ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА
на тему: «ТЕЛЕГРАМ-БОТ ДЛЯ ВИВЧЕННЯ ІНОЗЕМНИХ МОВ НА ОСНОВІ PYTHON»

на здобуття освітнього ступеня бакалавр
зі спеціальності 126 Інформаційні системи та технології
(код, найменування спеціальності)
освітньо-професійної програми Інформаційні системи та технології
(назва)

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

_____ Владислав ЛУК'ЯНОВ
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав: здобувач вищої освіти гр. ІСД-42
Владислав ЛУК'ЯНОВ
Ім'я, ПРІЗВИЩЕ

Керівник:
доцент кафедри Валентина ДАНИЛЬЧЕНКО
Ім'я, ПРІЗВИЩЕ

Рецензент:
_____
Ім'я, ПРІЗВИЩЕ

Київ 2026

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра Інформаційних систем та технологій

Ступінь вищої освіти Бакалавр

Спеціальність 126 Інформаційні системи та технології

Освітньо-професійна програма Інформаційні системи та технології

ЗАТВЕРДЖУЮ
Завідувач кафедрою ІСТ
_____ Каміла СТОРЧАК
«____» _____ 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Владислав Юрійович Лук'янов
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Телеграм-бот для вивчення іноземних мов на основі Python

керівник кваліфікаційної роботи Валентина ДАНИЛЬЧЕНКО, доцент кафедри

(Ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)

затверджені наказом Державного університету інформаційно-комунікаційних технологій

від «20» лютого 2026 р. №51

2. Строк подання кваліфікаційної роботи «1» червня 2026 р.

3. Вихідні дані до кваліфікаційної роботи:

Інформаційні ресурси системи включають навчальні словники, бази тестових завдань, тематичні набори слів та фраз, дані користувачів і результати проходження тестів;
Програмно-технічні компоненти охоплюють модулі обробки повідомлень у Telegram, механізми взаємодії з Telegram Bot API та бібліотеки мови Python;
Методична та нормативна база містить матеріали щодо використання ІТ в освіті, підходи до дистанційного навчання та застосування чат-ботів у навчальному процесі;

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

1. Аналіз предметної області та постановка задачі створення Telegram-бота
2. Огляд існуючих програмних рішень і сервісів.
3. Розробка та тестування Telegram-бота для організації навчального процесу.

5. Перелік ілюстративного матеріалу: *презентація*

6. Дата видачі завдання «20» лютого 2026

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітки
1	Аналіз предметної області та огляд сучасних систем і сервісів для вивчення іноземних мов з використанням чат-ботів	15.04.2026 р.	Виконано
2	Вивчення технічної документації Telegram Bot API та бібліотек Python для розробки чат-ботів	20.04.2026 р.	Виконано
3	Постановка задачі, визначення функціональних вимог до системи та формування технічного завдання	25.04.2026 р.	Виконано
4	Розробка архітектури програмної системи: серверна частина (Python), взаємодія з Telegram API, структура бази даних	05.05.2026 р.	Виконано
5	Програмна реалізація Telegram-бота на мові Python та створення модулів обробки повідомлень	15.05.2026 р.	Виконано
6	Реалізація системи тестування знань користувачів, налаштування параметрів навчання (мова, тема, рівень складності)	25.05.2026 р.	Виконано
7	Подання роботи в деканат	01.06.2026 р.	Виконано

Здобувач вищої освіти _____ Владислав ЛУК'ЯНОВ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Валентина ДАНИЛЬЧЕНКО
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 55 стор., 13 рис., 3 табл., 18 джерел.

Об'єкт дослідження – система автоматизованого вивчення іноземних мов із використанням чат-ботів та месенджер-платформ.

Предмет дослідження – методи розробки та використання Telegram-ботів для організації інтерактивного навчального процесу, тестування знань та персоналізації навчання користувачів.

Мета роботи – розробка програмного застосунку у вигляді Telegram-бота, що забезпечує інтерактивне вивчення іноземних мов, проведення тестів, налаштування рівня складності та тем навчання.

Методи дослідження:

1. Аналіз літературних та технічних джерел – дослідження сучасних підходів до використання чат-ботів у сфері онлайн-освіти та мовного навчання.
2. Методи системного аналізу та проєктування – моделювання структури Telegram-бота, визначення основних функціональних модулів та алгоритмів взаємодії з користувачем.
3. Методи конфігурації та програмної реалізації – розробка програмного забезпечення на мові Python із використанням Telegram Bot API та бібліотек для обробки повідомлень.
4. Експериментальні методи – тестування роботи Telegram-бота, перевірка коректності функціонування тестових модулів, налаштувань рівня складності та системи нагадувань.

TELEGRAM-БОТ, PYTHON, ІНОЗЕМНІ МОВИ, ЧАТ-БОТ, TELEGRAM API, ОНЛАЙН НАВЧАННЯ, ІНТЕРАКТИВНЕ ТЕСТУВАННЯ, ОСВІТНІ ТЕХНОЛОГІЇ

ЗМІСТ

ВСТУП

Актуальність дослідження. У сучасному світі знання іноземних мов є важливим фактором професійного розвитку, міжнародної комунікації та доступу до глобальних інформаційних ресурсів. Активна інтеграція інформаційних технологій у різні сфери життя, зокрема в освіту, сприяє появі нових підходів до організації навчального процесу. Сьогодні дедалі більшої популярності набувають цифрові платформи, мобільні додатки та онлайн-сервіси, що дозволяють користувачам самостійно вивчати іноземні мови у зручний для них час і темп.

Чат-боти забезпечують автоматизовану взаємодію з користувачем, дозволяють організувати інтерактивний навчальний процес, проводити тестування знань, надавати рекомендації та контролювати прогрес навчання. Використання чат-ботів у навчанні іноземних мов дає можливість створити доступний та зручний інструмент для постійної практики мовних навичок без необхідності встановлення спеціалізованого програмного забезпечення.

Серед сучасних месенджерів особливе місце займає платформа Telegram, яка надає розширені можливості для створення ботів за допомогою Telegram Bot API. Відкритість платформи, простота інтеграції та велика кількість користувачів роблять її ефективним середовищем для реалізації освітніх сервісів. Використання мови програмування Python для розробки Telegram-ботів дозволяє створювати функціональні та масштабовані програмні рішення завдяки наявності великої кількості бібліотек та інструментів для роботи з мережевими сервісами.

Розробка Telegram-бота для вивчення іноземних мов на основі технології програмування Python є актуальною задачею, оскільки дозволяє створити інтерактивний інструмент навчання, який поєднує можливості сучасних месенджерів та програмних технологій для підвищення ефективності процесу засвоєння мовних знань.

Об’єкт дослідження – система автоматизованого вивчення іноземних мов із використанням чат-ботів та месенджер-платформ.

Предмет дослідження – методи розробки та використання Telegram-ботів для організації інтерактивного навчального процесу, тестування знань та персоналізації навчання користувачів.

Мета роботи – розробка програмного застосунку у вигляді Telegram-бота, що забезпечує інтерактивне вивчення іноземних мов, проведення тестів, налаштування рівня складності та тем навчання.

Наукові завдання:

1. проаналізувати сучасні підходи до використання цифрових технологій у процесі вивчення іноземних мов;
2. дослідити можливості використання чат-ботів як інструменту автоматизації навчального процесу;
3. вивчити функціональні можливості платформи Telegram та її інтерфейс програмування Telegram Bot API для створення освітніх сервісів;
4. сформувати функціональні та нефункціональні вимоги до програмного застосунку для вивчення іноземних мов;
5. розробити архітектуру програмної системи Telegram-бота та визначити структуру основних програмних модулів;
6. реалізувати Telegram-бота на мові програмування Python із використанням відповідних бібліотек для взаємодії з Telegram API;
7. реалізувати систему тестування знань користувачів та механізм перевірки правильності відповідей;
8. розробити конфігураційні параметри навчального процесу, що дозволяють обирати мову навчання, тематику та рівень складності;
9. провести тестування розробленого програмного застосунку та оцінити ефективність його функціонування.

Методи дослідження:

1. аналіз літературних та технічних джерел – для вивчення сучасних підходів до використання чат-ботів у сфері онлайн-освіти та методів вивчення іноземних мов із застосуванням цифрових технологій;
2. методи системного аналізу – для дослідження структури системи

навчання за допомогою чат-бота та визначення її основних компонентів;

3. методи проєктування програмних систем – для розробки архітектури Telegram-бота та визначення взаємодії між його функціональними модулями;

4. методи програмної реалізації – для створення програмного застосунку на мові програмування Python із використанням бібліотек для роботи з Telegram Bot API;

5. експериментальні методи – для тестування роботи Telegram-бота, перевірки коректності функціонування модулів тестування та налаштування параметрів навчання;

6. методи аналізу результатів – для оцінювання ефективності роботи системи та визначення можливостей її подальшого вдосконалення.

Розроблений програмний застосунок реалізує можливість взаємодії користувача з ботом через месенджер Telegram, що дозволяє отримувати навчальні матеріали, виконувати тестові завдання та отримувати миттєвий зворотний зв'язок щодо правильності відповідей. Система демонструє можливість створення масштабованого освітнього рішення, яке може використовуватися для вивчення різних іноземних мов, а також для організації дистанційного навчання.

Наукова новизна одержаних результатів. Удосконалено архітектуру систем мікронавчання у мессенджерах. Запропоновано модульну модель взаємодії на базі асинхронного Python-фреймворку, що оптимізує доставку контенту та роботу з базою даних.

Практична значущість одержаних результатів. Розроблено готовий до використання Telegram-бот для вивчення мов. Гнучка структура бази даних дозволяє легко масштабувати систему та додавати нові модулі без зміни коду.

Апробація результатів та публікації. Основні положення і результати бакалаврської роботи доповідались на науково практичних конференціях, що проходили на базі Державного університету інформаційно-комунікаційних технологій.

1. III всеукраїнська науково-технічна конференція «Технологічні горизонти: дослідження та застосування інформаційних технологій для технологічного

прогресу України і світу»(Київ, ДУІКТ, 18 листопада 2025 р.) Тези доповіді на тему « Гейміфікація як засіб підвищення мотивації учнів та студентів» опубліковані у збірнику матеріалів конференції(стор.300).

2. VII Науково-технічна конференція «Сучасний стан та перспективи розвитку IoT» »(Київ, ДУІКТ, 20 квітня 2026 р.) Тези доповіді на тему «Застосування чат-бот технологій у мовній освіті: розробка telegram-бота для вивчення іноземних мов засобами python» опубліковані у збірнику матеріалів конференції(стор.47).

1 ТЕОРЕТИЧНІ ОСНОВИ ВИКОРИСТАННЯ ЧАТ-БОТІВ У НАВЧАННІ ІНОЗЕМНИХ МОВ

1.1 Сучасні підходи до вивчення іноземних мов за допомогою цифрових технологій

У сучасному світі знання іноземних мов є складовою професійного розвитку, міжнародної комунікації та доступу до глобальних інформаційних ресурсів. У зв'язку з активним розвитком інформаційних технологій підходи до організації навчального процесу суттєво змінюються. Традиційні методи навчання поступово доповнюються цифровими інструментами, онлайн-сервісами та мобільними застосунками, що дозволяє підвищити ефективність засвоєння навчального матеріалу та зробити процес навчання більш доступним і гнучким.

Традиційне навчання іноземних мов передбачає проведення занять у навчальних закладах під керівництвом викладача. Такий підхід базується на використанні підручників, навчальних посібників, аудіоматеріалів та безпосередньому спілкуванні між викладачем і студентами. Основною перевагою традиційного навчання є можливість отримання негайного зворотного зв'язку від викладача, а також організація живого спілкування між учасниками навчального процесу. Однак такий формат має певні обмеження, зокрема прив'язку до конкретного часу та місця проведення занять, що може ускладнювати доступ до навчання для окремих категорій користувачів [1].

З розвитком мережевих технологій широкого поширення набуло онлайн-навчання, яке дозволяє організувати освітній процес за допомогою мережі Інтернет. Онлайн-платформи забезпечують доступ до навчальних матеріалів, відеолекцій, інтерактивних вправ та тестових завдань. Користувачі мають можливість навчатися у зручний для себе час, що значно підвищує гнучкість освітнього процесу. Крім того, онлайн-навчання сприяє розвитку дистанційної освіти, яка дозволяє залучати до навчального процесу студентів з різних регіонів та

країн.

Досить часто використовують мобільні додатки, які забезпечують можливість навчання безпосередньо за допомогою смартфонів та планшетів. Мобільні застосунки дозволяють користувачам виконувати вправи, проходити тести, вивчати нові слова та фрази у будь-який зручний час. Завдяки інтерактивним елементам, гейміфікації та адаптивному підходу до навчання мобільні додатки роблять процес вивчення мов більш цікавим і мотивуючим.

Одним із найбільш поширених підходів до організації дистанційної освіти є e-learning. Даний підхід передбачає використання електронних освітніх ресурсів, навчальних платформ та систем управління навчанням (Learning Management Systems) [2]. У межах e-learning користувачі отримують доступ до електронних курсів, тестових систем, відеоматеріалів та інтерактивних завдань. Такий формат навчання дозволяє систематизувати освітній процес, забезпечити контроль знань та відстежувати прогрес користувачів.

Microlearning передбачає подання навчального матеріалу невеликими порціями. Такий формат навчання дозволяє користувачам швидко засвоювати нову інформацію та ефективно повторювати вже вивчений матеріал. Microlearning часто використовується у мобільних додатках та онлайн-сервісах для вивчення мов, де навчальні модулі представлені у вигляді коротких вправ, тестів або інтерактивних завдань. Цей підхід сприяє підвищенню концентрації уваги користувачів та забезпечує більш ефективне засвоєння навчального матеріалу.

Інформаційні технології суттєво розширюють можливості вивчення іноземних мов. Поєднання традиційних методів навчання з цифровими технологіями, онлайн-сервісами, мобільними додатками та чат-ботами дозволяє створювати нові ефективні освітні інструменти, що забезпечують доступність та гнучкість навчального процесу.

Існує велика кількість цифрових сервісів, що дозволяють користувачам вивчати іноземні мови за допомогою мобільних застосунків та онлайн-платформ. Такі системи поєднують інтерактивні вправи, тести, аудіоматеріали та елементи гейміфікації, що робить процес навчання більш ефективним та цікавим для

користувачів [3]. Найбільш популярними сервісами для вивчення іноземних мов є Duolingo, Babbel, Busuu та Memrise (рис. 1.1).

Рис. 1.1 – Популярні цифрові платформи для вивчення іноземних мов

Сервіс Duolingo є одним із найпоширеніших мобільних застосунків для вивчення іноземних мов. Платформа використовує ігровий підхід до навчання, де користувачі виконують короткі вправи, отримують бали та відкривають нові рівні. Завдання включають переклад слів, побудову речень, аудіювання та повторення лексики [14]. Основною перевагою Duolingo є простота використання, велика кількість доступних мов та можливість навчатися безкоштовно.

Платформа Babbel орієнтована на більш структуроване навчання та пропонує курси, розроблені лінгвістами. Уроки в Babbel побудовані таким чином, щоб користувач поступово засвоював граматику, лексику та правила побудови речень [16]. Система використовує інтерактивні вправи, аудіоматеріали та діалоги, що дозволяє покращувати навички спілкування іноземною мовою.

Сервіс Busuu також є популярною платформою для вивчення мов, яка поєднує мобільний додаток та соціальну взаємодію між користувачами. Однією з особливостей Busuu є можливість отримання зворотного зв’язку від носіїв мови, що дозволяє покращити навички письма та вимови. Крім того, сервіс пропонує

структуровані курси, тестування знань та систему відстеження прогресу користувача.

Платформа Memrise використовує методику повторення та запам’ятовування нових слів за допомогою спеціальних алгоритмів. Основна увага в Memrise приділяється розширенню словникового запасу та розвитку навичок сприйняття мови на слух [17]. Система пропонує короткі інтерактивні вправи, відеоматеріали з носіями мови та різноманітні тренувальні завдання.

Порівняння популярних сервісів для вивчення іноземних мов наведено в табл. 1.1.

Таблиця 1.1 – Порівняння популярних сервісів для вивчення іноземних мов

Сервіс	Основні можливості	Переваги	Недоліки
Duolingo	Вправи, тести, переклад слів, гейміфікація	Безкоштовний доступ, простий інтерфейс	Обмежене пояснення граматики
Babbel	Структуровані уроки, аудіоматеріали, діалоги	Чітка система навчання	Переважно платний доступ
Busuu	Курси, перевірка завдань носіями мови, тестування	Соціальна взаємодія користувачів	Частина функцій доступна лише у преміум
Memrise	Вивчення лексики, відео з носіями мови, повторення слів	Ефективне запам’ятовування нових слів	Менше уваги граматиці

Цифрові платформи значно розширюють можливості вивчення іноземних мов та дозволяють користувачам навчатися у зручний для них час. Разом з тим більшість таких сервісів реалізовані у вигляді мобільних застосунків або веб-платформ. Використання чат-ботів у месенджерах може стати альтернативним підходом до організації навчального процесу, що забезпечує швидку взаємодію з користувачем та доступність навчальних матеріалів у повсякденному середовищі спілкування.

1.2 Чат-боти як інструмент автоматизації навчального процесу

Розвиток інформаційних технологій сприяє появі нових інструментів автоматизації різних процесів, зокрема у сфері освіти. Одним із таких інструментів є чат-боти, які дозволяють організувати взаємодію між користувачем та програмною системою за допомогою текстових або голосових повідомлень. Чат-боти широко використовуються у різних галузях, включаючи електронну комерцію, банківську сферу, технічну підтримку та освіту.

Чат-бот — це програмний застосунок, який імітує спілкування з користувачем у режимі діалогу. Така система може відповідати на запити користувача, надавати інформацію, виконувати певні команди або допомагати у вирішенні конкретних задач. Основною особливістю чат-ботів є можливість автоматизації взаємодії з користувачем без необхідності постійної участі людини.

У сфері освіти чат-боти можуть використовуватися для організації навчального процесу, проведення тестування, надання навчальних матеріалів та контролю знань користувачів. Використання чат-ботів у навчанні дозволяє створити інтерактивне середовище, у якому користувач може отримувати інформацію, виконувати завдання та отримувати миттєвий зворотний зв’язок. Особливо ефективним є застосування чат-ботів у процесі вивчення іноземних мов, оскільки вони дозволяють регулярно практикувати лексику, граматику та навички перекладу [4].

Залежно від принципів роботи та технологій реалізації чат-боти можна поділити на декілька основних типів. Найбільш поширеними є rule-based чат-боти та боти, що використовують технології штучного інтелекту (рис. 1.2).

Рис. 1.2 – Класифікація чат-ботів за принципом роботи

Rule-based чат-боти працюють на основі заздалегідь визначених правил та сценаріїв взаємодії з користувачем. Такі системи використовують набір команд, ключових слів або шаблонів повідомлень, на які бот реагує певним чином. Наприклад, користувач може вводити команду для запуску тесту або вибору теми навчання, після чого бот пропонує відповідні варіанти завдань [5]. Основною перевагою rule-based чат-ботів є простота реалізації та передбачуваність роботи. Вони не потребують складних алгоритмів обробки природної мови та можуть бути реалізовані за допомогою стандартних програмних бібліотек.

Іншим типом є чат-боти, що використовують технології штучного інтелекту (AI-боти). Такі системи застосовують алгоритми обробки природної мови (Natural Language Processing), машинне навчання та нейронні мережі для аналізу повідомлень користувача. AI-боти здатні розуміти зміст запитів, аналізувати

контекст діалогу та формувати більш гнучкі відповіді. Вони можуть підтримувати складніші діалоги та адаптуватися до поведінки користувача. Однак розробка таких систем є більш складною та потребує значних обчислювальних ресурсів.

У навчальних системах часто використовуються саме rule-based чат-боти, оскільки вони дозволяють реалізувати чітко структуровані сценарії взаємодії з користувачем. Наприклад, бот може пропонувати вибір мови навчання, теми уроку або рівня складності завдань, після чого генерувати тестові питання та перевіряти правильність відповідей [6]. Такий підхід є достатньо ефективним для організації навчальних курсів та систем тестування знань.

Чат-боти активно використовуються у різних сферах діяльності, оскільки дозволяють автоматизувати процес взаємодії з користувачами та значно спрощують обробку запитів. Завдяки можливості працювати у популярних месенджерах та веб-сервісах, чат-боти можуть виконувати широкий спектр функцій у різних галузях (рис. 1.3).

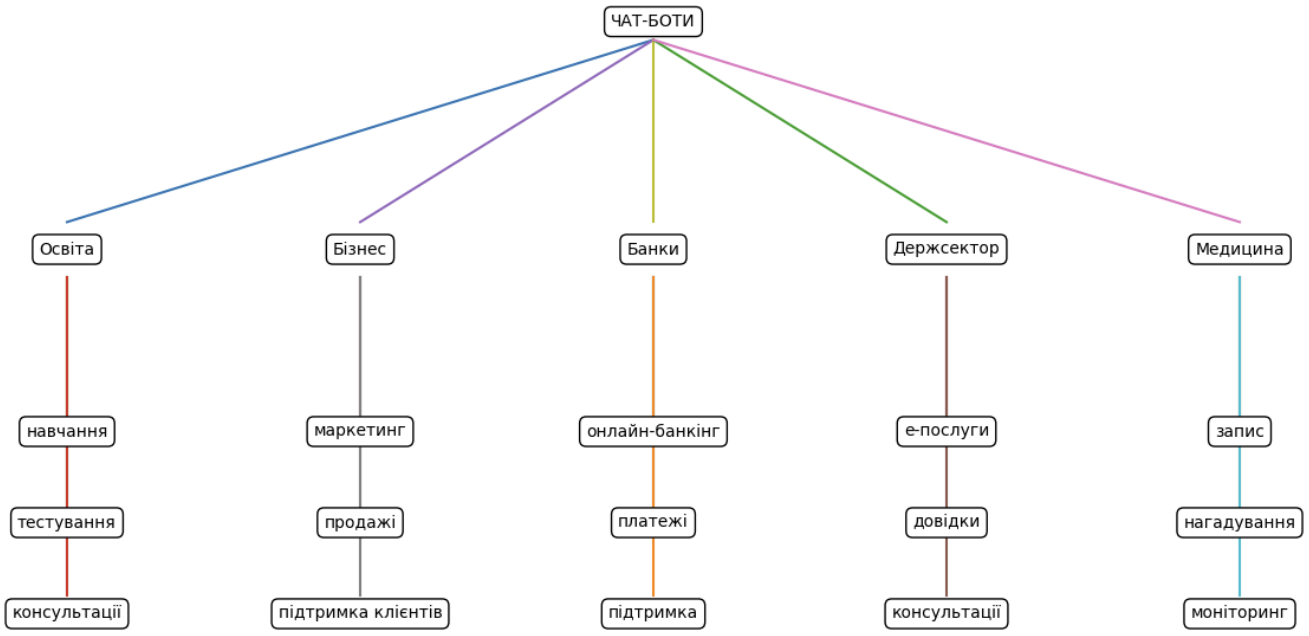

Рис. 1.3 – Сфери застосування чат-ботів

У навчальному процесі чат-боти можуть використовуватися для надання навчальних матеріалів, організації тестування, перевірки знань студентів та надання рекомендацій щодо подальшого навчання. Крім того, вони можуть

виконувати роль інтерактивних помічників, що допомагають користувачам повторювати навчальний матеріал, вивчати нові слова або виконувати вправи з іноземної мови.

У банківській сфері чат-боти використовуються для автоматизації обслуговування клієнтів. Вони можуть надавати інформацію про банківські послуги, перевіряти баланс рахунку, повідомляти про останні операції або допомагати клієнтам виконувати певні фінансові операції. Використання чат-ботів дозволяє значно зменшити навантаження на операторів служби підтримки.

Технічна підтримка, чат-боти можуть відповідати на типові запитання користувачів, допомагати у вирішенні технічних проблем, надавати інструкції з використання програмного забезпечення або спрямовувати користувача до відповідного спеціаліста. Це дозволяє значно пришвидшити процес обробки звернень та підвищити якість обслуговування.

У сфері маркетингу чат-боти використовуються для взаємодії з клієнтами, інформування про нові продукти та послуги, проведення рекламних кампаній та збору відгуків користувачів. Завдяки автоматизованому спілкуванню компанії можуть підтримувати постійний контакт з клієнтами та швидко реагувати на їхні запити.

Використання чат-ботів має низку важливих переваг. Однією з основних є доступність 24/7, що дозволяє користувачам отримувати необхідну інформацію або виконувати певні дії у будь-який час без необхідності очікування відповіді оператора. Автоматизація процесів, яка дозволяє значно скоротити витрати часу та ресурсів на обробку запитів користувачів. Чат-боти можуть одночасно обслуговувати велику кількість користувачів, що підвищує ефективність роботи інформаційних систем [7]. Система може враховувати попередні дії користувача, рівень його підготовки або індивідуальні налаштування. У навчальних системах це дозволяє адаптувати навчальний процес до потреб користувача, що підвищує ефективність засвоєння навчального матеріалу.

Чат-боти є ефективним інструментом автоматизації взаємодії з користувачами у різних сферах діяльності. Їх використання у навчальних системах,

зокрема для вивчення іноземних мов, відкриває нові можливості для створення інтерактивних та доступних освітніх сервісів.

1.3 Можливості платформи Telegram для створення освітніх ботів

Сучасні месенджери активно використовуються не лише для спілкування, але й для створення різноманітних інформаційних сервісів та автоматизованих систем. Однією з найбільш популярних платформ для реалізації чат-ботів є месенджер Telegram, який надає широкі можливості для створення програмних застосунків завдяки відкритому інтерфейсу програмування Telegram Bot API.

Telegram Bot API — це спеціальний програмний інтерфейс, який дозволяє розробникам створювати ботів та керувати їхньою взаємодією з користувачами. За допомогою цього API бот може отримувати повідомлення від користувачів, обробляти їх та надсилати відповідні відповіді [8]. Робота Telegram-бота базується на взаємодії між сервером Telegram, серверною частиною програми та користувачем, який спілкується з ботом через інтерфейс месенджера (рис. 1.4).

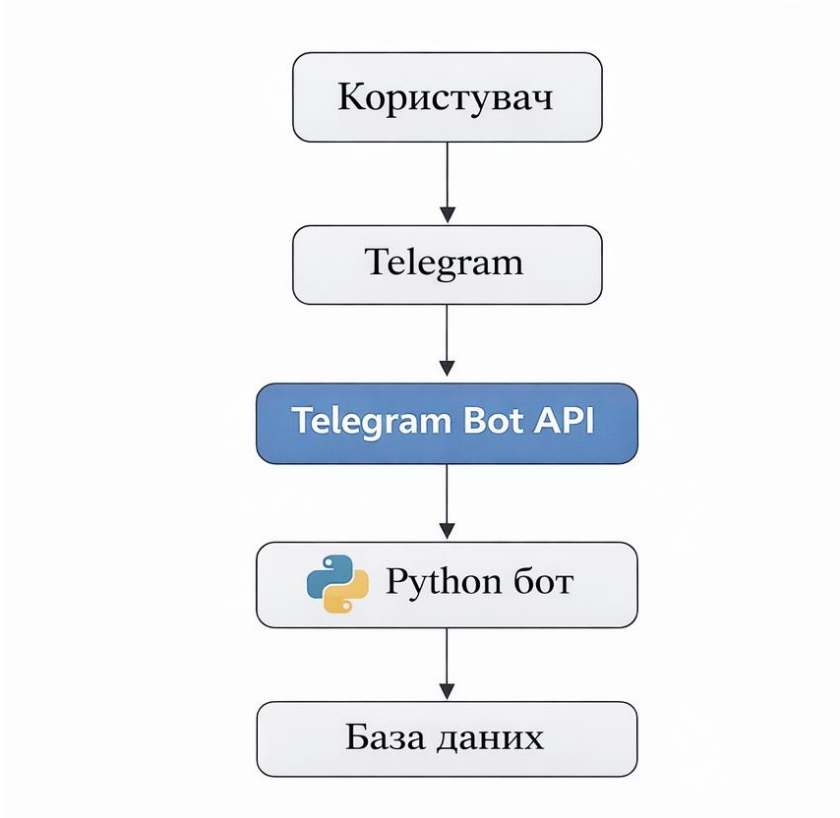

Рис. 1.4 – Архітектура взаємодії користувача з Telegram-ботом

Однією з функцій Telegram Bot API є підтримка команд, які дозволяють користувачеві взаємодіяти з ботом у зручний спосіб. Команди зазвичай починаються зі символу “/”, наприклад /start, /help або /test. За допомогою таких команд користувач може запускати бот, отримувати довідкову інформацію або переходити до певних функцій системи [9]. Ще однією функцією є можливість обміну повідомленнями між користувачем та ботом. Бот може надсилати текстові повідомлення, зображення, документи або інші типи контенту. У навчальних системах це дозволяє надсилати користувачам завдання, пояснення або результати тестування.

Для зручної взаємодії з користувачем Telegram Bot API підтримує використання клавіатур, які дозволяють створювати кнопки для вибору певних дій. Користувач може натискати кнопки замість введення текстових команд, що значно спрощує навігацію та використання бота.

Крім того, Telegram підтримує inline-кнопки, які розміщуються безпосередньо під повідомленням бота. За допомогою таких кнопок можна створювати інтерактивні меню, переходи між розділами або виконання певних функцій, наприклад запуск тесту або вибір теми навчання.

Telegram Bot API використовує механізм webhook, який дозволяє отримувати повідомлення від Telegram у режимі реального часу. У цьому випадку сервер Telegram автоматично надсилає дані про нові повідомлення на сервер бота, що забезпечує швидку обробку запитів користувачів та ефективну роботу системи.

Завдяки широкому набору функцій, простоті інтеграції та великій кількості користувачів месенджера Telegram створення ботів на цій платформі є ефективним рішенням для реалізації освітніх сервісів. Використання Telegram Bot API дозволяє створювати інтерактивні навчальні системи, що забезпечують зручний доступ до навчальних матеріалів, тестування знань та персоналізацію навчального процесу.

1.4 Ефективність використання чат-ботів у навчанні іноземних мов

Ефективність використання чат-ботів у процесі вивчення іноземних мов визначається їх здатністю забезпечувати безперервну взаємодію з користувачем, адаптацію навчального контенту та підвищення рівня залученості студентів. У умовах цифровізації освіти чат-боти виступають як інструмент, що доповнює традиційні методи навчання, надаючи можливість організувати індивідуалізований підхід до кожного здобувача освіти.

Доступність навчання у будь-який час та з будь-якого пристрою. Це дозволяє студентам самостійно обирати темп засвоєння матеріалу, повторювати складні теми та отримувати миттєвий зворотний зв’язок. Завдяки інтерактивному формату взаємодії, чат-боти сприяють розвитку навичок читання, письма та базового спілкування іноземною мовою.

Можливість автоматизації перевірки знань. Чат-боти здатні проводити тестування, оцінювати відповіді користувачів та надавати рекомендації щодо подальшого навчання. Це значно зменшує навантаження на викладачів і дозволяє зосередитися на більш складних завданнях, таких як розвиток мовлення та критичного мислення.

Використання чат-ботів сприяє підвищенню мотивації студентів. Елементи гейміфікації, такі як бали, рівні або досягнення, стимулюють регулярне використання навчального інструменту. Інтерактивність та простота використання роблять процес навчання більш привабливим, особливо для молодого покоління, яке активно користується цифровими технологіями.

Таблиця 1.2 – Оцінка ефективності використання чат-ботів у навчанні іноземних мов

Критерій	Переваги використання чат-ботів	Обмеження
Доступність	Доступ 24/7, можливість навчання з будь-якого пристрою	Потреба в інтернет-з’єднанні
Індивідуалізація навчання	Адаптація завдань під рівень користувача	Обмежена гнучкість у складних випадках
Зворотний зв’язок	Миттєва перевірка відповідей та рекомендації	Не завжди точна оцінка відкритих відповідей
Мотивація	Гейміфікація, інтерактивність	Може знижуватись інтерес при одноманітності
Автоматизація процесів	Зменшення навантаження на викладача	Не замінює повністю викладача
Розвиток мовних навичок	Практика читання та письма	Обмежений розвиток усного мовлення

Дані, наведені у таблиці, свідчать про те, що чат-боти є ефективним допоміжним інструментом у навчанні іноземних мов, проте їх використання має здійснюватися у поєднанні з традиційними методами навчання.

Разом з тим, ефективність чат-ботів має певні обмеження. Зокрема, вони не здатні повністю замінити живе спілкування з викладачем або носієм мови, що є важливим для розвитку усного мовлення та правильної вимови. Також існує ризик обмеженості функціоналу ботів, особливо у випадках складних мовних конструкцій або нестандартних запитів користувача.

Отже, чат-боти є ефективним інструментом підтримки процесу вивчення іноземних мов, який поєднує доступність, автоматизацію та інтерактивність. Вони значно підвищують ефективність навчання за умови їх використання у поєднанні з традиційними методами викладання, що забезпечує комплексний підхід до формування мовних компетентностей.

Висновок до розділу 1

Розглянуто теоретичні основи використання цифрових технологій та чат-ботів у процесі вивчення іноземних мов. Проведений аналіз сучасних підходів до організації навчання показав, що традиційні методи навчання поступово доповнюються новими інформаційними технологіями, такими як онлайн-платформи, мобільні застосунки, системи e-learning та методики microlearning. Використання цих підходів дозволяє підвищити ефективність навчального процесу, зробити його більш доступним та адаптованим до потреб користувачів.

Проаналізовано можливості використання чат-ботів як інструменту автоматизації навчального процесу. Чат-боти дозволяють організувати інтерактивну взаємодію з користувачем, автоматизувати надання навчальних матеріалів, проводити тестування знань та забезпечувати персоналізацію навчання. Встановлено, що чат-боти можуть застосовуватися у різних сферах діяльності, зокрема в освіті, банківській сфері, технічній підтримці та маркетингу.

Визначено, що використання Telegram Bot API дозволяє реалізувати функціональні програмні системи, які підтримують роботу з командами, повідомленнями, клавіатурами, inline-кнопками та механізмом webhook. Завдяки відкритому програмному інтерфейсу, простоті інтеграції та значній кількості користувачів месенджера Telegram дана платформа є ефективним середовищем для розробки навчальних сервісів.

2 ПРОЄКТУВАННЯ СИСТЕМИ TELEGRAM-БОТА ДЛЯ ВИВЧЕННЯ ІНОЗЕМНИХ МОВ

2.1 Аналіз функціональних вимог до програмного застосунку

Під час розробки програмних систем важливим етапом є визначення функціональних вимог до майбутнього застосунку. Функціональні вимоги визначають перелік можливостей системи, які повинні забезпечувати взаємодію користувача з програмним продуктом та виконання основних задач системи.

Основною метою розроблюваного програмного застосунку є створення Telegram-бота, який забезпечує інтерактивне вивчення іноземних мов за допомогою текстових вправ, тестових завдань та автоматизованої перевірки відповідей користувача. Система повинна забезпечувати зручну взаємодію користувача з ботом через месенджер Telegram та надавати можливість доступу до навчальних матеріалів у будь-який час.

До основних функціональних вимог програмного застосунку можна віднести такі можливості:

- запуск та ініціалізація роботи бота;
- взаємодія користувача з ботом через текстові команди;
- надання користувачеві навчальних завдань;
- перевірка правильності відповідей;
- збереження результатів навчання;
- налаштування параметрів навчання.

Важливою функцією системи є можливість налаштування параметрів навчального процесу, що дозволяє користувачу обирати мову навчання, тематику занять або рівень складності завдань. Це дозволяє адаптувати систему до різного рівня підготовки користувачів.

Крім функціональних вимог, важливу роль відіграють нефункціональні вимоги, які визначають характеристики роботи системи. До таких вимог можна віднести:

- швидкість обробки запитів користувача;
- стабільність роботи програмного застосунку;
- зручність користування системою;
- можливість подальшого розширення функціоналу.

Основні функціональні можливості системи представлено в табл. 2.1.

Таблиця 2.1 – Основні функціональні можливості Telegram-бота

Функція	Опис
Запуск бота	ініціалізація роботи бота за допомогою команди /start
Вибір параметрів навчання	вибір мови, теми або рівня складності
Отримання завдань	надсилання користувачу навчальних вправ
Перевірка відповідей	аналіз відповіді користувача та визначення її правильності
Збереження результатів	фіксація результатів виконання тестових завдань
Налаштування системи	можливість зміни параметрів навчального процесу

Для проєктування програмного застосунку важливим є визначення структури взаємодії між користувачем та системою. У випадку Telegram-бота взаємодія здійснюється через інтерфейс месенджера, де користувач надсилає команди або повідомлення, а бот обробляє їх та формує відповідну відповідь.

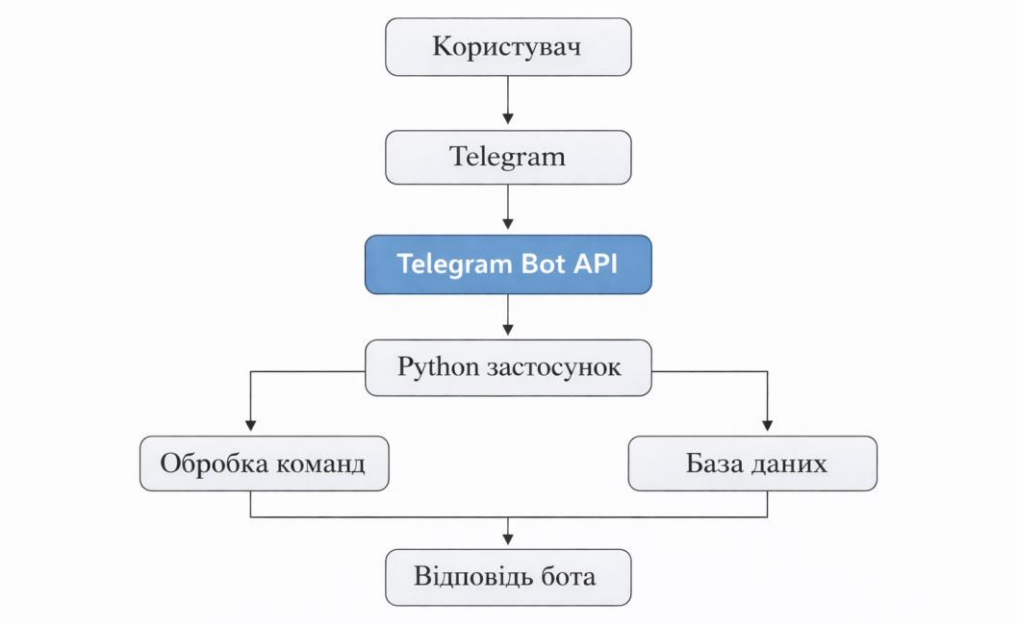

Рис. 2.1 – Загальна логіка роботи Telegram-бота

Після запуску бота користувач отримує доступ до основного меню системи, де може обрати необхідні функції навчання. Залежно від обраного режиму бот може пропонувати користувачеві виконати тестові завдання, повторити нові слова або переглянути результати попереднього навчання.

Обробка запитів користувача здійснюється серверною частиною застосунку, яка реалізована мовою програмування Python. Програмна логіка бота відповідає за аналіз отриманих повідомлень, визначення команди або дії користувача та виконання відповідної функції системи [10].

Також буде використання бази даних для збереження інформації. У базі даних можуть зберігатися дані про користувачів, навчальні матеріали, питання тестів та результати виконання завдань. Це дозволяє забезпечити персоналізацію навчального процесу та відстежувати прогрес користувача.

Крім того, система може підтримувати механізм автоматичних нагадувань або планування навчальних завдань. Завдяки цьому користувач отримує повідомлення з пропозицією виконати нове завдання або повторити матеріал у певний час.

Проведений аналіз функціональних вимог до програмного застосунку дозволив визначити основні можливості майбутньої системи та сформувати загальну модель її функціонування. Було встановлено, що Telegram-бот повинен забезпечувати взаємодію з користувачем через інтерфейс месенджера, обробку команд, надання навчальних завдань, перевірку відповідей та збереження інформації у базі даних.

Система повинна підтримувати механізми налаштування параметрів навчання, зокрема вибір мови, теми та рівня складності, а також реалізацію додаткових функцій, таких як нагадування про проходження тестування.

Отримані результати аналізу функціональних вимог є основою для подальшого проєктування програмної системи. На наступному етапі роботи доцільно розглянути архітектуру програмного застосунку, структуру його основних компонентів та принципи взаємодії між ними.

2.2 . Архітектура Telegram-бота та структура програмних модулів

У сучасних умовах розробка Telegram-ботів базується на використанні багаторівневих архітектур, що забезпечують гнучкість, масштабованість та інтеграцію із зовнішніми сервісами. Для аналізу підходів до побудови таких систем доцільно розглянути типову архітектуру Telegram-бота, яка реалізована із застосуванням хмарних технологій та модульного принципу побудови.

На рис. 2.2 представлено узагальнену архітектуру взаємодії користувача з Telegram-ботом через хмарну інфраструктуру.

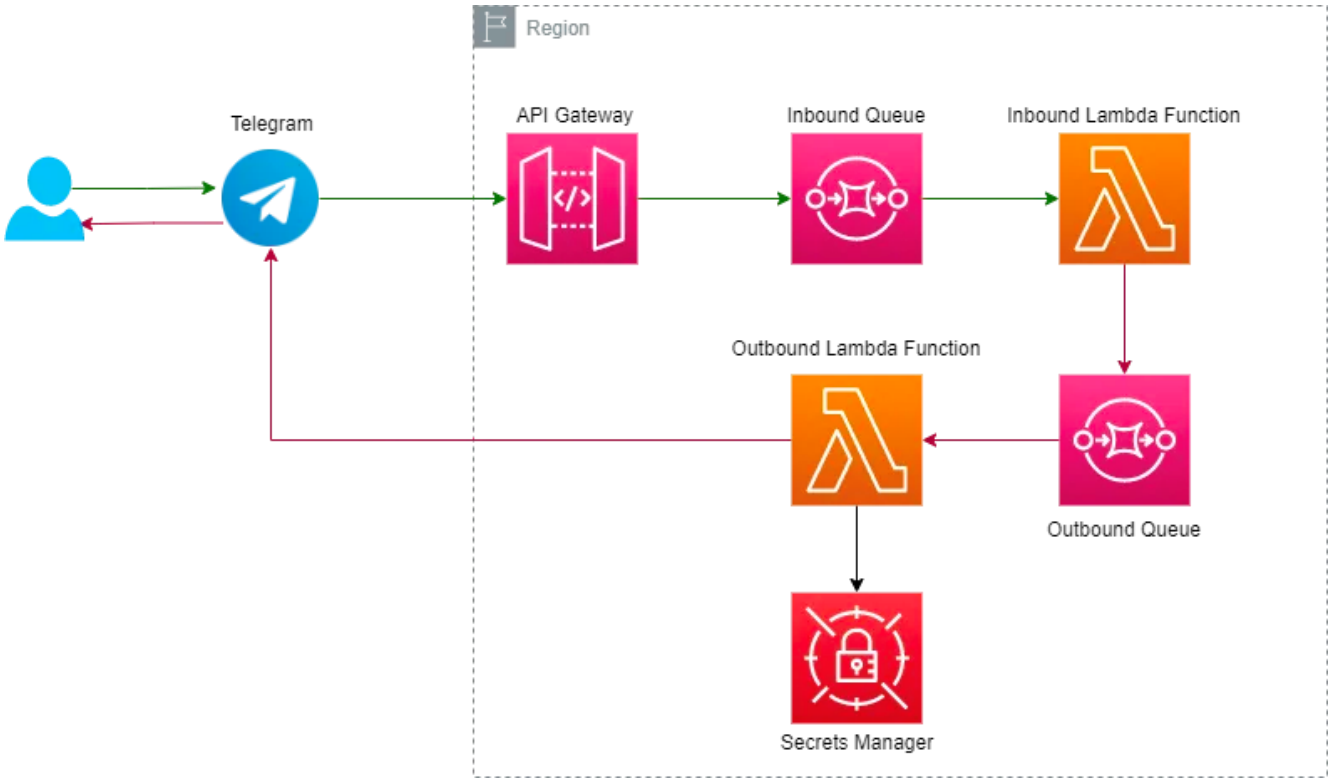

Рис. 2.2 – Архітектура взаємодії Telegram-бота з використанням хмарних сервісів

Як видно з рисунка, користувач взаємодіє з ботом через платформу Telegram, після чого запит передається до серверної частини системи через API-шлюз. Основні етапи обробки включають:

- прийом HTTP-запиту від Telegram;
- маршрутизацію запиту через API Gateway;
- обробку бізнес-логіки за допомогою обчислювальних функцій;
- координацію складних сценаріїв через механізми оркестрації;

– формування відповіді та її повернення користувачу.

Такий підхід відповідає концепції serverless-архітектури, що дозволяє уникнути постійного утримання серверної інфраструктури та забезпечує автоматичне масштабування системи залежно від навантаження.

Крім того, у структурі присутні компоненти моніторингу та безпеки, що дозволяють здійснювати контроль стану системи та реагувати на потенційні загрози.

Більш детальне представлення внутрішньої структури системи наведено на наступному рисунку 2.3.

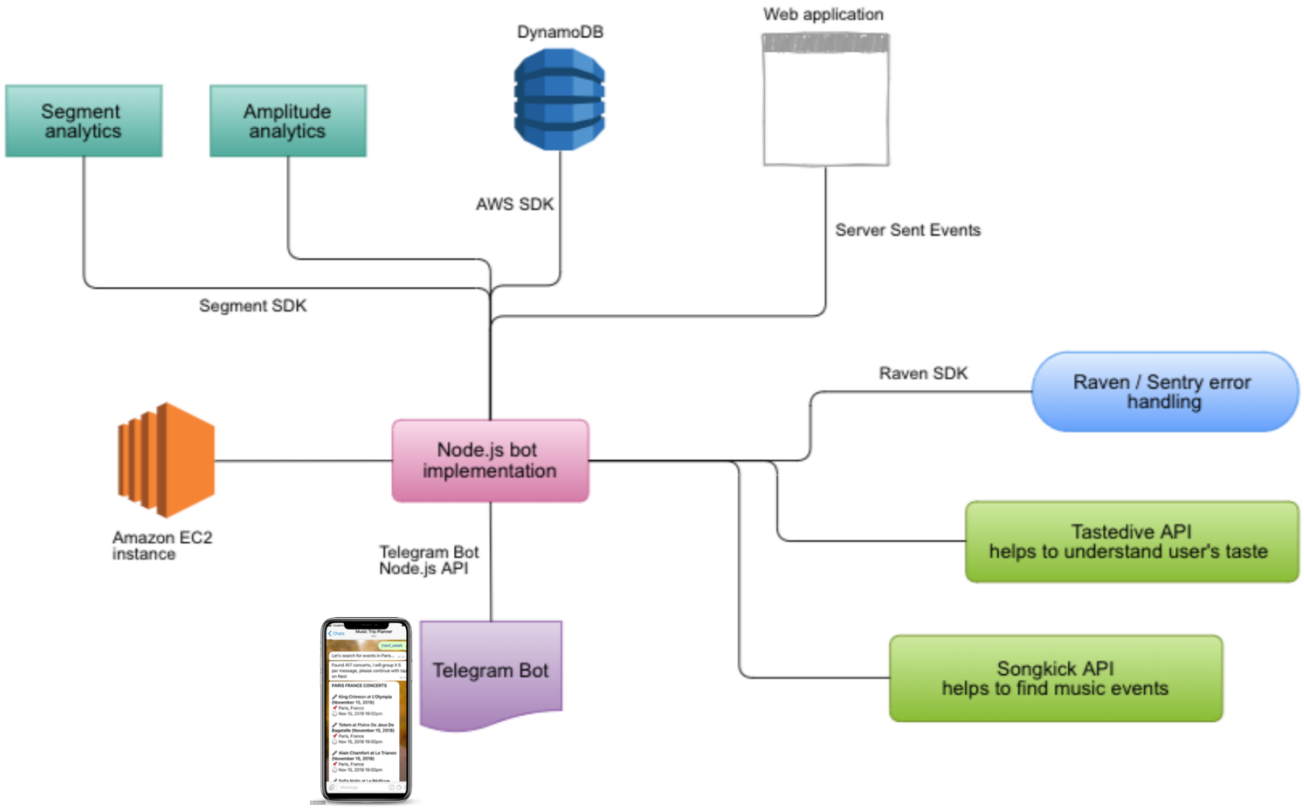

Рис. 2.3 – Структура програмної реалізації Telegram-бота та взаємодія модулів

Дана схема відображає внутрішню організацію програмного забезпечення Telegram-бота та його інтеграцію з зовнішніми сервісами.

Центральним елементом виступає серверна реалізація бота, яка виконує такі функції:

– обробка повідомлень користувача;

- управління логікою діалогу;
- взаємодія з базою даних;
- інтеграція з аналітичними та зовнішніми API.

У межах цієї архітектури можна виділити кілька ключових підсистем:

1. Клієнтський рівень

До нього належить Telegram-клієнт, через який користувач здійснює взаємодію з системою. Вся комунікація відбувається у форматі повідомлень, що обробляються ботом.

2. Серверна логіка (backend)

Реалізована на платформі Node.js, вона забезпечує:

- прийом і обробку запитів;
- формування відповідей;
- реалізацію сценаріїв взаємодії.

3. Рівень зберігання даних

Для збереження інформації використовується база даних (наприклад, NoSQL-рішення), яка містить:

- дані користувачів;
- історію запитів;
- параметри налаштувань.

4. Інтеграційний рівень

Система взаємодіє із зовнішніми сервісами:

- аналітичними платформами (для збору статистики);
- API сторонніх сервісів (для отримання додаткових даних);
- системами обробки подій.

Це дозволяє значно розширити функціональні можливості бота та забезпечити персоналізацію взаємодії.

5. Система моніторингу та обробки помилок

Включає інструменти логування та відстеження помилок, що дозволяє:

- оперативно виявляти проблеми;

- аналізувати роботу системи;
- підвищувати надійність програмного забезпечення.

На основі проаналізованої архітектури можна виділити типову модульну структуру Telegram-бота:

- Модуль обробки запитів – відповідає за прийом і первинну обробку повідомлень;
- Модуль бізнес-логіки – реалізує основні сценарії роботи системи;
- Модуль доступу до даних – забезпечує взаємодію з базою даних;
- Модуль інтеграції – відповідає за роботу з зовнішніми API;
- Модуль логування та моніторингу – здійснює контроль роботи системи.

Модульний підхід дозволяє ізолювати функціональні частини системи, що спрощує розробку, тестування та подальшу модернізацію програмного продукту.

Кожен із зазначених модулів виконує чітко визначену роль у загальній архітектурі системи, що дозволяє забезпечити її гнучкість та масштабованість. Детальний розгляд їх функціонування дозволяє глибше зрозуміти принципи побудови Telegram-ботів.

Модуль обробки запитів (Handler Module) є першою точкою входу в систему. Він забезпечує прийом повідомлень від Telegram API, їхню валідацію та попередню обробку. На цьому етапі виконується:

- визначення типу повідомлення (текст, команда, callback-запит);
- нормалізація вхідних даних;
- перевірка коректності структури запиту;
- передача запиту до відповідного обробника.

Цей модуль часто реалізується у вигляді маршрутизатора (router), який визначає, який саме сценарій обробки буде застосовано.

Модуль бізнес-логіки (Service Layer) є центральною частиною системи. Саме тут реалізуються основні алгоритми функціонування бота. Залежно від призначення системи, цей модуль може включати:

- обробку команд користувача;

- формування відповідей;
- управління станом діалогу (state management);
- реалізацію сценаріїв взаємодії (flows).

Особливістю цього рівня є те, що він не залежить від конкретної реалізації інтерфейсу або джерела даних, що забезпечує можливість повторного використання логіки в інших системах.

Модуль доступу до даних (Data Access Layer) забезпечує взаємодію з базами даних та іншими сховищами інформації. Він виконує такі функції:

- збереження інформації про користувачів;
- ведення історії взаємодій;
- зберігання конфігурацій та налаштувань;
- кешування часто використовуваних даних.

Використання окремого шару доступу до даних дозволяє ізолювати бізнес-логіку від конкретної СУБД, що спрощує зміну технологій зберігання даних у майбутньому.

Модуль інтеграції (Integration Layer) відповідає за взаємодію із зовнішніми сервісами та API. У сучасних Telegram-ботах цей модуль відіграє ключову роль, оскільки значна частина функціональності базується на сторонніх сервісах. Основні задачі модуля:

- формування HTTP-запитів до зовнішніх API;
- обробка отриманих відповідей;
- трансформація даних у внутрішній формат системи;
- обробка помилок інтеграції.

Цей модуль дозволяє реалізувати розширений функціонал, наприклад:

- отримання аналітичних даних;
- пошук інформації;
- персоналізацію контенту.

Модуль логування та моніторингу (Logging & Monitoring Module) забезпечує контроль за роботою системи. Він виконує:

- запис подій та дій користувачів;
- фіксацію помилок та винятків;
- моніторинг продуктивності;
- формування звітів.

Наявність такого модуля є критично важливою для забезпечення надійності системи, особливо у випадку масштабування або роботи з великою кількістю користувачів.

Узагальнений процес обробки запиту користувача можна описати наступним чином:

1. Користувач надсилає повідомлення через Telegram;
2. Модуль обробки запитів приймає та аналізує повідомлення;
3. Визначається відповідний сценарій обробки;
4. Викликається модуль бізнес-логіки;
5. За необхідності здійснюється звернення до бази даних;
6. Виконується інтеграція із зовнішніми сервісами;
7. Формується відповідь користувачу;
8. Результат передається назад через Telegram API;
9. Паралельно здійснюється логування події.

Такий підхід забезпечує чітку послідовність виконання операцій та дозволяє легко масштабувати систему.

Переваги модульної архітектури Telegram-бота

Використання модульного підходу має ряд суттєвих переваг:

- Масштабованість – кожен модуль може бути розширений або замінений незалежно від інших;
- Гнучкість – можливість інтеграції нових функцій без зміни всієї системи;
- Підтримуваність – спрощення супроводу та оновлення програмного забезпечення;
- Повторне використання коду – окремі модулі можуть використовуватись в інших проєктах;

– Зниження складності – розподіл системи на логічні частини спрощує її розуміння.

Незважаючи на значні переваги, модульна архітектура має і певні недоліки:

– збільшення складності взаємодії між модулями;
– необхідність додаткової обробки помилок;
– затримки при інтеграції із зовнішніми сервісами;
– складність тестування інтеграційних сценаріїв.

Однак при правильному проєктуванні ці недоліки можуть бути мінімізовані.

У випадку збільшення навантаження система може масштабуватися за рахунок:

– горизонтального масштабування серверної частини;
– використання serverless-технологій;
– кешування запитів;
– розподілу навантаження між сервісами.

Це дозволяє забезпечити стабільну роботу навіть при значній кількості користувачів.

2.3 Проєктування взаємодії користувача з Telegram-ботом

У процесі розроблення системи Telegram-бота для вивчення іноземних мов важливим етапом є проєктування взаємодії користувача із застосунком. Саме від зручності та логічності інтерфейсу залежить ефективність використання бота, рівень залученості користувачів та загальний досвід роботи із системою.

Взаємодія користувача з ботом реалізується через текстові повідомлення, кнопки та інтерактивні елементи, які надає платформа Telegram. Основою побудови інтерфейсу є сценарний підхід, при якому кожна дія користувача викликає відповідну реакцію системи. Такий підхід дозволяє створити зрозумілу послідовність дій та забезпечити простоту навігації.

Основні сценарії взаємодії включають:

– вибір мови навчання;

- налаштування рівня складності;
- вибір тематики завдань;
- проходження тестів;
- отримання результатів та рекомендацій;
- активацію системи нагадувань.

Для підвищення зручності використання реалізовано кнопкове меню, яке дозволяє користувачу швидко обирати необхідні функції без введення текстових команд. Крім того, використання інлайн-кнопок забезпечує динамічну зміну інтерфейсу залежно від обраного режиму роботи.

Особлива увага приділяється організації зворотного зв’язку. Після виконання тестових завдань користувач отримує миттєву інформацію про правильність відповіді, що сприяє кращому засвоєнню матеріалу. Також передбачено можливість повторного проходження завдань для закріплення знань.

Крім базових сценаріїв взаємодії, важливим аспектом є логіка керування станами користувача. У Telegram-боті кожен користувач може перебувати в певному стані (наприклад: вибір мови, проходження тесту, перегляд результатів), що визначає подальшу поведінку системи. Для цього використовується підхід із збереженням поточного стану користувача, що дозволяє коректно обробляти його дії та уникати помилок у навігації.

Також у системі реалізовано персоналізацію взаємодії, яка полягає у врахуванні раніше обраних параметрів користувача, таких як рівень складності, тематика завдань та мова навчання. Це дозволяє формувати індивідуальний навчальний сценарій та підвищує ефективність засвоєння матеріалу.

Важливу роль відіграє обробка помилок та некоректних дій користувача. У випадку введення невірних або неочікуваних даних бот надає підказки або пропонує повернутися до попереднього кроку. Такий механізм забезпечує стійкість системи та покращує користувацький досвід.

Додатково передбачено реалізацію системи повідомлень та нагадувань, яка дозволяє підтримувати регулярність навчання. Користувач може отримувати

повідомлення з пропозицією пройти нові тести або повторити вже вивчений матеріал, що сприяє формуванню стабільної навчальної активності.

Ще одним важливим елементом є структурування діалогів, яке забезпечує послідовність і логічність навчального процесу. Кожен етап взаємодії має чітко визначену мету та результат, що дозволяє уникнути перевантаження користувача інформацією та спрощує використання системи.

Таким чином, реалізація ефективної моделі взаємодії користувача з Telegram-ботом базується на поєднанні сценарного підходу, керування станами, персоналізації та обробки помилок, що у сукупності забезпечує зручність використання та високу ефективність навчального процесу.

Висновок до розділу 2

У другому розділі було розглянуто підходи до побудови архітектури Telegram-бота та проаналізовано основні принципи організації його програмної структури. Визначено, що сучасні рішення базуються на використанні клієнт-серверної взаємодії, хмарних технологій та модульного підходу до розробки. Проаналізована архітектура дозволяє забезпечити ефективну обробку запитів користувачів, масштабованість системи та її адаптацію до змін навантаження.

Досліджено структуру програмних модулів Telegram-бота, зокрема модуль обробки запитів, бізнес-логіки, доступу до даних, інтеграції із зовнішніми сервісами та моніторингу. Встановлено, що чіткий розподіл функцій між модулями сприяє підвищенню надійності системи, спрощує процес розробки та тестування, а також забезпечує можливість подальшого розширення функціональних можливостей програмного продукту.

Таким чином, проведений аналіз архітектури та структури Telegram-бота дозволив сформувати теоретичну та практичну основу для реалізації програмного застосунку. Отримані результати є базою для подальшого етапу розробки, що передбачає реалізацію функціональних можливостей системи та оцінку її ефективності.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ TELEGRAM-БОТА

3.1 Реалізація бота на мові програмування Python

Для кращого і більш легкого розуміння побудови телеграм бота на технології програмування Python було розділено код на логічні блоки на основі функціональності. Кожен блок містить фрагмент коду, за яким йде опис, який пояснює, що робить цей блок. Опис описує послідовність дій, які виконує код, з акцентом на ключові операції.

Блок 1: Імпорти (Imports)

```
import asyncio
import random
from datetime import datetime

from aiogram import Bot, Dispatcher, F
from aiogram.types import (
    Message, CallbackQuery,
    InlineKeyboardMarkup, InlineKeyboardButton
)
from aiogram.filters import Command
from aiogram.fsm.state import State, StatesGroup
from aiogram.fsm.context import FSMContext
from aiogram.fsm.storage.memory import MemoryStorage
import aiosqlite
from apscheduler.schedulers.asyncio import AsyncIOScheduler
from apscheduler.triggers.cron import CronTrigger
from aiogram.client.default import DefaultBotProperties
```

Перший блок це імпорт можливих інструментів, які знадобляться для коректної роботи телеграм бота, а саме:

1. Імпорт модуль `asyncio` для асинхронного програмування, що дозволяє виконувати кілька задач одночасно (наприклад, обробку повідомлень і планування завдань).

2. Імпорт `random` для генерації випадкових чисел (використовується для вибору випадкових питань у тесті).

3. Імпорт `datetime` для роботи з датою та часом (використовується в нагадуваннях для перевірки поточного часу).

4. Імпорт ключові компоненти з `aiogram`: `Bot` для створення бота, `Dispatcher` для маршрутизації подій, `F` для фільтрів, типи повідомлень (`Message`, `CallbackQuery` тощо) для обробки вхідних даних, фільтри (`Command`) для команд, FSM-класи (`State`, `StatesGroup`, `FSMContext`) для управління станами користувача, `MemoryStorage` для тимчасового зберігання станів у пам'яті.

5. Імпорт `aiosqlite` для асинхронної роботи з базою даних SQLite (для зберігання користувачів, мов, тем і питань).

6. Імпорт `AsyncIOScheduler` і `CronTrigger` з `apscheduler` для планування періодичних завдань (наприклад, щоденні нагадування).

7. Імпорт `DefaultBotProperties` для налаштування бота (наприклад, режим розбору HTML).

Блок 2: Конфігурація (Config)

```
TOKEN = "8502172478:AAEpGnPWpx61-rL9HmdfrVzG6WDUBn5hpMc"
ADMINS = [YOUR_ID]

bot = Bot(token=TOKEN, default=DefaultBotProperties(parse_mode="HTML"))
dp = Dispatcher(storage=MemoryStorage())
scheduler = AsyncIOScheduler()

DB_NAME = "bot.db"
```

Далі, у другому блоці встановлюється змінна TOKEN, що містить токен бота, отриманий від @BotFather і необхідний для його автентифікації. Далі визначається список ADMINS, який містить знайдений за допомогою @Get_myidrobot Telegram ID адміністраторів для надання доступу до адміністративної панелі. Після цього створюється об’єкт bot класу Bot із зазначеним токеном та налаштуваннями за замовчуванням, зокрема режимом розбору HTML для форматування тексту повідомлень. Також ініціалізується об’єкт dp класу Dispatcher із використанням MemoryStorage, що забезпечує зберігання станів користувачів у оперативній пам’яті для підвищення швидкодії. Окремо створюється об’єкт scheduler для планування завдань, зокрема надсилання нагадувань, і задається ім’я файлу бази даних DB_NAME (“bot.db”), який використовується для зберігання даних.

Блок 3: Визначення станів FSM (FSM States)

```python
class UserState(StatesGroup):
    add_new_language = State()
    add_new_topic = State()

    add_question_lang = State()
    add_question_topic = State()
    add_question_text = State()

    add_option_a = State()
    add_option_b = State()
    add_option_c = State()

    add_correct = State()
    set_reminder_hour = State()
```

Наступним кроком створюється клас UserState, який успадковує StatesGroup з aiogram FSM (Finite State Machine) та використовується для керування багатокроковими діалогами з користувачем. У межах цього класу визначаються стани як об’єкти State(): add_new_language для додавання нової мови, add_new_topic для створення теми, набір станів add_question_* для поетапного додавання питання (вибір мови, теми, введення тексту запитання, варіантів відповідей A/B/C та правильної відповіді), а також set_reminder_hour для налаштування часу нагадування. Завдяки цим станам бот зберігає контекст розмови та послідовно переходить від одного кроку до іншого, поступово збираючи необхідні дані.

Блок 4: Ініціалізація бази даних (DB Initialization)

```python
async def init_db():
    async with aiosqlite.connect(DB_NAME) as db:

        # USERS
        await db.execute("""
            CREATE TABLE IF NOT EXISTS users (
                telegram_id INTEGER PRIMARY KEY,
                language_id INTEGER DEFAULT 1,
                difficulty TEXT DEFAULT 'Medium',
                topic_id INTEGER DEFAULT 1,
                reminder INTEGER DEFAULT 0,
                reminder_hour INTEGER DEFAULT 18
            )
        """)

        # LANGUAGES
        await db.execute("""
            CREATE TABLE IF NOT EXISTS languages (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                name TEXT UNIQUE
            )
        """)
```

У четвертому блоці програма асинхронно підключається до бази даних за допомогою aiosqlite.connect(DB_NAME), після чого створює необхідні таблиці, якщо вони ще не існують. Формується таблиця users для зберігання даних користувачів, зокрема Telegram ID, обраної мови, рівня складності, теми, статусу нагадування та години його надсилання з установленими значеннями за замовчуванням. Далі створюється таблиця languages з автоінкрементним ідентифікатором і унікальною назвою мови, таблиця topics з автоінкрементним ID, прив'язкою до мови та назвою теми, а також таблиця questions, яка містить автоінкрементний ID, посилання на мову й тему, рівень складності, текст запитання, варіанти відповідей A, B, C та правильний варіант. Після цього до бази додається стандартна мова «Англійська» з ID 1 і тема «Grammar» з ID 1 для цієї мови, якщо вони відсутні, а всі внесені зміни зберігаються за допомогою db.commit().

Блок 5: Утилітарні функції (Utilities)

```
async def ensure_user(user_id):
    async with aiosqlite.connect(DB_NAME) as db:
        await db.execute(
            "INSERT OR IGNORE INTO users (telegram_id) VALUES (?)",
            (user_id,)
        )
        await db.commit()


async def get_user(user_id):
    async with aiosqlite.connect(DB_NAME) as db:
        async with db.execute(
            "SELECT language_id, difficulty, topic_id, reminder, reminder_hour "
            "FROM users WHERE telegram_id=?",
            (user_id,)
        ) as cur:
            return await cur.fetchone()


async def update_user(user_id, field, value):
    async with aiosqlite.connect(DB_NAME) as db:
        await db.execute(
            f"UPDATE users SET {field}=? WHERE telegram_id=?",
            (value, user_id)
        )
        await db.commit()
```

Далі у функції ensure_user виконується асинхронне підключення до бази даних, після чого здійснюється додавання нового запису користувача із зазначеним telegram_id, якщо такого запису ще не існує, та фіксація змін у базі. Функція

get_user також підключається до БД, вибирає дані користувача за telegram_id, зокрема мову, рівень складності, тему, статус нагадування та годину його надсилання, і повертає отримані значення у вигляді кортежу. У функції update_user відбувається підключення до бази, оновлення вказаного поля field новим значенням value для користувача з відповідним telegram_id, після чого зміни зберігаються.

Блок 6: Клавіатури (Keyboards)

```
def back_kb(cb: str):
    return InlineKeyboardMarkup(inline_keyboard=[
        [InlineKeyboardButton(text="⬅ Назад", callback_data=cb)]
    ])


def main_menu_kb(is_admin: bool, reminder: int):
    buttons = [
        [InlineKeyboardButton(text="▶ Почати тест", callback_data="test_start")],
        [InlineKeyboardButton(text="⚙ Налаштування", callback_data="settings")],
        [InlineKeyboardButton(
            text=f"🔔 Нагадування: {'Увімкнено' if reminder else 'Вимкнено'}",
            callback_data="reminder_settings"
        )],
    ]

    if is_admin:
        buttons.append(
            [InlineKeyboardButton(text="🛠 Адмін", callback_data="admin")]
        )

    return InlineKeyboardMarkup(inline_keyboard=buttons)
```

У 6 блоці реалізовано функції back_kb створюється inline-клавіатура з однією кнопкою «⬅ Назад», якій призначаються callback-дані cb, що забезпечують повернення користувача до попереднього меню. У функції main_menu_kb формується набір кнопок головного меню, зокрема «Почати тест», «Налаштування» та «Нагадування», причому текст останньої змінюється залежно від поточного статусу параметра reminder. Якщо користувач має права адміністратора (is_admin = True), додатково відображається кнопка «Адмін». Після цього всі кнопки об’єднуються в об’єкт InlineKeyboardMarkup, який повертається для подальшого відображення в повідомленні.

Блок 7: Головне меню (Main Menu)

```
async def show_main_menu(user_id: int):
    await ensure_user(user_id)

    user = await get_user(user_id)
    if not user:
        return

    lang_id, diff, topic_id, reminder, _ = user

    async with aiosqlite.connect(DB_NAME) as db:

        async with db.execute(
            "SELECT name FROM languages WHERE id=?",
            (lang_id,)
        ) as cur:
            lang = (await cur.fetchone() or ["-"])[0]

        async with db.execute(
            "SELECT name FROM topics WHERE id=?",
            (topic_id,)
        ) as cur:
            topic = (await cur.fetchone() or ["-"])[0]

    text = (
        f"<b>{lang}</b> • <b>{topic}</b> • "
        f"<b>{diff}</b>\n\nОберіть дію:"
    )

    await bot.send_message(
        user_id,
        text,
        reply_markup=main_menu_kb(user_id in ADMINS, reminder)
    )


@dp.message(Command("start"))
async def start_cmd(message: Message):
    await show_main_menu(message.from_user.id)
```

Сьомий блок реалізує функції show_main_menu спочатку перевіряється наявність користувача в базі даних за допомогою ensure_user. Далі через get_user отримуються його дані; якщо запис відсутній, виконання функції припиняється. Отримані значення розпаковуються в окремі змінні — мова, рівень складності, тема та статус нагадування. Після цього здійснюється підключення до бази даних для отримання назв обраної мови й теми; якщо відповідні записи не знайдено, відображається символ «-». На основі цих даних формується текст повідомлення з поточними налаштуваннями користувача, який надсилається разом із головною inline-клавіатурою з урахуванням перевірки прав адміністратора. В обробнику

@dp.message(Command("start")) при отриманні команди /start викликається функція show_main_menu для Telegram ID користувача, що ініціює відображення головного меню.

Блок 8: Повернення до головного меню (Back to Main)

```python
@dp.callback_query(F.data == "main")
async def back_to_main(call: CallbackQuery):
    await call.answer()

    try:
        await call.message.delete()
    except:
        pass

    await show_main_menu(call.from_user.id)
```

Восьмий блок – обробник callback із даними «main» (кнопка «Вийти» або «Назад») спочатку підтверджується отримання події за допомогою call.answer(), після чого виконується спроба видалити поточне повідомлення для очищення чату від попереднього меню. Після цього викликається функція show_main_menu, яка повторно відображає користувачеві головне меню.

Блок 9: Налаштування (Settings)

```python
@dp.callback_query(F.data == "settings")
async def settings(call: CallbackQuery):
    await call.answer()

    kb = InlineKeyboardMarkup(inline_keyboard=[
        [InlineKeyboardButton(text="🌐 Мова", callback_data="set_language")],
        [InlineKeyboardButton(text="🎨 Тема", callback_data="set_topic")],
        [InlineKeyboardButton(text="📊 Складність", callback_data="set_difficulty")],
        [InlineKeyboardButton(text="⬅ Назад", callback_data="main")]
    ])

    await call.message.edit_text("⚙ Налаштування:", reply_markup=kb)


@dp.callback_query(F.data == "set_language")
async def set_language(call: CallbackQuery):
    await call.answer()

    async with aiosqlite.connect(DB_NAME) as db:
        async with db.execute("SELECT id, name FROM languages") as cur:
            rows = await cur.fetchall()

    buttons = [
        [InlineKeyboardButton(text=name, callback_data=f"lang_{id}")]
        for id, name in rows
    ]
```

У дев’ятому блоці налаштувань реалізовано наступну логіку: при отриманні callback «settings» створюється клавіатура з кнопками для вибору мови, теми, рівня складності та кнопкою «Назад», після чого редагується повідомлення користувача. Для callback «set_language» з бази даних витягується список доступних мов, формується відповідна клавіатура з кнопками для кожної мови та додатковою кнопкою «Назад», і повідомлення редагується. Коли користувач обирає мову (callback «lang_»), оновлюється його параметр мови в базі, тема скидається на першу доступну для нової мови або на None, після чого видаляється поточне повідомлення і показується головне меню. Аналогічно, при виборі callback «set_topic» отримуються теми для поточної мови, створюються кнопки і редагується повідомлення; якщо тем немає, користувач бачить повідомлення про помилку. При обранні теми (callback «topic_») оновлюється відповідний параметр користувача, повідомлення видаляється і відображається головне меню. Для callback «set_difficulty» формуються кнопки для рівнів складності, редагується повідомлення, а при виборі конкретного рівня (callback «diff_») оновлюється складність користувача, видаляється повідомлення та повторно показується головне меню.

3.2 Реалізація системи тестування та перевірки відповідей

Блок 10: Тестування (Test)

```
async def get_random_question(user_id: int):
    user = await get_user(user_id)

    if not user:
        return None

    lang_id, diff, topic_id, _, _ = user

    if topic_id is None:
        return None
```

Десятий блок працює з функцією get_random_question в якій спочатку отримуються дані користувача та перевіряється наявність обраної теми. Потім з

бази даних витягуються питання, які відповідають вибраній мові, темі та рівню складності; якщо таких питань немає, функція повертає None, інакше випадковим чином обирається одне питання. Для callback «test_start» видаляється поточне повідомлення, отримується випадкове питання; якщо його немає, користувач отримує повідомлення про помилку з кнопкою «Назад», інакше формується клавіатура з варіантами відповіді та кнопкою «Вийти», після чого питання надсилається користувачеві. У callback «ans_» визначається ID питання та вибраний користувачем варіант, отримується правильна відповідь з бази даних і перевіряється відповідність, підтримуючи як літери (A, B, C), так і текстові варіанти. Після цього надсилається повідомлення з результатом, і виконується запуск нового питання через start_test.

Блок 11: Нагадування (Reminder)

```
async def reminder_job():
    current_hour = datetime.now().hour

    async with aiosqlite.connect(DB_NAME) as db:
        async with db.execute(
            "SELECT telegram_id FROM users WHERE reminder=1 AND reminder_hour=?",
            (current_hour,)
        ) as cur:

            users = await cur.fetchall()

    for (uid,) in users:
        try:
            await bot.send_message(
                uid,
                "⏰ Час пройти тест!\nНатисніть /start"
            )
        except:
            pass


@dp.callback_query(F.data == "reminder_settings")
async def reminder_settings(call: CallbackQuery):
    await call.answer()

    user = await get_user(call.from_user.id)

    if not user:
        return
```

У одинадцятому блоці нагадування зроблені через функцію reminder_job, яка

спочатку визначає поточну годину, після чого з бази даних витягуються всі користувачі з увімкненим нагадуванням на цю годину, і кожному з них надсилається повідомлення-нагадування.

Для callback «reminder_settings» отримується поточний статус та година нагадування користувача, створюється клавіатура для увімкнення/вимкнення, зміни часу або повернення назад, і редагується повідомлення з відображенням цих налаштувань. При виборі «toggle_reminder» статус нагадування змінюється на протилежний (0 або 1), повідомлення видаляється і показується головне меню. У callback «set_reminder_time» встановлюється стан set_reminder_hour, редагується повідомлення з інструкцією введення години та кнопкою «Назад». У обробнику цього стану введене значення перевіряється на коректність (0–23), оновлюється в базі, стан очищується і користувачеві відображається головне меню; при некоректному вводі генерується повідомлення про помилку.

Блок 12: Адмін-панель (Admin Panel)

```
@dp.callback_query(F.data == "admin")
async def admin_panel(call: CallbackQuery):
    await call.answer()

    kb = InlineKeyboardMarkup(inline_keyboard=[
        [InlineKeyboardButton(text="➕ Додати мову", callback_data="admin_add_lang")],
        [InlineKeyboardButton(text="➕ Додати топік", callback_data="admin_add_topic")],
        [InlineKeyboardButton(text="➕ Додати питання", callback_data="admin_add_question")],
        [InlineKeyboardButton(text="⬅ Назад", callback_data="main")]
    ])

    await call.message.edit_text("🛠 Адмін панель:", reply_markup=kb)


# --- Додавання мови ---
@dp.callback_query(F.data == "admin_add_lang")
async def admin_add_lang(call: CallbackQuery, state: FSMContext):
    await call.answer()

    await state.set_state(UserState.add_new_language)

    await call.message.edit_text(
        "Введіть назву нової мови:",
        reply_markup=back_kb("admin")
    )
```

Наступний, дванадцятий блок виконує системі адміністративні callback події, які реалізовано через повний цикл додавання мов, тем і питань. При отриманні callback «admin» створюється клавіатура з кнопками для додавання мови, теми, питання та кнопкою «Назад», і редагується повідомлення. Для «admin_add_lang» встановлюється стан введення назви нової мови, і повідомлення редагується з кнопкою «Назад». У обробнику стану add_new_language перевіряється унікальність назви, мова додається в базу, стан очищується і показується адміністративне меню. При callback «admin_add_topic» витягуються наявні мови, формуються кнопки та редагується повідомлення. Для «addt_lang_» обрана мова зберігається, встановлюється стан введення назви теми, повідомлення редагується, а в обробнику стану add_new_topic нова тема перевіряється, вставляється в БД, стан очищується і відображається меню. При callback «admin_add_question» спочатку витягуються мови, створюються кнопки та редагується повідомлення; вибір мови («addq_lang_») зберігає її, витягує теми й створює кнопки з опцією «Новий топік». Callback «addq_topic_» зберігає обрану тему та показує кнопки для вибору складності, а «addq_new_topic» встановлює стан введення нової теми. У обробнику стану add_question_topic введена назва перевіряється, вставляється в базу, зберігається ID теми і відображаються кнопки для вибору складності. Callback «addq_diff_» зберігає обрану складність і встановлює стан введення тексту питання, після чого у стані add_question_text зберігається текст питання і бот переходить до збору варіантів A, B та C по черзі, зберігаючи кожен у відповідному стані. У фінальному стані add_correct перевіряється правильна відповідь (A/B/C), питання з усіма зібраними даними вставляється в базу, стан очищується і адміністративне меню знову відображається.

Блок 13: Запуск бота (Run)

```
async def main():
    await init_db()

    # Запускаємо нагадування точно на початку кожної години
    scheduler.add_job(
        reminder_job,
        CronTrigger(minute=0)
    )

    scheduler.start()
```

І в останньому, тринадцятому блоці, функції main спочатку виконується ініціалізація бази даних через init_db. Потім у планувальник додається завдання reminder_job, яке запускається щогодини на початку години за допомогою CronTrigger, після чого планувальник стартує. Далі запускається polling бота через dp.start_polling(bot), що дозволяє йому слухати та обробляти вхідні повідомлення. Якщо скрипт виконується безпосередньо, асинхронний цикл запускається командою asyncio.run(main()).

3.3 Системні налаштування та сповіщення

Після запуску бота користувач отримує доступ до головного меню, яке містить основні функції системи.

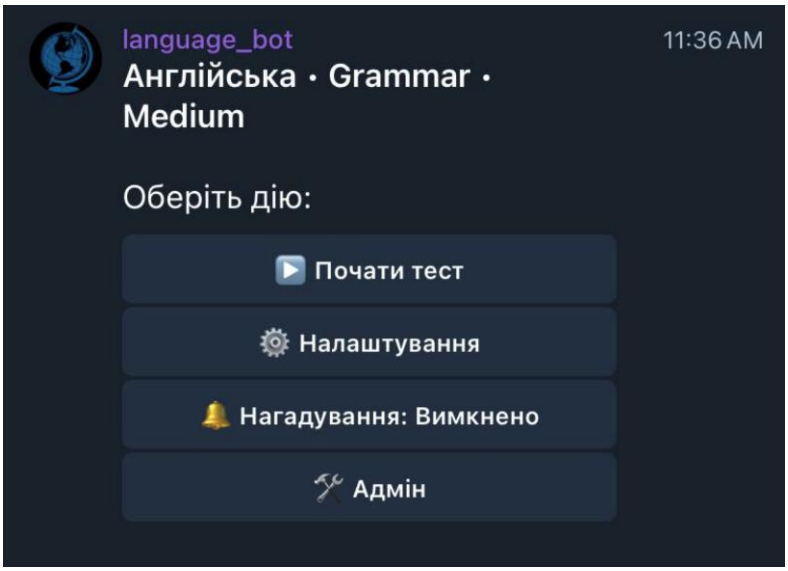

Рис. 3.1 – Головне меню Telegram-бота

Головне меню містить такі елементи:

- кнопка «Почати тест» – запускає процес проходження тестування;
- кнопка «Налаштування» – відкриває параметри конфігурації;
- кнопка «Нагадування» – дозволяє керувати системою сповіщень;
- кнопка «Адмін» – забезпечує доступ до адміністративного функціоналу.

Таке структурування меню дозволяє швидко орієнтуватися у функціональних можливостях системи.

У верхній частині інтерфейсу відображається інформація про поточні параметри навчання.

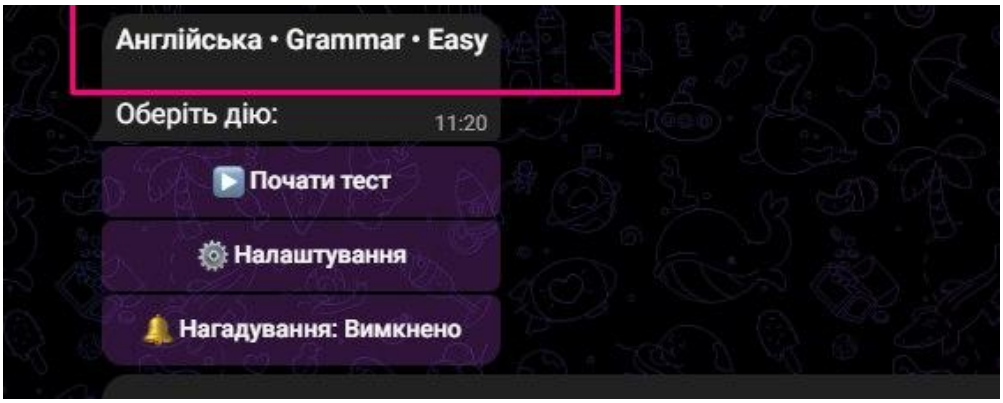

Рис. 3.2 – Відображення поточних параметрів навчання

Даний блок містить:

- обрану мову;
- тему навчання;
- рівень складності.

Це дозволяє користувачу контролювати параметри перед початком тестування та уникати помилок при виборі налаштувань.

Основною функцією системи є проходження тестів, що реалізовано через окрему кнопку в інтерфейсі.

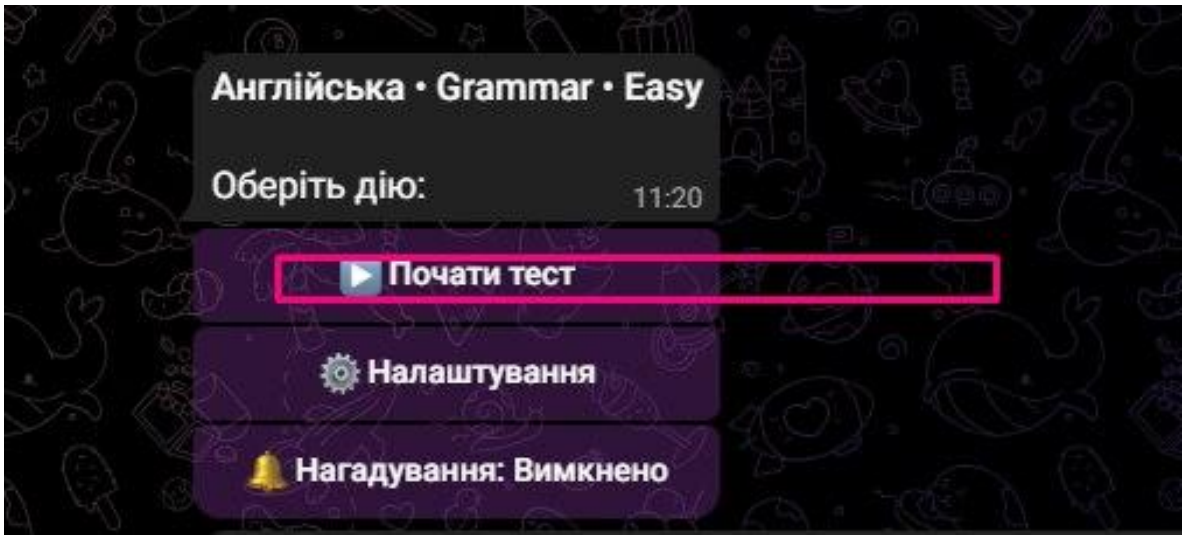

Рис. 3.3 – Інтерфейс запуску тестування

Після натискання кнопки система ініціює процес тестування, що включає:

- генерацію запитання;
- відображення варіантів відповіді;
- перевірку відповіді користувача.

Такий підхід забезпечує простий та швидкий доступ до навчального процесу.

Для зміни параметрів роботи бота передбачено окремий розділ налаштувань.

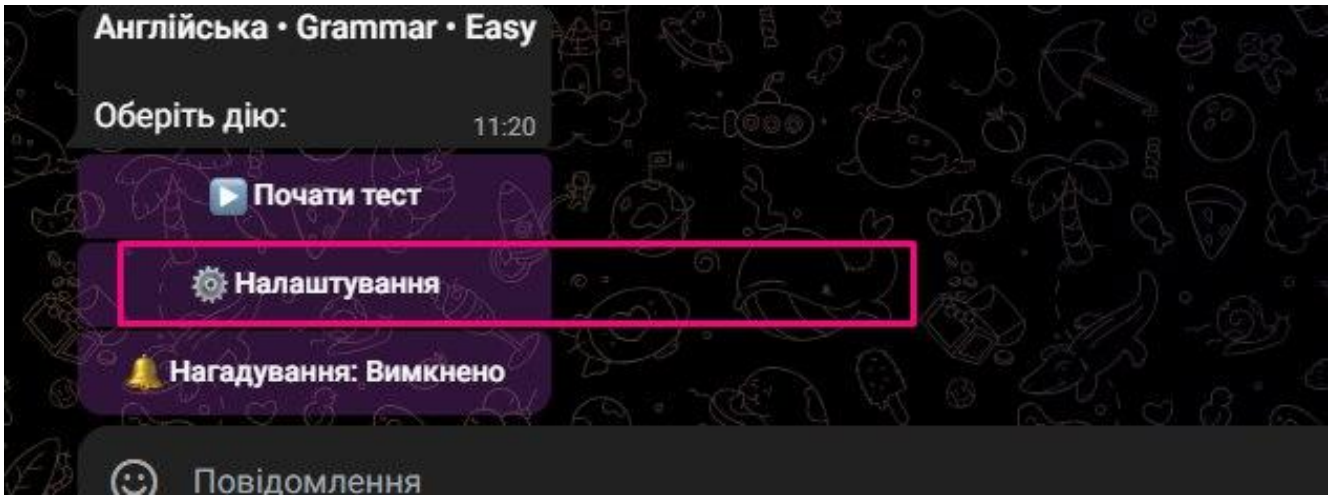

Рис. 3.4 – Перехід до меню налаштувань

Меню налаштувань дозволяє користувачу змінювати параметри роботи системи відповідно до власних потреб.

У розділі налаштувань представлено основні параметри конфігурації.

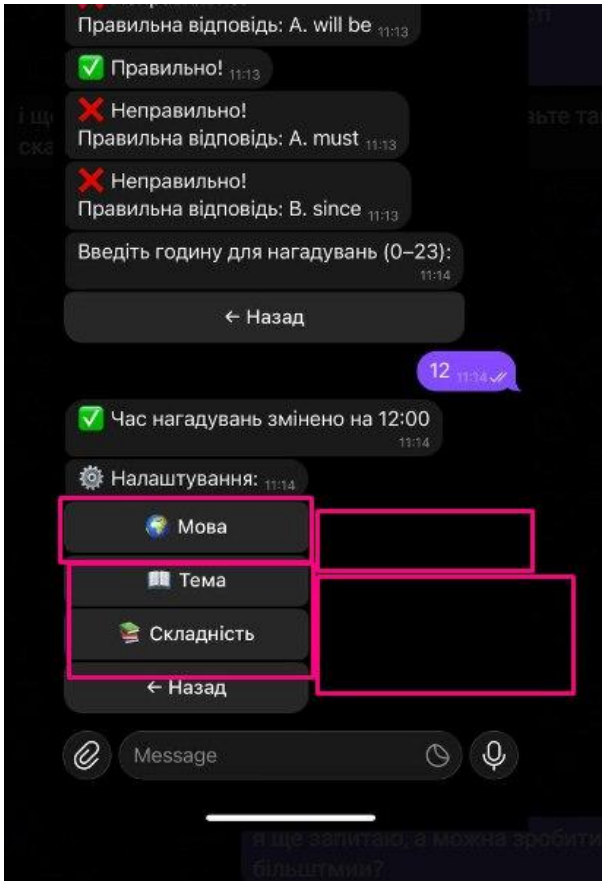

Рис. 3.5 – Основні параметри налаштувань бота

Користувач має можливість:

- обрати мову навчання;
- змінити тему;
- встановити рівень складності.

Це дозволяє адаптувати навчальний процес під індивідуальні потреби.

Після відповіді на запитання користувач отримує миттєвий зворотний зв’язок.

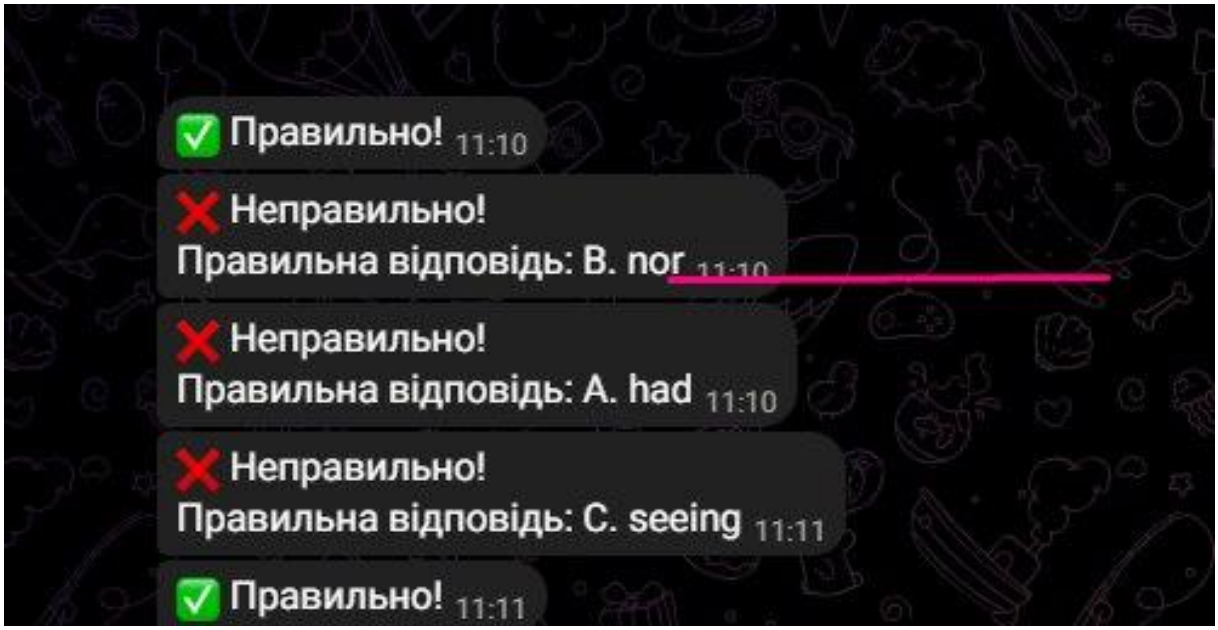

Рис. 3.6 – Відображення результатів тестування

Система повідомляє:

- правильність відповіді;
- правильний варіант у разі помилки.

Це дозволяє користувачу одразу аналізувати свої помилки та підвищувати рівень знань.

Висновок до розділу 3

У третьому розділі було розглянуто процес програмної реалізації Telegram-бота та особливості побудови його користувацького інтерфейсу. Визначено, що застосування кнопкової навігації, чіткої структури меню та логічної організації функціональних елементів забезпечує зручну та інтуїтивно зрозумілу взаємодію користувача із системою. Реалізований інтерфейс дозволяє швидко отримувати доступ до основних можливостей бота та ефективно виконувати необхідні дії.

Проаналізовано функціональні можливості системи, зокрема проходження тестування, отримання миттєвого зворотного зв’язку, налаштування параметрів навчання та використання системи нагадувань. Встановлено, що поєднання цих компонентів забезпечує персоналізацію навчального процесу та сприяє підвищенню ефективності засвоєння матеріалу. Особливу роль відіграє можливість адаптації складності та тематики завдань відповідно до потреб користувача.

Таким чином, результати третього розділу підтверджують, що розроблений Telegram-бот є функціонально завершеною системою, яка поєднує зручний інтерфейс, ефективну логіку роботи та можливості налаштування. Отримані результати демонструють практичну цінність розробленого застосунку та можуть бути використані для подальшого вдосконалення системи або розширення її функціональних можливостей.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було досліджено теоретичні та практичні аспекти розробки Telegram-бота як інструменту автоматизації навчального процесу. Проведений аналіз сучасних підходів до використання цифрових технологій показав, що інтерактивні програмні системи, зокрема чат-боти, є ефективним засобом підвищення якості навчання та залучення користувачів до освітнього процесу. Використання месенджерів як платформи для реалізації навчальних сервісів дозволяє забезпечити доступність та зручність використання системи.

У ході роботи було проаналізовано існуючі рішення та визначено доцільність використання Telegram як платформи для розробки бота. Встановлено, що Telegram Bot API надає широкі можливості для реалізації функціональних сервісів, включаючи обробку повідомлень, використання інтерактивних кнопок, налаштування сценаріїв взаємодії та організацію системи сповіщень. Завдяки простоті інтеграції та популярності месенджера, обрана платформа є оптимальним середовищем для реалізації поставлених задач.

У другому розділі було розглянуто архітектуру системи та визначено основні принципи побудови Telegram-бота. Встановлено, що використання модульного підходу до розробки дозволяє розподілити функціональність системи між окремими компонентами, такими як модуль обробки запитів, бізнес-логіки, доступу до даних, інтеграції та моніторингу. Такий підхід забезпечує гнучкість системи, спрощує її супровід та створює умови для подальшого масштабування.

У третьому розділі було реалізовано програмний застосунок Telegram-бота та досліджено особливості його функціонування. Розроблений інтерфейс забезпечує зручну взаємодію користувача із системою, включаючи можливість проходження тестування, отримання миттєвого зворотного зв’язку, налаштування параметрів навчання та використання системи нагадувань. Встановлено, що застосування інтерактивних елементів значно підвищує ефективність користування системою.

У процесі виконання роботи було досягнуто поставленої мети, а саме розроблено функціональний Telegram-бот для навчання іноземної мови, який реалізує основні можливості тестування знань, персоналізації навчального процесу та автоматизації взаємодії з користувачем. Розроблений застосунок може бути використаний як основа для створення більш складних навчальних систем або інтеграції з іншими інформаційними платформами.

Перспективами подальшого розвитку є розширення функціональних можливостей системи, зокрема впровадження адаптивних алгоритмів навчання, використання методів машинного навчання для персоналізації контенту, а також інтеграція з додатковими освітніми сервісами. Це дозволить підвищити ефективність навчання та розширити сферу застосування розробленого програмного продукту.

ПЕРЕЛІК ПОСИЛАНЬ

1. Telegram. Telegram Bot API Documentation [Електронний ресурс]. – Режим доступу: https://core.telegram.org/bots/api (дата звернення: 12.03.2026).
2. Telegram. Bots: An introduction for developers [Електронний ресурс]. – Режим доступу: https://core.telegram.org/bots (дата звернення: 12.03.2026).
3. aiogram developers. Aiogram documentation – asynchronous framework for Telegram Bot API [Електронний ресурс]. – Режим доступу: https://docs.aiogram.dev (дата звернення: 12.03.2026).
4. Python Software Foundation. Python Programming Language Documentation [Електронний ресурс]. – Режим доступу: https://docs.python.org/3 (дата звернення: 12.03.2026).
5. SQLite Consortium. SQLite Database Engine Documentation [Електронний ресурс]. – Режим доступу: https://sqlite.org/docs.html (дата звернення: 12.03.2026).
6. Holovaty B. The Ultimate Guide to Telegram Bots [Електронний ресурс]. – Режим доступу: https://www.freecodecamp.org/news/telegram-bot-guide (дата звернення: 12.03.2026).
7. Dale R. The return of the chatbots // Natural Language Engineering. – 2016. – Vol. 22(5). – P. 811–817.
8. Molnár G., Szüts Z. The role of chatbots in education // International Journal of Advanced Computer Science and Applications. – 2018. – Vol. 9(5). – P. 1–7.
9. Clark R., Mayer R. E-Learning and the Science of Instruction. – San Francisco: Pfeiffer, 2016. – 544 p.
10. Følstad A., Brandtzaeg P. B. Chatbots and the new world of HCI // Interactions. – 2017. – Vol. 24(4). – P. 38–42.
11. Adamopoulou E., Moussiades L. An overview of chatbot technology // Artificial Intelligence Applications and Innovations. – 2020. – P. 373–383.
12. Jurafsky D., Martin J. H. Speech and Language Processing. – 3rd ed. – Draft, 2021.

13. Russell S., Norvig P. Artificial Intelligence: A Modern Approach. – 4th ed. – Pearson, 2021. – 1152 p.

14. Goodfellow I., Bengio Y., Courville A. Deep Learning. – MIT Press, 2016. – 775 p.

15. Gartner Inc. Chatbots and Conversational AI Overview [Електронний ресурс]. – Режим доступу: https://www.gartner.com/en/information-technology/glossary/chatbot (дата звернення: 12.03.2026).

16. Duolingo Inc. Duolingo – language learning platform [Електронний ресурс]. – Режим доступу: https://www.duolingo.com (дата звернення: 12.03.2026).

17. Babbel GmbH. Babbel – Learn Languages Online [Електронний ресурс]. – Режим доступу: https://www.babbel.com (дата звернення: 12.03.2026).

18. Busuu Ltd. Busuu – Online Language Learning Platform [Електронний ресурс]. – Режим доступу: https://www.busuu.com (дата звернення: 12.03.2026).

19. Memrise Ltd. Memrise Language Learning Platform [Електронний ресурс]. – Режим доступу: https://www.memrise.com (дата звернення: 12.03.2026).

20. McTear M. F. Conversational AI: Dialogue Systems, Conversational Agents, and Chatbots. – Morgan & Claypool Publishers, 2020. – 150 p.

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА
«ТЕЛЕГРАМ-БОТ ДЛЯ ВИВЧЕННЯ ІНОЗЕМНИХ МОВ НА ОСНОВІ PYTHON»

Виконав студент: групи ІСД-42 Владислав ЛУК'ЯНОВ

Керівник роботи: доцент кафедри Валентина Данильченко

Київ – 2026

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ 2

Мета роботи:

Розробка програмного застосунку у вигляді Telegram -бота, що забезпечує інтерактивне вивчення іноземних мов, проведення тестів, налаштування рівня складності та тем навчання.

Об'єкт дослідження:

Система автоматизованого вивчення іноземних мов із використанням чат -ботів та месенджер -платформ.

Предмет дослідження:

Методи розробки та використання Telegram -ботів для організації інтерактивного навчального процесу, тестування знань та персоналізації навчання користувачів.

НАУКОВІ ЗАВДАННЯ ТА МЕТОДИ ДОСЛІДЖЕННЯ 3

Наукові завдання:

- Проаналізувати сучасні підходи до використання цифрових технологій у вивченні іноземних мов
- Дослідити можливості чат-ботів як інструменту автоматизації навчального процесу
- Вивчити Telegram Bot API для створення освітніх сервісів
- Сформувати функціональні та нефункціональні вимоги до застосунку
- Розробити архітектуру системи та структуру програмних модулів
- Реалізувати Telegram-бота на мові Python з використанням aiogram
- Реалізувати систему тестування та перевірки відповідей
- Провести тестування та оцінити ефективність роботи системи

Методи дослідження:

- Аналіз літературних та технічних джерел — сучасні підходи до чат-ботів в освіті
- Методи системного аналізу — дослідження структури навчальної системи
- Методи проєктування — розробка архітектури Telegram-бота
- Методи програмної реалізації — Python, aiogram, aiosqlite, APScheduler
- Експериментальні методи — тестування модулів та налаштувань
- Методи аналізу результатів — оцінка ефективності системи

АНАЛІЗ ІСНУЮЧИХ ПЛАТФОРМ ДЛЯ ВИВЧЕННЯ ІНОЗЕМНИХ МОВ 4

Таблиця 1 – Порівняння популярних сервісів для вивчення іноземних мов

Сервіс	Основні можливості	Переваги	Недоліки
Duolingo	Вправи, тести, переклад, гейміфікація	Безкоштовний, простий інтерфейс	Обмежене пояснення граматики
Babbel	Структуровані уроки, аудіо, діалоги	Чітка система навчання	Переважно платний доступ
Busuu	Курси, перевірка носіями мови, тестування	Соціальна взаємодія користувачів	Частина функцій у преміум
Memrise	Лексика, відео з носіями, повторення слів	Ефективне запам'ятовування слів	Менше уваги граматиці

Порівняльний аналіз показав, що жоден із розглянутих сервісів не інтегрований у середовище месенджерів. Розробка Telegram-бота усуває цей недолік, забезпечуючи навчання безпосередньо у звичному застосунку.

КЛАСИФІКАЦІЯ ЧАТ -БОТІВ ЗА ПРИНЦИПОМ РОБОТИ 5

Рисунок 1 – Класифікація чат-ботів за принципом роботи

ВИМОГИ ДО ПРОГРАМНОГО ЗАСТОСУНКУ 9

Таблиця 3 – Функціональні вимоги до Telegram-бота для вивчення іноземних мов

№	Вимога	Опис
1	Тестування знань	Система питань з вибором відповіді A/B/C, організована за мовою та темою
2	Перевірка відповідей	Миттєва перевірка та відображення правильного варіанту при помилці
3	Вибір параметрів	Мова навчання, тема та рівень складності (Easy, Medium, Hard)
4	Нагадування	Автоматичні сповіщення за розкладом для регулярного навчання
5	Адміністрування	Додавання мов, тем та питань через інтерфейс бота без коду

Таблиця 4 – Нефункціональні вимоги до Telegram-бота для вивчення іноземних мов

№	Вимога	Опис
1	Доступність	Робота 24/7 без перерв; можливість навчання у будь- який зручний час
2	Швидкодія	Час відповіді бота на команду — не більше 2 секунд
3	Масштабованість	Підтримка великої кількості одночасних користувачів
4	Зручність	Інтуїтивна навігація через Inline -кнопки без необхідності вводу команд

ДЕМОНСТРАЦІЯ ІНТЕРФЕЙСУ TELEGRAM -БОТА 10

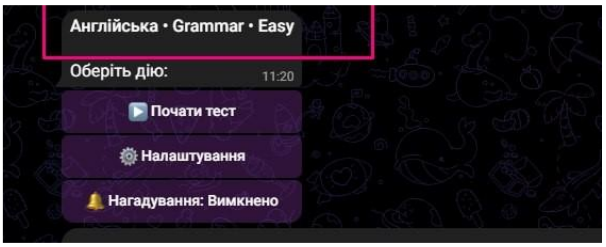

Рисунок 4 – Головне меню Telegram-бота

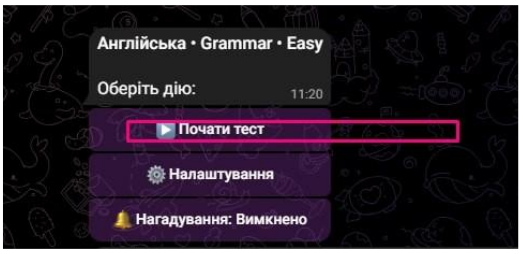

Рисунок 5 – Відображення поточних параметрів навчання

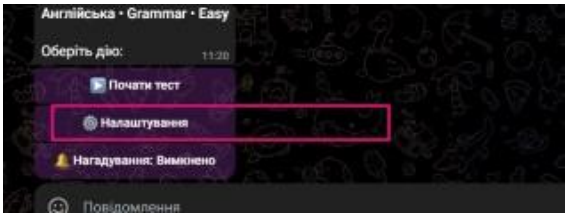

Рисунок 6 – Інтерфейс запуску тестування

ДЕМОНСТРАЦІЯ ІНТЕРФЕЙСУ TELEGRAM -БОТА 11

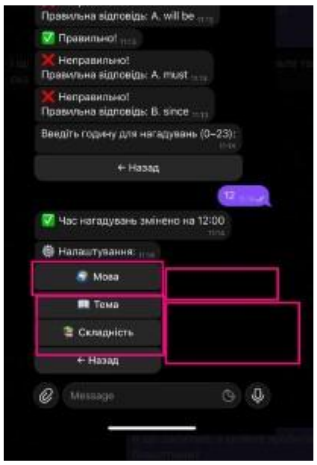

Рисунок 7 – Меню налаштувань бота

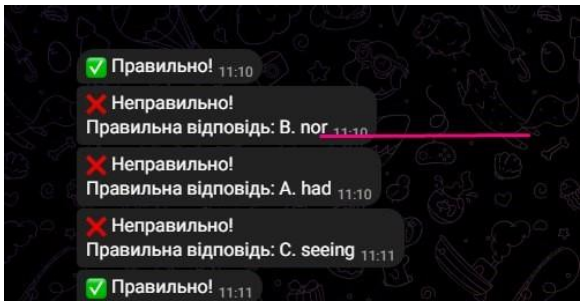

Рисунок 8 – Відображення результатів тестування

ТЕХНОЛОГІЧНИЙ СТЕК СИСТЕМИ 12

Таблиця 5 – Технології та бібліотеки, використані при розробці Telegram-бота

Компонент	Технологія / Бібліотека	Призначення
Мова розробки	Python 3.11+	Основна мова реалізації застосунку
Telegram-фреймворк	aiogram 3.x	Асинхронна взаємодія з Telegram Bot API
База даних	SQLite + aiosqlite	Зберігання питань, тем, налаштувань, користувачів
Планувальник	APScheduler (CronTrigger)	Система автоматичних нагадувань за розкладом
Управління станом	FSM aiogram	Сценарії взаємодії з користувачем (стани діалогу)
Платформа	Telegram Bot API	Середовище для взаємодії з кінцевим користувачем
Інтерфейс	InlineKeyboardMarkup	Кнопкова навігація без введення текстових команд

Структура програмних модулів:

Обробка запитів: Приймає повідомлення від Telegram API та маршрутизує до обробників

Бізнес-логіка: Генерація тестів, перевірка відповідей, управління навчальним процесом

Доступ до даних: CRUD-операції з SQLite через aiosqlite (async)

Моніторинг: APScheduler— планування нагадувань; FSM— управління станами

ВИСНОВКИ 13

1. Проаналізовано сучасні підходи до вивчення іноземних мов за допомогою цифрових технологій і встановлено доцільність використання Telegram як освітньої платформи
2. Досліджено функціональні можливості Telegram Bot API та обґрунтовано вибірPython/aiogram як технологічного стеку для розробки
3. Сформовано функціональні та нефункціональні вимоги та розроблено модульну архітектуру системи
4. Реалізовано систему тестування з генерацією питань за мовою, темою та рівнем складності (Easy/Medium/Hard)
5. Розроблено конфігуратор навчального процесу та систему автоматичних нагадувань на основі APScheduler
6. Реалізовано адміністративний модуль для додавання мов, тем і питань безпосередньо через інтерфейс бота
7. Проведено тестування та підтверджено практичну цінність застосунку як масштабованого освітнього рішення

Апробація:
1.ІІІ всеукраїнська науково-технічна конференція «Технологічні горизонти: дослідження та застосування інформаційних технологій для технологічного прогресу України і світу на тему « Гейміфікація як засіб підвищення мотивації учнів та студентів»
2. VII Науково -технічна конференція «Сучасний стан та перспективи розвитку IoT» на тему «Застосування чат -бот технологій у мовній освіті: розробка telegram-бота для вивчення іноземних мов засобами python»

ДЯКУЮ ЗА УВАГУ!