Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

магістра
(назва освітньо-кваліфікаційного рівня)

студента	*Скрипки Микити Євгеновича*
	(ПІБ)
академічної групи	*122М-20-2*
	(шифр)
спеціальності	*122 Комп'ютерні науки*
	(код і назва спеціальності)
освітньої програми	*«122 Комп'ютерні науки»*
	(назва освітньої програми)
на тему:	*Розробка програмного забезпечення з функціональністю інтерактивного чат-бота з використанням алгоритмів машинного навчання*

_____*М.Є. Скрипка*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
розділів кваліфікаційної роботи	*Проф. Мещеряков Л.І.*			
спеціальний	*Проф. Мещеряков Л.І.*			
економічний	*Проф. Вагонова О.Г.*			
Рецензент				
Нормоконтролер	*Ас. Харь А.Т.*			

Дніпро
2022

Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
Завідувач кафедри
Програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 20 22 Року

ЗАВДАННЯ
на виконання кваліфікаційної роботи

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

студенту *122м-20-2* *Скрипки Микити Євгеновича*
(група) (прізвище та ініціали)

Тема кваліфікаційної роботи *Розробка програмного забезпечення з функціональністю інтерактивного чат-бота з використанням алгоритмів машинного навчання*

1 ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Наказ ректора НТУ «Дніпровська політехніка» від 10.12.2021 р. № 1036 -с

2 МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об'єкт досліджень – зразки текстів та зображень, які використовуються для написання повідомлень, статей та книг.

Предмет досліджень – засоби та моделі обробки природної мови, методи навчання моделі, методи навчання та роботи моделі із даними на основі зображень.

Мета НДР – створення програми чат-бота з використанням алгоритму генерування тексту з обробкою природної мови та користувацького введення.

Вихідні дані для проведення роботи – збірники мовних даних, складені з прикладів людських діалогів, текстів, а також метадані, релевантні по введених користувачами зображень, моделі, створені за алгоритмом обробки природної мови.

3 ОЧІКУВАНІ НАУКОВІ РЕЗУЛЬТАТИ

Новизна запропонованих рішень полягає в новому алгоритмі генеруванні тексту на природної мові, для використання при написанні текстових повідомлень, статей, книг, тощо, заголовків для тексту, а також допомога при перекладі тексту на основі машинного навчання, який також може засовуватися для генерування зображень за заготовками. Існуючі програми-аналоги майже не підтримують

можливість миттєвого навчання на введених даних, а також не мають мультимедійної функціональності.

Практична цінність полягає у розробці програм та алгоритму генерації тексту та зображень, які можуть бути використані при написанні текстових повідомлень, статей, текстових перекладів.

4 ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Результати досліджень мають бути подані у вигляді, що дозволяє побачити та оцінити використання розумного алгоритму обробки природної мови. В результаті роботи повинен бути розроблений програмний комплекс для вирішення задачі спілкування із віртуальним супутником у мережі Інтернет.

5 ЕТАПИ ВИКОНАННЯ РОБІТ

Найменування етапів робіт	**Строки виконання робіт (початок – кінець**
Аналіз теми та постановка задачі	12.09.2021-30.09.2021
Побудова моделі та алгоритмів навчання на даних, датасету	01.10.2021-31.10.2021
Створення додаткової функціональності обробки зображень	01.11.2021-16.12.2021

6 РЕАЛІЗАЦІЯ РЕЗУЛЬТАТІВ ТА ЕФЕКТИВНІСТЬ

Економічний ефект від реалізації результатів роботи очікується позитивним завдяки впровадженню моделі монетизацію сервісу, в рамках якої користувач може придбати додаткові функції програми.

Соціальний ефект від реалізації результатів роботи очікується позитивним завдяки впровадженню нового переважно розважального чат-боту на зростаючому ринку віртуальної комунікації.

7 ДОДАТКОВІ ВИМОГИ

Завдання видав _____ *Мещеряков Л.І.*
(підпис) (прізвище, ініціали)

Завдання прийняв до виконання _____ *Скрипка М.Є.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *12.09.2021 р.*

Термін подання кваліфікаційної роботи до ЕК *19.01.2022*

РЕФЕРАТ

Пояснювальна записка: 91 с., 35 рис., 3 дод., 57 джерел.

Об'єкт розробки: розробка програмного забезпечення з функціоналом інтерактивного чат-бота з використанням алгоритмів машинного навчання.

Мета кваліфікаційної роботи: створення програми чат-бота з обробкою користувацького введення та алгоритмом машинного навчання на основі введення, включаючи обробку тексту та зображень.

У вступі було розглянуто сучасний стан проблеми, уточнено мету кваліфікаційної роботи та область її застосування, обґрунтовано актуальність теми, уточнено постановку проблеми.

У першому розділі проаналізовано предметну область, визначено актуальність завдання, сформульовано постановку проблеми, уточнено вимоги до реалізації програми, технології та програмні засоби.

У другому розділі розглянуто існуючі рішення, обрано платформу для розробки, описано роботу по розробці, названо основні алгоритми та структуру, надано довідкову інформацію та функціональні можливості програми, зроблено опис інтерфейсу користувача системи.

В економічному розділі було визначено трудомісткість та тривалість розробки програмного продукту, виконано оцінку витрат на його створення.

Практична цінність проекту полягає у створенні програми чат-бота, яка надає можливість розмовляти за допомогою введення користувача, а також можливість обробляти, вчитися та виводити дані на основі зображень.

Актуальність цієї програми визначається достатнім попитом на подібні програми, які дозволяють користувачам у всьому світі з доступом до Інтернету спілкуватися з віртуальним супутником. Більше того, існуючі програми майже не підтримують можливість миттєвого навчання на введених даних, що лише підвищує актуальність розробки програми в цій області.

Список ключових слів: ПРОГРАМА, ЧАТ-БОТ, МАШИННЕ НАВЧАННЯ, ОБРОБКА ЗОБРАЖЕНЬ, .NET CORE, VISUAL STUDIO.

ABSTRACT

Explanatory note: 91 p., 35 figs., 3 app., 57 sources.

The object of development: development of software with interactive chatbot functionality using machine learning algorithms.

The purpose of the qualification thesis: creation of a chatbot application with user input processing and input-based machine learning algorithm including text and image processing.

In introduction the current state of the problem was examined, the purpose of the qualification thesis and the field of its application were specified, the relevance of the topic was justified, and the problem statement was specified.

In the first section subject area was analyzed, the relevance of the task was determined, problem statement was formulated, requirements for program implementation, technologies and software tools were specified.

In the second section existing solutions were overviewed, platform for development was chosen, development work was described, main algorithms and structure were mentioned, the background information and application functionality were provided, a description of the system user interface was made.

In economic section labor input and duration of software product development was defined, the estimation of expenses for its creation was executed.

The practical value of the project is to create a chatbot application that provides the ability to create human-like conversations using user input, as well as the ability to process, learn from and output image-based data.

The relevance of this application is determined by the sufficient demand for similar applications that allow users all around the world with Internet access to speak to a virtual companion. Moreover, existing applications almost do not support the ability to instantly learn from the input, which only increases the relevance of the development of the application in this area.

Keywords: PROGRAM, CHATBOT, MACHINE LEARNING, IMAGE PROCESSING, .NET CORE, VISUAL STUDIO.

ЗМІСТ

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ

OS — Operating System;

IDE — Integrated Development Environment;

IS — Information System;

OOP — Object-Oriented Programming;

IT — Information Technologies;

SDK — Software Development Kit;

XML — Extensible Markup Language;

GUI — Graphical User Interface;

JSON — JavaScript Object Notation;

IO — Input Output;

API — Application Programming Interface;

PC — Personal Computer;

RAM — Random Access Memory;

CLR — Common Language Runtime;

AI — Artificial Intelligence;

ML — Machine Learning;

IL — Intermediate Language;

JIT — Just-In-Time;

FCL — Framework Class Library;

LINQ — Language Integrated Query;

NLP — Natural Language Processing;

ВСТУП

В наш час мільйони людей у всьому світі майже щодня використовують комп'ютери, ноутбуки та смартфони, вони використовують їх для спілкування з друзями та незнайомими людьми в іграх та програмах чату. Таким чином, є необхідність вдосконалювати та урізноманітнити існуючі засоби комунікації.

Безсумнівно, Інтернет-комунікація – це галузь, що стрімко розвивається, яка охоплює всі сфери життя людини та надає широкий спектр послуг. Це допомагає користувачам заощадити їх час, виконувати багато важливих функцій і чітко повідомляти про свої наміри. Це призводить до високої актуальності застосування чат-бота для збагачення процесу спілкування.

Ідея чат-бота з користувацьким навчанням не зовсім нова ідея для ринку. В даний час найбільш продуктивним і підходящим способом спілкування між людьми є програми миттєвого обміну повідомленнями. Більшість програм цього типу дозволяють створювати ботів з API, наданих месенджером.

У проекті таке програмне забезпечення буде проаналізовано та розглянуто з його перевагами та недоліками. Завдання цього проекту та його об'єкт діяльності безпосередньо пов'язані з напрямком підготовки «Комп'ютерні науки» і відповідають узагальненим темам кваліфікаційної роботи, типових завдань діяльності, навичок та компетенцій, які повинні проводитися магістрам відповідно до освітнього кваліфікації. Завершення кваліфікаційної роботи дає можливість автору отримати кваліфікацію спеціаліста з розробки та тестування програмного забезпечення.

Мета даної кваліфікаційної роботи – відпрацювати інструменти Visual Studio, принципи мови C#, працювати з її процесами та методами, застосовуючи отримані знання при створенні програми Bores.

Результатом цього проекту є консольний додаток, який надає користувачеві унікальний інтерактивний текстовий і мультимедійний чат-бот.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ПРОБЛЕМИ

1.1. Загальна інформація про предметну область

Світ технологій динамічно і безперервно розвивається. З кожним роком з'являються нові концепції та їх реалізації, а вже існуючі розширюють свої функціональні характеристики та можливості. Головна мета цієї спіралі прогресу – зробити умови життя людини максимально комфортними та продуктивними.

Чат-бот – це в першу чергу програма для текстового спілкування. Це незамінне доповнення для нашого комфорту, яке щодня використовується не лише для особистого користування, а й для ділових завдань.

Сучасний чат-бот – це зазвичай веб-додаток, який використовує обробку натуральної мови для участі в людських розмовах, доступних у месенджерах, на веб-сайтах або в спеціальних мобільних додатках. Вони створені для розпізнавання людських намірів шляхом обробки текстових повідомлень, наданих користувачем, і відповідають потребам користувачів практично без участі людини-оператора (рис. 1.1.).

Сучасні інновації чат-ботів (рис. 1.2.) пройшли довгий шлях, і зараз компанії використовують чат-ботів безліччю способів. Головною перевагою чат-бота є те, що він не затримується, оскільки бот запрограмований і швидко відповість на будь-яке повідомлення користувача. Звичайним випадком використання є вже існуючий клієнт, де чат-бот може легко та швидко відповісти на будь-яке запитання користувача. Основною перевагою цього підходу є економія часу, оскільки бот не змушує свого клієнта затримувати доступ до спілкування. Замість того, щоб клієнт проходив сегмент допомоги або розділ поширених запитань на вашому сайті, чат-бот виконує свою роботу та відповідає клієнту на основі наданих знань. Крім того, бот може збирати дані для компаній, запитуючи запитання клієнтів і використовуючи дані з чатів.

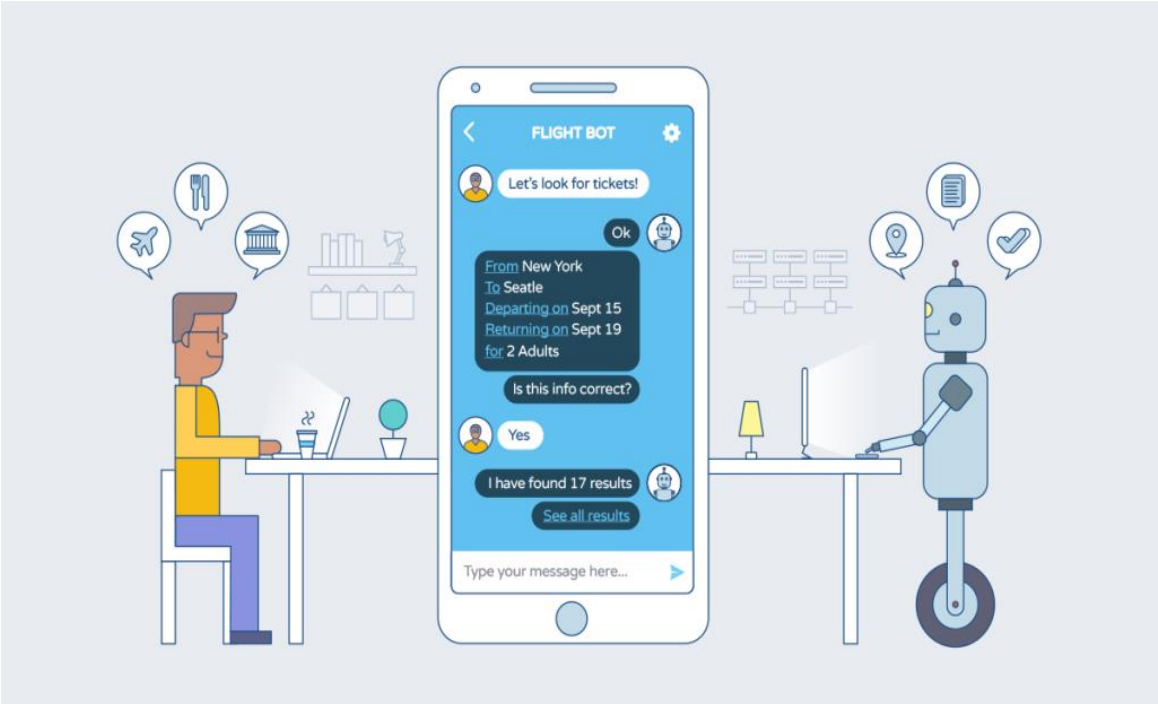

Рис. 1.1. Приклад комерційного чат-бота з продажу білетів

Рис. 1.2. Глобальний ринок чат-ботів

1.1.1. Огляд та аналіз існуючих аналогів

Перш ніж починати розробку того чи іншого програмного додатка чи системи, краще поглянути на аналоги, дослідити, проаналізувати їх переваги та недоліки, щоб зрозуміти, яких помилок слід уникати і які функції реалізувати. Були проаналізовані та вибрані декілька найкращих чат-ботів з тих, які вже існують і досить популярні в мережі. Вибрані були Cleverbot (рис. 1.3.), SimSimi (рис. 1.5.), щоб розглянути переваги та недоліки кожного з них. Ці програми мають значну базу користувачів і багато оглядів. Важливо враховувати індивідуальні особливості обох програм, їх структуру, стабільність, користувальницький досвід, надання послуг і приймати правильні рішення щодо розвитку проекту. У наступному розділі (2.3.4) буде розглянуто питання створення та застосування специфічних для функцій методів розробки програмного забезпечення для оптимізації циклу розробки.

Cleverbot [1]

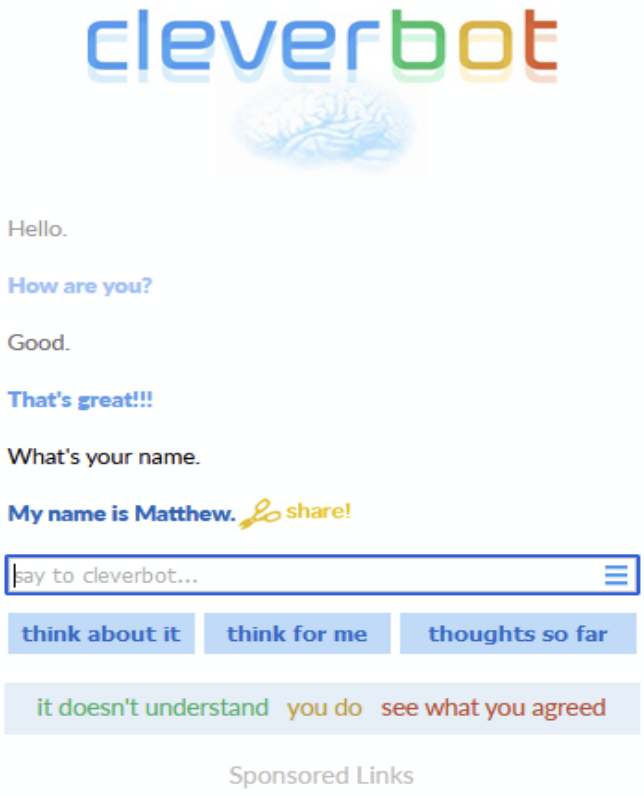

Рис. 1.3. Вікно чату Cleverbot

Cleverbot — це розважальний чат-бот із штучним інтелектом, який почав працювати в 2006 році. Однак його штучний інтелект був створений у 1988 році і, згідно з описом, з тих пір навчається. Веб-сайт простий і в основному складається з вікна чату на декількох кнопок.

Веб-сайт чат-бота можна безкоштовно використовувати без реєстрації користувача з моделлю монетизації веб-сайту на основі реклами. Однак реєстрація потрібна для використання додаткових функцій Cleverbot, таких як налаштування відповідей бота, збереження історії розмов, публікація фрагментів розмови, взаємодія з іншими користувачами тощо.

Cleverbot не пропонує мультимедійних функцій і підтримує різні мови введення. Неможливо зберегти розмови за допомогою вбудованих функцій. Cleverbot доступний в App Store за 0,99$ (рис. 1.4.).

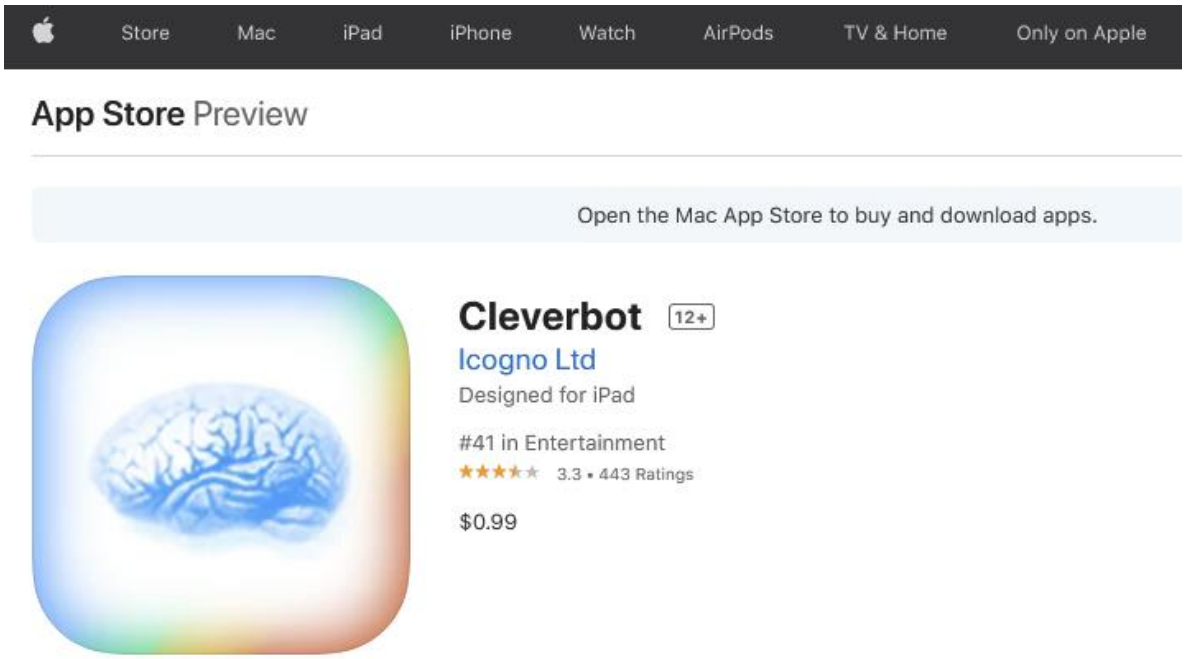

Рис. 1.4. Сторінка Cleverbot в App Store

Додаток iOS має платні функції, такі як розпізнавання голосу. Додаток має рейтинг 3,3 з 5 і змішані відгуки. Багато користувачів скаржаться на бракування

розмовних навичок і неналежну роботу розпізнавання голосу. Після проведення аналізу асортименту відгуків користувачів можна зробити наступний висновок.

Переваги:

- велика база знань;
- унікальні відповіді;
- підтримка різних мов введення;
- платний додатковий функціонал.

Недоліки:

- немає офіційного додатка для Android;
- бракування розмовних навичок;
- відсутність мультимедійної функціональності.

SimSimi [2]

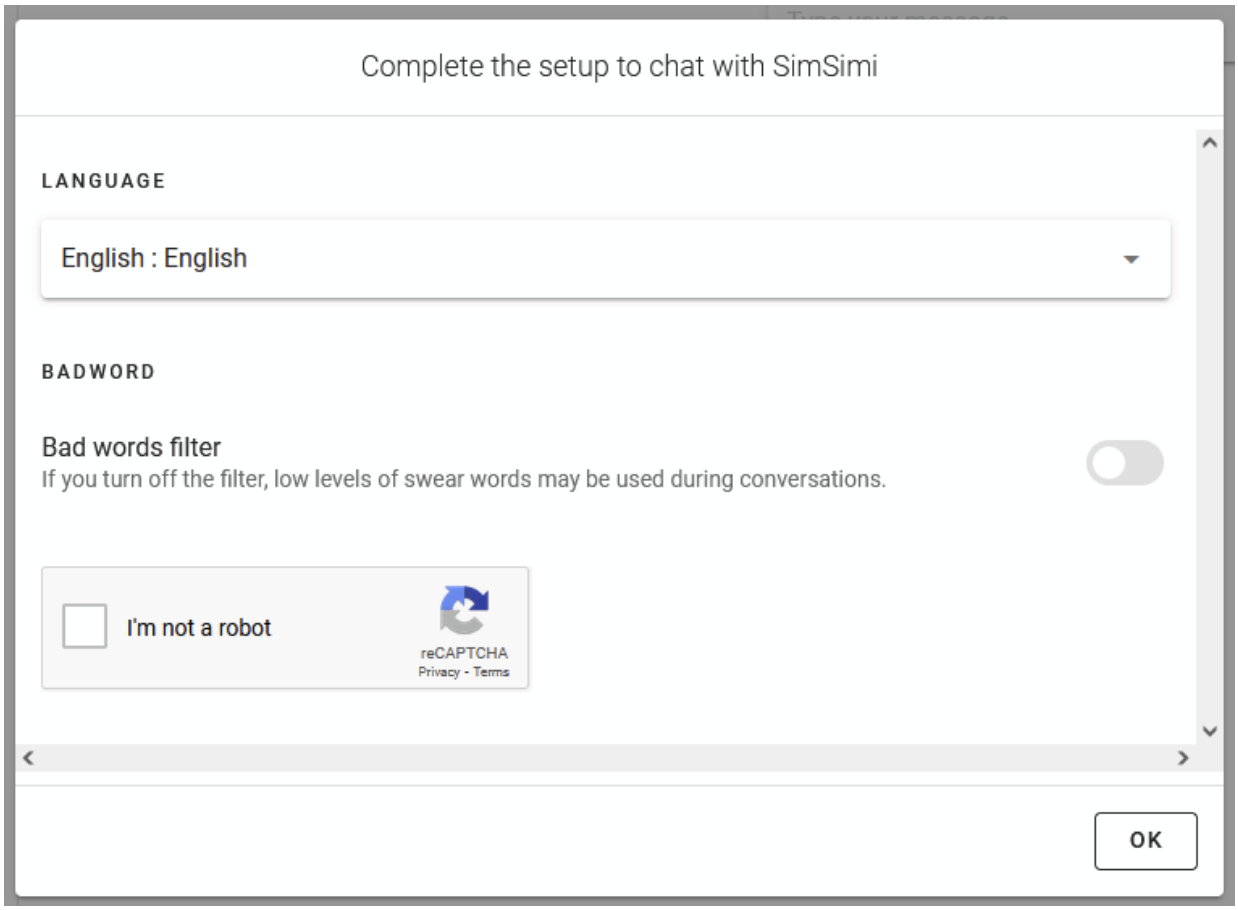

Рис. 1.5. Початкове вікно діалогу SimSimi

SimSimi — це розважальний чат-бот з глибоким навчанням, який почав свою діяльність у 2002 році. З тих пір він здійснив понад 130 мільйонів розмов із понад 350 мільйонами користувачів у всьому світі.

Веб-сайт чат-бота можна безкоштовно використовувати для перших 5 повідомлень без вимоги реєстрації користувача з моделлю монетизації сервісу на основі реклами та плати за використання. Реєстрація необхідна для оплати додаткових «повітряних куль» на SimSimi.

SimSimi не пропонує мультимедійних функцій і підтримує 81 мову введення. Неможливо зберегти розмови за допомогою вбудованих функцій. SimSimi доступний в App Store і Google Play (рис. 1.6.).

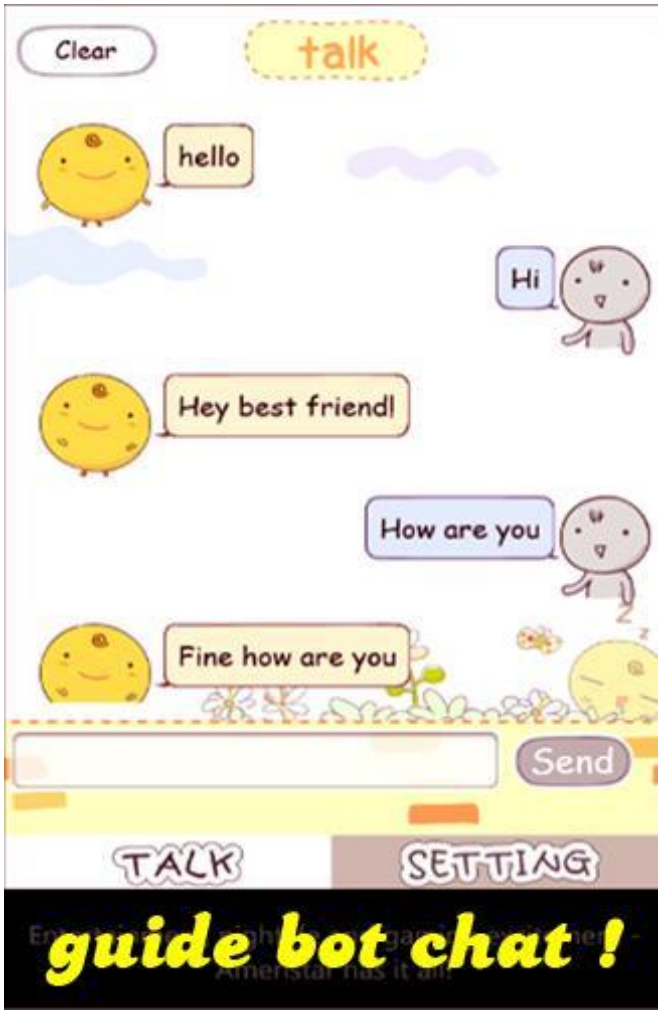

Рис. 1.6. Додаток для Android

Додаток пропонує покупки в програмі та містить рекламу. 100 повідомлень продаються за 1,99$, 300 за 4,99$ і 500 за 6,99$ (рис. 1.7.).

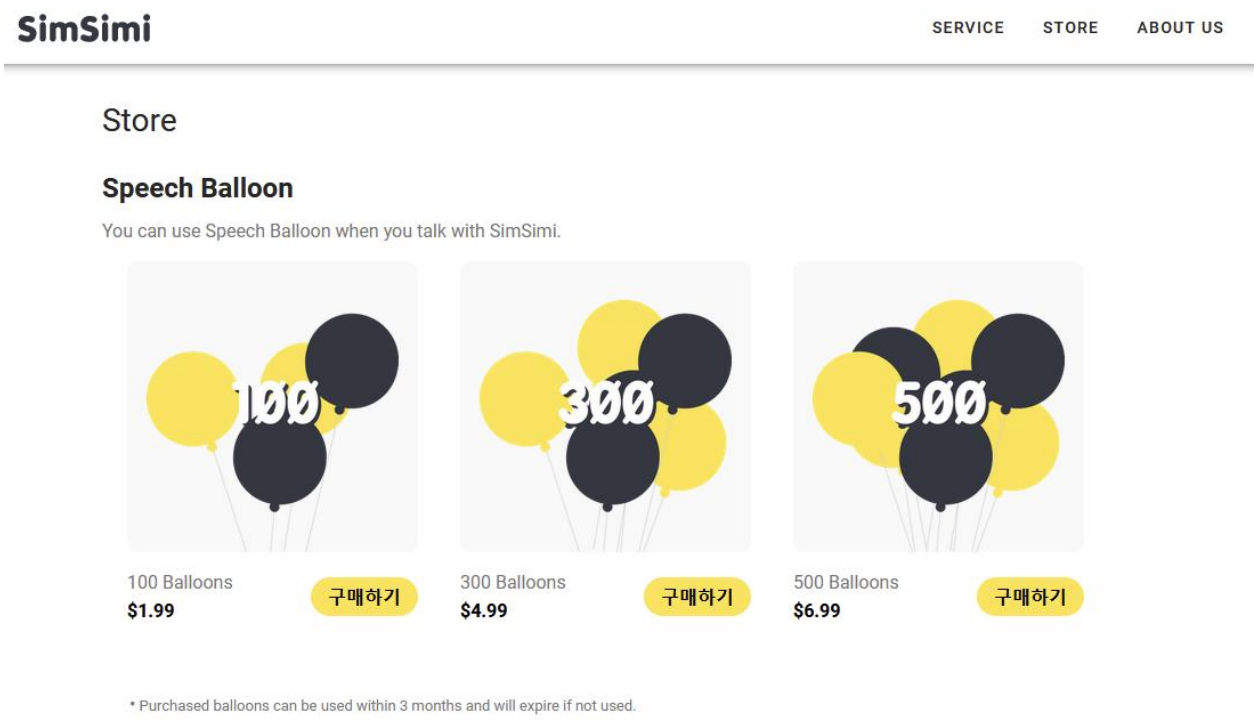

Рис. 1.7. Ціни на SimSimi

Переваги:

- великі знання та база користувачів;
- доступний на багатьох платформах.

Недоліки:

- безкоштовна демо-версія майже повністю відсутня;
- агресивна монетизація на основі реклами.

1.1.2. Вибір операційної системи

Практично кожен власник смартфона використовує його після ранкового пробудження і безпосередньо перед сном. Про те, скільки часу звичайна людина проводить за своїм смартфоном, говорити не доводиться. Відомо, що близько 40% інтернет-трафіку використовується зі смартфонів. І це не дивно, адже

смартфон – це універсальний мобільний пристрій, який завжди є вдома чи в дорозі і дозволяє користувачеві легко отримати доступ до необхідної інформації всього за кілька кліків у додатку чи браузері.

Смартфони настільки вкоренилися в нашому житті, що їм важко не користуватися. Можна сказати, що смартфони є найкориснішими пристроями цифрового зв'язку у цифрову епоху.

Отже, розробка програми для смартфонів як платформи є правильним рішенням. Тому необхідно розглянути існуючі операційні системи (рис. 1.8.) для мобільних комп'ютерів і вибрати найбільш необхідну для створення програми для певної мобільної операційної системи.

Важливо проаналізувати різні особливості розробки системи, частку ринку, доступні методи розповсюдження програмного забезпечення, простоту та практичність розробки, базу користувачів.

Висновок дозволить належним чином сконцентрувати ресурси розробки та згодом вибрати правильну модель розробки програмного забезпечення.

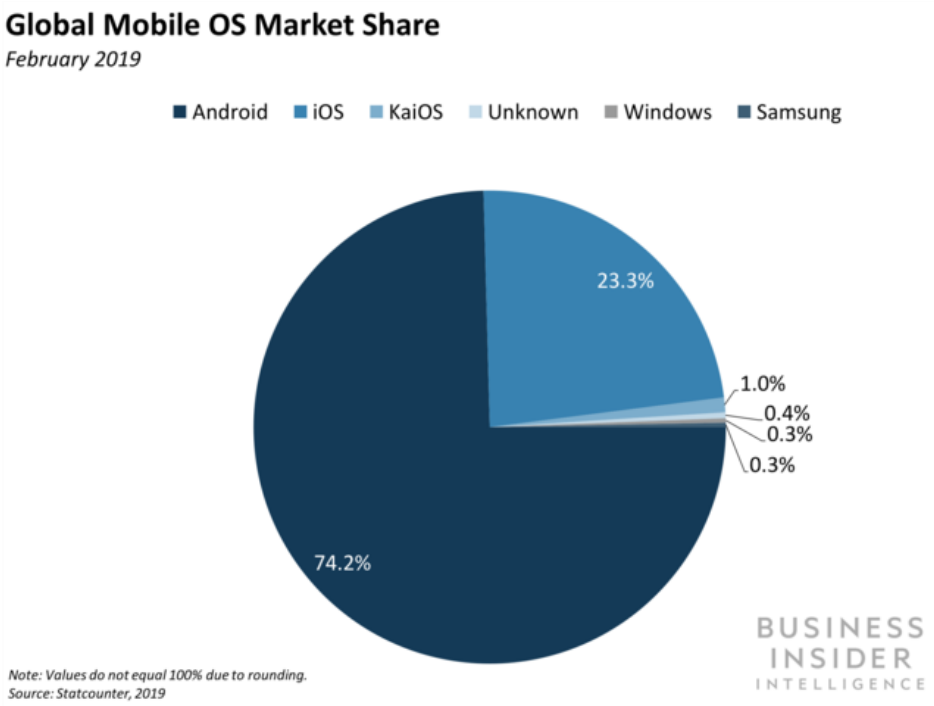

Рис. 1.8. Схема використання ОС на ринку смартфонів

Завдання вибору операційної системи досить важливе, оскільки ринок ОС для смартфонів досить широкий і кожна система має свої переваги. Існує велика кількість операційних систем для смартфонів, але всі вони мають різну популярність. Чим більше людей зможуть використовувати додаток, тим більша ймовірність знайти аудиторію, яка зацікавиться програмою.

Три найбільш часто використовувані операційні системи, які слід розглянути, – це Android, iOS, Windows Phone. Усі три ОС мають власний візуальний стиль, ефекти, зовнішній вигляд, дисплей, програми, доступність, панель навігації, інтерфейс та багато інших.

Кожна з перерахованих вище характеристик створює інтерфейс та уявлення про систему, складність та підходи щодо наявності, ціни та можливостей додатків під час користування смартфоном чи мобільним комп'ютером зі спеціальною ОС. Тому в цьому розділі буде розглянуто декілька ОС: Windows (рис. 1.9.), iOS (рис. 1.10.) та Android (рис. 1.11.).

Windows Phone

Рис. 1.9. Приклад екрана ОС Windows Phone

Windows Phone — це мобільна операційна система Microsoft і бренд численних конструкцій обладнання, які на ній працюють. З низькою популярністю серед користувачів смартфонів Windows Phone не оновлюється і не вдосконалюється постійно в порівнянні з операційними системами Mac OS і Android. Windows Store застарів і ледве може задовольнити майже всі потреби користувачів, оскільки кількість додатків у ньому перевищує 2 мільйони.

Windows використовує інтерфейс користувача (UI), отриманий від Metro, що точно відображає інтерфейс, подібний до плиток, представлений у настільних системах. Щоб знайти та завантажити програми, музику, відео та подкасти, користувачі можуть перейти до магазину Windows.

Найвідомішими виробниками пристроїв на базі цієї операційної системи можна назвати Nokia і HP. Основними особливостями Windows є мінімалізм, продуктивність і кросплатформенність, яка, хоча і має певні проблеми, орієнтована на користувачів системи Windows в цілому.

IOS

Рис. 1.10. Приклад екрану iOS

iOS — операційна система для планшетів і смартфонів, розроблена американською компанією Apple.

Кількість користувачів iOS на ринку операційних систем для смартфонів становить досить значну частину. iOS використовується лише пристроями від Apple, тоді як Windows і Android можуть використовуватися самими різними виробниками мобільних та інших пристроїв.

Варто визнати, що завантажувати нові програми можливе лише через App Store. Деякі додатки платні, інші традиційно безкоштовні, треті повністю безкоштовні. Кількість додатків iOS в App Store становить понад 300 мільйонів.

Ця операційна система дуже приваблива для розробки, оскільки має широку аудиторію користувачів. Незважаючи на це, є кілька недоліків:

- кількість користувачів iOS менша за кількість користувачів Android;
- вимоги облікового запису розробника для публікації додатків у App Store;
- відсутність необхідних інструментів для розвитку.

Платформа iOS добре розбирається в багатозадачності. Без особливих труднощів ви можете згорнути і розгорнути утиліту. Найголовніше, щоб згорнуті програми не впливали на операційну систему і не знижували заряд акумулятора.

Android

Рис. 1.11. Приклад екрану ОС Android

Android — це операційна система для великої кількості пристроїв. Частка Android на ринку операційних систем є найбільшою.

Ця операційна система має найбільшу користувальницьку аудиторію і, отже, приверне найбільшу кількість користувачів. Проаналізувавши найпопулярніші операційні системи для смартфонів, було прийнято рішення, що розробка програми буде здійснюватися за допомогою сторонніх програм обміну повідомленнями, доступних на всіх платформах.

1.2. Мета розробки та сфера застосування

«Bores» (скорочення від Bottomless Resentfulness) — це назва серверного додатка, створеного під час роботи. Програмним додатком будуть користуватися користувачі, які мають настільний ПК або смартфон під управлінням будь-якої сучасної ОС з доступом до Інтернету.

Мета програми — надати користувачеві можливість брати участь у розважальних текстових розмовах у програмах обміну миттєвими повідомленнями, які підтримують функції бота.

Операційна мета програми — створити комфортне середовище для веселого текстового спілкування та роботи з файлами зображень мемів, а також заощадити час користувача, забезпечити швидкий, зручний, простий та інтуїтивно зрозумілий користувальницький інтерфейс, який не вимагає особливих навичок та умінь користуватися додатком; підтримка переважної більшості сучасних пристроїв.

Bores складається з трьох основних функцій: текстове спілкування, спілкування на основі зображень, навчання за прикладом користувача. На ринку майже немає програм, які могли б поєднати всі такі функції в одному місці.

Ідея створення програми такого типу полягала в тому, щоб створити багатофункціональний і зручний програмний продукт, який допоможе користувачеві проводити вільний час по-новому та цікаво. Крім того, функціонал, яким володіє додаток Bores, не є стандартним і має суттєві

відмінності від таких програм. Ці додаткові функції підтвердять актуальність продукту серед користувачів різного віку та професій.

Підсумовуючи, цей додаток може бути дійсно необхідним користувачам з різних сфер діяльності. Bores завжди будуть під рукою, коли користувачеві потрібно з кимось швидко і дружно поспілкуватися.

1.3. Постановка проблеми

Метою роботи є створення серверного додатка Bores — інтерактивного чат-бота з основним акцентом на текстовому спілкуванні. Додаток розроблено для пристрою з підтримкою веб-перегляду. У IDE Visual Studio 2022 з використанням мови програмування C#.

Простота інтерфейсу повинна дати можливість користуватися утилітою, як молоді, так і людям старшого віку. Додаток разом із додатками для обміну миттєвими повідомленнями утворює повну працездатну систему.

Додаток Bores складається з конвеєра обробки тексту та різних команд, які мають різні цілі та функції. У цій частині буде представлений повний опис технічних завдань для кожної частини.

Структура команд програми складається з наступного:

- Отримати коротку/довгу текстову відповідь.
- Отримати відповідь на основі зображень.
- Змінити налаштування бота.
- Адміністративні.
- Команда перекладу тексту.
- Команда генерації імені.
- Отримайте відповідь на основі ймовірності.

Програму потрібно виконати в Visual Studio IDE для ОС Windows. Додаток Bores має мати головне вікно чату, де буде здійснюватися взаємодія з

користувачем. Крім того, має бути легко дістатися до посібника користувача, яке має бути інтуїтивно зрозумілим і легким для розуміння.

Команда «Отримати коротку/довгу текстову відповідь» дозволяє отримати нове згенероване текстове повідомлення на основі або не на основі введеного користувача. Розмір повідомлення є змінним.

Команда «Отримати відповідь на основі зображення» дозволяє отримати нове згенероване повідомлення з прикріпленим до нього зображенням. Згенерована картинка може бути заснована на попередньо визначеному шаблоні (наприклад, попередньо завантажені шаблони мемів), або на основі введеного тексту або на основі обох цих типів даних.

Команди зміни налаштувань бота в першу чергу призначені для адміністраторів чату, щоб точно налаштувати бота на свій смак. Налаштування включають автоматичне створення зображень, навчання, увімкнення/вимкнення посилань і згадок, шанс відповіді, базове обмеження навчання, максимальну довжину повідомлення та інші. Деякі команди можуть вимагати привілейованого доступу від користувача, який може бути наданий адміністратором.

Адміністративні команди в першу чергу призначені для маніпулювання базою знань бота та збереженими файлами за допомогою самого бота, а не іншими менш зручними засобами. Може бути корисно віддалено вимкнути, перезапустити, перевірити журнал тощо, поговоривши з ботом. Деякі команди можуть вимагати привілейованого доступу.

Переклад тексту дозволяють користувачам перекладати випадковий або наданий текст іншою мовою. За замовчуванням переклад здійснюється українською мовою, але можна вказати іншу підтримувану мову [3].

Команда генерування імені дозволяє користувачам генерувати випадкове ім'я. Розмір генеруємого імені можна зазначити.

Отримати відповідь на основі ймовірності дозволяє отримати ймовірність певного факту чи дії, зазначеної користувачем.

1.4. Вимоги до програмного забезпечення

Основна вимога – надсилати повідомлення через популярні програми миттєвого обміну повідомленнями або соціальні мережі, такі як Telegram, VK, WhatsApp. Ці сервіси виступають в ролі фрот-енду для програми.

1.4.1. Функціональні вимоги

Додаток Bores повинен забезпечувати можливість виконання наступних функцій і можливостей:

- Спілкування на основі тексту, схоже на людину.
- Інтерактивна обробка зображень.
- Можливість вчитися на користувачах та зберігати прогрес.

Надійну (стабільну) роботу Bores має забезпечити користувач, виконавши комплекс організаційно-технічних заходів, перерахованих нижче:

- організація безперебійного електропостачання технічного обладнання;
- використання ліцензійного програмного забезпечення;
- відсутність сторонніх або шкідливих програм, які можуть призвести до вимкнення цієї програми.

Необхідно враховувати доступність інтерфейсу для користувачів під час його проектування та розробки. Додатком можуть користуватися люди, які погано розбираються в технологіях та пошуку інформації.

Тому при розробці інтерфейсу важливо дотримуватися наступних правил:

1. Інтерфейс повинен бути візуальним, тобто розташування елементів має бути зрозумілим і зрозумілим користувачам.
2. Інтерфейс повинен бути виконаний у спокійних, ненав’язливих тонах і бути читабельним.

1.4.2. Вимоги інформаційної безпеки

Вимоги до інформаційних структур і методів вирішення програмного забезпечення та інформаційної безпеки полягають у тому, що системне програмне забезпечення, яке використовується програмою, має бути представлене ліцензованою локалізованою версією операційної системи для конкретного пристрою, сумісної з компонентами програми.

Щоб уникнути некоректної роботи програми, необхідно реалізувати контроль надходження даних, обробку виняткових ситуацій, а також бути впевненим у незмінному стані даних, що зберігаються в конфігураційних файлах, у разі збою програми чи екстремальних обставин.

1.4.3. Вимоги до апаратного середовища

Для стабільної роботи даного програмного забезпечення необхідний пристрій з наступними характеристиками.

Настільний або мобільний пристрій під ОС, який може отримати доступ до веб-браузера для клієнта. Сервер з Ubuntu має починати з версії не нижче 20.04 LTS, має бути встановлений фреймворк .NET Core 5.

1.4.4. Вимоги до сумісності

Програмний продукт розроблено в середовищі Visual Studio 2022.

Програмний код написаний мовою програмування C# за допомогою бібліотек VkNet, Telegram.Bot, Newtonsoft.Json, System.Drawing.Common.

Дані, які додаватимуть користувачі, будуть збережені у файлах .json.

2. РОЗДІЛ 2
2. ДИЗАЙН ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Функціональні цілі програми

Аналіз вимог – це визначення потреб і умов, які розглядає новий або оновлений продукт, беручи до уваги можливі суперечливі вимоги різних клієнтів, наприклад користувачів або бенефіціарів.

Аналіз вимог має вирішальне значення для успішного розвитку проекту. Вимоги повинні бути задокументовані, піддані вимірюванню, тестуванню та описані достатньо детально, щоб розробити програмну систему.

Функціональні вимоги перераховані як функції та послуги, які повинні надаватися системою, а також обмеження системи на основі даних і поведінки під час їх виконання та взаємодії с компонентами.

Специфікації функціональних вимог — опис функцій та їх властивостей, які не містять винятків і будь-яких протиріч.

Додаток Bores — в першу чергу сервіс для текстового спілкування з віртуальним супутником. Для системи, що розробляється, були сформульовані наступні функціональні вимоги, наведені нижче.

Розроблена програма повинна працювати в соцмережі VK, месенджері Telegram та мати необхідну адаптивність для реалізації в рамках інших сервісах даного типу, які надають східний типовий функціонал користувача.

Користувач може надсилати повідомлення, що містять текстові та графічні файли. Ці повідомлення можуть бути оброблені або (і) збережені.

Є можливість перевірити статус розмови, кількість збережених повідомлень, кількість збережених зображень, налаштування користувача.

Користувачі можуть змінювати налаштування бота, щоб увімкнути автоматичне створення зображень, навчання, увімкнення/вимкнення посилань і згадувань, шанс відповіді, базове обмеження навчання, максимальну довжину повідомлення та інші. Деякі команди можуть вимагати привілейованого доступу.

Функціональна мета програми Bores для користувачів – це можливість надавати людські розмови за допомогою введення користувача, а також можливість обробляти, вчитися та виводити дані на основі зображень.

2.2. Опис використаних математичних методів

Оскільки особливості предметної області розв’язуваної задачі не передбачають використання математичних методів, то при розробці програми для роботи з бібліотеками математичні методи не використовувалися.

2.3. Опис використаних шаблонів архітектури та дизайну

2.3.1. JSON

JSON (JavaScript Object Notation) — це простий у використанні формат і метод зберігання даних. Основною особливістю JSON є текстова структура, яка дозволяє легко читати та редагувати людину [10]. Файли даних легко обробляти та створювати машинами. Структура заснована на підмножині стандарту мови програмування JavaScript ECMA-262 3rd Edition — грудень 1999. Формат не залежить від мови програмування та використовує принципи, знайомі розробникам сімейства мов C, наприклад C#, Perl, Java, C, C++, JavaScript та інші. Ці функції роблять JSON найкращим форматом зберігання даних для невеликих файлів, таких як файли конфігурації та налаштувань [27].

JSON заснований на двох основних структурах. Колекції пар імен і значень. У різних мовах програмування він реалізується як структура, об'єкт, словник, списки, запис, хеш-таблиця або асоціативний масив

2.3.2. VkNet

VkNet — це бібліотека, заснована на платформі .NET і повністю написана на C#. Хоча це ще відносно молодий проект, він пропонує багато функцій, таких як доступ до повідомлень, аутентифікація на основі профілю та групи. А також доступ до мультимедійних функцій, звукових повідомлень тощо.

Бібліотека заснована на гнучкої архітектурі, що дозволяє без особливих зусиль адаптувати її до будь-яких потреб.

Основні особливості VkNet включають наступне (як зазначено на офіційному сайті):

- Users
    - Users.Get — Повертає розширену інформацію про користувачів.
    - Users.GetFollowers — Повертає список ідентифікаторів користувачів, які стежать за користувачем.
    - Users.GetSubscriptions — Повертає список ідентифікаторів користувачів і спільнот, які включені до списку підписок користувача.
- Friends
    - Friends.Add — Схвалює або створює запит на дружбу.
    - Friends.AddList — Створює новий список друзів для поточного користувача.
    - Friends.AreFriends — Повертає інформацію про те, чи додано поточного користувача як друга вказаних користувачів.
    - Friends.Delete — Видаляє користувача зі списку друзів або відхиляє запит на дружбу.
    - Friends.DeleteAllRequests — Позначає всі вхідні запити на дружбу як переглянуті.
    - Friends.Get — Повертає список ідентифікаторів друзів користувача або розширений
- Groups
    - Groups.addAddress — Дозволяє додати адресу до спільноти.
    - Groups.AddCallbackServer — Додає до спільноти сервер для API зворотного дзвінка.
    - Groups.AddLink — Дозволяє додавати посилання на спільноту.

    - Groups.ApproveRequest — Дозволяє схвалити запит до групи від користувача.
    - Groups.Ban — Додає користувача до чорного списку спільноти.
- Audio
    - Audio.Get — Повертає список аудіозаписів користувачів або спільноти.
    - Audio.GetById — Повертає інформацію про аудіозаписи.
    - Audio.GetLyrics — Повертає текст аудіозапису.
    - Audio.Search — Повертає список аудіозаписів за вказаними критеріями пошуку.
    - Audio.GetUploadServer — Повертає адресу сервера для завантаження аудіозаписів.
    - Audio.Save — Зберігає аудіозаписи після успішного завантаження.
    - Audio.Add — Копіює аудіозапис на сторінку користувача або групи.
- Messages
    - Messages.AddChatUser — Додає нового користувача до багатодіалогового вікна.
    - Messages.AllowMessagesFromGroup — Дозволяє надсилати повідомлення від спільноти поточному користувачеві.
    - Messages.CreateChat — Створює бесіду з кількома учасниками.
    - Messages.Delete — Видаляє повідомлення.
    - Messages.DeleteChatPhoto — Дозволяє видалити фотографію багатодіалогового вікна.
- Wall
    - Wall.CreateComment — Додає коментар до публікації на стіні.
    - Wall.Delete — Видаляє вхід зі стіни.
    - Wall.DeleteComment — Видаляє коментар поточного користувача до публікації на його чи чужій стіні.

2.3.3. Шаблони дизайну

Шаблон проектування — це спосіб вирішити конкретну проблему, яка є досить типовою і часто трапляється при розробці архітектури програми. Шаблон не можна просто скопіювати в програму, на відміну від готових функцій або бібліотек. Шаблон — це не конкретний код, а загальний принцип вирішення проблеми. Практично завжди його потрібно коригувати для вирішення певної проблеми. Високорівневий опис рішення, реалізація якого може відрізнятися в двох різних програмах, не слід сприймати як алгоритм, який являє собою чіткий набір дій.

Використання шаблонів і свідоме володіння інструментом дозволяє розробнику витрачати менше часу і використовувати готові рішення. У той же час шаблони мають стандартизований код. Однак використання шаблонів не завжди є необхідним і приносить лише ефективні рішення [7].

Багато користувачів, починаючи впроваджувати шаблон у свою програму, забувають налаштувати його під свій проект. У той же час використання візерунків не завжди виправдано.

Під час розробки настільного додатка Bores було використано кілька шаблонів, таких як Factory Method, Flyweight, Iterator, Command [22].

Шаблон проектування Factory Method, який зустрічається дуже часто, забезпечує інтерфейс для створення екземплярів певного класу. Під час створення спадкоємці можуть визначити, який клас створити. Шаблон проектування делегує спадкоємцям батьківського класу створення об'єктів.

Flyweight — це структурна модель. Його основна функція – економія пам’яті за рахунок розподілу загального стану між багатьма іншими об’єктами. Загальна умова повинна бути представлена в одному об'єкті.

Iterator — це поведінковий шаблон проектування. Ітератор дає можливість перевірити всі елементи складеного об'єкта. Ітератор повинен гарантувати, що внутрішня структура об'єкта прихована. Такі ітератори часто використовуються

під час використання фреймворку колекції. Для цього був створений спеціальний інтерфейс користувача, який відповідає .

Шаблон команди є поведінковим. Використання шаблону команд дозволяє розробнику збирати запити або прості операції в окремі об’єкти. Шаблон часто зустрічається у випадках, коли необхідно відкласти виконання команд, побудувати їх у чергу або зберегти історію.

2.3.4. Методологія розробки

Модель життєвого циклу може бути парадигмою розробки програмного забезпечення, яка допомагає програмісту визначити прийнятну стратегію розробки. Методології розробки програмного забезпечення мають власний набір інструментів, процедур і методів для очевидного визначення та визначення життєвого циклу розробки програмного забезпечення [28].

Як правило, життєвий цикл розробки програмного забезпечення включає наступні фази:

Етап 1. Збір та аналіз вимог:

Цей етап дає більш чітке уявлення про масштаби всього проекту і, отже, передбачувані проблеми, можливості та директиви, які ініціювали проект. На цьому етапі додатково виконується планування стандартних вимог щодо впевненості та розпізнавання ризиків.

Етап 2. Техніко-економічне обґрунтування:

Має бути присутнім техніко-економічне обґрунтування, оскільки важливо визначити та задокументувати потреби в програмному забезпеченні. Це означає кожну частину, яку необхідно запрограмувати та спроектувати протягом життєвого циклу проекту. В основному існує п’ять видів перевірок доцільності: технічні, практичні, юридичні, економічні, графіки.

Етап 3. Проектування:

На цьому етапі має бути підготовлений документ зі специфікацією вимог разом із документами з проектуванням системи та програмного забезпечення. Це

допомагає визначити загальну архітектуру системи. Цей етап проектування є вхідним для наступної фази моделі.

Етап 4. Кодування:

На етапі кодування завдання поділяються на блоки або модулі і призначаються різним розробникам. це найдовший етап процесу життєвого циклу розробки програмного забезпечення.

Розробники починають будувати всю систему з написання коду за допомогою обраної штучної мови. вони повинні дотримуватися певних попередньо визначених інструкцій щодо кодування та використовувати інструменти програмування, такі як компілятор, інтерпретатори, налагоджувач, щоб отримати та реалізувати програмний код.

Етап 5. Тестування:

Етап тестування дуже важлива, щоб переконатися, що вимоги замовника були виконані, і, отже, вся програма працює відповідно.

В ідеалі цей процес триває до тих пір, поки програма не буде звільнена від помилок, не буде стабільною та працюватиме відповідно до потреб системи.

Етап 6. Установка/розгортання:

Відповідно до відгуків, представлених менеджером проекту, остаточна версія програми випускається та додатково перевіряється на наявність проблем із розгортанням, якщо це необхідно у процесі.

Етап 7. Технічне обслуговування:

Метою цього етапу SDLC є перевірка того, що вимоги все ще виконуються, і, отже, система працює відповідно до специфікації, зазначеної на першому етапі.

Якість проектування визначає ефективність системи. тому, можливо, кожен етап свідомо поділяється на різні етапи і вимагає підготовки документації, що відображає результати програмної роботи.

Використання життєвих циклів розробки програмного забезпечення (SDLC) під час розробки пакета є дуже важливим [29]. По-перше, він пропонує основи для планування, планування та оцінки проекту, що підвищує видимість проекту

для всіх зацікавлених сторін процесу події. По-друге, SDLC збільшує та покращує швидкість розробки за рахунок наявності механізму відстеження та контролю проекту. Крім того, використання SDLC забезпечує основу для типового набору заходів і результатів, допомагає зменшити ризики проекту та накладні витрати на план управління проектом, а також покращує відносини з клієнтами, що також важливо в бізнесі.

Основними і, отже, найбільш популярними методологіями в розробці програмного забезпечення є модель водоспаду, інкрементальна модель, V-модель, спіральна модель, agile модель і scrum.

Під час розробки настільного додатка була застосована додаткова методологія. Це включає як захід, так і подальшу підтримку товару. він вважається завершеним у той час, коли він відповідає всім необхідним.

Інкрементальна модель не є окремою моделлю. по суті, це серія циклів водоспаду. під час цього методу кожен цикл діє, оскільки фаза обслуговування попереднього випуску програмного забезпечення. Модифікація інкрементальної моделі дозволяє перекривати цикли розробки. У такому випадку наступний цикл може початися до завершення попереднього циклу [29].

На початку проекту товари першої необхідності розподіляються на різні групи. У кожній групі під час заходу дотримуються моделі SDLC. метод SDLC повторюється неодноразово, кожен новий випуск додає нову функціональність до досягнення мети, коли будуть задоволені всі необхідні вимоги. стратегія інкрементного SDLC, в рамках якої проект планується, реалізується та тестується поступово (щоразу з незначними доповненнями), потім до початку циклу подій. Модель поєднує елементи каскадної моделі з прототипуванням.

Інкрементальна модель (рис. 2.1.) функціонує за принципом каскадної моделі з перекриттям, завдяки чому функціональність товару, придатного до експлуатації, виробляється раніше. для цього може знадобитися цілий, заздалегідь сформований набір вимог, які виконуються у вигляді послідовних

невеликих проектів, або реалізація проекту може розпочатися з формулювання спільних цілей, які потім впроваджуються командами розробників [30].

Таке вдосконалення в рамках каскадної моделі однаково ефективне, якщо використовується у випадку надзвичайно великих і малих проектів.

Модель допомагає «згладити кути», а не миттєво розгортати користувачеві зовсім нову систему. Проект розкладається на кілька компонентів, кожна з яких задумана і побудована незалежно від протилежної (build). Кожен компонент доставляється клієнту відразу, оскільки він готовий, що дозволяє клієнту миттєво почати використовувати товар і уникнути тривалої розробки. Це також стимулює великі інвестиційні витрати, але скорочує час очікування результату.

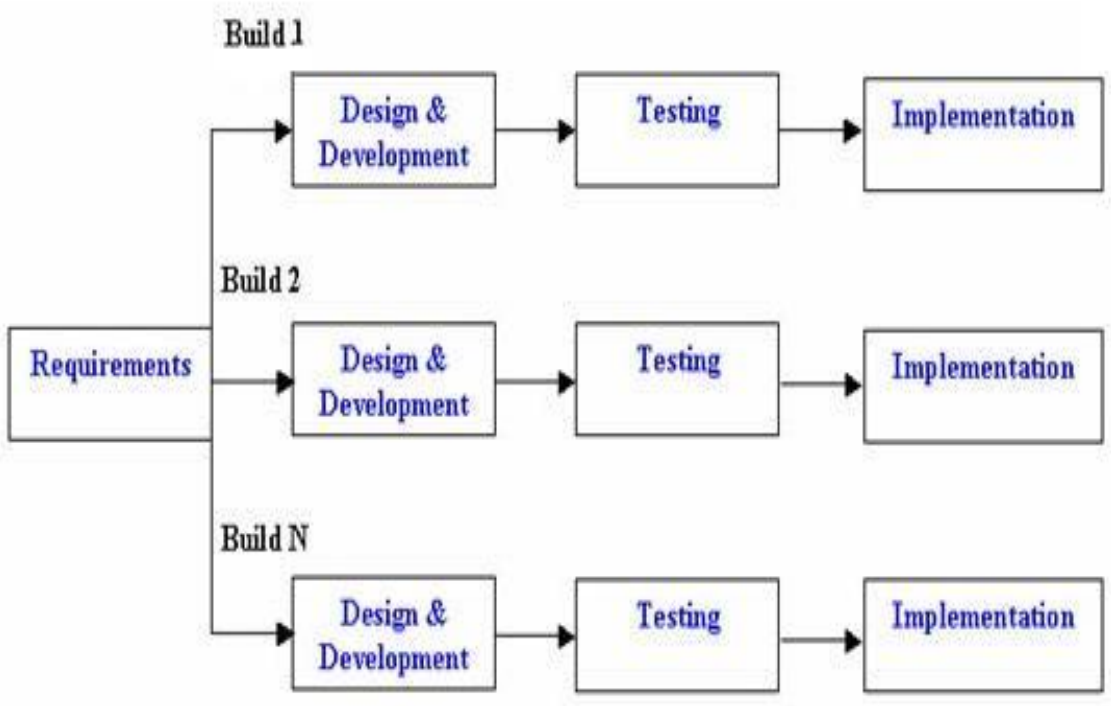

Рис. 2.1. Модель інкрементального життєвого циклу

Перевагами інкрементальної моделі є те, що робоча програма виходить на ранніх етапах життєвого циклу продукту та гнучкість. Змінити обсяг і вимоги проекту відносно дешевше. З невеликими ітераціями тестування та редагування

стають легшими. Можна визначити ризики. Кожна ітерація є простим у керуванні орієнтиром даного проекту.

Що стосується недоліків інкрементальної моделі, то це те, що кожна фаза ітерації є фіксованою. Також можуть виникнути проблеми щодо архітектури системи, оскільки не всі вимоги збираються заздалегідь на весь життєвий цикл програмного забезпечення [31].

Інкрементальна модель використовується разом із чіткими і зрозумілими вимогами, які реалізуються поетапно.

2.4. Опис використовуваних технологій та мов програмування

C# (вимовляється як «see Sharp» або «C Sharp») — це мова програмування з сімейства мов .NET. Він дозволяє створювати широкий спектр додатків і є об'єктно-орієнтованим. C# був представлений Microsoft 26 червня 2000 року, і він став популярною багатоцільовою мовою програмування в 21 столітті [21].

Мова є еволюцією сім'ї мов C. Він також запозичує функції з альтернативних мов програмування, наприклад Java і Delphi. Таким чином, основний синтаксис Java і C# виглядає дуже знайомим. Код також навмисно нагадує мову C++ [21]. C# підтримує певні функції або працює певним чином, оскільки має коріння в його спадщині C++. Новітні можливості мови (наприклад, LINQ — Language Integrated Query) та асинхронне програмування (асинхронне) присутні не тільки в C#, але роблять мову відносно унікальною.

C# — це керована мова, яка вимагає виконання .NET CLR. Під час виконання програми CLR відповідає за керування пам'яттю, винятки, що виникають, функціональність GC та інші речі. Компілятор C# компілює код не безпосередньо на машинну мову, а на проміжну мову (IL). Цей IL розуміє CLR. JIT-компілятор CLR компілює код IL, метод за методом, у машинний код і виконує його. Для виконання коду C# потрібен CLR. Усі нові ОС Windows мають

попередньо встановлену версію CLR, і вона доступна в старих системах через оновлення операційної системи [20].

Незважаючи на ці обмеження, оскільки C# має доступ до FCL, можна багато чого зробити. Наприклад, FCL дозволяє писати настільні програми з WPF, WinForms і консольними додатками. Для Інтернету можна використовувати програми ASP.NET. Для роботи з даними є ADO.NET Entity Framework і LINQ. Новітні функції .NET включають Windows Store і підтримку Windows Phone. Масштабовані хмарні програми можуть бути написані за допомогою Windows Azure. Це лише деякі з доступних функцій. З універсальною мовою програмування загального призначення, як-от C#, ви можете зробити набагато більше, ніж просто зі звичайними аналогами [9].

2.5. System Structure and Description of its Functional Algorithms

Структура програми наведена нижче на малюнку 2.2. За структурою він має п’ять основних функцій з підфункціями.

Функція текстової відповіді надає користувачеві можливість отримувати текстові повідомлення як відповіді на свої власні. Також можна навмисно створити коротку або довгу відповідь.

У функції налаштувань можна автоматично генерувати зображення, навчатися, увімкнути/вимкнути посилання та згадування, шанс відповіді, базове обмеження навчання, максимальну довжину повідомлення та інші. Деякі команди можуть вимагати привілейованого доступу.

Переклад дозволяє користувачам перекладати випадковий або наданий текст іншою мовою. За замовчуванням переклад здійснюється українською мовою, але можна вказати іншу підтримувану мову [3].

Адміністративні команди в першу чергу призначені для маніпулювання базою знань бота та збереженими файлами за допомогою самого бота, а не іншими менш зручними засобами. Може бути корисно віддалено вимкнути,

перезапустити, перевірити журнал тощо, поговоривши з ботом. Деякі команди можуть вимагати привілейованого доступу.

Команда генерування імені дозволяє користувачам генерувати випадкове ім'я. Розмір можна вказати у вхідних даних.

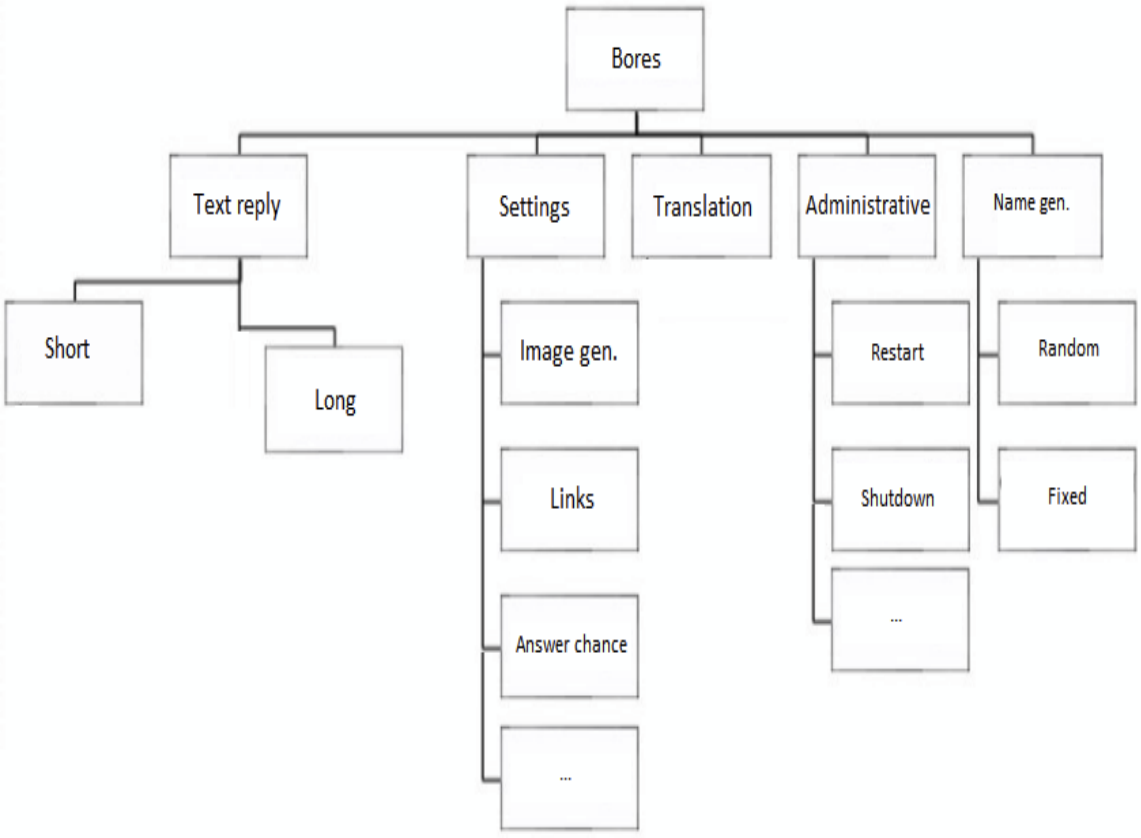

Рис. 2.2. Структура програми Bores.

Клас BoresEngine.cs є основним механізмом бота, який має обробку повідомлень і зображень. Клас BoresEngine реалізує основні текстові методи GetRandResponse, GetShortResponse, GetAdvancedResponse, LearnMsg. Вони допомагають обробляти NewMessage, що виникає в Program.cs з новими повідомленнями від користувачів програми.

Розмір бази знань може бути змінений, навчання відбувається автоматично. У методі ProcessNewMessage можуть виконуватися різні команди з параметрами консольного стилю, які передає користувач у повідомленнях.

Клас ChatSettings.cs створюється з налаштуваннями програми. Він має налаштування MustMakeMemes, MustLearn, EnableMentions, EnableLinks, SaveImages, AnswerChance, SavedImagesCap, DialogCap, ReplyRandomness. А також методи SetByName, Save, Load.

Метод Save зберігає налаштування у файлі JSON. Файл налаштувань є унікальним для кожної бесіди (ідентифікатор діалогового вікна).

```
public string GetList(bool isGod)
{
    string list = "";
    foreach (Setting setting in SettingsBool)
        if (!setting.SenderHasToBeCreator || isGod)
            list += setting.ToString() + "\n";
    foreach (Setting setting in SettingsInt)
        if (!setting.SenderHasToBeCreator || isGod)
            list += setting.ToString() + "\n";
    return list;
}
```

Щоб змінити налаштування за назвою, використовується метод SetByName:

public string SetByName(string settingName, string newValue, UserPrivileges privilege)

```
{
    if (privilege == UserPrivileges.Regular)
        return "Error;
    if (privilege == UserPrivileges.Unknown)
        return " Error";
    foreach (Setting setting in SettingsBool)
    {
        if (setting.Name == settingName || setting.NameShortcut == settingName.ToLower())
```

```
            {
                bool result = setting.Set(newValue, privilege);
                if (result)
                {
                    Save();
                    return $"{setting.Name} set to {newValue}";
                }
                else
                    return $" Error! {newValue} is either outside the range {setting.GetRange()}" +
                        $" or you don't have the permission to access it!";
            }
        }
        foreach (Setting setting in SettingsInt)
        {
            if (setting.Name == settingName || setting.NameShortcut == settingName.ToLower())
            {
                bool result = setting.Set(newValue, privilege);
                if (result)
                {
                    Save();
                    return $"{setting.Name} set to {newValue}";
                }
                else
                    return $" Error! {newValue} is either outside the range {setting.GetRange()}" +
                        $" or you don't have the permission to access it!";
            }
```

```
        }
        return $" Error! Setting {settingName} doesn't exist!";
    }
```

MemeTemplate.cs містить попередньо визначені шаблони на основі зображень, які можна використовувати, викликавши функцію Generate, усі необхідні параметри, пов’язані з зображеннями, такі як координати та метадані.

```
        public Bitmap Generate(Bitmap[] imgs, params string[] texts)
        {
            if (imgs != null && imgs.Length != ImgBlocks.Count)
                throw new ArgumentException($"Passed {imgs.Length}" +
        $" texts for a template with {ImgBlocks.Count} msg blocks!");
            if (TextBlocks.Count != texts.Length)
                throw new ArgumentException($"Passed {texts.Length}" +
                    $" texts for a template with {TextBlocks.Count} msg blocks!");

            Bitmap meme = new(OrigImage);

            for (int i = 0; i < texts.Length; i++)
            {
                using Graphics g = Graphics.FromImage(meme);
                TextBlocks[i].Draw(texts[i], g);
                //TextBlocks[i].DrawRepeating("test ", g);
            }
            if (imgs != null)
            {
                for (int i = 0; i < ImgBlocks.Count; i++)
                {
                    using Graphics g = Graphics.FromImage(meme);
                    //g.DrawImage(imgs[i], ImgBlocks[i]);
```

```
            ImgBlocks[i].Draw(imgs[i], g);
        }
    }

    return meme;
}
```

Конфігураційні файли програми – це файли JSON, які складаються з масивів елементів різного типу: Bool (рис. 2.3.) та Int (рис. 2.4.). Збережені дані представлені у вигляді файлу наступної структури. Різні змінні зберігаються як колекції пар імен і значень і можуть бути відредаговані в текстовому редакторі. Різні змінні зберігаються як колекції пар імен і значень і можуть бути відредаговані в текстовому редакторі.

Рис. 2.3. Налаштування типу Bool

Рис. 2.4 Налаштування типу Int

2.6. Раціоналізація та організація вхідних та вихідних даних

Коли програма запускається, вона завантажує свої бази знань і налаштування з кожної діалогової папки, ініціалізує механізми. Файл налаштувань і база знань створюються автоматично при відкритті нового діалогового вікна. Усі зміни, внесені користувачем, автоматично зберігаються протягом певної кількості ітерацій.

Було вирішено використовувати файли конфігурації JSON для конкретних користувацьких налаштувань діалогу, таких як увімкнення/вимкнення посилань та згадок, шанси відповідей, базове обмеження навчання тощо. Будь-які зміни зберігаються автоматично на диск.

За замовчуванням зображення користувачів зберігаються в папці img у відповідній діалоговій папці. Збереження зображень можна вимкнути, якщо для параметра SaveImages встановлено значення false.

2.7. Розроблений опис програмного продукту

2.7.1. Використані апаратні ресурси

Для розробки цього програмного забезпечення використовувався ПК з наступними характеристиками:

– AMD Ryzen 9 5950X (3.5 – 4.7 GHz);

– 64GB of RAM;
– Nvidia RTX 3090 Ti;
– OS Microsoft Windows 10;
– 49" monitor;
– keyboard, mouse;
– access to the Internet.

2.7.2. Використані програмні ресурси

Додаток Bores написаний об’єктно-орієнтованою мовою програмування C# в Windows Studio IDE. Для зберігання налаштувань вибрано JSON і XML.

Система сумісна з серверами Windows 10 і Ubuntu.

Для роботи серверної програми необхідна інсталяція платформи .NET Core.

2.7.3. Відкриття та завантаження програми

Для використання бота користувачеві необхідно завантажити або відвідати vk.com, або завантажити додаток Telegram. Користувач повинен знайти чат-бота за допомогою пошуку а потім відкрити діалогове вікно. Програма не вимагає спеціальної процедури встановлення.

2.7.4. Опис інтерфейсу користувача

Коли користувач вперше відкриває діалог з ботом, надсилається повідомлення з корисним посібником (рис. 2.5.).

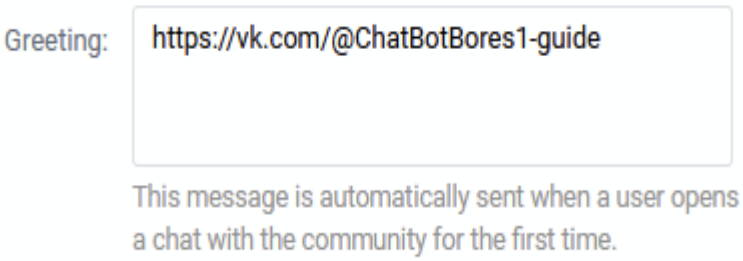

Рис. 2.5. Привітання в налаштуваннях ВК

Вікно чату (рис. 2.6.) складається з повідомлень від користувача та бота. Бот може говорити кількома мовами (рис. 2.7.) за умови, що він пройшов навчання на необхідному наборі даних і має достатню базу знань.

У діалоговому вікні з самим ботом властивість AnswerChange встановлюється на 100, тому на кожне повідомлення від користувача буде відповідь від Bores із спільної бази знань приватного чату.

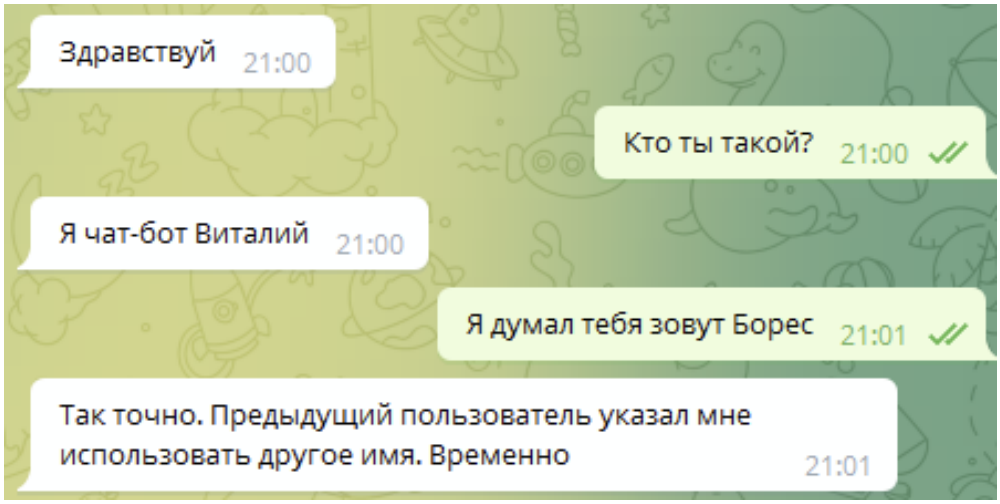

Рис. 2.6. Чат з Bores в месенджері Telegram російською

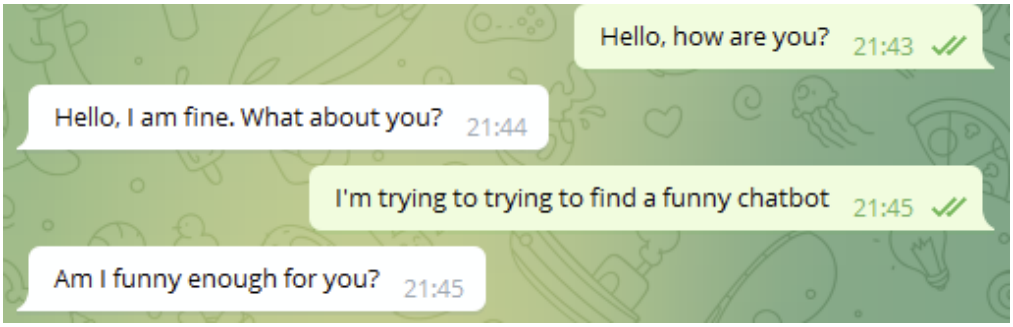

Рис. 2.7. Чат з Bores у месенджері Telegram англійською

Bores можна запросити на будь-який номер розмови з кількома користувачами та іншими ботами. За умови, що йому було надано дозвіл на перегляд повідомлень у чаті, Bores може автоматично обробляти їх,

використовуючи реалізацію довгого опитування та запускаючи події NewMessage для кожного нового повідомлення в чаті.

Деякі адміністративні функції можуть вимагати підвищених привілеїв користувача (рис. 2.8.) у чаті. Кожен чат складається з власної бази знань, файлів зображень, налаштувань та допоміжних файлів.

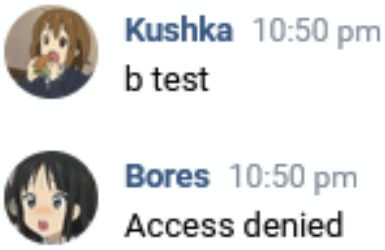

Рис. 2.8. Виклик адміністративної команди (недостатні привілеї)

Якщо під час обробки нового повідомлення має статися необроблений виняток, деталі винятку будуть записані, а адміністратор буде повідомлений та наданий необхідним описом, деталями, вмістом повідомлення, ідентифікатором чату в повідомленні від Bores. У цьому випадку (рис. 2.9.) команда “b test” призначена для виконання “throw new Exception(“test ex”)”, що призводить до необробленого повідомлення про виключення.

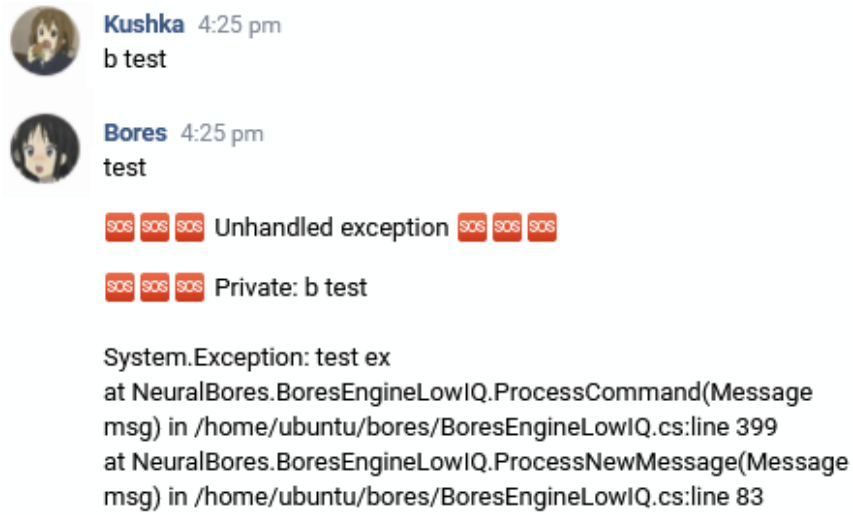

Рис. 2.9. Обробка винятків

Налаштування мають три різні рівні привілеїв: звичайний, адміністратор і автор. У прямому чаті користувачі мають дозволи за замовчуванням і не можуть змінювати будь-які налаштування або викликати адміністративні команди.

Адміністратори чату мають доступ до більшості налаштувань (рис. 2.10.). Їх можна змінити за допомогою команди «b set» (рис. 2.11.).

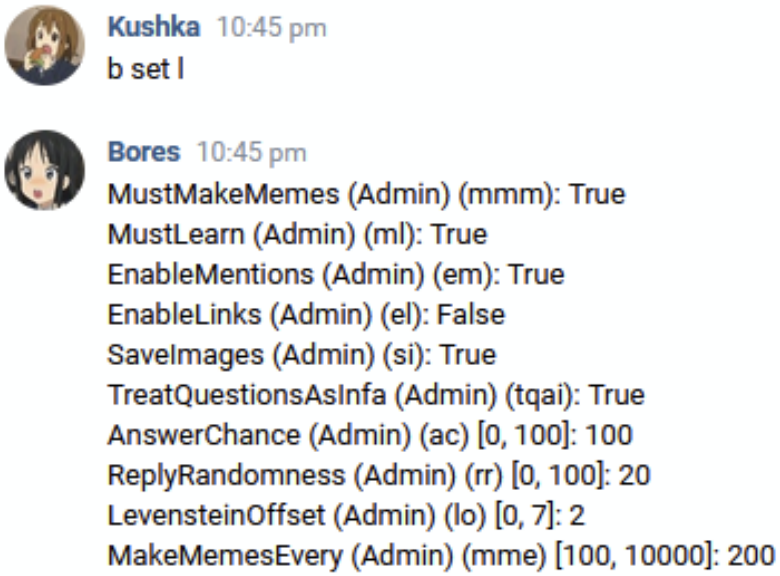

Рис. 2.10. Налаштування адміністратора

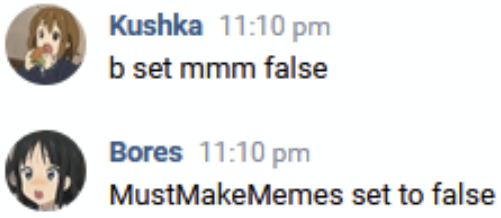

Рис. 2.11. Зміна значення налаштування

Користувачі можуть перевірити статус бота. Детальність статусу визначається рівнем привілеїв користувача (рис. 2.12.).

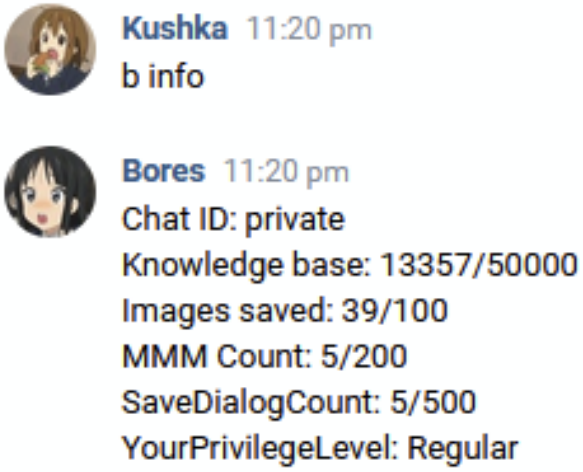

Рис. 2.12. Статус

Bores має різноманітні шаблони мемів. Їх можна перерахувати за допомогою команди «b it l» (рис. 2.13.). Шаблон може бути використаний командою «b it name text» (рис. 2.14.). Bores буде враховувати текст, наданий користувачем, здійснювати пошук у його базі знань або шукати відповідне зображення в мережі Інтернет за допомогою пошукових сервісів.

У випадку використаного шаблону «Guys literally only want one thing» Bores сформулює більш відповідний запит для пошуку зображень, запустивши текст користувача у свій алгоритм генерації тексту (рис. 2.15.). Після цього він вибере зображення, яке не перевищує певний розмір, і передасть відповідний BitMap у метод MemeTemplate.Generate().

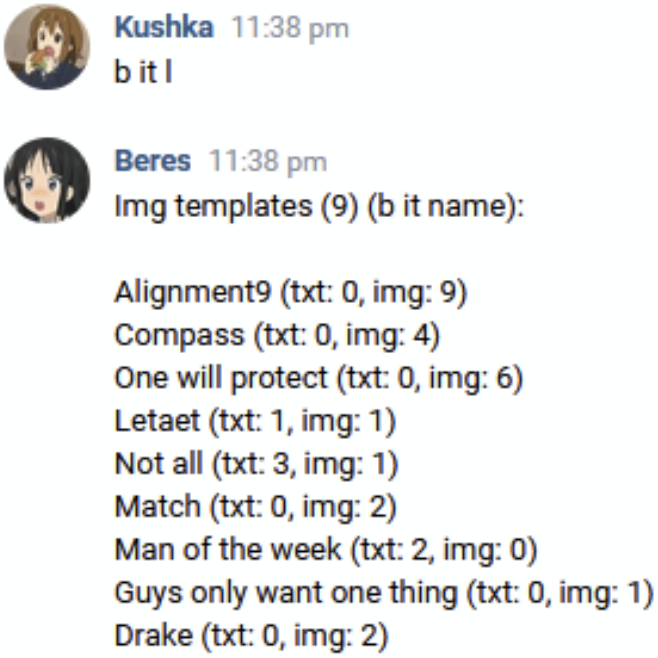

Рис. 2.13. Шаблони мемів

Рис. 2.14. Генерація мемів на основі шаблонів

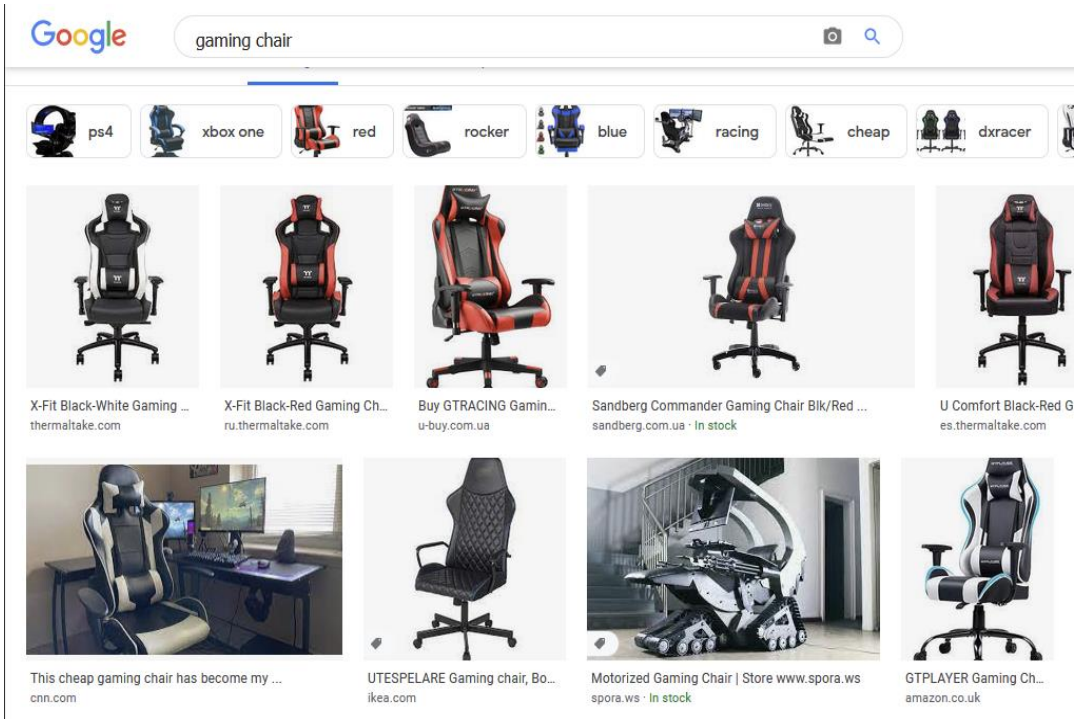

Рис. 2.15. Пошук зображень на основі тексту користувача

Шаблон мему може містити текстові та графічні блоки. Користувач може вказати використане зображення та потрібний текст. Bores візьме до уваги інформацію про запитуваний шаблон, історію чату, а також спробує знайти зв'язок між наданими даними для роботи.

Ще однією особливістю Bores є можливість створення демотиваційних плакатів (рис. 2.16.). Можна використовувати наданий користувачем текст та/або зображення. Також є можливість збільшення текстових блоків.

Команди генерації зображень можуть бути викликані як неявно, так і явно. У разі явного виклику користувач повинен викликати команду «b demo» із зображенням або без нього. Неявний виклик (рис. 2.17.) повинен містити намір, пов’язаний із зображенням (і бажано зображення). Якщо інструкція зрозуміла правильно, бот спробує придумати зображення, створене на основі даних, наданих користувачем у повідомленні.

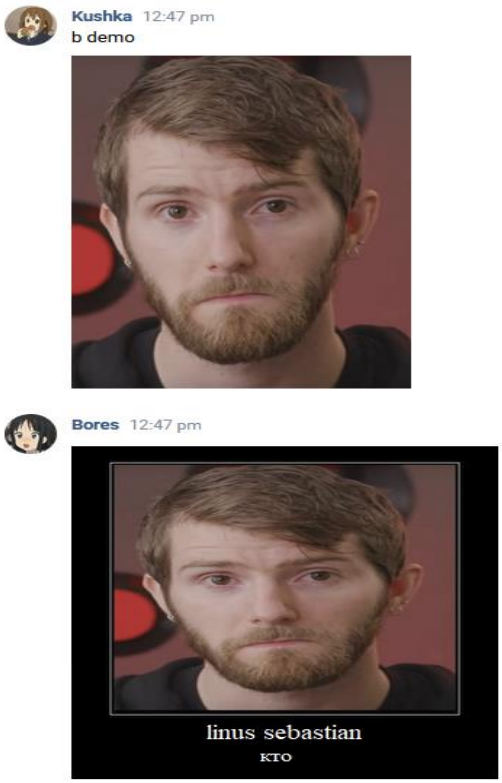

Рис. 2.16. Демотиваційний плакат із зображенням користувача (явна команда)

Рис. 2.17. Демотиваційний плакат із зображенням користувача (неявна команда)

Для отримання інформації про зображення використовується Google (рис. 2.18.) зворотний пошук зображень API (рис. 2.19.). При необхідності інформація передається в алгоритм генерації тексту.

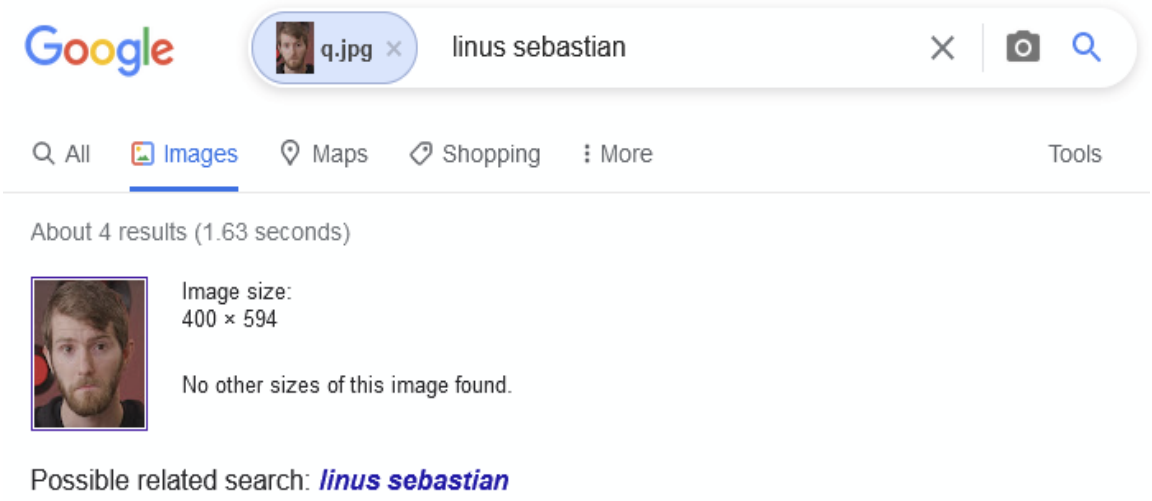

Рис. 2.18. Зворотний пошук зображень Google №1

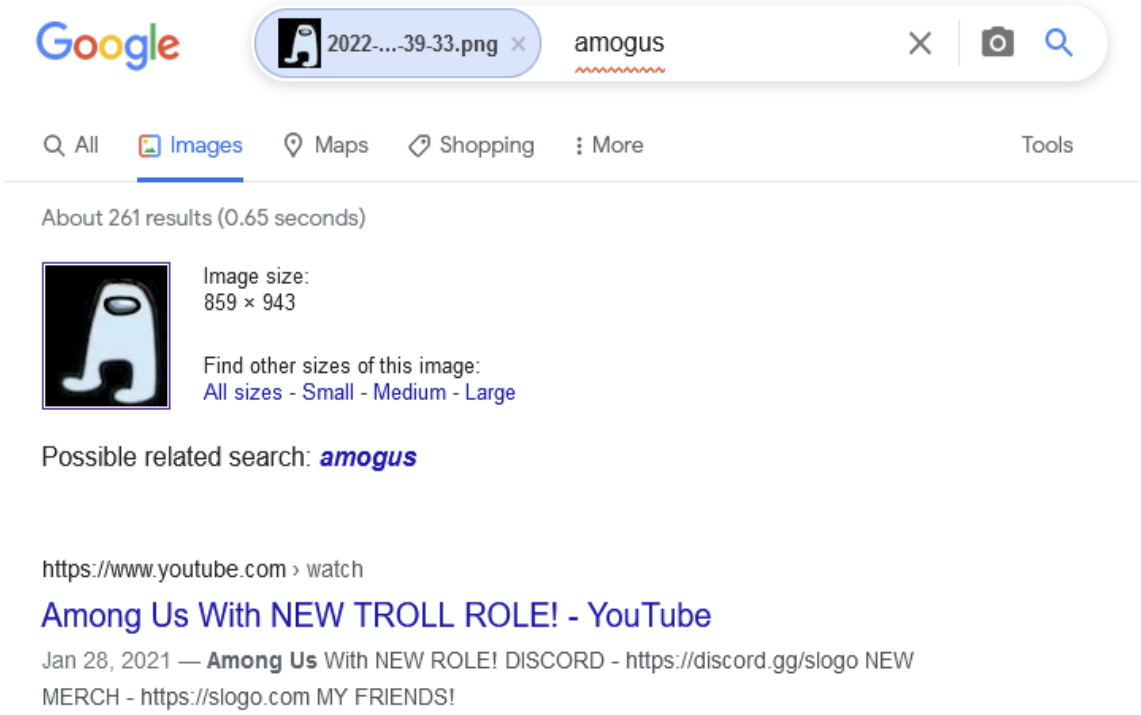

Рис. 2.19. Зворотний пошук зображень Google №2

Алгоритм генерації демотиватора можна кілька разів зациклити, додавши відповідне число після «demo» (рис. 2.20.). Ключове слово «last» повідомляє боту для повторного використання останнього кешованого зображення з чату.

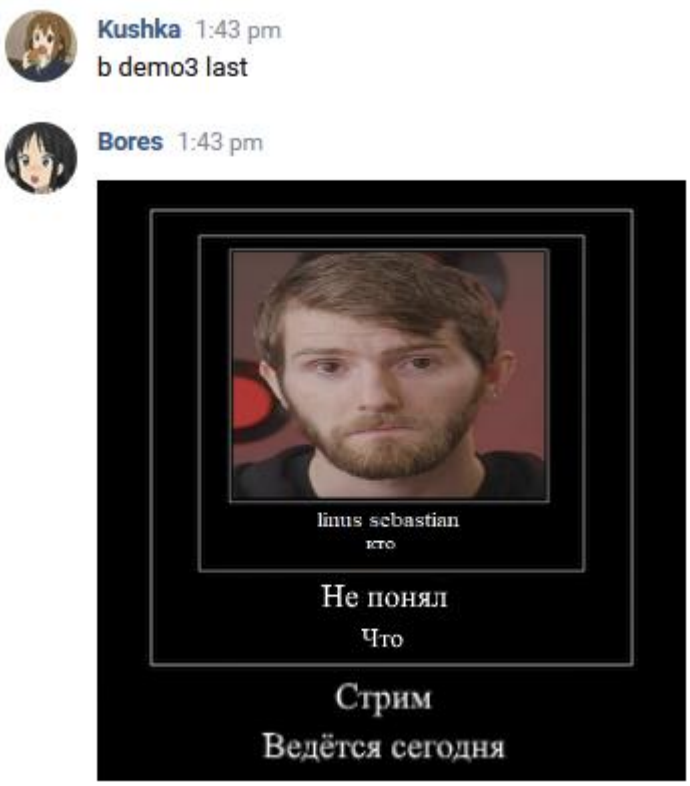

Рис. 2.20. Генерація зацикленого демотиватора

Згенерувати випадкове ім’я можна за допомогою бази знань діалогу (рис. 2.21.). Рекомендується використовувати команду, коли база знань досить велика.

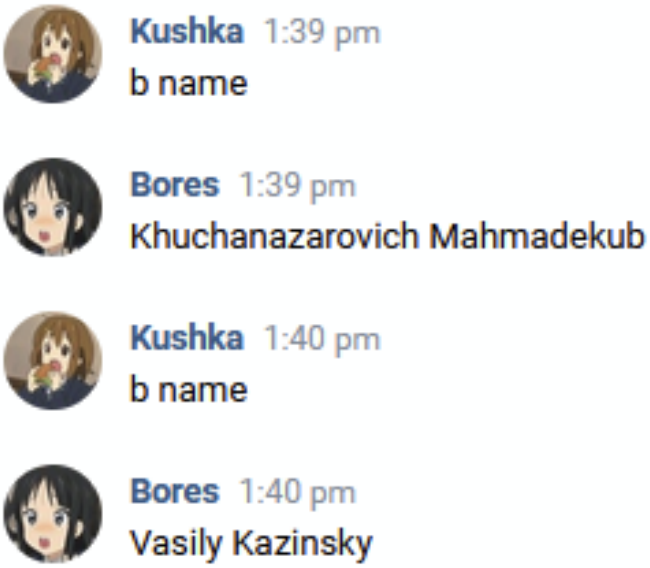

Рис. 2.21. Генерація імені

Команда перекладу (рис. 2.22.) дозволяє користувачам перекладати випадковий (рис. 2.23.) або наданий текст іншою мовою за допомогою Microsoft Translator API. За замовчуванням переклад здійснюється українською мовою, але можна вказати іншу підтримувану (рис. 2.24.) мову [3].

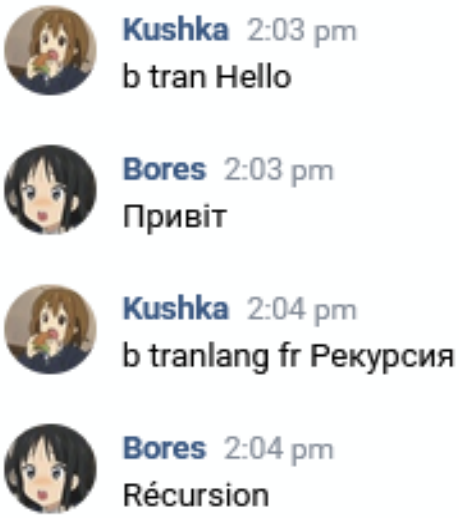

Рис. 2.22. Переклад тексту українською та французькою мовами

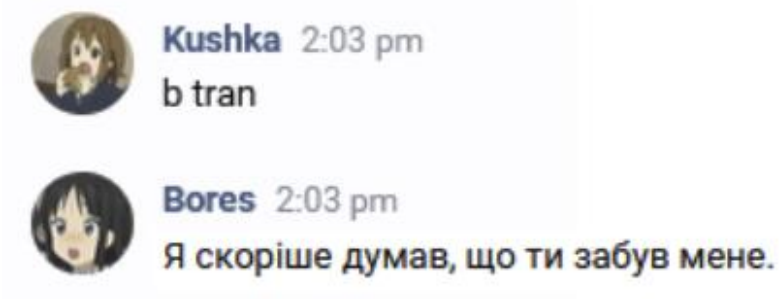

Рис. 2.23. Переклад тексту випадково згенерованого повідомлення

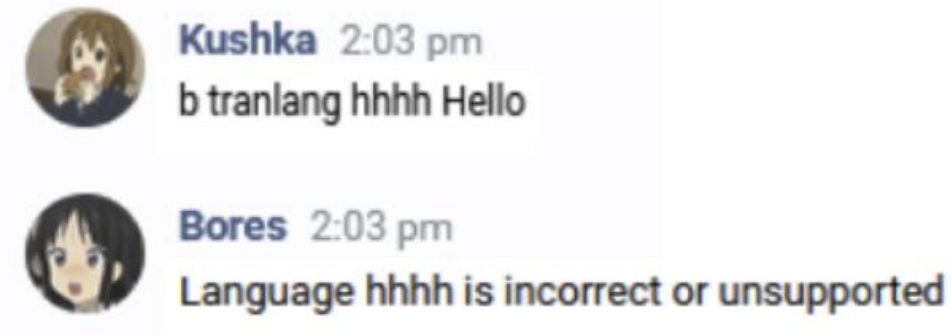

Рис. 2.24. Помилка про неправильну чи непідтримувану мову

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Визначення трудомісткості розробки програмного забезпечення

Початкові дані:

1) передбачуване число операторів програми – 1000;
2) коефіцієнт складності програми – 1,6;
3) коефіцієнт корекції програми в ході її розробки – 0,4;
4) годинна заробітна плата програміста, грн/год - 70;
5) коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,2;
6) коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1,5;
7) вартість машино-години ЕОМ, грн/год - 15.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_u + t_a + t_n + t_{отл} + t_д, \text{ людино-годин,} \qquad (3.1)$$

де t_o - витрати праці на підготовку й опис поставленої задачі (приймається 50);

$t_и$ - витрати праці на дослідження алгоритму рішення задачі;

$t_а$ - витрати праці на розробку блок-схеми алгоритму;

$t_п$ - витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ - витрати праці на налагодження програми на ЕОМ;

$t_д$ - витрати праці на підготовку документації.

Умовне число операторів (підпрограм):

$$Q = q \times C \times (1 + p),$$

де q - передбачуване число операторів,

C - коефіцієнт складності програми,

p - коефіцієнт кореляції програми в ході її розробки,

Q - передбачуване число операторів,

С коефіцієнт складності програми. Коефіцієнт складності завдання З характеризує відносну складність програми по відношенню до так званої типової задачі, що реалізує стандартні методи рішення, складність якої прийнята рівною одиниці (величина С лежить в межах від 1,25 до 2). Для даного програмного продукту, з урахуванням великої кількості і різноманітності оброблюваної інформації і складності складання звітів, коефіцієнт складності завдання візьмемо 1,6.

$$Q = q * C * (1 + p) \quad (3.2)$$

де q - передбачуване число операторів;

С - коефіцієнт складності програми;

р - коефіцієнт корекції програми в ході її розробки.

С - коефіцієнт складності програми.

Коефіцієнт складності завдання характеризує відносну складність програми по відношенню до так званої типової задачі, що реалізує стандартні методи рішення, складність якої прийнята рівною одиниці (величина С лежить в межах від 1,25 до 2). Для даного програмного продукту, з урахуванням великої кількості і різноманітності оброблюваної інформації і складності складання звітів, коефіцієнт складності завдання візьмемо 1,6.

Р коефіцієнт корекції програми в ході її розробки. Коефіцієнт корекції програми р - збільшення обсягу робіт за рахунок внесення змін до алгоритму або програму за результатами уточнення постановок. В даному випадку програма вимагала численних доробок. З урахуванням цього візьмемо коефіцієнт рівний 0,4.

$$Q=1000*1,6*(1+0,4)= 2240, \text{ людино-годин.} \quad (3.3)$$

Витрати праці на вивчення опису задачі $t_и$ визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q * B}{(75 \dots 85) * k}, \text{ людино-годин,} \quad (3.4)$$

де B - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

k - коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності.

В коефіцієнт збільшення витрат праці внаслідок недостатнього опису завдання. Коефіцієнт збільшення витрат праці в залежності від складності завдання приймається від 1,25 до 1,5, внаслідок недостатнього опису рішення задачі приймемо В = 1,3.

К коефіцієнт кваліфікації програміста, який визначається від стажу роботи за даною спеціальністю. К = 1,5.

$$t_u = \frac{2240*1,3}{80*1,5} = \frac{2912}{112,5} = 24,27, \text{ людино-годин.} \quad (3.5)$$

Витрати праці на розробку алгоритму рішення задачі:

$$t_a = \frac{Q}{(20 \dots 25) * k}, \text{ людино-годин.} \quad (3.6)$$

$$t_a = \frac{2240}{22*1,5} = 67,879, \text{ людино-годин.} \quad (3.7)$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20\ldots25)*k}, \text{ людино-годин} \quad (3.8)$$

$$t_n = \frac{2240}{23 * 1{,}5} = 64{,}928, \text{ людино-годин.} \quad (3.9)$$

Витрати праці на налагодження програми на ЕОМ:

- за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4\ldots5)*k}, \text{ людино-годин,} \quad (3.10)$$

$$t_{отл} = \frac{2240}{5 * 1{,}5} = 298{,}667, \text{ людино-годин,} \quad (3.11)$$

- за умови комплексного налагодження завдання:

$$t^k_{отл} = 1{,}5 * t_{отл}, \text{ людино-годин.} \quad (3.12)$$

$$t^k_{отл} = 1{,}5 * 298{,}667 = 448, \text{ людино-годин.} \quad (3.13)$$

Витрати праці на підготовку документації:

$$t_д = t_{др} + t_{до}, \text{ людино-годин,} \quad (3.14)$$

де $t_{др}$ - трудомісткість підготовки матеріалів і рукопису.

$$t_{др} = \frac{Q}{15..20*k}, \text{ людино-годин.} \quad (3.15)$$

$$t_{др}=\frac{2240}{18*1,5}=82,963, \text{ людино-годин,} \quad (3.16)$$

$t_{до}$ - трудомісткість редагування, печатки й оформлення документації

$$t_{до} = 0,75 * t_{др}, \text{ людино-годин} \quad (3.17)$$

$$t_{до}=0,75*82,963=62,222, \text{ людино-годин.} \quad (3.18)$$

$$t_{д}=82,963+62,222=145,185, \text{ людино-годин.} \quad (3.19)$$

Отримуємо трудомісткість розробки програмного забезпечення:

$$t=50+24,27+67,879+448 + 145,185=735,334, \text{ людино-годин.} \quad (3.20)$$

3.2. Витрати на створення програмного забезпечення

Витрати на створення ПЗ Кпо включають витрати на заробітну плату виконавця програми Зз/п і витрат машинного часу, необхідного на налагодження програми на ЕОМ

$$К_{по}=З_{зп}+З_{мв}, \text{ грн.} \quad (3.21)$$

Заробітна плата виконавців визначається за формулою:

$$З_{зп}=t*С_{пр}, \text{ грн,} \quad (3.22)$$

де: t - загальна трудомісткість, людино-годин;

$С_{пр}$ - середня годинна заробітна плата програміста, грн/година

$$З_{зп}=735,334*70=51473,38, \text{ грн.} \quad (3.23)$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{мв}=t_{отл}*C_{мч}, \text{ грн}, \quad (3.24)$$

де $t_{отл}$ - трудомісткість налагодження програми на ЕОМ, год.

$C_{мч}$ - вартість машино-години ЕОМ, грн/год.

Визначені в такий спосіб витрати на створення програмного забезпечення є частиною одноразових капітальних витрат на створення АСУП.

$$З_{мв}=298,667*15=4480, \text{ грн.} \quad (3.25)$$

$$К_{по}=51473,38+4480=55953,38, \text{ грн.} \quad (3.26)$$

Очікуваний період створення ПЗ:

$$Т=\frac{t}{B_k*F_p}, \text{ міс}, \quad (3.27)$$

де B_k - число виконавців;

F_p - місячний фонд робочого часу (при 40 годинному робочому тижні F_p=176 годин).

B_k = 1

$$Т=\frac{735,334}{1*176}=4,2, \quad \text{міс.} \quad (3.28)$$

3.3. Маркетингові дослідження ринку збуту розробленого програмного продукту

Так як тема кваліфікаційної роботи присвячена дослідженням, пов’язаним з обробкою природної мови, дослідженням алгоритмів машинного навчання у чатових системах, то можна зробити висновок, що робота комерційних перспектив не має, тому маркетингові дослідження проведені не були.

Однак, отримані результати роботи можуть бути використані під час створення систем чат-ботів, що дозволить досягти соціального ефекту завдяки впровадженню нового переважно розважального чат-боту на зростаючому ринку віртуальної комунікації.

3.4. Маркетингові дослідження ринку збуту розробленого програмного продукту

Виконана робота має дослідницький характер, не передбачає створення кінцевого продукту, тому чисельний розрахунок економічної ефективності не може бути виконаний.

Висновок: у результаті виконання кваліфікаційної роботи був створений застосунок “Bores”. У даному економічному розділі було визначено трудомісткість на розробку додатку, що складає 735.3 люд-год, проведено підрахунок вартості роботи по створенню програми, які склали 55953,4 грн. та розраховано час на його створення – 4,7 міс.

ВИСНОВКИ

Метою кваліфікаційної роботи є розробка програми чат-бота, яка забезпечує можливість створювати людські розмови з використанням введення користувача, а також можливість обробляти, вивчати та виводити дані на основі зображень.

Програма являє собою систему, яка дозволяє створювати розмови та зображення, виконувати команди, вивчати дані, надані користувачем. Додаток розроблено для серверів під керуванням Ubuntu або Windows Server під ОС Windows. Середовище розробки — Visual Studio з мовою програмування C#.

Настільна програма Bores написана об’єктно-орієнтованою мовою програмування C# в Visual Studio IDE. Система сумісна з Ubuntu 20.04, Windows 10 і пізнішими версіями.

Bores має конфігураційні файли JSON і XML для зберігання інформації та має набір даних, організований за певними правилами, включаючи загальні принципи опису, зберігання та маніпулювання даними. Для всіх конкретних даних користувача було вирішено використовувати JSON, який зберігає інформацію в спеціальних файлах «dialog.json», «settings.json» та папках.

Додаток Bores складається з кількох основних частин, які мають різні цілі та функції. Простота інтерфейсу дає можливість користуватися утилітою, як молоді, так і людям старшого віку. Додаток разом з VK, Telegram або іншим сервісом повідомлень утворює цілісну працездатну систему.

Реалізація даного програмного продукту є економічно вигідною, оскільки його повна вартість помірна за рахунок використання некомерційних інструментів для розробки, відсутності потреби в конфігурації та супроводі робочих місць користувачів. Також до та під час розробки була визначена трудомісткість розробки програми, яка становить 735,334 люд.-год., проведено розрахунок вартості робіт зі створення програми, яка склала 55953,38 грн. і розрахунковий час на його створення – 4,7 місяця.

В результаті виконання дипломного проекту удосконалено знання з програмування консольного додатка мовою C#. Знайдено нові можливості середовища розробки Visual Studio, досліджено набуті навички використання науково-технічної інформації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Cleverbot chatbot. https://www.cleverbot.com/
2. SimSimi chatbot. https://simsimi.com/
3. Microsoft Translator supported languages. https://docs.microsoft.com/en-us/azure/cognitive-services/translator/language-support
4. Mark J. Price. C# 8.0 and .NET Core 3.0 – Modern Cross-Platform Development: Build applications with C#, .NET Core, Entity Framework Core, ASP.NET Core, and ML.NET using Visual Studio Code, 4th Edition. 2019, 154 p.
5. John Sharp. Microsoft Visual C# Step by Step (8th Edition) (Developer Reference). 2018, 384 p.
6. Tony Gaddis. Starting out with Visual C# (4th Edition). 2017, 221 p.
7. Code Quickly. Learn C# Quickly: A Complete Beginner's Guide to Learning C#, Even If You're New to Programming. 2016, 150 p.
8. Stephen Cleary. Concurrency in C# Cookbook: Asynchronous, Parallel, and Multithreaded Programming. 2018, 164 p.
9. Jon Skeet. C# in Depth. 2016, 354 p.
10. C.P.A. Inc, CyberPunk Architects. The Programmers Code: A Deep Dive Into Mastering Computer Programming Including Python, C, C++, C#, Html Coding, Raspberry Pi3, And Black Hat Hacking. 2020, 398 p.
11. Ian Griffiths. Programming C# 8.0: Build Cloud, Web, and Desktop Applications. 2019, 234 p.
12. Huw Collingbourne. The Little Book Of C# Programming: Learn To Program C-Sharp For Beginners. 2019, 283 p.
13. Joyce Farrell. Microsoft Visual C#: An Introduction to Object-Oriented Programming (MindTap Course List). 2017, 231 p.
14. Joseph Albahari. C# 8.0 in a Nutshell: The Definitive Reference. 2020, 243 p.

15. Ockert J. du Preez, Sunny Sharma. Visual Studio 2019 In Depth: Discover and make use of the powerful features of the Visual Studio 2019 IDE to develop better and faster mobile, web, and desktop applications. 2019, 187 p.

16. Benjamin Smith. C#: A Comprehensive Beginner's Guide to Learn about the Realms of C# from A-Z. 2020, 343 p.

17. Jon Skeet. C# in Depth. 2008, 304 p.

18. Gabriel Baptista, Francesco Abbruzzese. Hands-On Software Architecture with C# 8 and .NET Core 3: Architecting software solutions using microservices, DevOps, and design patterns for Azure Cloud. 2019, 243 p.

19. Sean Burns. Hands-On Network Programming with C# and .NET Core: Build robust network applications with C# and .NET Core. 2019, 341 p.

20. John Paul Mueller, Bill Sempf, Chuck Sphar. C# 7.0 All-in-One For Dummies. 2017, 176 p.

21. Joydip Kanjilal. Mastering C# 8.0: Master C# skills with plentiful code examples (English Edition). 2019, 389 p.

22. Jeffrey Richter. CLR via C# (Developer Reference). 2012, 273 p.

23. Joyce Farrell. Microsoft Visual C#: An Introduction to Object-Oriented Programming (MindTap Course List). 2017, 134 p.

24. Gary Mclean. Adaptive Code via C#: Agile coding with design patterns and SOLID principles (Developer Reference). 2014, 234 p.

25. Benjamin Perkins, Jacob Vibe Hammer, Jon D. Reid. Beginning C# 7 Programming with Visual Studio 2017. 2018, 122 p.

26. DENNIS SHARP. C# Advanced Topics, Features and Programming Techniques: Take Your C# Skills and Expertise to the Next Level (Advanced Level). 2019, 130 p.

27. Life -Style Academy. C#: C# CRASH COURSE - Beginner's Course To Learn The Basics Of C# Programming Language: (c#, c programming, c, java, python, angularjs, c++ programming). 2016, 234 p.

28. Wally Parsons. C# Programming: Ultimate Guide For Advanced Users To Learn C# Programming (3 books in 1). 2019, 287 p.

29. Aristides S. Bouras. C# for Tweens and Teens (Black & White Edition): Learn Computational and Algorithmic Thinking. 2017, 141 p.

30. Capers Jones. Software Methodologies: A Quantitative Guide. 2017, 197 p.

31. David Harned. Hands-On Agile Software Development with JIRA: Design and manage software projects using the Agile methodology. 2018, 203 p.

32. David Kung. Object-Oriented Software Engineering: An Agile Unified Methodology. 2013, 201 p.

33. Michael E. Bays. Software Release Methodology. 1999, 342 p.

34. A step-by-step guide to train your own GPT-2 model for text generation in your choice of language from scratch. https://towardsdatascience.com/train-gpt-2-in-your-own-language-fc6ad4d60171

35. The Pros and Cons of Chatbots. https://www.chatdesk.com/blog/pros-and-cons-of-chatbots

36. Advantages and disadvantages of bots: everything you need to know. https://www.aivo.co/blog/advantages-and-disadvantages-of-chatbots

37. How to Create a Chatbot in 2022: an Ultimate Guide. https://www.cleveroad.com/blog/how-to-make-a-chatbot

38. Deep Learning Chatbot: Everything You Need to Know. https://shanebarker.com/blog/deep-learning-chatbot/

39. How To Build Your Own Chatbot Using Deep Learning. https://towardsdatascience.com/how-to-build-your-own-chatbot-using-deep-learning-bb41f970e281

40. Create smart chatbots for multi-channel messaging on our revolutionary platform. https://snatchbot.me/

41. Hector an Conversational AI Chatbot. https://ochatbot.com/hector/

42. BotSharp - The Open Source AI Chatbot Platform Builder in 100% C#. https://www.findbestopensource.com/product/scisharp-botsharp

43. 21 Best AI Chatbot Platforms. https://www.giosg.com/blog/ai-chatbot

44. Free AI-Powered Website Chat. https://smith.ai/chat/free-ai-chatbot

45. Chatbot or human? Either way, what matters for customer trust is "perceived humanness". https://www.eurekalert.org/news-releases/939360

46. Tay A.I. | The People's Chatbot. https://youtu.be/HsLup7yy-6I

47. The Open Source AI Chatbot Platform Builder. https://github.com/SciSharp/BotSharp

48. Building Chatbots with Markov Chains and Neural Networks. https://www.codingame.com/blog/markov-chain-automaton2000/

49. 2022 List of Fun Chatbots. https://www.ometrics.com/blog/list-of-fun-chatbots/

50. The Ultimate Guide to Chatbots. https://www.drift.com/learn/chatbot/

51. Chatbots | GPT-3 Demo. https://gpt3demo.com/category/chatbots

52. What Are GPT-3 Chatbots? https://www.simplr.ai/glossary/gpt-3-chatbot

53. How To Implement GPT-3 In Your Chatbot. https://cobusgreyling.medium.com/implement-gpt-3-in-your-chatbot-52a29b0a066d

54. Creating Chatbot using 100% .NET Core with Machine Learning. https://chatbotslife.com/creating-chatbot-using-100-net-core-with-machine-learning-f0dd2c34e891

55. How do I talk to GPT? https://lifearchitect.ai/how-do-i-talk-to-gpt/

56. Building Machine Learning Chatbots: Choose the Right Platform and Applications. https://neptune.ai/blog/building-machine-learning-chatbots-platforms-and-applications

57. The Definitive Guide to Using a Deep Learning or Machine Learning Chatbot for Your Business. https://www.ringcentral.co.uk/gb/en/blog/the-definitive-guide-to-using-a-deep-learning-or-machine-learning-chatbot-for-your-business/

ДОДАТОК А

ЛІСТИНГ ПРОГРАМИ

Program.cs

```
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;

using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Threading.Tasks;
using Newtonsoft.Json;
using System.Dynamic;
using Newtonsoft.Json.Converters;
using VkNet;
using System.Threading;
using VkNet.Model;
using System.Collections.ObjectModel;
using VkNet.Model.Attachments;
using VkNet.Model.RequestParams;
using System.Net.Http;
using System.Text;
using System.Net;

using System.Threading;
using System.Diagnostics;
using System.Runtime.InteropServices;

using System.Globalization;

using Microsoft.Extensions.DependencyInjection;
using VkNet.Enums.SafetyEnums;

namespace NeuralBores
{
   public class Program
   {
      static List<BoresEngineLowIQ> _BoresEngines = new();
```

```
static BoresEngineLowIQ _privateChatEngine;

static bool privateMode = true;

static public VkApi VkApi => BoresEngineLowIQ.VkApi;

public static async Task Main(string[] args)
{
    if (privateMode)
        Console.WriteLine($"PrivateMode ENABLED");

    //Thread.Sleep(20000);
    Console.InputEncoding = Encoding.Unicode;
    Console.OutputEncoding = System.Text.Encoding.UTF8;
    CultureInfo.DefaultThreadCurrentCulture = new CultureInfo("ru-RU");
    InfaHandler.LoadHachs();
    CancellationTokenSource cts = new CancellationTokenSource();
    CancellationToken token = cts.Token;

    string access = "1";
    ulong id = 1;
    StartUpTheRotors();
    VkApi.Authorize(new ApiAuthParams
    {
        AccessToken = access,
    });

    var longPollServer = VkApi.Groups.GetLongPollServer(id);
    var longPollHistoryParams = new BotsLongPollHistoryParams()
    {
        Server = longPollServer.Server,
        Ts = longPollServer.Ts,
        Key = longPollServer.Key,
        Wait = 25
    };
    BotsLongPollHistoryResponse poll;

    while (true)
    {
        try
        {
            poll = VkApi.Groups.GetBotsLongPollHistory(longPollHistoryParams);
        }
        catch (Exception ex)
```

```
            {
                Console.WriteLine($"Longpoll exception (key expired?): {ex}");

            start:
                try
                {
                    longPollServer = VkApi.Groups.GetLongPollServer(id);
                    longPollHistoryParams = new BotsLongPollHistoryParams()
                    {
                        Server = longPollServer.Server,
                        Ts = longPollServer.Ts,
                        Key = longPollServer.Key,
                        Wait = 25
                    };
                    poll =
VkApi.Groups.GetBotsLongPollHistory(longPollHistoryParams);
                }
                catch (Exception ex2)
                {
                    Console.WriteLine($"Longpoll exc 2: {ex2}");
                    Thread.Sleep(5000);
                    goto start;
                }
            }
            longPollHistoryParams.Ts = poll.Ts;
            if (poll?.Updates == null) continue;
            foreach (var a in poll.Updates)
            {
                if (a.Type == GroupUpdateType.MessageNew)
                {
                    {
                        MessageNewHandler(a.MessageNew.Message);
                    }
                }
            }
        }

        Console.WriteLine("If you see this, it's already too late");

    }
    public static string SaveAll()
    {
        _privateChatEngine.SaveDialog();
        foreach (var engine in _BoresEngines)
```

```
            engine.SaveDialog();
        return $"{_BoresEngines.Count + 1} Saved";
    }
    static void DeteleSettingsFiles() //for testing only
    {
        foreach (var dir in Directory.EnumerateDirectories("lowIQ"))
            if (File.Exists(Path.Combine(dir, "settings.json")))
                File.Delete(Path.Combine(dir, "settings.json"));
    }

    static void StartUpTheRotors()
    {
        Console.WriteLine("Starting engines...");
        _privateChatEngine = new BoresEngineLowIQ("private", false);


        Directory.GetDirectories(BoresEngineLowIQ.ModelName)
            .Select(dir => Path.GetFileName(dir)) //returns the correct directory name
because you
            .Where(dir => int.TryParse(dir, out _) == true)
            .ToList()
            .ForEach(chatId => _BoresEngines.Add(new BoresEngineLowIQ(chatId,
true)));

        Console.WriteLine($"{_BoresEngines.Count} engines and the private one
started!");
    }

    private static BoresEngineLowIQ GetEngineByChatId(string id)
    {
        if (!IsGroupChat(id))
        {
            return _privateChatEngine;
        }
        else
        {
            var engine = _BoresEngines.Where(eng => eng.ChatIdInternal ==
id).FirstOrDefault();
            if (engine is null)
            {
                engine = new BoresEngineLowIQ(id, true);
                _BoresEngines.Add(engine);
            }
            return engine;
```

```
        }
    }
    static bool IsGroupChat(string id) => id.StartsWith("200") && id.Length == 10;
    public static void MessageNewHandler(Message msg)
    {
        if (privateMode)
        {
            if (msg.PeerId != 1)
                return;
        }

        Console.WriteLine($"{msg.PeerId}: {msg.Text}");

        try
        {
            var engineById = GetEngineByChatId(msg.PeerId.ToString());
            engineById.ProcessNewMessage(msg);
        }
        catch (Exception ex)
        {
            File.AppendAllText("log.txt", $"{DateTime.Now} {msg.PeerId}" +
                $" {msg.Text}: {ex.InnerException.ToString()}");
        }
    }
    public static async Task<ReadOnlyCollection<Photo>>
UploadImageToMessage(VkApi vkApi, long peerId, string imagePath)
    {
        var uploadServerInfo = vkApi.Photo.GetMessagesUploadServer(peerId);
        MultipartFormDataContent form = new MultipartFormDataContent();
        using var wc = new WebClient();
        var response =
Encoding.ASCII.GetString(wc.UploadFile(uploadServerInfo.UploadUrl,
imagePath));
        return vkApi.Photo.SaveMessagesPhoto(response);
    }

    public static Task<ReadOnlyCollection<Photo>> UploadImage(long? peerId,
string path)
    {
        return UploadImageToMessage(VkApi, (long)peerId, path);
    }

    public static string[] CommandLineToArgs(string commandLine)
    {
```

```
            var parmChars = commandLine.ToCharArray();
            var inSingleQuote = false;
            var inDoubleQuote = false;
            for (var index = 0; index < parmChars.Length; index++)
            {
               if (parmChars[index] == "" && !inSingleQuote)
               {
                  inDoubleQuote = !inDoubleQuote;
                  parmChars[index] = '\n';
               }
               if (parmChars[index] == '\' && !inDoubleQuote)
               {
                  inSingleQuote = !inSingleQuote;
                  parmChars[index] = '\n';
               }
               if (!inSingleQuote && !inDoubleQuote && parmChars[index] == ' ')
                  parmChars[index] = '\n';
            }
            return (new string(parmChars)).Split(new[] { '\n' },
StringSplitOptions.RemoveEmptyEntries);
         }
      }
}
```

ImgBlock.cs

```
using System.Drawing;
using System.Drawing.Drawing2D;

namespace NeuralBores
{
   public class ImgBlock
   {
      public Rectangle Rect;
      public bool IsCircular = false;

      public void Draw(Bitmap img, Graphics g)
      {
         if (IsCircular)
            img = CropToCircle(img, Color.Transparent);
         g.DrawImage(img, Rect);
      }

      public ImgBlock(Rectangle rect, bool isCircular = false)
      {
```

```
            Rect = rect;
            IsCircular = isCircular;
        }

        public Bitmap CropToCircle(Bitmap srcImage, Color backGround)
        {
            srcImage = ResizeBitmap(srcImage, Rect.Width, Rect.Height);
            Bitmap dstImage = new Bitmap(srcImage.Width, srcImage.Height,
srcImage.PixelFormat);//, srcImage.PixelFormat);
            Graphics g = Graphics.FromImage(dstImage);
            using (Brush br = new SolidBrush(backGround))
            {
                g.FillRectangle(br, 0, 0, dstImage.Width, dstImage.Height);
            }
            float radius = Rect.Width/2;
            //PointF center = new Point(60, 60);
            GraphicsPath path = new GraphicsPath();
            path.AddEllipse(0, 0, radius * 2, radius * 2);
            g.SetClip(path);
            g.SmoothingMode = SmoothingMode.HighSpeed;
            g.InterpolationMode = InterpolationMode.Low;
            g.DrawImage(srcImage, 0, 0);

            return dstImage;
        }

        public Bitmap ResizeBitmap(Bitmap bmp, int width, int height)
        {
            Bitmap result = new Bitmap(width, height);
            using (Graphics g = Graphics.FromImage(result))
            {
                g.DrawImage(bmp, 0, 0, width, height);
            }

            return result;
        }
    }
}
```

MsgHandler.cs

```
using System;
using System.Net.Http;
using System.Text;
using System.Threading.Tasks;
```

```
using System.Collections.Generic;
using Newtonsoft.Json; // Install Newtonsoft.Json with NuGet
using System.Dynamic;
using Newtonsoft.Json.Converters;
using System.Text.Json;

namespace NeuralBores
{
   public class MsgHandler
   {
      static HttpClient httpClient = new HttpClient();

      private static readonly string subscriptionKey = "key";
      private static readonly string endpoint =
"https://api.cognitive.microsofttranslator.com/";

      private static List<string> _langs = new List<string>()
      {
         "af",
"sq",
"am",
"ar",
"hy",
"as",
"az",
"bn",
"ba",
"bs",
"bg",
"yue",
"ca",
"lzh",
"zh-Hans",
"zh-Hant",
"hr",
"cs",
"da",
"prs",
"dv",
"nl",
"en",
"et",
"fj",
"fil",
```

"fi",
"fr",
"fr-ca",
"ka",
"de",
"el",
"gu",
"ht",
"he",
"hi",
"mww",
"hu",
"is",
"id",
"iu",
"ga",
"it",
"ja",
"kn",
"kk",
"km",
"ko",
"ku",
"ky",
"lo",
"lv",
"lt",
"mk",
"mg",
"ms",
"ml",
"mt",
"mi",
"mr",
"mn-Cyrl",
"mn-Mong",
"my",
"ne",
"nb",
"or",
"ps",
"fa",
"pl",
"pt",

```
"pt-pt",
"pa",
"otq",
"ro",
"ru",
"sm",
"sr-Cyrl",
"sk",
"sl",
"es",
"sw",
"sv",
"ty",
"ta",
"tt",
"te",
"th",
"bo",
"ti",
"to",
"tr",
"tk",
"uk",
"ur",
"ug",
"uz",
"vi",
"cy",
        };

        // Add your location, also known as region. The default is global.
        // This is required if using a Cognitive Services resource.
        private static readonly string location = "global";

        public static int LevenshteinDistance(string s, string t, bool toLower)
        {
            if (toLower)
            {
                s = s.ToLower();
                t = t.ToLower();
            }
            int n = s.Length;
            int m = t.Length;
            int[,] d = new int[n + 1, m + 1];
```

```
    // Step 1
    if (n == 0)
    {
        return m;
    }

    if (m == 0)
    {
        return n;
    }

    // Step 2
    for (int i = 0; i <= n; d[i, 0] = i++)
    {
    }

    for (int j = 0; j <= m; d[0, j] = j++)
    {
    }

    // Step 3
    for (int i = 1; i <= n; i++)
    {
        //Step 4
        for (int j = 1; j <= m; j++)
        {
            // Step 5
            int cost = (t[j - 1] == s[i - 1]) ? 0 : 1;

            // Step 6
            d[i, j] = Math.Min(
                Math.Min(d[i - 1, j] + 1, d[i, j - 1] + 1),
                d[i - 1, j - 1] + cost);
        }
    }
    // Step 7
    return d[n, m];
}

public static async Task<string> Translate(string text, string langShort = "uk")
{
    if (string.IsNullOrWhiteSpace(text))
        return "Text empty";
```

```
        if (!LangExists(langShort))
            return $"Language {langShort} is incorrect or unsupported";
        // Output languages are defined as parameters, input language detected.
        string route = $"/translate?api-version=3.0&to={langShort}";
        object[] body = new object[] { new { Text = text } };
        var requestBody = JsonConvert.SerializeObject(body);

        using var request = new HttpRequestMessage();
        // Build the request.
        request.Method = HttpMethod.Post;
        request.RequestUri = new Uri(endpoint + route);
        request.Content = new StringContent(requestBody, Encoding.UTF8,
"application/json");
        request.Headers.Add("Ocp-Apim-Subscription-Key", subscriptionKey);
        request.Headers.Add("Ocp-Apim-Subscription-Region", location);

        // Send the request and get response.
        HttpResponseMessage response = await
httpClient.SendAsync(request).ConfigureAwait(false);
        // Read response as a string.
        if (!response.IsSuccessStatusCode)
            return await response.Content.ReadAsStringAsync();

        string reply;
        try
        {
            string json = await response.Content.ReadAsStringAsync();
            using var translation = JsonDocument.Parse(json);
            reply =
translation.RootElement[0].GetProperty("translations")[0].GetProperty("text").ToStri
ng();
        }
        catch { reply = "Error parsing response"; }
        return reply;
    }

    static bool LangExists(string lang) => _langs.Contains(lang);
}

public class Transl
{

}
```

```
}
```

Startup.cs

```
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Http;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net.Http;
using System.Threading.Tasks;

namespace NeuralBores
{
   public class Startup
   {
      public static IHttpClientFactory fact;

      // This method gets called by the runtime. Use this method to add services to the
container.
      // For more information on how to configure your application, visit
https://go.microsoft.com/fwlink/?LinkID=398940
      public static void ConfigureServices(IServiceCollection services)
      {
         fact = (IHttpClientFactory)services.AddHttpClient();
      }

      // This method gets called by the runtime. Use this method to configure the
HTTP request pipeline.
      public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
      {
         if (env.IsDevelopment())
         {
            app.UseDeveloperExceptionPage();
         }

         app.UseRouting();

         app.UseEndpoints(endpoints =>
         {
            endpoints.MapGet("/", async context =>
            {
```

```
                await context.Response.WriteAsync("Hello World!");
            });
        });
    }
  }
}
```

MemeManager.cs

```
using System;
using System.Collections.Generic;
using System.Drawing;
//using System.Drawing.Drawing2D;
using System.Globalization;
using System.IO;
using System.Linq;
using System.Threading;
using System.Threading.Tasks;
using Microsoft.AspNetCore;
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.Logging;

namespace NeuralBores
{
  public class MemeManager
  {
    public static Random RNG => BoresEngine.RNG;

    static Font _Impact48 = new Font("Impact", 48);
    static Font _Impact60 = new Font("Impact", 60);
    static Font _Impact28 = new Font("Impact", 25);
    static Font _Arial16 = new Font("Arial", 16);
    static Font _Arial18 = new Font("Arial", 18);
    static Font _Arial22 = new Font("Arial", 22);
    static Font _Arial26 = new Font("Arial", 26);
    static Font _Arial28 = new Font("Arial", 28);
    static Font _Arial30 = new Font("Arial", 30);
    static Font _Arial40 = new Font("Arial", 40);
    static Font _Times26 = new Font("Times New Roman", 26);

    static List<MemeTemplate> _Templates = new List<MemeTemplate>();

    static MemeTemplate GetRandTemp(int savedImgs)
    {
```

```
            var properTemplates = _Templates.Where(t =>
t.CheckImgCount(savedImgs)).ToArray();

            return properTemplates[RNG.Next(properTemplates.Length)];
        }
        /// <summary>
        /// Returns a MT that starts with name. Considers saved imgs count. Returns
random MT if unsuccessful.
        /// </summary>
        /// <param name="name"></param>
        /// <param name="imgsSaved"></param>
        /// <returns></returns>
        static MemeTemplate? GetProperByName(string name, int imgsSaved)
        {
            return _Templates.Where(mt =>
            mt.Name.ToLower().StartsWith(name.ToLower()) &&
            mt.CheckImgCount(imgsSaved))
                .FirstOrDefault() ?? GetRandTemp(imgsSaved);
        }
        /// <summary>
        /// Generates a random meme.
        /// </summary>
        /// <returns></returns>
        public static Bitmap Generate(BoresEngineLowIQ engine, FileHandler fh, bool
isRecursive = false, params string[] texts)
        {
            return Generate(engine, fh, memeTemp: null, isRecursive, texts);
        }
        public static Bitmap Generate(BoresEngineLowIQ engine, FileHandler fh, string
memeTemp, bool isRecursive = false, params string[] texts)
        {
            MemeTemplate requstedMT;
            if (string.IsNullOrWhiteSpace(memeTemp))
                requstedMT = GetRandTemp(fh.SavedImgsCount);
            else
                requstedMT = GetProperByName(memeTemp, fh.SavedImgsCount);
            Console.WriteLine($"Generating meme using temp {requstedMT.Name}");

            string[] suppliedTexts = new string[requstedMT.TextBlocks.Count];
            string lastText = "";
            for (int i = 0, indexTexts = 0; i < requstedMT.TextBlocks.Count; i++,
indexTexts++)
            {
                //string prevText = i > 0 ? texts[i - 1] : "";
```

```
            lastText = GetText(engine, isRecursive, texts, ref indexTexts, lastText);
            suppliedTexts[i] = lastText;
        }

        Bitmap[] randDistinctSavedImgs =
fh.GetRandomDistinctImgs(requstedMT.ImgBlocks.Count)
            .Select(imgPath => FromFileWithoutLockingIt(imgPath)).ToArray();
        //Bitmap[] randSavedImgs = new Bitmap[requstedMT.ImgBlocks.Count];
        //for (int i = 0; i < randSavedImgs.Length; i++)
        //{
        //    randSavedImgs[i] = FromFileWithoutLockingIt(fh.RandomSavedImg);
        //}

        return requstedMT.Generate(randDistinctSavedImgs, suppliedTexts);
    }
    static string GetText(BoresEngineLowIQ engine, bool isRecursive,
        string[] suppliedTexts, ref int index, string prevText = "")
    {
        if (isRecursive && suppliedTexts != null && suppliedTexts.Length > 0 &&
index >= suppliedTexts.Length)
        {
            index = 0;
            goto CheckForSpecial;
        }
        if (index >= suppliedTexts.Length)
            return engine.GetAdvancedResponse(prevText);

        CheckForSpecial:
        if (suppliedTexts[index] == "_")
            return engine.GetAdvancedResponse(prevText);
        if (suppliedTexts[index] == "_h_")
            return InfaHandler.GenerateHach(false);
        return suppliedTexts[index];
    }
    public static Bitmap GenerameDemo(BoresEngineLowIQ engine,
        string imgPath, bool useRecursiveText = false, int cycles = 1, params string[]
texts)
    {
        Bitmap tempImg = FromFileWithoutLockingIt(imgPath);
        //int multiplier = useRecursiveText ? 0 : 2;

        for (int i = 0, indexTexts = 0; i < cycles; i++, indexTexts++)
        {
            string upper;
```

```
            string lower;

            //int indexUpper = i * multiplier;
            //int indexLower = 1 + i * multiplier;

            upper = GetText(engine, useRecursiveText, texts, ref indexTexts);
            indexTexts++;
            lower = GetText(engine, useRecursiveText, texts, ref indexTexts);

            tempImg = Demo.Generate(upper, lower, tempImg);
         }
         return tempImg;
      }
      public static void Save(Bitmap bmp)
      {

      }
      public static Bitmap FromFileWithoutLockingIt(string path) //Should be cleaned
by GC
      {
         var bytes = File.ReadAllBytes(path);
         var ms = new MemoryStream(bytes);
         var img = (Bitmap)Image.FromStream(ms);
         return img;
      }
      public static string GetTemplateList()
      {

         foreach (var mt in _Templates)
            tempList += $"{mt.Name} (txt: {mt.TextBlocks.Count}, img:
{mt.ImgBlocks.Count})\n";
         return tempList;
      }
   }
}
```

SettingsAdditional.cs

```
   public class SettingInt : Setting
   {
      [JsonProperty]
      private int _value;
      [JsonProperty]
      private int _min;
      [JsonProperty]
```

```
        private int _max = 100;
        public int Value => _value;

        public override bool Set(string newValueString, UserPrivileges privileges)
        {
            if (CheckPriviledge(privileges))
            {
                if (int.TryParse(newValueString, out int newValueInt))
                {
                    if (IsWithinRange(newValueInt))
                    {
                        _value = newValueInt;
                        return true;
                    }
                }
            }
            return false;
        }

        public SettingInt(string name, int value, int min = 0, int max = 100, bool
hasToBeGod = true)
        {
            Name = name;
            _value = value;
            _min = min;
            _max = max;
            SenderHasToBeCreator = hasToBeGod;
        }
        private bool IsWithinRange(int newValue)
        {
            return (newValue >= _min && newValue <= _max);
        }
        public override string ToString()
        {
            string isCreat = SenderHasToBeCreator ? "God" : "Admin";
            return $"{Name} ({isCreat}) ({NameShortcut}) {GetRange()}: {Value}";
        }
        public override string GetRange()
        {
            return $"[{_min}, {_max}]";
        }
    }
    public class SettingBool : Setting
    {
```

```
        [JsonProperty]
        private bool _value;
        public bool Value => _value;
        public SettingBool(string name, bool value = false, bool hasToBeGod = true)
        {
            Name = name;
            _value = value;
            SenderHasToBeCreator = hasToBeGod;
        }
        public override bool Set(string newValueString, UserPrivileges privileges)
        {
            if (CheckPriviledge(privileges))
            {
                if (bool.TryParse(newValueString, out bool newValue))
                {
                    _value = newValue;
                    return true;
                }
            }
            return false;
        }
        public override string ToString()
        {
            string isCreat = SenderHasToBeCreator ? "God" : "Admin";
            return $"{Name} ({isCreat}) ({NameShortcut}): {Value}";
        }
        public override string GetRange()
        {
            return "[true/false]";
        }
    }

    public class Setting
    {
        public string Name;

        public string NameShortcut => string.Concat(Name.Where(c => c >= 'A' && c
<= 'Z')).ToLower();

        public bool SenderHasToBeCreator;

        protected bool CheckPriviledge(UserPrivileges priv)
        {
            if (priv == UserPrivileges.God)
```

```
            return true;
        if (priv == UserPrivileges.Admin && !SenderHasToBeCreator)
            return true;
        return false;
    }

    public virtual bool Set(string newValueStrin, UserPrivileges privileges) { throw new NotImplementedException(); }

    public virtual string GetRange() { throw new NotImplementedException(); }
}
```

ДОДАТОК Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК С

СПИСОК ФАЙЛІВ НА ДИСКУ

Ім'я файлу	**Опис**
Пояснювальні документи	
DiplomaSkrypkaMykyta.doc	Пояснювальна записка до проекту. Документ Word.
DiplomaSkrypkaMykyta.pdf	Пояснювальна записка до проекту у форматі PDF.
Program	
Program.zip	Архів. Містить програмні коди.
Presentation	
PresentationSkrypkaMykyta.ppt	Презентація проекту.