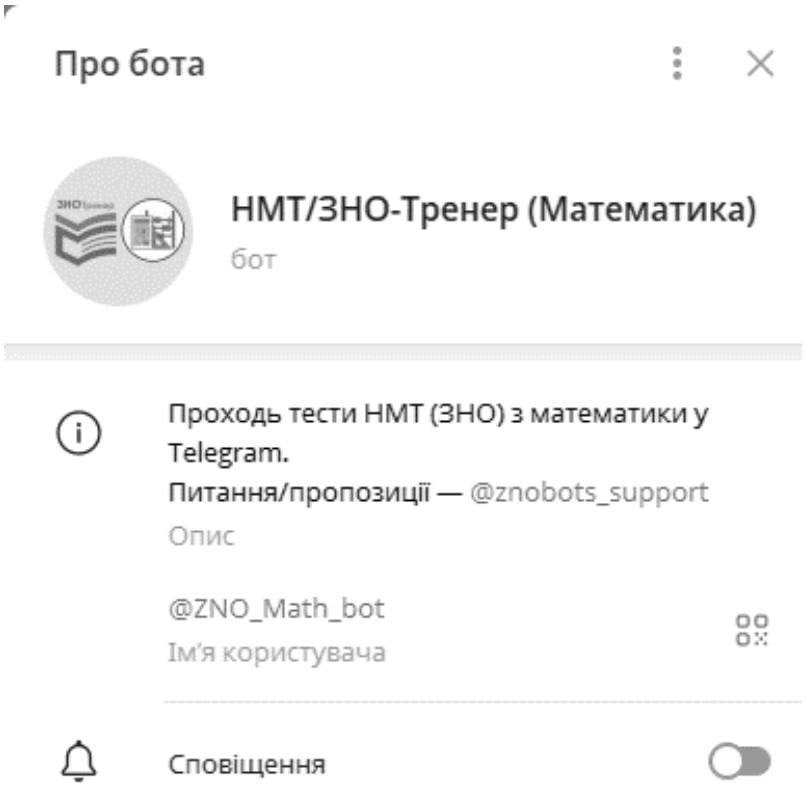

Рисунок 1.5 — Telegram-бот НМТ/ЗНО-Тренер (Математика)

Розробка Telegram-ботів потребує програмування, найчастіше використовується мова Python через зручні бібліотеки (наприклад, python-telegram-bot), але також можливе використання JavaScript, C# або інших мов. Бот може працювати на основі файлів формату JSON, що зручно для зберігання тестових даних, та реалізовувати обробку команд користувача (/start, /help,

/logout, /results тощо). Завдяки webhook або polling-з’єднанням бот може швидко реагувати на дії користувача та змінювати стан взаємодії.

Слід зазначити, що Telegram-боти частково вирішують проблеми, притаманні іншим платформам: зменшують вірогідність проходження тесту іншою особою, дозволяють контролювати логіку доступу до тесту через паролі, забезпечують гнучке формування інтерфейсу для кожного користувача. До того Telegram є звичним середовищем для більшості студентів, що значно спрощує інтеграцію таких ботів у навчальний процес.

Таким чином, у сучасних умовах Telegram-боти демонструють високий потенціал як інструмент для автоматизації онлайн-тестування знань. Їх переваги в гнучкості, зручності та доступності роблять їх ефективною альтернативою класичним системам тестування, особливо в освітніх установах, де важлива персоналізація, мобільність і простота взаємодії зі студентами [4].

1.2 Порівняльний аналіз наявних рішень на базі чат-ботів

Ще у 2016 році, коли чат-боти лише почали набирати популярність, не всі експерти вірили в їхню перспективність. Тоді їх сприймали радше як експериментальне рішення для автоматизації простих дій у месенджерах. Проте з часом ці технології стали доступними для широкого кола користувачів, включно з освітніми установами та невеликими компаніями. Зокрема, популярність таких платформ, як Telegram чи WhatsApp, створила сприятливі умови для впровадження ботів у навчальний процес [5].

Одним з перших прикладів масового впровадження чат-ботів був анонс компанії Facebook щодо використання ботів у Messenger. Тоді йшлося про прості сценарії взаємодії, де користувач міг надсилати текстові команди й отримувати запрограмовані відповіді. Однак із розвитком технологій штучного інтелекту можливості таких систем значно розширилися. Тепер вони дозволяють не лише

відповідати на запити, а й виконувати роль інтерактивного посередника між користувачем та навчальним контентом.

Серед найбільш уживаних ботів у сфері освіти можна виокремити чотири приклади: Quiz Bot, Duolingo Chatbot, TestMe Bot та WhatsApp Education Bot.

Quiz Bot — простий у використанні Telegram-бот, що дозволяє проводити тестування у вигляді вікторин. Він підтримує як одиночні, так і множинні варіанти відповідей, веде статистику та надає адміністраторам можливість налаштовувати базовий сценарій тесту. Проте цей бот не призначений для складних перевірок знань, а функціональність персоналізації чи адаптації навчання в ньому практично відсутня.

Duolingo Chatbot інтегрований у мобільний додаток Duolingo і орієнтований на вивчення іноземних мов. Він імітує розмови, поступово ускладнюючи завдання відповідно до прогресу користувача. Основна увага зосереджена на формуванні мовленнєвих навичок, а не на класичному тестуванні з вибором відповідей. Така модель взаємодії є ефективною для навчання, але не підходить для перевірки рівня знань за окремими темами або модулями.

TestMe Bot також працює на платформі Telegram і більше орієнтований на викладачів чи адміністраторів курсів. У ньому можна створювати структуровані тести, збирати відповіді студентів та формувати звіти. Хоча бот має гнучкі можливості, його використання вимагає певних технічних навичок, а інтерфейс — менш інтуїтивний у порівнянні з іншими аналогами.

WhatsApp Education Bot — рішення, що найчастіше застосовується для поширення навчальних матеріалів та перевірки знань у форматі коротких опитувань. Оскільки WhatsApp має величезну аудиторію, такий підхід забезпечує широку доступність. Проте, з технічної точки зору, функціональність таких ботів часто обмежена можливостями платформи та не дозволяє створювати складні тести або гнучко обробляти дані [6].

Для кращої наочності подано порівняльну таблицю характеристик зазначених рішень. Результати порівняння аналогів зведено в табл. 1.1.

У підсумку варто зазначити, що наявні рішення мають певні сильні сторони, але не є універсальними. Наприклад, Quiz Bot підійде для швидкого опитування, Duolingo Chatbot — для мовного навчання, TestMe Bot — для серйозного освітнього процесу, а WhatsApp Bot — для комунікації та розсилок.

Таблиця 1.1 — Порівняльні характеристики програмних продуктів

Характеристика	Quiz Bot	Duolingo Chatbot	TestMe Bot	WhatsApp Education Bot	Власний чат-бот
Підтримка тестів з вибором відповіді	+	-	+	+	+
Можливість адміністрування	-	-	+	-	+
Підтримка збереження результатів	+	-	+	-	+
Можливість реєстрації	-	-	-	-	+
Простота у використанні	+	+	-	+	+
Сума	**3**	**1**	**3**	**2**	**5**

Однак жоден із розглянутих ботів не охоплює повного функціоналу, необхідного для комплексної перевірки знань у межах вищої освіти: з авторизацією, предметною структурою, збереженням результатів та можливістю керування навчальним процесом з боку викладача.

1.3 Обґрунтування вибору Telegram як платформи для реалізації

У контексті створення ефективної системи оцінювання знань виникла необхідність визначення оптимальної платформи, яка відповідала б ключовим критеріям: зручна взаємодія з користувачем, широке охоплення цільової аудиторії, можливість інтеграції з програмними інтерфейсами та гнучкість у реалізації логіки тестування. Аналіз доступних технологічних рішень, включаючи веб-застосунки, мобільні додатки та платформи обміну повідомленнями, дозволив сформулювати пріоритетні вимоги: спрощене розгортання, доступність для кінцевого користувача та оптимізація ресурсних витрат. На підставі комплексного аналізу було визначено, що імплементація у форматі інтерактивного бота в екосистемі Telegram є найбільш раціональним рішенням.

Месенджер Telegram посідає провідні позиції за показниками популярності як на території України, так і на міжнародному рівні. Відповідно до статистичних даних 2025 року, месенджер налічує понад 900 мільйонів активних користувачів щомісячно, що свідчить про високий рівень проникнення сервісу серед різноманітних демографічних груп. Суттєвою перевагою даної платформи є її мінімальні вимоги до обчислювальних можливостей користувацьких пристроїв, а також підтримка численних операційних систем (Windows, Linux, Android, iOS, macOS). Важливим фактором при виборі став також той факт, що використання розроблюваної системи не потребуватиме інсталяції додаткового програмного

забезпечення, окрім власне месенджера, який вже є звичним інструментом комунікації для переважної більшості студентського контингенту. [7]

З позиції технічної імплементації, Telegram пропонує відкритий прикладний програмний інтерфейс та розвинуту інфраструктуру Telegram Bot Platform, що забезпечує можливість розробки автоматизованих систем різного рівня складності — від базових механізмів реагування до комплексних рішень з розвиненою логікою та персистентним зберіганням інформації. Серед ключових технічних переваг платформи варто відзначити:

– функціональність для створення інтерактивних елементів управління (кнопки, навігаційні меню, спеціалізовані клавіатури);

– архітектурну підтримку обробки callback-запитів, що дозволяє реалізувати багаторівневі сценарії тестування;

– комплексну систему захисту конфіденційності та персональних даних користувачів;

– належний рівень відмовостійкості та можливості горизонтального масштабування сервісу.

Особливої уваги заслуговує відсутність необхідності у проектуванні та розробці власного клієнтського застосунку — бот у месенджері Telegram автоматично стає доступним для будь-якого користувача, який має встановлену відповідну програму. Таким чином, відбувається суттєва економія ресурсів, які в іншому випадку були б спрямовані на розробку та підтримку окремого веб-інтерфейсу або нативного мобільного додатку.

Архітектура Telegram дозволяє ефективно забезпечувати як індивідуальну взаємодію з окремими користувачами, так і підтримку групових режимів роботи, що створює передумови для масштабування системи до рівня цілісного освітнього закладу. Додатковою технічною перевагою є наявність оптимізованого механізму розгортання webhook-інтерфейсу на серверній

інфраструктурі, що гарантує безперервне функціонування автоматизованої системи в режимі реального часу.

Враховуючи сукупність проаналізованих факторів, месенджер Telegram представляє собою оптимальне рішення для імплементації автоматизованої системи перевірки знань, оскільки інтегрує необхідний баланс між простотою використання, розширеним функціоналом, адаптивністю та технічною доцільністю з точки зору процесу розробки та впровадження.

1.4 Вимоги до системи з боку користувача

Ефективність впровадження автоматизованої системи тестування на базі Telegram-платформи безпосередньо залежить від урахування потреб цільової аудиторії. Аналіз функціональних вимог здійснюється з урахуванням двох ключових груп користувачів: студентів як безпосередніх учасників тестувального процесу та викладачів як організаторів контролю академічних досягнень.

Ключовими критеріями для студентів виступають доступність системи, мінімальна складність взаємодії та ефективність процесу тестування. Вибір Telegram як технологічної платформи обґрунтовується його широким розповсюдженням серед молоді та відсутністю необхідності інсталяції спеціалізованого програмного забезпечення.

Функціональні можливості системи для даної категорії користувачів охоплюють процедуру ідентифікації (внесення персональних даних та академічної групи), селекцію навчальної дисципліни, виконання тестових завдань з негайним оцінюванням результатів та можливістю повторного тестування при потребі. Критично важливими аспектами є логічна організація питань, підтримка множинного вибору відповідей та інформативний зворотний зв'язок по завершенні кожного запитання.

Педагогічний персонал потребує гарантованої збереженості результатів тестування та ефективного інструментарію для їх аналізу. У представленій програмній реалізації дані зберігаються у структурованому форматі, що включає персональну інформацію студента, назву дисципліни та кількісні показники успішності. Такий підхід забезпечує оперативність перевірки академічних досягнень та створює основу для статистичного аналізу ефективності навчального процесу.

Важливою складовою є система контролю доступу, реалізована через механізм парольного захисту для кожної навчальної дисципліни. Дана функціональність гарантує обмеження доступу виключно для авторизованих учасників навчального процесу та попереджає можливість несанкціонованого використання тестових матеріалів.

Результуючим чином, проектована система має відповідати принципам ергономічності, універсальної доступності, інформаційної безпеки та функціональної завершеності. Розроблений програмний продукт на базі Telegram-технології відповідає сформульованим критеріям завдяки реалізації механізмів користувацької автентифікації, адаптивної архітектури тестових модулів, алгоритмів верифікації відповідей та системи архівування результатів тестування.

1.5 Психолого-педагогічні аспекти застосування чат-ботів у навчанні

Сучасні тенденції розвитку освітніх систем характеризуються активною інтеграцією цифрових технологій у навчальний процес, що зумовлює необхідність комплексного дослідження їх впливу на психолого-педагогічні аспекти навчання. Впровадження автоматизованих систем тестування, зокрема чат-ботів на базі мессенджерів, відкриває нові перспективи для розвитку індивідуалізованого підходу до контролю знань студентів.

Аналіз психологічних особливостей використання чат-ботів у освітньому процесі виявляє декілька ключових переваг порівняно з традиційними методами контролю знань [8]. Першочергово, неформальний характер інтерфейсу мессенджера створює психологічно комфортне середовище для студентів, що сприяє редукції тестової тривожності та підвищенню об'єктивності результатів оцінювання.

Індивідуалізований характер взаємодії з ботом елімінує негативний вплив соціального порівняння, характерного для групових форм контролю, та створює умови для більш релаксованого стану студента під час тестування. Особливо важливим є аспект миттєвого зворотного зв'язку, який активізує механізми самомоніторингу та формує інтринсивну мотивацію до навчання через усвідомлення власних помилок у реальному часі.

Дослідження показують, що студенти демонструють вищий рівень залученості при використанні інтерактивних цифрових інструментів, що пов'язано з природною адаптацією сучасного покоління до цифрового середовища. Геймифікаційні елементи, притаманні чат-бот інтерфейсам, активізують дофамінергічні механізми винагороди, що позитивно впливає на мотиваційну складову навчального процесу.

З педагогічної перспективи, чат-бот технології забезпечують реалізацію принципів персоналізованого навчання через можливість створення диференційованих тестових завдань для різних категорій студентів. Адаптивність системи дозволяє викладачу конструювати модульні тести відповідно до специфіки навчальної дисципліни та рівня підготовки конкретної академічної групи.

Інтерактивний формат взаємодії забезпечує підвищення рівня студентської активності в процесі оцінювання, що трансформує традиційну пасивну роль студента у процесі тестування. Це сприяє формуванню суб'єкт-

суб'єктних відносин між учасниками освітнього процесу та розвитку критичного мислення студентів.

Важливим педагогічним аспектом є можливість накопичення та аналізу статистичних даних щодо результатів тестування, що дозволяє викладачу відстежувати динаміку засвоєння знань студентами та своєчасно корегувати навчальні програми. Аналітичні функції системи забезпечують об'єктивне оцінювання ефективності викладання та ідентифікацію проблемних аспектів курсу.

Використання Telegram-ботів у навчальному процесі сприяє формуванню цифрової грамотності студентів, що є критично важливим у контексті сучасних вимог ринку праці. Студенти не лише отримують навички роботи з автоматизованими системами, а й розвивають адаптивність до нових цифрових інструментів, що підвищує їх конкурентоспроможність у професійній сфері.

Крім того, робота з чат-ботами формує навички самоорганізації та автономного навчання, оскільки студенти мають можливість самостійно контролювати темп та час проходження тестування. Це сприяє розвитку метакогнітивних навичок та формуванню культури lifelong learning.

Незважаючи на численні переваги, впровадження чат-бот технологій у освітній процес потребує врахування певних обмежень. Необхідність забезпечення технічної підтримки, розробки якісного контенту та навчання викладачів роботі з новими інструментами вимагає значних ресурсних вкладень.

Перспективним напрямком розвитку є інтеграція елементів штучного інтелекту для створення адаптивних тестових систем, здатних автоматично коригувати складність запитань відповідно до рівня знань студента. Це дозволить максимізувати персоналізацію навчального процесу та підвищити ефективність контролю знань.

1.6 Постановка задачі розробки Telegram-бота для тестування знань

Трансформація освітнього ландшафту в умовах четвертої промислової революції зумовлює критичну необхідність інтеграції передових цифрових технологій у процеси академічного оцінювання. Заклади вищої освіти опиняються перед викликом створення інноваційних інструментів контролю знань, які б відповідали вимогам мобільності, ефективності та об'єктивності. Особливого значення в цьому аспекті набуває розробка адаптивних систем тестування, що поєднують технологічну досконалість із педагогічною доцільністю.

Головним завданням даного проекту є проектування та імплементація багатофункціонального бот-додатку в екосистемі Telegram, призначеного для організації комплексної системи академічного тестування студентів різних спеціальностей. Система має забезпечувати інтерактивну взаємодію з користувачами, автоматичну верифікацію відповідей, персистентне зберігання результатів та ефективний контроль доступу до тестових матеріалів.

Досягнення поставленої мети передбачає послідовне вирішення наступних технічних завдань.

Підсистема ідентифікації користувачів — розробка модуля автентифікації з можливістю введення та збереження персональної інформації студента, включаючи повне ім'я та академічну групу приналежності.

Модуль селекції навчальних дисциплін — створення інтерфейсу вибору предметної області тестування з інтегрованою системою парольного захисту для обмеження несанкціонованого доступу до тестових матеріалів.

Система завантаження та парсингу тестових даних — імплементація механізму імпорту структурованих тестових завдань у JSON-форматі з подальшою їх організацією у внутрішньому сховищі системи.

Інтерактивний модуль тестування — розробка алгоритму послідовної презентації запитань користувачеві з реалізацією миттєвого фідбеку щодо правильності наданих відповідей.

Система аналітики та звітності — створення функціоналу накопичення статистичних даних про результати тестування з можливістю їх подальшого перегляду та аналізу.

Інтерфейс користувацьких команд — забезпечення підтримки стандартного набору команд керування для забезпечення зручності навігації в системі.

Критичними аспектами розробки є імплементація алгоритмів обробки некоректних користувацьких вводів, механізмів повторної презентації запитань при необхідності, а також забезпечення надійного та безпечного зберігання результатів тестування. Використання Telegram API як базової платформи елімінує необхідність розробки спеціалізованого користувацького інтерфейсу та системи авторизації, що суттєво спрощує процес інтеграції розробленого рішення в існуючу освітню інфраструктуру.

Архітектурний дизайн проектованої системи характеризується модульністю та масштабованістю, що дозволяє легко адаптувати функціонал під специфічні потреби різних навчальних закладів. Система проектується з урахуванням принципів відокремлення логіки обробки даних від презентаційного рівня, що забезпечує можливість майбутніх модифікацій без кардинальних змін базової архітектури.

Розроблюване рішення має чітко визначену функціональну архітектуру, що базується на аналізі як технічних можливостей сучасних месенджерних платформ, так і педагогічних потреб освітнього процесу. Подальші розділи роботи розкриватимуть принципи архітектурної організації системи, специфіку її функціонування та деталі практичної імплементації розробленого рішення.

2 ПРОЕКТУВАННЯ ТА РОЗРОБКА СИСТЕМ

2.1 Основи побудови Telegram-ботів для онлайн-тестування

Інтерактивні програмні рішення на платформі Telegram представляють собою спеціалізовані автоматизовані агенти, що забезпечують комунікацію з користувачами через текстовий інтерфейс даного месенджера. Відкритий програмний інтерфейс, розвинена система обробки користувацьких дій, можливість асинхронної роботи з повідомленнями та широкий інструментарій для створення елементів взаємодії (різноманітні кнопки, меню, вбудовані клавіатури) зробили Telegram привабливою екосистемою для розробки навчальних застосунків, зокрема платформ для проведення електронного оцінювання знань.

У центрі функціонування кожного автоматизованого агента Telegram знаходиться програмний інтерфейс Telegram Bot API, який забезпечує механізми прийому та опрацювання користувацьких повідомлень, введених команд, відповідей на запити, а також надсилання текстового контенту, мультимедійних матеріалів та інтерактивних елементів керування. Реєстрація такого агента відбувається за посередництвом системного сервісу — [@BotFather], результатом взаємодії з яким є отримання індивідуального ідентифікатора доступу (токена), необхідного для комунікації з програмним інтерфейсом. [9]

При створенні автоматизованого агента на платформі Telegram, спрямованого на проведення дистанційного тестування, раціональним підходом є впровадження асинхронних механізмів опрацювання комунікацій. Це забезпечує оперативну реакцію на активність користувачів, одночасне обслуговування численних сесій оцінювання та скорочення часових затримок у процесі взаємодії. В рамках даного дослідження використовується мова програмування Python із застосуванням бібліотечного модуля python-telegram-bot, що пропонує функціональний інструментарій для конструювання

поведінкових алгоритмів у формі команд, обробників подій зворотного зв'язку та конверсаційних сценаріїв (ConversationHandler).

Автоматизовані агенти Telegram підтримують декілька моделей інформаційного обміну з серверною інфраструктурою Telegram: Polling (циклічне звернення до сервера для отримання актуальних подій) та Webhook (приймання повідомлень про події через HTTP-запити до власного серверного застосунку). У контексті реалізації даної системи було обрано модель Polling як методологічно простішу для етапу розробки та тестування. У перспективі планується трансформація до Webhook-моделі при розгортанні на серверному обладнанні або хмарній інфраструктурі. [10]

2.2 Архітектура Telegram-бота та принципи його роботи

Програмна реалізація системи оцінювання знань студентів здійснена з використанням мови програмування Python у поєднанні зі спеціалізованою бібліотекою python-telegram-bot, яка надає комплексний набір інструментальних засобів для ефективної взаємодії з прикладним програмним інтерфейсом Telegram Bot API. Концептуальна архітектура розробленого програмного рішення базується на асинхронній подієво-керованій моделі із залученням механізмів обробки подій (Handler) та структурованої організації діалогового процесу (ConversationHandler), що забезпечує методологічно обґрунтовану поетапну комунікацію з кінцевим користувачем [11]. Щоб краще розуміти архітектуру даного телеграм-бота у додатку В.1 буде його графічне зображення.

У структурі розробленого програмного комплексу можна виокремити наступні функціональні компоненти:

Система управління користувацькими сесіями: реалізована з використанням інфраструктури ConversationHandler, що дозволяє алгоритмізувати весь процес взаємодії користувача з системою через чітко

детерміновану послідовність етапів — початкова автентифікація з введенням персональних даних (прізвище, ім'я, по батькові та академічна група), селекція навчальної дисципліни, автентифікація через пароль для доступу до відповідного тестового модуля, безпосереднє проходження тестових завдань та системний аналіз наданих відповідей.

У структурі розробленого програмного комплексу можна виокремити наступні функціональні компоненти:

Інтерактивний користувацький інтерфейс: інтеграція адаптивних елементів керування у формі інтерактивних кнопок (InlineKeyboardButton), які забезпечують ергономічний візуальний механізм для вибору навчальної дисципліни та селекції варіантів відповідей у контексті тестових завдань.

Підсистема управління тестовим контентом: імплементована стратегія завантаження та структурованого зберігання тестових матеріалів у форматі JSON, що локалізуються у файлі tests.json. Архітектура даних оптимізована для ефективної класифікації та ідентифікації тестових завдань відповідно до навчальних дисциплін [12].

Модуль валідації користувацьких відповідей: реалізовано алгоритм динамічного співставлення наданої користувачем відповіді з еталонним значенням, яке зберігається в JSON-структурі даних, з подальшим оперативним інформуванням користувача про коректність виконання завдання.

Компонент аналітики та статистики: передбачено механізм тимчасового збереження результатів тестування у контейнері даних results, який містить структуровану інформацію щодо ідентифікації користувача, обраної дисципліни та кількісного показника правильних відповідей. Інтегрована команда /results надає функціональність для відображення аналітичної інформації щодо останніх 10 сесій тестування.

Механізм контролю доступу: впроваджено багаторівневу систему автентифікації, яка передбачає перевірку дисциплінарно-специфічних паролів перед ініціацією тестування, що забезпечує контрольований доступ до навчально-методичних матеріалів.

Допоміжний функціонал: інтегровано сукупність службових команд (/help, /logout, /cancel), які розширюють функціональні можливості системи та підвищують рівень користувацького досвіду.

Технічна експлуатація розробленого бота здійснюється через спеціалізований метод run_polling(), який забезпечує автоматизоване отримання оновлень безпосередньо від серверної інфраструктури Telegram без необхідності конфігурування webhook-інтерфейсу. Такий підхід суттєво оптимізує процес тестування та налагодження програмного комплексу в процесі розробки, а також підвищує показники стабільності та надійності системи [13].

Підсумовуючи, можна констатувати, що розроблена архітектура Telegram-бота характеризується раціональним балансом між структурною простотою та функціональною повнотою. Імплементована система забезпечує комплексну реалізацію всіх ключових етапів процесу тестування знань, одночасно демонструючи високі показники адаптивності, масштабованості та зручності використання як з боку студентського контингенту, так і для професорсько-викладацького складу.

2.3 Модель сценарію взаємодії з користувачем

В процесі розробки автоматизованої системи перевірки знань на базі месенджера Telegram було спроектовано та реалізовано структуровану модель діалогу з користувачем. Методологічною основою для побудови інтерактивної взаємодії обрано механізм керованої конверсації (ConversationHandler), що задається функціональними можливостями бібліотеки python-telegram-bot [14].

Даний підхід дозволив організувати контекстно-залежну обробку користувацьких дій та підтримку стану між окремими повідомленнями. Модель діалогу з користувачем зображено у додатку В.2.

Архітектурно, розроблена модель комунікації представлена у вигляді послідовності взаємопов'язаних станів:

— етап автентифікації користувача
— етап верифікації доступу
— етап проходження тестування
— етап представлення результатів

Реалізовано систему запитів для введення персональних даних (прізвище, ім'я, по батькові та група студента), що забезпечує ідентифікацію суб'єкта навчання та подальшу атрибуцію результатів тестування. Цей етап є фундаментальним для персоналізації всього процесу взаємодії з системою.

Імплементовано з використанням інтерактивних елементів управління (InlineKeyboardMarkup), що інтегруються в інтерфейс месенджера. Такий підхід мінімізує ризик виникнення помилок при введенні та підвищує ергономічність користувацького досвіду.

Впроваджено механізм парольної автентифікації для забезпечення конфіденційності тестових матеріалів. Система здійснює валідацію введеного паролю, асоційованого з конкретною навчальною дисципліною, що дозволяє викладачеві контролювати доступ до тестових завдань.

Центральний компонент діалогової моделі, що забезпечує послідовне представлення тестових запитань, фіксацію відповідей користувача, верифікацію їх правильності та надання миттєвого зворотного зв'язку. Особливістю імплементації є використання інтерактивних кнопок для вибору варіантів відповіді.

Завершальна фаза комунікації, в рамках якої користувачу надається структурована інформація щодо кількісних показників результативності проходження тесту. Одночасно система здійснює збереження отриманих даних для подальшого аналізу викладачем.

Інтеграція вищезазначених етапів реалізована через механізм управління станами, де кожен стан асоціюється з відповідними обробниками подій. Для забезпечення збереження контексту взаємодії використовується контейнер context.user_data, який функціонує як тимчасове сховище для сесійних даних користувача.

Ключовими перевагами розробленої моделі діалогу є:

— інтуїтивність послідовності етапів взаємодії;

— захищений доступ до навчальних матеріалів;

— ефективне управління процесом тестування в режимі реального часу;

— адаптивність до різних освітніх сценаріїв.

2.4 Форматування тестових даних та логіка їх обробки

Значущим аспектом функціонування автоматизованих систем на базі платформи Telegram, що спрямовані на реалізацію контролю рівня знань, є методологія структурування та обробки тестових матеріалів. У контексті розробленої системи нами застосовано підхід, заснований на використанні формату обміну даними JSON (JavaScript Object Notation) для збереження екзаменаційних запитань, що забезпечує необхідну гнучкість при моделюванні ієрархії діагностичних завдань, альтернативних відповідей та еталонних результатів.

Вибір формату JSON як базового механізму персистентності даних обґрунтовується його інтуїтивно зрозумілою структурою для сприйняття людиною, оптимальною інтеграцією з процесами серіалізації та десеріалізації у

середовищі Python, а також потенціалом для взаємодії з сторонніми сервісами при подальшому розвитку архітектури системи [15].

Концептуально кожен діагностичний блок у системі представлено як упорядкований набір елементів, де кожен елемент характеризується трьома основними атрибутивними компонентами:

— question — текстове формулювання діагностичного завдання;

— options — упорядкована множина варіативних відповідей;

— correct — еталонна відповідь (яка співвідноситься з одним із елементів множини options).

Прикладом структурної організації одного діагностичного завдання у файлі конфігурації tests.json

Лістинг 2.1 — Структура одного з питань у файлі tests.json

```
"networks": [
  {
   "question": "Який рівень відповідає за IP-адресацію?",
   "options": ["Канальний", "Мережевий", "Транспортний", "Прикладний"],
   "correct": "Мережевий"
  }
```

2.5 Зберігання та облік результатів тестування

Фіксація і документування підсумків тестування становить критично важливий компонент функціональності розробленої системи, оскільки забезпечує не лише формування інформативних повідомлень для суб'єктів навчання, але й уможливлює акумуляцію аналітичних даних для подальшої експертної обробки. У представленому автоматизованому рішенні на базі месенджера Telegram реєстрація діагностичних показників реалізована через інтегрований динамічний масив, що підтримується в оперативній пам'яті протягом активної фази функціонування програмного застосунку.

По завершенні кожної діагностичної сесії система автоматично документує наступні атрибутивні характеристики:

— персональні ідентифікаційні дані суб'єкта навчання (прізвище, ім'я, по батькові);

— структурна приналежність до навчальної групи;

— обраний предмет;

— кількісні показники успішності (співвідношення коректних відповідей до загального обсягу завдань).

З метою оптимізації використання системних ресурсів та запобігання надлишковій акумуляції інформації, архітектурою системи передбачено обмеження реєстраційного масиву до десяти актуальних записів. При перевищенні встановленого ліміту відбувається автоматичне видалення найменш актуального елемента реєстру через застосування алгоритмічної процедури pop(0). Такий підхід забезпечує збереження релевантної інформації без необхідності імплементації спеціалізованих систем управління базами даних чи зовнішніх сховищ на початковому етапі експлуатації.

Доступ до аналітичної інформації здійснюється через застосування системної інструкції /results. У відповідь на даний запит уповноважений користувач (здебільшого викладач-експерт або адміністративний працівник) отримує структуроване повідомлення з хронологічно впорядкованим переліком останніх діагностичних сесій. Запропонована методика надає можливість оперативного моніторингу динаміки освітнього процесу та ідентифікації учасників, які пройшли відповідні діагностичні процедури.

Враховуючи відносну нескладність структурної організації даних та пріоритетність оперативного доступу, тимчасова фіксація інформації в оперативній пам'яті є методологічно обґрунтованою. Проте в контексті потенційного розширення функціональності системи або необхідності тривалого

збереження статистичних показників доцільним є впровадження механізмів персистентності даних на основі спеціалізованих СУБД, які показані на рисунку 2.1 (зокрема, SQLite або PostgreSQL).

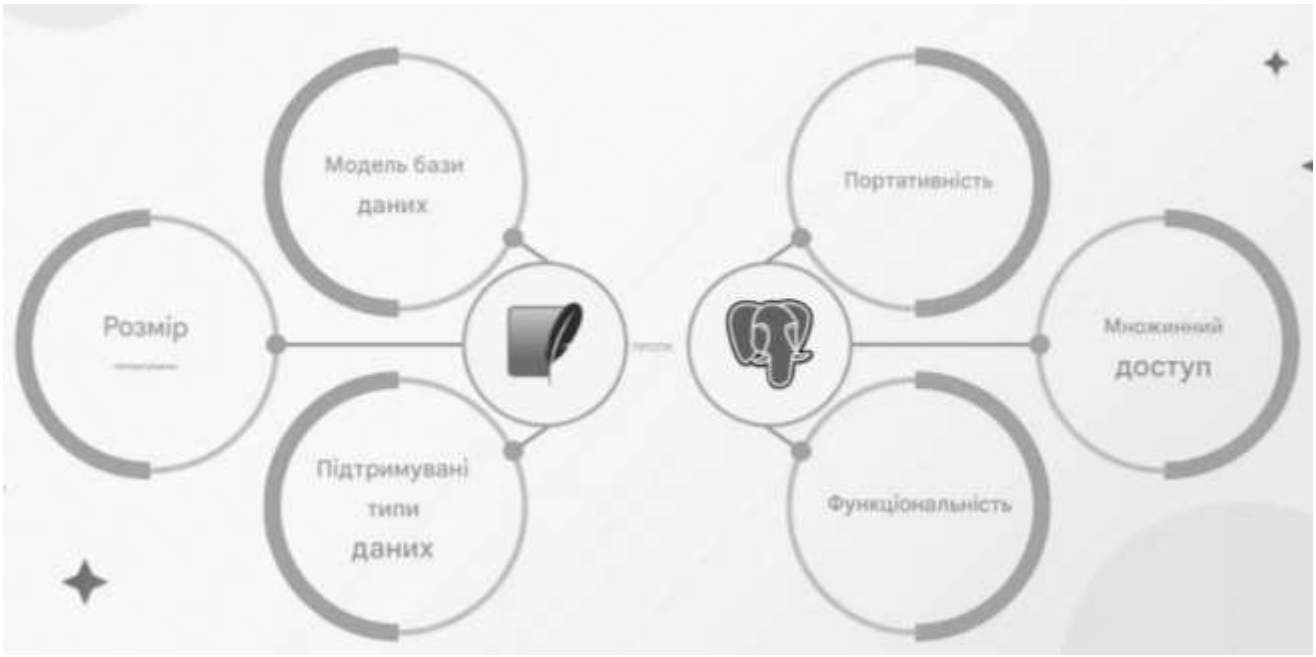

Рисунок 2.1 — Характеристики SQLite та PostgreSQL

Таким чином, імплементований механізм документування експертних оцінок у розробленій системі на базі платформи Telegram забезпечує базовий функціональний рівень, необхідний для ефективного контролю навчальних досягнень, одночасно створюючи архітектурний фундамент для подальшої модернізації та масштабування інформаційної інфраструктури.

3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Обґрунтування вибору мови програмування та бібліотек

Платформа Telegram пропонує розробникам програмного забезпечення два фундаментальні підходи до інтеграції з екосистемою месенджера: безпосереднє конструювання та опрацювання HTTP-запитів до програмного інтерфейсу Telegram Bot API або застосування спеціалізованих програмних компонентів, що забезпечують абстрагування від низькорівневих деталей взаємодії. Другий підхід надає програмістам суттєво вдосконалений інструментарій для конструювання автоматизованих агентів завдяки наявності готових функціональних модулів для опрацювання комунікацій, інтерактивних елементів, діалогових сценаріїв тощо.

Програмний інтерфейс Telegram Bot API являє собою HTTP-орієнтований інтерфейс, що надає можливість створення автоматизованих програмних агентів для комунікаційної платформи Telegram. Він забезпечує функціональні можливості з передачі та отримання текстових повідомлень, обробки зворотних запитів від інтерактивних елементів керування, отримання персональних даних про користувачів та численні інші функції. Для оптимізації процесу розробки використовуються спеціалізовані програмні бібліотеки, реалізовані для різноманітних мов програмування, зокрема Python, JavaScript (Node.js), Java, C# та інших.

В контексті даного дослідження було прийнято рішення імплементувати програмний агент на базі мови Python, що обумовлено рядом технічних та прагматичних аспектів:

Python представляє собою мову програмування високого рівня абстракції з інтуїтивно зрозумілим синтаксисом, що дозволяє концентрувати зусилля на розробці функціональних алгоритмів, а не на деталях технічної імплементації;

Програмна бібліотека python-telegram-bot забезпечує комплексну функціональну обгортку над програмним інтерфейсом Telegram Bot API,

включаючи підтримку як синхронної, так і асинхронної парадигми виконання, механізми опрацювання системних інструкцій, текстових повідомлень, інтерактивних елементів та запитів зворотного зв'язку;

Розширена підтримка інтегрованих середовищ розробки, зокрема Visual Studio Code, PyCharm та Jupyter, забезпечує оптимальні умови для швидкої розробки та верифікації програмного рішення;

Інтегровані механізми взаємодії з форматом JSON, який застосовується для структурування діагностичних матеріалів, надають можливість ефективної обробки значних обсягів структурованої інформації;

Незалежність від операційної платформи, що забезпечує функціонування програмного агента як на індивідуальних серверних комплексах, так і в розподілених хмарних інфраструктурах (наприклад, Microsoft Azure або Heroku).

Обрана програмна бібліотека python-telegram-bot характеризується наявністю активної професійної спільноти, вичерпної технічної документації та підтримкою актуальних версій програмного інтерфейсу Telegram. У представленій розробці реалізовано асинхронну модель виконання, що ґрунтується на модулі asyncio, завдяки чому програмний агент здатен здійснювати паралельну обробку множинних запитів без втрати функціональності [16].

З метою об'єктивного обґрунтування вибору технологічного стеку для розробки автоматизованої системи тестування та демонстрації конкурентних переваг обраного програмного інструментарію, доцільним є проведення детального аналізу основних характеристик Python у контексті створення Telegram-ботів. Для отриманих результатів дослідження та забезпечення наочності представлення ключових технічних аспектів буде наведено таблицю 3.1, яка демонструє функціональні переваги та технологічні особливості над іншими популярними мовами програмування:

Таблиця 3.1 — Порівняльні характеристики програмних продуктів

	Python	C#	Java
Інтеграція з хмарними платформами	+	+	+
Наявність зрозумілої документації	+	+	+
Зручна обробка JSON-даних	+	+	-
Швидкість розробки	+	-	-
Зручна робота з Telegram API	+	-	-
Сума	5	3	2

Таким чином, інтеграція мови програмування Python та бібліотечного модуля python-telegram-bot забезпечила можливість ефективної та оперативної імплементації програмного агента на базі месенджера Telegram для дистанційної оцінки академічних компетентностей здобувачів освіти, з високим потенціалом масштабування та модифікації системи відповідно до еволюції функціональних вимог.

3.2 Налаштування середовища та створення базової структури

Конструювання програмного рішення для дистанційного оцінювання на базі платформи Telegram вимагає формування відповідного інструментального комплексу, що забезпечує ефективне проєктування, верифікацію та експлуатацію програмного коду. В рамках даного дослідження для реалізації функціональних можливостей автоматизованого агента було обрано мову програмування Python,

яка характеризується лаконічним синтаксисом та розгалуженою екосистемою програмних модулів для взаємодії з програмними інтерфейсами, зокрема з Telegram Bot API.

В якості інструментальної платформи для розробки було застосовано інтегроване середовище PyCharm 2025, яке зображене на рисунку 3.1, що надає комплексну підтримку технологій Python, включає розвинену систему діагностики, семантичне підсвічування програмного коду, інтелектуальне доповнення синтаксичних конструкцій та інтегрований термінальний емулятор для взаємодії з віртуалізованими середовищами. Додатково, інструментальна платформа забезпечує оперативний запуск та моніторинг діагностичних повідомлень, що має критичне значення на етапі верифікації програмної системи.

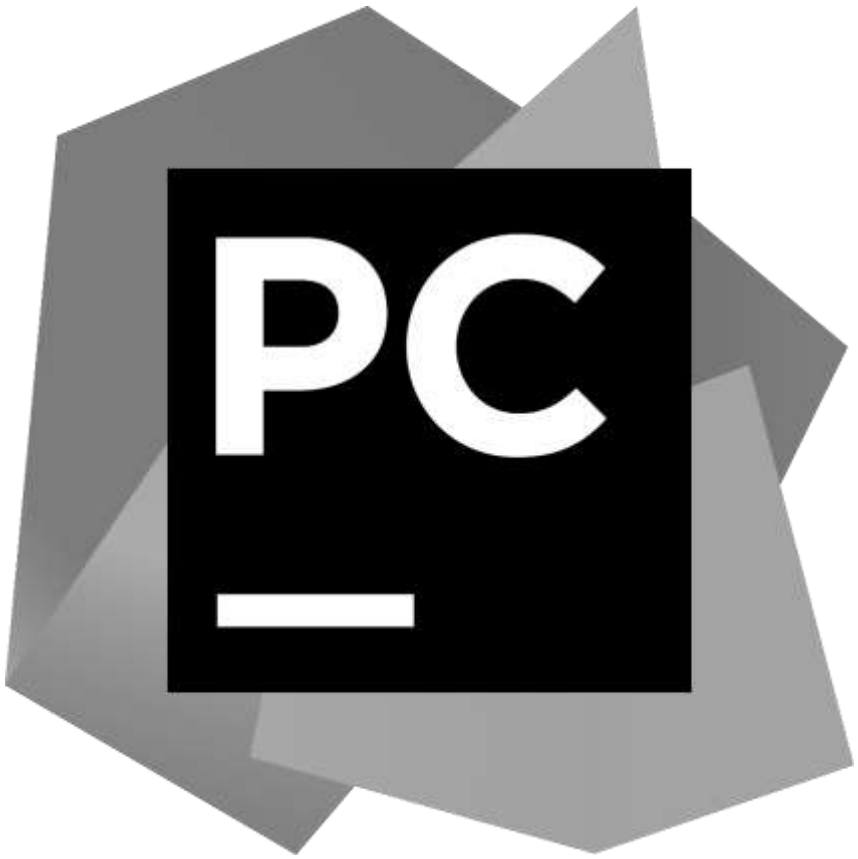

Рисунок 3.1 — Середовище PyCharm 2025

Процес формування інструментального середовища включав послідовність наступних технологічних етапів:

— внсталяція інтерпретатора Python;

— формування ізольованого віртуального середовища;

— впровадження необхідних програмних модулів;

— структурна організація програмного проєкту;

— конфігурація автентифікаційних параметрів програмного агента.

Для проєкту було використано актуальну версію інтерпретатора Python (3.12), що попередньо інстальована в обчислювальну систему та зображена на рисунку 3.2. Верифікація коректності інсталяції здійснювалась через виконання інструкції python --version в термінальному емуляторі.

Рисунок 3.2 — Програмне середовище PyCharm 2025

З метою забезпечення ізоляції програмних залежностей проєкту було сформовано віртуальне середовище за допомогою команди: python -m venv venv

Ініціалізація середовища виконувалась через термінальний емулятор інтегрованого середовища розробки, що дозволило уникнути потенційних конфліктів з глобальними бібліотечними компонентами.

Основним бібліотечним компонентом для інтеграції з Telegram є модуль python-telegram-bot. Його інсталяція здійснювалась через пакетний менеджер pip:

pip install python-telegram-bot

Додатково використовувались стандартні компоненти Python, зокрема модуль json для маніпуляції структурованими даними, та модуль logging (за необхідності) для діагностичних цілей.

У базовому каталозі проєкту було сформовано такі основні файлові компоненти які зображені на рисунку 3.3.

bot.py — імплементує інтерактивну комунікацію з користувачем через механізм ConversationHandler, реалізує функції автентифікації, селекції навчальної дисципліни, валідації парольних даних, проходження діагностичних завдань та аналізу результатів.

tests.json — містить структуровані діагностичні матеріали за кожною навчальною дисципліною. Інформація організована у вигляді асоціативного масиву, де ідентифікатор представляє назву дисципліни, а значення — впорядкований набір завдань з альтернативними варіантами відповідей.

results.json — використовується для персистентності результатів оцінювання у форматі послідовності текстових елементів.

subject_passwords.json — окремий файловий компонент для централізованого зберігання автентифікаційних даних з диференціацією за навчальними дисциплінами. Такий підхід забезпечує можливість модифікації параметрів доступу без втручання в основний програмний код.

users.json — за необхідності може акумулювати додаткову інформацію про зареєстрованих суб'єктів навчання, що оптимізує процедури подальшого аналізу та статистичної обробки даних.

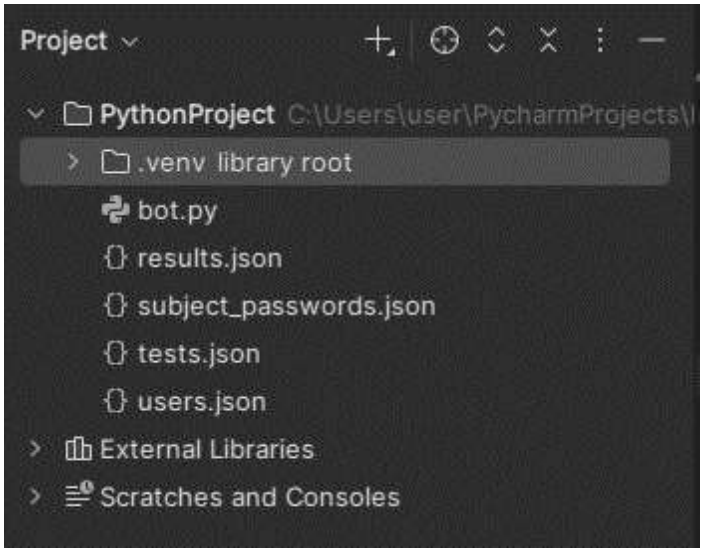

Рисунок 3.3 — Основні файлові компоненти

Унікальний ідентифікатор доступу, отриманий через службову систему BotFather, який зображено на рисунку 3.4, було інтегровано у відповідну програмну змінну у файлі bot.py.

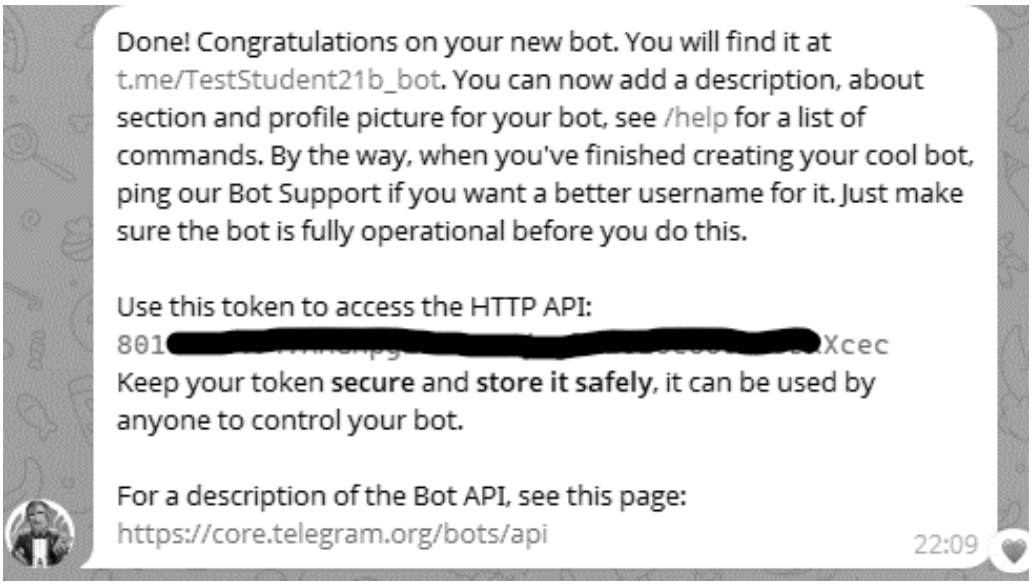

Рисунок 3.4 — Отримання ідентифікатор доступу в BotFather

З метою підвищення рівня інформаційної безпеки на подальших етапах можливе винесення автентифікаційних даних до окремого конфігураційного файлу .env з наступним програмним зчитуванням за допомогою спеціалізованого модуля python-dotenv.

3.3 Реалізація основних команд Telegram-бота (/start, /help, /logout, /results)

Автоматизовані боти на базі платформи Telegram повинні містити набір фундаментальних інструкційних елементів, що забезпечують логічну архітектуру комунікації з кінцевим користувачем та надають інтерфейси доступу до основних функціональних можливостей. У розробленому програмному рішенні було імплементовано та сконфігуровано чотири принципові елементи: /start, /help, /logout, /results, кожен з яких виконує специфічні функції в процесі взаємодії суб'єкта навчання з інтерактивним агент.

Команда /start функціонує як ініціаційний механізм взаємодії з програмним агентом. При активації даного елемента система верифікує наявність ідентифікаційних даних користувача. У випадку, коли персональні дані суб'єкта навчання (прізвище, ім'я, по батькові та академічна група) наявні в контекстному сховищі сесії, система автоматично пропонує селекцію навчальної дисципліни для проходження діагностичної процедури. За умови первинного звернення користувача до системи, ініціюється протокол реєстрації. Дана функціональність реалізована з використанням механізму ConversationHandler, що забезпечує ефективну структуризацію комунікаційних етапів у рамках визначеного сценарію [17].

Лістинг 3.1 — Фрагмент коду метода /start

```
# /start
async def start(update: Update, context: ContextTypes.DEFAULT_TYPE):
    user_data = context.user_data
    if "name" in user_data and "group" in user_data:
```

```
        # Якщо користувач вже зареєстрований — одразу вибір предмету
        keyboard = [
            [InlineKeyboardButton("💻 Комп’ютери",
callback_data='computers')],
            [InlineKeyboardButton("🌐 Мережі", callback_data='networks')],
            [InlineKeyboardButton("🇬🇧 Англійська", callback_data='english')]
        ]
        await update.message.reply_text("📚 Оберіть предмет:",
reply_markup=InlineKeyboardMarkup(keyboard))
        return PASSWORD
    else:
        await update.message.reply_text("👋 Вітаю! Введіть своє ПІБ:")
        return REGISTER
```

Команда /help призначена для надання користувачеві стислої методичної інформації щодо функціональних можливостей програмного агента. У відповідь на активацію даного елемента система генерує структурований перелік доступних інструкцій: ініціація діагностичної процедури, аналіз результатів, деавтентифікація та інші. Таким чином, цей інструкційний елемент виконує роль інформаційно-довідкового інтерфейсу та навігаційного засобу в функціональному просторі системи.

Лістинг 3.2 — Фрагмент коду метода /help

```
# /help
async def help_command(update: Update, context:
ContextTypes.DEFAULT_TYPE):
    await update.message.reply_text(
        "🛠️ Доступні команди:\n"
        "/start – Почати тестування або вибрати предмет\n"
        "/results – Переглянути останні 10 результатів\n"
        "/logout – Вийти з облікового запису\n"
        "/cancel – Скасувати тестування\n"
        "/help – Показати довідку"
    )
```

Команда /logout імплементує механізм деавтентифікації користувача в системі. Після активації даного елемента програмний агент здійснює очищення локальних сесійних даних (context.user_data.clear()), що забезпечує ініціацію процедури повторної реєстрації при наступному зверненні. Такий архітектурний підхід дозволяє здійснювати зміну активного користувача без необхідності перезапуску програмного агента чи видалення історії комунікацій.

Лістинг 3.3 — Фрагмент коду метода /logout

```
# /logout
async def logout(update: Update, context: ContextTypes.DEFAULT_TYPE):
    context.user_data.clear()
    await update.message.reply_text("🔒 Ви вийшли з профілю. Щоб почати спочатку, введіть /start.")
```

За допомогою команди /results суб'єкт навчання або уповноважена особа має можливість переглянути актуальні результати діагностичних процедур. Система здійснює збереження десяти найбільш актуальних записів у динамічному масиві, які містять персональні дані, ідентифікатор академічної групи, назву навчальної дисципліни та підсумкову кількісну оцінку. Даний функціональний підхід забезпечує оперативний доступ до результатів успішності суб'єктів навчання без необхідності інтеграції з системами управління базами даних.

Лістинг 3.4 — Фрагмент коду метода /results

```
# /results
async def results_command(update: Update, context:
ContextTypes.DEFAULT_TYPE):
    if not results:
        await update.message.reply_text("ℹ️ Немає результатів.")
    else:
        await update.message.reply_text("📊 Останні результати:\n\n" +
"\n".join(results))
```

Усі інструкційні елементи інтегруються до централізованого обробника ApplicationBuilder, який здійснює реєстрацію відповідних CommandHandler для кожного елемента. Завдяки цьому архітектурному рішенню забезпечується модульність та екстенсивність системи з перспективою подальшого функціонального розширення.

Таким чином, імплементація базових інструкційних елементів забезпечує інтуїтивно зрозумілу взаємодію з користувачем, підтримує структуровану логіку діагностичних процедур та створює середовище для збереження необхідної інформації про функціональний стан програмного агента.

3.4 Реалізація логіки тестування та перевірки відповідей

Функціональне ядро розробленого Telegram-бота для оцінювання знань становить комплексний механізм управління процесом тестування. Даний механізм забезпечує послідовну взаємодію з користувачем через етапи представлення питань, фіксацію та обробку наданих відповідей, обчислення набраних балів та генерацію підсумкового звіту після завершення всіх завдань.

При проектуванні системи було прийнято рішення використовувати структуру user_data, що надається інфраструктурою бібліотеки python-telegram-bot, для збереження інформації про стан сеансу кожного окремого користувача. Така архітектура дозволяє підтримувати незалежні сесії тестування для декількох користувачів одночасно.

Процедура тестування активується після успішного проходження користувачем етапів реєстрації та авторизації через введення валідного паролю до відповідного навчального предмета. Система динамічно завантажує тестові завдання з відповідного розділу структурованого сховища даних у форматі JSON (tests.json).

Лістинг 3.5 — Приклад запитання з файлу tests.json

```
"english": [
   {
    "question": "Choose the correct form: He ___ to school every day.",
    "options": ["go", "goes", "going", "gone"],
    "correct": "goes"
   }
```

Структурна організація тестових матеріалів дотримується уніфікованої схеми, що містить:

— формулювання запитання

— перелік можливих варіантів відповіді

— індикатор коректної відповіді

Ініціалізація тестового процесу супроводжується формуванням набору необхідних параметрів у структурі даних користувача.

Лістинг 3.6 — Стрічки коду, які відповідають за запис даних з тесту

```
context.user_data["questions"] = tests[subject]
   context.user_data["current"] = 0
   context.user_data["score"] = 0
```

Для забезпечення зручної взаємодії з користувачем під час відповіді на тестові запитання використовується технологія InlineKeyboardMarkup.
Вона створює інтерактивний інтерфейс з кнопками-варіантами відповідей, що суттєво підвищує ергономіку процесу тестування .

Система верифікації відповідей функціонує за алгоритмом порівняння обраного користувачем варіанту з еталонною відповіддю. Якщо у випадку співпадіння відбувається інкрементація лічильника правильних відповідей та надсилається повідомлення про успішне виконання завдання.

Лістинг 3.7 — Фрагмент, який відповідає за завершення тесту

```
   if answer == correct:
      context.user_data["score"] += 1
```

```
        reply = "✅ Правильно!"
    else:
        reply = f"❌ Неправильно. Правильна відповідь: {correct}"
```

По завершенні обробки поточного запитання система визначає наступний крок процедури: перехід до наступного тестового завдання або, у випадку вичерпання переліку питань, формування підсумкового результату з детальною статистикою проходження тесту.

Лістинг 3.8 — Фрагмент, який відповідає за завершення тесту

```
    if index >= len(questions):
        score = context.user_data["score"]
        name = context.user_data["name"]
        group = context.user_data["group"]
        subject = context.user_data["subject"]
        result = f"{name} ({group}) – {subject}: {score}/{len(questions)}"
        results.append(result)
        if len(results) > 10:
            results.pop(0)
        await update_or_query.  message.reply_text(f"✅ Тест
завершено!\nРезультат: {score} із {len(questions)}")
        return ConversationHandler.END
```

Особливої уваги заслуговує система тимчасового зберігання результатів проходження тестів у спеціалізованому масиві results. Кожен елемент цього масиву містить структуровану інформацію у текстовому форматі, наприклад: "Глиненко І.І. (1СП-21) – networks: 4/5". Ця інформація є доступною для перегляду через спеціальну команду /results та в перспективі може бути трансформована для зберігання у базі даних або форматі JSON.

Таким чином, імплементований алгоритм тестування формує комплексну систему для ефективної оцінки рівня знань студентів через інтуїтивно зрозумілий та функціонально повний інтерфейс взаємодії.

4 ТЕСТУВАННЯ І ВЕРИФІКАЦІЯ РОБОТИ БОТА

4.1 Засоби тестування Telegram-ботів та обробки JSON

У процесі створення Telegram-бота для оцінювання академічних компетенцій студентів критичною складовою розробки виступила комплексна верифікація його функціональності, стабільності роботи та адекватності реагування на користувацькі дії. Впровадження систематичного тестування уможливило ідентифікацію та усунення серії технічних недоліків, оптимізацію алгоритмічної складової програмного продукту та гарантування якісної інтеракції користувачів із ботом. У даному розділі проаналізовано методологічний апарат та технічний інструментарій, задіяний під час верифікаційного етапу.

Першорядним підходом до перевірки функціонального потенціалу розробленого рішення виступило безпосереднє тестування через інтерфейс Telegram із симуляцією реальних користувацьких сценаріїв. На даному етапі проводилася верифікація наступних функціональних компонентів:

— процедура реєстрації нових користувачів (включаючи введення персональних ідентифікаційних даних та академічної групи);

— механізм селекції навчальної дисципліни з подальшою автентифікацією через парольний захист;

— коректність візуалізації тестових запитань та альтернативних варіантів відповідей;

— алгоритмічна точність обчислення результативності та формування звітності;

— функціональність адміністративного інтерфейсу через команду /results.

Кожен із зазначених сценаріїв підлягав багаторазовій верифікації з варіюванням вхідних параметрів. Під час тестування оцінювалася як відповідність стандартним очікуванням функціонування системи, так і

коректність обробки некоректних, неповних або аномальних вхідних даних. На рисунку 4.1 практично показуються використання цих даних.

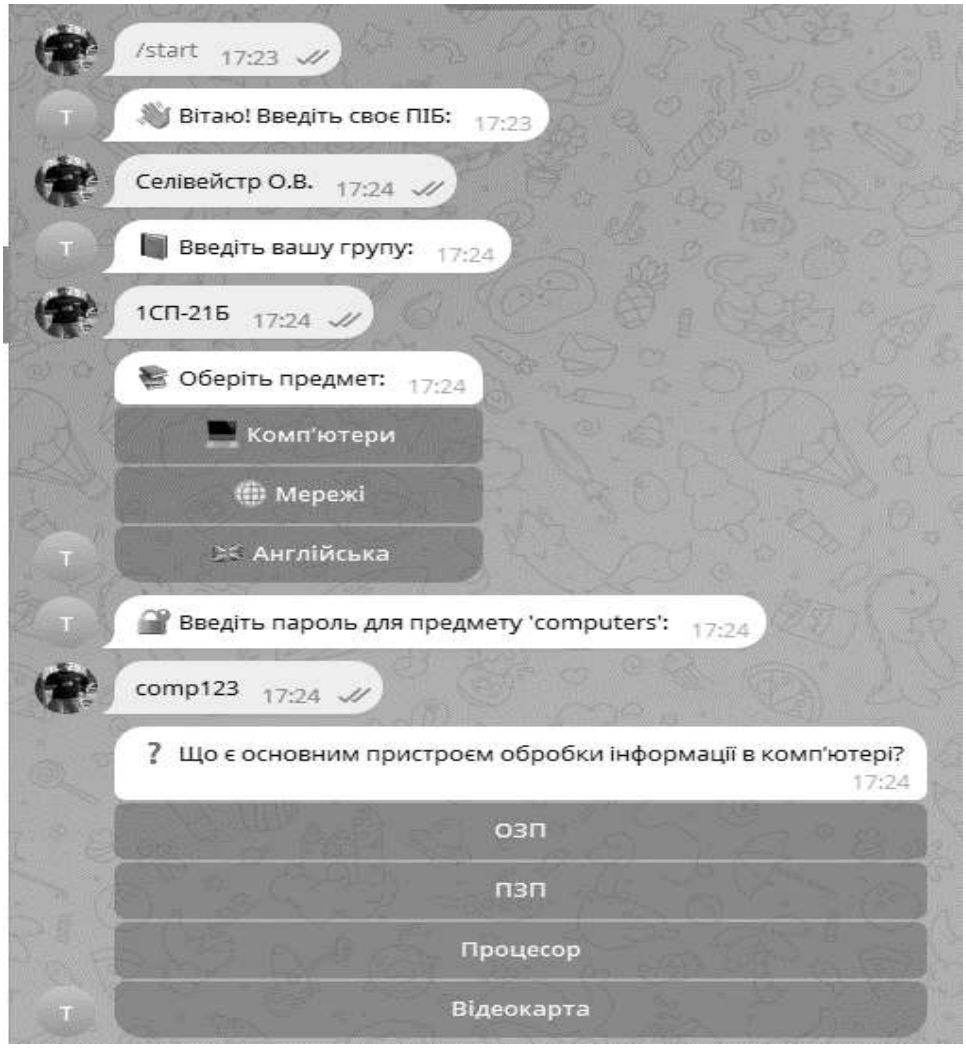

Рисунок 4.1 — Тестування в телеграм-боті

Усі етапи програмної реалізації та верифікації бота здійснювалися в середовищі інтегрованої розробки PyCharm 2025, що забезпечує розширений інструментарій для діагностики програмного коду. У процесі тестування активно застосовувалися:

— діагностичне виведення інформації до консолі через функцію print() — для моніторингу змінних стану та проміжних значень, яке буде зображене на рисунку 4.2;

— імплементація механізмів обробки винятків (try-except) — для виявлення критичних аномалій виконання;

— аналіз логічних послідовностей обробки команд та транзицій між станами в рамках архітектури ConversationHandler.

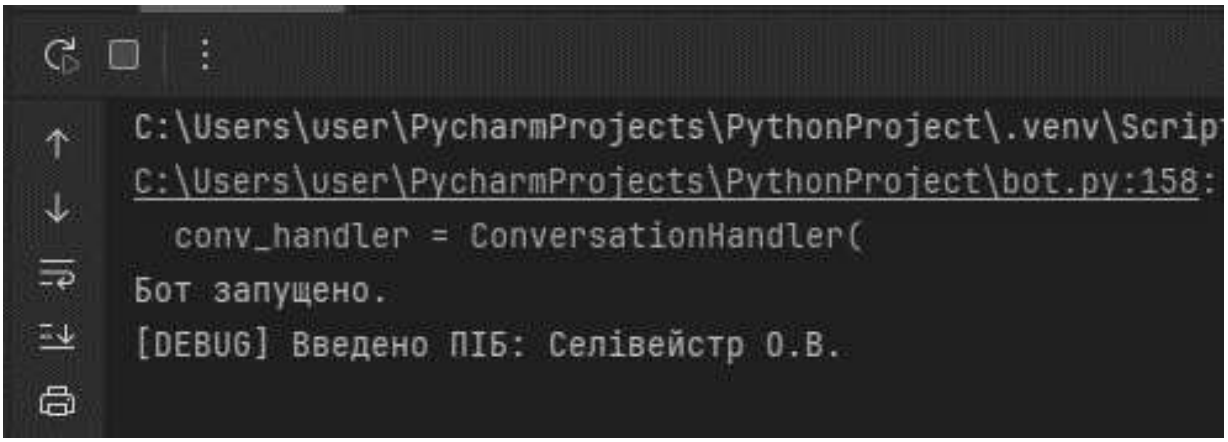

Рисунок 4.2 — Використання debugger з командою print()

Впровадження комплексного журналювання на рисунку 4.3 дозволило ефективно ідентифікувати проблемні аспекти, включаючи неправильну інтерпретацію даних, помилкову обробку командних інструкцій або відсутність очікуваної реакції бота.

Розроблений Telegram-бот використовує зовнішні файли формату JSON для збереження тестових матеріалів, користувацьких профілів, результатів оцінювання та автентифікаційних даних. Тестування охоплювало також перевірку коректності операцій зчитування, обробки та запису інформації до зазначених файлів.

Особлива увага приділялася наступним аспектам:

— відповідність структурної організації JSON-файлів проектним специфікаціям;

— стійкість до аномальних ситуацій, включаючи відсутність або пошкодження файлів;

— забезпечення цілісності даних при проведенні операцій оновлення (запобігання втраті інформації).

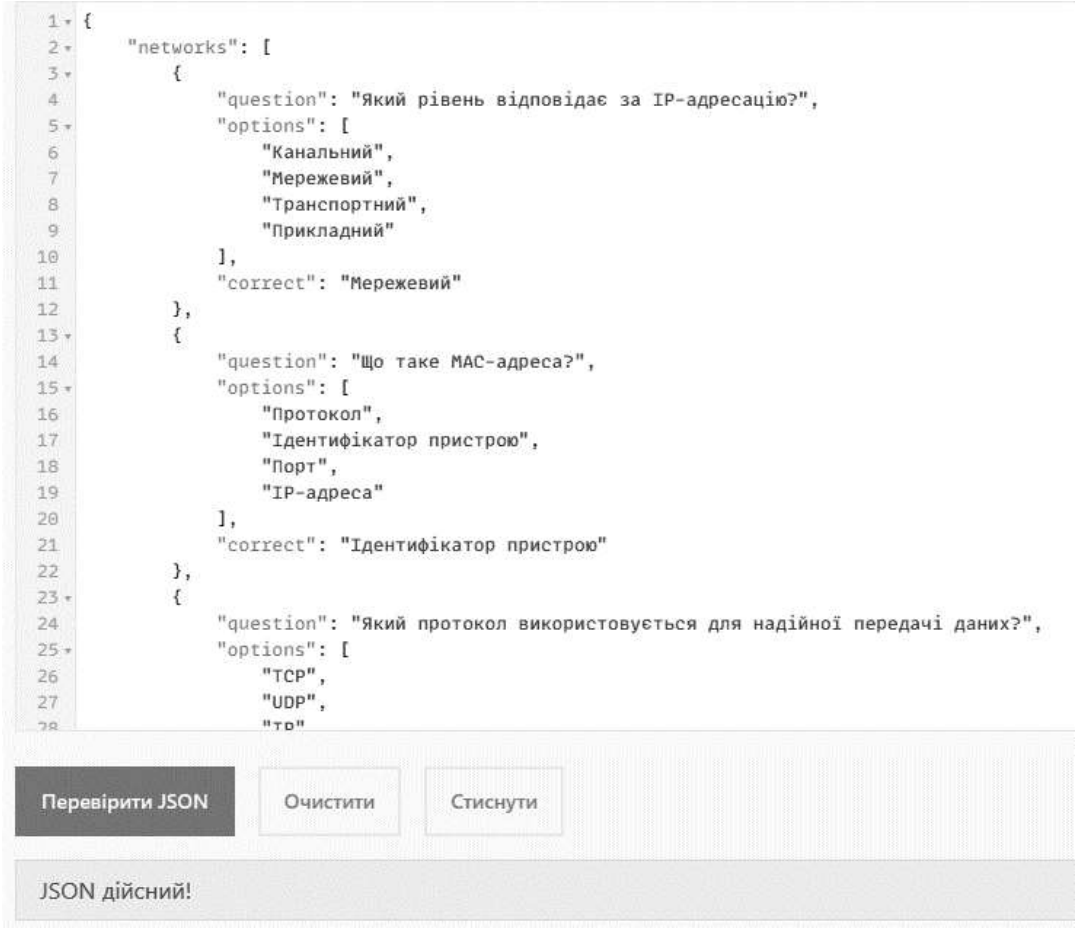

Рисунок 4.3 — Перевірка цілістності та роботи JSON файла на сайті JSONLint

Окрім основного функціонального тестування, для підвищення якості верифікації використовувалися допоміжні інструментальні засоби:

— TestStudent21b_bot — для автоматизованої перевірки реакції бота на різноманітні командні інструкції;

— JSONLint — для валідації синтаксичної коректності структур JSON-файлів;

— PyCharm Debugger — для покрокового контрольованого виконання коду з відстеженням динаміки змінних у режимі реального часу.

Комплексне застосування зазначеного інструментарію забезпечило всебічну оцінку стабільності функціонування програмного рішення та своєчасну елімінацію ідентифікованих проблемних аспектів.

4.2 Тестування сценаріїв користувача: реєстрація, проходження тестів

В процесі впровадження розробленого програмного рішення було здійснено комплексну перевірку ключових алгоритмів взаємодії з користувачем у середовищі месенджера Telegram. Головним завданням даного етапу виступала верифікація послідовного виконання всіх запрограмованих операцій від моменту ініціалізації діалогу до фіксації підсумкового результату оцінювання.

При активації бота через команду /start система запитує персональні дані респондента, зокрема, його прізвище, ім'я та по батькові, що зображено практично на рисунку 4.4, а в подальшому діалозі — академічну приналежність до навчальної групи. Експериментальна перевірка продемонструвала, що зазначена інформація належним чином зберігається у тимчасовому об'єкті user_data та використовується для індивідуалізації наступних комунікаційних повідомлень.

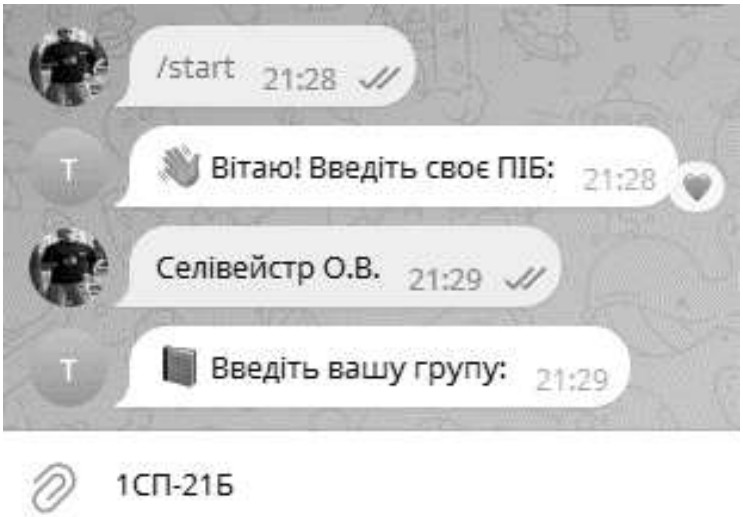

Рисунок 4.4 — Процес введення особистих даних та ідентифікації групи

Після успішної ідентифікації система пропонує користувачеві здійснити вибір навчальної дисципліни для проведення контрольного тестування. Наступним кроком виступає необхідність введення авторизаційного ключа для отримання доступу до відповідного набору тестових завдань. У процесі розробки для кожної навчальної дисципліни було сформовано індивідуальний пароль, інтегрований до словникової структури даних subject_passwords. Верифікація даного етапу, яка зображена на рисунку 4.5, засвідчила коректність обробки як позитивних, так і негативних сценаріїв авторизації.

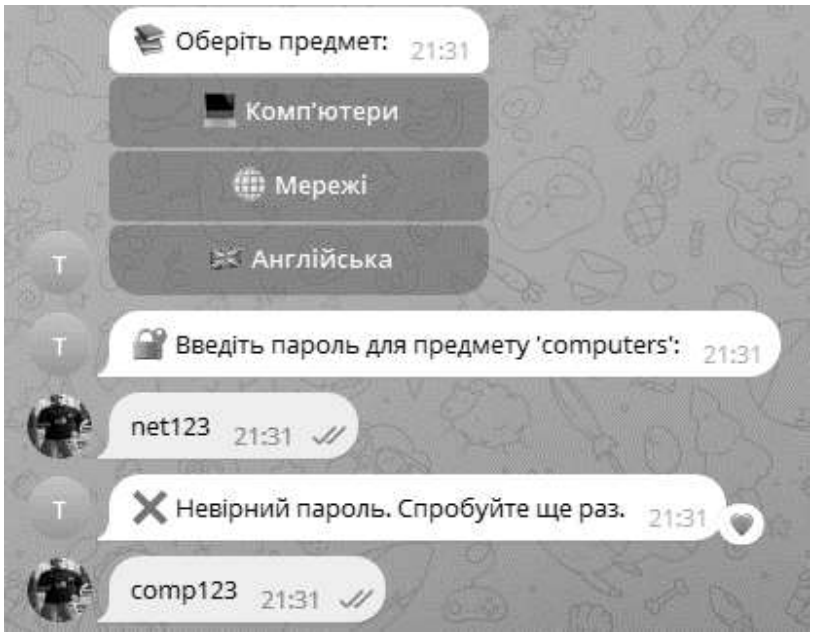

Рисунок 4.5 — Процес вибору навчальної дисципліни та системного сповіщення про невдалу спробу авторизації

Після валідації авторизаційних даних відбувається ініціалізація контрольного тестування. Система послідовно надсилає тестові питання з варіантами відповідей, представленими у вигляді інтерактивних елементів управління. Кожна відповідь респондента фіксується та підлягає миттєвій оцінці. У випадку коректної відповіді система надає позитивне підтвердження, при

помилковому виборі — відображає правильний варіант відповіді. Показано на рисунку 4.6.

Рисунок 4.6 — Варіанти відповідей та системні повідомлення щодо результату

Після завершення повного циклу тестових завдань система здійснює обчислення кількісних показників та формує підсумковий результат за формалізованим шаблоном: Прізвище Ім'я (Група) – Назва дисципліни: X/Y, де параметр X відображає кількість правильних відповідей, а Y — загальну кількість запропонованих завдань. Сформований результат автоматично інтегрується до загального реєстру, доступ до якого реалізовано на рисунку 4.7 через системну команду /results.

Архітектура програмного рішення передбачає можливість багаторазового проходження контрольних заходів після виходу з системи через команду /logout або повторної ініціалізації через команду /start, що забезпечує гнучкість у виборі різних навчальних дисциплін або модифікації персональних даних користувача. Дану команду зображено на рисунку 4.8.

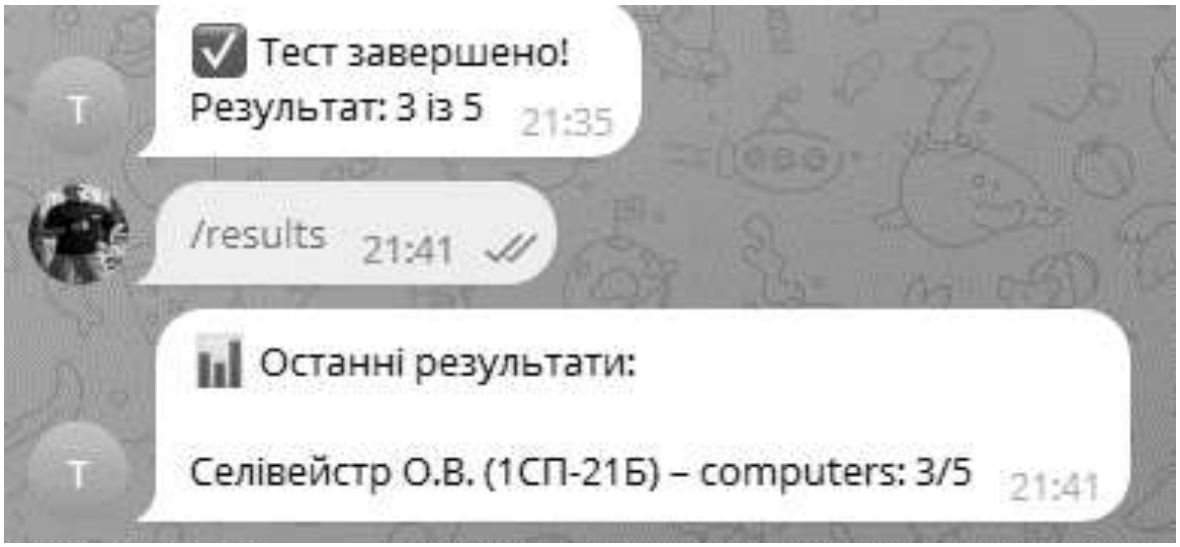

Рисунок 4.7 – Результат завершення тесту та використання /results

Всі компоненти повторного циклу тестування також пройшли успішну верифікацію функціональності.

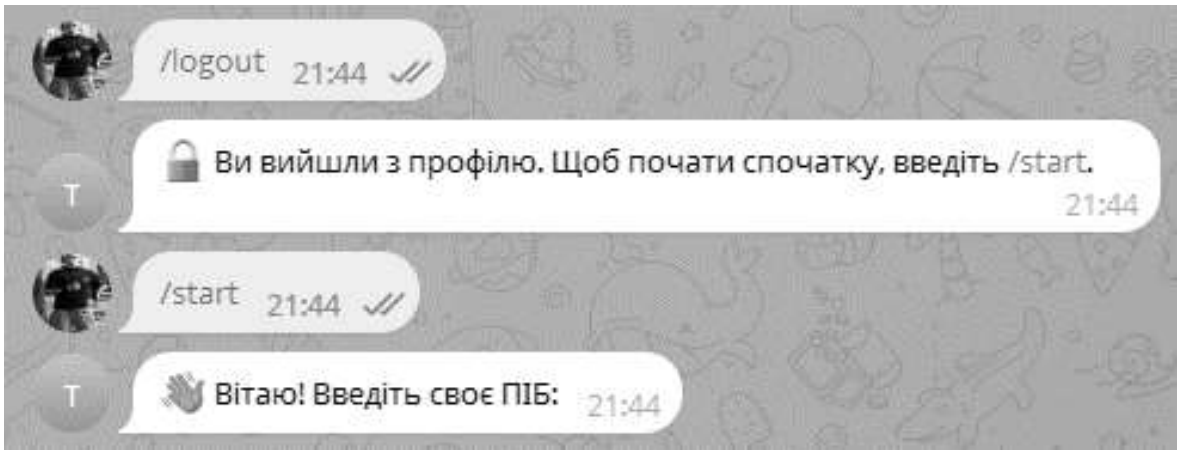

Рисунок 4.8 – Використання команди /logout

4.3 Аналіз результатів тестування та усунення помилок

В процесі діагностичного аналізу розробленого програмного рішення для месенджера Telegram було ідентифіковано низку функціональних недосконалостей, які підлягали подальшому коригуванню на етапі оптимізації коду. Стратегічним завданням даної фази виступало забезпечення надійного та прогнозованого функціонування програмного комплексу в середовищі практичного застосування.

На початковій стадії експлуатації програмного продукту було встановлено, що при введенні службових команд /start або /help під час виконання активного тестового сеансу, система не здійснювала належного завершення поточного діалогового циклу. Дана особливість призводила до порушення логічної послідовності алгоритму взаємодії. В рамках оптимізації було впроваджено механізм контролю подібних ситуацій за допомогою використання елементів fallbacks у структурі об'єкта ConversationHandler, що забезпечило коректне розпізнавання керуючих команд та регламентоване припинення активних сеансів.

Аналітичні показники проходження контрольних тестувань фіксувалися виключно у динамічній структурі даних results, що не гарантувало їх збереження після реініціалізації програмного комплексу. В контексті вирішення даної проблематики заплановано реалізацію механізму збереження даних у форматі results.json, проте в межах поточної імплементації було застосовано обмежений підхід із забезпеченням відображення десяти останніх результативних записів у оперативній пам'яті під час функціонування активної сесії.

При процедурі введення персональних ідентифікаційних даних користувача (ПІБ) або академічної групи існувала потенційна можливість внесення некоректної інформації (порожні значення, нерелевантні символи, цифрові коди тощо). З метою елімінації подібних сценаріїв було інтегровано

базові алгоритми перевірки форматної відповідності вхідних даних, а також імплементовано механізми обробки виняткових ситуацій (try-except) для унеможливлення аварійного припинення роботи програми [18].

В ході експериментального застосування програмного продукту було виявлено архітектурну недосконаліть, пов'язану з обробкою випадків введення некоректних паролів, коли система не здійснювала належну транзицію до відповідного стану. Дана проблематика була пов'язана з неоптимальною структуризацією логіки переходів у контексті ConversationHandler [19]. Вирішення даного питання було реалізовано шляхом деталізації алгоритмів повернення станів після процедури верифікації автентифікаційних даних.

Було діагностовано ситуації некоректного відображення системних повідомлень, коли програмний комплекс намагався застосувати метод message.reply_text у випадках, що передбачали обробку callback-запитів. У рамках оптимізації було внесено архітектурні модифікації у функціональні блоки, де замість конструкції update.message.reply_text() було застосовано альтернативний підхід update.callback_query.message.reply_text(), або імплементовано процедуру ідентифікації джерела запиту.

На рисунку 4.9, при ініціалізації програмного скрипта фіксувалось технічне застереження класу PTBUserWarning, асоційоване з використанням компонента CallbackQueryHandler без експліцитного параметра per_message=True. Проведено аналіз технічної документації бібліотеки та прийнято обґрунтоване рішення щодо збереження параметрів за замовчуванням, оскільки індиковане застереження не мало негативного впливу на функціональні характеристики, однак було відповідним чином документовано в системному журналі.

Комплекс ідентифікованих дефектів підлягав ретельному аналізу, локалізації та належному коригуванню [20]. Після завершення циклу повторного

тестування програмний комплекс демонстрував успішне виконання всіх сценаріїв взаємодії, передбачених функціональними вимогами проєкту.

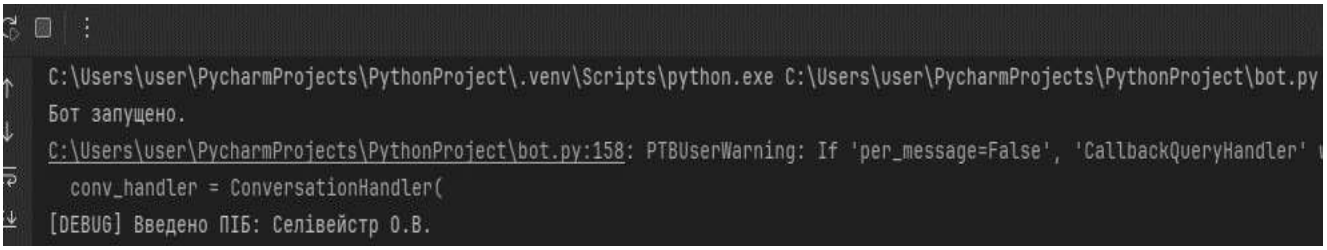

Рисунок 4.9 – Помилка PTBUserWarning

Дані показники дозволяють констатувати належний рівень програмної стабільності та експлуатаційної готовності до імплементації в середовищі кінцевих користувачів.

ВИСНОВКИ

Реалізовано програмне рішення на базі месенджера Telegram для автоматизації оцінювання академічних компетенцій здобувачів вищої освіти, що становить внесок у трансформацію освітніх практик.

У процесі дослідження проведено аналіз інструментарію електронного оцінювання та обрано технологічний стек Python з бібліотекою python-telegram-bot. Спроектовано систему діалогової взаємодії з користувачем, що забезпечує реєстрацію, вибір освітнього компонента, захист тестових матеріалів та надання результатів оцінювання. Тестові матеріали реалізовано у форматі JSON для масштабованості.

Програмна імплементація включає багаторівневу архітектуру з ConversationHandler, обробку системних команд та механізми зберігання користувацьких даних. Функціональне тестування підтвердило відповідність системи вимогам.

Система характеризується гнучкою архітектурою та інтуїтивним інтерфейсом. JSON-файли дозволяють педагогам оновлювати тести без зміни коду. Telegram забезпечує доступ з різних пристроїв без додаткових інсталяцій. Автономне функціонування спрощує розгортання, а механізми авторизації забезпечують безпеку.

Перспективи розвитку включають генерацію статистичних звітів, інтеграцію з базами даних, інтероперабельність з навчальними платформами та адаптивне тестування.

Програмний комплекс є практично орієнтованим рішенням для освітнього середовища та може застосовуватися в закладах вищої освіти для підвищення якості освітніх послуг.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Chatbots as Digital Language Tutors: Revolutionizing Education Through AI [Електронний ресурс]. — Режим доступу: https://ejournal.upi.edu/index.php/ijost/article/view/79514

2. Considerations and strategies for effective online assessment with a focus on the biomedical sciences [Електронний ресурс]. — Режим доступу: https://pmc.ncbi.nlm.nih.gov/articles/PMC8728109/

3. Do AI chatbots improve students learning outcomes? Evidence from a meta-analysis [Електронний ресурс]. — Режим доступу: https://bera-journals.onlinelibrary.wiley.com/doi/abs/10.1111/bjet.13334

4. Top 4 Best Chatbots for Higher Education 2025 [Електронний ресурс]. — Режим доступу: https://www.o8.agency/blog/top-4-best-chatbots-higher-education

5. A Comparative Study of WhatsApp and Telegram in Enhancing English Language Learning: A Case Study at the English Department, University of Benghazi, Suluq Campus [Електронний ресурс]. — Режим доступу: https://www.semanticscholar.org/paper/A-Comparative-Study-of-WhatsApp-and-Telegram-in-A-Aljarih/b63dc077f2d86135c4d00c29899f8a474f6f3eef

6. TeleScope: A Longitudinal Dataset for Investigating Online Discourse and Information Interaction on Telegram [Електронний ресурс]. — Режим доступу: https://arxiv.org/abs/2504.19536v1

7. Психологічні аспекти та основні мотиви використання соціальних мереж. ResearchGate, 2019. [Електронний ресурс]. — Режим доступу: https://www.researchgate.net/publication/339467568_Psihologicni_aspekti_ta_osnovni_motivi_vikoristanna_socialnih_merez

8. How to Create a Telegram Bot using Python [Електронний ресурс]. — Режим доступу: https://www.freecodecamp.org/news/how-to-create-a-telegram-bot-using-python/

9. Implementasi Telegram Bot untuk Proses Automatisasi Rekapan Data Menggunakan Metode Webhook [Електронний ресурс]. — Режим доступу: https://ojs.stmikdharmapalariau.ac.id/index.php/jikb/article/view/532

10. Telegram Bot API Documentation [Електронний ресурс]. — Режим доступу:https://core.telegram.org/bots/api

11. Validation of Modern JSON Schema: Formalization and Complexity [Електронний ресурс]. — Режим доступу:https://arxiv.org/abs/2307.10034

12. CommandHandler - python-telegram-bot v21.5 [Електронний ресурс]. — Режим доступу: https://docs.python-telegram-bot.org/en/v21.5/telegram.ext.commandhandler.html

13. Real Python - Build a Quiz Application With Python [Електронний ресурс]. — Режим доступу: https://realpython.com/python-quiz-application/

14. N. Modrzyk. Building Telegram Bots: Develop Bots in 12 Programming Languages using the Telegram Bot API. Apress, 2018. 256 p.

15. E. Gamma, R. Helm, R. Johnson, J. Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional, 1994. 395 p.

18. Difference between Testing and Debugging [Електронний ресурс]. — Режим доступу: https://www.browserstack.com/guide/difference-between-testing-and-debugging

19. Telegram bot - How to handle conversations? [Електронний ресурс]. — Режим доступу: https://stackoverflow.com/questions/74782052/telegram-bot-how-to-handle-conversations

20. Software Testing as a Debugging Tool [Електронний ресурс]. — Режим доступу: https://debugagent.com/software-testing-as-a-debugging-tool

ДОДАТОК А

Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
проф., д.т.н. Азаров О.Д.
«21» березня 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ
на виконання бакалаврської кваліфікаційної роботи
«Чат-бот для тестування знань студента»
08-54.БКР.013.00.000 ТЗ

Науковий керівник: к.т.н., доц. каф. ОТ
Кожем'яко А. В.
Студент групи 1СП-21б
Селівейстр О.В.

ВНТУ 2025

1 Підставою для виконання бакалаврської кваліфікаційної роботи (БКР)

1.1 Необхідність автоматизації процесів перевірки знань студентів, що дозволяє підвищити ефективність навчального процесу. Telegram-боти як інструмент мають високу популярність серед користувачів завдяки доступності, швидкодії та зручному інтерфейсу. Розробка таких систем дозволяє набути актуальних навичок у галузі створення інтерактивних застосунків із використанням API, асинхронного програмування та роботи з JSON-структурами.

1.2 Наказ про затвердження теми БКР.

2 Мета БКР і призначення розробки

2.1 Мета роботи – створення Telegram-бота, який дозволяє проводити онлайн-тестування студентів, забезпечуючи реєстрацію користувачів, вибір предметів, валідацію відповідей, підрахунок результатів і збереження статистики.

2.2 Призначення розробки – розроблений бот призначений для використання в навчальних закладах як інструмент дистанційного оцінювання знань студентів, доступний через платформу Telegram, з можливістю масштабування на різні навчальні дисципліни.

3 Вихідні дані для виконання БКР

3.1 Аналіз існуючих рішень у сфері тестування знань за допомогою ботів, зокрема Telegram-ботів, і вивчення бібліотеки python-telegram-bot.

3.2 Проектування логіки роботи бота, включаючи етапи реєстрації, вибору предмета, проходження тесту та обробки результатів.

3.3 Реалізація Telegram-бота мовою Python з використанням файлів JSON для зберігання тестів, результатів і даних користувачів, а також забезпечення зручного інтерфейсу взаємодії через кнопки.

4 Вимоги до виконання БКР

Робота повинна бути виконана відповідно до затвердженої теми та наказу. Telegram-бот має включати функції реєстрації, перевірки знань з різних предметів, збереження результатів та відображення статистики. Програмну частину необхідно реалізувати на мові Python з використанням бібліотеки python-telegram-bot. Код має бути структурованим, а інтерфейс – зручним для користувача.

5 Етапи БКР та очікувані результати

Етапи БКР та очікувані результати приведено в таблиці А.1

Таблиця А.1 — Етапи БКР

№	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Вибір, узгодження та затвердження теми БКР	21.03.2025	22.03.2025	
2	Аналіз літературних джерел та аналогів у сфері онлайн-тестування	23.03.2025	27.03.2025	
3	Постановка задачі, визначення вимог до функціоналу Telegram-бота	28.03.2025	30.03.2025	
4	Проєктування архітектури програмного забезпечення	31.03.2025	04.04.2025	
5	Розробка основного функціоналу бота (реєстрація, вибір предмету)	05.04.2025	09.04.2025	
6	Реалізація системи тестування та обробки відповідей	10.04.2025	17.04.2025	
7	Додавання збереження результатів і команди для перегляду статистики	18.04.2025	20.04.2025	
8	Проведення тестування Telegram-бота, усунення помилок	21.04.2025	27.04.2025	

9	Оформлення пояснювальної записки та графічних матеріалів	28.04.2025	06.05.2025	
10	Нормоконтроль та рецензування БКР	07.05.2025	12.05.2025	
11	Захист БКР	13.05.2025	13.05.2025	

6 Матеріали, що подаються до захисту БКР

До захисту подаються: пояснювальна записка БКР, графічні і ілюстративні матеріали, протокол попереднього захисту БКР на кафедрі, відгук наукового керівника, рецензія, анотації БКР українською та іноземною мовами, довідка про відповідність оформлення БКР діючим вимогам.

7 Порядок контролю виконання та захисту БКР

Виконання етапів графічної та розрахункової документації БКР контролюється керівником згідно зі встановленими термінами. Захист БКР відбувається на засіданні Екзамеційної комісії, затвердженої наказом ректора.

8 Вимоги до оформлювання та порядок виконання БКР

8.1 При оформлювання БКР використовуються:

– ДСТУ 3008: 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

– ДСТУ 8302: 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

– методичні вказівки до виконання бакалаврських кваліфікаційних робіт зі спеціальності 123 – «Комп’ютерна інженерія» (освітня програма «Комп’ютерна інженерія»). Кафедра обчислювальної техніки ВНТУ 2022;

– документами на які посилаються у вище вказаних.

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Чат-бот для тестування знань студента

Тип роботи: бакалаврська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ кафедра обчислювальної техніки
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 8,5 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

- ✓ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

- ☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

- ☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

Азаров О.Д., зав. каф. ОТ _____ _____
(прізвище, ініціали, посада) (підпис)

Тарновський М.Г., доц. каф. _____ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Захарченко С.М.
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Кожем'яко А. В.
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Селівейстр О.В
(підпис) (прізвище, ініціали)

ДОДАТОК В

Ілюстративна частина

ЧАТ-БОТ ДЛЯ ТЕСТУВАННЯ ЗНАНЬ СТУДЕНТА

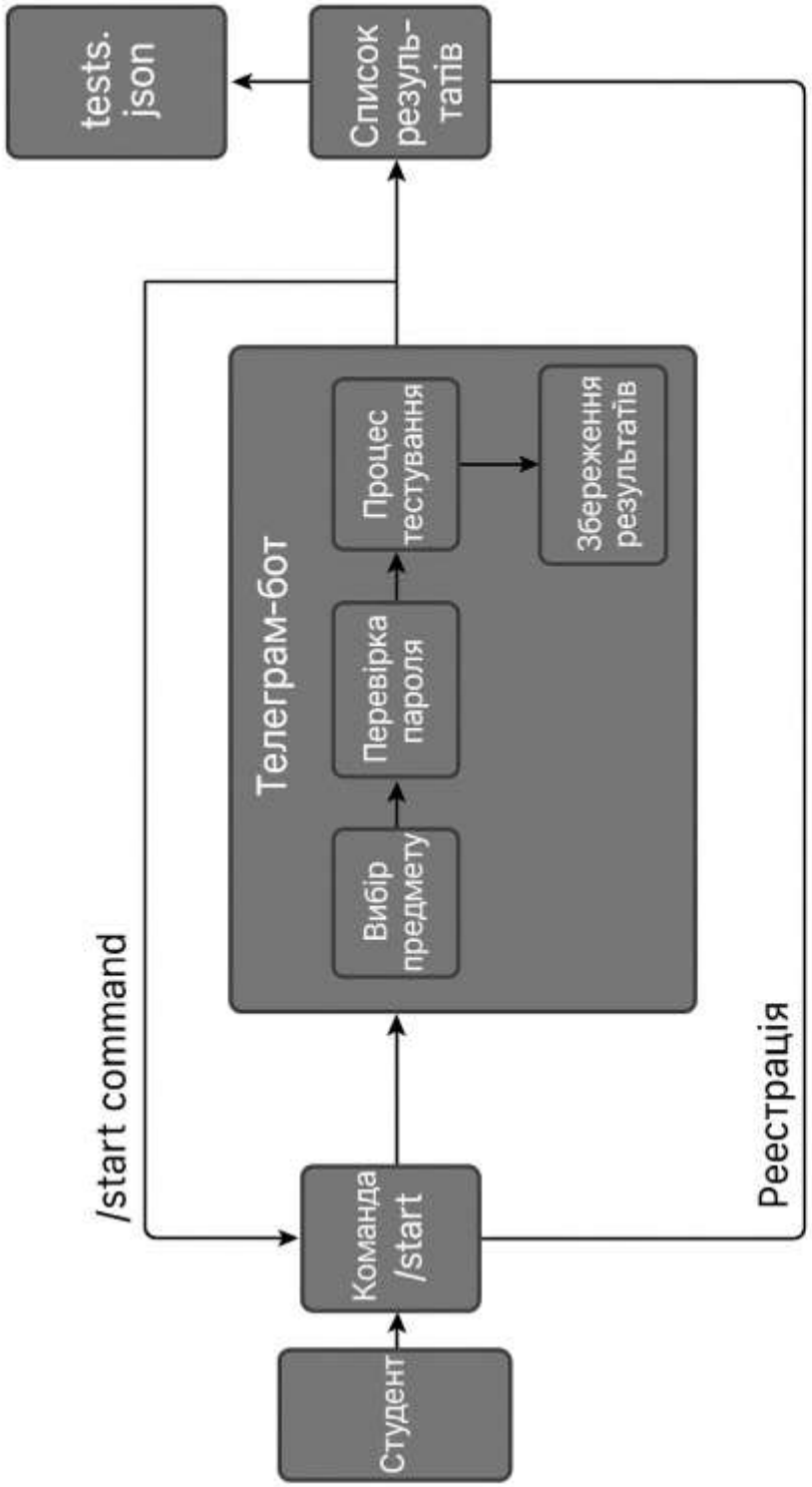

Рисунок В.1 — Схема архітектури Telegram-бота

Рисунок В.2 — Модель діалогу з користувачем

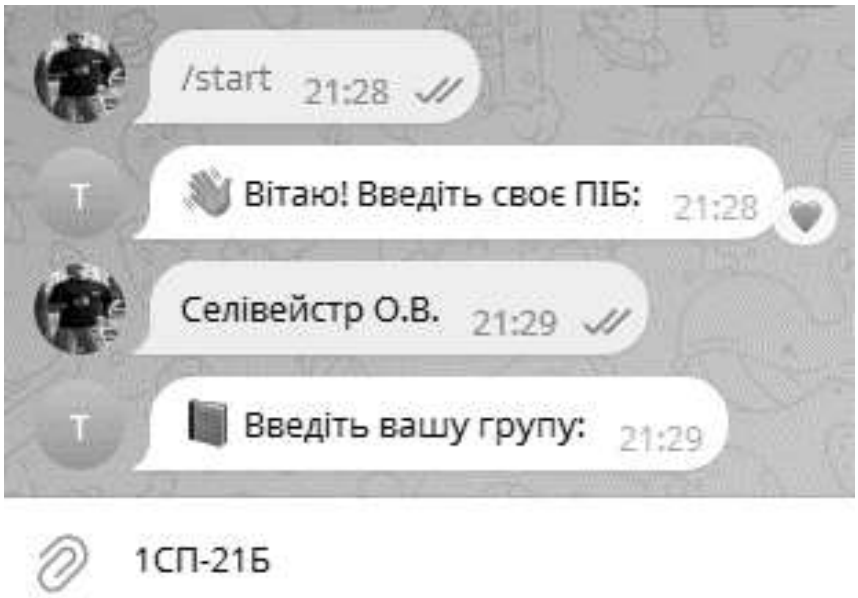

Рисунок В.3 — Введення особистих даних та ідентифікації груп

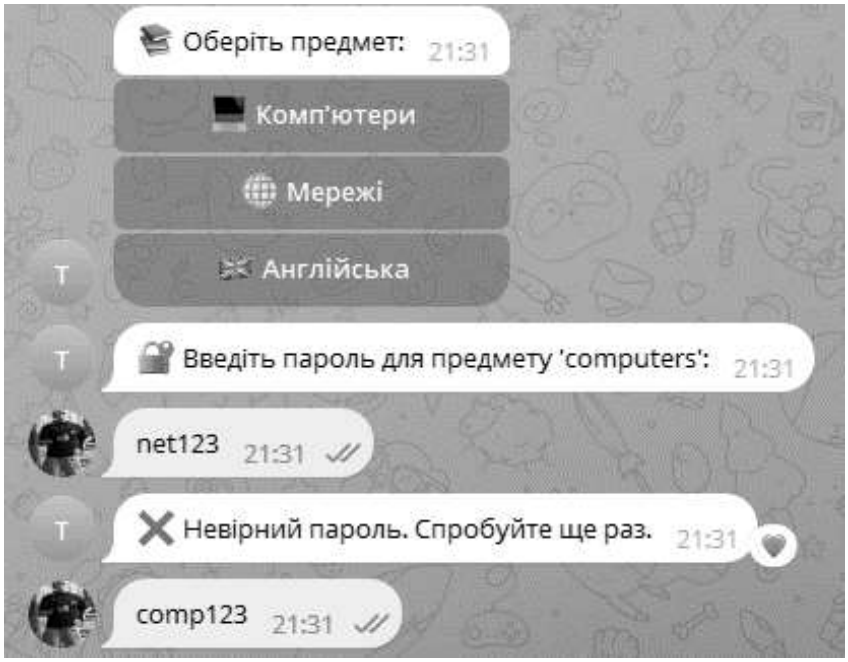

Рисунок В.3 — Процес вибору навчальної дисципліни та системного сповіщення про невдалу спробу авторизації

Рисунок В.5 — Варіанти відповідей та системні повідомлення щодо результату

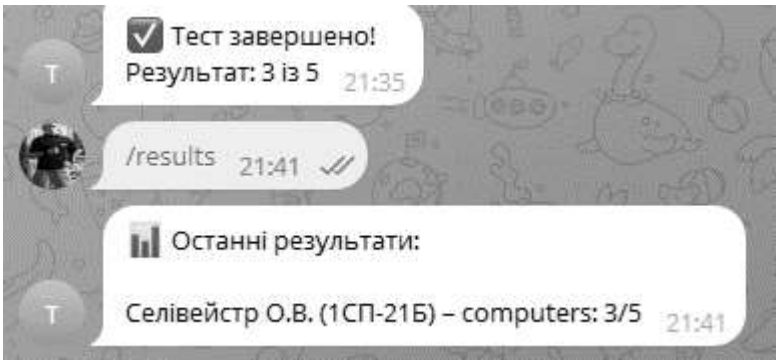

Рисунок В.6 — Результат завершення тесту та використання /results

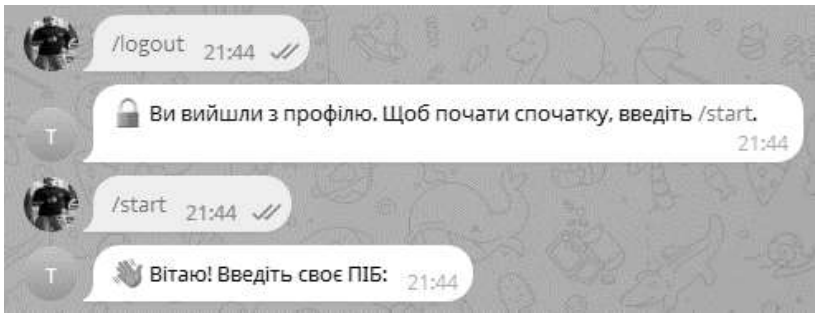

Рисунок В.7 — Використання команди /logout

ДОДАТОК Г

Лістинг програмного забезпечення

bot.py

```
import json
from telegram import Update, InlineKeyboardButton, InlineKeyboardMarkup
from telegram.ext import (
    ApplicationBuilder, CommandHandler, MessageHandler,
    filters, CallbackQueryHandler, ContextTypes, ConversationHandler
)

REGISTER, SUBJECT, PASSWORD, TEST = range(4)

with open("tests.json", "r", encoding="utf-8") as f:
    tests = json.load(f)

subject_passwords = {
    "networks": "net123",
    "computers": "comp123",
    "english": "eng123"
}

results = []

async def start(update: Update, context: ContextTypes.DEFAULT_TYPE):
    user_data = context.user_data
    if "name" in user_data and "group" in user_data:
        # Якщо користувач вже зареєстрований — одразу вибір предмету
        keyboard = [
            [InlineKeyboardButton("💻 Комп’ютери",
callback_data='computers')],
            [InlineKeyboardButton("🌐 Мережі", callback_data='networks')],
            [InlineKeyboardButton("🇬🇧 Англійська", callback_data='english')]
        ]
        await update.message.reply_text("📚 Оберіть предмет:",
reply_markup=InlineKeyboardMarkup(keyboard))
        return PASSWORD
    else:
        await update.message.reply_text("👋 Вітаю! Введіть своє ПІБ:")
        return REGISTER
```

```
async def register_name(update: Update, context:
ContextTypes.DEFAULT_TYPE):
    context.user_data["name"] = update.message.text
    print(f"[DEBUG] Введено ПІБ: {context.user_data['name']}")
    await update.message.reply_text("📝 Введіть вашу групу:")
    return SUBJECT

async def register_group(update: Update, context:
ContextTypes.DEFAULT_TYPE):
    context.user_data["group"] = update.message.text
    keyboard = [
        [InlineKeyboardButton("💻 Комп’ютери", callback_data='computers')],
        [InlineKeyboardButton("🌐 Мережі", callback_data='networks')],
        [InlineKeyboardButton("GB Англійська", callback_data='english')]
    ]
    await update.message.reply_text("📚 Оберіть предмет:",
reply_markup=InlineKeyboardMarkup(keyboard))
    return PASSWORD

async def subject_password(update: Update, context:
ContextTypes.DEFAULT_TYPE):
    query = update.callback_query
    await query.answer()
    subject = query.data
    context.user_data["subject"] = subject
    await query.message.reply_text(f"🔐 Введіть пароль для предмету
'{subject}':")
    return TEST

async def start_test(update: Update, context:
ContextTypes.DEFAULT_TYPE):
    subject = context.user_data["subject"]
    password = update.message.text
    if password != subject_passwords.get(subject):
        await update.message.reply_text("❌ Невірний пароль. Спробуйте ще
раз.")
        return TEST

    context.user_data["questions"] = tests[subject]
```

```
    context.user_data["current"] = 0
    context.user_data["score"] = 0

    return await send_question(update, context)

async def send_question(update_or_query, context:
ContextTypes.DEFAULT_TYPE):
    index = context.user_data["current"]
    questions = context.user_data["questions"]

    if index >= len(questions):
        score = context.user_data["score"]
        name = context.user_data["name"]
        group = context.user_data["group"]
        subject = context.user_data["subject"]

        result = f"{name} ({group}) – {subject}: {score}/{len(questions)}"
        results.append(result)
        if len(results) > 10:
            results.pop(0)

        await update_or_query.  message.reply_text(f"✅ Тест
завершено!\nРезультат: {score} із {len(questions)}")
        return ConversationHandler.END

    q = questions[index]
    keyboard = [[InlineKeyboardButton(opt, callback_data=opt)] for opt in
q["options"]]
    await update_or_query.message.reply_text(f"❓ {q['question']}",
reply_markup=InlineKeyboardMarkup(keyboard))
    return TEST

async def handle_answer(update: Update, context:
ContextTypes.DEFAULT_TYPE):
    query = update.callback_query
    await query.answer()
    answer = query.data

    index = context.user_data["current"]
    question = context.user_data["questions"][index]
    correct = question["correct"]
```

```
    if answer == correct:
        context.user_data["score"] += 1
        reply = "✅ Правильно!"
    else:
        reply = f"❌ Неправильно. Правильна відповідь: {correct}"

    await query.message.reply_text(reply)
    context.user_data["current"] += 1
    return await send_question(query, context)

async def results_command(update: Update, context:
ContextTypes.DEFAULT_TYPE):
    if not results:
        await update.message.reply_text("ℹ️ Немає результатів.")
    else:
        await update.message.reply_text("📊 Останні результати:\n\n" +
"\n".join(results))

async def logout(update: Update, context: ContextTypes.DEFAULT_TYPE):
    context.user_data.clear()
    await update.message.reply_text("🔒 Ви вийшли з профілю. Щоб почати
спочатку, введіть /start.")

async def help_command(update: Update, context:
ContextTypes.DEFAULT_TYPE):
    await update.message.reply_text(
        "🛠 Доступні команди:\n"
        "/start – Почати тестування або вибрати предмет\n"
        "/results – Переглянути останні 10 результатів\n"
        "/logout – Вийти з облікового запису\n"
        "/cancel – Скасувати тестування\n"
        "/help – Показати довідку"
    )

async def cancel(update: Update, context: ContextTypes.DEFAULT_TYPE):
    await update.message.reply_text("❌ Тест скасовано. Ви можете почати
заново командою /start.")
    return ConversationHandler.END
```

```
if __name__ == "__main__":
    app = ApplicationBuilder().token("8016952404:AAGMpgibbVnZsjuglNLo1Oe5569F3bRXcec").build()

    conv_handler = ConversationHandler(
        entry_points=[CommandHandler("start", start)],
        states={
            REGISTER: [MessageHandler(filters.TEXT & ~filters.COMMAND, register_name)],
            SUBJECT: [MessageHandler(filters.TEXT & ~filters.COMMAND, register_group)],
            PASSWORD: [CallbackQueryHandler(subject_password)],
            TEST: [
                MessageHandler(filters.TEXT & ~filters.COMMAND, start_test),
                CallbackQueryHandler(handle_answer)
            ]
        },
        fallbacks=[CommandHandler("cancel", cancel)]
    )

    app.add_handler(conv_handler)
    app.add_handler(CommandHandler("help", help_command))
    app.add_handler(CommandHandler("results", results_command))
    app.add_handler(CommandHandler("logout", logout))
    app.add_handler(CommandHandler("cancel", cancel))  # Щоб працював і поза тестом

    print("Бот запущено.")
    app.run_polling()
```

subject_password.json

```
{
    "networks": {
      "name": "Комп'ютерні мережі",
      "password": "net123"
    },
    "computers": {
      "name": "Організація та функціонування комп'ютерів",
      "password": "org456"
```

```
    },
    "english": {
      "name": "Англійська мова",
      "password": "eng789"
    }
}
```

tests.json

```
{
    "networks": [
      {
        "question": "Який рівень відповідає за IP-адресацію?",
        "options": ["Канальний", "Мережевий", "Транспортний",
"Прикладний"],
        "correct": "Мережевий"
      },
      {
        "question": "Що таке MAC-адреса?",
        "options": ["Протокол", "Ідентифікатор пристрою", "Порт", "IP-
адреса"],
        "correct": "Ідентифікатор пристрою"
      },
      {
        "question": "Який протокол використовується для надійної передачі
даних?",
        "options": ["TCP", "UDP", "IP", "HTTP"],
        "correct": "TCP"
      },
      {
        "question": "Який пристрій з’єднує різні мережі?",
        "options": ["Маршрутизатор", "Комутатор", "Міст", "Концентратор"],
        "correct": "Маршрутизатор"
      },
      {
        "question": "Що означає скорочення DNS?",
        "options": ["Data Network Service", "Domain Name System", "Digital
Number Service", "Direct Network System"],
        "correct": "Domain Name System"
      }
    ],
```

```
  "computers": [
    {
      "question": "Що є основним пристроєм обробки інформації в комп’ютері?",
      "options": ["ОЗП", "ПЗП", "Процесор", "Відеокарта"],
      "correct": "Процесор"
    },
    {
      "question": "Який компонент відповідає за збереження постійної інформації?",
      "options": ["ОЗП", "ПЗП", "Процесор", "Блок живлення"],
      "correct": "ПЗП"
    },
    {
      "question": "Що таке шина в архітектурі комп'ютера?",
      "options": ["Карта пам’яті", "Канал передачі даних", "Процесор", "Мережа"],
      "correct": "Канал передачі даних"
    },
    {
      "question": "Яка одиниця вимірювання тактової частоти процесора?",
      "options": ["Байт", "Герц", "Вольт", "Ом"],
      "correct": "Герц"
    },
    {
      "question": "Який тип пам’яті є енергозалежним?",
      "options": ["ROM", "Flash", "HDD", "RAM"],
      "correct": "RAM"
    }
  ],
  "english": [
    {
      "question": "Choose the correct form: He ___ to school every day.",
      "options": ["go", "goes", "going", "gone"],
      "correct": "goes"
    },
    {
      "question": "Choose the correct word: I ___ a book now.",
      "options": ["read", "reads", "am reading", "reading"],
      "correct": "am reading"
    },
```

```
            {
              "question": "Choose the correct tense: They ___ already eaten.",
              "options": ["have", "has", "had", "have had"],
              "correct": "have"
            },
            {
              "question": "Choose the correct form: She ___ TV when I called.",
              "options": ["watch", "was watching", "watched", "is watching"],
              "correct": "was watching"
            },
            {
              "question": "Choose the right word: There ___ a lot of people at the party.",
              "options": ["was", "were", "are", "is"],
              "correct": "were"
            }
          ]
}
```


users.json


```
{
          "757391386": {
            "username": "Селівейстр 21б",
            "subject": "computers",
            "current_question": 3,
            "score": 0
          }
}
```