МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В.Н. Каразіна

Факультет: **ННІ Каразінський банківський інститут**
Кафедра: **Інформаційних технологій та математичного моделювання**
Спеціальність: **122 Комп'ютерні науки**
Освітня програма: **Комп'ютерні науки**

Група: АК-21М **денна форма навчання**

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

на тему:

ТЕЛЕГРАМ-БОТ ДЛЯ НАДАННЯ ДОВІДКОВИХ МАТЕРІАЛІВ З ПЕРШОЇ МЕДИЧНОЇ ДОПОМОГИ НА ОСНОВІ МОДЕЛЕЙ ШТУЧНОГО ІНТЕЛЕКТУ

ЗА НАКАЗОМ № 4601-5_3262 ВІД 15 вересня 2025 РОКУ

здобувача вищої освіти **Левченка Андрія Олеговича**

Робота допущена до захисту в ЕК
протокол кафедри ІТММ №___від ___2025 р.
Завідувач кафедри ІТММ
к. п. н. доцент
_____Н. І. Стяглик
Науковий керівник
к.ф.-м.н., доцент
_____Г. В. Макарова

м. Харків 2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Харківський національний університет імені В. Н. Каразіна

Факультет навчально-науковий інститут "Каразінський банківський інститут"
Кафедра інформаційних технологій та математичного моделювання
Рівень вищої освіти другий (магістерський)
Спеціальність 122 Комп'ютерні науки
Освітня програма Комп'ютерні науки та інформаційні технології в бізнесі

ЗАТВЕРДЖУЮ

Завідувач кафедри

Н. І. Стяглик
Підпис ініціали, прізвище

"15" вересня 2025 року

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ (ПРОЄКТ)

Левченка Андрія Олеговича
(прізвище, ім'я, по батькові студента)

1. Тема проекту (роботи): Телеграм-бот для надання довідкових матеріалів з першої медичної допомоги на основі моделей штучного інтелекту.

керівник роботи к.ф.-м.н., доцент Г. В. Макарова
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від "15" вересня 2025 року № 4601-5_3262

2. Строк подання студентом роботи 24 листопада 2025 року

3. Перелік питань, які потрібно розробити:

У розділі 1: проаналізувати стан проблеми інформування населення про першу медичну допомогу, розглянути технології створення медичних чат-ботів та існуючі програмні рішення.
У розділі 2: дослідити методи машинного навчання для класифікації запитів, обґрунтувати вибір методу FastTree та визначити інструментальні засоби реалізації системи.
У розділі 3: спроєктувати архітектуру телеграм-бота, створити базу даних, інтегрувати модель машинного навчання, провести тестування й аналіз результатів роботи системи.

4. План роботи

№ з/п	Назви етапів роботи
1	Вибір здобувачем теми кваліфікаційної магістерської роботи
2	Затвердження плану і завдання кваліфікаційної магістерської роботи
3	Здача кваліфікаційної магістерської роботи керівнику
4	Підпис кваліфікаційної магістерської роботи керівника
5	Підпис кваліфікаційної магістерської роботи у нормоконтролера
6	Допуск завідувачем кафедри до захисту кваліфікаційної магістерської роботи
7	Захист кваліфікаційної магістерської роботи

5. Дата видачі завдання _____

Студент _____ А. О. Левченко
підпис ініціали, прізвище

Керівник роботи _____ Г. В. Макарова
підпис ініціали, прізвище

РЕФЕРАТ

НА КВАЛІФІКАЦІЙНУ МАГІСТЕРСЬКУ РОБОТУ

«ТЕЛЕГРАМ-БОТ ДЛЯ НАДАННЯ ДОВІДКОВИХ МАТЕРІАЛІВ З ПЕРШОЇ МЕДИЧНОЇ ДОПОМОГИ НА ОСНОВІ МОДЕЛЕЙ ШТУЧНОГО ІНТЕЛЕКТУ»

Левченка Андрія Олеговича

Кваліфікаційна магістерська робота містить: 117 сторінок, 10 таблиць, 34 рисунки, 1 формулу, список літератури з 29 найменувань.

Об'єктом дослідження є процес надання населенню інформації щодо першої медичної допомоги з використанням цифрових технологій.

Предметом дослідження є характерні особливості застосування моделей штучного інтелекту в телеграм-ботах для надання довідкових матеріалів з першої медичної допомоги.

Метою дослідження є розроблення інтелектуальної системи у вигляді телеграм-бота, здатного надавати користувачам довідкові матеріали з першої медичної допомоги на основі моделей штучного інтелекту.

Завданнями кваліфікаційної магістерської роботи є:

– у першому розділі дослідити сучасний стан проблеми інформування населення щодо надання першої медичної допомоги, проаналізувати технології створення медичних чат-ботів і розглянути існуючі програмні рішення у цій сфері;

– у другому розділі проаналізувати методи машинного навчання для класифікації користувацьких запитів, обґрунтувати вибір методу швидкого дерева та визначити інструментальні засоби реалізації системи;

– у третьому розділі спроєктувати архітектуру телеграм-бота, побудувати базу даних, розробити і інтегрувати модель машинного навчання, провести експериментальне тестування та аналіз отриманих результатів.

Актуальність дослідження зумовлена потребою у створенні доступних інтелектуальних засобів для надання довідкової медичної інформації населенню з використанням технологій штучного інтелекту.

За результатами дослідження розроблено функціональний Telegram-бот, що реалізує класифікацію користувацьких запитів на основі навченої моделі, забезпечуючи автоматичне надання довідкових матеріалів з першої допомоги.

Практична новизна полягає у створенні інтегрованої системи, яка поєднує засоби машинного навчання та телекомунікаційні сервіси, дозволяючи підвищити доступність медичних знань.

Отримані результати можуть бути використані для розробки аналогічних інтелектуальних ботів у сфері охорони здоров'я, освіти та соціальної підтримки.

КЛЮЧОВІ СЛОВА: ШТУЧНИЙ ІНТЕЛЕКТ, TELEGRAM-БОТ, МАШИННЕ НАВЧАННЯ, FASTTREE, ПЕРША МЕДИЧНА ДОПОМОГА, ML.NET.

ABSTRACT

AT QUALIFICATION MAGISTER WORK

«TELEGRAM BOT FOR PROVIDING REFERENCE MATERIALS ON FIRST AID BASED ON ARTIFICIAL INTELLIGENCE MODELS»

by Andriy Olegovich Levchenko

The master's qualification thesis consists of 117 pages, 10 tables, 34 figures, 1 formula, and a reference list of 29 sources.

The object of research is the process of providing the population with information on first aid using digital technologies.

The subject of research is the specific features of applying artificial intelligence models in Telegram bots for delivering reference materials on first aid.

The purpose of the study is to develop an intelligent system in the form of a Telegram bot capable of providing users with first aid reference materials based on artificial intelligence models.

The tasks of the master's qualification thesis are as follows:

– in the first chapter, to examine the current state of the problem of public awareness regarding first aid provision, to analyze technologies for developing medical chatbots, and to review existing software solutions in this field;

– in the second chapter, to analyze machine learning methods for classifying user queries, to justify the choice of the FastTree method, and to determine the instrumental tools for system implementation;

– in the third chapter, to design the architecture of the Telegram bot, build the database, develop and integrate the machine learning model, and conduct experimental testing and analysis of the obtained results.

The relevance of the study lies in the need to create accessible intelligent tools for providing the public with medical reference information using artificial intelligence technologies.

As a result of the research, a functional Telegram bot was developed, which implements classification of user queries based on a trained model, ensuring automated delivery of first aid reference materials.

The practical novelty consists in the creation of an integrated system combining machine learning techniques and telecommunication services, thereby enhancing the accessibility of medical knowledge.

The obtained results can be applied to the development of similar intelligent bots in the fields of healthcare, education, and social assistance.

KEY WORDS: ARTIFICIAL INTELLIGENCE, TELEGRAM BOT, MACHINE LEARNING, FASTTREE, FIRST AID, ML.NET.

ЗМІСТ

ВСТУП

Сучасні виклики у сфері охорони здоров’я висувають на перший план завдання забезпечення швидкого доступу населення до перевіреної інформації щодо першої медичної допомоги. Дослідження свідчать, що рівень базової підготовки громадян у більшості країн, включаючи Україну, залишається недостатнім, а обізнаність щодо алгоритмів надання допомоги часто обмежується фрагментарними знаннями або неперевіреними порадами з мережі. У кризових умовах, під час воєнних дій чи природних катастроф, відсутність таких знань може коштувати життя, що актуалізує проблему пошуку нових ефективних способів інформування.

Наукова спільнота протягом останніх років запропонувала низку рішень, зокрема мобільні застосунки та онлайн-курси, які покликані навчати основам першої допомоги. Проте ці інструменти мають низку обмежень: мобільні програми вимагають завантаження та встановлення, що не завжди зручно в екстрених ситуаціях; онлайн-курси більше орієнтовані на навчання, ніж на миттєву підтримку користувача. Деякі системи надають інтерактивні інструкції, але залишаються складними для людей без медичної освіти. Таким чином, існуючі рішення поки що не забезпечують поєднання доступності, простоти та інтелектуальної адаптивності.

У світовій практиці вже з’явилися приклади чат-ботів у сфері медицини, таких як Babylon Health у Великій Британії чи волонтерські проєкти FAST в Україні. Вони довели, що чат-боти можуть виступати ефективним інструментом у сфері охорони здоров’я, проте водночас показали і свої обмеження: обмежену точність алгоритмів, ризики конфіденційності та складність масштабування. З іншого боку, їхня популярність підтвердила попит на такі інструменти серед населення, що підкреслює потребу у вдосконаленні технологій.

Використання моделей штучного інтелекту відкриває нові можливості для подолання зазначених недоліків. На відміну від статичних програм,

системи з елементами машинного навчання здатні враховувати контекст запиту, адаптувати інструкції до конкретних умов і надавати персоналізовані рекомендації. Це особливо важливо для України, де поєднання воєнних ризиків і потреби у підвищенні медичної грамотності населення робить питання першої допомоги стратегічним. Швидкий доступ до актуальної інформації може суттєво підвищити рівень виживання постраждалих і знизити навантаження на медичну систему.

Окрім того, інтеграція подібних рішень у популярні месенджери забезпечує унікальну перевагу – відсутність потреби у встановленні спеціальних програм чи реєстрації. Це зменшує бар'єри для користувачів різного віку й рівня цифрової грамотності та робить технологію по-справжньому масовою. Простий і зрозумілий функціонал, підкріплений штучним інтелектом, дозволяє не лише інформувати, а й навчати користувачів, формуючи в суспільстві культуру відповідального ставлення до здоров'я та безпеки.

Метою дослідження є розроблення інтелектуальної системи у вигляді телеграм-бота, здатного надавати користувачам довідкові матеріали з першої медичної допомоги на основі моделей штучного інтелекту.

Для досягнення поставленої мети необхідно вирішити такі задачі:

– проаналізувати сучасний стан інформування населення щодо першої медичної допомоги та визначити ключові проблеми, що впливають на рівень обізнаності;

– дослідити технології побудови чат-ботів і методи штучного інтелекту, придатні для класифікації користувацьких запитів та формування персоналізованих рекомендацій;

– спроєктувати архітектуру програмного рішення з урахуванням бази даних і бізнес-процесів, необхідних для ефективної роботи інтелектуального телеграм-бота;

– реалізувати інтеграцію обраних моделей штучного інтелекту, провести експериментальні дослідження та здійснити аналіз отриманих результатів з метою підтвердження доцільності запропонованого підходу.

Об’єктом дослідження є процес надання населенню інформації щодо першої медичної допомоги з використанням цифрових технологій.

Предметом дослідження є характерні особливості застосування моделей штучного інтелекту в телеграм-ботах для надання довідкових матеріалів з першої медичної допомоги.

У першому розділі «Теоретичні основи використання штучного інтелекту для надання довідкових матеріалів з першої медичної допомоги» розглянуто актуальний стан проблеми інформування населення, проаналізовано сучасні медичні чат-боти та системи надання першої допомоги, визначено їхні переваги й обмеження.

У другому розділі «Аналіз методів та технологій розробки» досліджено алгоритми машинного навчання для класифікації текстових запитів і обґрунтовано вибір методу швидкого дерева. Визначено оптимальні інструментальні засоби реалізації – C#, Visual Studio 2022 та ML.NET.

У третьому розділі «Проєктування, розробка та дослідження інтелектуального телеграм-бота» сформульовано постановку задачі, спроєктовано базу даних і архітектуру системи, створено діаграми бізнес-процесів і реалізовано інтеграцію моделі машинного навчання. Проведено експериментальне тестування Telegram-бота, що підтвердило його коректну роботу, точність класифікації запитів і зручність користування.

Висновок узагальнює результати проведеного дослідження, відображає його наукову та прикладну цінність, а також окреслює перспективи подальшого вдосконалення системи й розширення можливостей штучного інтелекту в галузі надання першої медичної допомоги.

РОЗДІЛ 1.
ТЕОРЕТИЧНІ ОСНОВИ ВИКОРИСТАННЯ ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ НАДАННЯ ДОВІДКОВИХ МАТЕРІАЛІВ З ПЕРШОЇ МЕДИЧНОЇ ДОПОМОГИ

1.1. Сучасний стан проблеми інформування населення щодо першої медичної допомоги

Перша медична допомога є критично важливою навичкою, від якої залежить виживання постраждалих до прибуття професійної допомоги. Рівень знань і готовності до надання допомоги визначається багатьма чинниками – від соціально-демографічних характеристик і рівня освіти до доступності навчальних курсів та поширеності цифрових ресурсів. Особливу увагу дослідники приділяють тому, як суспільні кризи, пандемія чи військові дії актуалізують потребу у швидкому поширенні базових знань та формуванні впевненості людей у власних діях.

Дані європейських і американських організацій свідчать про тривожний стан підготовки до надання першої допомоги. За статистикою Американського Червоного Хреста, 55 % працівників не мають можливості пройти курси першої допомоги та серцево-легеневої реанімації (CPR) на робочому місці; водночас щороку у США відбувається близько 10 тис. зупинок серця на робочому місці, і лише 45 % постраждалих отримують раннє використання автоматичного зовнішнього дефібрилятора (AED) [1]. Ще тривожніше те, що понад 70 % випадків раптової зупинки серця трапляються вдома, а більшість людей зізнаються, що вони не навчені й бояться діяти. У багатьох штатах США та канадських провінціях прийнято закони, що зобов’язують школярів проходити курси CPR, однак охоплення дорослого населення залишається низьким.

Ситуація схожа у Західній Європі. Аналіз, проведений у рамках європейського проєкту EuReCa, показав, що частота надання сторонньої

серцево-легеневої реанімації при позалікарняній зупинці серця відрізняється від країни до країни: у Нідерландах вона сягає 66 %, у Швеції – 61%, тоді як у деяких регіонах показник не перевищує 20 % [2]. Різноманітні програми обов'язкового навчання в школах і на підприємствах дозволяють підвищувати ці показники, але досі значна частина населення не має базових навичок.

Для розуміння конкретних прогалин варто розглянути результати регіональних досліджень. У 2023 р. у Північній Греції було проведено опитування дорослого населення щодо надання допомоги при опіках. Результати засвідчили, що лише один із десяти опитаних знав оптимальний алгоритм дій при термічних опіках; більшість застосовувала зубну пасту, йогурт та інші народні засоби, які можуть погіршити стан постраждалого [3]. Автори підкреслюють, що жінки, люди з вищою освітою та особи, які мають дітей і пройшли курси першої допомоги, вірогідніше демонстрували правильні дії. Водночас рівень знань не залежав від віку, місця проживання чи професії. Традиційні «лікувальні» засоби, передані з покоління в покоління, залишаються поширеними навіть у країнах з високим рівнем медичної культури. Дослідження підкреслило необхідність модернізувати інформаційні кампанії, зокрема через соціальні мережі й інтерактивні онлайн-курси, щоб охопити вразливі групи населення.

Повномасштабна війна в Україні загострила потребу в базових медичних навичках для широких верств населення. Від початку вторгнення український Червоний Хрест проводить інтенсивне навчання громадян. Згідно з доповіддю ReliefWeb, станом на 1 вересня 2024 р. майже 270 тис. людей пройшли програми першої допомоги, що тривають 6, 12 чи 48 годин і передбачають 60 % часу для практичних навичок (серцево-легенева реанімація, зупинка кровотечі, лікування опіків) [4]. Тренінги проводяться щоденно навіть у віддалених містечках та під час обстрілів; окремі модулі адаптовані для людей з інвалідністю й дітей. Оскільки значна частина громадян сьогодні змушена працювати або проживати в умовах підвищеної

небезпеки, попит на навчання зростає, що стимулює використання мобільних платформ для розповсюдження інформації.

Окрему увагу привертає психологічна перша допомога. Під час війни добровольці й психотерапевти організували телефонні та цифрові служби для зниження стресу та панічних станів. Одне з досліджень повідомляє, що мобільний чат-бот і телефонна лінія психологічної допомоги, створені для українських користувачів, залучили понад 50 тис. осіб протягом перших трьох місяців, надаючи поради щодо самодопомоги та сприяючи зниженню симптомів тривоги [5]. Цей досвід демонструє не лише високу потребу в інформаційній підтримці, а й готовність населення користуватися цифровими інструментами.

Дослідження показують, що джерелами інформації найчастіше виступають соціальні мережі, відеоплатформи та неформальні поради. Однак частина цих матеріалів є неточною або застарілою, що призводить до хибних дій, особливо щодо опіків або травм.

Соціально-демографічні фактори впливають на рівень обізнаності. Опитування в Північній Греції встановило, що жінки та люди з вищою освітою частіше мають правильні знання, а сімейні обов’язки (наявність більше трьох дітей) збільшують мотивацію пройти навчання [3]. У США низький рівень підготовки часто пов’язують із нехваткою тренінгів на робочому місці та відсутністю вимог до навчання дорослих. В умовах України доступність тренінгів залежить від безпеки регіону: у прифронтових областях основним джерелом стають онлайн-курси та волонтерські телеграм-канали.

Високий рівень проникнення смартфонів відкриває можливості для інтерактивного навчання. Концепція «Перша допомога в цифровому світі», запропонована IFRC, рекомендує поєднувати відеоуроки, ігри та тести для дітей, дорослих та літніх людей. Цифрові ресурси дозволяють навчатися у зручний час, відстежувати прогрес та оновлювати знання. Автори підкреслюють, що онлайн-навчання повинно доповнюватися практичними

заняттями, оскільки навички, наприклад, накладання пов’язки чи проведення CPR, важко опанувати без фізичних тренувань [6].

Пандемія COVID-19 суттєво обмежила проведення очних тренінгів; частина курсів була скасована або перенесена в онлайн-формат. Це підштовхнуло організації до впровадження дистанційних тренінгів та мобільних додатків. Прикладом є віртуальний курс першої допомоги IFRC та національних товариств, який поєднує відеолекції з тестами й симуляціями та дозволяє отримати сертифікат онлайн.

Російсько-українська війна створила додаткові бар’єри: через обстріли та міграцію люди втрачають доступ до традиційних навчальних закладів. Водночас війна підвищила мотивацію громадян опанувати першу допомогу.

Таким чином, цифрові платформи (вебсайти, мобільні застосунки, телеграм-канали) стали основним каналом поширення знань. Психологічна перша допомога через чат-боти також набирає обертів: дослідження 2025 р. показало, що ChatGPT-4 у 90 % випадків правильно наслідує протоколи PFA, хоча може надавати недостовірні рекомендації, тому потрібен людський контроль [5].

Аналіз сучасних джерел свідчить, що попри значні зусилля організацій, більшість населення країн Заходу та України не має достатньої готовності до надання першої допомоги. Серед причин – нестача систематичних тренінгів, хибні уявлення, низький рівень залучення роботодавців і держави до підготовки дорослого населення та обмежений доступ до очних курсів у періоди пандемії чи воєнного стану.

Для подолання цих проблем потрібен комплексний підхід: розвиток офлайн-тренінгів у безпечних регіонах, обов’язкове навчання в школах та на підприємствах, створення національних інформаційних кампаній. Одночасно необхідно впроваджувати цифрові інструменти – мобільні додатки й телеграм-боти, які швидко надають перевірену інформацію та спрямовують користувачів до сертифікованих ресурсів. В умовах України, де велика частина населення користується Telegram, бот може стати універсальним

каналом для поширення матеріалів з фізичної та психологічної першої допомоги.

Розвиток таких сервісів повинен супроводжуватися етичними стандартами: повідомлення мають проходити експертну перевірку, користувачеві слід нагадувати, що інформація не замінює професійну медичну допомогу, а персональні дані повинні залишатися конфіденційними. У наступних розділах роботи буде розглянуто, як технології штучного інтелекту й телеграм-боти можуть реалізувати ці рекомендації та забезпечити надійний інструмент для підвищення готовності населення до дій у надзвичайних ситуаціях.

1.2. Огляд технологій створення чат-ботів у сфері охорони здоров’я

Чат-бот – це не просто інструмент для імітації розмови з користувачем, а програмний агент, що здатен забезпечувати природну комунікацію та адаптуватися до контексту діалогу. У сфері охорони здоров’я такі системи відіграють особливу роль, адже вони можуть підтримувати пацієнтів у критичних ситуаціях, надавати довідкову інформацію щодо першої допомоги та слугувати посередником між людиною й медичною інфраструктурою. Технологічною основою для цього є поєднання оброблення природної мови та алгоритмів машинного й глибинного навчання, що забезпечують здатність розпізнавати запити, класифікувати наміри й формувати доречні відповіді [7].

Важливим аспектом є інтеграція чат-ботів у вже існуючі цифрові системи – електронні медичні картки, бази даних чи сторонні сервіси. Це робить їх не лише зручним інструментом для користувача, а й ефективною частиною екосистеми охорони здоров’я [8]. Завдяки цьому забезпечується персоналізованість відповіді та швидкість доступу до релевантних даних. Крім того, сучасні моделі трансформерів і рекурентних нейронних мереж дають змогу системам ідентифікувати контекст і навіть прогнозувати подальший хід

розмови. Це, у свою чергу, дозволяє створювати більш природний і корисний діалог.

Поступове поєднання цих технологічних компонентів формує структуру, у якій різні модулі виконують взаємодоповнюючі функції. На рис. 1.1 показано основні складові медичного чат-бота, які визначають його функціональність і взаємодію з користувачем.

Рис. 1.1. Основні компоненти медичного чат-бота

У цій структурі важливе місце займає система управління діалогом, що відповідає за логіку ведення розмови та підтримку контексту. Вона забезпечує плавність і цілісність діалогу, дозволяючи не лише реагувати на окремі запити, а й вести послідовну комунікацію. Інтеграційні модулі взаємодіють із електронними медичними картками, базами даних і сторонніми сервісами, що робить можливим оперативний обмін медично значущою інформацією. Модуль оброблення мови виконує завдання виділення намірів і витягу сутностей, перетворюючи запит користувача на структуру даних, зрозумілу для алгоритмів. На цій основі модель машинного чи глибинного навчання класифікує наміри та формує відповіді, використовуючи трансформери або

рекурентні нейронні мережі. Завершальним етапом є генерація відповіді, яка робить взаємодію інтуїтивною та корисною для користувача.

Поширеною практикою є використання трансформерів (BERT, GPT) і великих мовних моделей (Large Language Models, LLM), які набувають знань шляхом само-навчання на гігантських обсягах текстових даних [9]. Огляд сучасних досліджень у сфері LLM для біомедицини констатує, що хоча універсальні моделі (GPT-4) досі домінують, існує тенденція до спеціалізації моделей для медичних текстів. У завданнях класифікації класичні трансформери поки що демонструють вищу точність, тоді як LLM краще працюють у режимі мало зразків та для генеративних завдань (резюмування). Дослідники підкреслюють необхідність уваги до приватності, упередженості й відтворюваності при використанні LLM.

З огляду на ці характеристики можна виділити ключові напрями практичного застосування, які розкривають переваги й обмеження сучасних моделей. Саме ці напрями подані у структурованому вигляді на рис. 1.2, що дає змогу наочно продемонструвати сфери, у яких використання ШІ в медицині є найбільш перспективним [10].

Рис. 1.2. Сфери застосування чат-бота

Суттєвим елементом функціонування є блок медичної інформації. У його межах чат-боти реалізують нагадування про прийом лікарських засобів, формують рекомендації щодо підтримки здорового способу життя та надають базову інтерпретацію симптомів. Такі можливості спрямовані на систематичний супровід пацієнта та створення умов для більш відповідального ставлення до власного стану здоров'я.

У сфері освітніх завдань чат-боти виступають джерелом доступу до інформаційних ресурсів, що охоплюють питання профілактики, надання першої медичної допомоги та ознайомлення з основними захворюваннями. Подібні матеріали є цінними для користувачів без спеціальної медичної підготовки, оскільки забезпечують швидке отримання перевірених відомостей у структурованій та доступній формі, зменшуючи ризик дезінформації.

Окремим напрямом виступає психологічна підтримка. Чат-боти можуть надавати інструменти для контролю емоційного стану, пропонувати базові рекомендації для зниження стресу та частково компенсувати дефіцит фахової допомоги. Використання таких систем у кризових умовах дозволяє знизити навантаження на психологів і забезпечити первинний рівень взаємодії з користувачем.

До адміністративного сегмента належать функції електронного запису на прийом, нагадування про візити та формування медичних записів. Автоматизація цих процесів оптимізує організацію роботи, скорочує часові витрати та підвищує зручність комунікації між пацієнтами й закладами охорони здоров'я. Додатково такі системи створюють передумови для безперервного моніторингу лікування та підтримки двостороннього інформаційного обміну між лікарем і пацієнтом.

Попри потенціал, вчені застерігають, що LLM-боти можуть поширювати неточні поради. У дослідженні про використання ChatGPT для інформування людей з епілепсією щодо фізичної активності виявлено, що чат-бот надає переважно консервативні та корисні відповіді, підкреслюючи користь фізичних вправ [12]. Однак у конкретних питаннях (наприклад, епілепсія та

хірургія) модель помилялась, тому рекомендується людський контроль. Авторами підкреслено, що LLM-боти можуть подолати мовні бар'єри й забезпечити доступність інформації для користувачів з країн з низьким доходом, але необхідно враховувати упередження, приватність і етичні міркування.

В огляді CADTH Horizon Scan зазначено, що чат-боти звільняють медичних працівників від рутинних завдань, можуть надавати симптом-чекер, нагадування про ліки та інформацію про самодопомогу [13]. Проте алгоритми здатні використовувати історичні дані, що не завжди відповідають актуальним клінічним рекомендаціям, тому важливий нагляд з боку фахівця.

З огляду на ці особливості особливої актуальності набуває використання телеграм-ботів як інструментів підтримки користувачів у сфері охорони здоров'я. Телеграм є однією з найпопулярніших комунікаційних платформ в Україні, а створення телеграм-бота на основі моделей штучного інтелекту для надання довідкових матеріалів з першої допомоги та психологічної підтримки має низку переваг, які наведено на рис. 1.3.

Рис. 1.3. Перспективи використання телеграм-бота на основі ШІ

Одним із ключових напрямів використання є підтримка під час воєнних дій та надзвичайних ситуацій. Телеграм-боти здатні виконувати функції психологічної першої допомоги, формувати алгоритми дій у небезпечних умовах і надавати оперативні рекомендації широкому колу користувачів. Такі системи демонструють ефективність у випадках, коли доступ до медичних закладів обмежений або відсутній, а інформаційна взаємодія має здійснюватися в максимально стислі терміни.

Характерною властивістю подібних сервісів є цілодобова доступність. Завдяки механізмам автоматизованого розпізнавання намірів користувачів чат-боти забезпечують фільтрацію другорядних даних і концентрують увагу на релевантних відповідях, зберігаючи структурований і зручний формат подання інформації.

Суттєвого значення набуває персоналізація та багатомовна підтримка. Для українського середовища вирішальним є застосування української мови поряд з англійською та іншими мовами, що дозволяє використовувати систему як внутрішніми користувачами, так і міжнародними партнерами. Це створює підґрунтя для масштабування сервісів у різних соціокультурних середовищах.

Окремим параметром є актуальність наданої інформації. Регулярне оновлення знань на основі сучасних клінічних протоколів та медичних рекомендацій забезпечує відповідність порад ботів сучасним стандартам. Такий підхід знижує ризик поширення застарілих даних і формує довіру до результатів взаємодії.

Перспективним напрямом є інтеграція чат-ботів з іншими сервісами. Автоматичні нагадування, інструменти збору статистики чи перенаправлення на онлайн-курси створюють умови для поступового підвищення рівня медичної обізнаності населення. Завдяки цьому чат-бот функціонує не лише як засіб отримання короткострокових рекомендацій, а й як інструмент довготривалої інформаційної підтримки в індивідуальних та організаційних системах охорони здоров’я.

Разом з тим необхідно передбачити етичні та юридичні аспекти. Чат-бот не може замінити лікаря і повинен містити відповідні застереження. Також важливо забезпечити конфіденційність даних, можливість позначати рівень достовірності інформації та механізм для прямого зв’язку з медичними фахівцями. Навчання моделі має включати останні клінічні протоколи, а відповіді – пройти верифікацію експертами.

1.3. Огляд існуючих систем для надання першої медичної допомоги

Розвиток цифрових технологій та широке поширення мобільних застосунків значною мірою вплинули на способи надання першої медичної допомоги. Якщо раніше навчання обмежувалося офлайн-курсами або друкованими посібниками, то сьогодні на ринку з’явилося чимало інтерактивних систем, орієнтованих на оперативне інформування та підтримку користувачів у надзвичайних ситуаціях. Такі рішення здатні забезпечити миттєвий доступ до покрокових інструкцій, алгоритмів дій чи рекомендацій відповідно до конкретних симптомів.

Серед існуючих систем можна виокремити як мобільні додатки, що пропонують інтерактивні інструкції й мультимедійні матеріали, так і чат-боти, інтегровані у популярні месенджери. Вони допомагають користувачам швидко зорієнтуватися в критичних обставинах і отримати базову інформацію ще до прибуття фахівців. Окрему категорію становлять інтелектуальні платформи, які застосовують алгоритми машинного навчання для інтерпретації введених даних і пропонують персоналізовані поради.

Розвиток цих систем зумовлений потребою в оперативності та доступності. У багатьох країнах дослідження показали, що рівень базових знань із першої допомоги серед населення залишається недостатнім, тому електронні інструменти стають ефективним способом зниження ризиків та поширення життєво важливих знань. Це визначає актуальність їхнього аналізу

й оцінки ефективності у контексті сучасної медицини та цифрового суспільства.

Чат-бот FAST «Джгут 2.0» – це волонтерський телеграм-бот, створений організацією FAST у співпраці з розробниками SayHiMedia, із метою надати кожному користувачеві доступ до покрокових інструкцій із першої медичної допомоги (рис. 1.4) [14]. Він орієнтований як на людей без спеціальної підготовки, так і на тих, хто вже проходив тренінги, і пропонує різні рівні деталізації порад.

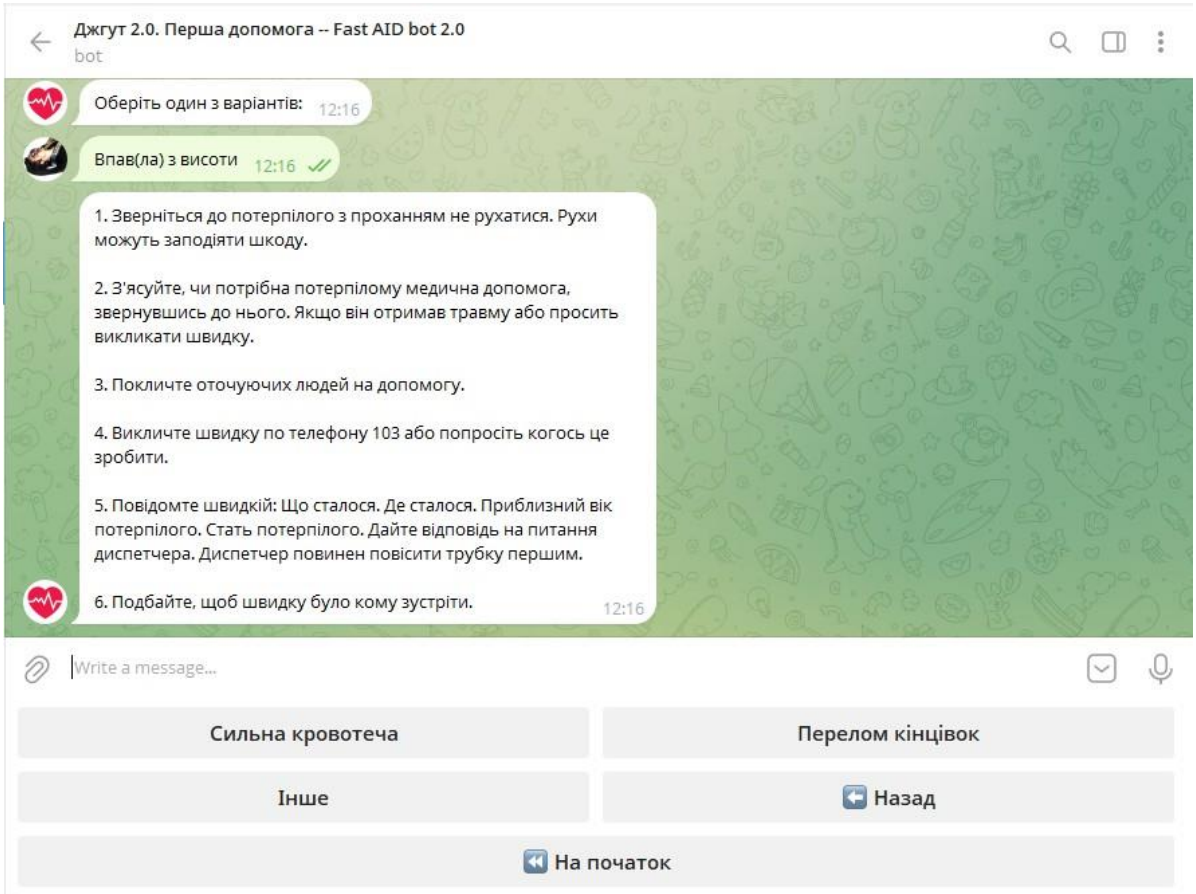

Рис. 1.4. Приклад інтерфейсу для роботи із чат-ботом FAST «Джгут 2.0»

Функціонал «Джгута 2.0» охоплює низку типових екстрених сценаріїв – що робити при травмах, втраті свідомості, епілептичному нападі, підозрі на інсульт, болі в грудях та інших критичних станах. Бот доступний українською, англійською та російською мовами й адаптований під кілька платформ: Telegram, Viber та Facebook Messenger. Користувач може обрати режим «для

новачків», де даються прості, мінімалізовані інструкції, або «для підготовлених», де подаються деталізовані алгоритми й нагадування про складніші етапи підтримки життя.

Переваги:

– доступність і багатомовність. Бот працює у кількох популярних месенджерах (Telegram, Viber, Facebook Messenger) та підтримує українську та англійську мови;

– практичність та адаптивність. Інструкції подаються у двох режимах: для новачків (спрощені кроки) та для підготовлених користувачів (детальні алгоритми);

– використання мультимедійних матеріалів. Фото й відео допомагають краще зрозуміти алгоритми першої допомоги;

– орієнтація на різні екстрені ситуації. Охоплює широкий спектр станів: від травм і втрати свідомості до інсультів та нападів.

Недоліки:

– відсутність офіційної сертифікації. Бот не є державним медичним інструментом, тому його поради не завжди можуть розглядатися як клінічно офіційні;

– обмежена інтерактивність. Попри корисність, бот не замінює живого консультанта і не може повністю адаптувати відповіді до унікальних умов;

– залежність від доступу до інтернету. У критичних умовах відсутність зв’язку унеможливлює використання;

– питання безпеки даних. Як волонтерський проєкт, він не має гарантій захисту персональної інформації на рівні державних або медичних систем.

FAST «Джгут 2.0» є практичним і зручним інструментом для підвищення рівня базової медичної грамотності та підтримки населення в екстрених ситуаціях. Він не може замінити кваліфіковану медичну допомогу, але суттєво знижує інформаційний бар’єр і допомагає швидко зорієнтуватися

в критичних обставинах. Як додатковий освітній та прикладний ресурс цей бот має значну соціальну цінність, особливо у воєнних і кризових умовах.

Babylon Health – це цифрова медична платформа, яка використовувала чат-бот як один із своїх основних сервісів: бот дозволяв користувачу ввести свої симптоми й відповісти на низку питань, після чого система аналізувала відповіді та пропонувала рекомендації щодо ймовірного діагнозу і подальших дій, наприклад звернення до лікаря чи самостійна терапія (рис. 1.5). Окрім функції «symptom checker», в екосистемі Babylon передбачалися відеоконсультації з лікарями, доступ до даних про стан здоров'я користувача, відстеження життєвих показників, а також поєднання з іншими сервісами охорони здоров'я, наприклад, направленнями до спеціалістів чи доставкою рецептів [15]. Система використовувала інтелектуальні моделі, зокрема мережі Байєсова типу та інші алгоритми, щоб моделювати ймовірні взаємозв'язки між симптомами й хворобами, а також надавати пояснення, чому запропоновано ту чи іншу дію користувачу.

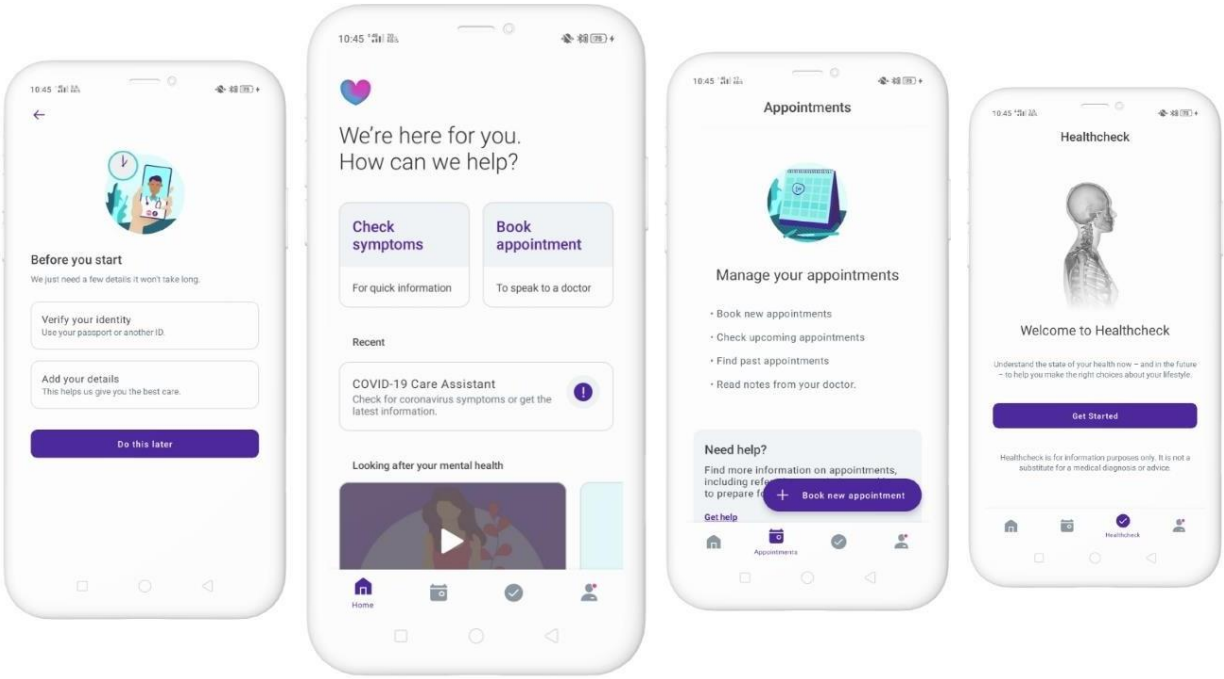

Рис. 1.5. Приклад інтерфейсу для роботи із чат-ботом «Babylon Health»

Babylon Health – це цифрова медична платформа, яка використовувала чат-бот як один із своїх основних сервісів: бот дозволяв користувачу ввести свої симптоми й відповісти на низку питань, після чого система аналізувала відповіді та пропонувала рекомендації щодо ймовірного діагнозу і подальших дій, наприклад звернення до лікаря чи самостійна терапія. Окрім функції «symptom checker», в екосистемі Babylon передбачалися відеоконсультації з лікарями, доступ до даних про стан здоров'я користувача, відстеження життєвих показників, а також поєднання з іншими сервісами охорони здоров'я, наприклад, направленнями до спеціалістів чи доставкою рецептів. Система використовувала інтелектуальні моделі, зокрема мережі Байєсова типу та інші алгоритми, щоб моделювати ймовірні взаємозв'язки між симптомами й хворобами, а також надавати пояснення, чому запропоновано ту чи іншу дію користувачу.

Проте сам бот ніколи не був автономною клінічною системою – він служив інструментом попередньої самооцінки і допомагав направити користувача до відповідного каналу – чи то відеоконсультація, чи звернення в лікарню. Крім того, один із викликів такої системи – забезпечення точності та безпеки рекомендацій, особливо у випадках із життєво небезпечними станами, коли помилка може дорого коштувати. У медичній спільноті були зауваження щодо можливості неправильного триажу або недооцінки симптомів, що змушувало Babylon періодично переглядати алгоритми й погоджувати їх із клінічними стандартами.

Переваги:

– швидкий доступ до медичної інформації. Користувач може оперативно отримати попередній аналіз симптомів без необхідності відвідувати лікаря;

– інтеграція з телемедичними сервісами. Бот поєднаний із відеоконсультаціями лікарів, електронними рецептами та іншими цифровими сервісами;

– масштабованість. Система може одночасно обслуговувати тисячі користувачів, зменшуючи навантаження на медичний персонал;

– інноваційність підходу. Використання алгоритмів штучного інтелекту, включно з байєсовими моделями та NLP, для моделювання зв’язків між симптомами й хворобами.

Недоліки:

– неточність триажу. У деяких випадках бот міг недооцінити серйозність симптомів або надати некоректну рекомендацію;

– відсутність клінічної автономності. Не замінює лікаря й потребує подальшого підтвердження діагнозу професіоналом;

– критика з боку медичної спільноти. Були зауваження щодо надійності алгоритмів і відповідності клінічним протоколам;

– питання конфіденційності. Як і більшість цифрових медичних сервісів, система вимагала обробки персональних даних, що викликало занепокоєння щодо їхнього захисту.

Отже, Babylon Health став одним із найбільш відомих прикладів використання чат-ботів у медицині, демонструючи потенціал штучного інтелекту у сфері первинного аналізу симптомів та телемедицини. Хоча він значно полегшив доступ до інформації й скоротив час очікування на консультацію, водночас постали виклики щодо точності, безпеки та клінічної валідності його рекомендацій. У результаті цей застосунок варто розглядати як допоміжний інструмент, що сприяє орієнтації користувача в системі охорони здоров’я, але не може повністю замінити професійну медичну допомогу.

Чат-бот «Турбота» створений як сервіс, що допомагає пацієнтам оперативно знаходити лікарів і отримувати безкоштовні онлайн-консультації (рис. 1.6). Ініційований платформою Bookimed, цей бот намагається зменшити перепони для доступу до медичної допомоги, особливо в умовах обмеженого фізичного доступу до клінік або під час надзвичайних ситуацій [16].

У своїй роботі «Турбота» дозволяє користувачу описати симптоми або свої проблеми зі здоров'ям і обрати спеціальність лікаря, який може надати консультацію. Бот формує список відповідних фахівців, з якими пацієнт може зв'язатися, та направляє контакти лікарів або повідомлення щодо подальшого зв'язку. Багато з лікарів консультують безкоштовно як волонтери, обираючи цей бот як інструмент доброчинної діяльності. Сервіс охоплює широкий спектр спеціалізацій – від терапевтів і інфекціоністів до невропатологів, психологів і педіатрів.

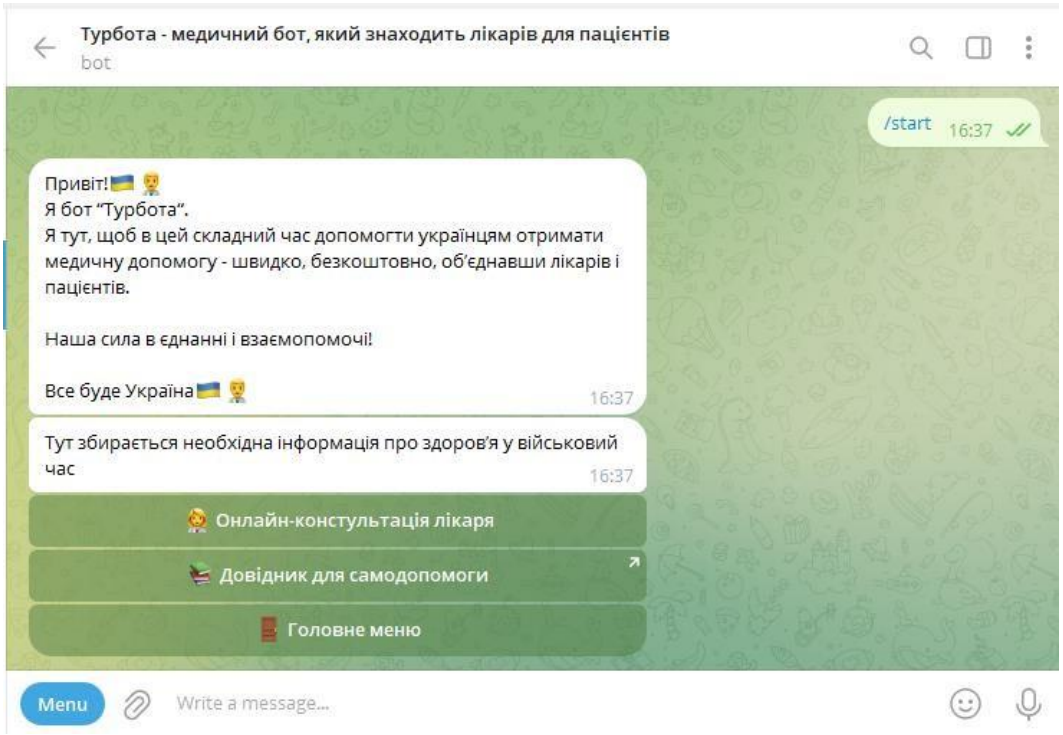

Рис. 1.6. Приклад інтерфейсу для роботи із чат-ботом «Турбота»

Переваги:

– безкоштовність та волонтерський характер. Значна частина лікарів консультує добровільно, що робить сервіс доступним для користувачів;

– швидкий зв'язок із фахівцями. Бот оперативно спрямовує пацієнта до лікаря потрібної спеціалізації, що зменшує час на пошук допомоги;

– широкий спектр спеціальностей. Користувач може отримати консультацію від терапевтів, психологів, педіатрів та інших лікарів;

– простота використання. Сервіс доступний у Telegram та не вимагає складної реєстрації чи завантаження додаткових додатків.

Недоліки:

– відсутність офіційної сертифікації. Поради не є клінічно затвердженими та не можуть розглядатися як медичний висновок;

– обмежена діагностична роль. Бот не ставить діагнози, а лише допомагає зорієнтуватися й зв’язатися з лікарем;

– залежність від доступу до мережі. У разі відсутності інтернету сервіс недоступний;

– ризики конфіденційності. Збір контактних даних та симптомів користувача може створювати певні загрози для приватності.

Отже, медичний чат-бот «Турбота» є корисним цифровим інструментом, що підвищує доступність медичних консультацій у кризових умовах та полегшує комунікацію з лікарями. Він не замінює професійну медичну діагностику, але значно скорочує шлях від виникнення симптомів до отримання поради фахівця. Як додатковий сервіс, орієнтований на оперативність і простоту використання, «Турбота» має соціальну цінність, хоча його ефективність обмежується консультативною функцією та питаннями безпеки даних.

Аналіз існуючих чат-ботів для надання першої медичної допомоги показав, що більшість із них або занадто складні для широкого кола користувачів, або не забезпечують достатньої точності та актуальності інформації. Це обґрунтовує потребу у створенні нового рішення, яке базуватиметься на сучасних моделях штучного інтелекту та забезпечуватиме більш персоналізовану взаємодію. Простий і зрозумілий функціонал дозволить швидко орієнтуватися у критичних ситуаціях, а доступність через популярні месенджери усуне необхідність встановлення додаткових застосунків. Такий підхід зробить сервіс не лише зручним, а й максимально корисним у кризових умовах, підвищуючи рівень готовності населення до надання першої допомоги.

РОЗДІЛ 2.
АНАЛІЗ МЕТОДІВ ТА ТЕХНОЛОГІЙ РОЗРОБКИ

2.1. Методи штучного інтелекту для обробки користувацьких запитів

Ефективність інтелектуальних систем, що надають довідкові матеріали з першої медичної допомоги, безпосередньо залежить від обраних методів обробки користувацьких запитів. Ключовим завданням є здатність системи правильно інтерпретувати введений текст, визначити наміри користувача та надати відповідь, що відповідає медичному контексту. Для цього застосовуються алгоритми машинного навчання, методи обробки природної мови та моделі класифікації, здатні працювати з різними формами формулювання питань.

У сучасних дослідженнях активно використовуються як класичні підходи, такі як логістична регресія, дерева рішень чи методи ансамблю, так і сучасні глибинні архітектури на основі трансформерів. Для системи, орієнтованої на швидку обробку запитів у режимі реального часу, особливе значення мають алгоритми, що забезпечують баланс між точністю класифікації та швидкістю роботи. Одним із таких методів є «швидке дерево» (ШД), яке довело свою ефективність у задачах класифікації текстів завдяки оптимізованому механізму навчання на великих обсягах даних.

Використання подібних підходів дає змогу досягати високої продуктивності навіть за умов обмежених ресурсів, що робить їх придатними для інтеграції в месенджери, де критичною є швидкість відповіді. Крім того, методи штучного інтелекту дозволяють враховувати не лише ключові слова, а й контекст запиту, що підвищує релевантність наданих матеріалів та формує більш природну взаємодію між користувачем і системою.

Метод *швидкого дерева* є оптимізованим алгоритмом побудови дерев рішень, який широко використовується у бібліотеці ML.NET для задач класифікації, регресії та ранжування [17]. Він належить до ансамблевих

методів машинного навчання і ґрунтується на принципі градієнтного бустингу над деревами рішень. Завдяки використанню стохастичної оптимізації та спеціальних евристик ШД забезпечує швидке навчання навіть на великих наборах даних, демонструючи при цьому високий рівень точності.

ШД активно застосовується у завданнях, де необхідна ефективна класифікація або прогнозування. Його використовують для аналізу текстових даних, визначення категорій повідомлень, виявлення спаму, прогнозування поведінки клієнтів, аналізу медичних даних та біоінформатики [18]. Однією з важливих переваг методу є здатність працювати з великою кількістю ознак та швидко адаптуватися до змін у структурі даних.

Даний метод може бути застосований для класифікації користувацьких запитів. Алгоритм здатен визначати, до якої категорії належить введене повідомлення: наприклад, «зупинка кровотечі», «опіки», «серцево-легенева реанімація», «травми» чи «інсульт». Завдяки цьому система швидко обирає відповідний блок довідкових матеріалів і повертає користувачу релевантну інструкцію.

На рис. 2.1 наведено узагальнену схему функціонування цього методу.

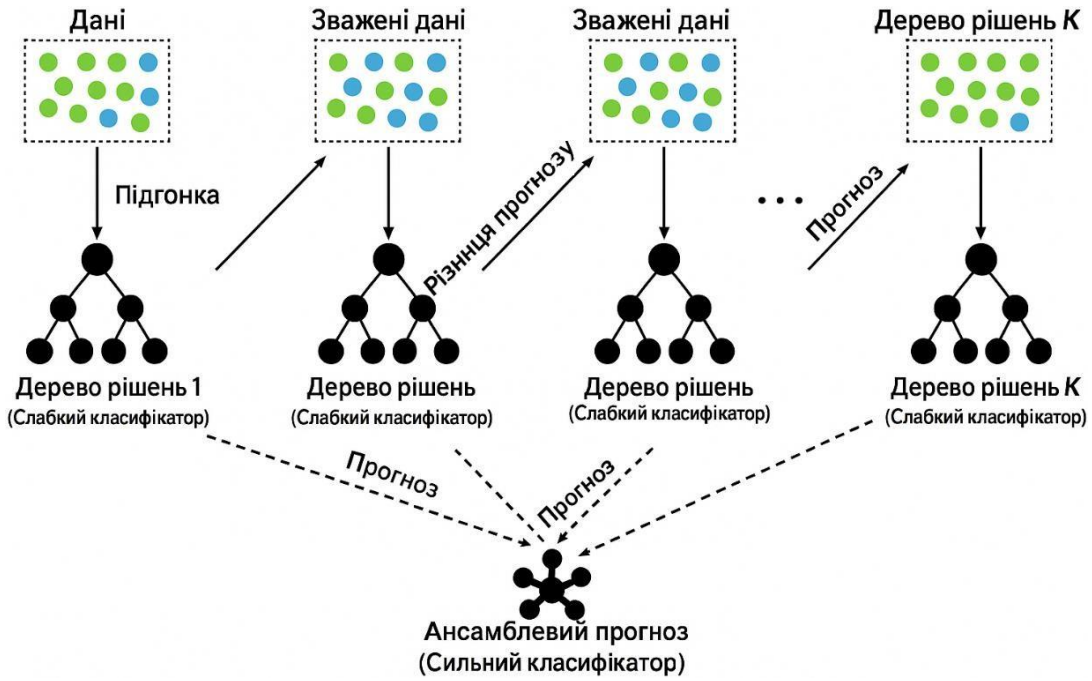

Рис. 2.1. Схема роботи методу ШД

Робота методу починається з наявних вхідних даних, які використовуються для побудови першого дерева рішень. Це дерево виступає слабким класифікатором, що намагається відобразити основні закономірності, проте має певні похибки. На наступному етапі дані піддаються зважуванню, де більша вага надається тим прикладам, які були неправильно класифіковані. Друге дерево рішень будується з урахуванням цих нових ваг, що дозволяє зменшити кількість помилок. Процес повторюється багаторазово, і на кожному кроці створюється нове дерево, яке уточнює прогноз попередніх.

Поступове нарощування дерев формує ансамбль, де кожен окремий класифікатор є відносно простим, але разом вони створюють сильну модель. У кінцевому результаті прогноз формується шляхом агрегації результатів усіх дерев рішень. Такий підхід забезпечує високу точність і стійкість до шуму в даних.

Переваги методу є:

– висока точність завдяки ансамблевому підходу. Комбінація великої кількості слабких класифікаторів забезпечує надійний результат і зменшує ймовірність помилок;

– ефективність у роботі з великими наборами даних. Алгоритм оптимізований для швидкого навчання та передбачення, що робить його придатним для реального часу;

– гнучкість застосування. Може використовуватися для задач класифікації, регресії та ранжування, включно з аналізом текстових запитів у медичних чат-ботах.

До недоліків методу можна віднести:

– високі обчислювальні витрати на тренування. Хоча прогнозування виконується швидко, процес навчання на великих обсягах даних може вимагати значних ресурсів;

– складність інтерпретації. Ансамблевий характер моделі ускладнює пояснення логіки прийняття рішення користувачам;

– чутливість до якості даних. У разі наявності шуму або дисбалансу класів модель може давати упереджені результати без додаткової обробки даних.

Отже, метод ШД є одним із найбільш збалансованих інструментів для побудови класифікаційних моделей, що поєднує високу точність та швидкість обробки запитів. Його використання у чат-ботах для надання довідкових матеріалів з першої медичної допомоги дозволяє оперативно класифікувати запити користувачів та формувати релевантні відповіді. Попри певні обмеження, пов’язані з обчислювальними ресурсами й складністю інтерпретації, ШД залишається надійним методом для застосування в інтелектуальних системах, орієнтованих на швидку підтримку користувачів у критичних ситуаціях.

Логістична регресія (ЛР) є одним із найпоширеніших статистичних методів, що використовується для задач класифікації. На відміну від лінійної регресії, яка прогнозує числові значення, логістична регресія визначає ймовірність належності прикладу до певного класу [19]. Основою методу є сигмоїдна (логістична) функція, яка перетворює лінійну комбінацію вхідних ознак у значення від 0 до 1, що інтерпретується як ймовірність. У найпростішому випадку йдеться про бінарну класифікацію, однак метод легко узагальнюється для задач із кількома класами.

Даний метод широко використовується в медицині для прогнозування наявності захворювань за сукупністю симптомів, у банківській сфері для оцінки кредитних ризиків, у маркетингу для аналізу поведінки споживачів [20]. Вона є базовим і водночас надійним алгоритмом, особливо коли дані мають відносно просту структуру та не вимагають надскладних моделей. Завдяки інтерпретованості її результати часто обирають у критично важливих сферах, де потрібно пояснити, чому система ухвалила певне рішення.

У системі для надання довідкових матеріалів логістична регресія може бути застосована для класифікації користувацьких запитів за попередньо визначеними категоріями. Наприклад, бот отримує текстове повідомлення:

«людина втратила свідомість» – і алгоритм логістичної регресії визначає з високою ймовірністю, що цей запит належить до категорії «невідкладні стани». На основі цього класифікаційного рішення користувачу миттєво пропонується відповідний довідковий матеріал або алгоритм дій: як перевірити дихання, викликати швидку допомогу, розпочати серцево-легеневу реанімацію.

На рис. 2.2 зображено схему роботи методу логістичної регресії, де послідовно показано ключові етапи перетворення вхідних даних у вихідне рішення. Вхідні незалежні змінні x_1, x_2,…,x_5 подаються на вхід моделі та зважуються за допомогою коефіцієнтів, що відображають їхню вагомість у загальній моделі. Далі всі ці добутки підсумовуються, утворюючи комбіноване значення. Отримана сума проходить через функцію активації, у цьому випадку – сигмоїдну функцію, яка переводить результат у значення від 0 до 1, що інтерпретується як ймовірність належності до певного класу.

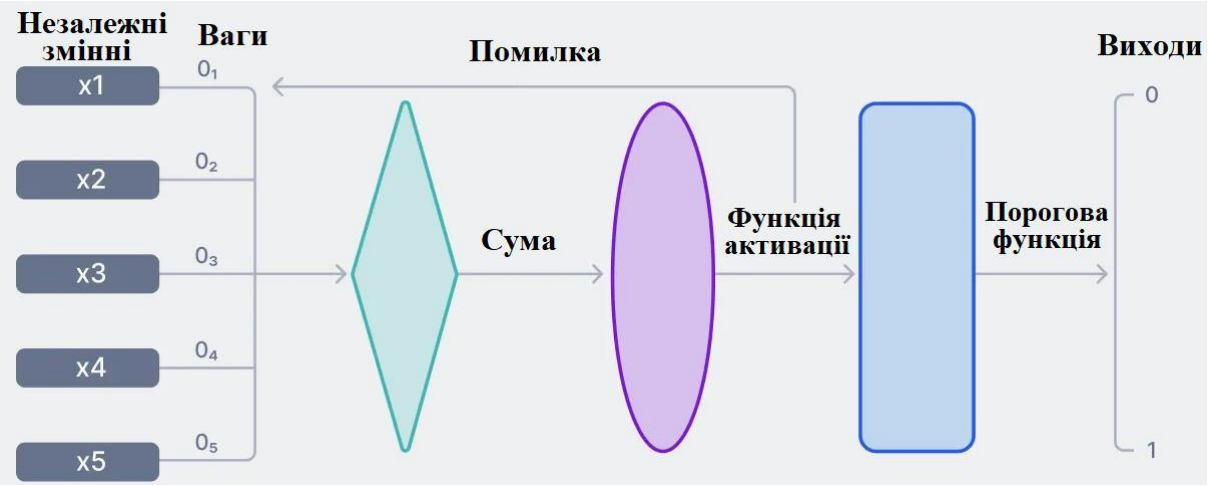

Рис. 2.2. Схема роботи методу логістичної регресії

На фінальному етапі застосовується порогова функція: якщо ймовірність перевищує встановлений поріг (наприклад, 0,5), система відносить запит до класу «1» (наприклад, «потребує медичної поради»), у протилежному випадку – до класу «0» (наприклад, «загальна інформація»). У разі використання в

медичному чат-боті така схема дозволяє не лише формувати прості рішення, а й робити їх зрозумілими та пояснюваними для користувача.

Переваги методу логістичної регресії:

– простота та зрозумілість. Метод має чітке математичне підґрунтя, що дозволяє легко інтерпретувати результати й пояснювати, як саме модель прийняла рішення;

– ефективність на малих і середніх наборах даних. Логістична регресія не потребує значних обчислювальних ресурсів і швидко навчається навіть при обмеженій кількості прикладів;

– ймовірнісний вихід. Модель генерує не лише клас (0 чи 1), а й оцінку ймовірності, що корисно у випадках, коли потрібно враховувати ступінь упевненості у прогнозі.

Недоліками логістичної регресії є:

– лінійність межі рішень. Метод добре працює лише тоді, коли класи можна розділити лінійно; у складних випадках його точність різко знижується;

– чутливість до корельованих ознак. У випадку мультиколінеарності (сильно пов’язаних між собою змінних) модель може давати нестабільні результати;

– обмежена здатність до моделювання складних залежностей. У порівнянні з методами на основі дерев чи нейронними мережами, логістична регресія менш гнучка для роботи зі складними багатовимірними даними.

Отже, ЛР є базовим, проте надійним методом класифікації, який дозволяє швидко та пояснювано аналізувати дані. У контексті надання довідкових матеріалів з першої медичної допомоги вона може ефективно застосовуватися для первинної обробки запитів користувачів і швидкої оцінки їхньої терміновості. Попри певні обмеження, пов’язані з відсутністю глибокого аналізу складних контекстів, цей метод залишається важливим інструментом у поєднанні з більш сучасними алгоритмами штучного інтелекту.

Метод *k найближчих сусідів* (KHH) належить до простих, але водночас дієвих алгоритмів машинного навчання з учителем, що застосовується як у задачах класифікації, так і в регресії [21]. Його ключова ідея полягає в тому, що схожі об’єкти, які мають близькі характеристики у багатовимірному просторі ознак, з великою ймовірністю належать до одного класу. Новий приклад класифікується шляхом пошуку k найближчих елементів у навчальній вибірці, після чого рішення ухвалюється на основі більшості «голосів» цих сусідів.

У практичних застосуваннях KHH виявив себе у багатьох сферах – від розпізнавання зображень та аналізу медичних даних до побудови рекомендаційних систем [22]. У медицині алгоритм корисний для ідентифікації хвороб за симптомами, прогнозування перебігу патологій та навіть у задачах роботи з медичними зображеннями. Важливою його перевагою є відсутність складного навчання: модель фактично зберігає дані, а прогноз формується вже під час порівняння нових прикладів із наявними.

Для задач довідкових систем із першої медичної допомоги KHH може стати інструментом класифікації користувацьких запитів. Наприклад, при введенні симптомів на кшталт болю у грудях, запаморочення чи сильної кровотечі алгоритм зіставляє їх з подібними прикладами в базі та визначає, якій категорії відповідає ситуація. Це дозволяє чат-боту запропонувати релевантні інструкції: порядок дій при інфаркті, правила накладання пов’язки чи необхідність виклику швидкої допомоги. Таким чином, метод забезпечує простий і зрозумілий механізм надання користувачеві перевіреної інформації в критичних обставинах.

Рис. 2.3 висвітлює узагальнену схему роботи алгоритму k найближчих сусідів, яка ілюструє, як вхідні дані співвідносяться з відомими прикладами, після чого відбувається визначення результату. Це дозволяє наочно продемонструвати, як простий принцип близькості між об’єктами може бути ефективно використаний для вирішення прикладних завдань у сфері охорони здоров’я.

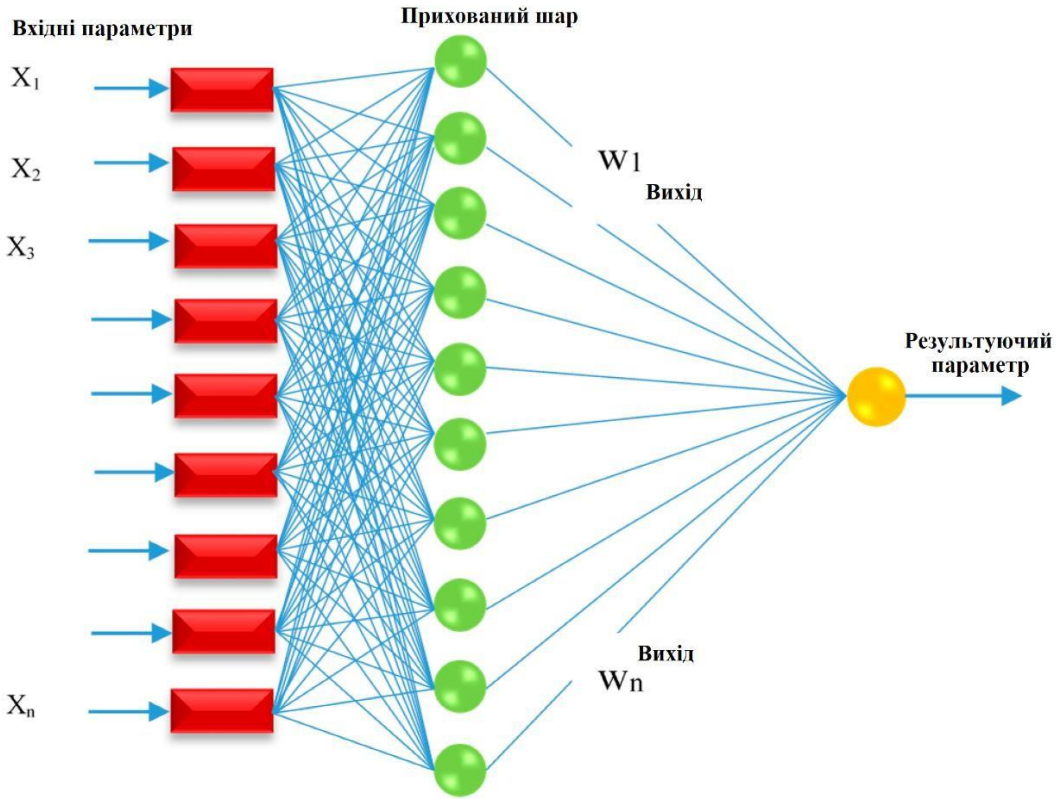

Рис. 2.3. Схема роботи методу k найближчих сусідів

На вхід подаються параметри, що описують об’єкт (наприклад, симптоми або інші характеристики стану користувача). Ці вхідні дані співставляються з множиною вже відомих прикладів у навчальній вибірці. Кожен приклад зберігається у багатовимірному просторі ознак, а схожість між ними оцінюється за допомогою метрик відстані (наприклад, евклідової чи косинусної). Далі алгоритм визначає k найближчих сусідів до нового об’єкта та на основі більшості їхніх міток робить прогноз. У випадку використання для надання довідкових матеріалів з першої медичної допомоги це може означати, що чат-бот, отримавши опис симптомів від користувача, знаходить у базі найбільш подібні ситуації й надає рекомендацію відповідно до тих сценаріїв, які вже закладені в навчальній вибірці.

Переваги методу КНН становлять:

– простота реалізації та інтерпретації. Алгоритм легко зрозуміти та пояснити, що робить його зручним для практичного застосування у медичних чат-ботах;

– гнучкість у використанні. Метод може застосовуватись як для класифікації, так і для регресії, що дозволяє адаптувати його до різних типів завдань;

– відсутність потреби у тренуванні моделі. КНН не вимагає попереднього навчання, він просто зберігає дані та працює на етапі передбачення, що спрощує підтримку системи.

Недоліки методу КНН:

– висока обчислювальна складність. При великій кількості даних пошук найближчих сусідів стає повільним, що може знижувати швидкодію чат-бота;

– чутливість до вибору метрики відстані. Результати класифікації залежать від способу вимірювання схожості між об’єктами, що може впливати на точність;

– проблема з нерівномірними даними. Якщо навчальна вибірка має дисбаланс класів або багато шуму, то метод може давати упереджені результати.

Отже, метод k найближчих сусідів є одним із найпростіших та інтуїтивно зрозумілих підходів у машинному навчанні. Його ефективність особливо помітна при невеликих обсягах даних та добре структурованих вибірках. У контексті надання довідкових матеріалів з першої медичної допомоги він дозволяє швидко знайти релевантні рекомендації, порівнюючи описані користувачем симптоми з уже наявними прикладами у базі. Однак для масштабних систем або для випадків, де важлива висока швидкість реакції, необхідно враховувати обмеження цього методу та по можливості комбінувати його з іншими алгоритмами.

2.2. Обґрунтування вибору методу штучного інтелекту

При розробці системи для надання довідкових матеріалів з першої медичної допомоги важливо обрати метод машинного навчання, який

забезпечує баланс між точністю, швидкістю та зручністю практичного використання. Оскільки користувацькі запити можуть бути короткими, містити обмежений контекст і водночас вимагати швидкої відповіді, вибір алгоритму повинен враховувати його здатність ефективно працювати з різнорідними даними та адаптуватися до нових прикладів без надмірних витрат ресурсів.

Серед розглянутих методів доцільно виділити «швидке дерево», логістичну регресію та алгоритм k найближчих сусідів. Кожен з них має власні переваги й обмеження, пов’язані з принципом роботи, вимогами до обчислювальних ресурсів та сферою застосування. Щоб обґрунтувати вибір у межах даного дослідження, доцільно порівняти ключові характеристики зазначених методів у вигляді табл. 2.1. Це дозволить визначити їхню придатність для задачі класифікації користувацьких запитів у сфері першої медичної допомоги.

Таблиця 2.1

Порівняння методів машинного навчання для класифікації запитів

Характеристика	Швидке дерево	Логістична регресія	k найближчих сусідів
Швидкість навчання	Висока	Середня	Низька
Швидкість прогнозування	Висока	Висока	Низька при великих даних
Робота з нелінійними залежностями	Добра	Обмежена	Залежить від метрики
Стійкість до шумних даних	Висока	Середня	Низька
Масштабованість	Добра	Висока	Обмежена
Інтерпретованість результатів	Середня	Висока	Середня

Вибір методу «швидкого дерева» для розробки системи пояснюється його здатністю забезпечувати оптимальний баланс між точністю прогнозу та швидкістю роботи. На відміну від алгоритму найближчих сусідів, який втрачає

продуктивність при великому обсязі даних, швидке дерево демонструє високу ефективність як на етапі навчання, так і під час класифікації нових запитів. Окрім того, цей метод краще справляється з нелінійними залежностями, що важливо при аналізі різноманітних формулювань користувачів у сфері першої медичної допомоги. Його стійкість до шумних даних і здатність масштабуватися без значної втрати якості робить його найбільш доцільним вибором для реалізації інтелектуального телеграм-бота.

2.3. Вибір інструментальних засобів і програмних бібліотек

Розробка інтелектуальної системи потребує ретельного добору інструментальних засобів, оскільки саме від них залежить не лише швидкість реалізації, але й стабільність та зручність подальшої експлуатації. У випадку чат-ботів, що працюють у сфері охорони здоров’я, особливо важливо враховувати сумісність інструментів із алгоритмами штучного інтелекту та можливість інтеграції з популярними платформами. Вибір програмних бібліотек і мов програмування повинен забезпечувати гнучкість для роботи з текстовими даними, простоту масштабування та зручність підтримки коду.

Серед найбільш поширених мов програмування для побудови подібних систем можна виділити Python, Java та C#. Кожна з них має власні переваги, набір бібліотек і середовище використання, що визначає її роль у сучасних проєктах.

Python вирізняється широким спектром бібліотек для штучного інтелекту та обробки природної мови (TensorFlow, PyTorch, scikit-learn), завдяки чому ця мова є популярною у дослідженнях і розробці прототипів [23]. Вона забезпечує швидку реалізацію експериментів, має простий синтаксис і велику спільноту розробників, що робить її придатною для побудови й тестування моделей у сфері медицини.

Java відома своєю надійністю, кросплатформеністю та використанням у великих корпоративних рішеннях [24]. Вона добре підходить для побудови

масштабованих серверних систем і підтримує інтеграцію з інструментами аналізу даних, хоча для роботи з AI потребує додаткових бібліотек, таких як Deeplearning4j.

C# має потужну підтримку в середовищі Microsoft та тісну інтеграцію з ML.NET, що дозволяє реалізовувати моделі машинного навчання безпосередньо у застосунках [25]. Завдяки цьому C# зручний для створення прикладних систем із графічним інтерфейсом, зокрема для чат-ботів, що працюють у Telegram і поєднують обробку запитів користувачів із базою знань.

Для обґрунтування вибору мови для розробки системи, ключові характеристики цих мов подані у табл. 2.2, де наведено їхні якісні та кількісні показники.

Таблиця 2.2

Порівняння мов програмування за основними характеристиками

Характеристика	Python	Java	C#
Простота синтаксису	Висока (навчання займає 2–4 тижні)	Середня (навчання займає 1–2 місяці)	Середня (навчання займає 1–2 місяці)
Бібліотеки для штучного інтелекту	Дуже широкий набір (TensorFlow, PyTorch, scikit-learn)	Обмежений вибір (Deeplearning4j та ін.)	Середній вибір (ML.NET, Accord.NET)
Продуктивність виконання програм	Середня	Висока	Висока
Масштабованість систем	Добра, але потребує оптимізації	Висока, особливо у корпоративних середовищах	Висока, особливо в рамках екосистеми .NET
Інтеграція з платформами	Універсальна	Кросплатформена	Тісна інтеграція з Windows/.NET
Швидкість розробки	Висока	Середня	Висока

Аналіз порівняння мов програмування показує, що кожна з них має власну сферу оптимального застосування. Python завдяки великій кількості бібліотек є ідеальним середовищем для досліджень і побудови прототипів у сфері штучного інтелекту, однак у прикладних системах може виникати потреба в додатковій оптимізації для забезпечення високої продуктивності. Java, своєю чергою, демонструє надійність та масштабованість, що робить її доречною для великих корпоративних рішень, але процес розробки на ній є відносно повільнішим і потребує більшого часу для інтеграції AI-моделей.

C# вирізняється збалансованістю між швидкістю розробки, продуктивністю та інтеграцією з бібліотеками машинного навчання. Завдяки ML.NET ця мова забезпечує можливість безпосередньо реалізовувати моделі класифікації в прикладних застосунках без необхідності використання сторонніх інструментів. Крім того, її тісна інтеграція з середовищем Visual Studio та платформою .NET створює умови для швидкої розробки, тестування й подальшого розгортання системи. Саме ці характеристики роблять C# найбільш доцільним вибором для реалізації інтелектуального телеграм-бота, орієнтованого на надання довідкових матеріалів з першої медичної допомоги.

Оскільки для реалізації системи було обрано мову програмування C#, наступним важливим кроком є визначення середовища розробки, яке забезпечить комфортну роботу з кодом, підтримку інструментів машинного навчання та інтеграцію з потрібними бібліотеками. Серед найбільш поширених варіантів виділяють Visual Studio 2022, Visual Studio Code та Rider. Кожне з цих середовищ має власні особливості, що впливають на процес створення програмного забезпечення.

Visual Studio 2022 (VS 2022) є повноцінним інтегрованим середовищем розробки від Microsoft, яке пропонує широкий набір інструментів для роботи з C# [26]. Воно підтримує ML.NET, має зручний відлагоджувач, інтеграцію з базами даних і засоби для побудови графічних інтерфейсів. Завдяки розширеному функціоналу VS 2022 є оптимальним вибором для масштабних проєктів та комплексної роботи з різними технологіями.

Visual Studio Code – це легкий кросплатформений редактор коду з можливістю розширення за допомогою плагінів [27]. Він підтримує C# через розширення OmniSharp і дозволяє швидко налаштовувати середовище під конкретні завдання. Хоча його функціонал поступається Visual Studio 2022, VS Code добре підходить для невеликих проєктів та ситуацій, коли важлива мобільність і мінімальне споживання ресурсів.

Rider – це IDE від компанії JetBrains, яка спеціалізується на розробці інструментів для програмістів. Rider поєднує можливості ReSharper і IntelliJ, пропонуючи потужний аналіз коду, підтримку різних мов та інтеграцію з .NET Core [28]. Його перевага полягає у зручності роботи з багатопроєктними рішеннями, однак для повного використання функціоналу потрібна платна ліцензія.

Ключові характеристики VS 2022, Visual Studio Code та Rider наведені у табл. 2.3, що дає змогу зіставити їх за основними параметрами, важливими для практичної реалізації.

Таблиця 2.3

Порівняння середовищ розробки VS 2022, Visual Studio Code та Rider

Характеристика	Visual Studio 2022	Visual Studio Code	Rider
1	2	3	4
Повнота інструментарію	Дуже висока (від графічного інтерфейсу до ML.NET)	Базова, розширюється за рахунок плагінів	Висока, орієнтована на .NET і JVM
Інтеграція з ML-бібліотеками	Вбудована підтримка ML.NET	Можлива через розширення та налаштування	Підтримує інтеграцію, але потребує додаткових модулів
Відлагодження та тестування	Розширені засоби, включаючи профілювання	Обмежені інструменти, здебільшого через плагіни	Добре інтегровані, але менш зручні для GUI

Продовження таблиці 2.3

1	2	3	4
Робота з великими проєктами	Оптимізоване середовище для багатокомпонентних рішень	Підходить для невеликих і середніх проєктів	Придатне для багатопроєктних систем, але менш інтуїтивне
Продуктивність IDE	Висока, оптимізована для Windows	Висока на різних платформах	Висока, але потребує більше ресурсів
Ліцензія та вартість	Безкоштовна Community-версія	Повністю безкоштовний	Платна ліцензія (з пробним періодом)

Для розробки інтелектуальної системи було обрано VS 2022, оскільки воно поєднує у собі повний набір інструментів для роботи з мовою C# та тісно інтегрується з бібліотекою ML.NET. Це середовище дозволяє не лише реалізовувати алгоритми машинного навчання, але й ефективно тестувати, відлагоджувати та профілювати код, що особливо важливо у системах, орієнтованих на стабільність і надійність. VS 2022 оптимізоване для роботи з великими проєктами та підтримує різні модулі – від роботи з базами даних до створення графічних інтерфейсів, що робить його універсальним інструментом для реалізації складних рішень. Крім того, наявність безкоштовної Community-версії забезпечує доступність для освітніх і наукових завдань, не знижуючи при цьому функціональності. Саме поєднання зручності, широкого функціоналу та можливостей інтеграції визначає доцільність вибору VS 2022 для реалізації даної системи.

Важливим є не лише вибір мови програмування та середовища розробки, а й визначення бібліотек машинного навчання, які забезпечують навчання та використання моделей. Саме вони формують основу обчислювальної логіки, дозволяючи реалізовувати алгоритми класифікації, регресії та аналізу даних. У випадку чат-ботів для надання довідкових матеріалів з першої медичної допомоги бібліотеки мають бути здатними швидко обробляти текстові запити,

забезпечувати інтеграцію з іншими інструментами та підтримувати сучасні методи штучного інтелекту.

Серед найбільш поширених бібліотек у середовищі C# варто виокремити ML.NET, Accord.NET та Encog [28]. Кожна з них має власні сильні сторони: ML.NET тісно інтегрована з екосистемою .NET і розрахована на розробників прикладних систем, Accord.NET пропонує широкий набір алгоритмів для аналізу сигналів і зображень, тоді як Encog орієнтована на моделювання нейронних мереж та навчання з різними підходами.

У табл. 2.4 наведено порівняння ключових характеристик цих інструментів, які відображають їхню практичну придатність у завданнях класифікації користувацьких запитів і надання довідкових матеріалів.

Таблиця 2.4

Порівняння бібліотек машинного навчання для C#

Характеристика	ML.NET	Accord.NET	Encog
Інтеграція з .NET	Повна, нативна підтримка	Висока, але менш тісна	Середня
Простота використання	Висока, орієнтована на прикладних розробників	Середня, потребує глибшого налаштування	Середня, вимагає знань про AI-моделі
Продуктивність	Висока	Середня	Висока на невеликих наборах даних
Підтримка документації	Дуже добра, офіційна підтримка Microsoft	Обмежена, переважно спільнотою	Середня, наявні приклади та гайдлайни
Ліцензія та доступність	Безкоштовна, відкритий код	Безкоштовна, відкритий код	Безкоштовна, відкритий код

Аналіз порівняння бібліотек машинного навчання демонструє, що кожна з них має свою спеціалізацію та область ефективного застосування. Accord.NET є корисною для задач, де потрібна обробка сигналів чи зображень,

а Encog орієнтована на побудову нейронних мереж і складних моделей оптимізації. Водночас ML.NET вирізняється комплексною інтеграцією з екосистемою .NET, простотою використання та підтримкою широкого спектра алгоритмів, включно з тими, що найкраще підходять для класифікації текстових запитів.

Завдяки високій продуктивності та наявності детальної документації ML.NET дозволяє ефективно поєднувати роботу з даними, навчання моделей і їх застосування безпосередньо в середовищі C#. Саме це робить його найбільш доцільним вибором для реалізації інтелектуального телеграм-бота, орієнтованого на надання довідкових матеріалів з першої медичної допомоги.

РОЗДІЛ 3.
ПРОЄКТУВАННЯ, РОЗРОБКА ТА ДОСЛІДЖЕННЯ ІНТЕЛЕКТУАЛЬНОГО ТЕЛЕГРАМ-БОТА

3.1. Формалізація задачі надання довідкових матеріалів з першої медичної допомоги

Формалізація задачі передбачає опис вхідних, вихідних даних та логіки взаємодії компонентів системи, що забезпечують надання користувачеві довідкової інформації з питань першої медичної допомоги через Telegram-бота. В основу моделі покладено взаємодію користувача з інтелектуальним ботом, який реалізує функції пошуку, відбору та представлення релевантних медичних рекомендацій, що зберігаються у базі знань системи або генеруються на основі моделей машинного навчання.

Задача може бути формально подана як функція:

$$R = F(Q, U, M, D), \tag{3.1}$$

де:

Q – текстовий запит користувача у Telegram-чаті;

U – множина зареєстрованих користувачів із відповідними атрибутами автентифікації;

M – множина моделей машинного навчання, що використовуються для класифікації запитів і генерації відповідей;

D – база знань, яка містить структуровані довідкові матеріали щодо надання першої допомоги;

R – відповідь системи у вигляді релевантної інформації або поради користувачеві.

Побудова Telegram-бота потребує деталізації цієї загальної функції через систему окремих задач, що забезпечують її реалізацію на практичному рівні. Кожна задача відповідає окремому компоненту системи та спрямована

на забезпечення цілісності процесу – від ідентифікації користувача до генерації готової довідкової відповіді. Відповідно до цього, необхідно реалізувати такі основні завдання системи:

– реалізувати механізм реєстрації та автентифікації користувачів. Система має забезпечити можливість створення облікових записів, перевірки автентичності користувачів та збереження інформації про них у базі даних;

– розробити підсистему керування користувачами. Адміністратор повинен мати змогу переглядати каталог користувачів, додавати нових, редагувати їхні дані та контролювати доступ до функціоналу Telegram-бота;

– створити модуль керування тематичними довідками. Необхідно реалізувати можливість створення, редагування та відображення тем, які охоплюють ключові аспекти першої допомоги: серцево-легеневу реанімацію, кровотечі, опіки, травми тощо. Контент повинен мати структуру, що дозволяє швидке оновлення і розширення;

– розробити модуль інтеграції моделей машинного навчання. Система повинна підтримувати додавання, тестування та оновлення моделей, призначених для класифікації запитів користувачів і визначення намірів;

– реалізувати Telegram-інтерфейс для інтерактивної взаємодії. Telegram-бот має приймати запити природною мовою, проводити попередню обробку тексту, надсилати його на обрану ML-модель і повертати користувачеві релевантну відповідь у зручному форматі;

– забезпечити збір статистики, відгуків і подій. Система повинна автоматично фіксувати звернення користувачів, збирати відгуки щодо якості наданих матеріалів та вести журнал подій для подальшого аналізу ефективності роботи моделей і системи в цілому.

3.2. Проєктування бази даних

Проєктування бази даних передбачає створення логічної структури, яка відображає основні сутності системи Telegram-бота для надання довідкових

матеріалів з першої медичної допомоги та взаємозв’язки між ними. База даних має забезпечувати збереження інформації про користувачів, теми, моделі машинного навчання, історію запитів, події та відгуки.

На цьому етапі формується концептуальна схема, у якій визначаються сутності, атрибути, первинні та зовнішні ключі, а також типи зв’язків між таблицями. Структура має забезпечувати цілісність даних і підтримку всіх функціональних можливостей системи, зокрема реєстрації користувачів, оброблення запитів, збирання статистики та керування контентом.

Відповідно до поставлених вимог, у базі даних передбачено такі основні таблиці:

Таблиця «Categories» призначена для зберігання відомостей про тематичні розділи, що використовуються у системі Telegram-бота. Кожен запис у таблиці відповідає окремій категорії довідкових матеріалів, наприклад «Серцево-легенева реанімація», «Опіки», «Кровотечі» тощо. Ця таблиця забезпечує структуровану організацію контенту, дозволяючи системі швидко визначати, до якої теми належить запит користувача або довідковий запис.

Таблиця 2.1

Атрибути сутності «Categories»

№	Найменування	Тип даних	Призначення
1	CategoriesId	INT	Первинний ключ, який однозначно ідентифікує категорію в системі
2	CategoriesName	NVARCHAR(250)	Назва категорії, що використовується для відображення теми у довідковому каталозі
3	Description	NVARCHAR(MAX)	Текстовий опис категорії, який може містити розширену інформацію або пояснення щодо змісту теми

Таблиця «Users» використовується для зберігання інформації про користувачів системи Telegram-бота, включаючи як звичайних користувачів, так і адміністраторів.

Таблиця 2.2

Атрибути сутності «Users»

№	Найменування	Тип даних	Призначення
1	UsersId	INT	Ідентифікує користувача в системі
2	FirstName	NVARCHAR(50)	Ім’я користувача
3	LastName	NVARCHAR(50)	Прізвище користувача
4	UsersName	NVARCHAR(50)	Унікальне ім’я користувача для входу
5	UsersPassword	NVARCHAR(50)	Пароль користувача
6	RoleId	INT	Ідентифікатор ролі користувача, що визначає його рівень доступу
7	Description	NVARCHAR(1000)	Додаткові відомості про користувача
8	Email	NVARCHAR(150)	Електронна адреса користувача

Таблиця «ChatWithBot» призначена для фіксування діалогів між користувачем і Telegram-ботом. У ній зберігаються запитання, що надсилаються користувачами, відповіді, сформовані системою, а також зв’язок із відповідною тематичною категорією.

Таблиця 2.3

Атрибути сутності «ChatWithBot»

№	Найменування	Тип даних	Призначення
1	ChatWithBotId	INT	Ідентифікатор запису діалогу між користувачем і ботом
2	ChatId	BIGINT	Ідентифікатор чату в Telegram, що пов’язує повідомлення з конкретним користувачем
3	Question	NVARCHAR (MAX)	Текст запитання, яке користувач надіслав боту
4	Answer	NVARCHAR (MAX)	Відповідь, сформована системою або отримана з бази знань
5	CategoriesId	INT	Посилання на категорію, до якої належить запит

Таблиця «Models» використовується для зберігання інформації про моделі машинного навчання, що застосовуються в Telegram-боті для оброблення користувацьких запитів і формування відповідей. Кожен запис у таблиці відповідає окремій моделі, пов’язаній із певною тематичною категорією або користувачем, який її створив.

Таблиця 2.4

Атрибути сутності «Models»

№	Найменування	Тип даних	Призначення
1	ModelsId	INT	Ідентифікатор моделі у системі
2	ModelsName	NVARCHAR(150)	Назва моделі машинного навчання, що використовується для генерації відповідей
3	CategoriesId	INT	Посилання на тематичну категорію
4	ModelsFileModel	NVARCHAR(MAX)	Шлях або ім’я файлу моделі
5	Description	NVARCHAR(MAX)	Текстовий опис призначення моделі
6	UsersId	INT	Ідентифікатор користувача, що створив модель

Таблиця «Logs» призначена для ведення журналу подій, що відбуваються у системі Telegram-бота. У ній фіксуються дії користувачів, час їх виконання та короткий опис події.

Таблиця 2.5

Атрибути сутності «Logs»

№	Найменування	Тип даних	Призначення
1	LogsId	INT	Ідентифікатор події у журналі системи
2	UsersId	INT	Ідентифікатор користувача, що ініціював подію
3	EventNameShow	NVARCHAR(MAX)	Короткий опис події
4	EventDate	DATETIME	Дата та час виконання події у системі

Таблиця «LeavingFeedback» призначена для зберігання відгуків користувачів про роботу Telegram-бота та якість наданих довідкових матеріалів. Кожен запис у цій таблиці містить текст повідомлення від користувача, дату його надсилання, а також дані, що дозволяють ідентифікувати автора.

Таблиця 2.6

Атрибути сутності «LeavingFeedback»

№	Найменування	Тип даних	Призначення
1	LeavingFeedbackId	INT	Первинний ключ, який ідентифікує відгук користувача в системі
2	ChatId	BIGINT	Ідентифікатор Telegram-чату, з якого було надіслано відгук
3	Message	NVARCHAR(MAX)	Текст повідомлення або коментаря, залишеного користувачем
4	FirstName	NVARCHAR(150)	Ім’я користувача, який залишив відгук, для відображення у звітах чи аналітиці
5	LastName	NVARCHAR(150)	Прізвище користувача, що дозволяє ідентифікувати автора повідомлення
6	LeavingFeedbackDate	DATETIME	Дата та час надсилання відгуку користувачем
7	UsersId	INT	Ідентифікатор користувача, пов’язаний із таблицею Users; використовується для встановлення зв’язку між відгуком і зареєстрованим користувачем

Після логічного проєктування структури бази даних було здійснено побудову її фізичної моделі, яка реалізує зв’язки між основними сутностями системи Telegram-бота: користувачами, категоріями, чатами, моделями, журналом подій та відгуками. Фізичне проєктування виконувалося у середовищі MS SQL Server Management Studio, де для кожної сутності створено відповідні таблиці з визначенням первинних і зовнішніх ключів,

типів даних та обмежень цілісності. На цьому етапі сформовано структуру, що забезпечує правильну взаємодію між об’єктами бази даних і гарантує логічну узгодженість інформації.

Усі таблиці – Users, Categories, ChatWithBot, Models, Logs та LeavingFeedback – взаємопов’язані через ключові атрибути, що дозволяє системі коректно обробляти запити, реєструвати події, зберігати відповіді ботів і забезпечувати аналітику взаємодій користувачів.

Фізична модель бази даних зображена на рис. 3.1, побудована на основі наведених вище таблиць і демонструє взаємозв’язки між ключовими сутностями системи.

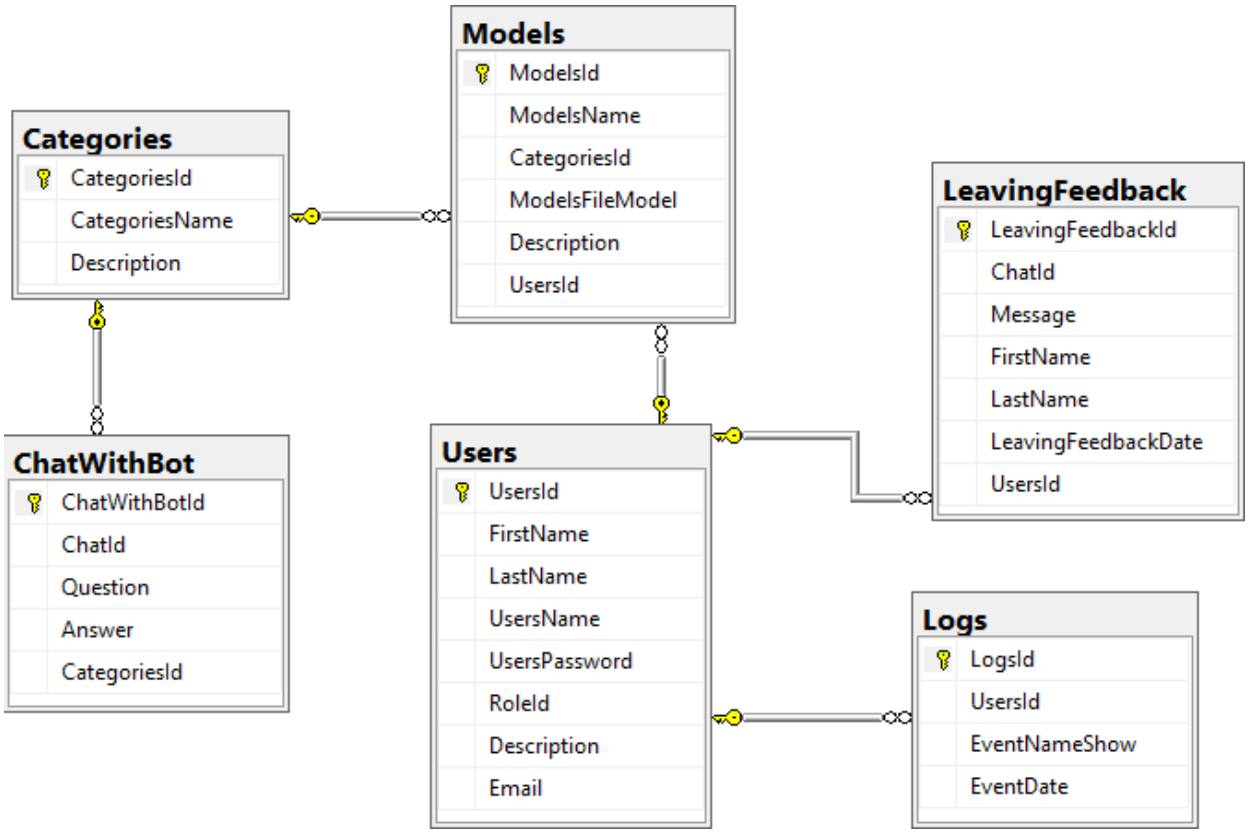

Рис. 3.1. Фізична модель бази даних системи

Наведена модель відображає структуру зберігання даних у вигляді реальних об’єктів бази, що дає змогу реалізувати цілісну архітектуру застосунку. Вона забезпечує ефективну взаємодію між модулями системи,

спрощує виконання запитів і гарантує узгоджене функціонування всіх компонентів Telegram-бота.

3.3. Проєктування ключових бізнес–процесів

На етапі проєктування ключових бізнес-процесів виконується моделювання основних сценаріїв взаємодії користувачів із системою Telegram-бота. Цей процес передбачає ідентифікацію ролей користувачів, визначення їхніх функціональних можливостей та побудову графічних моделей, що відображають логіку виконання операцій у системі. Таке представлення дозволяє формалізувати поведінку системи, встановити взаємозв’язки між діями користувачів і процесами, а також визначити межі відповідальності кожного учасника.

Для відображення функціональної поведінки системи застосовуються діаграми варіантів використання. Вони дають змогу візуалізувати основні сценарії взаємодії користувачів із системою, де кожен прецедент описує певну дію або функціональну можливість. На рис. 3.2 представлено діаграму прецедентів для ролі адміністратора системи Telegram-бота.

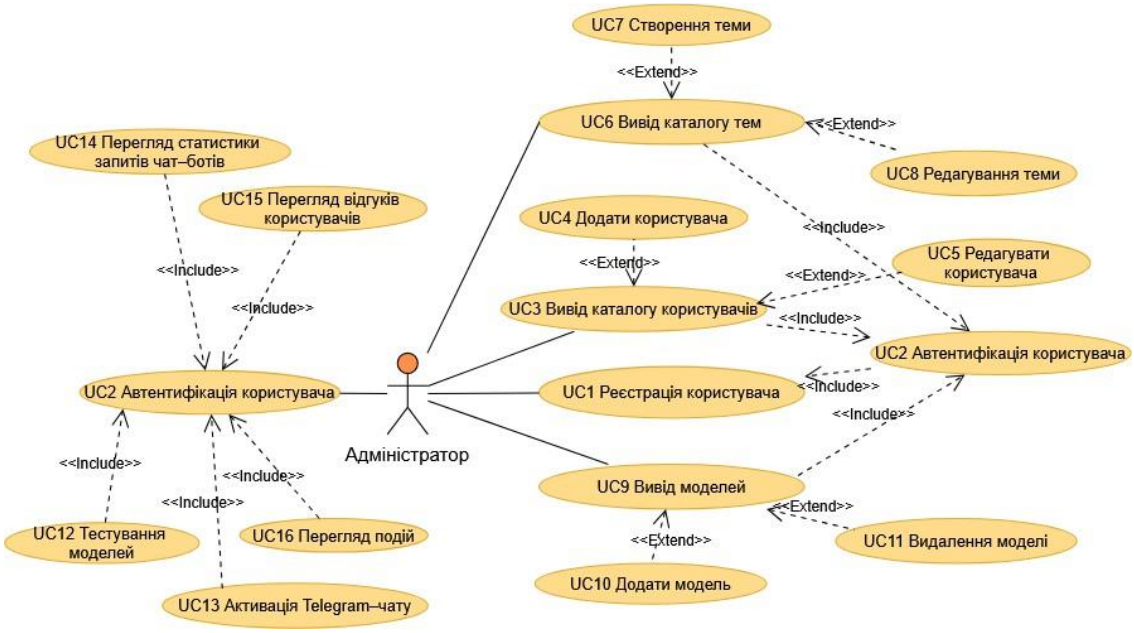

Рис. 3.2. Діаграма прецедентів для ролі «Адміністратор»

Діаграма ілюструє основні дії, які може виконувати адміністратор: реєстрацію та автентифікацію користувачів, керування темами, моделюванням, тестуванням моделей, перегляд статистики запитів, відгуків користувачів і подій системи. На діаграмі також показано відношення include та extend, що описують взаємозалежності між прецедентами, зокрема використання автентифікації як обов'язкової складової більшості адміністративних дій.

На рис. 3.3 зображено діаграму прецедентів для ролі «Користувач», яка відображає основні можливості взаємодії звичайного користувача із системою Telegram-бота. Діаграма демонструє логіку виконання дій, спрямованих на отримання довідкових матеріалів з першої медичної допомоги, тестування моделей та участь у процесі зворотного зв'язку.

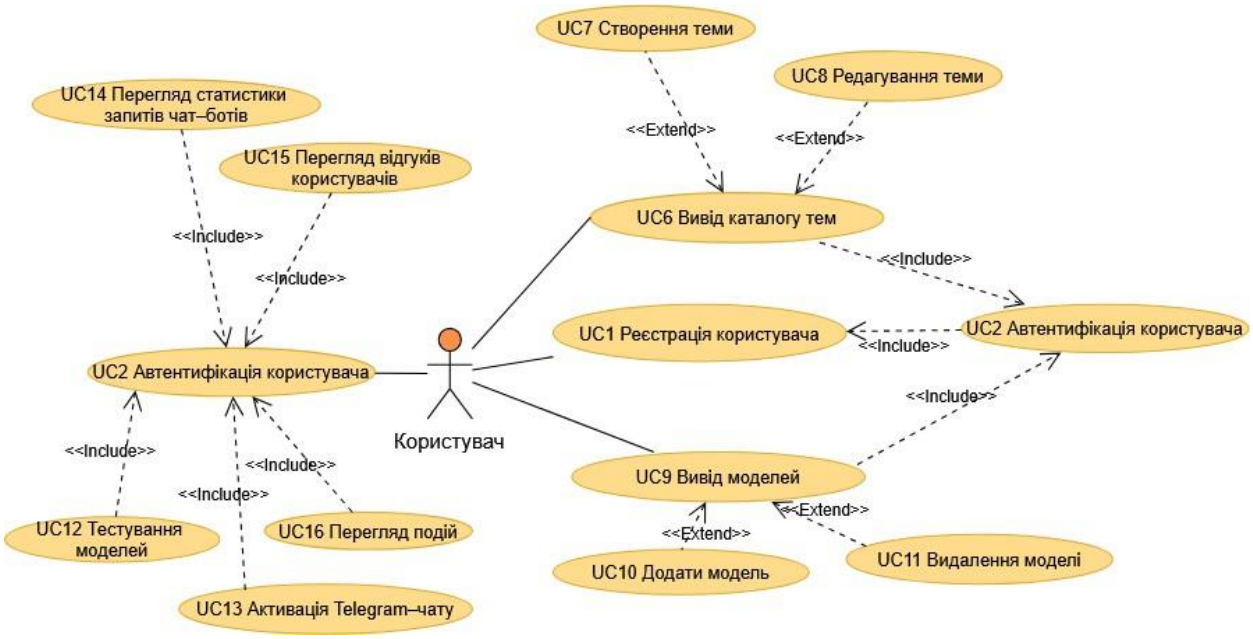

Рис. 3.3. Діаграма прецедентів для ролі «Користувач»

Користувач має змогу здійснити реєстрацію та автентифікацію, після чого отримує доступ до каталогу тем, перегляду моделей, а також активації Telegram-чату для безпосередньої взаємодії з ботом. Крім основних сценаріїв, користувач може тестувати доступні моделі машинного навчання,

переглядати події системи, залишати відгуки про роботу чат-бота і знайомитись зі статистикою запитів.

На етапі проєктування бізнес-процесів, окрім діаграм прецедентів, доцільно використовувати діаграми послідовності (sequence diagrams), які дозволяють відобразити динамічну взаємодію між об’єктами системи у часовій послідовності. Вони показують, як саме здійснюється обмін повідомленнями між користувачем, інтерфейсом, внутрішніми модулями та зовнішніми компонентами під час виконання певного сценарію.

На рис. 3.4 подано діаграму послідовності, що відображає процес взаємодії користувача з Telegram-ботом у межах системи надання довідкових матеріалів з першої медичної допомоги.

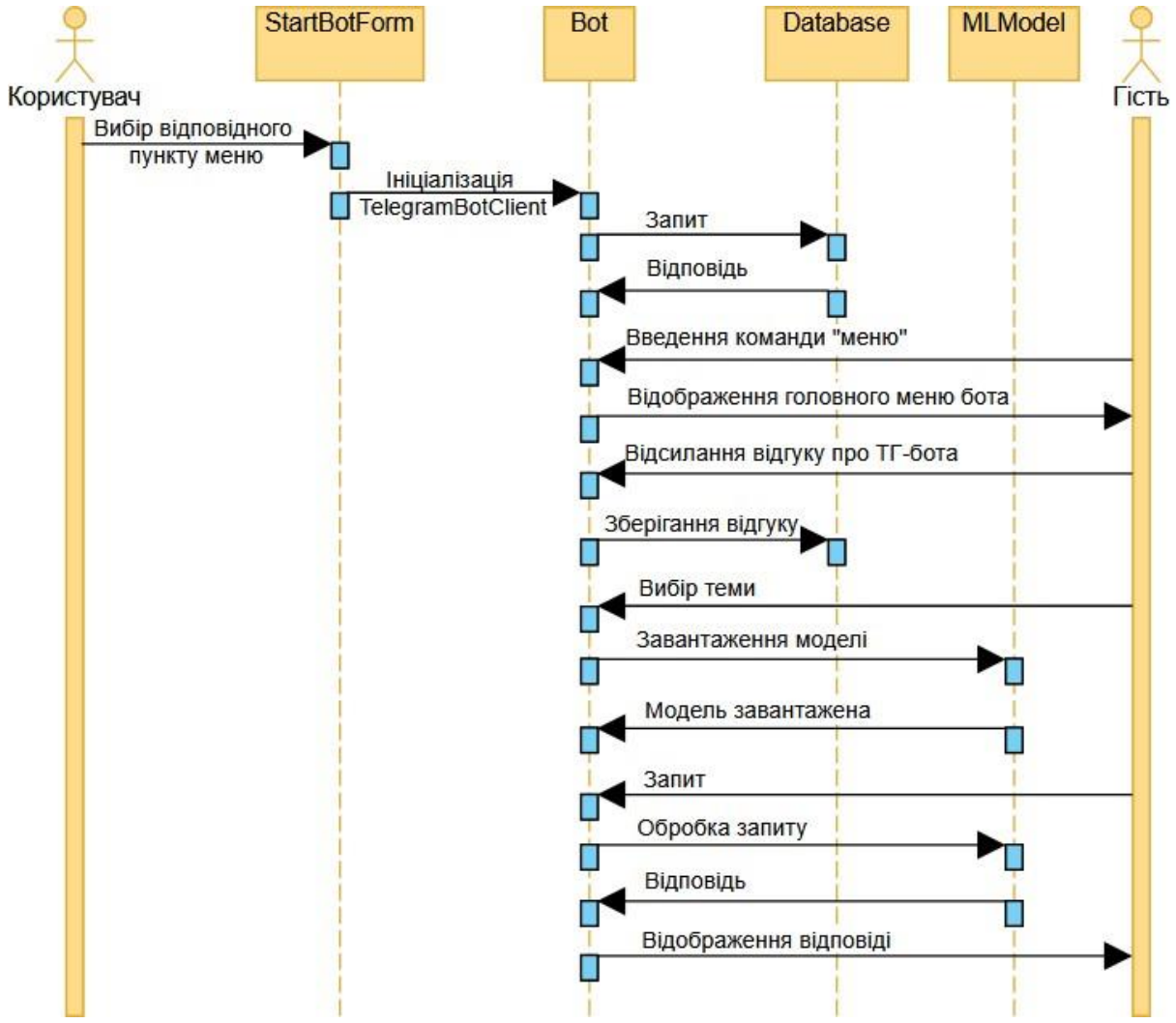

Рис. 3.4. Діаграма послідовності взаємодії користувача з ботом

У наведеній діаграмі послідовності користувач розпочинає роботу з Telegram-ботом через форму StartBotForm, яка виступає стартовим інтерфейсом застосунку. На цьому етапі відбувається ініціалізація клієнта TelegramBotClient, що забезпечує підключення до платформи Telegram і встановлення каналу зв’язку між користувачем та серверною частиною системи. Після цього формується початковий запит, який надсилається до компонента Bot, що відповідає за логіку оброблення команд і маршрутизацію повідомлень.

Далі розгортається процес обміну даними між модулями Bot, Database та MLModel. Компонент Bot отримує введену користувачем команду, наприклад «меню», виконує запит до бази даних для отримання інформації про доступні функції або теми та надсилає користувачеві структуровану відповідь у вигляді головного меню Telegram-бота. Якщо під час сеансу користувач вирішує залишити відгук, повідомлення передається до бази даних, де воно зберігається в таблиці LeavingFeedback із фіксацією часу та ідентифікатора користувача.

Після перегляду доступних тем користувач має змогу обрати конкретну категорію, що ініціює запит до модуля MLModel для завантаження відповідної моделі машинного навчання. Модель опрацьовує текст запиту користувача, аналізує його зміст і формує релевантну відповідь, спираючись на базу знань або попередньо навчені дані. Після оброблення інформації результат повертається через компонент Bot, який відображає сформовану відповідь у чаті Telegram.

Таким чином, наведена діаграма демонструє повну логіку взаємодії користувача з ботом – від ініціації сеансу до отримання довідкової інформації. Вона відображає послідовність операцій, синхронізацію між компонентами та обмін даними між різними рівнями системи.

Рис. 3.5 висвітлює діаграму послідовності, що описує процес додавання нової теми у чат-бот через інтерфейс адміністратора. Вона відображає взаємодію між користувачем, формою введення даних (CategoriesForm),

модулем перевірки валідності (Validation), базою даних (Database) та журналом подій (Logs).

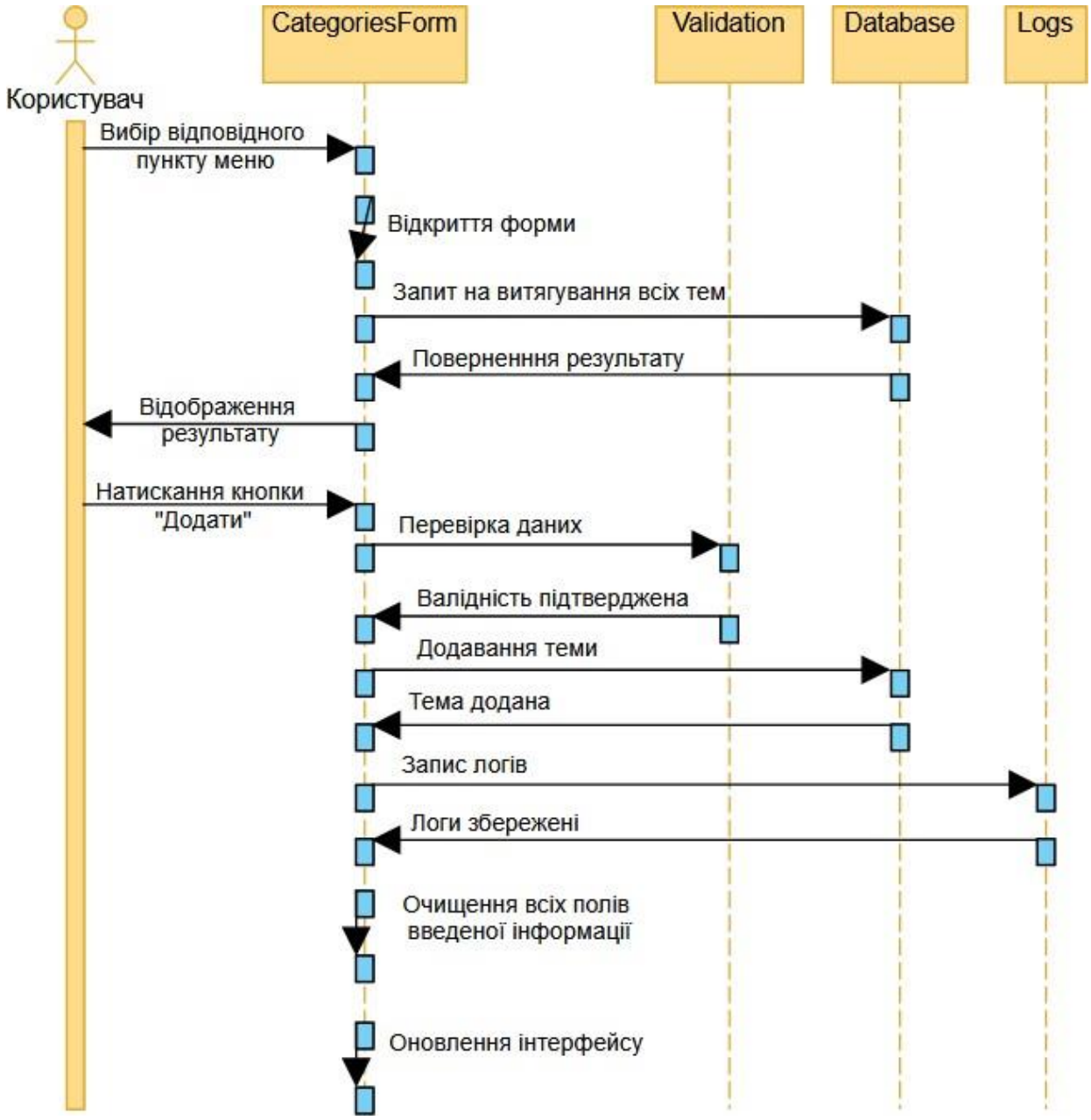

Рис. 3.5. Діаграма послідовності для додавання теми у чат–бот

Послідовність дій розпочинається з вибору користувачем відповідного пункту меню, після чого відбувається відкриття форми додавання теми. Система виконує запит до бази даних для отримання наявного переліку тем і повертає результати для відображення. Після цього користувач натискає

кнопку «Додати», що ініціює перевірку введених даних модулем Validation. У разі успішної валідації підтверджується коректність інформації, і система виконує операцію додавання нової теми до бази даних.

Після збереження теми виконується запис події в журнал (Logs), що дозволяє зафіксувати факт додавання нового елемента та забезпечити контроль за діями користувача. Після завершення операції система очищає всі поля введеної інформації та оновлює інтерфейс для відображення актуальних даних.

3.4. Архітектура програмного рішення

Архітектура програмного рішення визначає логічну структуру системи Telegram-бота для надання довідкових матеріалів з першої медичної допомоги, а також принципи взаємодії між її компонентами. На цьому етапі формується узагальнена модель побудови програмного забезпечення, що забезпечує узгоджену роботу користувацького інтерфейсу, серверної логіки та сховища даних. Така архітектура дозволяє структурувати систему на незалежні частини, спрощуючи процеси розроблення, тестування та подальшого супроводу.

Для реалізації Telegram-бота обрано трьохрівневу архітектуру (three-tier architecture), яка передбачає поділ системи на три логічні шари:

– рівень користувацького інтерфейсу (Presentation Layer) – відповідає за взаємодію з користувачем через інтерфейс Telegram і форми керування;

– рівень бізнес-логіки (Business Logic Layer) – реалізує основні функціональні процеси системи;

– рівень даних (Data Access Layer) – забезпечує доступ до бази даних, виконання запитів і збереження інформації про користувачів, запити, моделі та відгуки.

Вибір трьохрівневої архітектури зумовлений її високою гнучкістю, масштабованістю та розподілом відповідальності між частинами системи.

Така структура дозволяє легко оновлювати або замінювати окремі модулі без порушення роботи всього застосунку, забезпечує стабільність під час інтеграції нових функцій і полегшує супровід програмного продукту на всіх етапах життєвого циклу.

Для забезпечення взаємодії між рівнем бізнес-логіки та базою даних у проєктованій системі реалізовано рівень доступу до даних, який інкапсулює всі операції з таблицями бази даних і забезпечує централізоване керування запитами. Цей рівень побудовано на основі окремих класів-провайдерів, кожен з яких відповідає за роботу з певною сутністю. Такий підхід дозволяє знизити зв'язність компонентів, полегшує оновлення структури даних і забезпечує єдиний механізм взаємодії з джерелом інформації.

На рис. 3.6 подано діаграму класів рівня даних, яка демонструє взаємозв'язки між основними класами-провайдерами системи Telegram-бота для надання довідкових матеріалів з першої медичної допомоги.

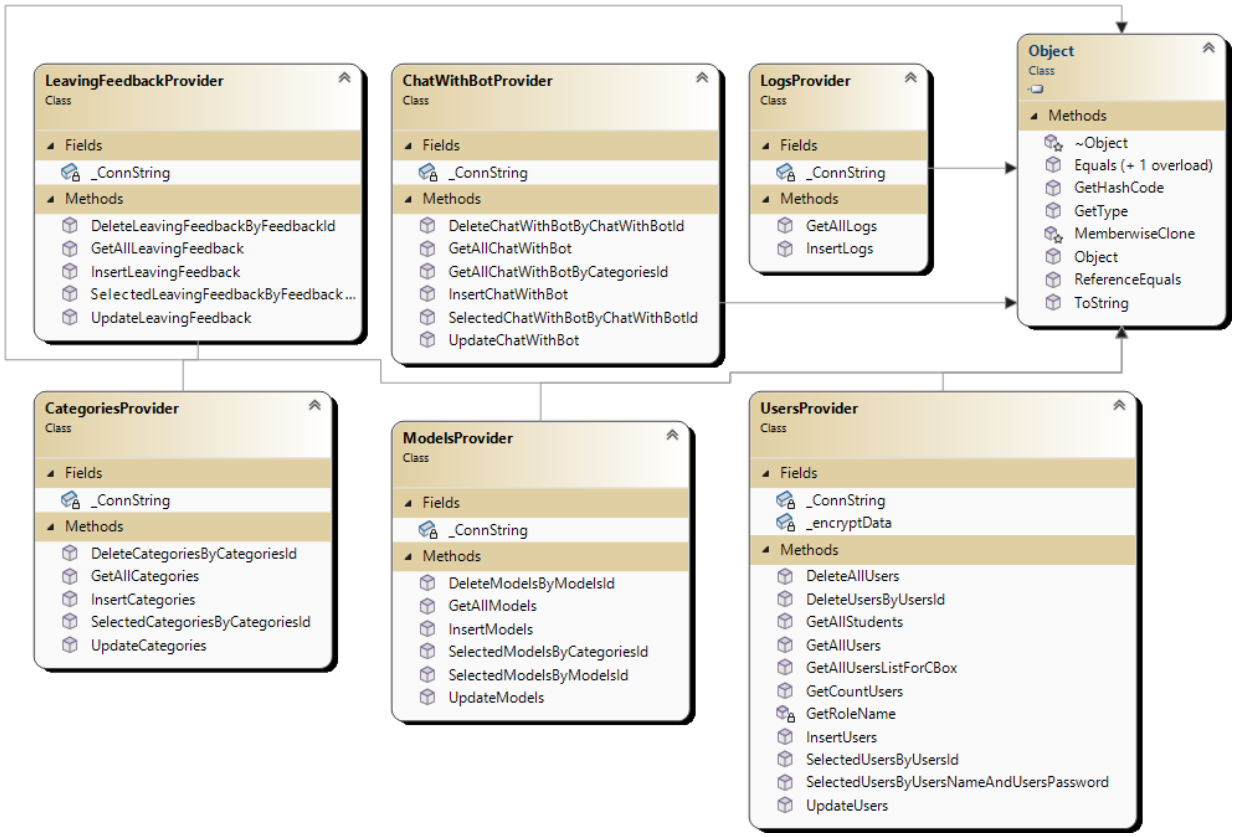

Рис. 3.6. Діаграма класів рівня даних

Діаграма класів рівня даних складається із таких основних класів:

– клас UsersProvider – відповідає за доступ до даних про користувачів системи. Реалізує методи для отримання списків користувачів, пошуку за ідентифікатором або обліковими даними, додавання, оновлення та видалення записів. Також забезпечує підтримку шифрування паролів (encryptData) для підвищення безпеки;

– клас CategoriesProvider – призначений для роботи з довідником тематичних категорій. Містить методи для отримання всіх категорій, вибірки за ідентифікатором, вставлення нових тем і оновлення існуючих записів;

– клас ChatWithBotProvider – забезпечує взаємодію з таблицею діалогів користувачів і бота. Реалізує методи для отримання історії чатів, пошуку повідомлень за ідентифікатором, додавання нових діалогів і оновлення відповідей бота;

– клас ModelsProvider – керує даними про моделі машинного навчання, що застосовуються у системі. Містить методи для завантаження переліку моделей, вибору конкретної моделі за ідентифікатором, додавання нових файлів моделей, їх оновлення або видалення;

– клас LogsProvider – відповідає за ведення журналу подій системи. Забезпечує запис інформації про виконані дії користувачів і адміністраторів, а також отримання повного переліку логів для подальшого аналізу;

– клас LeavingFeedbackProvider – використовується для роботи з відгуками користувачів. Дозволяє додавати, оновлювати, вилучати та переглядати повідомлення з таблиці LeavingFeedback, а також здійснювати вибірку за конкретним ідентифікатором.

Усі вищенаведені класи успадковують базову функціональність від об’єкта Object, який містить базові методи системи .NET, які необхідні для правильної інтеграції у середовище виконання. Така побудова забезпечує логічну структурованість рівня даних, модульність і легкість розширення при подальшому розвитку Telegram-бота.

На рис. 3.7 наведено діаграму класів рівня користувацького інтерфейсу системи, яка демонструє взаємозв’язки між усіма формами, що забезпечують функціональність Telegram-бота.

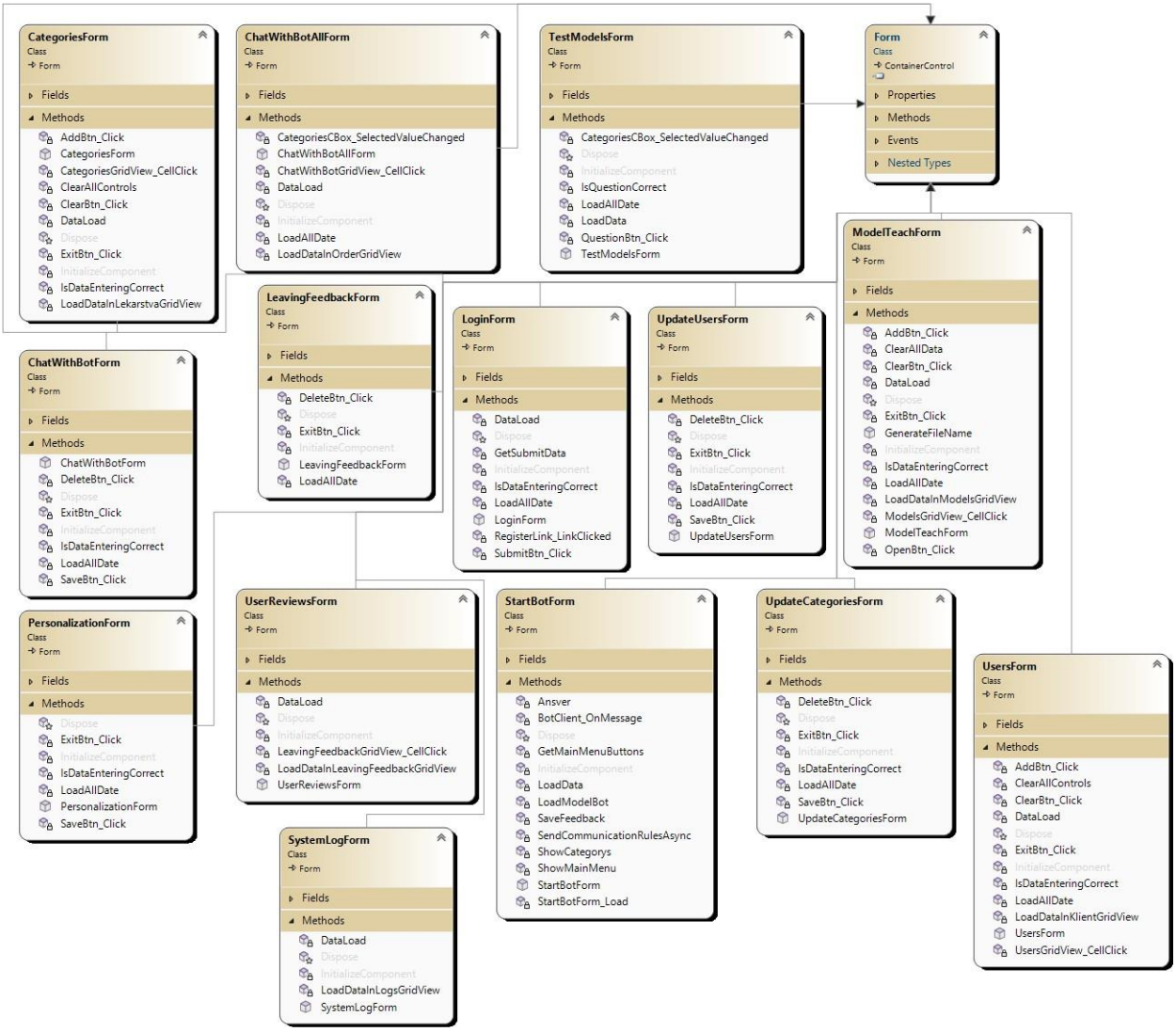

Рис. 3.7. Діаграма класів рівня користувацького інтерфейсу системи

Діаграма класів рівня користувацького інтерфейсу складається із таких основних класів:

– клас StartBotForm – є головною формою, з якої запускається Telegram-бот. Відповідає за ініціалізацію клієнта TelegramBotClient, оброблення вхідних повідомлень, відображення головного меню, завантаження моделей і передачу даних у відповідні модулі системи;

– клас CategoriesForm – використовується для відображення та управління категоріями довідкових матеріалів. Містить методи для додавання, редагування, оновлення та відображення списку категорій у таблиці;

– клас ChatWithBotForm – реалізує функції взаємодії з користувацькими діалогами Telegram-бота. Дає змогу переглядати історію повідомлень, фільтрувати записи, а також видаляти окремі діалоги;

– клас ChatWithBotAllForm – відображає повну історію спілкування між користувачами та ботом. Забезпечує пошук за категоріями або ідентифікаторами чатів і завантаження відповідних даних із бази;

– клас ModelTeachForm – призначений для навчання або оновлення моделей машинного навчання, що використовуються у Telegram-боті. Містить методи для вибору файлів моделей, генерації назв, завантаження даних у таблицю та збереження змін;

– клас TestModelsForm – забезпечує тестування моделей машинного навчання. Реалізує методи для завантаження моделі, вибору категорії, введення тестових запитів і відображення отриманих результатів;

– клас ModelsForm – відповідає за роботу з переліком моделей, включаючи їх додавання, редагування, оновлення або видалення. Сприяє підтримці актуальності моделей, використаних у процесі навчання бота;

– клас UsersForm – реалізує відображення зареєстрованих користувачів системи. Містить методи для додавання, редагування та видалення користувачів, а також перегляду даних у таблиці Users;

– клас UpdateUsersForm – використовується для оновлення відомостей про користувачів. Містить методи для перевірки введених даних, підтвердження змін і збереження результатів у базі даних;

– клас UserReviewsForm – відповідає за відображення відгуків користувачів, отриманих через Telegram-бота. Містить методи для завантаження повідомлень, фільтрації та перегляду вмісту таблиці LeavingFeedback;

– клас LeavingFeedbackForm – забезпечує оброблення відгуків користувачів. Реалізує методи для видалення, оновлення та перегляду отриманих повідомлень із бази даних;

– клас LoginForm – реалізує процедури реєстрації та входу до системи. Містить методи для перевірки введених даних, авторизації користувачів, а також переходу до головного інтерфейсу після успішної автентифікації;

– клас SystemLogForm – відповідає за відображення системних подій і журналу логів, що зберігаються у таблиці Logs. Дає змогу переглядати дії користувачів, час виконання операцій і зміст подій.

Отже, рівень користувацького інтерфейсу системи Telegram-бота побудований на принципах модульності та зручності взаємодії. Кожна форма відповідає за окремий функціональний напрям, що забезпечує логічну структурованість системи, спрощує навігацію та дозволяє легко розширювати функціонал без зміни основної архітектури застосунку.

На рис. 3.8 подано діаграму класів рівня бізнес-логіки системи Telegram-бота для надання довідкових матеріалів з першої медичної допомоги, яка відображає основні класи, методи й зв’язки між ними.

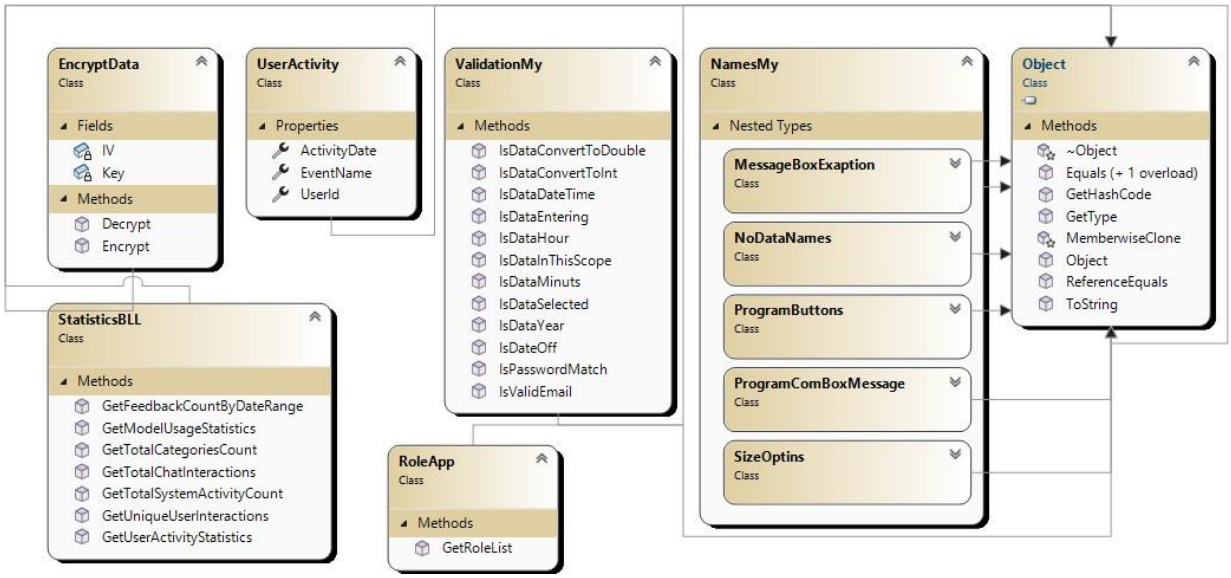

Рис. 3.8. Діаграма класів рівня бізнес-логіки системи Telegram-бота

Діаграма класів рівня бізнес-логіки складається із таких основних класів:

– клас EncryptData – відповідає за шифрування та дешифрування даних користувачів, зокрема паролів і конфіденційної інформації. Містить поля IV і Key, які використовуються для генерації криптографічних параметрів, та методи Encrypt і Decrypt, що забезпечують безпечне збереження даних у базі;

– клас UserActivity – використовується для відстеження дій користувачів у системі. Містить властивості ActivityDate, EventName та UserId, що дозволяють зберігати інформацію про події, які здійснював користувач, разом із датою їх виконання;

– клас ValidationMy – реалізує перевірку правильності введених користувачем даних. Містить широкий набір методів, які забезпечують валідацію введених значень, перевірку числових і текстових форматів, а також логічну коректність даних перед збереженням або обробленням;

– клас StatisticsBLL – виконує обчислення статистичних показників, що характеризують роботу системи. Реалізує методи, які дозволяють отримувати узагальнену інформацію про активність користувачів, використання моделей, кількість звернень і відгуків;

– клас RoleApp – забезпечує керування ролями користувачів у системі. Містить метод GetRoleList, який повертає перелік доступних ролей («Адміністратор» і «Користувач») та використовується при автентифікації й контролі доступу до функцій системи;

– клас NamesMy – містить вкладені допоміжні класи, що відповідають за стандартизацію назв, повідомлень і графічних елементів інтерфейсу.

Рівень бізнес-логіки формує основу інтелектуальної частини Telegram-бота, об’єднуючи функції безпеки, валідації, аналітики та управління ролями. Його структура забезпечує цілісність даних, правильну обробку запитів користувачів і стабільну взаємодію між інтерфейсом і базою даних. Завдяки чіткому поділу обов’язків цей рівень легко розширюється новими функціональними модулями без порушення роботи всієї системи.

3.5. Розробка та інтеграція моделей штучного інтелекту для класифікації запитів користувачів

Процес розробки та інтеграції моделей штучного інтелекту в Telegram-бот передбачає побудову алгоритмів, здатних автоматично розпізнавати зміст користувацьких запитів і визначати найбільш релевантні відповіді з бази знань. На цьому етапі реалізовано підсистему класифікації текстових повідомлень, яка працює на основі бібліотеки ML.NET. Модель аналізує вхідні фрази, що надходять від користувачів у Telegram-чаті, та зіставляє їх із попередньо навченою структурою даних, визначаючи, до якої категорії відноситься запит. Результатом роботи моделі є текстова відповідь або рекомендація, сформована відповідно до розпізнаної тематики.

Для реалізації зв'язку між Telegram-ботом та навченою моделлю необхідно визначити формат обміну даними. Цей формат описується спеціальними класами, які задають структуру вхідної та вихідної інформації, що використовуються під час навчання й прогнозування. На рис. 3.9 наведено фрагмент коду, який демонструє логіку опису цих структур даних у вигляді класів Input та Output.

```
// Input class
11 references
public class Input {
    [LoadColumn(0), ColumnName("Question")]
    public string Question;

    [LoadColumn(1), ColumnName("Answer")]
    public string Answer;
}
```

```
// Output class
4 references
public class Output {
    [ColumnName("Score")]
    public float[] Score;

    [ColumnName("PredictedLabel")]
    public uint PredictedLabel;

    [ColumnName("PredictedAnswer")]
    public string PredictedAnswer;
}
```

Рис. 3.9. Класи вхідних і вихідних даних для ML-моделі

У наведеному фрагменті визначено два класи – Input і Output, що забезпечують узгодженість між навчальною моделлю та програмним

середовищем Telegram-бота. Клас Input описує формат даних, які надходять на вхід моделі під час навчання або виконання прогнозу. Поля Question і Answer позначені атрибутами [LoadColumn] та [ColumnName], які визначають, з яких стовпців CSV-файлу або іншого джерела даних здійснюється завантаження інформації. Поле Question містить текст запиту користувача, тоді як Answer використовується як цільова змінна, що відповідає правильній або еталонній відповіді під час процесу навчання моделі.

Клас Output представляє результати роботи моделі після оброблення вхідного запиту. Поле Score містить набір числових значень, що відображають рівень імовірності належності запиту до кожного з можливих класів. Поле PredictedLabel зберігає ідентифікатор класу, який модель вважає найбільш ймовірним, а PredictedAnswer містить сформовану текстову відповідь, яку Telegram-бот надсилає користувачеві.

Наведений на рис. 3.10 фрагмент коду реалізує обробник події натискання кнопки «OpenBtn» у застосунку. Основне призначення цього методу полягає у відкритті файлу, який користувач обирає через стандартне діалогове вікно системи. Після активації кнопки створюється об’єкт класу OpenFileDialog, що забезпечує взаємодію з файловою системою.

```
private void OpenBtn_Click(object sender, EventArgs e) {
  // Створення діалогового вікна для відкриття файлу
  OpenFileDialog openFileDialog = new OpenFileDialog();
  // Налаштування властивостей діалогового вікна
  openFileDialog.Filter = "Text files (*.csv)|*.csv|All files (*.*)|*.*";
  openFileDialog.FilterIndex = 2;
  openFileDialog.RestoreDirectory = true;
  // Відображення діалогового вікна та обробка результату
  if (openFileDialog.ShowDialog() == DialogResult.OK) {
    try {
      _Path = openFileDialog.FileName;
      FileNameTBox.Text = openFileDialog.FileName;
```

Рис. 3.10. Обробник події натискання кнопки «OpenBtn»

Для зручності користувача задаються параметри фільтрації файлів, зокрема відображення лише файлів формату CSV або всіх доступних типів

файлів, а також встановлюється режим відновлення попереднього каталогу після завершення операції.

Після налаштування параметрів діалогове вікно відкривається методом ShowDialog. Якщо користувач підтверджує вибір файлу, програма перевіряє результат виконання діалогу і зчитує шлях до обраного файлу.

На рис. 3.11 наведено код процесу ініціалізації середовища машинного навчання та завантаження даних із CSV-файлу у внутрішню структуру програми. Створюється об’єкт MLContext, який слугує центральною точкою для роботи з інструментами ML.NET і забезпечує відтворюваність результатів завдяки фіксованому значенню параметра seed. Далі формується порожній список datas, призначений для зберігання колекції об’єктів типу Input, кожен з яких містить пару «запит – відповідь».

```
context = new MLContext(seed: 0);
List<Input> datas = new List<Input>();
using (StreamReader reader = new
  StreamReader(_Path, Encoding.GetEncoding(1251))) {
  while (!reader.EndOfStream) {
    var line = reader.ReadLine();
    var values = line.Split(';');
    Input item = new Input() { Question = values[0], Answer = values[1] };
    datas.Add(item);
  }
}
```

Рис. 3.11. Ініціалізація середовища МН та завантаження даних

Зчитування даних виконується за допомогою об’єкта StreamReader, який відкриває файл за шляхом _Path у кодуванні Windows-1251, що є типовим для кириличних текстів. Програма послідовно проходить усі рядки файлу до досягнення кінця потоку, використовуючи цикл while. Кожен рядок зчитується як текстовий рядок, після чого розділяється методом Split(';') на окремі елементи – запит (Question) та відповідь (Answer). Для кожного рядка створюється новий об’єкт класу Input, у поля якого записуються відповідні значення, після чого цей об’єкт додається до колекції datas.

У наведеному фрагменті на рис. 3.12 реалізовано підготовку корпусу запитів до формату, сумісного з конвеєром ML.NET. Колекція об’єктів datas, що містить пари «запит – відповідь», завантажується у внутрішнє представлення типу IDataView, завдяки чому надалі стає можливим застосування трансформацій і навчальних алгоритмів без додаткових перетворень. На цьому етапі фіксується структура схеми даних і забезпечується потокове опрацювання записів.

```
data = context.Data.LoadFromEnumerable<Input>(datas);
// Розділ даних на навчальний набір та тестовий набір
var trainTestData =
  context.Data.TrainTestSplit(data, testFraction: 0.2, seed: 0);
var trainData = trainTestData.TrainSet;
var testData = trainTestData.TestSet;
```

Рис. 3.12. Підготовка даних для навчання

Після завантаження даних виконується випадкове розбиття корпусу на навчальну та тестову вибірки методом TrainTestSplit з часткою тесту 0,2. Це означає, що 80 % записів використовуються для оцінки параметрів моделі, а 20 % — для незалежної перевірки якості класифікації. Встановлення seed = 0 гарантує відтворюваність розбиття між запусків, що дозволяє коректно порівнювати експерименти. У результаті формуються два об’єкти trainData та testData, які надалі передаються відповідно на етап навчання моделі та на етап підсумкового оцінювання її узагальнювальної здатності.

Фрагмент коду на рис. 3.13 реалізовує процесу машинного навчання – побудова та оцінювання моделі. Після формування конвеєра обробки даних, який містить усі необхідні перетворення та алгоритм класифікації, виконується його навчання на основі навчального набору даних. Це відбувається за допомогою методу Fit, який створює модель, що узагальнює закономірності у навчальних прикладах і формує вагові коефіцієнти для подальшого прогнозування.

```
model = pipeline.Fit(trainData);
// Оцінка моделі на тренувальному наборі даних
var trainPredictions = model.Transform(trainData);
var trainMetrics = context.MulticlassClassification.Evaluate(trainPredictions,
  labelColumnName: "Label", scoreColumnName: "Score");
```

Рис. 3.13. Реалізація процесу машинного навчання

Після навчання модель перевіряється безпосередньо на тому ж наборі даних, щоб оцінити її здатність відтворювати відомі результати. Для цього застосовується метод Transform, який генерує прогнозовані відповіді моделі для кожного прикладу з навчальної вибірки. Отримані результати зберігаються у змінній trainPredictions і використовуються для розрахунку метрик якості.

Оцінювання проводиться за допомогою функції Evaluate, яка належить до підмодуля MulticlassClassification бібліотеки ML.NET. У процесі розраховуються показники точності, повноти, F1-міри та інші статистичні характеристики, що дозволяють визначити, наскільки ефективно модель класифікує запити. У параметрах функції вказуються назви стовпців, які відповідають реальним міткам класів (Label) та оцінкам моделі (Score).

У наведеному на рис. 3.14 фрагменті коду реалізовано процес виведення результатів оцінювання навченої моделі на екран у текстовому полі звіту. Отримані під час попереднього етапу метрики ефективності моделі, збережені у змінній trainMetrics, використовуються для формування текстового звіту, що дозволяє користувачеві візуально оцінити якість навчання.

```
var predictions = model.Transform(testData);
var metrics = context.MulticlassClassification.Evaluate(predictions, "Label", "Score");
RaportTBox.Text += "Model evaluation metrics:" + "\r\n";
RaportTBox.Text += ($"Log-loss: {metrics.LogLoss:P2}") + "\r\n";
RaportTBox.Text += ($"Log-loss reduction: {metrics.LogLossReduction:P2}") + "\r\n";
RaportTBox.Text += ($"MicroAccuracy: {metrics.MicroAccuracy:P2}") + "\r\n";
RaportTBox.Text += ($"MacroAccuracy: {metrics.MacroAccuracy:P2}") + "\r\n";
```

Рис. 3.14. Виведення метрик для навчального набору

Спершу виводиться заголовок «Model evaluation metrics», який позначає, що наведені нижче значення стосуються саме навчального набору даних. Далі у звіті послідовно відображаються основні статистичні параметри моделі: Log-loss, який характеризує середню похибку класифікації; Log-loss reduction, що показує покращення моделі порівняно з базовим випадковим прогнозом; Macro accuracy – середня точність класифікації для всіх класів; та Micro accuracy, що оцінює точність на рівні всіх прикладів без урахування ваг класів.

Фрагмент коду на рис. 3.15 реалізує асинхронний метод UpdateHandler, який обробляє всі оновлення, що надходять від Telegram-сервера до клієнта бота. Метод приймає три параметри: об’єкт ITelegramBotClient bot, що представляє активний екземпляр клієнта Telegram; об’єкт Update, який містить дані про нову подію (наприклад, повідомлення або команду); та токен скасування CancellationToken ct, що використовується для контролю асинхронного виконання.

```
private async Task UpdateHandler(ITelegramBotClient bot,
  Update update, CancellationToken ct) {
  if (update == null) return;
  var msg = update.Message;
  if (msg == null) return;
  if (msg.Type != MessageType.Text) return;
  if (string.IsNullOrEmpty(msg.Text)) return;
  AppendLog("RX [" + msg.Chat.Id + "]: " + msg.Text);
  // лог у вікно
  RaportTBox.BeginInvoke((MethodInvoker)(() => {
    RaportTBox.AppendText(msg.Text + "\r\n");
    RaportTBox.ScrollToCaret();
  }));
```

Рис. 3.15. Виведення метрик для навчального набору

На початку роботи виконується послідовна перевірка отриманих даних для запобігання виникненню помилок. Якщо оновлення update не містить даних або не має вкладеного повідомлення, метод завершує виконання. Далі перевіряється тип повідомлення – обробляються лише ті, що мають текстовий формат, а також перевіряється, чи не є порожнім саме поле Text. Це дозволяє

фільтрувати службові або мультимедійні повідомлення, зосереджуючи обробку лише на текстовій взаємодії користувача з ботом.

Після перевірки коректності даних виконується логування отриманого повідомлення. Виклик методу AppendLog записує інформацію у внутрішній лог програми у форматі, що містить ідентифікатор чату та текст повідомлення. Таким чином забезпечується фіксація всієї активності користувачів для подальшого аналізу або діагностики роботи бота.

Для відображення отриманого тексту в інтерфейсі програми використовується елемент RaportTBox, який оновлюється за допомогою делегата BeginInvoke. Це дозволяє виконати операцію оновлення у межах головного потоку інтерфейсу, не блокуючи виконання асинхронного процесу. Повідомлення додається в кінець звіту, а курсор автоматично прокручується до останнього рядка, забезпечуючи відображення найновіших записів.

У наведеному фрагменті коду на рис. 3.16 реалізовано логіку роботи Telegram-бота в режимі збору відгуків від користувачів. Перевірка умови _IschatStart визначає, чи активовано цей режим – якщо так, програма переходить до обробки повідомлення як відгуку.

```
if (_IschatStart) {
  var first = (msg.From != null) ? msg.From.FirstName : null;
  var last = (msg.From != null) ? msg.From.LastName : null;
  SaveFeedback(msg.Chat.Id, msg.Text, first, last);
  await _BotClient.SendTextMessageAsync(msg.Chat.Id,
    "Дякуємо за відгук!", cancellationToken: ct);
  _IschatStart = false;
  return;
}
```

Рис. 3.16. Режим збору відгуків від користувачів

Після підтвердження, що бот перебуває у відповідному стані, з об’єкта msg.From зчитуються ім’я (FirstName) та прізвище (LastName) користувача, який надіслав повідомлення. Для уникнення можливих помилок перевіряється, чи об’єкт msg.From не є порожнім – якщо він існує, то дані ініціалізуються відповідними значеннями, інакше присвоюється null.

Далі викликається метод SaveFeedback, який зберігає отриманий відгук у базі даних разом з ідентифікатором чату, текстом повідомлення та інформацією про користувача. Таким чином забезпечується реєстрація всіх залишених відгуків для подальшого перегляду або аналізу адміністраторами системи. Після успішного збереження бот асинхронно надсилає користувачеві повідомлення подяки, використовуючи метод SendTextMessageAsync. Це створює інтерактивну взаємодію та підвищує рівень залученості користувача. На завершення змінна _IschatStart скидається у значення false, що сигналізує про вихід із режиму збору відгуків, і метод припиняє виконання, повертаючись до звичайного режиму роботи.

Фрагмент коду на рис. 3.17 реалізує логіку маршрутизації команд у Telegram-боті, яка визначає подальші дії системи залежно від отриманого текстового повідомлення від користувача.

```
var text = msg.Text.Trim();
switch (text) {
  case "меню":
  case "Меню":
    await ShowMainMenu(msg.Chat.Id, ct);
    break;
  case "Обрати тему допомоги":
    ShowCategorys(msg.Chat.Id); // async void – запускається, відповідь відправиться всередині
    break;
  case "Правила спілкування":
    await SendCommunicationRulesAsync(msg.Chat.Id, ct);
    break;
  case "Залишити відгук":
    await _BotClient.SendTextMessageAsync(msg.Chat.Id,
      "Будь-ласка, залиште відгук", cancellationToken: ct);
    _IschatStart = true;
    break;
  default:
    if (_IsCategoriesLoad) {
      Ansver(msg.Chat.Id, text);   // ML.NET прогноз + логування у БД
    } else {
      LoadModelBot(msg.Chat.Id, text); // вибір моделі по номеру категорії
    }
    break;
}
```

Рис. 3.17. Маршрутизація команд у Telegram-боті

На початку з тексту повідомлення видаляються зайві пробіли методом Trim, після чого оператор switch порівнює введене значення з відомими

командами, які підтримує бот. Такий підхід дозволяє чітко структурувати логіку роботи та спростити керування сценаріями взаємодії з користувачем.

Якщо повідомлення містить слово «меню» або «Меню», викликається асинхронний метод ShowMainMenu, який формує та надсилає головне меню Telegram-бота користувачеві. У разі вибору пункту «Обрати тему допомоги» виконується метод ShowCategorys, який відкриває перелік доступних тем довідкових матеріалів. Цей виклик здійснюється як async void, тобто його виконання відбувається у фоновому режимі, а надсилання відповіді користувачу здійснюється всередині самого методу.

Коли користувач обирає пункт «Правила спілкування», бот викликає асинхронний метод SendCommunicationRulesAsync, який надсилає повідомлення з коротким набором правил для коректної взаємодії в чаті. Якщо ж користувач натискає «Залишити відгук», бот надсилає запрошення ввести текст відгуку, встановлює змінну _IschatStart у значення true та переходить у режим очікування введення коментаря, який буде збережено після наступного повідомлення.

У випадку, коли введений текст не відповідає жодній із перелічених команд, виконується гілка default, яка визначає подальшу поведінку системи. Якщо прапорець _IsCategoriesLoad має значення true, тобто тема вже обрана, викликається метод Ansver, який здійснює прогнозування відповіді за допомогою моделі машинного навчання ML.NET і зберігає результат у базі даних. Якщо ж тема ще не вибрана, викликається метод LoadModelBot, який здійснює завантаження відповідної моделі на основі вибраної категорії.

3.6. Експериментальне дослідження та аналіз результатів

Експериментальна перевірка розробленого Telegram-бота здійснювалася з метою підтвердження його працездатності, функціональної повноти та відповідності поставленим вимогам. Дослідження проводилося у контексті практичного застосування інтелектуальних алгоритмів класифікації

для автоматизованого надання довідкових матеріалів з першої медичної допомоги. У ході тестування оцінювалися такі аспекти, як стабільність роботи програмного забезпечення, точність оброблення користувацьких запитів, швидкодія системи та коректність взаємодії між окремими її компонентами.

На рис. 3.18 наведено форму, призначену для управління темами довідкових матеріалів. Вона дозволяє здійснювати структуроване керування інформаційними категоріями, зокрема створення нових тем, редагування наявних і видалення застарілих записів.

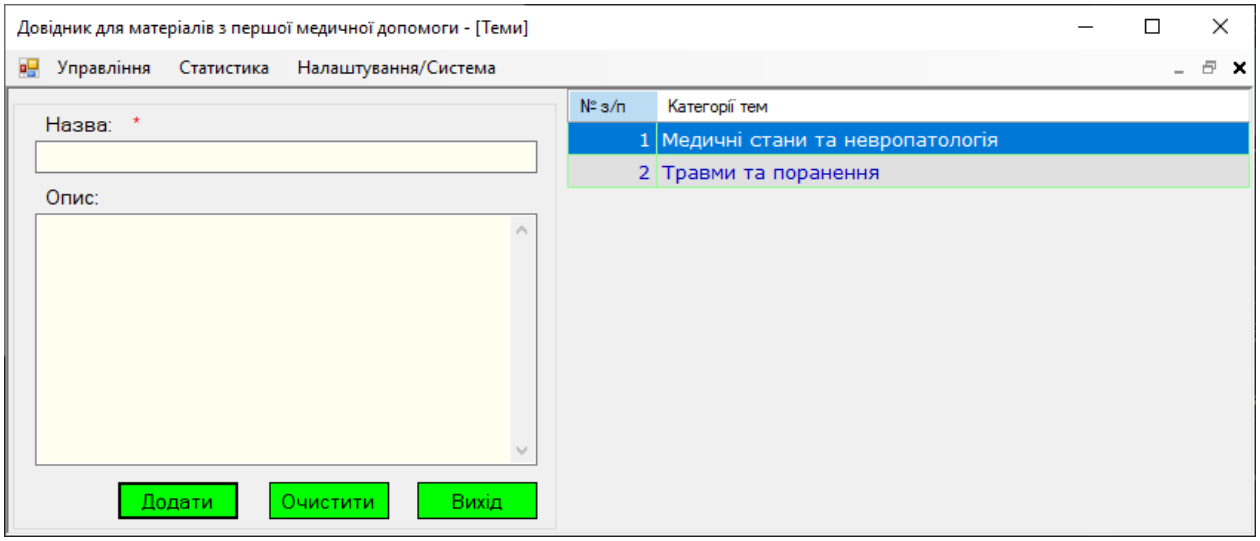

Рис. 3.18. Форма для управління темами довідкових матеріалів

Форма містить текстові поля для введення назви та опису теми, а також таблицю з переліком існуючих категорій. У результаті тестування встановлено, що інтерфейс забезпечує стабільну роботу всіх функцій, а операції додавання та оновлення виконуються без затримок. Зміни відображаються негайно, що підтверджує ефективну інтеграцію між клієнтською частиною програми та базою даних.

Рис. 3.19 відображає інтерфейс форми, що використовується для тренування моделі машинного навчання. У верхній частині форми розташовано елементи вибору файлу з навчальними даними, поле для відображення його шляху та текстову область для виведення результатів

процесу тренування. У центральній частині розміщено список доступних моделей із зазначенням їх назв, шляхів до збережених файлів і кнопкою для видалення застарілих версій. Нижня частина форми містить поля для вибору теми, введення назви моделі та короткого опису, а також кнопки для додавання, очищення або виходу з вікна.

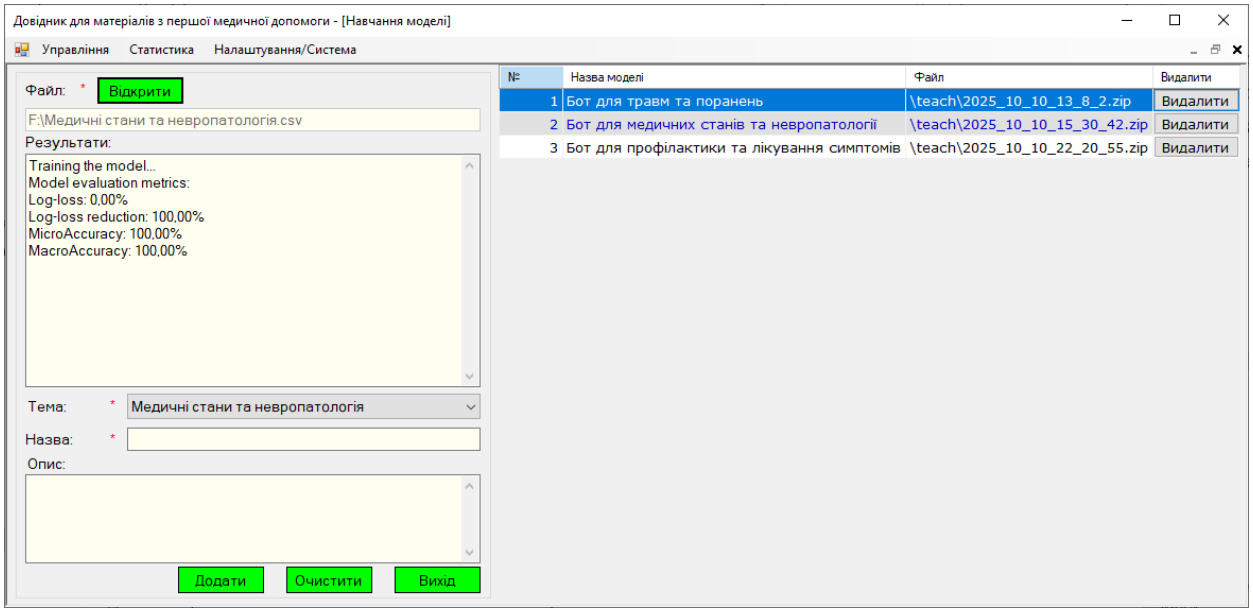

Рис. 3.19. Інтерфейс форми для тренування моделі

У ході тестування було підтверджено стабільну роботу всіх функціональних елементів форми. Процес навчання ініціюється після вибору файлу та супроводжується виведенням повідомлень про стан виконання, включно з показниками MicroAccuracy та MacroAccuracy, які дають змогу оцінити якість сформованої моделі. У тестовому запуску обидва показники сягнули 100%, що свідчить про повну відповідність навчальних даних очікуваним вихідним результатам. Збереження моделі у вигляді архіву дозволяє використовувати її повторно для прогнозування запитів користувачів у Telegram-боті.

На рис. 3.20 показано інтерфейс вікна для тестування моделі, яке призначене для введення користувацького запиту, його класифікації та відображення згенерованої відповіді. Ліва частина форми містить поля для

вибору теми обговорення та введення запитання, тоді як права частина відображає результати оброблення — знайдене питання з бази знань і відповідь, сформовану натренованою моделлю. Крім того, виводиться числове значення точності у відсотках, яке характеризує рівень впевненості моделі у своєму прогнозі.

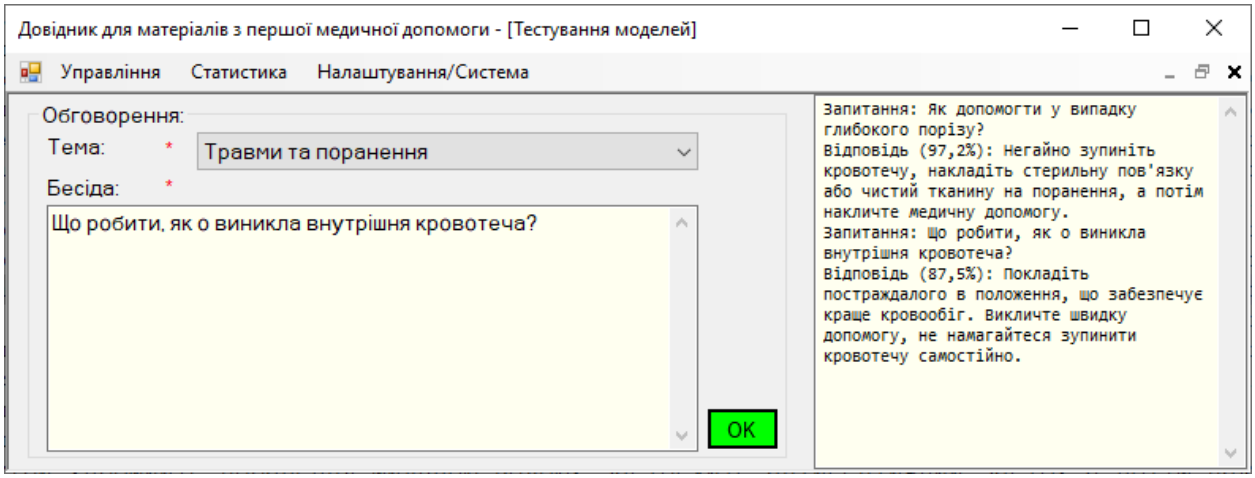

Рис. 3.20. Вікно для тестування моделі

У процесі тестування система продемонструвала високу стабільність і точність розпізнавання текстових запитів. Наприклад, при введенні запиту «Що робити, якщо виникла внутрішня кровотеча?» модель класифікувала його з упевненістю 87,5% і надала змістовну відповідь, що відповідає принципам першої медичної допомоги. Подібним чином, для запиту про глибокий поріз прогнозована відповідь мала впевненість 97,2%. Результати експериментів засвідчили, що алгоритм класифікації ефективно розпізнає формулювання, навіть якщо вони відрізняються від початкових прикладів у навчальному наборі. Система працює коректно, швидко генерує відповіді та демонструє здатність до узагальнення знань на основі отриманих навчальних даних.

Рис. 3.21 висвітлює основне меню чат-бота, яке відображається користувачеві після ініціації діалогу. Інтерфейс побудовано у вигляді трьох інтерактивних кнопок: «Обрати тему спілкування», «Залишити відгук» та «Правила спілкування». Такий підхід забезпечує інтуїтивну взаємодію з ботом

і дозволяє швидко перейти до необхідного режиму роботи без використання текстових команд.

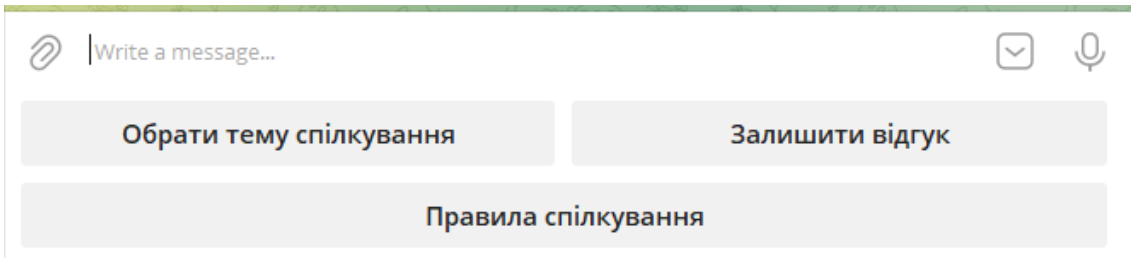

Рис. 3.21. Основне меню чат-бота

При виборі опції «Обрати тему спілкування» бот відображає перелік доступних категорій, що відповідають навчальним моделям, створеним у попередніх етапах роботи системи. Це дозволяє користувачеві обрати напрям, який його цікавить, наприклад «Травми та поранення» або «Медичні стани та невропатологія». Після вибору категорії бот переходить у режим оброблення запитів і готовий надати відповідь на будь-яке питання, що стосується обраної теми.

Кнопка «Залишити відгук» активує режим збору зворотного зв’язку. Користувач може ввести власне повідомлення, яке зберігається в базі даних для подальшого аналізу якості роботи системи. Опція «Правила спілкування» надає короткі рекомендації щодо коректного формулювання запитів та взаємодії з ботом.

Тестування показало, що Telegram-бот стабільно реагує на натискання кнопок, швидко формує відповіді та не допускає зависань навіть при інтенсивному обміні повідомленнями. Основне меню визнано зручним, логічно структурованим і придатним для використання користувачами без технічної підготовки.

На рис. 3.22 показано фрагмент діалогу, у якому користувач ініціює команду «Обрати тему допомоги». У відповідь бот відправляє відформатований список доступних тем із нумерацією, що дозволяє зручно

виконати вибір за допомогою введення відповідного номера. Після отримання вибору система підтверджує обрану тему повідомленням «Ви вибрали тему: Медичні стани та невропатологія», що сигналізує про успішне встановлення контексту для подальшої класифікації запитів.

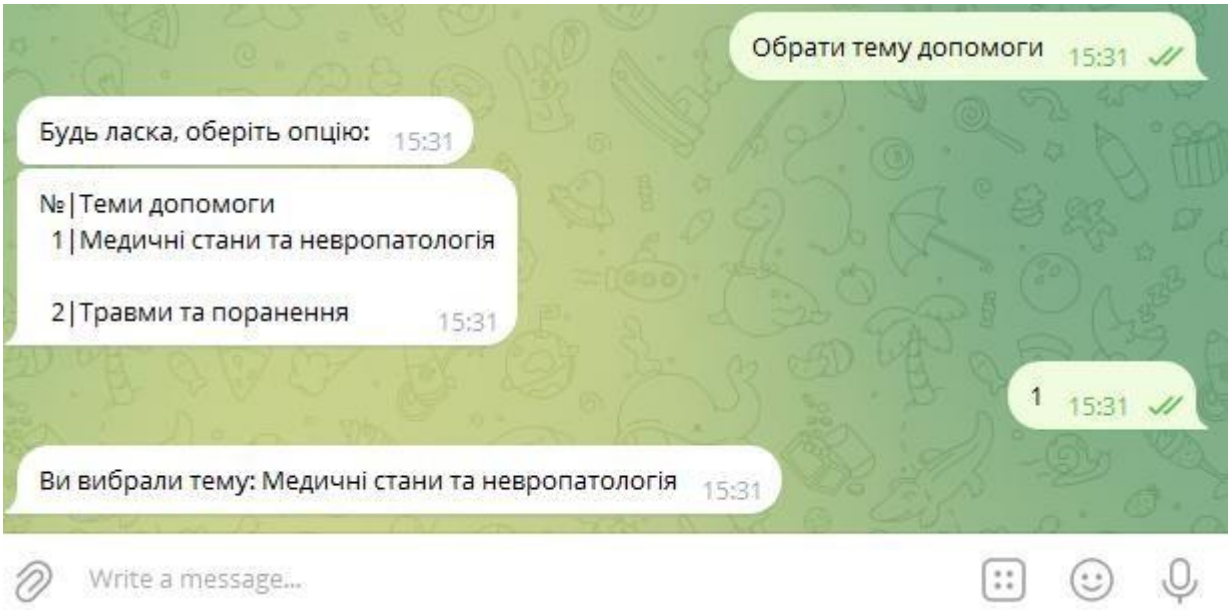

Рис. 3.22. Обрання медичної теми

Під час перевірки цього функціонального блоку було відзначено стабільність роботи механізму обробки повідомлень: бот швидко реагує на команди, не допускає дублювання запитів і коректно обробляє числові значення, навіть якщо вони вводяться з різними затримками. Крім того, система здатна адаптуватися до змін у базі даних тем — у разі додавання нових категорій список автоматично оновлюється без необхідності перезапуску бота. Такий підхід підтверджує правильність побудови логіки взаємодії між інтерфейсом користувача та модулями машинного навчання, що забезпечує цілісність і узгодженість роботи всієї системи.

Рис. 3.23 висвітлює приклад використання Telegram-бота під час симуляції запитів щодо надання першої медичної допомоги. Користувач звертається із запитаннями, пов’язаними з порізами, укусами або пошкодженнями шкіри, а бот оперативно надає відповідь, сформовану на

основі натренованої моделі ML.NET. Система не лише відтворює текстову рекомендацію, але й адаптує формулювання під контекст конкретної ситуації, що робить взаємодію максимально природною для користувача.

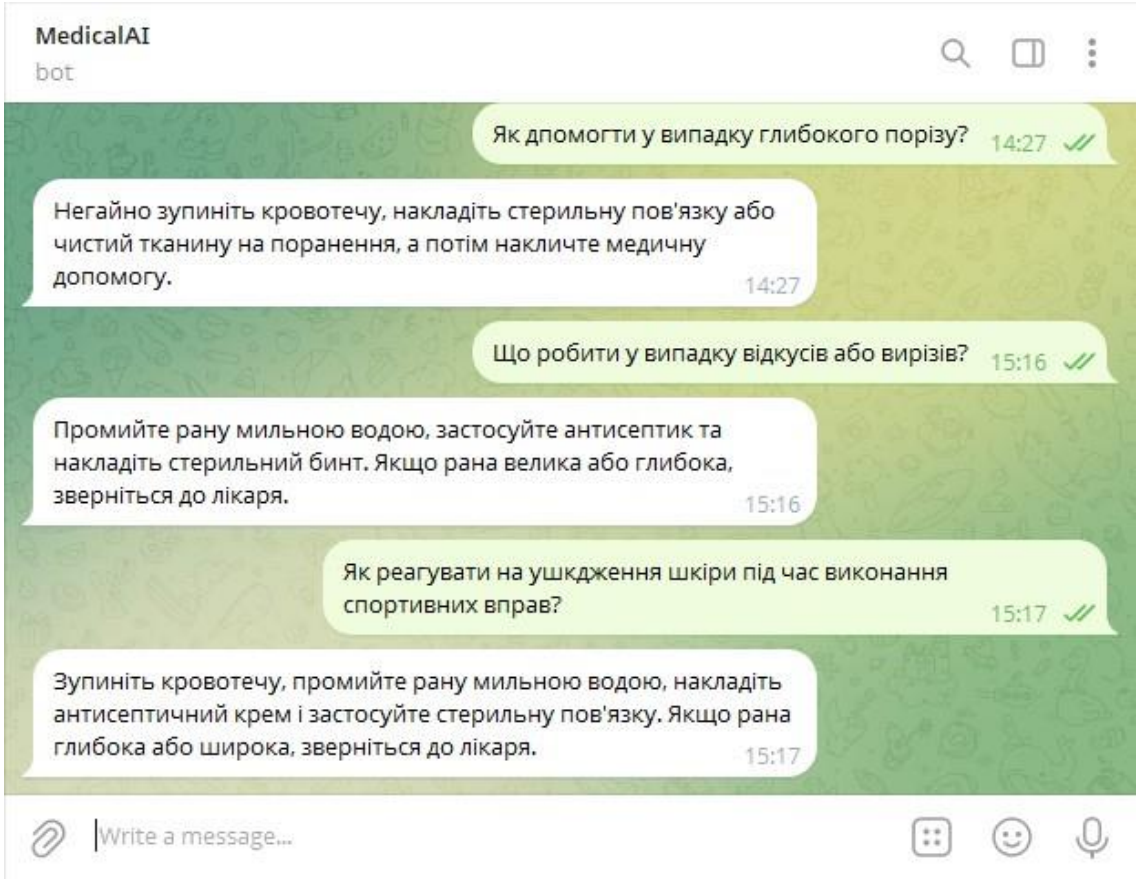

Рис. 3.23. Приклад використання Telegram-бота

Під час тестування виявлено, що бот правильно інтерпретує ключові слова запиту («глибокий поріз», «ушкодження шкіри»), знаходить найбільш відповідну відповідь і формує повідомлення без затримок. Це свідчить про ефективну інтеграцію навчальної моделі з механізмом оброблення текстових повідомлень Telegram API. Також підтверджено здатність системи до узагальнення знань: навіть при зміненому формулюванні запитання (наприклад, «як реагувати на ушкодження шкіри під час виконання спортивних вправ») бот надає релевантну пораду.

На рис. 3.24 подано приклад вікна, що демонструє правила користування чат-ботом. Після введення команди «Правила спілкування» система надсилає

користувачеві інформативне повідомлення, яке містить опис основних можливостей платформи, порядок навігації між розділами, інструкції щодо отримання медичних довідок і рекомендації з етикету під час спілкування.

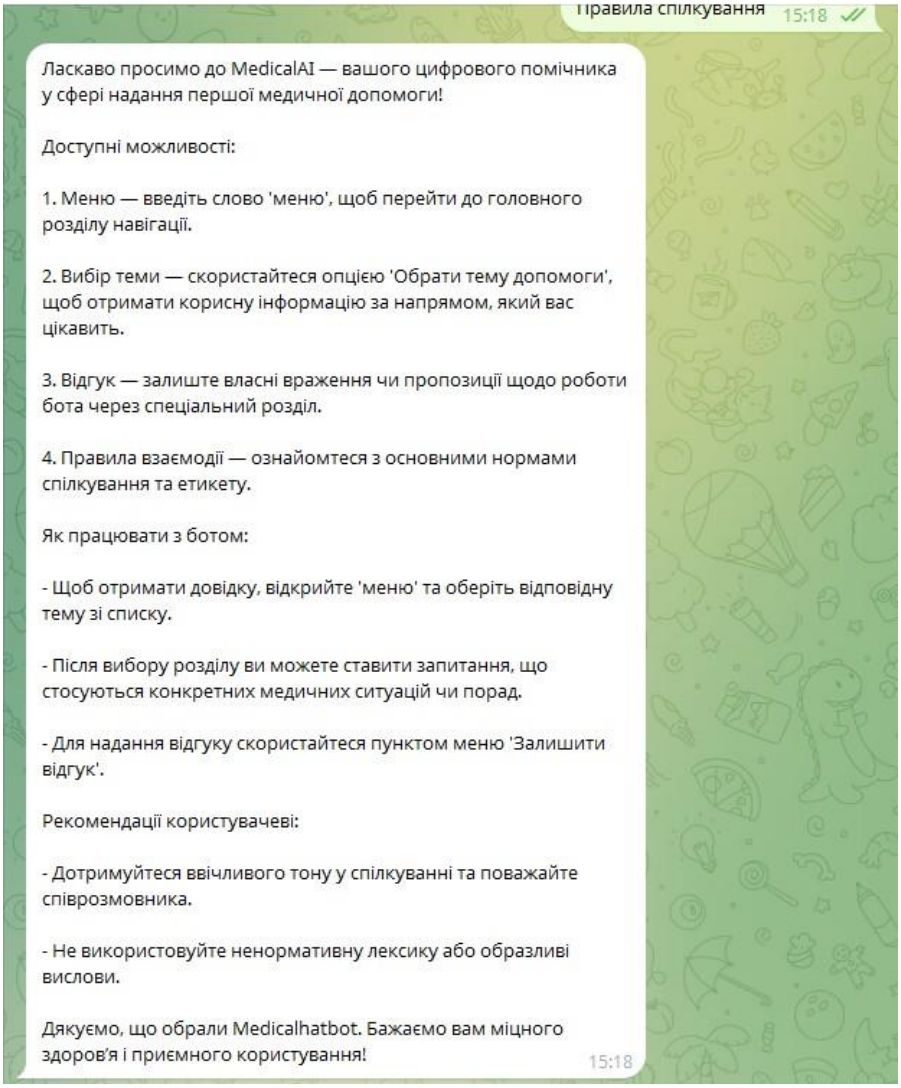

Рис. 3.24. Правила користування чат–ботом

У межах повідомлення пояснюється, як відкрити головне меню, обрати потрібну тему, ставити запитання або залишати відгуки. Крім того, наведено короткі рекомендації щодо культури взаємодії з ботом: дотримання ввічливого тону, поваги до співрозмовника та уникнення ненормативної лексики. Це сприяє формуванню позитивного користувацького досвіду і водночас запобігає некоректній поведінці під час тестування системи у відкритому середовищі.

На рис. 3.25 наведено приклад успішного діалогу з ботом під час надсилання відгуку. Після вибору відповідного пункту меню користувач отримує запрошення залишити коментар. У полі введення він формулює власні враження щодо роботи системи, точності відповідей та зручності використання інтерфейсу. У відповідь бот автоматично фіксує повідомлення, зберігає його у базі даних і надсилає підтвердження про успішне отримання тексту.

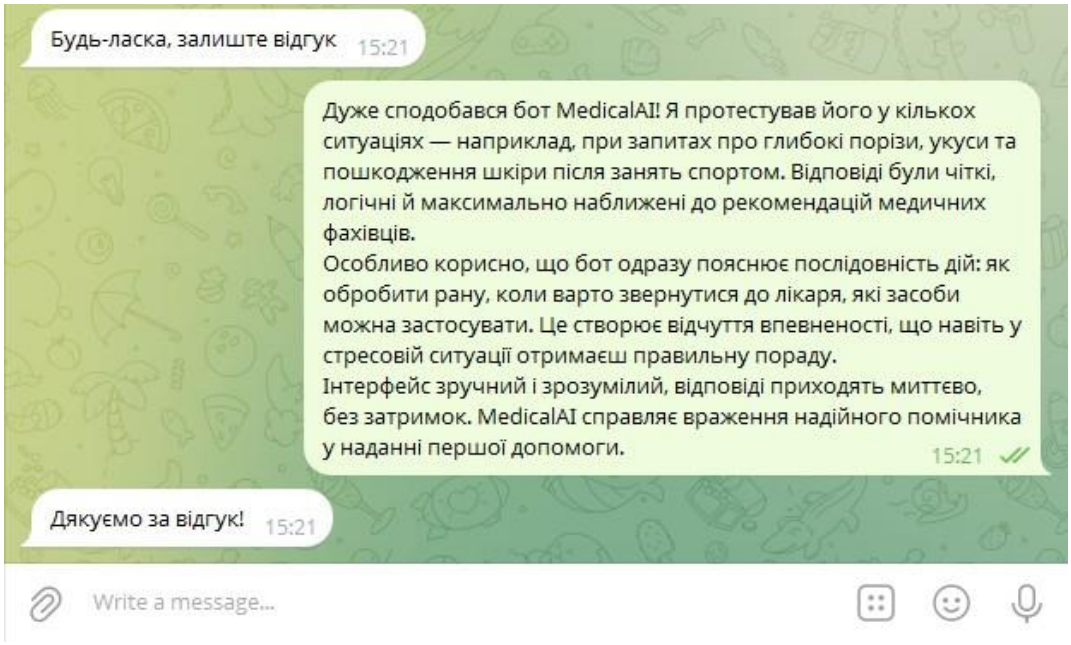

Рис. 3.25. Приклад відгуку користувача

Як видно з наведеного прикладу, користувач позитивно оцінює роботу системи, відзначаючи логічність відповідей, чіткість формулювань та швидкість реакції бота. Окремо підкреслюється користь від поетапних інструкцій, що дозволяють правильно реагувати у типових медичних ситуаціях, наприклад при порізах або пошкодженнях шкіри. Така форма зворотного зв’язку демонструє ефективність реалізованого механізму обробки відгуків, а також свідчить про високу ступінь довіри користувачів до інтелектуальної системи надання медичних консультацій.

ВИСНОВКИ

У результаті виконання кваліфікаційної магістерської роботи було розроблено та реалізовано інтелектуальну систему надання довідкових матеріалів з першої медичної допомоги у вигляді Telegram-бота, що функціонує на основі моделей машинного навчання. Розроблений програмний продукт поєднує модулі збору даних, навчання моделі, обробки користувацьких запитів і формування релевантних відповідей у режимі реального часу. Основою інтелектуальної складової системи став метод швидкого дерева, який забезпечив високу швидкість класифікації текстових запитів і стабільність результатів.

Структура програмного рішення побудована за трирівневою архітектурною моделлю, що включає рівень даних, рівень бізнес-логіки та рівень користувацького інтерфейсу. Такий підхід забезпечив логічну ізольованість компонентів, полегшив підтримку та масштабування системи, а також дав змогу реалізувати чітку взаємодію між модулями Telegram API, базою даних та ML.NET-моделлю.

У процесі навчання моделі було сформовано навчальний і тестовий набори даних, які містили пари «запит – відповідь». Після навчання моделі проведено тестування, результати якого показали максимальні значення точності, що свідчить про високу якість класифікації запитів користувачів і правильність побудови навчальної вибірки.

Під час експериментального дослідження реалізовано комплекс форм для управління темами, навчання моделей і тестування їх роботи. Крім того, створено повноцінну інтеграцію з Telegram, що забезпечує зручну взаємодію користувача з ботом через інтуїтивно зрозуміле меню, можливість вибору теми допомоги, ознайомлення з правилами спілкування та залишення відгуків. Результати тестування у реальному середовищі продемонстрували стабільну роботу системи, високу швидкість реагування на запити та коректність збереження зворотного зв’язку.

Наукова новизна дослідження полягає у поєднанні алгоритмів класифікації машинного навчання з телекомунікаційною платформою Telegram для створення інтерактивного помічника, здатного інтелектуально реагувати на запити користувачів. Використання методу швидкого дерева як базового алгоритму навчання дозволило досягти високої точності та мінімальних часових витрат при класифікації текстів медичної спрямованості.

Практична цінність розробленої системи полягає у можливості її використання як у навчальному процесі, так і в інформаційно-довідкових цілях, зокрема для ознайомлення користувачів із базовими алгоритмами дій при невідкладних медичних станах. Програмне рішення може бути розширене шляхом додавання нових тематичних розділів, підключення додаткових мов спілкування або інтеграції з зовнішніми базами медичних знань.

Подальший розвиток дослідження може бути спрямований на розширення функціональності Telegram-бота через впровадження механізмів діалогового навчання моделі, застосування методів глибинного навчання для аналізу контексту запитів та оптимізацію структури даних для обробки великих текстових корпусів. Отримані результати підтверджують практичну ефективність використання машинного навчання у системах автоматизованої медичної консультації та відкривають перспективи для створення більш складних інтелектуальних сервісів підтримки користувачів.

ПЕРЕЛІК ПОСИЛАНЬ

1. Out-of-Hospital Cardiac Arrest Facts. URL: https://www.redcross.org/take-a-class/resources/articles/cpr-facts-and-statistics (дата звернення 30.09.2025).

2. Survival after out-of-hospital cardiac arrest in Europe - results of the EuReCa TWO study. URL: https://wrap.warwick.ac.uk/id/eprint/132943/ (дата звернення 30.09.2025).

3. Papachristodoulou V., Tripsianis G., Constantinidis T. C., Kakagia D. D. Knowledge and attitudes in first aid practices for thermal burns: a cross-sectional study among adults in northern Greece. Materia socio-medica. 2023. Vol. 35, No 3, 228 p.

4. Nearly 270,000 people have completed first aid training provided by the Ukrainian Red Cross. URL: https://reliefweb.int/report/ukraine/nearly-270000-people-have-completed-first-aid-training-provided-ukrainian-red-cross-enuk (дата звернення 30.09.2025).

5. Psychological First Aid by AI: Proof-of-Concept and Comparative Performance of ChatGPT-4 and Gemini in Different Disaster Scenarios. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC12228088/ (дата звернення 30.09.2025).

6. First Aid in the Digital World. URL: https://www.globalfirstaidcentre.org/wp-content/uploads/2023/04/Final-Concept-Note-First-Aid-in-the-Digital-World-English_SV.pdf (дата звернення 30.09.2025).

7. Transforming healthcare with chatbots: Uses and applications–A scoping review. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC11915287/#:~:text=In%20this%20context%2C%20related%20to,2023 (дата звернення 30.09.2025).

8. Overview of Chatbots with special emphasis on artificial intelligence-enabled ChatGPT in medical science. URL:

https://pmc.ncbi.nlm.nih.gov/articles/PMC10644239/#:~:text=A%20schematic%20figure%20depicts%20Chatbot%27s,Chatbot%27s%20architecture%20and%20process%20flow (дата звернення 30.09.2025).

9. Sarker A., Zhang R., Wang Y., Xiao Y., Das S., Schutte D., Xu H. Natural Language Processing for Digital Health in the Era of Large Language Models. Yearbook of Medical Informatics. 2024. Vol. 33, No 1, pp. 229-240.

10. Chatbots in Health Care: Connecting Patients to Information. URL: www.ncbi.nlm.nih.gov/books/NBK602381/#:~:text=Key Messages (дата звернення 30.09.2025).

11. Alammar J., Grootendorst M. Hands-On Large Language Models. Sebastopol, CA : O'Reilly Media, 2024. 428 p.

12. Rocha-Silva R., de Lima, B. E., Costa T. G., Morais N. S., Jose G., Cordeiro D. F., de Lira C. A. B. Can people with epilepsy trust AI chatbots for information on physical exercise?. 2025. 163 p.

13. Clark M., Bailey S. Chatbots in health care: connecting patients to information. Canadian Journal of Health Technologies. 2024. Vol. 4, No 1, 22 p.

14. Джгут 2.0. Перша допомога Fast AID bot 2.0 @FastAid_bot. URL: https://dovidka.info/bots/dzhgut-2-0-persha-dopomoga-fast-aid-bot-2-0-fastaid_bot (дата звернення 30.09.2025).

15. Medical Advice From a Bot: The Unproven Promise of Babylon Health. URL: https://undark.org/2019/12/09/babylon-health-artificial-intelligence-medical-advice/ (дата звернення 30.09.2025).

16. Медичний чатбот для лікарів та пацієнтів: як надати та отримати допомогу. URL: https://life.pravda.com.ua/health/2022/03/03/247654/ (дата звернення 30.09.2025).

17. Psallidas F., Zhu Y., Karlas B., Henkel J., Interlandi M., Krishnan S., Karanasos K. Data science through the looking glass: Analysis of millions of github notebooks and ml. net pipelines. ACM SIGMOD Record. 2022. Vol. 51, No 2, pp. 30-37.

18. Mehmood K., Anees S. A., Muhammad S., Shahzad F., Liu Q., Khan W. R., Dube T. Machine learning and spatio temporal analysis for assessing ecological impacts of the billion tree afforestation project. Ecology and Evolution. 2025. Vol. 15, No 2, 30 p.

19. Mantika A. M., Triayudi A., Aldisa R. T. Sentiment analysis on twitter using Naïve Bayes and logistic regression for the 2024 presidential election. SaNa: Journal of Blockchain, NFTs and Metaverse Technology. 2024. Vol. 2, No 1, pp. 44-55.

20. Elkahwagy D. M., Kiriacos C. J., Mansour M. Logistic regression and other statistical tools in diagnostic biomarker studies. Clinical and translational oncology. 2024. Vol. 26, No 8, pp. 172-180.

21. Halder R. K., Uddin M. N., Uddin M. A., Aryal S., Khraisat A. Enhancing K-nearest neighbor algorithm: a comprehensive review and performance analysis of modifications. Journal of Big Data. 2024. Vol. 11, No 1, 113 p.

22. Sachdeva R. K., Bathla P., Rani P., Lamba R., Ghantasala G. P., Nassar I. F. A novel K-nearest neighbor classifier for lung cancer disease diagnosis. Neural Computing and Applications. 2024. Vol. 36, No 35, pp. 403-416.

23. Балабанов О.І., Павленко А.П. PyCharm для розробників Python: Навчальний посібник. - Львів: Видавництво Львівської політехніки, 2018. 300 с.

24. Карапецький В. П. Побудова графічного контенту додатків з використанням JavaFX і Swing компонентів і даних, взятих із баз даних / В. П. Карапецький. – Науковий вісник НЛТУ. 2021. 418 с.

25. Ashcraft A. Parallel Programming and Concurrency with C# 10 And . NET 6: A Modern Approach to Building Faster, More Responsive, and Asynchronous. NET Applications Using C#. Packt Publishing, Limited, 2022. 815 p.

26. Verma, R. Extending Visual Studio. In Visual Studio Extensibility Development: Extending Visual Studio IDE for Productivity, Quality, Tooling, Analysis, and Artificial Intelligence. 2023. pp. 73-113.

27. Kahlert T., Giza K. Visual Studio Code Tips & Tricks. Microsoft Deutschland GmbH. 2016. Vol. 1. 26 p.

28. Most Helpful Rider Reviews. URL: https://www.gartner.com/reviews/market/integrated-development-environment-ide-software/vendor/jetbrains/product/rider (дата звернення 30.09.2025).

29. Price M. J. C# 11 and . NET 7 - Modern Cross-Platform Development Fundamentals: Start Building Websites and Services with ASP. NET Core 7, Blazor, and EF Core 7, 7th Edition. Packt Publishing, Limited, 2022. 826 p.

ДОДАТКИ

Додаток А. Скрипти бази даних

```
/* ---------------------------------------------------------------------
   Ініціалізація
--------------------------------------------------------------------- */
SET ANSI_NULLS ON;
SET QUOTED_IDENTIFIER ON;
GO

/* ---------------------------------------------------------------------
   1) Базові довідники та сутності без зовнішніх залежностей
--------------------------------------------------------------------- */

-- 1.1 Categories
IF OBJECT_ID(N'dbo.Categories', N'U') IS NOT NULL DROP TABLE dbo.Categories;
GO
CREATE TABLE [dbo].[Categories] (
    [CategoriesId]   INT IDENTITY (1, 1) NOT NULL,
    [CategoriesName] NVARCHAR(250) NULL,
    [Description]    NVARCHAR(MAX) NULL,
    CONSTRAINT [PK_Categories] PRIMARY KEY CLUSTERED ([CategoriesId] ASC)
);
GO

-- 1.2 Users
IF OBJECT_ID(N'dbo.Users', N'U') IS NOT NULL DROP TABLE dbo.Users;
GO
CREATE TABLE [dbo].[Users] (
    [UsersId]       INT IDENTITY (1, 1) NOT NULL,
    [FirstName]     NVARCHAR(50)  NULL,
    [LastName]      NVARCHAR(50)  NULL,
    [UsersName]     NVARCHAR(50)  NULL,
    [UsersPassword] NVARCHAR(50)  NULL,
    [RoleId]        INT           NULL,          -- лишено як атрибут; довідник ролей не
створюється
    [Description]   NVARCHAR(1000) NULL,
    [Email]         NVARCHAR(150)  NULL,
    CONSTRAINT [PK_Users] PRIMARY KEY CLUSTERED ([UsersId] ASC)
);
GO

/* ---------------------------------------------------------------------
   2) Сутності, що посилаються на Categories та/або Users
--------------------------------------------------------------------- */

-- 2.1 ChatWithBot (FK -> Categories)
IF OBJECT_ID(N'dbo.ChatWithBot', N'U') IS NOT NULL DROP TABLE dbo.ChatWithBot;
```

```
GO
CREATE TABLE [dbo].[ChatWithBot] (
   [ChatWithBotId] INT IDENTITY (1, 1) NOT NULL,
   [ChatId]        BIGINT         NULL,
   [Question]      NVARCHAR(MAX)  NULL,
   [Answer]        NVARCHAR(MAX)  NULL,
   [CategoriesId]  INT            NULL,
   CONSTRAINT [PK_ChatWithBot] PRIMARY KEY CLUSTERED ([ChatWithBotId] ASC),
   CONSTRAINT [FK_ChatWithBot_Categories]
      FOREIGN KEY ([CategoriesId]) REFERENCES [dbo].[Categories]([CategoriesId])
      ON UPDATE CASCADE
      ON DELETE SET NULL
);
-- Індекс під FK
CREATE INDEX [IX_ChatWithBot_CategoriesId]
   ON [dbo].[ChatWithBot]([CategoriesId]);
GO

-- 2.2 Models (FK -> Categories, FK -> Users)  <<< ДОДАНО ЗВ'ЯЗОК З USERS >>>
IF OBJECT_ID(N'dbo.Models', N'U') IS NOT NULL DROP TABLE dbo.Models;
GO
CREATE TABLE [dbo].[Models] (
   [ModelsId]        INT IDENTITY (1, 1) NOT NULL,
   [ModelsName]      NVARCHAR(150) NULL,
   [CategoriesId]    INT           NULL,
   [ModelsFileModel] NVARCHAR(MAX) NULL,
   [Description]     NVARCHAR(MAX) NULL,
   [UsersId]         INT           NULL,  -- автор/власник моделі
   CONSTRAINT [PK_Models] PRIMARY KEY CLUSTERED ([ModelsId] ASC),
   CONSTRAINT [FK_Models_Categories]
      FOREIGN KEY ([CategoriesId]) REFERENCES [dbo].[Categories]([CategoriesId])
      ON UPDATE CASCADE
      ON DELETE SET NULL,
   CONSTRAINT [FK_Models_Users]
      FOREIGN KEY ([UsersId]) REFERENCES [dbo].[Users]([UsersId])
      ON UPDATE CASCADE
      ON DELETE SET NULL
);
-- Індекси під FK
CREATE INDEX [IX_Models_CategoriesId] ON [dbo].[Models]([CategoriesId]);
CREATE INDEX [IX_Models_UsersId]      ON [dbo].[Models]([UsersId]);
GO

/* ----------------------------------------------------------------
   3) Журнали та зворотний зв'язок (посилання на Users)
---------------------------------------------------------------- */

-- 3.1 Logs (FK -> Users)
IF OBJECT_ID(N'dbo.Logs', N'U') IS NOT NULL DROP TABLE dbo.Logs;
GO
CREATE TABLE [dbo].[Logs] (
   [LogsId]        INT IDENTITY (1, 1) NOT NULL,
```

```
    [UsersId]       INT           NULL,
    [EventNameShow] NVARCHAR(MAX)  NULL,
    [EventDate]     DATETIME       NULL,
    CONSTRAINT [PK_Logs] PRIMARY KEY CLUSTERED ([LogsId] ASC),
    CONSTRAINT [FK_Logs_Users]
        FOREIGN KEY ([UsersId]) REFERENCES [dbo].[Users]([UsersId])
        ON UPDATE CASCADE
        ON DELETE SET NULL
);
CREATE INDEX [IX_Logs_UsersId] ON [dbo].[Logs]([UsersId]);
GO

-- 3.2 LeavingFeedback (FK -> Users)
IF OBJECT_ID(N'dbo.LeavingFeedback', N'U') IS NOT NULL DROP TABLE
dbo.LeavingFeedback;
GO
CREATE TABLE [dbo].[LeavingFeedback] (
    [LeavingFeedbackId]   INT IDENTITY (1, 1) NOT NULL,
    [ChatId]              BIGINT         NULL,
    [Message]             NVARCHAR(MAX)  NULL,
    [FirstName]           NVARCHAR(150)  NULL,
    [LastName]            NVARCHAR(150)  NULL,
    [LeavingFeedbackDate] DATETIME       NULL,
    [UsersId]             INT            NULL,
    CONSTRAINT [PK_LeavingFeedback] PRIMARY KEY CLUSTERED
([LeavingFeedbackId] ASC),
    CONSTRAINT [FK_LeavingFeedback_Users]
        FOREIGN KEY ([UsersId]) REFERENCES [dbo].[Users]([UsersId])
        ON UPDATE CASCADE
        ON DELETE SET NULL
);
CREATE INDEX [IX_LeavingFeedback_UsersId]
    ON [dbo].[LeavingFeedback]([UsersId]);
GO
```

Додаток Б. Лістинги програмного коду

Лістинг 1. Код класу «ModelTeachForm»

```
using FirstAidAPP.AppCode;
using FirstAidAPP.Providers;
using Microsoft.ML;
using Microsoft.ML.Data;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace FirstAidAPP.Forms.Systems {
  public partial class ModelTeachForm : Form {
  private string _Path = "";
    private MLContext context;
    private TransformerChain<Microsoft.ML.Transforms.KeyToValueMappingTransformer>
model;
    private IDataView data;



    private int _selectedRowIndex = 0;
    private CategoriesProvider _CategoriesProvider = new CategoriesProvider();
    private List<Categories> _CategoriesList = new List<Categories>();
    private ValidationMy _Validation = new ValidationMy();
    private ModelsProvider _ModelsProvider = new ModelsProvider();
```

```
private List<Models> _ModelsList = new List<Models>();
private LogsProvider _LogsProvider = new LogsProvider();
private bool _IsModelTrain = false;

public ModelTeachForm() {
 InitializeComponent();
 LoadAllDate();
 DataLoad();
}

private void LoadAllDate() {
 _CategoriesList = _CategoriesProvider.GetAllCategories();
 CategoriesCBox.DataSource = _CategoriesList;
 CategoriesCBox.ValueMember = "CategoriesId";
 CategoriesCBox.DisplayMember = "CategoriesName";
}

private void DataLoad() {
 int firstRowIndex = 0;
 if (ModelsGridView.FirstDisplayedScrollingRowIndex > 0) {
  firstRowIndex = ModelsGridView.FirstDisplayedScrollingRowIndex;
 }
 try {
  _ModelsList = _ModelsProvider.GetAllModels();
  LoadDataInModelsGridView(_ModelsList);
  if (_selectedRowIndex == ModelsGridView.Rows.Count) {
   _selectedRowIndex = ModelsGridView.Rows.Count - 1;
  }
  if (_selectedRowIndex >= 0) {
   ModelsGridView.FirstDisplayedScrollingRowIndex = firstRowIndex;
   ModelsGridView.Rows[_selectedRowIndex].Selected = true;
```

```
    }
   } catch (Exception ex) {
    MessageBox.Show(ex.ToString());
   }
  }


  private void OpenBtn_Click(object sender, EventArgs e) {
   // Створення діалогового вікна для відкриття файлу
   OpenFileDialog openFileDialog = new OpenFileDialog();

   // Налаштування властивостей діалогового вікна
   openFileDialog.InitialDirectory = "C:\\";
   openFileDialog.Filter = "Text files (*.csv)|*.csv|All files (*.*)|*.*";
   openFileDialog.FilterIndex = 2;
   openFileDialog.RestoreDirectory = true;

   // Відображення діалогового вікна та обробка результату
   if (openFileDialog.ShowDialog() == DialogResult.OK) {
   try {
      _Path = openFileDialog.FileName;
      FileNameTBox.Text = openFileDialog.FileName;
      context = new MLContext(seed: 0);

      List<Input> datas = new List<Input>();
      using (StreamReader reader = new StreamReader(_Path, Encoding.GetEncoding(1251))) {
       while (!reader.EndOfStream) {
         var line = reader.ReadLine();
         //RaportTBox.Text += line + "\r\n"; // виводимо рядок, щоб переконатися, що він
містить правильну кількість полів
         var values = line.Split(';');
         Input item = new Input() {
```

```
            Question = values[0],
            Answer = values[1]
          };
          datas.Add(item);
        }
      }

      data = context.Data.LoadFromEnumerable<Input>(datas);
      // Розділ даних на навчальний набір та тестовий набір
      var trainTestData = context.Data.TrainTestSplit(data, testFraction: 0.2, seed: 0);
      var trainData = trainTestData.TrainSet;
      var testData = trainTestData.TestSet;

      // Побудова конвеєра обробки даних та навчання моделі
      var pipeline = context.Transforms.Text.FeaturizeText(outputColumnName: "Features",
        inputColumnName: "Question")
          .Append(context.Transforms.Conversion.ConvertType(outputColumnName: "Label",
          inputColumnName: "Answer", outputKind: DataKind.String))
          .Append(context.Transforms.Conversion.MapValueToKey(outputColumnName:
"Label"))
          .Append(context.Transforms.Conversion.MapKeyToValue(outputColumnName:
"Answer",
          inputColumnName: "Label"))
          .Append(context.MulticlassClassification.Trainers.SdcaNonCalibrated())
          .Append(context.Transforms.Conversion.MapKeyToValue(outputColumnName:
"PredictedAnswer",
          inputColumnName: "PredictedLabel"));

      RaportTBox.Text = "Training the model..." + "\r\n";
      model = pipeline.Fit(trainData);

      // Evaluate the model
      var predictions = model.Transform(testData);
```

```
      var metrics = context.MulticlassClassification.Evaluate(predictions, "Label", "Score");
      RaportTBox.Text += "Model evaluation metrics:" + "\r\n";
      RaportTBox.Text += ($"Log-loss: {metrics.LogLoss:P2}") + "\r\n";
      RaportTBox.Text += ($"Log-loss reduction: {metrics.LogLossReduction:P2}") + "\r\n";
      RaportTBox.Text += ($"MicroAccuracy: {metrics.MicroAccuracy:P2}") + "\r\n";
      RaportTBox.Text += ($"MacroAccuracy: {metrics.MacroAccuracy:P2}") + "\r\n";
      _IsModelTrain = true;
    } catch (Exception ex) {
      MessageBox.Show("Помилка: " + ex.Message);
    }
  }

}

private void AddBtn_Click(object sender, EventArgs e) {
  if (IsDataEnteringCorrect()) {
    //Save model
    string pathName = @"\teach\" + GenerateFileName() + ".zip";
    string localProj = System.IO.Path.GetDirectoryName(
    System.Reflection.Assembly.GetExecutingAssembly().Location);
    _ModelsProvider.InsertModels(ModelsNamesTBox.Text,
      Convert.ToInt32(CategoriesCBox.SelectedValue),
      pathName, DescriptionTBox.Text);
    context.Model.Save(model, data.Schema, localProj + pathName);
    ClearAllData();
    _LogsProvider.InsertLogs(LoginForm.CurrentUser.UsersId,
      "Було навчено модель" +
      ModelsNamesTBox.Text, DateTime.Now);
    MessageBox.Show("Дані успішно збережено!");
  }
}
```

```
  private void ClearBtn_Click(object sender, EventArgs e) {
    ClearAllData();
  }

  private void ExitBtn_Click(object sender, EventArgs e) {
    this.Close();
  }

  public string GenerateFileName() {
    DateTime now = DateTime.Now;
    string fileName = string.Format("{0}_{1}_{2}_{3}_{4}_{5}",
        now.Year, now.Month, now.Day, now.Hour, now.Minute, now.Second);

    return fileName;
  }

  private void ClearAllData() {
    _IsModelTrain = false;
    FileNameTBox.Text = String.Empty;
    ModelsNamesTBox.Text = String.Empty;
    RaportTBox.Text = String.Empty;
    DataLoad();
  }
  private bool IsDataEnteringCorrect() {
    bool isCorrect = true;
    if (!_IsModelTrain) {
      MessageBox.Show("Неможливо зберегти дані. \r\nЩе не навчено нейронну мережу!",
"Увага!");
      isCorrect = false;
    }
    if (Convert.ToInt32(CategoriesCBox.SelectedValue) > 0) {
```

```
        CategoriesValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        CategoriesValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_Validation.IsDataEntering(ModelsNamesTBox.Text)) {
        ModelsNamesValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        ModelsNamesValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      return isCorrect;
    }

    private void ModelsGridView_CellClick(object sender, DataGridViewCellEventArgs e) {
      if (e.ColumnIndex == 4 && ModelsGridView[0, e.RowIndex].Value.ToString() !=
_ModelsList[0].Message) {
        if (MessageBox.Show("Ви дійсно хочете видалити цей елемент?", "Видалити",
MessageBoxButtons.YesNo) == DialogResult.Yes) {
          _ModelsProvider.DeleteModelsByModelsId(Convert.ToInt32(ModelsGridView[0,
e.RowIndex].Value.ToString()));
          DataLoad();
        }
      }
    }
  }
}

// Input class

public class Input {
  [LoadColumn(0), ColumnName("Question")]
  public string Question;
```

```
  [LoadColumn(1), ColumnName("Answer")]
  public string Answer;
}

public class Output {
  [ColumnName("PredictedLabel")]
  public uint Prediction;

  [ColumnName("Score")]
  public float[] Scores;

  public string PredictedAnswer;
}
```

Лістинг 2. Код класу «StartBotForm»

```
using FirstAidAPP.AppCode;
using FirstAidAPP.Providers;
using Microsoft.ML;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading;
using System.Threading.Tasks;
using System.Windows.Forms;

using Telegram.Bot;
using Telegram.Bot.Polling;
using Telegram.Bot.Types;
using Telegram.Bot.Types.Enums;
using Telegram.Bot.Types.ReplyMarkups;
```

```
namespace FirstAidAPP.Forms.Controls {
  public partial class StartBotForm : Form {
    // ===== ПОЛЯ СТАНУ ТА ЗАЛЕЖНОСТІ =====
    private TelegramBotClient _BotClient;
    private CancellationTokenSource _cts;
    //MedicalAI - чат
    //MedicalAI344_Bot - назва бота
    //токен (без пробілів/лапок)
    private string _Token = "8397082241:AAGrxH4gL8Ec4vQZqrEF3M8HZUS7k-8mHlc";

    private bool _IschatStart = false;          // режим збору відгуку
    private bool _IsCategoriesLoad = false;     // вибір теми включено
    private bool isMenuFirslLoad = false;       // перше показування меню
    private const float ConfidenceThreshold = 0.65f; // поріг впевненості

    private readonly ValidationMy _Validation = new ValidationMy();
    private readonly CategoriesProvider _CategoriesProvider = new CategoriesProvider();
    private readonly LeavingFeedbackProvider _LeavingFeedbackProvider = new
LeavingFeedbackProvider();
    private readonly ChatWithBotProvider _ChatWithBotProvider = new ChatWithBotProvider();
    private readonly ModelsProvider _ModelsProvider = new ModelsProvider();

    private List<Categories> _CategoriesList = new List<Categories>();
    private Categories _SelectedCategories = new Categories();
    private Models _SelectedModels = new Models();

    private readonly MLContext context = new MLContext(seed: 123);
    private PredictionEngine<Input, Output> predictor;

    // ===== КОНСТРУКТОР =====
    public StartBotForm() {
```

```
    InitializeComponent();

    // (не обов’язково, але корисно для .NET Framework)
    System.Net.ServicePointManager.SecurityProtocol =
System.Net.SecurityProtocolType.Tls12;

    _CategoriesList = _CategoriesProvider.GetAllCategories();
    if (_CategoriesList.Count > 0 && _CategoriesList[0].CategoriesId != 0) {
      _SelectedCategories = _CategoriesList[0];
      _SelectedModels =
_ModelsProvider.SelectedModelsByCategoriesId(Convert.ToInt32(_SelectedCategories.Categor
iesId));
      LoadData(_SelectedModels.ModelsFileModel);
    } else {
      MessageBox.Show("Необхідно навчити хоча б одну модель!");
      this.Close();
    }
  }

  // ===== ПОДІЇ ФОРМИ =====
  private async void StartBotForm_Load(object sender, EventArgs e) {
    try {
      var token = CleanToken(_Token);
      if (!LooksLikeTelegramToken(token)) {
        MessageBox.Show("Токен має некоректний формат. Згенеруйте новий у @BotFather і
вставте без лапок/пробілів/переносів.");
        return;
      }

      _BotClient = new TelegramBotClient(token);
      _cts = new CancellationTokenSource();

      // Перевірка зв’язку
```

```
        try {
          var me = await _BotClient.GetMeAsync(_cts.Token);
          AppendLog("Бот запущено: @" + me.Username + " (id: " + me.Id + ")");
        } catch (Exception ex) {
          AppendLog("Помилка GetMeAsync: " + ex.Message);
          return;
        }

        var receiverOptions = new ReceiverOptions {
          AllowedUpdates = new UpdateType[] { UpdateType.Message } // приймаємо лише текстові повідомлення
        };

        _BotClient.StartReceiving(
          UpdateHandler,     // обробник оновлень
          ErrorHandler,      // обробник помилок
          receiverOptions,
          _cts.Token);

        AppendLog("StartReceiving() → OK. Напишіть щось боту в Telegram.");
      } catch (Exception ex) {
        AppendLog("Помилка запуску: " + ex.Message);
      }
    }

    private void StartBotForm_FormClosed(object sender, FormClosedEventArgs e) {
      try {
        if (_cts != null) _cts.Cancel();
      } catch { /* ignore */ }
    }

    // ===== ОБРОБНИК ОНОВЛЕНЬ (Telegram.Bot v18/19) =====
```

```
    private async Task UpdateHandler(ITelegramBotClient bot,
      Update update, CancellationToken ct) {
      if (update == null) return;
      var msg = update.Message;
      if (msg == null) return;
      if (msg.Type != MessageType.Text) return;
      if (string.IsNullOrEmpty(msg.Text)) return;
      AppendLog("RX [" + msg.Chat.Id + "]: " + msg.Text);
      // лог у вікно
      RaportTBox.BeginInvoke((MethodInvoker)(() => {
      RaportTBox.AppendText(msg.Text + "\r\n");
      RaportTBox.ScrollToCaret();
      }));

      // Показати меню один раз при першому повідомленні
      if (!isMenuFirslLoad) {
        await ShowMainMenu(msg.Chat.Id, ct);
        isMenuFirslLoad = true;
        // Не return — дозволимо одразу обробити й текст користувача після показу меню
      }

      // Режим збору відгуку
      if (_IschatStart) {
        var first = (msg.From != null) ? msg.From.FirstName : null;
        var last = (msg.From != null) ? msg.From.LastName : null;

        SaveFeedback(msg.Chat.Id, msg.Text, first, last);
        await _BotClient.SendTextMessageAsync(msg.Chat.Id, "Дякуємо за відгук!",
cancellationToken: ct);
        _IschatStart = false;
        return;
      }
```

```
    // Маршрутизація команд/дій
    var text = msg.Text.Trim();
    switch (text) {
      case "меню":
      case "Меню":
        await ShowMainMenu(msg.Chat.Id, ct);
        break;
      case "Обрати тему допомоги":
        ShowCategorys(msg.Chat.Id); // async void — запускається, відповідь відправиться всередині
        break;
      case "Правила спілкування":
        await SendCommunicationRulesAsync(msg.Chat.Id, ct);
        break;
      case "Залишити відгук":
        await _BotClient.SendTextMessageAsync(msg.Chat.Id,
          "Будь-ласка, залиште відгук", cancellationToken: ct);
        _IschatStart = true;
        break;
      default:
        if (_IsCategoriesLoad) {
          Ansver(msg.Chat.Id, text);   // ML.NET прогноз + логування у БД
        } else {
          LoadModelBot(msg.Chat.Id, text); // вибір моделі по номеру категорії
        }
        break;
    }
  }

  // ===== ОБРОБНИК ПОМИЛОК ПОЛЛІНГУ =====
  private Task ErrorHandler(ITelegramBotClient bot, Exception ex, CancellationToken ct) {
```

```
    AppendLog("Polling error: " + ex.Message);
    return Task.CompletedTask;
  }

  // ===== ЛОГІКА ДОДАТКУ (ТВОЇ МЕТОДИ, АДАПТОВАНІ) =====

  private void SaveFeedback(long chatId, string feedback, string firstName, string lastName) {
    _LeavingFeedbackProvider.InsertLeavingFeedback(chatId, firstName, lastName, feedback);
  }

  private async Task SendCommunicationRulesAsync(long chatId, CancellationToken ct) {
    string rules =
      "Ласкаво просимо до MedicalAI — вашого цифрового помічника у сфері надання першої медичної допомоги!\n\n" +
      "Доступні можливості:\n\n" +
      "1. Меню — введіть слово 'меню', щоб перейти до головного розділу навігації.\n\n" +
      "2. Вибір теми — скористайтеся опцією 'Обрати тему допомоги', щоб отримати корисну інформацію за напрямом, який вас цікавить.\n\n" +
      "3. Відгук — залиште власні враження чи пропозиції щодо роботи бота через спеціальний розділ.\n\n" +
      "4. Правила взаємодії — ознайомтеся з основними нормами спілкування та етикету.\n\n" +
      "Як працювати з ботом:\n\n" +
      "- Щоб отримати довідку, відкрийте 'меню' та оберіть відповідну тему зі списку.\n\n" +
      "- Після вибору розділу ви можете ставити запитання, що стосуються конкретних медичних ситуацій чи порад.\n\n" +
      "- Для надання відгуку скористайтеся пунктом меню 'Залишити відгук'.\n\n" +
      "Рекомендації користувачеві:\n\n" +
      "- Дотримуйтеся ввічливого тону у спілкуванні та поважайте співрозмовника.\n\n" +
      "- Не використовуйте ненормативну лексику або образливі вислови.\n\n" +
      "Дякуємо, що обрали Medicalhatbot. Бажаємо вам міцного здоров’я і приємного користування!";
```

```
    await _BotClient.SendTextMessageAsync(chatId, rules, cancellationToken: ct);
  }

  private IReplyMarkup GetMainMenuButtons() {
    var keyboard = new ReplyKeyboardMarkup(new[]
    {
      new KeyboardButton[] { "Обрати тему допомоги", "Залишити відгук" },
      new KeyboardButton[] { "Правила спілкування" }
    }) {
      ResizeKeyboard = true,
      OneTimeKeyboard = false
    };

    return keyboard;
  }

  private async Task ShowMainMenu(long chatId, CancellationToken ct) {
    var mainMenu = GetMainMenuButtons();
    await _BotClient.SendTextMessageAsync(chatId, "Будь ласка, оберіть опцію:",
replyMarkup: mainMenu, cancellationToken: ct);
  }

  private async void ShowCategorys(long ChatId) {
    _IsCategoriesLoad = false;
    string messages = string.Format("{0,3}|{1, -100}\r\n", "№", "Теми допомоги");
    for (int i = 0; i < _CategoriesList.Count; i++) {
      messages += string.Format("{0,3}|{1, -100}\r\n", _CategoriesList[i].Number,
_CategoriesList[i].CategoriesName) + "\r\n";
    }
    await _BotClient.SendTextMessageAsync(ChatId, messages);
  }
```

```
    private async void LoadModelBot(long ChatId, string msgText) {
      // Перевіряємо, чи текст — валідний номер
      if (_Validation.IsDataConvertToInt(msgText)) {
        int selectedNumber = Convert.ToInt32(msgText);
        Categories selectedCategory = _CategoriesList.FirstOrDefault(c => c.Number ==
selectedNumber);

        if (selectedCategory != null) {
          _SelectedCategories = selectedCategory;
          _SelectedModels =
_ModelsProvider.SelectedModelsByCategoriesId(Convert.ToInt32(_SelectedCategories.Categor
iesId));

          string messages = "Ви вибрали тему: " + selectedCategory.CategoriesName;

          LoadData(_SelectedModels.ModelsFileModel);
          _IsCategoriesLoad = true;
          await _BotClient.SendTextMessageAsync(ChatId, messages);
        } else {
          await _BotClient.SendTextMessageAsync(ChatId, "Тема з таким номером не
знайдена!");
        }
      } else {
        await _BotClient.SendTextMessageAsync(ChatId, "Введене число не є цифрою!");
      }
    }

    private void Ansver(long ChatId, string msgText) {
      string message;
      var input = new Input { Question = msgText };

      try {
```

```
        var prediction = predictor.Predict(input);


        if (prediction == null || prediction.Scores == null || prediction.Scores.Length == 0) {
          message = "Не вдалося отримати відповідь від моделі. Спробуйте повторити запит пізніше.";
        } else {
          // Обчислення softmax
          var probs = Softmax(prediction.Scores);


          // Індекс передбаченого класу
          int predIdx;
          if (prediction.Prediction > 0 && prediction.Prediction <= probs.Length)
            predIdx = (int)prediction.Prediction - 1;
          else
            predIdx = ArgMax(prediction.Scores);


          float confidence = probs[predIdx];
          string answerText = prediction.PredictedAnswer;


          if (confidence < ConfidenceThreshold) {
            message = "Я не впевнений у відповіді. Будь ласка, переформулюйте запитання або уточніть деталі.";
          } else if (!string.IsNullOrEmpty(answerText) &&
                  answerText.Equals("Answer", StringComparison.OrdinalIgnoreCase)) {
            message = "Вибачте, я не можу відповісти на поставлене запитання.";
          } else {
            message = !string.IsNullOrEmpty(answerText)
                    ? answerText
                    : "Немає відповіді.";
          }
        }
```

```
        // Відправлення повідомлення користувачу
        _BotClient.SendTextMessageAsync(ChatId,  message);


        // Збереження у БД
        _ChatWithBotProvider.InsertChatWithBot(ChatId, input.Question, message,
_SelectedCategories.CategoriesId);
      } catch (Exception ex) {
        _BotClient.SendTextMessageAsync(ChatId, "Сталася помилка під час обробки запиту: "
+ ex.Message);
      }
    }


    // === Допоміжні методи ===
    private static float[] Softmax(float[] scores) {
      float max = scores.Max();
      double sum = 0.0;
      var exps = new double[scores.Length];
      for (int i = 0; i < scores.Length; i++) {
      double v = Math.Exp(scores[i] - max);
      exps[i] = v;
        sum += v;
      }
      var probs = new float[scores.Length];
      if (sum <= 0.0) {
        float p = 1f / scores.Length;
        for (int i = 0; i < probs.Length; i++) probs[i] = p;
        return probs;
      }
      for (int i = 0; i < scores.Length; i++) {
        probs[i] = (float)(exps[i] / sum);
      }
      return probs;
```

```
}

private static int ArgMax(float[] arr) {
  int idx = 0;
  float best = arr[0];
  for (int i = 1; i < arr.Length; i++) {
    if (arr[i] > best) { best = arr[i]; idx = i; }
  }
  return idx;
}
private void LoadData(string FilePath) {
  string localProj = Application.StartupPath + FilePath;
  DataViewSchema modelSchema;
  ITransformer model = context.Model.Load(localProj, out modelSchema);
  predictor = context.Model.CreatePredictionEngine<Input, Output>(model);
}

// ===== Допоміжні утиліти =====
private void AppendLog(string line) {
try {
    if (RaportTBox != null && !RaportTBox.IsDisposed) {
      RaportTBox.BeginInvoke((MethodInvoker)(() => {
      RaportTBox.AppendText(line + Environment.NewLine);
      RaportTBox.ScrollToCaret();
      }));
    }
  } catch { /* ignore */ }
}

private static string CleanToken(string raw) {
  if (raw == null) return "";
```

```
      var s = raw.Trim()
        .Replace("\uFEFF", "")
        .Replace("\u200B", "")
        .Replace("\u200C", "")
        .Replace("\u200D", "")
        .Replace("\u2060", "")
        .Replace("“", "").Replace("”", "").Replace("\"", "").Replace("'", "")
        .Replace("\r", "").Replace("\n", "")
        .Replace("–", "-").Replace("—", "-").Replace("−", "-");
      return s;
    }

    private static bool LooksLikeTelegramToken(string t) {
      if (string.IsNullOrEmpty(t)) return false;
      int colon = t.IndexOf(':');
      if (colon < 6 || colon > 12) return false;
      if (t.Length < colon + 31) return false;
      for (int i = colon + 1; i < t.Length; i++) {
        char c = t[i];
        if (!(char.IsLetterOrDigit(c) || c == '_' || c == '-')) return false;
      }
      return true;
    }
  }
}
```

Лістинг 3. Код класу «TestModelsForm»

```
using FirstAidAPP.AppCode;
using FirstAidAPP.Forms.Systems;
using FirstAidAPP.Providers;
using Microsoft.ML;
using System;
```

```
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace FirstAidAPP.Forms.Controls {
  public partial class TestModelsForm : Form {
    private CategoriesProvider _CategoriesProvider = new CategoriesProvider();
    private ValidationMy _Validation = new ValidationMy();
    private List<Categories> _CategoriesList = new List<Categories>();

    private ModelsProvider _ModelsProvider = new ModelsProvider();
    private Models _SelectedModels = new Models();
    private bool _IsCategoriesLoad = false;
    private MLContext context = new MLContext();
    private PredictionEngine<Input, Output> predictor;
    private bool _isFormLoad = false;
    private const float ConfidenceThreshold = 0.65f; // поріг впевненості (0..1)
    public TestModelsForm() {
      InitializeComponent();
      LoadAllDate();
    }

    private void LoadAllDate() {
      _CategoriesList = _CategoriesProvider.GetAllCategories();
      CategoriesCBox.DataSource = _CategoriesList;
      CategoriesCBox.ValueMember = "CategoriesId";
```

```
  CategoriesCBox.DisplayMember = "CategoriesName";
  _IsCategoriesLoad = true;
  CategoriesCBox_SelectedValueChanged(CategoriesCBox, EventArgs.Empty);
}

private void CategoriesCBox_SelectedValueChanged(object sender, EventArgs e) {
  if (_IsCategoriesLoad) {
    if (Convert.ToInt32(CategoriesCBox.SelectedValue) > 0) {
      try {
        _SelectedModels = _ModelsProvider.SelectedModelsByCategoriesId(
          Convert.ToInt32(CategoriesCBox.SelectedValue));
        if (_SelectedModels.ModelsId != 0) {
          LoadData(_SelectedModels.ModelsFileModel);
        } else {
          if (_isFormLoad) {
            MessageBox.Show("По вибраній категорії ще не навчено чат-бота!");
          } else {
            _isFormLoad = true;
          }
        }
      } catch (Exception ex) {
        MessageBox.Show(ex.Message);
      }
    }
  }
}

private void LoadData(string FilePath) {
  string localProj = Application.StartupPath + FilePath;
  // Визначення DataViewSchema для конвеєра підготовки даних і навченої моделі
  DataViewSchema modelSchema;
```

```
  // Завантаження моделі
  ITransformer model = context.Model.Load(localProj, out modelSchema);
  // Evaluate the model

  // Використання моделі для прогнозування
  predictor = context.Model.CreatePredictionEngine<Input, Output>(model);
}

private bool IsQuestionCorrect() {
  bool isCorrect = true;
  if (_Validation.IsDataEntering(QuestionTBox.Text)) {
    QuestionValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
  } else {
    QuestionValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
    isCorrect = false;
  }
  if (Convert.ToInt32(CategoriesCBox.SelectedValue) > 0) {
    CategoriesValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
  } else {
    CategoriesValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
    isCorrect = false;
  }
  return isCorrect;
}

private void QuestionBtn_Click(object sender, EventArgs e) {
  if (!IsQuestionCorrect()) return;

  var input = new Input { Question = QuestionTBox.Text };

  if (predictor == null) {
```

```
  MessageBox.Show("Поки не створено моделі для обраної категорії", "Увага!");
  return;
 }

 try {
  var prediction = predictor.Predict(input);

  // Безпечно обробляємо можливі аномалії
  if (prediction == null || prediction.Scores == null || prediction.Scores.Length == 0) {
   MaterialsTBox.Text += "Запитання: " + QuestionTBox.Text + "\r\n";
   MaterialsTBox.Text += "Не вдалося отримати коректний прогноз моделі.\r\n";
   return;
  }

  // 1) Обчислюємо softmax-імовірності
  var probs = Softmax(prediction.Scores);

  // 2) Визначаємо індекс передбаченого класу
  int predIdx;
  if (prediction.Prediction > 0 && prediction.Prediction <= probs.Length) {
   predIdx = (int)prediction.Prediction - 1; // KeyType у ML.NET 1-базований
  } else {
   predIdx = ArgMax(prediction.Scores);       // запасний варіант
  }

  // 3) Імовірність для передбаченого класу
  float confidence = probs[predIdx];

  MaterialsTBox.Text += "Запитання: " + QuestionTBox.Text + "\r\n";

  // Якщо ваша пайплайн-модель заповнює PredictedAnswer (наприклад, через map ключ->текст),
```

```
        // використовуємо його; інакше ви можете самостійно витягти назву класу за індексом.
        string answerText = prediction.PredictedAnswer;

        // 4) Логіка повідомлень залежно від порога
        if (confidence < ConfidenceThreshold) {
          // Низька впевненість — просимо переформулювати або поставити уточнення
          MaterialsTBox.Text +=
            $"На жаль, я не впевнений у відповіді (впевненість: {(confidence * 100f):F1}%). " +
            $"Спробуйте переформулювати запитання або надати більше контексту.\r\n";
        } else {
          // Висока/достатня впевненість — віддаємо відповідь
          if (string.Equals(answerText, "Answer", StringComparison.OrdinalIgnoreCase)) {
            MaterialsTBox.Text +=
              $"Вибачте, я не можу відповісти на поставлене запитання (впевненість: {(confidence * 100f):F1}%).\r\n";
          } else {
            MaterialsTBox.Text +=
              $"Відповідь ({(confidence * 100f):F1}%): {answerText}\r\n";
          }
        }
      } catch (Exception ex) {
        MaterialsTBox.Text += "Сталася помилка під час прогнозування: " + ex.Message + "\r\n";
      }
    }

    // === Допоміжні методи ===

    private static float[] Softmax(float[] scores) {
      // чисельно стабільний softmax
      float max = scores.Max();
      double sum = 0.0;
```

```
    var exps = new double[scores.Length];
    for (int i = 0; i < scores.Length; i++) {
    double v = Math.Exp(scores[i] - max);
    exps[i] = v;
      sum += v;
    }
    var probs = new float[scores.Length];
    if (sum <= 0.0) {
      // захист від ділення на нуль — рівномірний розподіл
      float p = 1f / scores.Length;
      for (int i = 0; i < probs.Length; i++) probs[i] = p;
      return probs;
    }
    for (int i = 0; i < scores.Length; i++) {
      probs[i] = (float)(exps[i] / sum);
    }
    return probs;
  }

  private static int ArgMax(float[] arr) {
    int idx = 0;
    float best = arr[0];
    for (int i = 1; i < arr.Length; i++) {
      if (arr[i] > best) { best = arr[i]; idx = i; }
    }
    return idx;
  }

 }
}
```