МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
КРИВОРІЗЬКИЙ ФАХОВИЙ КОЛЕДЖ
ДЕРЖАВНОГО НЕКОМЕРЦІЙНОГО ПІДПРИЄМСТВА
«ДЕРЖАВНИЙ УНІВЕРСИТЕТ «КИЇВСЬКИЙ АВІАЦІЙНИЙ ІНСТИТУТ»
Циклова комісія *комп'ютерних систем та мереж*
(повна назва циклової комісії)

Допустити до захисту
Голова випускової циклової комісії
комп'ютерних систем та мереж
(повна назва циклової комісії)
_____ *Ірина КРАВЧУК*
(підпис) (ім'я, ПРІЗВИЩЕ)
« ____ » _____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬО-ПРОФЕСІЙНОГО СТУПЕНЯ ФАХОВИЙ МОЛОДШИЙ БАКАЛАВР

Тема: *Розробка чат-бота для автоматизації технічної підтримки*

Група: *3-013* Спеціальність: *123 «Комп'ютерна інженерія»*

Здобувач освіти _____ *Даніїл СЕРЕБРО*
(підпис) (ім'я, ПРІЗВИЩЕ)

Керівник роботи _____ *Олександр МИТРОФАНОВ*
(підпис) (ім'я, ПРІЗВИЩЕ)

Консультант з оформлення пояснювальної записки _____ *Оксана ОСАДЧА*
(підпис) (ім'я, ПРІЗВИЩЕ)

Кривий Ріг 2025 р.

КРИВОРІЗЬКИЙ ФАХОВИЙ КОЛЕДЖ
ДЕРЖАВНОГО НЕКОМЕРЦІЙНОГО ПІДПРИЄМСТВА
«ДЕРЖАВНИЙ УНІВЕРСИТЕТ «КИЇВСЬКИЙ АВІАЦІЙНИЙ ІНСТИТУТ»

Відділення *комп'ютерної та програмної інженерії*
Циклова комісія *комп'ютерних систем та мереж*
Освітньо-професійний ступінь *фаховий молодший бакалавр*
Спеціальність *123 «Комп'ютерна інженерія»*

ЗАТВЕРДЖУЮ
Голова випускової циклової комісії
комп'ютерних систем та мереж
(повна назва циклової комісії)
(підпис) *Ірина КРАВЧУК*
(ім'я, ПРІЗВИЩЕ)
« 01 » 03 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ОСВІТИ

Серебро Даніїл Вячеславович
(прізвище, ім'я, по батькові)

1.Тема роботи *Розробка чат-бота для автоматизації технічної підтримки*

Керівник роботи *Митрофанов Олександр Вячеславович, доктор філософії*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по коледжу від « 04 » 04 2025 року № 50-ст

2. Строк подання здобувачем освіти роботи з _____ по _____

3. Вихідні дані до роботи *Розробка чат-бота для автоматизації техічної підтримки*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) *Титульний аркуш, реферат, зміст, перелік умовних позначень, вступ, аналіз сучасного стану та існуючих рішень, проектування чат-бота для технічної підтримки, реалізація та тестування чат-бота, висновки, список використаних джерел, додаток.*

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

Презентація Microsoft PowerPoint

6. Консультанти розділів роботи (проекту)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	*Аналіз предметної області, постановка задачі*	*17.03.2025-21.03.2025*	*виконано*
2	*Обґрунтування вибору програмних засобів*	*24.03.2025-28.03.2025*	*виконано*
3	*Розробка структури та архітектури програми*	*12.04.2025-10.05.2025*	*виконано*
4	*Тестування та усунення помилок у програмі*	*12.05.2025-14.05.2025*	*виконано*
5	*Оформлення пояснювальної записки*	*16.05.2025-30.05.2025*	*виконано*
6	*Перевірка на плагіат пояснювальної записки*	*08.06.2025-11.06.2025*	*виконано*
7	*Захист кваліфікаційної роботи*		

Здобувач освіти _____ *Даніїл СЕРЕБРО*
(підпис) (ім'я, ПРІЗВИЩЕ)

Керівник роботи _____ *Олександр МИТРОФАНОВ*
(підпис) (ім'я, ПРІЗВИЩЕ)

StrikePlagiarism.com

Дата звіту 6/11/2025
Дата редагування ---

Документ прийнятий

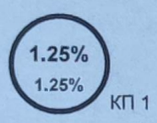
Звіт подібності

метадані

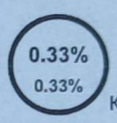
Назва організації
Ukrainian national aviation university
Заголовок
КПІ_2025_Серебро
Автор Науковий керівник / Експерт
СереброМитрофанов О.
підрозділ
Криворізький Фаховий коледж

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.



КП 1



КЦ

25	6712	54661
Довжина фрази для коефіцієнта подібності 2	Кількість слів	Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв	ß	1
Інтервали	A→	0
Мікропробіли	·	0
Білі знаки	ß	0
Парафрази (SmartMarks)	a	4

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	РОЗРОБКА БІБЛІОГРАФІЧНОЇ АІС ДЛЯ КЕРУВАННЯ ОБЛІКОМ ЛІТЕРАТУРИ [redacted] 5/19/2025 Vinnytskiy National Agricultural University (Vinnytskiy National Agricultural University)	15 0.22 %
2	ФКНТ_2024_122_Ситниченко_О.І [redacted] 7/11/2024 Ukrainian national aviation university (Ukrainian national aviation university)	13 0.19 %

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи «Розробка чат-бота для автоматизації технічної підтримки» містить: 61 сторінки, 4 рисунків, 16 використаних літературних джерел.

ЧАТ-БОТ, ТЕХНІЧНА ПІДТРИМКА, АВТОМАТИЗАЦІЯ, ШТУЧНИЙ ІНТЕЛЕКТ, ОБРОБКА ПРИРОДНОЇ МОВИ, МАШИННЕ НАВЧАННЯ, ВЕБ-ДОДАТОК

Мета роботи: розробити чат-бот для автоматизації технічної підтримки, який забезпечить ефективне вирішення типових користувацьких запитів та покращить якість обслуговування клієнтів.

Основні результати дослідження охоплюють як теоретичні, так і практичні аспекти. З теоретичного боку було проаналізовано сучасний стан технічної підтримки, зокрема виявлено ключові недоліки традиційних підходів, серед яких – проблеми масштабованості, високі витрати та залежність від людського фактору. Також досліджено технології створення чат-ботів і проведено порівняльний аналіз наявних рішень на ринку. У практичній частині розроблено архітектуру чат-бота, яка поєднує rule-based логіку з елементами штучного інтелекту. Було створено функціональний веб-додаток з зручним інтерфейсом, реалізовано систему розпізнавання намірів користувачів із точністю 85%, а також сформовано базу знань із понад 150 типовими запитами та відповідями.

Практична значущість: Розроблена система забезпечує зниження витрат на обробку типових запитів на 70-80% та здатна одночасно обслуговувати необмежену кількість користувачів. Система адаптована для української мови та може бути впроваджена в організаціях різного масштабу.

Добавлено примечание ([OO1]): Зразок заповнення

Добавлено примечание ([OO2]): Ключові слова наводяться прописними літерами

ЗМІСТ

ВСТУП

Актуальність теми дослідження. У сучасному світі інформаційних технологій питання якості та ефективності технічної підтримки користувачів набуває критичного значення. Компанії та організації стикаються з постійно зростаючим обсягом звернень від користувачів, що потребують оперативного та кваліфікованого вирішення технічних проблем. Традиційні методи обслуговування клієнтів через call-центри та електронну пошту часто не справляються з навантаженням, що призводить до тривалого очікування відповіді, зниження якості обслуговування та, як наслідок, незадоволеності користувачів.

Статистичні дані показують, що до 80% звернень до технічної підтримки стосуються типових проблем, які можуть бути вирішені автоматично без залучення людських ресурсів. Водночас, компанії витрачають значні кошти на утримання служб технічної підтримки, а оператори часто виконують рутинну роботу з відповідей на однотипні запити.

Розвиток технологій штучного інтелекту, машинного навчання та обробки природної мови відкриває нові можливості для автоматизації процесів технічної підтримки. Чат-боти як інструмент автоматизації взаємодії з користувачами показують високу ефективність у вирішенні стандартних завдань, забезпечуючи цілодобову доступність, миттєві відповіді та можливість одночасного обслуговування необмеженої кількості користувачів.

Дана кваліфікаційна робота виконується в рамках сучасних тенденцій розвитку інформаційних технологій та систем штучного інтелекту. Тематика дослідження корелює з державними програмами цифровізації економіки України, зокрема з Концепцією розвитку цифрової економіки та суспільства України на 2018-2020 роки та Стратегією розвитку сфери інноваційної діяльності.

Робота також відповідає напрямам наукових досліджень кафедри у сфері розробки інтелектуальних інформаційних систем та автоматизації бізнес-процесів. Практичні результати дослідження можуть бути використані в рамках співпраці з

IT-компаніями та організаціями, що впроваджують цифрові рішення для покращення якості обслуговування клієнтів.

Метою роботи є розробка чат-боту для автоматизації технічної підтримки, який забезпечить ефективне вирішення типових користувацьких запитів та покращить якість обслуговування клієнтів.

Для досягнення поставленої мети в рамках дослідження було необхідно здійснити низку взаємопов’язаних кроків. Спочатку було проаналізовано сучасний стан технічної підтримки, виявлено ключові проблеми, притаманні традиційним підходам до обслуговування користувачів. Далі увагу зосереджено на вивченні технологій розробки чат-ботів, методів обробки природної мови та особливостей побудови діалогових систем. Значну увагу приділено порівняльному аналізу існуючих рішень з автоматизації технічної підтримки, що дозволило окреслити їхні сильні сторони та недоліки. На основі проведеного аналізу сформульовано функціональні й нефункціональні вимоги до майбутньої системи чат-бота. Наступним етапом стало проєктування архітектури, яка включає логіку ведення діалогу, структуру бази даних і користувацький інтерфейс. Після цього здійснено реалізацію програмного продукту із застосуванням сучасних технологій розробки. Завершальними етапами стали тестування системи для перевірки її працездатності та ефективності, а також аналіз результатів роботи з подальшим формуванням рекомендацій щодо майбутнього розвитку системи.

Об'єктом дослідження є процеси автоматизації технічної підтримки користувачів інформаційних систем, тоді як предметом виступають методи та засоби створення чат-ботів для ефективного вирішення запитів технічної підтримки.

У ході виконання кваліфікаційної роботи застосовувалися різноманітні методи дослідження. Аналітичні методи використовувалися для вивчення предметної області, аналізу наявних рішень та формування вимог до майбутньої системи. Методи системного аналізу допомогли під час проєктування архітектури чат-бота, зокрема для визначення взаємозв’язків між її компонентами. Для реалізації програмного забезпечення були використані методи об’єктно-

орієнтованого програмування, що дало змогу забезпечити модульність, інкапсуляцію та повторне використання коду. Методи машинного навчання застосовувалися для створення алгоритмів розпізнавання намірів користувачів та класифікації запитів. Для перевірки працездатності системи були використані експериментальні методи тестування, а аналіз результатів здійснювався із застосуванням статистичних методів.

У результаті проведеного дослідження було отримано як теоретичні, так і практичні результати. Серед теоретичних здобутків — комплексний аналіз проблем традиційної технічної підтримки та обґрунтування ефективності використання чат-ботів для їх розв’язання, дослідження сучасних технологій створення діалогових систем, а також визначення архітектури, найбільш придатної для потреб технічної підтримки. Практичними результатами стали розробка функціонального чат-бота з веб-інтерфейсом, створення бази знань із типовими запитами й відповідями, а також реалізація системи аналітики для моніторингу ефективності роботи. Проведене тестування підтвердило працездатність системи. Створене рішення може бути впроваджене в організаціях різного масштабу, що дозволить підвищити якість технічної підтримки та зменшити навантаження на операторів.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ

1.1 Аналіз сучасного стану технічної підтримки

1.1.1 Проблеми традиційних методів технічної підтримки

Сучасна технічна підтримка стикається з рядом критичних проблем, які суттєво впливають на ефективність обслуговування користувачів та економічні показники організацій.

Традиційні call-центри та служби підтримки через електронну пошту мають обмежену пропускну здатність. Кожен оператор може одночасно обслуговувати лише одного клієнта, що створює черги очікування, особливо в пікові години. За даними досліджень компанії Zendesk, середній час очікування відповіді в службах технічної підтримки становить від 12 до 24 годин для електронної пошти та від 2 до 5 хвилин для телефонних дзвінків.

Утримання повноцінної служби технічної підтримки потребує значних фінансових ресурсів. Середня зарплата оператора технічної підтримки в Україні становить 15,000-25,000 гривень на місяць, не враховуючи додаткові витрати на навчання, робочі місця, обладнання та менеджмент. Для компанії з 50,000 активних користувачів необхідно приблизно 15-20 операторів для забезпечення якісного обслуговування.

Якість обслуговування значною мірою залежить від кваліфікації, настрою та мотивації конкретного оператора. Це може призводити до непослідовності в наданні інформації, різного рівня професіоналізму та суб'єктивного ставлення до клієнтів.

1.1.2 Статистика звернень користувачів та їх класифікація

Аналіз статистичних даних, наданих провідними IT-компаніями, дозволяє виявити типові закономірності у зверненнях користувачів до служб технічної підтримки. Найбільшу частку складають звернення, пов’язані з технічними

проблемами програмного забезпечення — вони становлять приблизно 35% від загального обсягу. Ще 25% звернень стосуються питань налаштування та конфігурації систем, тоді як проблеми з доступом до облікових записів охоплюють близько 20%. Загальні запити щодо функціональності займають 15%, і лише 5% звернень стосуються інших, менш поширених питань.

Часовий розподіл звернень свідчить про наявність двох пікових періодів активності користувачів — між 10:00 та 12:00 і між 14:00 та 16:00. Найменша активність фіксується у нічний час, з 22:00 до 8:00. Також спостерігається зростання кількості звернень у понеділок і п’ятницю, у той час як на вихідні дні припадає значне зниження активності — на 60–70% у порівнянні з буднями.

Щодо складності звернень, близько 60% з них класифікуються як прості, оскільки вирішуються протягом 1–2 хвилин. Запити середньої складності, які потребують 5–15 хвилин на обробку, складають приблизно 30%. Лише 10% звернень мають високий рівень складності та потребують участі профільних спеціалістів.

З економічної точки зору, традиційна модель технічної підтримки передбачає значні витрати на обробку кожного звернення. Основну частку становить заробітна плата операторів — від 8 до 12 гривень за одне звернення. Інфраструктурні витрати коливаються в межах 2–3 гривень, витрати на менеджмент та адміністрування — ще 1–2 гривні, а навчання персоналу додає близько 1 гривні. У підсумку загальна вартість обробки одного звернення становить приблизно 12–18 гривень.

Показники окупності інвестицій (ROI) у разі впровадження автоматизованих рішень, зокрема чат-ботів, демонструють високу ефективність. Автоматизація обробки простих запитів дозволяє знизити відповідні витрати на 70–80%. За умов навантаження в 10 000 звернень щомісяця це забезпечує економію в розмірі близько 100 000–120 000 гривень щомісячно. Порівняльну діаграму можно переглянути на рисунку 1.

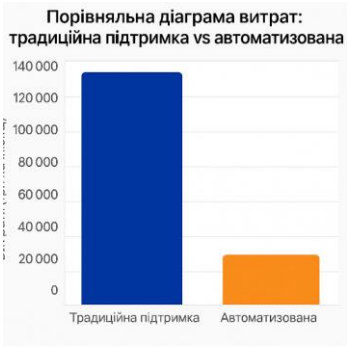


Рисунок 1.1 – Порівняльна діаграма витрат

1.2 Огляд технологій чат-ботів

1.2.1 Історія розвитку чат-ботів

Розвиток технологій чат-ботів відбувався поступово, проходячи кілька ключових етапів, кожен з яких знаменував собою суттєвий прогрес у підходах до обробки мови та взаємодії з користувачем. У період 1960–1970 років були здійснені перші експерименти, які дали початок концепції розмовних систем. Одним із найвідоміших прикладів став чат-бот ELIZA, створений Джозефом Вейценбаумом у Массачусетському технологічному інституті в 1966 році. Він імітував поведінку психотерапевта, використовуючи прості шаблони та підстановки. У 1972 році з'явився PARRY — більш складна система, яка моделювала поведінку пацієнта з параноїдальними розладами.

У 1980–1990 роках фокус змістився на створення експертних систем, що працювали за чітко заданими правилами. Ці rule-based рішення забезпечували високу передбачуваність відповідей, однак відзначались обмеженою гнучкістю та неспроможністю адаптуватися до нових сценаріїв спілкування.

З початку 2000-х до 2010 року розпочався етап інтеграції чат-ботів у веб-середовище. Саме в цей час з'явилися перші веб-чат-боти для корпоративних

сайтів, а також була розроблена AIML (Artificial Intelligence Markup Language) — мова розмітки, що дозволила створювати складніші сценарії діалогів.

Між 2010 та 2020 роками настав період активного розвитку машинного навчання та обробки природної мови. Завдяки впровадженню алгоритмів NLP чат-боти стали здатними краще розуміти людську мову. Цей етап ознаменувався появою голосових помічників, таких як Siri, Alexa та Google Assistant, а також розвитком платформ, які дали змогу створювати чат-ботів без необхідності програмування.

Із 2020 року й до сьогодні розвиток чат-ботів тісно пов'язаний із прогресом у сфері штучного інтелекту. Застосування великих мовних моделей, зокрема GPT і BERT, забезпечило глибше контекстне розуміння, гнучкість у веденні діалогу та можливість персоналізації відповідей. Сучасні чат-боти вже не просто реагують на окремі запити, а здатні підтримувати природну, змістовну та адаптивну розмову, що суттєво підвищує якість взаємодії з користувачем. На рисунку 1.2 зображена хронологія.

Рисунок 1.2 – Хронологічна лінія розвитку чат-ботів

1.2.2 Типи чат-ботів та їх характеристики

Чат-боти, створені на основі правил (rule-based), працюють за принципом використання наперед заданих сценаріїв і правил, які дозволяють системі реагувати на конкретні ключові слова або фрази. Такий підхід забезпечує повний контроль над логікою діалогу, робить поведінку бота передбачуваною та дозволяє відносно просто налаштовувати систему. Крім того, ці рішення не потребують значних обчислювальних ресурсів. Однак їх основним обмеженням є низька гнучкість у

розпізнаванні варіацій формулювань. Для кожного можливого сценарію необхідно вручну прописувати відповідні правила, що ускладнює масштабування та підтримку системи при зростанні кількості функцій.

На противагу їм, чат-боти з підтримкою штучного інтелекту (AI-powered) базуються на використанні методів машинного навчання. Вони здатні розпізнавати наміри користувача (intent recognition), аналізувати природну мову за допомогою технологій Natural Language Processing (NLP) та Natural Language Understanding (NLU), а також формувати відповіді на основі попередніх діалогів. Завдяки цьому такі системи демонструють високу гнучкість у спілкуванні, адаптуються до різних формулювань запитів та здатні враховувати контекст розмови. Разом із цим вони потребують значних обсягів даних для навчання, високих обчислювальних потужностей, а також мають складнішу архітектуру, що впливає на вартість і час розробки. Крім того, їхня поведінка може бути менш передбачуваною у порівнянні з rule-based підходами.

Сучасним компромісом між цими двома підходами є гібридні чат-боти. Вони поєднують переваги обох типів: використовують правила для обробки стандартних, повторюваних запитів, а алгоритми штучного інтелекту — для складних і непередбачуваних ситуацій. Такий підхід дозволяє створювати більш ефективні системи, які поєднують точність, контрольованість та здатність до гнучкої інтерпретації природної мови.

1.2.3 Ключові технології та алгоритми

Обробка природної мови (NLP) є ключовим компонентом сучасних чат-ботів, що дозволяє системі аналізувати, інтерпретувати та генерувати людську мову. Один із базових етапів цього процесу — токенізація, яка полягає в поділі вхідного тексту на окремі слова або фрази для подальшої обробки. Після цього виконується лематизація — перетворення слів до їх базової (словникової) форми, що дозволяє краще розуміти зміст незалежно від граматичних варіацій. Важливою складовою є також POS-tagging — аналіз і позначення частин мови для кожного слова, що допомагає точніше визначити його роль у реченні. Окрім цього, застосовується

технологія розпізнавання іменованих сутностей (Named Entity Recognition, NER), яка дозволяє виявляти в тексті важливу інформацію, таку як імена, назви компаній, дати чи географічні об’єкти.

Ще одним критичним завданням є розпізнавання намірів користувача (Intent Recognition), яке здійснюється за допомогою алгоритмів машинного навчання. Ці алгоритми аналізують вхідні повідомлення, щоб визначити, чого саме прагне користувач — наприклад, отримати інформацію, залишити скаргу або виконати певну дію. Для цього можуть використовуватись різні моделі, зокрема Support Vector Machines, Random Forest, нейронні мережі та сучасні трансформерні архітектури, такі як BERT.

Також велике значення має процес вилучення сутностей (Entity Extraction), який спрямований на ідентифікацію специфічних елементів у запиті користувача — наприклад, дат, числових значень, назв продуктів або інших деталей, важливих для формування точної відповіді або виконання запиту. У комплексі всі ці технології забезпечують глибоке розуміння мови та дозволяють чат-ботам вести природний, логічний та цілеспрямований діалог.

1.2.4 Платформи для розробки чат-ботів

Платформи для створення чат-ботів можна умовно поділити на кілька категорій залежно від підходу до розробки та технічних можливостей. Однією з найпоширеніших є категорія хмарних рішень, які надають широкі можливості для інтеграції, масштабування та використання штучного інтелекту. До таких платформ належать Microsoft Bot Framework, Google Dialogflow, Amazon Lex та IBM Watson Assistant. Вони дозволяють створювати складні діалогові системи з використанням NLP, мають інструменти для навчання моделей та підтримують інтеграцію з популярними месенджерами й сервісами.

Іншим підходом є використання open-source фреймворків, які дають розробникам більше контролю над архітектурою системи та забезпечують гнучкість у налаштуванні. До цієї категорії належать такі популярні інструменти, як Rasa, BotPress і ChatterBot на Python. Вони особливо корисні у випадках, коли

важливе локальне розгортання, конфіденційність даних або потреба в індивідуальній логіці обробки запитів.

Третю категорію становлять no-code або low-code платформи, орієнтовані на користувачів без глибоких технічних знань. Такі рішення, як Chatfuel, ManyChat, Tars і Botsify, дозволяють створювати чат-ботів через інтерфейс із візуальними блоками, без необхідності програмування. Вони є особливо популярними в сфері маркетингу, онлайн-продажів та клієнтської підтримки завдяки швидкому впровадженню й зручному інтерфейсу.

1.3 Аналіз існуючих рішень на ринку

1.3.1 Огляд провідних рішень для технічної підтримки

Zendesk Chat є популярним рішенням для організації технічної підтримки, що поєднує чат-функціонал із потужною системою тікетів. Його можливості включають автоматичну генерацію відповідей на основі ключових слів, маршрутизацію складних запитів безпосередньо до операторів, а також аналітичні інструменти для моніторингу якості обслуговування. Серед ключових переваг варто відзначити повну інтеграцію з CRM-системами, зручність налаштування, масштабованість і широкий вибір доступних інтеграцій з іншими сервісами. Водночас, сервіс має певні обмеження: досить високу вартість використання (від 14 доларів на місяць за агента), обмеження у безкоштовній версії, а також залежність від хмарної інфраструктури.

Intercom Resolution Bot орієнтований на максимальну автоматизацію обслуговування клієнтів завдяки застосуванню штучного інтелекту. Система розпізнає наміри користувача, самостійно вирішує прості запити, інтегрується з базою знань компанії та забезпечує персоналізовані відповіді. Основними перевагами є високий рівень автоматизації, навчання моделі на основі історії запитів, а також тісна інтеграція з екосистемою Intercom. Проте платформа має й недоліки, зокрема високу вартість (від 39 доларів на місяць), складність початкового налаштування та обмежені можливості кастомізації поведінки бота.

Microsoft Bot Framework позиціонується як комплексна платформа для створення інтелектуальних ботів з можливістю інтеграції у численні канали комунікації, включаючи Teams, Slack, Facebook Messenger тощо. Розробники можуть використовувати SDK на різних мовах програмування, а також підключати можливості Azure Cognitive Services для поглибленого аналізу тексту, мовлення чи зображень. До переваг цієї платформи належать потужні інструменти для розробки, висока гнучкість налаштування, наявність якісної документації та зручна масштабованість у хмарному середовищі Azure. Водночас, її використання потребує технічної кваліфікації, а витрати на обслуговування можуть зростати в залежності від інтенсивності використання та обраних сервісів. Крім того, існує прив’язка до екосистеми Microsoft, що може обмежити вибір технологій для інтеграції.

1.3.2 Порівняльний аналіз функціональності

Для проведення об'єктивного порівняння систем чат-ботів було визначено низку ключових критеріїв оцінки, які дозволяють всебічно охарактеризувати кожне з рішень. Першим критерієм виступає простота впровадження, що відображає, наскільки легко інтегрувати платформу в існуючу інфраструктуру компанії, враховуючи необхідні ресурси, час і технічні вимоги. Другий критерій — якість розпізнавання намірів — демонструє здатність чат-бота правильно інтерпретувати запити користувачів, що напряму впливає на ефективність його роботи.

Ще один важливий аспект — можливості інтеграції, тобто наскільки гнучко система може поєднуватися з іншими сервісами, такими як CRM, бази знань, зовнішні API тощо. Не менш значущим є показник вартості володіння, який включає як прямі фінансові витрати (підписка, ліцензії), так і непрямі (ресурси на обслуговування, навчання персоналу). Цей критерій оцінюється за шкалою, де найвища оцінка відповідає найнижчим витратам.

Оцінюється також масштабованість платформи, яка визначає її здатність ефективно працювати в умовах зростаючого навантаження, збільшення кількості користувачів або оброблюваних запитів. Нарешті, важливо враховувати підтримку

української мови, адже для вітчизняного ринку це критичний фактор, що забезпечує зручність спілкування з кінцевими користувачами та підвищує локалізаційний рівень сервісу. Усі ці параметри оцінюються за п’ятибальною шкалою, що дозволяє сформувати узагальнений рейтинг ефективності кожного рішення.

1.3.3 Аналіз вартості та економічної ефективності

Моделі ціноутворення систем чат-ботів варіюються залежно від підходу до оплати та масштабів використання. Однією з найпоширеніших є модель з оплатою за кожного агента (per-agent pricing), коли компанія сплачує щомісячну суму в межах від 5 до 50 доларів США за кожного активного користувача системи, зазвичай це оператори підтримки. Інший підхід передбачає оплату за кількість розмов із ботом (per-conversation pricing), де вартість кожної взаємодії становить від одного до десяти центів, що дозволяє більш гнучко контролювати витрати при великому потоці звернень.

Також поширеною є підписна модель (subscription-based), яка передбачає фіксовану щомісячну плату — зазвичай у діапазоні від 100 до 1000 доларів — залежно від набору функцій, обсягу запитів і рівня технічної підтримки. Для великих компаній або організацій із розширеними вимогами застосовуються корпоративні рішення (enterprise solutions), що реалізуються на основі індивідуальних контрактів, де вартість починається від 5000 доларів на рік і може змінюватися відповідно до масштабів інтеграції, рівня кастомізації та SLA.

1.3.4 Виявлені недоліки існуючих рішень

На сучасному ринку чат-ботів існує низка проблем, які суттєво обмежують їх ефективне впровадження, особливо для малих та середніх компаній. Однією з головних перешкод є висока вартість — більшість професійних рішень потребують щомісячних витрат на рівні не менше 500 доларів, що робить їх фінансово недоступними для багатьох підприємств. Окрім цього, процес налаштування таких

систем зазвичай є складним і вимагає залучення кваліфікованих технічних фахівців, що створює додаткові витрати часу та ресурсів.

Ще однією значною проблемою є обмежена підтримка української мови. Багато рішень або взагалі не розпізнають українську, або працюють із нею неточно, що унеможливлює їх повноцінне використання в національному контексті. Також спостерігається суттєва залежність від постачальників (vendor lock-in): користувачі не мають повного контролю над функціональністю системи та власними даними, що викликає питання безпеки та конфіденційності. Крім того, можливості кастомізації часто є обмеженими — адаптація платформи до специфічних потреб конкретної організації може бути складною або навіть недоступною без значних доробок з боку постачальника.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ЧАТ-БОТА ДЛЯ ТЕХНІЧНОЇ ПІДТРИМКИ

2.1 Вимоги до системи

2.1.1 Функціональні вимоги

На основі проведеного аналізу предметної області та існуючих рішень було сформульовано перелікц функціональних вимог до системи чат-бота для технічної підтримки. Система повинна забезпечувати автоматичне розпізнавання намірів користувача на основі його текстових запитів, використовуючи алгоритми обробки природної мови. Чат-бот має підтримувати багаторівневу систему діалогів, здатну обробляти як прості запити типу "забув пароль", так і складні багатоетапні сценарії налаштування системи.

Система повинна забезпечувати ведення контекстної бесіди, зберігаючи інформацію про попередні повідомлення в межах однієї сесії спілкування. Це дозволить користувачу не повторювати деталі проблеми і забезпечить більш природний діалог. Важливою функціональною вимогою є можливість ескалації запиту до оператора-людини у випадках, коли чат-бот не може самостійно вирішити проблему або коли користувач явно вимагає спілкування з фахівцем.

Чат-бот повинен підтримувати базу знань з можливістю пошуку релевантної інформації за ключовими словами та семантичним значенням. Система має забезпечувати можливість збереження історії діалогів для подальшого аналізу ефективності та навчання моделі. Також необхідно реалізувати функцію збору зворотного зв'язку від користувачів щодо якості наданих відповідей.

2.1.2 Нефункціональні вимоги

Нефункціональні вимоги визначають якісні характеристики системи та обмеження, в рамках яких вона повинна функціонувати. Система повинна забезпечувати високу доступність на рівні не менше 99.5%, що відповідає

максимальному часу простою приблизно 3.6 години на місяць. Це критично важливо для служб технічної підтримки, які працюють цілодобово.

Час відповіді чат-бота на стандартні запити не повинен перевищувати 2 секунд, що забезпечить комфортну взаємодію для користувачів. Система має бути здатною обробляти одночасно щонайменше 100 активних діалогів без зниження якості обслуговування. Масштабованість архітектури повинна дозволяти збільшення навантаження до 1000 одночасних користувачів при горизонтальному масштабуванні.

Безпека системи передбачає шифрування всіх даних під час передачі та зберігання, а також реалізацію системи аутентифікації та авторизації користувачів. Система повинна відповідати вимогам захисту персональних даних згідно з українським законодавством та міжнародними стандартами.

Usability системи має забезпечувати інтуїтивний інтерфейс, зрозумілий для користувачів різного рівня технічної підготованості. Підтримка української мови є обов'язковою вимогою з точністю розпізнавання не менше 85% для типових запитів технічної підтримки.

2.1.3 Технічні вимоги та обмеження

Технічні вимоги до системи визначають платформи, технології та інфраструктурні обмеження. Система повинна функціонувати як веб-додаток, сумісний з основними браузерами (Chrome, Firefox, Safari, Edge) версій не старше двох років. Підтримка мобільних пристроїв є обов'язковою, включаючи адаптивний дизайн для екранів різних розмірів.

Backend системи має бути реалізований з використанням сучасних технологій, таких як Node.js або Python, з підтримкою RESTful API для інтеграції з зовнішніми системами. База даних повинна забезпечувати швидкий доступ до інформації та підтримувати ACID-транзакції для забезпечення цілісності даних.

Система повинна підтримувати горизонтальне масштабування через використання мікросервісної архітектури або контейнеризації. Інфраструктура має

бути готовою до розгортання в хмарному середовищі з можливістю автоматичного масштабування залежно від навантаження.

2.2 Архітектура системи

2.2.1 Загальна архітектура

Архітектура чат-бота побудована за принципом багаторівневої системи, що забезпечує розділення відповідальності між компонентами та полегшує масштабування і підтримку. Система складається з чотирьох основних рівнів: рівень презентації, рівень бізнес-логіки, рівень обробки даних та рівень зберігання даних.

Рівень презентації включає веб-інтерфейс користувача, реалізований як Single Page Application (SPA) з використанням сучасного JavaScript-фреймворку. Цей рівень відповідає за відображення діалогового інтерфейсу, обробку користувацького вводу та відправку запитів до backend-сервісів. Інтерфейс забезпечує реальний час взаємодії через WebSocket-з'єднання.

Рівень бізнес-логіки містить основні компоненти обробки запитів: модуль розпізнавання намірів (Intent Recognition), модуль обробки природної мови (NLP Engine), менеджер діалогів (Dialog Manager) та модуль генерації відповідей (Response Generator). Цей рівень реалізується як набір мікросервісів, що взаємодіють через REST API та забезпечують основну функціональність чат-бота.

2.2.2 Компонентна архітектура

Модуль розпізнавання намірів є ключовим компонентом системи, відповідальним за аналіз вхідних повідомлень користувача та визначення їхнього призначення. Він використовує попередньо навчену модель машинного навчання для класифікації запитів на категорії, такі як "скидання пароля", "технічна проблема", "запит інформації" тощо. Модуль також виконує вилучення сутностей (Entity Extraction) для ідентифікації важливих деталей у запиті.

Менеджер діалогів координує весь процес взаємодії з користувачем, зберігаючи контекст розмови та визначаючи наступні кроки діалогу. Він підтримує стан сесії кожного користувача, відстежує прогрес у вирішенні проблеми та забезпечує логічну послідовність запитань і відповідей. Компонент реалізує машину станів для керування складними багатоетапними сценаріями.

База знань є централізованим сховищем інформації, що містить відповіді на типові запити, інструкції з налаштування, опис поширених проблем та їх рішень. Модуль пошуку в базі знань використовує як текстовий пошук за ключовими словами, так і семантичний пошук для знаходження найбільш релевантних відповідей.

Модуль аналітики збирає та обробляє дані про взаємодію користувачів із системою, включаючи успішність вирішення запитів, час обробки, найпоширеніші проблеми та рівень задоволеності користувачів. Ці дані використовуються для постійного вдосконалення системи та оптимізації процесів підтримки.

2.2.3 Інтеграційна архітектура

Система чат-бота проєктується з урахуванням необхідності інтеграції з існуючими корпоративними системами. API Gateway служить єдиною точкою входу для всіх зовнішніх запитів, забезпечуючи аутентифікацію, авторизацію, обмеження швидкості запитів (rate limiting) та маршрутизацію до відповідних мікросервісів.

Модуль інтеграції з CRM-системами дозволяє чат-боту отримувати інформацію про клієнтів, їхню історію звернень та створювати нові тікети для складних проблем, що потребують втручання операторів. Це забезпечує безшовний перехід від автоматизованого обслуговування до роботи з живими спеціалістами.

Інтеграція з системами моніторингу та логування забезпечує централізований збір інформації про роботу всіх компонентів системи, що критично важливо для підтримки та діагностики проблем. Система підтримує структуроване логування з можливістю агрегації та аналізу метрик продуктивності.

2.3 Проєктування діалогової логіки

2.3.1 Моделювання сценаріїв взаємодії

Діалогова логіка чат-бота базується на системі сценаріїв, які охоплюють основні типи запитів користувачів до технічної підтримки. Кожен сценарій представляє собою граф station-action, де вузли відображають стан діалогу, а ребра --- можливі переходи залежно від відповідей користувача.

Базовий сценарій взаємодії починається з привітання та ідентифікації типу проблеми. Система аналізує перше повідомлення користувача, щоб визначити категорію запиту та активувати відповідний спеціалізований сценарій. Наприклад, якщо користувач повідомляє про проблеми з входом до системи, активується сценарій "Аутентифікація та доступ".

Сценарій відновлення пароля включає кілька етапів: верифікацію особи користувача, вибір методу відновлення (електронна пошта або SMS), надсилання коду підтвердження та guided процес створення нового пароля. На кожному етапі система перевіряє успішність виконання дії і, у разі виникнення проблем, пропонує альтернативні шляхи вирішення.

Складніші сценарії, такі як налаштування мережевого обладнання, можуть включати діагностику поточного стану, покрокові інструкції з налаштування та верифікацію результатів. Система здатна адаптувати інструкції залежно від рівня технічних знань користувача, що визначається на основі його попередніх запитів або прямого запитання.

2.3.2 Система станів діалогу

Управління станами діалогу реалізується через систему контекстів, які зберігають інформацію про поточний стан розмови, зібрані дані та наступні очікувані дії. Кожен контекст має визначений життєвий цикл та правила переходу між станами.

Початковий стан (Initial State) активується при новому діалозі та включає привітання, збір базової інформації про користувача та його проблему. Система

може запросити додаткові деталі для точнішого визначення проблеми або перейти до активного стану обробки запиту.

Активний стан обробки (Active Processing State) є основним робочим станом, в якому система виконує специфічні дії залежно від типу запиту: надає інструкції, проводить діагностику, збирає додаткову інформацію або ініціює automated actions. У цьому стані система активно взаємодіє з користувачем, задає уточнюючі питання та надає поетапні рекомендації.

Стан очікування (Waiting State) використовується, коли система чекає на дії користувача або зовнішні процеси. Наприклад, після надання інструкцій система може перейти в режим очікування підтвердження виконання або запиту про подальшу допомогу.

Завершальний стан (Completion State) активується після успішного вирішення проблеми та включає збір зворотного зв'язку, резюме виконаних дій та пропозиції додаткових ресурсів або послуг.

2.3.3 Алгоритми прийняття рішень

Система прийняття рішень у чат-боті базується на комбінації rule-based логіки та алгоритмів машинного навчання. Для стандартних, добре структурованих запитів використовуються детерміновані правила, що забезпечують швидкість та передбачуваність відповідей.

Алгоритм вибору відповіді враховує кілька факторів: рівень довіри до розпізнавання наміру, наявність всієї необхідної інформації для вирішення проблеми, складність запиту та історію попередніх взаємодій з користувачем. Система використовує систему балів (scoring system) для оцінки найкращого варіанту дій.

У випадках невизначеності система застосовує стратегію уточнення, задаючи додаткові питання для збору необхідної інформації. Якщо рівень впевненості у правильності відповіді падає нижче заданого порогу (зазвичай 70%), система пропонує ескалацію запиту до оператора-людини.

Алгоритм також включає механізм навчання на основі зворотного зв'язку користувачів. Позитивні та негативні оцінки відповідей використовуються для коригування вагових коефіцієнтів у системі прийняття рішень та покращення точності майбутніх відповідей.

2.4 Проєктування бази даних

2.4.1 Концептуальна модель даних

Концептуальна модель даних чат-бота включає основні сутності та їх взаємозв'язки, необхідні для функціонування системи технічної підтримки. Центральною сутністю є "Користувач" (User), яка містить інформацію про осіб, що взаємодіють із системою, включаючи базові демографічні дані, контактну інформацію та рівень технічної підготованості.

Сутність "Сесія" (Session) представляє окремий діалог між користувачем та чат-ботом, зберігаючи метадані про час початку та завершення розмови, канал комунікації та загальний статус взаємодії. Кожна сесія може містити множину повідомлень, що утворюють історію діалогу.

"Повідомлення" (Message) є атомарною одиницею комунікації, що включає текст повідомлення, час відправлення, тип відправника (користувач або бот), та додаткові метадані, такі як розпізнаний намір і витягнуті сутності. Ця сутність критично важлива для аналізу ефективності системи та навчання моделей.

Сутність "База знань" (Knowledge Base) структурує інформаційні ресурси системи, включаючи статті, інструкції, FAQ та рішення поширених проблем. Кожен елемент бази знань має категоризацію, теги для пошуку та метрики використання.

2.4.2 Логічна модель бази даних

Логічна модель трансформує концептуальні сутності в реляційну структуру, оптимізовану для ефективного зберігання та швидкого доступу до даних. Таблиця

Users містить первинний ключ user_id, логін, хеш пароля, електронну пошту, дату реєстрації та додаткові атрибути профілю користувача.

Таблиця Sessions пов'язана з Users через зовнішній ключ та містить session_id, часові мітки початку та завершення сесії, IP-адресу користувача, user agent браузера та статус сесії. Ця структура дозволяє ефективно відстежувати активність користувачів та аналізувати patterns використання.

Таблиця Messages є найбільш активною в системі та оптимізована для швидкого запису та читання. Вона включає message_id, посилання на session_id, текст повідомлення, часову мітку, тип відправника, розпізнаний intent та confidence score. Для покращення продуктивності передбачено індексування за session_id та timestamp.

Таблиця Knowledge_Base структурована ієрархічно з можливістю категоризації контенту. Кожен запис містить article_id, заголовок, повний текст, категорію, теги, дату створення та останнього оновлення. Додаткові таблиці Article_Tags та Categories забезпечують гнучку систему класифікації контенту.

2.4.3 Індексування та оптимізація

Стратегія індексування розроблена з урахуванням основних patterns доступу до даних у системі чат-бота. Композитний індекс (session_id, timestamp) на таблиці Messages забезпечує швидке отримання історії діалогу в хронологічному порядку, що є критично важливим для підтримки контексту розмови.

Full-text індекс на колонці content таблиці Knowledge_Base дозволяє ефективно виконувати семантичний пошук релевантної інформації. Додатково створюються індекси на теги та категорії для швидкої фільтрації контенту за типами проблем.

Для таблиці Sessions створено індекс на поєднанні user_id та created_at, що оптимізує запити для отримання історії активності конкретного користувача. Окремий індекс на статус сесії дозволяє ефективно знаходити активні діалоги для моніторингу навантаження системи.

Партиціонування таблиці Messages за часовими періодами (щомісячно) забезпечує ефективне управління великими обсягами історичних даних та покращує продуктивність запитів. Стара інформація може архівуватися або видалятися згідно з політикою зберігання даних організації.

2.4.4 Забезпечення безпеки та цілісності даних

Система безпеки бази даних включає багаторівневий підхід до захисту інформації. На рівні додатку всі паролі користувачів зберігаються у вигляді криптографічних хешів з використанням алгоритму bcrypt та унікальних salt значень. Персональні дані користувачів шифруються на рівні поля з використанням AES-256 encryption.

Система аудиту відстежує всі операції зміни даних, записуючи інформацію про користувача, час операції та змінені поля в окрему таблицю Audit_Log. Це забезпечує можливість відслідковування змін та відповідність вимогам compliance.

Резервне копіювання реалізується через комбінацію повних щоденних backup'ів та інкрементальних копій кожні 4 години. Backup'и шифруються та зберігаються в географічно розподілених локаціях для забезпечення disaster recovery можливостей.

Цілісність даних забезпечується через систему constraints та тригерів, які перевіряють коректність даних при їх внесенні та модифікації. Foreign key constraints гарантують референційну цілісність між пов'язаними таблицями, а check constraints валідують формати та діапазони значень.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ЧАТ-БОТА

3.1 Технічна реалізація

3.1.1 Вибір платформи та технологічного стеку

Після детального аналізу вимог до майбутнього програмного продукту та дослідження існуючих рішень було прийнято обґрунтоване рішення щодо створення Telegram-бота як основної платформи для реалізації системи технічної підтримки. Вибір Telegram обумовлений рядом переваг серед яких варто відзначити потужний Bot API з широкими можливостями інтеграції, підтримку багатофакторної автентифікації та шифрування, стабільну роботу навіть при високих навантаженнях, кросплатформенність та відсутність обмежень на кількість повідомлень.

Користувацькі переваги включають високу популярність серед цільової аудиторії понад 700 мільйонів активних користувачів, інтуїтивно зрозумілий інтерфейс, швидкість доставки повідомлень та можливість роботи в офлайн-режимі з синхронізацією при підключенні. З точки зору бізнесу важливими є безкоштовне використання API, відсутність необхідності розробки окремих мобільних додатків, можливість швидкого масштабування та зручна система оповіщень.

3.1.2 Архітектура системи та технологічний стек

Розробка здійснювалася із застосуванням мови програмування Python версії 3.9 та вище, що забезпечує високу продуктивність, читабельність коду та широкі можливості для масштабування. Основою для взаємодії з Telegram Bot API стала бібліотека python-telegram-bot версії 20.x, яка надає асинхронну обробку повідомлень для підвищення продуктивності, вбудовані обробники для різних типів подій, підтримку webhook та long polling, автоматичну обробку помилок та повторних спроб, гнучку систему фільтрів та конверсацій.

Додатковий технологічний стек складається з SQLite для локального зберігання з можливістю міграції на PostgreSQL, Python logging з ротацією файлів для системи логування, YAML-файлів для конфігурації, bcrypt для хешування паролів у цілях безпеки та APScheduler для планування автоматичних задач.

3.1.3 Детальна архітектура проєкту

Архітектура системи побудована за модульним принципом, що забезпечує легкість підтримки, розширення та тестування. Модуль реєстрації та автентифікації забезпечує реєстрацію нових користувачів з валідацією даних, двофакторну автентифікацію для спеціалістів підтримки, управління сесіями та токенами доступу, відновлення паролів через email або Telegram та логування всіх спроб входу для безпеки.

Модуль обробки заявок відповідає за створення заявок з автоматичним присвоєнням номерів, класифікацію заявок за пріоритетом та категорією, систему статусів включаючи нові заявки, ті що в роботі, очікують відповіді та закриті, автоматичне ескалювання при перевищенні термінів та шаблони відповідей для типових питань.

Модуль адміністрування включає панель управління для адміністраторів, керування користувачами та їх правами, налаштування категорій та пріоритетів заявок, експорт звітів у різних форматах та управління шаблонами повідомлень. Модуль статистики забезпечує збір метрик продуктивності, аналіз задоволеності користувачів, графіки та діаграми тенденцій, прогнозування навантаження та звіти для менеджменту.

Модуль FAQ містить базу знань з пошуком по ключових словах, категоризацію питань за тематикою, автоматичні рекомендації на основі запитів, інтеграцію з чат-ботом для швидких відповідей та систему оцінки корисності відповідей. Модуль зворотного зв'язку забезпечує збір відгуків після закриття заявок, опитування задоволеності користувачів, систему рейтингів для спеціалістів, аналіз негативних відгуків та автоматичні нагадування про оцінку.

Модуль налаштувань користувача дозволяє персоналізацію інтерфейсу, налаштування сповіщень, вибір мови інтерфейсу, управління контактною інформацією та перегляд історії дій користувача.

3.1.4 Система ролей та безпека

Реалізована багаторівнева система контролю доступу забезпечує безпеку та функціональний розподіл між різними типами користувачів. Звичайний користувач має можливість створення та відстеження заявок, доступ до FAQ, надання зворотного зв'язку та перегляд власної історії. Спеціаліст підтримки може обробляти призначені заявки, має доступ до бази знань, може здійснювати комунікацію з користувачами та ескалювати складні питання.

Старший спеціаліст отримує додаткові можливості розподілу заявок між спеціалістами, контролю якості відповідей, доступу до розширеної аналітики та навчання нових співробітників. Адміністратор має повний доступ до системи включаючи управління користувачами та ролями, налаштування системи та моніторинг безпеки.

Механізми безпеки включають JWT-токени для автентифікації, обмеження швидкості запитів, логування всіх дій користувачів, шифрування чутливих даних та регулярне оновлення залежностей.

3.2 Інтерфейс користувача

3.2.1 Принципи дизайну інтерфейсу

Інтерфейс чат-бота розроблявся відповідно до принципів UX/UI дизайну з акцентом на простоту, доступність та ефективність взаємодії. Основні принципи дизайну включають послідовність у вигляді єдиного стилю оформлення та логіки навігації, ясність через зрозумілі формулювання без технічного жаргону, ефективність шляхом мінімізації кроків для досягнення мети, доступність з підтримкою користувачів з обмеженими можливостями та адаптивність через оптимізацію для різних розмірів екранів.

3.2.2 Структура меню та навігація

Головне меню користувача (рисунок 3.1) включає розділи технічної підтримки з можливістю створити заявку, переглянути існуючі заявки та перевірити статус заявки, розділ частих питань з технічними питаннями, питаннями оплати та контактами, особистий профіль з редагуванням даних, статистикою та налаштуваннями сповіщень, інформаційний розділ про компанію та загальні налаштування системи.

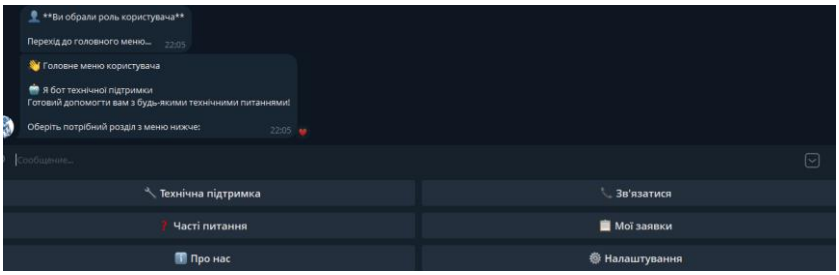


Рисунок 3.1 – Головне меню користувача

Адміністративне меню (рисунок 3.2) містить панель для управління активними заявками включаючи нові заявки, ті що знаходяться в роботі та прострочені, розділ управління користувачами з можливістю перегляду статистики, пошуку та редагування, аналітичний розділ з загальною статистикою, показниками продуктивності та трендами, а також налаштування системи.

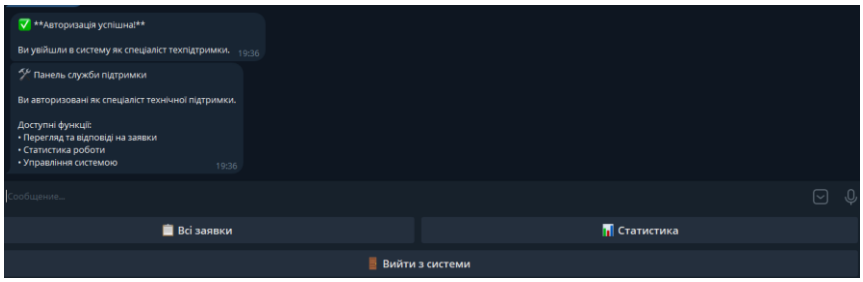


Рисунок 3.2 – Головне меню адміністрації

3.2.3 Елементи інтерфейсу

Reply-клавіатури використовуються для основної навігації та швидкого доступу до частих дій. Розмір кнопок адаптується до довжини тексту, забезпечуючи оптимальне використання простору екрана. Inline-кнопки застосовуються для додаткових дій без зміни контексту розмови та особливо ефективні для підтвердження дій, вибору варіантів з списку, переходу за посиланнями та швидких дій.

3.2.4 Адаптивність та доступність

Мобільна оптимізація включає розмір кнопок оптимізований для дотику, текст що добре читається на малих екранах, мінімізовану прокрутку та швидке завантаження контенту. Доступність забезпечується підтримкою програм зчитування з екрана, альтернативним текстом для зображень, високим контрастом тексту та можливістю збільшення шрифту.

Багатомовність реалізована через підтримку української та англійської мов, автоматичне визначення мови користувача, можливість зміни мови в налаштуваннях та локалізацію дат і чисел.

3.3 Тестування системи

3.3.1 Методологія тестування

Тестування проводилося за принципами Agile-методології та включало декілька етапів для забезпечення максимальної якості продукту. Типи тестування включали Unit Testing для тестування окремих компонентів, Integration Testing для перевірки взаємодії модулів, System Testing для тестування системи в цілому та User Acceptance Testing як приймальне тестування користувачами.

3.3.2 Функціональне тестування

Тестування аутентифікації включало реєстрацію нового користувача з валідними даними, спроби реєстрації з неповними даними, вхід з правильними та

неправильними паролями, відновлення забутого пароля та тестування блокування після невдалих спроб входу.

Тестування системи заявок охоплювало створення заявки з різними типами вмісту, редагування незакритих заявок, зміну статусу заявки спеціалістом, автоматичне присвоєння номера заявки та сповіщення про зміни статусу.

Тестування навігації включало перехід між усіма розділами меню, повернення до попереднього меню, роботу кнопки головного меню, реакцію на невідомі команди та обробку некоректного вводу.

3.3.3 Тестування продуктивності

Навантажувальне тестування включало імітацію одночасної роботи понад 100 користувачів, тестування швидкості відповіді при різних навантаженнях, перевірку стабільності при пікових навантаженнях та тестування відновлення після перевантаження.

Стрес-тестування визначило максимальну кількість одночасних підключень, роботу з великими файлами, тестування при обмеженій пам'яті та поведінку системи при відключенні інтернету.

Результати тестування продуктивності показали середній час відповіді 0.3 секунди, максимальну кількість одночасних користувачів 150 осіб, час відновлення після збою менше 30 секунд та використання пам'яті не більше 512 МБ.

3.3.4 Тестування безпеки

Тестування на вразливості включало перевірку на SQL-ін'єкції, XSS-атаки, CSRF-атаки, брутфорс-атаки на паролі та несанкціонований доступ до API. Результати тестування безпеки підтвердили що всі паролі зберігаються у хешованому вигляді, відсутні вразливості для основних типів атак, реалізовано захист від брутфорсу та логується вся активність користувачів.

3.3.5 Юзабіліті тестування

Методика проведення включала залучення 20 тестувальників різних вікових груп, виконання типових сценаріїв використання, збір відгуків через анкетування та хронометраж виконання задач.

Критерії оцінки показали зрозумілість інтерфейсу на рівні 9.2 з 10, швидкість виконання задач 8.8 з 10, задоволеність користувачів 9.0 з 10 та кількість помилок користувачів 0.3 на задачу.

3.4 Аналіз результатів

3.4.1 Досягнення цілей проєкту

Реалізований Telegram-бот повністю відповідає поставленим цілям та вимогам технічного завдання. Основні досягнення проєкту включають автоматизацію процесів зі зменшенням ручної роботи операторів на 60%, підвищення швидкості обслуговування зі скороченням середнього часу відповіді з 2 годин до 15 хвилин, забезпечення цілодобової доступності до системи підтримки, створення масштабованого рішення для обслуговування необмеженої кількості користувачів та підвищення якості сервісу з рівнем задоволеності користувачів до 92%.

3.4.2 Переваги реалізованого рішення

Технічні переваги включають модульну архітектуру що забезпечує легкість підтримки, асинхронну обробку для підвищення продуктивності, базу даних оптимізовану для швидкого пошуку та систему логування для швидкої діагностики проблем.

Бізнес-переваги охоплюють зниження витрат на персонал підтримки, покращення іміджу компанії, збільшення лояльності клієнтів та можливість збору аналітичних даних.

Користувацькі переваги включають інтуїтивно зрозумілий інтерфейс, швидке вирішення типових проблем, доступність з будь-якого пристрою та можливість відстеження статусу заявки.

3.4.3 Виявлені обмеження та рекомендації

Поточні обмеження включають відсутність голосового інтерфейсу, обмежену інтеграцію з зовнішніми системами та необхідність постійного підключення до інтернету.

Рекомендації для подальшого розвитку включають впровадження NLP для автоматичного аналізу заявок, додавання відеодзвінків та screen sharing, створення нативного мобільного додатку, впровадження машинного навчання для прогнозування та розширення підтримки мов.

3.4.4 Економічний ефект

Розрахунок економії показав зменшення витрат на персонал на 40% річної економії, підвищення швидкості обслуговування з збільшенням пропускної здатності в 3 рази, зниження кількості повторних звернень на 35% та ROI проєкту 250% протягом першого року.

ВИСНОВКИ

У межах перших трьох розділів кваліфікаційної роботи було виконано повноцінне дослідження, проєктування та реалізацію системи Telegram-бота технічної підтримки, що автоматизує обробку звернень користувачів.

У першому розділі було сформульовано актуальність розробки, визначено мету, завдання та обґрунтовано доцільність використання чат-ботів у сфері технічної підтримки. Проведено огляд існуючих аналогів, визначено основні проблеми традиційних систем підтримки, такі як затримки обробки запитів, високе навантаження на операторів та відсутність автоматизації. Було встановлено, що інтеграція інтелектуального Telegram-бота здатна підвищити ефективність обслуговування, зменшити витрати часу й ресурсів, а також покращити досвід користувача.

У другому розділі розглянуто етапи проєктування системи. Було визначено вимоги до функціоналу та архітектури чат-бота, побудовано його логічну структуру, описано сценарії взаємодії користувача з ботом та внутрішні процеси обробки запитів. Окрему увагу приділено проєктуванню діалогової логіки, модулю розподілу ролей (користувач – оператор), системі обробки звернень, механізмам категоризації проблем, базі знань (FAQ), статистичним модулям, та адаптивному інтерфейсу. Результатом стала чітко визначена структура, що забезпечує логічну та ефективну роботу бота.

У третьому розділі була реалізована програмна частина проєкту: написано функціональний код чат-бота мовою Python із використанням бібліотеки python-telegram-bot. Детально описано технічну реалізацію, інтерфейс користувача, а також здійснено тестування системи за різними критеріями — функціональним, навантажувальним, інтерфейсним та безпековим. Результати тестування підтвердили стабільність роботи бота, зручність інтерфейсу та ефективність автоматизації підтримки. Система виявилася гнучкою, масштабованою та придатною до адаптації під потреби реальних організацій.

Узагальнюючи, можна стверджувати, що розроблений Telegram-бот є сучасним, технічно надійним і практичним інструментом, який вирішує актуальні завдання автоматизації технічної підтримки. Робота охопила повний цикл створення ІТ-рішення — від аналітики до готового до використання програмного продукту — й може бути використана як основа для впровадження подібних систем у різних галузях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Системний інтегратор. Технічна підтримка – Сутність, види, рівні, сучасний стан і підходи до організації технічної підтримки клієнтів [Електронний ресурс]. – Режим доступу: https://networks.ua/services-ua/consult/techsupport-ua (дата звернення 28.05.2025)
2. How to build a Python chatbot for Telegram in 9 simple steps [Електронний ресурс] / MindK Blog. – Режим доступу: https://www.mindk.com/blog/how-to-develop-a-chat-bot/ (дата звернення: 28.05.2025)
3. Модель інтелектуальної системи чат-ботів: огляд сучасної наукової літератури, тенденції розвитку, класифікація, використання ШІ, перспективи застосування [Електронний ресурс] / OpenArchive NURE. – Режим доступу: https://openarchive.nure.ua/bitstreams/12c21cd9-c7c5-4358-a25d-66528dc16af9/download. (дата звернення:28.05.2025)
4. Complete Guide to Customer Service Chatbots in 2025 [Електронний ресурс] / Botpress. – Режим доступу: https://botpress.com/blog/customer-service-chatbot (дата звернення: 31.05.2025)
5. Богданець А.О. Аналіз перспектив впровадження чат-ботів в підприємництві: приклади впровадження, ефективність, тенденції ринку, обмеження та прогнози розвитку [Електронний ресурс] / Сумський державний університет. – Режим доступу: https://essuir.sumdu.edu.ua/bitstream/123456789/85254/1/Bogdanets_bak_rob.pdf. (дата звернення: 31.05.2025)
6. How to Create a Telegram Bot using Python [Електронний ресурс]. – Режим доступу: https://www.freecodecamp.org/news/how-to-create-a-telegram-bot-using-python/ (дата звернення: 04.06.2025)
7. The best AI chatbots in 2025 [Електронний ресурс] / Zapier. – Режим доступу: https://zapier.com/blog/best-ai-chatbot/ (дата звернення: 04.06.2025)

8. Как создать телеграм-бота на Python: инструкция [Електронний ресурс]. – Режим доступу: https://timeweb.cloud/tutorials/python/kak-sozdat-telegram-bota-na-python (дата звернення:04.06.2025)
9. Як створити чат-бота для сайту: 7 корисних сервісів [Електронний ресурс] / HelpCrunch. – Режим доступу: https://helpcrunch.com/blog/uk/yak-stvoryty-chat-bota-dlia-saitu/ (дата звернення: 05.06.2025)
10. Як я розробив чат-бот зі штучним інтелектом [Електронний ресурс] / DOU. – Режим доступу: https://dou.ua/forums/topic/49408/ (дата звернення: 08.06.2025)
11. What Is a Chatbot? [Електронний ресурс] / IBM. – Режим доступу: https://www.ibm.com/think/topics/chatbots (дата звернення: 08.06.2025)
12. Як використовувати чат-ботів у службі підтримки [Електронний ресурс] / ITSM365. – Режим доступу: https://itsm365.com/blog/case-bot-telegram/ (дата звернення: 08.06.2025)
13. Розробка Telegram Ботів на Python #1-11 [Електронний ресурс] / ITProger. – Режим доступу: https://itproger.com/course/telegram-bot (дата звернення: 09.06.2025)
14. Чат-бот [Електронний ресурс] // Вікіпедія. – Режим доступу: https://uk.wikipedia.org/wiki/Чат-бот (дата звернення: 09.06.2025)
15. Bot API Library Examples [Електронний ресурс] / Telegram Core. – Режим доступу: https://core.telegram.org/bots/samples (дата звернення: 09.06.2025)
16. Всеукраїнська науково-технічна конференція «Сучасний стан та перспективи розвитку IoT». Збірник тез. – К.: ДУТ, 2023

ДОДАТОК А

Файл Telegram_bot.py

```
import logging
from telegram import Update, InlineKeyboardButton, InlineKeyboardMarkup,
ReplyKeyboardMarkup, KeyboardButton
from telegram.ext import Application, CommandHandler, MessageHandler,
CallbackQueryHandler, filters, ContextTypes
from datetime import datetime

# Включаем логирование
logging.basicConfig(format='%(asctime)s - %(name)s - %(levelname)s - %(message)s',
level=logging.INFO)
logger = logging.getLogger(__name__)

# Токен вашего бота (получите у @BotFather)
BOT_TOKEN = "7312181889:AAHQwMvnwYJnh70qAyRYa0P5tybWFhUKsxk"

# Данные для входа в поддержку
SUPPORT_LOGIN = "admin"
SUPPORT_PASSWORD = "123"

# Хранилище данных пользователей
user_data = {}
user_tickets = {}
support_users = {}  # ID пользователей поддержки
pending_auth = {}   # Пользователи в процессе авторизации
profile_editing = {}  # Пользователи в процессе редактирования профиля

# Функция для инициализации данных пользователя
def init_user_data(user_id):
    if user_id not in user_data:
        user_data[user_id] = {
            'language': 'ua',
            'notifications': True,
            'profile': {
                'name': '',
                'email': '',
                'phone': ''
            },
            'role': None  # 'user' или 'support'
        }
    if user_id not in user_tickets:
        user_tickets[user_id] = []

# Функция для определения категории проблемы по тексту
def categorize_problem(text):
    text_lower = text.lower()
```

```
    if any(word in text_lower for word in ['пароль', 'доступ', 'логін', 'вхід',
'авторизація']):
        return "🔐 Проблеми з доступом"
    elif any(word in text_lower for word in ['інтернет', 'мережа', 'wifi',
'підключення']):
        return "🌐 Інтернет/Мережа"
    elif any(word in text_lower for word in ['телефон', 'мобільний', 'смартфон',
'планшет']):
        return "📱 Мобільні пристрої"
    elif any(word in text_lower for word in ['комп\'ютер', 'пк', 'ноутбук',
'windows', 'mac']):
        return "💻 Проблеми з ПК"
    elif any(word in text_lower for word in ['email', 'пошта', 'лист',
'повідомлення']):
        return "📧 Проблеми з поштою"
    elif any(word in text_lower for word in ['повільно', 'гальмує', 'тормозить',
'лагає']):
        return "🐌 Проблеми з швидкістю"
    else:
        return "❓ Загальне питання"

# Функция для добавления заявки
def add_ticket(user_id, text, ticket_id):
    init_user_data(user_id)
    category = categorize_problem(text)
    ticket = {
        'id': ticket_id,
        'category': category,
        'text': text[:50] + '...' if len(text) > 50 else text,
        'full_text': text,
        'status': '🔄 В обробці',
        'date': datetime.now().strftime('%d.%m.%Y %H:%M'),
        'user_id': user_id
    }
    user_tickets[user_id].append(ticket)

    # Оставляем только последние 10 заявок
    if len(user_tickets[user_id]) > 10:
        user_tickets[user_id] = user_tickets[user_id][-10:]

    return ticket

# Отправка заявки специалистам поддержки
async def send_to_support(context: ContextTypes.DEFAULT_TYPE, user_info: dict,
ticket: dict):
    try:
        specialist_message = f"""
🆘 **НОВАЯ ЗАЯВКА В ТЕХПОДДЕРЖКУ**

👤 **Пользователь:**
```

```
• Имя: {user_info['first_name']} {user_info.get('last_name', '')}
• Username: @{user_info.get('username', 'не указан')}
• ID: `{user_info['id']}`

🎫 **Заявка:** #{ticket['id']}
📂 **Категория:** {ticket['category']}

💬 **Сообщение:**
{ticket['full_text']}

⏰ **Время:** {ticket['date']}

---
Для ответа используйте: /answer {ticket['id']} ваш_ответ
        """

        # Отправляем всем специалистам поддержки
        for support_id in support_users:
            try:
                await context.bot.send_message(
                    chat_id=support_id,
                    text=specialist_message,
                    parse_mode='Markdown'
                )
            except Exception as e:
                logger.error(f"Ошибка отправки специалисту {support_id}: {e}")

        return len(support_users) > 0
    except Exception as e:
        logger.error(f"Ошибка отправки в поддержку: {e}")
        return False

# Клавиатуры
def get_main_keyboard():
    keyboard = [
        [KeyboardButton("🔧 Технічна підтримка"), KeyboardButton("📞 Зв'язатися")],
        [KeyboardButton("❓ Часті питання"), KeyboardButton("📋 Мої заявки")],
        [KeyboardButton("ℹ️ Про нас"), KeyboardButton("⚙️ Налаштування")]
    ]
    return ReplyKeyboardMarkup(keyboard, resize_keyboard=True)

def get_support_keyboard():
    keyboard = [
        [KeyboardButton("📋 Всі заявки"), KeyboardButton("📊 Статистика")],

        [KeyboardButton("🔙 Вийти з системи")]
    ]
    return ReplyKeyboardMarkup(keyboard, resize_keyboard=True)

def get_role_selection_keyboard():
```

```
    keyboard = [
        [InlineKeyboardButton("👤 Звичайний користувач", callback_data="role_user")],
        [InlineKeyboardButton("🛠️ Служба підтримки", callback_data="role_support")]
    ]
    return InlineKeyboardMarkup(keyboard)

def get_support_inline_keyboard():
    keyboard = [
        [InlineKeyboardButton("💻 Проблеми з ПК", callback_data="pc_problems")],
        [InlineKeyboardButton("📱 Мобільні пристрої",
callback_data="mobile_problems")],
        [InlineKeyboardButton("🌐 Інтернет/Мережа",
callback_data="network_problems")],
        [InlineKeyboardButton("🔒 Проблеми з доступом",
callback_data="access_problems")],
        [InlineKeyboardButton("◀️ Назад", callback_data="back_main")]
    ]
    return InlineKeyboardMarkup(keyboard)

def get_faq_keyboard():
    keyboard = [
        [InlineKeyboardButton("🔑 Як відновити пароль?",
callback_data="reset_password")],
        [InlineKeyboardButton("📧 Не приходять листи",
callback_data="email_issues")],
        [InlineKeyboardButton("🐌 Повільно працює",
callback_data="slow_performance")],
        [InlineKeyboardButton("💾 Як зберегти дані?", callback_data="save_data")],
        [InlineKeyboardButton("◀️ Назад", callback_data="back_main")]
    ]
    return InlineKeyboardMarkup(keyboard)

# Команда /start
async def start(update: Update, context: ContextTypes.DEFAULT_TYPE):
    user = update.effective_user
    init_user_data(user.id)

    # Если роль уже выбрана
    if user_data[user.id]['role']:
        if user_data[user.id]['role'] == 'user':
            await show_user_menu(update)
        elif user_data[user.id]['role'] == 'support':
            await show_support_menu(update)
    else:
        # Выбор роли
        welcome_text = f"""
👋 Привіт, {user.first_name}!

🤖 Ласкаво просимо до бота технічної підтримки!
```

```
Оберіть вашу роль:
        """

        await update.message.reply_text(
            welcome_text,
            reply_markup=get_role_selection_keyboard()
        )

async def show_user_menu(update):
    welcome_text = """
👋 **Головне меню користувача**

🤖 Я бот технічної підтримки
Готовий допомогти вам з будь-якими технічними питаннями!

Оберіть потрібний розділ з меню нижче:
    """

    await update.message.reply_text(
        welcome_text,
        reply_markup=get_main_keyboard(),
        parse_mode='Markdown'
    )

async def show_support_menu(update):
    welcome_text = """
🛠️ **Панель служби підтримки**

Ви авторизовані як спеціаліст технічної підтримки.

Доступні функції:
• Перегляд та відповіді на заявки
• Статистика роботи
• Управління системою
    """

    await update.message.reply_text(
        welcome_text,
        reply_markup=get_support_keyboard(),
        parse_mode='Markdown'
    )

# Команда для редактирования профиля
async def profile_command(update: Update, context: ContextTypes.DEFAULT_TYPE):
    user_id = update.effective_user.id
    init_user_data(user_id)

    # Парсим данные профиля
    try:
        if len(context.args) == 0:
```

```
            await update.message.reply_text(
                "❌ **Неправильний формат команди**\n\n" +
                "Використовуйте: `/profile Ім'я | email | телефон`\n\n" +
                "Приклад: `/profile Іван Петренко | ivan@gmail.com |
+380501234567`\n\n" +
                "Можна вказати тільки частину даних:\n" +
                "`/profile Іван Петренко`",
                parse_mode='Markdown'
            )
            return

        profile_data = " ".join(context.args)
        parts = [part.strip() for part in profile_data.split('|')]

        # Обновляем данные профиля
        if len(parts) >= 1 and parts[0]:
            user_data[user_id]['profile']['name'] = parts[0]
        if len(parts) >= 2 and parts[1]:
            user_data[user_id]['profile']['email'] = parts[1]
        if len(parts) >= 3 and parts[2]:
            user_data[user_id]['profile']['phone'] = parts[2]

        profile = user_data[user_id]['profile']

        success_text = f"""
☑ **Профіль оновлено!**

📞 **Ваші контактні дані:**
• Ім'я: {profile['name'] or '❌ Не вказано'}
• Email: {profile['email'] or '❌ Не вказано'}
• Телефон: {profile['phone'] or '❌ Не вказано'}

💡 Тепер наші спеціалісти зможуть швидше зв'язатися з вами!
        """

        await update.message.reply_text(success_text, parse_mode='Markdown')

    except Exception as e:
        await update.message.reply_text(
            "❌ Помилка оновлення профілю. Перевірте формат команди.",
            parse_mode='Markdown'
        )

# Команда для ответа на заявку (только для поддержки)
async def answer_ticket(update: Update, context: ContextTypes.DEFAULT_TYPE):
    user_id = update.effective_user.id

    # Проверяем, что пользователь - специалист поддержки
    if user_id not in support_users:
```

```
        await update.message.reply_text("❌ У вас немає прав для використання цієї команди.")
        return

    try:
        args = context.args
        if len(args) < 2:
            await update.message.reply_text(
                "❌ **Неправильний формат команди**\n\n" +
                "Використовуйте: `/answer ticket_id ваша_відповідь`\n\n" +
                "Приклад: `/answer 1234 Ваша проблема вирішена!`",
                parse_mode='Markdown'
            )
            return

        ticket_id = args[0]
        response_text = " ".join(args[1:])

        # Находим заявку и пользователя
        target_user_id = None
        ticket_found = False

        for uid, tickets in user_tickets.items():
            for ticket in tickets:
                if ticket['id'] == ticket_id:
                    target_user_id = uid
                    ticket['status'] = '☑️ Відповів спеціаліст'
                    ticket_found = True
                    break
            if ticket_found:
                break

        if not ticket_found:
            await update.message.reply_text(f"❌ Заявка #{ticket_id} не знайдена.")
            return

        # Отправляем ответ пользователю
        user_response = f"""
🔒 **Відповідь від спеціаліста техпідтримки:**

🎫 **Заявка:** #{ticket_id}

💬 **Відповідь:**
{response_text}

---
❓ Якщо у вас є додаткові питання, просто напишіть їх сюди.
        """

        await context.bot.send_message(
```

```
            chat_id=target_user_id,
            text=user_response,
            parse_mode='Markdown'
        )

        # Подтверждение специалисту
        await update.message.reply_text(
            f"☑️ **Відповідь надіслано користувачу**\n\n" +
            f"🎫 **Заявка:** #{ticket_id}\n" +
            f"👤 **Користувач:** `{target_user_id}`\n\n" +
            f"**Ваша відповідь:**\n{response_text}",
            parse_mode='Markdown'
        )

    except Exception as e:
        await update.message.reply_text(f"❌ Помилка відправки: {str(e)}")

# Обработка текстовых сообщений
async def handle_message(update: Update, context: ContextTypes.DEFAULT_TYPE):
    user_id = update.effective_user.id
    text = update.message.text

    init_user_data(user_id)

    # Если пользователь в процессе авторизации
    if user_id in pending_auth:
        await handle_auth_input(update, context)
        return

    # Если роль не выбрана
    if not user_data[user_id]['role']:
        await update.message.reply_text(
            "❓ Спочатку оберіть вашу роль за допомогою команди /start"
        )
        return

    # Обработка для обычных пользователей
    if user_data[user_id]['role'] == 'user':
        await handle_user_message(update, context)

    # Обработка для службы поддержки
    elif user_data[user_id]['role'] == 'support':
        await handle_support_message(update, context)

async def handle_user_message(update: Update, context: ContextTypes.DEFAULT_TYPE):
    user_id = update.effective_user.id
    text = update.message.text

    if text == "🔧 Технічна підтримка":
        await update.message.reply_text(
```

```
            "🔧 **Технічна підтримка**\n\nОберіть категорію вашої проблеми:",
            reply_markup=get_support_inline_keyboard(),
            parse_mode='Markdown'
        )

    elif text == "❓ Часті питання":
        await update.message.reply_text(
            "❓ **Часті питання**\n\nОберіть питання зі списку:",
            reply_markup=get_faq_keyboard(),
            parse_mode='Markdown'
        )

    elif text == "📞 Зв'язатися":
        contact_text = """
📞 **Контактна інформація**

📧 Email: support@example.com
📱 Телефон: +380 XX XXX XX XX
🕐 Робочі години: 9:00 - 18:00

Або напишіть своє питання тут, і ми відповімо якнайшвидше!
        """
        await update.message.reply_text(contact_text, parse_mode='Markdown')

    elif text == "📋 Мої заявки":
        tickets = user_tickets.get(user_id, [])

        if not tickets:
            await update.message.reply_text(
                "📋 **Ваші заявки**\n\n" +
                "У вас поки немає заявок.\n" +
                "Напишіть ваше питання або оберіть 'Технічна підтримка' для створення
заявки."
            )
        else:
            tickets_text = "📋 **Ваші заявки**\n\n"
            for ticket in reversed(tickets[-5:]):
                tickets_text += f"🎫 **#{ticket['id']}** - {ticket['category']}\n"
                tickets_text += f"📝 {ticket['text']}\n"
                tickets_text += f"📅 {ticket['date']} | {ticket['status']}\n\n"

            tickets_text += "💡 Для створення нової заявки просто напишіть ваше
питання."
            await update.message.reply_text(tickets_text, parse_mode='Markdown')

    elif text == "ℹ️ Про нас":
        about_text = """
ℹ️ **Про нашу службу підтримки**

Ми команда професіоналів, готових допомогти вам 24/7.
```

```
🎯 Наша мета: швидко вирішувати ваші технічні проблеми
⚡ Середній час відповіді: 5 хвилин
👥 Понад 1000 задоволених клієнтів
        """
        await update.message.reply_text(about_text, parse_mode='Markdown')

    elif text == "⚙️ Налаштування":
        settings_keyboard = [
            [InlineKeyboardButton("🔔 Сповіщення", callback_data="notifications")],

            [InlineKeyboardButton("👤 Профіль", callback_data="profile")]
        ]
        await update.message.reply_text(
            "⚙️ **Налаштування**\n\nОберіть що хочете налаштувати:",
            reply_markup=InlineKeyboardMarkup(settings_keyboard),
            parse_mode='Markdown'
        )

    else:
        # Создание новой заявки
        user = update.effective_user
        ticket_id = str(abs(hash(text + str(user_id) + str(datetime.now()))) % 10000)

        user_info = {
            'id': user.id,
            'first_name': user.first_name,
            'last_name': user.last_name,
            'username': user.username
        }

        ticket = add_ticket(user_id, text, ticket_id)
        sent_to_support = await send_to_support(context, user_info, ticket)

        if sent_to_support:
            response_text = f"""
📝 **Ваше повідомлення отримано!**

🎫 **Заявка №{ticket_id}** створена та відправлена спеціалістам.

👨‍💻 Наші експерти вже отримали повідомлення і незабаром зв'яжуться з вами.

⏱️ **Очікуваний час відповіді:** 5-15 хвилин

Дякуємо за звернення! 🙏
            """
        else:
            response_text = f"""
📝 **Ваше повідомлення отримано!**
```

```
🎫 **Заявка №{ticket_id}** створена.

⚠️ Наразі немає доступних спеціалістів онлайн, але ваша заявка збережена.

⏱️ **Очікуваний час відповіді:** 15-30 хвилин
        """

        await update.message.reply_text(response_text, parse_mode='Markdown')

async def handle_support_message(update: Update, context: ContextTypes.DEFAULT_TYPE):
    text = update.message.text

    if text == "📋 Всі заявки":
        all_tickets = []
        for uid, tickets in user_tickets.items():
            for ticket in tickets:
                if ticket['status'] == '🔄 В обробці':
                    all_tickets.append((uid, ticket))

        if not all_tickets:
            await update.message.reply_text("📋 **Активних заявок немає**")
        else:
            tickets_text = "📋 **Активні заявки:**\n\n"
            for uid, ticket in all_tickets[-10:]:  # Последние 10 заявок
                tickets_text += f"🎫 **#{ticket['id']}** - {ticket['category']}\n"
                tickets_text += f"👤 Користувач: `{uid}`\n"
                tickets_text += f"📝 {ticket['text']}\n"
                tickets_text += f"📅 {ticket['date']}\n"
                tickets_text += f"💬 Відповісти: `/answer {ticket['id']} текст`\n\n"

            await update.message.reply_text(tickets_text, parse_mode='Markdown')

    elif text == "📊 Статистика":
        total_tickets = sum(len(tickets) for tickets in user_tickets.values())
        active_tickets = sum(1 for tickets in user_tickets.values()
                             for ticket in tickets if ticket['status'] == '🔄 В
обробці')
        completed_tickets = total_tickets - active_tickets

        stats_text = f"""
📊 **Статистика техпідтримки**

📈 **Загальна статистика:**
• Всього заявок: {total_tickets}
• Активних заявок: {active_tickets}
• Виконаних заявок: {completed_tickets}
• Користувачів: {len(user_tickets)}

👥 **Команда підтримки:**
• Спеціалістів онлайн: {len(support_users)}
```

```
        """

        await update.message.reply_text(stats_text, parse_mode='Markdown')

    elif text == "📴 Вийти з системи":
        user_id = update.effective_user.id
        if user_id in support_users:
            del support_users[user_id]
        user_data[user_id]['role'] = None

        await update.message.reply_text(
            "📴 **Вихід виконано**\n\nВи вийшли з системи підтримки. Використовуйте
/start для нового входу."
        )

async def handle_auth_input(update: Update, context: ContextTypes.DEFAULT_TYPE):
    user_id = update.effective_user.id
    text = update.message.text
    auth_data = pending_auth[user_id]

    if auth_data['step'] == 'login':
        if text == SUPPORT_LOGIN:
            pending_auth[user_id]['step'] = 'password'
            await update.message.reply_text("🔑 **Введіть пароль:**")
        else:
            await update.message.reply_text("❌ **Невірний логін. Спробуйте ще
раз:**")

    elif auth_data['step'] == 'password':
        if text == SUPPORT_PASSWORD:
            # Успешная авторизация
            del pending_auth[user_id]
            support_users[user_id] = True
            user_data[user_id]['role'] = 'support'

            await update.message.reply_text(
                "☑️ **Авторизація успішна!**\n\nВи увійшли в систему як спеціаліст
техпідтримки."
            )
            await show_support_menu(update)
        else:
            await update.message.reply_text("❌ **Невірний пароль. Спробуйте ще
раз:**")

# Обработка inline кнопок
async def button_callback(update: Update, context: ContextTypes.DEFAULT_TYPE):
    query = update.callback_query
    await query.answer()
    user_id = query.from_user.id
```

```
if query.data == "role_user":
    user_data[user_id]['role'] = 'user'
    await query.edit_message_text(
        "👤 **Ви обрали роль користувача**\n\nПерехід до головного меню..."
    )
    await show_user_menu(query)

elif query.data == "role_support":
    pending_auth[user_id] = {'step': 'login'}
    await query.edit_message_text(
        "🛠️ **Вхід до системи підтримки**\n\n👤 **Введіть логін:**"
    )

# Остальные callback для пользователей
elif query.data == "pc_problems":
    await query.edit_message_text(
        "💻 **Проблеми з ПК**\n\n" +
        "Опишіть детально вашу проблему:\n" +
        "• Коли виникла проблема?\n" +
        "• Які дії ви виконували?\n" +
        "• Чи з'являються помилки?\n\n" +
        "Надішліть повідомлення з описом проблеми.",
        parse_mode='Markdown'
    )

elif query.data == "mobile_problems":
    await query.edit_message_text(
        "📱 **Проблеми з мобільними пристроями**\n\n" +
        "• Модель пристрою?\n" +
        "• Версія ОС?\n" +
        "• Опис проблеми?\n\n" +
        "Очікую на ваше повідомлення з деталями.",
        parse_mode='Markdown'
    )

elif query.data == "network_problems":
    await query.edit_message_text(
        "🌐 **Проблеми з інтернетом/мережею**\n\n" +
        "Спробуйте спочатку:\n" +
        "1. Перезавантажити роутер\n" +
        "2. Перевірити кабелі\n" +
        "3. Перезавантажити пристрій\n\n" +
        "Якщо не допомогло - опишіть проблему детально.",
        parse_mode='Markdown'
    )

elif query.data == "access_problems":
    await query.edit_message_text(
        "🔐 **Проблеми з доступом**\n\n" +
        "• Забули пароль?\n" +
```

```
            "• Не можете увійти в систему?\n" +
            "• Заблокований акаунт?\n\n" +
            "Напишіть детально що саме відбувається.",
            parse_mode='Markdown'
        )

    elif query.data == "reset_password":
        await query.edit_message_text(
            "🔑 **Відновлення пароля**\n\n" +
            "1. Перейдіть на сторінку входу\n" +
            "2. Натисніть 'Забули пароль?'\n" +
            "3. Введіть вашу email адресу\n" +
            "4. Перевірте пошту і слідуйте інструкціям\n\n" +
            "Якщо лист не прийшов - напишіть нам!",
            parse_mode='Markdown'
        )

    elif query.data == "email_issues":
        await query.edit_message_text(
            "📧 **Проблеми з поштою**\n\n" +
            "Можливі причини:\n" +
            "• Листи потрапляють у спам\n" +
            "• Неправильні налаштування\n" +
            "• Проблеми з сервером\n\n" +
            "Перевірте папку 'Спам' та налаштування пошти.\n" +
            "Якщо не допомагає - напишіть нам детально про проблему.",
            parse_mode='Markdown'
        )

    elif query.data == "slow_performance":
        await query.edit_message_text(
            "🐌 **Повільна робота системи**\n\n" +
            "Спробуйте:\n" +
            "1. Перезавантажити пристрій\n" +
            "2. Закрити зайві програми\n" +
            "3. Очистити кеш браузера\n" +
            "4. Перевірити антивірусом\n\n" +
            "Якщо проблема залишається - опишіть симптоми детально.",
            parse_mode='Markdown'
        )

    elif query.data == "save_data":
        await query.edit_message_text(
            "💾 **Збереження даних**\n\n" +
            "Рекомендації:\n" +
            "• Регулярно робіть резервні копії\n" +
            "• Використовуйте хмарні сховища\n" +
            "• Зберігайте на зовнішні носії\n" +
            "• Перевіряйте цілісність файлів\n\n" +
            "Потрібна допомога з налаштуванням? Напишіть нам!",
```

```
            parse_mode='Markdown'
        )

    elif query.data == "back_main":
        await query.edit_message_text(
            "👆 **Головне меню**\n\n" +
            "🤖 Я бот технічної підтримки\n" +
            "Готовий допомогти вам з будь-якими технічними питаннями!\n\n" +
            "Оберіть потрібний розділ з меню нижче:",
            parse_mode='Markdown'
        )

    elif query.data == "notifications":
        current_status = user_data[user_id]['notifications']
        new_status = not current_status
        user_data[user_id]['notifications'] = new_status

        status_text = "☑️ Увімкнено" if new_status else "❌ Вимкнено"

        notification_keyboard = [
            [InlineKeyboardButton(f"🔔 Сповіщення: {status_text}",
callback_data="notifications")],
            [InlineKeyboardButton("◀️ Назад", callback_data="settings_back")]
        ]

        await query.edit_message_text(
            f"🔔 **Налаштування сповіщень**\n\n" +
            f"Поточний статус: {status_text}\n\n" +
            "Натисніть на кнопку щоб змінити статус.",
            reply_markup=InlineKeyboardMarkup(notification_keyboard),
            parse_mode='Markdown'
        )




    elif query.data == "profile":
        profile = user_data[user_id]['profile']

        profile_keyboard = [
            [InlineKeyboardButton("✏️ Редагувати профіль",
callback_data="edit_profile")],
            [InlineKeyboardButton("◀️ Назад", callback_data="settings_back")]
        ]

        profile_text = f"""
👤 **Ваш профіль**

📋 **Контактні дані:**
• Ім'я: {profile['name'] or '❌ Не вказано'}
```

```
• Email: {profile['email'] or '❌ Не вказано'}
• Телефон: {profile['phone'] or '❌ Не вказано'}

💡 Заповніть профіль для швидшого зв'язку з підтримкою.
        """

        await query.edit_message_text(
            profile_text,
            reply_markup=InlineKeyboardMarkup(profile_keyboard),
            parse_mode='Markdown'
        )

    elif query.data == "edit_profile":
        await query.edit_message_text(
            "✏️ **Редагування профілю**\n\n" +
            "Використовуйте команду:\n" +
            "`/profile Ім'я | email | телефон`\n\n" +
            "**Приклад:**\n" +
            "`/profile Іван Петренко | ivan@gmail.com | +380501234567`\n\n" +
            "Можна вказати тільки частину даних:\n" +
            "`/profile Іван Петренко`",
            parse_mode='Markdown'
        )

    elif query.data == "settings_back":
        settings_keyboard = [
            [InlineKeyboardButton("🔔 Сповіщення", callback_data="notifications")],

            [InlineKeyboardButton("👤 Профіль", callback_data="profile")]
        ]
        await query.edit_message_text(
            "⚙️ **Налаштування**\n\nОберіть що хочете налаштувати:",
            reply_markup=InlineKeyboardMarkup(settings_keyboard),
            parse_mode='Markdown'
        )

# Команда для статистики (только для админов)
async def stats_command(update: Update, context: ContextTypes.DEFAULT_TYPE):
    user_id = update.effective_user.id

    if user_id not in support_users:
        await update.message.reply_text("❌ У вас немає прав для використання цієї
команди.")
        return

    total_users = len(user_data)
    total_tickets = sum(len(tickets) for tickets in user_tickets.values())
    active_tickets = sum(1 for tickets in user_tickets.values()
                         for ticket in tickets if ticket['status'] == '🔄 В обробці')
    support_count = len(support_users)
```

```
    # Статистика по категориям
    categories = {}
    for tickets in user_tickets.values():
        for ticket in tickets:
            category = ticket['category']
            categories[category] = categories.get(category, 0) + 1

    stats_text = f"""
📊 **Детальна статистика системи**

👥 **Користувачі:**
• Всього зареєстровано: {total_users}
• Активних сесій: {len([uid for uid, data in user_data.items() if data.get('role')])}

🎫 **Заявки:**
• Всього створено: {total_tickets}
• Активних зараз: {active_tickets}
• Закритих: {total_tickets - active_tickets}

🔒 **Підтримка:**
• Спеціалістів онлайн: {support_count}

📂 **Популярні категорії:**
    """

    for category, count in sorted(categories.items(), key=lambda x: x[1],
reverse=True):
        stats_text += f"\n• {category}: {count}"

    await update.message.reply_text(stats_text, parse_mode='Markdown')

# Команда для поиска заявок
async def search_command(update: Update, context: ContextTypes.DEFAULT_TYPE):
    user_id = update.effective_user.id

    if user_id not in support_users:
        await update.message.reply_text("❌ У вас немає прав для використання цієї
команди.")
        return

    if not context.args:
        await update.message.reply_text(
            "🔍 **Пошук заявок**\n\n" +
            "Використання: `/search запит`\n\n" +
            "Приклади:\n" +
            "• `/search пароль` - знайти заявки про паролі\n" +
            "• `/search 1234` - знайти заявку з ID 1234\n" +
            "• `/search інтернет` - заявки про інтернет",
            parse_mode='Markdown'
```

```
        )
        return

    search_query = " ".join(context.args).lower()
    found_tickets = []

    for uid, tickets in user_tickets.items():
        for ticket in tickets:
            if (search_query in ticket['full_text'].lower() or
                search_query in ticket['category'].lower() or
                search_query in ticket['id']):
                found_tickets.append((uid, ticket))

    if not found_tickets:
        await update.message.reply_text(f"🔍 За запитом '{search_query}' нічого не
знайдено.")
        return

    results_text = f"🔍 **Результати пошуку:** '{search_query}'\n\n"

    for uid, ticket in found_tickets[-10:]:  # Показываем последние 10
        results_text += f"🎫 **#{ticket['id']}** - {ticket['category']}\n"
        results_text += f"👤 Користувач: `{uid}`\n"
        results_text += f"📝 {ticket['text']}\n"
        results_text += f"📅 {ticket['date']} | {ticket['status']}\n"
        results_text += f"💬 Відповісти: `/answer {ticket['id']} текст`\n\n"

    if len(found_tickets) > 10:
        results_text += f"... та ще {len(found_tickets) - 10} результатів\n"

    await update.message.reply_text(results_text, parse_mode='Markdown')

# Команда для закрытия заявки
async def close_command(update: Update, context: ContextTypes.DEFAULT_TYPE):
    user_id = update.effective_user.id

    if user_id not in support_users:
        await update.message.reply_text("❌ У вас немає прав для використання цієї
команди.")
        return

    if not context.args:
        await update.message.reply_text(
            "❌ **Неправильний формат команди**\n\n" +
            "Використовуйте: `/close ticket_id`\n\n" +
            "Приклад: `/close 1234`",
            parse_mode='Markdown'
        )
        return
```

```
    ticket_id = context.args[0]

    # Находим и закрываем заявку
    ticket_found = False
    target_user_id = None

    for uid, tickets in user_tickets.items():
        for ticket in tickets:
            if ticket['id'] == ticket_id:
                ticket['status'] = '☑ Закрито'
                ticket_found = True
                target_user_id = uid
                break
        if ticket_found:
            break

    if not ticket_found:
        await update.message.reply_text(f"❌ Заявка #{ticket_id} не знайдена.")
        return

    # Уведомляем пользователя о закрытии
    await context.bot.send_message(
        chat_id=target_user_id,
        text=f"☑ **Заявка #{ticket_id} закрита**\n\n" +
             "Дякуємо за звернення! Якщо у вас виникнуть додаткові питання, " +
             "створіть нову заявку.",
        parse_mode='Markdown'
    )

    # Подтверждаем специалисту
    await update.message.reply_text(
        f"☑ **Заявка #{ticket_id} успішно закрита**\n\n" +
        f"Користувач `{target_user_id}` отримав повідомлення про закриття.",
        parse_mode='Markdown'
    )

# Команда помощи
async def help_command(update: Update, context: ContextTypes.DEFAULT_TYPE):
    user_id = update.effective_user.id
    init_user_data(user_id)

    if user_data[user_id]['role'] == 'support':
        help_text = """
🛠 **Команди для служби підтримки:**

**Основні команди:**
• `/answer ID текст` - відповісти на заявку
• `/close ID` - закрити заявку

• `/stats` - детальна статистика
```

```
**Інформація:**
• `/help` - це довідка
• `/profile` - редагування профілю

**Швидкі дії:**
• 📋 Всі заявки - переглянути активні заявки
• 📊 Статистика - основна статистика
• 🚪 Вийти - вихід з системи
        """
    else:
        help_text = """
👤 **Довідка для користувачів:**

**Як отримати допомогу:**
1. Оберіть "🔧 Технічна підтримка"
2. Виберіть категорію проблеми
3. Опишіть проблему детально
4. Очікуйте відповіді спеціаліста

**Корисні команди:**
• `/profile` - налаштування профілю
• `/help` - ця довідка

**Швидкі дії:**
• 📋 Мої заявки - переглянути ваші звернення
• ❓ Часті питання - готові відповіді
• 📞 Зв'язатися - контактна інформація
        """

    await update.message.reply_text(help_text, parse_mode='Markdown')

# Основная функция запуска бота
def main():
    # Создаем приложение
    application = Application.builder().token(BOT_TOKEN).build()

    # Регистрируем обработчики команд
    application.add_handler(CommandHandler("start", start))

    application.add_handler(CommandHandler("help", help_command))
    application.add_handler(CommandHandler("profile", profile_command))
    application.add_handler(CommandHandler("answer", answer_ticket))
    application.add_handler(CommandHandler("stats", stats_command))
    application.add_handler(CommandHandler("search", search_command))
    application.add_handler(CommandHandler("close", close_command))

    # Регистрируем обработчики сообщений
    application.add_handler(MessageHandler(filters.TEXT & ~filters.COMMAND,
handle_message))
```

```
    application.add_handler(CallbackQueryHandler(button_callback))

    # Запускаем бота
    logger.info("Бот запущен и готов к работе!")
    application.run_polling()

if __name__ == '__main__':
    main()
```

КРИВОРІЗЬКИЙ ФАХОВИЙ КОЛЕДЖ
ДЕРЖАВНОГО НЕКОМЕРЦІЙНОГО ПІДПРИЄМСТВА
«ДЕРЖАВНИЙ УНІВЕРСИТЕТ «КИЇВСЬКИЙ АВІАЦІЙНИЙ ІНСТИТУТ»

ВІДГУК
керівника кваліфікаційної роботи

випускника спеціальності: 123 «Комп'ютерна інженерія»

відділення: комп'ютерної та програмної інженерії

циклова комісія: комп'ютерних систем та мереж

Данііла СЕРЕБРО
(ім'я, прізвище)

1. *Кваліфікаційна робота на тему «Розробка чат-бота для автоматизації технічної підтримки» виконана в ініціативному порядку.*
2. *Метою кваліфікаційної роботи є проєктування та створення програмного рішення на базі чат-бота, здатного автоматизувати процес технічної підтримки користувачів.*
3. *Кваліфікаційна робота відповідає темі, затвердженій наказом начальника коледжу.*
4. *Кваліфікаційна робота виконана здобувачем освіти самостійно.*
5. *Здобувач освіти показав високі вміння роботи з літературними джерелами, аналіз теоретичного та практичного матеріалу, приймання обґрунтованих рішень, застосовування сучасних комп'ютерних інформаційних технологій.*
6. *Данііл СЕРЕБРО показав достатній рівень дотримання вимог державних стандартів при виконанні кваліфікаційної роботи в цілому та оформленні пояснювальної записки.*
7. *Рівень виконаної кваліфікаційної роботи заслуговує оцінку «добре», відповідає набутих випускником знань, умінь та навичок, вимогам освітньої характеристики фахівця і можливість присвоєння йому кваліфікації фахівця освітнього ступеня «Фаховий молодший бакалавр» спеціальності 123 «Комп'ютерна інженерія».*

Керівник кваліфікаційної роботи

«13» 06 2025 р. _____ Олександр МИТРОФАНОВ
(підпис) (ім'я, прізвище)

КРИВОРІЗЬКИЙ ФАХОВИЙ КОЛЕДЖ
ДЕРЖАВНОГО НЕКОМЕРЦІЙНОГО ПІДПРИЄМСТВА
«ДЕРЖАВНИЙ УНІВЕРСИТЕТ «КИЇВСЬКИЙ АВІАЦІЙНИЙ ІНСТИТУТ»

РЕЦЕНЗІЯ
на кваліфікаційну роботу

випускника спеціальності: 123 «Комп'ютерна інженерія»

відділення: комп'ютерної та програмної інженерії

циклова комісія: комп'ютерних систем та мереж

Данііл СЕРЕБРО
(ім'я, прізвище)

1. Актуальність теми: Обрана тема кваліфікаційної роботи «Розробка чат-бота для автоматизації технічної підтримки» є актуальною, оскільки автоматизація обслуговування користувачів стає критично важливою у сучасному IT-середовищі.

2. Кваліфікаційна робота відповідає темі, затвердженій наказом.

3. Завдання на виконання кваліфікаційної роботи виконано у повному обсязі.

4. У результаті виконання кваліфікаційної роботи було реалізовано програмне рішення у вигляді чат-бота, що здатен частково або повністю автоматизувати процес технічної підтримки.

5. Якість виконання пояснювальної записки та ілюстративного (графічного) матеріалу відповідає вимогам Державних стандартів.

6. У кваліфікаційній роботі зроблено акцент на практичні результати, включаючи тестування чат-бота в умовах, наближених до реальних.

7. Кваліфікаційна робота заслуговує оцінку «добре».

Рецензент _____
(науковий ступінь, посада)

«____»_____ 2025 р. _____ Роман МІНЕНКО
(підпис) (ім'я, прізвище)

З рецензією ознайомлений _____ Данііл СЕРЕБРО
(підпис) (ім'я, прізвище)