Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інтелектуальних інформаційних технологій та автоматизації
(повне найменування інституту, назва факультету (відділення))

Кафедра автоматизації та інтелектуальних інформаційних технологій
(повна назва кафедри (предметної, циклової комісії))

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка багатофункціонального телеграм боту для вивчення англійської мови з використанням сторонніх API»

Виконав: студент 2 курсу, групи 1ІСТ-21м спеціальності 126 – Інформаційні системи та технології
(шифр і назва спеціальності)

Керівник: к.т.н., доцент кафедри АІІТ Богач І. В.
(прізвище та ініціали)
«____» _____2022 р.

Опонент: д.т.н., доцент кафедри КСУ Ковтун В.В.
(прізвище та ініціали)
«____» _____2022 р.

Допущено до захисту
Завідувач кафедри АІІТ
_____О.В.Бісікало
(прізвище та ініціали)

«____»_____2022 року

Вінниця ВНТУ - 2022 року

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій
Рівень вищої освіти 2-й (магістерський)
Галузь знань – 12 Інформаційні технології
Спеціальність 126 – Інформаційні системи та технології
Освітньо-професійна програма - Інформаційні технології аналізу даних та зображень

ЗАТВЕРДЖУЮ
Завідувач кафедри АІІТ
О.В.Бісікало
(прізвище та ініціали)

«____» _____2022 року

З А В Д А Н Н Я
НА МАГІСТРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Семенюку Андрію Васильовичу
(прізвище, ім'я, по батькові)

1. Тема роботи: «Розробка багатофункціонального телеграм боту для вивчення англійської мови з використанням сторонніх API»
керівник роботи:
Богач Ілона Віталіївна, к.т.н., доцент кафедри АІІТ
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від «__»___ 2022 року №__

2. Строк подання студентом роботи 10.12.2022

3. Вихідні дані до роботи: стек технологій для розробки продукту; загальновідомий месенджер «Telegram» та його загальнодоступний API; Перелік API для отримання необхідних даних;

4. Зміст текстової частини: Вступ; Дослідження предметної галузі; Вибір технологічного стеку та дата-провайдерів; Розробка програмного забезпечення; Тестування створеного функціоналу та порівняння з аналогами; Економічна частина. Висновки. Список використаних джерел.

5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень): UML-діаграма роботи RESTful API; Діаграма компонентів чат-боту; Діаграма послідовності роботи чат-боту; UML-діаграма навігації Start Scene; UML-діаграма навігації Schedule Scene; UML-діаграма навігації Level Scene; UML-діаграма навігації Main Scene; UML-діаграма навігації Word Scene; UML-діаграма навігації Shorts Scene; UML-діаграма навігації Text Scene.

6. Консультанти розділів роботи

Розділ змістової частини роботи	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Розділи 1, 2, 3, 4	доцент кафедри АІІТ, доцент, Богач І. В.		
Економічний розділ	професор кафедри ЕПВМ, професор, к.е.н. Лесько О.Й.		

7. Дата видачі завдання «____»_____2022 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Дослідження предметної галузі	12.10.2022	
2	Вибір технологічного стеку та дата-провайдерів	19.10.2022	
3	Розробка програмного забезпечення	20.11.2022	
4	Тестування створеного функціоналу та порівняння з аналогами	30.11.2022	
5	Підготовка економічної частини	03.12.2022	
6	Оформлення пояснювальної записки, графічного матеріалу і презентації	06.12.2022	
7	Апробація результатів дослідження	07.12.2022	
8	Публікації	09.12.2022	
9	Графічні матеріали	12.12.2022	
10	Захист МКР	21.12.2022	

Студент _____ _____Семенюк А.В._
(підпис) (прізвище та ініціали)

Керівник роботи _____ _____Богач І.В.__
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

УДК 004.512.2

Семенюк А.В. Розробка багатофункціонального телеграм боту для вивчення англійської мови з використанням сторонніх API. Магістерська кваліфікаційна робота зі спеціальності 126 – Автоматизація та комп'ютерно-інтегровані технології, освітня програма – Інформаційні технології аналізу даних та зображень. Вінниця: ВНТУ, 2022. 132 с.

На укр. мові. Бібліогр.: 31 назв; рис.: 39; табл.11.

Магістерська кваліфікаційна робота присвячена розробці телеграм бота(на базі месенджера «Telegram»), що має на меті збільшити лексичний запас англійських слів, витренувати та покращити вимову вивчених слів. Бот включає в себе функціонал комфортного інтерфейсу користувача, інтеграцію з різними API, що будуть грати роль дата провайдерів та інтеграцію з базою даних для відслідковування даних користувача. Використано власний інструментарій месенджера для розробки бота.

Також в роботі присутній детальний огляд подібних систем зі схожим функціоналом, аналізуються їх ключові особливості, переваги та недоліки. Для програмного забезпечення, що буде розроблене, підбирається технологічний стек та описується архітектура. На основі актуальних та доцільних технологій розробки та у відповідності з існуючими дизайн макетами створюється кодова база додатку.

Метою даного бота є демонстрація та використання принципово нового методу навчання за допомогою месенджера, що надає функціонал обміну повідомленнями. На основі проведеного аналізу вже наявних методів та платформ для навчання зроблено та обґрунтовано вибір месенджера «Telegram» та проведено аналогію з іншими платформами для навчання.

Ключові слова: чат-бот, Telegram, навчання, база даних, API.

ABSTRACT

UDC 004.512.2

Semeniuk A.V. Development of a multifunctional Telegram bot for learning English using third-party APIs. Master's thesis on specialty 126 - Automation and computer-integrated technologies; educational program - Information technologies of data and image analysis. Vinnytsia: VNTU, 2022. 132 p.

In Ukrainian language. Bibliogr.: 31 titles; Fig.: 39; table 11.

The master's thesis is devoted to the development of a telegram bot (based on the Telegram messenger), which aims to increase the vocabulary of English words, practice and improve the pronunciation of learned words. The bot includes the functionality of a comfortable user interface, integration with various APIs that will play the role of data providers, and integration with a database for tracking user data. The messenger's own toolkit was used to develop the bot.

The work also includes a detailed review of similar systems with similar functionality, their key features, advantages and disadvantages are analyzed. For the software to be developed, the technology stack is selected and the architecture is described. On the basis of current and appropriate development technologies and in accordance with existing design layouts, the code base of the application is created.

The purpose of this bot is to demonstrate and use a fundamentally new method of learning with the help of a messenger that provides messaging functionality.

Based on the analysis of already existing methods and platforms for training, the choice of the Telegram messenger was made and justified, and an analogy with other platforms for training was made.

Keywords: chatbot, Telegram, training, database, API.

ЗМІСТ

ВСТУП

Актуальність теми. Важко представити сучасну людину у нинішніх реаліях без розумних гаджетів. Смартфон, лептоп, планшет – один із цих девайсів ми, беззаперечно, використовуємо щоденно, при чому витрачаючи у ньому значну частину нашого часу. Чому ми це робимо? Пояснення досить просте – можливості. Використовуючи умовний смартфон людина має змогу зробити неймовірну кількість речей: від створення заміток щодо розкладу дня і аж до оформлення замовлення на придбання власного авто. Глобальна павутина почала пронизувати собою всі сфери людської діяльності: бізнес, навчання, комунікації, промисловість тощо.

Зокрема онлайн комунікація - напрямок, що останнім часом має досить стрімкий розвиток. Нам зручно використовувати онлайн месенджери для миттєвого зв'язку з іншими людьми. Зазвичай для цього використовуються спеціальні додатки чи платформи – месенджери або соціальні мережі, метою яких є встановлення зв'язку між «користувачем А» та «користувачем Б».:

Трішки аналітики. Дані згідно досліджень агентства «WeAreSocial». У світі налічується понад 5,19 млрд користувачів мобільних телефонів[20], їх кількість невпинно зростає. Переважна більшість цих телефонів має змогу підключитися до мережі Інтернет. Що ж стосується самої мережі, то маємо наступне: на початок 2020 року інтернетом користувалися 4,54 млрд осіб, або майже 60% населення планети[20].

Доступ до мережі Інтернет перетворює телефон із простого засобу голосового(переважно) зв'язку в девайс типу «все в одному». Як наслідок цього ми виділяємо все більше часу на взаємодію з ним. По статистиці, якщо брати середній показник, то можливо заявити, що людина проводить в інтернеті 6 годин 43 хвилини на добу, це приблизно 40% її добової активності.

Статистика щодо месенджерів або соціальних мереж свідчить про наступне: месенджери і соцмережі – найпопулярніші категорії додатків, у них сидить 89% інтернет-користувачів у віці 16–64 років[20, 30]. Після них йдуть

додатки для шопінгу (66%), карти і розважальні програми (65%), музика (52%) та ігри (47%). На соцмережі і месенджери люди витрачають найбільше власного часу. При врахуванні і смартфонів, і ПК, то в середньому на соцмережі витрачають 2 години 24 хвилини в день[13]. Кожного місяця аудиторія соцмереж та месенджерів досягає планки в 4 млрд користувачів. Ці цифри лише доводять те, що акаунтом у соцмережах володіє 53% населення Землі. Про це йдеться у звіті «Digital 2020».

У зв'язку з такою шаленої популярністю розробники соціальних мереж та месенджерів почали запроваджувати таку технологію як «чат-бот». Чат-бот (від англ. chatbot) – це програма, яка що веде прямо комунікацію з користувачем й імітує реальну розмову за допомогою текстових або аудіо/відео повідомлень на сайтах, по телефону, в додатках і месенджерах.

Власники платформ, сервісів, з напрямками B2C(бізнес-споживач) , B2B (бізнес-бізнес) все частіше використовують віртуальних помічників чат-ботів для виконання простих завдань[2]. Додавання помічників чат-ботів зменшує накладні витрати, ефективніше використовує час персоналу служби підтримки та дає змогу організаціям обслуговувати клієнтів у години , коли живі агенти недоступні. Організації, які прагнуть збільшити продуктивність продажів або послуг, можуть використовувати чат-боти для економії часу та ефективності, оскільки чат-боти зі штучним інтелектом (AI) можуть спілкуватися з користувачами та відповідати на повторювані запитання[3].

Також останнім часом набуває популярності навчання на основі чат ботів. Живий час стимулює людей повертатись до навчання знову і знову. Освітні чат-боти блискуче змінюють спосіб взаємодії навчальних закладів зі своїми учнями. Вони працюють над тим, щоб полегшити студентам навчання та охопити всі види діяльності, якими вони можуть займатися протягом курсу навчання[4]. Досить актуальною є проблема залучення студентів. Серед віртуального навчання під час пандемії залучення студентів скоротилося до мінімуму. Взаємодія по моделі «учень-викладач» і взаємодія «учень-учень» відійшли на другий план. Вчителі намагаються пояснити якнайкраще аби учні

отримували знання та розуміли предмети. Чат-боти можуть допомогти в цьому сценарії, постійно взаємодіючи зі студентами та миттєво з’ясовуючи їхні запити. Студенти можуть скористатися їхньою допомогою під час і після занять і переконатися, що вони не шкодять навчанню через віртуальну платформу[4].

Трішки статистики: очікується, що до 2026 року глобальне електронне навчання зростатиме на 9,1% на рік. Люди обирають дистанційне корпоративне навчання та курси, щоб їм не потрібно було відриватися від роботи та сім’ї для підвищення кваліфікації.

Метою роботи є покращення доступного навчального функціоналу для вивчення англійської мови на основі месенджера «Telegram». На початкових етапах розробки планується функціонал перекладу слів, навчання за обраним розкладом, конфігурація рівня складності та ведення статистики користувача. Частина вище перелічених функцій можуть бути реалізовані за допомогою сторонніх API сервісів

Для досягнення мети необхідно розв'язати наступні задачі:

1. Провести аналіз існуючих подібних платформ навчального напрямку та аналіз наявних корисних API, що можуть бути використанні в власному боті.

2. Проаналізувати технологічний стек для розробки боту та обрати допоміжні елементи для розробки.

3. Розробити клієнт-частину (бот функціонал) додатку за допомогою обраного технологічного стеку, інтегрувати клієнт-частину з серверною частино.

4. Створити базу даних, створити необхідне наповнення БД.

5. Створити сервер-частину та інтегрувати її базою даних.

6. Протестувати програмне забезпечення та порівняти з аналогами.

Об'єктом дослідження є методи взаємодії користувача з сервісом.

Предметом дослідження є технології та методи забезпечення інформацією та реалізації чат-боту.

Методи дослідження. У роботі застосовано наступні методи дослідження: аналіз, прогнозування результатів, моделювання системи, класифікація наявних сутностей, спостереження за результатами розробки та їх коригування, узагальнення проведених робіт.

Наукова новизна роботи полягає в поєднанні популярного месенджера «Telegram» з функціоналом, що має навчальний напрямок і на відміну від існуючих аналогів дозволяє мати більш простий доступ до навчальних ресурсів, користуватись уже звичним інтерфейсом, отримувати повідомлення з навчальним матеріалом, оминути зайвих кроків перед початком навчання (авторизація, підтвердження, пошук останнього уроку) та проводити навчання за комфортним користувачу розкладом.

Практична цінність роботи полягає в створенні боту, що є зручною платформою для систематизації вивчення певного матеріалу. Користувач має можливість налаштувати навчання під свій графік та можливості.

Апробація результатів роботи. Роботу представлено на Всеукраїнській науково-практичній Інтернет-конференції студентів, аспірантів та молодих науковців «Молодь в науці: дослідження, проблеми, перспективи (МН-2023)»[1] та подано документи на отримання свідоцтва про авторське право.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ГАЛУЗІ

Чат-бот — це програмне забезпечення або комп'ютерна програма, яка імітує людську розмову за допомогою текстових або голосових взаємодій.

1.1 Дослідження технології чат ботів

1.1.1 Типізація чат ботів за станом та структуризацією

Чат-боти мають різні рівні складності: без стану або без стану . Чат-боти без стану підходять до кожної розмови так, ніби до взаємодії з новим користувачем. Навпаки, чат-боти зі збереженням стану можуть переглядати минулі взаємодії та формувати нові відповіді в контексті.

Люди, що впроваджують чат боти у власні сервіси також повинні вирішити, який тип розмов вони підтримувати - структуровані чи неструктуровані. Чат-боти, що були створені для структурованих розмов, мають велику кількість сценаріїв, що спрощує програмування, але обмежує запити користувачів. У середовищі B2B чат-боти зазвичай створені для відповідей на поширені запитання або виконання простих завдань, що повторюються. Наприклад, чат-боти можуть дозволити торговим представникам швидко отримувати номери телефонів.

При використанні неструктурованих чат-ботів є необхідність розпізнавати контекст розмови і вже відштовхуючись від нього направляти користувача до потрібної категорії відповідей. Це потребує значно більших ресурсів та імплементацій ніж структурований аналог. Прикладом неструктурних чат-ботів є боти з використанням штучного інтелекту для розпізнавання контексту мовлення. Вони широко використовуються організаціями, які прагнуть збільшити продуктивність продажів або послуг, оскільки такі боти можуть як спілкуватися з користувачами відповідаючи на

повторювані запитання, так і взаємодіяти з користувачами, які відходять від традиційних форм спілкування.

1.1.2 Типізація чат ботів за принципами роботи

Чат боти типізують за принципами роботи:

- зі сценарієм або швидкими відповідями;
- на основі розпізнання ключових слів;
- гібридні;
- контекстні чат-боти;
- голосові.

Розглянемо кожен з них більш детальніше.

Зі сценарієм або швидкими відповідями - як найпростіші чат-боти, вони діють як ієрархічне дерево рішень . Ці боти взаємодіють з користувачами за допомогою попередньо визначених запитань, які просуваються, поки чат-бот не відповість на запитання користувача.

Подібним до цього бота є чат-бот на основі меню, який вимагає від користувачів робити вибір із попередньо визначеного списку або меню, щоб надати боту глибше розуміння того, що потрібно клієнту.

На основі розпізнавання ключових слів - ця категорія ботів намагаються слухати те, що вводить користувач, і відповідати відповідним чином, використовуючи ключові слова з відповідей клієнтів. Цей бот поєднує настроюванні ключові слова та штучний інтелект, щоб відповідати належним чином. На жаль, цим чат-ботам важко використовувати повторювані ключові слова або зайві запитання.

Гібридні - поєднують елементи ботів на основі меню та ботів на основі розпізнавання ключових слів. Користувачі можуть вибрати прямі відповіді на свої запитання або скористатися меню чат-бота, щоб зробити вибір, якщо розпізнавання ключових слів не є ефективним.

Контекстні - ці чат-боти одними із найскладніших, вони вимагають орієнтації на дані. Ця категорія використовує штучний інтелект і машинне навчання, щоб запам’ятовувати розмови та взаємодію користувачів і використовує попередні дані для вдосконалення з часом. Замість того, щоб покладатися на ключові слова, ці боти використовують те, що запитують клієнти, і те, як вони запитують, щоб надавати відповіді та самовдосконалюватися.

Голосові - за цим типом чат-ботів майбутнє цієї технології. Чат-боти з підтримкою голосу використовують голосовий діалог від користувачів як введення, яке спонукає до відповідей або творчих завдань. Розробники створюють цю категорію ботів за допомогою API перетворення тексту в мовлення та розпізнавання голосу : Amazon Alexa, Siri та Google Voice.

1.1.3 Цільова аудиторія

Цільова аудиторія поділяється на

- інтернет-магазини;
- обслуговування клієнтів;
- віртуальні помічники;
- навчальні заклади та платформи.

Розглянемо кожну з них більш детальніше.

Інтернет-магазини - відділи продажів можуть використовувати чат-ботів, щоб відповідати на нескладні запитання про продукт або надавати корисну інформацію, яку споживачі можуть шукати в будь-яку пору доби, зокрема ціну доставки та наявність.

Відділи обслуговування також можуть використовувати чат-ботів, щоб допомогти агентам колл-центру відповідати на повторювані запити. Наприклад, клієнт може надати чат-боту номер замовлення та запитати, коли

замовлення буде відправлено. Якщо розмова стає надто складною - чат-бот передає повідомлення клієнта в службу підтримки.

Чат-боти також можуть виконувати роль віртуальних помічників. Apple, Amazon, Google і Microsoft мають різні форми віртуальних помічників. Додаток Siri від Apple, Cortana від Microsoft або Google Home - усі відіграють роль особистого чат-бота.

Люди мають звичку проводити багато часу в месенджерах без цільової потреби, тому звичний для них інтерфейс, легка й невимушена комунікація де-факто заставляють користувача повертатись до навчання повторно у найкоротші проміжки часу.

1.1.4 Переваги та недоліки технології

До загальних переваг технологій відносять:

- Ведення багатьох діалогів одночасно; чат-боти можуть спілкуватися одночасно з тисячами покупців. Це підвищує продуктивність бізнесу та скорочує час очікування.

- Економічна ефективність; чат-бот — це швидша та дешевша одноразова інвестиція, що є куди доцільнішою ніж наймання додаткових співробітників. Крім того, чат-боти можуть зменшити кількість дороговартісних проблем, що були спричинені помилками людини. Витрати через відтік користувачів також зменшуються завдяки здатності чат-бота відповідати протягом кількох секунд.

- Економія часу; чат-боти можуть автоматизувати завдання, які виконуються часто та в певний час. Це дає працівникам час зосередитися на більш важливих завданнях дає змогу клієнтам не чекати на отримання простих відповідей.

- Проактивна взаємодія з клієнтами. Ще не так давно бізнес покладався лише на пасивну взаємодію з клієнтами та чекав, поки покупці звернуться

першими. За допомогою чат-ботів бізнес може активно взаємодіяти, оскільки боти можуть ініціювати розмови та контролювати, як клієнти використовують веб-сайти та цільові сторінки. Як результат - організації можуть використовувати інформацію, зібрану в результаті моніторингу, щоб пропонувати конкретні стимули покупцям, допомагати користувачам орієнтуватися на сайті та відповідати на майбутні запитання.

- Збір аналітичних даних. Чат-боти збирають відгуки про кожну взаємодію, щоб допомогти компаніям покращити їхні послуги та продукти або оптимізувати свої веб-сайти. Боти також можуть записувати дані користувачів, щоб відстежувати поведінку та моделі покупок. Ця інформація може запропонувати організаціям уявлення про те, як краще продавати свої продукти та послуги, а також про типові перешкоди, з якими клієнти стикаються під час процесу купівлі.

- Розширення клієнтської бази. Чат-боти можуть покращити процес формування потенційних клієнтів. Чат-боти можуть задавати запитання протягом усього шляху покупця та надавати інформацію, яка може переконати користувача та створити потенційного клієнта. Потім чат-боти можуть надавати інформацію про потенційних клієнтів команді продажів, яка може взаємодіяти з потенційними клієнтами. Боти можуть підвищити коефіцієнт конверсії та забезпечити, щоб потенційний клієнт рухався в правильному напрямку – до покупки.

До загальних недоліків відносять:

- Безпека. Користувачі повинні достатньо довіряти чат-боту, щоб ділитися особистими даними. Тому організації повинні переконатися, що вони розробляють свої чат-боти, щоб запитувати лише відповідні дані та безпечно передавати ці дані через Інтернет. Чат-боти повинні мати безпечний дизайн і бути здатними запобігти доступу хакерів до інтерфейсів чату. Чат-боти можуть спілкуватися одночасно з тисячами покупців. Це підвищує продуктивність бізнесу та скорочує час очікування.

- Різноманітність лексичної поведінки. Користувач може використовувати сленг, слова з помилками або абревіатури. На жаль, навіть штучний інтерфейс має свої обмеження і не здатний повністю вирішити цю проблему.

- Різноманітність способів донесення потреби. Навіть одне слово, що має декілька значень та використовується у запиті користувача може призвести до неправильно зрозумілих намірів. Чат-боти повинні обробляти як довгі, так і короткі речення, а також «бабл» чат, де користувач описує потреби використовуючи одне або декілька слів в одному повідомленні.

- Непередбачуваність, настрої та емоції людини. Природа людини дозволяє їх різко та кардинально змінити думку за лічені секунди. Після довгого слідування ієрархією бота, людина може вмить захотіти повернутися на декілька рівнів вище. Чат-боти повинні адаптуватися до цієї випадковості та спонтанності та розуміти їх.

- Обмеженість функціоналу. Який би “розумний” бот ви не використовували, в будь-якому випадку у нього є свій «ліміт» можливостей. В більшості випадків бот має змогу надати лише базову інформаційну консультацію, забезпечити користувача інформацією про наявність, допомогти відстежити доставку, уточнити адресу, щось записати або відтворити.

1.2 Дослідження існуючих ресурсів для вивчення англійської мови

1.2.1 Words

Принцип роботи Words досить стандартний: вибираємо тему, слова за якою хотіли б вивчити, і починаєте заняття. Вивчивши тему, будуть доступні тренування з картками, де необхідно буде вгадувати переклад слова, збирати

мовні пари, перекладати слова в аудіоформаті та збирати слова по літерах. Вигляд додатку наведено на рисунку 1.1.

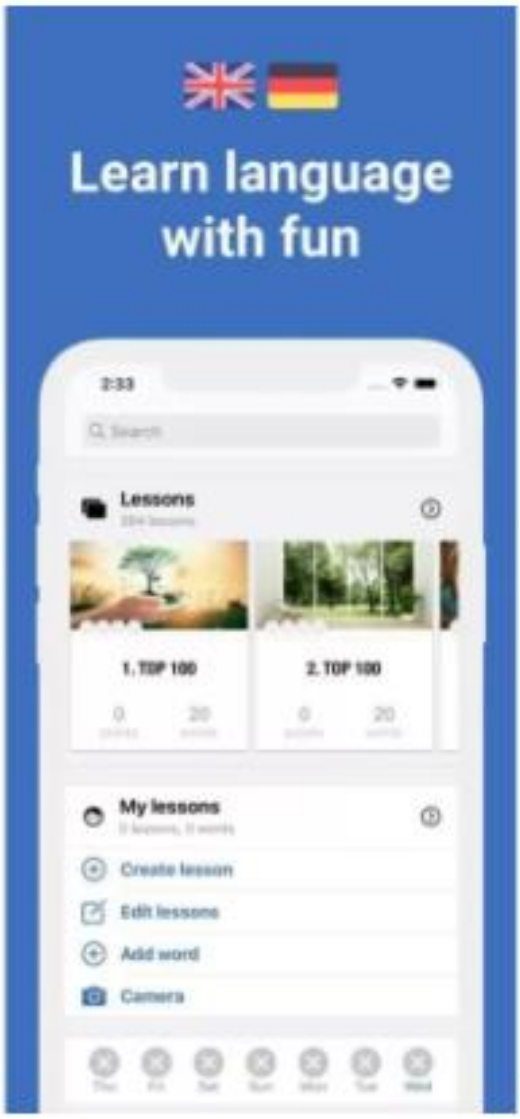

Рисунок 1.1 – Додаток «Words»

Перевагами додатку «Words» є:

- наявність власного додатку;
- велика варіація тем для вивчення слів;
- відтворення слів за допомогою аудіо
- збереження тем, що сподобались у закладки
- інтерактивні тести.

Недоліками додатку «Words» є:

- нав’язлива монетизація;

- незрозумілий багатьом користувачам інтерфейс;
- підвисання додатку при першому старті;
- відсутність української мови.

Останній у переліку недолік є ключовим для багатьох людей, які бажають вчити англійську мову маючи переклад незрозумілих їм слів українською.

1.2.2 Rosetta Stone

Якісний додаток для вивчення англійських слів. Безкоштовно можна проходити невеликі уроки, проте, якщо Ви бажаєте більшого і якщо Вам хочеться навчатися частіше, слід сплатити за платну підписку. Цей додаток тренує всі аспекти мови: аудіювання, читання, говоріння та письмо. На початку Вашого шляху вивчення англійської мови Вам буде запропоновано почати з карток, де спікер буде говорити слово, Ваше завдання – це слово повторити[5]. Також у додатку реалізована система карток, де Вам потрібно вибрати правильне зображення та слово, відповідно до того, що сказав спікер. Вигляд додатку наведено на рисунку 1.2.

Перевагами додатку «Rosetta Stone» є:

- тренування промови;
- система тестування за допомогою карток;
- відтворення слів за допомогою аудіо;
- вибір рівня складності.

Недоліками додатку «Rosetta Stone» є:

- базовий безкоштовний функціонал;
- проблеми з розпізнавання запису слова;
- проблеми з авторизацією додатку;
- відсутність української мови.

Подібно до переднього додатку ключовим мінусом є відсутність української мови. Ще один вагомий мінус – занадто обмежений безкоштовний функціонал додатку.

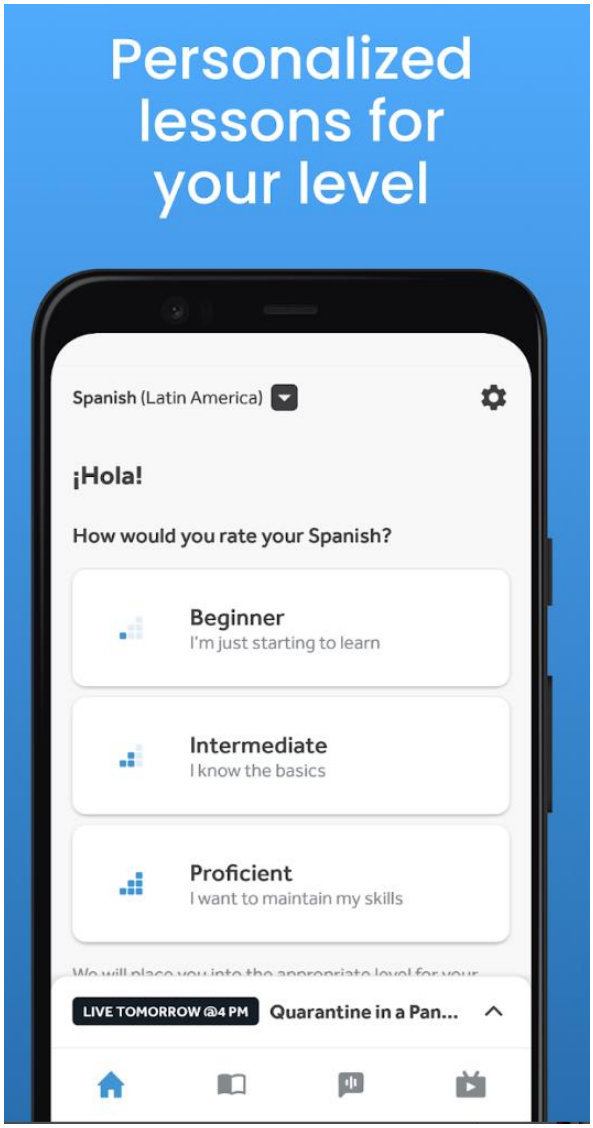

Рисунок 1.2 – Додаток «Rosetta Stone»

1.2.3 Memrise

Найкращий спосіб запам’ятовувати слова в англійській – це читати їх, прописувати і, звичайно ж, постійно слухати і повторювати. Програма для вивчення англійської мови Memrise[5] побудована на принципі інтервального повторення (повторення вже вивченого матеріалу з певною періодичністю).

Більше того, працюючи в цьому додатку, Ви матимете можливість прослуховувати безліч разів одні й ті самі слова, вимовлені різними людьми за різних обставин, що дозволить Вам сприймати на слух різні акценти. Приклад вигляду даного додатку наведено на рисунку 1.3.

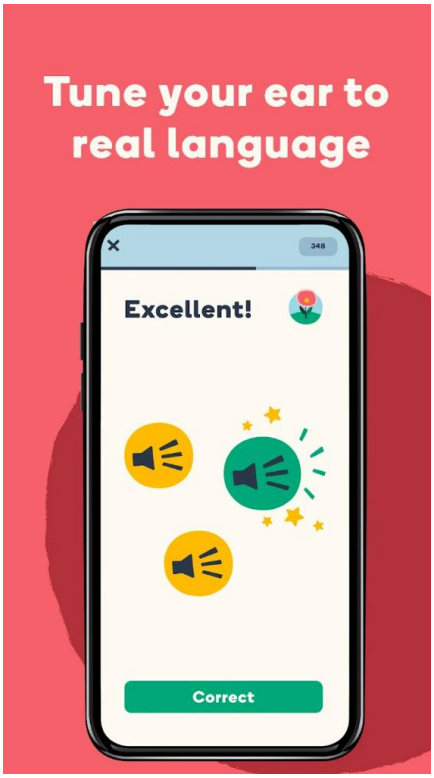

Рисунок 1.3 – Додаток «Memrise»

Перевагами додатку «Memrise» є: тренування за допомогою аудіо та відео; система тестування за допомогою карток; вибір рівня складності.

Недоліками додатку «Memrise» є базовий безкоштовний функціонал та відсутність української мови.

1.2.4 Andy Bot

Цікавий та багатофункціональний бот. Він може без проблем підтримувати просту розмову та ставити запитання. Деякі заняття вимагають встановлення програми. Але, наприклад, напрацьовувати словниковий запас і

проходити вправи з граматики можна у Telegram[6]. Особливо цікавою нам здалася гра «Опиши емодзі». Бот дає емодзі, а вам потрібно написати, що це таке. Бот веде статистику ваших успіхів. Він покаже, скільки днів ви вже займаєтесь, скільки нових слів вивчили за останній тиждень та весь час навчання, а також загальну кількість пройдених уроків з граматики. Як додатковий інструмент навчання Andy bot підійде відмінно. У налаштуваннях бота ви можете вказати інтенсивність тренувань. Є кілька варіантів: кожен день, кожні 2-3 дні та раз на тиждень[6]. Залежно від обраної інтенсивності бот надсилатиме вправи на граматику та нові слова для вивчення. Приклад комунікації з ботом наведено на рисунку 1.4.

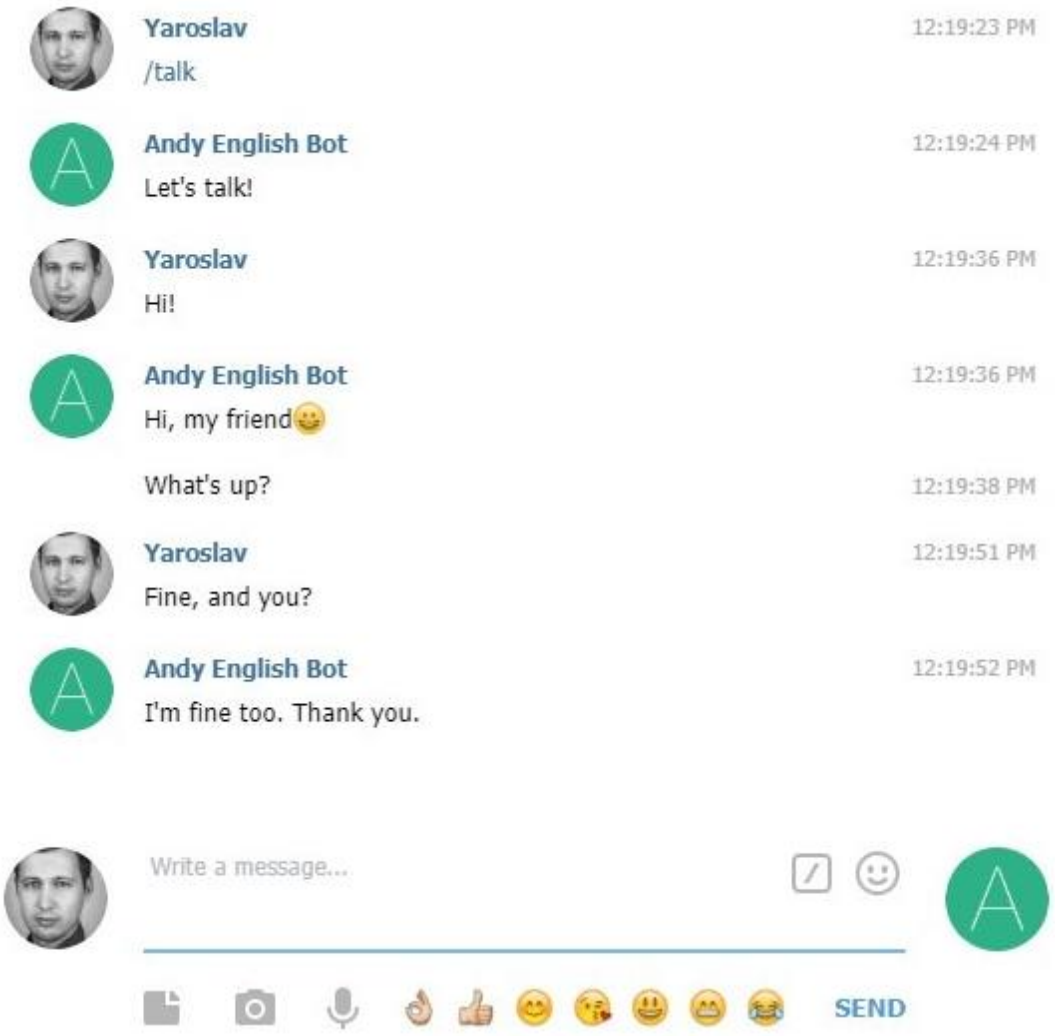

Рисунок 1.4 – Бот «Andy Bot»

Перевагами боту «Andy Bot» є:

- ведення діалогу англійською;
- система тестування за допомогою карток;
- автоматичні повідомлення.

Недоліками боту «Andy Bot» є:

- відсутність розкладу повідомлень:
- відсутність конфігурування кількості слів за день;
- відсутність української мови.

1.2.5 Memoize

Дуже цікавий бот, який дозволяє вивчати слова прямо в контексті їхнього використання. Бот надає вибір слів, з яких потрібно вибрати одне. Після виводить список фільмів та мультфільмів, де використовується це слово. Ви маєте змогу вибирати відео фрагмент із вибраного фільму або мультика, і таким чином Ви вивчаєте лексику. Наприклад, ми вибрали слово «gun» та у списку фільмів клікнули на «Термінатор». В результаті отримали 5-секундний фрагмент відео , де використовується це слово[6]. Приклад комунікації з ботом наведено на рисунку 1.5.

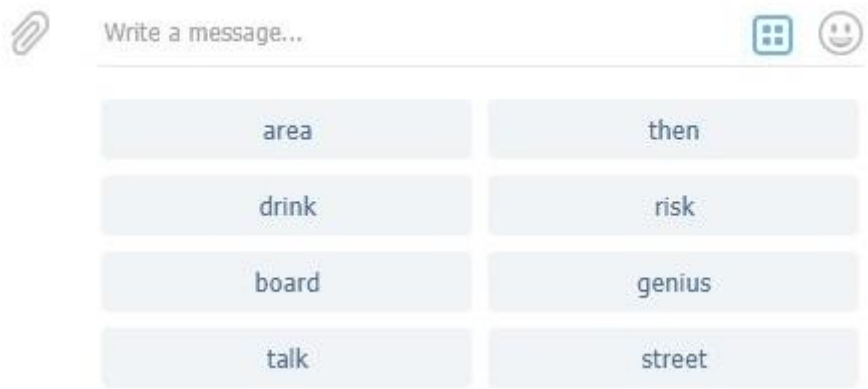

Рисунок 1.5 – Бот «Memoize»

Перевагами боту «Memoize» є:

- велика база відео фрагментів;
- зручний вибір слів.

Недоліками боту «Memoize» є:

- відсутність можливості отримати переклад;
- відсутність можливості зберегти слово для подальшого вивчення;
- відсутність історії вивчення.

1.3 Висновки

Багато експертів очікують, що популярність чат-ботів зростатиме. У майбутньому штучний інтелект і машинне навчання продовжуватимуть розвиватися, пропонуючи нові можливості для чат-ботів і запроваджуючи нові рівні текстового та голосового досвіду користувачів. Подібні вдосконалення також можуть вплинути на збір даних і запропонувати глибшу інформацію про клієнтів, що призведе до прогнозованої поведінки бізнес користувачів технології та збільшить їх інвестиції у цю технологію.

Відповідно до проведеного вище аналізу є можливим виділити декілька основних та критичних недоліків сервісів для вивчення мови:

- базовий безкоштовний функціонал;
- відсутність української мови.

Якщо перший пункт залишається на розсуд користувача, то наступний є значним недоліком більшості таких сервісів.

2 ВИБІР ТЕХНОЛОГІЧНОГО СТЕКУ ТА ДАТА-ПРОВАЙДЕРІВ

2.1 JavaScript

JavaScript — це високорівнева інтерпретована мова програмування, що містить підтримку функціонального, імперативного та подієво-орієнтованого стилів. JavaScript - мова програмування з динамічною типізацією. На створення синтаксису цієї мови програмування вплинули такі мови як: C/C++ і Java [10]. JavaScript був офіційно представлений ще в 1995 році. Тоді JS швидко зайняв свої позиції, цьому посприяв розвиток технології AJAX - оновлення інформації на сторінці без перезавантаження вкладки браузера.

Завдяки ініціативам організації, що створює стандарти JS – ECMA, швидко вводяться більш сучасніші можливості та інструментарій.

ES6 (специфікація JS) — перетворює JavaScript в гнучку та виразну мову програмування. Синтаксис є досить простим для вивчення, а інтерпретатор «мовчки» сприймає великий перелік помилок розробника. Особливістю являється те, що JS має змогу підлаштовуватись під користувача, якщо присутнє відчуття неправильного чи неповного налаштування , то розробник зможе відносно просто налаштувати більш суворіші правила для написання більш чистого та читабельного коду, використовуючи різні інструменти. До прикладу, ESlint (перевірка стилю коду), Prettier — форматування коду. TypeScript — типізація[10]. Саме за допомогою такого інструментарію, JS з класами, декораторами, інтерфейсами, типізацією все більше набуває вигляду Java (у хорошому сенсі цього слова).

Розглянемо сфери застосування JavaScript.

Як зазначалося вище, JavaScript входить в число найпопулярніших мов програмування. Сумарно JavaScript (включно з CoffeeScript і TypeScript) користуються близько 16,4 млн розробників. Традиційно GitHub опублікував рейтинг найпопулярніших мов програмування. Лідером упродовж багатьох років незмінно залишається мова програмування JavaScript.

Варто також вказати об’єм даних, на яких базується цей рейтинг: кількість користувачів GitHub у 2022 році зросла на 30% [9] і досягла 94 млн, при цьому у 2020 році на платформі налічувалося 56 млн користувачів. Кількість зареєстрованих на GitHub репозиторіїв перевищила 340 млн. Рейтинг мов програмування за 2014-2022 роки можливо побачити на рисунку 2.1.

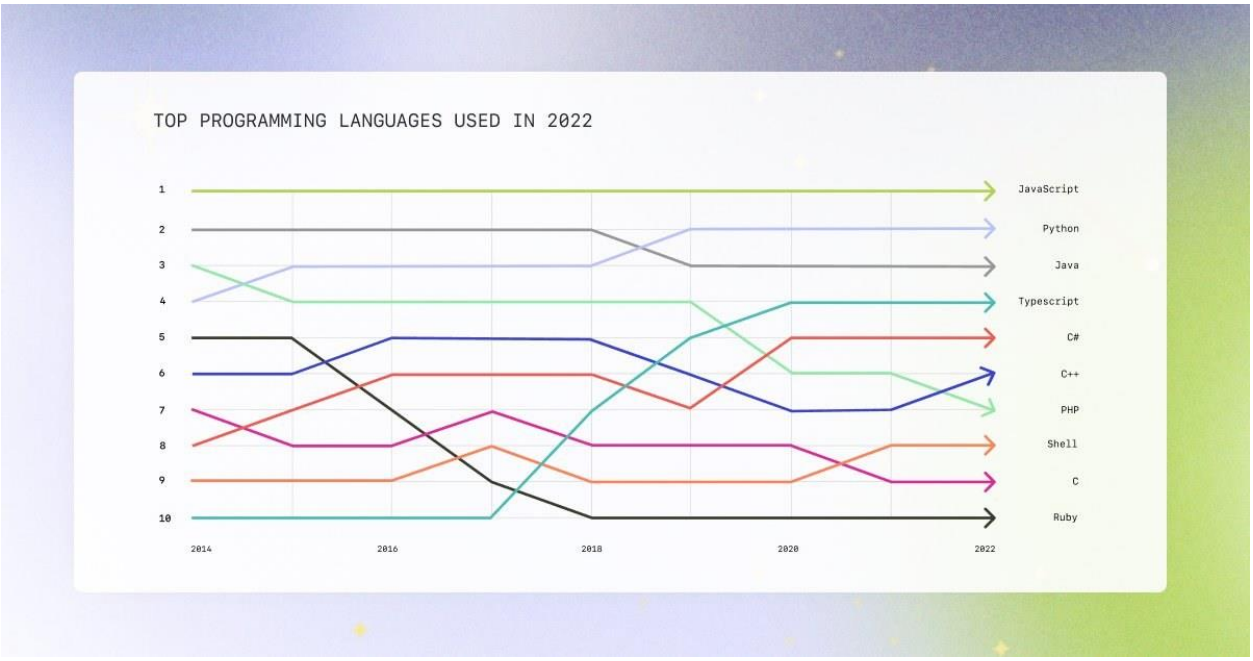

Рисунок 2.1 – Рейтинг популярності мов програмування 2014-2022

Розглянемо перспективи JavaScript.

Фреймворки JS, очевидно, на піку популярності. Два верхні рядки рейтингу StackOverflow за 2017 рік займають Node.js та AngularJS. React займає четверте місце, поступаючись .NET Core[12].

Це допомагає JavaScript мати одне з найактивніших спільнот. В Україні за JS та різними його фреймворками постійно проходять ком'юніті-мітапи і все частіше великі конференції. Регулярні зустрічі дозволяють ділитися досвідом досвідченим фахівцям та стають точкою входу до спільноти для новачків.

У Києві, де зосереджена найбільша кількість розробників, є одразу кілька регулярних мітапів: KyivJS , React Kyiv , NodeUA , AngularKyiv та

щотижневий неформальний BeerJS . Загалом за цими мітапами стежать кілька тисяч людей[11].

Все це неспішно натякає на подальшу підтримку мови зі сторони ECMA та її використання великими проектами.

2.2 Фреймворк NestJS

NestJS — це платформа для створення ефективних та масштабованих серверних програм Node.js. Він використовує прогресивний JavaScript, був створений на TypeScript та повністю підтримує його, при цьому все ще дозволяє розробникам кодувати на чистому JavaScript. Фреймворк поєднує в собі елементи ООП (об’єктно-орієнтоване програмування), FP (функціональне програмування) і FRP (функціональне реактивне програмування)[7]. Приклад коду та його архітектури продемонстровано на рисунку 2.2.

app.controller.ts

```ts
import { Get, Controller, Render } from '@nestjs/common';
import { AppService } from './app.service';

@Controller()
export class AppController {
  constructor(private readonly appService: AppService) {}

  @Get()
  @Render('index')
  render() {
    const message = this.appService.getHello();
    return { message };
  }
}
```

Рисунок 2.2 – Зразок коду з використанням NestJS

Під капотом NestJS використовує перевірені часом фреймворки HTTP-сервера, такі як Express (за замовчуванням). Також за бажанням може бути налаштований на використання Fastify.

Nest забезпечує та нав’язує певні рівні абстракції, які є вищими ніж ті, що використовуються в фреймворках Node.js (Express/Fastify), але в цей же час надає їхні API безпосередньо розробнику. Це дає розробникам свободу використовувати незліченну кількість сторонніх модулів, які доступні для основної платформи[22].

Розглянемо філософію фреймворку. В останні роки, завдяки Node.js, JavaScript став «lingua franca» інтернету для обох сторін: зовнішніх і внутрішніх програм. Це призвело до появи чудових проектів, таких як Angular , React і Vue , які покращують продуктивність розробників і дозволяють створювати швидкі та розширювані зовнішні програми. Однак, незважаючи на те, що існує безліч чудових бібліотек, допоміжних засобів та інструментів для NodeJS, жоден із них не вирішує ефективно головну проблему — архітектури[7] .

NestJS надає готову архітектуру програмного забезпечення, яка дозволяє розробникам і командам створювати високоякісні, масштабовані додатки, що є слабко пов’язаними між собою та легко дозволяють легко підтримувати свій функціонал. Архітектура NestJS значною мірою натхненна Angular.

Універсальність і розширюваність фреймворку не викликає сумніву.

NestJS надає розробникам максимальну свободу використання будь-яких додаткових модулів[8]. Він забезпечує високий рівень абстракції, що дозволяє використовувати API інших фреймворків, бібліотек тощо, збираючи унікальну серверну програму будь-якого типу з будь-яких модулів. Nest має відкритий вихідний код і має практично необмежену масштабованість, що активно використовують розробники у своїх проектах. Зокрема, вже є модулі для підключення баз даних PostgreSQL, MongoDB, MySQL та інтеграції технологій Caching, Mongoose, GraphQL, WebSockets тощо[8].

NestJS досить легко піддається тестуванню, тому що навіть при всій своїй різноманітності можливостей та масштабуванні, система змушує використовувати сувору архітектуру, подібно до Angular[21]. Це сприяє уникненню проблем з величезними витратами ресурсів на масштабування програми при необхідності - кожен мікросервіс можливо розширити окремо, без шквалу змін у всьому функціоналі.

Розглянемо перспективи фреймворку.

Зараз NestJS — це фреймворк, що має неймовірно стрімкі темпи розвитку. Він імпонує розробникам через можливість створювати додатки з гнучким функціоналом та втілювати оригінальні ідеї замовників у життя. Він вже налічує у своїх «коридорах» досить багато модулів. Також можливо відмітити значну кількість прикладів вирішення задач, які знаходяться у відкритому доступі і можуть стати в нагоді у вашому проекті – це означає, що дана платформа вже має своє чимале ком'юніті. Багато відомих компаній використовують цей фреймворк для створення власного функціоналу. Невеликий перелік таких компаній наведено на рисунку 2.3.

Рисунок 2.3 – Перелік компаній, що використовують NestJS у свої сервісах

Отже, адаптивна екосистема та масштабованість є основними причинами, чому ви можете обрати Nest для свого проекту, особливо якщо це стартап або

програма зі специфічною бізнес-логікою. Популярність даного фреймворка представлена у статистиці на рисунку 2.4.

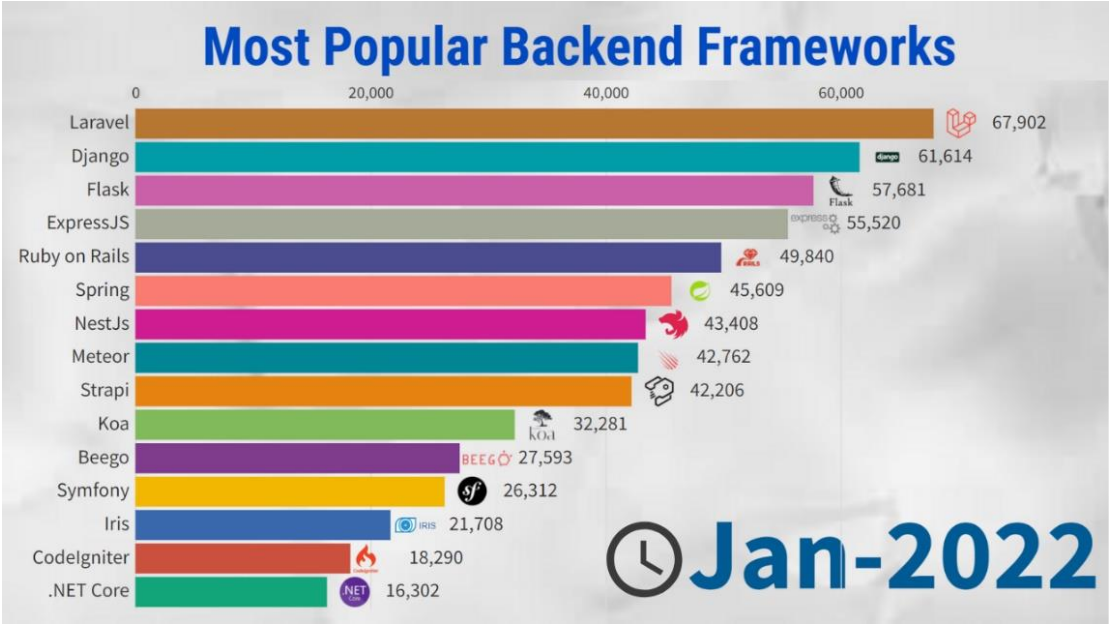

Рисунок 2.4 –Рейтинг найпопулярніших бекенд-фреймворків

2.3 NestJS-Telegraf

Telegram дозволяє розробникам створювати «third-party» програмне забезпечення й запускати його на своїй платформі. Він надає Bot API - добре задокументований інтерфейс на основі HTTP, де можливо знайти приклади для швидкого початку роботи або той чи інший варіант використання всього інструментарію Telegram. Приклад цього наведено на діаграмі послідовності у додатку Б, на рисунку Б.3.

На основі нативного Telegram API розробники створили чималу кількість обгорток для більш простої взаємодії з ним. Прикладом однієї із таких обгорток і є TelegrafJS[24].

Поєднання з фреймворком NestJS

У попередньому розділі вже було перераховано всі переваги використання NestJS. У поєднанні з TelegrafJS вони утворюють неймовірно

потужний симбіоз чіткої архітектури, модульного розмежування та ефективної взаємодії. При розробці обширного бот-функціоналу постає питання розмежування блоків коду за признаками приналежності, функціоналу та ієрархії. NestJS чудово вирішує цю задачу. Приклад коду npm пакету «nestjs-telegraf» наведено на рисунку 2.5.

```
@Update()
export class AppUpdate {
  @Start()
  async start(@Ctx() ctx: TelegrafContext) {
    await ctx.reply('Welcome');
  }

  @Help()
  async help(@Ctx() ctx: TelegrafContext) {
    await ctx.reply('Send me a sticker');
  }

  @On('sticker')
  async on(@Ctx() ctx: TelegrafContext) {
    await ctx.reply('👍');
  }

  @Hears('hi')
  async hears(@Ctx() ctx: TelegrafContext) {
    await ctx.reply('Hey there');
  }
}
```

Рисунок 2.5 – Приклад коду з використанням «nestjs-telegraf»

Відмітимо універсальність та розширювальність NestJS. Так як NestJS має змогу підключити велику кількість модулів – це значить, що кількість інструментарію, що розробник має змогу використати, значно збільшується і при цьому витрати часу не ростуть у геометричній прогресії. Яскравим прикладом можливо вказати підключення бази даних, де зберігаються дані користувачів бота. Підключення бази даних є простим, гнучким і орієнтованим на швидку роботу, його приклад вказано на рисунку 2.6.

```
TypeOrmModule.forRootAsync( options: {
  inject: [ConfigService],
  useFactory: (configService: ConfigService) => ({
    type: 'postgres',
    host: configService.get('db.host'),
    port: configService.get('db.port'),
    database: configService.get('db.name'),
    username: configService.get('db.user'),
    password: configService.get('db.password'),
    entities: [],
    synchronize: true,
    autoLoadEntities: true,
  }),
}),
```

Рисунок 2.6 – Приклад підключення БД

2.4 Axios

Axios - JavaScript-бібліотека, що має широку популярність в рядах розробників. Вона являє собою HTTP-клієнт, заснований на «Promise» конструкції та призначений як для клієнтської, так і для серверної частин функціоналу.

У грудні 2022 року пакет Axios був завантажений з npm 37,109,188 разів[13].

До моменту популярності Axios, в браузерах був відсутній API, що реалізує HTTP-клієнт. Стандартний інтерфейс XML HTTP Request (XHR) був та й досі лишається незручним, працювати з ним важко та вимагає додаткових затрат часу.

Короткий перелік можливостей axios:

- створення XMLHttpRequests із браузера;
- створення http-запитів з node.js;

- підтримка Promise API;
- перехоплення запитів і відповідей;
- перетворення даних запитів і відповідей;
- скасування запитів;
- автоматичне перетворення даних у JSON.

Приклад запиту за допомогою axios наведено на малюнку 2.7.

```
axios.get('/user?ID=12345')
  .then(function (response) {
    // handle success
    console.log(response);
  })
  .catch(function (error) {
    // handle error
    console.log(error);
  })
  .finally(function () {
    // always executed
  });
```

Рисунок 2.7 – HTTP запит за допомогою axios

2.5 PostgreSQL

PostgreSQL — це потужна об’єктно-реляційна база даних з відкритим вихідним кодом, активна розробка якої триває понад 35 років, що забезпечує їй надійну репутацію, надійність функціоналу і високу продуктивність. В офіційній документації можна знайти велику кількість інформації, яка описує, як встановити та використовувати PostgreSQL. Репозиторій з відкритим кодом надає можливість доступу до багатьох корисних місць, де можна ознайомитися з PostgreSQL, дізнатися, як вона працює.

Розглянемо мотивацію використання.

PostgreSQL має багатий інструментарій, що допомагає розробникам створювати додатки, адміністраторам — захищати цілісність даних і створювати відмово стійке середовище, а також допомагає вам керувати своїми даними, незалежно від того, наскільки велика чи мала їх кількість. Окрім того, що PostgreSQL є безкоштовним і відкритим вихідним кодом , він дуже просто розширюватися. Наприклад, ви можете визначати власні типи даних, створювати власні функції, навіть писати код з різних мов програмування без перекомпіляції бази даних.

PostgreSQL намагається відповідати стандарту SQL, якщо така відповідність не суперечить традиційним функціям або не призведе до невдалих архітектурних рішень. Багато функцій, необхідних стандарту SQL, підтримуються, хоча іноді з дещо відмінним синтаксисом або функціями. Починаючи з випуску версії 15 у жовтні 2022 року, PostgreSQL відповідає принаймні 170 із 179 обов’язкових функцій для відповідності SQL:2016 Core. На даний момент жодна інша реляційна база даних не відповідає цьому стандарту.

Розглянемо інструментарій.

Типи даних в себе включають:

- примітиви: Integer, Numeric, String, Boolean;
- структуровані: Date/Time, Array, Range / Multirange, UUID;
- документ орієнтовані: JSON/JSONB, XML, Key-value (Hstore);
- геометричні : Point, Line, Circle, Polygon;
- кастомізація: Composite, Custom Types.

Цілісність даних надається за допомогою:

- UNIQUE, NOT NULL;
- первинні ключі;
- іноземні ключі;
- обмеження виключення;
- явні блокування, рекомендаційні блокування.

Безпека складається з:

- автентифікації: GSSAPI, SSPI, LDAP, SCRAM-SHA-256;
- надійна система контролю доступу;
- безпека на рівні стовпців і рядків;
- багатофакторна аутентифікація з сертифікатами та додатковим методом.
- інтернаціоналізація, текстовий пошук:
- підтримка міжнародних наборів символів, наприклад через порівняння ICU;
- сортування без урахування регістру та наголосу;
- повнотекстовий пошук.

2.6 Дослідження існуючих API та дата провайдерів

Абревіатура API розшифровується як Application Programming Interface або програмний інтерфейс програми. API – це набір конструкцій, інтерфейсів, методів, способів і правил взаємодії та обміну даними між різними програмами та їх компонентами[13].

Спілкування відбувається через класи, методи, функції і різні структури. Іноді комунікації можливі за допомогою сталих змінних однієї програми, до яких звертаються інші.

Якщо підсумувати, API виступає в ролі посередника(брокера), що надає змогу функціоналу 1 отримати доступ до даних та можливостям функціоналу 2. Це дозволяє розробникам покращити функціональність продуктів, що розробляються і пов’язати їх з іншими системами, тощо.

Розглянемо чому API називають інтерфейсом

Інтерфейс – певна межа між двома програмами чи частинами їхнього функціоналу, за допомогою якої вони можуть проводити обмін даними та виконувати пов’язані між ними операції. При цьому процеси, що відбуваються

всередині кожної з них, є інкапсульованими для програми, що встановлює зв’язок.

Завдяки такому підходу швидко налагоджується взаємодія між декількома програмами чи їх частинами (можливий навіть варіант налагодження подібного типу взаємодії між внутрішніми частинами однієї програми), не переймаючись питанням про їх внутрішнє влаштування, програмну логіку або способі внутрішньої обробки даних. За допомогою інтерфейсів розробникам не потрібно витрачати купу часу та розбиратися в програмному коді інших фахівців, щоб підключити свій продукт до іншого.

Користувачі також мають з цього свою вигоду: до прикладу, користувачам не потрібно замислюватися про те, що стоїть за ширмою звичних функцій в девайсу - для надсилання повідомлення з телефону їм не потрібно знати принцип роботи його «внутрішньої кухні». Замість цього достатньо набрати бажаний текст та натиснувши кнопку інтерфейсу.

Підсумок: API надає можливість користуватися інструментарієм певного функціоналу, не маючи уявлення про те, як вона працює. Саме через це API прирівнюють до інтерфейсу.

2.6.1 Мотивація використання

Головною задачею API є налагодження взаємодії між певними сторонами функціоналу та інкапсуляції внутрішнього функціоналу.

Головними перевагами API є: збільшення безпеки розробки. За допомогою API розробник має змогу винести в окрему частину той функціонал, що повинен бути захищений від стороннього ока. Це знижує шанси некоректного використання стороннім функціоналом. Ще однією перевагою є зниження кошторису розробки та економія часу. Придбання готового API доволі часто є вигіднішим, ніж витрачати дорогоцінний час на створення власного функціоналу з нуля з усіма наслідками, що можуть

з’явитися. Коли є змога використовувати вже готовий інструментарій, а не винаходити велосипед з нуля – чому б цим не скористатися[13].

Розмежування зв’язків між сторонніми системами, сервісами чи компонентами. Розробники можуть імплементувати у власні продукти підтримку стороннього функціоналу, навіть не замислюючись про те, хто це створював та яким чином це працює.

2.6.2 RESTful API

REST – абревіатура від Representational State Transfer, що перекладається як «передача репрезентативного стану». Це підхід до проектування розподілених систем за допомогою впровадження певний обмежень. Центральною абстракцією в REST є ресурс[31]. А головні обмеження виглядають так:

- клієнт-серверна модель;
- взаємодія без збереження стану;
- логічний інтерфейс.

RESTful API є одним із самих популярних підходів для веб-розробки. Принцип його роботи наведено на діаграмі у додатку Б, на рисунку Б.1.

2.6.3 Google Translate API

Google Перекладач — це знайомий всім і кожному сервіс від компанії Google. Через складність перекладу деяких мов або, в залежності від виду чи контексту тексту, користувачам доступні на вибір кілька варіантів перекладу одного слова чи фрази, це досить корисна можливість, наприклад, для наукових термінів чи технічних статтях.

Цей сервіс, що на відміну від аналогічних, які «під капотом» використовують технологію SYSTRAN, має власне програмне забезпечення. Тут працює алгоритм з машинним перекладом, який здатний до самонавчання.

Також цікавим моментом є те, що за допомогою Google Перекладача можна перекладати сторінку сайту повністю. Цей сервіс також є корисним для веб-дизайнерів. Фахівці Google розробили скрипт, що дає змогу перекладати сайти на будь-які доступні мови[14].

Google Перекладач, як і інші подібні інструменти для перекладу тексту, окрім свого багатого функціоналу також включає в себе певні недоліки та обмеження. Він допоможе користувачеві перекласти текст та зрозуміти загальний контекст речення, але не надасть точний та досконалий переклад наприклад тих же ідіом, що може зіграти злий жарт при роботі з перекладом фраз кінострічок, тощо.

Оскільки алгоритм Google Перекладача контролює видачу варіантів перекладу загальновживаних слів, він може генерувати переклад для нецензурних слів. Розробники компанії постійно працюють над вдосконаленням точності перекладу вже наявних мов та над інтеграцією нових.

Також важливо відмітити, що далеко не зайвим в API буде можливість відтворення тексту перекладу. Приклад Google Translate API наведено за допомогою сервісу «RapidAPI» на рисунку 2.8.

2.6.4 Words API

WordsAPI — це RESTful API, який дозволяє користувачеві запитувати базу даних визначень для понад 150 000 слів англійської мови. [15]. Приклад JSON відповіді WordsAPI наведено на рисунку 2.9.

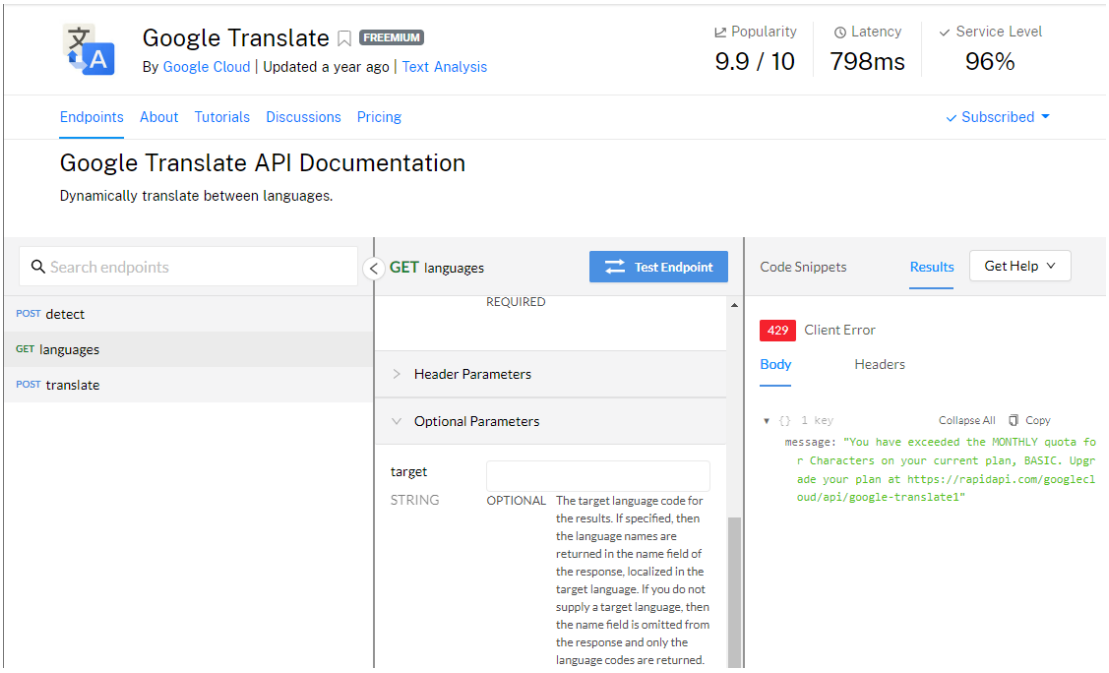

Рисунок 2.8 – Google Translate API в інтеграції сервісу «RapidAPI»

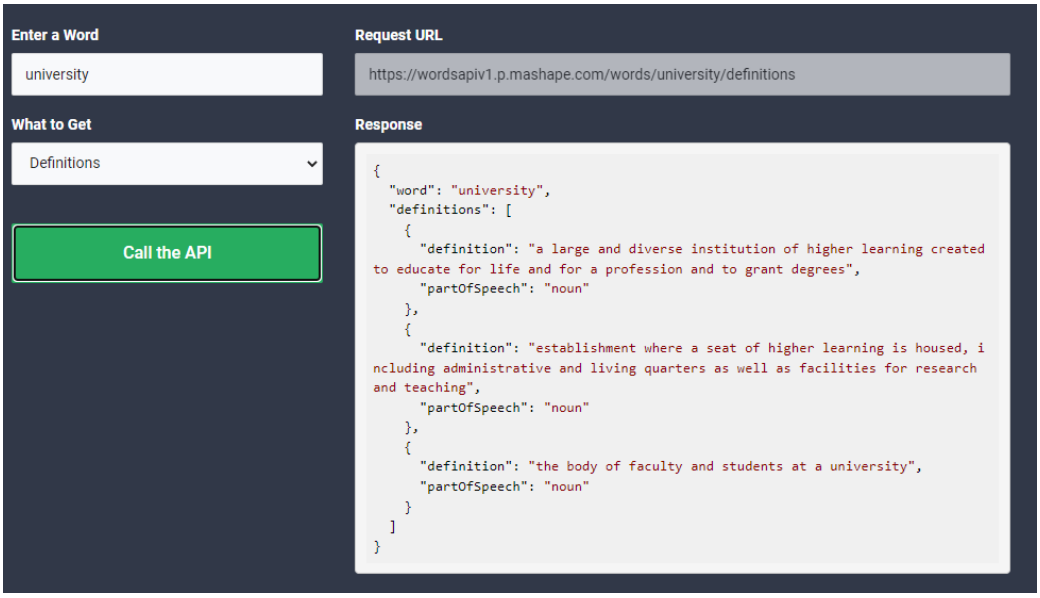

Рисунок 2.9– Приклад JSON відповіді WordsAPI

WordsAPI дозволяє програмам отримувати різноманітну, а іноді й специфічну інформацію про англійські слова. Коли ми робимо запит на окреме слово, воно може містити наступну інформацію:

- визначення;
- синоніми;
- приклад використання;

- частина мови;
- рими.

Більш цікаві ендпоінти можуть продемонструвати співвідношення слова з іншими словами в реальному світі. Ці ендпоінти включають наступну інформацію[16]:

- регіон використання (наприклад, Канада);
- знаходиться в категорії;
- включає категорії;
- подібний до;
- включає частини («будівля» може повертати список із «покрівлею», «кладкою», «сантехнікою» тощо).

Ціноутворення даного API є досить помірним. При оптимізації та кешуванні запитів можливо обійтися мінімальними планами. Ціна вказує на передбачувані випадки використання. API призначений для частого використання. Базовий план для WordsAPI охоплює 2500 запитів на день, а потім 0,004 доларів США за запит. Місячний план становить лише 10 доларів США, а квота збільшується до 25 000 на день. Середня затримка на момент написання становить 301 мс, що не є високим показником. Приклад цін наведено на рисунку 2.10.

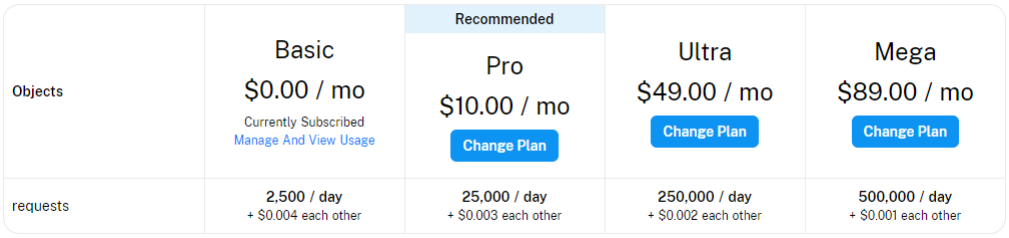

Рисунок 2.10 – Ціни на використання WordsAPI

2.6.5 PlayPhrase

PlayPhrase – досить потужний провайдер відеоматеріалів[29], що дозволяє отримати фрагменти кінострічки, серіалу, тощо за ключовим словом чи фразою. На момент написання сервіс налічує 7.600.186 фрагментів, що мають чіткі відношення до певних слів(фраз).

Також сервіс вказує на приналежність відео-фрагменту до конкретного фільму, серіалу, тощо та точний час, де була використані ключові елементи. Приклад роботи сервісу наведено на рисунку 2.11.

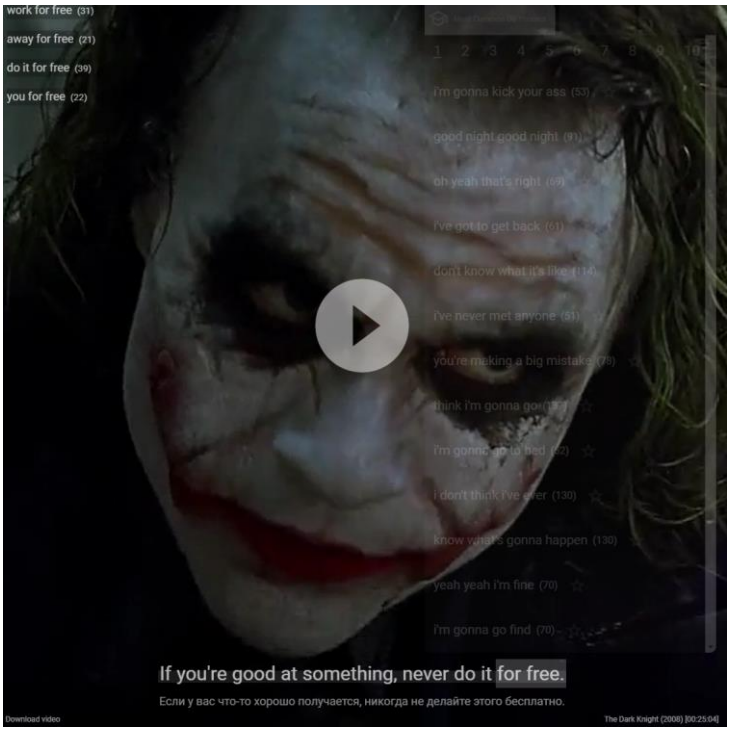

Рисунок 2.11 – Приклад роботи PlayPhrase

2.6.6 RandomWord

За допомогою сервісу Random Word, ви можете знайти випадкове слово, фразу, ідіому, цитатучи речення. Веб-сайт дає можливість щодня вивчати нові слова, щоб збільшити словниковий запас. Якщо ви поспішаєте, ви можете

вивчити перше слово, яке з’явиться як слово дня, проте якщо у вас є більше часу, ви можете натиснути на посилання «наступне слово» та вивчити стільки, скільки хочете, або скільки дозволить час.

Однією з головних особливостей є те, що цей сервіс надає визначення кожного слова, що відображається. Оскільки багато слів є досить незвичайними, а не такими, які ви використовуєте регулярно, це дає вам змогу дізнатися значення слова без необхідності розкривати словник, щоб спробувати з’ясувати визначення. Приклад веб-інтерфейсу RandomWord наведено на рисунку 2.12.

Рисунок 2.12 – Веб-інтерфейс RandomWord

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Структура бота

Структурна складова проекту базується на основі діаграми компонентів. Сама діаграма компонентів наведена у додатку Б, на рисунку Б.2.

Структурною одиницею при розробці функціоналу за допомогою nest-telegraf є сцена. Вона включає в себе певні повідомлення, які пов’язані між собою за допомогою взаємодії користувача з ботом. Наприклад, при натисканні кнопки «Розпочати навчання» розпочинається зона відповідальності сцени «main.scene».

Для реалізації закінченого функціоналу повного циклу необхідно створити наступний перелік сцен:

- start.scene - початок роботи та знайомство з ботом;
- level.scene - встановлення налаштувань рівня знань англійської мови та можливе проходження тестування;
- schedule.scene - встановлення налаштувань частоти повідомлень бота англійської мови;
- main.scene – меню користувача, вибір підрозділу навчання;
- word.scene – навчання за допомогою слів
- short.scene – навчання за допомогою відеофрагментів;
- idiom.scene – навчання за допомогою ідіом;
- quote.scene – навчання за допомогою цитат;
- sentence.scene – навчання за допомогою речень;
- translate.scene – переклад слів;
- stats.scene – статистика користувача.

3.2 Початкове налаштування бота

3.2.1 Створення бота

Перш за все бот було створено та ініціалізовано за допомогою внутрішнього API Telegram – BotFather. З його допомогою було отримано токен для подальшої взаємодії з API Telegram[23]. Процес створення бота наведено на рисунку 3.1.

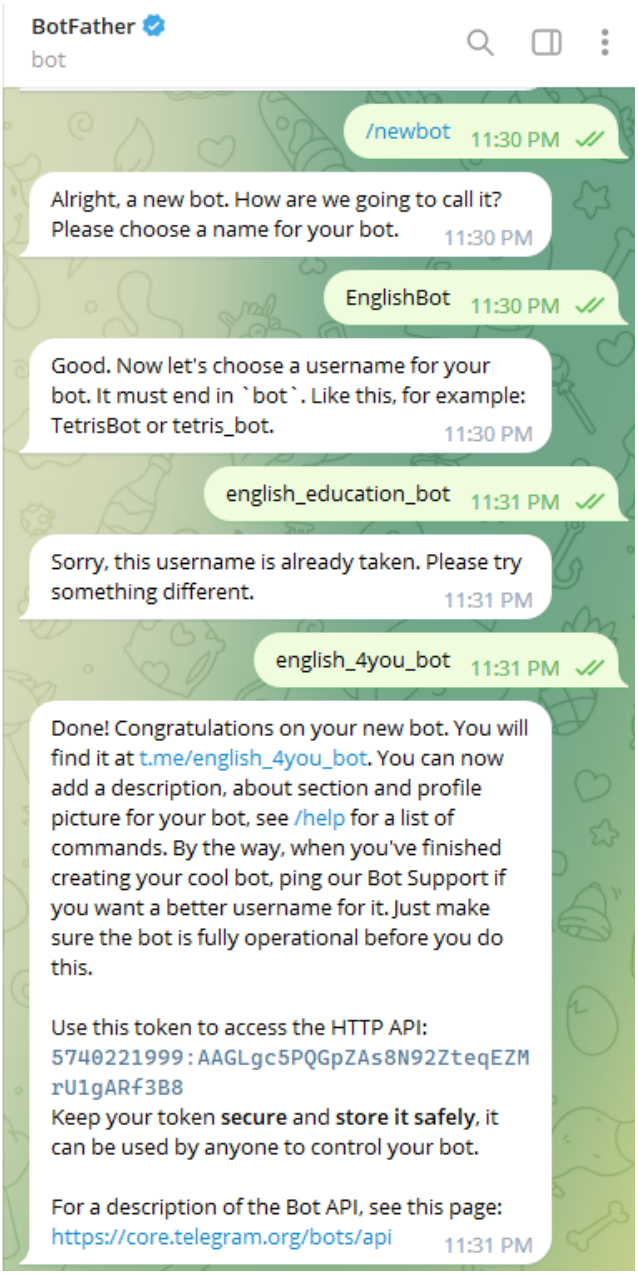

Рисунок 3.1 – Створення та ініціалізація бота за допомогою BotFather

3.2.2 Конфігурація опису та інших початкових елементів

За допомогою вищезгаданого BotFather також були добавлені опис та ім’я бота, що є базою інформаційного вікна бота. Вони наведені на рисунку 3.2.

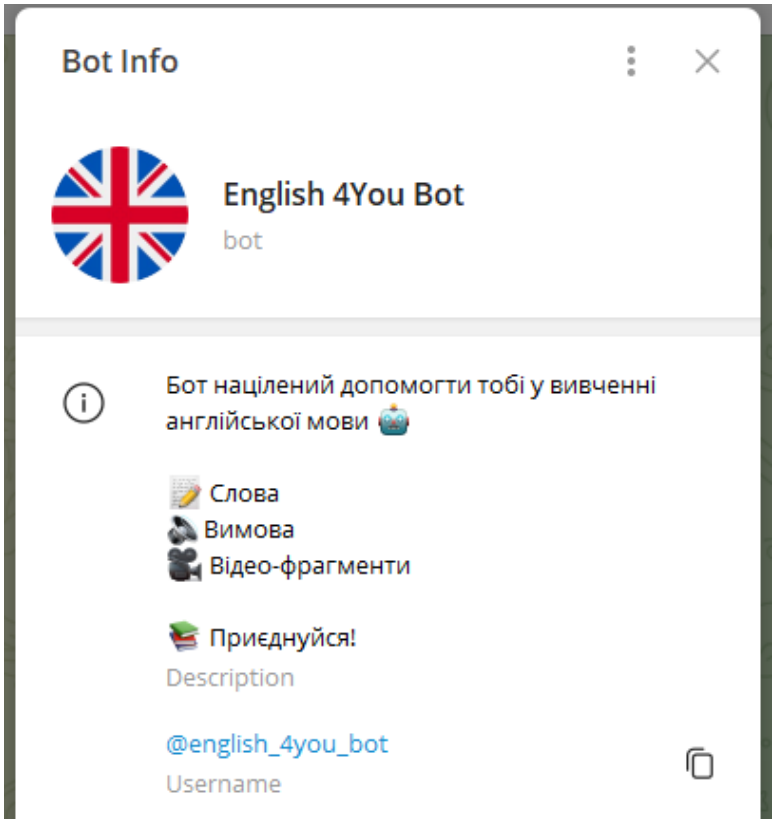

Рисунок 3.2 – Інформаційне вікно бота

Сконфігурований початковий екран бота та добавлені швидкі команди користувача. Це потрібно для швидкого орієнтуванні користувача в можливостях бота та швидкого початку роботи з ним. Зразок початкового екрану бота наведено на рисунку 3.3.

3.2.3 Швидкі команди бота

Команди несуть в собі роли швидкої взаємодії користувача з функціоналом бота:

- «start» - початок роботи та ініціалізація бота(буде розглянуто у розділах нижче);
- «help» - інформаційна допомога користувачу;
- «stats» - початок роботи та ініціалізація бота (буде розглянуто у розділах нижче).

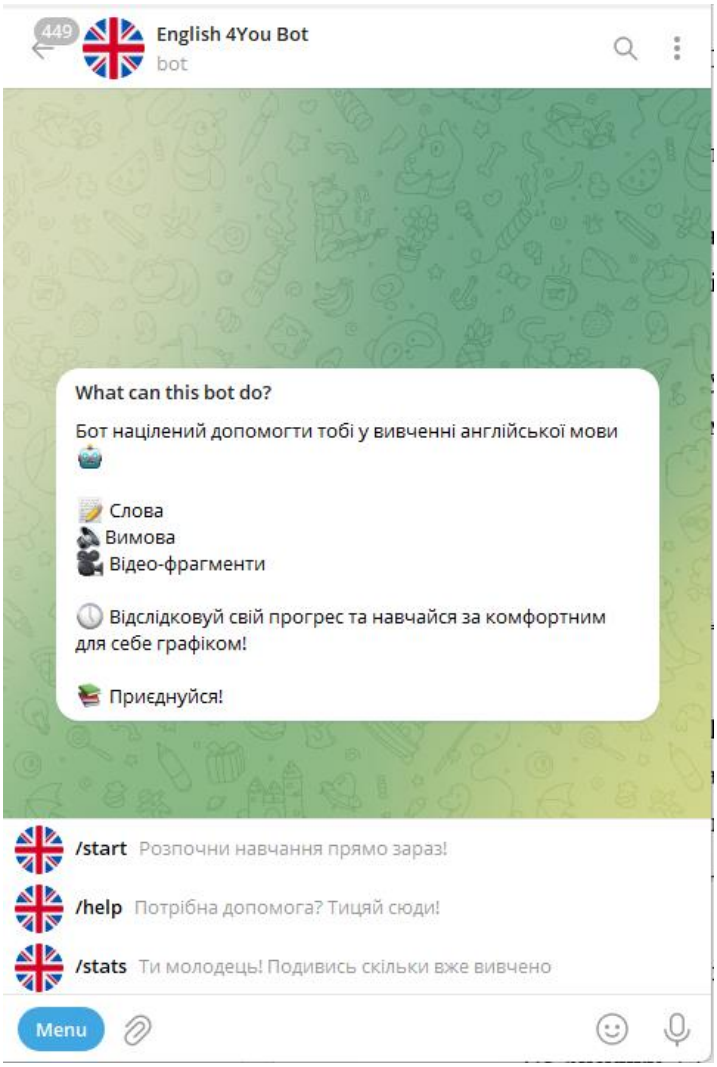

Рисунок 3.3 – Початковий екран бота

3.2.4 Швидкі команда «help»

Команда «help» надсилає допоміжне повідомлення щоб допомогти користувачеві зорієнтуватись у подальших діях. Приклад повідомлення наведено на рисунку 3.4.

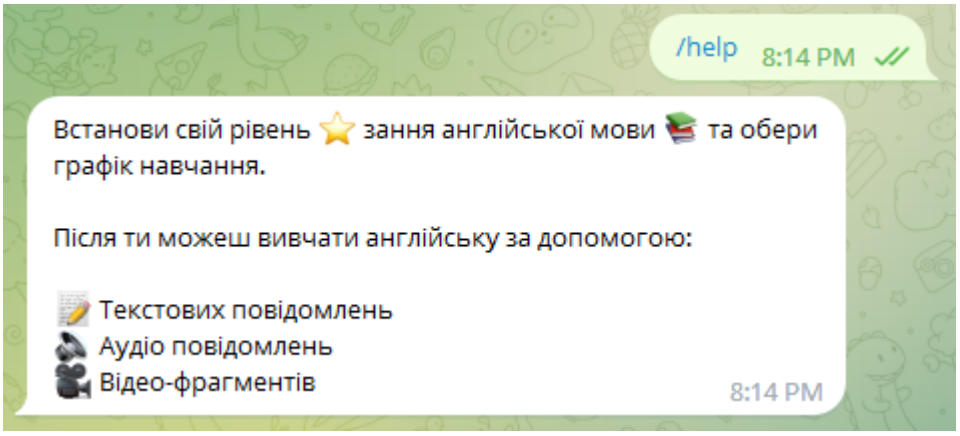

Рисунок 3.4 – Команда «help»

3.2 Start Scene

Сцена потрібна для початку роботи з ботом, вона є ключовим моментом розгалуження для налаштування параметрів конфігурації бота. Після її активації користувач розпочинає свою взаємодію з ботом шляхом отримання повідомлення, що пропонує йому подальшу ієрархію дій. Також в повідомленні присутня інструкція щодо подальших дій користувача. Приклад початкової сцени наведено на рисунку 3.5.

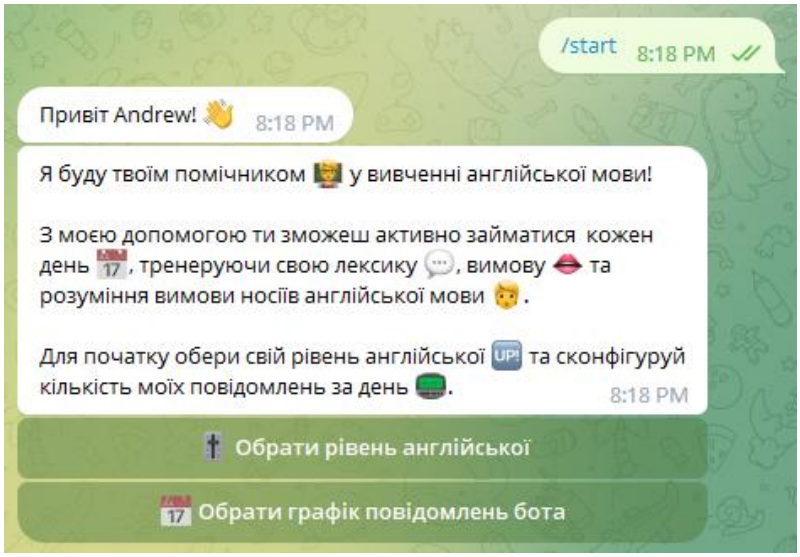

Рисунок 3.5 – Команда «start»

Сцена несе в собі дуже відповідальну роль – створення локальної сесії користувача та створення початкового запису в базі даних. Подальше розгалуження сцени відбувається або до Level Scene, або до Schedule Scene.

Доступні дії користувача:

- обрати графік повідомлень бота– перехід на Level Scene;
- обрати рівень англійської – перехід на Schedule Scene.

Блок-схему Start Scene наведено у додатку Б, на рисунку Б.4.

3.3 Level Scene

Сцена є необхідною для встановлення користувачем рівня знань англійської мови для початку роботи з ботом. Надає інструкцію для розуміння подальших дій користувача. Приклад повідомлення сцени наведено на рисунку 3.6.

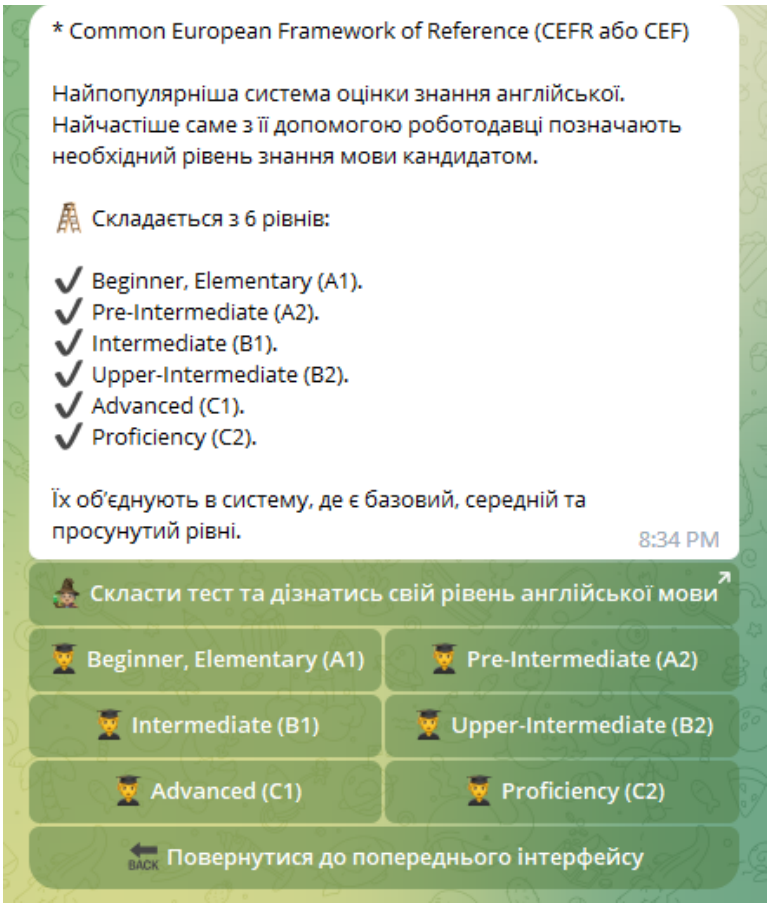

Рисунок 3.6 – Початкове повідомлення Level Scene

Доступні дії користувача:

- складання тесту рівня знань англійської мови за допомогою стороннього ресурсу, перенаправлення на посилання в браузері;
- рівень англійської – конфігурація рівня знань англійської мови, запис його в базу даних та продовження конфігурації бота або (якщо розклад вже було сконфігуровано раніше) перехід до Main Scene (навчальне меню)Б приклад наведено на рисунку 3.8. Якщо ж користувач ще не сконфігурував розклад навчання – його буде повідомлено про це та запропоновано його конфігурацію. Такий сценарій наведено на рисунку 3.7;
- назад – повернення до попереднього меню.

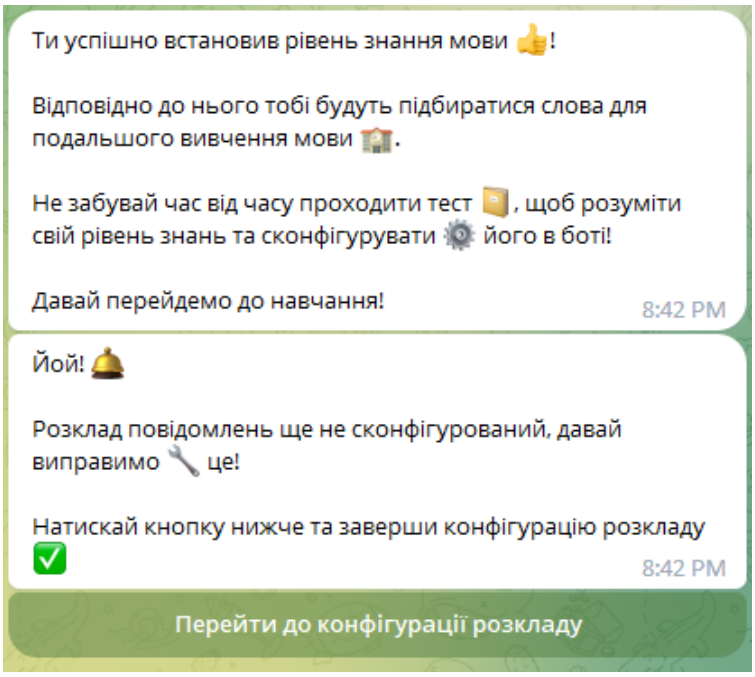

Рисунок 3.7 – Вибір користувачем рівня знань англійської мови (варіант 1)

Доступні дії користувача:

- перехід до Schedule Scene та конфігурація розкладу;
- перехід до Main Scene та початок навчання(якщо розклад вже було сконфігуровано раніше).

Блок-схему Level Scene наведено у додатку Б, на рисунку Б.6.

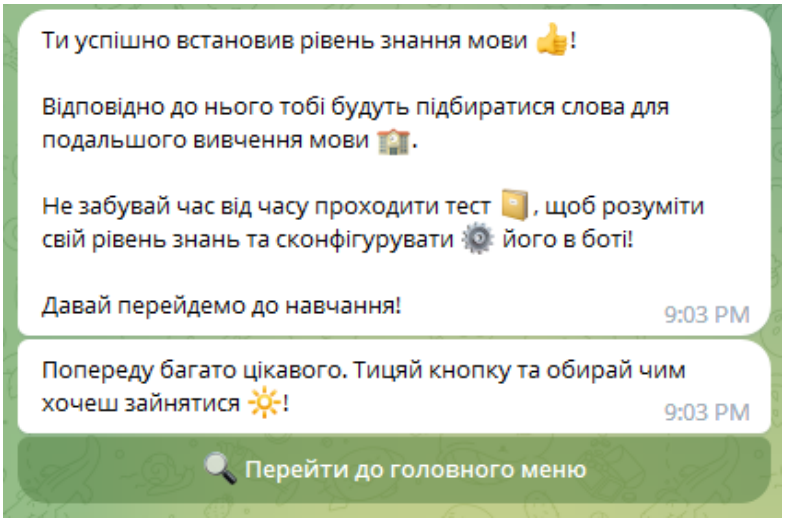

Рисунок 3.8 – Вибір користувачем рівня знань англійської мови (варіант 2)

3.4 Schedule Scene

Сцена потребує встановлення користувачем частоти отримання повідомлень з навчальним матеріалом в межах доби. Це є необхідною умовою для початку роботи з ботом. Також вона надає інструкцію для розуміння подальших дій користувача. Приклад повідомлення сцени наведено на рисунку 3.9.

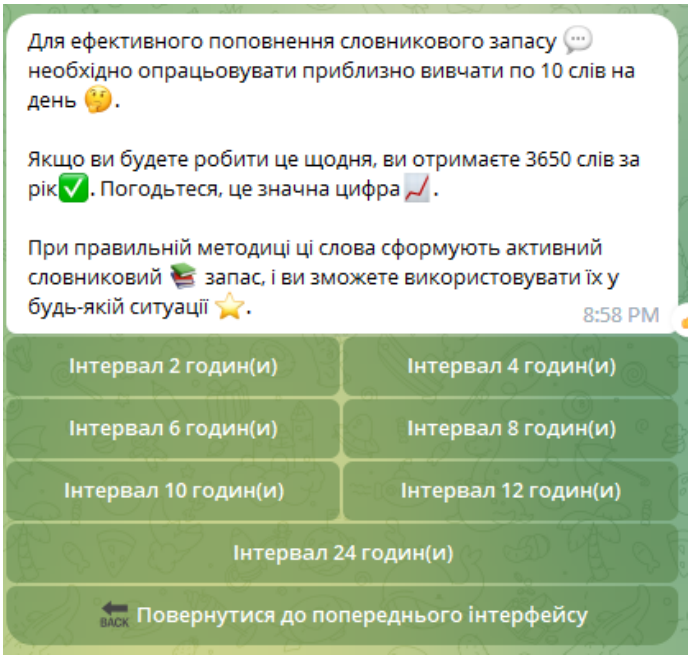

Рисунок 3.9 – Початкове повідомлення Schedule Scene

Доступні дії користувача:

- встановлення частоти повідомлень в межах доби, її запис в базу даних та продовження конфігурації бота або (якщо рівень знань вже було сконфігуровано раніше) перехід до Main Scene (навчальне меню), сценарій наведено на рисунку 3.11. Якщо ж користувач ще не сконфігурував рівень знань – його буде повідомлено про це та запропоновано його конфігурацію. Такий сценарій наведено на рисунку 3.10;
- назад – повернення до попереднього меню.

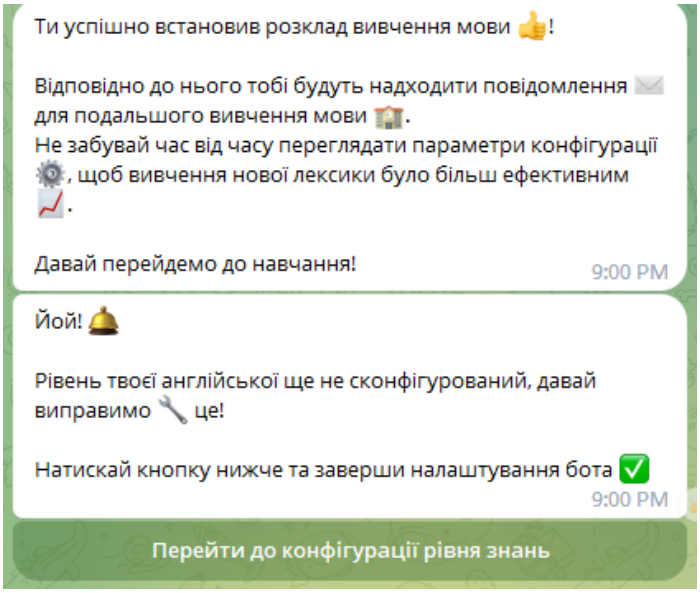

Рисунок 3.10 – Вибір користувачем частоти повідомлень в межах доби (варіант 1)

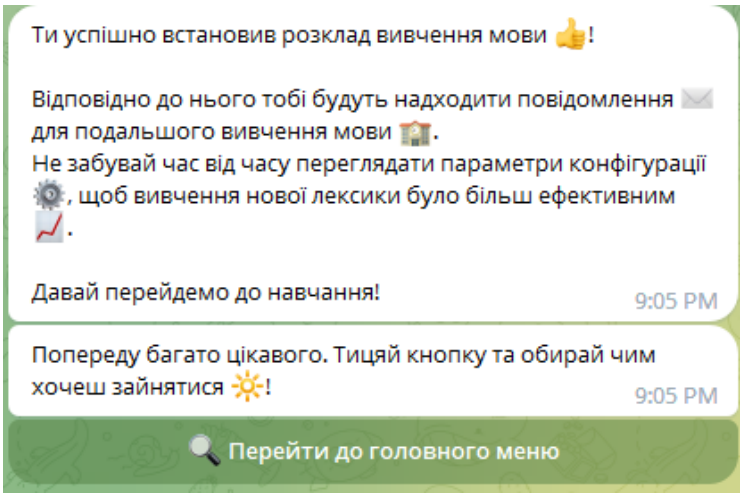

Рисунок 3.11 – Вибір користувачем частоти повідомлень в межах доби (варіант 2)

Доступні дії користувача:

- перехід до Level Scene та конфігурація розкладу;
- перехід до Main Scene та початок навчання(якщо рівень знань вже було сконфігуровано раніше).

Блок-схему Schedule Scene наведено у додатку Б, на рисунку Б.5.

3.5 Main Scene

Сцена є розгалужувачем між подальшими сценами та надає невелику підказку користувачу щодо подальших дій. Приклад меню головної сцени наведено на рисунку 3.12.

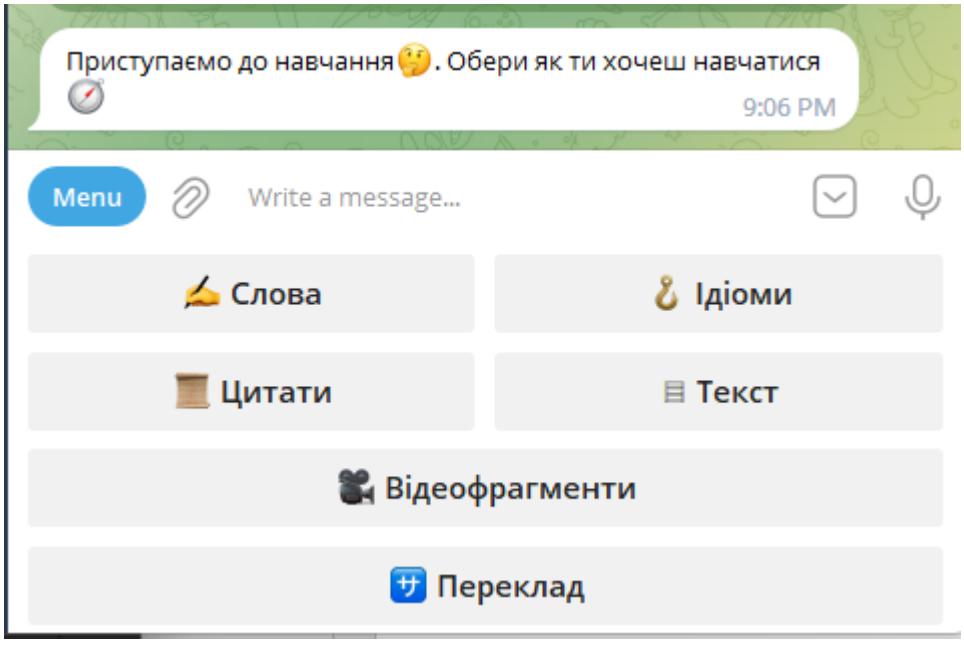

Рисунок 3.12 – Розгалуження методів навчання

Доступні дії користувача:

- вибір методу навчання «Слова», перехід до Word Scene;
- вибір методу навчання «Ідіоми», перехід до Idioms Scene;
- вибір методу навчання «Цитати», перехід до Quotes Scene;
- вибір методу навчання «Відеоматеріали», перехід до Word Scene;

- вибір перекладача, перехід до Translate Scene;
- перехід до Main Scene та початок навчання(якщо рівень знань вже було сконфігуровано раніше).

3.6 Word Scene

Беззаперечно одна із найскладніших сцен, де використовуються всі залучені API.

3.6.1 Збір даних

Сцена займається пошуком слова та збиранням даних стосовно цього слова:

- переклад;
- транскрипції;
- складів;
- частина мови;
- синоніми;
- антоніми;
- приклади використання;
- значення.

3.6.2 Виведення повідомлення

Після збору усіх наявних даних відбувається їх обробка та перетворення у відповідне повідомлення бота. Так, як деякі слова можуть мати подвійне

значення, у цьому випадку бот виведе усі знайдені значення у коректній для користувача формі. Приклад виводу повідомлення наведено на рисунку 3.13.

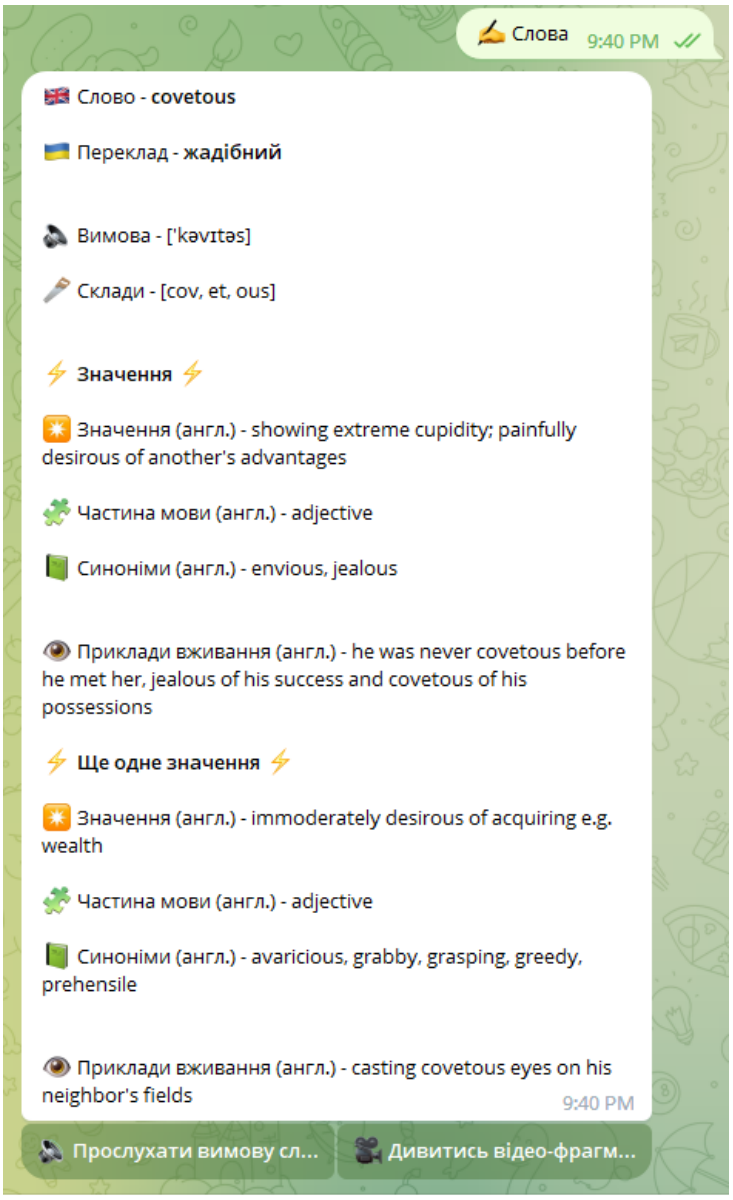

Рисунок 3.13 – Метод навчання за допомогою слів

Доступні дії користувача:

- прослухати правильну промову слова;
- перегляну відео-фрагмент.

3.6.3 Прослуховування вимови слова

Після того, як користувач натиснув кнопку прослуховування вимови, він отримає від боту файл, що буде містити записаний аудіо текст. Користувач може прослухати та зберегти його. Приклад повідомлення наведено на рисунку 3.14.

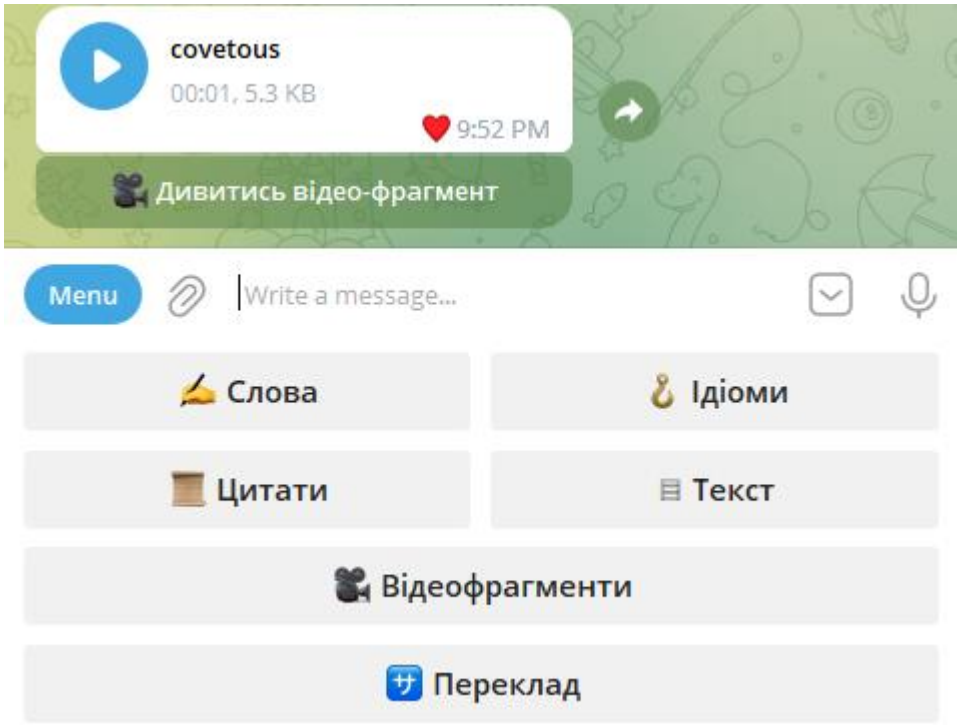

Рисунок 3.14 – Прослуховування вимови слова

Доступні дії користувача:

- обрати будь який вид навчання з нижнього меню;
- подивитись відеофрагмент.

3.6.4 Відеофрагмент з словом, що вивчається

Обравши перегляд відеофрагменту будь-то з головного меню, будь-то після прослуховування вимови слова – користувач отримає відрізок

кінострічки чи серіалу (приклад наведено на рисунку 3.15), де застосовується слово, що вивчається. Користувач має змогу завантажити цей фрагмент у свою колекцію та переглянути у будь-який зручний час знову. При перегляді кінофрагменту бот вказує його джерело то таймкод, так користувач зможе подивитись цей фільм після та згадати вивчене слово у обраному моменті. Опісля перегляду даного відеофрагменту бот пропоную (за наявності) перегляну наступний (приклад наведено на рисунку 3.16), якщо фрагменти закінчились - кнопка перегляду наступного та підказка до неї зникають.

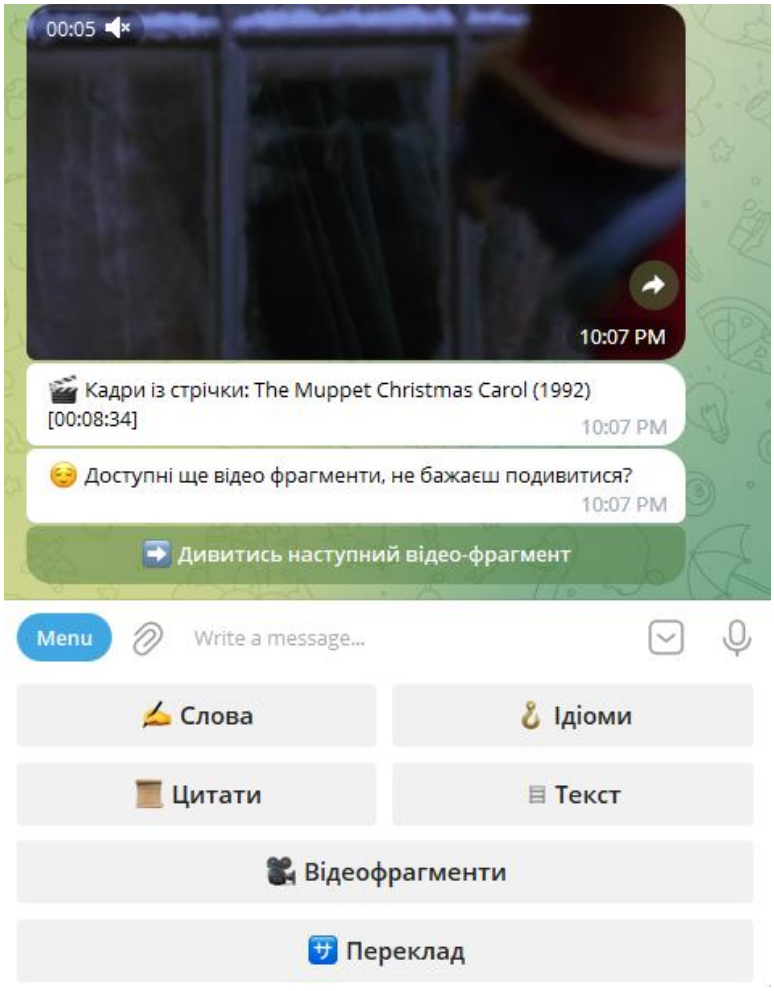

Рисунок 3.15 – Перегляд відеофрагменту

Доступні дії користувача:

- обрати будь який вид навчання з нижнього меню;
- подивитись наступний відеофрагмент.

Також можливо відмітити, що користувач має змогу завантажити фрагмент на пристрій та подивитись його пізніше.

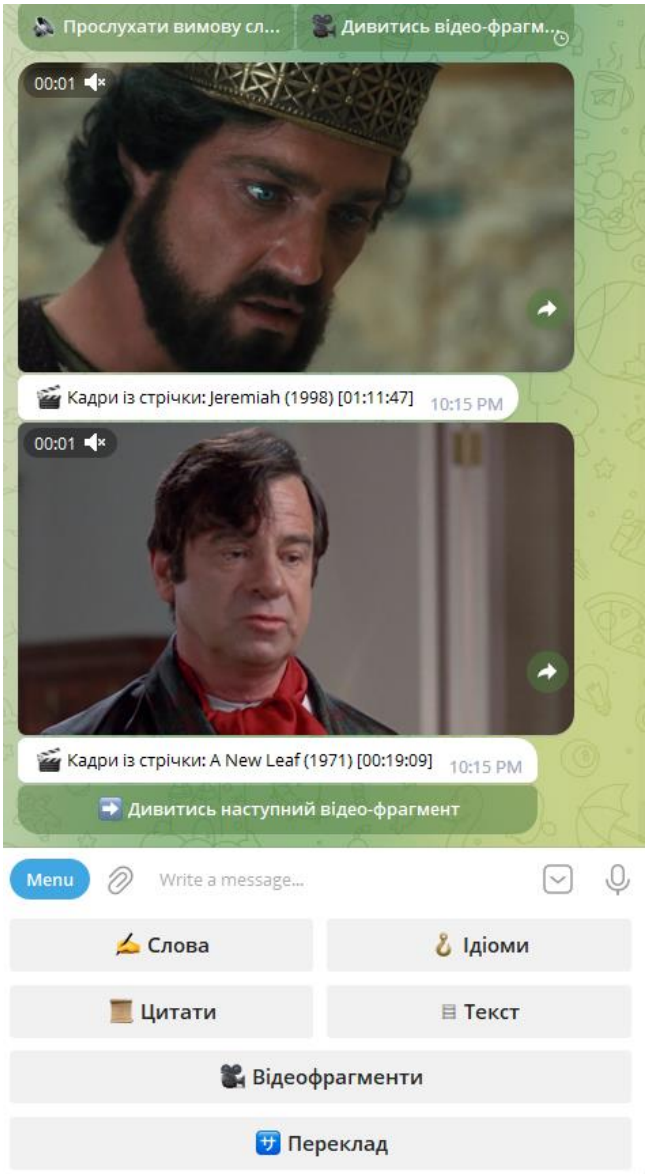

Рисунок 3.16 – Перегляд наступного відеофрагменту

Блок-схему Word Scene наведено у додатку Б, на рисунку Б.7.

3.7 Idiom Scene

За допомогою API бот генерує ідіому англійкою мовою, надає переклад, приклади використання, вимову та намагається знайти для ідіоми відеофрагменти з її використанням. У подальшому функціонал і дії

максимально схожі на подібний функціонал із попереднього пункту. Приклад наведено на рисунку 3.17.

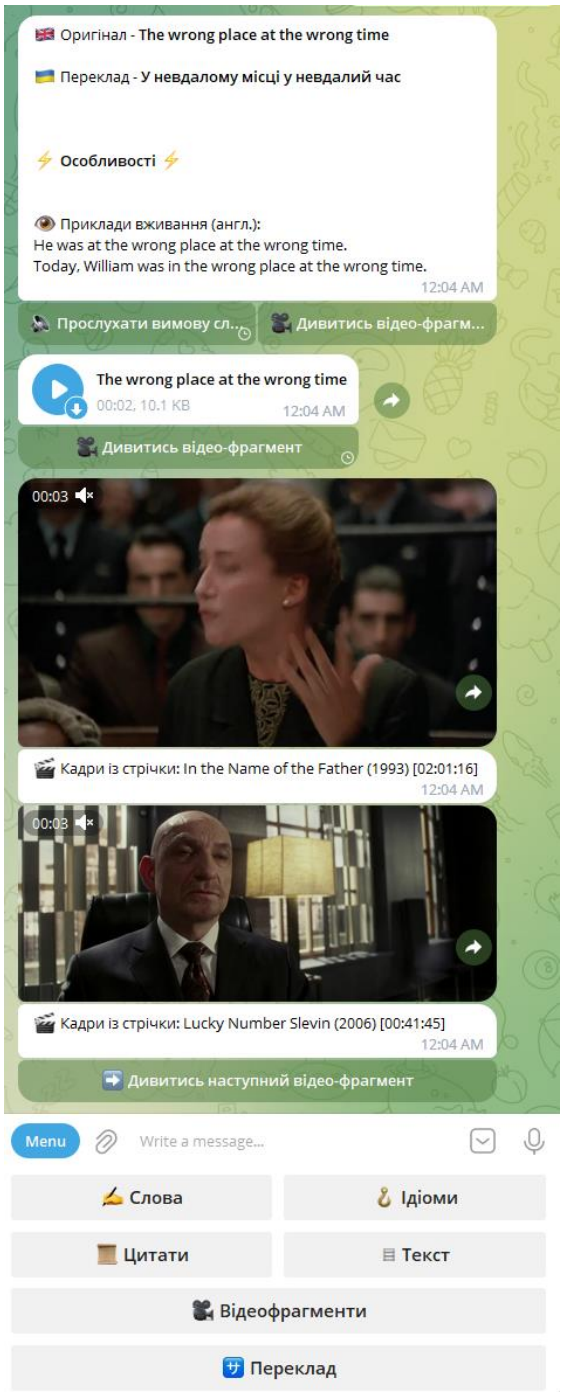

Рисунок 3.17 – Приклад функціональності методу навчання за ідіомами

Доступні дії користувача:

- обрати будь який вид навчання з нижнього меню;
- подивитись наступний відеофрагмент.

3.8 Quote Scene

За допомогою API бот генерує цитату англійкою мовою, надає переклад, приклади використання, вимову та намагається знайти для цитати відеофрагменти з її використанням. У подальшому функціонал і дії максимально схожі на подібний функціонал із пункту 3.6. Приклад користування функціоналом цитат наведено на рисунку 3.18.

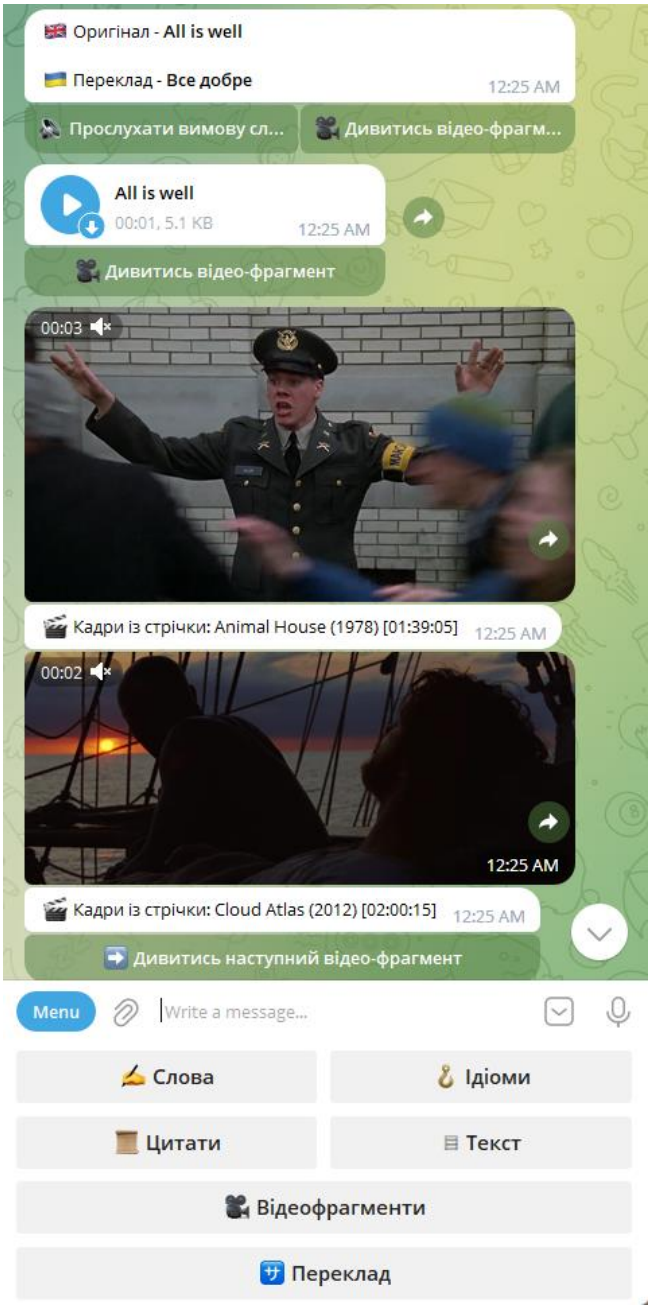

Рисунок 3.18 – Приклад функціональності методу навчання за цитатами

Доступні дії користувача:

- обрати будь який вид навчання з нижнього меню;
- подивитись наступний відеофрагмент.

3.9 Text Scene

За допомогою API бот генерує параграф тексту англійкою мовою, надає переклад. Після прочитання тексту користувач натискає кнопку подальшої дії та отримує переклад для тексту, що допоможе його перевірити правильність розуміння тексту.

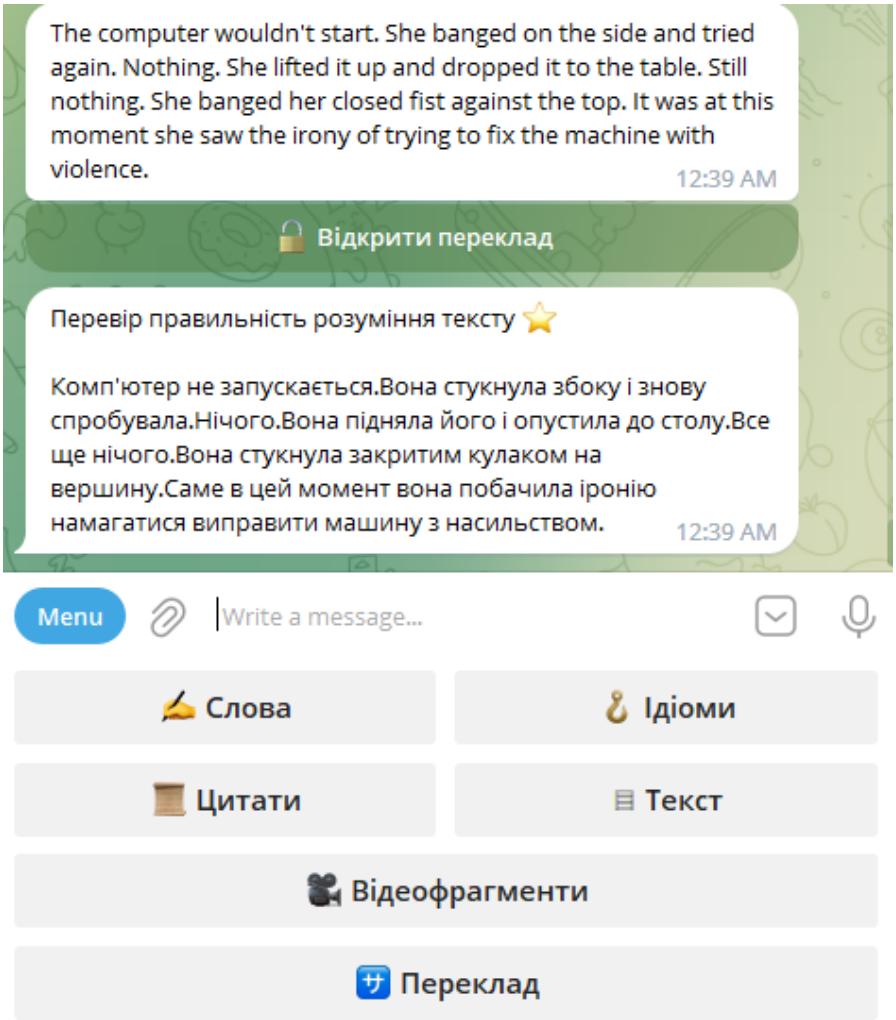

Рисунок 3.19 – Приклад функціональності методу навчання за допомогою текстового змісту

Доступні дії користувача:

- відкрити переклад у наступному повідомленні;
- обрати будь який вид навчання з нижнього меню.

Блок-схему Text Scene наведено у додатку Б, на рисунку Б.10.

3.10 Translate scene

Сцена має на меті переклад слів за запитом користувача. Для перекладу користувачу потрібно ввести слово на натиснути кнопку відправки. Бот знайде переклад слова, особливості, транскрипцію, вимову та відеофрагменти з ним. Приклад наведено на рисунку 3.20.

Доступні дії користувача:

- обрати будь який вид навчання з нижнього меню;
- подивитись наступний відеофрагмент.

3.11 Shorts Scene

Сцена дозволяє вести тренувати розуміння вимови носіїв за допомого перегляду коротких відеороликів. При перегляді кінофрагменту бот вказує його джерело та таймкод, так користувач зможе подивитись цей фільм після навчання. Після перегляду останнього відеоролику бот пропонує переглянути наступний. Приклад наведено на рисунку 3.21.

Доступні дії користувача:

- обрати будь який вид навчання з нижнього меню;
- подивитись наступний відеофрагмент.

Блок-схему Shorts Scene наведено у додатку Б, на рисунку Б.9.

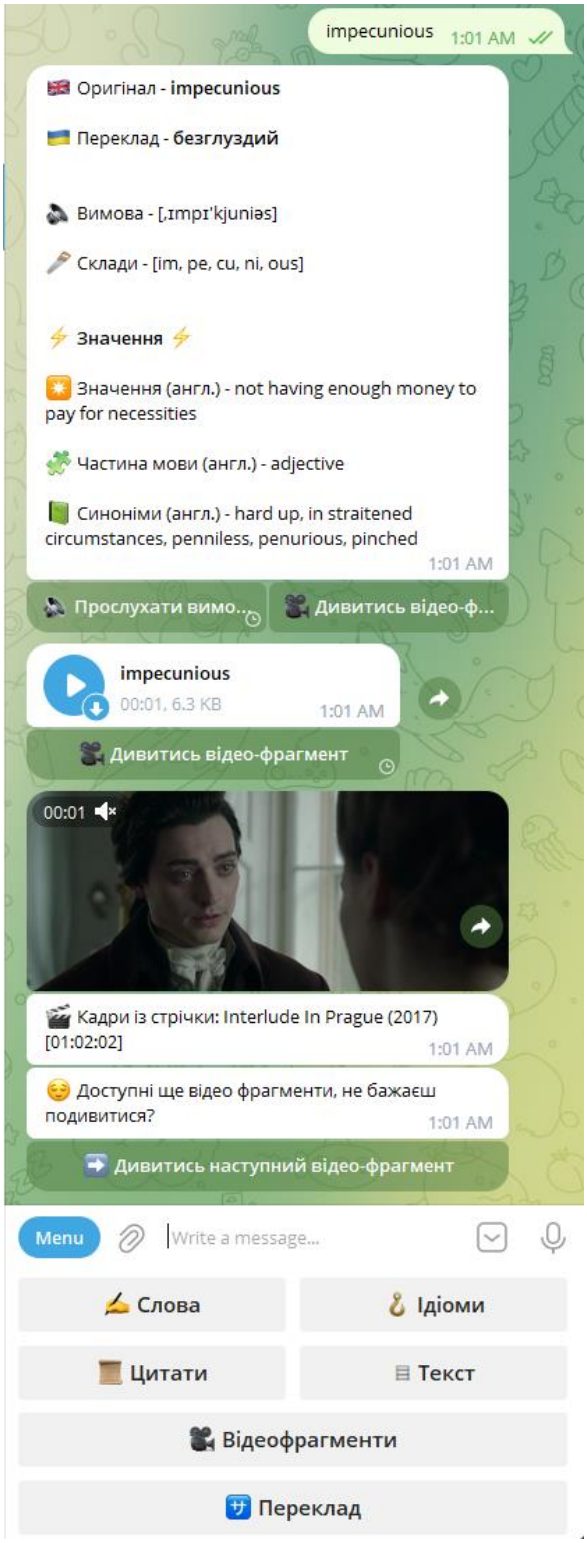

Рисунок 3.20 – Приклад роботи функціоналу перекладача в боті

Рисунок 3.21 – Приклад роботи Shorts Scene

3.12 Stats Scene

Сцена дозволяє отримати статистику користувача в будь-який момент у вигляді повідомлення. Приклад повідомлення наведено на рисунку 3.22. Статистика складається з таких даних:

- інтервал отримання повідомлень;

- рівень знань мови;
- початок навчання з ботом;
- останнє вивчення;
- кількості показаних слів;
- кількості показаних ідіом;
- кількості показаних цитат;
- кількості показаних текстів;
- кількості показаних відеофрагметів;
- кількості запитів на вимову.

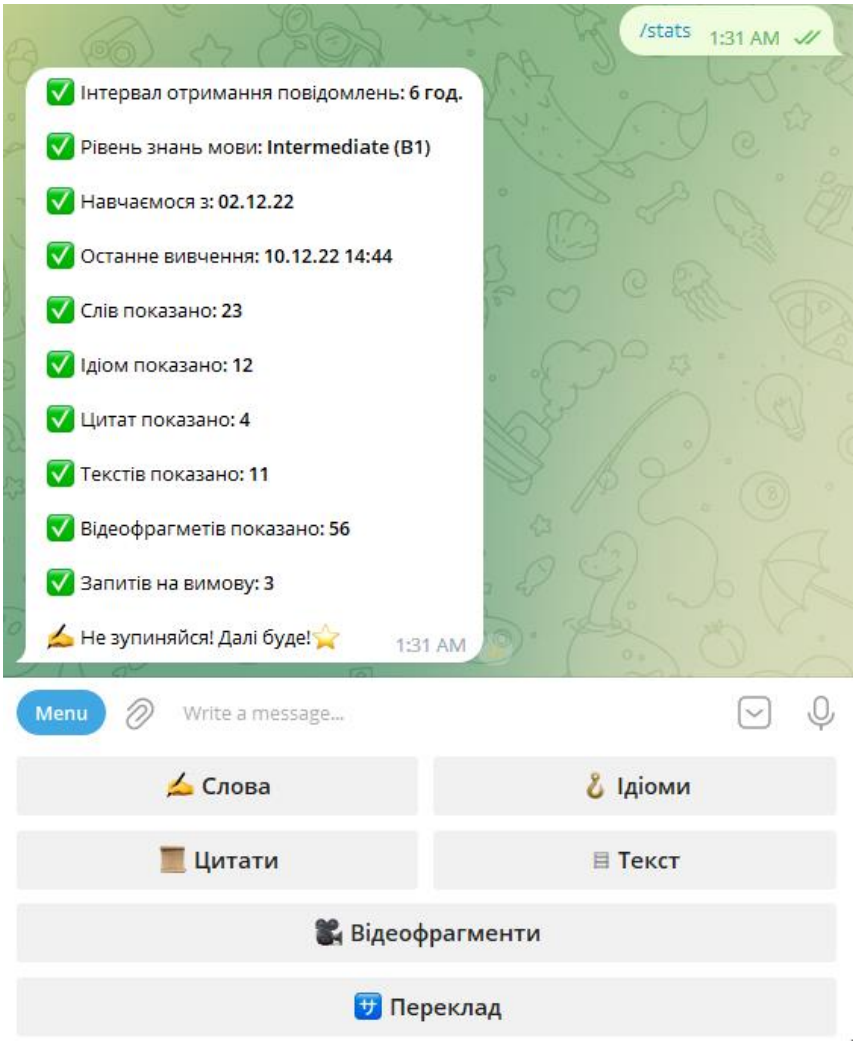

Рисунок 3.22 – Приклад відображення статистики користувача

4 ТЕСТУВАННЯ СТВОРЕНОГО ФУНКЦІОНАЛУ ТА ПОРІВНЯННЯ З АНАЛОГАМИ

Тестування програмного забезпечення — це процес, що направлений на виявлення рівня якості створеного функціоналу та має на меті його покращення, також тестуванням називають перевірку відповідності створеного функціоналу до технічних вимог, що були попередньо написані для нього.

Залежно від специфіки, стадії та потреб продукту, тестування може розподілятися на такі типи:

- функціональне тестування (functional testing);
- нефункціональне тестування (non-functional testing);
- структурне тестування (structural testing);
- тестування змін (change related testing).

4.1 Функціональне тестування

Функціональне тестування[25] на сьогоднішній день є одним із найпопулярніших видів тестування, адже таки вид тестування покриває будь-який тип проекту та є досить гнучким у порівнянні з іншими видами.

Функціональне тестування включає в себе такі елементи:

- опрацювання вхідної документації;
- опрацювання бізнес-вимог функціоналу;
- опрацювання вхідних/вихідних даних;
- написання та опрацювання тест-кейсів;
- опрацювання отриманих результатів та їх співставлення з вимогами.

Функціональне тестування проводиться відповідно до специфікації та на основі бізнес-вимог[18].

Цей тип тестування має такі переваги:

- опрацювання сценаріїв, що буде проходити реальний користувач;
- наближення до дій реального користувача системи.

Також присутні наступні недоліки:

- значні часові витрати;
- через інкрементальний тип розробки та можливий рефакторінг певного функціоналу потрібне повторне тестування.

4.2 Нефункціональне тестування

В межах функціонального тестування ми перевіряємо працездатність системи в цілому, а от в межах нефункціонального тестування опрацьовуються властивості системи які не відносяться до її основного функціоналу але складають її додаткову частину.

Нефункціональне поділяється на підвиди:

- тестування стабільності системи (reliability testing) – виявлення падінь функціоналу в наслідок його використання;
- тестування юзабіліті (usability testing) – виявлення слабких моментів в UX;
- тестування ефективності системи (efficiency testing) – виділення окремих функціональних частин та їх бенчмаркінг;
- тестування ремонтопридатності (maintainability testing) – визначення кількості потрібних ресурсів для підтримання працездатності системи;
- тестування портативності (portability testing) – тестування можливості перенесення функціоналу з одного середовища в інше;
- тестування відповідності критеріям готовності (acceptance testing) – перевірка створеного функціоналу на відповідність критеріям готового продукту;
- тестування документації (documentation testing) – тестування документообігу проекту;

- тестування витривалості функціоналу (endurance testing) – тестування системи при високому навантаженні протягом тривалого періоду часу з метою вивчення її поведінки;
- тестування навантаження (load testing) – перевірка поведінки системи під нормальним навантаженням.
- тестування продуктивності (performance testing) – тестування швидкості роботи розробленого функціоналу;
- тестування сумісності (compatibility testing) – тестування при різних варіаціях середовищ та апаратної частини;
- тестування безпеки (security testing) – перевірка безпеки додатку та даних, що він опрацьовує;
- об'ємне тестування (volume testing) – тестування з використання в умовах реальних об’ємів даних;
- стрес тестування (stress testing) – тестування в умовах високого та максимального навантажень;
- тестування швидкості відновлення (recovery testing) – тестування часового простору необхідного для відновлення повної функціональності;
- тестування локалізації (localization testing) – тестування в умовах використання різних мовних інтерфейсів функціоналу.

4.3 Структурне тестування

Перевірка процесів «під капотом». Опрацювання різних компонентів системи[18]. Відноситься до тестування за принципом «gray box» чи «white box».

Структурне тестування включає наступне:

- строкове покриття (statement Coverage) – тестування операторів;
- покриття шляху (path coverage) – тестування роутингу функціоналу до відповідності критеріям;

- покриття рішення (branch coverage) – тестування умов розгалуження.

Переваги:

- проведення рефакторингу кодової бази;
- виявлення та попередження майбутніх багів;
- тестування функціоналу зсередини.

Недоліки:

- потрібні знання кодової бази;
- необхідність більш висококваліфікованого спеціаліста.

4.4 Тестування змін

Регресійне тестування (regression testing)[26] – тестування вже існуючого функціонала після його оновлення до наступної весії (інтеграції чи збірки).

Переваги:

- перевірка працездатності функціоналу після нової інтеграції;
- є можливість автоматизованих тестів;
- можливість виявлення попередньо пропущених вразливостей.

Недоліки:

- великі витрати часу.

Повторне тестування (retesting) – тестування, що проводиться як повторна перевірка функціоналу, в якому виправили вразливість.

Переваги:

- підвищення загальної якості функціоналу;
- відносно малі затрати часу;
- не вимагає додаткової конфігурації середовища.

Недоліки:

- повторне проходження одного й того ж самого функціоналу.

4.5 Тестування розробленого функціоналу

У різні цикли розробки функціоналу застосовувались різні види тестування. Весь функціонал було піддано регресійному тестуванні при завершенні його розробки. Етапи тестування наведені у таблиці 4.1.

Таблиця 4.1 – Тестування розробленого функціоналу

Цикл розробки	Тип тестування	Результат тестування
Ініціалізація проекту	Функціональне тестування	Підключення до БД, створення таблиці, підключення сервісів
Ініціалізація боту	Функціональне тестування	Зв'язок з ботом, реагування бота на базові дії коритувача
Підключення API	Функціональне тестування Структурне тестування	Отримання відповіді (string) від randomword; отримання відповіді (object) від words; отримання відповіді (object) від Play Phrase; отримання відповіді (string) від translate google.
Розробка Level Scene	Функціональне тестування	Готовий діалог в межах конфігурації рівня знань.
Розробка Schedule Scene	Функціональне тестування	Готовий діалог в межах конфігурації частоти повідомлень.
Виявлення проблем та їх усунення	Повторне тестування	Усунуто вразливості унаслідок різноманітних дій користувача (перехід між сценами)
Розробка Word, Text, Idioms, Quotes Scenes	Функціональне тестування	Готовий функціонал запиту до API, парсинг відповідей, відтворення інформації для користувача
Перевірка роботи функціоналу після імплементації текстових сцен	Регресійне тестування Тестування юзабіліті Тестування стабільності системи	Виявлення проблем та робота над їх усуненням

Продовження таблиці 4.1 - Тестування розробленого функціоналу

Усунення проблем після регресійного тестування	Повторне тестування	Усунуто вразливості унаслідок різноманіття типів відповідей API, оброблено додаткові сценарії користувача
Розробка Shorts Scene	Функціональне тестування	Готовий діалог в межах запитів користувача на відеофрагменти.
Розробка Translate Scene	Функціональне тестування	Готовий діалог в межах перекладу тексту.
Розробка Stats Scene	Функціональне тестування	Готовий діалог в межах відображення статистики користувача.
Рефакторинг коду	Структурне тестування Регресійне тестування	Конфігурація додаткових змінних, оптимізація кодової бази.

4.6 Порівняльна характеристика з системами аналогами

Кожна платформа для навчання має свої переваги та недоліки. Деякі з них грають ключову роль у виборі платформи для навчання, адже технології та методи у них відрізняються. Сучасна людина здебільшого має обмаль часу на навчальну діяльність, тому важливим фактором є універсальність та гнучкість системи.

4.6.1 Тип ресурсу

У більшості навчальних платформ тип ресурсу – веб-сайт. При такому сценарії користувачу потрібно щоразу використовувати браузер, шукати веб-сайт(або додавати його у закладки), проходити авторизацію, шукати розділ чи момент де користувач зупинився останній раз. У випадку з ботом все набагато простіше, користувач відкриває звичний йому месенджер та просто знаходить

діалог з ботом, історією його останнього моменту навчання слугують останні повідомлення бота, тобто користувач має змогу щоразу переглядати попередній урок та пригадати або повторити його.

У випадку з нативним додатком, він потребує попереднього встановлення.

4.6.2 Авторизація

Щоб розпочати роботу з системою типу веб-сайт чи нативний додаток, Вам в будь-якому випадку потрібно буде пройти реєстрацію[27]. Це досить нудний, а іноді і довгий процес. Ось кроки, що потрібно виконати для авторизація в таких системах:

1) перехід на сторінку реєстрації;
2) введення даних;
3) генерація паролю, який ще потрібно тримати в голові для повторної авторизації у майбутньому);
4) перевірка «на робота»;
5) перехід у сервіс поштової скриньки та авторизація там;
6) пошук листа та підтвердження реєстрації аккаунта за допомогою посилання, що міститься у листі;
7) вхід у особистий кабінет після проходження процесу реєстрації.

В свою чергу, Telegram у користувача вже авторизований, тому що людина активно користується ним для підтримання щоденного спілкування з іншими. Сам бот не потребує ніякої авторизації, всі дані він має змогу взяти з внутрішнього API Telegram. Як результат, для початку роботи з ботом потрібно натискання чи введення однієї команди «start».

4.6.3 Повідомлення

У випадку з веб-ресурсом, користувачу потрібно надати доступ браузеру для відправлення повідомлень. Зазвичай, браузер пропонує це зробити у окремому повідомленні, але якщо його закрити чи відмовити в дозволі, у такому випадку потрібно буде переходити у налаштування браузера, шукати заборону на повідомлення для певного веб-сайту та видаляти її, потім займати пошуком того ,як надати дозвіл. У додатку до вищесказаного, можливо ще відмітити, що механізм надання подібного дозволу у різних браузерах може відрізнятися, тому користувачеві потрібно буде розбиратися з цим власноруч. У нативних додатках також потрібно підтвердити дозвіл на повідомлення, але якщо схибити чи проігнорувати його, історія з пошуком налаштувань може повторитися і тут.

На противагу веб-додатку, бот використовую стандартний функціонал месенджера, що абсолютно точно нівелює навіть можливість виникнення подібної проблеми.

4.6.4 Складність створення

Для створення веб-сайту чи нативного додатку потрібно значно більше зусиль, тому що при його створенні вам потрібно займатися розробкою клієнтської частини, тобто UI та UX, що може зайняти навіть більше часу ніж написання серверної частини функціоналу[28]. Також неможливо не відмітити, що при розробці сайтів та мобільних додатків враховується різноманіття девайсів, які може використовувати користувач для взаємодії з цією системою. Адаптація системи до подібного – неймовірно трудомістка процедура, до того ж, гарантій, що така система (після адаптації) буде коректно виглядати на якомусь недавно представленому девайсі з нестандартними пропорціями сторін – немає.

Водночас при використанні месенджера усі ці проблеми уже вирішені його розробниками, які також будуть надавати постійні оновлення та виправляти проблеми співвідношення сторін для різних девайсів.

4.6.5 Витрати

При використанні веб-сайту, розробнику системи потрібно буде придбати доменне ім’я, місце на хостингу.

У випадку мобільного застосунку, розробнику потрібно буде проходити верифікацію у App Store та Google Play, якщо з останнім все доволі просто, то от App Store має досить жорсткі правила, що можуть призвести до рефакторингу значної частини функціональності після завершення її розробки.

Варіант з ботом – найлегший, тому як потребує лише створення бота через внутрішнє API Telegram, що є умовно легкою процедурою та займає декілька хвилин часу.

4.7 Висновки

Всі визначені тести було пройдено і порівняно бот з аналогами.

Бот має перевагу у багатьох позиціях, які є ключовими для сучасної людини: гнучкість, надійність, вартість, доступність, швидкість. А дешевизна розробки призведе до появи значної кількості навчальних ресурсів на його базі, активну конкуренцію і як наслідок, справедливу ціну.

5 ЕКОНОМІЧНА ЧАСТИНА

Науково-технічна розробка має право на існування та впровадження, якщо вона відповідає вимогам часу, як в напрямку науково-технічного прогресу та і в плані економіки. Тому для науково-дослідної роботи необхідно оцінювати економічну ефективність результатів виконаної роботи.

5.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту дослідження за темою «Розробка багатофункціонального телеграм боту для вивчення англійської мови з використанням сторонніх API» є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями [19]. (Таблиця 5.1).

Таблиця 5.1 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	5	5	5
2. Ринкові переваги (наявність аналогів)	1	2	2
3. Ринкові переваги (ціна продукту)	1	1	1
4. Ринкові переваги (технічні властивості)	1	2	2
5. Ринкові переваги (експлуатаційні витрати)	2	2	2

Продовження таблиці 5.1.

6. Ринкові перспективи (розмір ринку)	3	3	3
7. Ринкові перспективи (конкуренція)	3	2	3
8. Практична здійсненність (наявність фахівців)	5	5	5
9. Практична здійсненність (наявність фінансів)	3	4	3
10. Практична здійсненність (необхідність нових матеріалів)	5	5	5
11. Практична здійсненність (термін реалізації)	5	4	4
12. Практична здійсненність (розробка документів)	4	5	4
Сума балів	38	40	39
Середньоарифметична сума балів $СБ_c$	39,0		

За результатами розрахунків, наведених в таблиці 5.1, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в [19].

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Розробка багатофункціонального телеграм боту для вивчення англійської мови з використанням сторонніх API» становить 39,0 бала, що, відповідно до [19], свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

5.2 Визначення рівня конкурентоспроможності розробки

В процесі визначення економічної ефективності науково-технічної розробки також доцільно провести прогноз рівня її конкурентоспроможності за сукупністю параметрів, що підлягають оцінюванню.

Загальні технічні та економічні характеристики розробки представлено в таблиці 5.2.

Таблиця 5.2 – Основні техніко-економічні показники аналога та розробки, що проектується

Показники (параметри)	Одиниця вимірю-вання	Аналог	Проектований пристрій	Відношення параметрів нової розробки до аналога	Питома вага показника
Використовуваний об'єм ОЗУ	МБ	540	400	1,35	0,2
Використовуваний об'єм ПЗУ	МБ	200	20	10	0,25
Швидкість розгортання ресурсу	бал	4	8	2	0,2
Рівень пізнаваності домену	бал	8	3	2,67	0,1
Дружність інтерфейсу	бал	7	9	1,23	0,25
Експлуатаційні витрати (вартість підписки)	грн.	125	75	0,6	0,5
Запланована вартість бота	грн.	920	630	0,68	0,5

Одиничний параметричний індекс розраховуємо за формулою [19]:

$$q_i = \frac{P_i}{P_{базі}}. \quad (5.1)$$

де q_i – одиничний параметричний індекс, розрахований за i-м параметром;

P_i – значення i-го параметра виробу;

$P_{базі}$ – аналогічний параметр аналога, з яким проводиться порівняння.

Нормативні параметри оцінюємо показником, який отримує одне з двох значень: 1 – пристрій відповідає нормам і стандартам; 0 – не відповідає.
За нормативними параметрами розроблюваний пристрій відповідає вимогам ДСТУ, тому $I_{нп} = 1$.

Значення групового параметричного індексу за технічними параметрами визначаємо з урахуванням вагомості (частки) кожного параметра [19]:

$$I_{ТП} = \sum_{i=1}^{n} q_i \cdot \alpha_i \ , \qquad (5.3)$$

де $I_{тп}$ – груповий параметричний індекс за технічними показниками (порівняно з виробом-аналогом);

q_i – одиничний параметричний показник i-го параметра;

α_i – вагомість i-го параметричного показника, $\sum_{i=1}^{n} \alpha_i = 1$;

n – кількість технічних параметрів, за якими оцінюється конкурентоспроможність.

Проведемо аналіз параметрів згідно даних таблиці 5.2.

$$I_{тп} = 1{,}35 \cdot 0{,}2 + 10 \cdot 0{,}25 + 2 \cdot 0{,}2 + 2{,}67 \cdot 0{,}1 + 1{,}23 \cdot 0{,}25 = 3{,}74.$$

Груповий параметричний індекс за економічними параметрами розраховуємо за формулою [19]:

$$I_{ЕП} = \sum_{i=1}^{m} q_i \cdot \beta_i \ , \qquad (5.4)$$

де $I_{ЕП}$ – груповий параметричний індекс за економічними показниками;

q_i – економічний параметр i-го виду;

β_i – частка i-го економічного параметра, $\sum_{i=1}^{m} \beta_i = 1$;

m – кількість економічних параметрів, за якими здійснюється оцінювання. Проведемо аналіз параметрів згідно даних таблиці .

$$I_{ЕП}=0,6 \cdot 0,5 + 0,68 \cdot 0,5 = 0,64.$$

На основі групових параметричних індексів за нормативними, технічними та економічними показниками розрахуємо інтегральний показник конкурентоспроможності за формулою [19]:

$$K_{ІНТ} = I_{НП} \cdot \frac{I_{ТП}}{I_{ЕП}}, \tag{5.5}$$

$$K_{ІНТ} = 1 \cdot 3,74 / 0,64 = 5,85.$$

Інтегральний показник конкурентоспроможності $К_{ІНТ} > 1$, отже розробка переважає відомі аналоги за своїми техніко-економічними показниками.

5.3 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему «Розробка багатофункціонального телеграм боту для вивчення англійської мови з використанням сторонніх API», під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

5.3.1 Витрати на оплату праці

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою [19]:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \quad (5.6)$$

де k – кількість посад дослідників залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дн.;

T_p – середнє число робочих днів в місяці, T_p=20 дні.

$$З_о = 17890{,}00 \cdot 40 / 20 = 35780{,}00 \; грн.$$

Проведені розрахунки зведемо до таблиці 5.3.

Таблиця 5.3 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник проекту	17890,00	894,50	40	35780,00
Інженер-програміст 1-ї категорії	16900,00	845,00	38	32110,00
Консультант (викладач іноземної мови)	15250,00	762,50	5	3812,50
Технік	7600,00	380,00	20	7600,00
Всього				79302,50

Основна заробітна плата робітників

Витрати на основну заробітну плату робітників ($З_р$) (таблиця 5.4) за відповідними найменуваннями робіт НДР на тему «Розробка багатофункціонального телеграм боту для вивчення англійської мови з використанням сторонніх API» розраховуємо за формулою:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i , \tag{5.7}$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}}, \tag{5.8}$$

де M_M – розмір мінімальної місячної заробітної плати, приймемо *Мм*=6700,00 грн;

K_i – коефіцієнт міжкваліфікаційного співвідношення (табл. Б.2, додаток Б) [19];

K_c – мінімальний коефіцієнт співвідношень місячних тарифних ставок;

T_p – середнє число робочих днів в місяці, приблизно T_p = 20 дн;

$t_{зм}$ – тривалість зміни, год.

$$C_1 = 6700{,}00 \cdot 1{,}10 \cdot 1{,}65 / (20 \cdot 8) = 76{,}00 \text{ грн.}$$

$$З_{р1} = 76{,}00 \cdot 8{,}64 = 656{,}67 \text{ грн.}$$

Таблиця 5.4 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Підготовка робочого місця розробника програмного забезпечення	8,64	2	1,10	76,00	656,67
Інсталяція програмного забезпечення середовища розробки	5,60	3	1,35	93,28	522,35
Компіляція програмних блоків	4,80	5	1,70	117,46	563,81
Всього					1742,82

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (5.9)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 10%.

$З_{дод} = (79302,50 + 1742,82) \cdot 10 / 100\% = 8104,53$ *грн.*

5.3.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (5.10)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%.

Зн = (79302,50 + 1742,82 + 8104,53) · 22 / 100% = 19612,97 грн.

5.3.3 Сировина та матеріали

Витрати на матеріали (*М*), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$M = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} В_j \cdot Ц_{вj}, \quad (5.11)$$

де $Н_j$ – норма витрат матеріалу *j*-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу *j*-го найменування, грн/кг;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ = *1,1 ... 1,15*);

$В_j$ – маса відходів *j*-го найменування, кг;

$Ц_{вj}$ – вартість відходів *j*-го найменування, грн/кг.

$М_1$ = 3,0 · 292,00 · 1,11 - 0 · 0 = 972,36 грн.

Проведені розрахунки зведемо до таблиці 5.5.

Таблиця 5.5 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн	Норма витрат, кг	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Офісний папір А4 500	292,00	3,0	0	0	972,36
Папір для записів А5 250	161,00	4,0	0	0	714,84
Органайзер офісний	198,00	4,0	0	0	879,12
Набір канцелярський офісний	216,00	3,0	0	0	719,28
Картридж для принтера	1110,00	1,0	0	0	1232,10
Диск оптичний CD-RW	22,10	2,0	0	0	49,06
Flesh-пам'ять DATA 32 GB	389,00	1,0	0	0	431,79
Всього					4998,55

5.3.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему «Розробка багатофункціонального телеграм боту для вивчення англійської мови з використанням сторонніх API» відсутні.

5.3.5 Спецустаткування для наукових (експериментальних) робіт

Балансову вартість спецустаткування розраховуємо за формулою:

$$B_{спец} = \sum_{i=1}^{k} Ц_i \cdot C_{пр.i} \cdot K_i \ , \qquad (5.12)$$

де $Ц_i$ – ціна придбання одиниці спецустаткування даного виду, марки, грн;

$C_{пр.i}$ –кількість одиниць устаткування відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує доставку, монтаж, налагодження устаткування тощо, (K_i = *1,10…1,12);*

k – кількість найменувань устаткування.

$$B_{спец} = 31500{,}00 \cdot 1 \cdot 1{,}12 = 35280{,}00 \text{ грн.}$$

Отримані результати зведемо до таблиці 5.6.

Таблиця 5.6 – Витрати на придбання спецустаткування по кожному виду

Найменування устаткування	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Серверне обладнання на основі EOM Brain A345-F71BC	1	31500,00	35280,00
Всього			35280,00

5.3.6 Програмне забезпечення для наукових (експериментальних) робіт

Балансову вартість програмного забезпечення розраховуємо за формулою:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \ , \qquad (5.13)$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.i}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, ($K_i = 1,10…1,12$);

k – кількість найменувань програмних засобів.

$$B_{прг} = 7950,00 \cdot 1 \cdot 1,1 = 8745,00 \text{ грн.}$$

Отримані результати зведемо до таблиці 5.7.

Таблиця 5.7 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Прикладне програмне забезпечення розробки ПЗ	1	7950,00	8745,00
Всього			8745,00

5.3.7 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \qquad (5.14)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$$A_{обл} = (26320{,}00 \cdot 2) / (2 \cdot 12) = 2193{,}33 \text{ грн.}$$

Проведені розрахунки зведемо до таблиці 5.8.

5.3.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію ($B_е$) розраховуємо за формулою:

$$B_е = \sum_{i=1}^{n} \frac{W_{уі} \cdot t_i \cdot Ц_е \cdot K_{впі}}{\eta_i}, \quad (5.15)$$

де $W_{уі}$ – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_е$ – вартість 1 кВт-години електроенергії, грн; приймемо $Ц_е = 6{,}15$ грн;

$K_{впі}$ – коефіцієнт, що враховує використання потужності, $K_{впі} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

$$B_е = 0{,}25 \cdot 320{,}0 \cdot 6{,}15 \cdot 0{,}95 / 0{,}97 = 492{,}00 \text{ грн.}$$

Проведені розрахунки зведемо до таблиці 5.9.

Таблиця 5.8 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Персональний комп'ютер проведення розробки ПЗ	26320,00	2	2	2193,33
Робоче місце інженера-розробника ПЗ	9200,00	5	2	306,67
Пристрої передачі даних	8690,00	4	2	362,08
Пристрій виводу інформації	8650,00	5	2	288,33
Оргтехніка	6740,00	4	2	280,83
Приміщення лабораторії	700000,00	20	2	5833,33
ОС Windows 11	8600,00	2	2	716,67
Прикладний пакет Microsoft Office 2019	7960,00	2	2	663,33
Всього				10644,58

5.3.9 Службові відрядження

Витрати за статтею «Службові відрядження» відсутні

Таблиця 5.9 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Персональний комп'ютер проведення розробки ПЗ	0,25	320,0	492,00
Робоче місце інженера-розробника ПЗ	0,12	300,0	221,40
Пристрої передачі даних	0,01	250,0	15,38
Пристрій виводу інформації	0,42	15,0	38,75
Оргтехніка	0,50	5,0	15,38
Серверне обладнання на основі EOM Brain A345-F71BC	0,32	320,0	629,76
Всього			1412,66

5.3.10 Витрати на роботи, які виконують сторонні підприємства, установи і організації

Витрати розраховуємо як 30…45% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{сп} = (З_о + З_р) \cdot \frac{H_{сп}}{100\%}, \quad (5.16)$$

де $Н_{сп}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації», приймемо $Н_{сп}$= 32%.

$$B_{сп} = (79302,50 + 1742,82) \cdot 32 / 100\% = 25934,50 \text{ грн.}$$

5.3.11 Інші витрати

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (5.17)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 55%.

$I_в$ = (79302,50 + 1742,82) · 55 / 100% = 44574,93 грн.

5.3.12 Накладні (загальновиробничі) витрати

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$В_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (5.18)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 100%.

$В_{нзв}$ = (79302,50 + 1742,82) · 100 / 100% = 81045,32 грн.

Витрати на проведення науково-дослідної роботи на тему «Розробка багатофункціонального телеграм боту для вивчення англійської мови з використанням сторонніх API» розраховуємо як суму всіх попередніх статей витрат за формулою:

$$В_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв}. \quad (5.19)$$

$$B_{заг} = 79302,50 + 1742,82 + 8104,53 + 19612,96762 + 4998,55 + 0,00 + 35280,00 + 8745,00 + 10644,58 + 1412,66 + 0,00 + 25934,50 + 44574,93 + 81045,32 = 321398,36 \text{ грн.}$$

Загальні витрати $ЗВ$ на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{B_{заг}}{\eta}, \tag{5.20}$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η =0,9.

$$ЗВ = 321398,36 / 0,9 = 357109,29 \text{ грн.}$$

5.4 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

Результати дослідження проведені за темою «Розробка багатофункціонального телеграм боту для вивчення англійської мови з використанням сторонніх API» передбачають комерціалізацію протягом 4-х років реалізації на ринку (таблиця 5.10).

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

ΔN – збільшення кількості споживачів продукту, у періоди часу, що аналізуються, від покращення його певних характеристик;

Таблиця 5.10 - Комерціалізація чат боту

Показник	1-й рік	2-й рік	3-й рік	4-й рік
Збільшення кількості споживачів, осіб	6000	15000	25000	10000

N – кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки, приймемо 60000 осіб;

$Ц_б$ – вартість програмного продукту у році до впровадження результатів розробки, приймемо 900,00 грн;

$\pm\Delta Ц_o$ – зміна вартості програмного продукту від впровадження результатів науково-технічної розробки, приймемо зниження на -79,95 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 4-х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою [19]:

$$\Delta П_i = (\pm\Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \qquad (5.21)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2022 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0{,}8333$;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту). Приймемо ρ =35%;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2022 році $\vartheta = 18\%$;

Збільшення чистого прибутку 1-го року:

$\Delta П_1$ = (-79,95·60000,00+820,05·6000)·0,83·0,35·(1-0,18/100%)=29371,29 грн.

Збільшення чистого прибутку 2-го року:

$\Delta П_2 =$(-79,95·60000,00+820,05·21000)·0,83·0,35·(1-0,18/100%)=2959532,95 грн.

Збільшення чистого прибутку 3-го року:

$\Delta П_3 =$(-79,95·60000,00+820,05·46000)·0,83·0,35·(1-0,18/100%)=7843135,71 грн.

Збільшення чистого прибутку 4-го року:

$\Delta П_4 =$(-79,95·60000,00+820,05·56000)·0,83·0,35·(1-0,18/100%)=9796576,82 грн.

Приведена вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (5.22)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =0,19;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$ПП =$29371,29/(1+0,19)^1+2959532,95/(1+0,19)^2+7843135,71/(1+0,19)^3+
+9796576,82/(1+0,19)^4=24681,76+2089918,05+4654240,76+4885246,73=
=11654087,30 грн.

Величина початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ, \quad (5.23)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=2,2;

$ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 357109,29 грн.

$PV = k_{інв} \cdot ЗВ = 2{,}2 \cdot 357109{,}29 = 785640{,}44$ грн.

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV, \quad (5.24)$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 11654087,30 грн;

PV – теперішня вартість початкових інвестицій, 785640,44 грн.

$$E_{абс} = ПП - PV = 11654087{,}30 - 785640{,}44 = 10868446{,}87 \text{ грн.}$$

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1, \quad (5.25)$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, 10868446,87 грн;

PV – теперішня вартість початкових інвестицій, 785640,44 грн;

$T_{ж}$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 4 роки.

$$E_{в} = \sqrt[Tж]{1+\frac{E_{абс}}{PV}} - 1 = (1+10868446{,}87/785640{,}44)^{1/4} = 0{,}96.$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$:

$$\tau_{мін} = d + f, \qquad (5.26)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2022 році в Україні d =0,1;

f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,25.

$\tau_{мін}$ = 0,1+0,25 = 0,35 < 0,96 свідчить про те, що внутрішня економічна дохідність інвестицій $E_{в}$, вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою «Розробка багатофункціонального телеграм боту для вивчення англійської мови з використанням сторонніх API» доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_{в}}, \qquad (5.27)$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій.

$$T_{ок} = 1 / 0,96 = 1,04 \text{ р.}$$

$T_{ок} <$ 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

5.5 Висновки до розділу

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Розробка багатофункціонального телеграм боту для вивчення англійської мови з використанням сторонніх API» становить 39,0 бала, що, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

При оцінюванні рівня конкурентоспроможності, згідно узагальненого коефіцієнту конкурентоспроможності розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 5,85 рази.

Також термін окупності становить 1,04 р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Розробка багатофункціонального телеграм боту для вивчення англійської мови з використанням сторонніх API».

ВИСНОВКИ

В даній магістерській кваліфікаційній роботі було здійснено аналіз існуючих платформ навчального напрямку для вивчення англійської мови. Особливу увагу було приділено методам подачі інформації подібних платформ, потокам відображення даних, наявним функціоналом для навчання. Оцінено швидкість роботи даних систем-аналогів, швидкість завантаження та рендерингу даних, зручність використання користувачем при різних сценаріях та приблизні витрати на розробку цих систем.

Було проаналізовано, перебрано стек технологій та API, що використовуються для створення подібного функціоналу, оцінено витрати, які включає в себе та чи інша технологія. Після детального аналізу було обрано технології та API які якнайкраще вписуються в архітектуру розробленого функціоналу та забезпечують якомога меншу затримку в отриманні результатів їх роботи. У зв’язку з принципово новою специфікою навчання на базі месенджера багато часу було витрачено на вирішення питань інтеграції в систему месенджера та розробки правильної масштабованої архітектури проекту. Була створена діаграма варіантів використання розробленого функціоналу.

У відповідністю з діаграмою варіантів використання була розроблена повна документація проекту. Основною сутністю проекту є сцена. Сцена має на меті вести діалог, що належить лише до одного напрямку (вивчення слова – перша сцена, вивчення ідіом – друга сцена). Згідно з діаграмою було розроблено структуру сцен, кожна з яких має свою чітку зону відповідальності. Також було створено й іншу кодову базу, яка має на меті обслуговування функціоналу сцен, роботу з API, запити до бази даних тощо. Важливо відмітити що кодова база була оптимізована під її подальше розширення та інтеграцію нового функціоналу. Це потрібно для полегшення інтегрування нового інструментарію у майбутньому. Для розробника

підключення нового функціоналу зводиться до простого приєднання нового модулю до вже існуючих.

У результаті виконання даної роботи було створено готовий клієнтоорієнтований чат-бот для вивчення англійської мови, що має змогу вести комунікацію з людиною у будь-який комфортний для неї час. Для забезпечення потреб датасетів та іншого функціоналу були використанні сторонні API.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Семенюк А.В., Богач І.В. «Розробка багатофункціонального телеграм боту для вивчення англійської мови з використанням сторонніх API» в Матеріали конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2023)», Вінниця, 2023. [Електронний ресурс]. Режим доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2023/paper/view/16874.

2. Chatbot [Електронний ресурс]. URL: https://www.techtarget.com/searchcustomerexperience/definition/chatbot

3. Чат боти: плюси і мінуси [Електронний ресурс]. URL: https://ukr-bot.com/plyusi-ta-minusi-chat-botiv/

4. 10 Powerful Use Cases Of Educational Chatbots In 2022 [Електронний ресурс]. URL: https://yellow.ai/chatbots/use-cases-of-chatbots-in-education-industry/

5. Найкращі додатки для вивчення англійської мови [Електронний ресурс]. URL: https://englishprime.ua/uk/prilozheniya-dlya-izucheniya-anglijskogo

6. 4 Telegram-бота для прокачки англійського [Електронний ресурс]. URL: https://ain.ua/ru/2018/10/30/4-telegram-bota-dlya-prokachki-anglijskogo

7. NestJS [Електронний ресурс]. URL: https://docs.nestjs.com

8. ПРИЧИНИ ВИБРАТИ NEST.JS ДЛЯ ВАШОГО ПРОЄКТУ [Електронний ресурс]. URL: https://brander.ua/technologies/nestjs/

9. JavaScript - №1, хто за нею? [Електронний ресурс]. URL: https://dev.ua/news/movy-1669291519

10. Чому варто вивчати JavaScript? [Електронний ресурс]. URL: https://qagroup.com.ua/publications/why-learn-javascript/

11. Навіщо розробникам переходити на JavaScript [Електронний ресурс]. URL: https://ain.ua/special/zachem-perehodit-na-javascript

12. Axios [Електронний ресурс]. URL: https://www.npmjs.com/package/axios.

13. API – що це таке простими словами [Електронний ресурс]. Джерело інформації: https://xn--90aamhd6acpq0s.xn--j1amh/teoriya/api-shcho-tse-take-

prostymy-slovamy. URL: https://xn--90aamhd6acpq0s.xn--j1amh/teoriya/api-shcho-tse-take-prostymy-slovamy/

14. Google Translate API [Електронний ресурс]. URL: https://apix-drive.com/ua/google-translate-api

15. Words API [Електронний ресурс]. URL: https://www.programmableweb.com/api/words

16. Build a Dictionary App with the WordsAPI (JavaScript). URL: https://rapidapi.com/blog/build-a-dictionary-app-with-the-wordsapi/

17. RandomWord [Електронний ресурс]. URL: https://randomword.com/

18. ОГЛЯД ВИДІВ ТЕСТУВАННЯ [Електронний ресурс]. URL: https://docs.nestjs.com/

19. В. О. Козловський, О. Й. Лесько, В. В. Кавецький Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : Вінниця : ВНТУ, 2021. 42 с.

20. За даними досліджень, щомісячна аудиторія соцмереж зросла до 4 млрд користувачів [Електронний ресурс]. – URL: https://armyinform.com.ua/2020/10/24/za-danymy-doslidzhen-shhomisyachna-audytoriya-soczmerezh-zrosla-do-4-mlrd-korystuvachiv/

21. Best Way to Structure Your Directory/Code (NestJS) [Електронний ресурс]. URL: https://medium.com/the-crowdlinker-chronicle/best-way-to-structure-your-directory-code-nestjs-a06c7a641401

22. 10 NestJS Folder Structure Best Practices [Електронний ресурс]. URL: https://climbtheladder.com/10-nestjs-folder-structure-best-practices/

23. Telegram APIs [Електронний ресурс]. URL: https://core.telegram.org/api

24. telegraf.js [Електронний ресурс]. URL: https://github.com/telegraf/telegraf

25. Complete Functional Testing Guide With Its Types And Example [Електронний ресурс]. URL: https://www.softwaretestinghelp.com/guide-to-functional-testing/

26. What is Regression Testing? [Електронний ресурс]. URL: https://www.javatpoint.com/regression-testing

27. Authorization [Електронний ресурс]. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Authorization

28. Project Cost Management: Steps, Basics and Benefits [Електронний ресурс]. URL: https://www.ecosys.net/knowledge/project-cost-management/#:~:text=What%20is%20Project%20Cost%20Management,expenditures%20within%20the%20approved%20budget.&text=it's%20completed%20within%20budget.

29. Data Providers [Електронний ресурс]. URL: https://galaxyproject.org/data-providers/

30. Global Digital Communication [Електронний ресурс]. URL: https://www.pewresearch.org/global/2011/12/20/global-digital-communication-texting-social-networking-popular-worldwide/

31. What is REST [Електронний ресурс]. URL: https://restfulapi.net/

ДОДАТКИ

Додаток А (обов'язковий)

Технічне завдання

ЗАТВЕРДЖЕНО

Завідувач кафедри АІІТ

_____

(прізвище та ініціали)

«____» _____2022 року

ТЕХНІЧНЕ ЗАВДАННЯ

на магістерську кваліфікаційну роботу

«Розробка багатофункціонального телеграм боту для вивчення англійської мови з використанням сторонніх API»

08-02.МКР.000.09.000 ТЗ

Виконав: студент 2 курсу, групи 1ІСТ-21м спеціальності 126 – Інформаційні системи та технології

(шифр і назва спеціальності)

Керівник: к.т.н., доцент кафедри АІІТ Богач І. В.

(прізвище та ініціали)

Вінниця ВНТУ 2022

1. Назва та галузь застосування

Магістерська кваліфікаційна робота: «Розробка багатофункціонального телеграм боту для вивчення англійської мови з використанням сторонніх API». Галузь застосування – інформаційні технології.

2. Підстава для проведення розробки

Тема магістерської кваліфікаційна роботи затверджена наказом по ВНТУ №___ від «__» _____ 2022 р.

3. Мета та призначення розробки

Метою роботи є покращення доступного навчального функціоналу для вивчення англійської мови на основі месенджера «Telegram».

4. Джерела розробки

- ISO/IEC 2382-1:1993, Information technology – Vocabulary – Part 1: Fundamental terms.
- ISO/IEC 9126-1:2001, Software engineering – Product quality – Part 1: Quality model.
- ISO/IEC TR 9126-2:2003, Software engineering – Product quality – Part 2: External metrics.
- ISO/IEC TR 9126-3:2003, Software engineering – Product quality – Part 3: Internal metrics.

5. Показники призначення

Необхідні вхідні дані для коректної роботи програми:

- стабільний зв'язок з мережею Інтернет;
- встановлений додаток «Telegram» чи його веб-версія;
- наявність запасу пом’яті на девайсі для можливості кешування медіа;
- посилання на чат-бот.

Результати роботи програми:

- виведення перекладу та основних властивостей слова для вивчення;

- виведення вимови слова у аудіо форматі;
- конфігурація повідомлень та рівня знань англійської мови;
- виведення слова у фрагменті фільму чи відео;
- виведення статистики користувача;
- виведення інформаційних повідомлень.

6. Економічні показники

- Прогнозовані витрати на розробку – не більше 20 тис. грн;
- Абсолютна ефективність розробки – не менше 60 тис. грн;
- Термін окупності витрат для виробника – не більше 1 року.

7. Стадії розробки

1. Розділ 1 « Дослідження предметної галузі » має бути виконаний до 12.10.2022.

2. Розділ 2 « Вибір технологічного стеку та дата-провайдерів » має бути виконаний до 19.10.2022.

3. Розділ 3 « Розробка програмного забезпечення » має бути виконаний до 20.11.2022.

4. Розділ 4 « Тестування створеного функціоналу та порівняння з аналогами» має бути виконаний до 30.11.2022.

5. Економічний розділ має бути виконаний до 03.12.2022.

8. Порядок контролю та приймання

1. Рубіжний контроль. Провести до 05.12.2022.

2. Попередній захист магістерської кваліфікаційної роботи. Провести до 12.12.2020.

3. Захист магістерської кваліфікаційної роботи. Провести в період з 19.12.2022 до 21.12.2022.

Додаток Б (обов'язковий)

Графічна частина

Зав. кафедри АІІТ _____ _____д.т.н., професор, Бісікало О.В.___

(підпис) (науковий ступінь, вчене звання, ініціали та прізвище)

Науковий керівник _____ _____к.т.н., доц. каф.АІІТ Богач І.В.____

(підпис) (науковий ступінь, вчене звання, ініціали та прізвище)

Тех. контроль _____ _____к.т.н., доц. каф.АІІТ Богач І.В.____

(підпис) (науковий ступінь, вчене звання, ініціали та прізвище)

Нормоконтроль _____ _____к.т.н., доц. каф.АІІТ Богач І.В.____

(підпис) (науковий ступінь, вчене звання, ініціали та прізвище)

Опонент _____ _____д.т.н., доц. каф.КСУ Ковтун В.В.____

(підпис) (науковий ступінь, вчене звання, ініціали та прізвище)

Студент гр. 1ІСТ-21м _____ Семенюк А.В.

(підпис) (ініціали та прізвище)

Продовження додатка Б

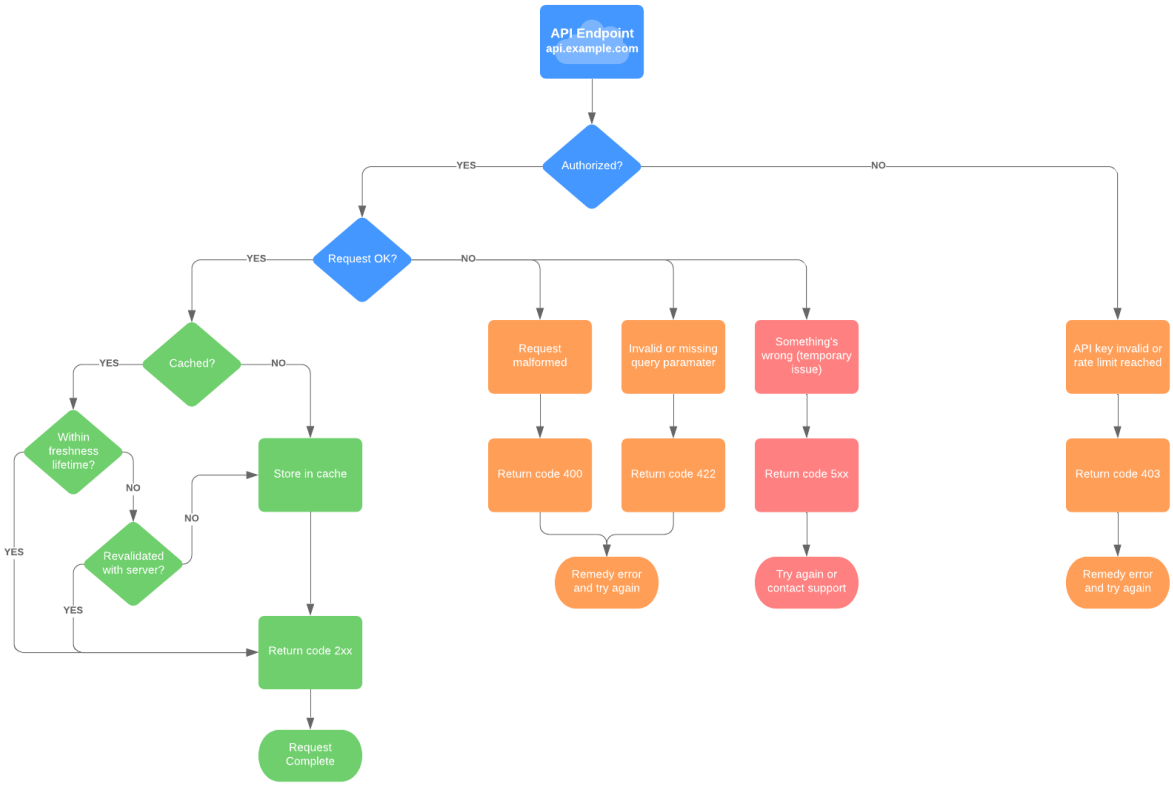

Рисунок Б.1 – UML-діаграма роботи RESTful API

Продовження додатка Б

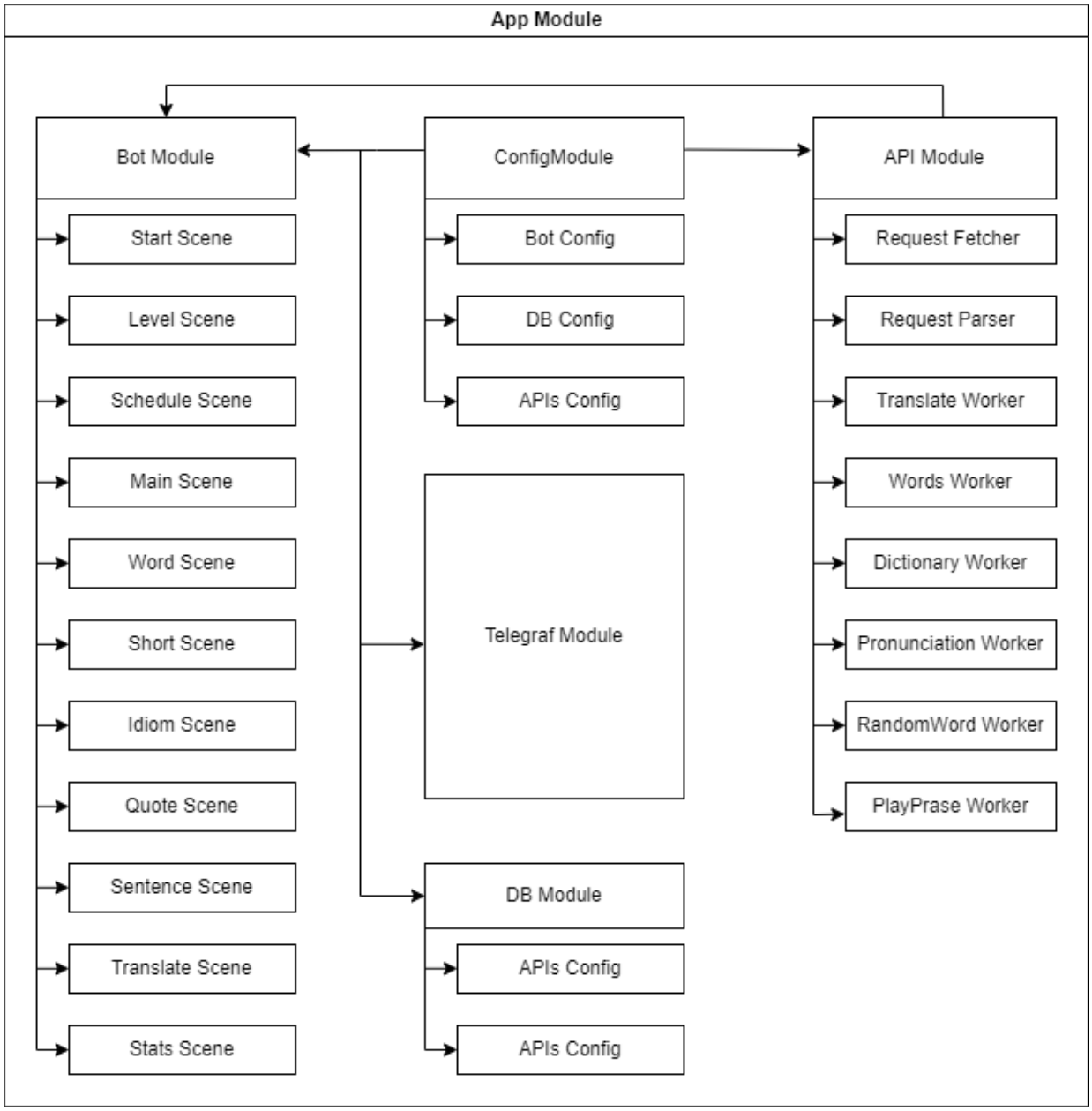

Рисунок Б.2 – Діаграма компонентів чат-боту

Продовження додатка Б

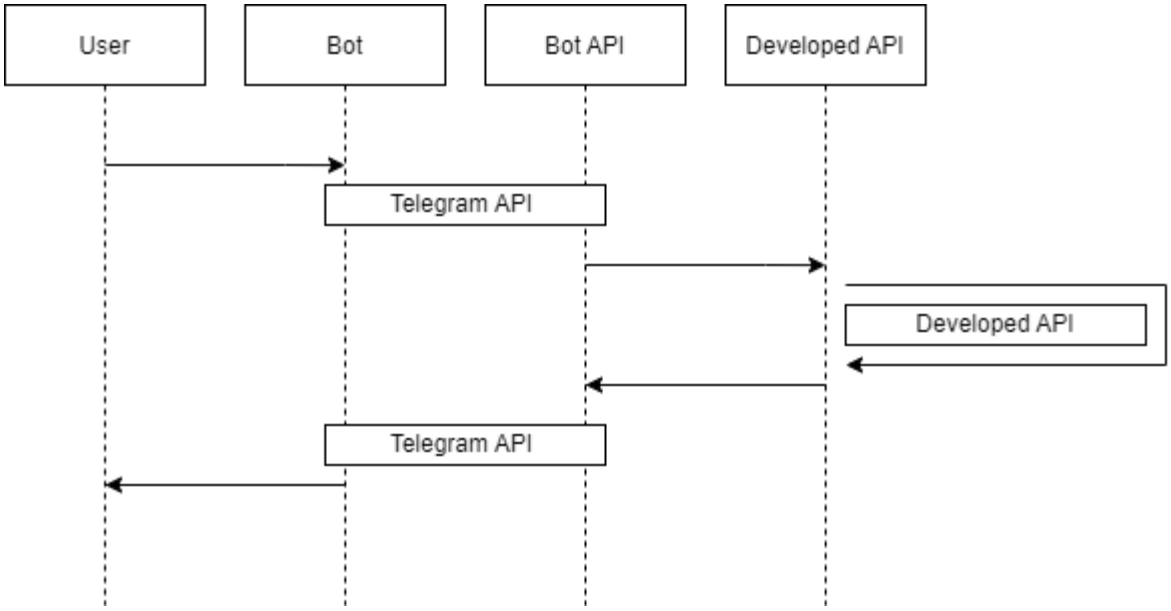

Рисунок Б.3 – Діаграма послідовності роботи чат-боту

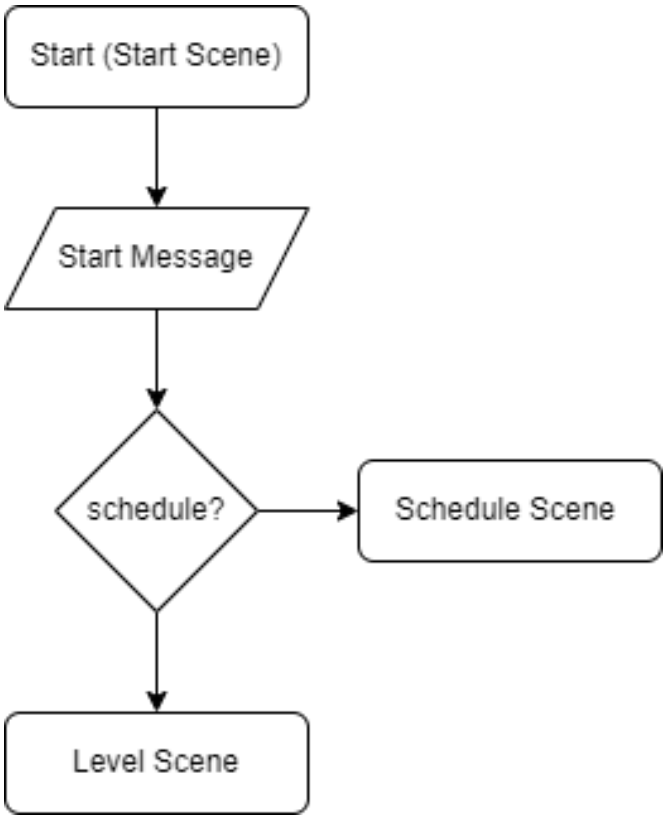

Рисунок Б.4 – UML-діаграма навігації Start Scene

Продовження додатка Б

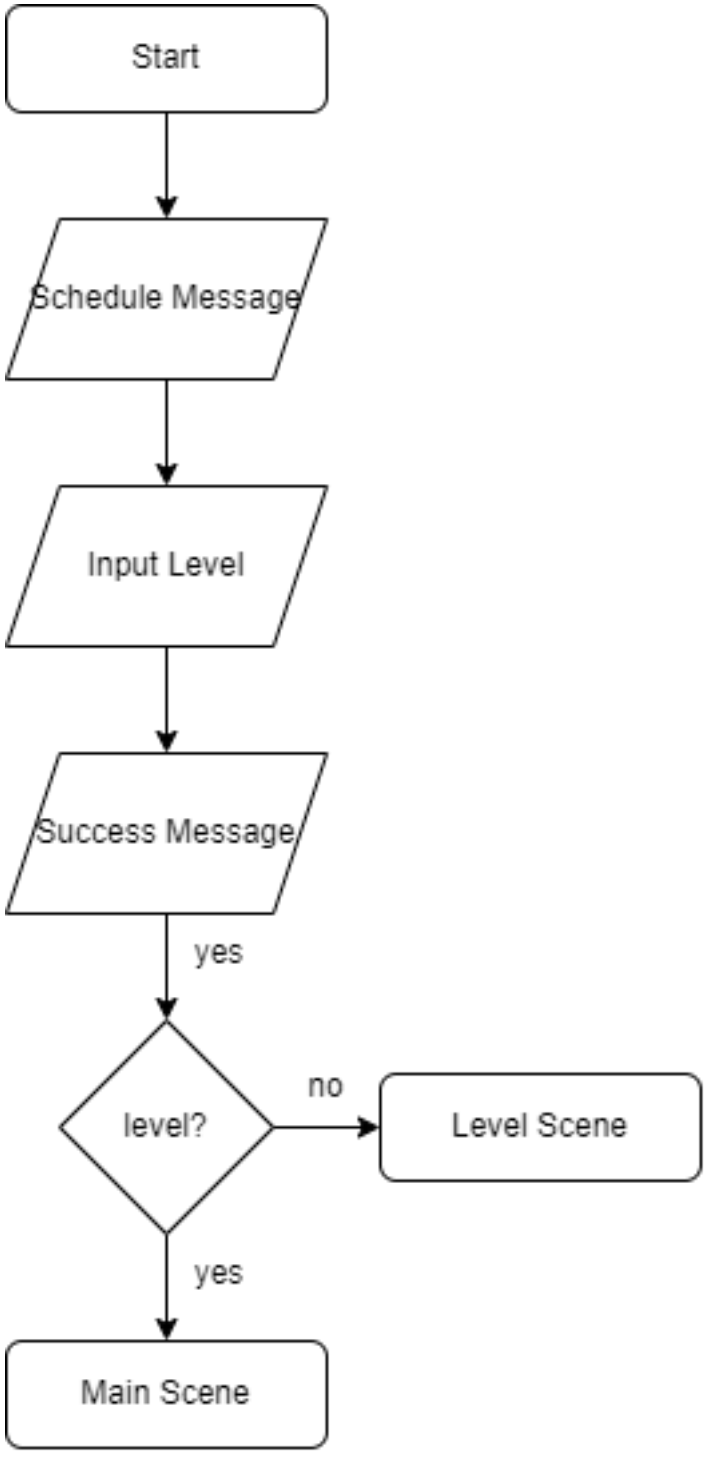

Рисунок Б.5 – UML-діаграма навігації Schedule Scene

Продовження додатка Б

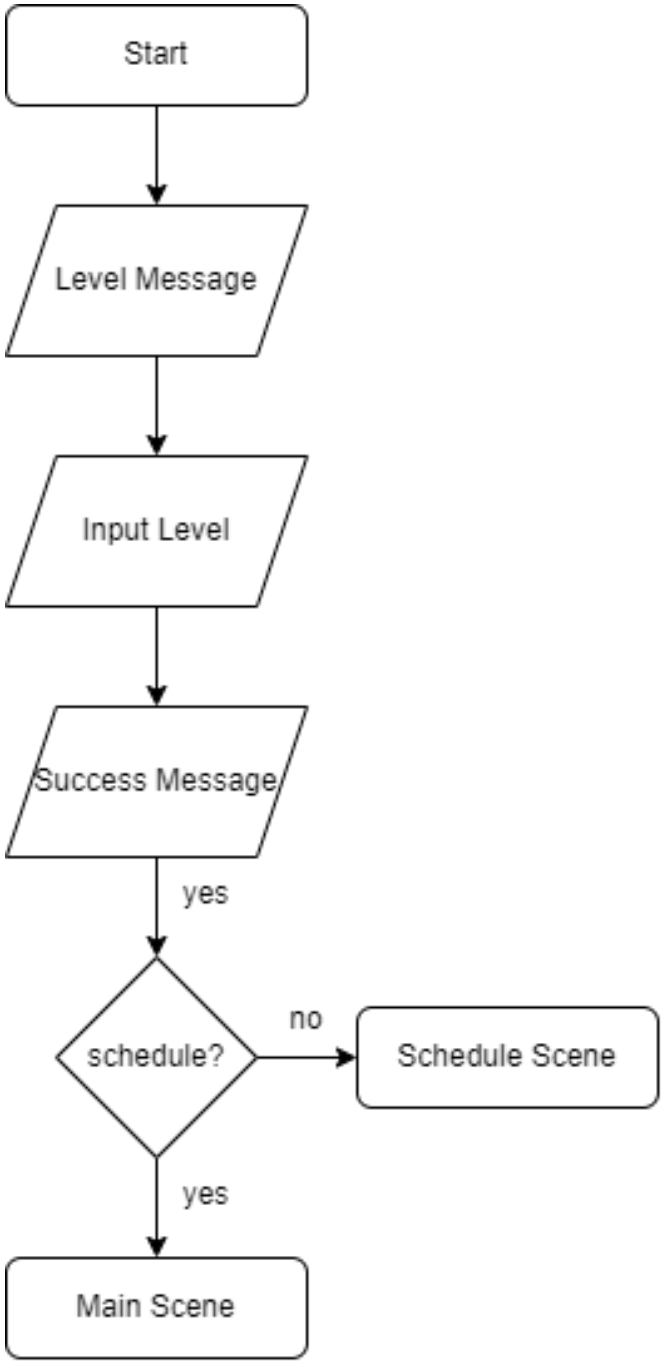

Рисунок Б.6 – UML-діаграма навігації Level Scene

Продовження додатка Б

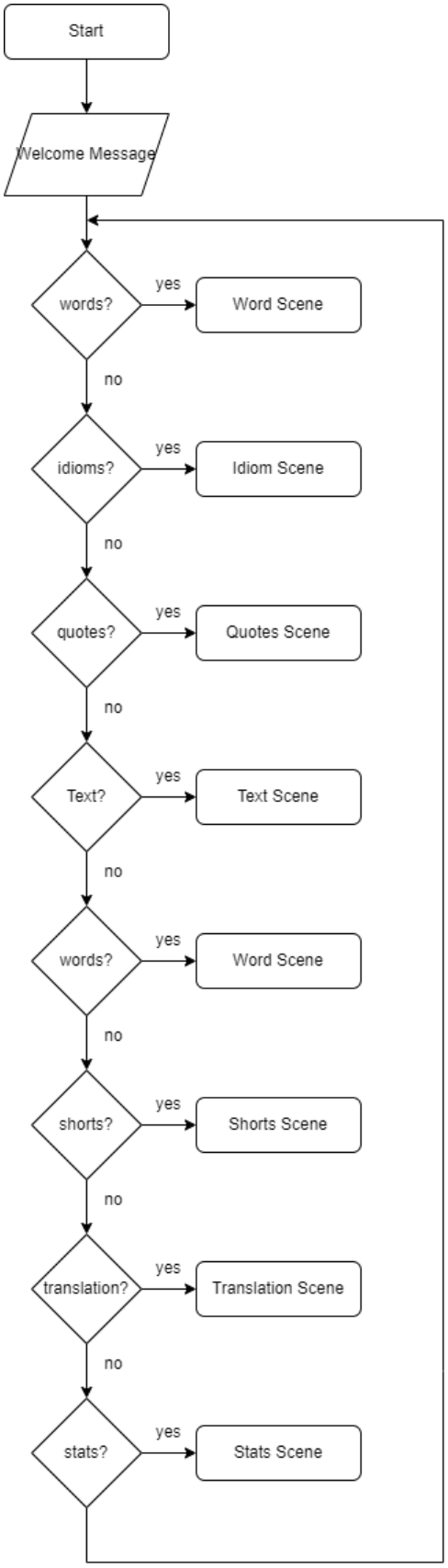

Рисунок Б.7 – UML-діаграма навігації Main Scene

Продовження додатка Б

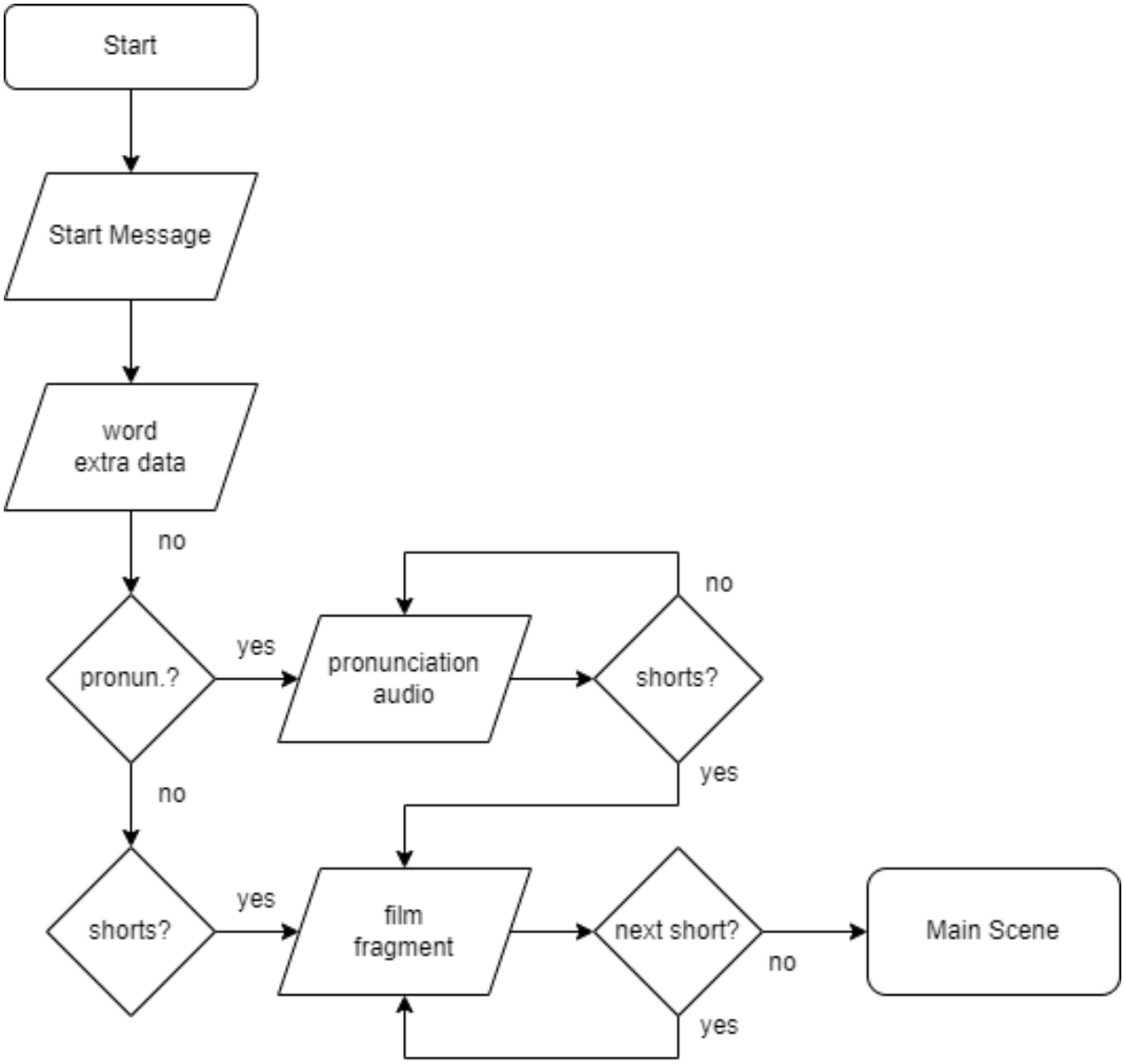

Рисунок Б.8 – UML-діаграма навігації Word Scene

Продовження додатка Б

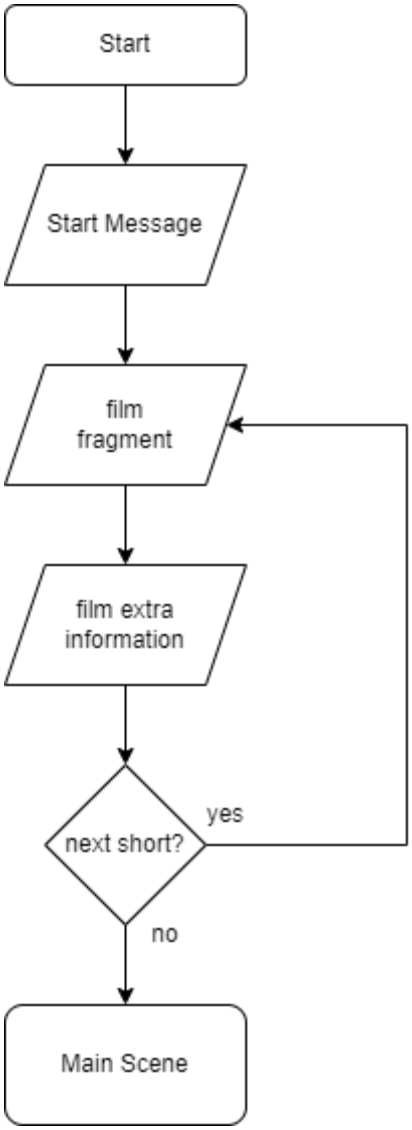

Рисунок Б.9 – UML-діаграма навігації Shorts Scene

Продовження додатка Б

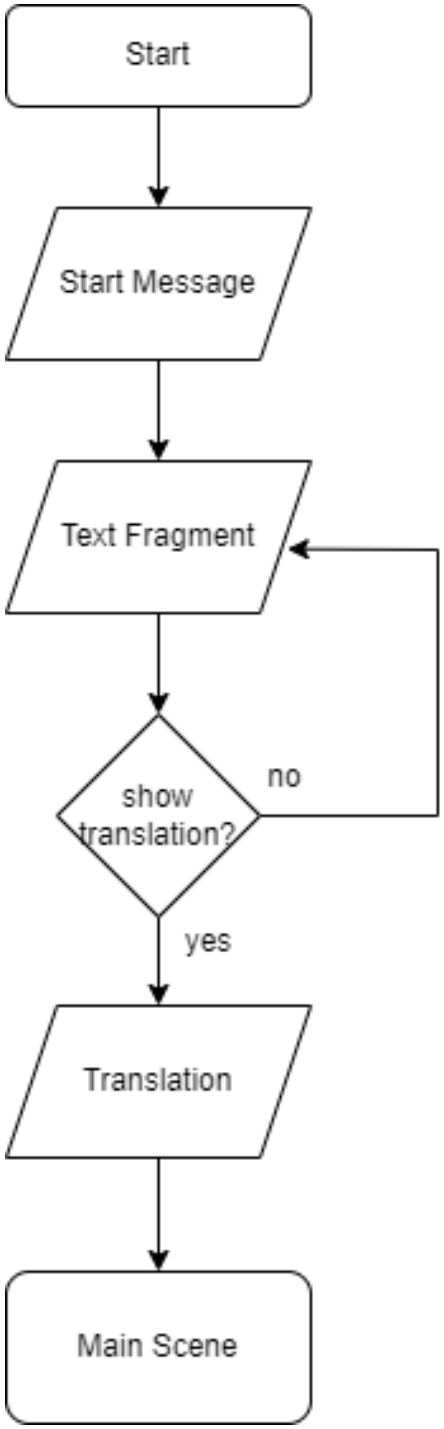

Рисунок Б.10 – UML-діаграма навігації Text Scene

Додаток В

Лістинг програмного забезпечення

```
async function bootstrap() {
  const app = await NestFactory.create(  AppModule);
  await app.listen(3030)
  app.useGlobalPipes(new ValidationPipe());
}
bootstrap();


@Module({
  imports: [
    ConfigModule.forRoot({
      isGlobal: true,
      load: [botConfiguration, dbConfiguration, apiConfiguration],
    }),
    TelegrafModule.forRootAsync({
      inject: [ConfigService],
      useFactory: (configService: ConfigService) => ({
        include: [BotModule],
        middlewares: [sessionMiddleware],
        token: configService.get('bot.botToken') as string,
      }),
    }),
    TypeOrmModule.forRootAsync({
      inject: [ConfigService],
      useFactory: (configService: ConfigService) => ({
        type: 'postgres',
        host: configService.get('db.host'),
        port: configService.get('db.port'),
        database: configService.get('db.name'),
        username: configService.get('db.user'),
        password: configService.get('db.password'),
```

```
        entities: [],
        synchronize: true,
        autoLoadEntities: true,
      }),
    }),
    UserModule,
    BotModule,
  ],
})
export class AppModule {}

export const sessionMiddleware = new LocalSession({
  property: 'session',
  database: 'sessions_db.json',
  storage: LocalSession.storageMemory,
}); import { Action, Ctx, InjectBot, Sender, Start, Update } from 'nestjs-telegraf';

@Update()
export class BotUpdate {
  constructor(
    @InjectBot()
    private readonly botService: BotService,
    private readonly usersService: UserService,
  ) {}

  @Start()
  async onStart(
    @Ctx() context: ISceneContext,
    @Sender('id') tgId: number,
    @Sender('first_name') firstName: string,
    @Sender('last_name') lastName: string,
  ): Promise<void> {
    const user = await this.usersService.findOne(tgId);
```

```
    if (!user) {
      await this.usersService.create({ tgId, firstName, lastName });
    }
    if (user?.level && user.schedule) {
      await context.scene.enter(SCENES.MAIN);
      return;
    }
    await context.scene.enter(SCENES.START);
  }
  @Action(COMMANDS.SET_LEVEL)
  async onSetLevel(@Ctx() context: ISceneContext) {
    await context.scene.enter(SCENES.SET_LEVEL);
  }
  @Action(COMMANDS.SET_SCHEDULE)
  async onSetSchedule(@Ctx() context: ISceneContext) {
    await context.scene.enter(SCENES.SET_SCHEDULE);
  }
}

export const COMMANDS = {
  START: 'START',
  BACK: 'BACK',
  MAIN: 'MAIN',
  PASS_TEST: 'PASS_TEST',
  SET_LEVEL: 'SET_LEVEL',
  SET_SCHEDULE: 'SET_SCHEDULE',
  MISS_SCHEDULE: 'MISS_SCHEDULE',
  MISS_LEVEL: 'MISS_LEVEL',
  LEARN_WORDS: 'LEARN_WORDS',
  LEARN_IDIOMS: 'LEARN_IDIOMS',
  LEARN_QUOTES: 'LEARN_QUOTES',
  LEARN_SHORTS: 'LEARN_SHORTS',
  LEARN_PARAGRAPHS: 'LEARN_PARAGRAPHS',
  LISTEN_PRONUNCIATION: 'LISTEN_PRONUNCIATION',
```

```
  NEXT_SHORT: 'NEXT_SHORT',
  WATCH_SHORT: 'WATCH_SHORT',
};

export const LINK = { levelTest: 'https://a-b-c.pl.ua/quiz/' };

export const TEXT = {
  getGreeting: (firstName: string) => `Привіт ${firstName}! 👋`,
  SCHEDULE_INTRO: `Для ефективного поповнення словникового запасу 💬 необхідно опрацьовувати приблизно вивчати по 10 слів на день 🧐.\n\nЯкщо ви будете робити це щодня, ви отримаєте 3650 слів за рік✅. Погодьтеся, це значна цифра📈.\n\nПри правильній методиці ці слова сформують активний словниковий 📚 запас, і ви зможете використовувати їх у будь-якій ситуації ⭐.`,
  getBotIntro: () =>
    `Я буду твоїм помічником 💂 у вивченні англійської мови!\n\nЗ моєю допомогою ти зможеш активно займатися кожен день 🗓️, тренеруючи свою лексику 💬, вимову 👄 та розуміння вимови носіїв англійської мови 👴.\n\nДля початку обери свій рівень англійської 🆙 та сконфігуруй кількість моїх повідомлень за день 📟.`,
  getLevelIntro: () =>
    `* Common European Framework of Reference (CEFR або CEF)
\nНайпопулярніша система оцінки знання англійської. Найчастіше саме з її допомогою роботодавці позначають необхідний рівень знання мови кандидатом.
\n📔 Складається з 6 рівнів:\n
✔️ Beginner, Elementary (A1).
✔️ Pre-Intermediate (A2).
✔️ Intermediate (B1).
✔️ Upper-Intermediate (B2).
✔️ Advanced (C1).
✔️ Proficiency (C2).
\nЇх об’єднують в систему, де є базовий, середній та просунутий рівні.`,
  READY_LEVEL: `Ти успішно встановив рівень знання мови 👍!\n\nВідповідно до нього тобі будуть підбиратися слова для подальшого вивчення мови 🏫.\n\nНе забувай час від часу проходити тест 📱, щоб розуміти свій рівень знань та сконфігурувати ⚙️ його в боті!\n\nДавай перейдемо до навчання! `,
  READY_SCHEDULE: `Ти успішно встановив розклад вивчення мови 👍!\n\nВідповідно до нього тобі будуть надходити повідомлення ✉️ для подальшого вивчення мови 🏫.\nНе забувай час від часу переглядати параметри конфігурації ⚙️, щоб вивчення нової лексики було більш ефективним 📈.\n\nДавай перейдемо до навчання!`,
```

```
  MISS_SCHEDULE: `Йой! 🔔\n\nРозклад повідомлень ще не сконфігурований, давай виправимо 🔧 це!\n\nНатискай кнопку нижче та заверши конфігурацію розкладу ✅`,

  MISS_LEVEL: `Йой! 🔔\n\nРівень твоєї англійської ще не сконфігурований, давай виправимо 🔧 це!\n\nНатискай кнопку нижче та заверши налаштування бота ✅`,

  PRESS_BTN: `Попереду багато цікавого. Тицяй кнопку та обирай чим хочеш зайнятися 🔆!`,

  START_EDUCATION: `Приступаємо до навчання🧐. Обери як ти хочеш навчатися 🧭`,

  HELP: `Встанови свій рівень ⭐ зання англійської мови 📚 та обери графік навчання.\n\nПісля ти можеш вивчати англійську за допомогою:\n\n📝 Текстових повідомлень\n📢 Аудіо повідомлень\n🎥 Відео-фрагментів`,

  renderWordMsg: (
    original: string,
    translation: string,
    pronunciation?: string,
  ) =>
    `🇬🇧 Слово: ${original}\n🇺🇦 Переклад: ${translation}\n📢 Вимова:${
      pronunciation || ''
    }`,

  renderMovieNameMsg: (name: string) => `🎬 Кадри із стрічки: ${name}`,

  EXIST_MORE_SHORTS: `😏 Доступні ще відео фрагменти, не бажаєш подивитися?`,
};

export const BUTTONS = {
  LEVEL: Markup.button.callback(
    '🚦 Обрати рівень англійської',
    COMMANDS.SET_LEVEL,
  ),
  SCHEDULE: Markup.button.callback(
    '🗓️ Обрати графік повідомлень бота',
    COMMANDS.SET_SCHEDULE,
  ),
  BACK: Markup.button.callback(
    '🔙 Повернутися до попереднього інтерфейсу',
    COMMANDS.BACK,
  ),
  MAIN: Markup.button.callback('🔍 Перейти до головного меню', COMMANDS.MAIN),
```

```
PASS_TEST: Markup.button.url(
  '💅 Скласти тест та дізнатись свій рівень англійської мови',
  LINK.levelTest,
),
[ELevels.A1]: Markup.button.callback(
  '👨‍ Beginner, Elementary (A1)',
  ELevels.A1,
),
[ELevels.A2]: Markup.button.callback('👨‍ Pre-Intermediate (A2)', ELevels.A2),
[ELevels.B1]: Markup.button.callback('👨‍ Intermediate (B1)', ELevels.B1),
[ELevels.B2]: Markup.button.callback(
  '👨‍ Upper-Intermediate (B2)',
  ELevels.B2,
),
[ELevels.C1]: Markup.button.callback('👨‍ Advanced (C1)', ELevels.C1),
[ELevels.C2]: Markup.button.callback('👨‍ Proficiency (C2)', ELevels.C2),
MISS_SCHEDULE: Markup.button.callback(
  'Перейти до конфігурації розкладу',
  COMMANDS.MISS_SCHEDULE,
),
MISS_LEVEL: Markup.button.callback(
  'Перейти до конфігурації рівня знань',
  COMMANDS.MISS_LEVEL,
),
renderScheduleBtn: (hours: ESchedule) =>
  Markup.button.callback(`Інтервал ${hours.slice(1)} годин(и)`, hours),

LISTEN_PRONUNCIATION: Markup.button.callback(
  '📢 Прослухати вимову слова',
  COMMANDS.LISTEN_PRONUNCIATION,
),
WATCH_SHORT: Markup.button.callback(
  '📽️ Дивитись відео-фрагмент',
```

```
    COMMANDS.WATCH_SHORT,
  ),
  NEXT_SHORT: Markup.button.callback(
    '➡️ Дивитись наступний відео-фрагмент',
    COMMANDS.NEXT_SHORT,
  ),
};
export const SCENES = {
  WORD: 'WORD',
  MAIN: 'MAIN',
  START: 'START',
  SET_LEVEL: 'SET_LEVEL',
  SET_SCHEDULE: 'SET_SCHEDULE',
};
interface IRenderWordViewProps {
  word: string;
  translation: string;
  pronunciation?: string;
  syllables?: string[];
  meanings?: {
    definition?: string;
    partOfSpeech?: string;
    synonyms?: string[];
    antonyms?: string[];
    examples?: string[];
  }[];
}
export const renderWordView = (wordItem: IRenderWordViewProps) => `
GB Оригінал - <b>${wordItem.word}</b>

UA Переклад - <b>${wordItem.translation}</b>

${wordItem.pronunciation ? `\n🔊 Вимова - [${wordItem.pronunciation}]` : ''}
```

```
${
  wordItem.syllables && wordItem.syllables.length
    ? `\n 🪚 Склади - [${wordItem.syllables.join(', ')}]`
    : ''
}



⚡ <b>Значення</b> ⚡
${
  wordItem.meanings
    ? wordItem.meanings
        .map(
          (meaning) => `
${meaning.definition ? `🔆 Значення (англ.) - ${meaning.definition}` : ''}
${
  meaning.partOfSpeech
    ? `\n🧩 Частина мови (англ.) - ${meaning.partOfSpeech}`
    : ''
}
${
  meaning.synonyms && meaning.synonyms.length
    ? `\n📗 Синоніми (англ.) - ${meaning.synonyms.join(', ')}`
    : ''
}
${
  meaning.antonyms && meaning.antonyms.length
    ? `\n📕 Антоніми (англ.) - ${meaning.antonyms.join(', ')}`
    : ''
}
${
  meaning.examples && meaning.examples.length
    ? `\n👁 Приклади вживання (англ.) - ${meaning.examples.join(', ')}`
    : ''
```

```
}
`,
        )
        .join('\n ⚡ <b>Ще одне значення</b> ⚡ \n')
    : ''
}
interface IWordSceneState {
  word: string;
  shortIndex: number;
}

@Scene(SCENES.WORD)
export class WordScene {
  constructor(
    private readonly userService: UserService,
    private readonly wordService: WordService,
  ) {}
  @SceneEnter()
  async onSceneEnter(@Ctx() context: TSceneContext<IWordSceneState>) {
    const word = await this.wordService.requestRandom(
      EAPIRandomTypes.VOCABULARY,
    );

    const translation = await this.wordService.requestTextTranslation2(word);

    const wordsRes = await this.wordService.requestWordsApi(word);

    const buttons = Markup.inlineKeyboard([
      [BUTTONS.LISTEN_PRONUNCIATION, BUTTONS.WATCH_SHORT],
    ]);

    await context.replyWithHTML(
      renderWordView({
        word,
```

```
      translation,
      meanings: wordsRes.results,
      syllables: wordsRes.syllables?.list,
      pronunciation: wordsRes.pronunciation?.all,
    }),
    buttons,
  );

  context.scene.session.state.word = word;
}

@Hears('impecunious')
async hear(@Ctx() context: TSceneContext<IWordSceneState>) {
  const word = 'impecunious';

  const translation = await this.wordService.requestTextTranslation2(word);

  const wordsRes = await this.wordService.requestWordsApi(word);

  const buttons = Markup.inlineKeyboard([
    [BUTTONS.LISTEN_PRONUNCIATION, BUTTONS.WATCH_SHORT],
  ]);

  await context.replyWithHTML(
    renderWordView({
      word,
      translation,
      meanings: wordsRes.results,
      syllables: wordsRes.syllables?.list,
      pronunciation: wordsRes.pronunciation?.all,
    }),
    buttons,
  );
  context.scene.session.state.word = word;
```

```
}

@Action(COMMANDS.LISTEN_PRONUNCIATION)
async onListenPronunciationAction(
  @Ctx() context: TSceneContext<IWordSceneState>,
) {
  const word = context.scene.session.state?.word;
  const pronunciation = this.wordService.requestTextPronunciation(word);

  const buttons = Markup.inlineKeyboard([[BUTTONS.WATCH_SHORT]]);

  await context.replyWithAudio(
    { url: pronunciation, filename: word },
    buttons,
  );
}

@Action(COMMANDS.WATCH_SHORT)
async onWatchShortAction(@Ctx() context: TSceneContext<IWordSceneState>) {
  const word = context.scene.session.state?.word;
  const short = await this.wordService.requestTextShort(word);

  const buttons = Markup.inlineKeyboard([[BUTTONS.NEXT_SHORT]]);

  const shortVideo = short[0]['video-url'];
  const shortInfo = short[0]['video-info']?.info;
  const nextShortVideo = short[1]?.['video-url'];

  await context.replyWithVideo(shortVideo);

  await context.reply(TEXT.renderMovieNameMsg(shortInfo));

  if (nextShortVideo) {
    await context.reply(TEXT.EXIST_MORE_SHORTS, buttons);
  }
```

```
    context.scene.session.state.shortIndex = 0;
  }

  @Action(COMMANDS.NEXT_SHORT)
  async onWatchNextShortAction(@Ctx() context: TSceneContext<IWordSceneState>) {
    const word = context.scene.session.state?.word;
    const short = await this.wordService.requestTextShort(word);

    const newShortIndex = context.scene.session.state.shortIndex + 1;
    context.scene.session.state.shortIndex = newShortIndex;

    const buttons = Markup.inlineKeyboard([[BUTTONS.NEXT_SHORT]]);

    const shortVideo = short[newShortIndex]['video-url'];
    const nextShortVideo = short[newShortIndex + 1]?.['video-url'];
    const shortInfo = short[newShortIndex]['video-info']?.info;

    await context.deleteMessage();

    await context.replyWithVideo(shortVideo);

    if (nextShortVideo) {
      await context.reply(TEXT.renderMovieNameMsg(shortInfo), buttons);
    } else {
      await context.reply(TEXT.renderMovieNameMsg(shortInfo));
    }
  }
};
@Scene(SCENES.START)
export class StartScene {
  @SceneEnter()
  async onSceneEnter(
    @Ctx() ctx: IContext,
```

```
    @Sender('first_name') firstName: string,
  ) {
    const buttons = Markup.inlineKeyboard([
      [BUTTONS.LEVEL],
      [BUTTONS.SCHEDULE],
    ]);

    await ctx.reply(TEXT.getGreeting(firstName));
    await ctx.reply(TEXT.getBotIntro(), buttons);
  }

  @Action(COMMANDS.SET_LEVEL)
  async onSetLevel(@Ctx() context: ISceneContext) {
    await context.scene.enter(SCENES.SET_LEVEL);
  }

  @Action(COMMANDS.SET_SCHEDULE)
  async onSetSchedule(@Ctx() context: ISceneContext) {
    await context.scene.enter(SCENES.SET_SCHEDULE);
  }
}
@Scene(SCENES.SET_SCHEDULE)
export class ScheduleScene {
  constructor(private readonly userService: UserService) {}
  @SceneEnter()
  async onSceneEnter(@Ctx() ctx: ISceneContext) {
    const buttons = Markup.inlineKeyboard([
      [
        BUTTONS.renderScheduleBtn(ESchedule.H2),
        BUTTONS.renderScheduleBtn(ESchedule.H4),
      ],
      [
        BUTTONS.renderScheduleBtn(ESchedule.H6),
        BUTTONS.renderScheduleBtn(ESchedule.H8),
```

```
    ],
    [
      BUTTONS.renderScheduleBtn(ESchedule.H10),
      BUTTONS.renderScheduleBtn(ESchedule.H12),
    ],
    [BUTTONS.renderScheduleBtn(ESchedule.H24)],
    [BUTTONS.BACK],
  ]);

  await ctx.reply(TEXT.SCHEDULE_INTRO, buttons);
}

@Action(Object.values(ESchedule))
async onSubmitLevelAction(
  @Ctx() ctx: ISceneContext,
  @Sender('id') tgId: number,
  @CallbackQuery('data') schedule: ESchedule,
) {
  const user = await this.userService.update({ tgId, schedule });
  const isUserLevelConfigured = user?.level;
  await ctx.reply(TEXT.READY_SCHEDULE);

  if (isUserLevelConfigured) {
    await ctx.reply(TEXT.PRESS_BTN, Markup.inlineKeyboard([BUTTONS.MAIN]));
    return;
  }

  await ctx.reply(
    TEXT.MISS_LEVEL,
    Markup.inlineKeyboard([BUTTONS.MISS_LEVEL]),
  );
}

@Action(COMMANDS.MISS_LEVEL)
```

```
  async onMyOrdersAction(@Ctx() ctx: ISceneContext) {
    await ctx.scene.enter(SCENES.SET_LEVEL);
  }

  @Action(COMMANDS.BACK)
  async onBackAction(@Ctx() context: ISceneContext) {
    await context.scene.enter(SCENES.START);
  }

  @Action(COMMANDS.MAIN)
  async onMainAction(@Ctx() context: ISceneContext) {
    await context.scene.enter(SCENES.MAIN);
  }
}
@Scene(SCENES.MAIN)
export class MainScene {
  @SceneEnter()
  async onSceneEnter(@Ctx() context: IContext) {
    const buttons = Markup.keyboard([
      [Markup.button.text('✍️ Слова'), Markup.button.text('🫰 Ідіоми')],
      [Markup.button.text('📜 Цитати'), Markup.button.text('📃 Текст')],
      [Markup.button.text('🎥 Відеофрагменти')],
      [Markup.button.text('🈂️ Переклад')],
    ]);
    await context.reply(TEXT.START_EDUCATION, buttons);
  }
  @Help()
  async onHelp(@Ctx() context: ISceneContext) {
    await context.reply(TEXT.HELP);
    await context.scene.leave();
  }
  @Hears('✍️ Слова')
  asynconTextWords(@Ctx() context: ISceneContext) {
```

```
    awai context.scene.enter(SCENES.WORD);
  }
@Scene(SCENES.SET_LEVEL)
export class LevelScene {
  constructor(private readonly userService: UserService) {}
  @SceneEnter()
  async onSceneEnter(@Ctx() ctx: ISceneContext) {
    const buttons = Markup.inlineKeyboard([
      [BUTTONS.PASS_TEST],
      [BUTTONS[ELevels.A1], BUTTONS[ELevels.A2]],
      [BUTTONS[ELevels.B1], BUTTONS[ELevels.B2]],
      [BUTTONS[ELevels.C1], BUTTONS[ELevels.C2]],
      [BUTTONS.BACK],
    ]);
    await ctx.reply(TEXT.getLevelIntro(), buttons);
  }
  @Action(Object.values(ELevels))
  async onSubmitLevelAction(
    @Ctx() ctx: ISceneContext,
    @Sender('id') tgId: number,
    @CallbackQuery('data') level: ELevels,
  ) {
    const user = await this.userService.update({ tgId, level });
    const isUserScheduleConfigured = user?.schedule;
    await ctx.reply(TEXT.READY_LEVEL);
    if (isUserScheduleConfigured) {
      await ctx.reply(TEXT.PRESS_BTN, Markup.inlineKeyboard([BUTTONS.MAIN]));
      return;
    }
    await ctx.reply(
      TEXT.MISS_SCHEDULE,
      Markup.inlineKeyboard([BUTTONS.MISS_SCHEDULE]),
    );
  }
```

```
  @Action(COMMANDS.MISS_SCHEDULE)
  async onMyOrdersAction(@Ctx() ctx: ISceneContext) {
    await ctx.scene.enter(SCENES.SET_SCHEDULE);
    return;
  }

  @Action(COMMANDS.BACK)
  async onBackAction(@Ctx() context: ISceneContext) {
    await context.scene.enter(SCENES.START);
  @Action(COMMANDS.MAIN)
  async onMainAction(@Ctx() context: ISceneContext) {
    await context.scene.enter(SCENES.MAIN);
  }
}
```

Додаток Г (обов'язковий)

ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи: *Комплексна магістерська кваліфікаційна робота «Розробка багатофункціонального телеграм боту для вивчення англійської мови з використанням сторонніх API».*

Тип роботи: магістерська кваліфікаційна робота
(кваліфікаційна робота, курсовий проект (робота), реферат, аналітичний огляд, інше – зазначити)

Підрозділ: кафедра АІІТ, ФІІТА, 1ІСТ-21м
(кафедра, факультет, навчальна група)

Науковий керівник: к.т.н., доцент кафедри АІІТ Богач І. В
(прізвище, ініціали, посада)

Показники звіту подібності

Plagiat.pl (StrikePlagiarism)		*Unicheck*	
КП1	-	Оригінальність	96.1%
КП2	-		
Тривога/Білі знаки	/	Схожість	3.9%

Аналіз звіту подібності (відмітити потрібне):
X Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

Заявляю, що ознайомлений (-на) з повним звітом подібності, який був згенерований Системою щодо роботи (додається)

Автор: _____ Семенюк А.В.
(підпис) (прізвище, ініціали)

Опис прийнятого рішення: Допустити до захисту

Особа, відповідальна за перевірку: _____ Маслій Р.В.
(підпис) (прізвище, ініціали)