Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет інформаційних систем та технологій
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: Розробка автоматизованої системи для надання першої медичної допомоги з використанням чат-бота

Виконав: студент VI курсу, групи СТм-61
спеціальності: 126 Інформаційні системи та технології
(шифр і назва спеціальності)

Маланчук М.І.
(підпис) (прізвище та ініціали)

Керівник _____ Козбур Г.В.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Никитюк В.В.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ Загородна Н.В.
(підпис) (прізвище та ініціали)

Тернопіль
2024

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет Інформаційних систем та технологій
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
« » _____ 2024 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Магістр
(назва освітнього ступеня)

за спеціальністю 126 Інформаційні системи та технології
(шифр і назва спеціальності)

Студенту Маланчуку Максиму Ігоровичу
(прізвище, ім'я, по батькові)

1. Тема роботи
Розробка автоматизованої системи для надання першої медичної допомоги з використанням чат-бота

Керівник роботи к.т.н., доц. Козбур Галина Володимирівна
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «_» _____ 2024 року № _____

2. Термін подання студентом завершеної роботи

3. Вихідні дані до роботи Наукові публікації про моделі класифікації, веб-ресурси з оглядами та аналізом різних систем, документація до C#, ML.NET
Веб-ресурси в яких описана робота та навчання штучного інтелекту.

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1 Огляд та аналіз тематичної сфери та визначення проблематки. 2. Обгрунтування вибору технологій та особливості розробки програмного рішення.
3. Розробка, впровадження і тестування програмного забезпечення.
4. Охорона праці та безпека в надзвичайних ситуаціях. Висновки. Перелік джерел. Додатки

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

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Сенчишин В.С., доцент		
Безпека в надзвичайних ситуаціях	Клепчик В.М., ст. викладач		

7. Дата видачі завдання __ _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	19.09.2024 – 20.09.2024	*Виконано*
2.	Підбір джерел про розробку чатбота на основі моделей класифікацій	21.09.2024 – 02.10.2024	*Виконано*
3.	Опрацювання джерел в галузі дослідження	03.10.2024 – 12.10.2024	*Виконано*
4.	Дослідження використання чатботів на основі штучного інтелекту	13.10.2024 – 17.10.2024	*Виконано*
5.	Оформлення розділу «ОГЛЯД ТА АНАЛІЗ ТЕМАТИЧНОЇ СФЕРИ ТА ВИЗНАЧЕННЯ ПРОБЛЕМАТИКИ»	18.10.2024 – 01.11.2024	*Виконано*
6.	Оформлення розділу «ОБҐРУНТУВАННЯ ВИБОРУ ТЕХНОЛОГІЙ ТА ОСОБЛИВОСТІ РОЗРОБКИ ПРОГРАМНОГО РІШЕННЯ»	02.11.2024 – 20.11.2024	*Виконано*
7.	Оформлення розділу «РОЗРОБКА, ВПРОВАДЖЕННЯ І ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ»	21.11.2024 – 04.12.2024	*Виконано*
8.	Виконання завдання до підрозділу «Безпека життєдіяльності, основи охорони праці»	05.12.2024 – 07.12.2024	*Виконано*
9.	Оформлення кваліфікаційної роботи	08.12.2024 – 11.12.2024	*Виконано*
10.	Нормконтроль	11.12.2024 – 15.12.2024	*Виконано*
11.	Перевірка на плагіат	16.12.2024 – 18.12.2024	*Виконано*
12.	Попередній захист кваліфікаційної роботи	20.12.2024	*Виконано*
13.	Захист кваліфікаційної роботи	27.12.2024	

Студент _____ __ Маланчук М.І.
(підпис) (прізвище та ініціали)

Керівник роботи _____ __ Козбур Г.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка автоматизованої системи для надання першої медичної допомоги з використанням чат-бота.// Кваліфікаційна робота освітнього рівня «Магістр» // Маланчук Максим Ігорович // Тернопільський національний технічний університет імені Івана Пулюя, факультет інформаційний систем та технологій, кафедра комп'ютерних наук, група СТм-61 // Тернопіль, 2024 // С. 117, рис. – 57, табл. – 19, додат. – 2, бібліогр. – 52.

Ключові слова: ТЕЛЕГРАМ-БОТ, ПЕРША МЕДИЧНА ДОПОМОГА, АВТОМАТИЗАЦІЯ, МОДЕЛІ КЛАСИФІКАЦІЇ, МАШИННЕ НАВЧАННЯ, ІНФОРМАЦІЙНА СИСТЕМА, АЛГОРИТМИ КЛАСИФІКАЦІЇ, МЕДИЧНІ РЕКОМЕНДАЦІЇ.

Мета дослідження полягає у розробці системи надання автоматизованих рекомендацій з першої медичної допомоги на основі алгоритмів класифікації, інтегровану у платформу Telegram, для швидкого і доступного вирішення медичних потреб користувачів в екстрених ситуаціях.

У роботі проведено огляд медичних джерел і законодавчої бази з надання першої допомоги, а також аналіз існуючих мобільних додатків для медичної допомоги та їхніх аналогів, що дозволило виявити переваги та недоліки сучасних рішень.

На етапі розробки було обґрунтовано вибір технологій для реалізації системи, а також побудовано діаграми варіантів використання та основних бізнес-процесів. Система спроектована на основі трьохрівневої архітектури, а також було створено базу даних і детально розроблено діаграми класів.

У третьому розділі реалізовано алгоритми машинного навчання для тренування моделей рекомендацій, модуль для інтеграції моделей у Telegram-бот, а також окремі програмні компоненти для керування функціями ботів.

Об’єкт дослідження: процес автоматизації надання першої медичної допомоги через інтеграцію з чат-ботами на базі платформ машинного навчання та аналізу даних.

Предмет дослідження: алгоритми класифікації та їх використання в системах надання медичних рекомендацій через Telegram-бот для покращення оперативності та точності рекомендацій в екстрених ситуаціях.

ANNOTATION

Development of an Automated System for Providing First Aid Using a Chatbot// Qualification work for Master's Degree / Malanchuk Maksym Ihorovych // Ternopil National Technical University named after Ivan Puluj, Faculty of Information Systems and Technologies, Department of Computer Science, group STm-61 // Ternopil, 2024 // P. 117, figs. 57, tables 19, annexes 2, bibliography 52.

Keywords: TELEGRAM BOT, FIRST AID, AUTOMATION, CLASSIFICATION MODELS, MACHINE LEARNING, INFORMATION SYSTEM, CLASSIFICATION ALGORITHMS, MEDICAL RECOMMENDATIONS.

The purpose of the study is to develop a system for providing automated first aid recommendations based on classification algorithms, integrated into the Telegram platform, to quickly and easily address the medical needs of users in emergency situations.

The paper reviews medical sources and the legislative framework for first aid, as well as analyzes existing mobile applications for medical care and their analogues, which allowed us to identify the advantages and disadvantages of modern solutions.

At the development stage, the choice of technologies for implementing the system was substantiated, and diagrams of use cases and key business processes were built. The system was designed on the basis of a three-tier architecture, and a database was created and class diagrams were developed in detail.

The third section implements machine learning algorithms for training recommendation models, a module for integrating models into a Telegram bot, and separate software components for managing bot functions.

Object of research: the process of automating first aid through integration with chatbots based on machine learning and data analysis platforms.

The subject of the study: classification algorithms and their use in systems for providing medical recommendations through a Telegram bot to improve the efficiency and accuracy of recommendations in emergency situations.

ЗМІСТ

ВСТУП

Питання надання першої медичної допомоги завжди було надзвичайно важливим, особливо у випадках, коли час є критичним фактором. Сучасні тенденції розвитку технологій відкривають нові можливості для автоматизації та оптимізації процесів, пов'язаних із допомогою при надзвичайних ситуаціях. Одним із перспективних напрямів є використання цифрових платформ та інструментів штучного інтелекту для надання рекомендацій з першої допомоги в режимі реального часу. Зокрема, такі технології, як Telegram-боти, поєднані з алгоритмами класифікації, дають змогу надавати точні й оперативні рекомендації залежно від конкретної ситуації, що може бути критично важливим у випадках невідкладної медичної допомоги [1].

Проблема полягає в тому, що багато людей не володіють достатніми знаннями або впевненістю у своїх діях під час екстрених ситуацій, коли необхідно надавати першу допомогу. Помилки або затримки в діях можуть мати серйозні наслідки для здоров'я або навіть життя постраждалих. Окрім того, навіть наявні медичні додатки часто потребують значного часу на введення інформації або не мають можливості адаптувати рекомендації до конкретних обставин. Це ускладнює процес надання допомоги та знижує ефективність її автоматизації.

Актуальність даної роботи зумовлена необхідністю розвитку систем, що здатні оперативно, точно й інтуїтивно зрозуміло надавати користувачам рекомендації з першої медичної допомоги. У світлі розвитку технологій штучного інтелекту і можливостей класифікаційних моделей, що здатні аналізувати різноманітні медичні симптоми й стан пацієнта, виникає потреба у створенні рішень, які можуть функціонувати на популярних комунікаційних платформах, таких як Telegram. Важливо, що подібні системи можуть бути інтегровані в повсякденне життя людей, що дозволить швидко реагувати на надзвичайні ситуації без потреби у спеціалізованих медичних знаннях.

Розвиток подібних рішень є особливо важливим для України, де швидка й ефективна перша допомога може значно знизити рівень смертності та

ускладнень при нещасних випадках або травмах. Також варто зазначити, що актуальність роботи підкріплюється складною епідеміологічною ситуацією в країні та світі, де забезпечення доступу до першої допомоги, навіть дистанційно, стає важливим елементом охорони здоров'я.

Практичне значення отриманих результатів полягає у створенні функціональної системи, яка може бути використана для оперативного надання першої медичної допомоги за допомогою чат-бота, що є зручним для широкого кола користувачів. Впроваджена система забезпечує швидке і точне надання медичних рекомендацій, особливо в умовах обмеженого доступу до кваліфікованої медичної допомоги. Це дозволяє використовувати розроблене рішення в різноманітних сферах, включаючи підтримку медичного персоналу, а також надання рекомендацій звичайним користувачам в екстрених ситуаціях, що сприяє підвищенню рівня загальної безпеки та ефективності першої допомоги.

1 ОГЛЯД ТА АНАЛІЗ ТЕМАТИЧНОЇ СФЕРИ ТА ВИЗНАЧЕННЯ ПРОБЛЕМАТИКИ

1.1 Огляд медичних джерел, присвячених темі надання першої допомоги

Надання першої допомоги – це процес виконання базових медичних заходів, які спрямовані на збереження життя, полегшення страждань постраждалого та запобігання можливим ускладненням до прибуття професійних медиків [2]. Особливістю першої допомоги є її доступність – вона може бути надана як професійними рятувальниками, так і звичайними громадянами в екстрених ситуаціях. Ключова роль цієї допомоги полягає в тому, що її правильне та швидке надання до прибуття медичних служб часто є вирішальним фактором для збереження життя людини.

Незважаючи на те, що першу допомогу здебільшого надають фахівці медичної сфери, є чимало ситуацій, коли екстрені служби не можуть оперативно прибути. У таких випадках важливим є знання базових навичок надання допомоги серед населення. Це особливо стосується певних категорій людей, таких як правоохоронці, рятувальники, працівники авіаційної чи залізничної галузей, де такі навички обов’язкові для виконання професійних обов'язків.

Основна мета першої допомоги – це збереження життєво важливих функцій організму постраждалого, мінімізація загроз здоров'ю та запобігання розвитку серйозних ускладнень до моменту надання професійної медичної допомоги [3]. Оперативність та правильність дій під час надання першої допомоги можуть суттєво вплинути на кінцевий результат, включаючи збереження життя або попередження інвалідності. Це підкреслює важливість навчання базовим медичним навичкам не лише медичних працівників, але й широкого кола громадян, що дозволяє створити більш захищене суспільство.

Важливо зазначити, що для успішної організації навчальних програм і створення умов для надання першої допомоги необхідне чітке правове регулювання. Законодавчі та нормативно-правові акти в Україні встановлюють

саме такі правила, забезпечуючи правову базу для впровадження стандартів надання допомоги, визначення відповідальних суб'єктів і захисту осіб, що її надають. Ці акти створюють правове регулювання для ситуацій, коли перша допомога надається до прибуття професійних медичних служб, таких як аварії, травми чи інші надзвичайні події. Вони також підтримують систему навчання та сертифікації громадян, що дозволяє багатьом людям здобути навички надання невідкладної допомоги. У цьому контексті важливо відзначити кілька основних документів, які забезпечують правову базу для таких дій. Для ілюстрації наведемо порівняльний аналіз ключових нормативних актів у табл. 1.1.

Таблиця 1.1 – Законодавчі та нормативно-правові акти надання першої допомоги в Україні

Назва закону чи нормативного акту	Короткий опис	Суб'єкти, відповідальні	Юридична відповідальність
Закон України "Про екстрену медичну допомогу"	Визначає порядок надання медичної допомоги та обов'язки фахівців [4]	Лікарі, працівники екстрених служб, добровольці	Відповідальність за ненадання допомоги в межах професійних обов'язків
Кримінальний кодекс України (ст. 136)	Встановлює відповідальність за ненадання допомоги особі, що перебуває в небезпеці [5]	Будь-які громадяни	Кримінальна відповідальність за ненадання допомоги в небезпечній для життя ситуації
Закон України "Про охорону праці"	Регулює обов’язки щодо навчання працівників навичкам першої допомоги [6]	Роботодавці, працівники, інструктори з охорони праці	Адміністративна відповідальність за невиконання вимог щодо навчання або обладнання для надання допомоги

В Україні створена досить чітка правова база для регулювання питань надання першої допомоги. Важливим є той факт, що законодавство охоплює як професійну медичну допомогу, так і домедичну допомогу, яку можуть надавати

звичайні громадяни [7]. Однак відповідальність за ненадання допомоги також передбачена, що стимулює громадян активно долучатися до вирішення кризових ситуацій.

Окрім національного регулювання, велику роль відіграють світові стандарти, що формуються на основі рекомендацій міжнародних організацій. Зокрема, Всесвітня організація охорони здоров'я (ВООЗ), Міжнародна федерація Червоного Хреста та Червоного Півмісяця (IFRC) і Міжнародний комітет з реанімації (ILCOR) розробляють стандарти надання першої допомоги, що ґрунтуються на наукових дослідженнях і практичному досвіді. Ці стандарти спрямовані на уніфікацію підходів до надання допомоги з метою мінімізації смертності та ускладнень під час надання допомоги постраждалим до прибуття професійних медичних служб. Завдяки таким нормативам надається можливість застосовувати найкращі практики на міжнародному рівні, що є важливим кроком у забезпеченні безпеки та здоров'я громадян.

Розглядаючи міжнародні нормативи та стандарти виділено кілька ключових документів та рекомендацій, які регулюють надання першої допомоги на світовому рівні та представлено у табл. 1.2.

Таблиця 1.2 – Міжнародні стандарти надання першої допомоги

Назва організації	Короткий опис	Цільові групи	Юридичні або етичні аспекти
Good Samaritan Laws (закони доброго самарянина)	Закони у США та інших країнах, які захищають тих, хто надає допомогу, від судових позовів [8]	Громадяни, що надають допомогу	Захист від судових позовів у випадку ненавмисних помилок при наданні першої допомоги, за умови відсутності навмисної шкоди
ВООЗ	Глобальні стандарти для надання домедичної допомоги, орієнтовані на широке населення [9]	Усі громадяни, медичні працівники	Важливість доступу до навчання для всіх верств населення; також забезпечення рівного доступу до першої допомоги в країнах із низьким рівнем медичних послуг
IFRC	Рекомендації щодо надання першої допомоги у надзвичайних ситуаціях[10]	Добровольці, громадяни, які проходять підготовку	IFRC також надає рекомендації щодо створення національних стандартів та законів для підтримки першої допомоги

Аналіз міжнародних стандартів і рекомендацій показує, що глобальне правове регулювання надання першої допомоги фокусується на уніфікації підходів і захисті осіб, які надають допомогу. Основними аспектами є навчання населення базовим навичкам, стандартизація алгоритмів дій та юридичний захист тих, хто надає допомогу. Це сприяє більшій готовності громадян до активної участі в ситуаціях, які потребують невідкладного реагування.

Наукові дослідження, присвячені ефективності навчальних програм з першої допомоги, є важливим елементом для оцінки підготовки населення до реагування в кризових ситуаціях. Однією з основних цілей цих досліджень є визначення, наскільки ефективними є такі тренінги для осіб без медичної освіти, а також ідентифікація бар'єрів, що заважають якісному засвоєнню знань та навичок.

Одним із ключових досліджень є систематичний огляд шкільних програм з навчання першої допомоги. Цей огляд аналізував ефективність навчання базовим медичним навичкам серед школярів. У ньому було виявлено, що більшість програм зосереджуються на навчанні базової підтримки життя (BLS) [11], серцево-легеневої реанімації (СЛР) [12] та використання автоматичних зовнішніх дефібриляторів (AED) [13]. Навчання цим навичкам є критично важливим для підтримки життя в екстрених ситуаціях, таких як серцеві напади чи важкі травми.

Огляд програми навчання першої допомоги в школах показав, що короткі інтерактивні курси з використанням відеоматеріалів та манекенів для відпрацювання СЛР значно покращують навички учнів. Наприклад, у дослідженні на основі програми Injury Minimization Programme for Schools (IMPS) виявлено, що учні, які брали участь у тривалих практичних заняттях з надання першої допомоги, показали кращі результати порівняно з тими, хто проходив короткі теоретичні курси [14].

Дослідження показало, що основними перешкодами для ефективного навчання є страх перед невиконанням дій правильно, відсутність регулярного повторення матеріалу, а також обмежений доступ до матеріалів та обладнання для тренування. Наприклад, було відзначено, що в багатьох країнах не вистачає

дефібриляторів у громадських місцях, що ускладнює практичне засвоєння навичок з використання AED.

Також варто зазначити, що кілька досліджень підкреслили необхідність інтеграції програм з першої допомоги в загальноосвітній процес. Наприклад, у деяких країнах, таких як Німеччина та Норвегія, навчання навичкам першої допомоги є обов'язковим для отримання водійських прав, що значно підвищує рівень обізнаності населення про дії в кризових ситуаціях [15]. Це сприяє збільшенню кількості осіб, які можуть надати первинну допомогу у разі аварій або інших надзвичайних ситуацій.

Ефективність програм навчання першій допомозі залежить від багатьох чинників: практичного досвіду, регулярного оновлення знань і доступу до сучасних навчальних ресурсів. В цьому контексті впровадження інструментів, які можуть допомагати людям надавати допомогу в реальному часі, є дуже перспективним.

Інтерактивні інструменти, такі як чат-боти, мають великий потенціал для покращення надання допомоги. Вони можуть забезпечити постійний доступ до оновлених алгоритмів дій та допомогти користувачам долати психологічні бар’єри, пов’язані з невпевненістю в своїх знаннях. Автоматизовані рекомендації можуть підвищити точність і швидкість реагування в критичних ситуаціях, тим самим збільшуючи шанси на успішне надання допомоги.

Отже, інтеграція технологій та інструментів, що базуються на аналізі ситуацій, дозволяє покращити підготовку та підвищити готовність до надзвичайних ситуацій, що в кінцевому підсумку сприяє зменшенню рівня смертності та ускладнень у випадках, коли надання допомоги має вирішальне значення.

1.2 Аналіз існуючих мобільних додатків для медичної допомоги та їх аналогів

На сьогоднішній день, мобільні додатки для медичної допомоги набули широкого поширення завдяки зручності доступу до інформації та можливості

оперативного отримання рекомендацій у критичних ситуаціях. Бізнес-інфраструктура постійно розвивається, і за останнє десятиліття численні тенденції та інноваційні технології значно підвищили рівень конкуренції. Це змусило більшість компаній активно змагатися за свої ресурси. Серед основних чинників росту бізнесу варто виділити технології штучного інтелекту, 5G, Інтернет речей та інші [16][17]. Багато з цих додатків орієнтовані на надання користувачам інструкцій щодо першої допомоги, управління симптомами або навіть прямого зв'язку з медичними службами. Однак, ефективність таких додатків значною мірою залежить від якості інтегрованих алгоритмів та доступу до актуальних медичних даних.

До ключових прикладів мобільних додатків, які забезпечують медичну допомогу, належать «First Aid» від Червоного Хреста, "My3Medical" та "WebMD". Ці рішення мають набір функцій, що включають інтерактивні довідники першої допомоги, інструменти для діагностики симптомів та рекомендації щодо подальших дій. Незважаючи на їхню корисність, більшість додатків обмежені в точності рекомендацій через відсутність глибокої інтелектуальної обробки запитів та врахування індивідуальних особливостей користувача.

First Aid призначений для надання користувачам швидкої допомоги у надзвичайних ситуаціях (рис. 1.1). Додаток містить прості інструкції та інтерактивні поради щодо надання першої допомоги для різних типів травм або медичних станів, таких як кровотечі, опіки або серцеві напади. Основна мета – забезпечити користувачів чіткими кроками, які можна виконати до прибуття медичної допомоги.

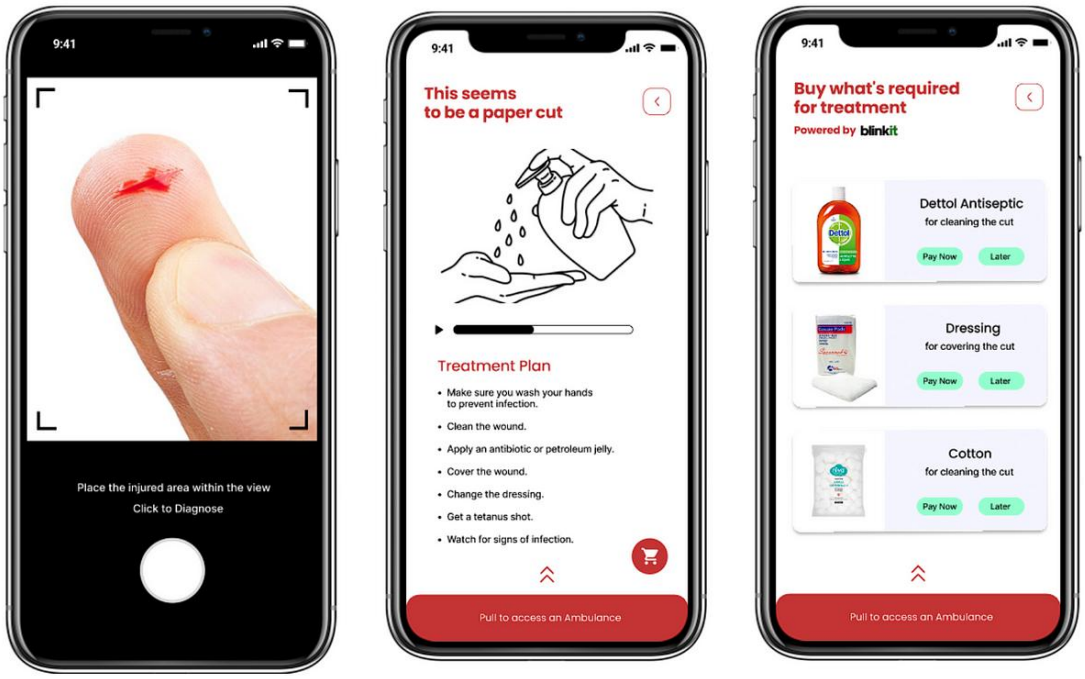

Рисунок 1.1 – Приклад інтерфейсу системи «First Aid»

Архітектура програми побудована на основі клієнт-серверної моделі, де основна частина даних зберігається на сервері, а користувацький інтерфейс реалізований на мобільному пристрої [18]. Додаток включає локальні бази даних для зберігання критичних інструкцій, що дозволяє доступ до них без інтернет-з'єднання. Важливим компонентом є інтеграція з GPS для надання рекомендацій про найближчі медичні заклади. Додаток також підтримує відеоінструкції, інтерактивні тести та вбудовані повідомлення для підтримки користувачів у надзвичайних ситуаціях.

Переваги:

– доступ до інструкцій офлайн, що дозволяє використовувати додаток без інтернет-з'єднання;

– інтерактивні відео та покрокові інструкції, які полегшують процес надання першої допомоги;

– простий та зрозумілий інтерфейс, орієнтований на швидке реагування;

– інтеграція з GPS для пошуку найближчих медичних установ [19].

Недоліки:

– відсутність персоналізованих рекомендацій, що може знизити ефективність допомоги в специфічних випадках [20];

– обмежені можливості для автоматичного аналізу симптомів користувача;

– відсутність можливостей для взаємодії з медичними працівниками в режимі реального часу;

– мала інтеграція із сучасними методами класифікації або штучного інтелекту для покращення точності рекомендацій.

Отже, First Aid від Американського Червоного Хреста є зручним та корисним інструментом для надання першої допомоги в надзвичайних ситуаціях, особливо завдяки можливості використання офлайн та інтуїтивному інтерфейсу. Проте, додаток має обмежені можливості щодо інтелектуальної обробки даних, що знижує його ефективність у складних випадках, де важлива персоналізація або швидкий аналіз симптомів.

Ada – це мобільний додаток, призначений для надання медичних рекомендацій та допомоги користувачам на основі симптомів, які вони вводять (рис. 1.2) [21]. Його основна мета – полегшити процес первинної самодіагностики, допомагаючи користувачам отримати попередню оцінку свого стану та рекомендації щодо подальших дій. Система обробляє введені дані про симптоми та ставить додаткові запитання для уточнення ситуації, після чого надає попередню медичну оцінку та можливі діагнози.

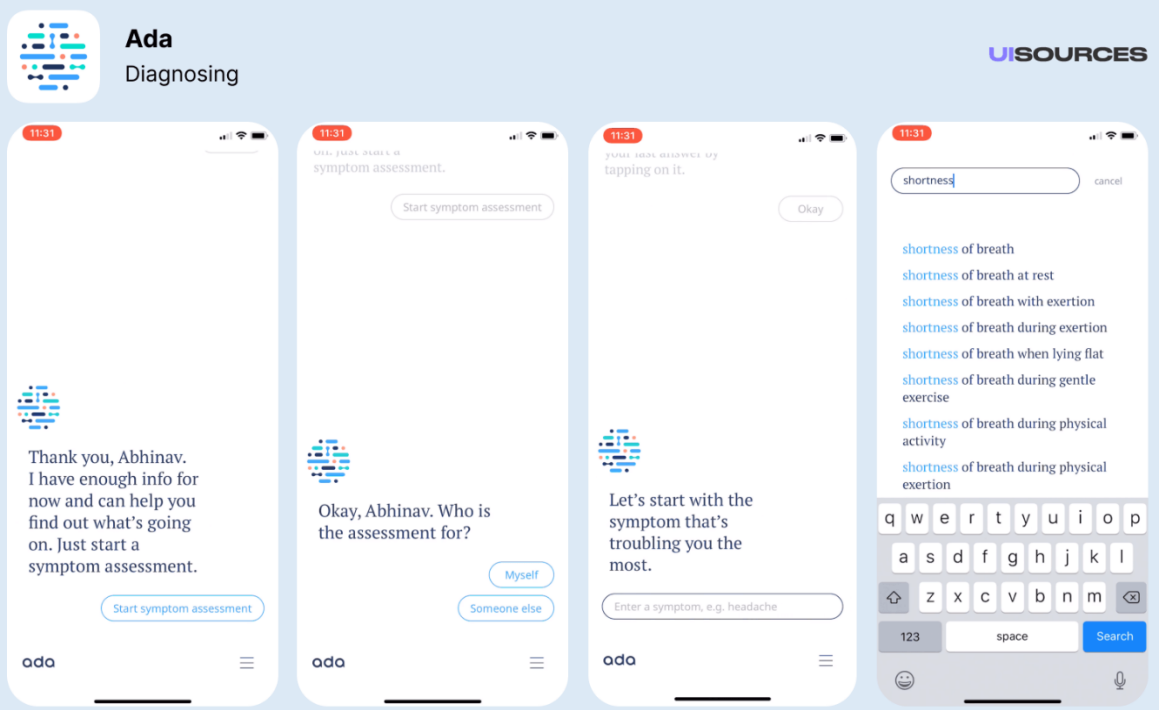

Рисунок 1.2 – Приклад інтерфейсу системи «Ada» [22]

Архітектура додатка заснована на штучному інтелекті та машинному навчанні, які працюють на серверній частині системи. На клієнтському боці, у мобільному додатку, реалізовано інтерфейс для збору симптомів та взаємодії з користувачем [23]. Всі дані про користувача передаються на сервер, де відбувається їх аналіз на основі великих медичних баз даних та попередньо навчених моделей. Ada постійно оновлює свої моделі та базу даних для забезпечення точності результатів. Система також забезпечує інтеграцію з електронними медичними записами, що дозволяє більш детально оцінювати історію хвороби користувача.

Переваги:

– використання штучного інтелекту для точнішого аналізу симптомів і надання індивідуальних медичних рекомендацій [24];

– інтерактивний процес збору даних, який дозволяє отримувати більш детальну інформацію про симптоми користувача;

– постійне оновлення бази знань і моделей, що підвищує точність діагностики;

– можливість інтеграції з електронними медичними записами для більш глибокої оцінки стану здоров'я користувача.

Недоліки:

– відсутність прямого зв’язку з медичними працівниками для підтвердження діагнозу [25];

– може пропонувати широке коло можливих діагнозів, що іноді призводить до невизначеності у виборі правильних дій;

– обмежена підтримка різних мов, що звужує коло користувачів;

– для використання деяких функцій потрібне стабільне інтернет-з'єднання, що обмежує доступність у деяких ситуаціях.

Отже, Ada є потужним інструментом для попередньої медичної діагностики на основі симптомів, який відрізняється високим рівнем інтелектуальної обробки даних та адаптивністю до індивідуальних потреб користувача. Незважаючи на високу точність і можливості персоналізації, додаток має певні обмеження, зокрема, відсутність прямої взаємодії з лікарями та необхідність доступу до інтернету для повноцінної роботи, що може бути критичним у деяких ситуаціях.

Doctor-on-Demand – це мобільний додаток, призначений для забезпечення користувачів можливістю отримувати консультації від ліцензованих медичних працівників через відеозв’язок (рис. 1.3) [26]. Основна мета додатка – надання швидкої і зручної дистанційної медичної допомоги без потреби відвідувати лікарню або медичний центр. Через платформу користувачі можуть звертатися до лікарів для вирішення різноманітних питань, від незначних симптомів до серйозніших проблем, таких як психічне здоров'я або хронічні захворювання.

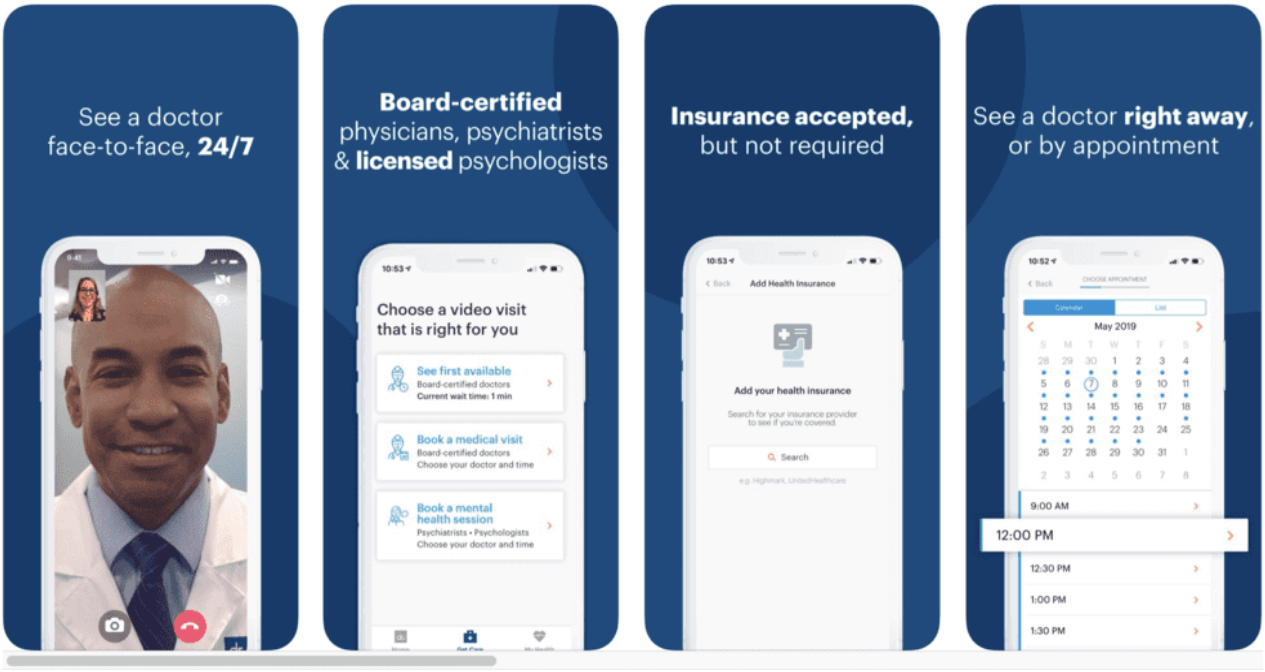

Рисунок 1.3 – Приклад інтерфейсу системи «Doctor-on-Demand» [27]

Архітектура Doctor-on-Demand побудована на основі клієнт-серверної моделі, де на стороні клієнта знаходиться мобільний додаток, що забезпечує користувацький інтерфейс для реєстрації, вибору спеціаліста та проведення відеоконсультацій [28]. Серверна частина відповідає за опрацювання користувацьких запитів, збереження електронних медичних записів, планування візитів і керування базою даних лікарів. Платформа підтримує інтеграцію з платіжними системами для здійснення оплати медичних послуг, а також з медичними страхуваннями, що робить її зручною для широкого кола користувачів. Дані про пацієнтів передаються захищеними каналами зв'язку, що гарантує конфіденційність медичної інформації.

Переваги:

– можливість отримати консультацію від ліцензованого лікаря через відеозв'язок у зручний для користувача час [29];

– швидкий доступ до медичної допомоги без необхідності відвідувати медичний заклад;

– інтеграція з медичними страховками, що дозволяє покривати вартість послуг через страхові компанії;

– платформа підтримує широкий спектр медичних послуг, включаючи як фізичне, так і психічне здоров'я.

Недоліки:

– залежність від стабільного інтернет-з’єднання, без якого неможливо здійснювати відеоконсультації;

– обмеження у наданні допомоги при серйозних медичних станах, що потребують фізичного огляду [30];

– висока вартість послуг для користувачів без страхування;

– можливі обмеження в доступності лікарів залежно від регіону та часу.

Отже, Doctor-on-Demand є зручним рішенням для тих, хто потребує оперативної медичної консультації від кваліфікованого фахівця, не виходячи з дому. Додаток забезпечує швидкий доступ до лікарів та широкий спектр медичних послуг, що робить його корисним для повсякденного використання. Однак він має обмеження, зокрема залежність від інтернету та неможливість проведення фізичних оглядів, що робить його менш придатним для вирішення складних медичних випадків.

Аналіз існуючих мобільних додатків для надання медичної допомоги, таких як First Aid, Ada та Doctor-on-Demand, виявив ряд обмежень, які можна усунути через розробку Telegram-бота для автоматизованої рекомендації першої медичної допомоги. Зокрема, наявні додатки або вимагають постійного інтернет-з'єднання для консультацій з лікарями, або мають обмежені можливості персоналізації рекомендацій на основі введених симптомів. Telegram-бот може стати доступним рішенням, оскільки він дозволяє швидко надавати рекомендації у реальному часі навіть у середовищах з низькою доступністю медичних послуг. Завдяки автоматизованим алгоритмам класифікації, бот може надавати точні й персоналізовані рекомендації, що значно підвищить ефективність первинної медичної допомоги. Крім того, інтеграція з популярною платформою Telegram забезпечить зручність і доступність для широкого кола користувачів.

1.3 Аналіз методів класифікації в машинному навчанні та їх застосування для надання медичних рекомендацій

Методи класифікації в машинному навчанні відіграють важливу роль у сфері медичних систем, що надають автоматизовані рекомендації на основі аналізу симптомів та стану пацієнта. Класифікація дозволяє системам, побудованим на основі машинного навчання, правильно ідентифікувати певні категорії захворювань або медичних станів, ґрунтуючись на великій кількості вхідних даних. Основна задача полягає в тому, щоб на основі обмеженої кількості симптомів, які вводить користувач, точно визначити можливий діагноз або надати рекомендації щодо подальших дій.

Використання методів класифікації, таких як випадкові ліси, метод опорних векторів, дерева рішень або нейронні мережі, дозволяє системам інтегрувати широкий спектр медичних даних: від аналізу результатів лабораторних досліджень до історій хвороб пацієнтів. Наприклад, модель може аналізувати демографічну інформацію пацієнта, як-от вік і стать, а також враховувати його анамнез, поточні скарги та результати обстежень. Такі алгоритми використовуються для визначення ризиків розвитку серцево-судинних захворювань, діабету, онкологічних хвороб тощо.

Окрім діагностики, методи класифікації застосовуються для надання рекомендацій щодо лікування. Наприклад, система може запропонувати перелік можливих медикаментів чи терапевтичних процедур, враховуючи персональні особливості пацієнта та його історію захворювання. Більш того, сучасні медичні рекомендаційні системи здатні вказати на необхідність додаткових обстежень, які можуть уточнити стан пацієнта, або спрямувати користувача до спеціалізованого лікаря. Завдяки цьому зменшується ймовірність помилкового діагнозу та оптимізується процес медичного обслуговування.

Варто відзначити, що точність таких систем великою мірою залежна від повноти вхідних даних та їх якості. Навчання моделей класифікації проводиться на основі великих наборів даних, які включають як негативні, так і позитивні приклади медичних станів. Такий підхід забезпечує не лише високу точність

прогнозів, а й адаптивність системи до нових симптомів або атипових проявів захворювань.

Логістична регресія є одним із найпоширеніших методів класифікації, який використовується для оцінки ймовірності настання певної події, особливо коли результат може бути тільки бінарним – тобто має лише два можливі значення (наприклад, 0 або 1, "так" або "ні") [31]. Цей метод ідеально підходить для задач, де потрібно передбачити належність до одного з двох класів. Основною задачою логістичної регресії є побудова математичної моделі, що описує зв’язок між набором незалежних змінних та ймовірністю певного результату.

Загальну форму регресійної моделі можна записати як рівняння:

$$y = F(x_1, x_2, \dots x_n)\,, \tag{1.1}$$

де y – це залежна змінна, а x_1,x_2,...,x_n – незалежні змінні. Однак, коли йдеться про класифікацію, зокрема бінарну, стандартна регресія не враховує, що результат може бути обмеженим двома значеннями. Це призводить до того, що моделі можуть передбачати значення поза межами діапазону від 0 до 1, що неприйнятно для ймовірностей. Для вирішення цієї проблеми використовують логістичну функцію (логіт-перетворення), яка обмежує результат значеннями між 0 та 1, що краще відповідає ймовірнісній природі завдання. Формула цього перетворення виглядає так:

$$p = \frac{1}{1+e^{-y}}\,, \tag{1.2}$$

де p – це ймовірність настання події, e – математична константа (близько 2.71), а y – стандартне рівняння регресії. Логістична регресія таким чином дозволяє моделювати ймовірність належності об'єкта до певного класу на основі його характеристик.

На рис. 1.4 представлена класична логістична функція (сигмоїда), яка описує залежність ймовірності настання певної події від значення лінійної комбінації вхідних змінних [32]. Логістична функція зазвичай застосовується в класифікаційних задачах для перетворення вихідної величини в діапазон від 0 до 1, що представляє ймовірність. Як видно з графіку, значення функції різко змінюється в області близько 0, і в цій точці ймовірність події дорівнює 0,5. Для значень на вході, що перевищують 0, ймовірність швидко наближається до 1, тоді як для від'ємних значень вона прагне до 0. Така властивість логістичної функції дозволяє ефективно застосовувати її для класифікації на два класи, де порогове значення на рівні 0,5 є роздільною межею між класами.

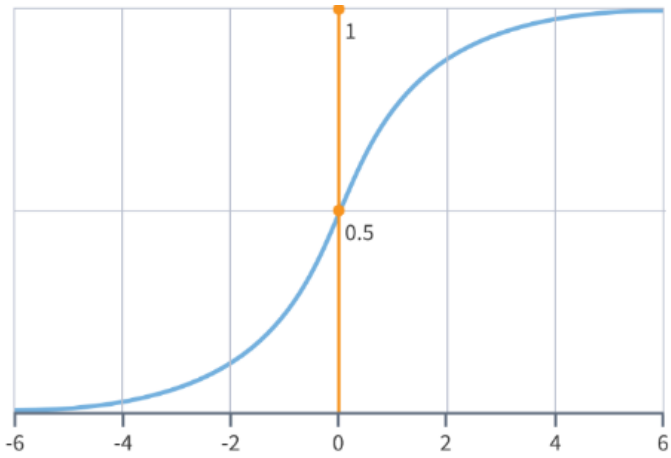

Рисунок 1.4 – Ілюстрація логістичної функції

Така залежність показує, як навіть незначні зміни вхідних даних у межах значень близьких до нуля можуть істотно вплинути на кінцеву ймовірність. В той час як екстремальні значення *x* практично не змінюють результат, залишаючи його близьким до 0 або 1. Таким чином, логістична функція слугує природним способом моделювання ймовірностей для задач бінарної класифікації, дозволяючи інтерпретувати результат як ймовірність належності об'єкта до одного з двох класів.

Дерево прийняття рішень є одним із найпопулярніших рішень інтелектуального аналізу даних та передбачувальної аналітики, що застосовується для вирішення задач класифікації та регресії [33]. Його розробка бере початок майже 40 років тому завдяки Аделю Катлеру та Лео Брейману, які

впровадили цей алгоритм для аналізу складних даних. Структурно дерево прийняття рішень нагадує природне дерево, де кожен вузол відповідає певній ознаці або питанню, а "гілки" представляють можливі шляхи або відповіді, що ведуть до наступних вузлів (рис. 1.5).

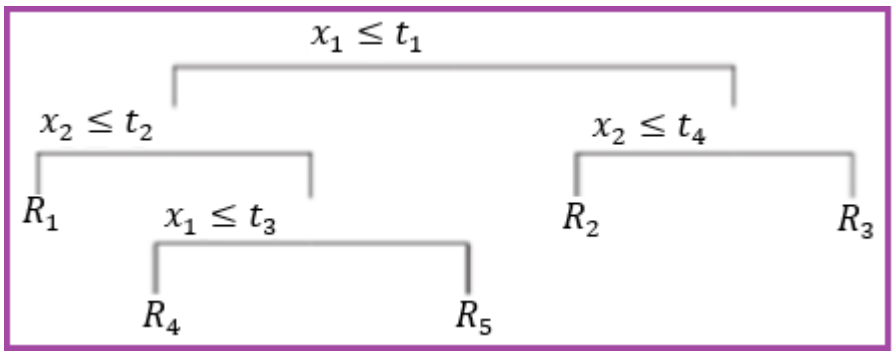

Рисунок 1.5 – Структура дерева прийняття рішень

Головна особливість такого алгоритму полягає у тому, що кожен вузол є певним критерієм розподілу даних на підмножини. Вузли, що не мають нащадків, називаються листками і представляють остаточне рішення задачі класифікації або прогнозування. Дерево будується шляхом автоматичного генерування правил виду «Якщо … то …», і ці правила дозволяють алгоритму класифікувати нові дані. Залежно від типу задачі, дерево може бути класифікаційним (для дискретних цільових змінних) або регресійним (для неперервних змінних).

На відміну від складних нейронних мереж, дерева рішень прості в інтерпретації, оскільки їхні правила формулюються природною мовою, що полегшує їхнє розуміння. Наприклад, у медицині: «Якщо температура пацієнта перевищує 38°C і супроводжується кашлем, то існує ймовірність вірусної інфекції» Процес навчання дерева рішень базується на індукції: алгоритм узагальнює навчальні приклади і будує модель, що дозволяє класифікувати нові дані [34]. Завдяки своїй гнучкості і простоті, алгоритм дерева рішень використовується в багатьох галузях для вирішення різноманітних задач класифікації, розділяючи дані на підмножини, поки не буде отримано достатньо однорідні групи.

Часовий ряд x, що складається з певної послідовності значень, можна представити як кінцеву впорядковану множину:

$$x = [x(1), \dots, x(t)]\,. \tag{1.3}$$

Дерево прийняття рішень розділяє весь простір об'єктів на J окремих областей: $R_1,\dots,R_J$. У кожній з цих областей дерево повертає постійне значення $b_1,\dots,b_J$.

Таким чином, модель дерева рішень можна записати як суму:

$$b(x) = \sum_{j=1}^{J}[x \in R_j]b_J\,, \tag{1.4}$$

де x – це аналізований об'єкт, а b_J – ймовірність того, що об'єкт xx належить до J-го класу в області R_J.

Для побудови дерева прийняття рішень використовуються різноманітні критерії оцінювання якості розбиття даних. Одним із таких критеріїв є ентропія Шеннона, яка вимірює ступінь невпорядкованості або «хаосу» в системі. Чим вища ентропія, тим менш впорядкована система:

$$S = -\sum_{i=1}^{n} p_i log_2 p_i\,, \tag{1.5}$$

де p_i – це ймовірність того, що система знаходиться в i-му стані.

Зменшення ентропії при кожному розбитті називається приростом інформації, і його можна обчислити за допомогою формули:

$$IG(Q) = S_0 - \sum_{i=1}^{m}\frac{n_i}{n}S_i\,, \tag{1.6}$$

де Q – це ознака, за якою здійснюється розбиття; S_i – ентропія в i-му вузлі дерева; m – кількість груп розбиття; n_i – кількість об'єктів, для яких ознака Q має i-те значення.

Інший критерій, який часто застосовується в алгоритмах побудови дерев рішень, – це індекс Джині. Він спрямований на мінімізацію ймовірності помилкової класифікації та має наступну математичну форму:

$$G = 1 - \sum_{i=1}^{n} p_i^2 \,, \tag{1.7}$$

де p_i – це ймовірність того, що об’єкт належить до іі-го класу. Індекс Джині дозволяє обчислити ступінь чистоти підмножин після кожного розбиття даних, що сприяє точнішій класифікації.

Ще одним важливим критерієм є оцінка помилки класифікації. Вона використовується для підрізання дерев, що допомагає запобігти їх надмірному розростанню, особливо коли це призводить до перенавчання. Формула для розрахунку помилки виглядає так:

$$E = 1 - max_i p_i \,, \tag{1.8}$$

де p_i – максимальна ймовірність серед усіх класів. Цей показник допомагає ідентифікувати вузли, де алгоритм схильний до помилкових рішень, і підрізати зайві гілки.

За замовчуванням у багатьох реалізаціях дерев рішень застосовується індекс Джині як базовий критерій для оцінки якості розбиття. Окрім цього, важливим параметром при побудові дерева є його глибина. Обмеження глибини дерева дозволяє уникнути його надмірного ускладнення та хаотичного розростання, що може призвести до перенавчання. Чим більше гілок і вузлів у дереві, тим більше ймовірність, що воно почне занадто точно підлаштовуватися під тренувальні дані, втрачаючи загальну здатність до узагальнення.

Правильний вибір критеріїв розбиття, таких як індекс Джині чи ентропія Шеннона, а також контроль глибини дерева є ключовими елементами для побудови ефективної моделі, здатної уникати перенавчання та зберігати високу точність класифікації на нових даних [35].

Однак, навіть з правильним вибором критеріїв розбиття та контролем глибини дерева, дерево рішень залишається вразливим до проблеми перенавчання, особливо на складних та неоднорідних даних. Невеликі зміни у навчальній вибірці можуть суттєво вплинути на структуру дерева, що призводить до нестабільності моделі та втрати її здатності до узагальнення. У таких випадках використання ансамблевих методів, які об’єднують кілька моделей, може значно підвищити надійність та точність класифікації.

Одним із найбільш ефективних ансамблевих методів є алгоритм «випадковий ліс» (Random Forest), що вирішує проблему перенавчання шляхом об’єднання прогнозів кількох дерев рішень (рис. 1.6) [36]. Основна ідея цього підходу полягає у використанні двох випадкових процесів під час побудови кожного дерева: по-перше, створюються випадкові навчальні вибірки (бутстрап-вибірки), по-друге, на кожному етапі вибираються випадкові підмножини ознак для розщеплення вузлів. Така випадковість дозволяє зменшити вплив особливостей конкретних даних і підвищити стабільність моделі. Після того, як всі дерева побудовані, підсумкове рішення приймається на основі мажоритарного голосування: обирається клас, який отримав найбільше підтримки від усіх дерев.

Бутстрап-вибірка формується шляхом випадкового вибору об’єктів із початкового набору даних з поверненням [37]. Це означає, що деякі об’єкти можуть з’явитися кілька разів у одній вибірці, тоді як інші можуть бути відсутні. Така техніка дозволяє зменшити залежність дерев від конкретної вибірки даних, забезпечуючи кращу узагальнюючу здатність моделі.

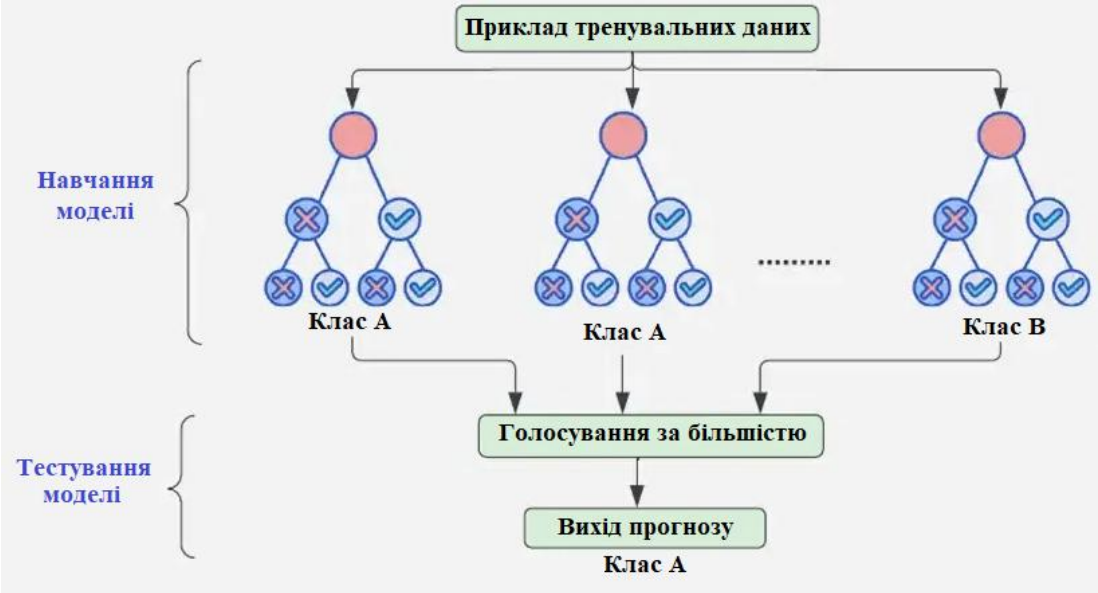

Рисунок 1.6 – Принцип роботи методу «випадковий ліс»

Для кожного дерева T_i формується бутстрап-вибірка D_i, що складається з об’єктів початкового набору даних D, але вибраних з поверненням. Це означає, що деякі об’єкти можуть з’являтися кілька разів у одній вибірці, тоді як інші можуть бути відсутні.

У кожному вузлі дерева T_i вибирається випадкова підмножина ознак розміру mm із загальної кількості ознак M. Для кожної вибраної ознаки обчислюється критерій розщеплення (наприклад, інформаційний приріст або індекс Джині), і вибирається найкраща ознака для розщеплення вузла.

Дерево T_i будується на бутстрап-вибірці D_i до моменту, поки не буде досягнуто певної максимальної глибини або не залишиться достатньо об’єктів у вузлах для подальшого розщеплення. Дерево не підрізається, що дає йому можливість навчатися на глибоких структурах даних.

Для класифікаційних задач кожне дерево T_i видає прогноз для нового об’єкта x, а остаточний результат класифікації визначається мажоритарним голосуванням серед усіх дерев у лісі.

Формально:

$$\hat{y} = mode(\{T_i(x)|i = 1,2, \dots, N\}) , \quad (1.9)$$

де N – кількість дерев у лісі, $T_i(x)$ – прогноз іі-го дерева для об'єкта x, а $\hat{y}$ – кінцева прогнозована мітка класу.

У задачах регресії кожне дерево T_i дає числовий прогноз, і кінцевий результат визначається як середнє значення прогнозів усіх дерев:

$$\hat{y} = \frac{1}{N}\sum_{i=1}^{N} T_i(x) , \quad (1.10)$$

де $\hat{y}$ – прогнозована числова величина.

Метод випадкових лісів знижує ризик перенавчання завдяки випадковості у виборі даних і ознак для кожного дерева, що робить його стійким до шуму в даних і дозволяє побудувати більш точні прогнози [38].

Застосування методів класифікації в медичних додатках для надання рекомендацій є критично важливим для забезпечення точності й надійності таких систем. Класифікаційні моделі дозволяють швидко аналізувати симптоми і пропонувати користувачам відповідні медичні рекомендації, базуючись на великому обсязі історичних медичних даних. У поєднанні з сучасними технологіями, такими як хмарні обчислення та штучний інтелект, класифікаційні моделі можуть значно покращити якість медичної допомоги, знижуючи ризики помилкових діагнозів та підвищуючи оперативність реагування на медичні запити користувачів.

1.4 Формулювання задачі для розробки системи

Однією з актуальних проблем сучасного суспільства є надання оперативної та якісної медичної допомоги у ситуаціях, коли доступ до професійного лікаря є ускладненим або неможливим. У таких випадках технологічні рішення, зокрема чат-боти, можуть стати ефективним інструментом для підтримки здоров'я людей. Створення Telegram-бота для надання першої медичної допомоги дозволить забезпечити користувачів оперативними рекомендаціями та інструкціями в реальному часі, що може мати критичне значення для збереження життя та здоров'я.

Для визначення цієї актуальної проблеми визначено низку функціональних та нефункціональних вимог, які забезпечать ефективність роботи бота та його зручність для користувачів. Розроблений бот буде спрямований на надання персоналізованих рекомендацій, опираючись на сучасні технології машинного навчання, аналізу даних і алгоритмах обробки природної мови.

Функціональні вимоги

Чат-бот, орієнтований на надання першої допомоги, володітиме низкою важливих функцій, які підвищать його ефективність та корисність:

– навчання на основі реальних даних. Бот буде тренуватися на основі даних про випадки надання першої допомоги. Завдяки використанню алгоритмів машинного навчання та технологій обробки природної мови, бот зможе аналізувати запити користувачів і надавати коректні рекомендації, враховуючи контекст;

– збереження навчальних моделей. Моделі, що продемонструють високу точність і ефективність, будуть збережені для подальшого використання, щоб уникнути необхідності повторного навчання й забезпечити стабільність результатів;

– інтеграція з Telegram. Бот буде повністю інтегровано з платформою Telegram, що дозволить користувачам легко взаємодіяти з ним через API месенджера, використовуючи зручні команди та налаштований інтерфейс;

– інтерактивність та підтримка діалогу. Бот буде здатний підтримувати природну взаємодію з користувачем, відповідаючи на запитання, утримуючи контекст розмови та адаптуючи відповіді до змінних умов спілкування;

– персоналізовані консультації. Користувачі зможуть обирати конкретні теми, які їх цікавлять, наприклад, серцево-легенева реанімація або зупинка кровотечі, що дозволить боту надавати релевантні рекомендації у відповідних випадках;

– збір зворотного зв'язку. Користувачам буде надана можливість залишати відгуки про роботу бота, що сприятиме подальшому покращенню його функціональності та якості обслуговування.

Нефункціональні вимоги

Для успішної реалізації Telegram-бота важливо враховувати не лише функціональні, а й нефункціональні аспекти, які визначають загальну якість роботи системи та впливають на користувацький досвід. Ці вимоги забезпечують стабільність, безпеку та продуктивність системи. До основних нефункціональних вимог відносяться:

– продуктивність. Бот повинен оперативно обробляти запити, незалежно від кількості користувачів або обсягів даних. Час відповіді повинен бути максимально швидким, не перевищуючи кількох секунд, щоб підтримувати швидку й плавну взаємодію з користувачем;

– масштабованість. Система має бути побудована з урахуванням можливого зростання кількості користувачів і запитів. Це забезпечить збереження високого рівня продуктивності навіть при підвищеному навантаженні;

– безпека. Захист персональних даних користувачів є важливим елементом роботи бота. Система повинна включати сучасні методи шифрування, безпечної передачі й зберігання даних, а також бути стійкою до несанкціонованого доступу та кібератак;

– надійність. Для надання безперервної підтримки в режимі 24/7 бот повинен бути стабільним і мінімізувати кількість збоїв або простоїв. Важливо

забезпечити швидке виявлення й усунення можливих помилок, щоб користувачі могли завжди отримувати необхідну допомогу;

– сумісність. Бот має бути доступним для використання на різних платформах і пристроях, включаючи мобільні телефони та комп'ютери з різними операційними системами. Це гарантуватиме його доступність для широкого кола користувачів;

– зручність використання. Інтерфейс користувача має бути простим та інтуїтивно зрозумілим, щоб користувачі будь-якого віку і рівня технічної підготовки могли легко взаємодіяти з ним. Це сприятиме комфортному використанню системи й забезпечить позитивний досвід для всіх користувачів.

1.5 Висновок

У рамках даного розділу було проведено детальний аналіз нормативно-правової бази та міжнародних стандартів щодо надання першої медичної допомоги. Було розглянуто основні закони України, зокрема Закон України "Про екстрену медичну допомогу", Кримінальний кодекс України (ст. 136), а також міжнародні стандарти, такі як закони "доброго самарянина" і рекомендації Всесвітньої організації охорони здоров’я та Міжнародної федерації товариств Червоного Хреста і IFRC. Це дало змогу визначити правову основу для функціонування системи, що забезпечує надання першої медичної допомоги, а також вивчити існуючі міжнародні підходи.

Проаналізовано існуючі мобільні додатки, такі як First Aid, Ada та Doctor-on-Demand. Виявлено, що кожен із них має свої переваги, але всі вони стикаються з обмеженнями, що можуть бути вирішені шляхом створення Telegram-бота для автоматизованої рекомендації першої допомоги. На основі цього аналізу сформовано основні вимоги до розроблюваного бота, що дозволить уникнути недоліків, характерних для існуючих рішень, і забезпечить покращення користувацького досвіду.

Розглянуто основні методи класифікації в машинному навчанні, які можуть бути застосовані для побудови системи надання медичних рекомендацій.

Логістична регресія, дерева рішень і метод випадкових лісів стали ключовими підходами для реалізації ефективної та надійної системи. Ці алгоритми дозволяють класифікувати запити користувачів і надавати релевантні рекомендації на основі аналізу вхідних даних.

На основі проведеного аналізу було сформульовано функціональні та нефункціональні вимоги до розробки Telegram-бота. Отримані результати дозволять обґрунтувати вибір необхідних технологій для реалізації програмного рішення та визначити архітектуру системи, яка забезпечить високу ефективність, масштабованість і безпеку.

2 ОБҐРУНТУВАННЯ ВИБОРУ ТЕХНОЛОГІЙ ТА ОСОБЛИВОСТІ РОЗРОБКИ ПРОГРАМНОГО РІШЕННЯ

2.1 Обґрунтування вибору технологій та архітектури системи

Цифрові технології стали фундаментом для розробки нових продуктів, цінностей і характеристик, що, в свою чергу, створює можливості для здобуття конкурентних переваг на численних ринках. [39]. Дослідження цифрової зрілості підприємств та її впливу на результативність бізнесу активно проводяться. Водночас, застосування штучного інтелекту для аналізу даних, пов'язаних з цифровою зрілістю, здатне виявляти фальшиві новини та дезінформацію, що може мати негативний вплив на бізнес-структури [40]. Тому важливими є дослідження, спрямовані на створення систем для оцінки процесів цифрової трансформації бізнесу, а також на забезпечення проведення періодичних аналізів цифрового прогресу і впровадження регулярних, структурованих статистичних моніторингів [41].

Вибір технологій та архітектури для розробки системи автоматизованої рекомендації першої медичної допомоги є основним етапом, який визначає ефективність, ефективність та масштабованість розроблюваного рішення. Система, побудована у форматі Telegram-бота, потребує інтеграції низки технологій, що забезпечують як безперервний доступ до сервісу користувачів, так і високу точність класифікації для рекомендаційної складової. Платформа Telegram, завдяки широкій популярності, інтуїтивному інтерфейсу та наявності API для розробки ботів, надає ідеальні можливості для створення простого та ефективного каналу комунікації з користувачами. Окрім функціональних переваг Telegram, важливим є також вибір технологій, що доповнять його можливості для досягнення цілей системи, зокрема для підтримки стабільності, масштабованості та інтеграції з іншими сервісами, такими як бази даних та зовнішні API.

Сучасні інструменти розробки, включаючи різноманітні мови програмування, фреймворки і платформи, дозволяють підлаштувати систему під

конкретні технічні вимоги та забезпечити її гнучкість для майбутнього розширення. У розробці Telegram-ботів для автоматизованої обробки запитів доцільним є використання таких мов, як Python, Java або C#, кожна з яких має свої переваги щодо продуктивності, підтримки інтеграції та можливостей роботи з алгоритмами класифікації. Це забезпечує більш точний аналіз та обробку даних для надання рекомендацій у сфері першої медичної допомоги.

Python – інтерпретована мова програмування, відома своєю простотою та читабельністю, що полегшує швидкий розвиток і підтримку коду. Вона широко використовується в області машинного навчання та наукових обчислень завдяки численним бібліотекам та інструментам для аналізу даних. Python забезпечує високу продуктивність при створенні прототипів і є зручною для розробки крос-платформних застосунків.

Java – об’єктно-орієнтована мова програмування, яка є стабільною та достатньо масштабною, тому вона підходить для розробки великих корпоративних систем. Завдяки віртуальній машині Java (JVM) вона є крос-платформною, а її сильна типізація робить код більш надійним. Java активно використовується для веб-додатків, мобільних програм під Android та складних серверних систем.

C# – мова програмування від Microsoft, яка найкраще підходить для розробки Windows-додатків і веб-служб. Вона підтримує сучасні об'єктно-орієнтовані можливості та тісно інтегрується з .NET-платформою, що дозволяє створювати продуктивні та масштабовані програми. C# також підтримує зручну розробку ігор через Unity, що робить її популярною у різних галузях.

У табл. 2.1 наведено ключові аспекти продуктивності, зручності інтеграції, підтримки багатопотоковості та інших властивостей розглянутих мов програмування.

Таблиця 2.1 – Порівняння мов програмування

Характеристика	Python	Java	C#
Швидкість виконання коду	Середня	Висока	Висока
Багатопотоковість	Підтримка обмежена (GIL)	Добра	Висока
Інтеграція з Windows	Обмежена	Середня	Висока (особливо з .NET платформою)
Підтримка компіляції	Інтерпретована	Байткод (JVM)	Компіляція в машинний код
Наявність статичної типізації	Відсутня	Є	Є
Продуктивність у середовищах з високим навантаженням	Середня	Висока	Висока
Графічні інтерфейси	Підтримка сторонніх бібліотек	JavaFX	Інтегровано в .NET (Windows Forms, WPF)

Виходячи з аналізу, мову C# обрано для розробки системи завдяки її високій продуктивності та оптимальній підтримці багатопотоковості, що є важливим для швидкої обробки даних і збереження стабільної роботи в умовах високого навантаження. Завдяки інтеграції з платформою .NET, C# забезпечує ефективну розробку і гнучке налаштування графічного інтерфейсу, що полегшує створення зручних користувацьких інтерфейсів і взаємодію з системами на базі Windows. Окрім того, статична типізація та можливість компіляції в машинний код підвищують стабільність та безпеку коду, зменшуючи ймовірність помилок у процесі виконання. Інтеграція з іншими сервісами та базами даних також є безперешкодною завдяки розширеним можливостям платформи .NET, що забезпечує надійний функціонал для розробки високоякісного програмного продукту.

Оскільки мову програмування C# обрано для розробки системи, наступним кроком є вибір відповідної архітектури, яка забезпечить ефективне розподілення функціональних частин і дозволить реалізувати гнучкий, масштабований дизайн.

Трьохрівнева архітектура складається з трьох основних рівнів: користувацького інтерфейсу, прикладної логіки та рівня зберігання даних [42].

Такий підхід розділяє відповідальність між окремими компонентами, що дозволяє легко змінювати або оновлювати один рівень без впливу на інші. Це особливо корисно для великих систем, де підтримка, тестування та масштабування можуть виконуватись незалежно для кожного рівня.

Мікросервісна архітектура полягає в розподілі програми на окремі, автономні сервіси, кожен з яких виконує конкретну функцію [43]. Кожен мікросервіс є автономним і мо же бути розроблений та розгорнутий окремо, що значно спрощує підтримку та розширення системи. Завдяки такій гнучкості, мікросервіси добре підходять для проєктів з високими вимогами до масштабованості та частих оновлень.

Сервіс-орієнтована архітектура (SOA) орієнтована на інтеграцію різних сервісів, що забезпечують бізнес-логіку у вигляді окремих компонентів, які можуть взаємодіяти через стандартизовані протоколи [44]. SOA дозволяє створювати систему, де сервіси можуть бути викликані як самостійні модулі та об'єднані для створення комплексних бізнес-процесів. Це сприяє легкому повторному використанню компонентів і забезпечує високу гнучкість при інтеграції з іншими системами.

Для кращого розуміння особливостей та можливостей кожної з розглянутих архітектур, у табл. 2.2 наведено їх порівняння за основними характеристиками.

Таблиця 2.2 – Порівняння мов програмування

Характеристика	Трьохрівнева архітектура	Мікросервісна архітектура	Сервіс-орієнтована архітектура (SOA)
Простота розгортання	Висока, один комплексний додаток	Середня, незалежне розгортання	Середня, вимагає інтеграції сервісів
Легкість підтримки	Висока, всі рівні у межах однієї системи	Середня, залежить від окремих сервісів	Середня, потрібне управління сервісами

Продовження таблиці 2.2

Масштабованість	Помірна, масштабування всієї системи	Висока, кожен сервіс окремо	Висока, модульність сервісів
Продуктивність	Висока, мінімальна міжрівнева взаємодія	Середня, залежить від комунікації між сервісами	Середня, потрібні ресурси для сервіс-орієнтованого зв'язку
Кількість компонентів	3 основні рівні	Залежить від функціоналу, часто понад 10	Залежить від функцій, зазвичай 5-10
Час на реалізацію	Відносно короткий	Довший, потребує окремого проектування кожного сервісу	Довший, потрібна стандартизація протоколів
Придатність для великих систем	Висока, особливо для систем середнього розміру	Дуже висока, підходить для великих систем	Висока, з урахуванням вимог до взаємодії сервісів

Трьохрівнева архітектура є оптимальним вибором для розробки системи автоматизованих рекомендацій першої медичної допомоги завдяки своїй простоті розгортання та високій продуктивності. Вона забезпечує чітке розділення на рівні інтерфейсу, прикладної логіки та зберігання даних, що спрощує підтримку та внесення змін у будь-який з рівнів без впливу на інші. Така архітектура дозволяє швидко реалізувати основні функції з мінімальними витратами часу та ресурсів, а також легко масштабувати систему для обробки запитів великої кількості користувачів. Крім того, продуктивність та зручність підтримки трьохрівневої архітектури роблять її ідеальною для проєктів середнього розміру, забезпечуючи баланс між складністю та ефективністю роботи системи.

2.2 Проєктування діаграм варіантів використання системи

Проєктування діаграм варіантів використання дозволяє визначити взаємодії користувачів із системою та зрозуміти основні сценарії її

використання. Для побудови ефективної системи автоматизованих рекомендацій з першої медичної допомоги через Telegram-бот необхідно чітко окреслити ключові функції, які мають бути доступні користувачам, а також визначити ролі користувачів і їхні можливі дії. Розробка діаграм варіантів використання допомагає структурувати ці вимоги, формуючи загальне уявлення про функціонал системи та взаємодію її компонентів. У табл. 2.3 наведено детальний опис основних варіантів використання, що охоплюють різні аспекти функціональності системи.

Таблиця 2.3 – Опис варіантів використання системи

№	Ім’я	Опис
1	2	3
UC1	Реєстрація користувача	Забезпечує користувачам доступ до створення власного профілю в системі
UC2	Автентифікація користувача	Дозволяє користувачам підтвердити свою особу для входу до системи
UC3	Перегляд списку користувачів	Надає адміністраторам можливість переглядати інформацію про всіх зареєстрованих користувачів
UC4	Додати користувача	Дозволяє адміністраторам вносити нових користувачів у систему
UC5	Оновлення даних користувача	Дозволяє адміністраторам редагувати інформацію про користувачів у базі даних
UC6	Перегляд каталогу тем	Забезпечує користувачам доступ до списку доступних тем у системі
UC7	Створення нової теми	Надає можливість створення нових тем у межах системи
UC8	Редагування існуючої теми	Дозволяє оновлювати або коригувати інформацію про вже наявні теми
UC9	Перегляд моделей	Дозволяє користувачам переглядати перелік доступних для використання моделей
UC10	Додавання нової моделі	Дає можливість користувачам створювати та інтегрувати нові моделі машинного навчання
UC11	Видалення моделі	Дозволяє користувачам видаляти неактуальні моделі з системи

Продовження таблиці 2.3

UC12	Перевірка моделей	Надає користувачам можливість тестувати моделі машинного навчання для визначення їхньої точності
UC13	Активація чату в Telegram	Забезпечує запуск чат-бота у Telegram для взаємодії з користувачами
UC14	Перегляд статистичних даних	Дозволяє переглядати й аналізувати інформацію про запити користувачів у чат-ботах
UC15	Перегляд зворотного зв’язку	Забезпечує перегляд відгуків, залишених користувачами у системі
UC16	Вибір теми для консультації	Дає користувачам можливість обрати тему для отримання рекомендацій з першої медичної допомоги
UC17	Ознайомлення з правилами	Дозволяє користувачам переглядати правила використання та взаємодії з чат-ботом
UC18	Надання зворотного зв’язку	Дозволяє користувачам залишати відгуки щодо роботи бота та якості отриманих рекомендацій
UC19	Запитання до бота	Дозволяє користувачам надсилати боту запитання щодо першої медичної допомоги, на які він надає відповідь
UC20	Перегляд історії подій	Забезпечує користувачам можливість переглядати інформацію про події та дії, що відбулися у системі

Виходячи з вимог до функціональності, було визначено три основні ролі: «Адміністратор», «Аналітик» та «Користувач ТГ-бота». Кожна з цих ролей має свій набір сценаріїв використання, що дозволяє оптимізувати процес взаємодії з системою та покращити виконання специфічних завдань. Для ролі «Адміністратор» визначено широкий спектр функцій, включаючи управління користувачами, темами та моделями, а також моніторинг активності користувачів і статистичних даних (рис. 2.1). Роль «Аналітик» має доступ до аналізу даних, тестування моделей та управління темами (рис. 2.2). Роль «Користувач ТГ-бота» представлена основними сценаріями, які дозволяють вибрати тему, отримати рекомендації та надати зворотний зв'язок (рис. 2.3).

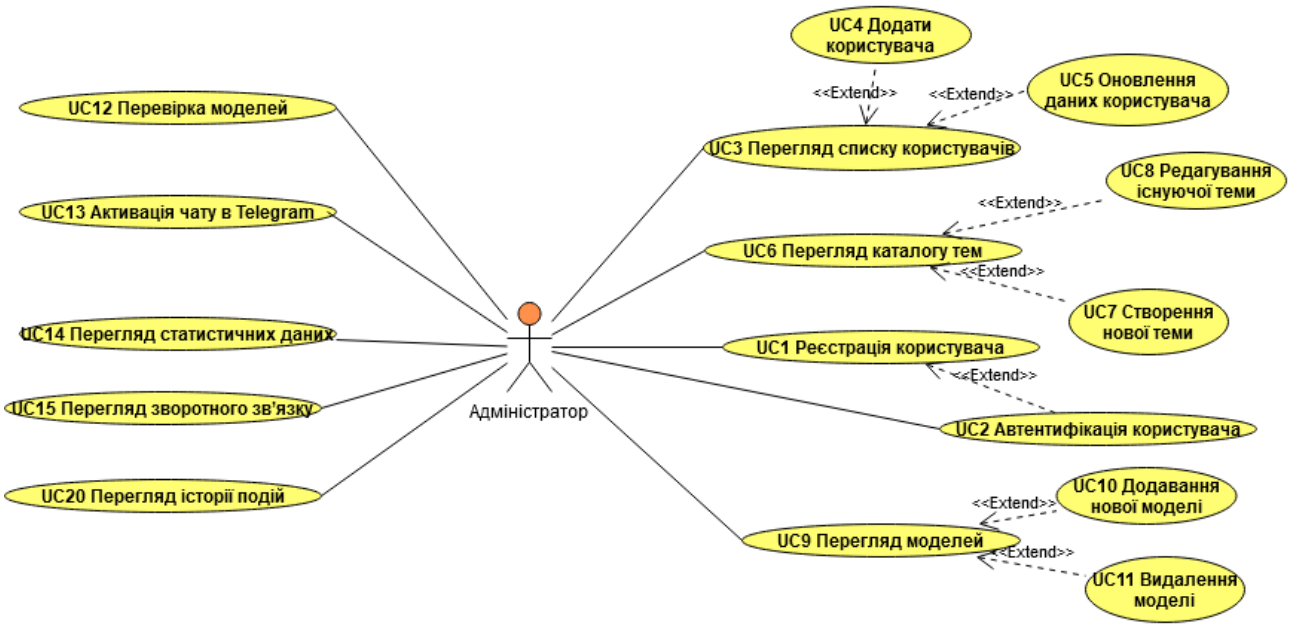

Рисунок 2.1 – Діаграма прецедентів для ролі «Адміністратор»

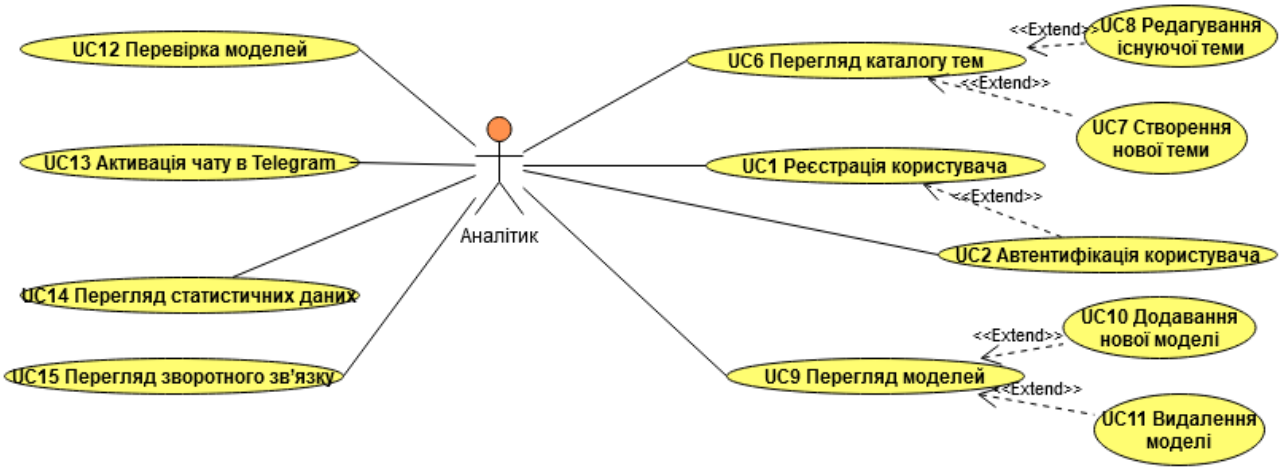

Рисунок 2.2 – Діаграма прецедентів для ролі «Аналітик»

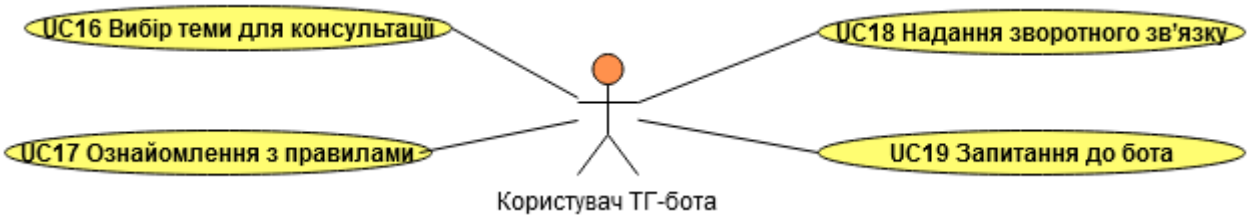

Рисунок 2.3 – Діаграма прецедентів для ролі «Користувач ТГ бота»

На наведених діаграмах варіантів використання системи показано основні взаємозв'язки між ролями та функціями системи. Вони ілюструють, як кожен прецедент, від UC1 до UC20, забезпечує різноманітні можливості для виконання

ключових операцій у системі, що включає реєстрацію, обробку даних і доступ до тематичного контенту.

2.3 Побудова діаграм основних бізнес-процесів

Для кращого розуміння бізнес-логіки системи була розроблена діаграма основних бізнес-процесів, яка відображає взаємодію користувача з Telegram-ботом на кожному етапі. Ця діаграма дозволяє простежити послідовність дій, починаючи від ініціалізації сесії до обробки запитів та збереження зворотного зв'язку [45]. Завдяки цьому стає зрозумілим, як дані проходять через усі ключові компоненти системи, включаючи базу даних і модель машинного навчання. На рис. 2.4 наведено діаграму послідовності взаємодії користувача з ботом, яка демонструє процес обробки запитів і заповнення відгуків.

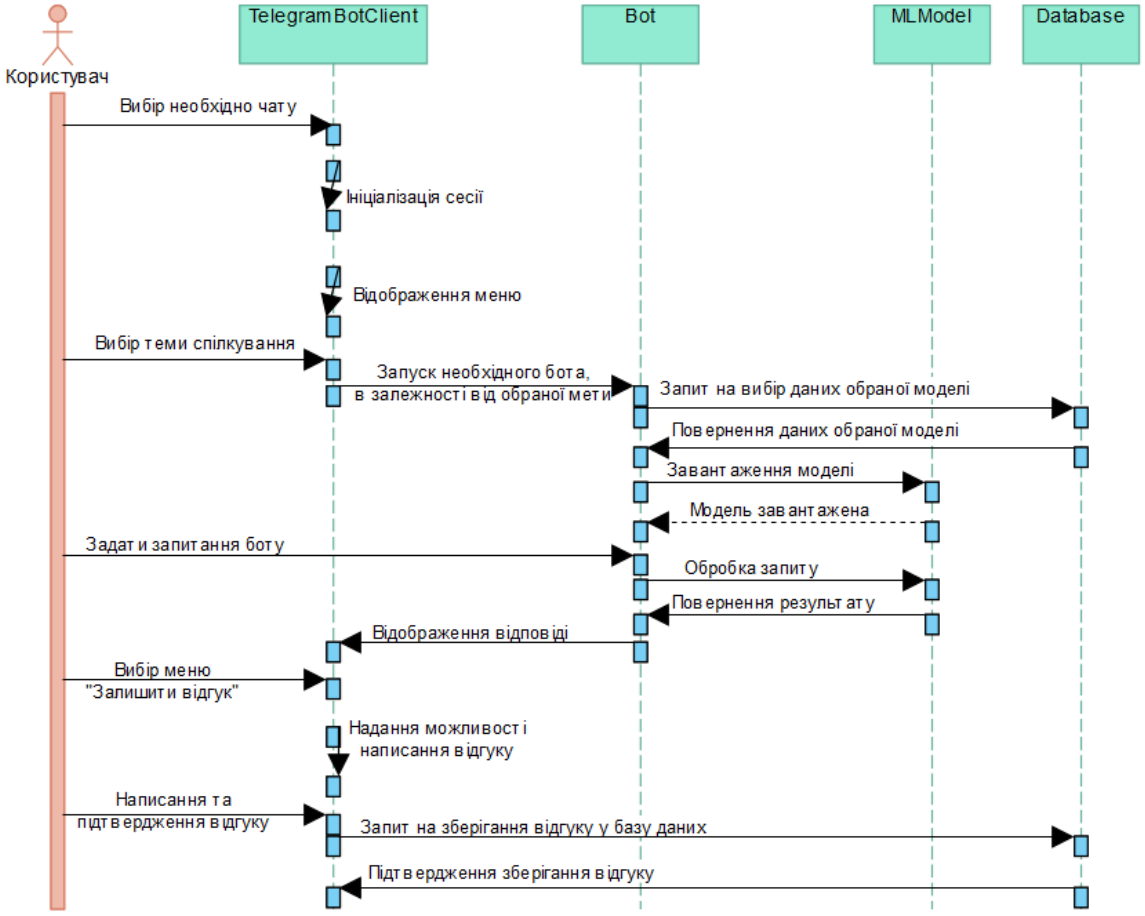

Рисунок 2.4 – Взаємодія користувача з ТГ-ботом

На діаграмі послідовності зображено взаємодію користувача з ботом через кілька послідовних етапів. Першим кроком є вибір необхідного чату, після чого відбувається ініціалізація сесії, і TelegramBotClient відображає меню для подальших дій. Користувач обирає тему для спілкування, яка визначає специфіку взаємодії з ботом. TelegramBotClient запускає потрібного бота залежно від обраної теми, і надсилається запит на вибір даних обраної моделі до блоку Bot, який передає його моделі машинного навчання (MLModel). Модель завантажується і виконує обробку запиту, після чого повертає результати до бота. Отримана відповідь відображається користувачеві.

Користувач може обрати опцію «Залишити відгук» і отримати можливість написати коментар щодо досвіду використання бота. Після підтвердження відгуку TelegramBotClient надсилає запит на збереження інформації до бази даних. База даних підтверджує успішне збереження, і ця інформація відображається користувачеві. Така послідовність дозволяє структуровано обробляти запити та відгуки користувачів, забезпечуючи злагоджену взаємодію всіх компонентів системи.

На рис. 2.5 показано діаграму послідовності процесу додавання нової теми в чат-бот розпочинається з відкриття користувачем форми CategoriesForm, яка призначена для управління темами в системі. При відкритті форми ініціюється її завантаження за допомогою виклику DataLoad, що включає підготовку інтерфейсу та необхідних ресурсів для роботи з темами. CategoriesForm відправляє запит до бази даних для отримання списку всіх збережених тем, що вже наявні у системі. Після успішного повернення даних цей список відображається на формі, дозволяючи користувачу ознайомитися з існуючими темами перед додаванням нової.

Користувач вводить необхідну інформацію про нову тему, заповнюючи відповідні поля на формі, і натискає кнопку "Додати", що ініціює перевірку коректності введених даних. Далі дані передаються до модуля Validation, який виконує перевірку на відповідність формату та можливі помилки. У разі успішного проходження валідації формується запит до бази даних для

збереження нової теми, після чого база даних підтверджує, що тема була успішно додана.

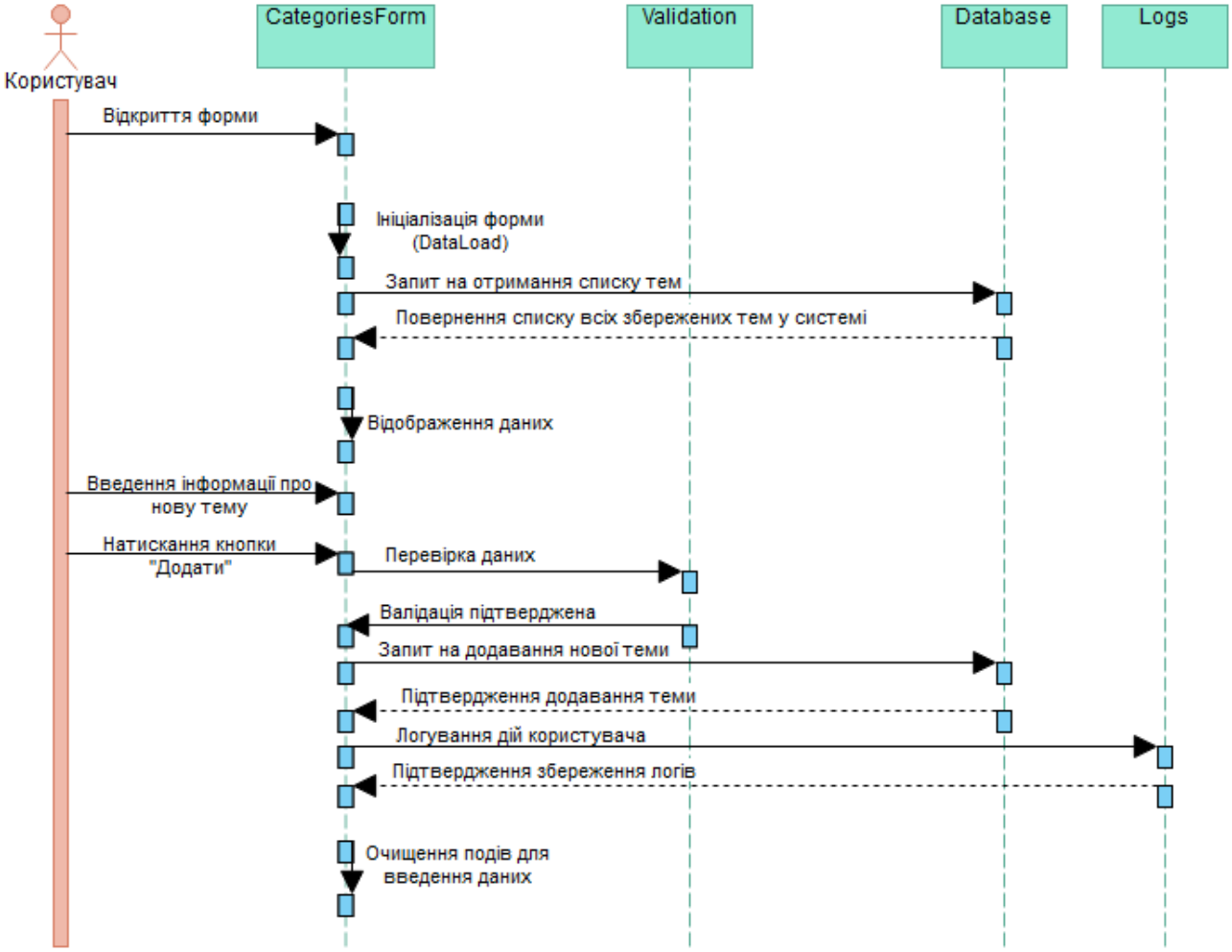

Рисунок 2.5 – Процес додавання нової теми у чат–бот

Система логування одночасно фіксує дію користувача, записуючи подію додавання нової теми у журнал для подальшого моніторингу та аудиту. Після збереження логу база даних надсилає підтвердження про успішне занесення запису. На завершення CategoriesForm автоматично очищає поля для введення, забезпечуючи готовність форми для можливого введення нової інформації. Така послідовність забезпечує чітку та надійну роботу процесу додавання нових тем у систему, зберігаючи всі дії та дані в актуальному стані.

На рис. 2.6 наведено послідовність дій для процесу тестування моделей машинного навчання перед їх використанням у чат-боті. Процес розпочинається з того, що користувач відкриває форму TestModelsForm, спеціально призначену для тестування моделей, і обирає потрібну категорію для подальшого аналізу.

TestModelsForm ініціює запит до CategoriesProvider для отримання списку доступних категорій, а потім, залежно від обраної категорії, запитує модель у ModelsProvider. Після повернення моделі вона завантажується в MLContext, де формується PredictionEngine, який дозволяє виконати передбачення на основі моделі.

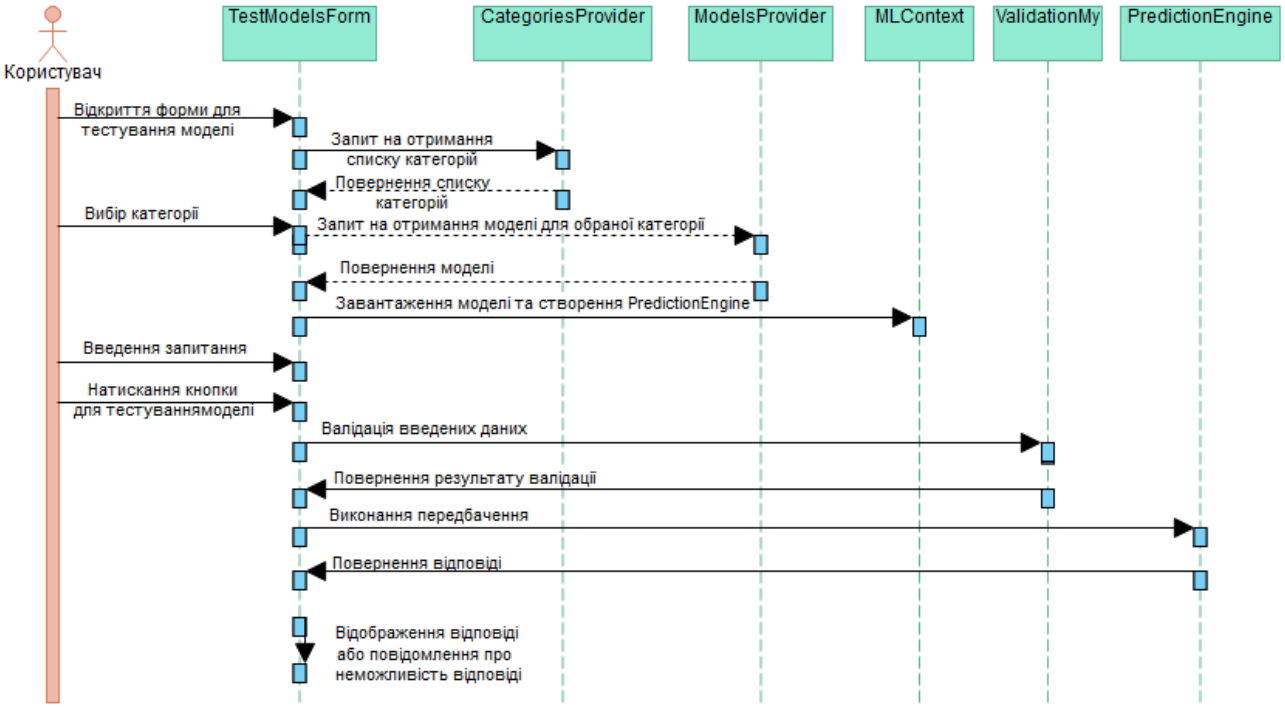

Рисунок 2.6 – Процес тестування моделей машинного навчання перед використанням у чат–боті

Користувач вводить запит для тестування, і перед його обробкою запит проходить валідацію через модуль ValidationMy. Якщо введені дані відповідають встановленим вимогам, повертається позитивний результат валідації, і PredictionEngine виконує передбачення на основі завантаженої моделі. Після виконання передбачення відповідь передається назад до TestModelsForm, де вона відображається користувачу. У разі неможливості надати відповідь користувач отримує відповідне повідомлення про помилку. Це дозволяє переконатися в коректності та готовності моделі до використання у чат-боті, забезпечуючи високу якість відповіді для кінцевого користувача.

2.4 Проектування бази даних

Проектування бази даних забезпечує належну організацію даних для ефективного зберігання, пошуку та обробки. Враховуючи специфіку даної системи, структура бази даних спроектована таким чином, щоб задовольнити вимоги щодо зберігання інформації про користувачів, події, а також управління доступом і моніторингом активності.

Таблиця «Logs» (Журнал подій) призначена для зберігання даних про різноманітні події, які відбуваються в системі, включаючи дії, здійснені користувачами (табл. 2.4). Вона забезпечує можливість відстеження активності в системі, що є важливим інструментом для аудиту та аналізу подій. Завдяки цій таблиці адміністратори можуть переглядати історію дій користувачів, оцінювати виконані операції та, у разі необхідності, вживати заходів для підвищення безпеки.

Таблиця 2.4 – Структура таблиці «Logs»

Назва поля	Тип даних	ПК	ЗК	Опис поля
LogsId	int	+	-	Ідентифікатор події, первинний ключ
UsersId	int	-	-	Ідентифікатор користувача, який виконав подію
EventNameShow	nvarchar(MAX)	-	-	Опис події або її назва
EventDate	datetime	-	-	Дата і час, коли відбулась подія

Таблиця «ChatWithBot» створена для зберігання даних про взаємодію користувачів із ботом, включаючи запитання, відповіді, а також категорії, до яких належать звернення (табл. 2.5). Ця таблиця дозволяє здійснювати аналіз діалогів між користувачами та ботом, що є корисним для вдосконалення відповіді бота на типові запити та підвищення якості обслуговування користувачів.

Таблиця 2.5 – Структура таблиці «ChatWithBot»

Назва поля	Тип даних	ПК	ЗК	Опис поля
ChatWithBotId	int	+	-	Ідентифікатор діалогу, первинний ключ
ChatId	bigint	-	-	Ідентифікатор чату
Question	nvarchar(MAX)	-	-	Запитання, яке поставив користувач
Answer	nvarchar(MAX)	-	-	Відповідь бота
CategoriesId	int	-	-	Ідентифікатор категорії, до якої належить запитання

Таблиця «Users» створена для збереження даних про користувачів системи (табл. 2.6). Вона містить основні відомості, необхідні для ідентифікації користувачів, їхньої автентифікації та управління ролями. Завдяки цій таблиці забезпечується можливість керування доступом до системи, а також надається інформація для контакту з користувачами.

Таблиця 2.6 – Структура таблиці «Users»

Назва поля	Тип даних	ПК	ЗК	Опис поля
UsersId	int	+	-	Ідентифікатор користувача, первинний ключ
FirstName	nvarchar(50)	-	-	Ім'я користувача
LastName	nvarchar(50)	-	-	Прізвище користувача
UsersName	nvarchar(50)	-	-	Логін користувача
UsersPassword	nvarchar(120)	-	-	Пароль користувача
RoleId	int	-	-	Ідентифікатор ролі, яка визначає права користувача
Description	nvarchar(1000)	-	-	Додаткова інформація про користувача
Email	nvarchar(150)	-	-	Електронна адреса користувача

Таблиця «Models» призначена для зберігання інформації про різноманітні моделі, які використовуються у системі (табл. 2.7). Вона включає відомості про назву кожної моделі, її категорію, шлях до файлу, а також додатковий опис, який допомагає деталізувати призначення та функціональність моделі. Це забезпечує

легкий доступ до збережених моделей, що є необхідним для коректної роботи системи та управління ресурсами.

Таблиця 2.7 – Структура таблиці «Models»

Назва поля	Тип даних	ПК	ЗК	Опис поля
ModelsId	int	+	-	Ідентифікатор моделі, первинний ключ
ModelsName	nvarchar(150)	-	-	Назва моделі
CategoriesId	int	-	-	Ідентифікатор категорії
ModelsFileModel	nvarchar(MAX)	-	-	Шлях до файлу з моделлю
Description	nvarchar(MAX)	-	-	Опис моделі, що надає додаткову інформацію

Таблиця «Categories» створена для управління та зберігання категорій, до яких належать різноманітні моделі та інші ресурси системи (табл. 2.8). Кожна категорія має унікальний ідентифікатор, назву та опис, що дає змогу більш точно класифікувати та організувати інформацію, пов’язану з різними аспектами системи.

Таблиця 2.8 – Структура таблиці «Categories»

Назва поля	Тип даних	ПК	ЗК	Опис поля
CategoriesId	int	+	-	Унікальний ідентифікатор категорії, первинний ключ
CategoriesName	nvarchar(250)	-	-	Назва категорії
Description	nvarchar(MAX)	-	-	Опис категорії, що деталізує її призначення

Таблиця «LeavingFeedback» призначена для зберігання відгуків, залишених користувачами, що взаємодіяли із системою. Вона містить інформацію про сам відгук, а також дані про користувача, який залишив цей відгук, і дату його подання. Це дозволяє проводити аналіз якості обслуговування та вдосконалювати роботу системи, враховуючи побажання та зауваження користувачів.

Таблиця 2.9 – Структура таблиці «LeavingFeedback»

Назва поля	Тип даних	ПК	ЗК	Опис поля
LeavingFeedbackId	int	+	-	Ідентифікатор відгуку
ChatId	bigint	-	-	Ідентифікатор чату, до якого належить відгук
Message	nvarchar(MAX)	-	-	Текст відгуку, залишеного користувачем
FirstName	nvarchar(150)	-	-	Ім'я користувача, який залишив відгук
LastName	nvarchar(150)	-	-	Прізвище користувача, який залишив відгук
LeavingFeedbackDate	datetime	-	-	Дата та час, коли був залишений відгук

Процес проектування бази даних завершився розробкою фізичної моделі, яка відображає структуру зберігання даних на рівні фізичних таблиць та зв’язків між ними. Фізична модель бази даних забезпечує конкретизацію логічної структури, включаючи всі ключові таблиці, такі як Users, Logs, ChatWithBot, Models, Categories та LeavingFeedback, що об’єднані відповідно до потреб функціональності системи. Рис. 2.7 відображає фізичну модель бази даних, яка ілюструє взаємозв’язки між таблицями та поля, що підтримують цілісність даних у системі.

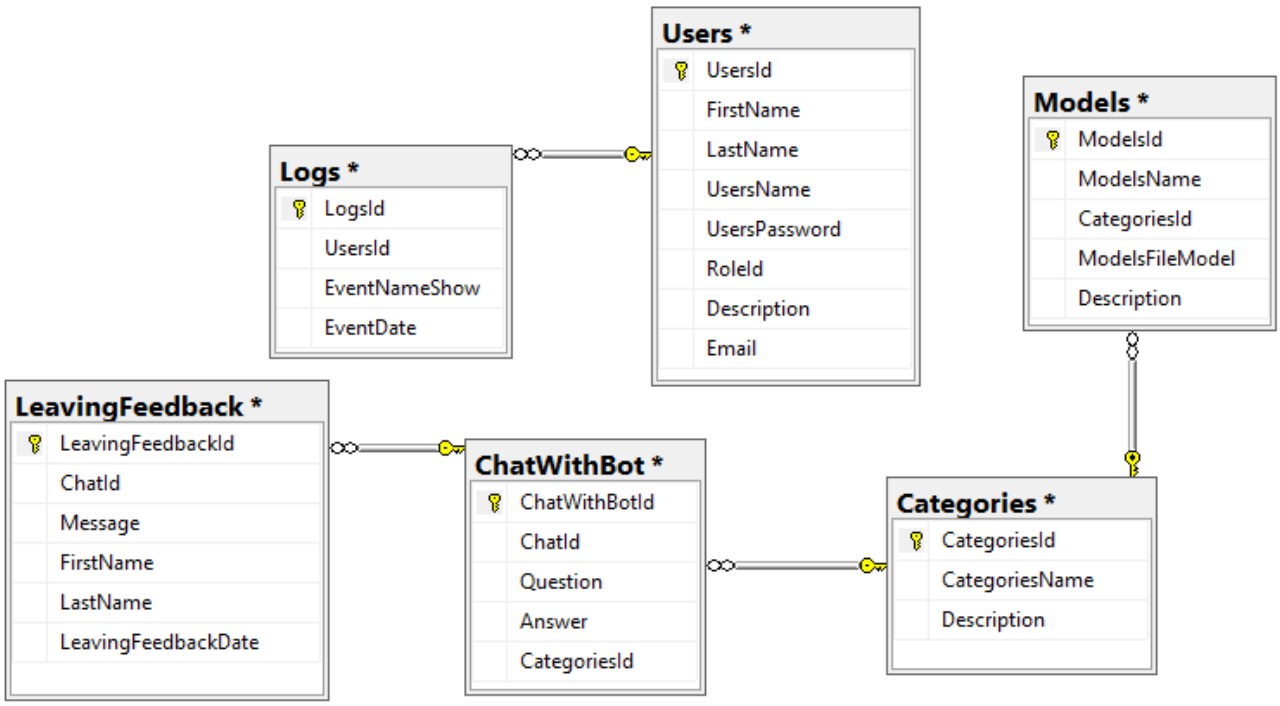

Рисунок 2.7 – Фізична модель бази даних

Фізична модель забезпечує ефективне зберігання та доступ до даних, дозволяючи оптимізувати процеси пошуку, аудиту та аналітики. Вона створює міцну основу для подальшої реалізації бізнес-логіки та підтримує масштабованість і зручність використання системи.

2.5 Архітектурне проектування системи

Архітектурне проектування системи забезпечує її стійкість, продуктивність та масштабованість. Для даної системи була обрана трьохрівнева архітектура, яка складається з рівнів презентації, логіки та даних [46]. Цей підхід дозволяє розподілити функціональність системи таким чином, щоб кожен рівень відповідав за конкретні задачі, сприяючи більш ефективному управлінню ресурсами. Рівень презентації забезпечує зручний інтерфейс для користувача, відокремлюючи його від логіки додатку та зберігання даних, що спрощує підтримку і модифікацію системи. Логічний рівень відповідає за обробку бізнес-логіки, реалізацію функціоналу та управління процесами, що підвищує гнучкість і дозволяє легше масштабувати систему. Рівень даних забезпечує зберігання та

доступ до інформації, гарантує цілісність даних та підвищує безпеку завдяки можливості централізованого контролю.

Рівень доступу до даних розробленої системи забезпечує взаємодію з базою даних, обробку запитів на зберігання, оновлення та отримання інформації, що є критичним для коректного функціонування всіх компонентів. На рис. 2.8 представлена діаграма класів цього рівня, яка демонструє основні класи, відповідальні за управління даними в системі.

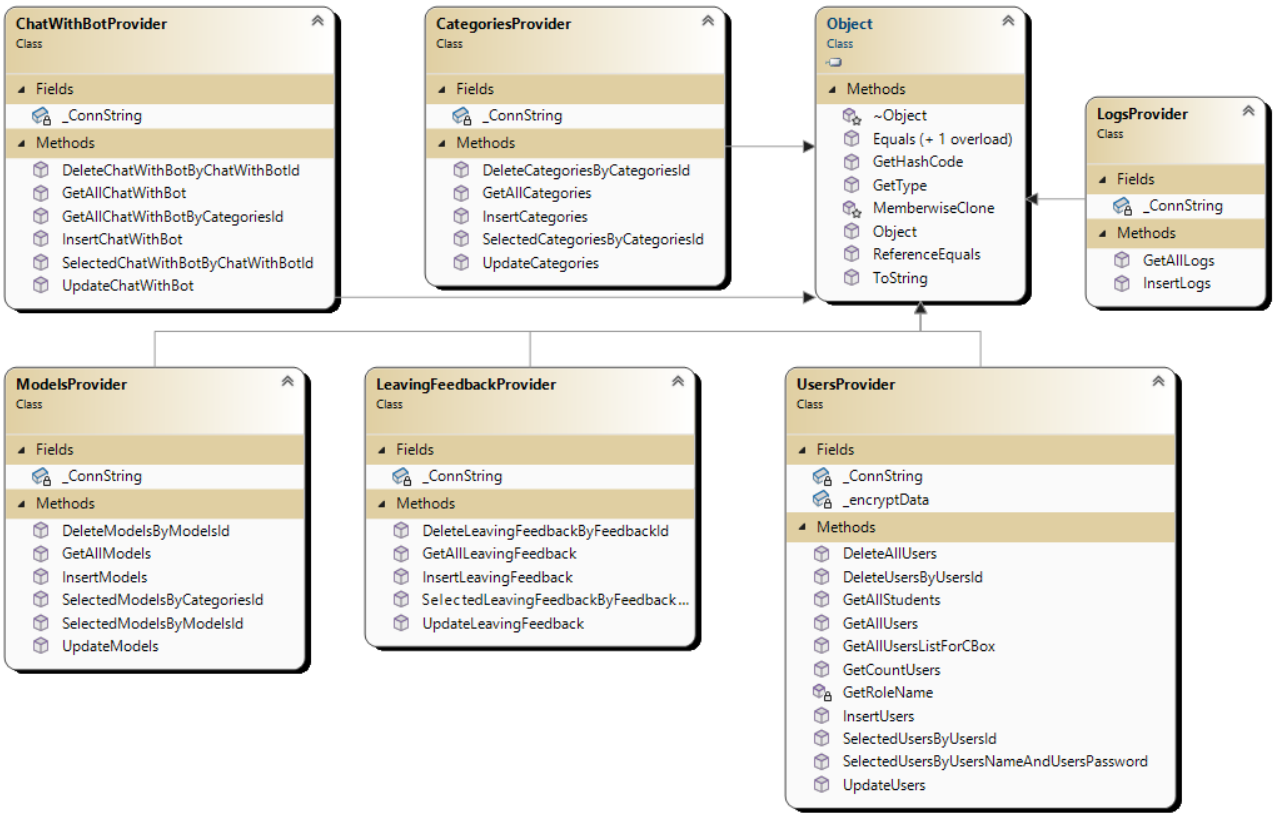

Рисунок 2.8 – Діаграма класів рівня даних

Діаграма класів рівня даних складається із:

– класу CategoriesProvider, який відповідає за доступ до даних про категорії медичних тем. Він містить методи для додавання, оновлення, видалення та пошуку інформації про різні категорії, що застосовуються у системі;

– класу ChatWithBotProvider, призначеного для обробки запитів до чат-ботів. Він дозволяє отримувати, зберігати та оновлювати дані про діалоги між

користувачами та ботом, що важливо для аналізу та вдосконалення функціоналу бота;

– класу LeavingFeedbackProvider, який відповідає за доступ до залишених користувачами відгуків. Він забезпечує можливість зберігати та аналізувати відгуки, що дозволяє покращувати якість обслуговування та підтримувати зворотний зв'язок із користувачами;

– класу LogsProvider, призначеного для роботи з даними про події в системі. Він містить методи для запису та отримання даних про події, що відбуваються, забезпечуючи можливість аудиту та моніторингу активності у системі;

– класу ModelsProvider, який забезпечує управління моделями класифікації тем, використаними в системі. Клас підтримує функціонал для додавання, оновлення та отримання моделей, що є основою класифікаційного процесу в системі;

– класу UsersProvider, який відповідає за доступ до даних про користувачів системи. Він містить методи для управління інформацією про користувачів, їхні ролі та аутентифікаційні дані, що дозволяє забезпечувати безпеку та організацію доступу до системи.

Ці класи забезпечують узгоджену та ефективну роботу з даними, що дозволяє системі швидко та надійно обробляти користувацькі запити та підтримувати актуальність даних.

Рівень інтерфейсу користувача забезпечує взаємодію між користувачем і системою, надаючи комфртний доступ до всіх функцій і дозволяючи користувачам ефективно використовувати можливості програми. Інтерфейс містить форми для різних аспектів роботи, включаючи керування ботом, перегляд даних та управління користувачами. На рис. 2.9 представлена діаграма класів цього рівня, що демонструє структуру основних форм і їхні функції.

Рисунок 2.9 – Діаграма класів рівня користувацького інтерфейсу

Така діаграма класів складається з:

– класу StartBotForm, призначеного для запуску та моніторингу роботи Телеграм-бота. Він містить елементи для ініціалізації бота та відстеження його активності у реальному часі;

– класу TestModelsForm, який дозволяє тестувати моделі штучного інтелекту для подальшого використання в Телеграм-боті. Він надає інтерфейс для перевірки точності та ефективності моделей на тестових даних;

– класу ChatWithBotAllForm, призначеного для перегляду всіх запитів до чат-бота. Цей клас забезпечує можливість аналізу та відстеження взаємодії користувачів із ботом;

– класу ChatWithBotForm, який дозволяє переглядати деталі конкретного запиту до Телеграм-бота. Це зручно для детального аналізу поведінки бота та оцінки точності відповідей;

– класу LeavingFeedbackForm, що надає можливість перегляду деталей окремого відгуку. Це дозволяє адміністраторам аналізувати зворотний зв'язок та оперативно реагувати на зауваження;

– класу UserReviewsForm, призначеного для перегляду всіх відгуків користувачів. Цей клас дозволяє переглядати зворотний зв'язок у зручному вигляді, що полегшує аналіз загальної оцінки системи;

– класу CategoriesForm, який забезпечує доступ до категорій медичних тем, що використовуються в системі. Це дозволяє управляти категоріями для більш точного налаштування бота та рекомендацій;

– класу ModelTeachForm, що дозволяє переглядати та тренувати моделі для надання медичних рекомендацій. Інтерфейс підтримує навчання нових моделей або доопрацювання існуючих, покращуючи їхню ефективність;

– класу LoginForm, який відповідає за авторизацію користувачів. Він надає інтерфейс для введення логіну та паролю, забезпечуючи безпеку доступу до системи;

– класу PersonalizationForm, що дозволяє користувачам налаштовувати персоналізовані параметри роботи з ботом. Це забезпечує індивідуальний підхід до кожного користувача;

– класу SystemLogForm, призначеного для перегляду подій у системі. Він містить функції для відстеження важливих подій, таких як зміни в базі даних та дії користувачів;

– класу UpdateUsersForm, який дозволяє редагувати або видаляти інформацію про користувачів системи. Це забезпечує зручний доступ до управління правами доступу та особистими даними;

– класу UsersForm, що забезпечує додавання та перегляд користувачів. Інтерфейс дозволяє створювати нові облікові записи та переглядати інформацію про існуючих користувачів.

Ці класи забезпечують зручний доступ до всіх функцій системи, що сприяє підвищенню ефективності роботи користувачів та надає можливість комплексного управління функціоналом системи.

Рівень бізнес-логіки в розробленій системі відповідає за реалізацію ключових процесів і обробку даних, забезпечуючи виконання функціональних завдань, які визначають основний функціонал додатку. Цей рівень включає класи, що керують логікою шифрування даних, перевіркою валідності, формуванням статистики та управлінням ролями користувачів. На рис. 2.10 представлена діаграма класів цього рівня, яка відображає основні компоненти бізнес-логіки та їх взаємодію.

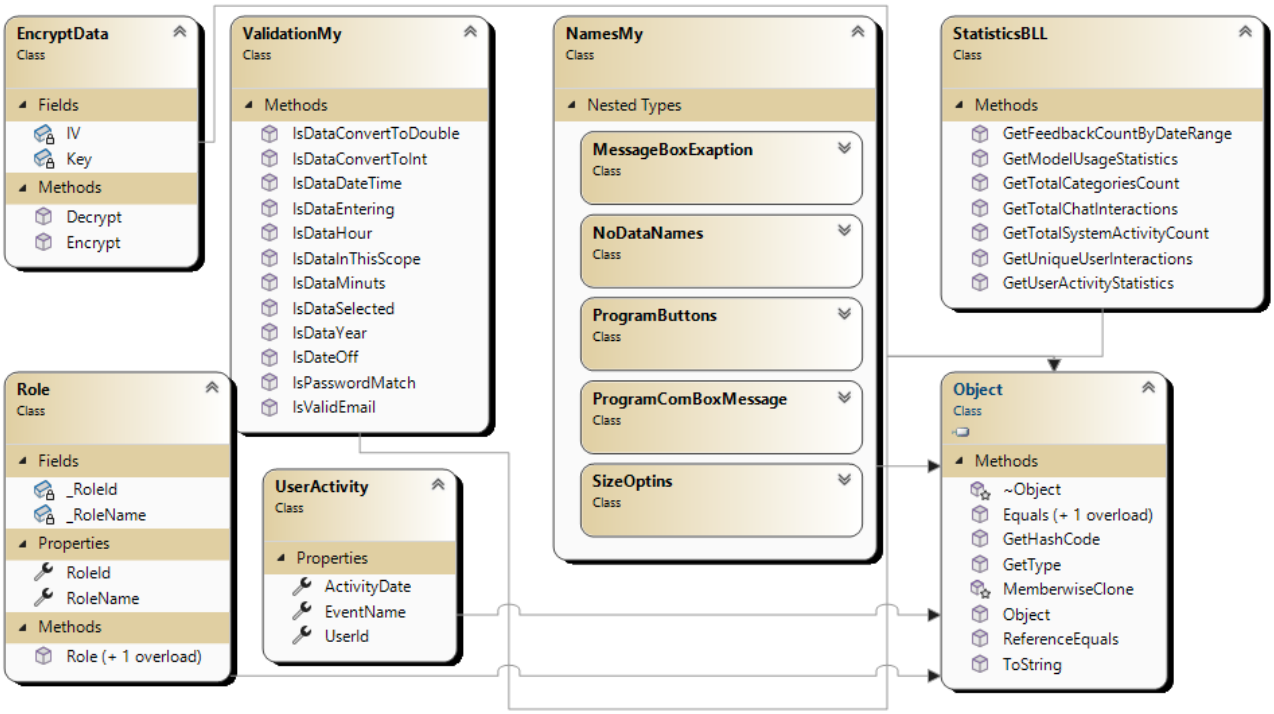

Рисунок 2.10 – Діаграма класів бізнес–логіки

Описана діаграма класів складається з:

– класу EncryptData, який забезпечує шифрування та розшифрування даних із застосуванням алгоритму AES. Він містить методи для захисту

інформації, яка є конфіденційною, що дозволяє підвищити безпеку системи та захист від несанкціонованого доступу;

– класу NamesMy, який зберігає всі назви, розміри елементів інтерфейсу та повідомлення для користувача. Це дозволяє централізовано керувати інтерфейсними елементами та повідомленнями, спрощуючи підтримку й оновлення системи;

– класу RoleApp, що визначає ролі користувачів у системі. Він забезпечує управління доступом до функцій на основі ролей, що дозволяє чітко розмежувати права та обов'язки користувачів;

– класу ValidationMy, який виконує перевірку валідності введеної інформації. Він дозволяє забезпечити коректність даних, введених користувачами, що знижує ризик помилок і підвищує надійність роботи системи;

– класу StatisticsBLL, який формує різноманітну статистику, використовуючи дані з інших рівнів системи. Він забезпечує доступ до зведених даних, корисних для аналізу ефективності та прийняття рішень;

– класу UserActivity, який використовується для відстеження та формування активності користувачів у системі. Це дозволяє збирати дані про використання функцій системи, що сприяє кращому розумінню поведінки користувачів і підвищенню загальної ефективності.

Перелічені класи забезпечують виконання всіх основних функцій бізнес-логіки, підтримуючи цілісність і безпеку даних та сприяючи стабільній роботі системи.

2.6 Висновок

У рамках даного розділу було проведено комплексне обґрунтування вибору технологій і архітектурних рішень для розробки програмного забезпечення, що відповідає вимогам стабільності, масштабованості та безпеки. На основі порівняльного аналізу мов програмування Python, Java та C# було обрано мову C# як оптимальну для реалізації функціоналу системи, оскільки вона забезпечує високу продуктивність, зручну роботу з базами даних та

підтримку розширеного функціоналу для інтеграції з інтерфейсом користувача. Серед архітектурних рішень, таких як трьохрівнева архітектура, мікросервісна архітектура та сервіс-орієнтована архітектура, було обрано трьохрівневу, оскільки вона дозволяє чітко розподілити рівні користувацького інтерфейсу, бізнес-логіки та даних, що значно підвищує ефективність системи в умовах багатозадачності та знижує витрати на підтримку й оновлення.

Були розроблені діаграми варіантів використання, які охоплюють всі можливі ролі, включаючи адміністратора, аналітика та користувача Телеграм-бота, що дозволяє чітко окреслити можливості та обмеження кожної з ролей. Це створює основу для правильного управління доступом і забезпечує коректність бізнес-процесів у системі. На основі діаграм варіантів використання були побудовані діаграми послідовності основних бізнес-процесів, включаючи взаємодію користувача з ботом, процес додавання нової теми та тестування моделей машинного навчання, що підвищує розуміння роботи системи на етапі її розробки.

Проектування бази даних охоплювало повний опис таблиць, таких як Users, Logs, ChatWithBot, Models, Categories та LeavingFeedback, що дозволило створити фізичну модель бази даних. Побудована модель забезпечує зручну організацію даних для зберігання й обробки та сприяє безпечному доступу до інформації, а також полегшує моніторинг активності користувачів і аудит подій. Також було детально розроблено архітектуру системи, побудовано діаграми класів для кожного рівня – користувацького інтерфейсу, бізнес-логіки та рівня даних, що сприяє ефективній взаємодії між компонентами системи та забезпечує модульність і легкість у підтримці та модернізації.

Отримані результати та розроблені моделі є міцною основою для подальшої реалізації програмного забезпечення в наступному розділі, який присвячено розробці та тестуванню системи. Вибір технологій, побудова діаграм та опис архітектурних рішень дозволять ефективно перейти до кодування та забезпечити високу якість і надійність програмного забезпечення.

3 РОЗРОБКА, ВПРОВАДЖЕННЯ І ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Розробка алгоритмів машинного навчання

Для розробки алгоритмів машинного навчання для інтерактивного Telegram-бота, спрямованого на автоматизовану рекомендацію першої медичної допомоги, критично важливо забезпечити високу точність класифікації для кожного сценарію. Цей етап включає вибір оптимальних моделей і методів навчання, які підвищать здатність бота надавати швидкі та точні рекомендації у відповідь на запити користувачів. На рис. 3.1 наведено блок-схему, що відображає основні етапи процесу тренування моделей.

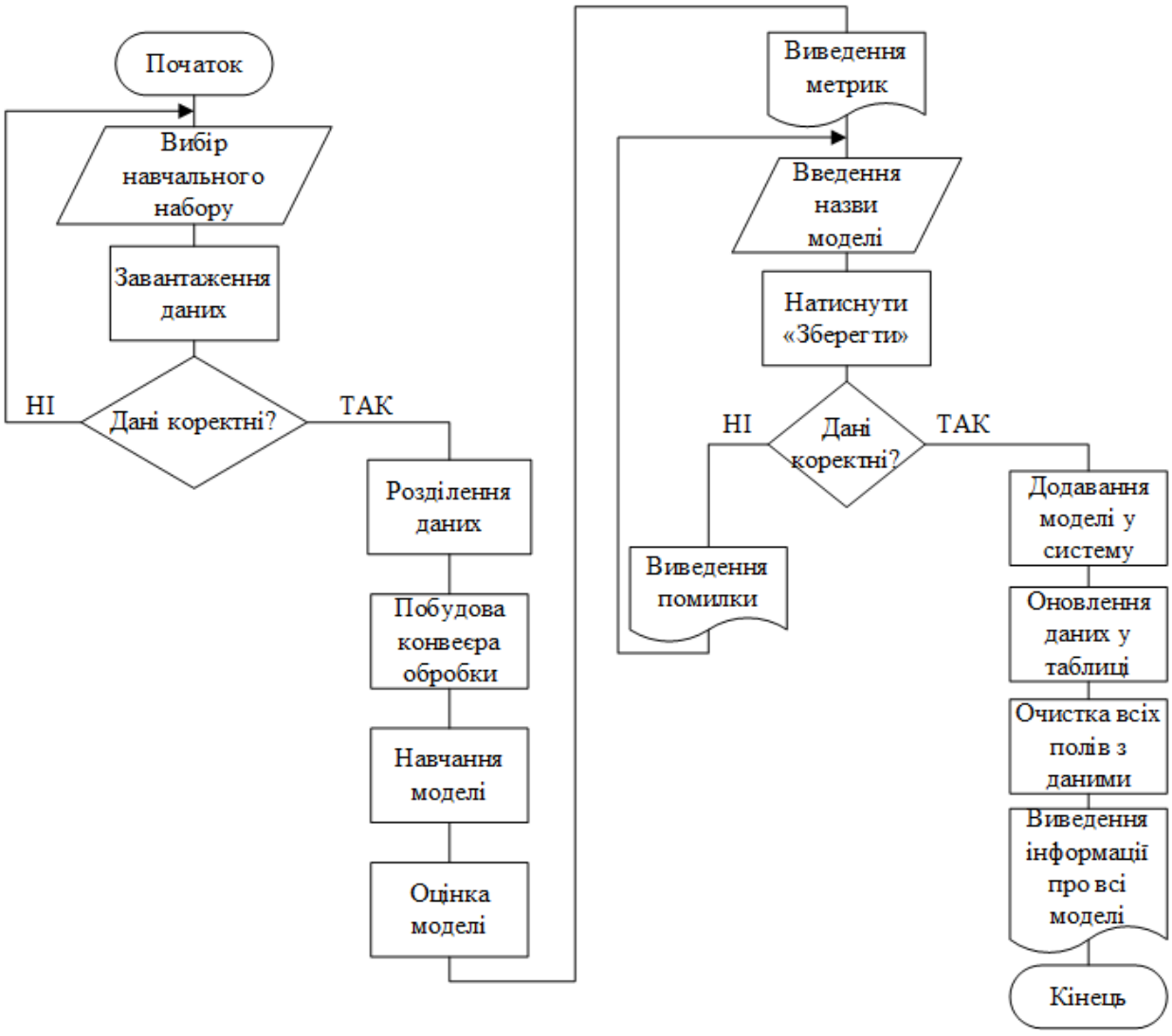

Рисунок 3.1 – Алгоритм тренування моделей автоматизованих рекомендацій

Алгоритм тренування моделей розпочинається з вибору та завантаження навчального набору даних, після чого здійснюється перевірка коректності даних. Якщо дані виявляються коректними, вони поділяються на частини, і на їх основі будується конвеєр обробки для подальшого тренування моделі, яка потім оцінюється на основі обраних метрик. У разі успішного тренування та позитивної оцінки моделі, вводиться її назва, і дані зберігаються, після чого модель додається до системи з оновленням таблиці та виведенням інформації про всі моделі. Якщо дані чи введена назва є некоректними на будь-якому етапі, система видає відповідну помилку для виправлення.

На рис. 3.2 представлено алгоритм взаємодії користувача з інтерактивним чат-ботом, де застосовується модель машинного навчання для відповіді на користувацькі запитання.

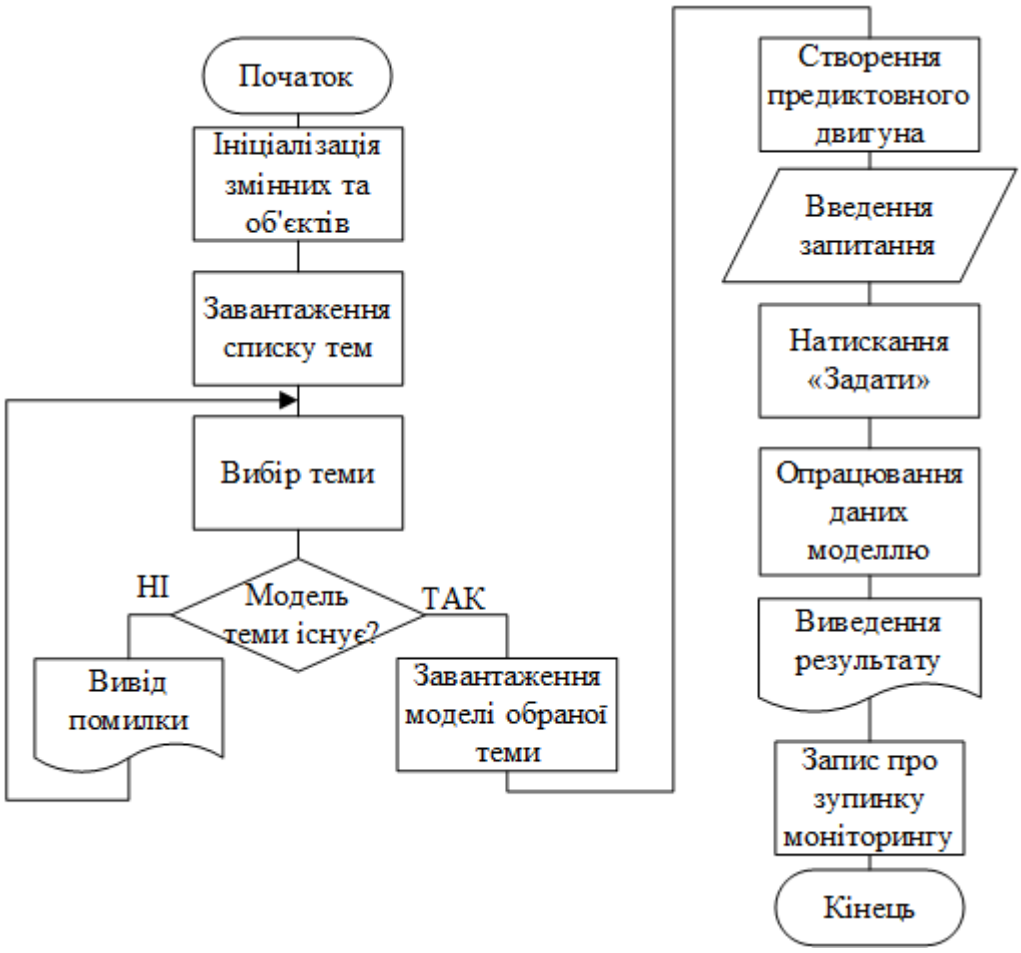

Рисунок 3.2 – Алгоритм взаємодії користувача із моделлю чат-бота

Алгоритм взаємодії користувача з моделлю чат-бота розпочинається з ініціалізації змінних та об'єктів, необхідних для роботи системи, та завантаження списку доступних тем для обробки запитів. Користувач обирає тему, і система перевіряє наявність моделі для цієї теми. Якщо відповідна модель існує, вона завантажується для подальшої роботи; у протилежному випадку, користувач отримує повідомлення про помилку. Далі створюється предиктивний двигун, до якого вводиться запит користувача, після чого він активує обробку запиту через натискання кнопки «Задати». Модель обробляє дані та виводить результат, надаючи користувачу відповідь на його запит. Після завершення взаємодії записується інформація про зупинку моніторингу, завершуючи процес.

3.2 Розробка модуля для Telegram-бот та інтеграція у систему

Процес розробки модуля для Telegram-бота передбачає створення структури, що дозволяє обробляти запити користувачів у реальному часі та надавати їм відповідні рекомендації з першої медичної допомоги. У цьому контексті ключову роль відіграє інтеграція моделі машинного навчання в бот, що забезпечує точність і швидкість обробки запитів. На рис. 3.3 зображено код, який демонструє структуру вхідних та вихідних даних, використовуваних моделлю.

```
public class Input {
    [LoadColumn(0), ColumnName("Question")]
    public string Question;
    [LoadColumn(1), ColumnName("Answer")]
    public string Answer;
}
// Клас прогнозування
4 references
public class Output {
    [ColumnName("Score")]
    public float[] Score;
    [ColumnName("PredictedLabel")]
    public uint PredictedLabel;
    [ColumnName("PredictedAnswer")]
    public string PredictedAnswer;
}
```

Рисунок 3.3 – Класи вхідних та вихідних даних моделі

Клас Input містить атрибути для завантаження даних із колонок, що позначають питання та відповідь, що дозволяє ботові розпізнавати запит користувача. Водночас клас Output призначений для збереження результатів прогнозування, включаючи оцінку відповідей, передбачений індекс відповіді та її текст. Ця структура дозволяє не лише точно передавати інформацію, а й забезпечує зворотний зв'язок для користувача у зручній формі, що значно підвищує ефективність системи інтеграції Telegram-бота.

Код на рис. 3.4 реалізує асинхронну обробку події, яка запускається при натисканні кнопки OpenBtn. Коли користувач натискає кнопку, відображається діалогове вікно для вибору файлу. Для цього створюється об'єкт OpenFileDialog, де налаштовуються фільтри файлів, щоб дозволити вибір лише текстових файлів у форматі CSV або всіх типів файлів.

```
private async void OpenBtn_Click(object sender, EventArgs e) {
  // Створення діалогового вікна для вибору файлу
  OpenFileDialog openFileDialog = new OpenFileDialog {
    // Вибір фільтру для типів файлів
    Filter = "Text files (*.csv)|*.csv|All files (*.*)|*.*",
    FilterIndex = 2,
    RestoreDirectory = true
  };
  // Показ діалогового вікна та обробка вибору
  if (openFileDialog.ShowDialog() == DialogResult.OK) {
    try {
      _Path = openFileDialog.FileName;
      FileNameTBox.Text = openFileDialog.FileName;
```

Рисунок 3.4 – Асинхронна обробка події

Вибір фільтру за замовчуванням встановлено на другий, а також активується опція відновлення останньої відкритої директорії після завершення роботи діалогового вікна. Якщо користувач обирає файл і підтверджує свій вибір, програма отримує шлях до цього файлу, зберігаючи його у змінній _Path, та відображає ім'я файлу у відповідному текстовому полі FileNameTBox.

У фрагменті коду на рис. 3.5 показано процес ініціалізації контексту машинного навчання та асинхронного запуску навчання моделі.

```
    // Ініціалізація ML-контексту
    context = new MLContext(seed: 0);
    // Асинхронний виклик методу для навчання моделі
    await TrainModelAsync();
} catch (Exception ex) {
    MessageBox.Show("Помилка: " + ex.Message);
}
```

Рисунок 3.5 – Процес ініціалізації контексту МН та асинхронного запуску НМ

Спершу створюється об'єкт MLContext із заданим початковим значенням seed, що забезпечує відтворюваність результатів навчання, зберігаючи однакову послідовність випадкових чисел. Далі здійснюється асинхронний виклик методу TrainModelAsync, який відповідає за навчання моделі, що дає змогу програмі не блокувати основний потік під час обробки. Якщо під час виконання виникає помилка, вона перехоплюється блоком catch, і користувач отримує повідомлення із детальним описом проблеми через MessageBox.

Фрагмент коду на рис. 3.6 ілюструє процес збору даних для навчання моделі, при цьому фіксується час, витрачений на цей процес. Спочатку створюється об’єкт Stopwatch, який запускається для вимірювання тривалості навчання.

```
Stopwatch stopwatch = new Stopwatch();
stopwatch.Start(); // Запуск таймера для замірів часу навчання
// Виведення повідомлення про початок навчання
RaportTBox.Text = "Розпочато навчання моделі..." + Environment.NewLine;
// Завантаження даних з файлу у колекцію
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);
  }
}
```

Рисунок 3.6 – Процес збору даних для навчання моделі

У текстове поле RaportTBox записується повідомлення про початок навчання, що дозволяє інформувати користувача про поточний етап виконання програми. Завантаження даних із файлу відбувається за допомогою StreamReader, який читає дані рядок за рядком, використовуючи відповідне

кодування. Кожен рядок розбивається на частини, створюється об’єкт типу Input з параметрами питання та відповіді, і додається до колекції datas. Це забезпечує структурування даних для наступного використання при навчанні моделі.

На рис. 3.7 висвітлено код, у якому здійснюється трансформація колекції даних у формат, придатний для роботи з ML.NET.

```
// Перетворення колекції на датасет ML.NET
data = context.Data.LoadFromEnumerable<Input>(datas);
```

Рисунок 3.7 – Трансформація колекції даних для роботи з ML.NET

Для цього використовується метод LoadFromEnumerable, який приймає на вхід колекцію datas, що складається з об'єктів типу Input. Отриманий результат зберігається у змінній data, перетворюючи дані у спеціалізований набір, який можна використовувати для подальших операцій машинного навчання в ML.NET. Це забезпечує інтеграцію з ML.NET, дозволяючи системі обробляти дані безпосередньо в необхідному для аналізу форматі.

У коді на рис. 3.8 реалізовано поділ даних на навчальний та тестовий набори, що є важливим етапом для забезпечення точності та надійності моделі.

```
// Розділення даних на навчальний і тестовий набори
var trainTestData = context.Data.TrainTestSplit(data, testFraction: 0.2, seed: 0);
var trainData = trainTestData.TrainSet;
var testData = trainTestData.TestSet;
```

Рисунок 3.8 – Поділ даних на навчальний та тестовий набори

Використовуючи метод TrainTestSplit, обирається певна частка даних (у даному випадку 20%) для тестування, а решта даних призначається для навчання. Завдяки використанню параметра seed з фіксованим значенням досягається стабільність у розподілі даних при кожному виконанні, що робить результати відтворюваними. Після виконання цього поділу, навчальні дані зберігаються у змінній trainData, а тестові – у змінній testData, що забезпечує чітке розмежування між етапами тренування та оцінки моделі.

У коді на рис. 3.9 здійснюється налаштування конвеєра для обробки та навчання моделі, що дозволяє послідовно перетворювати дані та адаптувати їх до потреб алгоритму класифікації.

```
var pipeline = context.Transforms.Text.FeaturizeText(outputColumnName:
  "Features", inputColumnName: "Question")
    .Append(context.Transforms.Conversion.MapValueToKey(outputColumnName:
    "Label", inputColumnName: "Answer"))
    .Append(context.MulticlassClassification.Trainers.SdcaMaximumEntropy())
    .Append(context.Transforms.Conversion.MapKeyToValue("PredictedAnswer", "PredictedLabel"));
```

Рисунок 3.9 – Поділ даних на навчальний та тестовий набори

Спершу текстові дані зі стовпця Question перетворюються у числові ознаки, які зберігаються у стовпці Features, забезпечуючи підготовку тексту до аналізу. Далі відбувається кодування відповідей у числовий формат за допомогою методу MapValueToKey, що дозволяє використовувати їх як мітки (Label) для навчання. Основний етап тренування реалізований через алгоритм SdcaMaximumEntropy, оптимізований для багатокласової класифікації. Після навчання модель перетворює передбачувані мітки назад у текстовий формат за допомогою MapKeyToValue, що дозволяє отримати результат у вигляді текстової відповіді (PredictedAnswer), забезпечуючи зручність для користувача.

Код на рис. 3.10 виконує асинхронне навчання моделі, забезпечуючи оптимізацію продуктивності шляхом запуску тренувального процесу у фоновому потоці. Використовуючи Task.Run, конвеєр pipeline, налаштований на попередньому етапі, застосовується до навчального набору даних trainData для створення навченої моделі.

```
// Асинхронне навчання моделі у фоновому потоці
model = await Task.Run(() => pipeline.Fit(trainData));
```

Рисунок 3.10 – Асинхронне навчання моделі

Даний підхід дозволяє уникнути блокування основного потоку, підвищуючи ефективність програми та забезпечуючи безперервну роботу

користувацького інтерфейсу під час виконання ресурсоємних операцій тренування.

У фрагменті коду на рис. 3.11 відбувається завершення процесу навчання, яке супроводжується вимірюванням і записом витраченого часу, а також виведення метрик навченої моделі.

```
stopwatch.Stop();
TimeSpan trainingTime = stopwatch.Elapsed;
// Оцінка моделі на навчальному наборі даних
var trainPredictions = model.Transform(trainData);
var trainMetrics = context.MulticlassClassification.Evaluate(trainPredictions,
  labelColumnName: "Label", scoreColumnName: "Score");
// Виведення результатів оцінки та часу навчання
RaportTBox.Text += "Результати навчання:" + Environment.NewLine;
RaportTBox.Text += $"Log-loss: {trainMetrics.LogLoss:P2}" + Environment.NewLine;
RaportTBox.Text += $"Зменшення Log-loss: {trainMetrics.LogLossReduction:P2}" + Environment.NewLine;
RaportTBox.Text += $"Макро точність: {trainMetrics.MacroAccuracy:P2}" + Environment.NewLine;
RaportTBox.Text += $"Мікро точність: {trainMetrics.MicroAccuracy:P2}" + Environment.NewLine;
RaportTBox.Text += $"Час навчання: {trainingTime.TotalSeconds} секунд" + Environment.NewLine;
```

Рисунок 3.11 – Вимірювання часу тренування та виведення метрик

Після того, як таймер stopwatch зупиняється, обчислюється тривалість навчання, яку зберігають у змінній trainingTime. Далі модель перевіряється на навчальному наборі даних – проводиться трансформація trainData, і результат передається для оцінки якості класифікації. Метод Evaluate обчислює низку ключових показників, серед яких Log-loss, його зменшення, а також макро- та мікро точність, що дозволяє детально оцінити ефективність моделі.

Наступним кроком реалізовано метод для збереження навченої моделі після підтвердження коректності введених користувачем даних (рис. 3.12).

```
private void Save_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!");
  }
}
```

Рисунок 3.12 – Зберігання моделі

Для тестування моделей розроблено спеціальну форму (рис. 3.13), яка забезпечує зручний інтерфейс для взаємодії користувача з моделлю.

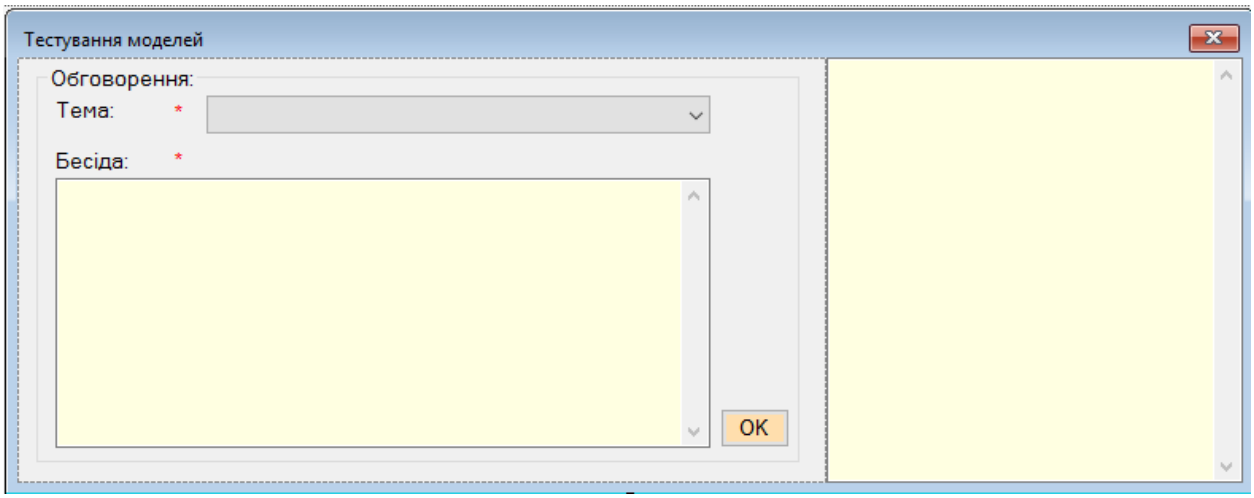

Рисунок 3.13 – Форма тестування моделей

Форма дозволяє обрати тему обговорення та ввести текст запиту в полі «Бесіда», після чого дані можуть бути відправлені на обробку за допомогою кнопки «Задати». Результати обробки відображаються у правій частині форми, що спрощує процес тестування і забезпечує наочність отриманих відповідей від моделі.

У фрагменті коду на рис. 3.14 ініціалізуються різні об'єкти, що використовуються в додатку для управління моделями та даними. CategoriesProvider та ModelsProvider забезпечують доступ до тем і моделей, які можна вибирати, що дозволяє користувачеві працювати з різними категоріями і моделями в системі.

```
// Створення екземпляра провайдера тем
private CategoriesProvider _CategoriesProvider = new CategoriesProvider();
private ValidationMy _Validation = new ValidationMy(); // Валідатор для перевірки введених даних
private List<Categories> _CategoriesList = new List<Categories>(); // Список тем
private ModelsProvider _ModelsProvider = new ModelsProvider(); // Провайдер моделей для вибору моделей
private Models _SelectedModels = new Models(); // Поточно обрана модель
private bool _IsCategoriesLoad = false; // Флаг для перевірки стану завантаження тем
private MLContext context = new MLContext(); // Контекст для машинного навчання ML.NET
private PredictionEngine<Input, Output> predictor; // Об'єкт для прогнозування з моделі
private bool _isFormLoad = false; // Флаг для перевірки стану завантаження форми
```

Рисунок 3.14 – Ініціалізація змінних та об’єктів

Об’єкт _Validation відповідає за перевірку введених користувачем даних, що допомагає зберегти коректність інформації на етапі взаємодії з інтерфейсом. MLContext використовується як середовище для виконання операцій машинного навчання, а об'єкт predictor реалізує механізм прогнозування, застосовуючи модель для нових даних. Два логічні прапорці _IsCategoriesLoad та _isFormLoad допомагають відстежувати стан завантаження тем і форми, що оптимізує керування процесами в додатку.

Фрагмент коду на рис. 3.15 реалізує завантаження даних для випадаючого списку тем медицини, що забезпечує користувачеві можливість вибору теми. Спершу виконується запит до провайдера _CategoriesProvider, щоб отримати всі доступні теми, які зберігаються в списку _CategoriesList. Потім цей список встановлюється як джерело даних для елемента CategoriesCBox, дозволяючи відобразити тему у вигляді вибору. Вибираються конкретні поля для значень і відображення у списку, де CategoriesId використовується як значення, а CategoriesName як відображуване ім'я. Після цього змінна _IsCategoriesLoad встановлюється в true, сигналізуючи про успішне завантаження тем, і програмно викликається подія CategoriesCBox_SelectedValueChanged, щоб оновити інші елементи інтерфейсу на основі обраної теми.

```
private void LoadAllData() {
    // Отримує всі категорії
    _CategoriesList = _CategoriesProvider.GetAllCategories();
    // Встановлює джерело даних для випадаючого списку
    CategoriesCBox.DataSource = _CategoriesList;
    // Вказує поле для значення в списку
    CategoriesCBox.ValueMember = "CategoriesId";
    // Вказує поле для відображення в списку
    CategoriesCBox.DisplayMember = "CategoriesName";
    // Встановлює флаг завантаження тем
    _IsCategoriesLoad = true;
    // Викликає обробник зміни значення списку
    CategoriesCBox_SelectedValueChanged(CategoriesCBox, EventArgs.Empty);
}
```

Рисунок 3.15 – Завантаження даних для списку тем медицини

Фрагмент коду на рис. 3.16 обробляє подію, яка виникає при зміні вибраного значення у випадаючому списку теми CategoriesCBox. Якщо

прапорець _IsCategoriesLoad вказує на те, що теми вже завантажені, виконується перевірка, чи вибране значення є валідним (більше за нуль).

```
private void CategoriesCBox_SelectedValueChanged(object sender, EventArgs e) {
  if (_IsCategoriesLoad) {
    if (Convert.ToInt32(CategoriesCBox.SelectedValue) > 0) {
      try {
        // Отримує модель на основі ID обраної теми
        _SelectedModels = _ModelsProvider.SelectedModelsByCategoriesId(
          Convert.ToInt32(CategoriesCBox.SelectedValue));
        if (_SelectedModels.ModelsId != 0) {
          LoadData(_SelectedModels.ModelsFileModel); // Завантаження моделі
        } else {
          // Повідомлення користувачу, якщо для обраної теми модель не навчено
          if (_isFormLoad) {
            MessageBox.Show("По вибраній теми ще не навчено чат-бота!");
          } else {
            _isFormLoad = true;
          }
        }
      } catch (Exception ex) {
        MessageBox.Show(ex.Message); // Виведення повідомлення про помилку
      }
    }
  }
}
```

Рисунок 3.16 – Завантаження даних для списку тем медицини

За допомогою методу SelectedModelsByCategoriesId, відбувається запит до провайдера _ModelsProvider для отримання моделі, пов’язаної з вибраною темою. Якщо для обраної теми знайдено модель із коректним ідентифікатором, запускається метод LoadData для завантаження цієї моделі. Якщо ж модель для вибраної теми ще не навчена, і форма вже завантажена, користувачеві відображається повідомлення з відповідним попередженням. У випадку, якщо під час виконання коду виникає помилка, програма перехоплює її та виводить відповідне повідомлення, що забезпечує користувача зворотнім зв’язком про можливі проблеми.

За завантаження даних для моделі машинного навчання на основі вказаного шляху до файлу реалізовано код, який показано на рис. 3.17.

```
private void LoadData(string FilePath) {
  string localProj = Application.StartupPath + FilePath; // Формує повний шлях до файлу моделі
  DataViewSchema modelSchema; // Схема даних для завантаженої моделі
  ITransformer model = context.Model.Load(localProj, out modelSchema); // Завантаження моделі ML.NET
  predictor = context.Model.CreatePredictionEngine<Input, Output>(model); // Створення двигуна прогнозування
}
```

Рисунок 3.17 – Завантаження даних для моделі машинного навчання

На початку формується повний шлях до моделі шляхом об’єднання шляху запуску додатка з отриманим параметром FilePath. Далі визначається змінна modelSchema, яка буде містити схему даних завантаженої моделі. Метод Load завантажує модель з використанням ML.NET, при цьому одночасно отримується схема, що дозволяє коректно працювати з даними. Після цього створюється об’єкт predictor, який є двигуном прогнозування, здатним приймати нові дані у форматі класу Input та повертати результати у вигляді класу Output. Такий підхід забезпечує легку інтеграцію моделі в додаток для подальшого використання у прогнозуванні.

Метод на рис. 3.18 обробляє подію, яка відбувається при натисканні на кнопку QuestionBtn. Спершу перевіряється, чи коректно введені дані за допомогою методу IsQuestionCorrect. Якщо дані відповідають вимогам, створюється об'єкт input класу Input, де зберігається текст запитання з текстового поля QuestionTBox. Далі програма перевіряє, чи ініціалізований об'єкт predictor, що вказує на готовність моделі до прогнозування. Якщо модель доступна, здійснюється прогнозування на основі введеного запитання. Запитання відображається в полі MaterialsTBox для документування.

```
private void QuestionBtn_Click(object sender, EventArgs e) {
    if (IsQuestionCorrect()) { // Перевірка коректності введених даних
        // Формування запиту для моделі
        var input = new Input { Question = QuestionTBox.Text };
        if (predictor != null) {
            // Виконання прогнозування
            var prediction = predictor.Predict(input);
            // Виведення запитання
            MaterialsTBox.Text += "Запитання: " + QuestionTBox.Text + "\r\n";
            // Перевірка відповіді та її виведення
            if (prediction.PredictedAnswer.ToString() == "Answer") {
                MaterialsTBox.Text += "Вибачте, я не можу відповісти на поставлене запитання" + "\r\n";
            } else {
                MaterialsTBox.Text += "Відповідь: " + prediction.PredictedAnswer + "\r\n";
            }
        } else {
            // Повідомлення про відсутність моделі
            MessageBox.Show("Поки не створено моделі для обраної теми", "Увага!");
        }
    }
}
```

Рисунок 3.18 – Метод обробника події при натисканні на кнопку QuestionBtn

Якщо модель повертає відповідь "Answer," що вказує на відсутність конкретної відповіді, користувач отримує повідомлення про неможливість відповіді. Інакше виводиться прогнозована відповідь. У випадку відсутності моделі для обраної теми користувач отримує попередження про необхідність обрання або завантаження відповідної моделі.

3.3 Розробка окремих програмних компонентів

Розробка окремих програмних компонентів передбачає створення спеціалізованих модулів для інтеграції та ефективної роботи системи. Одним із ключових елементів є ініціалізація форми запуску бота, де забезпечується перевірка наявності навченої моделі, необхідної для коректної роботи програми. На рис. 3.19 представлено структуру коду конструктора форми StartBotForm, яка відповідає за початкову налаштування інтерфейсу та завантаження моделі.

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

Рисунок 3.18 – Конструктор форми StartBotForm

Під час ініціалізації форми викликається метод InitializeComponent, що налаштовує всі графічні елементи. Потім через _CategoriesProvider отримується список всіх тем для подальшого вибору. Якщо перший елемент списку містить коректний ідентифікатор (відмінний від нуля), цей елемент призначається як поточно обрана тема _SelectedCategories. Далі на основі цієї теми визначається відповідна модель за допомогою методу SelectedModelsByCategoriesId, яка завантажується через виклик LoadData. У випадку відсутності навченої моделі користувач отримує повідомлення про необхідність проведення навчання, після

чого форма автоматично закривається, запобігаючи запуску без доступної моделі.

Фрагмент коду на рис. 3.19 відповідає за ініціалізацію Telegram-бота під час завантаження форми StartBotForm.

```
private void StartBotForm_Load(object sender, EventArgs e) {
  _BotClient = new TelegramBotClient(_Token);
  _BotClient.OnMessage += BotClient_OnMessage;
  _BotClient.StartReceiving();
}
```

Рисунок 3.19 – Ініціалізація Telegram-бота

У межах події Load створюється об’єкт _BotClient на основі класу TelegramBotClient, який використовує заданий токен _Token для автентифікації. Після цього налаштовується обробник подій OnMessage, який відповідає за отримання та обробку повідомлень від користувачів, асоціюючи його з методом BotClient_OnMessage. Завершальним кроком є запуск процесу прослуховування з боку бота, що активується через StartReceiving, дозволяючи програмі обробляти вхідні повідомлення в режимі реального часу.

Фрагмент коду на рис. 3.20 обробляє вхідні повідомлення від користувачів у Telegram-боті через асинхронний метод BotClient_OnMessage. Спершу отримується текст повідомлення, після чого воно додається до текстового поля RaportTBox за допомогою делегата BeginInvoke, що дозволяє оновлювати інтерфейс у багатопоточному режимі. Це забезпечує відображення тексту повідомлення з автоматичним прокручуванням до останнього запису.

```
private async void BotClient_OnMessage(object sender, MessageEventArgs e) {
  var msg = e.Message;
  RaportTBox.BeginInvoke((MethodInvoker)(delegate {
    RaportTBox.AppendText(msg.Text + "\r\n");
    RaportTBox.ScrollToCaret();
  }));
  if (!isMenuFirslLoad) {
    await ShowMainMenu(msg.Chat.Id);
    isMenuFirslLoad = true;
  }
  // Перевірка, чи користувач зараз залишає відгук
  if (_IschatStart) {
    SaveFeedback(msg.Chat.Id, msg.Text, msg.From.FirstName, msg.From.LastName);
    await _BotClient.SendTextMessageAsync(msg.Chat.Id, "Дякуємо за відгук!");
    _IschatStart = false;
    return;
  }
```

Рисунок 3.20 – Ініціалізація Telegram-бота

Якщо головне меню ще не було завантажено (isMenuFirslLoad дорівнює false), бот викликає асинхронний метод ShowMainMenu, передаючи ідентифікатор чату користувача, після чого прапорець isMenuFirslLoad змінюється на true для уникнення повторного завантаження меню. Далі виконується перевірка, чи користувач перебуває у режимі залишення відгуку (_IschatStart дорівнює true). Якщо так, то метод SaveFeedback зберігає відгук з деталями про користувача, включно з ім'ям і прізвищем. Після цього бот надсилає користувачеві повідомлення з подякою за відгук, змінює прапорець _IschatStart на false, що завершує процес збору відгуку, і виходить з методу, щоб уникнути подальшої обробки цього повідомлення.

Код на рис. 3.21 реалізує логіку обробки тексту повідомлень від користувача, використовуючи конструкцію switch. Залежно від тексту вхідного повідомлення (msg.Text), бот виконує певні дії. Якщо користувач надсилає "меню," бот асинхронно викликає метод ShowMainMenu, показуючи головне меню. При виборі команди "Обрати тему допомоги" бот відображає список доступних категорій через метод ShowCategorys. У разі, коли обрано "Правила спілкування," запускається асинхронний метод SendCommunicationRulesAsync, який надсилає правила взаємодії.

```
switch (msg.Text) {
  case "меню":
    await ShowMainMenu(msg.Chat.Id);
    break;
  case "Обрати тему допомоги":
    ShowCategorys(msg.Chat.Id);
    break;
  case "Правила спілкування":
    await SendCommunicationRulesAsync(msg.Chat.Id);
    break;
  case "Залишити відгук":
    await _BotClient.SendTextMessageAsync(msg.Chat.Id, "Будь-ласка, залиште відгук");
    _IschatStart = true;
    break;
  default:
    if (_IsCategoriesLoad) {
      Answer(msg.Chat.Id, msg.Text);
    } else {
      LoadModelBot(msg.Chat.Id, msg.Text);
    }
    break;
}
```

Рисунок 3.21 – Ініціалізація Telegram-бота

Якщо користувач обирає "Залишити відгук," бот просить залишити відгук, надсилаючи відповідне повідомлення, а прапорець _IschatStart встановлюється на true, що вказує на початок процесу збору відгуку. Для всіх інших випадків за умовчанням бот перевіряє, чи були завантажені категорії (_IsCategoriesLoad). Якщо так, викликається метод Answer, який формує відповідь для користувача, інакше метод LoadModelBot використовується для завантаження моделі на основі тексту запиту, що дає змогу боту адаптуватися до різних запитів користувача.

Код на рис. 3.22 створює головне меню для Telegram-бота у вигляді кнопок, використовуючи об'єкт ReplyKeyboardMarkup. Спершу формується клавіатура з двома рядами кнопок: перший ряд містить кнопки "Обрати тему допомоги" та "Залишити відгук," тоді як у другому ряду розміщено кнопку "Правила спілкування." Параметр ResizeKeyboard налаштований так, щоб клавіатура автоматично адаптувалася до розмірів кнопок, а OneTimeKeyboard встановлено на false, що дозволяє зберегти клавіатуру активною після натискання. У результаті метод повертає об'єкт keyboard, який виступає як основне меню для взаємодії користувача з ботом, спрощуючи навігацію по функціях.

```
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;
}
```

Рисунок 3.22 – Створення головне меню Telegram-бота

Метод LoadModelBot, який завантажує відповідну модель для обраної теми на основі введеного користувачем номера в повідомленні (рис. 3.23). Спочатку перевіряється, чи введений текст можна перетворити на ціле число, використовуючи метод _Validation.IsDataConvertToInt. Якщо введене значення коректне, воно конвертується в ціле число та шукається в списку категорій _CategoriesList. Якщо знайдено відповідну категорію, вона призначається як обрана, а також здійснюється вибір моделі, пов’язаної з цією категорією, через метод SelectedModelsByCategoriesId.

```
private async void LoadModelBot(long ChatId, string msgText) {
  // Перевіряємо, чи текст є валідним номером
  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.CategoriesId));
      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!");
  }
}
```

Рисунок 3.23 – Код методу «LoadModelBot»

Після цього здійснюється завантаження даних моделі за допомогою LoadData, а також встановлюється прапорець _IsCategoriesLoad для позначення, що модель завантажена. Користувачу надсилається повідомлення про обрану тему. Якщо ж обраний номер не знайдено серед доступних категорій,

користувачу відправляється повідомлення про те, що такої теми не існує. У випадку, коли введене значення не є коректним числом, бот інформує користувача про помилку, вказуючи, що очікувалося числове значення.

Метод Ansver, який відповідає за формування відповіді бота на запитання, отримане від користувача (рис. 3.24). Спочатку створюється змінна message для зберігання тексту відповіді. Далі формуються вхідні дані для моделі: створюється об’єкт input із запитанням, введеним користувачем, яке надсилається на прогнозування за допомогою методу predictor.Predict.

```
private void Ansver(long ChatId, string msgText) {
  string message = "";
  var input = new Input { Question = msgText };
  var prediction = predictor.Predict(input);
  if (prediction.PredictedAnswer.ToString() == "Answer") {
    message = "Вибачте, я не можу відповісти на поставлене запитання\r\n";
  } else {
    message = prediction.PredictedAnswer + "\r\n";
  }
  // Додано відправлення повідомлення користувачу
  _BotClient.SendTextMessageAsync(ChatId, message);
  _ChatWithBotProvider.InsertChatWithBot(ChatId,
    input.Question, message, _SelectedCategories.CategoriesId);
}
```

Рисунок 3.24 – Код методу «Ansver»

Якщо модель повертає результат "Answer," що вказує на неможливість надати конкретну відповідь, у змінну message записується повідомлення з вибаченням. Якщо ж модель дає конкретну відповідь, результат прогнозування зберігається в message. Після цього бот надсилає сформоване повідомлення користувачеві за допомогою SendTextMessageAsync. Додатково запис про взаємодію з користувачем зберігається у базі даних за допомогою методу _ChatWithBotProvider.InsertChatWithBot, включаючи ID чату, запит, відповідь, а також ідентифікатор обраної категорії.

3.4 Проведення тестування системи

Під час розробки системи важливим етапом є тестування функціоналу для забезпечення коректної роботи всіх компонентів. Одним з ключових елементів у системі є форма авторизації, яка забезпечує доступ користувачів до внутрішніх ресурсів. Для забезпечення належної безпеки та зручності користування, було проведено тестування функціоналу авторизації з метою перевірки відповідності роботи системи очікуваним результатам.

У табл. 3.1 представлені тестові сценарії для перевірки форми авторизації. Кожен тестовий сценарій містить конкретні кроки, що необхідно виконати, а також очікуваний та фактичний результати.

Таблиця 3.1 – Тестові сценарії форми авторизації

№	Назва сценарію	Кроки	Очікуваний результат	Отриманий результат
1	Введення правильних даних	1. Ввести правильний логін 2. Ввести правильний пароль 3. Натиснути "Підтвердити"	Користувач успішно авторизується, відкривається головна сторінка	ок
2	Введення неправильного пароля	1. Ввести правильний логін 2. Ввести неправильний пароль 3. Натиснути "Підтвердити"	Відображається повідомлення про невірний пароль	Ок
3	Введення порожніх полів	1. Залишити поле логіну порожнім 2. Залишити поле пароля порожнім 3. Натиснути "Підтвердити"	Відображається повідомлення про обов'язковість заповнення полів	Ок
4	Використання незареєстрованого логіну	1. Ввести незареєстрований логін 2. Ввести будь-який пароль 3. Натиснути "Підтвердити"	Відображається повідомлення про відсутність користувача	Ок

Ці сценарії дозволили оцінити стійкість форми авторизації до типових помилок користувача та підтвердити, що система реагує належним чином, забезпечуючи необхідний рівень безпеки та інформативності.

Для перевірки функціональності форми тренування моделей були розроблені різні тестові сценарії, що забезпечують відповідність роботи системи очікуваним результатам при різних умовах використання. У табл. 3.2 представлено п'ять ключових тестових сценаріїв для даної форми.

Таблиця 3.2 – Тестові сценарії форми тренування моделей

№	Назва сценарію	Кроки	Очікуваний результат	Отриманий результат
1	Завантаження файлу для навчання	1. Натиснути "Відкрити" 2. Обрати файл у файловій системі 3. Натиснути "Відкрити" для завантаження файлу	Файл успішно завантажено, шлях до файлу відображається у відповідному полі	Ок
2	Заповнення всіх обов'язкових полів	1. Завантажити файл 2. Обрати тему 3. Ввести назву моделі 4. Ввести опис 5. Натиснути "Зберегти"	Дані успішно збережені, модель додається до списку моделей з коректними даними	Ок
3	Залишення обов'язкових полів порожніми	1. Натиснути "Зберегти" без заповнення обов’язкових полів	Відображається повідомлення про необхідність заповнення обов'язкових полів	Ок
4	Перегляд результатів навчання	1. Завантажити файл для навчання 2. Запустити навчання моделі 3. Переглянути метрики навчання в полі "Результати"	В полі "Результати" відображаються метрики навчання, включаючи Log-loss, Макро та Мікро точність, час навчання	Ок
5	Видалення моделі з таблиці	1. Обрати модель зі списку 2. Натиснути кнопку "Видалити" поруч із моделлю	Модель видаляється зі списку моделей, таблиця оновлюється без видаленої моделі	Ок

Дані сценарії дозволили перевірити коректність основних функцій форми тренування моделей, забезпечуючи правильність виведення повідомлень та обробку дій користувача при роботі з формою.

Для перевірки коректності роботи програмного забезпечення було використано MSTest – інструмент для тестування, який дозволяє автоматизовано перевірити функціональні можливості системи, забезпечуючи надійність та стабільність її роботи. Проведення MSTest тестування дозволило перевірити широкий набір методів і функцій, що використовуються в системі, зокрема функціонал, пов'язаний з обробкою даних чат-бота для медичної допомоги.

На рис. 3.25 показано результат проведення тестування за допомогою MSTest, де всі тестові сценарії пройшли успішно, а загальний час виконання тестів склав лише 7 мс. Це свідчить про високу швидкодію та ефективність реалізованих функцій у системі.

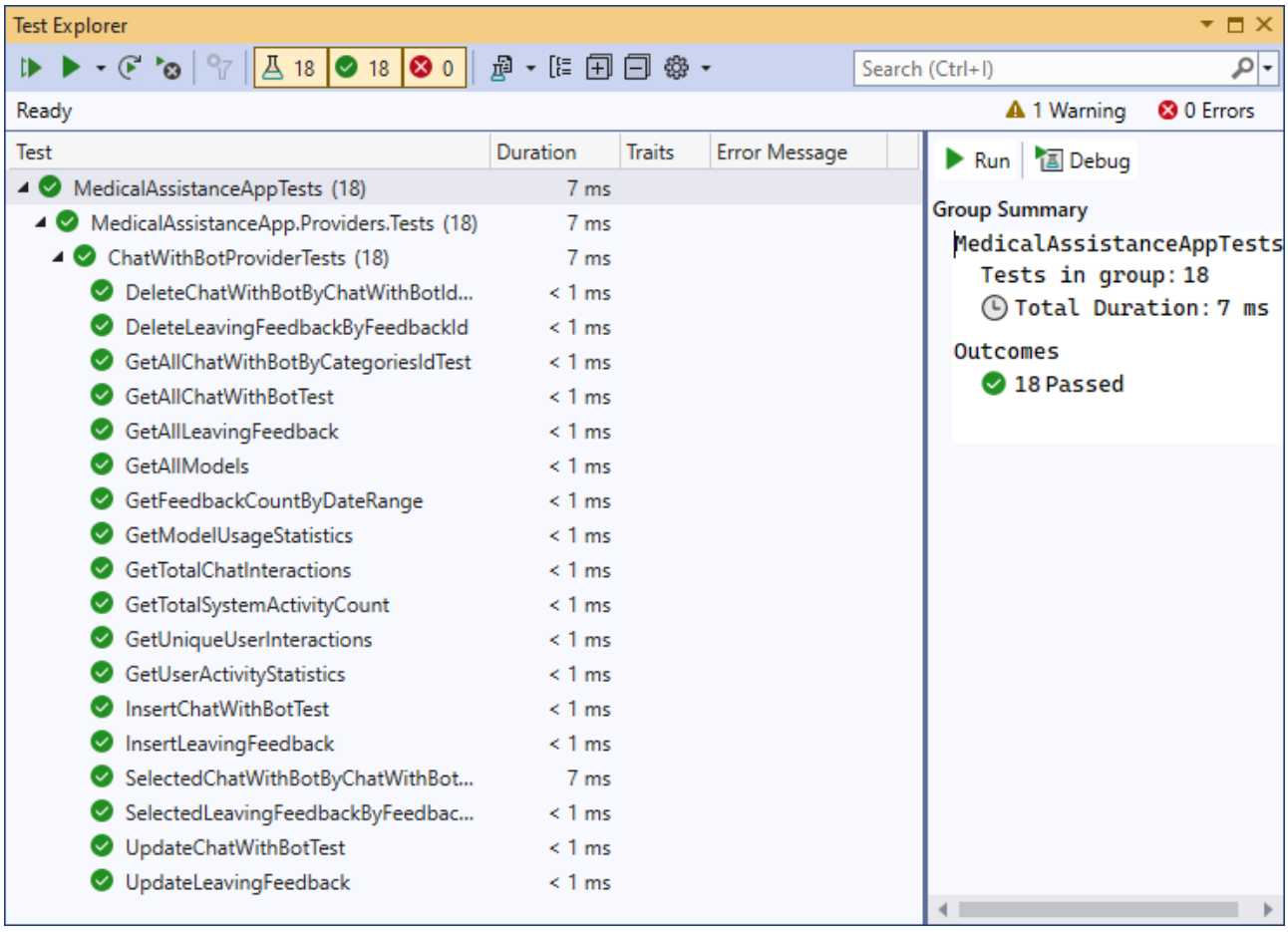

Рисунок 3.25 – Результат проведення MSTest тестування

Загалом, результати тестування як функціональних сценаріїв взаємодії користувача, так і автоматизованого MSTest показали стабільну роботу системи.

Усі тестові сценарії пройдено успішно, що підтверджує готовність програмного забезпечення до подальшої експлуатації.

3.5 Оцінка роботи системи в експериментальних умовах

Для створення датасету, що використовується в Telegram-боті для надання першої медичної допомоги, було проведено комплексний процес збору, обробки та впорядкування даних. Першочерговим етапом стало визначення ключових тематичних напрямів, на які орієнтуватиметься система. Враховуючи основну мету – надання рекомендацій у критичних ситуаціях, було обрано дві основні тематичні групи: «Медичні стани та невропатологія» і «Травми та поранення».

Збір даних розпочався з аналізу відкритих джерел, таких як посібники з надання першої допомоги, що містять стандартизовані інструкції, розроблені на основі сучасних медичних протоколів [47]. Також була використана інформація з перевірених онлайн-ресурсів, зокрема матеріали сайту «Рецептіка», що спеціалізується на рекомендаціях для домедичної допомоги [48]. У процесі збору важливим завданням було забезпечити відповідність зібраної інформації вимогам практичності та релевантності. Дані вибиралися таким чином, щоб вони охоплювали найтиповіші сценарії, з якими можуть зіткнутися користувачі.

Зібрані дані спочатку були представлені у вигляді текстових описів, що включали повні інструкції для дій у різних ситуаціях. Після цього тексти були сегментовані на окремі запитання та відповіді. Наприклад, загальний розділ про укуси змій був розділений на конкретні запитання, такі як «Що робити при укусі змії?» або «Як зупинити отруєння від укусу?», із відповідями, що надають чіткі рекомендації.

Процес формування датасету передбачав структурування даних у форматі «запитання – відповідь». Для цього всі записи були приведені до єдиного формату, що забезпечує їх легку інтеграцію в модель машинного навчання та використання в системі Telegram-бота. Кожен запис складався з текстового запитання, яке може бути поставлене користувачем, і відповіді, що містить рекомендації з надання першої допомоги. З метою забезпечення високої точності

системи були використані лише ті дані, які пройшли ретельну перевірку та мають актуальність для реальних сценаріїв.

Під час обробки текстів було враховано вимоги до стиснення та адаптації інформації для швидкого розуміння користувачами. Замість громіздких описів, які містяться в оригінальних джерелах, відповіді були оптимізовані, щоб передавати ключову інформацію в стислому та зрозумілому вигляді. Наприклад, рекомендації щодо дій при травмі грудної клітки містили лише конкретні кроки: накласти холод, забезпечити спокій, дати анальгетик і звернутися до лікаря у разі ускладнень.

Після сегментації дані були впорядковані у вигляді таблиці, яка включає два основні стовпці: «Question» (запитання) і «Answer» (відповідь). Кожен запис у таблиці був перевірений на предмет наявності дублювання чи можливих неточностей. На фінальному етапі було створено два окремі датасети, що відповідають тематичним напрямам. Перший охоплює запитання, пов’язані з медичними станами, такими як шок, непритомність або алергічні реакції. Другий містить інформацію щодо травм і поранень, включно з порізами, забоями, переломами та укусами тварин.

Сформовані датасети були збережені у форматі CSV для подальшого використання у процесі тренування моделей машинного навчання. Загалом, у тематичній групі «Медичні стани та невропатологія» було підготовлено 84 унікальні записи, тоді як для групи «Травми та поранення» – 76 записів.

На рис. 3.26 наведено скріншот фрагменту інтерфейсу з підготовленими даними для навчання, що включають інформацію для тематики «Медичні стани та невропатологія».

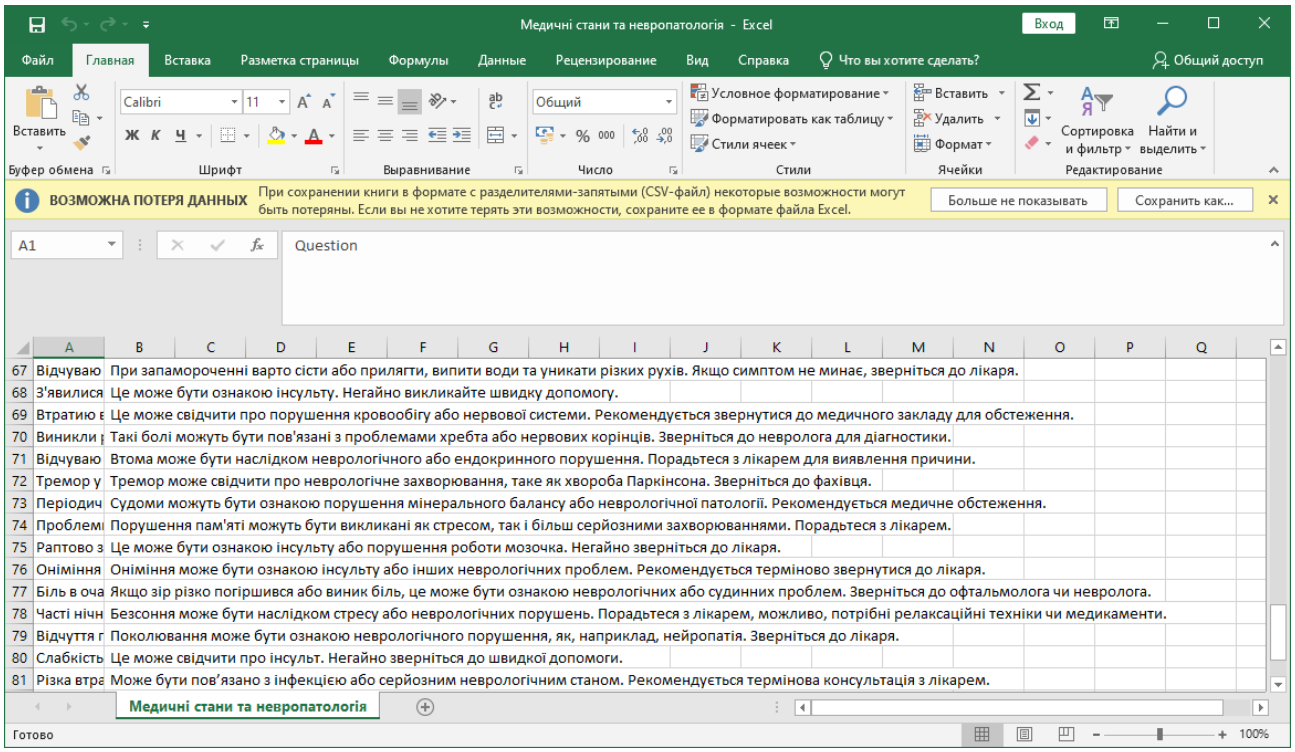

Рисунок 3.26 – Фрагмент інтерфейсу з даними для навчання

На основі зібраних даних було розпочато процес навчання моделі, що дозволило системі опрацювати специфічну інформацію, пов’язану з медичними станами та невропатологією. Після завантаження файлу з навчальним набором даних "Медичні стани та невропатологія.csv" було виконано навчання з обчисленням ключових метрик, що дозволяють оцінити якість отриманої моделі.

На рис. 3.27 зображено процес навчання моделі, де видно прогрес виконання операцій та обчислені метрики точності та втрат.

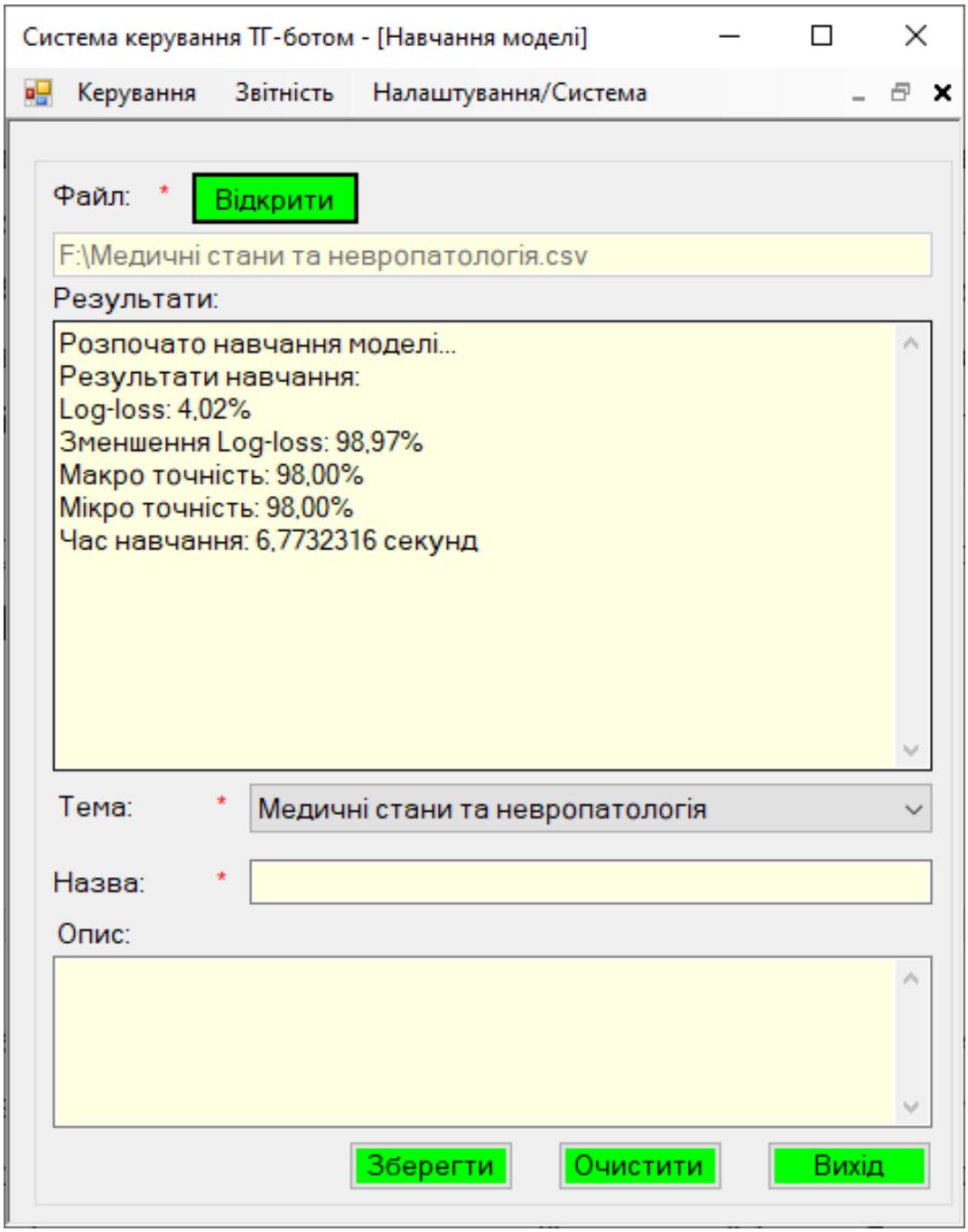

Рисунок 3.27 – Результати навчання моделі

В ході навчання система виміряла метрики цієї моделі, зокрема:

– Log-loss становить 4.02%, що є низьким показником і свідчить про невелику похибку в передбаченні відповідей моделі. Це означає, що модель здатна досить точно ідентифікувати відповіді, з мінімальними втратами інформації при обробці симптомів. Зменшення Log-loss до 98.97% вказує на значне покращення моделі під час навчання, що є хорошим результатом для систем, які орієнтовані на обробку медичних даних;

– макро точність на рівні 98.0% свідчить про високу точність для всіх категорій відповідей, враховуючи загальне середнє значення без урахування частоти класів. Це особливо важливо для системи, де всі симптоми та відповіді мають однаково важливе значення;

– мікро точність також складає 98.0%, що показує успішне передбачення по всіх симптомах і випадках, враховуючи частоту кожного з них. Це метрика є значущою для перевірки загальної продуктивності моделі;

– час навчання склав 6.7732316 секунд, що свідчить про швидкість та ефективність алгоритму при роботі з відносно невеликим набором даних.

Ці метрики вказують на високу якість навченої моделі, яка демонструє здатність забезпечувати точні та надійні результати при діагностиці симптомів, пов’язаних з медичними станами та невропатологією.

Після навчання моделі було проведено експериментальні тести для оцінки її здатності правильно відповідати на запитання, пов’язані з медичними станами та невропатологією. Система дозволяла користувачеві вводити запити, а модель генерувала відповідні відповіді на основі навчальних даних, що дало змогу оцінити адекватність та точність її реакції на різні симптоми.

На рис. 3.28 зображено приклади експериментальних тестів, де користувач вводить запитання, наприклад, "Виникли різкі болі в шиї та спині", а модель відповідно генерує рекомендацію для таких випадків. Це підтверджує здатність системи надавати корисні відповіді в контексті заданих медичних тем.

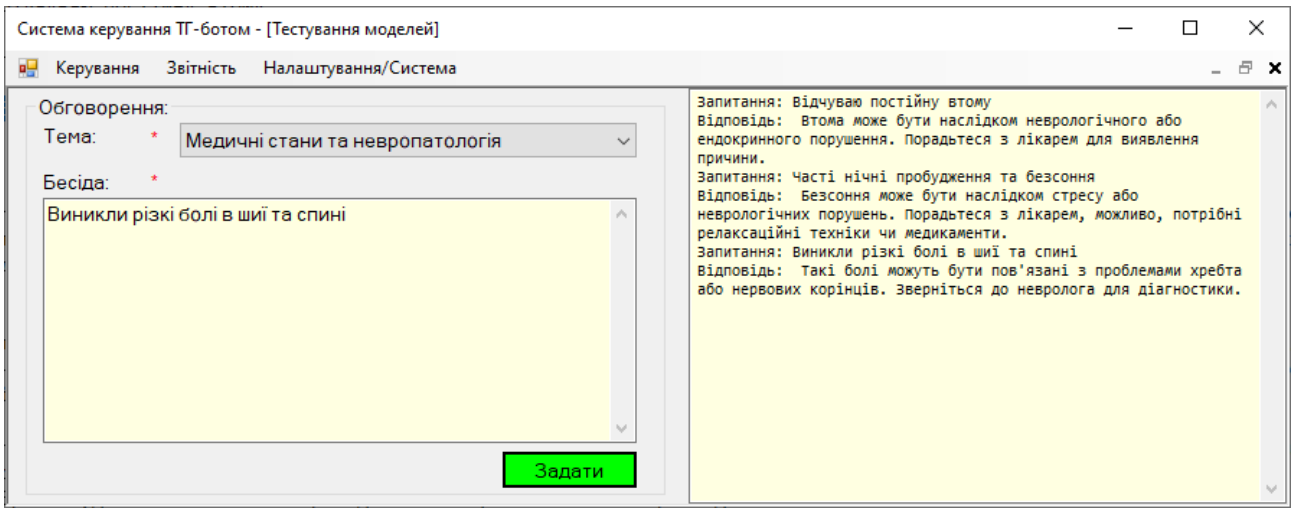

Рисунок 3.28 – Експериментальні тести з запитаннями

Також для перевірки роботи чат-бота було проведено додаткове тестування, спрямоване на аналіз його здатності коректно обробляти запити з граматичними помилками, пропусками букв і неповними формулюваннями. У процесі тестування навмисно вводилися запити, що містили орфографічні помилки, такі як відсутність окремих букв у словах або їх перестановка, а також граматичні недоліки, наприклад неправильне узгодження слів у реченні (рис. 3.29). Також перевірялося, чи здатний бот інтерпретувати запити, що містять надмірні пропуски або скорочення, які характерні для реального спілкування.

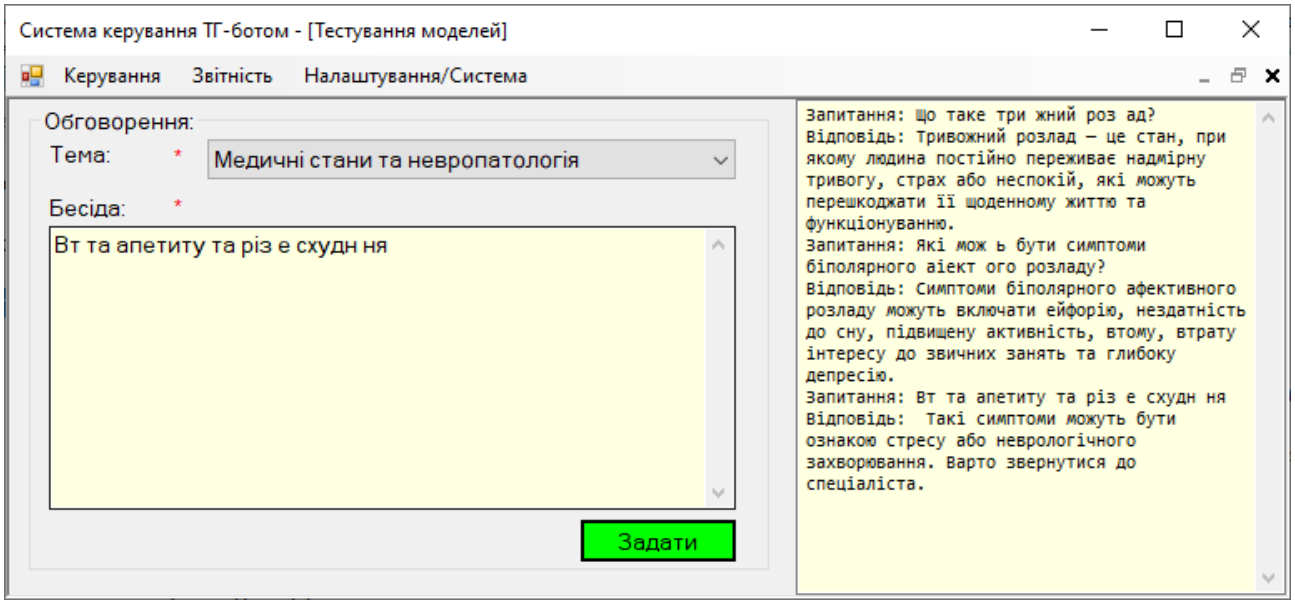

Рисунок 3.29 – Експериментальні тести із помилками введення

Виявлено, що бот демонстрував високу стійкість до таких похибок у запитах. Наприклад, у ситуаціях, коли користувач вводив неповну фразу або слова із пропусками букв, система правильно розпізнавала контекст запиту та пропонувала коректні відповіді. На скріншоті видно, що система змогла успішно відреагувати на запит "Вт та апетиту та різ є сухн ня", який містив явні помилки в написанні. Бот інтерпретував це як питання, що стосується втрати апетиту та сухості у роті, і надав точну відповідь, яка пояснювала можливі причини такого стану та рекомендації щодо звернення до фахівця.

На рис. 3.30 наведено приклад діалогу з ботом, де були задані питання про склероз множинної енцефалопатії, симптоми розладу уваги та гіперактивності

(РУГ), а також симптоми біполярного афективного розладу. Чат-бот надав точні й структуровані відповіді, демонструючи здатність забезпечувати користувачів необхідною інформацією.

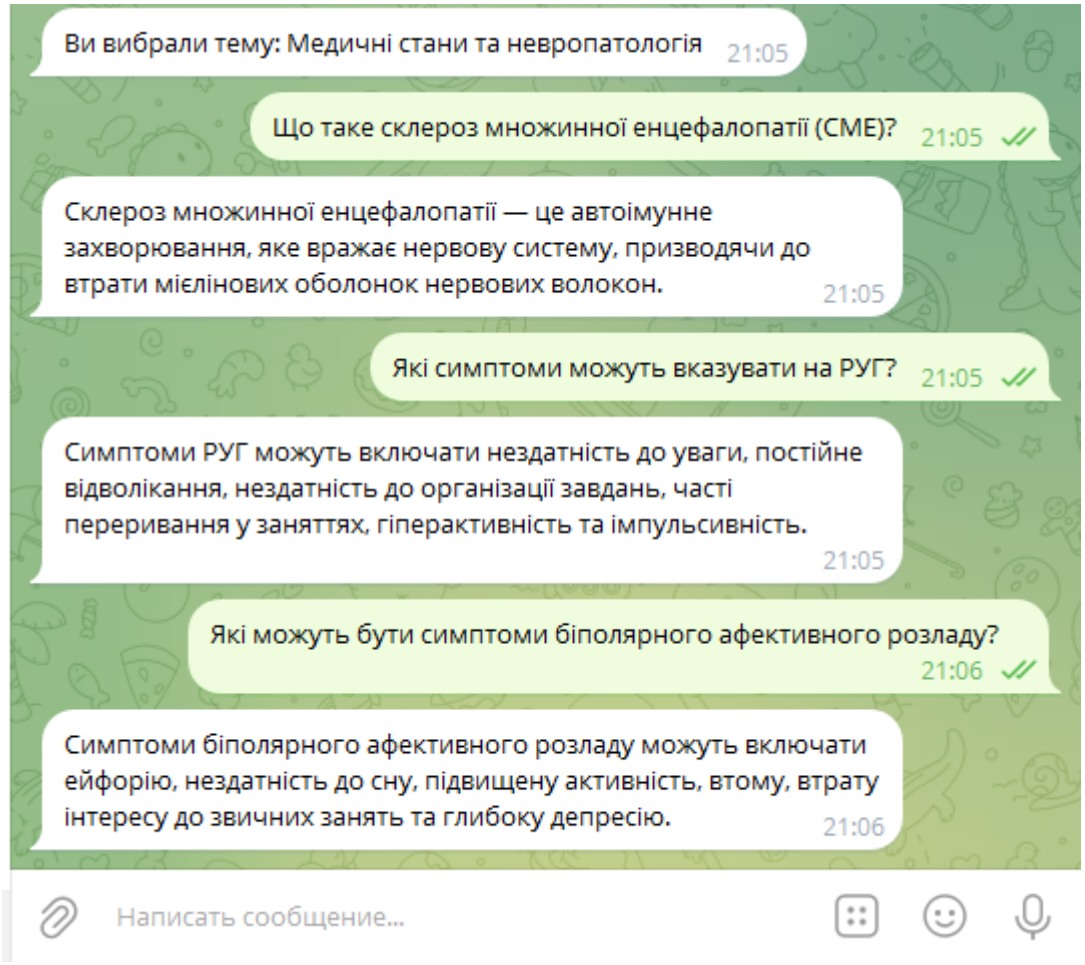

Рисунок 3.30 – Результати тестування чат–боту

Результати тестування підтвердили ефективність роботи чат-боту, його здатність коректно інтерпретувати запити та надавати відповідну інформацію. Це свідчить про високу якість підготовлених навчальних даних і правильну реалізацію алгоритмів обробки інформації.

3.6 Розробка інструкцій для користувачів, що надають і отримують медичну допомогу

Для забезпечення зручності та ефективності використання системи чат-боту, були розроблені детальні інструкції для користувачів, які надають та отримують медичну допомогу. Ці інструкції допомагають користувачам ознайомитися з процесом взаємодії з системою, від авторизації до виконання основних функцій, таких як надання рекомендацій, пошук медичної інформації або спілкування з фахівцем.

На рис. 3.31 представлено скріншот форми авторизації користувача в системі, що є першим кроком для входу та доступу до функцій системи. У формі користувач має можливість обрати свій логін зі списку та ввести пароль.

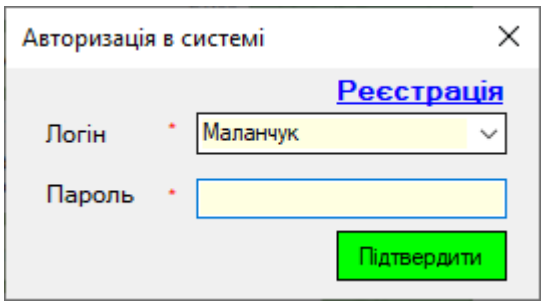

Рисунок 3.31 – Форма авторизації користувача

Функція авторизації забезпечує захист особистих даних користувача і гарантує, що доступ до інформації матимуть лише авторизовані особи. Це є важливим етапом для підтримки безпеки і конфіденційності даних у системі, яка працює з чутливою медичною інформацією.

Форма керування медичними темами є важливим інструментом для організації та додавання нових категорій медичних знань у систему. Як показано на рис. 3.32, інтерфейс цієї форми дозволяє адміністратору вводити назву нової теми в полі "Назва" та надавати їй детальний опис у відповідному текстовому полі. Після заповнення цих полів адміністратор може натиснути кнопку "Додати", щоб зберегти нову тему в системі.

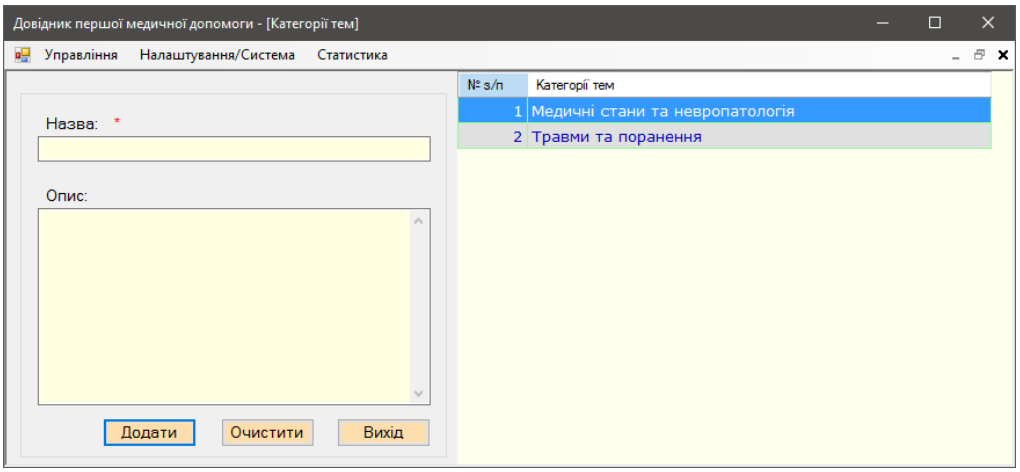

Рисунок 3.32 – Форма для керування темами

На правій стороні форми розташований список існуючих тем, де кожна категорія має свій номер і назву. Це дозволяє зручно переглядати вже наявні теми, наприклад, "Медичні стани та невропатологія" та "Травми та поранення". Форма також включає кнопки "Очистити" для скидання введених даних і "Вихід" для завершення роботи з формою. Така структура форми забезпечує зручне керування темами та полегшує організацію знань у системі.

Форма редагування тем призначена для внесення змін до існуючих категорій медичних знань у системі. На рис. 3.33 показано, що користувач може оновити як назву теми в полі "Назва", так і її детальний опис у відповідному текстовому полі "Опис". Це дозволяє адміністраторам системи уточнювати інформацію про тему, додаючи нові аспекти або корегуючи наявні дані.

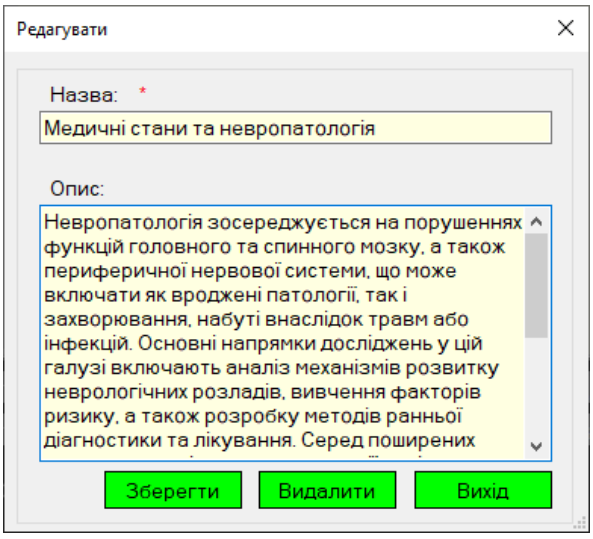

Рисунок 3.33 – Форма для редагування теми

Форма містить три функціональні кнопки: "Зберегти" для підтвердження внесених змін, "Видалити" для повного видалення теми з системи, і "Вихід" для завершення роботи з формою редагування.

Форма для тренування моделі надає користувачеві можливість завантажувати файли з навчальними даними, запускати процес навчання моделей, переглядати результати та керувати вже створеними моделями (рис. 3.34). Як видно на скріншоті, користувач може завантажити файл, натиснувши кнопку "Відкрити", після чого обраний файл відображається у відповідному полі.

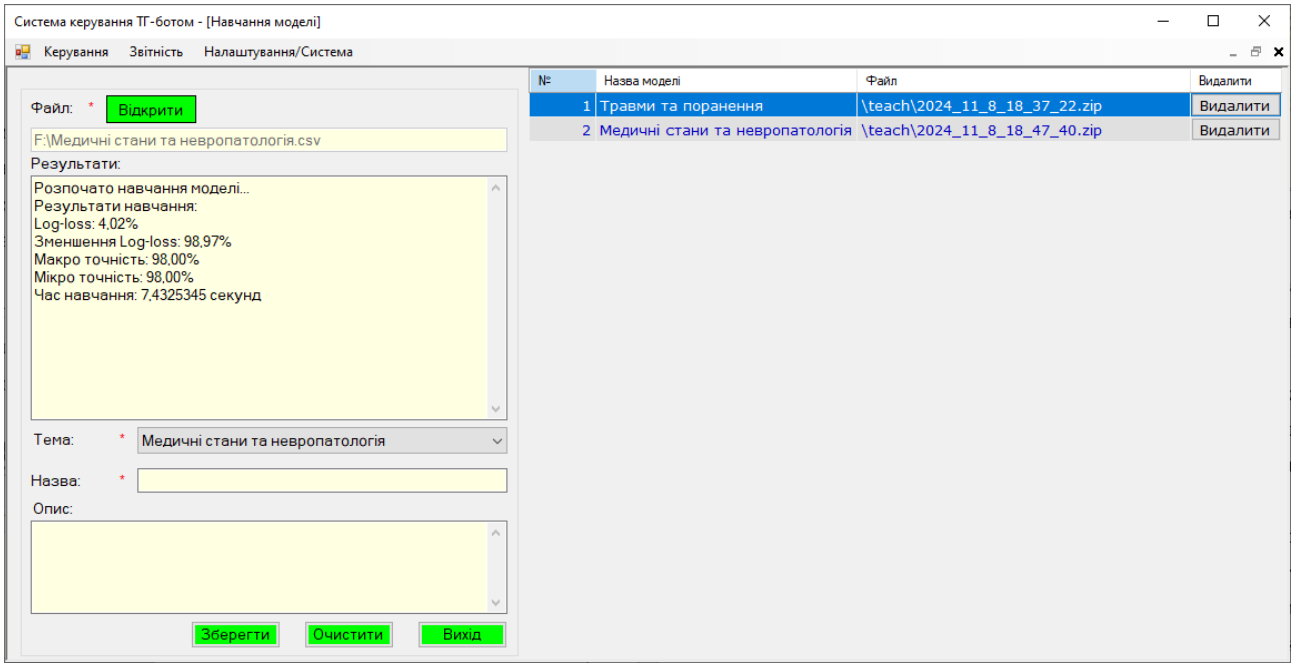

Рисунок 3.34 –Форма тренування моделей

У полі "Результати" відображається інформація про метрики навчання моделі, такі як Log-loss, Зменшення Log-loss, Макро точність, Мікро точність, а також Час навчання. Ці показники дозволяють оцінити ефективність моделі, яка була навчена на конкретному наборі даних. На правій стороні форми наведений список створених моделей із зазначенням їхніх назв та файлів, які використовувалися для тренування. Для кожної моделі доступна опція "Видалити," що дозволяє видалити непотрібну модель із системи.

Окрім цього, користувач може вибрати тему з випадаючого списку, ввести назву та опис моделі перед її збереженням у системі, натиснувши кнопку

«Зберегти». Форма також оснащена кнопками «Очистити» для скидання введених даних і "Вихід" для завершення роботи з інтерфейсом. Така структура дозволяє зручно завантажувати та тренувати моделі для різних медичних тем, забезпечуючи контроль і гнучкість у роботі з навчальними даними.

Форма для тестування моделей призначена для перевірки здатності системи надавати коректні відповіді на запитання користувачів у межах заданих тем. Як видно на рис. 3.35, інтерфейс дозволяє користувачеві вибрати тему для тестування з випадаючого списку, наприклад, "Медичні стани та невропатологія". Після цього користувач може ввести конкретне запитання в полі "Бесіда" і натиснути кнопку "Задати" для отримання відповіді.

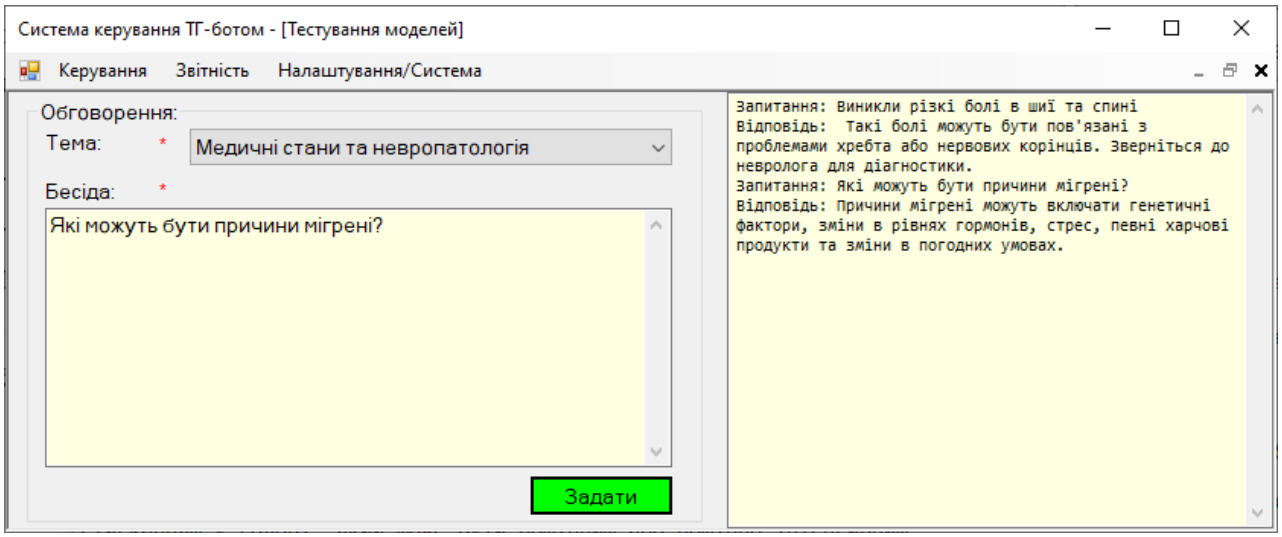

Рисунок 3.35 – Вікно для тестування моделі

На правій панелі форми відображається історія запитань та відповідей, що дає можливість аналізувати результати тестування в реальному часі. Наприклад, користувач задав питання "Які можуть бути причини мігрені?", і модель надала відповідь з можливими факторами, такими як генетичні причини, стрес або певні харчові продукти. Це дозволяє оцінити, наскільки точно та інформативно модель відповідає на запити.

Форма для тестування моделей забезпечує зручний інтерфейс для введення запитань і швидкого отримання відповідей, що є важливим етапом у процесі налаштування та вдосконалення алгоритмів. Вона допомагає розробникам і

користувачам оцінювати якість роботи системи в умовах, наближених до реального використання, і при необхідності коригувати модель.

Основне меню чат-бота у Telegram забезпечує користувачам швидкий доступ до ключових функцій системи. На рис. 3.36 показано три основні кнопки, розташовані в інтуїтивно зрозумілому інтерфейсі, які полегшують навігацію та взаємодію з ботом.

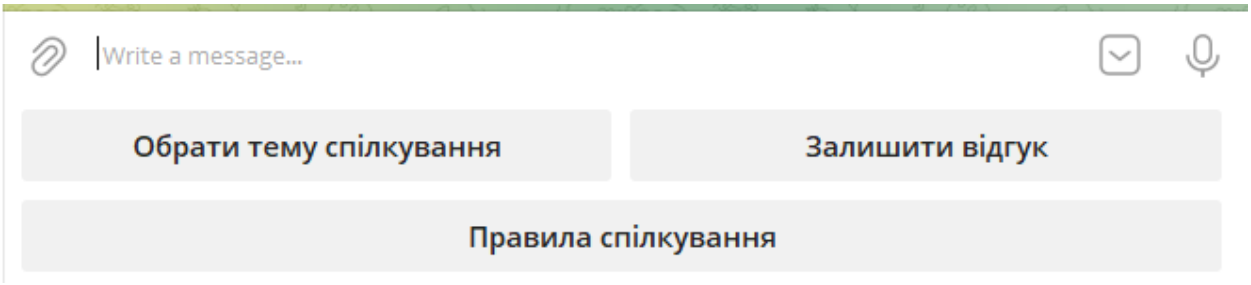

Рисунок 3.36 – Основне меню чат-бота

До меню бота належать:

– обрання теми спілкування – ця кнопка дозволяє користувачам обрати конкретну тему для отримання інформації або допомоги. Вибір теми допомагає чат-боту надати більш релевантні відповіді, що відповідають запиту користувача.

– залишення відгуку – кнопка надає можливість користувачам залишити відгук про роботу чат-бота, що є важливим для збору зворотного зв’язку та вдосконалення системи.

– правила спілкування – цей розділ містить етичні норми та рекомендації щодо взаємодії з ботом, що сприяє підтриманню ввічливого і продуктивного спілкування.

Ці функції в основному меню надають користувачам зручний доступ до основних сервісів чат-бота, забезпечуючи простоту та ефективність у використанні системи для медичних консультацій та отримання першої допомоги.

При використанні чат-боту для надання першої медичної допомоги користувачі можуть залишати відгуки, що сприяють вдосконаленню сервісу та

його функціональних можливостей. Для цього в основному меню бота передбачена кнопка "Залишити відгук", яка дозволяє користувачам висловити свою думку щодо роботи бота, а також надавати конструктивні пропозиції для покращення його функціоналу (рис. 3.37).

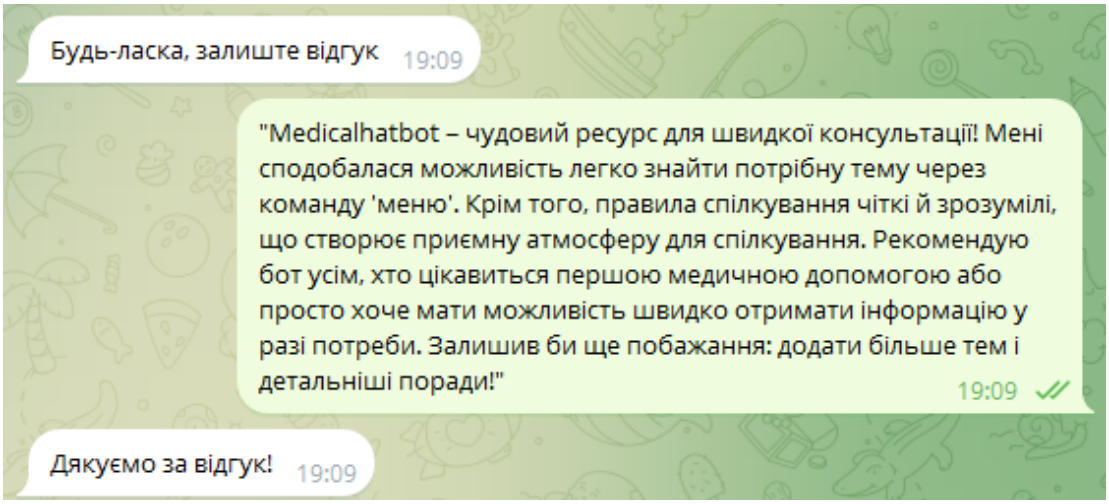

Рисунок 3.37 – Приклад відгуку користувача

Користувачі, які взаємодіють з Medicalhatbot, мають дотримуватися основних правил спілкування, що сприяють створенню приємної та ввічливої атмосфери в процесі отримання медичної допомоги (рис. 3.38).

При зверненні до бота, користувачеві рекомендується ознайомитися з базовими принципами етикету, зазначеними в розділі "Правила спілкування", що доступний у головному меню бота.

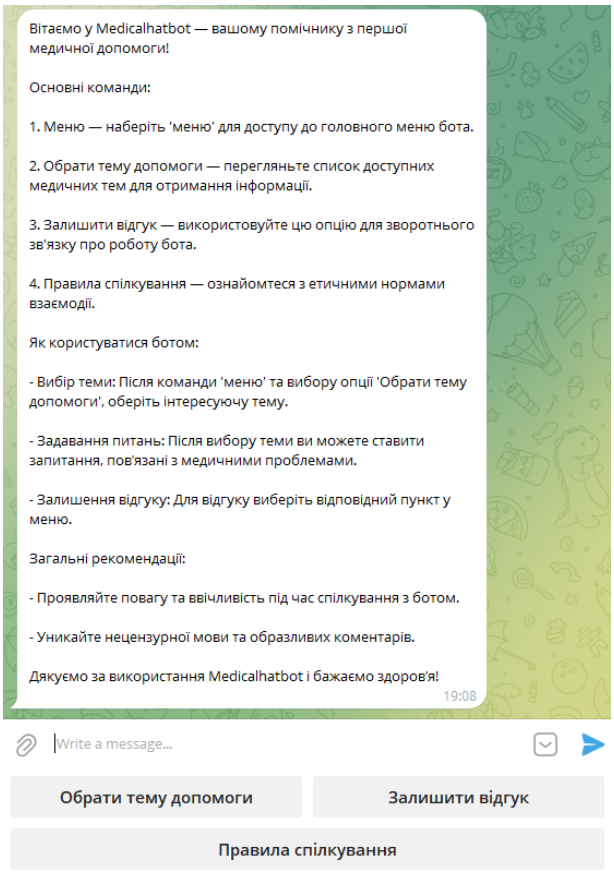

Рисунок 3.38 – Правила користування чат–ботом

Форма керування обліковими записами забезпечує адміністраторам системи можливість створення, редагування та управління обліковими записами користувачів у чат-боті (рис. 3.39). Інтерфейс форми дозволяє вводити основну інформацію про користувача, включаючи прізвище, ім'я, електронну пошту, опис, логін і пароль. Важливим елементом форми є вибір ролі користувача із запропонованого списку, який дозволяє призначити різні рівні доступу, такі як "Адміністратор". Для забезпечення безпеки, передбачено поле для повторного введення пароля, що допомагає уникнути помилок при його встановленні.

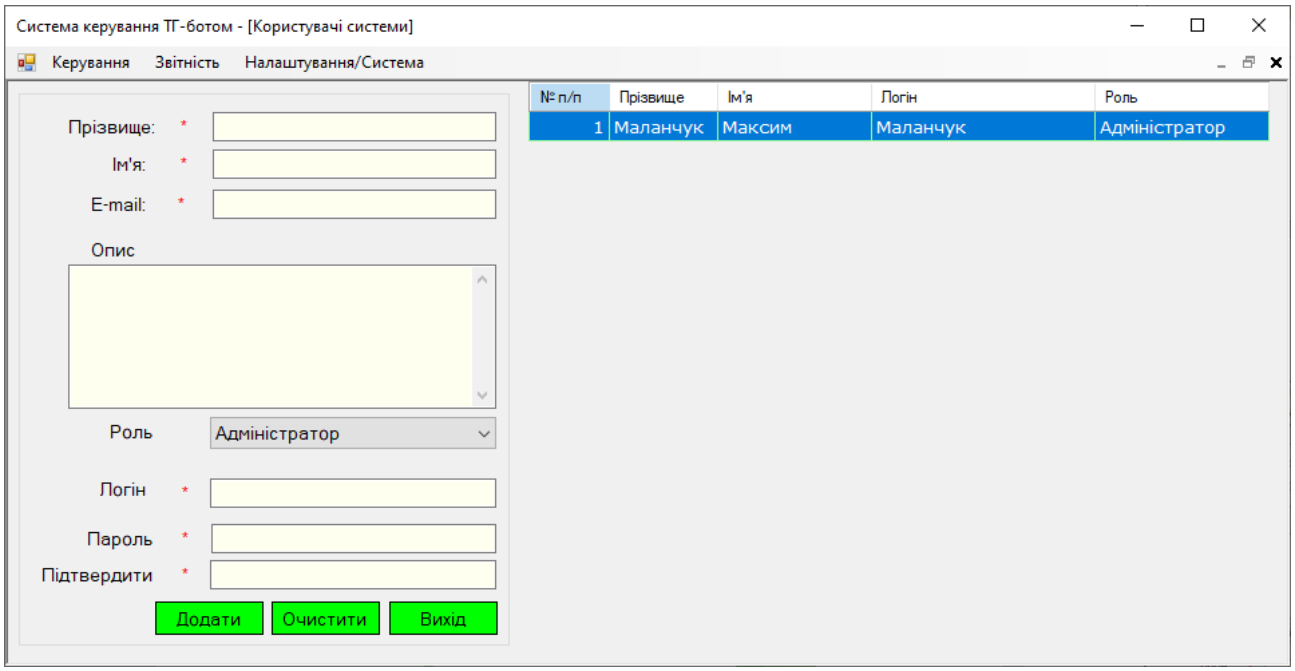

Рисунок 3.39 – Форма керування обліковими записами

На правій стороні форми представлено таблицю зі списком зареєстрованих користувачів, яка містить ключові поля, такі як прізвище, ім'я, логін та роль. Це дозволяє адміністратору легко переглядати наявні облікові записи, а також ідентифікувати їхні ролі та основні дані. Кнопки "Додати", "Очистити" та "Вихід" полегшують процес управління обліковими записами. Кнопка "Додати" дозволяє зберегти новий обліковий запис після введення всіх необхідних даних, кнопка "Очистити" скидає введену інформацію, а кнопка "Вихід" завершує роботу з формою. Така структура форми робить процес керування обліковими записами зручним і ефективним, забезпечуючи легкий доступ до основної інформації про користувачів системи та необхідний контроль за їхніми обліковими записами.

В системі можна також оглядати статистичні дані та запити, які користувачі залишили у діалогах з чат-ботами, як показано на рис. 3.40, де відображено інтерфейс для перегляду таких діалогів. Інтерфейс форми відображає список запитів, які користувачі надсилали боту, дозволяючи вибрати конкретну тему для фільтрації, наприклад, "Медичні стани та невропатологія". Це допомагає зосередитися на діалогах, що стосуються певної медичної тематики, і забезпечує зручний доступ до запитів за заданою категорією.

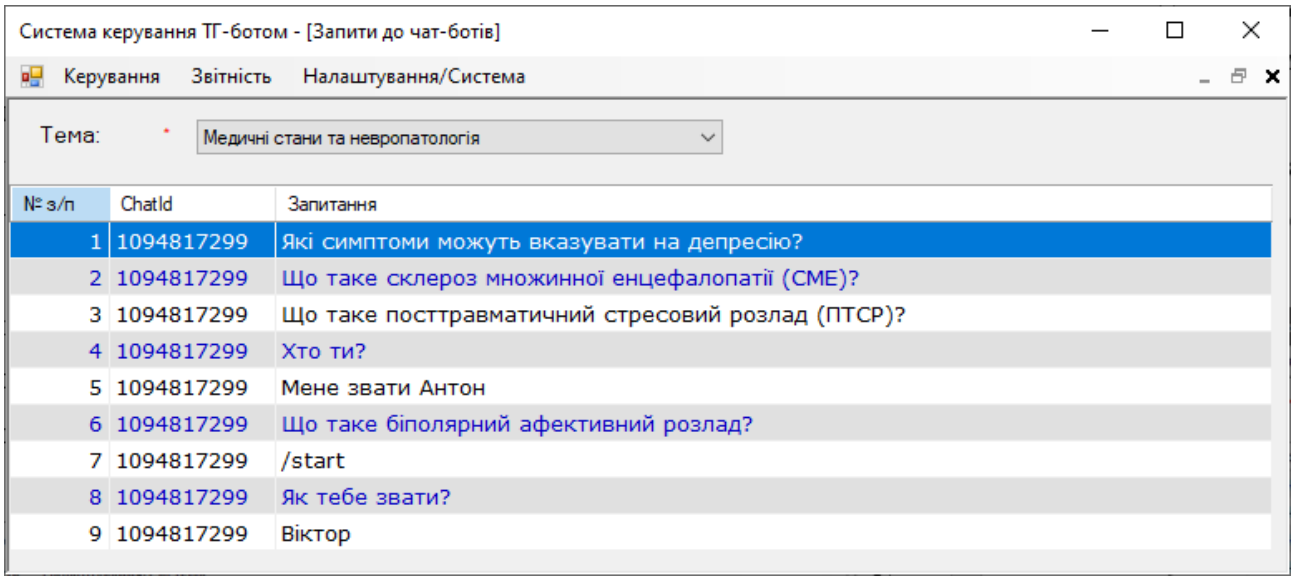

Рисунок 3.40 – Форма перегляду діалогів

Таблиця на формі включає кілька колонок: порядковий номер запиту, ідентифікатор чату (ChatId) і текст запиту. Ідентифікатор чату дає змогу відстежувати запити окремих користувачів, що особливо корисно для аналізу частих запитів від одного користувача або для розгляду випадків, де необхідні додаткові уточнення. Текст запиту надає інформацію про конкретні питання, які користувачі ставлять боту, що допомагає адміністраторам оцінювати релевантність відповідей, наданих системою, та виявляти потенційні теми для покращення. Форма забезпечує адміністративний контроль за роботою бота, дозволяючи зберігати історію діалогів, що є важливим для підтримки якості обслуговування та вдосконалення роботи чат-боту на основі реальних потреб користувачів.

Форма перегляду відгуків призначена для зручного доступу до зворотного зв’язку, який користувачі залишають про роботу чат-бота (рис. 3.41). Ця форма дозволяє адміністраторам системи переглядати деталі кожного відгуку, що включають унікальний ідентифікатор користувача (назва чату), прізвище, ім'я користувача, дату залишення відгуку та сам текст відгуку. Такий детальний формат забезпечує точне ідентифікування користувача, що залишив відгук, і дозволяє відслідковувати дати та зміст отриманого зворотного зв’язку.

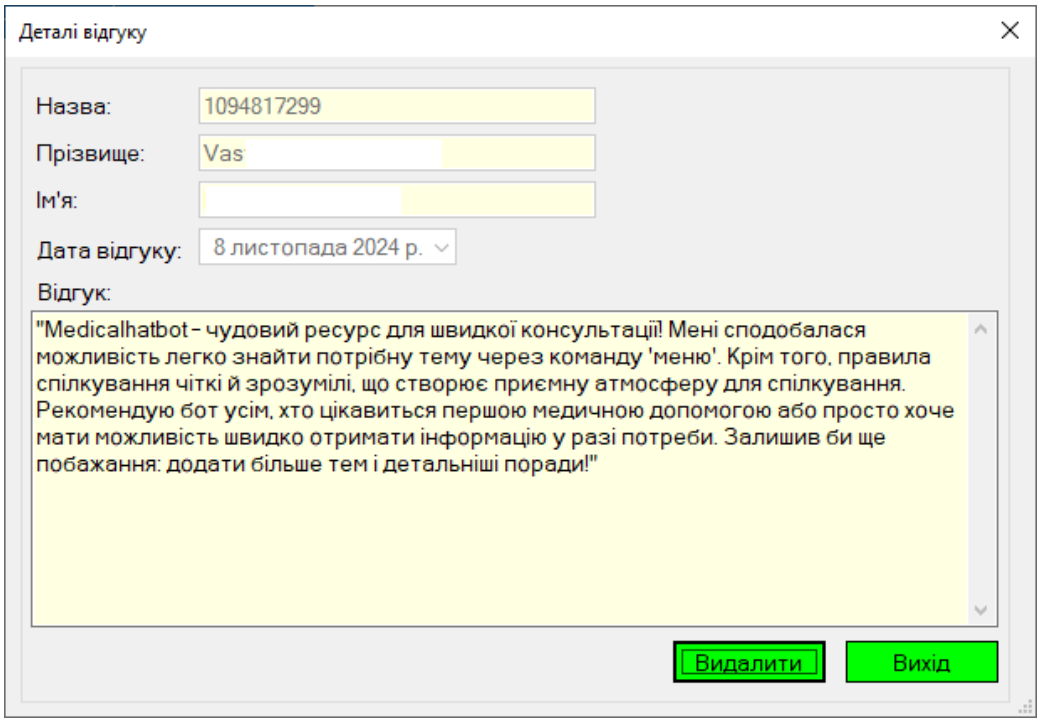

Рисунок 3.41 – Форма перегляду деталей відгуку

У текстовому полі "Відгук" відображається повний текст коментаря користувача, що дає можливість адміністраторам аналізувати позитивні аспекти взаємодії з ботом, а також враховувати пропозиції та побажання користувачів для подальшого вдосконалення сервісу.

Форма перегляду відгуків є важливим інструментом для забезпечення якості обслуговування, оскільки дозволяє системі бути більш орієнтованою на потреби користувачів і адаптуватися до їхніх очікувань, забезпечуючи своєчасний аналіз та обробку отриманого зворотного зв’язку.

При виборі пункту меню «Керування» –> «Вихід» користувач завершує роботу з програмою, і система автоматично закриває всі відкриті форми та виводить користувача з інтерфейсу. Це забезпечує швидке і коректне завершення роботи, зберігаючи налаштування та стан програми.

3.7 Висновок

У рамках даного розділу було розроблено, впроваджено та протестовано програмне забезпечення для автоматизованих рекомендацій на основі машинного навчання в Telegram-боті для медичної допомоги. Були розроблені

алгоритми машинного навчання для тренування моделей рекомендацій та алгоритм взаємодії користувача з моделлю чат-бота, що забезпечили коректне функціонування системи рекомендацій відповідно до запитів користувачів. Був створений спеціалізований модуль для інтеграції моделей у Telegram-бот, що включає написання коду для навчання і використання моделей, що дозволило ефективно впровадити їх у структуру системи.

Реалізовані окремі програмні компоненти, які забезпечують управління функціями ботів, організацію основного меню та взаємодію з користувачем. Для перевірки якості та надійності роботи системи проведено серію тестувань, зокрема, було розроблено і виконано тестові сценарії для форми авторизації та процесу тренування моделей, а також проведено MSTest тестування, результати якого підтвердили стабільну роботу системи.

Виконано оцінку роботи системи в експериментальних умовах на прикладі тестування моделі для теми «Медичні стани та невропатологія». В процесі навчання системою були виміряні метрики, такі як Log-loss на рівні 4.02% та макро і мікро точність, що становлять 98.0%, що свідчить про високу ефективність моделі. Окрім того, були розроблені інструкції для користувачів, що допомагають забезпечити зручність та коректність взаємодії з ботом під час надання і отримання медичної допомоги.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Питання щодо охорони праці

У процесі створення інформаційних систем, таких як чат-бот для автоматизованої рекомендації першої медичної допомоги, важливим є забезпечення належних умов праці розробників. Одним із ключових аспектів є дотримання правил охорони праці, спрямованих на попередження професійних ризиків, покращення продуктивності працівників та збереження їхнього здоров'я [49]. Сучасна IT-сфера ставить високі вимоги до фізичного та психологічного стану спеціалістів, оскільки тривала робота за комп'ютером може призвести до зорового та фізичного перенапруження.

Особливу увагу в умовах охорони праці приділяють ергономіці робочого місця. Ергономіка є комплексом заходів, спрямованих на створення комфортного та безпечного середовища для роботи. Вона враховує фізіологічні та психологічні особливості людини, оптимізуючи взаємодію працівника з робочим обладнанням і середовищем. Основна мета ергономіки – зменшення ризику травм та професійних захворювань, підвищення продуктивності та комфортності роботи [50].

Ергономічні принципи активно впроваджуються на підприємствах, що працюють у сфері розробки програмного забезпечення, де необхідно забезпечити ефективну та безпечну роботу за комп'ютером. Вони враховують розташування технічних засобів, освітлення, вентиляцію, а також режим праці та відпочинку. Оцінка цих параметрів дозволяє уникнути негативних наслідків, таких як втома чи порушення постави.

Нижче наведена табл. 4.1, яка містить основні рекомендації щодо організації ергономічного робочого місця програмістів.

Таблиця 4.1 – Рекомендації з організації ергономічного робочого місця

Параметр	Рекомендація	Мета впровадження	Наслідки недотримання
1	2	3	4
Розташування монітора	Відстань 50–70 см від очей, верхній край на рівні очей	Зменшення зорового напруження	Втома очей, зниження зору
Робочий стілець	Регульована висота, підтримка попереку	Забезпечення правильної постави	Болі у спині, порушення постави
Освітлення	Рівномірне, без бліків на екрані	Зниження втоми очей	Перенапруження очей, головний біль
Робочий стіл	Висота 70–75 см, достатній простір для рук	Комфортне розташування тіла	Порушення кровообігу, напруження м'язів
Частота перерв	Перерва 5–10 хв кожну годину роботи	Зменшення стомлюваності	Зниження продуктивності, перевтома
Клавіатура та миша	Розташування на одному рівні з ліктями	Зменшення навантаження на руки	Болі у зап’ястях, тунельний синдром
Вентиляція приміщення	Постійний доступ свіжого повітря	Підтримка комфортного мікроклімату	Задуха, зниження концентрації
Психологічний комфорт	Низький рівень шуму, комфортна температура	Збільшення продуктивності	Стрес, дратівливість

Дотримання рекомендацій щодо розташування технічних засобів, вибору меблів та створення комфортного мікроклімату допомагає знизити ризик професійних захворювань. Це також сприяє підвищенню ефективності роботи програмістів, дозволяючи уникнути негативних наслідків, пов’язаних з втомою та перенапруженням. Оптимальне середовище роботи є ключовим елементом збереження здоров’я працівників та покращення загального робочого процесу.

Робота програмістів пов’язана із тривалим перебуванням за комп’ютером, що може спричинити проблеми із зором (комп’ютерний зоровий синдром) та захворювання опорно-рухового апарату (остеохондроз, тунельний синдром). Для зниження ризиків важливо дотримуватись чітких рекомендацій щодо

тривалості роботи за комп’ютером, поєднуючи періоди активної діяльності з перервами. Режим праці повинен передбачати регулярні паузи для відпочинку очей і м’язів, що сприяє збереженню працездатності та запобіганню втомі.

Ці рекомендації використовуються в ІТ-компаніях, на підприємствах, а також у навчальних закладах, де робота з комп’ютером є невід’ємною частиною діяльності. Дотримання цих правил дозволяє зменшити негативний вплив тривалого використання комп’ютерів на здоров’я, підвищуючи комфорт та продуктивність роботи.

Нижче наведена табл. 4.2, яка містить основні рекомендації щодо тривалості роботи за комп’ютером для програмістів.

Таблиця 4.2 – Рекомендації щодо тривалості роботи за комп’ютером

Параметр	Рекомендація	Мета впровадження	Наслідки недотримання
Тривалість безперервної роботи	Не більше 2 годин	Зменшення навантаження на очі та спину	Втома, зниження продуктивності
Тривалість перерв	10–15 хвилин кожні 1–2 години	Відновлення фізичного та зорового балансу	Перевтома, погіршення зору
Фізичні вправи	Кожну годину: легкі вправи для розтягування спини та шиї	Запобігання захворюванням опорно-рухового апарату	Напруження, біль у м’язах
Максимальна тривалість роботи за день	Не більше 8 годин	Підтримка загального стану здоров’я	Хронічна втома, зниження концентрації
Організація робочого місця	Дотримання ергономічних вимог	Зменшення фізичного навантаження	Біль у спині, дискомфорт
Контроль перерв	Використання таймерів або програм для нагадування	Забезпечення регулярності перерв	Недотримання режиму праці
Використання спеціальних окулярів	Окуляри з фільтром для роботи з монітором	Зменшення шкідливого впливу екранного світла	Погіршення зору, підвищена втома очей

Регулярні перерви, фізичні вправи та виконання вправ для очей сприяють зменшенню негативного впливу тривалого використання комп’ютера. Впровадження таких заходів підвищує комфорт роботи, зберігаючи високу продуктивність та працездатність спеціалістів. Це важливий аспект організації праці в сучасних умовах високих інтелектуальних навантажень.

Розробка програмного забезпечення передбачає інтенсивну розумову діяльність, що може супроводжуватися значними психологічними навантаженнями. Високий рівень відповідальності, необхідність вирішення складних задач та дотримання жорстких дедлайнів часто стають причиною стресу та емоційного вигорання [51]. Для збереження психологічного комфорту важливим є впровадження спеціальних заходів, які допомагають мінімізувати негативний вплив стресових факторів та підтримувати загальний стан працівників.

Ці заходи включають організацію зручного робочого середовища, гнучкі графіки роботи, створення сприятливого мікроклімату в колективі, а також підтримку професійного розвитку та мотивації. У сучасних IT-компаніях усе частіше застосовують такі інструменти, як тренінги з управління стресом, групові психологічні консультації та програми підтримки ментального здоров’я. Ці практики допомагають зберегти ефективність і працездатність персоналу, зменшуючи ризик емоційного вигорання.

Нижче наведена табл. 4.3, яка містить основні рекомендації для забезпечення психологічного комфорту розробників.

Таблиця 4.3 – Рекомендації щодо забезпечення психологічного комфорту

Параметр	Рекомендація	Мета впровадження	Наслідки недотримання
1	2	3	4
Робочий графік	Гнучкий графік або віддалена робота	Зменшення рівня стресу через обмеження часу	Висока напруга, зниження мотивації
Психологічна підтримка	Проведення тренінгів з управління стресом	Підвищення стійкості до стресових ситуацій	Емоційне вигорання, стрес

Продовження таблиці 4.3

Перерви	Регулярні перерви для відпочинку	Зменшення інтелектуального навантаження	Перевтома, зниження концентрації
Соціальна взаємодія	Спільні заходи та командні активності	Зміцнення відносин у команді	Відчуття ізоляції, конфлікти
Організація робочого середовища	Забезпечення комфортного простору	Підвищення рівня задоволеності умовами праці	Роздратування, зниження продуктивності
Менторство	Надання підтримки та допомоги молодшим спеціалістам	Формування позитивного досвіду роботи	Стрес через брак знань чи ресурсів
Стимулювання професійного росту	Участь у тренінгах, курсах	Підвищення самооцінки та професійної впевненості	Низька мотивація, стагнація
Здоровий спосіб життя	Заохочення до спорту, відпочинку та правильного харчування	Покращення загального психофізичного стану	Хронічна втома, дратівливість

Забезпечення психологічного комфорту розробників є невід’ємною складовою їхньої продуктивної та тривалої діяльності. Впровадження заходів підтримки ментального здоров’я, створення комфортних умов роботи та стимулювання соціальної взаємодії дозволяє зменшити рівень стресу та уникнути емоційного вигорання. Це підвищує мотивацію працівників, сприяє збереженню їхнього здоров’я та оптимізує робочі процеси у компанії.

4.2 Питання щодо безпеки в надзвичайних ситуаціях

Безпека в надзвичайних ситуаціях спрямована на захист людей, матеріальних цінностей та довкілля у разі виникнення аварій, катастроф або природних стихійних явищ. Важливим аспектом є забезпечення швидкої реакції на надзвичайні події та мінімізація їхніх наслідків. Особливо актуально це для

систем охорони здоров’я, де оперативність у наданні першої медичної допомоги може врятувати життя постраждалих.

Інформаційні системи, такі як чат-боти на основі моделей класифікації, є сучасним інструментом підтримки безпеки в таких умовах. Вони забезпечують автоматизацію процесу аналізу симптомів, допомагають ухвалювати швидкі рішення та надають користувачам рекомендації у критичні моменти [52]. Впровадження таких рішень є надзвичайно важливим для служб швидкого реагування, медичних установ та індивідуальних користувачів.

Одним із ключових елементів у безпеці надзвичайних ситуацій є оперативність у наданні першої медичної допомоги, адже від швидкості реагування залежить подальший стан постраждалого. Наведена нижче табл. 4.4, містить основні аспекти, пов’язані із значенням оперативності.

Таблиця 4.4 – Значення оперативності у наданні першої медичної допомоги

Аспект	Рекомендація	Мета впровадження	Наслідки недотримання
Швидкість аналізу стану постраждалого	Використання автоматизованих систем	Скорочення часу для постановки попереднього діагнозу	Втрата часу, погіршення стану
Доступність інформації	Використання мобільних пристроїв і чат-ботів	Надання рекомендацій у будь-якому місці	Відсутність доступу до медичної допомоги
Навчання населення	Проведення курсів першої допомоги	Підвищення готовності до реагування	Паніка, неправильні дії
Стандартизація дій	Розробка алгоритмів для медичного персоналу та рятувальників	Забезпечення ефективності реагування	Некоректна або неорганізована допомога
Взаємодія служб	Інтеграція інформаційних систем різних організацій	Підвищення координації між службами	Хаотичність дій, втрата часу

Продовження таблиці 4.4

Постійний моніторинг	Використання сенсорів і систем відстеження стану пацієнтів	Профілактика ускладнень	Погіршення стану здоров’я
Розробка планів дій	Створення планів реагування на основі попереднього аналізу ризиків	Забезпечення готовності до кризових ситуацій	Неефективність у надзвичайних умовах

Оперативність у наданні першої медичної допомоги є критично важливим чинником у мінімізації наслідків надзвичайних ситуацій. Застосування автоматизованих систем, ефективна комунікація між службами та стандартизація дій сприяють швидшому та більш якісному наданню допомоги. Такі підходи дозволяють значно зменшити втрати часу та підвищити шанси на успішне врятування життя постраждалих.

Інформаційні системи дозволяють збирати, обробляти та передавати дані, необхідні для швидкого реагування на загрози. Такі системи використовуються в державних службах, лікарнях, службах екстреної допомоги, а також у приватному секторі для моніторингу ризиків, попередження небезпек та координації дій у кризових умовах. Наприклад, автоматизовані чат-боти можуть оперативно надавати рекомендації з надання першої допомоги, зменшуючи час на пошук інформації. Інформаційні системи застосовуються для оповіщення населення про небезпеку, управління евакуацією, прогнозування та аналізу ризиків, а також для організації роботи служб швидкого реагування. Вони забезпечують ефективну взаємодію між різними відомствами, дозволяючи швидше ухвалювати рішення та надавати необхідну підтримку. Це стає можливим завдяки використанню технологій штучного інтелекту, машинного навчання та аналітичних інструментів.

Нижче наведена табл. 4.5, яка містить основні аспекти використання інформаційних систем у забезпеченні безпеки населення.

Таблиця 4.5 – Використання інформаційних систем для забезпечення безпеки населення

Аспект	Рекомендація	Мета впровадження	Наслідки недотримання
1	2	3	4
Моніторинг загроз	Використання сенсорів і систем аналізу даних	Своєчасне виявлення потенційних небезпек	Затримка у виявленні загроз
Оповіщення населення	Мобільні додатки та системи SMS-оповіщення	Швидке інформування про небезпеку	Відсутність інформації у критичний момент
Координація дій служб	Інтегровані платформи для взаємодії між організаціями	Підвищення ефективності реагування	Некоректна або несинхронізована робота
Аналіз ризиків	Використання прогнозних моделей	Запобігання небезпекам	Неефективне планування заходів
Підтримка евакуації	Інформаційні карти з маршрутами евакуації	Забезпечення безпеки під час евакуації	Паніка, хаос, втрата часу
Підтримка медичних служб	Системи класифікації для аналізу симптомів	Швидка діагностика і перша допомога	Втрата часу на діагностику
Збір та обробка даних	Хмарні платформи для збереження та обміну даними	Централізоване управління інформацією	Дублювання або втрата даних
Навчання населення	Мобільні додатки з інструкціями щодо дій в екстрених ситуаціях	Підвищення обізнаності населення	Низька готовність до надзвичайних ситуацій

Інформаційні системи дозволяють своєчасно виявляти загрози, оповіщати населення, координувати дії служб та надавати підтримку у вирішенні кризових ситуацій. Використання таких систем значно підвищує ефективність заходів з безпеки, зменшуючи ризики для життя і здоров’я громадян.

Чат-боти є сучасним інструментом автоматизації, який дозволяє оперативно надавати інформацію та рекомендації в умовах надзвичайних ситуацій. Вони створені для аналізу запитів користувачів, генерування відповідей та допомоги у критичних ситуаціях, таких як аварії, стихійні лиха або

техногенні катастрофи. Зокрема, чат-боти можуть оперативно оповіщати населення про небезпеку, надавати маршрути евакуації, роз'яснювати дії в екстрених умовах або допомагати визначити необхідність медичної допомоги.

Застосування таких систем можливе у службах швидкого реагування, медичних установах, організаціях цивільного захисту, а також серед індивідуальних користувачів. Наприклад, чат-бот може інформувати про місця укриттів, безпечні зони чи доступні ресурси в реальному часі. Використання моделей класифікації дозволяє системі швидко обробляти введені дані та знаходити оптимальні рішення для користувача.

Нижче наведена табл. 4.6, яка містить основні аспекти застосування чат-ботів у випадках надзвичайних ситуацій.

Таблиця 4.6 – Використання чат-ботів у надзвичайних ситуаціях

Аспект	Рекомендація	Мета впровадження	Наслідки недотримання
Інформування населення	Оперативне оповіщення через мобільні платформи	Запобігання паніці та надання точних даних	Відсутність актуальної інформації
Аналіз запитів	Автоматичний аналіз текстових повідомлень	Прискорення реакції на запити користувачів	Затримки у відповідях
Рекомендації з першої допомоги	Надання інструкцій відповідно до описаних симптомів	Підтримка постраждалих до прибуття медиків	Неправильна або відсутня допомога
Навігація в умовах кризи	Генерація маршрутів евакуації або до укриттів	Забезпечення безпеки населення	Втрати часу, дезорієнтація
Підтримка служб	Інтеграція з системами екстреного виклику	Координація дій між службами	Некоректна організація роботи
Мультимовність	Забезпечення доступності інформації для іноземних громадян	Розширення охоплення населення	Непорозуміння, відсутність комунікації
Робота в умовах обмеженого зв'язку	Використання офлайн-режимів або мінімального інтернету	Забезпечення стабільної роботи навіть у кризових умовах	Відмова системи, втрата зв'язку
Персоналізація допомоги	Аналіз індивідуальних потреб користувача	Оптимізація наданої інформації	Надання нерелевантної інформації

Отже, чат-боти відіграють важливу роль у реагуванні на надзвичайні ситуації, забезпечуючи оперативне інформування та підтримку населення. Вони можуть аналізувати запити, надавати рекомендації та координувати дії служб, що сприяє зменшенню наслідків кризових подій. Впровадження таких систем дозволяє значно підвищити ефективність заходів безпеки, мінімізуючи ризики для життя та здоров’я громадян.

4.3 Висновок

У даному розділі висвітлюються ключові аспекти, які впливають на ефективність роботи програмістів та застосування розроблених інформаційних систем у кризових умовах. Було акцентовано увагу на необхідності створення безпечного та комфортного робочого середовища для розробників, зокрема шляхом забезпечення ергономіки робочого місця, дотримання режиму роботи за комп’ютером і підтримки психологічного комфорту. Це дозволяє мінімізувати ризики для здоров’я спеціалістів та підвищити продуктивність їхньої праці.

Окрему увагу приділено ролі інформаційних систем у забезпеченні безпеки населення під час надзвичайних ситуацій. Відзначено важливість оперативності у наданні першої медичної допомоги, використання автоматизованих рішень для аналізу ризиків та координації дій між службами. Чат-боти, як інструмент сучасних інформаційних технологій, довели свою ефективність у забезпеченні швидкого реагування, інформуванні та підтримці постраждалих.

Впровадження подібних рішень сприяє не лише покращенню умов праці програмістів, але й підвищенню загального рівня безпеки населення. Це забезпечує стабільну роботу інформаційних систем у кризових ситуаціях, зменшуючи ризики для здоров’я і життя громадян.

ВИСНОВКИ

У результаті проведеного дослідження та розробки в даній кваліфікаційній роботі була вирішена науково-технічна задача, яка полягала у створенні ефективної системи автоматизованої медичної допомоги на основі машинного навчання, реалізованої у вигляді Telegram-бота. Така система дозволяє оперативно надавати користувачам рекомендації з першої медичної допомоги, базуючись на сучасних медичних стандартах і використовуючи методи класифікації для формування персоналізованих рекомендацій.

Теоретичне значення отриманих результатів полягає у розробці алгоритмів машинного навчання для надання медичних рекомендацій, які можуть бути застосовані в інших системах підтримки прийняття рішень у сфері охорони здоров’я. Практичне значення роботи виявляється у створенні Telegram-бота, здатного працювати з великими обсягами медичних запитів, забезпечуючи точність і швидкість відповіді. В ході роботи був проведений аналіз існуючих рішень у сфері мобільних додатків для медичної допомоги та алгоритмів машинного навчання, що дозволило визначити оптимальні технології та методи для побудови системи.

Отримані результати свідчать про високу ефективність розробленого програмного рішення. Зокрема, модель для теми «Медичні стани та невропатологія» показала Log-loss на рівні 4.02% та досягла макро- і мікро точності у 98.0%, що підтверджує точність і надійність алгоритмів класифікації. Тестування, проведене з використанням MSTest, показало стабільну роботу всіх компонентів системи, зокрема функціоналу авторизації, тренування моделей та взаємодії з користувачами.

Достовірність отриманих результатів обґрунтовується використанням загальноприйнятих методів тестування та аналізу моделей машинного навчання, що підтверджує надійність розробленої системи. Результати тестувань, отримані під час експериментальних перевірок у реальних умовах, свідчать про готовність системи до використання для надання медичних консультацій.

Для практичного використання розробленої системи рекомендовано впровадження Telegram-бота у закладах охорони здоров’я як додаткового інструменту для інформування населення про першу медичну допомогу. Використання такого бота сприятиме швидкому доступу громадян до достовірної інформації щодо базових медичних рекомендацій, що особливо важливо у випадках надзвичайних ситуацій або віддаленого доступу до фахівців.

Окрім того, система може бути корисною для навчальних цілей у медичних закладах, надаючи студентам і молодим спеціалістам можливість ознайомитися з алгоритмами діагностики та першої допомоги в інтерактивному форматі. Такий підхід підвищує якість навчального процесу і дозволяє закріпити знання на практиці. Система також може бути інтегрована з іншими медичними інформаційними системами для надання автоматизованих рекомендацій та підтримки медичних рішень, що підвищує її цінність як універсального інструменту для сучасної медицини.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Prismana, I. G. L. P. E., Prehanto, D. R., Dermawan, D. A., Herlingga, A. C., & Wibawa, S. C. Nazief & Adriani Stemming Algorithm With Cosine Similarity Method For Integrated Telegram Chatbots With Service. In IOP Conference Series: Materials Science and Engineering. 2021. Vol. 1125, No. 1, 9 p.

2. Домедична допомога в умовах бойових дій: Методичний посібник / В.Д. Юрченко, В.О.Крилюк, А.А.Гудима та ін.. – К.: Середняк Т.К., 2014, - 80 с.

3. Гур'єв С., Волянський П., Терент'єва А. та ін. Організація та управління процесом надання медичної допомоги постраждалим внаслідок землетрусів: монографія. – Переяслав-Хмельницький: СКД, 2008. – 188 с.

4. ЗАКОН УКРАЇНИ. Про екстрену медичну допомогу. URL: https://zakon.rada.gov.ua/laws/show/5081-17#Text (дата звернення 23.10.2024).

5. Хуторянський, О., & Останін, В. Потерпілий від злочину, передбаченого статтею 136 Кримінального кодексу України: кримінально-правовий аналіз. Юридичний часопис Національної академії внутрішніх справ, 2011. Вип. 1 №1, с. 71-77.

6. ЗАКОН УКРАЇНИ. Про охорону праці. URL: https://zakon.rada.gov.ua/laws/show/2694-12#Text (дата звернення 23.10.2024).

7. Мислива О. Основи надання патрульною поліцією невідкладної (домедичної та медичної) допомоги постраждалим особам: навч. посібник / О. О. Мислива. Дніпро: Дніпроп. держ. ун-т внутр. справ, 2018. 144 с.

8. Кришмарел, В. Релігієзнавчий складник у формуванні предметної історичної компетентності учнів ЗНЗ. Релігія та Соціум. 2016. Вип.3, с. 197-202.

9. Дрига Н. Медико-соціальне обґрунтування оптимізації системи управління якістю медичної допомоги пацієнтам із цукровим діабетом 2-го типу на рівні первинної ланки охорони здоров'я : дис.д-ра філософії : 222. Суми, 2022. 168 с.

10. Денова, Л., Іванов, Д., Андруневич, Р., Корж, О., & Красюк, Е. Нефрологічна допомога в умовах воєнного стану в Україні. Нирки. 2022. Вип. 11 №3. 2022, 372 с.

11. Olasveengen, T. M., Mancini, M. E., Perkins, G. D., Avis, S., Brooks, S., Castrén, M., ... & Morley, P. T. Adult basic life support: 2020 international consensus on cardiopulmonary resuscitation and emergency cardiovascular care science with treatment recommendations. Circulation. 2020. Vol. 142, No. 16, pp. 41-91.

12. Greif, R., Bhanji, F., Bigham, B. L., Bray, J., Breckwoldt, J., Cheng, A., ... & Finn, J. C. Education, implementation, and teams: 2020 international consensus on cardiopulmonary resuscitation and emergency cardiovascular care science with treatment recommendations. Circulation. 2020. Vol. 142, No. 16, pp. 222-283.

13. Delhomme, C., Njeim, M., Varlet, E., Pechmajou, L., Benameur, N., Cassan, P., ... & Karam, N. Automated external defibrillator use in out-of-hospital cardiac arrest: Current limitations and solutions. Archives of cardiovascular diseases. 2019. Vol. 112, No. 3, pp. 217-222.

14. Bou-Karroum, L., El-Jardali, F., Jabbour, M., Harb, A., Fadlallah, R., Hemadi, N., & Al-Hajj, S. Preventing unintentional injuries in school-aged children: a systematic review. Pediatrics, 2022. 149 p.

15. Jost, G., Allsop, R., Steriu, M., & Enculescu, S. A Challenging Start towards the EU 2020 Road Safety Target. European Transport Safety Council. 2012. 92 p.

16. I. Strutynska, H. Kozbur, L. Dmytrotsa, O. Sorokivska, L. Melnyk, R. Grytseliak. Regarding to the Concept of Small and Medium-Sized Enterprises Digitalization in Ukraine: Problems and Solutions. IEEE Deggendorf, Germany, 2021,pp.276–279.

17. I. Strutynska, L. Dmytrotsa, H. Kozbur, L. Melnyk, R.Sherstiuk: The Unification of Approaches to Measuring the Digital Maturity of Business Structures (International and Domestic Approaches). ICTERI 2021: Kherson, Ukraine, September 28 - October 2, 2021. CEUR Workshop Proceedings 3013, CEUR-WS.org 2021.pp.10-23.

18. First Aid - IFRC 4+ . URL: https://apps.apple.com/us/app/first-aid-ifrc/id1312876691 (дата звернення 23.10.2024).

19. Comparative review of the First Aid App. URL: https://preparecenter.org/wp-content/sites/default/files/gdpc_first_aid_app_review_d5_final_trilateral.pdf (дата звернення 23.10.2024).

20. First Aid Goes Mobile: A Review. URL: https://redcrosschat.org/2012/07/11/first-aid-goes-mobile-a-review/ (дата звернення 23.10.2024).

21. Ada – check your health. URL: https://play.google.com/store/apps/details?id=com.ada.app&hl=en_US (дата звернення 23.10.2024).

22. Ada App - Chat based health diagnostics | UI Sources. URL: https://www.uisources.com/app/ada (дата звернення 23.10.2024).

23. Survey of Multimodal Medical Question Answering. URL: https://www.researchgate.net/publication/377026586_Survey_of_Multimodal_Medical_Question_Answering (дата звернення 23.10.2024).

24. About Ada. URL: https://www.softwareadvice.com/product/290363-ada/ (дата звернення 23.10.2024).

25. Comparison of Diagnostic and Triage Accuracy of Ada Health and WebMD Symptom Checkers. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC10582809/ (дата звернення 23.10.2024).

26. Doctor On Demand Review: Tested by Our Experts in 2024. URL: https://www.helpguide.org/handbook/online-therapy/doctor-on-demand-review (дата звернення 23.10.2024).

27. Doctor-on-Demand: A Practical Guide to Creating a Telemedicine App. URL: https://easternpeak.com/blog/doctor-on-demand-a-practical-guide-to-creating-a-telemedicine-app/ (дата звернення 23.10.2024).

28. Doctor On Demand Review. URL: https://www.onlinedoctor.com/doctor-on-demand-review/ (дата звернення 23.10.2024).

29. Doctor on Demand Review 2024: Pros & Cons, Cost, & My Experience. URL: https://www.choosingtherapy.com/doctor-on-demand-review/ (дата звернення 23.10.2024).

30. Doctor on Demand App Development: Features, Cost. URL: https://www.cleveroad.com/blog/doctor-on-demand-app-development/ (дата звернення 23.10.2024).

31. Geng, Y., Li, Q., Yang, G., & Qiu, W. Logistic regression. In Practical Machine Learning Illustrated with KNIME 2024. pp. 99-132.

32. Khandezamin, Z., Naderan, M., & Rashti, M. J. Detection and classification of breast cancer using logistic regression feature selection and GMDH classifier. Journal of Biomedical Informatics. 2020. Volume 111, 250 p.

33. Yoo, S. H., Geng, H., Chiu, T. L., Yu, S. K., Cho, D. C., Heo, J., ... & Lee, H. Deep learning-based decision-tree classifier for COVID-19 diagnosis from chest X-ray imaging. Frontiers in medicine, 2020. Vol. 7, 427 p.

34. Mendonça, Y. V., Naranjo, P. G. V., & Pinto, D. C. The role of technology in the learning process: a decision tree-based model using machine learning. Emerg. Sci. J. 2023. Vol., pp. 280-295.

35. Tangirala, S. Evaluating the impact of GINI index and information gain on classification using decision tree classifier algorithm. International Journal of Advanced Computer Science and Applications. 2020. Vol. 11, No. 2, pp. 612-619.

36. Sahin, E. K. Assessing the predictive capability of ensemble tree methods for landslide susceptibility mapping using XGBoost, gradient boosting machine, and random forest. SN Applied Sciences. 2020. Vol. 2, No. 7, 1308 p.

37. Wang, Q., Zhou, Y., Ding, W., Zhang, Z., Muhammad, K., & Cao, Z. Random forest with self-paced bootstrap learning in lung cancer prognosis. ACM Transactions on Multimedia Computing, Communications, and Applications. 2020. Vol. 16, No. 1, pp. 1-12.

38. Martínez-Gramage, J., Albiach, J. P., Moltó, I. N., Amer-Cuenca, J. J., Huesa Moreno, V., & Segura-Ortí, E. A random forest machine learning framework to reduce running injuries in young triathletes. 2020. Vol. 20, No. 21, 638 p.

39. Strutynska, I., Dmytrotsa, L., Kozbur, H., Ermolayev, V., Mallet, F., Yakovyna, V., ... & Spivakovsky, A. (2019, June). The Main Barriers and Drivers of the Digital Transformation of Ukraine Business Structures. In ICTERI (pp. 50-64).

40. Analysis of the SMEs' Digitalization State Using HIT Index and Machine Learning Technique Strutynska, I., Kozbur, H., Dmytrotsa, L., Kozbur, I., Gashchyn, N. Proceedings - International Conference on Advanced Computer Information Technologies, ACIT, 2023, pp. 332–337

41. Working-Out of Recommendation System to Increase the Digital Maturity Level of Enterprises Strutynska, I., Dmytrotsa, L., Kozbur, H., Hlado, O., Sorokivska, O. 2020 IEEE International Conference on Problems of Infocommunications Science and Technology, PIC S and T 2020 - Proceedings, 2021, pp. 147–151

42. Villalobos, K., Ramirez-Duran, V. J., Diez, B., Blanco, J. M., Goni, A., & Illarramendi, A. A three level hierarchical architecture for an efficient storage of industry 4.0 data. Computers in Industry, 2020. – 121p.

43. Кравчук, О. А., & Синюк, О. М. Переваги та недоліки мікросервісної архітектури. 2020. 4 с.

44. Кулинич, О. О. Теоретичне обґрунтування доцільності використання сервіс-орієнтованої архітектури в діяльності підприємства. 2015. 5 с.

45. Ganesh, R., & Prabu, G. Determination of internet banking usage and purpose with explanation of data flow diagram and use case diagram. International Journal of Management and Humanities, 4(7), 2020. – pp. 52-58.

46. Гладун А.В. Трьохрівнева архітектура побудови веб-систем. Вісник Національного університету "Львівська політехніка". 2018. – 315 с.

47. Посібники з надання першої допомоги. URL: https://medical-club.net/uk/spravochniki-po-okazaniyu-pervoy-pomoshchi/#pervpomua3 (дата звернення 15.11.2024).

48. Рецептіка. URL: https://receptika.ua/ua/stati/kak-okazat-pervuyu-pomosch-obschie-pravila (дата звернення 15.11.2024).

49. Health, lifestyle and occupational risks in Information Technology workers. URL: https://academic.oup.com/occmed/article/71/2/68/6124582?utm_source=chatgpt.com&login=false (дата звернення 15.11.2024).

50. The Importance of Ergonomics for Software Engineers: A Comprehensive Guide. URL: https://dev.to/ysinghchouhan/the-importance-of-ergonomics-for-

software-engineers-a-comprehensive-guide-4a22?utm_source=chatgpt.com (дата звернення 30.11.2024).

51. Penzenstadler, B., Torkar, R., & Martinez Montes, C. Take a deep breath: Benefits of neuroplasticity practices for software developers and computer workers in a family of experiments. Empirical Software Engineering. 2022. Vol. 27, No. 4, 98 p.

52. Grassini, E., Buzzi, M., Leporini, B., & Vozna, A. A systematic review of chatbots in inclusive healthcare: insights from the last 5 years. Universal Access in the Information Society. 2024. pp. 1-9.