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

Навчально-науковий
інститут електроенергетики
(навчально-науковий інститут)
Факультет інформаційних технологій
(факультет)
Кафедра інформаційних технологій та комп'ютерної інженерії
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня магістра

Здобувача вищої освіти Тимченка Богдана Миколайовича
(ПІБ)
академічної групи 123м-23-1
(шифр)
спеціальності 123 Комп'ютерна інженерія
(код і назва спеціальності)
за освітньо-професійною програмою «Комп'ютерна інженерія»
(офіційна назва)

на тему «Обґрунтування структури комп'ютерної системи компанії «Avaris» з використанням телеграм-бота для консультацій»
(назва за наказом ректора)

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	доц. Бешта Д.О.			
розділів:				
синтез системи	доц. Бешта Д.О.			
розроблення програмного забезпечення	ас. Панферова Я.В.			

Рецензент				

Нормоконтролер	проф. Цвіркун Л.І.			

Дніпро
2024

ЗАТВЕРДЖЕНО:
завідувач кафедри
інформаційних технологій
та комп'ютерної інженерії
(повна назва)
_____ _В.В. Гнатушенко
(підпис) (ініціали, прізвище)
«_____» _____ 2024 року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня магістра
(бакалавра, магістра)

здобувача вищої освіти _____Тимченко Б.М.____ **академічної групи** ____123м-23-1____
(прізвище та ініціали) (шифр)
спеціальності _____123 Комп'ютерна інженерія_____
за освітньою-професійною програмою _____«Комп'ютерна інженерія»_____
(офіційна назва)
на тему «Обґрунтування структури комп'ютерної системи компанії «Avaris» з використанням телеграм-бота для консультацій»,
затверджену наказом ректора НТУ «Дніпровська політехніка» від 17 жовтня 2024 р. №1388-с

Розділ	**Зміст**	**Термін виконання**
Стан питання та постановка завдання	На основі матеріалів виробничих практик, інших науково-технічних джерел сформулювати наукове завдання, конкретизується предмет та мета досліджень	16.10.2024
Теоретичний	Обґрунтувати теоретичну базу розв'язання наукового завдання, якому присвячено роботу	30.10.2024
Синтез системи	Розробка комп'ютерної системи	08.11.2024
Розроблення програмного забезпечення	Розробка програмного забезпечення	25.11.2024
Експериментальний розділ	Проведення і обробка результатів експериментів	06.12.2024

Завдання видано _____ доц. Бешта Д.О.
(підпис керівника) (ініціали, прізвище)
Дата видачі 06 вересня 2024 р.
Дата подання до екзаменаційної комісії 10.12.2024 р.
Прийнято до виконання _____ Тимченко Б.М.
(підпис здобувача вищої освіти) (ініціали, прізвище)

РЕФЕРАТ

Пояснювальна записка: 84 с., 1 таблиця, 48 рисунків, 1 додаток, 11 джерел.

ЮРИДИЧНА КОМПАНІЯ, БОТ, TELEGRAM, MySQL, PYTHON, ПІДТРИМКА, ЗВОРОТНІЙ ЗВ’ЯЗОК

Об'єкт дослідження: платформа для автоматизації юридичних консультацій юридичної компанії «Avaris».

Мета роботи: автоматизація консультування клієнтів юридичної компанії та забезпечення для них зворотного зв’язку за допомогою Telegram-бота.

Методи дослідження: використання мови Python, бібліотек PyTelegramBotAPI, pymysql, реляційної бази даних MySQL для створення боту з можливостями зворотного зв’язку, розробка та тестування ПЗ.

Дана пояснювальна записка аргументує використання обраних інструментів та бібліотек, обґрунтовує архітектуру створеного ПЗ, описує обрані засоби автоматизації надання консультацій клієнтам юридичної компанії.

У теоретичному розділі було проведено аналіз сучасних платформ і технологій для створення чат-ботів, зокрема Telegram Bot API, а також інструментів інтеграції з базами даних і веб-сервісами. В розділі синтезу системи викладено загальну концепцію архітектури системи, визначено основні компоненти Telegram-бота та їх взаємодію із серверними та клієнтськими частинами.

В розділі розробки ПЗ представлено розробку програмного забезпечення, включаючи Telegram-бота та базу даних. Експериментальний розділ описує тестування системи, перевірку її працездатності.

ЗМІСТ

1 СТАН ПИТАННЯ І ПОСТАНОВКА ЗАВДАННЯ

1.1 Стисла характеристика галузі

Юридична сфера охоплює систему норм і правил, які регулюють права й обов'язки людей і організацій, а також визначають механізми їх вирішення. Вона включає такі напрямки, як цивільне, кримінальне, адміністративне, комерційне право тощо.

Ця галузь відіграє ключову роль у забезпеченні порядку та стабільності в суспільстві. Вона регулює відносини між громадянами й організаціями, захищає права та свободи, а також створює умови для розвитку бізнесу й економіки, визначаючи правила взаємодії між компаніями та їх клієнтами.

Основу функціонування юридичної сфери складають такі принципи, як законність, верховенство права та правова відповідальність. Принцип законності зобов’язує всіх учасників суспільних відносин дотримуватися встановлених правил, а правова держава гарантує рівність перед законом і захист основних прав громадян.

Юридична сфера постійно змінюється та адаптується до викликів сучасного світу, враховуючи соціальні та економічні трансформації. Вона охоплює низку професій, серед яких юристи, адвокати, нотаріуси, судді, прокурори та інші фахівці, які забезпечують функціонування правової системи.

Робота в цій сфері супроводжується великим обсягом документів, які потребують зберігання, обробки та обміну, а також роботою з конфіденційною інформацією. Використання інформаційних технологій, зокрема комп’ютерних мереж і спеціалізованого програмного забезпечення, значно полегшує ці процеси, забезпечуючи швидкий і захищений доступ до даних. Це дозволяє підвищити ефективність роботи юридичних компаній і якість їхніх послуг.

В Україні зараз триває реформа правової системи, спрямована на вдосконалення законодавства, покращення судових процедур та забезпечення більш дієвого захисту прав громадян. Важливим аспектом цього процесу є

інтеграція сучасних технологій, які сприяють автоматизації роботи юридичних установ і підвищують їхню продуктивність.

1.2 Характеристика і структура об’єкта впровадження

Компанія «Avaris Law Group» відома як провідний український постачальник юридичних послуг, який забезпечує комплексну правову підтримку для бізнесу та приватних клієнтів.

Компанія була заснована у 2005 році і за час своєї діяльності здобула значний досвід роботи з широким колом клієнтів. Серед них — українські та міжнародні компанії, державні установи, неурядові організації, а також фізичні особи.

Основні напрями діяльності компанії охоплюють такі сфери права:

- корпоративне право та структурування бізнесу: команда юристів «Avaris» надає консультації та підтримку у створенні, реєстрації та управлінні корпоративними структурами. Крім того, компанія допомагає вирішувати питання корпоративного управління, організації торговельної діяльності, реструктуризації та ліквідації компаній;
- податкове право: спеціалісти компанії займаються питаннями податкового планування, аналізом та мінімізацією податкових ризиків, а також співпрацею з податковими органами для забезпечення законності дій клієнтів;
- трудове право: юридична підтримка охоплює питання укладання трудових договорів, урегулювання трудових конфліктів, консультації з питань зайнятості, заробітної плати, а також соціального забезпечення;
- міжнародне право: компанія надає професійну допомогу у сфері зовнішньоекономічної діяльності, міжнародної торгівлі, захисту прав інтелектуальної власності та юридичного супроводу міжнародних операцій;

- судові та арбітражні справи: юристи «Avaris» забезпечують професійне представництво інтересів клієнтів у судових інстанціях та арбітражних процесах. Компанія має великий досвід вирішення корпоративних, трудових, податкових, земельних та інших спорів;
- право нерухомості: компанія консультує клієнтів щодо операцій із нерухомістю, таких як купівля, продаж, оренда, а також супроводжує реєстрацію прав власності;
- банківське та фінансове право: послуги включають консультації з питань кредитування, інвестування, лізингу та інших фінансових операцій.

«Avaris» користується репутацією компанії, яка дотримується найвищих професійних стандартів та суворо захищає конфіденційність своїх клієнтів.

Окрім надання юридичних послуг, компанія активно займається благодійною діяльністю. «Avaris» співпрацює з благодійними фондами та організаціями, надає безкоштовну юридичну допомогу, а також підтримує різні соціальні та культурні ініціативи. Наприклад, у співпраці з благодійним фондом «Країна Мрій» компанія допомагала талановитій молоді з малозабезпечених сімей отримати доступ до вищої освіти.

Компанія має офіси в таких містах України, як Київ, Одеса, Харків, Дніпро, Миколаїв, Суми, Запоріжжя та Львів. Використовуючи сучасні технології та онлайн-інструменти, «Avaris» забезпечує можливість співпрацювати з клієнтами як на національному, так і міжнародному рівні.

Загалом, «Avaris» є однією з провідних юридичних компаній в Україні, яка не лише забезпечує професійну підтримку в різних галузях права, а й орієнтується на індивідуальні потреби кожного клієнта. Компанія постійно вдосконалюється та впроваджує інноваційні підходи, щоб гарантувати найкращий результат для своїх клієнтів.

1.3 Вивчення стану і можливостей подальшого розвитку автоматизації окремих складових об'єкта й об'єкта в цілому

1.3.1 Наявні принципи автоматизації робочих процесів

Юридична компанія "Avaris" впровадила кілька систем для автоматизації ключових робочих процесів, які суттєво полегшують управління документообігом, справами, бухгалтерським обліком, а також підвищують ефективність комунікацій між співробітниками.

Управління документами: компанія вже використовує систему електронного документообігу, що є важливим кроком у напрямку автоматизації. Це забезпечує зручний доступ до юридичних документів, їх зберігання та обмін. Проте подальший розвиток може включати інтеграцію з автоматизованими системами аналітики, що дозволить автоматично класифікувати документи та надавати рекомендації на основі аналізу.

Керування справами: спеціалізована CRM-система для управління справами полегшує процеси відстеження етапів роботи над справами, нагадує про важливі терміни та автоматично генерує звіти. Однак, можливості CRM можна розширити шляхом впровадження штучного інтелекту для прогнозування результатів справ або автоматичного генерування шаблонів юридичних документів.

Бухгалтерський облік сучасна бухгалтерська програма автоматизує основні операції, розрахунок зарплати та податків. Розвиток може включати більш тісну інтеграцію з CRM-системами для кращої звітності та автоматичного аналізу фінансових ризиків, а також впровадження автоматизованих систем оплати та моніторингу платежів клієнтів.

Юридичний аналіз: використання спеціалізованих програм для аналізу документів є важливим кроком в автоматизації юридичних процесів. У майбутньому можна інтегрувати системи штучного інтелекту для автоматичного визначення прецедентів і рекомендацій на основі судових рішень, що ще більше скоротить час аналізу.

Комунікації: наразі компанія використовує корпоративну пошту, системи відеоконференцій та інструменти для спільної роботи. Однак розвиток може включати створення єдиної платформи для всіх видів комунікацій, що дозволить централізовано керувати повідомленнями, обговореннями та відеоконференціями з можливістю інтеграції з CRM-системою.

1.3.2 Недоліки та можливі шляхи їх усунення

Один з головних недоліків полягає в тому, що більшість комунікації з клієнтами здійснюється через телефон та електронну пошту.

Можливий розвиток включає впровадження чат-ботів для автоматизованого відповіді на часті запити клієнтів, що значно скоротить час обробки стандартних запитів. Крім того, можна впровадити єдину клієнтську платформу, яка дозволить клієнтам бачити стан своїх справ у режимі реального часу та комунікувати з адвокатами через безпечний канал зв'язку.

Також великим недоліком є відсутність самообслуговування для клієнтів.

Впровадження порталу самообслуговування для клієнтів дозволить їм переглядати статус справ, документи, здійснювати платежі та контактувати з адвокатами без потреби безпосереднього контакту. Це значно підвищить зручність роботи для клієнтів і зменшить навантаження на співробітників компанії.

1.3.3 Можливості подальшого розвитку автоматизації

Впровадження Telegram-бота для юридичної компанії "Avaris" відкриває нові можливості для автоматизації комунікацій та надання послуг клієнтам. Це сучасний, ефективний інструмент, який може значно підвищити рівень зручності та швидкості взаємодії з клієнтами, оптимізувати внутрішні процеси та покращити обслуговування.

Нижче наведено перспективи автоматизації.

Швидкий доступ до інформації: Telegram-бот може надати клієнтам швидкий доступ до основної інформації щодо їхніх справ. Клієнти зможуть

дізнатися статус своєї справи, перевірити дати судових засідань, отримати посилання на необхідні документи або інші матеріали безпосередньо через бот.

Відповіді на часті запитання (FAQ): бот може автоматично відповідати на типові запитання клієнтів, такі як процедура звернення до адвоката, вартість послуг, перелік необхідних документів або етапи судового процесу. Це значно зменшить кількість ручних відповідей юристів або адміністративного персоналу на стандартні запити.

Автоматизовані сповіщення: Telegram-бот може відправляти автоматичні сповіщення про важливі події у справі клієнта: нагадування про судові засідання, необхідність подачі документів або сплату рахунків. Такі сповіщення допоможуть клієнтам бути в курсі всіх важливих подій без постійних дзвінків або електронних листів.

Цілодобова підтримка: впровадження Telegram-бота дозволить клієнтам отримувати відповіді на їхні запити 24/7. Бот може надавати базову інформацію або пересилати запити до відповідного спеціаліста для подальшого вирішення. Це особливо корисно для клієнтів, які потребують оперативних відповідей або живуть у різних часових поясах.

Запит документів та інформації через бот: клієнти можуть запитувати документи або інформацію через бот, а потім автоматично отримувати доступ до відповідних файлів, завантажених у хмарне сховище компанії. Це зменшить кількість часу, витраченого на обмін документами через пошту чи інші канали.

Маркетинг та підтримка потенційних клієнтів: Telegram-бот може бути ефективним інструментом для залучення нових клієнтів. Через бот потенційні клієнти можуть отримувати інформацію про компанію "Avaris", її послуги, успішні кейси та рекомендації щодо звернення. Крім того, бот може автоматично пересилати контактні дані потенційних клієнтів до CRM для подальшого зв’язку з менеджерами компанії.

Автоматизовані консультації для нових клієнтів: бот може допомогти новим клієнтам отримати попередню консультацію щодо їхньої ситуації, відповідаючи на базові запитання та направляючи клієнта до відповідного

фахівця для детальнішої консультації. Це скоротить час на обробку первинних звернень і дозволить юристам зосередитися на складніших справах.

1.4 Завдання та мета роботи

Завданням кваліфікаційної роботи є розробка телеграм-бота для юридичної фірми "Avaris", який забезпечуватиме автоматизацію процесів консультування клієнтів. Бот повинен надавати клієнтам можливість отримувати інформацію про компанію, ціни на послуги, шаблони документів, а також забезпечувати оперативний зворотний зв'язок.

Для вирішення поставленої мети в роботі слід виконати наступні завдання:

- провести аналіз бізнес-процесів компанії "Avaris" та визначити основні потреби клієнтів у сфері юридичного консультування;
- розробити технічні вимоги до телеграм-бота, включаючи функціонал, інтерфейс і вимоги до безпеки;
- розробити алгоритм функціонування бота, який охоплюватиме обробку запитів клієнтів, пошук інформації та надання шаблонів документів;
- обрати платформу для розробки телеграм-бота, інструменти програмування та бази даних для зберігання інформації;
- реалізувати інтеграцію бота з базою даних компанії для автоматизованого надання актуальної інформації клієнтам;
- забезпечити можливість завантаження клієнтами шаблонів документів із бота та інтеграцію з відповідними сервісами (за потреби);
- реалізувати функцію зворотного зв'язку, яка дозволить клієнтам отримувати відповіді на запитання або залишати заявки на консультації;
- протестувати функціонал телеграм-бота для виявлення помилок і оптимізації роботи;

- оцінити ефективність впровадження телеграм-бота в процеси консультування клієнтів на основі тестових сценаріїв та отриманих результатів.

Розроблений телеграм-бот має забезпечувати високу швидкість відповіді, зручність використання, відповідність вимогам конфіденційності даних та можливість масштабування під зростаючі потреби компанії.

2 ТЕОРЕТИЧНИЙ РОЗДІЛ

2.1 Визначення потреб компанії «Avaris» у використанні автоматизованих консультацій

Для юридичної компанії «Avaris», яка спеціалізується на наданні консультаційних послуг, швидкий і якісний зворотний зв'язок із клієнтами є одним із ключових чинників успішної діяльності. З розвитком цифрових технологій та підвищенням очікувань клієнтів щодо доступності інформації, виникла потреба у впровадженні інструментів автоматизації бізнес-процесів, зокрема автоматизованих консультацій.

Основні потреби компанії у впровадженні автоматизованих консультацій наведено нижче.

Покращення доступності послуг для клієнтів:

- сучасні клієнти очікують можливості отримувати консультації та інформацію в режимі 24/7. Впровадження автоматизованих систем, таких як чат-боти, дозволить задовольнити ці очікування;
- юридичні послуги часто потребують оперативності, особливо в умовах термінових питань. Чат-бот зможе надати базову консультацію або необхідну інформацію миттєво, без необхідності залучення працівників компанії.

Оптимізація часу та ресурсів:

- автоматизовані консультації дозволять скоротити навантаження на співробітників компанії, звільняючи їхній час для вирішення більш складних завдань;
- телеграм-бот може виконувати рутинні завдання, такі як надання інформації про послуги, ціни чи шаблони документів, що зменшить кількість запитів до співробітників.

Стандартизація консультацій: використання автоматизованих рішень гарантує єдиний стандарт відповіді для всіх клієнтів. Це мінімізує ризик людської помилки або різночитань у наданій інформації.

Розширення клієнтської бази:

- доступність автоматизованих консультацій через телеграм-бот може залучити нових клієнтів, які цінують зручність і швидкість комунікації;
- можливість отримати базову інформацію безпосередньо в чат-боті підвищить довіру клієнтів до компанії, створюючи позитивний імідж технологічно розвиненої фірми.

Ефективність роботи з шаблонами документів: клієнти часто потребують базових юридичних документів, таких як договори чи заяви. Телеграм-бот зможе автоматично надавати відповідні шаблони. Це зменшить час, який працівники витрачають на підготовку стандартних документів.

Моніторинг і аналіз клієнтських запитів: телеграм-бот дозволяє автоматично збирати дані про запити клієнтів, що сприятиме аналізу їхніх потреб. Це допоможе компанії адаптувати послуги до реальних запитів ринку та оптимізувати бізнес-процеси.

Виклики, які вирішує автоматизація:

- швидка реакція на запити. Автоматизовані консультації дозволять клієнтам отримувати відповіді одразу, без необхідності очікування;
- зниження витрат на обслуговування. Впровадження телеграм-бота дозволить компанії мінімізувати витрати на рутинні консультації;
- покращення комунікації. Завдяки інтеграції з базами даних, бот може надавати точну та актуальну інформацію про компанію та її послуги.

Таким чином, впровадження автоматизованих консультацій відповідає сучасним викликам і потребам компанії «Avaris». Це дозволить підвищити ефективність її роботи, покращити клієнтський досвід та створити додаткову конкурентну перевагу на ринку юридичних послуг.

2.2 Огляд наукових досліджень у сфері автоматизації бізнес-процесів з використанням чат-ботів

Сучасні наукові дослідження активно аналізують можливості застосування чат-ботів у різних галузях.

Наукові роботи підтверджують, що чат-боти дозволяють автоматизувати ключові бізнес-процеси, такі як консультування клієнтів, прийом замовлень, обробка платежів та управління базами даних. Наприклад, дослідження Bui et al. (2020) демонструє, що чат-боти ефективно замінюють традиційні способи комунікації, такі як телефонні дзвінки чи електронна пошта, забезпечуючи миттєвий доступ до інформації. Це дозволяє компаніям економити до 30% витрат на обслуговування клієнтів. [9]

Серед основних переваг впровадження чат-ботів дослідники виділяють їхню цілодобову доступність, високу швидкість реагування на запити, можливість персоналізації обслуговування та зниження навантаження на співробітників. Використання алгоритмів машинного навчання дозволяє чат-ботам адаптуватися до індивідуальних потреб клієнтів, підвищуючи рівень задоволеності їхньою роботою. Завдяки автоматизації стандартних задач співробітники компанії можуть зосередитися на виконанні більш складних і творчих завдань.

У юридичній сфері чат-боти стають перспективним інструментом автоматизації. Вони можуть надавати клієнтам базову інформацію про законодавство, пропонувати шаблони документів та відповідати на стандартні запити. Дослідження Wang et al. (2021) вказують, що такі інструменти дозволяють зменшити час консультацій на 40% і водночас підвищити точність відповідей завдяки інтеграції з базами даних законодавства. Це робить чат-боти ефективними помічниками у правовій сфері.[10]

Чат-боти також є важливою складовою цифрової трансформації бізнесу. Як зазначають у дослідженнях Gartner (2020), вони не лише оптимізують існуючі процеси, але й сприяють створенню нових бізнес-моделей. Інтеграція чат-ботів у популярні месенджери, такі як Telegram чи Facebook Messenger, дозволяє

компаніям розширювати канали взаємодії з клієнтами, покращувати аналітику запитів і спрощувати використання послуг.[11]

Попри численні переваги, впровадження чат-ботів має і свої виклики. Технічні обмеження можуть призводити до некоректного оброблення складних або погано сформульованих запитів, що негативно впливає на досвід користувачів. Питання безпеки даних також є критичним, оскільки автоматизація потребує зберігання й обробки персональної інформації клієнтів. Високі початкові витрати на створення й налаштування чат-ботів можуть бути перешкодою для малого бізнесу, як зазначає Patel et al. (2021).

Науковці прогнозують подальший розвиток технологій чат-ботів у напрямках інтеграції голосових інтерфейсів, використання досконаліших алгоритмів штучного інтелекту та створення спеціалізованих ботів для окремих галузей, таких як юридична чи медична. Огляд досліджень свідчить, що чат-боти є потужним інструментом автоматизації, здатним трансформувати підходи до роботи компаній і створювати нові можливості для розвитку бізнесу.

2.3 Сучасні платформи для створення чат-ботів: порівняльний аналіз

Сучасні платформи для створення чат-ботів надають різноманітні можливості для автоматизації бізнес-процесів, забезпечення зручності взаємодії з клієнтами та покращення обслуговування. Вибір платформи залежить від конкретних потреб компанії, рівня технічної підготовки персоналу та бюджету. У цьому розділі буде розглянуто можливості популярних платформ, таких як Telegram, Viber, а також спеціалізовані інструменти для створення чат-ботів.

Однією з найбільш популярних платформ є Telegram, яка дозволяє створювати чат-ботів за допомогою Telegram Bot API. Telegram відзначається простотою інтеграції, широкими можливостями налаштувань та підтримкою таких функцій, як автоматичні відповіді, надсилання медіа-файлів і кнопок дій. Завдяки відкритій архітектурі API розробники можуть створювати ботів із кастомними функціями, інтегруючи їх з іншими системами. Telegram-боти є

особливо популярними завдяки швидкості роботи платформи, високому рівню захисту даних та великій аудиторії користувачів.

Viber також пропонує функціонал для створення чат-ботів, використовуючи Viber Bot API. Ця платформа підходить для компаній, які хочуть взаємодіяти з клієнтами через месенджер із широкою аудиторією, особливо у країнах, де Viber є популярним (наприклад, Україна). Viber-боти підтримують текстові повідомлення, мультимедіа, кнопки та опитування, що робить їх ефективними для маркетингових кампаній і клієнтської підтримки. Однак Viber має обмежену гнучкість порівняно з Telegram і менш розвинену екосистему сторонніх інтеграцій.

Серед платформ для створення ботів варто згадати Facebook Messenger, який дозволяє налаштовувати інтерактивних ботів для взаємодії з клієнтами на базі Facebook. Ця платформа є особливо корисною для бізнесів, орієнтованих на аудиторію Facebook, адже інтеграція з іншими інструментами Facebook, такими як реклами чи сторінки, спрощує взаємодію. Messenger-боти підтримують текстові відповіді, кнопки, каруселі, а також автоматизацію сценаріїв.

Ще однією популярною платформою є WhatsApp Business API, яка дозволяє компаніям створювати ботів для обслуговування клієнтів. WhatsApp має величезну базу користувачів і забезпечує високу швидкість та зручність комунікації. Основною перевагою є інтеграція з CRM-системами та можливість відправлення персоналізованих повідомлень. Однак використання WhatsApp Business API вимагає ліцензії, що робить її менш доступною для малого бізнесу.

Окрім зазначених месенджерів, існують спеціалізовані платформи для створення чат-ботів, такі як Chatfuel, ManyChat та Tars. Ці інструменти надають зручні інтерфейси для створення ботів без програмування, інтеграцію з популярними платформами (Telegram, Facebook Messenger, Viber) та широкий спектр функцій для автоматизації бізнес-процесів. Наприклад, ManyChat дозволяє створювати маркетингові кампанії з використанням чат-ботів, а Chatfuel ідеально підходить для автоматизації продажів.

Порівнюючи платформи, важливо враховувати як технічні можливості, так і цільову аудиторію. Telegram та WhatsApp підходять для універсального використання завдяки їхній популярності та функціональності. Viber є відмінним вибором для регіональних кампаній, орієнтованих на східноєвропейські ринки. Спеціалізовані інструменти, такі як Chatfuel та ManyChat, є ідеальними для бізнесів, які потребують швидкого запуску ботів без значних технічних витрат.

Таким чином, платформи для створення чат-ботів мають різні переваги, і вибір залежить від потреб компанії. Telegram і WhatsApp підходять для масштабних проектів, Viber — для регіональних потреб, а ManyChat і Chatfuel — для швидкої автоматизації маркетингових процесів.

У таблиці 2.1 наведено порівняльну характеристику усіх описаних платформ.

Таблиця 2.1 – Порівняльний аналіз сучасних платформ для створення чат-ботів

Платформа	Основні переваги	Обмеження	Цільова аудиторія
Telegram	Відкрите API, гнучкі налаштування, висока швидкість, безпека, широка аудиторія.	Вимагає певних технічних навичок для інтеграції складних функцій.	Універсальна, підходить для будь-якого бізнесу.
Viber	Популярність у певних регіонах (Україна та інші країни), підтримка мультимедіа, кнопок і опитувань.	Обмежена гнучкість порівняно з Telegram, менше інтеграцій зі сторонніми системами.	Локальні компанії, орієнтовані на східноєвропейські ринки.
Facebook Messenger	Інтеграція з екосистемою Facebook (реклама, бізнес-сторінки), підтримка каруселей, кнопок і сценаріїв.	Залежність від Facebook, менш популярний серед користувачів у деяких регіонах	Бренди, які активно використовують Facebook для просування.

		порівняно з WhatsApp.	
WhatsApp	Широка аудиторія, висока швидкість і зручність, інтеграція з CRM, персоналізовані повідомлення.	Потребує ліцензії, висока вартість для малого бізнесу.	Великі компанії, які надають сервіс через WhatsApp.
Chatfuel	Зручний інтерфейс для створення ботів без програмування, інтеграція з популярними платформами.	Обмежений функціонал для розробників, які хочуть реалізувати кастомні рішення.	Маркетологи, компанії малого та середнього бізнесу.
ManyChat	Інструменти для автоматизації маркетингових кампаній, простота використання.	Обмеження у функціональності для складних бізнес-процесів.	Бізнес, орієнтований на автоматизацію продажів.
Tars	Орієнтація на створення ботів для генерації лідів, простота інтеграції в веб-сторінки.	Підходить лише для вузького спектру завдань (лідогенерація).	Компанії, які потребують автоматизації процесу лідів.

2.4 Вибір та обґрунтування методів дослідження

2.4.1 Методологія розробки телеграм-ботів для автоматизації консультування

Визначення вимог та постановка цілей: на початковому етапі здійснюється детальне визначення завдань, які має вирішувати телеграм-бот. Зокрема, для юридичної компанії важливо врахувати потреби клієнтів у швидкому отриманні інформації про послуги, шаблони документів, а також доступ до консультацій. Основними вимогами до функціональності бота є: забезпечення доступу до інформації 24/7, інтеграція з базами даних компанії, персоналізація обслуговування клієнтів, а також можливість масштабування системи.[6]

Вибір технологічного стеку: для розробки телеграм-ботів зазвичай використовуються такі програмні мови, як Python, JavaScript або PHP, завдяки їхній сумісності з Telegram API. Найчастіше застосовуються фреймворки, які

спрощують процес розробки, наприклад, Aiogram або Telebot для Python, а також бібліотеки на основі Node.js, такі як Telegraf.js. Серверна частина зазвичай базується на хмарних платформах (AWS, Google Cloud, Heroku) або локальному сервері, залежно від обсягу даних та вимог до швидкодії.

Проектування архітектури бота: на цьому етапі створюється логічна структура бота, яка враховує ключові функції. Наприклад, бот повинен мати меню для навігації, систему авторизації клієнтів, інтеграцію з базами даних юридичної компанії для відображення послуг, шаблонів документів і консультаційних матеріалів.[6] Архітектура бота складається з кількох компонентів:

- frontend: взаємодія з користувачем через інтерфейс у Telegram;
- backend: обробка запитів, логіка роботи бота, інтеграція з базами даних та API;
- база даних: зберігання інформації про клієнтів, послуги та шаблони документів.

Реалізація функціональності: на етапі розробки здійснюється кодування функцій бота, зокрема створення команд, обробка запитів і відповідей. Для юридичної компанії важливими є такі функції:

- надання списку послуг із цінами;
- доступ до шаблонів документів, які користувач може завантажити;
- відповіді на поширені запитання через базу знань компанії;
- тестування та оптимізація.

Перед впровадженням бота в експлуатацію проводиться комплексне тестування. Це включає:

- перевірку коректності роботи всіх команд;
- тестування навантаження для визначення максимальної кількості одночасних користувачів;
- аналіз зручності інтерфейсу та швидкості обробки запитів.

На основі результатів тестування вносяться коригування для підвищення продуктивності бота та усунення можливих недоліків.

Після тестування бот інтегрується в бізнес-процеси компанії. Це включає налаштування доступу, навчання співробітників роботі з ботом, а також запуск його у публічному доступі. На етапі підтримки здійснюється моніторинг роботи бота, оновлення баз даних та регулярне вдосконалення функціональності на основі зворотного зв’язку від клієнтів.

2.4.2 Опис методів кількісної та якісної оцінки факторів, що впливають на продуктивність системи

Оцінка продуктивності інформаційних систем, таких як телеграм-боти для автоматизації консультування, є важливим етапом забезпечення їхньої ефективності. Для цього використовуються методи кількісного та якісного аналізу, які дають змогу оцінити швидкодію, надійність і зручність використання системи.

Кількісна оцінка базується на вимірюванні об'єктивних числових показників. Зокрема, ключовим фактором є час відгуку системи, тобто період, необхідний для обробки запиту користувача. Оптимальним вважається час у межах 1-3 секунд. Пропускна здатність системи визначається кількістю запитів, які вона може обробити за одиницю часу, і залежить від ефективності алгоритмів та серверної архітектури. Також важливим є аналіз стійкості системи до високих навантажень, що тестується через симуляцію одночасного доступу великої кількості користувачів. Іншим показником є частота помилок, яка свідчить про стабільність системи, а витрати ресурсів оцінюються через споживання пам’яті, процесорного часу та місця для зберігання даних.[6]

Якісний аналіз спрямований на оцінку суб’єктивних характеристик, пов’язаних із досвідом користувачів. Одним із ключових аспектів є зручність інтерфейсу, яка аналізується через опитування користувачів і оцінку легкості навігації та зрозумілості інтерфейсу. Тестування функціональності дозволяє перевірити відповідність системи поставленим завданням. Особлива увага

приділяється доступності для різних категорій користувачів, зокрема осіб із обмеженими можливостями. Додатково аналізуються відгуки користувачів і спостереження за їхньою взаємодією із системою, щоб виявити потенційні проблеми.

Поєднання кількісних і якісних методів дає змогу отримати всебічну оцінку продуктивності системи. Наприклад, об’єктивні дані про час відгуку або частоту помилок можуть бути зіставлені із відгуками користувачів для виявлення ключових слабких місць. Такий інтегрований підхід дозволяє краще зрозуміти пріоритети для вдосконалення.

Застосування кількісних і якісних методів забезпечує всебічну оцінку продуктивності, що дозволяє виявити слабкі місця системи та визначити напрями для її вдосконалення. Це сприяє створенню ефективних інструментів автоматизації, таких як телеграм-боти, які здатні значно покращити процес консультування клієнтів.

2.5 Модель взаємодії телеграм-бота з іншими компонентами системи

Телеграм-бот, що працює на сервері компанії, є ключовим елементом автоматизованої системи консультування. Його функціонування забезпечується інтеграцією з іншими компонентами інформаційної інфраструктури компанії, такими як бази даних, зовнішні API та клієнтські пристрої. Взаємодія між цими компонентами базується на чітко визначеній архітектурі та протоколах обміну даними.

Бот працює на сервері компанії, який виконує функцію основної платформи для обробки запитів користувачів. Сервер приймає повідомлення від клієнтів через API Telegram, обробляє їх, звертаючись до необхідних внутрішніх компонентів системи, і формує відповіді.

Для зберігання даних про клієнтів, їхні запити та відповіді бот використовує корпоративну базу даних, розташовану на окремому сервері. Ця база даних містить інформацію про клієнтів, історію їхніх запитів, ціни на послуги та іншу важливу інформацію. Бот через запити до бази даних забезпечує

швидкий доступ до цих даних, що дозволяє ефективно формувати персоналізовані відповіді.

На клієнтському боці користувачі взаємодіють із ботом через інтерфейс месенджера Telegram. Ця взаємодія є простою й інтуїтивно зрозумілою, оскільки Telegram підтримує текстові запити, кнопки, меню, а також мультимедійні матеріали.

Таким чином, телеграм-бот на сервері компанії взаємодіє з різними компонентами системи через централізовану архітектуру, що забезпечує його ефективність, зручність для клієнтів і відповідність сучасним вимогам безпеки.

2.6 Огляд технологій та інструментів для створення Telegram бота

2.6.1 Python

Python є однією з найпопулярніших мов програмування у світі, яка здобула визнання завдяки своїй простоті, універсальності та широкому спектру застосувань. Її зручний синтаксис дозволяє швидко освоїти мову навіть новачкам, тоді як досвідчені розробники цінують її потужність і гнучкість. У розробці Telegram ботів Python використовується як основний інструмент завдяки ряду унікальних переваг.

Перш за все, Python має величезну кількість бібліотек і фреймворків, які значно спрощують реалізацію різноманітних функцій. Для створення Telegram ботів існують спеціалізовані бібліотеки, такі як PyTelegramBotAPI та python-telegram-bot, які забезпечують повну інтеграцію з Telegram API. Ці бібліотеки надають готові інструменти для роботи з повідомленнями, кнопками, вебхуками та іншими функціями, що дозволяє розробникам зосередитися на бізнес-логіці бота, а не на технічних деталях.

Ще однією важливою перевагою Python є його сумісність з іншими технологіями. Мова підтримує інтеграцію з базами даних, такими як MySQL, PostgreSQL та MongoDB, що дозволяє ефективно зберігати та обробляти дані. Також Python має потужні інструменти для роботи з вебсервісами, обробки HTTP-запитів, інтеграції API інших платформ та реалізації серверної логіки.[3]

2.6.2 PyTelegramBotAPI

PyTelegramBotAPI — це популярна бібліотека для мови програмування Python, яка надає зручний інтерфейс для роботи з Telegram Bot API. Вона дозволяє розробникам швидко й ефективно створювати та налаштовувати Telegram ботів, реалізовуючи як базові, так і складні функціональні можливості.

Однією з ключових переваг PyTelegramBotAPI є простота використання. Бібліотека має інтуїтивно зрозумілий синтаксис і добре задокументовану структуру, що робить її доступною навіть для новачків у програмуванні. Вона забезпечує базові інструменти для обробки запитів від Telegram API, такі як отримання та надсилання повідомлень, обробка медіафайлів, створення кнопок і клавіатур, а також підтримка інтерактивних елементів.[1]

PyTelegramBotAPI пропонує два основні підходи до роботи з повідомленнями: довге опитування (long polling) та вебхуки (webhooks). Довге опитування дозволяє отримувати оновлення від Telegram у режимі реального часу, виконуючи запити до сервера Telegram через регулярні інтервали. Цей підхід є простим у реалізації та зручним для невеликих проєктів. Вебхуки, своєю чергою, надають можливість автоматично отримувати оновлення через HTTP-запити на визначений URL, що підходить для масштабованих і продуктивних систем.[1]

Бібліотека також підтримує різноманітні типи контенту, включаючи текстові повідомлення, фотографії, відео, аудіо, документи та навіть інтерактивні функції, такі як опитування чи вікторини. Крім того, PyTelegramBotAPI дозволяє створювати кастомізовані клавіатури, inline-кнопки, а також працювати з callback-запитами, що значно розширює можливості взаємодії бота з користувачами.

2.6.3 MySQL

MySQL — це потужна і широко використовувана система управління реляційними базами даних (RDBMS), яка забезпечує зберігання, обробку та

доступ до даних у структурованому вигляді. Вона працює на основі мови SQL (Structured Query Language), яка є стандартом для маніпулювання даними в реляційних базах даних. MySQL здобула популярність завдяки своїй продуктивності, надійності, простоті використання та відкритому коду.[2]

Однією з ключових переваг MySQL є її швидкість і масштабованість. Ця СУБД підходить як для невеликих проєктів із кількома таблицями, так і для великих корпоративних додатків із мільйонами записів. Вона забезпечує високу продуктивність завдяки ефективним алгоритмам індексації та оптимізації запитів.

MySQL підтримує транзакції, що дозволяє забезпечувати цілісність даних навіть у випадку збоїв. Це досягається через механізми, такі як блокування записів, перевірка цілісності даних і можливість відкату транзакцій. Для забезпечення надійності використовується підтримка реплікації — копіювання даних між різними серверами для підвищення відмовостійкості та розподілення навантаження.[2]

СУБД MySQL підтримує широкий спектр типів даних, включаючи числові, текстові, часові, двійкові та спеціалізовані типи, що дає змогу зберігати дані у зручному форматі. Крім того, MySQL дозволяє створювати складні структури баз даних за допомогою зв’язків між таблицями, забезпечуючи високу гнучкість у проектуванні баз даних.

Однією з сильних сторін MySQL є її сумісність із різними мовами програмування, такими як Python, Java, PHP, C++, що дозволяє легко інтегрувати її з веб-додатками та іншими системами. Наприклад, для взаємодії з Python можна використовувати популярні бібліотеки, такі як mysql-connector чи SQLAlchemy, що забезпечують просту інтеграцію бази даних у програмний код.

2.9 Висновки

У теоретичному розділі проведено всебічний аналіз сучасних технологій, платформ та інструментів для розробки чат-ботів, а також методів автоматизації бізнес-процесів. Встановлено, що впровадження чат-ботів є важливим кроком у

цифровій трансформації компаній, адже вони дозволяють оптимізувати взаємодію з клієнтами, підвищити ефективність роботи та знизити операційні витрати.

Дослідження показали, що Python, у поєднанні з бібліотекою PyTelegramBotAPI та базами даних MySQL, є одним із найкращих рішень для розробки чат-ботів завдяки їх гнучкості, доступності та потужному функціоналу. Окрему увагу приділено аналізу існуючих платформ і підходів до інтеграції чат-ботів у бізнес-системи, що дозволило визначити ключові переваги та недоліки різних інструментів.

3. СИНТЕЗ КОМП'ЮТЕРНОЇ СИСТЕМИ

3.1 Вибір і обґрунтування принципів розробки функціональної схеми програмного забезпечення

Розробка функціональної схеми запланованого об'єкта дослідження ґрунтується на принципах забезпечення ефективності, зручності та стабільності функціонування системи. Основна мета — створення взаємопов'язаної інфраструктури, яка підтримує безперебійну роботу Telegram-бота, операторів служби підтримки та бази даних компанії.

Принципи розробки функціональної схеми наведено нижче.

Модульність:

- обробка повідомлень від користувачів і операторів;
- управління базою даних;
- реалізація зворотного зв'язку через Telegram-бота;
- модульна структура забезпечує зручність у підтримці, масштабуванні та модернізації системи.

Централізація обробки даних:

- швидкий доступ до даних операторам і користувачам;
- узгодженість і цілісність інформації;
- центральна база даних об'єднує функціональні елементи системи, дозволяючи ефективно управляти потоками даних.

Взаємодія через API Telegram:

- простота інтеграції з існуючими інструментами (Python, MySQL);
- готові функції для обробки повідомлень, запитів і callback-даних;
- використання API Telegram дозволяє мінімізувати час на розробку базового функціоналу та сконцентруватися на кастомізації рішення.

Автоматизація процесів:

- надання інформації про компанію та послуги;
- здійснення пошуку даних у базі даних;
- захист від спаму та обмеження некоректної взаємодії з користувачами.

3.2 Функціональна схема інтеграції ПЗ в КС компанії

Згідно вимог було розроблено функціональну схему системи. Дана функціональна схема описує інфраструктуру взаємодії Telegram-бота із зовнішніми та внутрішніми компонентами системи. Основні елементи схеми:

1. Користувачі системи – взаємодіють із Telegram-ботом для отримання послуг або виконання певних запитів.
2. Telegram-бот – приймає запити від користувачів, надсилає їх на сервер, та отримує відповіді.
3. Інтернет – забезпечує зв’язок між користувачами, ботом і сервером.
4. Firewall – виконує функцію захисту мережі від несанкціонованого доступу.
5. Пограничний маршрутизатор – з’єднує локальну мережу компанії з Інтернетом.
6. Локальна мережа компанії – включає робочі станції, мережеві периферійні пристрої, сервери та інші компоненти.
7. Веб-сервер – обробляє запити, які надходять від Telegram-бота, і взаємодіє з іншими сервісами.
8. Файловий сервер – зберігає документи користувачів системи.
9. База даних бота – використовується для збереження даних, необхідних для роботи бота.
10. DNS-сервер – забезпечує трансляцію доменних імен у IP-адреси для запитів.

Ця структура забезпечує ефективну обробку запитів, збереження даних користувачів і надійність системи завдяки захисту на мережевому рівні.

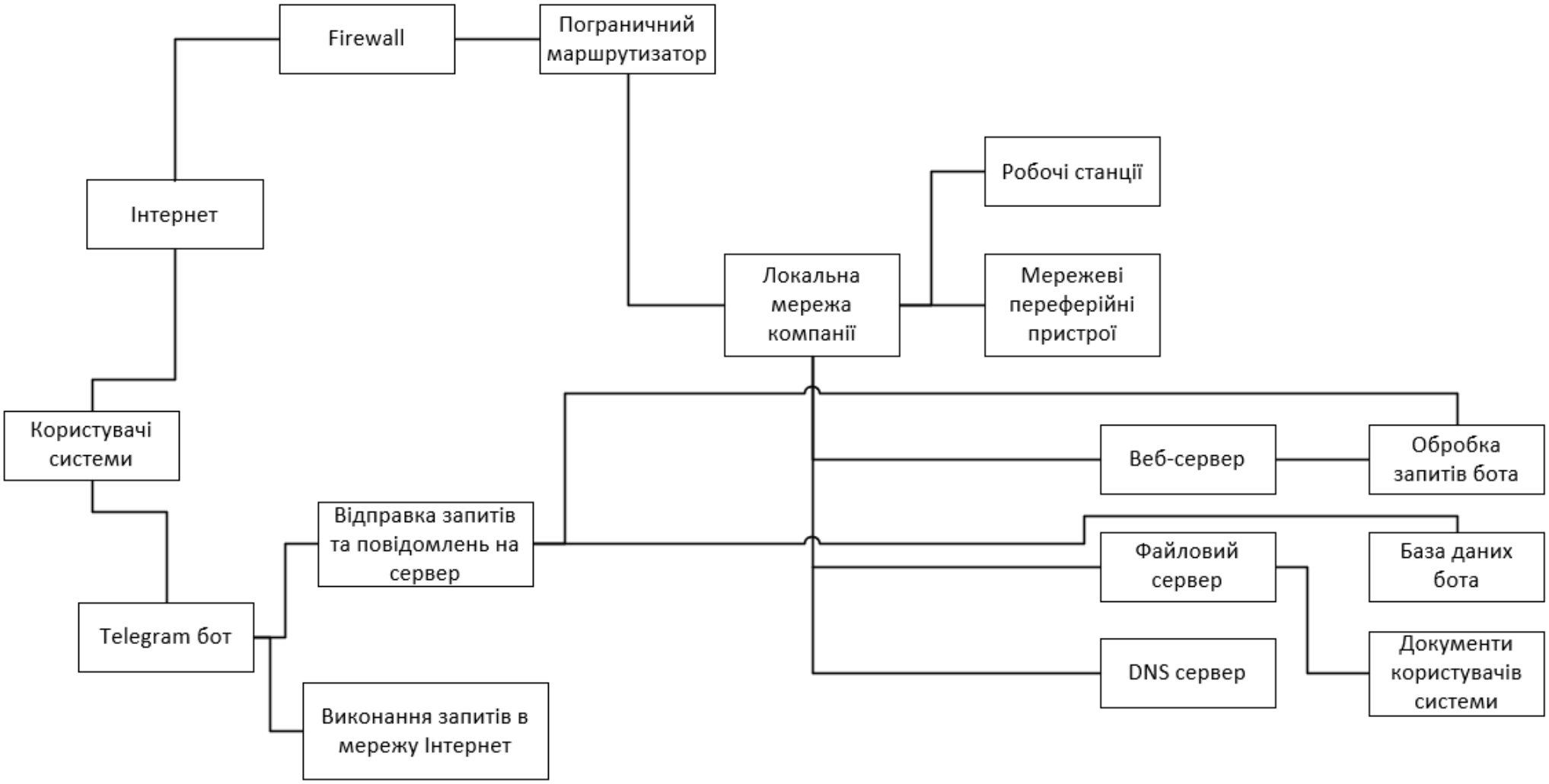

Рисунок 3.1 – Функціональна схема комп’ютерної системи компанії

3.3 Вимоги до функціональності телеграм-бота

Телеграм-бот, розроблений для автоматизації взаємодії з користувачами, повинен відповідати низці функціональних і нефункціональних вимог. Основна задача бота полягає у забезпеченні зручного та ефективного зв'язку користувачів із операторами служби підтримки.

Перш за все, бот повинен забезпечувати прямий зв'язок користувачів з операторами. Це передбачає можливість передачі повідомлень від користувача до представника служби підтримки та надсилання зворотної відповіді. У разі, якщо оператори недоступні, бот повинен повідомляти про це користувачів і надавати можливість залишити запит для подальшого розгляду.

Бот також повинен надавати користувачам інформацію про компанію, включаючи її напрями діяльності, історію, місію та основні цілі. Окрім цього, необхідно забезпечити доступ до контактних даних компанії, таких як адреса офісу, телефони, електронна пошта та посилання на офіційний вебсайт.

Однією з ключових функцій є забезпечення доступу до сховища шаблонів документів, розміщених на Google Диску.

Для підтримки високого рівня безпеки та контролю бот повинен включати механізм блокування користувачів. Команда підтримки має мати змогу блокувати недобросовісних або небажаних користувачів, що створюють загрозу нормальному функціонуванню системи.

Ще однією важливою вимогою є забезпечення захисту від спаму. Бот має обмежувати кількість повідомлень, які користувач може відправити, не отримавши відповіді. Це дозволить уникнути перевантаження системи та зберегти ефективність роботи операторів підтримки.

Крім функціональних можливостей, бот повинен відповідати загальним вимогам до продуктивності та надійності. Він має працювати безперебійно навіть за значного навантаження, обробляючи великий обсяг запитів користувачів у реальному часі. Інтерфейс бота повинен бути інтуїтивно зрозумілим, щоб забезпечити комфортне використання для широкого кола користувачів.

Таким чином, телеграм-бот повинен бути багатофункціональним, безпечним і зручним у використанні, забезпечуючи високу якість обслуговування та сприяючи ефективній взаємодії між користувачами та компанією.

3.4 Вимоги до системи

3.4.1 Вимоги до серверного обладнання системи

Серверне обладнання, на якому розгорнуто телеграм-бот, є критичним компонентом системи, від якого залежить стабільність, продуктивність та безпека всієї системи автоматизації. Для забезпечення коректної роботи серверів необхідно враховувати кілька ключових аспектів, таких як продуктивність, надійність, масштабованість та безпека.

Продуктивність сервера має бути достатньою для обробки великої кількості запитів від користувачів у реальному часі. Сервер повинен бути оснащений потужним багатоядерним процесором (рекомендується щонайменше 4-8 ядер із тактовою частотою від 3 ГГц) та мати достатній обсяг оперативної пам’яті (не менше 16 ГБ, з можливістю розширення). Це дозволить забезпечити швидку обробку запитів телеграм-бота, а також виконання інших сервісних завдань, таких як робота з базою даних або обробка великого обсягу інформації.

Обсяг сховища даних повинен відповідати потребам системи. Для зберігання логів, користувацьких рекомендується використовувати HDD диски у RAID-масиві, оскільки це забезпечує стійкість до відмови обладнання та захищає дані від втрат.

Надійність серверного обладнання є важливим фактором для забезпечення безперебійної роботи системи. Рекомендується використовувати сервери корпоративного класу з апаратними компонентами високої якості. Сервери повинні підтримувати механізми апаратного моніторингу для відстеження стану компонентів та виявлення можливих збоїв. Також важливо передбачити резервні джерела живлення (PSU) для запобігання перебоям у роботі через відключення електроенергії.

Безпека серверів є критично важливою, особливо якщо сервери зберігають чутливу інформацію або підключені до мережі інтернет. Серверне обладнання повинно підтримувати сучасні протоколи шифрування для захисту переданих даних. Рекомендується також використовувати апаратні засоби безпеки, такі як TPM-модулі (Trusted Platform Module) для додаткового захисту доступу.

Масштабованість серверного обладнання дозволить адаптувати систему до зростання навантаження. Для цього сервери повинні мати можливість розширення обчислювальних ресурсів, таких як додавання оперативної пам’яті, збільшення кількості процесорів чи накопичувачів. У разі збільшення кількості користувачів чи обсягу даних система має бути готова до горизонтального масштабування за рахунок підключення нових серверів до кластеру.

Підключення до мережі повинно бути високошвидкісним і надійним. Сервери повинні мати мережеві адаптери, які підтримують з'єднання на швидкості 1 Гбіт/с або більше. Для забезпечення стійкості до збоїв рекомендується використовувати резервні мережеві інтерфейси (Dual NIC).

Операційна система та програмне забезпечення на серверах повинні бути оптимізовані для роботи телеграм-бота. Рекомендовано використовувати Ubuntu Server.

У підсумку, серверне обладнання має відповідати вимогам щодо продуктивності, надійності, безпеки та масштабованості, що дозволить забезпечити стабільну та безперебійну роботу телеграм-бота і всієї системи автоматизації консультування.

3.5 Вимоги до програмного забезпечення

3.5.1 Вимоги до ПЗ працівників компанії

Комп’ютери працівників повинні бути оснащені ліцензованою операційною системою. Рекомендується використовувати сучасні версії ОС із регулярними оновленнями безпеки, такі як Windows 10/11 Pro або корпоративні дистрибутиви Linux (Ubuntu LTS, Fedora).

Працівники компанії повинні мати доступ до програм для роботи з текстами, таблицями, презентаціями та базами даних. Рекомендується використовувати Microsoft Office 365 або його альтернативи.

Для оперативного обміну інформацією між працівниками та взаємодії з клієнтами необхідно використовувати корпоративні платформи для обміну повідомленнями та організації відеоконференцій, такі як Microsoft Teams, Slack або Zoom. Доступ до телеграм-бота компанії також повинен бути налаштований через месенджер Telegram, який встановлений на робочих станціях працівників та мобільних пристроях, що використовуються в робочих цілях.

3.5.2 Вимоги до ПЗ бота

Програмне забезпечення телеграм-бота повинно відповідати низці технічних вимог, які забезпечать його стабільність, ефективність і безпеку:

- архітектура системи: розроблюване програмне забезпечення має бути побудоване на основі серверно-клієнтної архітектури. Серверна частина виконує всі обчислення, обробку запитів та взаємодію з Telegram API, а клієнтська частина представлена інтерфейсом користувача в месенджері Telegram. Бот повинен підтримувати модульну структуру, що дозволяє додавати нові функції без порушення роботи основного функціоналу;
- інтеграція з Telegram API: програмне забезпечення повинно забезпечувати повну сумісність із Telegram API для реалізації основних функцій бота. Використання бібліотеки PyTelegramBotAPI дозволяє спростити інтеграцію і надати підтримку всіх можливостей платформи, таких як надсилання текстових повідомлень, обробка команд, інтерактивні кнопки та відповіді на запити користувачів;
- робота з базою даних: для зберігання інформації про користувачів, цін на послуги а також дані про блокування користувачів використовується реляційна база даних MySQL. База даних повинна бути оптимізована для високої швидкості читання і запису;

- продуктивність: бот повинен забезпечувати обробку великої кількості запитів одночасно, навіть у випадках підвищеного навантаження. Розробка повинна враховувати оптимізацію коду, щоб уникати надмірного споживання ресурсів сервера;
- безпека: безпека системи є критично важливим аспектом. Комунікація між ботом і сервером повинна бути захищена за допомогою протоколу HTTPS. Доступ до адміністративного функціоналу, наприклад блокування користувачів, повинен бути обмежений і захищений авторизацією;
- масштабованість: програмне забезпечення повинно бути побудоване з урахуванням можливості масштабування. Це дозволить у майбутньому розширювати функціонал, наприклад, інтегрувати бот із CRM-системами, додати нові сервіси чи забезпечити роботу з більшою кількістю користувачів.

3.6 Структурна схема логіки бази даних бота

На рисунках 3.1 та 3.2 зображено структурні схеми логічних операцій бази даних бота.

Ця схема логіки бази даних для бота описує основні сутності та їх зв'язки:

- користувачі (USERS) містить інформацію про користувачів бота, зокрема їхні унікальні ідентифікатори, імена, електронні адреси, паролі та дату реєстрації;
- запити (QUERIES) зберігає інформацію про запити користувачів, включаючи текст запиту, дату створення, статус та зв'язок з конкретним користувачем;
- відповіді (RESPONSES) надані ботом на запити користувачів, зберігають текст відповіді та час її надання. Кожна відповідь прив'язана до конкретного запиту;

– історія запитів (HISTORICAL_QUERIES) для зберігання історії запитів, що були зроблені користувачами. Вона містить текст запиту та час, коли він був надісланий.

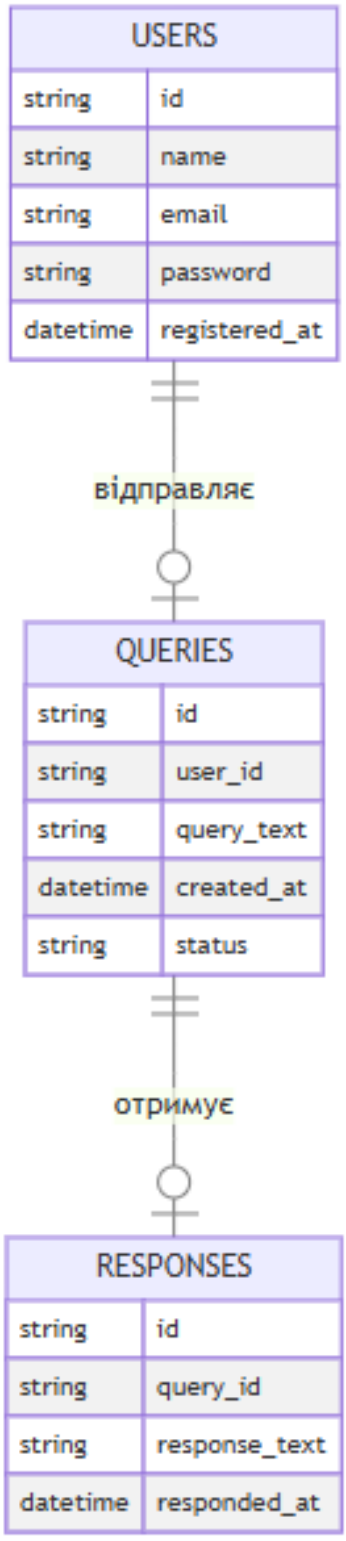

Рисунок 3.2 - База даних: Користувачі, Запити, Відповіді

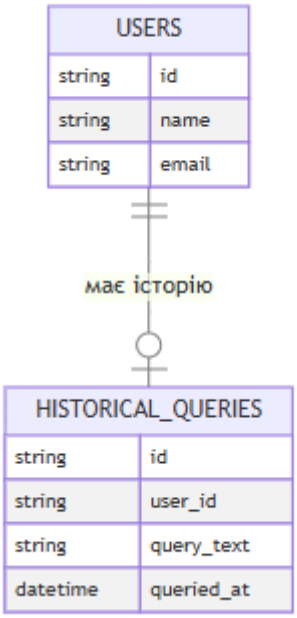

Рисунок 3.3 – Історія запитів

3.7 Структура взаємодій в системі

Система взаємодій у компанії організована таким чином, щоб забезпечити ефективну комунікацію між ключовими суб'єктами: операторами центру підтримки, користувачами та базою даних компанії. Центральним елементом цієї структури є Telegram-бот, який виступає як посередник у всіх комунікаційних процесах.

Користувачі взаємодіють із системою через Telegram-бота. Бот автоматизує звернення до операторів, дозволяючи оперативно передавати запити, надсилати повідомлення або блокувати користувачів у разі порушення правил.

Оператори центру підтримки отримують запити користувачів через Telegram-бота до окремого чату, в якому вони опрацьовують запити та надсилають відповіді користувачам. Вони мають доступ до інструментів для адміністрування ботом, включаючи управління доступом користувачів, аналіз вхідних запитів та внесення оновлень у базу даних компанії.

База даних компанії зберігає ключову інформацію, яка використовується для обробки запитів. Telegram-бот забезпечує автоматизований доступ до бази, обробляючи запити користувачів і передаючи необхідні дані операторам.

Таким чином, структура взаємодій забезпечує безперебійний обмін інформацією між усіма суб'єктами системи, автоматизуючи рутинні процеси й підвищуючи ефективність роботи компанії.

3.8 Висновки

У процесі синтезу системи було розроблено архітектуру, яка відповідає сучасним вимогам щодо автоматизації бізнес-процесів і взаємодії користувачів із сервісами компанії. Визначено ключові компоненти системи, серед яких Telegram бот, оператори центру підтримки, база даних та інші технічні засоби.

Було сформульовано технічні та організаційні вимоги, що є базисом для подальшої реалізації.

Запропоновані технічні рішення дозволяють досягти високої продуктивності системи, оптимізувати роботу операторів підтримки, мінімізувати затримки у відповіді користувачам та забезпечити зручність взаємодії. У результаті синтезована система здатна задовольнити як внутрішні потреби компанії, так і запити клієнтів, сприяючи підвищенню ефективності бізнес-процесів та покращенню якості обслуговування.

4 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Призначення й область застосування програмного забезпечення

Дана програма призначена для автоматизованого надання інформації про компанію користувачам, забезпечення можливості зворотного зв’язку з операторами підтримки через систему тікетів, також бот надає доступ до архіву шаблонів документів. Бот забезпечує ефективну роботу системи зворотного зв’язку, дозволяючи компанії швидко відповідати клієнтам.

4.2 Постановка завдання на розробку програми

Об’єкт розробки: Telegram-бот для юридичної компанії "Avaris".

Мета розробки: Створення програмного забезпечення для забезпечення зручної та ефективної взаємодії клієнтів із операторами підтримки компанії "Avaris" через систему Telegram.

Нижче наведено завдання на створення боту підтримки для юридичної компанії:

- бот повинен забезпечувати клієнтам можливість створення запитів до операторів підтримки через систему тікетів;
- повідомлення клієнтів автоматично пересилаються в окремий чат підтримки, де оператори можуть відповідати на них;
- бот повинен надавати користувачам інформацію про компанію через меню, реалізоване за допомогою inline-клавіатур;
- бот повинен надавати користувачам інформацію про ціни на послуги компанії за допомогою команди /price;
- перелік послуг та відповідних цін зберігається у базі даних MySQL;
- оператори підтримки повинні мати можливість блокування користувачів;

- система повинна забезпечувати захист від спаму, обмежуючи кількість повідомлень, які користувач може надіслати до отримання відповіді;
- основна бібліотека для роботи з Telegram API: pyTelegramBotAPI;
- база даних MySQL для зберігання інформації про послуги, користувачів і тікети.

Очікуваний результат: Telegram-бот, який дозволяє клієнтам юридичної компанії "Avaris" зручно отримувати зворотній зв’язок від операторів, дізнаватися інформацію про послуги та їх вартість, а також забезпечує ефективну модерацію та захист від спаму.

4.3 Обґрунтування технічних характеристик програм

Розроблений Telegram-бот для юридичної компанії "Avaris" створений із використанням бібліотеки PyTelegramBotAPI та бази даних MySQL, що обумовлює вибір відповідного програмного забезпечення. Для роботи бота достатньо комп’ютера середньої потужності із доступом до інтернету, здатного забезпечити виконання запитів до бази даних і обробку повідомлень у реальному часі.

Програма ефективно функціонує на сервері з процесором, який підтримує багатопоточну обробку даних, мінімум 4 ГБ оперативної пам’яті та постійним сховищем для збереження логів і резервних копій бази даних. Операційна система Linux або Windows забезпечує стабільне середовище для запуску бота. Обрана бібліотека PyTelegramBotAPI оптимізована для роботи з Telegram API, що дозволяє реалізувати всі необхідні функції, включаючи управління повідомленнями, зворотний зв'язок і взаємодію з inline клавіатурами.

Така конфігурація технічних засобів гарантує швидкодію, стійкість до навантажень і надійну роботу бота, що відповідає потребам клієнтів компанії "Avaris".

4.4 Розробка програмного забезпечення

4.4.1 Вибір програмних засобів для розробки ПЗ

Для ефективної розробки програмного забезпечення були обрані сучасні інструменти, які забезпечують зручність роботи, гнучкість налаштувань та підтримку необхідних функціональних можливостей. Розглянемо використані програмні засоби:

- Visual Studio Code – це популярний текстовий редактор, який забезпечує широкі можливості для написання, налагодження та тестування коду. Він підтримує велику кількість мов програмування та має вбудовану інтеграцію з системами керування версіями, такими як Git. Завдяки багатому вибору розширень, зокрема для роботи з Python, цей редактор є зручним і функціональним інструментом для створення програмного забезпечення;
- бібліотека PyTelegramBotAPI є основною бібліотекою для взаємодії з API Telegram. Вона надає зручний інтерфейс для створення чат-ботів, обробки повідомлень, роботи з командами та іншими функціями Telegram;
- бібліотека PyMySQL використовується для взаємодії Python-програми з базою даних MySQL. Вона дозволяє виконувати запити до бази даних, обробляти результати, додавати чи змінювати записи, а також забезпечує надійний зв’язок між Telegram ботом та базою даних;
- MySQL обрано як основну базу даних завдяки її високій продуктивності, стабільності та підтримці масштабованості. Вона зберігає всі дані, необхідні для функціонування системи, зокрема дані користувачів та інформацію про блокування. MySQL забезпечує швидку обробку запитів та підтримує складні реляційні структури;
- для обробки регулярних виразів використовується стандартна бібліотека Python – re. Вона дозволяє ефективно аналізувати текст, виконувати перевірки форматів даних, здійснювати пошук і заміну текстових

фрагментів, що є важливим для обробки повідомлень користувачів у чат-боті;

- MySQL Workbench використовувався для перегляду, адміністрування та аналізу вмісту бази даних. Цей інструмент дозволяє зручно створювати таблиці, виконувати SQL-запити, переглядати дані та здійснювати налаштування бази даних.

Вибір зазначених програмних засобів був обумовлений їхньою надійністю, популярністю серед розробників та можливістю ефективної реалізації поставлених завдань. Ці інструменти забезпечують узгоджену роботу всіх компонентів системи, що є ключовим для досягнення високої якості розробленого програмного забезпечення.

4.4.2 Розробка клієнтської частини

Клієнтський інтерфейс бота реалізовано за допомогою Inline клавіатур та текстових повідомлень, які допомагають користувачам орієнтуватися в головному меню бота.

Для користувачів представлено додаткові команди для доступу до інформації: /price, яка дозволяє знаходити інформацію про вартість послуг, та /faq, яка виводить відповіді на часті запитання.

4.4.3 Розробка серверної частини

Серверна частина Telegram-бота юридичної компанії "Avaris" є основою функціонування системи, забезпечуючи обробку даних, взаємодію з базою даних MySQL і координацію між користувачами та операторами підтримки.

Розробка серверної частини виконана на мові програмування Python із використанням бібліотеки PyTelegramBotAPI, яка надає інструменти для роботи з Telegram API. Серверна частина відповідає за обробку команд і повідомлень від користувачів, передачу відповідей і забезпечення захисту від спаму.

Основним елементом серверної частини є обробник повідомлень, який отримує запити від користувачів і визначає, які дії потрібно виконати. Для цього реалізовано систему команд, наприклад, команда /start ініціалізує діалог із ботом і показує головне меню, а команда /price отримує з бази даних список послуг і цін та надсилає відповідь користувачеві.Для реалізації системи тікетів серверна частина обробляє повідомлення користувачів і пересилає їх у чат підтримки операторів.

Відповіді операторів автоматично передаються назад до користувачів. Цей механізм дозволяє забезпечити ефективну комунікацію між клієнтами та підтримкою.

Захист від спаму реалізовано через лічильник повідомлень: серверна частина обмежує кількість повідомлень, які користувач може надіслати без відповіді від оператора. У разі перевищення ліміту користувач отримує повідомлення про блокування подальших дій.

Для зберігання даних про користувачів, тікети, послуги компанії та ціни серверна частина взаємодіє з базою даних MySQL. Запити до бази виконуються через бібліотеку pymysql, яка забезпечує зручний спосіб виконання SQL-запитів із Python-коду.

Серверна частина також має логіку для управління доступом. Оператори підтримки отримують права блокувати користувачів, зберігаючи інформацію про заблокованих осіб у базі даних. Це дозволяє запобігти зловживанням і забезпечити безпеку комунікації.

Таким чином, серверна частина Telegram-бота розроблена з урахуванням вимог продуктивності, безпеки та гнучкості. Її архітектура забезпечує надійну роботу бота, зручність для клієнтів і ефективну підтримку операторами.

4.5 Опис розробленого ПЗ

4.5.1 Загальні відомості про програму

Розроблений Telegram-бот для юридичної компанії "Avaris" є програмним рішенням, яке забезпечує автоматизацію обслуговування клієнтів та взаємодію з

операторами підтримки. Програма дозволяє надавати інформацію про компанію, послуги та їхні ціни, а також реалізує систему тікетів для зворотного зв’язку. Бот створено з використанням бібліотеки PyTelegramBotAPI та бази даних MySQL, що забезпечує його функціональність, гнучкість і продуктивність.

4.5.2 Функціональне призначення програмного забезпечення

Розроблюване програмне забезпечення призначене для автоматизації процесів взаємодії між користувачами та службою підтримки компанії через Telegram бот. Воно покликане підвищити ефективність обробки запитів, покращити якість обслуговування клієнтів та забезпечити оперативний доступ до необхідної інформації.

Основні функціональні можливості програмного забезпечення:

- забезпечення швидкої двосторонньої комунікації між користувачами та операторами підтримки через бот. Повідомлення користувачів автоматично передаються до чату операторів, які відповідають на них через спеціалізовану панель;
- надання користувачам доступу до інформації про компанію, включно з напрямами її діяльності та контактними даними;
- надання операторам можливості блокувати порушників, обмежувати спам та контролювати активність користувачів;
- обмеження кількості повідомлень від користувачів, які ще не отримали відповіді, з метою запобігання перевантаження системи.

Розробка даного програмного забезпечення має на меті знизити навантаження на операторів підтримки, прискорити вирішення запитів клієнтів та покращити загальну продуктивність системи.

4.5.3 Опис структури розробленої програми

Розроблена програма є багатомодульною системою, що забезпечує ефективну реалізацію функціоналу Telegram-бота. Програма організована таким

чином, щоб забезпечити модульність, зручність підтримки та легкість масштабування. У цьому розділі детально описано структуру програми, функції кожного модуля та каталогів.

Головний модуль main.py є точкою входу для запуску програми. Він виконує наступні основні функції:

- ініціалізацію всіх необхідних компонентів бота, таких як клавіатури, обробники повідомлень та зворотних викликів;
- встановлення з'єднання з базою даних через спеціалізовані модулі;
- запуск основного циклу Telegram-бота, використовуючи бібліотеку PyTelegramBotAPI;
- моніторинг роботи бота та логування помилок для забезпечення стабільності системи;
- модуль main.py відповідає за координацію роботи всієї системи та забезпечує взаємодію між іншими модулями.

Модуль config.py використовується для зберігання ключових параметрів і конфігураційних даних, які необхідні для функціонування системи. Його основні функції:

- зберігання токенів Telegram API для авторизації бота;
- вказання ID адміністраторів і операторів підтримки для керування правами доступу;
- містить параметри для підключення до бази даних, такі як ім'я хоста, ім'я користувача, пароль і назва бази даних;
- забезпечує централізоване зберігання налаштувань, що спрощує адміністрування програми.

Модуль розмітки клавіатур: keyboards.py. Модуль keyboards.py містить функції для створення та управління інтерфейсними клавіатурами бота. Основні функції цього модуля:

- створення вбудованих клавіатур (inline keyboards), які використовуються для вибору опцій або виклику функціональних команд;
- створення стандартних клавіатур для навігації, які дозволяють користувачам легко взаємодіяти з ботом;
- забезпечення динамічного формування кнопок залежно від контексту чи потреб користувача;
- модуль забезпечує інтуїтивно зрозумілу взаємодію користувачів із ботом через зручний інтерфейс.

Модуль mysql_handler.py реалізує всі операції, пов'язані з базою даних. Його завдання включають:

- встановлення з'єднання з базою даних MySQL за допомогою бібліотеки PyMySQL;
- виконання SQL-запитів для додавання, видалення та оновлення даних у таблицях бази;
- отримання даних з бази для відображення користувачам чи операторам підтримки;
- управління транзакціями для забезпечення цілісності та надійності даних;
- модуль забезпечує ефективну взаємодію між Telegram-ботом та базою даних, виконуючи всі необхідні запити;
- Модуль msg_handler.py відповідає за обробку вхідних повідомлень від користувачів та операторів підтримки.

Основні функції:

- реалізація механізму зворотного зв'язку між користувачами та операторами через спеціальні чати;
- аналіз змісту повідомлень для визначення запитів і подальших дій;
- відправлення відповідей користувачам залежно від отриманого запиту;
- цей модуль забезпечує якісний сервіс підтримки користувачів, обробляючи всі їхні звернення.

Модуль обробки callback-значень: callback_handler.py. Модуль callback_handler.py відповідає за обробку зворотних викликів (callback data), які генеруються під час натискань на кнопки вбудованих клавіатур. Основні функції модуля:

- обробка даних, переданих із кнопок клавіатури;
- реалізація навігації користувачів між різними розділами бота;
- виведення відповідних даних чи повідомлень залежно від вибору користувача;
- цей модуль є критично важливим для забезпечення інтерактивності та зручності використання бота.

Модуль find_price.py реалізує функціонал пошуку даних у базі, таких як ціни на послуги чи інша релевантна інформація. Його основні функції:

- формування SQL-запитів для отримання даних із бази відповідно до запитів користувача;
- форматування результатів пошуку у вигляді зручному для відображення;
- надсилання отриманих результатів користувачу через Telegram-бота;
- цей модуль значно підвищує функціональність бота, дозволяючи швидко надавати користувачам важливу інформацію.

Каталог resources містить усі текстові файли, які використовуються ботом для відправлення користувачам. Сюди входять шаблони відповідей, документи, посилання тощо. Основні переваги такого підходу:

- централізація ресурсів для зручного редагування та оновлення;
- відокремлення статичних даних від основного коду програми, що спрощує її підтримку.

Розроблена структура програми забезпечує модульність, що дозволяє вносити зміни або розширювати функціонал без значного впливу на інші компоненти. Завдяки чіткій організації, кожен модуль виконує свою роль, забезпечуючи надійну та ефективну роботу Telegram-бота.

Структуру робочого каталогу бота наведено на рисунку 4.1.

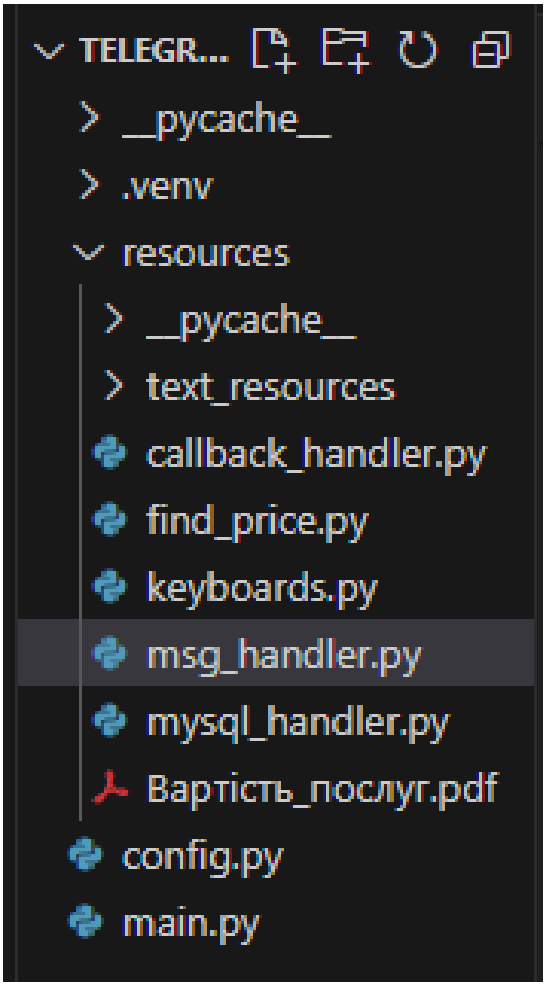

Рисунок 4.1 – Робочий каталог бота

4.5.4 Вхідні дані

Бот обробляє наступні вхідні команди користувачів:

- /start — для початку роботи з ботом;
- /price — для отримання інформації про послуги та їх вартість.

Також бот обробляє вхідні через inline-клавіатури для отримання інформації про компанію або виклику оператора і запити від користувачів у вигляді текстових повідомлень у рамках системи тікетів.

Від операторів підтримки бот приймає команди для блокування користувачів або закриття тікетів.

4.5.5 Вихідні дані

У якості вихідних даних бот генерує наступні відповіді користувачам:

- інформаційні повідомлення про компанію;
- перелік послуг і їх вартість;
- підтвердження створення тікетів і статус їх обробки.

Для операторів підтримки бот надає наступні вихідні дані:

- інформація про нові запити від користувачів;
- можливість відповідати на запити та керувати статусами тікетів.

4.5.6 Виклик і завантаження бота

Запуск бота виконується у створеному віртуальному середовищі. Перед запуском бота необхідно запустити роботу бази даних MySQL. Запуск самого бота виконується через запуск файлу main.py командою python.exe.

Користувач розпочинає роботу з ботом виконавши команду /start в особистому чаті з ботом у Телеграмі.

Після виконання команди користувач отримує вітальне повідомлення, де описується базовий функціонал бота, під даним повідомленням розміщується

клавіатура, за допомогою якої виконується навігація ботом та доступ до інформації.

4.5.7 Опис логічної структури програми

Програма Telegram-бота має модульну структуру, яка забезпечує її масштабованість та зручність у підтримці. Головний модуль main.py виконує роль точки входу, ініціалізуючи всі підсистеми та модулі. Модуль config.py відповідає за зберігання конфігураційних параметрів, таких як токени, ідентифікатори та параметри підключення до бази даних. Модуль keyboards.py забезпечує генерацію inline-клавіатур для зручної навігації користувачів у боті.

Модуль mysql_handler.py виконує всі операції з базою даних, включаючи збереження запитів, обробку списку заблокованих користувачів та отримання інформації про послуги. Модуль msg_handler.py відповідає за обробку текстових повідомлень користувачів, маршрутизуючи їх відповідно до типу запиту. Callback-запити, що виникають при натисканні кнопок у меню, обробляються модулем callback_handler.py, який забезпечує динамічну взаємодію з користувачами. Для команди /price використовується модуль find_price.py, який виконує пошук відповідних даних у базі та формує відповіді для користувача. Усі текстові файли та шаблони повідомлень зберігаються у каталозі resources, що дозволяє легко змінювати їх без редагування коду програми.

4.5.8 Схематичне зображення алгоритму роботи програмного забезпечення

На рисунку 4.2 зображено загальну схему алгоритму програми. Дана схема описує роботу програми після виконання користувачем команди /start. Після початку роботи бота користувач може або скористуватися меню, представленим у вигляді Inline клавіатури, або надіслати повідомлення, для того, щоб зв’язатися з оператором підтримки.

На рисунку 4.3 зображено функціональну схему процесу пересилання повідомлень між особистим чатом користувача з ботом та чатом операторів підтримки.

На рисунку 4.4 зображено алгоритм роботи команди /ban, яка дозволяє заблокувати користувача, заборонивши йому надсилати повідомлення боту. На рисунку 4.5 зображено алгоритм роботи аналогічної команди, яка дозволяє розблокувати користувача.

На рисунку 4.6 зображено алгоритм роботи команди /price, яка дозволяє користувачеві здійснювати пошук цін на послуги, які його цікавлять. Пошук здійснюється серед значень відповідної таблиці в базі даних, обираючи усі значення, подібні до того, що увів користувач.

На рисунку 4.7 зображено функціональну схему алгоритму функції, яка виводить користувачу попередження про спам, інформуючи його, що він може надіслати ще одне повідомлення.

На рисунку 4.8 зображено алгоритм роботи функції, яка виводить користувачу сповіщення про те, що його повідомлення більше не будуть передаватися команді підтримки до тих пір, поки він не отримає відповіді.

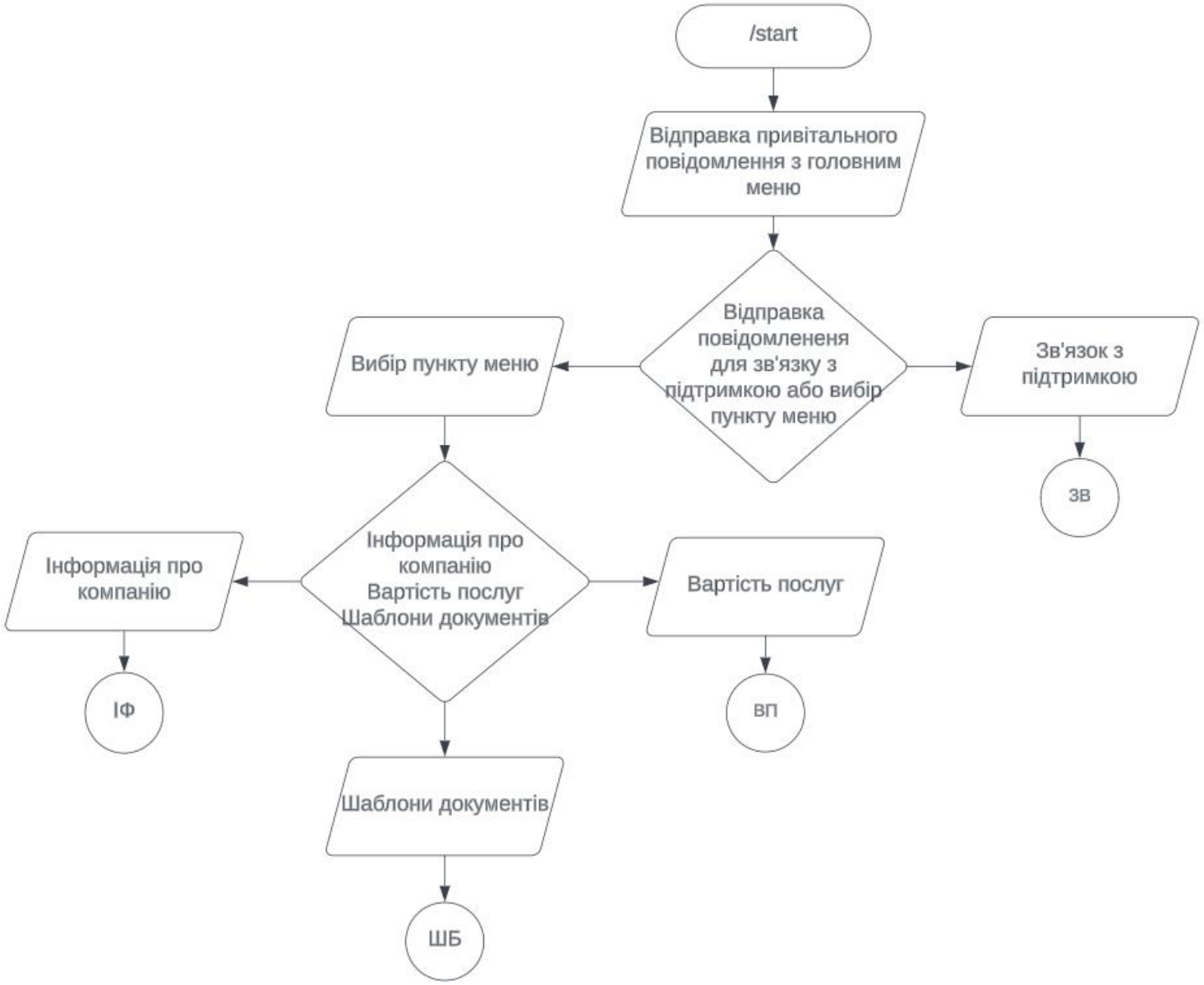

Рисунок 4.2 – Блок-схема вхідної точки програми

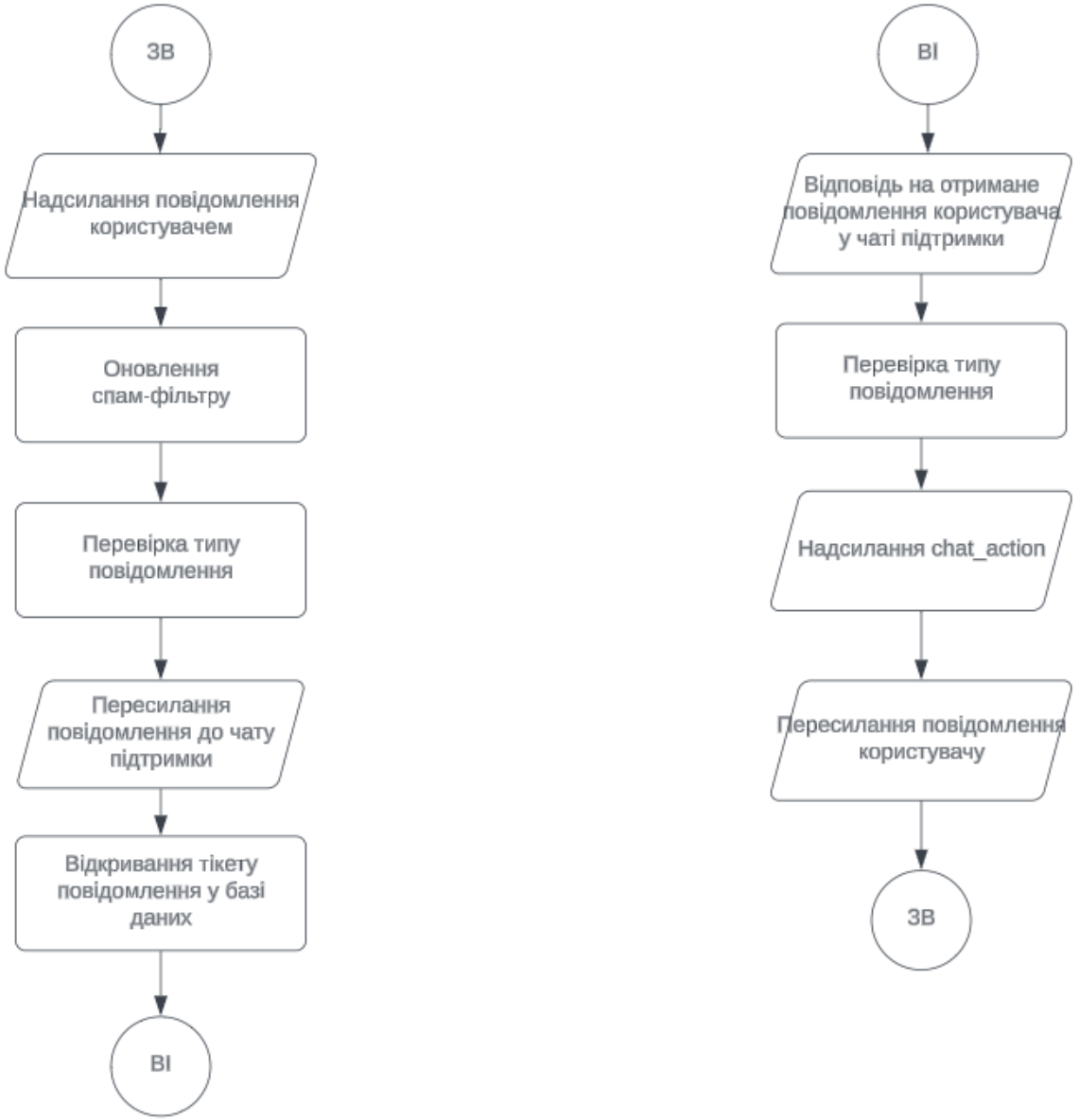

Рисунок 4.3 – Схема пересилання повідомлень між ботом та чатом підтримки

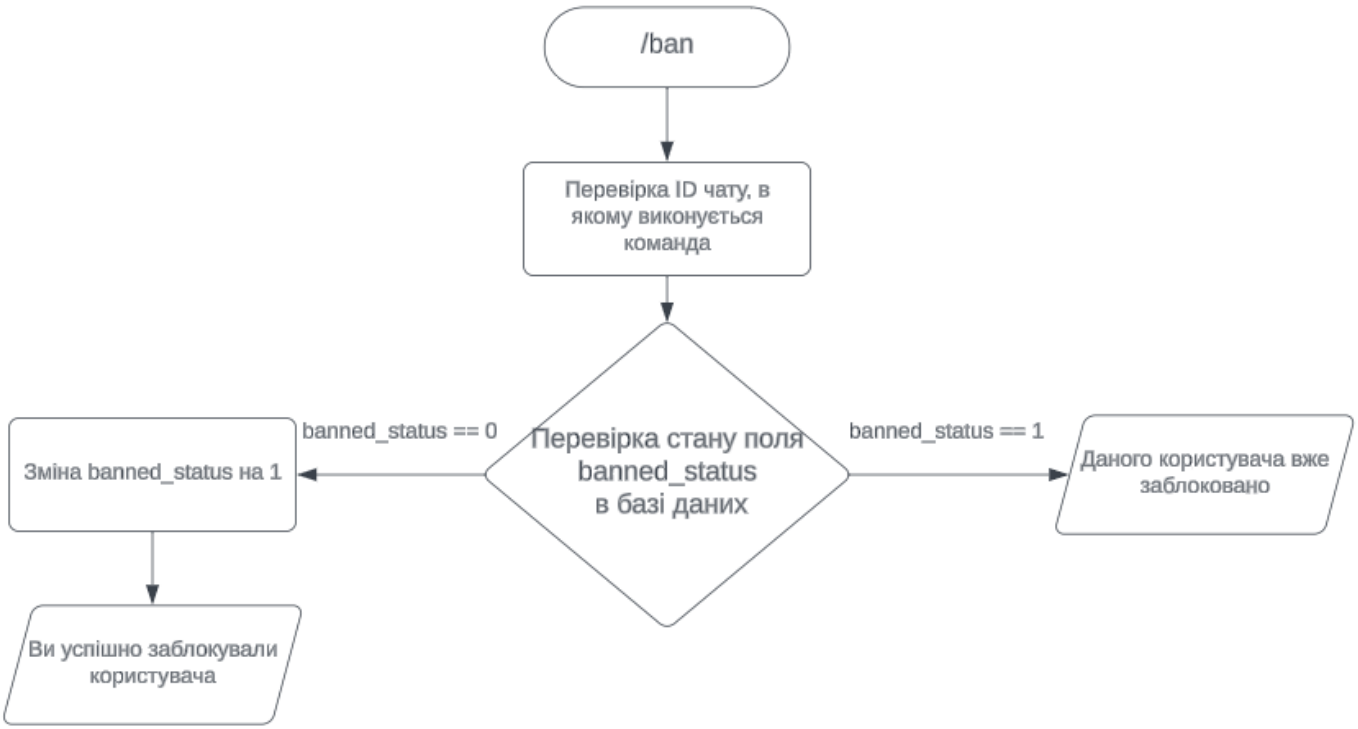

Рисунок 4.4 - Алгоритм роботи команди /ban

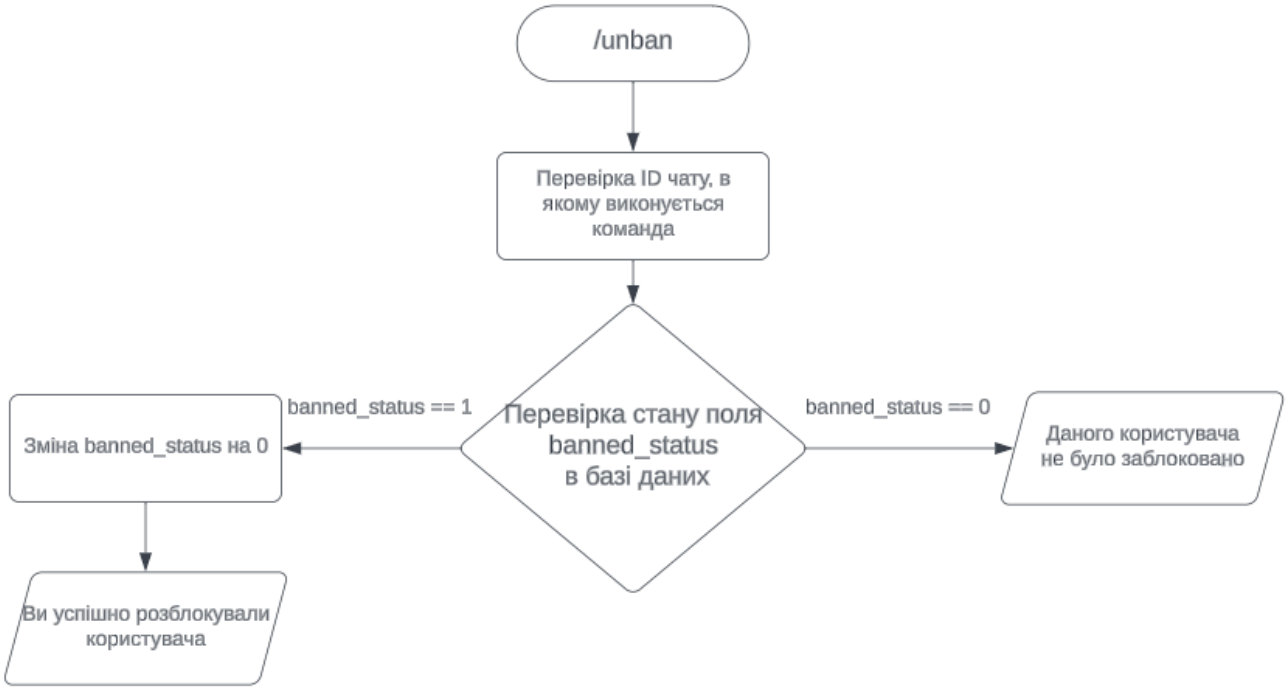

Рисунок 4.5 - Алгоритм роботи команди /unban

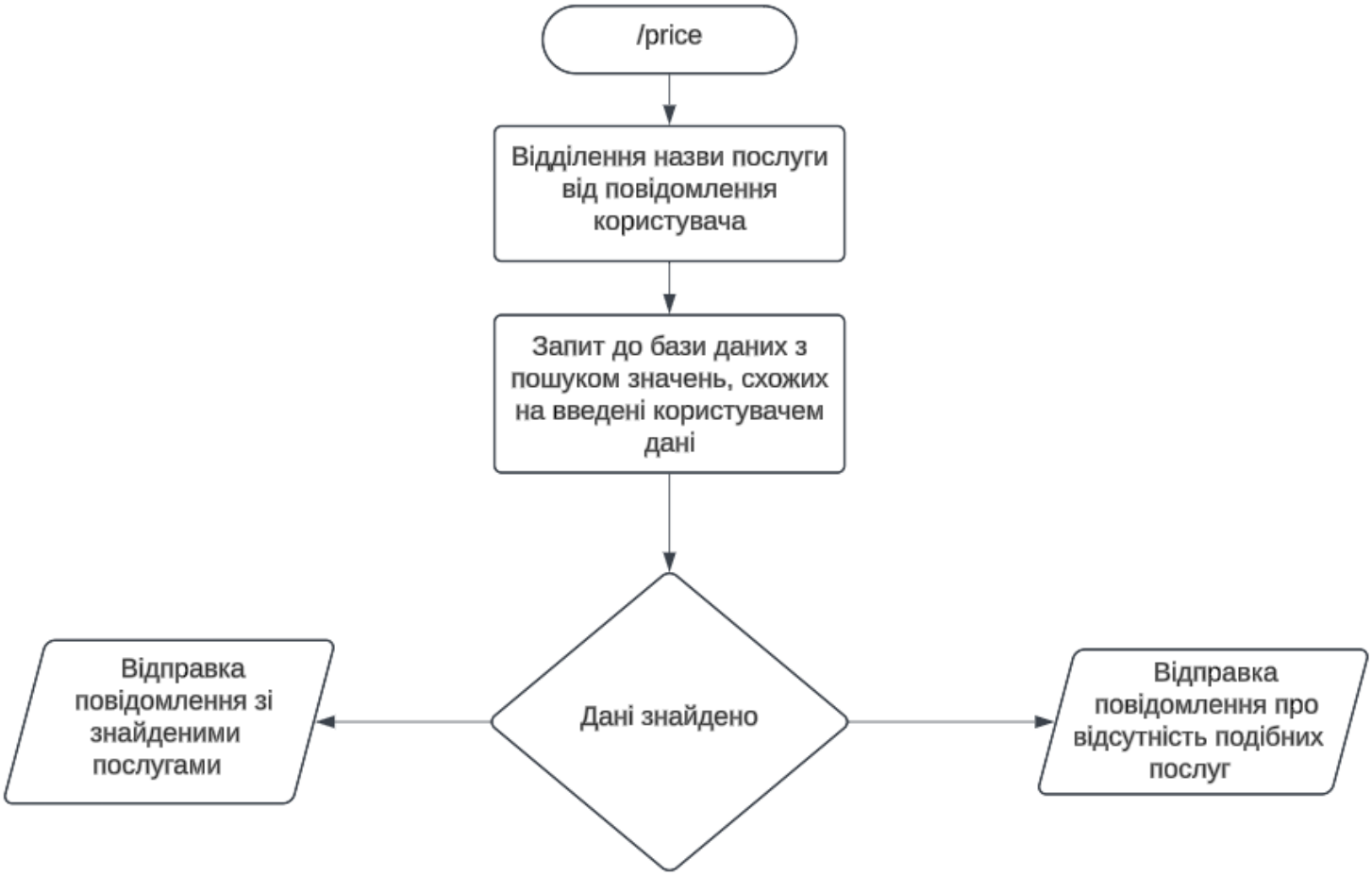

Рисунок 4.6 – Алгоритм команди пошуку цін на послуги

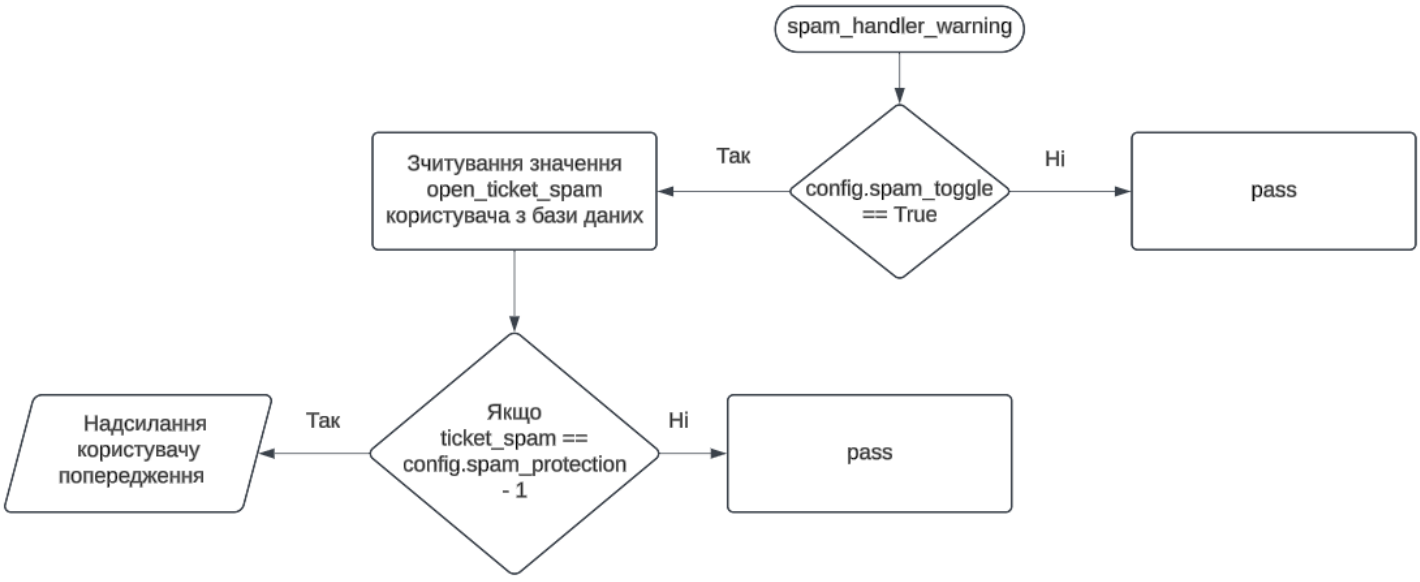

Рисунок 4.7 – Алгоритм функції попередження користувачів про спам

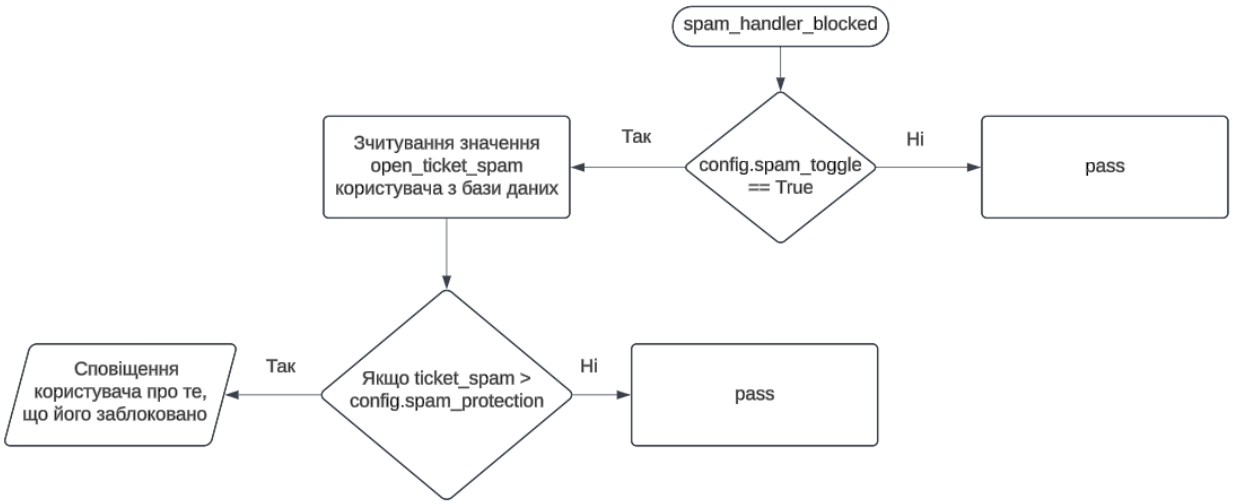

Рисунок 4.8 – Алгоритм функції інформування користувачів про обмеження надсилання повідомлень

4.6 Опис функціоналу та демонстрація роботи бота

На рисунку 4.9 зображено виконання команди /start користувачем та отримання вітального повідомлення.

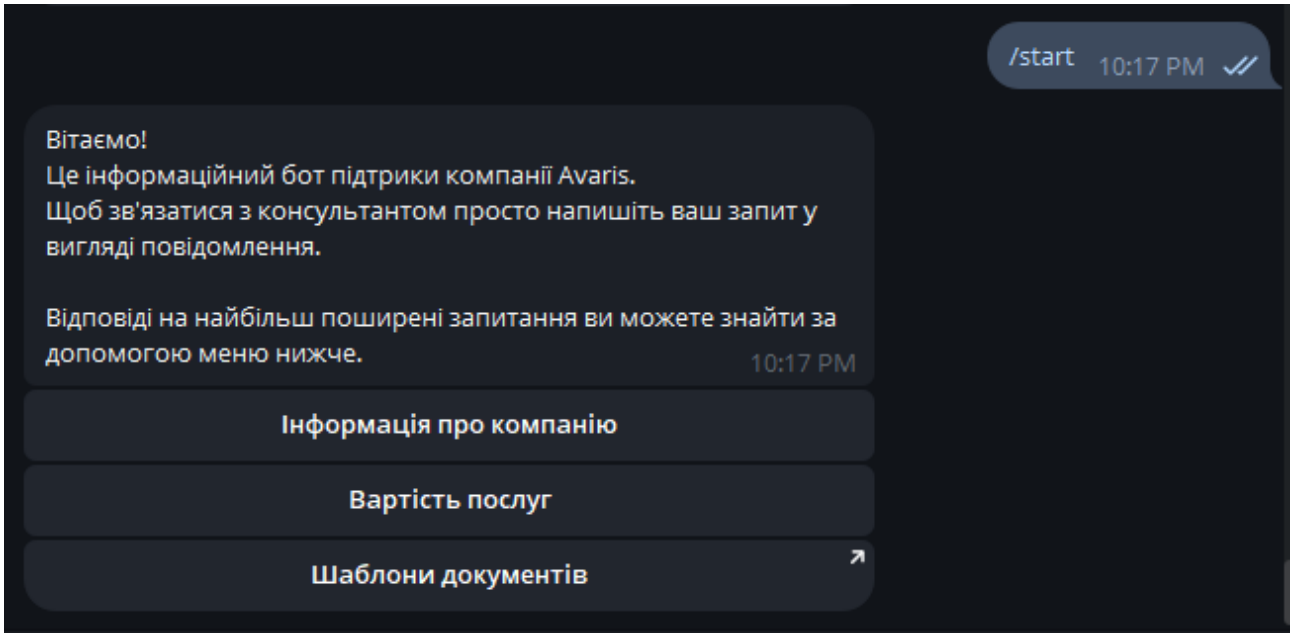

Рисунок 4.9 – Початок роботи з ботом

Виконання даної команди забезпечується наступним кодом:

```
@bot.message_handler(commands=['start'])
def start(message):
    if message.chat.type == 'private':
```

```
        bot.send_message(message.chat.id,
                         config.text_messages['start'].format(message.from_user.first_name),
                         parse_mode='html', disable_web_page_preview=True,
reply_markup=kb.main)
        mysql.start_bot(message.chat.id)
    else:
        bot.reply_to(message, 'Будь ласка, надішліть мені особисте повідомлення, якщо ви хочете зв\'язатися з командою підтримки.')
```

При натисканні кнопки «Інформація про компанію» вітальне повідомлення змінюється на повідомлення, яке пропонує обрати інформацію, яка цікавить користувача. Також клавіатура змінюється на перелік доступної інформації про компанію (рисунок 4.10).

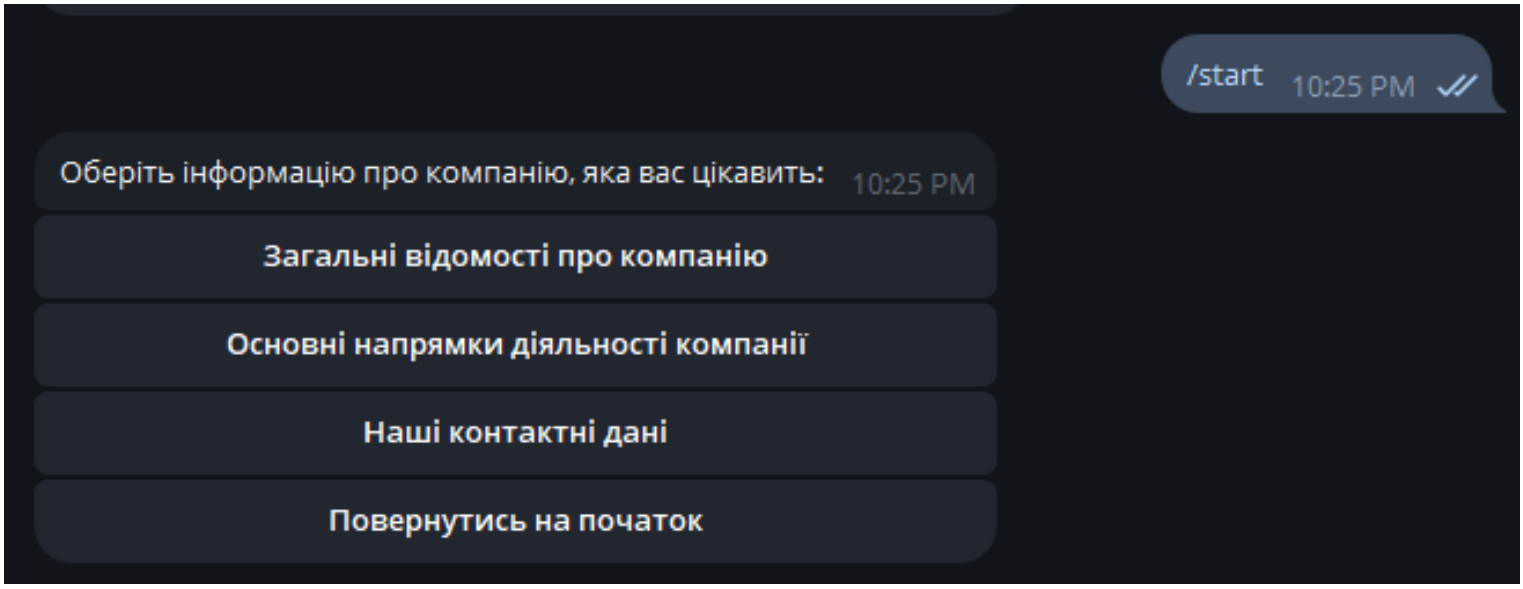

Рисунок 4.10 – Меню вибору інформації про компанію

Нижче наведено код розмітки клавіатури даного розділу бота:

```
aboutCompany = InlineKeyboardMarkup(keyboard=[
    [InlineKeyboardButton(text="Загальні відомості про компанію", callback_data="about_company")],
    [InlineKeyboardButton(text="Основні напрямки діяльності компанії", callback_data="activity")],
    [InlineKeyboardButton(text="Наші контактні дані", callback_data = "contacts")],
```

[InlineKeyboardButton(text="Повернутись на початок", callback_data="back")]
])

На рисунку 4.11 зображено виведення загальної інформації про компанію. Це реалізовано через callback_data, даний параметр привласнюється кнопці клавіатури, при натисканні кнопки спрацьовує callback і виконується відповідний функціонал. Код наведено нижче:

```
elif call.data == "about_company":
        file_path = "resources/text_resources/company_info.txt"
        bot.edit_message_text(chat_id=call.message.chat.id,
message_id=call.message.message_id,
                              text=read_info_from_file(file_path),
parse_mode='Markdown',
                              disable_web_page_preview=True,
reply_markup=kb.back_to_about)
```

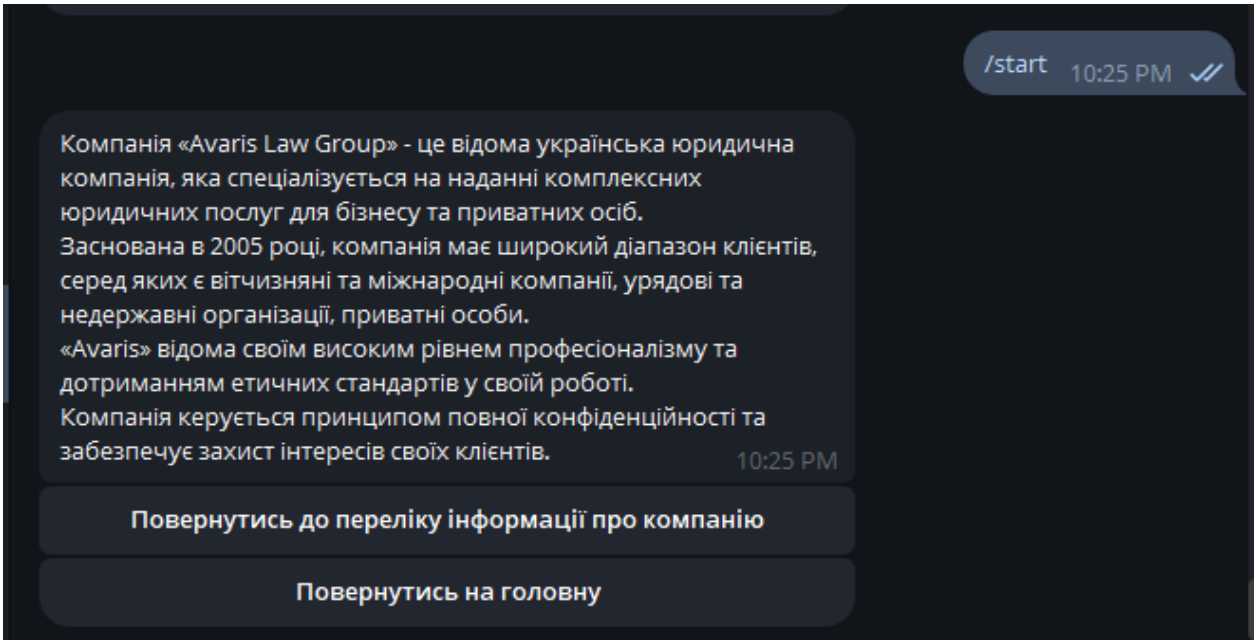

Рисунок 4.11 – Загальна інформація про компанію

На рисунку 4.12 зображено перелік напрямків діяльності компанії. При натисканні на кнопку виводиться інформація про відповідний напрям. На

рисунку 4.13 зображено хендлери, які відповідають за виведення інформації про напрямки діяльності компанії.

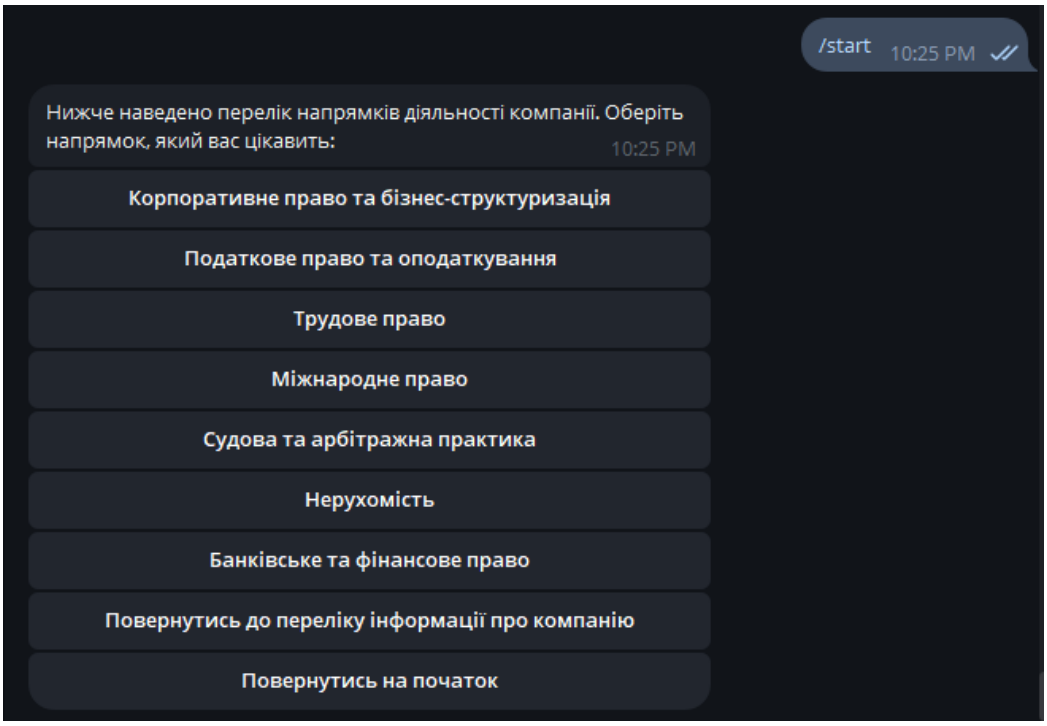

Рисунок 4.12 – Виведення переліку інформації про напрямки діяльності компанії

```
elif call.data == "activity":
    bot.edit_message_text(chat_id=call.message.chat.id, message_id=call.message.message_id,
                          text="Нижче наведено перелік напрямків діяльності компанії. Оберіть напрямок, який вас цікавить:\n\n", parse_mode='Markdown',
                          disable_web_page_preview=True, reply_markup=kb.aboutActivity)
elif call.data == "contacts":
    file_path = "resources/text_resources/contacts.txt"
    bot.edit_message_text(chat_id=call.message.chat.id, message_id=call.message.message_id,
                          text=read_info_from_file(file_path), parse_mode='Markdown',
                          disable_web_page_preview=True, reply_markup=kb.back_to_about)
elif call.data == "corporate":
    file_path = "resources/text_resources/corporate.txt"
    bot.edit_message_text(chat_id=call.message.chat.id, message_id=call.message.message_id,
                          text=read_info_from_file(file_path), parse_mode='Markdown',
                          disable_web_page_preview=True, reply_markup=kb.back_to_activity)
elif call.data == "taxes":
    file_path = "resources/text_resources/taxes.txt"
    bot.edit_message_text(chat_id=call.message.chat.id, message_id=call.message.message_id,
                          text=read_info_from_file(file_path), parse_mode='Markdown',
                          disable_web_page_preview=True, reply_markup=kb.back_to_activity)
elif call.data == "work_rights":
    file_path = "resources/text_resources/work_rights.txt"
    bot.edit_message_text(chat_id=call.message.chat.id, message_id=call.message.message_id,
                          text=read_info_from_file(file_path), parse_mode='Markdown',
                          disable_web_page_preview=True, reply_markup=kb.back_to_activity)
elif call.data == "international":
    file_path = "resources/text_resources/international.txt"
    bot.edit_message_text(chat_id=call.message.chat.id, message_id=call.message.message_id,
                          text=read_info_from_file(file_path), parse_mode='Markdown',
                          disable_web_page_preview=True, reply_markup=kb.back_to_activity)
elif call.data == "court":
    file_path = "resources/text_resources/court.txt"
    bot.edit_message_text(chat_id=call.message.chat.id, message_id=call.message.message_id,
                          text=read_info_from_file(file_path), parse_mode='Markdown',
                          disable_web_page_preview=True, reply_markup=kb.back_to_activity)
elif call.data == "realty":
    file_path = "resources/text_resources/realty.txt"
    bot.edit_message_text(chat_id=call.message.chat.id, message_id=call.message.message_id,
                          text=read_info_from_file(file_path), parse_mode='Markdown',
                          disable_web_page_preview=True, reply_markup=kb.back_to_activity)
elif call.data == "bank":
    file_path = "resources/text_resources/bank.txt"
    bot.edit_message_text(chat_id=call.message.chat.id, message_id=call.message.message_id,
                          text=read_info_from_file(file_path), parse_mode='Markdown',
                          disable_web_page_preview=True, reply_markup=kb.back_to_activity)
```

Рисунок 4 .13 – Перелік хендлерів інформації про напрямки діяльності

На рисунку 4.14 зображено виведення інформації про контактні дані компанії.

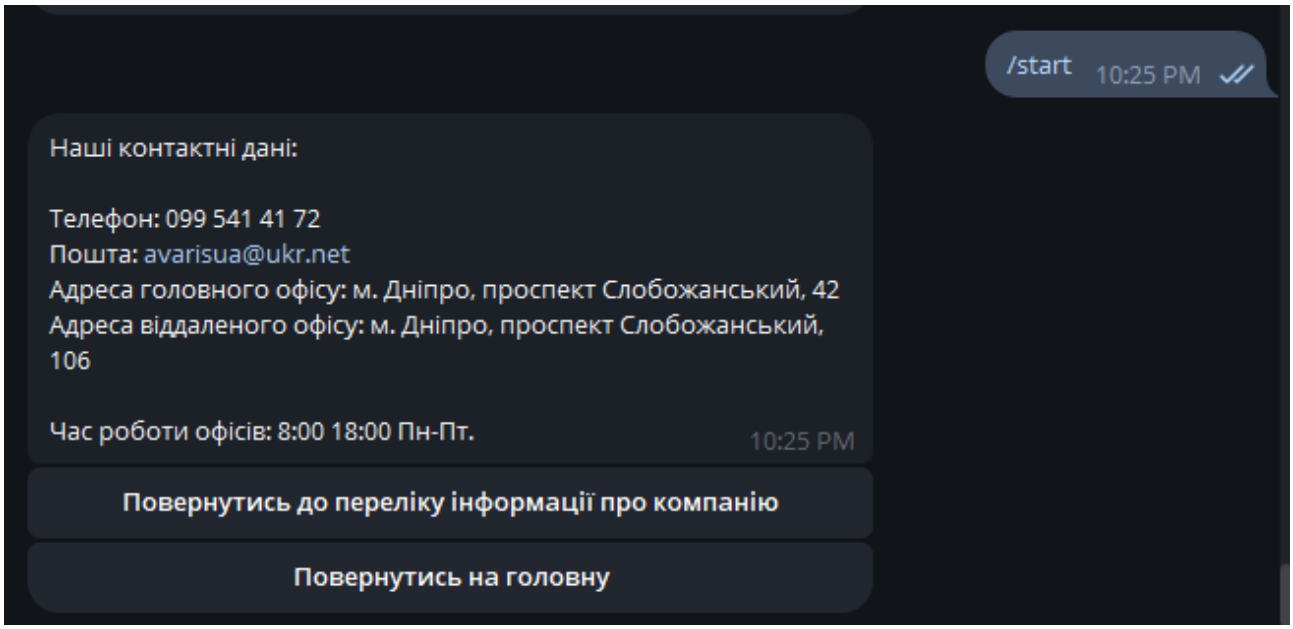

Рисунок 4.14 – Контактні дані компанії

На рисунках 4.15 – 4.21 зображено інформацію про напрямки діяльності компанії, яку виводить бот.

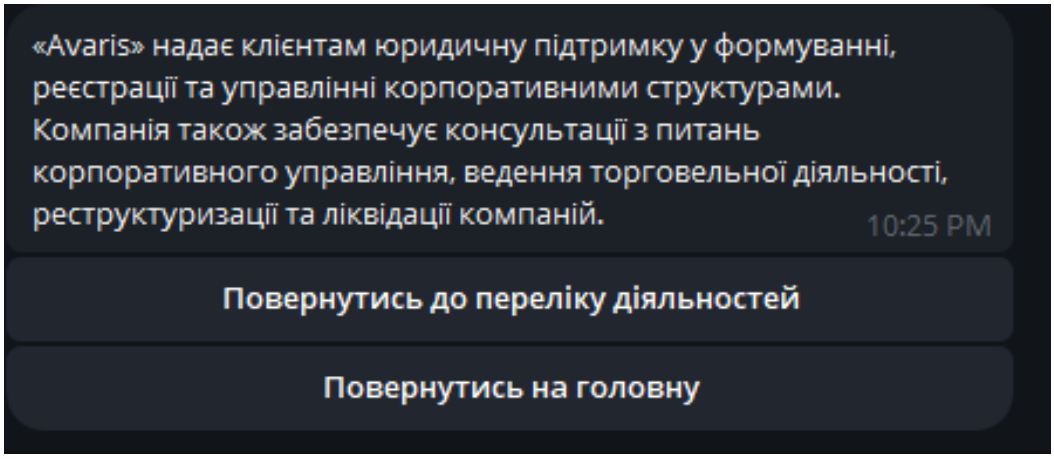

Рисунок 4.15 – Інформація про корпоративне право

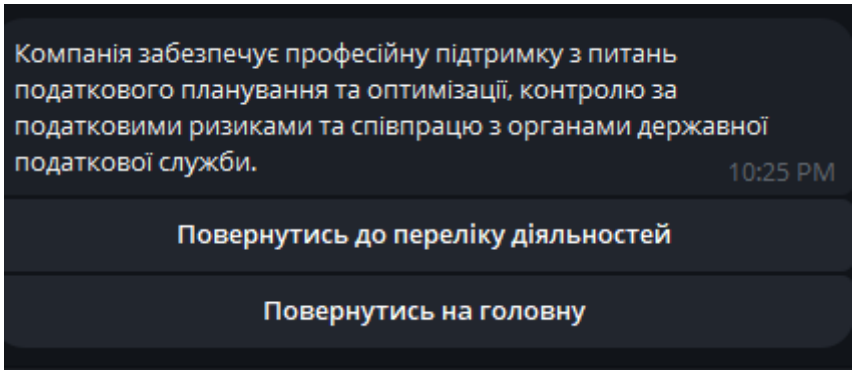

Рисунок 4.16 – Інформація про податкове право

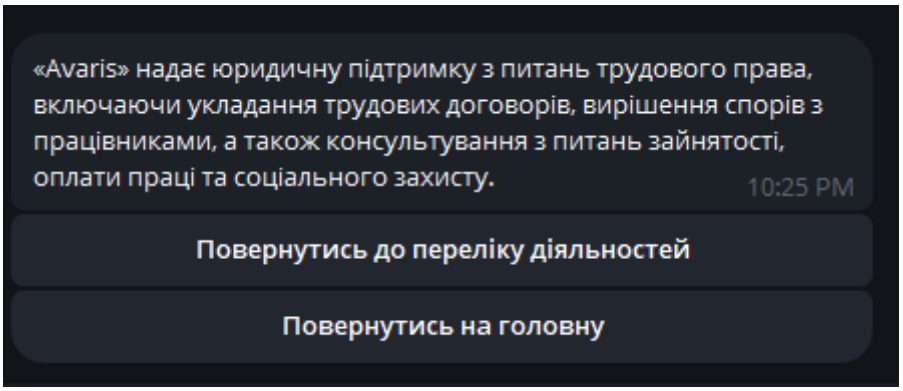

Рисунок 4.17 – Інформація про трудове право

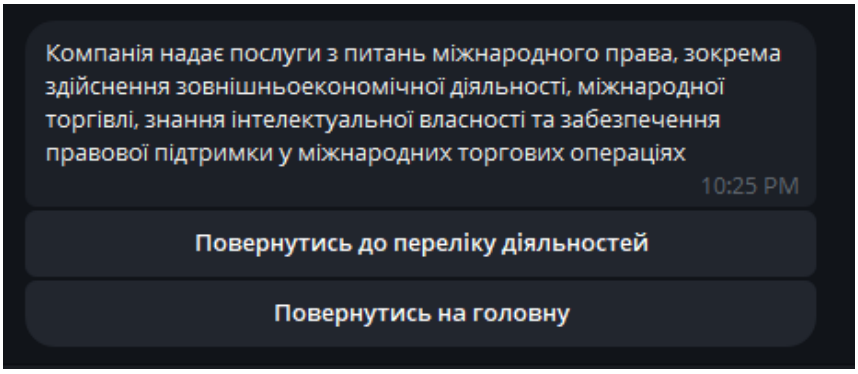

Рисунок 4.18 – Інформація про міжнародне право

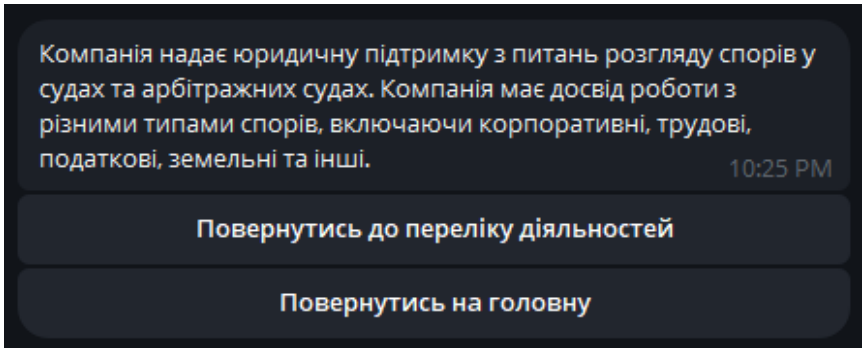

Рисунок 4.19 – Інформація про судову практику

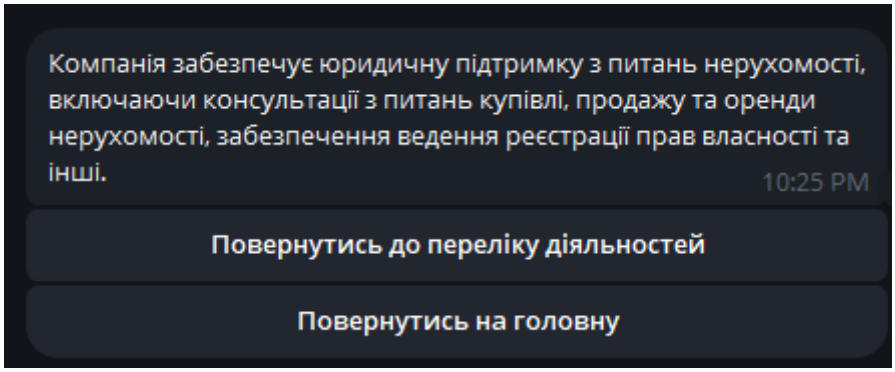

Рисунок 4.20 – Інформація про нерухомість

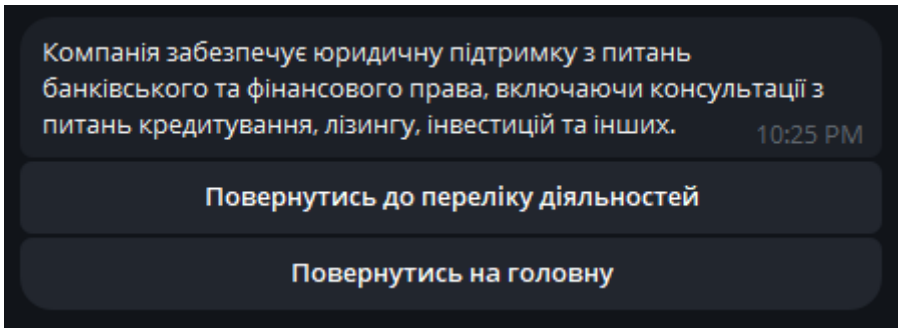

Рисунок 4.21 – Інформація про банківське право

На рисунку 4.22 зображено результат натискання на кнопку «Вартість послуг», яка знаходиться в головному меню бота. Дана команда виводить довідку по використанню команди /price, а також прикріплює pdf файл з повним переліком послуг компанії.

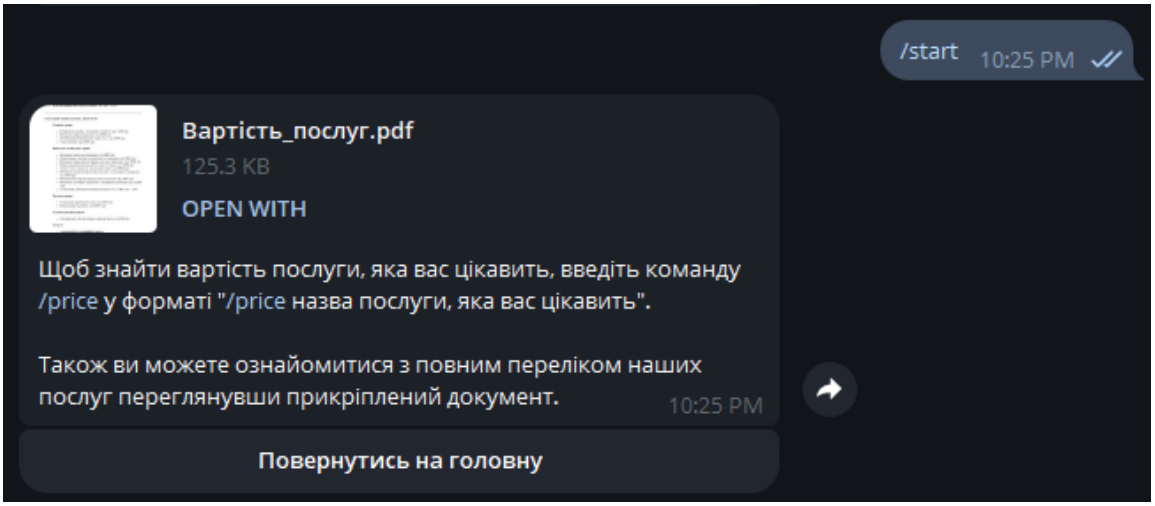

Рисунок 4.22 – Виведення інформації про вартість послуг компанії

На рисунку 4.23 зображено виконання команди /price без передачі значення боту.

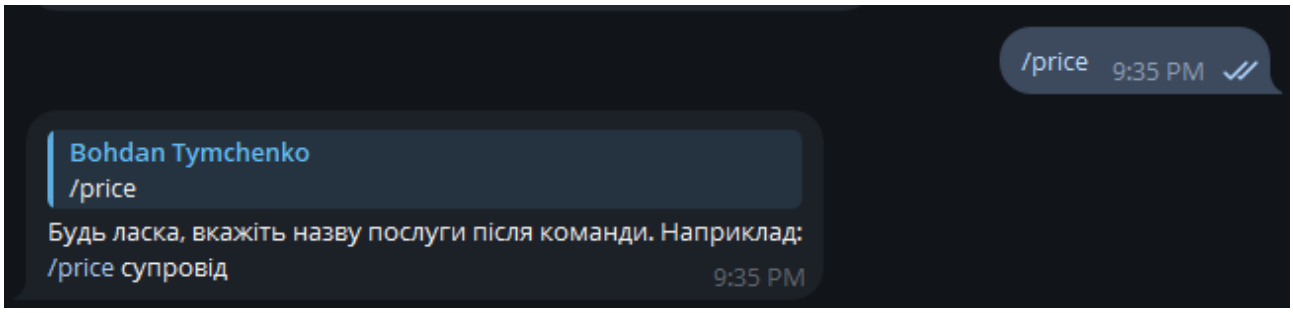

Рисунок 4.23 – Виконання команди /price

На рисунку 4.24 зображено виконання пошуку послуги та її ціни у базі даних після передачі приблизної назви послуги боту.

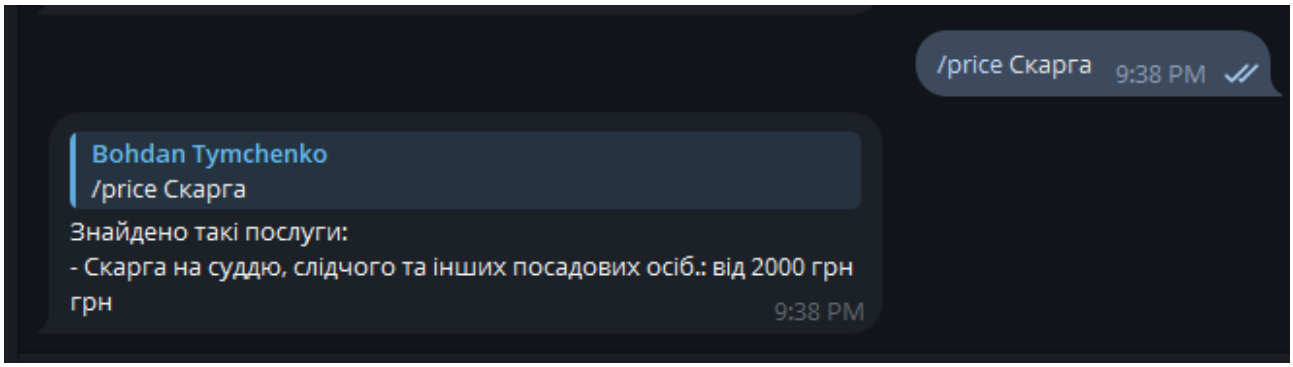

Рисунок 4.24 – Виконання пошуку ціни на послугу

Усі послуги з їх цінами зберігаються у таблиці services у базі даних MySQL. Вигляд таблиці наведено на рисунку 4.25.

Result Grid | Filter Rows: | Edit:

id	service_name	price
1	Розірвання шлюбу, стягнення аліментів	від 1000 грн
2	Розподіл майна подружжя	від 6000 грн
3	Позбавлення батьківських прав та ін.	від 5000 грн
4	Усиновлення	від 6000 грн
5	Визнання права на спадщину (усунення від с...	від 6000 грн
6	Поновлення строків для прийняття спадщини	від 3000 грн
7	Визнання таким, що втратив право користув...	від 5000 грн
8	Відшкодування шкоди (залиття, ДТП тощо),...	від 5000 грн
9	Захист честі, гідності та ділової репутації	від 10000 грн
10	Визнання фізичної особи безвісно відсутньою...	від 3000 грн
11	Встановлення фактів, що мають юридичне з...	від 3000 грн
12	Визнання договору недійсним (дійсним), розі...	від 11000 грн
13	Стягнення дебіторської заборгованості	від 11000 гр...
14	Стягнення заробітної плати, стягнення опла...	від 2500 грн
15	Поновлення на роботі	від 4000 грн
16	Оскарження дій, рішень державних органів ...	від 6000 грн
17	Спори з сімейного права	від 3000 грн
18	Спори з цивільних правовідносин	від 4000 грн
19	Спори з трудового права	від 5000 грн
20	Спори з адміністративного права	від 10000 грн
21	Спори з кримінального права	від 7000 грн
22	Оскарження постанови про притягнення до ...	від 3000 грн
23	Скарга на суддю, слідчого та інших посадов...	від 2000 грн
24	Адвокатський запит про витребування доку...	від 1500 грн
25	Пояснення по справах про адміністративні пр...	від 1500 грн
26	Пояснення по сімейних, цивільних, трудових...	від 1500 грн
27	Заперечення по сімейних, цивільних, трудов...	від 1500 грн
28	Претензія, клопотання та інше	від 1500 грн
29	Договір купівлі-продажу (в тому числі роздрі...	від 1500 грн
30	Договір найму (оренди) житла	від 1500 грн
31	Договір про надання послуг	від 2000 грн
32	Інші договори	від 2000 грн
33	Досудовий аналіз адміністративної справи на...	від 1500 грн
34	Складання окремих процесуальних докумен...	від 2500 грн

Рисунок 4.25 – Таблиця з переліком послуг компанії

На рисунку 4.26 зображено натискання на кнопку «Шаблони документів» в головному меню бота. Дана кнопка представляє з себе посилання на теку на Гугл диску, яка містить архів шаблонів популярних документів.

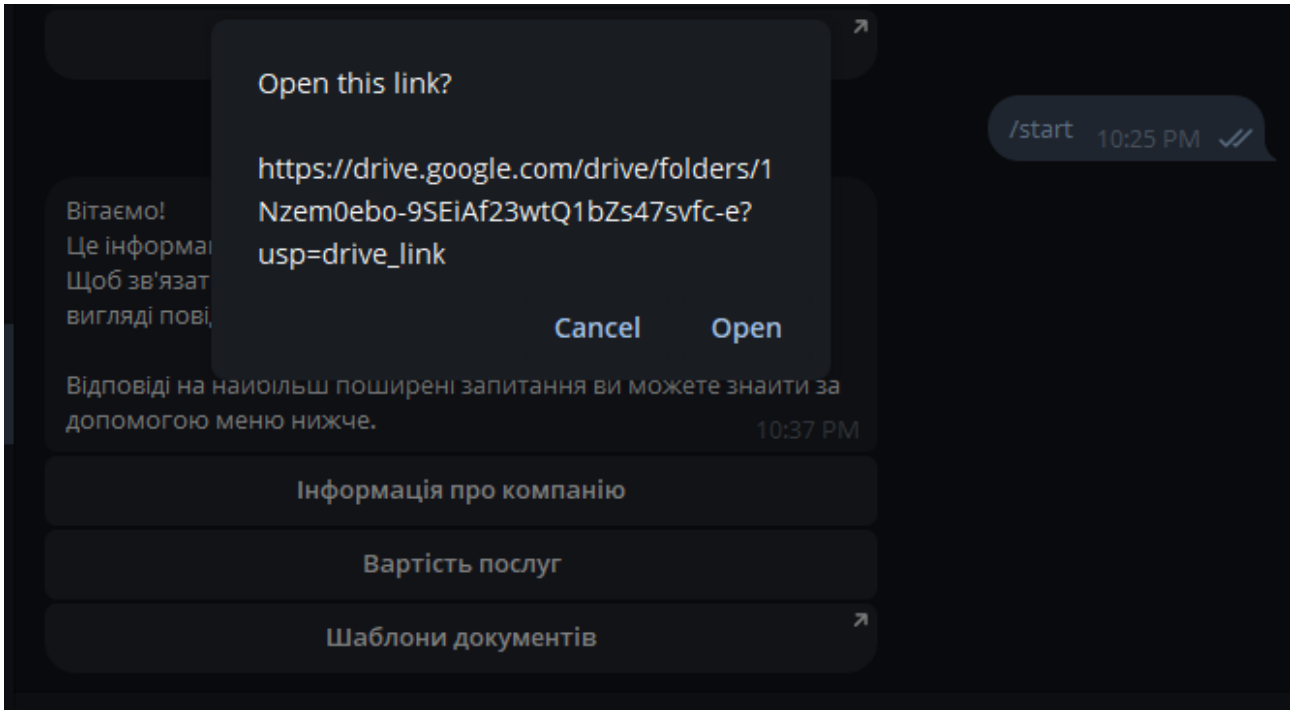

Рисунок 4.26 – Натискання кнопки «Шаблони документів»

На рисунку 4.27 зображено архів шаблонів документів, які можуть використовувати користувачі.

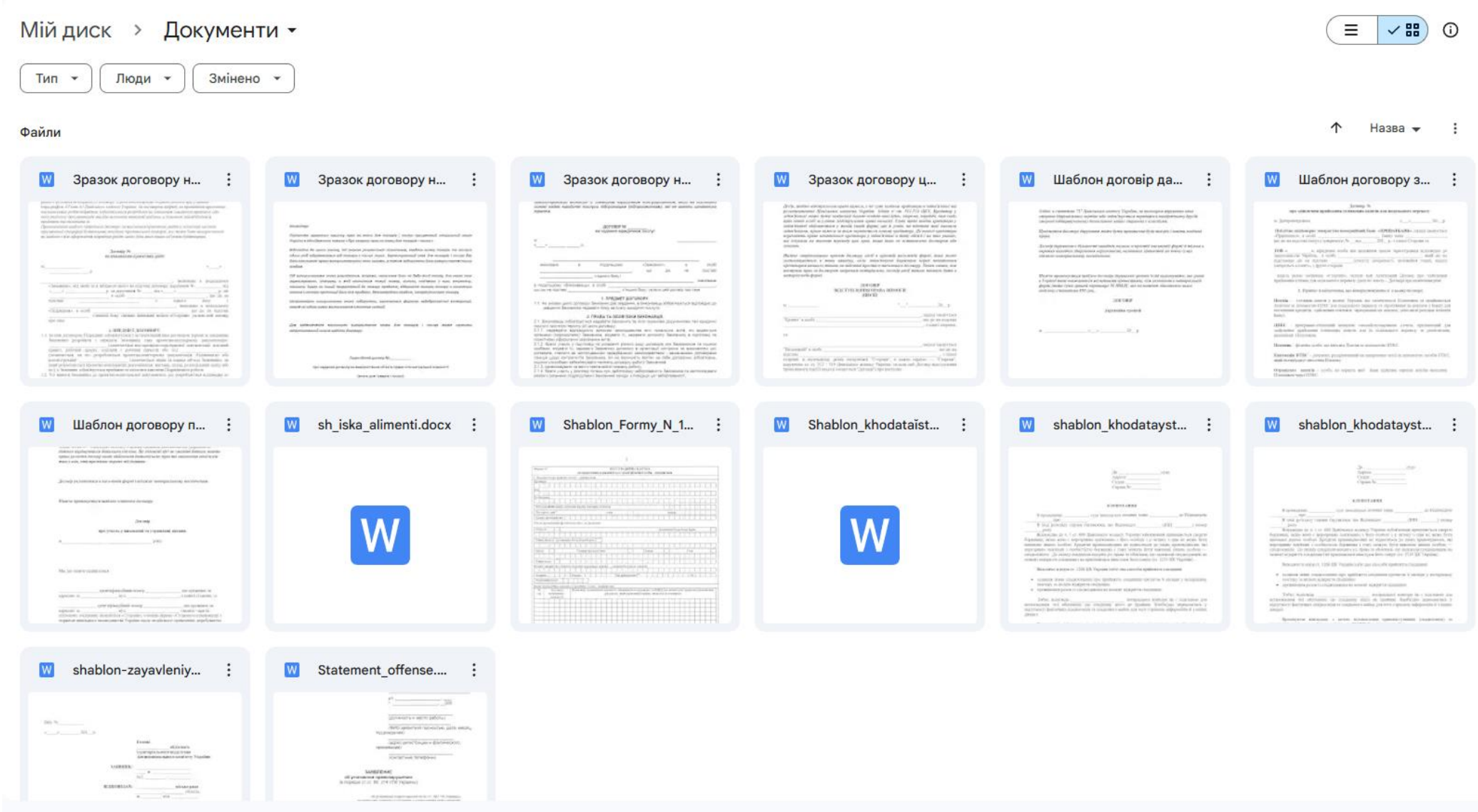

Рисунок 4.27 – Архів шаблонів документів для користувачів

4.7 Очікувані техніко-економічні показники

Розробка Telegram-бота для юридичної компанії "Avaris" дозволяє автоматизувати процес обробки клієнтських запитів, що значно знижує витрати на ручне обслуговування. Очікується скорочення часу реагування на запити клієнтів до кількох секунд, підвищення ефективності роботи операторів підтримки завдяки системі тікетів і зменшення витрат на технічну підтримку до 30%. Інтеграція з базою даних MySQL забезпечує точність і швидкість обробки даних, а також можливість масштабування системи у майбутньому.

4.8 Висновки

У межах розділу "Розробка програмного забезпечення" було реалізовано всі необхідні етапи створення Telegram-бота для автоматизації взаємодії між користувачами, операторами підтримки та базою даних компанії. Сформовано чітку структуру програми, що включає кілька модулів, кожен із яких відповідає за конкретну функціональність, що забезпечує високу модульність, зручність розробки та масштабованість.

Для реалізації програмного забезпечення було обрано сучасні програмні засоби, зокрема, середовище розробки Visual Studio Code, бібліотеки PyTelegramBotAPI, PyMySQL, MySQL Workbench, які забезпечили ефективність процесу розробки. База даних MySQL дозволила організувати централізоване зберігання інформації та її швидку обробку.

Особливу увагу було приділено створенню зручного інтерфейсу для користувачів за допомогою динамічних клавіатур, розроблених у модулі keyboards.py. Функціональність програми доповнена такими модулями, як msg_handler.py для обробки повідомлень, callback_handler.py для роботи з callback-значеннями, а також find_price.py, який дозволяє здійснювати пошук необхідної інформації в базі даних.

Розроблене програмне забезпечення відповідає всім функціональним та додатковим вимогам, сформульованим у процесі синтезу системи. Воно

забезпечує ефективну взаємодію між усіма суб’єктами системи, гнучкість у налаштуванні, а також можливість подальшого розширення функціоналу. Таким чином, завершено ключовий етап розробки програмного забезпечення, що є основою для подальшого тестування, впровадження та експлуатації системи.

5 ЕКСПЕРИМЕНТАЛЬНИЙ РОЗДІЛ

5.1 Постановка завдання експерименту

Метою експерименту є перевірка коректності роботи Telegram-бота в частині функціоналу зворотного зв’язку та оцінка ефективності взаємодії з базою даних компанії під час виконання відповідних операцій. Зокрема, завдання експерименту включає тестування таких аспектів:

1. Функціонал зворотного зв’язку. Необхідно перевірити, як Telegram-бот обробляє запити від користувачів і передає їх до операторів підтримки через чат операторів. Тестування повинно включати:

- Надсилання повідомлень користувачами та перевірку їхнього коректного відображення в чаті операторів.
- Надсилання відповідей операторами та підтвердження доставки цих відповідей користувачам.
- Роботу з повідомленнями різних типів: текстовими, графічними чи з вкладеннями.

2. Оцінка роботи бази даних. Під час експерименту необхідно перевірити стабільність роботи бази даних у процесі виконання запитів, що виникають під час зворотного зв’язку. Основні аспекти:

- Коректність запису даних користувачів до бази даних.

3. Взаємодія компонентів системи. Особливу увагу слід приділити тестуванню взаємодії між Telegram-ботом, базою даних та чатами операторів. Важливо підтвердити, що всі компоненти системи працюють узгоджено.

Експеримент має на меті виявити можливі недоліки в реалізації функціоналу зворотного зв’язку, а також оцінити загальну ефективність роботи програмного забезпечення. Отримані результати будуть використані для вдосконалення системи перед її остаточним впровадженням.

5.2 Опис умов тестування програмного забезпечення

Тестування програмного забезпечення Telegram-бота проводилося в наступному середовищі: у якості серверного обладнання використовувався комп'ютер із наступними технічними характеристиками:

- процесор із 12 логічними ядрами, що забезпечує високу продуктивність обчислень;
- 16 ГБ оперативної пам'яті, достатньої для виконання багатозадачних операцій і стабільної роботи програмного забезпечення;
- NVMe-накопичувач, який гарантує швидку обробку операцій введення-виведення і забезпечує мінімальні затримки в роботі бази даних.

Для тестування функціоналу Telegram-бота використовувався десктопний клієнт Telegram, який дозволив оцінити взаємодію з ботом з точки зору користувача. Цей клієнт забезпечує доступ до всіх функцій бота та можливість перевірки коректності роботи основних команд і клавіатур.

База даних перевірялася за допомогою MySQL Workbench, що дозволило здійснювати моніторинг збереження і зміни даних у реальному часі. Цей інструмент також використовувався для верифікації коректності SQL-запитів, що формуються програмним забезпеченням, і аналізу роботи бази даних під навантаженням.

Тестування проводилося з урахуванням потенційного навантаження на систему, включаючи одночасну роботу кількох користувачів і операцій взаємодії із базою даних. Це дозволило ідентифікувати можливі недоліки та забезпечити високу якість програмного забезпечення перед його впровадженням.

5.3 Результати проведених досліджень

5.3.1 Експеримент з пересиланням повідомлень між чатами

Початковим етапом дослідження виконаємо запуск бота та відправку повідомлення користувачем до особистого чату з ботом. Після цього бот повинен

надіслати відповідні повідомлення до чату підтримки, де їх зможуть переглядати оператори підтримки та надавати на них відповіді. Відправку повідомлень користувачем зображено на рисунку 5.1.

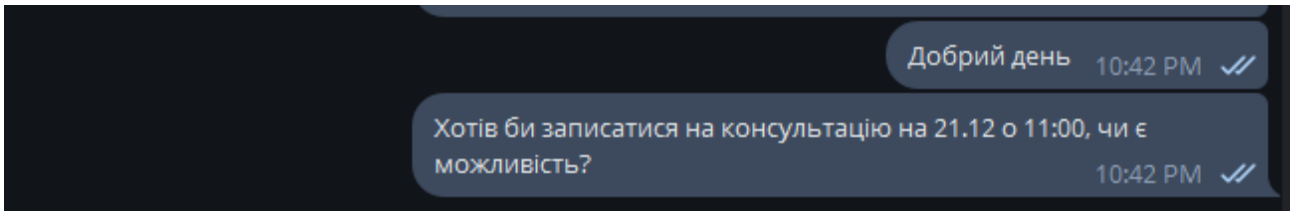

Рисунок 5.1 – Відправка повідомлень боту користувачем

На рисунку 5.2 зображено отримання повідомлень користувача у чаті підтримки. Разом з повідомленням бот надсилає посилання на профіль користувача та його ID.

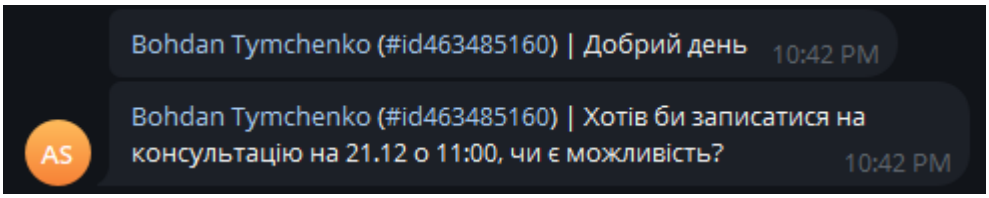

Рисунок 5.2 – Пересилання повідомлень до чату підтримки

Після надсилання повідомлення користувачем до бази даних додається запис, де міститься інформація про користувача. Тут вказано його ID, наявність відкритих квитків (повідомлень, які не отримали відповіді), статус користувача (заблоковано або не заблоковано), кількість відправлених повідомлень без відповіді для запобігання спаму (open_ticket_spam), також тут міститься посилання на відкритий квиток користувача та час відправки цього повідомлення користувачем.

	userid	open_ticket	banned	open_ticket_spam	open_ticket_link	open_ticket_time
▶	463485160	1	0	2	https://t.me/c/2367665326/142	2024-12-17 20:47:41

Рисунок 5.3 – Запис користувача в базі даних

Після того, як повідомлення користувача було переадресовано до чату підтримки оператори мають змогу відповісти на них просто зробивши відповідь (reply) на дане повідомлення.

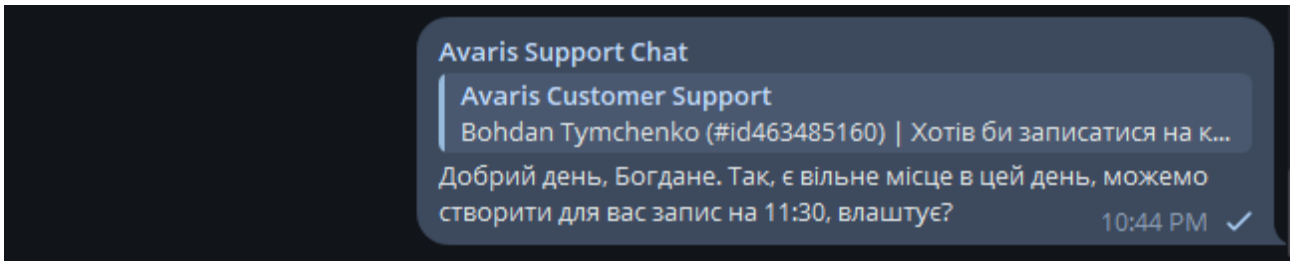

Рисунок 5.4 – Відповідь оператором на повідомлення користувача

На рисунку 5.5 зображено отримання користувачем відповіді від оператора.

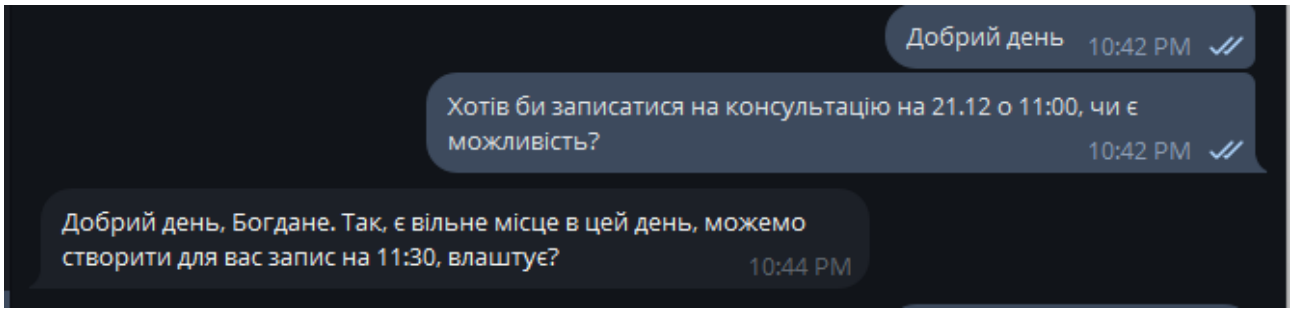

Рисунок 5.5 – Отримання користувачем відповіді

5.3.2 Експеримент з запобіганням спаму від користувачів

Наступним кроком в роботі необхідно перевірити функціонал забезпечення захисту від спаму з боку користувачів бота. Кількість повідомлень, які користувач може відправити до отримання попередження задається у файлі конфігурації. На рисунку 5.6 зображено налаштування кількості повідомлень до попередження про спам на 5 повідомлень.

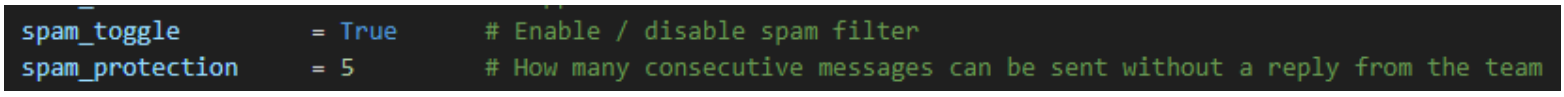

Рисунок 5.6 – Налаштування спам-фільтру

На рисунку 5.7 зображено відправку користувачем декількох повідомлень без отримання відповіді. Бот успішно попереджує користувача про те, що він зможе відправити всього 1 додаткове повідомлення до того часу, поки не отримає відповіді.

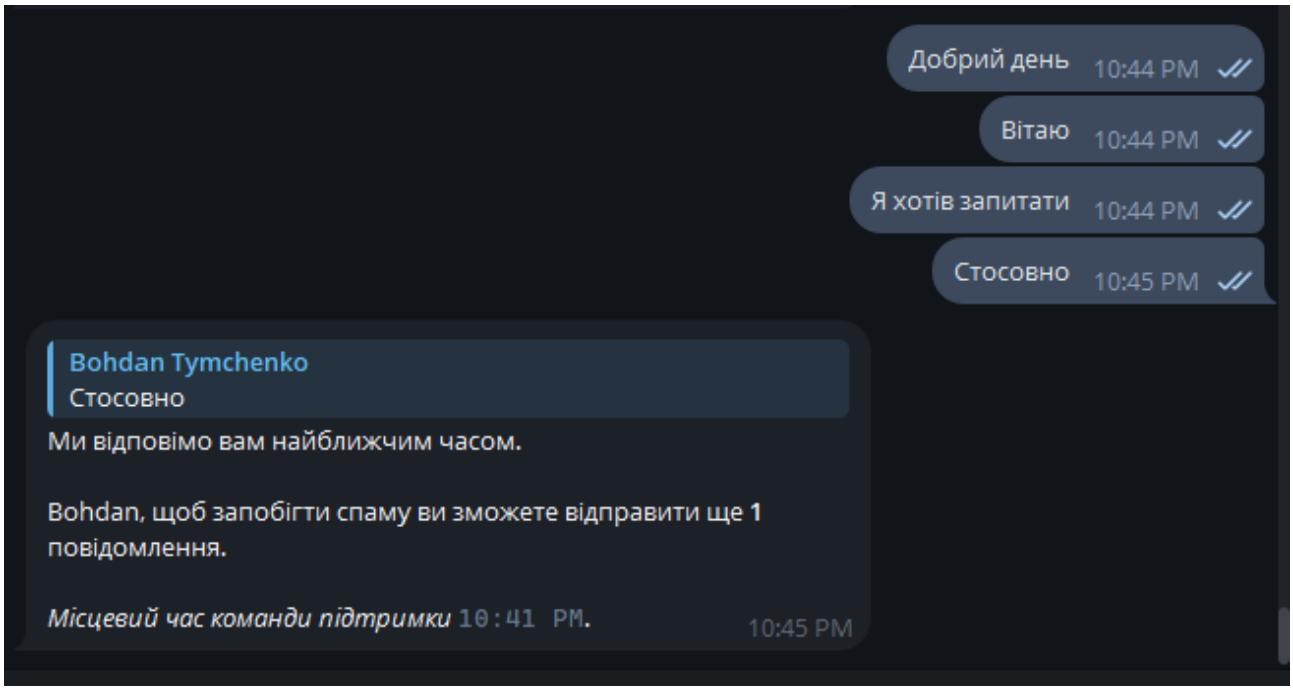

Рисунок 5.7 – Спроба спаму користувачем

Якщо користувач після відправки останнього повідомлення спробує знову звернутися до підтримки, то отримає сповіщення про те, що його повідомлення більше не пересилаються до тих пір, поки він не отримає відповіді від підтримки.

Спрацювання спам-фільтру обумовлено записом до бази даних кількості повідомлень, які користувач надіслав без відповіді. На рисунку 5.8 показано, що значення open_ticket_spam в користувача змінилося на 6. Через це повідомлення користувача втрачають змогу пересилатися до чату підтримки.

	userid	open_ticket	banned	open_ticket_spam	open_ticket_link	open_ticket_time
▶	463485160	1	0	6	https://t.me/c/2367665326/146	2024-12-17 20:47:41

Рисунок 5.8 – Зміна значення open_ticket_spam

На рисунку 5.9 зображено спробу користувача надіслати нові повідомлення, не отримавши відповіді на попередні.

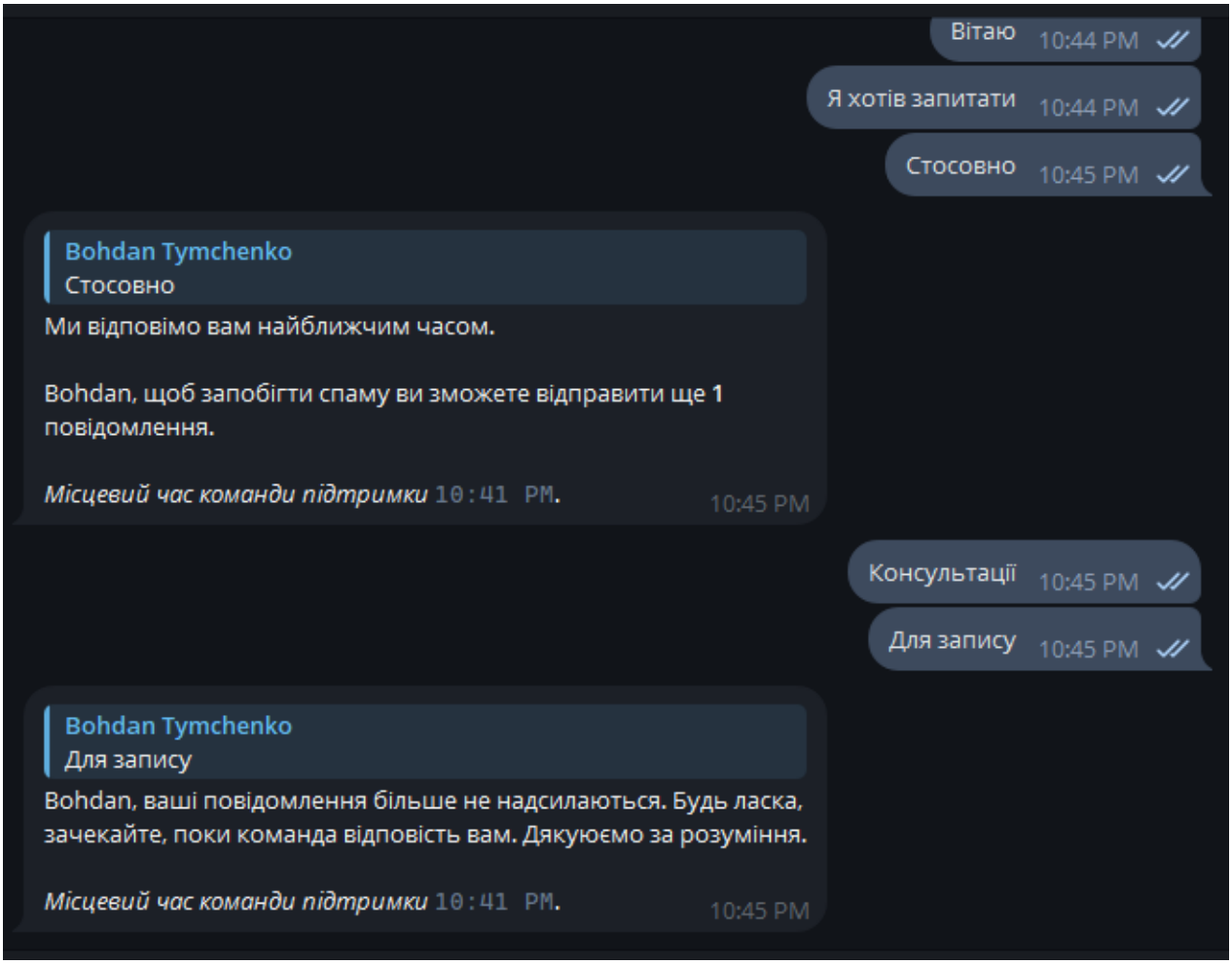

Рисунок 5.9 – Спроба користувачем відправити повідомлення після спрацювання спам-фільтру

5.3.3 Перевірка роботи бота з різними типами повідомлень

На даному етапі експерименту необхідно перевірити взаємодію бота з різними видами повідомлень. Бот повинен надсилати до чату підтримки текстові повідомлення, медіа файли та документи. Бот повинен ігнорувати інші види повідомлень, включаючи голосові повідомлення, стікери та інші.

На рисунку 5.10 зображено надсилання ботом файлу до чату підтримки.

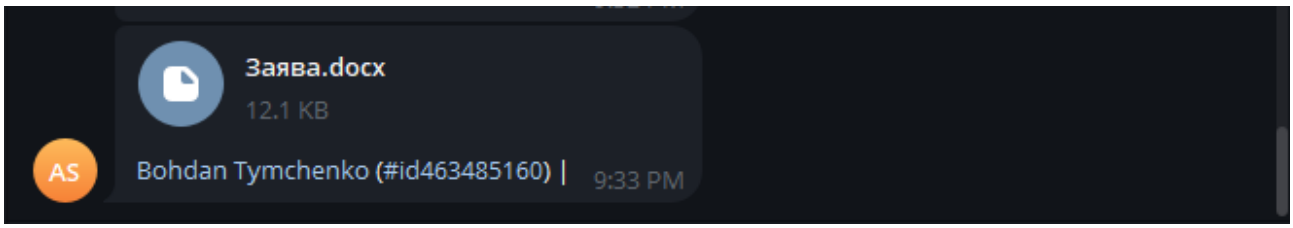

Рисунок 5.10 – Пересилання ботом документу

На рисунку 5.11 зображено відправку користувачем фото, стікера та голосового повідомлення. На рисунку 5.12 бачимо, що бот переслав до чату підтримки тільки фото, ігноруючи стікер та голосове повідомлення.

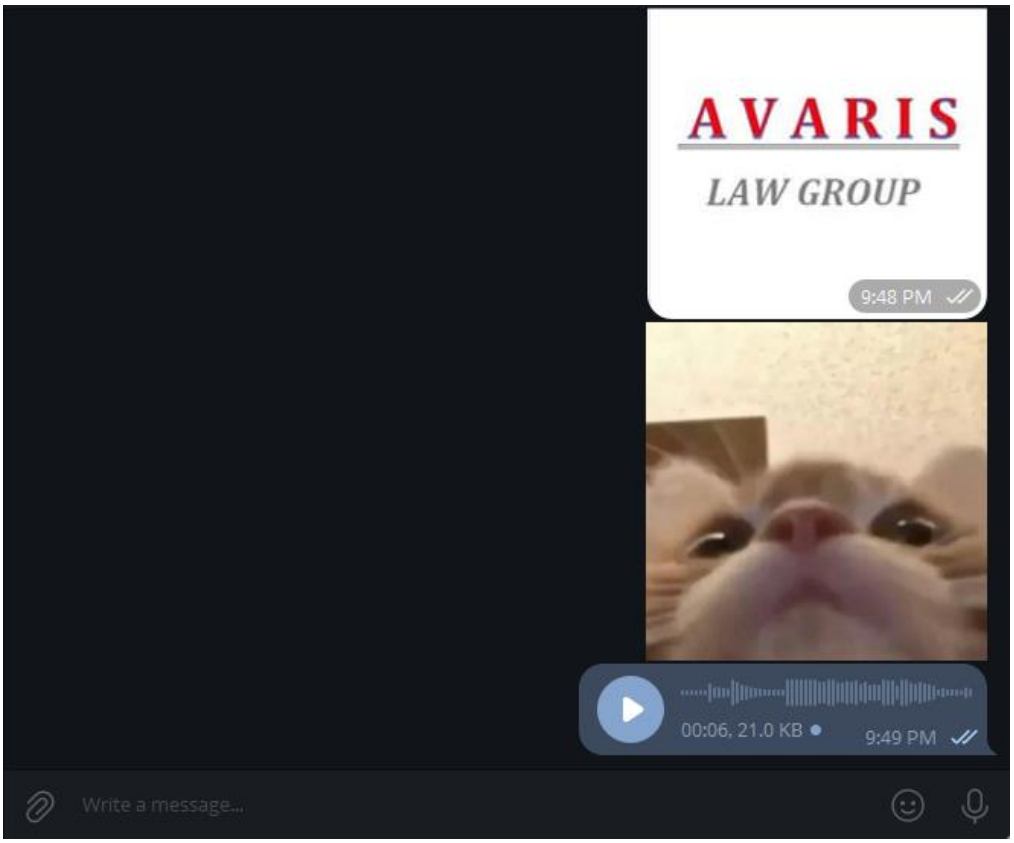

Рисунок 5.11 – Надсилання користувачем фото, наліпки та голосового повідомлення

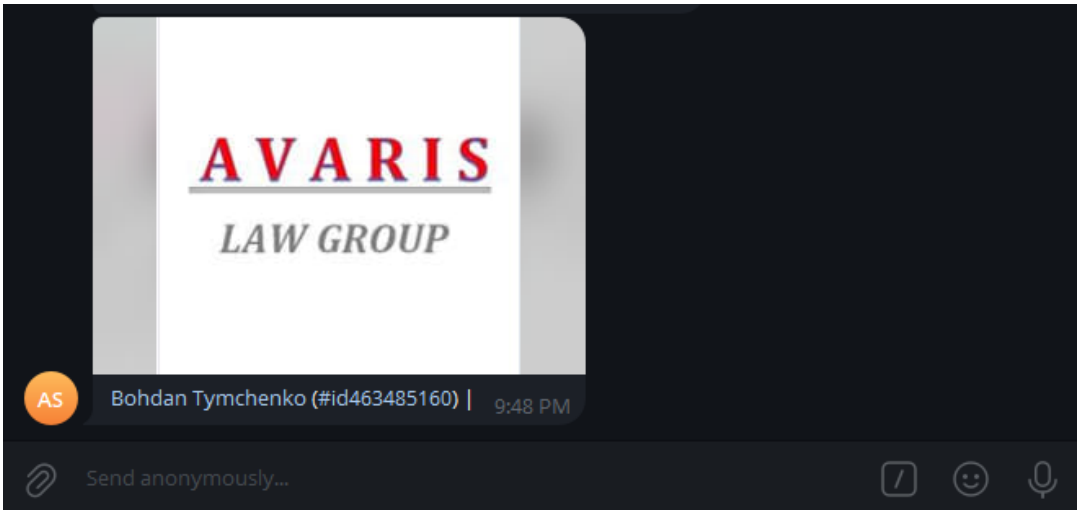

Рисунок 5.12 – Пересилання ботом фото до чату підтримки, ігнорування непідтримуваних повідомлень

5.3.4 Експеримент пошуком відкритих квитків

Для операторів підтримки доступна команда /tickets, при виконанні якої повинен виводитися перелік відкритих квитків з вказаним користувачем, який надіслав повідомлення та посиланням на його повідомлення. Також повинна виводитися інформація про те, скільки часу тому користувач надіслав повідомлення. На рисунку 5.13 зображено перевірку дієздатності команди /tickets.

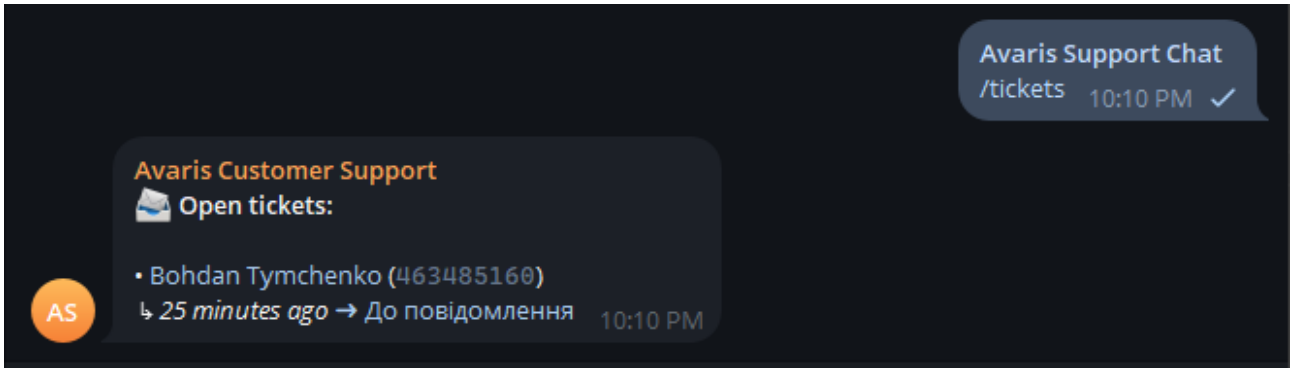

Рисунок 5.13 – Отримання списку відкритих квитків

Якщо наразі відкриті квитки відсутні, то оператор отримає про це сповіщення від бота (рисунок 5.14).

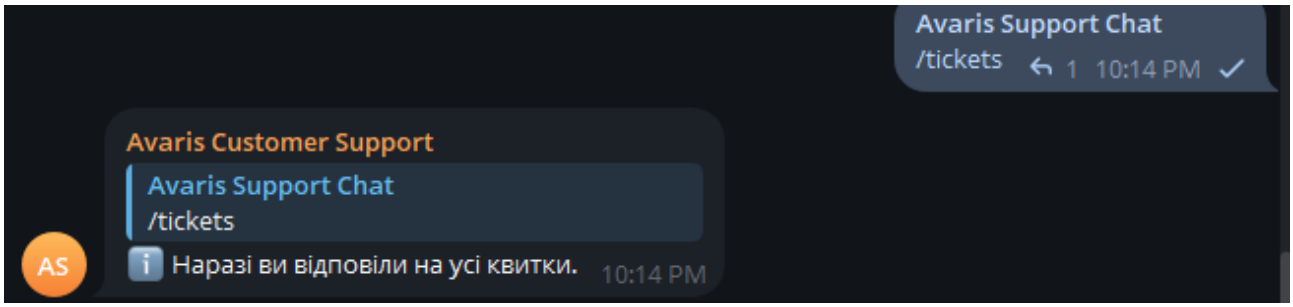

Рисунок 5.14 – Сповіщення про відсутність відкритих квитків

Також оператори підтримки мають змогу за потреби вручну закрити квиток. Перевірку даної команди наведено на рисунку 5.15.

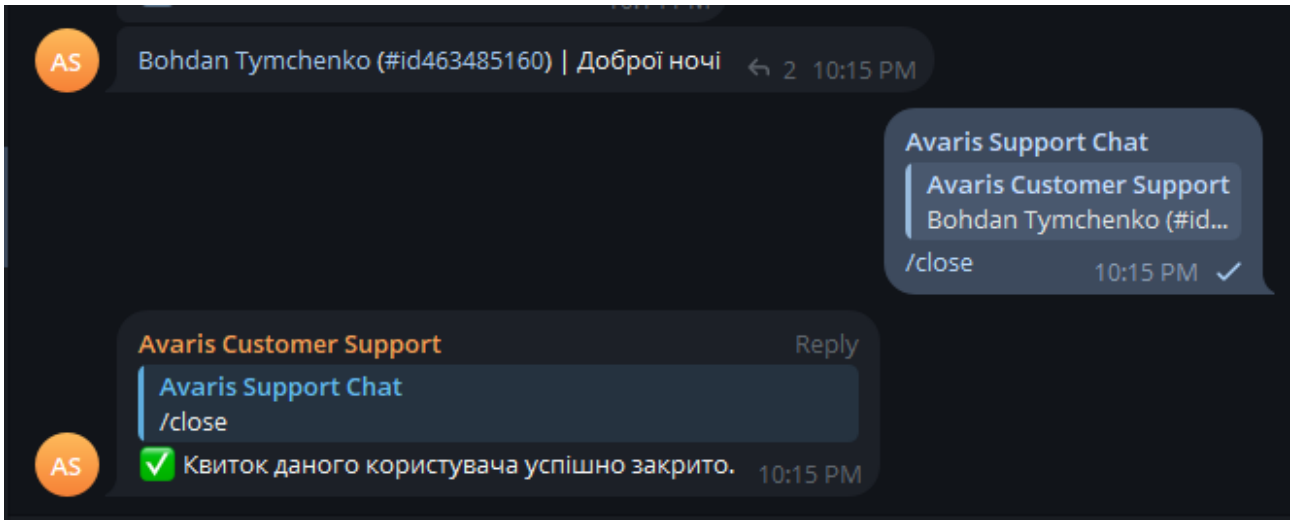

Рисунок 5.15 – Примусове закривання квитка користувача

5.3.5 Експеримент з блокуванням користувачів

Оператори підтримки мають змогу блокувати користувачів за допомогою команди /ban. При цьому в базі даних в користувача з'являється відповідний статус і його повідомлення втрачають змогу пересилатися до чату підтримки.

На рисунку 5.16 зображено блокування користувача оператором.

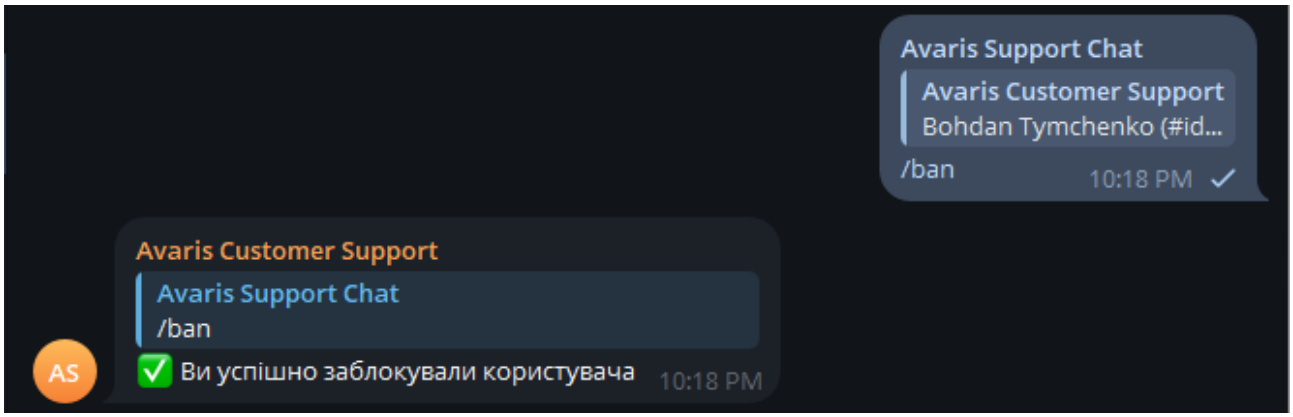

Рисунок 5.16 – Блокування користувача

Після виконання блокування користувача можемо спостерігати, як в базі даних в нього з'являється відповідний статус (поле banned приймає значення 1). Після цього повідомлення користувача не будуть спрямовуватися до чату підтримки.

	userid	open_ticket	banned	open_ticket_spam	open_ticket_link	open_ticket_time
▶	463485160	0	1	1	https://t.me/c/2367665326/176	2024-12-17 22:15:32
	347738012	0	0	1	https://t.me/c/2367665326/117	2024-11-30 16:13:54

Рисунок 5.17 – Зміна статусу блокування користувача

Для розблокування користувача використовується команда /unban (рисунок 5.18).

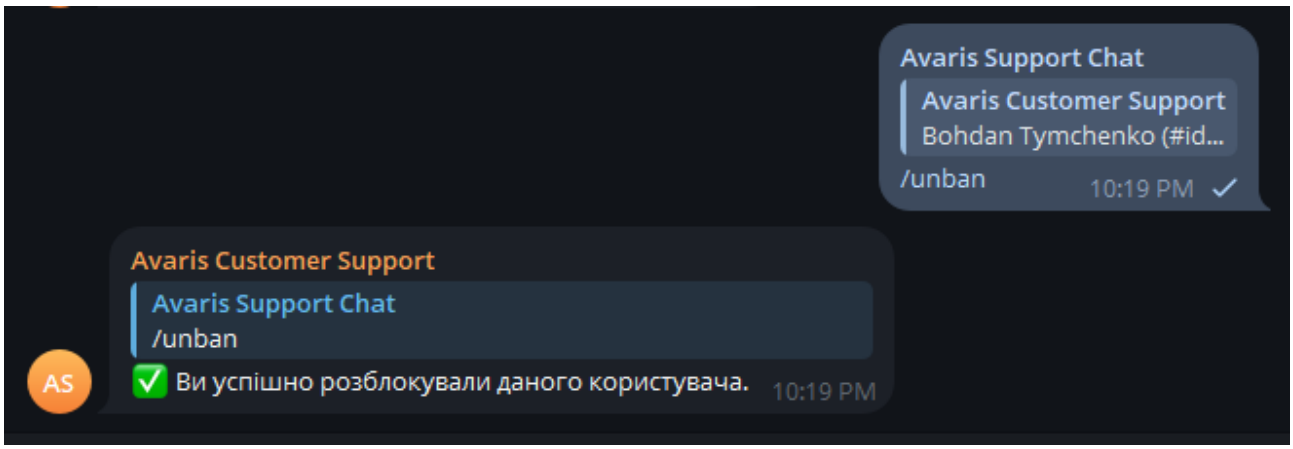

Рисунок 5.18 – Розблокування користувача

5.4 Висновки

У ході експериментального дослідження було проведено тестування функціоналу Telegram-бота, зокрема перевірку зворотного зв'язку та роботи бази даних. Тестування здійснювалося в умовах, максимально наближених до реальних, з використанням сучасного обладнання, що забезпечило високу продуктивність і точність отриманих результатів.

Було підтверджено коректність виконання основних функцій системи: надсилання та обробки повідомлень користувачів і операторів, взаємодії з базою даних для отримання та збереження інформації. Оцінка показала, що бот належним чином виконує команди, забезпечує швидкий доступ до необхідних даних та підтримує ефективний зв'язок між користувачами і операторами підтримки.

Під час тестування не знайдено явні недоліки, що підтверджує високу готовність програмного забезпечення до подальшого впровадження. Таким

чином, результати експерименту засвідчили функціональну надійність системи, стабільну роботу бази даних та відповідність поставленим вимогам.

ВИСНОВКИ

У кваліфікаційній роботі було виконано повний цикл розробки системи автоматизації консультування за допомогою Telegram-бота. Було проведено аналіз сучасних платформ для створення чат-ботів та обґрунтовано вибір технологій для реалізації проєкту. Основою розробки стала мова програмування Python із використанням бібліотек PyTelegramBotAPI, pymysql, а також бази даних MySQL для зберігання і обробки інформації.

Під час роботи було визначено вимоги до функціонального та технічного забезпечення системи, розроблено архітектуру програми та описано структуру її модулів. Реалізовано функціонал, що забезпечує зворотний зв'язок між користувачами та операторами підтримки, надання необхідної інформації про компанію, а також доступ до шаблонів документів.

Експериментальне тестування підтвердило коректність роботи основних функцій системи, стабільну взаємодію з базою даних та відповідність розробленого програмного забезпечення заданим вимогам. Були створені умови для подальшої інтеграції бота в робочий процес компанії, що дозволить підвищити ефективність обслуговування клієнтів та оптимізувати роботу служби підтримки.

Таким чином, результати роботи продемонстрували досягнення поставленої мети, вирішення поставлених завдань та підтвердили практичну цінність розробленого програмного забезпечення для автоматизації процесів консультування.

ПЕРЕЛІК ПОСИЛАНЬ

1. PyTelegramBotAPI – [Електронний ресурс] URL: https://pytba.readthedocs.io/en/latest/ (дата звернення: 17.12.2024)

2. MySQL Documentation – [Електронний ресурс] URL: https://dev.mysql.com/doc/ (дата звернення: 18.12.2024)

3. Telegram Bot API – [Електронний ресурс] URL: https://core.telegram.org/bots/api (дата звернення: 14.12.2024)

4. Цвіркун Л.І. Атестація здобувачів вищої освіти. Методичні рекомендації до виконання кваліфікаційної роботи магістра здобувачами галузі знань 12 Інформаційні технології спеціальності 123Комп'ютернаінженерія / Л.І. Цвіркун, В.В. Гнатушенко, С.М. Ткаченко ; М-во освіти і науки України, Нац. техн. ун-т «Дніпровська політехніка». –Дніпро: НТУ«ДП», 2024. – 54 с (дата звернення: 14.12.2024)

5. Компанія Avaris – [Електронний ресурс] URL: https://avaris.com.ua/ (дата звернення: 18.12.2024)

6. Панасюк, О.В., Іваненко, Р.С. (2021). Особливості створення чат-ботів для автоматизації бізнес-процесів. Вісник НТУУ "КПІ", 47(2), 76–83. (дата звернення: 18.12.2024)

7. Create a Telegram Ticket Bot for Customer Support – [Електронний ресурс] – https://www.mava.app/blog/how-to-create-a-telegram-ticket-bot-for-customer-support (дата звернення: 17.12.2024)

8. PyMySQL documentation – [Електронний ресурс] URL: https://pymysql.readthedocs.io/en/latest/ (дата звернення: 12.12.2024)

9. Bui, Q., et al. (2020). "Chatbots as a new customer communication channel: An application study." Journal of Business Research, 117, 852-859 (дата звернення: 10.12.2024)

10. Wang, Y., et al. (2021). "Legal Chatbots: Reducing Consultation Time and Improving Accuracy through Legislative Database Integration." International

Journal of Law and Information Technology, 29(2), 123-140. (дата звернення: 10.12.2024)

11. Gartner. (2020). "Chatbots Will Appeal to Modern Workers." – [Електронний ресурс] URL: http://surl.li/qwfdox (дата звернення: 10.12.2024)

ДОДАТОК А

Текст програми Telegram-бота

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
“ДНІПРОВСЬКА ПОЛІТЕХНІКА”

ТЕКСТ ПРОГРАМИ ЧАТ-БОТА ДЛЯ КОНСУЛЬТАЦІЙ ЮРИДИЧНОЇ ФІРМИ «AVARIS»

Текст програми
804.02070743.24019-01 12 01
Листів 22

2024

АНОТАЦІЯ

Даний додаток містить програмний код Telegram-бота, призначеного для надання зворотного зв’язку для клієнтів юридичної компанії «Avaris».

Дана програма призначена для автоматизованого надання інформації про компанію користувачам, забезпечення можливості зворотного зв’язку з операторами підтримки через систему тікетів, також бот надає доступ до архіву шаблонів документів. Бот забезпечує ефективну роботу системи зворотного зв’язку, дозволяючи компанії швидко відповідати клієнтам.

Дана програма написана мовою Python з використанням бібліотеки pyTelegramBotAPI, реляційної бази даних MySQL та бібліотеки pymysql. Обробка клієнтських запитів та повідомлень виконується веб сервером компанії. Вся розробка проводилася у середовищі Visual Studio Code.

ЗМІСТ

1 Код ключових модулів створеного бота

1.1 main.py

```
import config
from resources import mysql_handler as mysql
from resources import msg_handler as msg
from resources import keyboards as kb
from resources import callback_handler
from telebot.types import Message
from resources import find_price
import telebot
from datetime import datetime
import arrow

bot = telebot.TeleBot(config.token)

mysql.createTables



# Callback Handlers
@bot.callback_query_handler(func=lambda call: True)
def callback_inline(call):
    callback_handler.callback_inline(call)



# Start Command
@bot.message_handler(commands=['start'])
def start(message):
    if message.chat.type == 'private':
```

```
            bot.send_message(message.chat.id,

config.text_messages['start'].format(message.from_user.first_name),
                             parse_mode='html',          disable_web_page_preview=True,
reply_markup=kb.main)
            mysql.start_bot(message.chat.id)
        else:
            bot.reply_to(message, 'Будь ласка, надішліть мені особисте повідомлення, якщо ви хочете зв\'язатися з командою підтримки.')

    # FAQ Command
    @bot.message_handler(commands=['faq'])
    def start(message):
        if message.chat.type == 'private':
            bot.reply_to(message,                    config.text_messages['faqs'],
parse_mode='Markdown', disable_web_page_preview=True)
        else:
            pass

    @bot.message_handler(commands=['price'])
    def handle_price_command(message: Message):
        find_price.handle_price_command(message)

    # Get All Open Tickets
    @bot.message_handler(commands=['tickets', 't'])
    def ot_handler(message):
        if message.chat.id == config.support_chat:
            if not mysql.open_tickets:
                bot.reply_to(message, "ℹ Наразі ви відповіли на усі квитки.")
```

```
            return

        ot_msg = '📨 *Open tickets:*\n\n'
        for user in mysql.open_tickets:
            bot.send_chat_action(message.chat.id, 'typing')
            ot_link = mysql.user_tables(int(user))['open_ticket_link']

            now = arrow.now()
            diff = datetime.now() - mysql.user_tables(int(user))['open_ticket_time']
            diff.total_seconds() / 3600  # seconds to hour
            time_since_secs = float(diff.seconds)
            time_since = now.shift(seconds=-time_since_secs).humanize()

            # Bring attention to 1 day old tickets
            if time_since_secs > config.open_ticket_emoji * 3600:
                alert = ' ↳ ⚠️ '
            else:
                alert = ' ↳ '

            ot_msg += "• [{0}{1}](tg://user?id={2}) (`{2}`)\n{5}_{3}_ [➜ До повідомлення]({4})\n".format(
                bot.get_chat(int(user)).first_name,
                ' {0}'.format(bot.get_chat(int(user)).last_name) if bot.get_chat(int(user)).last_name else '',
                int(user), time_since, ot_link, alert)

        bot.send_message(message.chat.id, ot_msg, parse_mode='Markdown')
    else:
        pass
```

```
# Close a ticket manually
@bot.message_handler(commands=['close', 'c'])
def ot_handler(message):
    if message.chat.id == config.support_chat:
        if message.reply_to_message and '(#id' in message.reply_to_message.text:
            bot.send_chat_action(message.chat.id, 'typing')
            user_id = int(message.reply_to_message.text.split('(#id')[1].split(')')[0])
            ticket_status = mysql.user_tables(user_id)['open_ticket']

            if ticket_status == 0:
                bot.reply_to(message, '❌ В даного користувача немає відкритих квитків')
            else:
                # Reset Open Tickets as well as the Spamfilter
                mysql.reset_open_ticket(user_id)
                bot.reply_to(message, '☑ Квиток даного користувача успішно закрито.')
        else:
            bot.reply_to(message, '**i** Вам необхідно відповісти на повідомлення')
    else:
        pass


# Get Banned User
@bot.message_handler(commands=['banned'])
def ot_handler(message):
    if message.chat.id == config.support_chat:
```

```
            if not mysql.banned:
                bot.reply_to(message, "ℹ Наразі заблоковані користувачі відсутні.")
                return

            ot_msg = '⛔ *Заблоковані користувачі:*\n\n'
            for user in mysql.banned:
                bot.send_chat_action(message.chat.id, 'typing')
                ot_link = mysql.user_tables(int(user))['open_ticket_link']

                ot_msg += "• [{0}{1}](tg://user?id={2}) (`{2}`)\n[➜ Go to last msg]({3})\n".format(
                    bot.get_chat(int(user)).first_name,
                    ' {0}'.format(bot.get_chat(int(user)).last_name) if bot.get_chat(int(user)).last_name else '',
                    int(user), ot_link)

            bot.send_message(message.chat.id, ot_msg, parse_mode='Markdown')
        else:
            pass


    # Ban User
    @bot.message_handler(commands=['ban'])
    def ot_handler(message):
        try:
            if message.chat.id == config.support_chat:
                if message.reply_to_message and '(#id' in msg.msgCheck(message):
                    user_id = msg.getUserID(message)
                    banned_status = mysql.user_tables(user_id)['banned']
```

```
                if banned_status == 1:
                    bot.reply_to(message, '❌ Даного користувача вже заблоковано')
                else:
                    mysql.ban_user(user_id)
                    try:
                        # Reset Open Tickets as well as the Spamfilter
                        mysql.reset_open_ticket(user_id)
                    except Exception as e:
                        pass
                    bot.reply_to(message, '☑ Ви успішно заблокували користувача')

            elif msg.getReferrer(message.text):
                user_id = int(msg.getReferrer(message.text))
                banned_status = mysql.user_tables(user_id)['banned']
                if banned_status == 1:
                    bot.reply_to(message, '❌ Даного користувача вже заблоковано')
                else:
                    mysql.ban_user(user_id)
                    try:
                        # Reset Open Tickets as well as the Spamfilter
                        mysql.reset_open_ticket(user_id)
                    except Exception as e:
                        pass
```

```
                bot.reply_to(message, '☑ Ви успішно заблокували користувача')
            else:
                bot.reply_to(message, '**i** Вам необхідно відповісти на повідомлення користувача або вказати його "User ID"',
                             parse_mode='Markdown')
        except TypeError:
            bot.reply_to(message, '❌ Ви впевнені, що я взаємодіяв з даним користувачем до цього?')


    # Un-ban Useer
    @bot.message_handler(commands=['unban'])
    def ot_handler(message):
        try:
            if message.chat.id == config.support_chat:
                if message.reply_to_message and '(#id' in msg.msgCheck(message):
                    user_id = msg.getUserID(message)
                    banned_status = mysql.user_tables(user_id)['banned']

                    if banned_status == 0:
                        bot.reply_to(message, '❌ Даного користувача не було заблоковано')
                    else:
                        mysql.unban_user(user_id)
                        bot.reply_to(message, '☑ Ви успішно розблокували даного користувача.')
```

```
            elif msg.getReferrer(message.text):
                user_id = int(msg.getReferrer(message.text))
                banned_status = mysql.user_tables(user_id)['banned']

                if banned_status == 0:
                    bot.reply_to(message, '❌ Даного користувача не було заблоковано')
                else:
                    mysql.unban_user(user_id)
                    bot.reply_to(message, '☑ Ви успішно розблокували даного користувача.')
            else:
                bot.reply_to(message, 'ℹ Вам необхідно відповісти на повідомлення користувача або вказати його "User ID"',
                             parse_mode='Markdown')
        except TypeError:
            bot.reply_to(message, '❌ Ви впевнені, що я взаємодіяв з даним користувачем до цього?')


    # Message Forward Handler (User - Support)
    @bot.message_handler(func=lambda message: message.chat.type == 'private', content_types=['text', 'photo', 'document'])
    def echo_all(message):
        while True:
            mysql.start_bot(message.chat.id)
            user_id = message.chat.id
            message = message
            banned = mysql.user_tables(user_id)['banned']
```

```
            ticket_status = mysql.user_tables(user_id)['open_ticket']
            ticket_spam = mysql.user_tables(user_id)['open_ticket_spam']

            if banned == 1:
                return
            elif msg.spam_handler_warning(bot, user_id, message):  # First spam warning
                return
            elif msg.spam_handler_blocked(bot, user_id, message):  # Final spam warning // user cant send messages anymore
                return
            elif ticket_status == 0:
                mysql.open_ticket(user_id)
                continue
            else:
                msg.fwd_handler(user_id, bot, message)
                return


        # Message Forward Handler (Support - User)
        @bot.message_handler(content_types=['text', 'photo', 'document'])
        def echo_all(message):
            while True:
                try:
                    try:
                        user_id = msg.getUserID(message)
                        message = message
                        text = message.text
                        msg_check = msg.msgCheck(message)
```

```
                ticket_status = mysql.user_tables(user_id)['open_ticket']
                banned_status = mysql.user_tables(user_id)['banned']

                if banned_status == 1:
                    # If User is banned - un-ban user and sent message
                    mysql.unban_user(user_id)
                    bot.reply_to(message, 'ℹ *FYI: That user was banned.*\n_Un-banned and sent message!_',
                                 parse_mode='Markdown')

                elif ticket_status == 1:
                    # Reset Open Tickets as well as the Spamfilter
                    mysql.reset_open_ticket(user_id)
                    continue

                else:
                    if message.reply_to_message and '(#id' in msg_check:
                        msg.snd_handler(user_id, bot, message, text)
                        return

            except telebot.apihelper.ApiException:
                bot.reply_to(message, '❌ В мене не вийшло надіслати повідомлення, можливо користувач заблокував мене.')
                return

        except Exception as e:
            bot.reply_to(message, '❌ Перевірте правильність написання команди.')
            return
```

```
print("Telegram Support Bot started...")
bot.polling()
```

1.2 find_price.py

```
import pymysql
import telebot
from telebot.types import Message
import config
from resources import mysql_handler as mysql
from resources import msg_handler as msg
from resources import keyboards as kb
from resources import callback_handler

import telebot
from datetime import datetime
import arrow

bot = telebot.TeleBot(config.token)

# Функція для пошуку послуги в базі даних
def find_service_in_db(service_name):
    try:
        connection = mysql.getConnection()
        cursor = connection.cursor()

        # Пошук за схожими назвами (чутливий до регістру)
        query = """
```

```
            SELECT service_name, price
            FROM services
            WHERE service_name LIKE %s
            """
            cursor.execute(query, (f"%{service_name}%",))
            results = cursor.fetchall()
            return results
        except pymysql.connect.Error as e:
            print(f"Error connecting to MySQL: {e}")
            return None
        finally:
            if connection:
                cursor.close()
                connection.close()

    # Обробник команди /price

    def handle_price_command(message: Message):
        # Отримання тексту послуги, який ввів користувач
        user_query = message.text[len('/price '):].strip()

        if not user_query:
            bot.reply_to(message, "Будь ласка, вкажіть назву послуги після команди. Наприклад:\n/price супровід")
            return

        # Пошук послуги в базі даних
        results = find_service_in_db(user_query)
```

```
    # Відповідь користувачеві
    if results:
        response = "Знайдено такі послуги:\n"
        for result in results:
            response += f"- {result['service_name']}: {result['price']} грн\n"
    else:
        response = "На жаль, не знайдено жодної послуги за вашим запитом. Спробуйте уточнити назву."

    bot.reply_to(message, response)
```

1.3 mysql_handler.py

```
import pymysql
import config
from datetime import datetime


# Connect to MySQL Database
def getConnection():
    connection = pymysql.connect(host=config.mysql_host,
                                 user=config.mysql_user,
                                 password=config.mysql_pw,
                                 db=config.mysql_db,
                                 charset='utf8mb4',
                                 cursorclass=pymysql.cursors.DictCursor,
                                 autocommit=True)
    return connection
```

```
def createTables():
    connection = getConnection()
    with connection.cursor() as cursor:
        tablename = "users"
        try:
            cursor.execute(
                "  CREATE TABLE `" + tablename + "` ( `userid` int(11) DEFAULT NULL,  `open_ticket` int(4) DEFAULT 0,  `banned` int(4) DEFAULT 0,  \
                    `open_ticket_spam`  int(4)  DEFAULT  1,  `open_ticket_link` varchar(50) DEFAULT NULL,  `open_ticket_time` datetime NOT NULL DEFAULT '1000-01-01 00:00:00')")
            return createTables
        except Exception as e:
            print(e)


def spam(user_id):
    connection = getConnection()
    with connection.cursor() as cursor:
        sql = "SELECT banned, open_ticket, open_ticket_spam FROM users WHERE userid = %s"
        cursor.execute(sql, user_id)
        data = cursor.fetchone()
        ticket_spam = data['open_ticket_spam']

        sql = "UPDATE users SET open_ticket_spam = %s WHERE userid = %s"
        spam = ticket_spam + 1
        cursor.execute(sql, (spam, user_id))
        return spam
```

```
    def user_tables(user_id):
        connection = getConnection()
        with connection.cursor() as cursor:
            sql = "SELECT open_ticket, banned, open_ticket_time, open_ticket_spam,
open_ticket_link FROM users WHERE userid = %s"
            cursor.execute(sql, user_id)
            data = cursor.fetchone()
            return data
    def getOpenTickets():
        connection = getConnection()
        with connection.cursor() as cursor:
            tmp = []
            cursor.execute("SELECT userid FROM users WHERE open_ticket = 1")
            for i in cursor.fetchall():
                tmp.append(i['userid'])
            return tmp
    def getBanned():
        connection = getConnection()
        with connection.cursor() as cursor:
            tmp = []
            cursor.execute("SELECT userid FROM users WHERE banned = 1")
            for i in cursor.fetchall():
                tmp.append(i['userid'])
            return tmp



    def start_bot(user_id):
```

```
        connection = getConnection()
        with connection.cursor() as cursor:
            sql = "SELECT EXISTS(SELECT userid FROM users WHERE userid =
%s)"
            cursor.execute(sql, user_id)
            result = cursor.fetchone()
            # If the User never started the bot before, add the Telegram ID to the
database
            if not list(result.values())[0]:
                sql = "INSERT INTO users(userid) VALUES (%s)"
                cursor.execute(sql, user_id)


    def open_ticket(user_id):
        connection = getConnection()
        with connection.cursor() as cursor:
            sql = "UPDATE users SET open_ticket = 1, open_ticket_time = %s
WHERE userid = %s"
            now = datetime.now()
            cursor.execute(sql, (now, user_id))
            open_tickets.append(user_id)
            return open_ticket


    def post_open_ticket(link, msg_id):
        connection = getConnection()
        with connection.cursor() as cursor:
            sql = "UPDATE users SET open_ticket_link = %s WHERE userid = %s"
            cursor.execute(sql, (link, msg_id))
```

```
            return post_open_ticket


    def reset_open_ticket(user_id):
        connection = getConnection()
        with connection.cursor() as cursor:
            sql = "UPDATE users SET open_ticket = 0,  open_ticket_spam = 1
WHERE userid = %s"
            cursor.execute(sql, user_id)
            open_tickets.pop(open_tickets.index(user_id))
            return reset_open_ticket


    def ban_user(user_id):
        connection = getConnection()
        with connection.cursor() as cursor:
            sql = "UPDATE users SET banned = 1 WHERE userid = %s"
            cursor.execute(sql, user_id)
            banned.append(user_id)
            return ban_user


    def unban_user(user_id):
        connection = getConnection()
        with connection.cursor() as cursor:
            sql = "UPDATE users SET banned = 0 WHERE userid = %s"
            cursor.execute(sql, user_id)
            banned.pop(banned.index(user_id))
            return unban_user
```

```
createTables = createTables()
open_tickets = getOpenTickets()
banned = getBanned()
```