МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма: «Розробка програмного забезпечення»

Група: 4РП-08

Дипломний проект

здобувача освіти денної форми навчання
РП.08.22.000.ДП

СТАЙКУЦИ
МАКСИМА СЕРГІЙОВИЧА

м. Одеса
2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: **121 «Інженерія програмного забезпечення»**

Освітньо-професійна програма: **«Розробка програмного забезпечення»**

Група: **4РП-08**

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проекту на тему:

Розробка телеграм боту для перевірки знань з англійської мови на базі ШІ

Проектний матеріал складається з пояснювальної записки на 84 сторінках та графічного (презентаційного) матеріалу на 14 аркушах (слайдах)

Дипломник _____ (Стайкуца М. С.)

Керівник _____ (Закроєв Ю.М.)

Консультанти:

з економічного розділу _____ (Канський М.Ю.)

з розділу охорони праці та техніки безпеки _____ (Чорновол Н.І.)

з нормоконтролю _____ (Петрашова В.І.)

старший консультант _____ (Кривченко Ю.В.)

До захисту допущений

Голова циклової комісії _____ (Кривченко Ю.В.)

Завідувач відділення _____ (Краснокутська К.Г.)

Захист «21» 06 2025 р. Протокол ЕК № 1

Оцінка ЕК 5/93

Секретар ЕК _____

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Відділення *комп'ютерних систем* Комісія *КТ та ПІ*

Спеціальність *121 «Інженерія програмного забезпечення»*

Освітньо-професійна програма *«Розробка програмного забезпечення»*

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР *Беркань І.В.*

"*12*" *05* 2025 р.

ЗАВДАННЯ

на дипломний проект

Здобувачеві освіти *Стайкуці Максиму Сергійовичу*

(прізвище, ім'я, по батькові)

1. Тема проекту *Розробка телеграм боту для перевірки знань з англійської мови на базі ШІ.*

затверджена наказом по коледжу від "*14*" *листопада* 2024р. № *246*

2. Термін здачі закінченого проекту

3. Вихідні данні до проекту

1. Передбачити зручний UX боту для забезпечення зручності та легкості;

2. Застосовувати сучасну та актуальну мову програмування Python;

3. Впровадити систему зберігання даних про користувачів;

4. Реалізувати запити до штучного інтелекту засобами API.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити)

1. Чат-боти та особливості їх використання в фокусі освіти;

2. Штучний інтелект у сфері освітніх технологій;

3. Розробка чат-боту для перевірки знань з англійської мови;

4. Економічний розрахунок; 5. Аспекти охорони праці та техніки безпеки.

5. Перелік графічного (презентаційного) матеріалу (з точним зазначенням обов'язкових креслень, кількості слайдів)

Титул; Актуальність теми та мета проекту; Класифікація чат-ботів; Роль штучного інтелекту у сфері освіти; Методи та технології ШІ в аспекті екосистеми чат-ботів; Роль ШІ у чат-ботах; Алгоритм розробки чат-боту для перевірки знань з англійської мови; Список використаних технологій; Схема роботи боту; Робота з Telegram API; Логіка контроллерів та робота з базою даних; Робота з API штучного інтелекту; Демонстрація роботи чат-боту; Демонстрація QR кодом.

6. Консультанти по проекту, із зазначенням розділів проекту, що їх стосується

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Основний розділ	*Закроєв Ю.М.*		
Економічний розділ	*Канський М.Ю.*		
Розділ охорони праці	*Чорновол Н.І.*		
Нормоконтроль	*Петрашова В.І.*		
Старший консультант	*Кривченко Ю.В.*		

7. Дата видачі завдання 12.05.2025

Керівник *Закроєв Ю.М.* _____
(підпис)

Завдання прийняв до виконання *Стайкуца М.С.* _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/р	Назва етапів дипломного проекту	Термін виконання етапів дипломного проекту (роботи)	Відмітка про виконання
1	*Формування вступу*	15.05.2025	виконано
2	*Дослідження предметної області*	16.05.2025	виконано
3	*Огляд аналогів*	19.05.2025	виконано
4	*Вибір технічної літератури*	21.05.2025	виконано
5	*Аналіз технологій розробки*	23.05.2025	виконано
6	*Проєктування архітектури боту*	26.05.2025	виконано
7	*Розробка боту*	02.06.2025	виконано
8	*Тестування створеного боту*	05.06.2025	виконано
9	*Оформлення пояснювальної записки*	06.06.2025	виконано
10	*Підготовка графічних матеріалів*	09.06.2025	виконано
11	*Економічний розрахунок*	10.06.2025	виконано
12	*Опис аспектів охорони праці та техніки безпеки*	12.06.2025	виконано
13	*Підведення висновків*	13.06.2025	виконано
14	*Підготовка доповіді для захисту*	16.06.2025	виконано

Дипломник _____
(підпис)

Керівник _____
(підпис)

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			РП 08. 22 000. 00 ДП	Дипломний проект		
А4			РП. 08. 22 000. 00 ДП ПЗ	Пояснювальна записка	1	

Зм.	Арк.	№ докум.	Підпис	Дата	РП 22. 01 000. 00 ДП			
Розробив		Стецюра И.С.		16.06	Розробка телеграм боту для перевірки знань з англійської мови на базі ШІ	Літ.	Аркуш	Аркушів
Перевірив		Сапрєєв Ю.М.		16.06		Н Д П	4	84
Н. контр.		Петрашова В.І.		16.06		ВСП "ОТФК ОНТУ" гр.4РП-08		
Затверд.		Кривченко Ю.В.		16.06				

ЗМІСТ

Вступ 7
1 Основний розділ 8
1.1 Чат-боти та особливості їх використання в фокусі освіти 8
1.1.1 Роль англійської мови у сучасному цифровому суспільстві 8
1.1.2 Проблематика вивчення англійської мови та можливі рішення 10
1.1.3 Базові поняття та історія розвитку чат-ботів 12
1.1.4 Класифікація чат-ботів та їх застосування в сфері освіти 16
1.2 Штучний інтелект у сфері освітніх технологій 21
1.2.1 Суть та етапи розвитку штучного інтелекту 21
1.2.2 Методи та технології ШІ в аспекті екосистеми чат-ботів 27
1.2.3 Дослідження можливостей використання ШІ в цифровій освіті 34
1.3 Розробка чат-боту для перевірки знань з англійської мови 35
1.3.1 Організація середовища 35
1.3.2 Структура проєкту 36
1.3.3 Робота з Telegram API 39
1.3.4 Структура та логіка контролерів 45
1.3.5 Інтеграція з ШІ 48
1.3.6 Робота з MongoDB. Опис моделей 49
1.3.7 Розробка сервісів 52
1.3.8 Демонстрація роботи чат-боту 55
2 Економічний розділ 60
2.1 Резюме 60
2.2 Розрахунок ціни програмного продукту нормативним методом 60
2.2.1 Визначення трудомісткості розробки програмного забезпечення 60
2.2.2 Розрахунок ціни програмного продукту 63
3 Розділ охорони праці та техніки безпеки 65
3.1 Основні положення 65
3.2 Вплив комп'ютерної роботи на здоров'я користувача 65
3.3 Гігієнічні норми 65

3.3.1 Умови приміщення для роботи з комп’ютером 65
3.3.2 Організація робочого місця при роботі з комп’ютером 66
3.3.3 Освітлення робочої зони 67
3.3.4 Рівень шуму та вібрацій у робочому середовищі 67
3.3.5 Мікрокліматичні умови на робочому місці 68
3.3.6 Дотримання вимог електробезпеки 68
3.3 Забезпечення пожежної безпеки 68
3.4 Заходи для збереження здоров’я під час роботи з комп’ютером.....69
Висновки.....70
Перелік використаних інформаційних джерел 71
Додаток А. Фрагмент програмного коду боту та сервісу ШІ 72
Додаток Б. Слайди мультимедійної презентації 77

ВСТУП

Стрімкий розвиток інформаційних технологій сприяє зростанню популярності чат-ботів як ефективного інструменту взаємодії між людиною та комп'ютерними системами. Вони використовуються у різних сферах – від електронної комерції до освіти, забезпечуючи швидкий та зручний доступ до інформації і послуг. Завдяки інтеграції штучного інтелекту чат-боти здатні адаптуватися до користувача, вести природну розмову та навіть здійснювати оцінювання знань, що робить їх цінним інструментом в освітньому процесі.

Окремої уваги заслуговує вивчення англійської мови, яка є мовою міжнародного спілкування, науки, IT-сфери та бізнесу. Уміння володіти англійською відкриває широкі можливості для особистого розвитку, кар'єрного зростання та доступу до світових інформаційних ресурсів. Проте у нашому швидкоплинному світі, де людина щоденно стикається з великою кількістю інформації та має обмежений вільний час, традиційні методи вивчення мови не завжди є ефективними. Вони часто вимагають значних часових витрат, регулярного відвідування занять та інтенсивної самостійної підготовки.

У зв'язку з цим виникає потреба у створенні нових, більш гнучких та інтерактивних засобів навчання. Одним із таких рішень є чат-бот на базі штучного інтелекту, який може проводити тестування знань, допомагати у вивченні нової лексики, граматики та покращенні навичок спілкування. Такий підхід дозволяє поєднати зручність цифрових технологій із навчальним процесом, зробити вивчення англійської мови більш доступним, цікавим та ефективним.

Метою цієї дипломної роботи є розробка чат-боту для перевірки знань з англійської мови із використанням сучасних технологій штучного інтелекту, який сприятиме покращенню процесу навчання та самоконтролю користувача [1][2][3].

1 ОСНОВНИЙ РОЗДІЛ

1.1 Чат-боти та особливості їх використання в фокусі освіти

1.1.1 Роль англійської мови у сучасному цифровому суспільстві

У 21 столітті не можна не помітити, що англійська мова відіграє важливу роль у цифровому світі. Цифрова епоха змінила способи поширення, споживання та обміну інформацією в усьому світі. Англійська мова, як «лінгва франка» Інтернету, відіграє ключову роль у спілкуванні людей з різним мовним походженням. Для тих, хто вивчає англійську як другу мову (English as a Second Language – ESL), оволодіння цією мовою відкриває безліч можливостей – від доступу до освітніх ресурсів до розширення кар'єрних перспектив.

В основі цифрової трансформації лежить інтернет – величезне сховище знань і основний засіб комунікації. Англійська мова домінує у понад 50% контенту в Інтернеті, забезпечуючи спільну мову для користувачів по всьому світу. Платформи соціальних мереж, потокові сервіси та онлайн-форуми часто використовують англійську мову як мову за замовчуванням, сприяючи глобальній взаємодії та культурному обміну. Така поширеність робить розуміння та спілкування англійською незамінним у сучасну цифрову епоху.

Як для викладачів, так і для студентів, використання цифрових ресурсів у вивченні мови доводить свою високу ефективність. Інтернет пропонує цілий ряд інструментів і платформ, які задовольняють різноманітні навчальні уподобання, від інтерактивних додатків до мультимедійного контенту. Включення цих ресурсів у навчальну програму з англійської мови професійного спрямування може значно покращити процес навчання, зробити його більш цікавим та ефективним. Більше того, цифровий світ надає безліч можливостей для практики та занурення в реальному часі, що має вирішальне значення для досягнення рівня володіння мовою.

Тому розуміння ролі англійської мови в цифровій сфері є важливим для будь-якого студента, який вивчає англійську мову як іноземну. Заглиблюючись у значення англійської мови онлайн, ми можемо краще зрозуміти її вплив на

спілкування, освіту, ринок праці та глобальний зв'язок.

Однією з ключових причин, чому англійська мова є такою домінуючою силою в Інтернеті, є її роль як глобальної мови Інтернету. Багато технологічних досягнень, включаючи сам інтернет, виникли в англомовних країнах. Як наслідок, ранні фреймворки, мови кодування та контент були переважно англійською мовою. Таке раннє впровадження підготувало підґрунтя для того, щоб англійська стала мовою цифрового світу за замовчуванням.

Такі платформи, як Google, Facebook і YouTube, спочатку були запущені англійською мовою, а згодом розширилися до підтримки кількох мов. Однак їхньою основною робочою мовою залишається англійська. Ця тенденція не обмежується лише соціальними мережами та пошуковими системами. Більшість наукових статей, сайтів електронної комерції, новинних порталів і навіть документація до програмного забезпечення доступні переважно англійською мовою. Це забезпечує єдину платформу для користувачів з різних куточків світу для обміну інформацією та безперешкодного доступу до неї.

Крім того, англійська діє як мова-міст у багатомовному просторі. Наприклад, на міжнародних технічних форумах або в спільнотах програмістів учасники часто спілкуються англійською, незважаючи на те, що їхні рідні мови відрізняються. Це дозволяє об'єднувати знання та спільно вирішувати проблеми на глобальному рівні. У цей спосіб, володіння англійською мовою все частіше розглядається як критично важлива навичка для тих, хто прагне процвітати в цифровому просторі, незалежно від сфери інтересів.

До того ж, пошукова оптимізація (search engine optimization – SEO) часто надає перевагу англомовному контенту через його широке охоплення та базу користувачів. Для творців контенту та бізнесу створення англомовного контенту означає доступ до ширшої аудиторії. Отже, володіння англійською мовою може значно посилити цифрову присутність та залученість.

У професійній сфері англійська виділяється як основна мова спілкування в різних галузях. З розвитком віддаленої роботи, транснаціональних корпорацій та глобальних ринків фрілансу володіння англійською мовою стало критично

важливим, як ніколи раніше.

Електронна пошта, відеоконференції, інструменти для спільної роботи та ділова документація часто за замовчуванням складаються англійською мовою, що сприяє безперешкодній взаємодії між різними працівниками. Для професіоналів, які прагнуть досягти успіху на світових ринках, оволодіння англійською – це не лише вивчення мови, але й розуміння тональності, етикету та культурних нюансів професійного спілкування.

Цифровий світ також пропонує сучасні інструменти, такі як додатки для перекладу та програмне забезпечення для вивчення мови, які покращують робочу комунікацію. Однак, незважаючи на їхню користь, вони не можуть повністю замінити глибину розуміння, яка приходить зі справжнім знанням мови. Ефективна двомовність або навіть багатомовність, закріплена англійською мовою як основною компетенцією, може значно покращити кар'єрні перспективи та можливості для співпраці.

Портали з працевлаштування та професійні мережі, такі як LinkedIn, переважно використовують англійську мову, що дозволяє шукачам роботи орієнтуватися в більшій кількості можливостей. Оскільки компанії все частіше використовують англійську мову як мову внутрішнього спілкування, перевага надається працівникам, які володіють англійською мовою, що відображає глобальний зсув у бік більш взаємопов'язаного робочого місця [4].

1.1.2 Проблематика вивчення англійської мови та можливі рішення

Цифрова революція також змінила сферу освіти, а електронне навчання стало помітною тенденцією. Такі освітні онлайн-платформи, як Coursera, EdX та Khan Academy, пропонують свої курси переважно англійською мовою. Це пов'язано не лише з орієнтацією на англомовні навчальні заклади, а й з роллю мови як міжнародного засобу навчання.

Для студентів, які вивчають англійську мову як іноземну, це одночасно і виклик, і можливість. Хоча вивчення складних предметів нерідною мовою може бути складним завданням, це також забезпечує досвід занурення в навчальний процес, який сприяє розвитку мовних навичок. Доступ до цих ресурсів допомагає

студентам покращити свій словниковий запас, розуміння та критичне мислення в академічному контексті.

Крім того, платформи онлайн-освіти є прибутковою сферою для викладачів англійської мови професійного спрямування. Викладання англійської мови онлайн стало життєздатною кар'єрою, оскільки викладачі можуть спілкуватися зі студентами з усього світу через віртуальні класи. Зручність і гнучкість цих платформ створюють динамічне навчальне середовище, долаючи географічні та культурні розриви.

Технологічний прогрес взагалі змінив спосіб викладання та вивчення мов, у тому числі і англійської. Існує безліч цифрових інструментів та ресурсів, які допомагають студентам, що вивчають англійську мову як іноземну, ефективно розвивати свої навички.

Такі додатки для вивчення мови, як Duolingo, Babbel та Rosetta Stone, пропонують інтерактивні, гейміфіковані заняття, які роблять вивчення англійської мови цікавим та захоплюючим. Ці додатки розроблені з урахуванням принципів рознесеного повторення, зворотного зв'язку в реальному часі та персоналізованої траєкторії навчання, що сприяє більш ефективному засвоєнню мови.

Крім того, онлайн-словники та інструменти для перекладу, такі як Google Translate та Linguee, надають миттєвий доступ до значень, прикладів використання та вимови. Ці інструменти особливо корисні для початківців та студентів середнього рівня для розширення словникового запасу та покращення навичок розуміння.

Віртуальна реальність (VR) і доповнена реальність (AR) також змінили традиційні методи навчання. Ці технології створюють захоплююче середовище, де студенти можуть практикувати англійську мову в реалістичних сценаріях, від симуляції шопінгу до віртуальних подорожей. Такі стратегії занурення дуже корисні для запам'ятовування лексики та розуміння контексту.

На додаток до цих інструментів, участь в онлайн-спільнотах, таких як веб-сайти та форуми мовного обміну, пропонує практичний досвід. Спілкування з носіями мови та іншими учнями через практичні заняття, дискусії та зворотній

зв'язок допомагає розвинути впевненість і компетентність у використанні англійської мови в реальних ситуаціях [5].

1.1.3 Базові поняття та історія розвитку чат-ботів

Чат-боти – це програмні рішення, які імітують спілкування з людиною через текстовий або голосовий інтерфейс, а також вони можуть слугувати інтерфейсом для доступу до програмі, що розміщена на сервері. Вони призначені для автоматизації взаємодії з користувачами, надаючи відповіді на запитання, виконуючи команди або допомагаючи у вирішенні завдань. Чат-боти можуть бути простими скриптовими програмами, які працюють за заздалегідь визначеними сценаріями, або складнішими системами, що використовують штучний інтелект для обробки природної мови. У контексті цієї роботи основна увага приділяється простим чат-ботам, які не потребують складних технологій, що відповідає рівню для розробки чат-бота цієї роботи. Їхня головна перевага полягає в доступності, швидкості роботи та можливості інтеграції в повсякденні цифрові платформи, з якими люди вже знайомі.

Чат-ботами користується широка аудиторія: від пересічних користувачів, які шукають швидкої інформації чи розваг, до компаній, які застосовують їх для автоматизації бізнес-процесів. Наприклад, до початку 2015 у нашому університеті існував чат-бот, що допомагав швидко отримувати важливу інформацію для студентів, а викладачі, у свою чергу, могли також там знайти інформацію, що стосується їх роботи. Бізнес-сфера також активно впроваджує чат-ботів у службу підтримки клієнтів, маркетинг і продажі, що дає додатковий прибуток і надає більш цікавий користувацький досвід для своїх клієнтів (див. рис. 1.1).

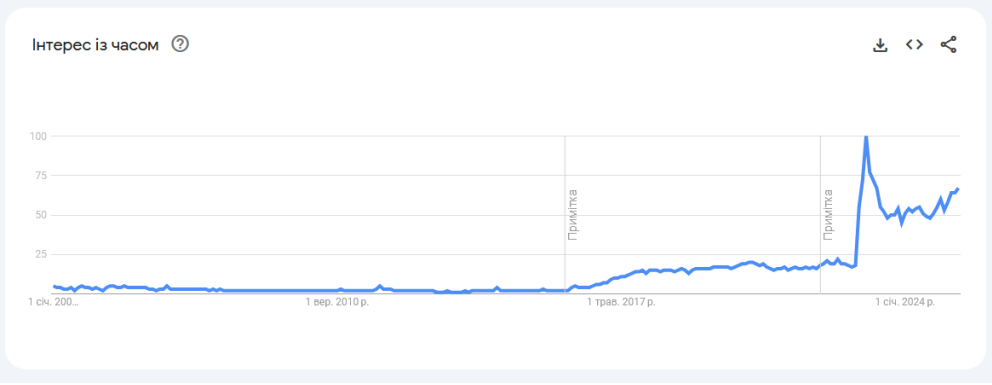


Рисунок 1.1. Статистика популярності чат-ботів

На рис. 1.1 можна побачити статистику популярності чат-ботів. Чат-боти почали свій активний розвиток починаючи з 2016 року і стає ще більш актуальним із часом. Ця статистка відображає користувацький інтерес до цієї тематики, а як слідство – статистику створення нових чат-ботів, бо всім відомо, що будь-який попит породжує пропозицію.

Популярність чат-ботів стрімко зростає з кількох причин. По-перше, розвиток месенджерів, таких як Telegram, WhatsApp і Viber, створив ідеальне середовище для їхнього розгортання, адже користувачам не потрібно встановлювати додаткові програми, а лиш достатньо відкрити знайомий додаток, що вже встановлений на їх пристрій. По-друге, чат-боти економлять час і ресурси – вони можуть одночасно обробляти запити тисяч людей, що робить їх незамінними для компаній і організацій різних масштабів. Зростання цифрової грамотності населення сприяє тому, що люди все частіше звертаються до автоматизованих рішень для вирішення рутинних завдань (див. рис. 1.2).

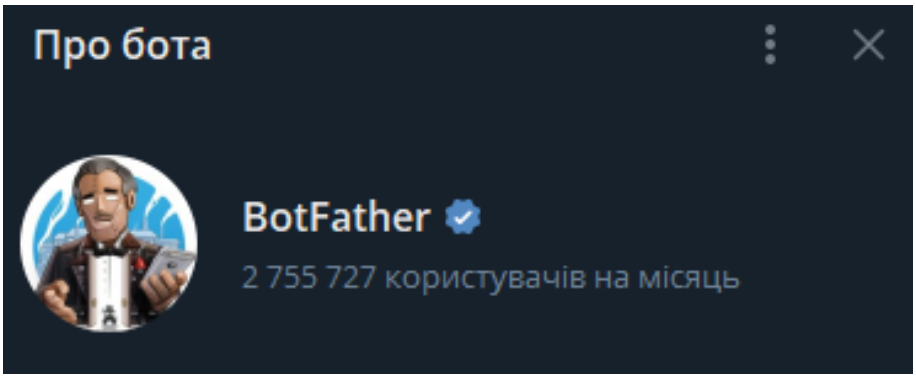


Рисунок 1.2. Інформація про BotFather

На рис. 1.2 можна побачити інформацію про офіційного чат-бота Telegram – BotFather. Це офіційний чат-бот, що дозволяє людям реєструвати нових чат-ботів у цьому месенджері, а також редагувати інформацію про вже створеного чат-бота. На рисунку також можна побачити, що кількість користувачів складає понад 2.5 мільйони за останній місяць, що гарно відображає великий попит на технологію чат-ботів у цілому.

Розробка чат-ботів може відбуватися різними способами, залежно від потреб і технічних можливостей розробника. Існує два основних підходи: використання конструкторів і написання коду вручну. Конструктори – це онлайн-платформи, які

дозволяють створювати чат-ботів без знання програмування. Серед популярних сервісів можна виділити Botmaker, ManyChat, Chatfuel і BotMother. Ці інструменти пропонують інтуїтивно зрозумілий інтерфейс, де користувач налаштовує сценарії за допомогою drag-and-drop елементів, додавати кнопки, тексти чи медіафайли. Наприклад, у ManyChat можна створити бота для Telegram, який вітає користувача і пропонує меню з опціями. Такі платформи ідеально підходять для новачків або невеликих проєктів, вони мають велике обмеження у функціональності, складні логічні операції чи інтеграції зазвичай недоступні, оскільки універсальний алгоритм дій для всіх користувачів побудувати неможливо (див. рис. 1.3).

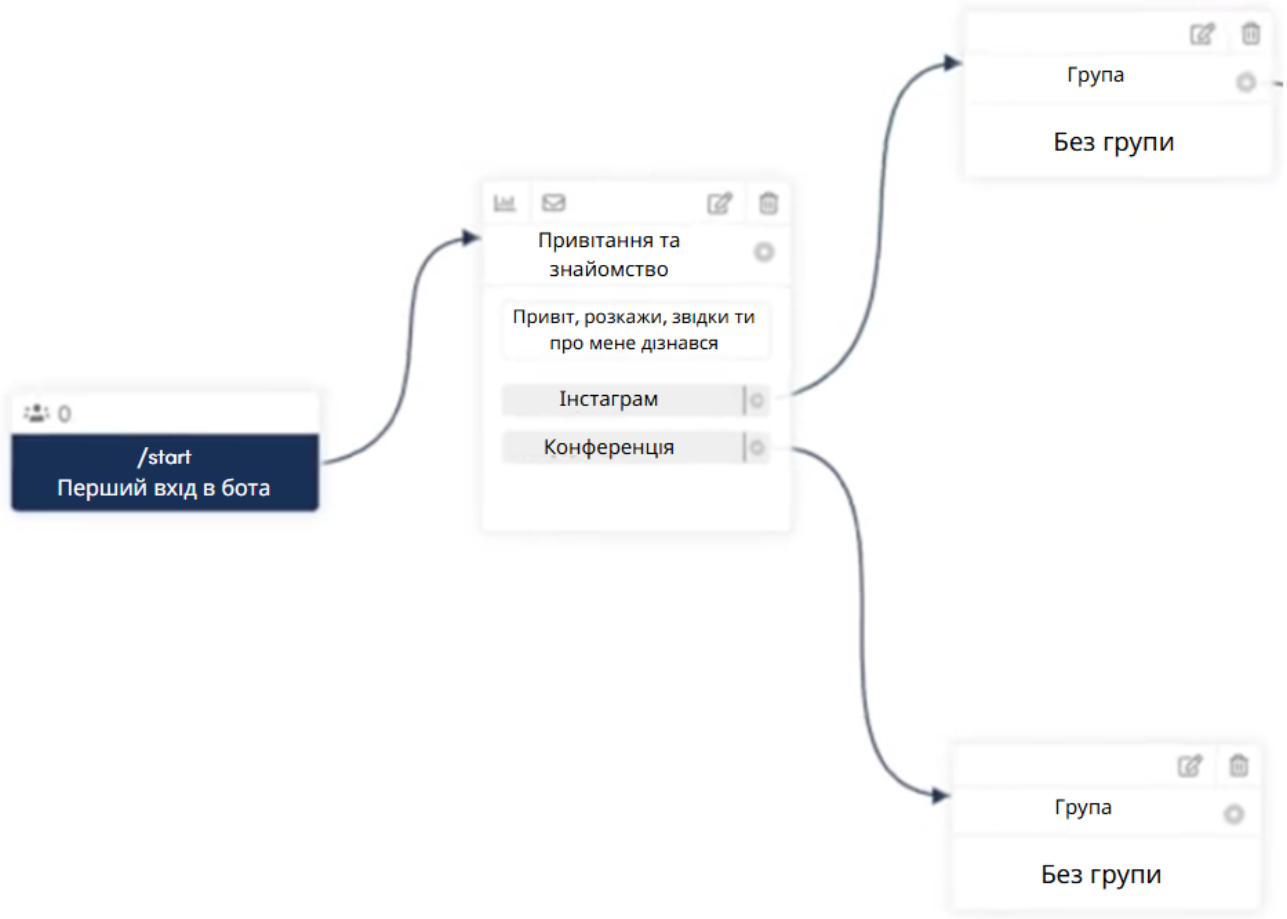


Рисунок 1.3. Приклад створення чат-боту в Botmaker

Другий підхід – розробка чат-ботів вручну за допомогою програмування. Цей метод вимагає знання мов програмування, таких як Python, JavaScript або PHP, а також розуміння принципів взаємодії з API. У випадку Telegram-ботів однією з можливостей є використання HTTP-запитів безпосередньо до Telegram Bot API, що

реалізується за допомогою бібліотеки requests. Цей підхід дозволяє надсилати та отримувати дані шляхом побудови власних запитів до серверів Telegram, забезпечуючи гнучкість та повний контроль над логікою роботи бота. Розробник самостійно реалізує обробку повідомлень, відповіді на команди, інтеграцію з базами даних та інші функції. Основна перевага такого способу – можливість реалізації нестандартного функціоналу, зокрема систем тестування, аналізу відповідей та збереження результатів. Водночас він потребує більше часу на реалізацію та відповідного технічного рівня. У цій роботі обрано саме програмний підхід із використанням бібліотеки requests, що дозволило реалізувати гнучку систему перевірки знань з англійської мови.

Важливим аспектом розробки чат-ботів є вибір платформи, на якій вони розміщуватимуться. Серед найпоширеніших платформ можна виділити Telegram, Viber, WhatsApp, Discord і Facebook Messenger. Telegram вирізняється простотою API, широкою аудиторією та підтримкою ботів через раніше згаданий BotFather. Наприклад, розробник може отримати токен для бота за кілька хвилин і одразу почати програмувати. Viber також популярний, особливо в Україні, завдяки своїм функціям масового розсилання та зручному інтерфейсу, але його API менш гнучке порівняно з Telegram. WhatsApp більше орієнтований на бізнес-користувачів і має складніший процес інтеграції через офіційний Business API. Discord підходить для геймерів і спільнот, але менш універсальний для масового використання. Facebook Messenger, хоча й має велику аудиторію, вимагає дотримання суворих правил платформи. Кожна з цих платформ має свої переваги та недоліки, які необхідно враховувати під час розробки.

Розробка чат-ботів вручну відкриває широкі можливості для кастомізації. У Python, наприклад, замість використання готових бібліотек, таких як pyTelegramBotAPI, можливо безпосередньо взаємодіяти з Telegram Bot API за допомогою бібліотеки requests. Такий підхід дозволяє вручну обробляти запити користувачів, надсилати відповіді, керувати повідомленнями, а також реалізовувати індивідуальну логіку для кожної функції бота, включаючи команди типу /start або /help. Для JavaScript популярним інструментом є бібліотека

Telegraf.js, яка значно спрощує роботу з Telegram API. У PHP також можлива реалізація ботів, особливо в контексті інтеграції з веб-додатками, хоча він менш активно використовується у сфері месенджер-ботів. З іншого боку, конструктори, такі як Chatfuel, дають змогу швидко запустити бота без необхідності програмування – достатньо обрати шаблон, налаштувати логіку відповідей і підключити платформу. Обидва підходи мають свої переваги: конструктори економлять час і зручні для типових задач, тоді як програмний підхід забезпечує високу гнучкість. У контексті реалізації освітнього чат-бота для перевірки знань з англійської мови було обрано програмний підхід із використанням бібліотеки requests, що дало змогу створити адаптований функціонал з перевіркою відповідей, аналізом статистики та інтеграцією з базою даних.

1.1.4 Класифікація чат-ботів та їх застосування в сфері освіти

Чат-боти можна поділити на два види:

- звичайні чат-боти;
- юзер-боти.

Звичайні чат-боти – це ті боти, яких звичайна людина може розпізнати майже одразу, якщо, звісно, не з першого погляду.

Для прикладу, можна звернути увагу на чат-бота Укрпошти (рис. 1.4). Вони мають гарний користувацький інтерфейс та зрозумілі описані функції, які прописує розробник в довідці, яку можна отримати завдяки командам.

Звичайних чат-ботів створюють для того, щоб зробити систему відповідей на питання, як це зроблено в чат-боті Дія, поширення інформації, викликів операторів чату, які зможуть допомогти із проблемою тощо. Чат-ботів вигідно розробляти підприємствам та установам, які бажають покращити свій сервіс, зробити його більш комфортним для звичайних користувачів (див. рис. 1.4).

Вони не потребують складної інтеграції з акаунтом користувача та зазвичай працюють через інтерфейс популярних месенджерів. Звичайні чат-боти мають обмежений набір функцій, який відповідає найбільш поширеним запитам користувачів. Їх можна швидко оновлювати та адаптувати під нові потреби завдяки простоті реалізації. Такі боти не мають доступу до особистих даних користувача,

що робить їх безпечнішими в загальнодоступних сервісах.

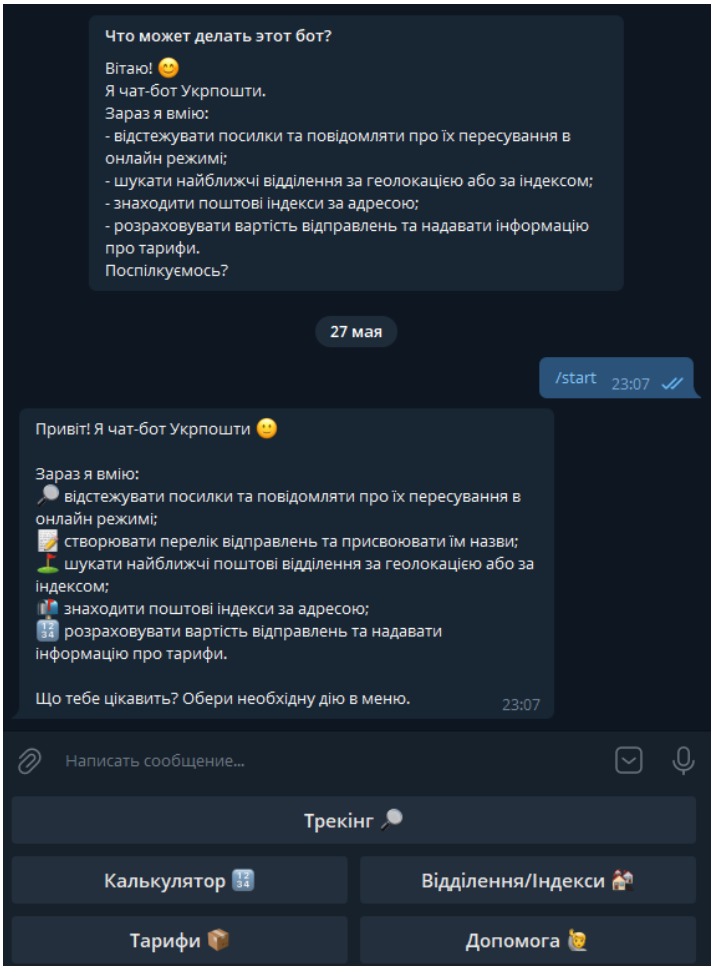


Рисунок 1.4. Бот Укрпошти

Інший вид – юзер-боти. Цей вид також може мати частковий функціонал звичайного чат-боту, але він не буде мати гарного інтерфейсу, адже платформи, наприклад, Telegram, дає можливість робити його тільки в звичайних чат-ботах, завдяки створенню різних кнопок, зручного виклику меню команд.

Завдяки юзер-ботам можна керувати своїм аккаунтом в месенджері, наприклад, міняти ім’я через написану програму або, зробити годинник із статусу “про мене”.

Більшість людей не використовують цей тип серйозно, а створюють щось потішне чи дуже вузькоспеціалізоване «під себе».

Якщо ж намагатися поділяти чат-ботів на більш конкретні категорії, то логічно класифікувати їх, розрізняючи за певними критеріями. Дослідження та аналіз особливостей сучасних чат-ботів дали змогу поділити програми чат-ботів на сім класів (див. рис. 1.5, 1.6, 1.7 та 1.8).

Рисунок 1.5. Класифікація чат-ботів

За призначенням:

Чат-боти для розмов на широкий спектр тем – призначені для ведення діалогу з користувачем на абстрактні теми та не мають чіткої мети;

Чат-боти, зосереджені на діалозі лише з конкретної теми або для розв'язання певної задачі чи досягнення мети – найпоширеніші. Наприклад, для регулярного розповсюдження інформації, встановлення нагадувань тощо.

За розташуванням:

На сайтах. Найчастіше компанії вбудовують чат-ботів на свої веб-сайти, щоб допомогти клієнтам отримати відповіді на запитання або вирішити інші комунікаційні запити чи проблеми, пов'язані з унікальними завданнями чи налаштуваннями;

У месенджерах. В основному використовуються для швидкої взаємодії з клієнтами, навіть за умов повільного інтернету або в роумінгу.

У спеціалізованих програмних застосунках. Використання чат-ботів полегшує та пришвидшує процес замовлення товарів або послуг.

Чат-боти, зосереджені на конкретних завданнях, зазвичай мають простішу логіку та зрозумілий сценарій взаємодії з користувачем. Вони часто інтегруються з внутрішніми системами компаній для автоматизації обробки запитів. Розміщення чат-бота на сайті дозволяє зменшити навантаження на службу підтримки. У месенджерах чат-боти працюють зручніше для користувачів, оскільки не потребують переходу на сторонні платформи. У спеціалізованих застосунках вони здатні обробляти індивідуальні запити на основі облікового запису користувача. Усі ці варіанти реалізації спрямовані на підвищення ефективності комунікації між сервісом і клієнтом.

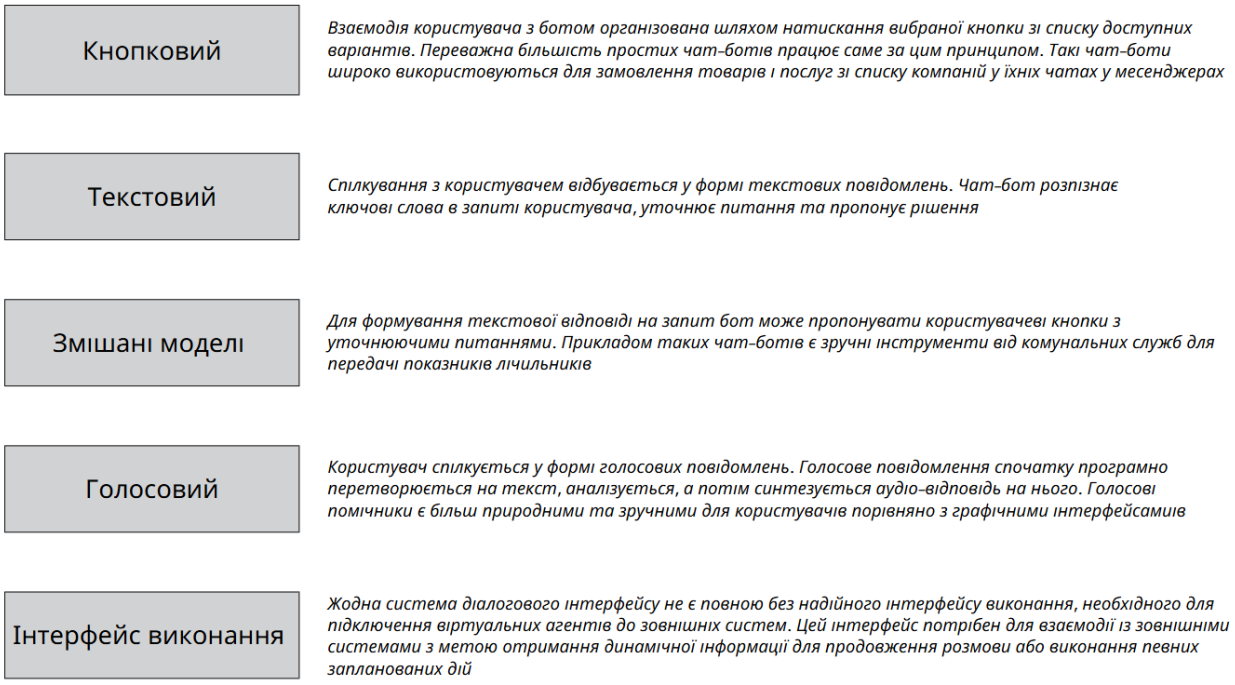


Рисунок 1.6. Класифікація чат-ботів за типом інтерфейсу

За кількістю користувачів:

Персональні чат-боти

Персональні боти можна поділити на дві групи: Для особистого використання без передачі даних іншим. Це можуть бути особисті сховища систематизованих даних;

Інтерактивні чат-боти – своєрідні помічники користувача у взаємодії (обміні даними) з іншими користувачами або іншими програмами для виконання певних дій від імені користувача, наприклад, для управління календарем, відправлення текстів, прийому особистих дзвінків, пошуку та відтворення аудіо та відеофайлів тощо;

Бізнес чат-боти призначені для забезпечення одночасного використання в бізнесі для автоматичного спілкування з багатьма клієнтами без залучення вручну працівників компанії до процесу обслуговування. Такі чат-боти використовуються в багатьох сферах бізнесу для автоматизації комунікаційних комерційних процесів з клієнтами.

Використання персональних чат-ботів допомагає оптимізувати повсякденні завдання та зменшити навантаження на користувача. Бізнес чат-боти, у свою чергу, сприяють підвищенню ефективності обслуговування клієнтів і скороченню витрат

на персонал.

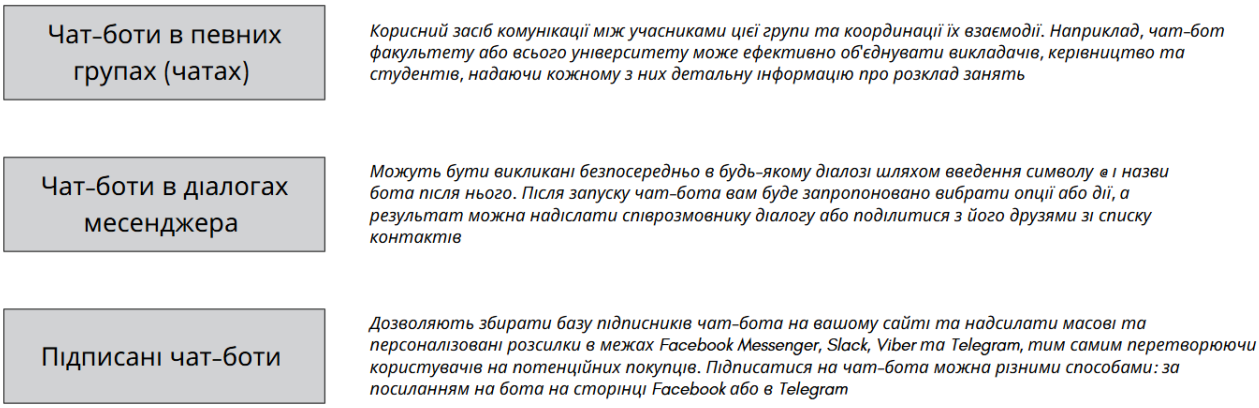


Рисунок 1.7. Класифікація чат-ботів за формою доступу

За алгоритмом:

Прості (обмежені) чат-боти взаємодіють з користувачами за заздалегідь підготовленим сценарієм – деревом рішень, яке містить набір відповідей на поширені питання, тобто відповіді вибираються з шаблонних фраз сценарію за ключовими словами. Якщо користувач не використовує ключові слова під час спілкування, бот не розуміє його і виконує дії, передбачені для таких випадків, наприклад, пропонує звернутися до оператора. Чат-боти цього типу зазвичай уникатимуть питань, що потребують вільних відповідей, і замість цього містять велику кількість кнопок;

Інтелектуальні ("розумні") чат-боти базуються на штучній нейронній мережі, яка "розуміє" зміст розмови. Шлях розмови визначається неявно на основі навчальних даних (навчальних зразків), які використовуються для навчання моделі машинного навчання. Такі програмні помічники розробляються індивідуально і набагато дорожчі в розробці, оскільки для створення якісного чат-бота вклали велику кількість зусиль у розвиток штучного інтелекту (ШІ) і нейронних мереж машинного навчання. Основу, на якій будується інтелектуальний чат-бот, складають NLP (Обробка природної мови), NLU (Розуміння природної мови) та NLG (Генерація природної мови). NLP – це здатність машини обробляти сказане, розуміти його зміст, визначати необхідні дії у відповідь і відповідати мовою, зрозумілою користувачу, перетворюючи комп'ютерний текст на структуровані

дані. NLU є основою будь-якого чат-бота і є підмножиною процесів NLP;

Гібридні чат-боти є поєднанням перших двох типів чат-ботів. Боти цього типу спілкуються з користувачем за заздалегідь визначеним сценарієм, але використовують ШІ для розпізнавання намірів користувача, а також для вилучення цінних даних з повідомлень користувача. Цей тип чат-ботів є найбільш широко використовуваним у комерційних застосуваннях [6].

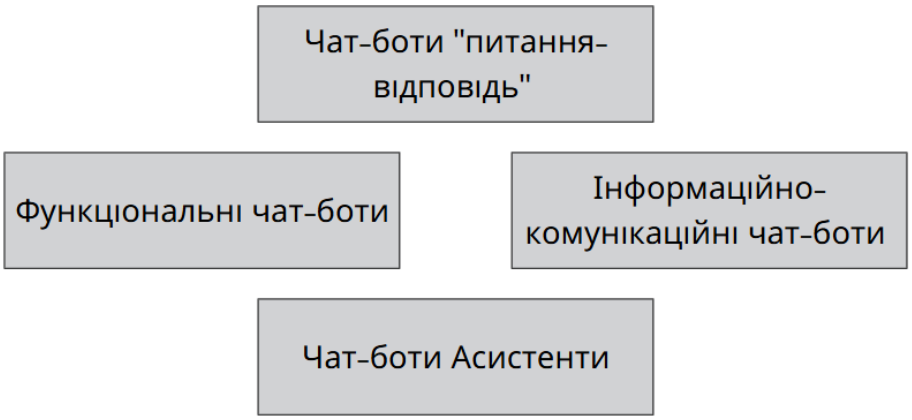


Рисунок 1.8. Класифікація чат-ботів за типом функціональності

1.2 Штучний інтелект у сфері освітніх технологій

1.2.1 Суть та етапи розвитку штучного інтелекту

Штучний інтелект (ШІ) – це галузь комп'ютерної науки, що зосереджується на створенні систем, здатних виконувати завдання, які зазвичай потребують людського інтелекту. До таких завдань належать розпізнавання мови, розуміння природної мови, виявлення закономірностей у даних, планування, навчання з досвіду, прийняття рішень тощо. Загалом, суть ШІ полягає в імітації когнітивних функцій людини за допомогою алгоритмів і моделей машинного навчання.

Концепція штучного інтелекту бере свій початок ще з 1950-х років, коли Алан Тюрінг поставив питання: «Чи може машина думати?» Відтоді розвиток цієї сфери пройшов шлях від простих математичних моделей до складних нейромереж, які здатні самостійно навчатися, аналізувати інформацію та генерувати нові рішення. Сучасне розуміння ШІ охоплює не лише програмування алгоритмів, а й створення систем, які можуть адаптуватися до нових ситуацій та вдосконалювати свою ефективність без безпосереднього втручання людини.

З технічної точки зору, ШІ – це поєднання підходів і технологій, включно з математичним моделюванням, статистикою, нейромережами та обробкою великих обсягів даних. Алгоритми штучного інтелекту здатні виводити нові знання на основі існуючих даних, а також підлаштовуватись під зміну вхідних умов. Це робить їх особливо цінними в умовах інформаційного перевантаження, де людина не завжди може швидко й точно ухвалити рішення.

Штучний інтелект охоплює кілька ключових напрямків, кожен з яких вирішує специфічні завдання. Серед найважливіших варто виокремити машинне навчання (machine learning), обробку природної мови (natural language processing), комп'ютерний зір (computer vision), робототехніку (robotics) та експертні системи. Кожен із цих напрямків має свою унікальну сферу застосування, але часто вони взаємопов'язані в рамках комплексних рішень (рис. 1.9).

Штучний інтелект у сучасному світі не обмежується лише складними теоріями чи технічними системами – він давно став частиною повсякденної реальності, зокрема й в освіті. Одним із найпотужніших інструментів, який допомагає системам ставати розумнішими, є машинне навчання. Завдяки йому комп'ютери не просто виконують заздалегідь задані команди, а самі вчаться на прикладах. Уявімо платформу, яка з кожним використанням краще розуміє, які завдання давати студенту – простіші чи складніші. Це результат роботи алгоритмів, що аналізують відповіді, знаходять закономірності та на їх основі формують рекомендації.

Ще один напрямок, який буквально дозволяє комп'ютерам "заговорити" з людиною – це обробка природної мови. Завдяки цьому підходу ми маємо чат-ботів, які не просто реагують на ключові слова, а намагаються зрозуміти сенс того, що пише людина. Це вже не сухий набір заготовлених фраз, а діалогові системи, які вміють вести розмову, пояснювати складні теми чи навіть створювати текст за запитом. Особливо помітний прорив стався останніми роками, коли з'явились потужні мовні моделі – саме вони стоять за такими інструментами, як сучасні освітні помічники.

Є й інші напрямки, де штучний інтелект тісно пов'язаний із візуальним

сприйняттям. Наприклад, здатність розпізнавати зображення, відео чи навіть рукописний текст. У сфері освіти це відкриває нові можливості – наприклад, системи можуть автоматично зчитувати та оцінювати відповіді, написані від руки, або допомагати дітям з особливими освітніми потребами через візуальну взаємодію.

Навіть робототехніка, яка на перший погляд здається далекою від навчання, теж активно проникає у цю сферу. Роботи, що допомагають дітям вивчати програмування, реагують на дії учнів, пояснюють помилки або демонструють приклади розв'язання завдань – усе це теж результат поєднання штучного інтелекту з фізичними пристроями.

Існують і системи, які не мають фізичної оболонки, але працюють як досвідчені консультанти. Вони можуть аналізувати знання учня, пропонувати наступний крок у навчанні або навіть автоматично складати нові завдання. Такі «невидимі помічники» особливо цінні в онлайн-освіті, де важко забезпечити індивідуальний підхід до кожного користувача без додаткових інструментів.

Загалом, усі ці технології – хоч і дуже різні – працюють на спільну мету: зробити освіту гнучкішою, доступнішою та ефективнішою. І навіть якщо вони не замінять учителя, вони вже сьогодні здатні суттєво посилити навчальний процес.

Завдяки стрімкому розвитку кожен із цих напрямків поглиблює інтеграцію ШІ в різні аспекти життя, зокрема – в освітні технології. Вони створюють передумови для появи нових моделей взаємодії між людиною і машиною, серед яких освітні чат-боти є яскравим прикладом (див. рис. 1.9).

Штучний інтелект продовжує активно трансформувати традиційні підходи до навчання та викладання. Його інтеграція дозволяє враховувати індивідуальні потреби кожного учня в режимі реального часу. Це сприяє формуванню персоналізованих маршрутів навчання, які адаптуються до рівня знань і темпу засвоєння матеріалу. Використання ШІ також допомагає зменшити навантаження на педагогів, автоматизуючи перевірку робіт та аналіз результатів. У цей спосіб, штучний інтелект стає потужним інструментом підвищення якості та доступності освіти.

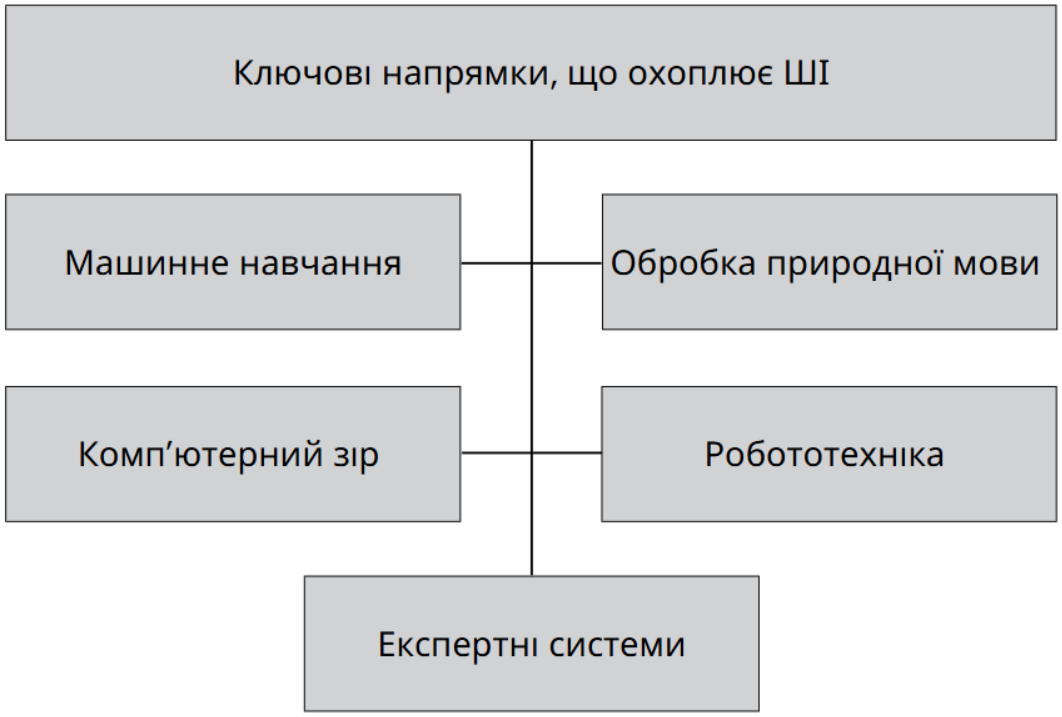


Рисунок 1.9. Ключові напрямки, що охоплює ШІ

Коли ми говоримо про штучний інтелект, часто маємо на увазі дуже різні речі. Одні системи можуть відповідати на запитання в чаті, інші – прогнозувати погоду або пропонувати фільми для вечірнього перегляду. Щоб краще зрозуміти цей спектр можливостей, дослідники умовно поділяють ШІ за рівнем складності – від базових алгоритмів до гіпотетичних надможливостей, про які поки що більше дискутують, ніж застосовують.

У центрі сучасних технологій – так званий «вузький» ШІ. Це системи, які створені для вирішення конкретного завдання. Вони не «думають» у звичному розумінні, але надзвичайно ефективно справляються зі своєю роллю. Наприклад, мовні помічники, рекомендації у відеосервісах чи чат-боти, які допомагають перевіряти знання – усе це приклади технологій, що вже інтегровані у наше життя. У межах навчальних платформ саме такі інструменти забезпечують індивідуальний підхід до кожного учня, адаптуючи контент залежно від рівня знань.

Але є й більш амбітні уявлення про ШІ – ті, що стосуються створення універсального розуму. Ідея в тому, щоб машина могла навчатися й мислити так само гнучко, як людина: не лише вирішувати задачі, а й самостійно шукати нові

підходи, аналізувати контексти, змінювати напрямок дій. Такий «загальний» ШІ поки залишається на рівні досліджень. Проте навіть гіпотетична поява подібних систем суттєво змінить підхід до навчання, адже вони зможуть не просто допомагати, а й повноцінно співпрацювати з учнями.

Ще далі йде уява про надінтелект – щось, що перевершить найрозумнішу людину в усіх можливих сферах. Це вже не про технології, які існують, а радше про етичні та філософські питання: як контролювати такі системи, хто нестиме відповідальність за їхні дії і чи маємо ми моральне право створювати щось подібне. Для освіти такі роздуми ще дуже далекі, але варто розуміти – усе починається з простих кроків.

Цікаво, що ШІ поділяють не лише за рівнем, а й за способом мислення, якщо можна так сказати. Одні системи працюють за чіткими правилами – якщо трапиться А, то зроби Б. Інші ж, навпаки, вчаться на прикладах, виявляють закономірності без прямої інструкції. Саме такий підхід, оснований на нейронних мережах і глибокому навчанні, зараз домінує у сфері створення освітніх чат-ботів. Завдяки цьому вони можуть розпізнавати мову, адаптувати відповіді, реагувати на контекст і вдосконалюватись із часом.

Коли розроблявся дипломний проєкт, ключовим було розуміння, який саме рівень ШІ буде доцільним. У підсумку було обрано саме вузький варіант – систему, яка ефективно працює в межах поставленого завдання: аналізує відповіді, ставить нові запитання, коригує складність. Вона не має амбіцій стати всезнаючим помічником, але вже зараз здатна значно полегшити навчальний процес і зробити його більш індивідуалізованим.

Інтенсивне впровадження штучного інтелекту в усі сфери життя породжує не лише технологічні, а й глибоко етичні, правові та соціальні питання. Особливо чутливою є ця тематика в контексті освіти, де йдеться про формування критичного мислення, безпеку персональних даних, доступність знань і збереження людського впливу в процесі навчання.

Однією з головних етичних проблем є прозорість рішень, які приймає ШІ. Більшість сучасних алгоритмів, особливо побудованих на основі глибокого

навчання, функціонують як «чорні скриньки» – складно пояснити, чому саме була надана певна відповідь. Це ускладнює верифікацію результатів та може підривати довіру до системи, особливо коли вона використовується в освітньому процесі.

Другим важливим аспектом є конфіденційність та захист даних. Системи ШІ часто потребують великого обсягу персональної інформації – зокрема, у чат-ботах з перевірки знань може зберігатися історія відповідей, вибрані теми, рівень складності тощо. Порушення захисту цих даних може призвести до витоку приватної інформації. Тому особливої уваги потребує дотримання принципів GDPR(General Data Protection Regulation) або аналогічних стандартів при проєктуванні та впровадженні таких систем.

Не менш важливою є доступність технологій. Незважаючи на зростаюче поширення цифрових інструментів, частина населення досі обмежена в доступі до високошвидкісного Інтернету або сучасних пристроїв. Це створює ризик освітньої нерівності, якщо системи ШІ будуть розроблені без урахування цієї проблеми.

Також обговорюється вплив ШІ на роль викладача. Існують побоювання, що автоматизовані платформи можуть частково витіснити людину з освітнього процесу. Проте більшість дослідників вказує на те, що ШІ повинен розглядатися як інструмент підтримки – він може звільнити викладача від рутинних завдань і дозволити більше зосередитись на індивідуальній роботі з учнями.

Нарешті, у дискусії часто порушується питання упередженості алгоритмів. ШІ-системи навчаються на історичних даних, які можуть містити соціальні, мовні або культурні упередження. В результаті система може демонструвати некоректні результати – наприклад, при автоматичній оцінці рівня знань чи підборі матеріалу.

Загалом, упровадження ШІ в освіту вимагає не лише технічної майстерності, але й відповідального підходу до дизайну, тестування та етичного аналізу систем. Лише збалансоване поєднання інновацій та гуманітарних принципів дозволить реалізувати потенціал ШІ у навчанні без шкоди для цінностей та прав людини (див. рис. 1.10) [7]. Впровадження ШІ в освіту потребує не лише технологічного розвитку, а й глибокого розуміння етичних, правових і соціальних наслідків його застосування.

Рисунок 1.10. Основні етичні проблеми, викликані у навчальній сфері розвитком ШІ

1.2.2 Методи та технології ШІ в аспекті екосистеми чат-ботів

Історія розвитку чат-ботів демонструє чіткий перехід від простих алгоритмічних рішень до складних систем, заснованих на штучному інтелекті. Найперші чат-боти функціонували виключно на основі заздалегідь прописаних правил. Такі системи, відомі як rule-based чат-боти, використовували шаблони ключових слів та сценарії, які визначали реакцію на повідомлення користувача. Наприклад, якщо користувач вводив фразу «Привіт», бот відповідав «Добрий день!». Вся логіка роботи таких систем обмежувалась умовними операторами типу «if-then», і будь-яке відхилення від очікуваного вводу призводило до збоїв у взаємодії.

Rule-based підхід мав певні переваги: простоту реалізації, передбачуваність відповідей і високу контрольованість поведінки бота. Водночас, він був надзвичайно обмеженим у масштабованості та неспроможним працювати в умовах природної, живої мови. Такі чат-боти не могли «зрозуміти» контекст, перефразування чи граматичні помилки – що робило їх непридатними для складних комунікаційних задач, зокрема в освітньому середовищі.

З розвитком машинного навчання та обробки природної мови з’явилися AI-based чат-боти, які здатні до адаптивного діалогу та контекстної взаємодії. Ці системи базуються на використанні алгоритмів машинного навчання, зокрема методів навчання з учителем (supervised learning), підкріплення (reinforcement

learning) і нейронних мереж. На відміну від rule-based підходів, інтелектуальні боти не покладаються виключно на жорстко закодовані правила, а здатні аналізувати великі обсяги даних, виявляти закономірності у мові та формувати відповіді на основі попереднього досвіду.

Переломним етапом у розвитку AI-базованих ботів стала інтеграція Natural Language Processing (NLP). Саме завдяки NLP чат-боти отримали змогу «розуміти» значення тексту, виділяти сутності, класифікувати наміри користувача та навіть формувати нові текстові конструкції. Додатковий імпульс дало використання глибинного навчання та мовних моделей нового покоління, таких як BERT, GPT, T5 – які можуть ефективно опрацьовувати контекст, підтримувати діалог і навіть навчатися на ходу.

Сьогодні архітектура чат-ботів набагато складніша, ніж у перших rule-based систем. Вона складається з багатьох шарів: модулів аналізу вводу, управління діалогом, генерації відповіді, підключення до баз знань чи сторонніх сервісів. У цей спосіб, еволюція чат-ботів від умовно «реактивних» до «когнітивних» систем дозволила вивести їх за межі допоміжних сервісів – і зробити повноцінними учасниками освітнього процесу.

Щоб реалізувати таку гнучкість і багатофункціональність, сучасні чат-боти будуються як комплексні системи, де кожен елемент виконує чітко визначену роль. Саме злагоджена взаємодія між цими модулями – від розпізнавання мовного вводу до прийняття рішення про відповідь – і визначає, наскільки ефективною буде взаємодія з користувачем. Така модульність дозволяє адаптувати ботів до різних освітніх сценаріїв, поступово ускладнювати їхню поведінку та розширювати функціонал відповідно до потреб навчального середовища.

Одним із ключових модулів є вище зазначена система обробки природної мови (Natural Language Processing, NLP). Саме вона відповідає за перетворення текстового вводу користувача у структуровану інформацію, яку може опрацювати система. NLP-процеси охоплюють сегментацію тексту, токенізацію, аналіз частин мови, розпізнавання іменованих сутностей (наприклад, імен, дат або локацій), а також визначення інтенції (наміру) користувача. Без цього компоненту бот не

здатен «зрозуміти» зміст повідомлення й адекватно відреагувати.

Наступним важливим елементом виступає менеджер діалогу (dialog manager). Це логічний центр чат-бота, який керує контекстом розмови, зберігає стан діалогу та приймає рішення про наступну дію бота. Менеджер відстежує історію взаємодії, враховує попередні повідомлення, іноді – дані з бази користувачів, і формує логіку реагування на основі заданих сценаріїв або машинного навчання. Його робота дозволяє системі підтримувати осмислений, послідовний діалог навіть у довших розмовах.

Ще одним важливим модулем є механізм генерації відповідей (response generation module). У більшості систем відповідь формують або на основі шаблонів, або за допомогою мовних моделей, які використовують машинне навчання для побудови природної мови. Сучасні AI-боти здатні створювати відповіді не лише за заздалегідь підготовленими зразками, а й динамічно – залежно від контексту та змісту повідомлення. Це робить спілкування з ботом більш гнучким, природним і наближеним до людської мови.

Окрім цього, важливу роль відіграє інтерфейс підключення до бази знань або зовнішніх API. Інтелектуальні чат-боти часто використовують зовнішні джерела інформації – від баз даних користувачів до енциклопедичних чи мовних ресурсів. Завдяки такій інтеграції система може надавати користувачам фактичну інформацію, перевіряти правильність відповідей, оновлювати прогрес або пропонувати персоналізовані рекомендації. У випадку освітніх ботів це можуть бути бази запитань, словники, системи тестування або аналітики.

У цей спосіб, ефективна робота інтелектуального чат-бота базується на тісній взаємодії кількох технологічних компонентів. У сукупності вони забезпечують здатність системи «розуміти» мову, підтримувати логіку діалогу, формувати осмислені відповіді та використовувати зовнішні знання для покращення взаємодії з користувачем.

Щоб ці процеси були справді динамічними та адаптивними, необхідно залучати алгоритми, здатні вчитися з досвіду. Саме тому машинне навчання стало основою сучасних інтелектуальних чат-ботів. Воно дає змогу не лише

налаштовувати поведінку системи під конкретного користувача, а й покращувати якість взаємодії з часом, виявляючи нові звички, помилки чи очікування співрозмовника. Залежно від цілей розробки, можуть застосовуватись різні підходи до навчання моделей – від заздалегідь підготовлених прикладів до адаптації на основі зворотного зв'язку під час спілкування.

Одним із найрозповсюдженіших є кероване навчання (supervised learning). У цьому випадку модель навчається на розмічених даних, де кожен приклад містить вхідні повідомлення та правильні реакції (відповіді або дії). Завдяки цьому чат-бот здатен з високою точністю класифікувати інтенції користувача, обирати правильну відповідь із наявного набору або навіть формулювати її динамічно. Сценарії такого навчання часто застосовуються в ботах служби підтримки або в освітніх системах, де передбачена велика кількість прикладів взаємодії.

Іншим перспективним підходом є навчання з підкріпленням (reinforcement learning). У цій парадигмі бот «вчиться» через досвід, отримуючи винагороду за правильні дії або штраф за помилки. Такі алгоритми дозволяють адаптувати поведінку чат-бота до складніших діалогів, коли рішення залежать від довгострокових наслідків. Наприклад, у навчальному чат-боті це може означати підвищення рівня залученості користувача або кращі результати проходження тестів. Хоча цей метод є складнішим у реалізації, він забезпечує гнучкість і еволюцію поведінки бота в реальному середовищі.

Для реалізації таких підходів використовуються різноманітні моделі машинного навчання. Серед класичних варто згадати дерева рішень (Decision Trees), які добре працюють на невеликих наборах даних з чіткою логікою. Також застосовуються нейронні мережі (Neural Networks), які особливо ефективні при обробці тексту та складних патернів у даних. На більш просунутому рівні популярністю користуються моделі трансформерів, серед яких – BERT (Bidirectional Encoder Representations from Transformers). Вони здатні розуміти контекст у двох напрямках, що робить їх особливо корисними в діалогових системах.

Загалом, вибір методів машинного навчання залежить від поставлених цілей:

чи йдеться про просту класифікацію запитів, адаптацію до користувача, чи про повноцінне розуміння мови й генерування відповідей. Інтеграція таких методів у архітектуру чат-бота створює передумови для побудови систем, що справді імітують інтелектуальну взаємодію.

Основним етапом обробки тексту при використанні NLP є токенізація (tokenization) – поділ вхідного повідомлення на окремі слова, символи або фрази. Це дозволяє системі аналізувати структуру повідомлення й далі застосовувати правила або моделі до кожного з елементів. Після токенізації виконується морфологічний аналіз, зокрема визначення частин мови (POS-tagging), що допомагає розпізнавати функції слів у реченні. Також важливим є розпізнавання іменованих сутностей (Named Entity Recognition, NER) – виявлення імен, місць, дат тощо. Це особливо актуально в ботах, які взаємодіють із конкретними даними користувача або надають інформацію на запит.

На другому рівні застосовуються векторні подання слів, відомі як word embeddings. Технології на зразок Word2Vec або GloVe дозволяють перетворити слова на багатовимірні вектори, де відстані між ними відображають семантичну подібність. Наприклад, слова "cat" і "dog" будуть ближчими одне до одного, ніж "cat" і "car". Це дає змогу чат-боту не лише "бачити" окремі слова, а й розуміти контекст їх вживання.

Прорив у розумінні мови забезпечили великі мовні моделі (Large Language Models, LLM), зокрема GPT (Generative Pretrained Transformer), BERT та T5. Вони здатні не лише аналізувати зміст, а й генерувати логічні, зв’язні відповіді. Наприклад, BERT добре справляється з завданнями класифікації запитів, тоді як GPT підходить для генерації тексту в реальному часі. Саме такі моделі найчастіше лежать в основі сучасних чат-ботів, які вміють вести діалог, пояснювати складні поняття або адаптуватися до стилю користувача.

Інтеграція інтелектуальних систем у популярні месенджери, такі як Telegram, Viber, Messenger чи WhatsApp, відкриває широкі можливості для реалізації освітніх або сервісних чат-ботів. Проте одночасно виникають і технічні, і архітектурні виклики, пов’язані з обмеженнями самих платформ, обробкою

повідомлень у реальному часі та необхідністю адаптації моделей до обмежених обчислювальних ресурсів.

Насамперед розробник має врахувати специфіку API-інтерфейсів месенджерів. Наприклад, Telegram Bot API дозволяє надсилати та приймати повідомлення, реагувати на кнопки або inline-запити. Проте він не містить вбудованих механізмів для тривалого збереження контексту чи обробки природної мови – ці функції мають реалізовуватися на стороні сервера бота. Крім того, усі дії є асинхронними, що вимагає ретельного планування логіки обробки запитів.

Інтелектуальне ядро, побудоване на основі ШІ-моделі, зазвичай розміщується окремо від інтерфейсу месенджера. Такий підхід дозволяє змінювати або оновлювати інтелектуальну частину незалежно від самого месенджера. Наприклад, бот може передавати текст користувача на обробку в GPT- або BERT-подібну модель, отримувати сформовану відповідь і повертати її користувачу у форматі повідомлення (див. рис. 1.11).

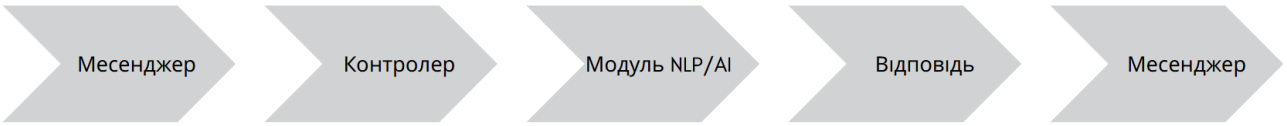


Рисунок 1.11. Принцип взаємодії інтелектуального ядра на базі ШІ та інтерфейсу месенджера

Особливої уваги потребує адаптація моделей до ресурсних обмежень. На відміну від повноцінних хмарних сервісів, бот, який запускається на локальному сервері чи обмеженому хостингу, не може підтримувати важкі трансформери без оптимізації. У таких випадках використовують або полегшені версії моделей (наприклад, DistilBERT), або зовнішні API (як-от OpenAI або Hugging Face), до яких надсилається запит і повертається результат. Це дозволяє зменшити навантаження на інфраструктуру, але потребує контролю за витратами та приватністю.

На завершення варто зазначити, що ефективна інтеграція AI-ядра у месенджер залежить не лише від вибору моделі, а й від загальної архітектури системи. Добре спроєктований чат-бот повинен бути масштабованим, адаптивним

і здатним підтримувати діалог у різних контекстах – від коротких відповідей до глибоких пояснень. Саме така гнучкість робить поєднання ШІ з месенджерами потужним інструментом у сучасних цифрових сервісах.

Попри значні досягнення у сфері штучного інтелекту, чат-боти з AI-ядром все ще мають низку обмежень, які впливають на якість діалогової взаємодії, особливо в освітньому контексті. Однією з головних проблем залишається генерація контекстно точних відповідей. Навіть сучасні великі мовні моделі, як-от GPT чи BERT, не завжди здатні коректно інтерпретувати запити користувача, якщо контекст розмови тривалий або містить неявні посилання на попередні частини діалогу.

З технічного боку, виклик становить необхідність fine-tuning'у моделей під конкретні потреби застосунку. Загальні мовні моделі мають широкий спектр знань, але для досягнення високої точності у вузькій галузі – наприклад, граматиці англійської мови чи кібергігієні – потрібна адаптація моделей за допомогою додаткового навчання на спеціалізованих корпусах. Це вимагає значних ресурсів: як обчислювальних, так і людських (зокрема експертів у предметній області).

Ще одним перспективним напрямком розвитку є гібридні моделі, які поєднують rule-based логіку (чіткі сценарії та правила) із AI-підходами. Такий підхід дозволяє зберігати контроль за критично важливими частинами діалогу, одночасно використовуючи ШІ для гнучкішої обробки природної мови. Наприклад, бот може використовувати ML-модель лише тоді, коли не розпізнає команду з попередньо визначеного переліку.

У майбутньому очікується ще більша персоналізація взаємодії між користувачем і ботом, удосконалення механізмів підтримки довготривалого контексту, зростання точності генерації відповідей і розширення мовних можливостей. Поява нових архітектур, зокрема моделей, здатних до мультимодальної обробки (обробка тексту, зображення, голосу одночасно), відкриває нові горизонти для створення інтелектуальних чат-ботів нового покоління. Інтеграція AI-ядра в месенджери вимагає адаптації моделей до ресурсних обмежень, використання полегшених версій або зовнішніх API, а також

врахування викликів генерації контекстно точних відповідей, fine-tuning'у, гібридних підходів і перспектив розвитку персоналізації та мультимодальних моделей.

1.2.3 Дослідження можливостей використання ШІ в цифровій освіті

Як вже зазначалося раніше, штучний інтелект (ШІ) відіграє дедалі важливішу роль у трансформації цифрових освітніх середовищ. Попередні розділи торкались загальних понять, архітектур та технічних аспектів реалізації інтелектуальних систем, у тому числі чат-ботів. Тому тут будуть представлені частково вищезгадані концепти думок, але вже у прикладному вимірі – з акцентом саме на освітні потреби, можливості впровадження ШІ в навчальний процес та існуючі виклики.

Одним із найперспективніших напрямів використання ШІ в освіті є персоналізація навчання. Завдяки обробці великих обсягів даних системи можуть адаптувати навчальний контент до потреб конкретного учня: підбирати завдання відповідної складності, змінювати темп подачі матеріалу, пропонувати додаткові пояснення або навпаки – переходити до нових тем. Це дозволяє значно покращити індивідуальний освітній досвід і підвищити рівень залучення студентів.

Водночас, ШІ активно використовується і в автоматизації рутинних освітніх процесів. Наприклад, платформи на основі ШІ вже зараз здійснюють перевірку тестових завдань, генерують навчальні рекомендації або навіть ведуть діалог із користувачем у форматі цифрового репетитора. Завдяки цьому викладачі мають можливість зосередитися на аналітичній, методичній чи менторській роботі, не витрачаючи час на технічні операції.

Однак впровадження ШІ в освітнє середовище не є виключно позитивним процесом. Воно супроводжується низкою ризиків, серед яких – зниження самостійності студентів, можливе копіювання знань без їх глибокого осмислення, упередженість алгоритмів або загроза порушення конфіденційності даних. Також спостерігається нерівномірність доступу до сучасних технологій у різних соціальних групах, що лише підсилює існуючі освітні розриви.

Перспективи розвитку ШІ в освіті залишаються оптимістичними.

Розширення функціональності освітніх платформ, удосконалення алгоритмів адаптивного навчання, інтеграція етичних принципів у розробку систем – усе це створює умови для подальшого розвитку якісної, інклюзивної та ефективної цифрової освіти. Очевидно, що майбутні рішення потребуватимуть як технологічного прогресу, так і глибокого осмислення цінностей, на яких будується освітній процес [8].

1.3 Розробка чат-боту для перевірки знань з англійської мови

1.3.1 Організація середовища

Для забезпечення стабільного та передбачуваного середовища виконання телеграм-бота було використано контейнеризацію з Docker. Це дозволяє швидко розгортати застосунок на будь-якій платформі без необхідності локального встановлення залежностей.

Файл Dockerfile відповідає за створення образу на базі легкого образу python:3.11-slim та містить лише найнеобхідніші залежності:

```
FROM python:3.11-slim
WORKDIR /app
RUN apt-get update && apt-get install -y --no-install-recommends \
    gcc \
    libssl-dev \
    ca-certificates \
    && rm -rf /var/lib/apt/lists/*
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt
RUN pip install -U python-dotenv
COPY ./src /app
CMD ["python", "main.py"]
```

Особливостями цього Dockerfile являються використання --no-install-recommends та очищення кешу apt зменшує розмір образу. Також було встановлено python-dotenv для коректної обробки змінних середовища всередині застосунку. Код програми копіюється лише з каталогу src/, щоб уникнути зайвих файлів у контейнері.

Для зручності запуску контейнера було використано docker-compose, який дозволяє автоматично монтувати код та змінні середовища:

```
version: '3.8'
services:
  app:
    build: .
    volumes:
```

```
      - ./src:/app
      - ./.env:/app/.env
    env_file:
      - path: ./.env
        required: true
```

volumes: дві точки монтування – каталог src/, що дозволяє змінювати код без перестворення образу, та .env, щоб передати налаштування в контейнер. env_file: вказує, що для сервісу необхідно використовувати .env файл, причому він є обов'язковим (required: true). Такий підхід також дозволяє уникнути "зашивання" чутливої інформації у сам образ.

Файл із залежностями виглядає так:

```
dotenv==0.9.9
python-dotenv==1.1.0
pymongo==4.12.1
requests==2.31.0
urllib3==1.26.15
```

Де бібліотека dotenv потрібна для роботи з .env файлами, pymongo для роботи з базами даних mongoDB, requests та urllib3 для мережевих запитів.

У файлі .env зберігаються конфіденційні дані, такі як токен для чат-боту та посилання-підключення до бази даних на MongoDB Atlas.

```
BOT_TOKEN=your-telegram-token
MONGO_URI=mongodb://localhost:27017
```

1.3.2 Структура проєкту

Розробка програмного забезпечення для Telegram-бота з можливістю генерації тестів англійською мовою передбачає чітке структурування коду та організацію окремих логічних модулів. У ході реалізації практичної частини проекту було створено структуру, яка відповідає сучасним підходам до розробки серверних застосунків мовою Python. Архітектура застосунку базується на принципах модульності, повторного використання коду та чіткого розділення відповідальностей.

У кореневій директорії проекту розміщено усі конфігураційні файли та скрипти, необхідні для розгортання застосунку як у середовищі розробника, так і в продакшн-середовищі. Основними з них є Dockerfile і compose.yaml, які відповідають за контейнеризацію застосунку. Завдяки Docker застосунок може бути ізольовано від системного оточення, що спрощує як розгортання, так і супровід. Dockerfile побудований на основі полегшеного образу python:3.11-slim,

що дозволяє суттєво зменшити розмір фінального контейнера. У ньому виконуються лише необхідні інструкції: встановлення системних бібліотек, завантаження залежностей із requirements.txt, копіювання коду та запуск програми. Файл compose.yaml конфігурує сервіс, монтує локальний каталог src до внутрішнього каталогу /app в контейнері, а також забезпечує доступ до змінних середовища з .env файлу, що зручно для зберігання ключів API, токена бота або конфігурації бази даних. Це дозволяє легко змінювати параметри роботи без необхідності модифікації коду.

Код застосунку зберігається у папці src, що є основним джерелом прикладної логіки. Точкою входу є файл main.py, який виконує лише одну функцію – виклик методу ініціалізації Telegram-бота, що винесений у окремий модуль bot.py. Такий підхід дозволяє спростити запуск програми та логічно відокремити інфраструктурний код від прикладної логіки. У модулі bot.py реалізовано повноцінну процедуру налаштування й запуску Telegram-бота: завантаження змінних середовища з .env файлу, отримання API-ключа, створення екземплярів сервісів, контролерів та представлення, а також запуск безперервного циклу опрацювання подій. Внутрішній цикл run_bot_loop() організований у вигляді нескінченного опитування серверу Telegram, із розділенням логіки на обробку команд та callback-запитів. Це дозволяє реалізувати реактивну поведінку бота, а також централізовано обробляти винятки без втрати стабільності сервісу. У цей спосіб, структура застосунку забезпечує чітке розділення логіки запуску, взаємодії з API Telegram, обробки повідомлень і логіки бізнес-рівня, що відповідає принципам чистої архітектури й сприяє легшій підтримці та масштабуванню проєкту..

У підкаталозі controllers розміщено модулі, відповідальні за обробку вхідних повідомлень від користувачів Telegram. Тут реалізовано класи контролерів, кожен з яких відповідає за певну функціональність (див. рис. 1.12). Для забезпечення масштабованості та зручності тестування кожен контролер спроектований з урахуванням принципів інверсії залежностей. Це дозволяє легко додавати нові функції або змінювати існуючі без значних змін у загальній архітектурі проекту.

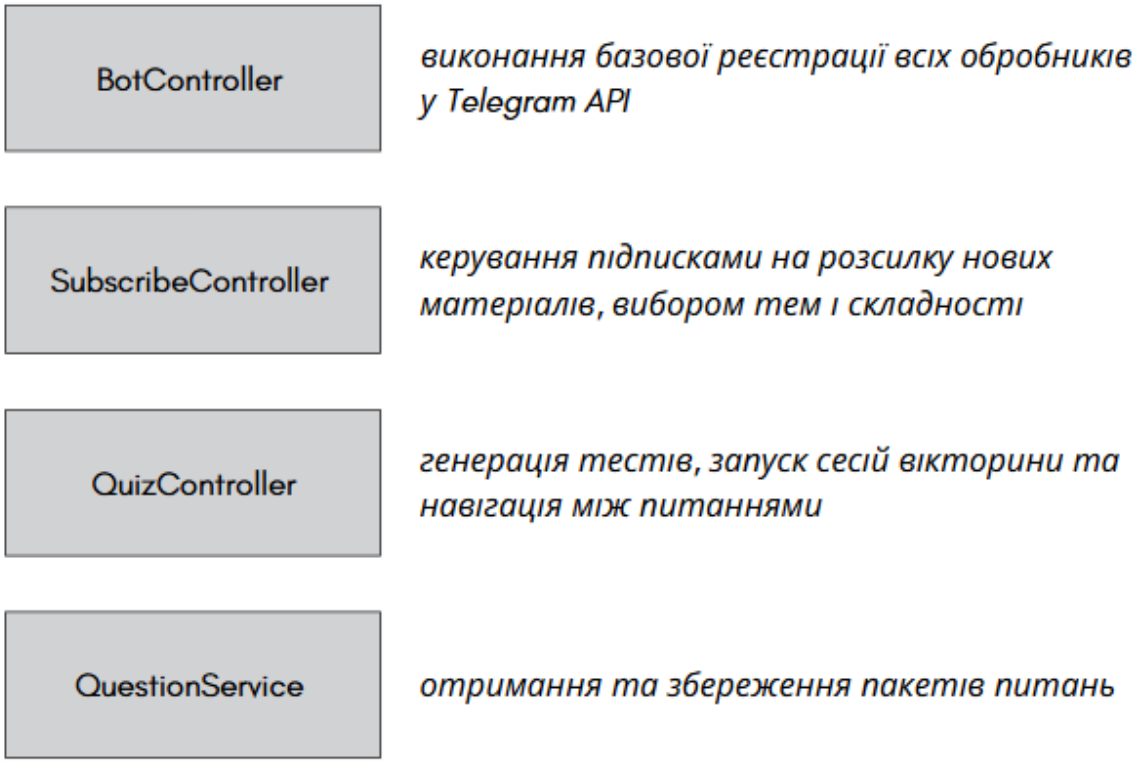


Рисунок 1.12. Класи контролерів та їх функціонал

Модулі, що взаємодіють із базою даних, згруповано у папці models. Тут описано логічну структуру колекцій MongoDB, які використовуються для зберігання інформації про користувачів, створені запитання, теми тестів, а також рівні складності. Наприклад, у UserModel описано поля, які зберігаються для кожного користувача: Telegram ID, ім’я, рівень складності, історію проходження тестів тощо. QuestionModel містить структуру згенерованих запитань, включаючи формулювання, варіанти, правильну відповідь та її пояснення. Усі моделі працюють через загальний модуль db.py, у якому налаштовано підключення до MongoDB з урахуванням конфігурації із .env файлу.

Логіка взаємодії між контролерами, моделями та зовнішніми сервісами реалізована у папці service. Саме тут містяться ключові функціональні компоненти, зокрема AIService, який відповідає за інтеграцію зі штучним інтелектом. Він приймає запити на генерацію запитань англійською мовою на задану тему, а також формує варіанти відповідей на основі результату. Інші сервіси – як-от UserService чи QuestionService – інкапсулюють логіку роботи з відповідними моделями, щоб спростити виклики з боку контролерів.

Для відображення реакції бота на запити користувача використовується папка views. У модулі TelegramView.py реалізовано всі повідомлення, які Telegram-бот надсилає користувачу: початкове вітання, інструкції, результати тесту тощо. Такий підхід дозволяє централізовано змінювати вигляд повідомлень, не вдаючись до редагування логіки контролерів.

Окрему роль відіграє папка utils, де зосереджено допоміжну функціональність, зокрема систему логування, яка спрощує відлагодження та ведення журналів подій. Модуль logger.py створює налаштований логер, який виводить повідомлення у консоль (інформація, помилка, попередження тощо).

1.3.3 Робота з Telegram API

У цьому розділі розглянуто, як реалізовано інтеграцію з Telegram API для створення чат-бота, який взаємодіє з користувачами, обробляє повідомлення та відповідає в реальному часі, а також розглянуто архітектуру, ключові модулі та алгоритми обробки подій.

Telegram дозволяє інтеграцію через Bot API, що працює на основі HTTP-запитів. У поточній реалізації застосовано метод long polling через ендпоінт getUpdates, який дозволяє періодично перевіряти наявність нових подій (повідомлень, callback-натискань тощо) (див. рис. 1.13).

Цей метод є простим у реалізації та не вимагає підтримки складних протоколів зв’язку. Обробка подій відбувається у циклі, що забезпечує безперервний моніторинг оновлень від серверів Telegram. Кожне нове повідомлення або дія користувача передається на обробку відповідному контролеру. Контролери відповідають за логіку взаємодії з користувачем, наприклад, генерацію тестів або відповіді на запити. Для підвищення надійності застосовано централізований механізм обробки винятків, що дозволяє уникнути аварійного завершення роботи бота. Всі ключові параметри, такі як токен бота та налаштування бази даних, зберігаються у змінних середовища, що підвищує безпеку. Архітектура передбачає розділення логіки взаємодії, бізнес-правил та доступу до даних для кращої підтримуваності коду. Реалізовано підтримку callback-запитів, що дозволяє створювати інтерактивні кнопки в інтерфейсі бота.

Також передбачено асинхронну обробку запитів, що підвищує продуктивність і швидкодію. Загалом обрана архітектура сприяє масштабованості та гнучкості проекту.

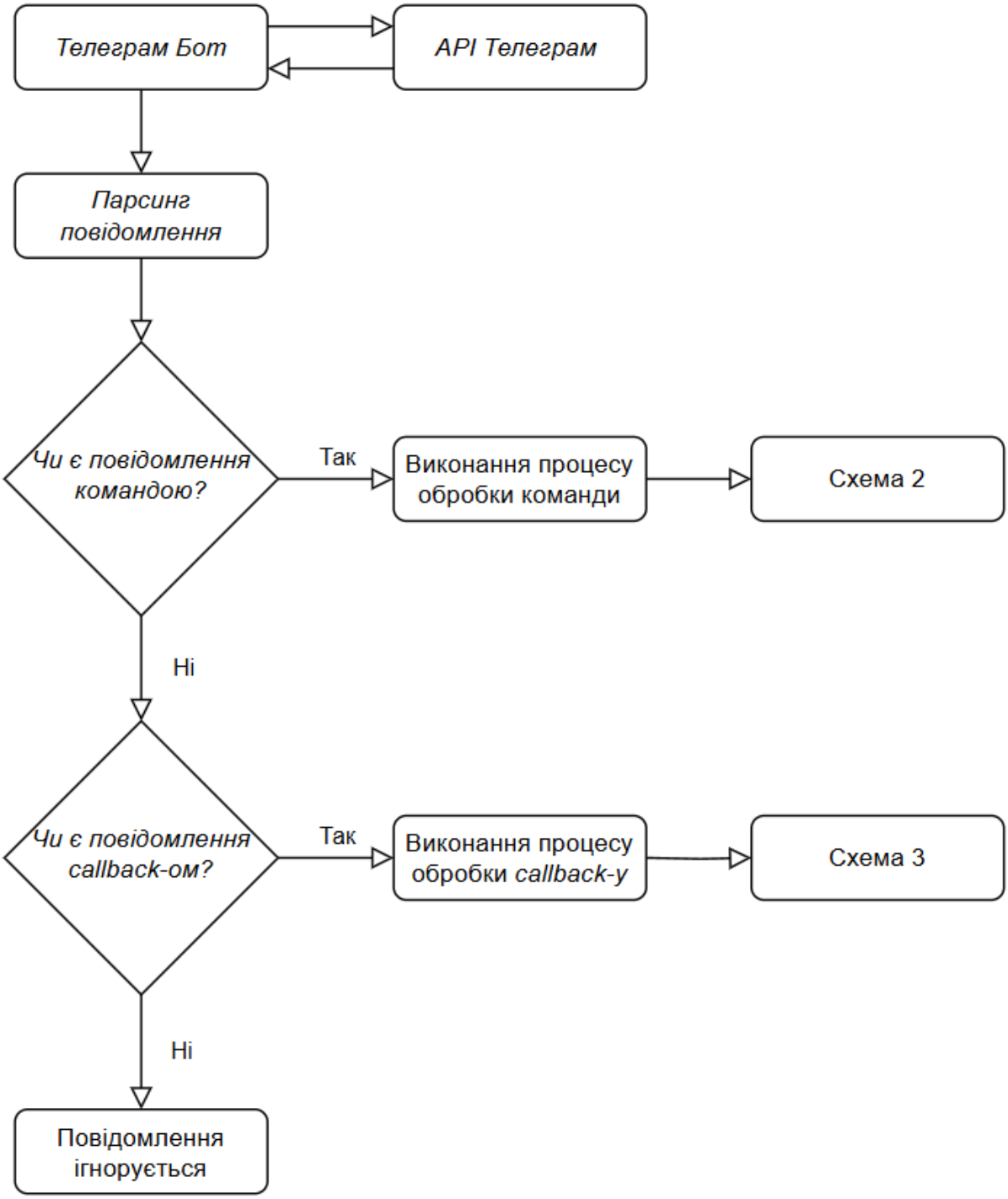


Рисунок 1.13. Загальна структура Телеграм боту

Бот запускається у файлі bot.py, де ініціалізуються залежності, включаючи сервіси, контролери та головний цикл run_bot_loop. У цьому циклі відбувається безперервне опитування Telegram API на наявність нових подій.

Після отримання повідомлення від користувача, бот викликає функцію handle_message, яка відповідає за розпізнавання та реакцію на текст команди. Основними командами є /start, /subscribe, /unsubscribe, /options і /test. Якщо користувач надсилає команду /start, бот надсилає вітальне повідомлення, у якому коротко пояснює, як працює вікторина. У випадку команди /subscribe бот зберігає інформацію про підписку користувача для подальших сповіщень. Команда /unsubscribe скасовує підписку, а /options дозволяє користувачеві обрати тему і складність вікторини через інтерактивні кнопки. Нарешті, команда /test ініціює саму вікторину, після чого користувач починає отримувати питання із варіантами відповідей (див. рис. 1.14).

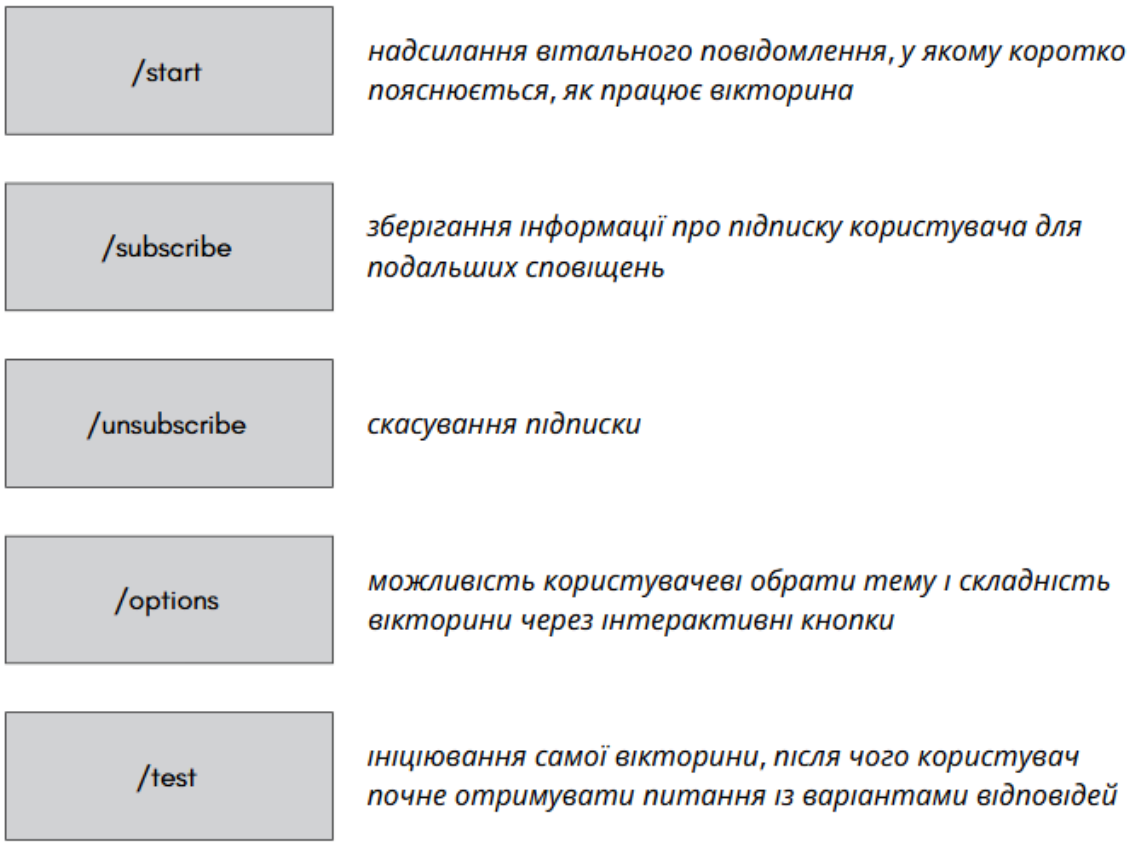


Рисунок 1.14. Основні команди та їх функціонал

Бот використовує простий механізм розгалуження: він порівнює отриманий текст із відомими командами і викликає відповідні дії у сервісах або формує відповіді через TelegramView. Відповіді можуть містити як прості текстові повідомлення, так і інтерактивні кнопки (inline keyboard), що дозволяють реалізувати опитування без необхідності введення тексту (див. рис. 1.15).

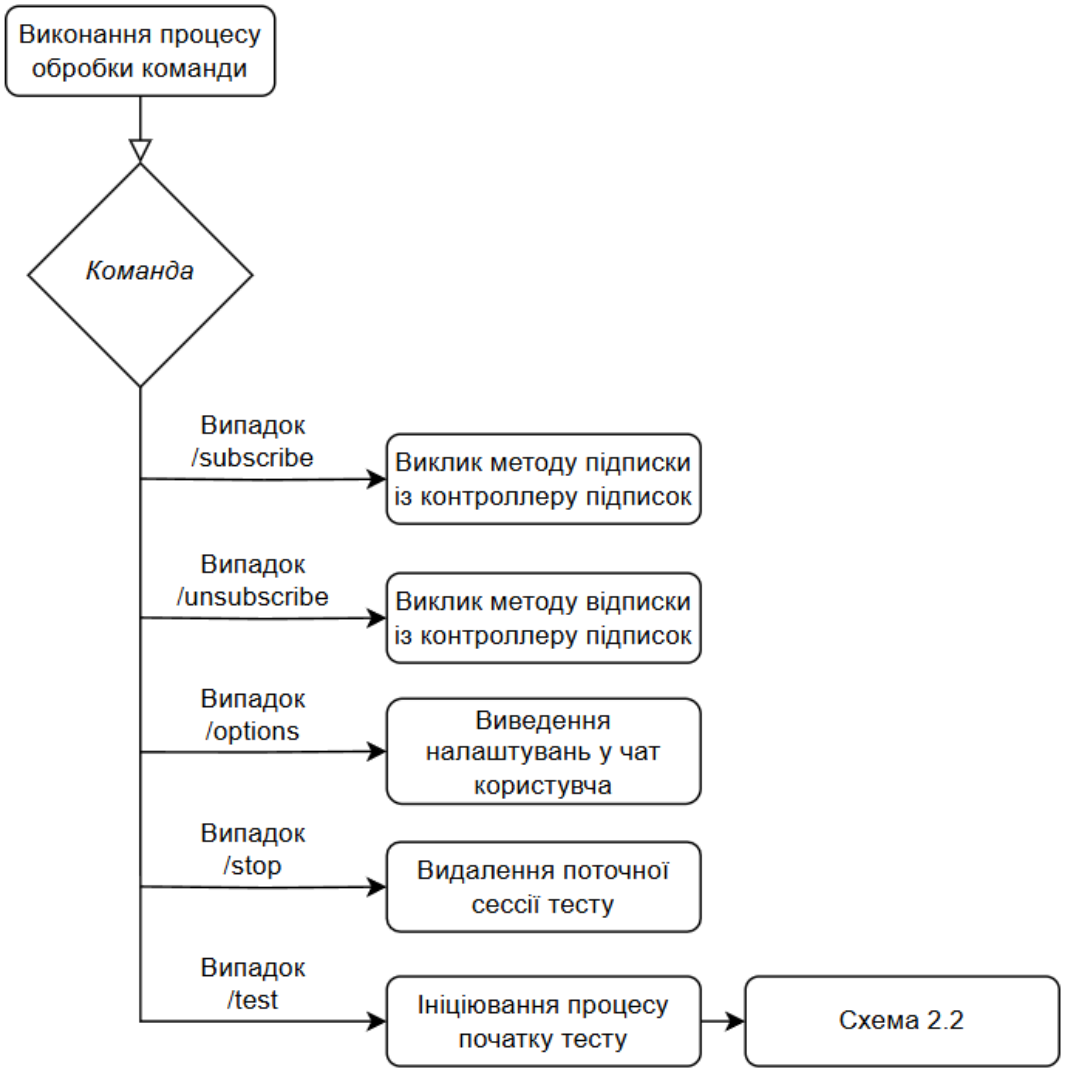


Рисунок 1.15. Схема виконання процесу обробки команд

Крім текстових команд, бот також реагує на callback-події, які виникають після натискання кнопок у Telegram-інтерфейсі. Ці події обробляються функцією handle_callback. Callback-повідомлення надходять у вигляді структурованих рядків, що містять інформацію про дію і її параметри, наприклад theme_pick:3, difficult_pick:2 або T:1, де перша частина вказує на тип дії, а друга – на її значення.

Функція обробки callback-ів розпізнає тип взаємодії: якщо це вибір теми (theme_pick), бот оновлює відповідний параметр у стані користувача; якщо вибрано складність (difficult_pick), аналогічно оновлюється рівень складності. У випадку, коли callback містить варіант відповіді на питання (наприклад, T:2 для true або F:1 для false), бот перевіряє правильність відповіді, оновлює інтерфейс кнопок і надсилає наступне питання або повідомляє про завершення вікторини, якщо питань більше не залишилося (див. рис. 1.16 та 1.17).

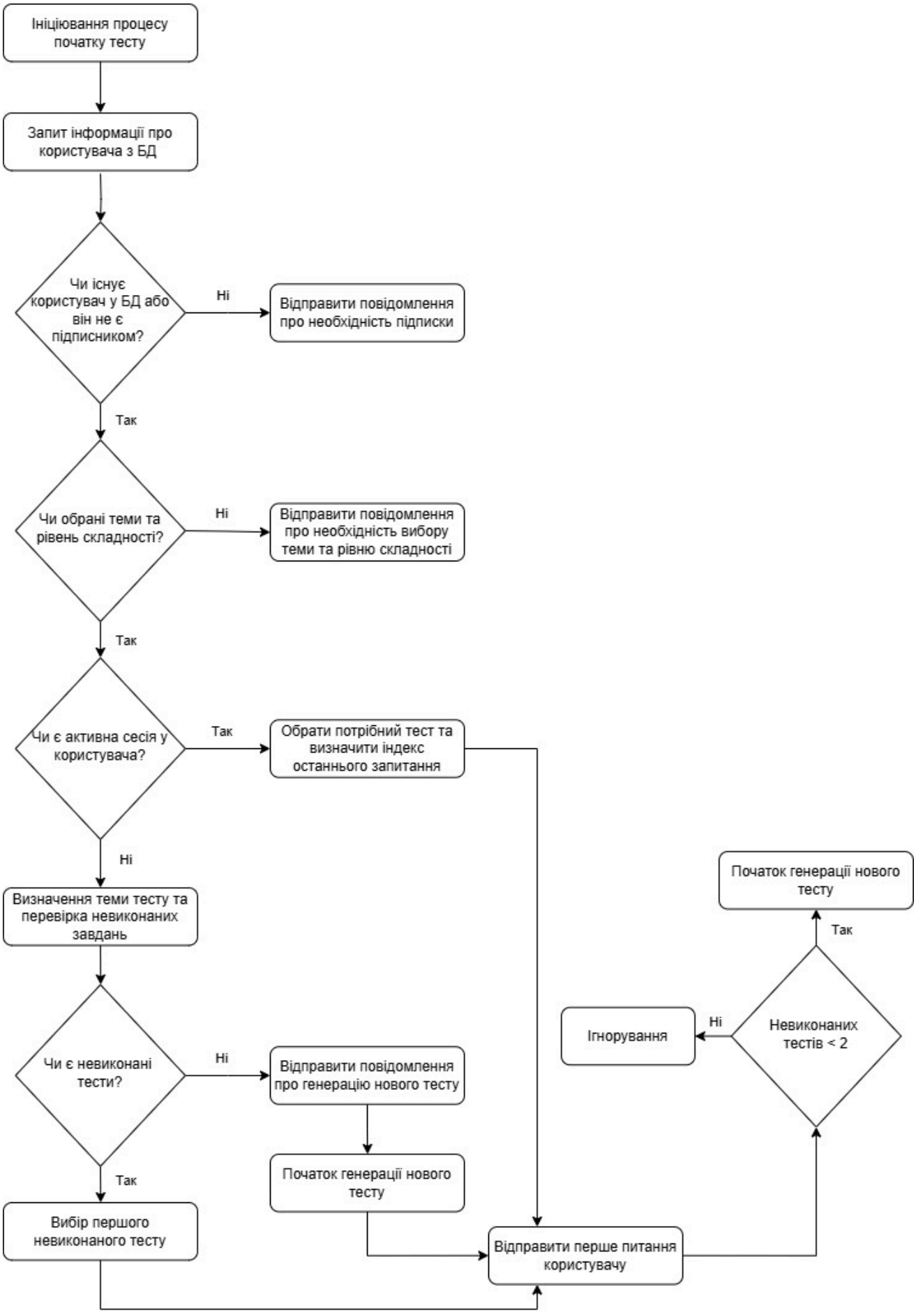


Рисунок 1.16. Схема ініціювання процесу початку тесту

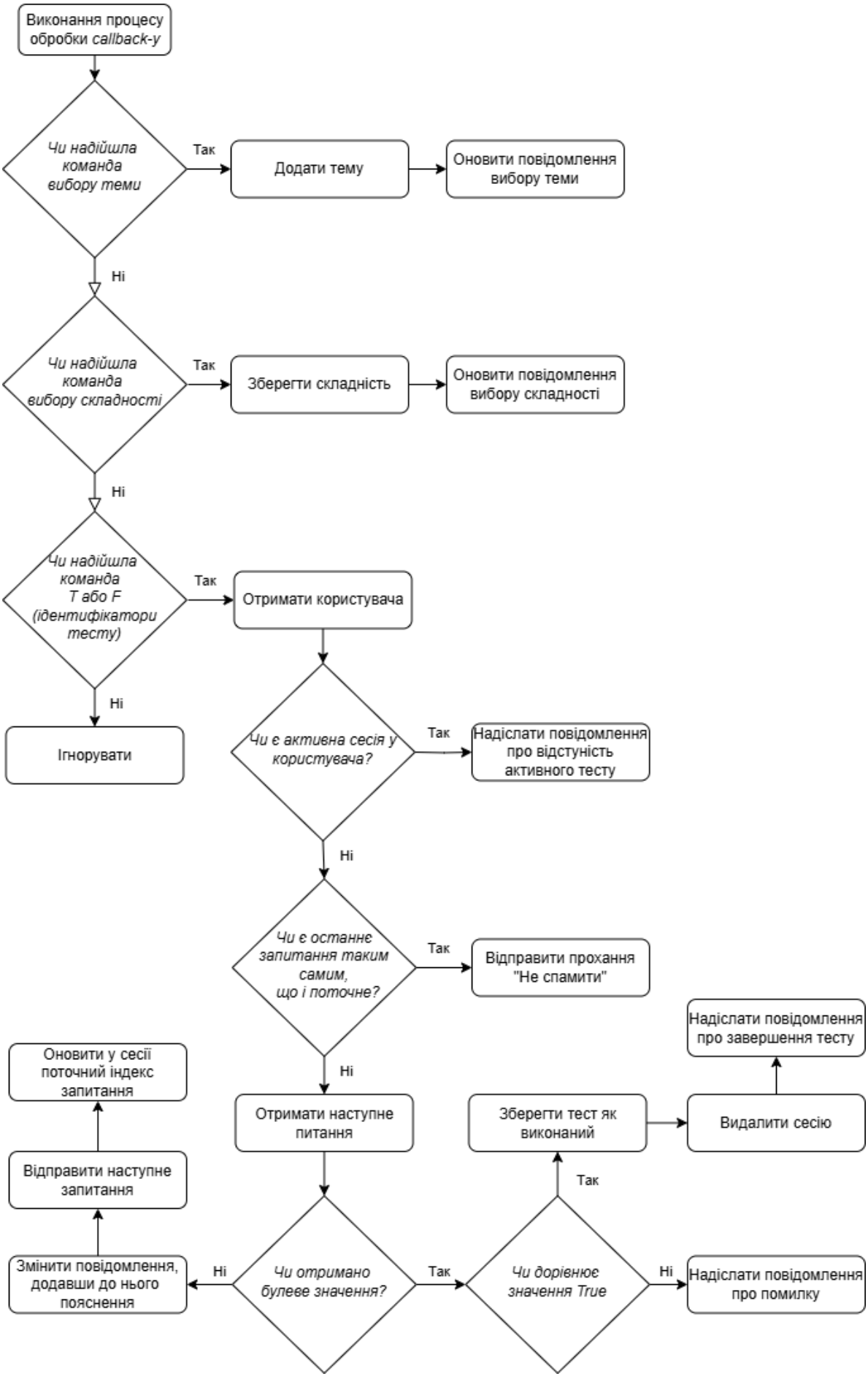


Рисунок 1.17. Схема виконання процесу обробки callback-у

1.3.4 Структура та логіка контролерів

У Telegram-боті логіка обробки ключових сценаріїв розділена між окремими контролерами. Це забезпечує чітке розділення відповідальностей. Цей розділ присвячено двум основним контролерам: QuizController – для генерації вікторин і керування сесією користувача, та SubscribeController – для підписки користувачів, вибору тем і складності.

QuizController відповідає за генерацію тестів, запуск сесій вікторини та навігацію між питаннями. Він активно взаємодіє з AIService (генерація питань) та QuestionService (отримання та збереження пакетів питань) і веде журнал подій через logger.

Генерація нового тесту:

```
def generate_quiz(self, theme, difficult=»»):
    if theme:
        history = QuestionService.get_questions_by_theme_and_difficult(theme[«id»],
difficult)
    if difficult:
        difficult = QuestionService.get_difficult_name(difficult)

    logger([history, theme[«title»], difficult])

    data     =     self.ai_service.get_new_ai_question(history,     theme[«title»],
difficult[«title»])
    if data:
        data = QuestionService().createPack({
            «theme»: ObjectId(theme[«id»]),
            «questions»: data,
            «difficult»: difficult[«_id»]
        }).to_dict()
        logger(«End of generating test»)
        return data
    else:
        return «Failed to generate quiz data.»
```

Цей метод генерує новий набір питань з використанням штучного інтелекту. Він також фільтрує вже наявні питання за темою та складністю, щоб уникнути повторів. Якщо генерація успішна – створюється новий запис у базі.

Початок сесії тесту:

```
def start_quiz(self, user, ready_pack):
    try:
        selected_pack_id = str(ready_pack.get(«_id»)) if ready_pack else None
        selected_question_index    =    user[«active_session»][«current_index»]    if
user[«active_session»] else 0
    except Exception as ex:
        import traceback
        traceback.print_exc()
        logger(f»🔥 Exception occurred: {repr(ex)}»)
```

```
            return «Something went wrong when starting the quiz.»

        Questions = ready_pack
        if not questions:
            return «⚠ Could not load the quiz. Try again later.»

        Return {
            «current»: selected_question_index,
            «selected_pack_id»: selected_pack_id,
            «options»: {
                «text»: questions[«questions»][selected_question_index][«text»],
                «qeustions»: questions[«questions»][selected_question_index][«options»]
            }
        }
```

Цей метод ініціює проходження тесту, визначаючи поточне питання та повертаючи його текст і варіанти відповідей. У разі помилок користувач отримує повідомлення про збій.

Перехід до наступного питання:

```
def next_quiz(self, user):
    pack = user[«active_session»].get(«question_pack_id»)
    index = user[«active_session»].get(«current_index»)
    indexNext = index + 1
    questions = QuestionService.getPack(pack)[«questions»] if pack else []

    reason = questions[index].get(«reason») or None

    if len(questions) == indexNext:
        return True  # фінал
    if not questions:
        return False  # помилка

    return {
        «current»: indexNext,
        «selected_pack_id»: pack,
        «options»: {
            «text»: questions[indexNext][«text»],
            «qeustions»: questions[indexNext][«options»]
        },
        «reason»: reason
    }
```

Метод next_quiz перевіряє наявність наступного питання і формує структуру наступного кроку. Також додає пояснення (reason) до попереднього питання, якщо воно передбачене.

SubscribeController забезпечує логіку підписки користувачів, збереження обраних тем та рівня складності. Він використовує UserService для взаємодії з моделлю користувача та TelegramView для формування повідомлень з кнопками.

Підписка нового користувача:

```
def subscribe_user(self, chatID, username=None):
    try:
        user = UserService.get_one(chatID)
```

```
            if not user or not user.get(«picked_themes»):
                UserService.subscribe_user(chatID, username)
                self.pick_themes_message(chatID)
                self.pick_difficult(chatID)
            else:
                UserService.subscribe_user(chatID, username)
                self.telegram.send_message(chatID, self.view.subscribe_message())
        except Exception as e:
            logger(f»{e}»)
```

Користувачам, які ще не обрали теми, надсилаються повідомлення з варіантами вибору. Іншим – звичайне підтвердження підписки.

Формування списку тем:

```
def pick_themes_message(self, chatID, messageID=None):
    userPickedThemes = UserService.get_subscribed_themes(chatID)
    themes = ThemeModel.find_all()
    options = []
    for theme in themes:
        name = theme[«theme_name»]
        id = str(theme[«theme_id»])
        display_name = f»{name} ✅» if id in userPickedThemes else name
        options.append({ ‘text’: display_name, ‘callback_data’: f»theme_pick:{id}»
})

    if messageID:
        self.telegram.send_message(chatID,         self.view.pick_themes_message(),
options, messageID)
    else:
        self.telegram.send_message(chatID,         self.view.pick_themes_message(),
options)
```

Метод формує динамічний список тем, де вже вибрані теми позначаються галочкою. Дані кнопки генеруються як callback_data.

Вибір складності:

```
def pick_difficult(self, chatID, messageID=None):
    user = UserService.get_one(chatID)
    userPickedDifficult = user.get(«difficult») if user else []
    difficulties = DifficultModel.find_all()
    options = []
    for diff in difficulties:
        name = diff[«diff_name»]
        id = str(diff[«diff_id»])
        display_name = f»{name} ✅» if id in userPickedDifficult else name
        options.append({        ‘text’:        display_name,        ‘callback_data’:
f»difficult_pick:{id}» })

    if messageID:
        self.telegram.send_message(chatID,   self.view.pick_difficult(),   options,
messageID)
    else:
        self.telegram.send_message(chatID, self.view.pick_difficult(), options)
```

Аналогічно до тем, цей метод формує список варіантів складності, з можливістю позначення вибраного варіанту. Користувач обирає варіант за

допомогою кнопок.

Оновлення теми або складності:

```
def add_theme(self, chatID, themeID):
    return UserService.update_theme(chatID, themeID)

def set_difficult(self, chatID, difficult):
    return UserService.update_difficult(chatID, difficult)
```

Ці методи обробляють вибір користувача у callback-ах і зберігають оновлену інформацію в базі.

Завдяки структурі контролерів логіка розподілена за функціональними ознаками: QuizController відповідає за тестування, а SubscribeController – за персоналізацію досвіду користувача. Кожен з них делегує завдання сервісам, що спрощує супровід і розширення проекту.

1.3.5 Інтеграція з ШІ

Сервіс AIService відповідає за генерацію нових запитань за допомогою зовнішньої AI-моделі (наприклад, OpenAI). Він формує промпти відповідно до заданих параметрів, надсилає їх до API, обробляє відповідь і перетворює її у внутрішній формат, зручний для подальшої роботи в системі.

Підключення до API.

При ініціалізації сервіс завантажує змінні середовища:

```
self.url = os.getenv("AI_API_URL")
self.key = os.getenv("AI_API_KEY")
self.model = os.getenv("AI_API_MODEL")
```

Це дає можливість динамічно змінювати модель, ключ або URL без потреби змінювати код.

Генерація тестових запитань.

Метод get_new_ai_question(history, theme, difficult) викликає функцію build_quiz_prompt, що формує текст запиту до AI. Він включає:

Назву теми (наприклад, “Internet”)

Рівень складності (наприклад, “Intermediate”)

Історію раніше згенерованих запитань, які не можна повторювати

Формат відповіді очікується у вигляді JSON-масиву з об’єктами:

```
[
  {
    "question": "Текст питання",
```

```
    "answers": ["варіант 1", "варіант 2", ...],
    "correctAnswer": "правильний варіант",
    "reasonOfAnswer": "пояснення вибору"
  }
]
```

Метод parse_ai_questions обробляє отриману відповідь, шукає в ній JSON-блок за допомогою регулярного виразу re.search, перевіряє його і перетворює кожне запитання на об’єкт внутрішнього формату:

text: текст запитання

options: список варіантів відповіді у вигляді кнопок Telegram із відповідною callback_data, де “T” означає правильну відповідь

reason: пояснення до відповіді

Приклад однієї сформованої опції:

```
{
  "text": "variant A",
  "callback_data": "F:0"
}
```

Надсилання запиту.

Запит надсилається методом get_questions, який використовує бібліотеку requests для POST-запиту до AI API. Тіло запиту містить модель, ключ, а також масив messages, де prompt вказується як вміст повідомлення користувача. У разі помилки в запиті чи декодуванні JSON вона логуються через logger.

Цей сервіс є ключовим компонентом інтелектуальної генерації контенту, який дозволяє адаптувати тематику та складність вікторини для кожного користувача динамічно та уникає повторень завдяки механізму врахування історії запитань.

1.3.6 Робота з MongoDB. Опис моделей

Підключення до MongoDB реалізовано за допомогою бібліотеки pymongo. URL для підключення зчитується з .env файлу за допомогою модуля dotenv, що дозволяє зберігати конфіденційні дані окремо від коду.

```
from pymongo import MongoClient
load_dotenv(find_dotenv())

url = os.getenv("MONGODB_URI")
client = MongoClient(url, tls=True)
db = client["english_teacher"]
```

Весь проект використовує одну базу даних з назвою english_teacher. У ній

розміщено кілька колекцій: users, themes, difficults, question_packs.

Кожна сутність (наприклад, користувач або тема) реалізована у вигляді окремого класу Python, який містить методи для створення, збереження, пошуку та оновлення даних у відповідній колекції MongoDB.

1. Модель складності (DifficultModel).

Модель використовується для зберігання рівнів складності, які користувач може обрати для проходження тесту. Наприклад: "Beginner", "Intermediate", "Advanced".

Структура документа:

1) title – назва рівня складності (типу str);
2) _id – унікальний ідентифікатор, генерується автоматично MongoDB.

Основні методи:

1) save() – зберігає новий рівень складності у колекцію difficults;
2) find_all() – повертає список усіх складностей у форматі списку словників;
3) find(id) – шукає один запис за _id.

Ця модель допомагає зберігати обмежений набір допустимих складностей, що далі використовуються у моделях користувача та тестів.

2. Модель тем (ThemeModel).

Відповідає за зберігання тематик, які користувач може обрати для тестування. Наприклад: "Internet", "Business", "Travel".

Структура документа:

1) title – назва теми (типу str);
2) description – короткий опис теми (типу str);
3) _id – унікальний ідентифікатор.

Основні методи:

1) save() – зберігає нову тему в колекцію themes;
2) find_all() – повертає всі наявні теми;
3) find_by_name(name) – шукає тему за назвою;
4) pick_least_used_theme(sorted_themes) – вибирає найменш використану тему на основі статистики використання (наприклад, для генерації нових питань

без повторів).

3. Модель тестових запитань (QuestionModel).

Містить згенеровані запитання від штучного інтелекту, згруповані у "пакет" для однієї теми з певною складністю.

Структура документа:

1) theme – посилання на тему (ObjectId);
2) questions – список питань, кожне з яких може містити текст запитання, варіанти відповідей, правильну відповідь тощо;
3) difficult – рівень складності (наприклад: "Beginner");
4) created_at – дата і час створення;
5) _id – унікальний ідентифікатор MongoDB.

Основні методи:

1) save() – зберігає тест у колекцію question_packs;
2) from_dict(data) – створює об'єкт моделі з документа MongoDB;
3) findOne(query) – знаходить одне питання за довільним запитом;
4) findAllBy(query) – повертає унікальні значення певного поля;
5) findAll(query) – повертає список усіх документів, які задовольняють умову;
6) findlast(query) – повертає найновішу дату створення тесту.

4. Модель користувача (UserModel).

Центральна модель, яка містить всю інформацію про користувача Telegram, включаючи його прогрес, обрані теми та поточну сесію тестування.

Структура документа:

1) chat_id – унікальний ідентифікатор користувача Telegram (int або str);
2) name – ім’я користувача;
3) difficult – обраний рівень складності;
4) subscribed – логічне поле, чи підписаний користувач (для розсилок тощо);
5) picked_themes – список ID тем, які користувач обрав;
6) completed_quizzes – список завершених тестів;
7) active_session – словник з інформацією про незавершений тест (наприклад, список питань, відповідей, таймер тощо);

8) _id – унікальний MongoDB ID.

Основні методи:

1) save() – зберігає нового користувача;
2) from_dict(data) – конвертує словник MongoDB у модель;
3) find_one(query) – знаходить одного користувача за умовою (наприклад, за chat_id);
4) update_user(chat_id, update_fields) – оновлює дані користувача. Підтримуються стандартні оператори MongoDB;
5) $set – оновлення значення;
6) $push – додавання до масиву;
7) $pull – видалення з масиву;
8) $addToSet – додавання без дублікатів.

Приклад використання:

UserModel.update_user(123456, {"$push": {"completed_quizzes": new_quiz}})

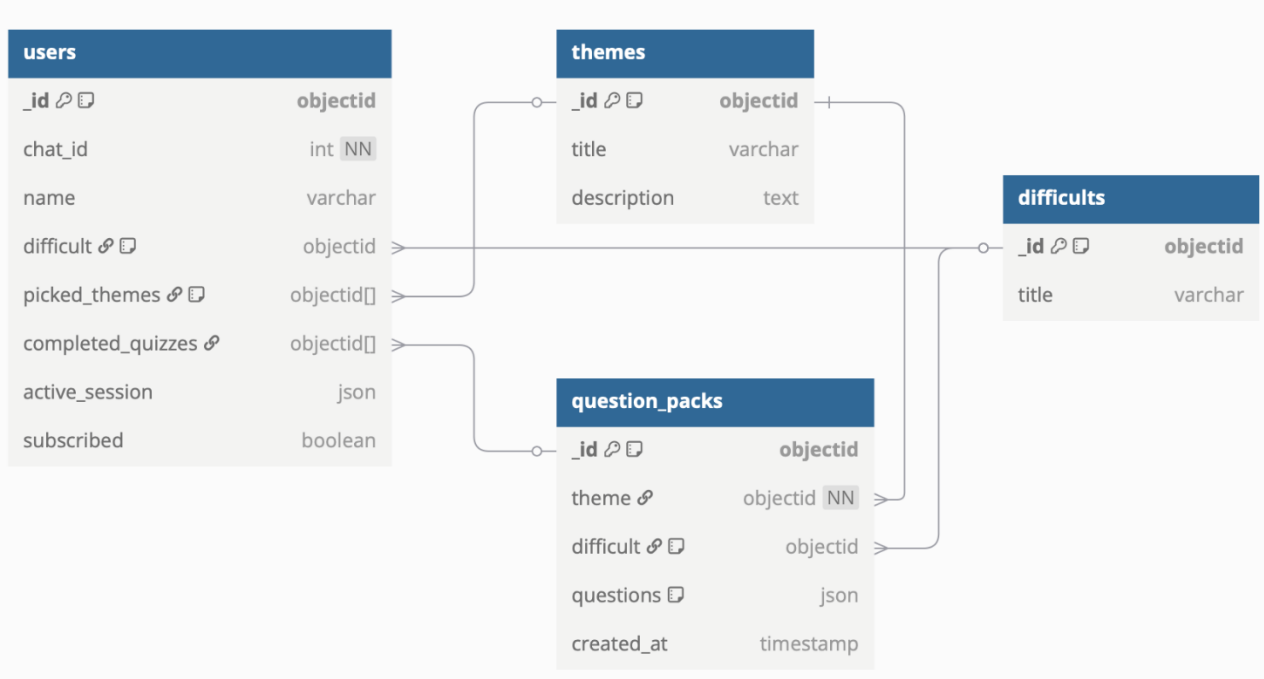


Рисунок 1.18. Діаграма відношення таблиць

1.3.7 Розробка сервісів

У межах серверної частини Telegram-бота реалізовано кілька сервісів, кожен із яких відповідає за окрему сферу логіки застосунку: керування користувачами (UserService), робота з Telegram API (TelegramService), а також взаємодія з базою

даних питань (QuestionService). Ці сервіси ізольовані один від одного, використовують відповідні моделі, і можуть бути повторно використані в інших частинах застосунку.

1. Сервіс користувача (UserService).

Сервіс обробляє логіку створення, оновлення та зчитування даних користувача, зокрема його сесію, складність, підписки на теми та виконані пакети.

Методи:

1) subscribe_user(chat_id, username): реєструє нового користувача в системі або оновлює існуючого, встановлюючи поле subscribed = True. Якщо користувач існує, реєстрація не повторюється;
2) unsubscribe_user(chat_id): знімає підписку користувача, встановлюючи поле subscribed = False;
3) update_session(chat_id, pack_id, message_id, index=0, last_question=""): оновлює сесію користувача: зберігає ID пакета питань, індекс поточного питання, текст останнього питання, список неправильних відповідей та ID повідомлення Telegram;
4) delete_session(chat_id): видаляє активну сесію користувача, очищаючи поле active_session;
5) update_theme(chat_id, theme_id): додає або видаляє тему зі списку обраних тем користувача. Якщо тема вже обрана – видаляє її, інакше додає;
6) update_difficult(chat_id, difficult): встановлює рівень складності, який вибрав користувач;
7) get_subscribed_themes(chat_id): повертає список тем, на які підписаний користувач. Якщо теми не знайдено – повертає порожній список;
8) push_completed_task(chat_id, task_id): додає ID виконаного пакета питань до списку completed_quizzes;
9) get_one(chat_id): повертає повну інформацію про користувача у вигляді словника (dict).

2. Сервіс Telegram (TelegramService).

Цей сервіс інкапсулює взаємодію з Telegram Bot API: обробку оновлень,

надсилання та редагування повідомлень, а також їхнє видалення. Сервіс дозволяє централізовано керувати всією логікою, пов'язаною з Telegram.

Методи:

1) get_updates(offset=None): виконує запит до Telegram API getUpdates. Параметр offset дозволяє обробляти тільки нові оновлення. Повертає JSON-відповідь з переліком нових подій;
2) send_message(chat_id, text, options=None, message_id=None): надсилає нове або редагує існуюче повідомлення користувачу. Якщо message_id передано – виконується редагування через editMessageText. Якщо options задано – будується інлайн-клавіатура. Кнопки з коротким текстом (< 6 символів) групуються в один рядок;
3) delete_message(chat_id, message_id): видаляє повідомлення Telegram з вказаним ID. Результат логуються через logger.

3. Сервіс питань (QuestionService).

Сервіс відповідає за створення, фільтрацію, підбір та статистику пакетів питань. Він використовує моделі QuestionModel і DifficultModel, а також взаємодіє з темами, складністю та виконаними тестами.

Методи:

1) getPack(id): повертає пакет питань за ID;
2) getPacks(): повертає всі доступні пакети питань у базі;
3) getUncompletedTasks(completed_models, theme, difficult): повертає всі пакети, які відповідають вказаним темі й складності, але ще не були завершені користувачем. Ігнорує пакети, ID яких у списку completed_models;
4) createPack(pack): створює новий пакет питань з указаними полями: theme, questions, difficult;
5) get_questions_by_theme_and_difficult(theme, difficult): повертає всі питання (у вигляді списку рядків) з усіх пакетів, які відповідають вказаній темі та складності;
6) getLastPackTime(): повертає останній створений пакет питань (за датою);
7) get_users_packs_usage(user): формує статистику використання тем

користувачем. вираховує, скільки пакетів було завершено в кожній темі. Повертає відсортований список тем із найменш використаними першими.

8) get_difficult_name(id): Повертає об’єкт складності за її ID через модель DifficultModel.

1.3.8 Демонстрація роботи чат-боту

У цьому підрозділі представлено короткий опис основних функціональних команд чат-бота, реалізованого в межах дипломного проєкту. Бот розгорнутий у середовищі месенджера Telegram і орієнтований на самостійне тестування користувачів із англійської мови. Його взаємодія з користувачем здійснюється через стандартні текстові команди, кожна з яких відповідає окремій логічній функції.

Ця команда ініціює перший контакт користувача з ботом. Після її виклику бот надсилає вітальне повідомлення з короткою інструкцією щодо можливостей застосунку, а також надає список доступних команд для подальшої роботи. Команда також може використовуватися для повторного запуску, якщо користувач хоче «перезавантажити» сесію.

Відповіді бота максимально адаптовані для зручності користувача та простоти сприйняття інформації. Такий підхід сприяє інтуїтивній взаємодії і підвищує ефективність самостійного навчання.

Розглянемо команду /start. Взаємодя користувача з командою представлена на рисунку 1.19.

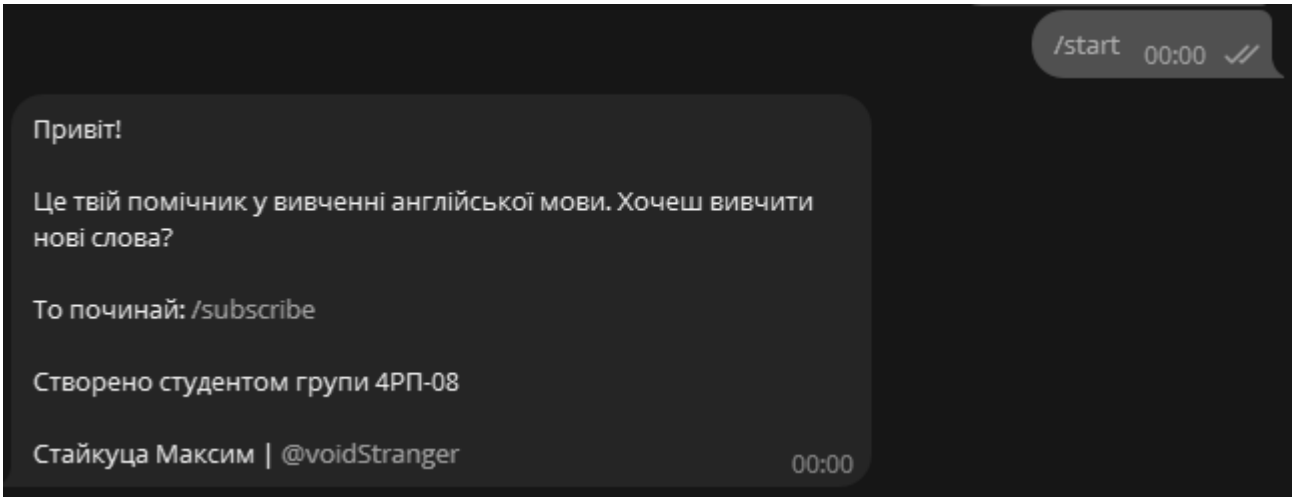


Рисунок 1.19. Демонстрація команди /start

Розглянемо команду / subscribe. Взаємодя користувача з командою представлена на рисунку 1.20.

Ця команда призначена для підписки користувача на бота. Після її активації користувач надає згоду на отримання автоматичних розсилок, зокрема – тестів чи нагадувань. Хоча механізм розсилки ще не реалізований повністю, команда є заділом для майбутньої розширеної функціональності.

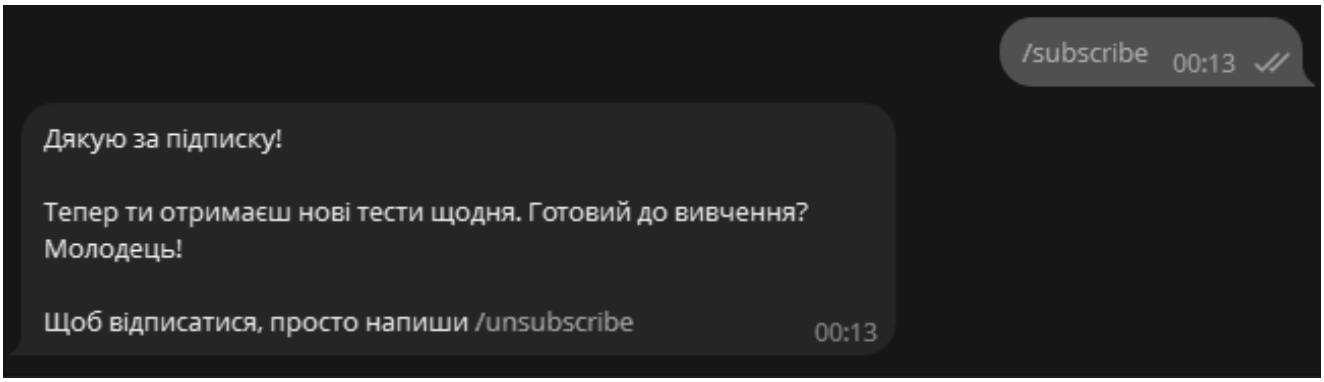


Рисунок 1.20. Демонстрація команди /subscribe

Розглянемо команду /unsubscribe. Взаємодія користувача з цією командою представлена на рисунку 1.21.

Ця команда дозволяє користувачеві скасувати підписку на розсилки чат-бота. Після її виконання бот вилучає користувача зі списку підписників і припиняє надсилання будь-яких автоматичних повідомлень, якщо вони були активовані.

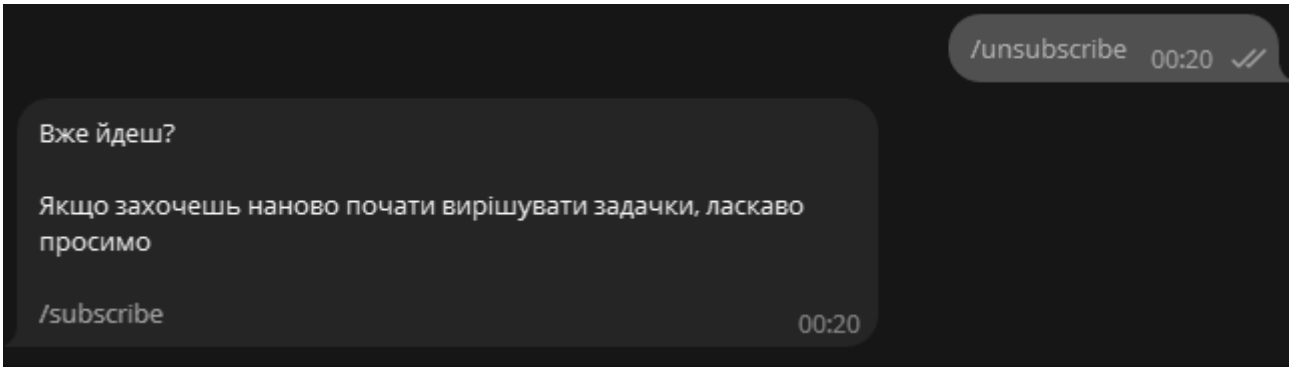


Рисунок 1.21. Демонстрація команди / unsubscribe

Розглянемо команду /test. Взаємодія користувача з цією командою представлена на рисунку 1.22 та 1.23.

За допомогою цієї команди користувач ініціює початок тестування. Після запуску бот деякий час формує тест відповідно до обраної теми та рівня складності. Потім послідовно надсилає 10 питань, кожне з яких містить чотири варіанти

відповіді. Після вибору варіанту бот відображає результат вибору: правильна відповідь позначається зеленою галочкою, неправильна – червоним хрестиком. Такий підхід забезпечує інтерактивність і зворотний зв’язок у режимі реального часу.

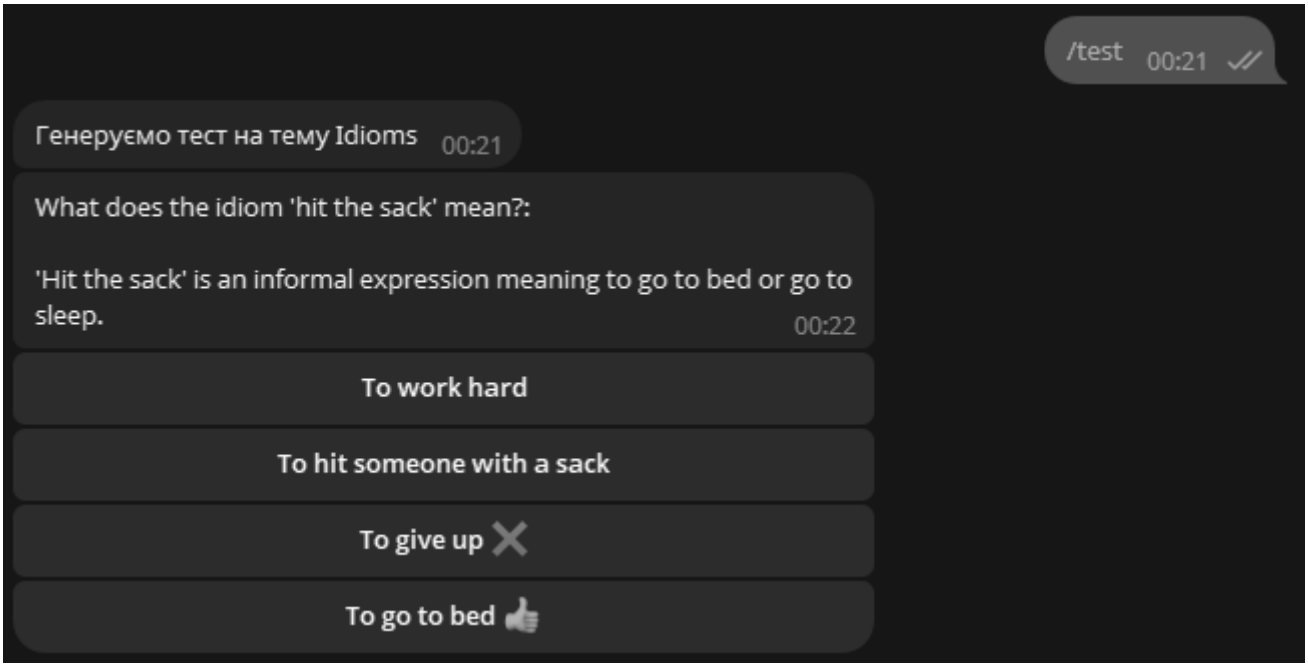


Рисунок 1.21. Демонстрація команди /test (частина 1)

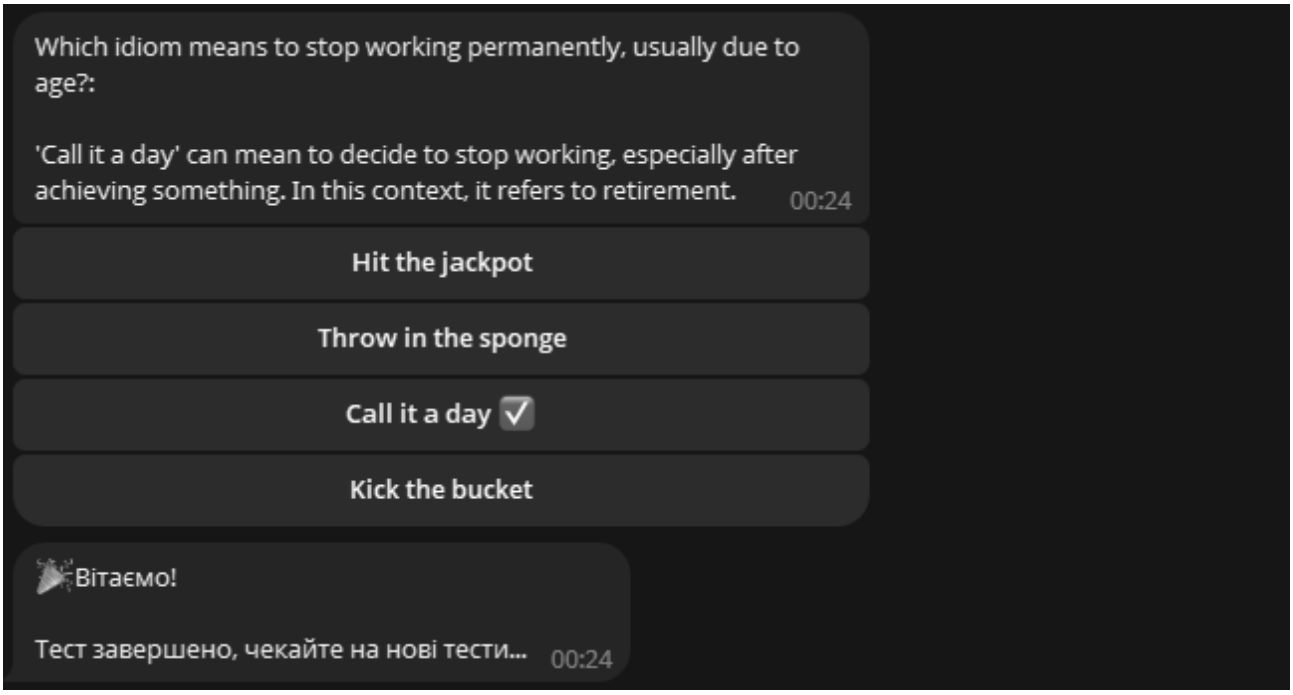


Рисунок 1.22. Демонстрація команди /test (частина 2)

Розглянемо команду /stop. Взаємодія користувача з цією командою представлена на рисунку 1.24.

Команда зупиняє поточну взаємодію з ботом. Вона не повертає жодного повідомлення у відповідь, однак припиняє виконання активної сесії або поточного процесу тестування.

Рисунок 1.24. Демонстрація команди /stop

Розглянемо команду /options. Взаємодія користувача з цією командою представлена на рисунку 1.25.

Команда відкриває меню вибору параметрів тесту. Користувач має можливість обрати одну з попередньо визначених тем (наприклад, «Граматика», «Фразові дієслова» тощо), а також встановити бажаний рівень складності: легкий, середній, складний або експертний. Обрані налаштування використовуються при формуванні наступного тесту.

Меню параметрів реалізоване у вигляді інтерактивних кнопок, що значно спрощує вибір для користувача та запобігає помилкам введення. Після вибору теми та рівня складності бот підтверджує обрані налаштування повідомленням, яке містить короткий огляд вибраних опцій. Користувач може в будь-який момент змінити параметри, повторно викликавши команду /options, що забезпечує гнучкість у налаштуванні процесу навчання. Окрім основних тем, у майбутніх версіях планується додати можливість створення кастомних тем та збереження улюблених налаштувань для швидкого доступу. Такий підхід дозволяє адаптувати тести під індивідуальні потреби кожного користувача та підвищує ефективність навчального процесу. Завдяки інтеграції з базою даних, обрані параметри зберігаються між сесіями, що робить роботу з ботом більш зручною та персоналізованою. Крім того, розроблено механізм валідації вибраних опцій, який гарантує коректність даних і запобігає виникненню помилок під час генерації тестів. У разі некоректного вибору користувачу надсилається відповідне повідомлення із підказкою щодо виправлення. Така структура взаємодії забезпечує інтуїтивність і простоту користування, що особливо важливо для користувачів із різним рівнем підготовки.

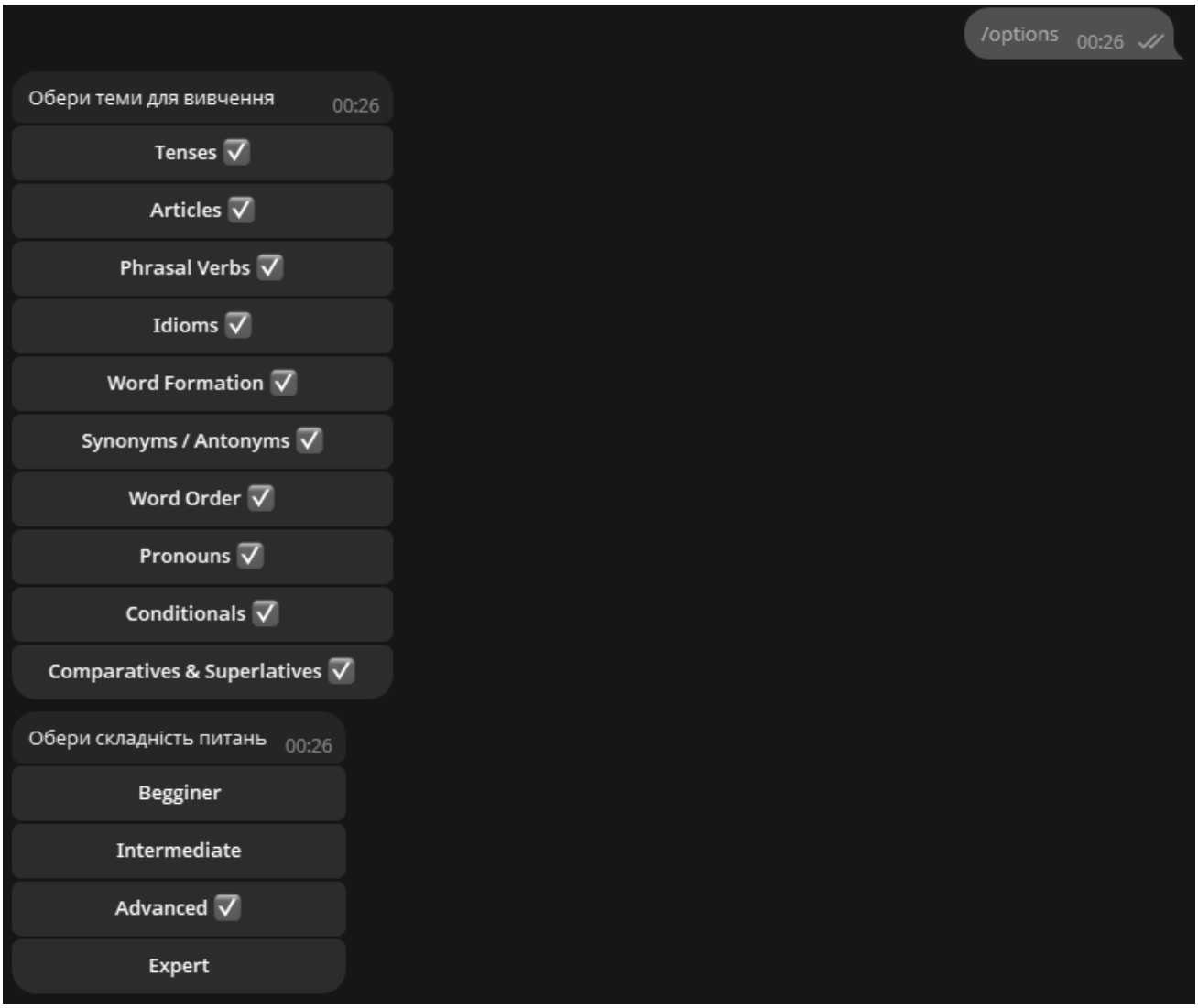


Рисунок 1.24. Демонстрація команди /options

Отриманий чат-бот успішно реалізує поставлені завдання, забезпечуючи зручну взаємодію користувачів із тестуванням англійської мови через Telegram. Використані підходи та методи підтвердили свою ефективність і дозволили створити стабільний та функціональний інструмент для самостійного навчання.

2 ЕКОНОМІЧНИЙ РОЗДІЛ

2.1 Резюме

У межах дипломного проєкту створено Telegram-чат-бот для перевірки знань англійської мови з інтерактивним тестуванням та миттєвим зворотним зв'язком. Застосунок реалізовано на Python із використанням Telegram API, що забезпечує адаптацію до рівня користувача та можливість подальшого розширення. Якість ПЗ оцінюється за функціональністю, ефективністю використання ресурсів та відповідністю технічним вимогам.

2.2 Розрахунок ціни програмного продукту нормативним методом

2.2.1 Визначення трудомісткості розробки програмного забезпечення

Тривалість розробки програмного продукту залежить від функціонального обсягу, трудомісткості етапів, кваліфікації виконавців та ринкових умов, при цьому обсяг програмних засобів визначається методом структурної аналогії на основі каталогів аналогів, приклади яких наведені в табл. 2.1.

Таблиця 2.1. Каталог аналогів

Найменування ПП	**Обсяг функції ПП – V_o, умов. машинних командах.**
1. ПП СУБД	2500 – 9800
2. Комплексні системи ведення БД	950 – 7430
3. ПП введення інформації	1060 – 5750

Вибравши аналог ПП, що містить Vo в умовних машинних командах, трудомісткості визначати на основі табл. 2.2.

Таблиця.2.2. Обсяг умовних команд

Обсяг ПП, тис.умов.машинних команд	**Норма часу, люд/год**
1.00	229
2.00	244
3.00	262
4.00	283
5.00	306

Виходячи з обсягу програмного забезпечення, за таблицею 2.2 визначають укрупнену норму часу на створення продукту, яка коригується поправочним коефіцієнтом Кк (0,7–0,8) з урахуванням умов розробки та використання обчислювальної техніки.

$$T^a p = \quad 229 * 0,7 = 160,3 \quad (люд/годин) \tag{2.1}$$

Розрахунок трудомісткості програмного продукту виконується окремо для кожного етапу (розробка технічного завдання, розробка технічного проєетк, розробка робочого проєкту), враховуючи складність, новизну функціоналу та використання типових модулів за відповідними формулами:

$$T_{ТЗ} = T^a p \times L_1 \times K_H \tag{2.2}$$

$$T_{ТП} = T^a p \times L_2 \times K_H \tag{2.3}$$

$$T_{РП} = T^a p \times L_3 \times K_H \times K_T \tag{2.4}$$

Для розрахунку необхідні наступні коефіцієнти: Li – питома вага і-го етапу розробки (див. табл. 2.3.); Кн – поправочний коефіцієнт, що враховує ступінь новизни (див. табл. 2.4.); Кт – поправочний коефіцієнт, що враховує ступінь використання в розробці типових програм (див. табл. 2.5.).

Таблиця 2.3. Значення питомих коефіцієнтів трудомісткості стадії в загальній трудомісткості розробки ПП.

Код стадії	**Ступінь новизни**		
	А	Б	В
ТЗ (L_1)	0,15	0,12	0,12
ТП (L_2)	0,16	0,15	0,11
РП (L_3)	0,55	0,58	0,61

Таблиця 2.4. Значення поправочного коефіцієнта, що враховує ступінь новизни

Код ступеня новизни	**Ступінь новизни**	**Значення $К_н$**
А	Принципово нові ПП	1,75 – 1,2
Б	ПП – розвиток визначеного параметричного ряду	1,0 – 0,8
В	ПП маючий аналог	0,7

Оскільки програмне забезпечення має функціональні аналоги, його ступінь новизни визначено як "В" з коефіцієнтом Кн=0,7, що за таблицями 2.3 і 2.4

дозволяє розрахувати питомі коефіцієнти трудомісткості L1=0,12, L2=0,11 та L3=0,61.

Таблиця 2.5. Значення коефіцієнта ступеня використання в розробці типових програм

Ступінь охоплення реалізованих функцій розроблювального ПП типовими програмами, %	**Значення $К_т$**
60 і вище	0,6
40-60	0,7
20-40	0,8
До 20	0,9

У розробленому програмному продукті використано близько 40–60 % типового функціоналу, що відповідає коефіцієнту типовості Кт=0,7, після чого можна приступити до покрокового розрахунку трудомісткості кожного етапу розробки.

Трудомісткість технічного завдання

$$T_{тз}=T^a p *L_1*К_н= 160,3 * 0,12 * 0,7 = 13,46 \text{ (люд/годин)} \quad (2.5)$$

Трудомісткість розробки технічного проекту

$$T_{тп}=T^a p *L_2*К_н = 160,3 * 0,11 * 0,7 = 12,34 \text{ (люд/годин)} \quad (2.6)$$

Трудомісткість розробки робочого проекту

$$T_{рп}=T^a p *L_3*К_н*К_т= 160,3 * 0,61 * 0,7 = 47,9 \text{ (люд/годин)} \quad (2.7)$$

Для подальших розрахунків визначено обсяг документації: Nтз = 1 сторінка, Nтп = 24 сторінки, Nрп = 21 сторінка, Nпз = 28 сторінок (див. табл. 2.6).

Таблиця 2.6. Розрахунок трудомісткості ПП

Найменування етапів	**Розрахунок, години.**		
	Розробка ПП	Контроль керівника	Нормоконтроль
1.ТЗ	$Т_{Ртз}$=13,46	$Т_{кк}=0,7*N_{тз}=$ 0,7*1=0,7	$Т_{нк}=0,15*N_{тз}=$ 0,15*1 =0,15
2.Розробка ТП	$Т_{Ртп}$= 12,34	$Т_{кк}=0,7*N_{тп}=0,7*24 =$ 16,8	$Т_{нк}=0,15*N_{тп}=$ 0,15*24 =3,6
3.Розробка РП	ТРрп=47,91	Ткк=0,7*Nрп=0,7*21 14,7	Тнк=0,15*Nрп= 0,15*21 =3,15

Продовження таблиці 2.6

4.Розробка пояснювальної записки	Трпз=1,5*Nпз=1,5*28= 42	Ткк=0,7*Nпз=0,7*28 =19,6	Тнк=0,15*Nпз= 0,15*28 =4,2
Усього, в т.ч.:	Тпп		
- на розробку	ΣТр=42+ 12,34+ 47,91+ 13,46 =115,71		
- контроль керівника		ΣТкк=19,6+14,7+16,8 +0,7 = 51,8	
- нормоконтроль			ΣТнк= 0,15+3,6+3,15+4,2=11,45

2.2.2 Розрахунок ціни програмного продукту

У цьому розділі розраховується вартість програмного продукту з урахуванням заробітної плати, матеріальних витрат і вартості машино-години, при цьому мінімальна погодинна тарифна ставка на 2025 рік встановлена на рівні 48 гривень (див. табл. 2.7).

Таблиця 2.7 Розрахунок основної заробітної плати виконавців.

Найменування робіт	**Трудомісткість робіт, роб.години**	**Годинна тарифна ставка,грн.**	**Розрахунок, грн.**
1.Розробка ПП	$\Sigma T_р$=115,71	50	5785
2.Контроль керівника	$\Sigma T_{кк}$=51,8	100	5180
3.Нормоконт-роль	$\Sigma T_{кк}$=11,45	100	1145
Усього ($З_о$)	-	-	Σ Зо= 12110

Розрахунок основної заробітної плати виконуємо по формулі:

$$Зо = Тмj \times Згод \quad (2.8)$$

де

- $Т_{мj}$ – трудомісткість j – того виду робіт, робоч. год;
- Згод – погодинна тарифна ставка, грн.

Зробимо розрахунок матеріальних витрат на розробку ПП. Розрахунок зведемо в таблицю 2.8

Таблиця 2.8. Розрахунок матеріальних витрат на розробку ПО

Найменування матеріальних витрат	**Тип, модель**	**Кількість, шт**	**Ціна одиниці, грн.**	**Вартість, грн.**
Папір А1	аркуш	-	-	0
Папір А4	аркуш	80	5	400
Разом	-	-	-	$В_{м\,і} = 80$
Транспортно – заготівельні Витрати 10%				$Втр\_з = 0{,}1 \times Вм1 = 40$
Усього				$В_м = В_{м1} + В_{тр\_з} = 440$

На підставі отриманих даних по окремих статтях витрат складена калькуляція планової собівартості в цілому ПП за формою, приведеною в таблиці 2.9.

Таблиця 2.9. Розрахунок статей витрат планової собівартості

Стаття витрат	**Значення, грн.**	**Формула розрахунку**
1. Матеріали	440	$В_м$ (див. табл. 2.8)
2. Основна заробітна плата	12110	$З_о$ (див. табл. 2.7)
3.Додаткова заробітна плата	121	$Зд = 0{,}1 \times Зо$
4.Відрахування до єдиного фонду соціального внеску	2690	$Вє.с.в. = 0{,}22 \times (Зо + Зд)$
5. Накладні витрати	7266	$Внак. = 0{,}6 \times Зо$
6. Повна собівартість	22627	$С_{пов} = В_м + З_о + З_д + Вє.с.в. +$ Внак.

Розмір прибутку, що включається в ціну, визначається по наступній формулі:

$$П=(С_{пов}*Р)/100 = (22627*10)/100 = 2262{,}7\ (грн) \qquad (2.6)$$

Де Р– плановий рівень рентабельності (10-15%).

Оптова ціна (кошторисна вартість) визначається по формулі:

$$Ц_о=С_{пов}+П= 22627 + 2262{,}7 = 24\,889{,}7\ (грн) \qquad (2.7)$$

Податок на додану вартість визначається по наступній формулі:

$$ПДВ = 0.2 * Ц_о = 24\,889{,}7* 0{,}2 = 4\,978\ (грн) \qquad (2.8)$$

Виходячи з отриманих даних, ціна програмного продукту становитиме:

$$Ц_р=Ц_о+ПДВ= 24\,889{,}7 + 4\,978 = 29\,867{,}7\ (грн) \qquad (2.9)$$

3 РОЗДІЛ ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

3.1 Основні положення

Охорона праці є важливою складовою будь-якої професійної діяльності та спрямована на збереження життя і здоров'я працівників. Оскільки розробка дипломного проєкту здійснювалась за комп'ютером, основну увагу приділено безпечним умовам праці при роботі з відеотерміналами.

У цьому розділі розглянуто вимоги до організації робочого місця користувача ПК відповідно до законодавства України. Зокрема, аналізуються фактори, що впливають на безпеку й комфорт: освітлення, мікроклімат, ергономіка, електробезпека, пожежна безпека та рівень шуму.

3.2 Вплив комп'ютерної роботи на здоров'я користувача

Тривале перебування за комп'ютером негативно позначається як на фізичному, так і на психоемоційному стані людини. Малорухомість спричиняє м'язове напруження, болі в спині, порушення кровообігу та загальне зниження фізичної активності.

Очі зазнають значного навантаження через постійну фіксацію погляду на екрані на одній відстані, що з часом призводить до втоми та погіршення зору.

Також часто спостерігаються психологічні наслідки – зниження концентрації, дратівливість, втома та проблеми зі сном, особливо при роботі у вечірній час. Постійна взаємодія з цифровими пристроями також може зменшувати рівень соціальної активності й призводити до емоційного виснаження.

3.3 Гігієнічні норми

3.3.1 Умови приміщення для роботи з комп'ютером

Для забезпечення безпечних та зручних умов праці було організовано індивідуальне робоче місце загальною площею 6,0 м². Такий простір відповідає мінімальним нормам згідно з чинними санітарно-гігієнічними вимогами для розміщення комп'ютерного обладнання.

Особливу увагу приділено оздобленню стін. Враховуючи те, що деякі полімерні оздоблювальні матеріали можуть виділяти леткі шкідливі речовини при нагріванні або з часом, у даному приміщенні використано безпечні, нетоксичні фарби. Крім того, кольорова гама відіграє роль у зниженні зорового та психоемоційного навантаження. Перевага надана нейтральним та м'яким кольорам – стіни мають пастельний зелений відтінок, який сприяє заспокоєнню нервової системи та створює сприятливу атмосферу для концентрації.

Підлога у приміщенні має матове покриття з неслизького матеріалу, що виключає ймовірність випадкових падінь. До того ж покриття володіє антистатичними властивостями, що знижує накопичення пилу та захищає електроніку від потенційних електростатичних розрядів.

3.3.2 Організація робочого місця при роботі з комп'ютером

Робоче місце було організоване з урахуванням ергономічних вимог для забезпечення комфорту та зниження навантаження на опорно-руховий апарат і зорову систему. Висота столу становила близько 680 мм, що дозволяло зручно розмістити клавіатуру, мишу і монітор. Робоча поверхня мала достатню ширину та глибину для вільного доступу до обладнання без зайвих нахилів, а простір під столом відповідав рекомендованим параметрам (не менше 600 мм у висоту, 500 мм у ширину, 450 мм у глибину) для комфортного положення ніг.

Використовувався офісний стілець із регулюванням висоти, нахилу спинки і положення сидіння, з ергономічною формою та напівм'якою оббивкою, що не ковзає та легко очищується.

Монітор розташовувався на відстані 600–700 мм від очей користувача, під кутом близько 30° вниз від рівня очей, що знижувало навантаження на шию та очі. Клавіатура розміщувалась на відстані 100–300 мм від краю столу з матовою поверхнею (коефіцієнт відбиття світла до 0,4) і висотою клавіш до 30 мм для зниження втоми.

Всі пристрої введення та виведення були розміщені в межах зони доступу рук (900–1300 мм по висоті, до 500 мм по ширині), що забезпечувало зручне і безпечне керування без зайвих рухів. Всі параметри можна побачити на рисунку

3.1

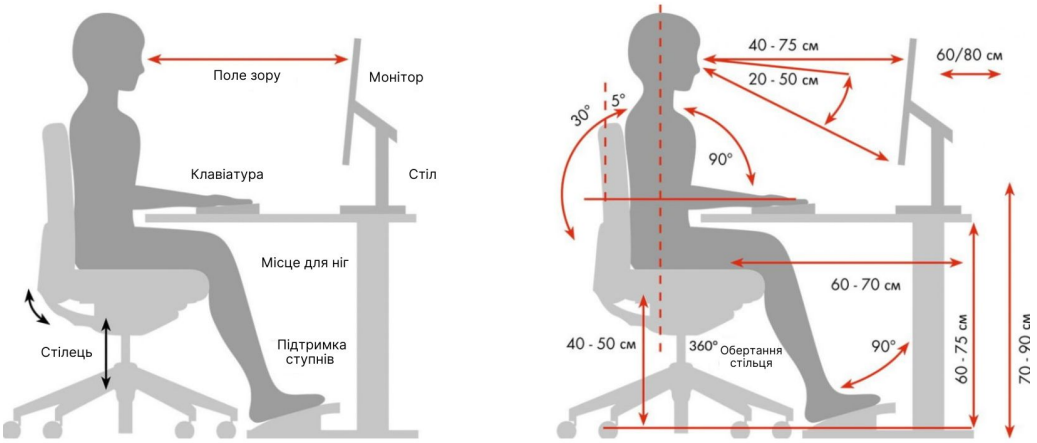


Рисунок 3.1. Параметри для робочого місця

3.3.3 Освітлення робочої зони

Освітлення робочого місця було організовано відповідно до нормативних вимог для роботи з електронними пристроями, враховуючи природне та штучне освітлення. Природне світло надходило через вікна з площею світлопрозорих конструкцій, що забезпечували коефіцієнт природної освітленості не менше 1,5%. Для запобігання відблискам використовували жалюзі та розташовували робоче місце так, щоб уникнути прямого попадання сонячних променів на монітор.

Штучне освітлення забезпечувалося рівномірним розсіюваним світлом від люмінесцентних або світлодіодних світильників, що уникало яскравих променів і тіней. Рівень освітленості на робочій поверхні становив 300–500 лк, що відповідає гігієнічним нормам. Заборонялося використання світильників без розсіювачів, які можуть створювати зайву яскравість і погіршувати зір.

3.3.4 Рівень шуму та вібрацій у робочому середовищі

Під час роботи над проектом забезпечували сприятливе акустичне середовище з рівнем шуму не вище 50 дБА, що відповідає гігієнічним нормам і сприяє концентрації без психоемоційного перенапруження. Для зниження шуму використовували малошумне обладнання та звукопоглинальні матеріали (м’які оздоблення, жалюзі).

Вібрації від техніки були незначними та не перевищували нормативних меж. Стабільне розміщення обладнання й антивібраційні підставки запобігали негативному впливу вібрацій на користувача.

3.3.5 Мікрокліматичні умови на робочому місці

Для комфортної роботи підтримувався оптимальний мікроклімат: температура 22–25 °C, вологість 40–60% – нормативні показники для тривалого перебування за комп'ютером. Швидкість руху повітря не перевищувала 0,1 м/с, що запобігає протягам.

Приміщення оснащене системою опалення і кондиціонування, що дозволяє регулювати умови незалежно від сезону. Регулярне провітрювання (приблизно кожні 2 години) забезпечує свіжість повітря і підтримує продуктивність праці.

3.3.6 Дотримання вимог електробезпеки

Оскільки робота здійснюється з електронним обладнанням, обов'язковим є дотримання норм електробезпеки для запобігання ураженню струмом та аваріям. Перед початком роботи перевіряють технічний стан: справність кабелів, ізоляції, корпусів та з'єднань. Використовується лише справна техніка, що відповідає нормам.

Після роботи обладнання вимикають з мережі, особливо при тривалих перервах, щоб уникнути коротких замикань і перегріву. Заборонено користуватися пристроями з мокрими руками або у вологих умовах.

У разі виявлення несправностей (іскріння, запах, шум) обладнання негайно вимикають і передають на ремонт. Усі дії виконуються відповідно до законодавства України з охорони праці та електробезпеки.

3.3 Забезпечення пожежної безпеки

Пожежна безпека є важливою складовою охорони праці при роботі з електронним обладнанням. Під час реалізації проєкту застосовувався комплекс заходів для мінімізації ризиків пожеж.

Перед початком роботи користувач пройшов інструктаж з пожежної безпеки відповідно до норм. Особлива увага приділялась експлуатації електроприладів:

заборонялося використовувати пошкоджене обладнання, перевантажувати електромережу та залишати пристрої ввімкненими без нагляду. Регулярно перевіряли справність розеток, фільтрів живлення та подовжувачів для виявлення дефектів чи перегріву.

У робочому приміщенні встановлені первинні засоби пожежогасіння – порошковий або вуглекислотний вогнегасник, які безпечні для електрообладнання. Вогнегасник розміщений у легкодоступному місці, на безпечній відстані від джерел загоряння, поряд із інструкцією, а його технічний стан регулярно контролюється.

Також у приміщенні розміщена схема евакуації з чіткими позначеннями напрямків руху при надзвичайних ситуаціях. Всі заходи відповідають чинним нормативам пожежної безпеки України.

3.4 Заходи для збереження здоров’я під час роботи з комп’ютером

Під час роботи над програмним забезпеченням дотримувалися рекомендацій з охорони праці та режиму праці й відпочинку: після кожної години роботи робилися 5–10-хвилинні перерви з фізичними вправами для зняття м’язового напруження і гімнастикою для очей.

Для підтримки фізичного тонусу практикували активний відпочинок або заняття спортом, що сприяло покращенню кровообігу, зниженню стресу і профілактиці професійних захворювань.

Щоб зменшити вплив синього світла від монітора, використовували спеціальні фільтри або режими зниження синього світла, які активуються автоматично або вручну.

Відповідно до норм охорони здоров’я, проходили щорічні медичні огляди, включно з офтальмологом і терапевтом, для раннього виявлення можливих проблем і своєчасного коригування.

ВИСНОВКИ

У рамках виконання дипломної роботи було реалізовано повнофункціональний чат-бот для перевірки знань з англійської мови на базі штучного інтелекту. Цей програмний продукт поєднує інструменти автоматизації тестування та персоналізованого підходу до самоперевірки, що дозволяє користувачеві оперативно взаємодіяти з навчальним контентом у звичному середовищі – месенджері Telegram.

У процесі розробки було досягнуто поставлену мету: створено зручний та ефективний застосунок, який дозволяє проходити тести на різні теми та рівні складності, отримувати миттєвий зворотний зв'язок і пояснення до кожної відповіді. Бот також зберігає статистику користувача, що сприяє більш усвідомленому самоконтролю та підвищенню мотивації до навчання.

Технічна реалізація поєднує сучасні інструменти: мову програмування Python, бібліотеку requests для взаємодії з Telegram API, ШІ-сервіс для генерації навчального контенту, а також MongoDB для зберігання інформації про користувачів, питання, теми й рівень складності. Усі компоненти було інтегровано в єдину архітектуру, що забезпечує надійність і стабільну роботу бота.

Загальне тестування серед користувачів підтвердило відповідність системи поставленим задачам: бот демонструє очікувану поведінку, коректно реагує на команди та виконує свою основну функцію – перевірку знань з англійської мови. Зворотний зв'язок від тестових користувачів був переважно позитивним.

Попри завершеність функціоналу, під час розробки виникали нові ідеї щодо його розширення. Серед можливих покращень – реалізація щоденних тестів, система досягнень, таблиця лідерів, підтримка групових чатів та інтеграція елементів гейміфікації. Хоча ці функції не входили до первинного плану дипломного проєкту, їх потенціал варто враховувати при подальшому розвитку.

Підсумовуючи, розроблений чат-бот може бути використаний як повноцінний освітній інструмент для самостійної перевірки знань з англійської мови. Він поєднує в собі простоту використання, адаптивність і можливості штучного інтелекту, що відповідає сучасним тенденціям цифрової освіти.

ПЕРЕЛІК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. Ось оформлений список джерел відповідно до ДСТУ 8302:2015, без нумерації, з додаванням дат звернення в діапазоні 14.05.2025 – 10.06.2025:
2. Олексій Васильєв. Програмування мовою Python. – Київ : Арій, 2023. – 328 с.
3. Когут, Ю. Штучний інтелект і безпека. – Харків : Сідкон, 2024. – 256 с.
4. Chowdhary, K. R. Fundamentals of Artificial Intelligence [Електронний ресурс]. – Режим доступу: https://link.springer.com/book/10.1007/978-81-322-3972-7 – Назва з екрана. – Мова англійська. – Дата звернення: 15.05.2025.
5. ICLS. 10 most spoken languages in the world in 2025 [Електронний ресурс]. – Режим доступу: https://www.icls.edu/blog/most-spoken-languages-in-the-world – Назва з екрана. – Мова англійська. – Дата звернення: 17.05.2025.
6. 5 Minute English. The Role of English in the Digital World [Електронний ресурс]. – Режим доступу: https://5minuteenglish.com/the-role-of-english-in-the-digital-world – Назва з екрана. – Мова англійська. – Дата звернення: 18.05.2025.
7. National Center for Biotechnology Information. Artificial Intelligence in Healthcare [Електронний ресурс]. – Режим доступу: https://pmc.ncbi.nlm.nih.gov/articles/PMC7256567/ – Назва з екрана. – Мова англійська. – Дата звернення: 20.05.2025.
8. BuiltIn. Artificial Intelligence Overview and Trends [Електронний ресурс]. – Режим доступу: https://builtin.com/artificial-intelligence – Назва з екрана. – Мова англійська. – Дата звернення: 25.05.2025.
9. SpringerOpen. Systematic Review of Research on Artificial Intelligence Applications in Higher Education – Where Are the Educators? [Електронний ресурс]. – Режим доступу: https://educationaltechnologyjournal.springeropen.com/articles/10.1186/s41239-019-0171-0 – Назва з екрана. – Мова англійська. – Дата звернення: 10.06.2025.

10..

ДОДАТОК А. Фрагмент програмного коду боту та сервісу ШІ

```
#BOT.PY

import threading
import time
from dotenv import load_dotenv, find_dotenv
import os

from utils import logger
from views.TelegramView import TelegramView
from models import ThemeModel
from service import QuestionService, TelegramService, UserService
from controllers import BotController, QuizController, SubscribeController

def init_bot():
    load_dotenv(find_dotenv())
    token = os.getenv("TELEGRAM_API_KEY")
    if not token:
        raise ValueError("TELEGRAM_API_KEY is not set")

    telegram_view = TelegramView()
    quiz_controller = QuizController()
    telegram_service = TelegramService(token)
    theme_service = ThemeModel()
    subscribe_controller = SubscribeController(telegram_service)
    bot_controller       =       BotController(telegram_service,       quiz_controller,
subscribe_controller)

    run_bot_loop(bot_controller,                              VeiwInstance=telegram_view,
QuizControllerInstance=quiz_controller,            TelegramBotController=bot_controller,
ThemeServiceInstance=theme_service, SubscribeControllerInstance=subscribe_controller)

serverData = {
    "offset": 0,
    "lastQuiz": 0
}

def  run_bot_loop(bot,  VeiwInstance,  QuizControllerInstance,  TelegramBotController,
ThemeServiceInstance, SubscribeControllerInstance):
    while True:
        try:
            updateChecker = bot.handle_updates(offset=serverData["offset"])
            serverData["offset"] = updateChecker["offset"]
            data = bot.process_message(updateChecker)

            if data and data["type"] == "command":
                logger(f"Bot: {data}")
                handle_command(data=data,
SubscribeControllerInstance=SubscribeControllerInstance,
TelegramBotController=TelegramBotController,
QuizControllerInstance=QuizControllerInstance,                VeiwInstance=VeiwInstance,
ThemeServiceInstance=ThemeServiceInstance)
            elif data and data["type"] == "callback":
                handle_callback(data, SubscribeControllerInstance, TelegramBotController,
VeiwInstance, QuizControllerInstance)

        except Exception as e:
            logger(f"Bot encountered an error: {e}")
            time.sleep(1)
```

```
def handle_command(data, SubscribeControllerInstance, TelegramBotController,
QuizControllerInstance, ThemeServiceInstance, VeiwInstance):
    if data["command"] == 'subscribe':
        SubscribeControllerInstance.subscribe_user(data["chat_id"])
    elif data["command"] == 'unsubscribe':
        SubscribeControllerInstance.unsubscribe_user(data["chat_id"])
    elif data["command"] == 'options':
        SubscribeControllerInstance.pick_themes_message(data["chat_id"])
        SubscribeControllerInstance.pick_difficult(data["chat_id"])
    elif data["command"] == 'stop':
            UserService.delete_session(data["chat_id"])
    elif data["command"] == 'test':
        user = UserService.get_one(data["chat_id"])
        if not user or not user.subscribed:
            TelegramBotController.send_message(data["chat_id"],
VeiwInstance.subscribe_first_message())
        elif not user.picked_themes or not user.difficult:
            TelegramBotController.send_message(data["chat_id"],
VeiwInstance.theme_and_difficulty_first_message())
        else:
            active_session = user.active_session
            if active_session:
                pack = QuestionService.getPack(user.active_session["question_pack_id"])
                if pack:
                    quizData = QuizControllerInstance.start_quiz(user, pack)
                else:
                    UserService.delete_session(data["chat_id"])
            else:
                sortedUserThemesUsage = QuestionService.get_users_packs_usage(user)
                theme = ThemeServiceInstance.pick_least_used_theme(sortedUserThemesUsage)

                completed_packs = user.completed_quizzes
                difficult = user.difficult
                uncompletedTasks  =  QuestionService.getUncompletedTasks(completed_packs,
theme, difficult)

                if not uncompletedTasks:
                    TelegramBotController.send_message(data["chat_id"],
VeiwInstance.generating_process_message(theme["title"]))
                    newPack = QuizControllerInstance.generate_quiz(theme, difficult)
                    quizData = QuizControllerInstance.start_quiz(user, newPack)
                else:
                    pack = uncompletedTasks[0]
                    quizData = QuizControllerInstance.start_quiz(user, pack)
                    TelegramBotController.send_message(data["chat_id"],
VeiwInstance.starting_theme_message(theme["title"]))
            if isinstance(quizData, str):
                UserService.delete_session(data["chat_id"])
                TelegramBotController.send_message(data["chat_id"], quizData)
            else:
                quizMessage     =     TelegramBotController.send_message(data["chat_id"],
quizData["options"].get("text"), quizData["options"].get("qeustions"))
                UserService.update_session(data["chat_id"],
quizData["selected_pack_id"],              quizMessage,              quizData["current"],
quizData["options"].get("text"))

            if not active_session and theme and uncompletedTasks and len(uncompletedTasks)
< 2:
                def generate_second_quiz():
                    try:
                        logger(f"starting genereting new pack on second thread...")
                        QuizControllerInstance.generate_quiz(theme, difficult)
                    except Exception as e:
                        logger(f"Failed to pre-generate second quiz pack: {e}")
```

```
                threading.Thread(target=generate_second_quiz).start()

def     handle_callback(data,     SubscribeControllerInstance,     TelegramBotController,
VeiwInstance, QuizControllerInstance):
    if data["command"] == 'theme_pick':
        SubscribeControllerInstance.add_theme(data["chat_id"], data["data"])
        SubscribeControllerInstance.pick_themes_message(data["chat_id"],
data["callback_message_id"])
    if data["command"] == 'difficult_pick':
        SubscribeControllerInstance.set_difficult(data["chat_id"], data["data"])
        SubscribeControllerInstance.pick_difficult(data["chat_id"],
data["callback_message_id"])
    if data["command"] == 'T' or data["command"] == 'F':
        user = UserService.get_one(data["chat_id"])
        if   user.active_session   and   not   user.active_session["last_question"]   ==
data["callback_text"]:
            TelegramBotController.send_message(data["chat_id"],
VeiwInstance.do_not_rush_message())
        elif not user.active_session:
            TelegramBotController.send_message(data["chat_id"], VeiwInstance.no_test())
        else:
            quizData = QuizControllerInstance.next_quiz(user)

            textString     =     f'{data["callback_text"]}:\n\n{quizData["reason"]     if
quizData["reason"] else ""}'
            TelegramBotController.send_message(data["chat_id"],               textString,
data["updated_reply_markup"], user.active_session["message_id"])

            if isinstance(quizData["current"], bool):
                print(quizData["current"])
                if quizData["current"] == True:
                    UserService.push_completed_task(data["chat_id"],
user.active_session.get("question_pack_id"))
                    UserService.delete_session(data["chat_id"])
                    TelegramBotController.send_message(data["chat_id"],
VeiwInstance.end_test())
                else:
                    return
                    TelegramBotController.send_message(data["chat_id"],
VeiwInstance.error())
            else:
                quizMessage     =     TelegramBotController.send_message(data["chat_id"],
quizData["options"].get("text"), quizData["options"].get("qeustions"))
                UserService.update_session(data["chat_id"],
quizData["selected_pack_id"],              quizMessage,              quizData["current"],
quizData["options"].get("text"))

#AIService

import json
import os
import random
import re
from dotenv import load_dotenv, find_dotenv
import requests

from utils import logger

load_dotenv(find_dotenv())

def build_theme_prompt(themes):
    themeData = f"Не повторюй такі як: {json.dumps(themes, ensure_ascii=False, indent=2)}"
if themes else ""
    return f"""
```

```
    Згенеруй масив із 10 унікальних тем для тестування знань з англійської мови.
     Кожна тема повинна бути короткою (1-3 слова) і зрозумілою (наприклад: "Подорожі",
"Інтернет", "Кіно", "Ресторан").

    Формат відповіді – ЧИСТИЙ JSON масив рядків:
    [
        "тема1",
        "тема2",
        ...
    ]
    {themeData}
    """

def build_quiz_prompt(history, theme, difficult):
    themeData = theme if theme else "IT"
    difficultData = difficult if difficult else "Begginer"
     historyData = f"Не повторюй такі питання: {json.dumps(history, ensure_ascii=False,
indent=2)}" if history else ""
    return f"""
         Згенеруй масив із 10 об'єктів для тестування знань з англійської мови на рівні
{difficultData} на тему {themeData} у форматі:
        ```json
        [
        {{
            "question": "Текст питання",
            "answers": ["варіант 1", "варіант 2", "варіант 3", "варіант 4"],
            "correctAnswer": "правильний варіант",
            "reasonOfAnswer": "аргументована відповіть чому відповідь саме така"
        }}
        ]
        ```
        {historyData}
        Формат відповіді – ЧИСТИЙ JSON, без додаткового тексту.
    """

class AIService:
    def __init__(self):
        self.url = os.getenv("AI_API_URL")
        self.key = os.getenv("AI_API_KEY")
        self.model = os.getenv("AI_API_MODEL")

    def get_questions(self, prompt):
        logger("fetching new qestion pack")
        try:
            response = requests.post(
                self.url,
                headers={
                    "Authorization": f"Bearer {self.key}",
                    "Content-Type": "application/json",
                },
                data=json.dumps({
                    "model": f"{self.model}",
                    "messages": [
                        {
                            "role": "user",
                            "content": prompt
                        }
                    ],
                })
)
                      data   =   response.json().get('choices',   [{}])[0].get('message',
{}).get('content')
            logger(data)
            return data
```

```
    except Exception as e:
        logger(f"Error fetching AI question: {e}")
        return None

def get_new_ai_question(self, history, theme, difficult):
   logger(f"Started generation new qestion pack: theme={theme} difficult={difficult}")
    prompt = build_quiz_prompt(history, theme, difficult)
    data = self.get_questions(prompt)
    return self.parse_ai_questions(data) if data else None

def get_new_ai_themes(self, themes):
    logger("Started generation new themes")
    prompt = build_theme_prompt(themes)
    data = self.get_questions(prompt)
    return self.parse_ai_themes(data) if data else None

@staticmethod
def parse_ai_questions(ai_message):
    logger("parsing new qestion pack")
    try:
        match = re.search(r'```json\s*(.*?)\s*```', ai_message, re.DOTALL)

        if match:
            json_content = match.group(1)
        else:
            logger("No JSON block found.")
            return None

        parsed = json.loads(json_content)

        quizzes = []
        for quiz in parsed:
            question_text = quiz.get('question', 'No question provided')
            answers = quiz.get('answers', quiz.get('answer', []))
            correct_answer = quiz.get('correctAnswer')
            reason = quiz.get('reasonOfAnswer')

            options = []
            for i, answer in enumerate(answers):
                isCorrect = 'T' if answer == correct_answer else 'F'
                callback = f"{isCorrect}:{i}"
                options.append({
                    'text': answer,
                    'callback_data': callback
                })

            random.shuffle(options)

            quizzes.append({
                'text': question_text,
                'options': options,
                'reason': reason
            })

        return quizzes
```

ДОДАТОК Б. Слайди мультимедійної презентації

ВІДОКРЕМЛЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ
«ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОДЕСЬКОГО НАЦІОНАЛЬНОГО ТЕХНОЛОГІЧНОГО УНІВЕРСИТЕТУ»

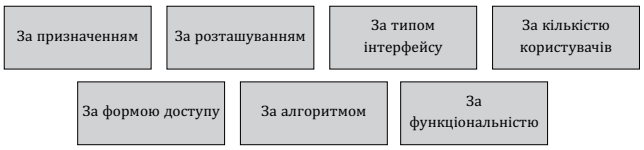
РОЗРОБКА ТЕЛЕГРАМ БОТУ ДЛЯ ПЕРЕВІРКИ ЗНАНЬ З АНГЛІЙСЬКОЇ МОВИ НА БАЗІ ШІ

ДИПЛОМНА РОБОТА

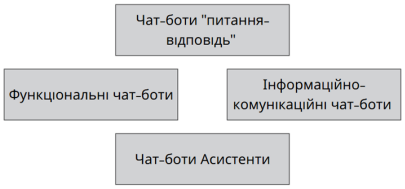
Керівник: Жадан А.С.
Виконав: Стайкуца М.С.

2025

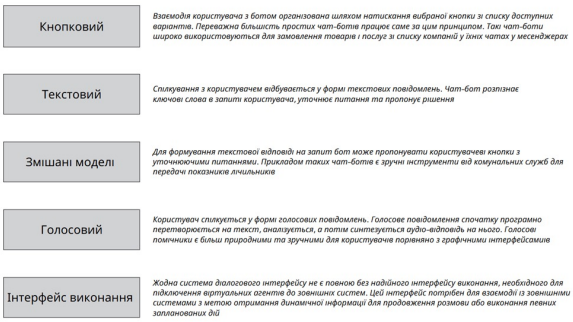
Базова класифікація чат-ботів



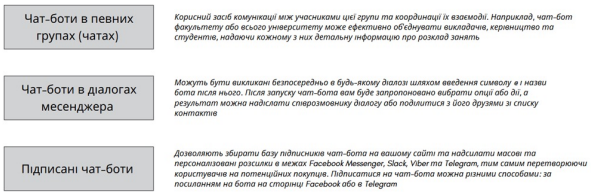
Загальна класифікація чат-ботів



Класифікація чат-ботів за типом функціональності



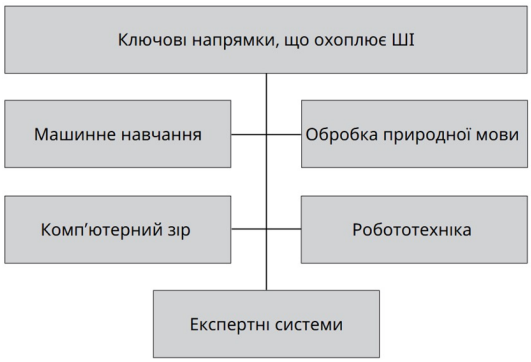
Класифікація чат-ботів за типом інтерфейсу



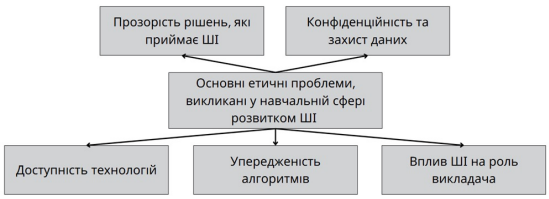
Класифікація чат-ботів за формою доступу

Роль штучного інтелекту у сфері освіти

Штучний інтелект (ШІ) *– це галузь комп'ютерної науки, що зосереджується на створенні систем, здатних виконувати завдання, які зазвичай потребують людського інтелекту. Загалом, суть ШІ полягає в імітації когнітивних функцій людини за допомогою алгоритмів і моделей машинного навчання.*

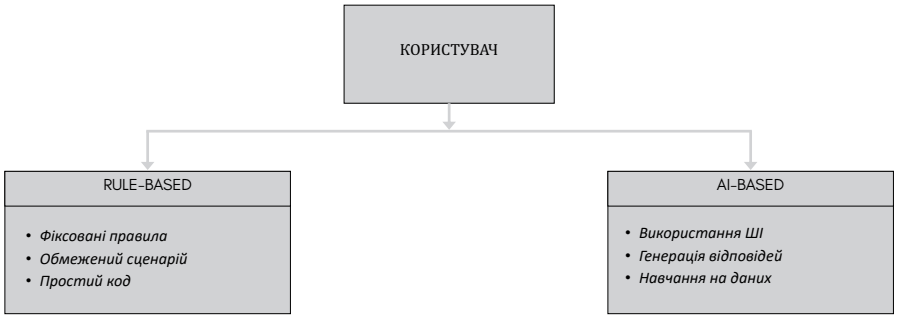


Ключові напрямки, що охоплює ШІ

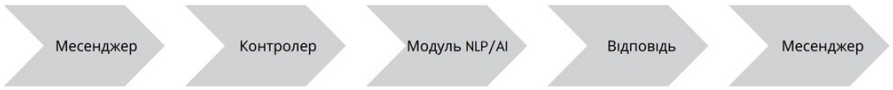


Основні етичні проблеми, викликані у навчальній сфері розвитком ШІ

Методи та технології ШІ в аспекті екосистеми чат-ботів



Основні Методи та технології ШІ в аспекті екосистеми чат-ботів

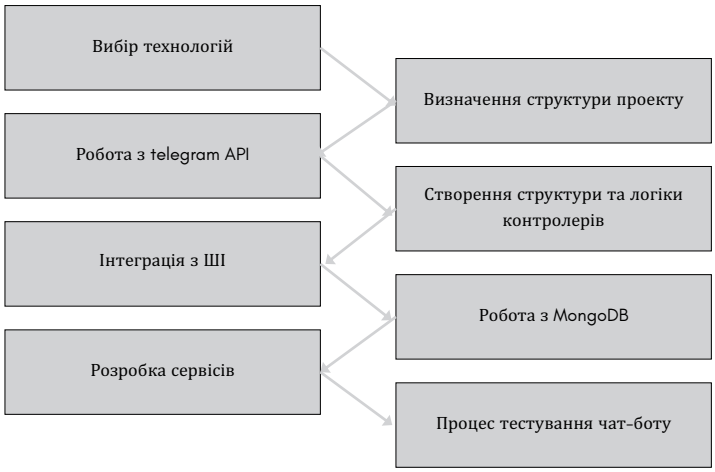


Принцип взаємодії інтелектуального ядра на базі ШІ та інтерфейсу месенджера

Роль ШІ у чат-ботах

Аналіз	Генерація	Пояснення
ШІ враховує тему й складність Формує запит із урахуванням історії користувача	Генерує 10 тестових питань у форматі JSON Кожне питання має 4 варіанти відповідей	Відмічає правильну відповідь і пояснює чому Формулює пояснення природною мовою та відсилаючись на правила

Алгоритм розробки чат-боту для перевірки знань з англійської мови



Список використаних технологій

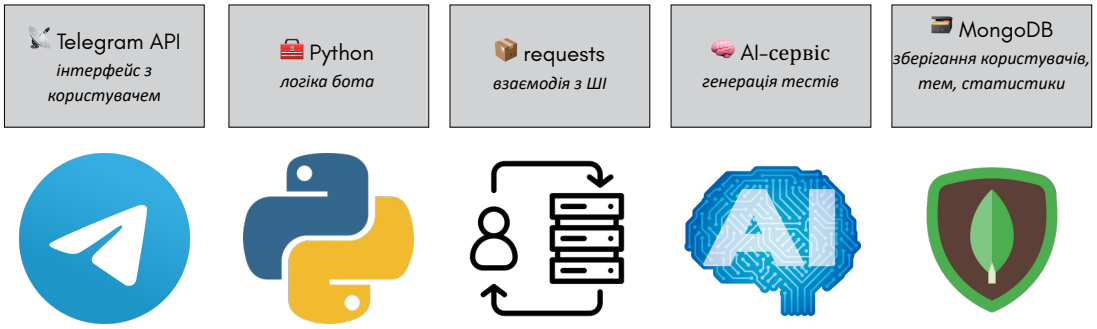
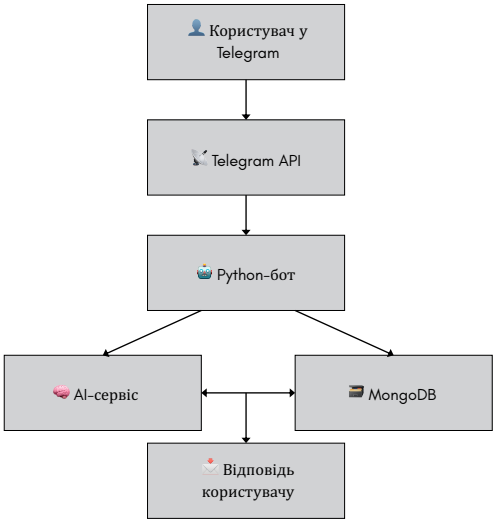
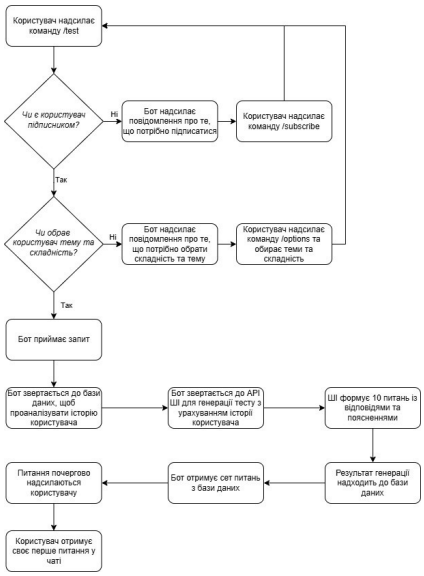


Схема роботи боту



Загальна структура телеграм-боту



Блок-схема роботи телеграм-боту

Робота з telegram API

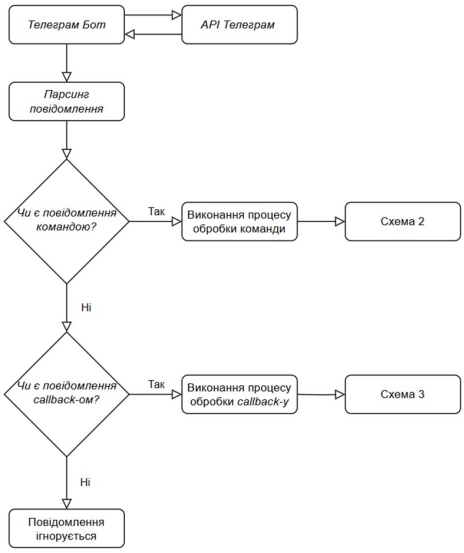
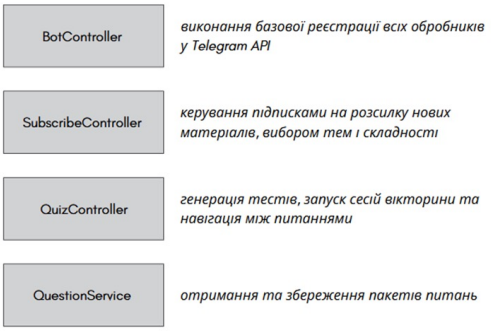


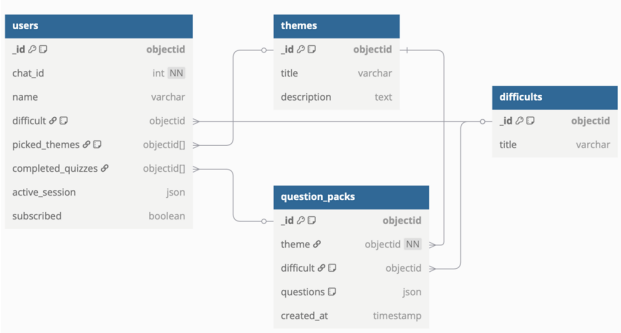
Схема зв'язку чат-боту з Телеграм API

/start	*надсилання вітального повідомлення, у якому коротко пояснюється, як працює вікторина*
/subscribe	*зберігання інформації про підписку користувача для подальших сповіщень*
/unsubscribe	*скасування підписки*
/options	*можливість користувачеві обрати тему і складність вікторини через інтерактивні кнопки*
/test	*ініціювання самої вікторини, після чого користувач почне отримувати питання із варіантами відповідей*

Основні команди та їх функціонал



Класи контролерів та їх функціонал



Діаграма відношення таблиць в базі даних


```python
def get_new_ai_question(self, history, theme, difficult):
    logger(f"Started generation new qestion pack: theme={theme} difficult={difficult}")
    prompt = build_quiz_prompt(history, theme, difficult)
    data = self.get_questions(prompt)
    return self.parse_ai_questions(data) if data else None
```

Виклик генерації нових питань

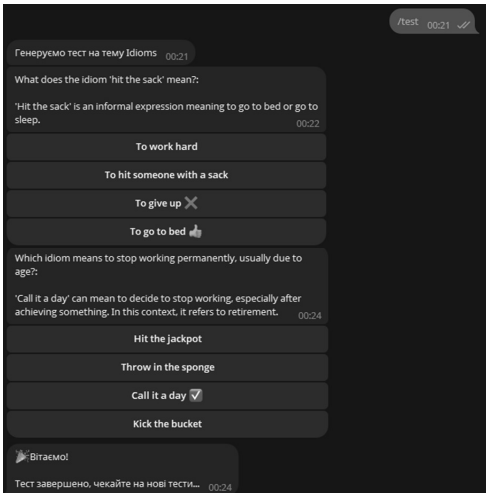
```python
def build_quiz_prompt(history, theme, difficult):
    themeData = theme if theme else "IT"
    difficultData = difficult if difficult else "Begginer"
    historyData = f"Не повторяй такі питання: {json.dumps(history, ensure_ascii=False, indent=2)}" if history else ""
    return f"""
        Згенеруй масив із 10 об'єктів для тестування знань з англійської мови на рівні {difficultData} на тему {themeData} у форматі:
        ```json
        [
        {{
            "question": "Текст питання",
            "answers": ["варіант 1", "варіант 2", "варіант 3", "варіант 4"],
            "correctAnswer": "правильний варіант",
            "reasonOfAnswer": "аргументована відповідь чому відповідь саме така"
        }}
        ]
        ```
        {historyData}
        Формат відповіді — ЧИСТИЙ JSON, без додаткового тексту.
    """
```

Створення промпта

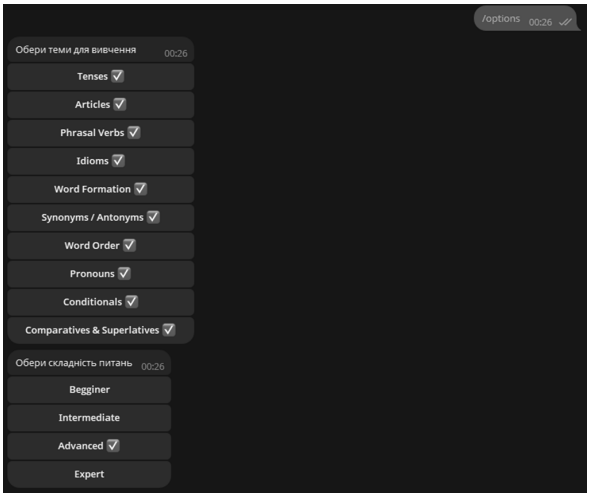
```python
def get_questions(self, prompt):
    logger("Fetching new qestion pack")
    try:
        response = requests.post(
            self.url,
            headers={
                "Authorization": f"Bearer {self.key}",
                "Content-Type": "application/json",
            },
            data=json.dumps({
                "model": f"{self.model}",
                "messages": [
                    {
                        "role": "user",
                        "content": prompt
                    }
                ],
            })
        )
        data = response.json().get('choices', [{}])[0].get('message', {}).get('content')
        logger(data)
        return data
    except Exception as e:
        logger(f"Error fetching AI question: {e}")
        return None
```

Запит до штучного інтелекту

Демонстрація роботи чат-боту



Демонстрація команди /test



Демонстрація команди /options

@ENGTEACHER_LILBOT

Демонстрація

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

РЕЦЕНЗІЯ

на дипломний проект здобувача (здобувачки) освіти

відділення комп'ютерних систем

Стайкуци Максима Сергійовича

(прізвище, ім`я та по батькові)

Спеціальність *121 «Інженерія програмного забезпечення»*

Освітня програма *«Розробка програмного забезпечення»*

Керівник дипломного проекту (роботи) *Закроєв Юрій Михайлович*

(прізвище, ім`я та по батькові)

Тема дипломного проекту (роботи) *Розробка телеграм боту для перевірки знань з англійської мови на базі ШІ*

Обсяг розрахунково-пояснювальної записки *84* сторінок

Обсяг графічної (презентаційної) частини *14* аркушів (слайдів)

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ (РОБОТИ)

а) заключення про ступінь відповідності виконаного дипломного проекту завданню

Представлений на рецензію дипломний проект відповідає затвердженій темі та виконаний відповідно технічному завданню. Дипломний проект присвячений проблемі вивчення іноземних мов та складається з пояснювальної записки, додатку з програмним кодом та мультимедійної презентації, що містить приклади роботи програми.

б) характеристика виконання кожного розділу дипломного проекту

Пояснювальна записка складається з основного розділу (аналізу предметної області, проектування застосунку, реалізації застосунку, тестування застосунку), економічного розділу, розділу охорони праці та додатків. Перелічені розділи поетапно охоплюють розробку, виконані докладно та обґрунтовано. Розділ охорони праці містить загальну інформацію та вимоги до техніки безпеки оператора КТ. Економічний розділ проекту містить розрахунок витрат на НДР та реалізацію проекту.

в) оцінка якості виконання пояснювальної записки та графічної частини дипломного проекту

Графічна частина складається з 14 слайдів мультимедійної презентації, виконаної у програмному продукті MS PowerPoint, які містять ілюстративні схеми, скріншоти роботи програмного застосунку, передбачені технічним завданням. Пояснювальна записка виконана акуратно та у відповідності до норм. Якість виконання графічної частини проекту та пояснювальної записки добра, розробку виконано у повному обсязі.

г) перелік позитивних якостей дипломного проекту *Використання Python, Telegram API, MongoDB відповідає сучасним інженерним практикам.*

Реалізовано зв'язок із зовнішнім API для генерації або перевірки запитань, що розширює функціонал бота.

д) основні недоліки дипломного проекту *UX/UI аспект може потребувати доопрацювання для підвищення зручності використання. Немає чітко описаних юніт- або інтеграційних тестів, що важливо для системи з користувацьким інтерфейсом. Хоча в документі згадується тестування роботи бота, аналіз можливих помилок або обговорення труднощів, що виникали під час експлуатації, не розкритий*

Оцінка розрахункової частини *відмінно*

Оцінка графічної частини *відмінно*

Загальна оцінка *відмінно*

Прізвище, ім'я, по батькові рецензента *к.т.н. Рудніченко Микола Дмитрович*

Місце роботи і посада рецензента *Національний університет «Одеська політехніка», доцент кафедри інформаційних технологій*

Підпис

« *20* » *червня* 2025 р.

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

ВІДГУК

керівника на дипломний проєкт здобувача (здобувачки) освіти

відділення комп'ютерних систем

Стайкуца Максим Сергійович

(прізвище, ім`я та по батькові)

Спеціальність: *121 " Інженерія програмного забезпечення"*

Освітньо-професійна програма: *«Розробка програмного забезпечення»*

Тема дипломного проєкту: *Розробка телеграм боту для перевірки знань з англійської мови на базі ШІ*

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЄКТУ

а) обсяг і якість виконання проєкту (графічного матеріалу і розрахунково-пояснювальної записки) *Дипломний проєкт виконано відповідно технічному завданню. Пояснювальна записка містить 84 сторінки. У пояснювальній записці виконано опис етапів розробки програмного забезпечення для телеграм боту для перевірки знань з англійської мови , а також його програмного забезпечення. Графічна частина складається з 14 слайдів мультимедійної презентації, які також містять схеми і скріншоти, передбачені технічним завданням. Якість виконання пояснювальної записки та графічної частини добра, розробку виконано в повному обсязі.*

б) самостійність роботи над проєктом: *Протягом всього строку дипломного проєктування та переддипломної практики здобувач освіти Стайкуца М.С. поступово та послідовно виконував всі етапи розробки. Всі роботи здобувач освіти виконував самостійно, з оглядом на рекомендації керівника*

в) теоретична підготовка випускника (випускниці): *Здобувач освіти Стайкуца М.С. під час роботи над дипломним проєктом вивчив достатню кількість літературних джерел та матеріалів за даною тематикою.*

Вважаю, що теоретична підготовка дипломника добра і він готовий до захисту дипломного проєкту

г) вміння розв'язувати виробничі та конструкторські питання _____

Під час дипломного проєктування здобувач освіти мав змогу самостійно приймати рішення з реалізації алгоритмів і елемнетів чат-боту, що розробляв, та показав вміння організовано працювати над поставленим завданням, складати схеми та проводити розробку коду за допомогою актуальних для теми комп'ютерних програмних засобів.

Оцінка розрахункової частини *Відмінно*

Оцінка графічної частини *Добре*

Загальна оцінка *Відмінно*

Прізвище, ім'я, по батькові керівника дипломного проєкту

Закроєв Юрій Михайлович

Місце роботи і посада керівника дипломного проєкту

Директор ТОВ «Біг ВОШ»

Підпис _____

«16» 06 2025 р.

ДОЗВІЛ
НА РОЗМІЩЕННЯ
ВИПУСКНОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
(ДИПЛОМНОГО ПРОЕКТУ)
В ЕЛЕКТРОННОМУ РЕПОЗИТАРІЇ ВСП «ОТФК ОНТУ»

Ми, що нижче підписалися,

Стайкуца Максим Сергійович,
здобувач освіти гр. 4РП-08, та

Закроєв Юрій Михайлович,
керівник дипломного проекту,

не заперечуємо щодо розміщення електронного варіанту пояснювальної записки до дипломного проекту фахового молодшого бакалавра на тему:

«Розробка телеграм боту для перевірки знань з англійської мови на базі ШІ» (автор роботи – Стайкуца М.С., керівник роботи – Закроєв Ю.М.)

виконаного у ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету» в 2025 році, у повному обсязі в електронному репозитарії ВСП «ОТФК ОНТУ» для вільного доступу через мережу Інтернет.

Несемо відповідальність за ідентичність електронного та друкованого варіантів випускної кваліфікаційної роботи і даємо згоду на обробку персональних даних.

Виконавець _____ / Стайкуца М.С. /

Керівник _____ / Закроєв Ю.М. /

«16» червня 2025 р.

ДОВІДКА

циклової комісії КТ та ПІ
про допуск до захисту дипломного проєкту
здобувача (здобувачки) освіти IV курсу
відділення комп'ютерних систем групи 4РП-08

Стайкуци Максима Сергійовича

на тему *Розробка чат-боту для перевірки знань з англійської мови на базі ШІ*

Висновок відповідальної особи за проведення нормоконтролю:
пояснювальна записка до дипломного проєкту виконана з деякими порушеннями ДСТУ та оформлена відповідно до вимог Положення про дипломне проєктування

(підпис) *16.06.2025* (дата) *Петрашова В.І.* (П.І.Б.)

Висновок відповідальної особи за перевірку роботи на наявність академічного плагіату *згідно звіту про перевірку від 15.06.2025 р. значення коефіцієнту подібності в роботі становить 16,73%, коефіцієнт цитування – 3,50%.*

(підпис) *16.06.2025* (дата) *Краснокутська К.Г.* (П.І.Б.)

Попередня експертиза (малий захист) дипломного проєкту

здобувача (здобувачки) освіти *Стайкуци М.С.* (П.І.Б.)

проведена «*16*» *червня* 2025 р.

Висновки *Пояснювальна записка до дипломного проєкту виконана у повному обсязі. Випускна кваліфікаційна робота (дипломний проєкт) відповідає вимогам Положення про дипломне проєктування та рекомендована до захисту.*

Голова ЦК КТ та ПІ (підпис) *Кривченко Ю.В.* (П.І.Б.)

StrikePlagiarism.com

Дата звіту 6/15/2025
Дата редагування 6/15/2025

Документ прийнятий

Звіт подібності

метадані

Назва організації
Odesa Technical Professional College of Odesa National University of Technology
Заголовок
Розробка чат-боту для перевірки знань з англійської мови на базі ШІ

Автор
Стайкуца Максим Сергійович
Науковий керівник / Експерт
Закроєв Юрій Михайлович

підрозділ
Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету"

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

КП 1

КЦ

25
Довжина фрази для коефіцієнта подібності 2

16924
Кількість слів

137554
Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв	ß	22
Інтервали	A→	0
Мікропробіли	·	1
Білі знаки	ß	0
Парафрази (SmartMarks)	a	87

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз

Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Розробка чат-бот системи з навчання кібергігієні [redacted] 5/25/2025 State University of Intellectual Technologies and Communications (Кафедра кібербезпеки та технічного захисту інформації)	199 1.18 %
2	Розробка чат-бот системи з навчання кібергігієні [redacted] 5/25/2025 State University of Intellectual Technologies and Communications (Кафедра кібербезпеки та технічного захисту інформації)	138 0.82 %

3	Розробка чат-бот системи з навчання кібергігієні [redacted] 5/25/2025 State University of Intellectual Technologies and Communications (Кафедра кібербезпеки та технічного захисту інформації)	131 0.77 %
4	Розробка чат-бот системи з навчання кібергігієні [redacted] 5/25/2025 State University of Intellectual Technologies and Communications (Кафедра кібербезпеки та технічного захисту інформації)	92 0.54 %
5	Розробка чат-бот системи з навчання кібергігієні [redacted] 5/25/2025 State University of Intellectual Technologies and Communications (Кафедра кібербезпеки та технічного захисту інформації)	84 0.50 %
6	https://card-file.ontu.edu.ua/server/api/core/bitstreams/a141b658-5fa7-4f90-b0bd-7f0ccaed21e5/content	77 0.45 %
7	https://card-file.ontu.edu.ua/bitstreams/34a6756b-592f-4b77-a805-183aa03a6a26/download	61 0.36 %
8	Телеграм бот для заповнення шаблонів [redacted] 3/16/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	50 0.30 %
9	https://card-file.ontu.edu.ua/server/api/core/bitstreams/a141b658-5fa7-4f90-b0bd-7f0ccaed21e5/content	47 0.28 %
10	https://card-file.ontu.edu.ua/server/api/core/bitstreams/a141b658-5fa7-4f90-b0bd-7f0ccaed21e5/content	47 0.28 %

з домашньої бази даних (0.57 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Розробка 3D-гри у жанрі survival-horror з налаштуваннями рівнів складності [redacted] 6/12/2025 **Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")**	**92 (7) 0.54 %**
2	Розробка web-застосунку для генерації повідомлень із використанням технологій штучного інтелекту [redacted] 6/14/2025 Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")	5 (1) 0.03 %

з програми обміну базами даних (5.80 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Розробка чат-бот системи з навчання кібергігієні [redacted] 5/25/2025 **State University of Intellectual Technologies and Communications (Кафедра кібербезпеки та технічного захисту інформації)**	**690 (9) 4.08 %**
2	Телеграм бот для заповнення шаблонів [redacted] 3/16/2025 **National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)**	**258 (15) 1.52 %**
3	ВИКОРИСТАННЯ МОЖЛИВОСТЕЙ ШТУЧНОГО ІНТЕЛЕКТУ ПЛАТФОРМИ CLASSTIME ДЛЯ ОЦІНЮВАННЯ РЕЗУЛЬТАТІВ НАВЧАННЯ [redacted] 4/1/2024 Hlukhiv National Pedagogical University of Oleksandr Dovzhenko (Hlukhiv National Pedagogical University of Oleksandr Dovzhenko)	15 (1) 0.09 %

4	Веб-застосунок для організації роботи IT-підприємств в умовах надзвичайного стану [redacted] 3/16/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	10 (1) 0.06 %
5	ФКПІ_ІПЗ_2023_121_Дараселія_Д.В [redacted] 7/11/2024 Ukrainian national aviation university (Ukrainian national aviation university)	8 (1) 0.05 %

з Інтернету (10.36 %)

ПОРЯДКОВИЙ НОМЕР	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://card-file.ontu.edu.ua/server/api/core/bitstreams/44c16132-5f53-48e2-b6c0-61e9a2f0fd75/content	634 (52) 3.75 %
2	**https://card-file.ontu.edu.ua/server/api/core/bitstreams/a141b658-5fa7-4f90-b0bd-7f0ccaed21e5/content**	**518 (22) 3.06 %**
3	https://card-file.ontu.edu.ua/bitstreams/1dff552d-7200-49b8-ae1d-ba76a1335685/download	191 (17) 1.13 %
4	**https://card-file.ontu.edu.ua/bitstreams/34a6756b-592f-4b77-a805-183aa03a6a26/download**	**87 (2) 0.51 %**
5	**https://card-file.ontu.edu.ua/bitstreams/29489599-0581-4ce6-8890-c3b13d9f2e0e/download**	**55 (3) 0.32 %**
6	https://card-file.ontu.edu.ua/bitstreams/549ee9fe-7574-4ae5-b500-9fe2711f33e6/download	49 (4) 0.29 %
7	https://card-file.ontu.edu.ua/bitstreams/bbed74c8-2ea7-44c5-8d00-0fe3fd9790ee/download	40 (4) 0.24 %
8	https://card-file.ontu.edu.ua/bitstreams/035f6436-20b4-4ee6-8e99-bede670e308b/download	32 (2) 0.19 %
9	https://card-file.ontu.edu.ua/bitstreams/82a6d375-2b69-4233-b80f-fbfd149b7747/download	30 (2) 0.18 %
10	https://card-file.ontu.edu.ua/bitstreams/5240e379-7721-49f0-8ee8-27140b0b473a/download	21 (1) 0.12 %
11	https://card-file.ontu.edu.ua/bitstreams/53ed22ad-8700-4162-b97a-082a1ad472d6/download	17 (2) 0.10 %
12	https://card-file.ontu.edu.ua/bitstreams/6cf43324-8f08-4031-ba42-f80b18efbbc8/download	15 (2) 0.09 %
13	https://github.com/ViTAE-Transformer/ViTPose/issues/151	14 (2) 0.08 %
14	https://cambridge.ua/uk/blog/resources-for-learning-english/	14 (2) 0.08 %
15	https://andrewlock.net/exploring-the-net-core-docker-files-dotnet-vs-aspnetcore-vs-aspnetcore-build/	11 (1) 0.06 %
16	https://card-file.ontu.edu.ua/server/api/core/bitstreams/c63b91ba-d04f-4715-890d-b16277695c7e/content	8 (1) 0.05 %
17	https://www.essuir.sumdu.edu.ua/bitstream/123456789/95577/1/Indyk_bak_rob.pdf	6 (1) 0.04 %
18	https://card-file.ontu.edu.ua/server/api/core/bitstreams/ead3fa83-2e3d-4cd7-bfbd-1d5ed04c1ce4/content	6 (1) 0.04 %
19	https://card-file.ontu.edu.ua/bitstreams/8f088d70-9465-490c-8fa6-2eb74516c620/download	6 (1) 0.04 %

Список прийнятих фрагментів (немає прийнятих фрагментів)

ПОРЯДКОВИЙ НОМЕР	ЗМІСТ	КІЛЬКІСТЬ ОДНАКОВИХ СЛІВ (ФРАГМЕНТІВ)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 121 «Інженерія програмного забезпечення»_
Освітньо-професійна програма: **«Розробка програмного забезпечення»**
Група: 4РП- 08