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

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

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня магістра

Здобувача вищої освіти Караськіна Іллі Євгеновича
(ПІБ)
академічної групи 123м-23-1
(шифр)
спеціальності 123 Комп'ютерна інженерія
(офіційна назва)
за освітньо-професійною програмою «Комп'ютерна інженерія»
(офіційна назва)

на тему «Обґрунтування структури комп'ютерної системи КП «Дніпропетровська обласна клінічна офтальмологічна лікарня» з комплексом онлайн-консультацій пацієнтів»
(назва за наказом ректора)

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	проф. Цвіркун Л.І.			
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 Караськіна І.Є. **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 КП «Дніпропетровська обласна клінічна офтальмологічна лікарня» з комплексом онлайн-консультацій пацієнтів»,
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 практик, інших науково-технічних джерел сформулювати наукове завдання, конкретизувати предмет та мету досліджень	11.10.2024
Теоретичний	Обґрунтувати теоретичну базу для впровадження та роботи комплексу онлайн-консультацій.	25.10.2024
Синтез системи	Розробка комп'ютерної системи із комплексом онлайн-консультацій КП «ДОКОЛ»	15.11.2024
Розроблення програмного забезпечення	Розробка програмного забезпечення чат-боту та бази даних для функціонування комплексу-онлайн консультацій на базі КП «ДОКОЛ»	29.11.2024
Експериментальний розділ	Проведення і обробка результатів експериментів щодо навантаження на систему від роботи комплексу	06.12.2024

Завдання видано _____ проф. Цвіркун Л.І.
(підпис керівника) (прізвище, ініціали)

Дата видачі 06 вересня 2024 р.

Дата подання до екзаменаційної комісії 10.12.2024 р.

Прийнято до виконання _____ Караськін І.Є.
(підпис здобувача вищої освіти) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 116 с., 57 рисунок, 2 табл., 1 дод., 17 джерел.

СИСТЕМА, КОМПЛЕКС, ТЕНДЕНЦІЯ, МЕРЕЖА, ЦИФРОВІЗАЦІЯ, ПІДПРИЄМСТВО, СЕРВЕР, БАЗА ДАНИХ, ЧАТ-БОТ, MONGODB.

Об’єкт розробки: комп'ютерна система КП «ДОКОЛ» з комплексом онлайн-консультацій.

Мета роботи: розробка комплексу онлайн-консультацій пацієнтів на базі комп’ютерної системи КП «ДОКОЛ» та обґрунтування програмних та технічних рішень для забезпечення її безперебійного функціонування.

Методи дослідження: нами застосовувались методи збору та аналізу даних про навантаження на сервер та БД при роботі комплексу онлайн-консультацій, що включає базу даних MongoDB та чат-бота Telegram. Для вимірювання навантаження використовувались спеціалізовані програмні засоби (MongoDB Compass, Mongostat, Mongo Exporter, Prometheus, Grafana), а також деякі методи математичної статистики для аналізу отриманих даних.

В пояснювальній записці наведено аналіз розвитку цифровізації, зокрема онлайн-консультацій, та сформульовано завдання для дослідження.

В теоретичному розділі проаналізовано існуючі рішення для налагодження онлайн-комунікації, її організації, зберігання великих обсягів даних, зняття та виведення метрик навантаження на БД та сервер.

У розділі синтезу системи сформульовано технічні вимоги до створюваного комплексу, проаналізовано апаратне забезпечення системи та топологію мережі підприємства, створено схему функціональної структури.

У розділі розроблення програмного забезпечення створено ПЗ комплексу онлайн-консультацій та надано його детальний опис.

В експериментальну розділі – поставлено задачу експерименту та проведено дослідження створеного комплексу онлайн-консультацій. Практична значимість отриманих результатів полягає в можливості забезпечення підприємства комплексом проведення онлайн-консультацій.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД – база даних;

ДБЖ – джерело безперебійного живлення;

КП – комунальне підприємство;

ОП – оперативна пам’ять;

ПЗ – програмне забезпечення;

API – Application Programming Interface;

DHCP – Dynamic Host Configuration Protocol;

IoT – Internet of Things;

IP – Internet Protocol;

NAT – Network Address Translation.

ВСТУП

На даний час виникла потреба у впроваджені системи телемедицини в рамках комп’ютерної системи підприємства галузі охорони здоров’я за для покращення комунікації між пацієнтами та фахівцями. Це надасть можливість розширити доступ пацієнтів до медичних послуг, полегшити отримання консультацій з лікарями за допомогою онлайн спілкування, а також можливість співпраці та взаємного консультування між лікарями області.

Впровадження комплексу онлайн-консультацій призведе до збільшення навантаження на комп’ютерну систему організації, що вимагає від нас провести дослідження оптимальних програмних рішень для функціонування цього комплексу та здійснити розрахунок сумарного обсягу навантаження на апаратну частину системи за для аналізу спроможності комп’ютерної системи забезпечити нові потреби підприємства.

Об’єкт дослідження: комп’ютерна система КП «Дніпропетровська обласна клінічна офтальмологічна лікарня», що забезпечує роботу підприємства.

Предмет дослідження: комплекс онлайн-консультацій.

Мета і завдання дослідження: обґрунтування параметрів комп’ютерної системи та її спроможності забезпечити можливість впровадження комплексу онлайн-консультацій, що передбачає проведення аналізу та дослідження можливостей програмних та технічних рішень у цьому напрямку.

Методи дослідження: теорія збору та аналізу даних, теорія математичної статистики.

Ідея роботи: створення комплексу для забезпечення інтерактивного онлайн спілкування пацієнтів та фахівців з можливістю передачі файлів, їх зберігання та автоматичної розсилки.

Наукові положення:

Встановлено, що для ефективної роботи комплексу онлайн-консультацій, який включає чат-бота та базу даних, необхідно оптимізувати серверні ресурси, підготувати відповідне ПЗ та налаштувати збір і моніторинг метрик навантаження на систему для забезпечення стабільності роботи комплексу та дослідження спроможності системи підтримувати додаткові навантаження на сервер. Встановлено, що зростання використання віртуальної та резидентної пам’яті під час активної роботи комплексу складає не більше 0.05 Gb та 0.1 Gb відповідно. Також встановлено навантаження на мережу, що у піковх значеннях складає 345 kb/s та 2.4 mb/s для вхідного та вихідного трафіків відповідно. Згідно з цим спрогнозовано пікове навантаження на мережу підприємства від роботи комплексу онлайн-консультацій для вихідного трафіку, що складатиме ~7.2mb/s.

Наукові результати:

Обґрунтовано застосування MongoDB як оптимальної бази даних для роботи чат-бота, що дозволяє забезпечити більш ефективне зберігання та швидкий доступ до даних та файлів пацієнтів під час онлайн-консультацій. Підтверджено, що використання спеціалізованого програмного забезпечення (Prometheus, Grafana) для моніторингу системи сприяє ефективному та зручному аналізу і швидкому реагуванню на пікові навантаження.

Достовірність та обґрунтованість наукових висновків і рекомендацій ґрунтується на результатах проведених аналітичних та експериментальних досліджень із використанням засобів моніторингу та аналізу навантаження на базу даних та серверну інфраструктуру.

Практичне втілення досягнутих результатів полягає в розробці програмного забезпечення, необхідного для функціонування комплексу онлайн-консультацій пацієнтів та фахівців, та доведенню можливості впровадження такого комплексу у комп’ютерній системі КП «ДОКОЛ».

1 СТАН ПИТАННЯ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Загальні тенденції у розвитку та цифровізації галузі охорони здоров’я

В сучасному світі має місце бурхливий розвиток цифрових та телекомунікаційних технологій. Це призвело до великих змін та новацій у галузі надання медичних послуг. З’явилося новітнє медичне обладнання, а також галузеве програмне забезпечення, що автоматизує всі робочі процеси. Найновіші технології дозволяють проводити найскладніші операції, обстеження, прискорювати обробку лабораторних аналізів, консультувати та оглядати пацієнтів на відстані та багато іншого. За допомогою спеціальних програм для медичних центрів вибудовується робота з клієнтами, ведеться облік стану їхнього здоров'я, забезпечується взаємодія структурних підрозділів, контролюється склад препаратів, проводяться розрахунки з пацієнтами та персоналом тощо.

Іншими словами, мова йде про впровадження цифрової революції в сектор охорони здоров’я за допомогою революційних технологій і культурних змін.

Згідно з дослідження, яке опублікувала в квітні 2024 року канадсько-індійська компанія Precedence Research, яка є одним із провідних постачальників стратегічних аналізів ринку, обсяг глобального цифрового ринку охорони здоров’я у 2023 році оцінювався в 309,93 мільярда доларів США, і очікується, що він перевищить приблизно 1019,89 мільярда доларів США до 2033 року та зросте на 12,19% за прогнозований період з 2024 по 2033 рік (рисунок 1.1) [12]. Зростання ринку цифрового здоров’я обумовлене постійним технологічним прогресом – використанням штучного інтелекту, появою Інтернету речей, робототехніці в секторі охорони здоров’я разом із пристроями віддаленого моніторингу та керування, розвитком, поширенням та вдосконаленням телекомунікацій.

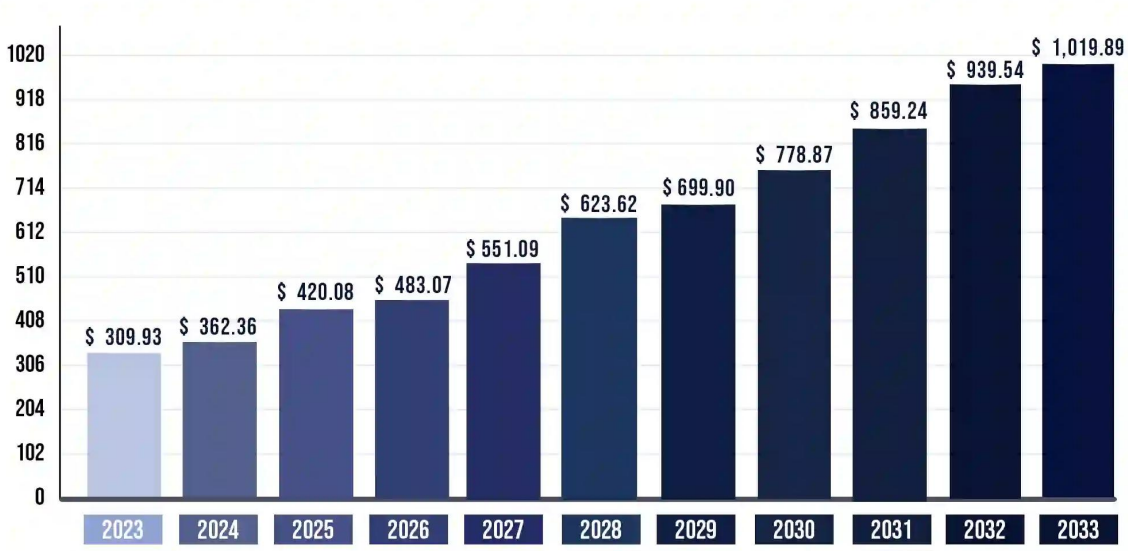

Рисунок 1.1 – Прогноз об'єму глобального ринку цифрової медицини з 2023 до 2033 рр., в млрд. доларів США

Аналогічно оптимістичні прогнози компанія робить та й щодо загально європейських тенденцій у даній сфері (рисунок 1.2). Згідно з ними в 2023 році розмір європейського цифрового ринку охорони здоров'я оцінювався в 108,74 мільярда доларів США, а до 2033 року прогнозується, що він становитиме близько 316,22 мільярда доларів США при середньорічному темпі зростання 10,8% з 2024 по 2033 рік [12].

Рисунок 1.2 – Прогноз об'єму загальноєвропейського ринку цифрової медицини з 2023 до 2033 рр., в млрд. доларів США

На європейському цифровому ринку охорони здоров’я спостерігається помітне зростання. Очікується, що ринок продовжуватиме стабільно розвиватися в найближчі роки, а його розмір оцінюється в мільярди євро.

Цифрова охорона здоров’я включає цифрову трансформацію в галузі охорони здоров’я, до якої активно залучені програмне забезпечення, обладнання та послуги. Згідно з цією концепцією цифрового здоров’я використовуються різні технології, такі як програми для мобільного здоров’я (mHealth), переносні пристрої, електронні медичні записи (EHR та EMR), телемедицина разом із персоналізованою медициною тощо. Потреба в медичній трансформації була особливо пов’язана зі зростанням старіння населення, дитячими хворобами, смертністю, проблемами, пов’язаними з високою вартістю та бідністю, а також расовою дискримінацією, пов’язаною з доступом до медичних послуг, що викликало технологічний прогрес. Крім того, епідемії та пандемії створили високу важливість для цифрового здоров’я, яке все ще продовжує розвиватися, тим самим сприяючи зростанню ринку в прогнозований період.

Факторами, які рухають глобальний цифровий ринок охорони здоров’я, є зростаючий попит на мобільні додатки для здоров’я, зростання попиту на послуги віддаленого моніторингу пацієнтів, зростання використання смартфонів і планшетів ще більше прискорили зростання ринку.

Очікується, що зростаюче поширення смартфонів разом із численними додатками для смартфонів, у всьому світі стане ключовим фактором зростання ринку цифрового здоров’я.

Окрім цього, спалах COVID-19 запровадив суворі норми соціального дистанціювання та карантини з боку урядів у всьому світі. У цій ситуації цифрові технології охорони здоров’я, наприклад телемедицина, надали постачальникам послуг величезні можливості для лікування пацієнтів, дотримуючись заходів карантину. Очікується, що це, у свою чергу,

сприятиме зростанню користуванням можливостями, що надають онлайн консультації з лікарями [4].

Тенденції стрімкого розвитку цифрової медицини також спостерігаються і в Україні. В медичній галузі нашої держави впроваджено велика кількість реформ та удосконалень, модернізацій систем з метою покращення надання медичних послуг населенню та наближення їх якості до європейського рівня. Наприклад успішно працює система електронної охорони здоров’я (eHealth), яка стосується медичної інформатики, що використовує інтернет та пов'язані технології для організації та розповсюдження медичних послуг та інформації.

Так, велике значення в галузі нової цифрової охорони здоров’я набула телемедицина. За допомогою комп'ютерних технологій стало можливим надання допомоги хворим на відстані, що робить медичні послуги більш доступними. Такі онлайн-консультації необхідні мешканцям віддалених районів, в екстрених ситуаціях, для пацієнтів з обмеженими можливостями або в замкнутому просторі. Лікар може провести віртуальний огляд, ознайомитися з результатами обстежень та аналізів, призначити лікування та вести регулярний контроль за станом здоров'я.

Крім того, телемедицина включає проведення онлайн-конференцій, зборів, навчання, швидкий обмін науковими відкриттями, проведення екстрених комісій щодо пацієнтів, тощо.

Використання телемедицини в Україні набуло більшого поширення в комерційних медичних закладах, але не менш важливо впровадження та розвиток цього напрямку на комунальних підприємствах охорони здоров’я. Особливо це має значення для комунальних підприємств обласного рівню, які обслуговують пацієнтів зі всієї області та є місцем роботи найкваліфікованіших фахівців області, а іноді і країни. Саме таким підприємством є об’єкт нашого дослідження – комунальне підприємство

«Дніпропетровська обласна клінічна офтальмологічна лікарня» (ск. КП «ДОКОЛ»).

1.2 Характеристика об’єкту дослідження

КП «Дніпропетровська обласна клінічна офтальмологічна лікарня» є місцем роботи досвідчених фахівців з високою кваліфікацією, що володіють найсучаснішими методиками лікування органів зору. Деякі з них є науковими співробітниками кафедри неврології та офтальмології ДДМА, що мають свої дослідження та розробки в галузі медицини, та відомі не тільки в нашій області, а й далеко за її межами.

Лікарня має найсучасніше медичне обладнання, яке надає можливість проводити діагностику на високому рівні, здійснювати лікувальні та профілактичні заходи при різноманітних захворюваннях органів зору та операційні втручання на найвищому технологічному та професійному рівні.

Лікарня має на своєму рахунку величезний досвід операцій на очах, як для дорослих, так и для дітей з усіх кінців України.

Завдяки унікальним методикам фахівцям клініки вдається боротися з різноманітними захворюваннями, такими як міопія, катаракта, глаукома та ін.

Тому лікарня є провідним медичним закладом в своїй галузі.

1.3 Аналітичний огляд існуючих проблем

Саме технічна та фахова унікальність лікарні, та те, що вона – обласний центр в галузі офтальмології, є причиною виникнення необхідності активного використання телекомунікації.

Існує проблема того, що пацієнти, яким на місцевому рівні надано рекомендацію звернутися з приводу стаціонарного лікування або хірургічного втручання, потерпають великих труднощів, як матеріальних, так і фізичних, щоб потрапити до обласного центру з віддалених районів нашої області або навіть з інших областей нашої країни.

Одночасно з цим існує можливість того, що фахівець клініки після ознайомлення з документами, що надав такий пацієнт, може не погодитись з необхідністю стаціонарного лікування або хірургічного втручання та запропонувати медикаментозне лікування проблеми. Крім того існує велика ймовірність того, що можливість поступити в стаціонар та потрапити на операцію з'явиться лише через деякий час у зв'язку з великою чергою серед пацієнтів. Тобто неодноразово виникають ситуації, при яких людина з вадами здоров'я та можливими матеріальними труднощами буде вимушена повернутись до свого місця проживання щоб або лікуватися вдома, або подолати цю дорогу ще раз для повторного візиту з метою госпіталізації.

Саме цю проблему може вирішити телемедицина, завдяки якої пацієнт може направити до лікаря результати всіх необхідних йому для постановки діагнозу досліджень, та отримати відповідь фахівця дистанційно.

Також є проблема або недостатності фахівців у віддалених районах області, або недостатньої кваліфікації та досвіду у наявних регіональних фахівців.

На базі КП «ДОКОЛ» створена та функціонує спілка офтальмологів Дніпропетровщини. Фахівці КП «ДОКОЛ» мають контакти з багатьма фахівцями з різних міст та населених пунктів області, з якими проводять консультації та обмін досвідом. Тому на даний час стоїть потреба в організації для даної спільноти інформаційного простору з можливістю спілкування з приводу складних випадків у професійній діяльності та спільним доступом до файлів з дослідженнями з метою колегіального розглядання проблеми та консультування з її вирішення.

1.4 Постановка завдання дослідження

Загальні тенденції розвитку телекомунікацій надають можливість задовольнити потреби об'єкту нашого дослідження до віддаленого спілкування з пацієнтами та фахівцями в Дніпропетровській області.

Отже метою нашої роботи є обґрунтування комп’ютерної системи КП «Дніпропетровська обласна клінічна офтальмологічна лікарня» із дослідженням можливостей по впровадженню комплексу онлайн-консультацій.

Впровадження комплексу потребує налагодження телекомунікаційних систем на базі об’єкту впровадження, для чого необхідно:

1. Проаналізувати та сформулювати вимоги до системи та комплексу;
2. Дослідити та обґрунтувати можливості комп’ютерної системи об’єкту впровадження;
3. Розрахувати технічні характеристики обладнання, необхідного для забезпечення роботи комплексу;
4. Проаналізувати спроможність комп’ютерної системи забезпечити безперебійну роботу комплексу;
5. Проаналізувати існуючи системи керування базами даних та обрати найбільш оптимальну, що задовольнятиме потребам організації;
6. Обрати оптимальні засоби для забезпечення комунікації та розрахувати оптимальний режим їх роботи;
7. Дослідити навантаження на комп’ютерну систему від обміну даними, у тому числі медіа даними;
8. За результатом проведених досліджень обґрунтувати можливості по впровадженню запропонованого комплексу у існуючи комп’ютерні системи закладів охорони здоров’я.

2 ТЕОРЕТИЧНИЙ РОЗДІЛ

2.1 Відомості про топологічну структуру об’єкту

КП «ДОКОЛ», який є об’єктом впровадження системи телемедицини, розташований за адресою площа Соборна, 14, м. Дніпро, Дніпропетровська область (рисунок 2.1).

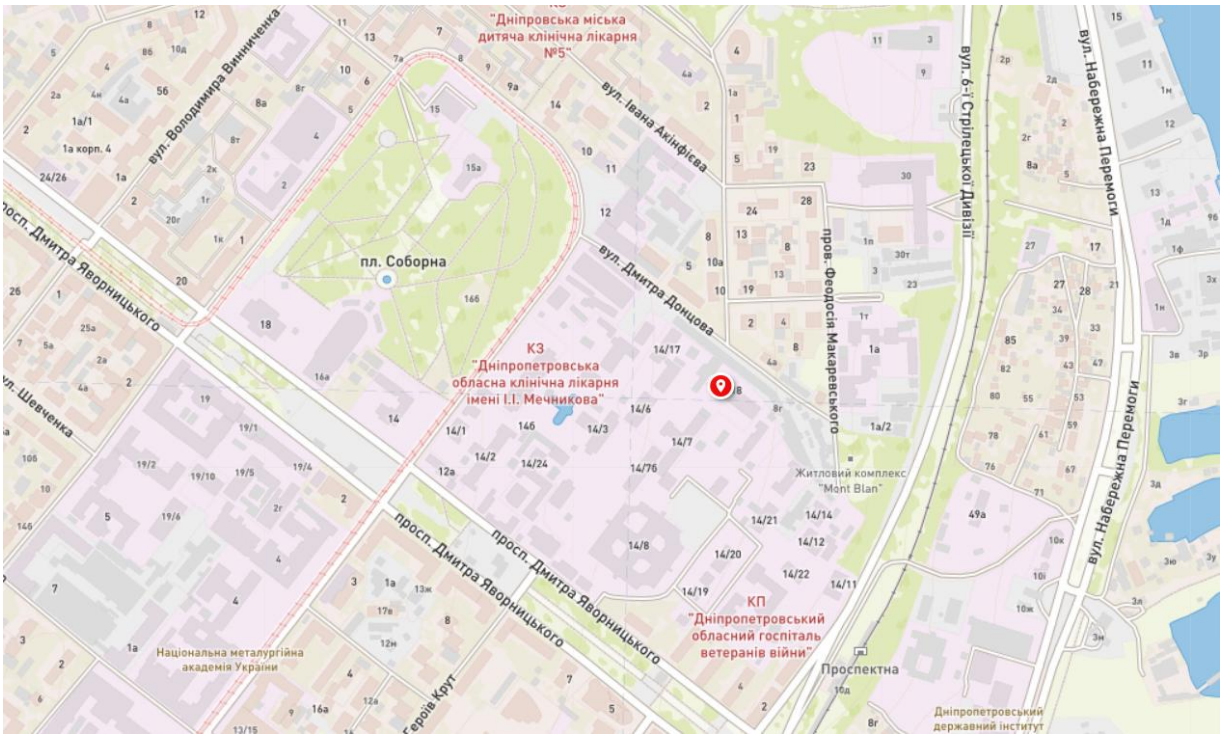

Рисунок 2.1 – Географічне розташування КП «ДОКОЛ»

2.2 Значущість технологій API та сфери їх застосування

Актуальність технологій API стрімко зростає в сучасному світі програмного забезпечення через їхню здатність забезпечувати зручну інтеграцію між різними системами та сервісами. Ця технологія є невід'ємною частиною розвитку цифрової економіки, оскільки вона дозволяє різним додаткам та платформам обмінюватися даними, функціональністю та сервісами незалежно від їхньої архітектури або мови програмування.

Сьогодні API широко використовуються в різних сферах, таких як мобільні додатки, веб-сервіси, інтернет речей, хмарні обчислення та ін.. Ця

технологія є основою багатьох популярних сервісів, зокрема соціальних мереж, онлайн-банкінгу, електронної комерції та стрімінгових платформ. Через це API значно спрощують масштабування систем та прискорюють інновації в різних галузях.

Однією з причин широкого впровадження API є їхня гнучкість і ефективність. Вони дозволяють компаніям швидко адаптуватися до змін, інтегрувати нові функції та об'єднуватися з зовнішніми сервісами, що робить різноманітні процеси більш динамічними та ефективними.

API можуть бути побудовані на основі трьох основних архітектур: RPC, REST і SOAP. Кожен із цих підходів має свої особливості та використовується для вирішення різних завдань.

RPC (віддалений виклик процедур) є одним з найбільш базових способів обміну інформацією між системами, при якому передаються параметри, і отримується результат. У порівнянні з іншими типами API, RPC орієнтований переважно на виконання дій або запуск процесів, тоді як інші архітектури спрямовані на обмін даними та ресурсами. RPC може працювати з двома форматами даних – JSON та XML. SOAP, з іншого боку, використовує тільки XML для передачі даних, і через його складну та суворо регламентовану структуру, цей метод є найбільш вимогливим у реалізації. REST – це більш спрощений підхід у порівнянні з SOAP. Для кожного ресурсу в REST створюється окрема URL-адреса, за якою можна отримати або відправити інформацію.

2.3 Вибір оптимального середовища для організації комунікації

При виборі можливих середовищ комунікації були проаналізовані найпоширеніші існуючі засоби та рішення, такі як email-сервіси, програми для відеоконференцій та месенджери.

Ці засоби повинні задовольняти комплексу вимог:

1. Бути доступними та розповсюдженими серед населення;

2. Мати можливість забезпечити безоплатну або доволі дешеву можливість аудіо та відео зв’язку;

3. Забезпечувати можливість пересилання файлів різних форматів та достатньо великого обсягу, в тому числі медіаданих.

2.3.1 Email-сервіси

На даний час Email-сервіси є одним з найпоширеніших засобів з забезпечення можливості для спілкування користувачів, як у діловій, так і в особистій сфері, оскільки виконують асинхронний обмін повідомленнями. Функції Email-сервісів дозволяють здійснювати пересилку як текстових повідомлень, так і вкладених файлів, листуватися та архівувати переписку, виконувати сортування та пошук листів, тощо. Задля підвищення комфорту користувачів та збільшення продуктивності праці, дані сервіси інтегруються з іншими інструментами, наприклад, з системами керування проектами, календарями, корпоративними месенджерами, тощо. Популярними прикладами таких сервісів є Gmail, Microsoft Outlook та ProtonMail.

Gmail – мабуть найпопулярніший Email-сервіс у світі, створений компанією Google, який інтегрується з іншими сервісами компанії, такими як Google Workspace, такими як Google Drive, Calendar, Google Meet. Gmail має зручний та зрозумілий інтерфейс, надає можливість пошуку, сортування, зберігання та архівування повідомлень, запобігання спаму, тощо.

Однією із складових сервісу Microsoft 365 є Microsoft Outlook, який забезпечує повну інтеграцію з іншими продуктами Microsoft, такими як календарі, контакти та завдання. Outlook дозволяє зручно організовувати електронну пошту, створювати правила для автоматизації робочих процесів та налаштовувати зустрічі через інший сервіс Microsoft 365 – Microsoft Teams.

ProtonMail – це ще один Email-сервіс, який відрізняється від інших своєю направленістю на забезпечення безпеки та конфіденційності. Він виконує захист електронної пошти шляхом наскрізного шифрування. Це є

причиною того, що ProtonMail завоював популярність серед користувачів, для яких велике значення має надійність захисту конфіденційності даних.

2.3.2 Програми для відеоконференцій

Враховуючи зростання поширеності у світі таких явищ як віддалена робота та дистанційне навчання, велике значення набули відеоконференції.

Вони створюють можливість спільнотам користувачів в режимі реального часу здійснювати спілкування за допомогою відео та аудіо, обмінюватись екраном та спільно працювати над документами. Основними критеріями вибору з великої кількості інструментів для проведення відеоконференцій є стабільності зв'язку, розмір аудиторії, а також наявність та кількість додаткових функцій. Основними та найбільш відомими серед таких інструментів є Zoom, Google Meet, Microsoft Teams.

Zoom забезпечує наявність стабільного сервісу для відеозустрічей, які передбачають велику кількість учасників, а також підтримку відео- і аудіодзвінків, можливість записати зустріч, можливість спільного використання екрана. Також Zoom дозволяє створення віртуальних переговорних кімнат.

Google Meet є інструментом для відеоконференцій, створеним Google та інтегрованим у Google Workspace. Він підходить для невеликих нарад і класів, оскільки не вимагає встановлення стороннього програмного забезпечення та простий у використанні.

Вищезгаданий Microsoft Teams є платформою для роботи у команді, оскільки у ньому об'єднуються обмін файлами, чат та відеоконференції. Відеозустрічі інтегровані з документами Microsoft 365 та календарем. Все вищевказане робить Teams ефективним інструментом для бізнес-спілкування.

2.3.3 Месенджери

В наш час Месенджери набули великої популярності, які сприяють як особистому спілкуванню, так і співпраці цілих спільнот. Вони надають можливість для миттєвого обміну повідомленнями та мультимедійними файлами, здійснювання аудіо- та відеодзвінків. Серед їх функціоналу однією з найважливіших функцій вважається спроможність створювати групові чати. Це сприяє організації командної співпраці та налагодженню ефективного спілкування працівників. Найпоширенішими та найвідомішими меседжерами у нашому інформаційному просторі вважаються Telegram, WhatsApp, Viber.

Telegram – це месенджер, ставший особливо популярним та розповсюджений в останні роки. Він має високий рівень безпеки та широкі можливості для налагодження каналів та чатів. Крім того, він також забезпечує можливість обміну файлами великого розміру, здійснення відеодзвінків та створення ботів для автоматизації різноманітних завдань.

WhatsApp є одним з месенджерів, що вважаються найпопулярнішими у світі. Він забезпечує як особистий зв’язок, так і командне спілкування для, у тому числі і для бізнесу. Також WhatsApp підтримує розсилку текстових повідомлень, аудіо та відеодзвінки, виконує обмін документами та мультимедійними файлами.

Ще одним популярним месенджером є Viber, який створює комфортну та зручну можливість текстового та аудіовізуального зв'язку. Він має простий інтерфейс, забезпечує підтримку створення групових чатів, обміну файлами, аудіо та відеодзвінками. Такі функції сприяють його частому використанню для організації спілкування як в особистій, так і в діловій сфері в невеликих компаніях. І хоча в останні роки він поступився своїми позиціями іншим месенджерам, завдяки всьому вищесказаному Viber як і раніше має дуже широке поширення серед користувачів з багатьох країн світу.

2.3.4 Результати аналізу

При проведенні комплексного аналізу по всім напрямкам, було зроблено вибір на користь додатку Telegram, що задовільняє всім цим вимогам. Він є одним з розповсюджених та легкодоступних месенджерів, в якому вже зареєстрований великий відсоток населення, та, якщо людина ще не користується даним месенджером, – процедура підключення до нього доволі проста, швидка та безпроблемна. Також цей месенджер забезпечує безкоштовний аудіо- та відеозв'язок, та має можливість передачі даних необхідного нам обсягу, в тому числі медіа файлів.

2.4 Інтеграція телеграм-ботів із технологіями API для розширення функціоналу

В процесі впровадження комплексу онлайн-консультацій в КП «ДОКОЛ», функцію передачі даних та організації інтерактивного спілкування, аудіо та відео зв'язку між користувачами буде покладено на бот – тобто на автоматизований програмний додаток, який буде здійснювати повторювані операції в мережі згідно із спеціальними алгоритмами, незалежно від втручання людини.

Такі боти є можливість створити у різних соціальних мережах та месенджерах, наприклад, у Telegram, Discord, Viber ті ін.

Цілям та вимогам по забезпеченню впровадження комплексу онлайн-консультацій в КП «ДОКОЛ» цілком відповідає Телеграм-бот, оскільки він є інтелектуальним інструментом, що здатний налагодити спілкування в широкодоступному месенджері Telegram. Такі боти мають велику кількість можливостей: автоматизують взаємодію між респондентами, надають інформацію згідно з запитами, та забезпечують можливість обміну файлами різних форматів та достатнього обсягу.

Телеграм бот функціонує завдяки інтеграції з Telegram API, що робить доступним широкий спектр можливостей месенджера Telegram. Серед них:

- можливість інтерактивної взаємодії з користувачами за допомогою текстових звернень та запитів з однієї сторони та відповідей з іншої;
- функція автоматизованого розсилання повідомлень та файлів окремим особам, або великим групам осіб;
- можливість отримання та аналізу даних;

Завдяки використанню ботів сумісно з Telegram API, ми можемо значно розширити функціонал, який пропонує месенджер для користувачів. Для роботи по створенню та програмуванню Telegram ботів існують різні вільно поширювані фреймворки та бібліотеки, такі як python-telegram-bot та aiogram. Вони роблять простішим процес програмування боту телеграм мовою Python.

2.5 Вибір мови програмування

Python – це універсальна, високорівнева мова програмування, що характеризується простим синтаксисом та широкими можливостями. Вона була створена Гвідо ван Россумом у кінці 1980-х років і отримала популярність завдяки своїй читабельності, зручності у використанні та величезній спільноті розробників, яка підтримує її розвиток. Python активно застосовується у різних сферах програмування – від веб-розробки та автоматизації процесів до машинного навчання, штучного інтелекту та аналізу даних [7].

Одна з головних особливостей Python – це його інтерпретована природа, що дозволяє розробникам виконувати код без необхідності компіляції, що робить процес розробки більш швидким та інтерактивним. Python підтримує кілька програмних парадигм, зокрема об’єктно-орієнтоване, процедурне та функціональне програмування. Це дозволяє адаптувати мову до різних задач і робить її надзвичайно гнучкою.

Відмінними рисами Python є його лаконічний і зрозумілий синтаксис, який робить мову легкою для вивчення як початківцями, так і досвідченими

програмістами. Наприклад, завдяки використанню відступів замість фігурних дужок для позначення блоків коду, Python дозволяє підтримувати чистоту та структурованість програм. Крім того, мова має величезну стандартну бібліотеку, яка містить готові рішення для багатьох типових завдань: робота з файлами, регулярними виразами, мережевими протоколами, базами даних, тестуванням та багато іншого.

Python активно використовується для створення телеграм-ботів завдяки своїй простоті та наявності спеціальних бібліотек, таких як python-telegram-bot та aiogram, що значно полегшує інтеграцію з Telegram API. Це дає змогу швидко налаштовувати взаємодію з користувачами та розширювати функціональні можливості ботів. У рамках взаємодії з API телеграм-боти на Python можуть виконувати різноманітні завдання: від обробки повідомлень і команд користувачів до роботи з веб-сервісами, базами даних і зовнішніми API.

Python також має автоматичне управління пам'яттю, що звільняє розробників від необхідності ручного звільнення ресурсів, як це робиться у мовах, таких як C або C++. Механізм збору сміття автоматично видаляє невикористовуванні об'єкти, що робить мову безпечнішою та більш надійною для великих проектів.

Завдяки своїм можливостям та простоті, Python є ідеальним вибором для створення проектів, таких як телеграм-боти, оскільки забезпечує високу продуктивність та зручність у розробці.

В нашому випадку, для виконання дій з програмування чат-боту нами було використано оболонку Visual Studio Code, що має зручний інтерфейс та дозволяє працювати з багатьма мовами програмування, включаючи мову Python.

2.6 Обґрунтування вибору бази даних для системи

Для розробки системи, яка дозволяє пацієнтам та спеціалістам передавати результати обстежень, зокрема зображення, рентгенівські та КТ-знімки або тощо, через телеграм-бот, важливим є вибір бази даних, що забезпечить надійне зберігання великих файлів та зручний доступ до них для подальшого аналізу та консультацій. У рамках цієї роботи було розглянуто дві популярні бази даних – MongoDB та PostgreSQL – з метою визначення, яка з них найкраще підходить для вирішення завдань, поставлених перед системою.

2.6.1 MongoDB

MongoDB є документно-орієнтованою базою даних, яка зберігає дані у форматі JSON-подібних документів. Це забезпечує високу гнучкість при роботі з різними типами даних, що є вагомим аргументом для систем, де структура даних може змінюватися або коли необхідно зберігати полуструктуровану інформацію. У нашій системі, де пацієнти та спеціалісти будуть надсилати файли з результатами обстежень (зокрема зображення), особливо важливим є використання вбудованої функції MongoDB GridFS, яка дозволяє зберігати великі файли, розбиваючи їх на частини та зберігаючи у колекціях бази даних.

Крім того, однією з головних переваг MongoDB є підтримка горизонтального масштабування, що дозволяє додавати сервери для збільшення потужностей бази даних без зміни основної архітектури системи. Це робить MongoDB привабливим вибором у випадку збільшення обсягів даних або зростання кількості користувачів системи. Від версії 4.0 MongoDB також підтримує транзакції за принципом ACID, що забезпечує цілісність даних навіть у складних сценаріях роботи з файлами.

Проте у даної БД є і свої недоліки. Так, хоча MongoDB і добре працює з великими обсягами даних, вона може виявитися менш ефективною при

великій кількості операцій читання та запису документів. Крім того, вимоги до ресурсів можуть зрости, особливо при роботі з великими файлами, що потребує уважного налаштування сервера для забезпечення стабільної роботи системи.

2.6.2 PostgreSQL

PostgreSQL – це реляційна база даних, відома своєю надійністю, відповідністю стандартам SQL та широкими можливостями обробки структурованих даних. Важливим аспектом для нашої системи є підтримка PostgreSQL великих об'єктів (Large Objects), що дозволяє ефективно зберігати великі файли, такі як зображення та результати медичних досліджень. Це робить PostgreSQL також привабливим варіантом для зберігання медичних даних, які можуть бути об'ємними.

PostgreSQL відрізняється підтримкою складних SQL-запитів та можливістю аналізу даних, що є важливим для дослідження навантаження на сервер залежно від обсягу та кількості файлів, які надсилаються через систему. Крім того, PostgreSQL забезпечує високу гарантію цілісності даних завдяки підтримці транзакцій ACID, що є критично важливим при роботі з медичними даними, де навіть незначні збої можуть спричинити втрату цінної інформації.

Однак, робота з великими об'єктами у PostgreSQL вимагає більш складного налаштування та розробки. Використання великих об'єктів не настільки гнучке, як в MongoDB, і може потребувати додаткових зусиль при інтеграції з іншими компонентами системи.

2.6.3 Порівняльний аналіз та вибір

Для впроваджуваного в організації комплексу онлайн-консультацій MongoDB пропонує більш гнучке рішення для зберігання великих файлів, особливо з огляду на можливість використання GridFS для обробки

медичних зображень. Це робить MongoDB більш привабливою БД для випадків, коли дані мають варіативну структуру або коли потрібно забезпечити швидке масштабування системи.

З іншого боку, PostgreSQL надає кращі можливості для роботи зі структурованими даними та складними SQL-запитами, що може бути корисним при детальному аналізі даних та оцінці навантаження на систему. І хоча до недавнього часу однією з вагомих переваг PostgreSQL також була можливість забезпечувати високу надійність при роботі з критично важливими даними завдяки своїм транзакційним можливостям, MongoDB останніх версій може становити непогану конкуренцію на цьому полі.

Отже, після проведення аналізу всіх переваг та недоліків, а також потреб в рамках впроваджуваного комплексу онлайн-консультацій, для створення та розгортання бази даних на сервері підприємства нами було прийнято рішення про застосування MongoDB.

2.7 Вибір засобів та методів дослідження

Для проведення дослідження навантаження на сервер підприємства під час роботи комплексу онлайн-консультацій, що включатиме в себе базу даних та чат-бота, перед нами постала необхідність в обранні методів збору та аналізу метрик продуктивності системи. Основна мета полягала у відстеженні й аналізі ключових показників, які можуть впливати на стабільність роботи комплексу, ефективність роботи БД та обробки запитів.

Нами було сформульовано наступні критерії вибору для цих програмних методів:

– програмне забезпечення, що використовуватиметься для збору, виведенню та/або аналізу метрик, має бути сумісним із MongoDB, оскільки нами було вирішено використовувати Mongo в процесі впровадження комплексу (п. 2.6.3);

– цей інструмент повинен мати можливість виводити дані метрик в зручному для їх перегляду та структурованому вигляді, що дозволить проведення подальшого аналізу цих даних.

За цими критеріями, нами було проаналізовано та обрано наступний ряд інструментів та програмних засобів для збору та обробки метрик:

1. MongoDB Compass – графічний інтерфейс, який, в числі інших своїх функцій, дозволяє переглядати базову статистику роботи MongoDB у реальному часі та надає інформацію про стан колекцій і документів в базі.

2. Mongostat – консольний інструмент з пакету MongoDB Tools, який виводить в реальному часі показники роботи бази даних, такі як кількість запитів на запис і читання, використання пам'яті та навантаження на процесор. Mongostat надає оновлювану інформацію про продуктивність бази даних у вигляді консольних таблиць, що є корисним для аналізу пікових навантажень.

3. MongoDB Exporter – окремий спеціалізований інструмент для збору матричних даних про роботу MongoDB. Він форматує дані в стандартизованому вигляді для подальшої обробки іншими програмами. MongoDB Exporter надає розширений набір метрик в порівнянні з Compass та Mongostat, що дозволяє відстежувати специфічні показники, такі як тривалість операцій читання і запису, обсяг використаної пам'яті та інші характеристики продуктивності.

4. Prometheus – система моніторингу та збору метрик з підтримкою можливості налаштування запитів і сповіщень. Prometheus використовується для збору й збереження метрик за допомогою таких інструментів, як MongoDB Exporter, а також для їх обробки та відображення у вигляді графіків. Основна перевага Prometheus полягає в можливості автоматично опитувати джерела даних із певною частотою і зберігати історію вимірювань, що важливо для здійснення аналізу тенденцій у роботі системи.

5. Grafana – ще один інструмент, але призначений вже для візуалізації метрик. Він дозволяє створювати панелі (дашборди) для відображення показників навантаження, наприклад, у вигляді графіків. Використання Grafana дає змогу легко аналізувати поведінку системи в реальному часі та визначати аномалії в роботі серверного комплексу, побудованого на основі MongoDB і чат-бота Telegram.

2.8 Висновки до теоретичного розділу

В цьому розділі нами було досліджено та проведено аналіз існуючих програмних рішень щодо можливості організації онлайн-комунікації, збереження даних та моніторингу навантаження на обладнання.

В процесі цього аналізу, нами було обрано мессенджер Telegram, як середовище для комплексу онлайн-комунікацій. Для створення програми чат-боту Telegram, який виступатиме посередником у спілкуванні, було обрано мову програмування Python, як найбільш оптимальну для цих цілей. Для збереження інформаційних та файлових даних та використання їх в процесі роботи комплексу було обрано базу даних Mongo завдяки її зручному графічному інтерфейсу та можливості зберігати файли різних форматів в документах.

Як результат нами було обрано ряд програмних методів для проведення дослідження навантаження на сервер та БД підприємства від роботи комплексу онлайн-консультацій.

3 СИНТЕЗ СИСТЕМИ

3.1 Вибір і обґрунтування принципів розробки комплексу

При розробці системи нами було закладено можливість того, що може виникнути потреба у її коректуванні або додаванні до неї окремих модулів за необхідності. Наприклад, може виникнути ситуація, коли адміністрація КП «ДОКОЛ» прийме рішення про надання онлайн-консультацій не тільки безкоштовно за електронними направленнями, але й на платній основі, з наданням пацієнтом замість електронного направлення – квитанції про оплату.

Крім того слід опрацювати інтеграцію даних. Це необхідно для того, щоб документи, надані від пацієнтів на розгляд фахівцям, зберігалися необхідний час на сервері з організацією доступу до них фахівців. Окремо мають бути інтегровані дані, що стосуються взаємних консультацій лікарів зі спілки офтальмологів Дніпропетровщини.

Обов’язково потрібно при побудові функціональної схеми приділити увагу забезпеченню конфіденційності та захисту особистих медичних даних пацієнтів. В системі має бути забезпечено функцію аутентифікації для фахівців, що матимуть доступ до конфіденційної інформації про пацієнтів.

Система розробляється для застосування у сфері охорони здоров’я. Це є стимулом для забезпечення максимального комфорту та зручності для людей із вадами здоров’я. Тому вона повинна мати простий та зручний у використанні інтерфейс. Це дозволить оптимізувати роботу персоналу медичного закладу з одного боку, та доступність для всіх бажаючих скористуватися можливістю онлайн доступа до медичних послуг – з іншого.

3.2 Постановка технічних вимог до комплексу онлайн-консультацій та комп'ютерної системи в цілому

3.2.1 Вимоги до реалізації системи

Комп'ютерна система КП «ДОКОЛ», яка на даний час забезпечує внутрішні потреби користувачів щодо функціонування бухгалтерських та медичних програм, та Internet зв'язку із зовнішніми респондентами, повинна також задовольняти новим вимогам щодо реалізації проектів онлайн доступу пацієнтів до медичних послуг та співпраці фахівців спілки офтальмологів Дніпропетровщини, з можливістю передачі та зберігання даних.

Модернизована таким чином система повинна реалізовуватись на базі сучасних програмних та апаратних рішень, які мають забезпечити її високий рівень безпеки, надійності та ергономічності.

З метою забезпечення можливості швидкого реагування на нові вимоги користувачів системи, вона повинна мати модульну структуру, що надасть змогу вносити до неї зміни або доповнювати її новими компонентами.

Система повинна мати достатню потужність для забезпечення пересилки великої кількості та/або великого обсягу даних через мережу Internet, що забезпечуватиме доступність системи для користувача, та зберігання на сервері цих даних протягом запланованого проміжку часу.

3.2.2 Вимоги до функцій комплексу онлайн-консультацій, що впроваджується

Впроваджувана система буде працювати у двох напрямках – онлайн консультації пацієнтів з лікарями та онлайн консультацій між фахівцями зі спілки офтальмологів Дніпропетровщини. По кожному з напрямків вона має забезпечувати певні функції.

Стосовно напрямку онлайн консультацій пацієнтів з лікарями, система повинна забезпечувати наступні функції:

1. Можливість передачі пацієнтом файлів із медичними дослідженнями, з метою попереднього ознайомлення з ними лікаря.

2. Ідентифікацію наданих пацієнтом файлів за номером його електронного направлення.

3. Аудіо чи відео зв’язок з пацієнтом для проведення онлайн консультації.

4. Можливість отримання пацієнту фідбеку від лікаря.

Стосовно напрямку забезпечення спілкування та співпраці фахівців зі спілки офтальмологів Дніпропетровщини, система має забезпечувати такі функції:

1. Звернення одного з фахівців на отримання консультації стосовно складного випадку у лікарській діяльності.

2. Прийняття та зберігання файлів з медичними дослідженнями за для ознайомлення з ними іншими фахівцями.

3. Розсилку іншим членам спілки цих даних з запрошенням на визначений час до прийняття участі у відеоконференції щодо розгляду складного випадку у медичній практиці та консультування з його приводу.

3.2.3 Показники призначення

Для чіткої роботи комплексу онлайн-консультацій, що впроваджується за допомогою чат-боту Telegram, необхідно забезпечити умови його функціонування відповідно з вимогами до технологічного процесу. Насамперед це важливо для серверу, оскільки на ньому зберігаються облікові дані фахівців та медичні дані пацієнтів, та сама програмна структура чат-бота. Перед Системою постає задача забезпечення доступу до зазначених вище даних як у мережі підприємства, так і поза її межами, для уповноважених осіб. Також вони мають бути захищеними від пошкодження та втрати при нештатних ситуаціях, збоях та кібератаках.

Обов’язково має бути у наявності один системний адміністратор, з бакалаврським ступенем вищої освіти за спеціальністю «Комп’ютерна інженерія». До його обов’язків має входити забезпечення функціонування впровадженого комплексу онлайн-консультацій. Має бути створений чіткий графік, перевірок та підтримки системним адміністратором працездатності серверу та чат-боту, що входять до комплексу онлайн-консультацій. Крім цього необхідне проведення позапланових заходів з коригування та налагодження роботи комп’ютерної системи та комплексу у відповідь на заявки користувачів та повідомлення про їх некоректну роботу.

Обов’язкова наявність фахівця, до обов’язків якого буде входити обробка вхідної кореспонденції, яка надходить з чат-боту від його користувачів.

3.2.4 Вимоги до інформаційної та програмної сумісності

Розроблюване програмне забезпечення комплексу має бути виконано для роботи на платформі Telegram та бути сумісним з його Android та Desktop версіями. У разі необхідності внесення змін, також необхідна наявність програмної оболонки, що підтримує середу мови Python.

3.2.5 Вимоги до патентної чистоти

Обладнання і програмне забезпечення, для роботи у використовуваних режимах, обов’язково повинно бути сертифікованим (за потреби) та мати патентну чистоту.

3.2.6 Вимоги до надійності

ПЗ в цілому повинно забезпечувати надійне і безперебійне функціонування комплексу, програмний код якого знаходиться на сервері підприємства, протягом робочого дня.

Час пошуку несправності і відновлення після відмови повинен складати не більше 3 годин.

Обов’язкова організація безперебійного електропостачання для забезпечення неможливості виникнення ситуацій з аваріним припиненням роботи компонентів Системи, зокрема апаратної складової комплексу онлайн-консультацій. З цією метою мають бути використані ДБЖ, стабілізатори напруги та підключений до загальної електричної мережі резервний генератор електропостачання.

3.2.7 Додаткові вимоги до Системи, пов'язані з особливими умовами її експлуатації

На даний час, у зв’язку із воєнним станом та ризиком виникнення перебоїв в електропостачанні, підприємству необхідно бути оздобленим як власним електрогенератором, так і джерелами безперебійного живлення, стабілізаторами напруги та перетворювачами напруги. Крім того також обов’язкова наявність фахівця з обслуговування генератору та іншої електричної техніки, електричних мереж та іншого устаткування. Це необхідно для забезпечення безперебійної роботи в цілому, та в тому числі і для забезпечення безпечного функціонування серверу з чат-ботом та базами даних підприємства та доступності комплексу онлайн-консультацій через мережу Internet.

3.2.8 Вимоги до видів забезпечення комплексу онлайн-консультацій

3.2.8.1 Вимоги до інформаційного забезпечення

Система онлайн-консультацій повинна мати зручну для користувачів форму, що забезпечує комфортний доступ, легке та зрозуміле користування. Необхідно щоб данні були ідентифіковані, структуровані та збережені на сервері.

Ідентифікація даних в базі даних серверу має відбуватись за допомогою автоматично виданих кожному документу унікальних id самою БД, а також за допомогою id створених програмою чат-боту та номерами електронного направлення, що входять до переліку інформації, яку вносять пацієнти. Крім того мають бути передбачені записи в документах з обліковими даними фахівців, що відповідатимуть за розподіл рівнів доступу до різних функцій чат-боту.

Інформаційні дані в БД мають бути структуровані таким чином, щоб в БД були присутні окремі колекції для записів пацієнтів та фахівців, колекція з білим списком користувачів фахівців, колекції для збереження файлових даних, що поступатимуть разом із запитами від користувачів.

3.2.8.2 Вимоги до лінгвістичного забезпечення

Для системи має бути забезпечена можливість обробляти, структурувати та видавати інформацію на англійській та українській мовах.

3.2.8.3 Вимоги до програмного забезпечення

Система має бути забезпечена програмною оболонкою Visual Studio Code або іншою оболонкою, що дозволяє працювати із програмами, створеними мовою програмування Python.

В системі має бути встановлений додаток Telegram за для контролю над роботою чат-боту та його тестуванням за необхідністю.

Система повинна мати встановленим пакет програм MS Office або аналог, за для роботи з кореспонденцією, що надходить з чат-боту.

В системі має бути встановленим пакети програмного забезпечення для роботи з базами даних MongoDB, а також для роботи з медіафайлами.

3.2.8.4 Вимоги до технічного забезпечення

Система повинна мати інтерфейс, який буде пристосований як для використання миші та клавіатури, так і для сенсорного управління. Також важливо забезпечити такі характеристики інтерфейсу, при яких навантаження на зір буде мінімальним.

Пристрої користувачів комплексу мають пітримувати роботу додатку Telegram. Для мобільної версії додатку, на пристрої має бути встановлено програмну оболонку Android 6.0 або вище. Для встановлення desktop версії додатку на пристрій з Windows, версія цієї програмної оболонки повинна бути новішою за 7.

Необхідне забезпечення робочих місць лікарів-консультантів ноутбуками для взаємодії з комплексом онлайн-консультацій із можливістю доступу до мережі інтернте через комп'ютерну мережу лікарні.

3.2.8.5 Вимоги до організаційного забезпечення

Системний адміністратор підприємства є відповідальною особою з контролю за працездатністю Систем, в тому числі комплексу онлайн-консультацій. Необхідно проводити для фахівців, що відповідають за комунікацію з користувачами, обов'язковий технічний інструктаж та організовувати їх навчання.

Рекомендовано розповсюдження інформації про можливість нового засобу комунікації з фахівцями організації через доступні для неї ресурси, такі як власний веб-сайт, сторінки у соціальних мережах, офіційні сторінки лікарні на інших сервісах, а також надання інформації у реєстратурі лікарні.

3.2.8.6 Вимоги до методичного забезпечення

Проект, що надається підприємству-замовнику, повинен обов'язково супроводжуватись методичними документами за наступними напрямками :

1. Структура передачі інформації через чат-бот;

2. Структура та засоби організації комунікації;

3. Інструкція та правила користування чат-ботом та системою в цілому.

3.2.9 Вимоги до захисту інформації

Система має бути налаштована таким чином, щоб забезпечити надійний постійний контроль над захищеністю даних користувачів таким чином, щоб унеможливити несанкційований доступ до даних, їх незаконне копіювання, використання, внесення змін та/або знищення.

При роботі з системою мають використовуватись новітні методи та алгоритми по захисту та безпеці інформації. Так, необхідне використання протоколів шифрування даних для забезпечення їх конфіденційності, таких як криптографічний протокол MTProto, що використовується в системі обміну повідомленнями в Telegram.

Одним з пріоритетних завдань системи має бути постійний контроль доступу до інформації, оскільки вона переважно містить персональні дані користувачів. За для цього має бути налаштовано аутентифікацію користувачів при доступі до функцій чат-боту, а також при доступі до баз даних та коду чат-боту на самому сервері.

В чат-бот має бути включено інформаційне вікно з повідомленням про уважне відношення до персональних даних, що використовуються для автентифікації та доступу до конфіденційних даних, та рекомендаціями з неприпустимості їх розголошення, поширення та необхідності прийняття мір для неможливості доступу до них сторонніх осіб.

3.2.10 Вимоги до ергономіки системи

Система має бути продуманої, логічно побудованою за для забезпечення максимального комфорту користування нею.

Інтерфейс чат-боту має бути максимально простим та зрозумілим, оскільки серед цільової аудиторії присутні користувачі, що мають невеликий досвід взаємодії з чат-ботами.

Має бути побудовано чітку логічну структуру взаємодії з чат-ботом, з можливістю навігації на різних етапах користування та мінімізацією кількості кроків необхідних для виконання поставлених завдань.

Відповіді та підказки чат-боту повинні бути чітко та зрозуміло сформульовані, щоб у користувача не виникало тлумачення, відмінного від задуманого.

Розміри шрифту та кнопок клавіатури мають бути зручними для читання та клікабельності як на ПК, так і на мобільній версії.

При першому запуску діалогу з чат-ботом повинно бути виведено повідомлення із коротким описом функцій чат-боту та інструкцій щодо їх використання.

Інтерфейс чат-боту має забезпечувати захист введеної користувачем інформації від помилок допущених ним. З цією метою мають бути прописані діалогові вікна на відповідних етапах взаємодії з чат-ботом, що закликатимуть до перевірки внесеної інформації та надаватимуть можливість її виправлення користувачем.

Програма чат-боту має функціонувати в асинхронному режимі за для забезпечення швидкого відгуку на дії користувача та безперебійної роботи.

3.3 Стислі відомості про комп’ютерну мережу об’єкту дослідження

При побудові комп’ютерної мережі організації КП «ДОКОЛ», керувались принципом оптимального співвідношення між ціною, якістю та потребами. Тобто вибір обладнання відбувався з урахуванням характеристик, що цілковито задовольняють потребам та вимогам до побудови комп’ютерної системи організації, та з урахуванням економічної доцільності, оскільки лікарня має в розпорядженні доволі обмежений грошовий ресурс.

В процесі побудови системи та розгортання мережі, схему логічної топології якої можна побачити на рисунку 3.1, було використано ряд протоколів та сервісів, таких як:

1. Сервіс DHCP для динамічного розповсюдження IP-адрес та інших параметрів серед вузлів мережі.

2. Протокол динамічної маршрутизації RIP за для встановленню зв’язку між вузлами мережі та мережею Internet.

3. Віртуальні локальні мережі для внутрішнього штучного розподілу вузлів системи по окремим відділам організації.

4. Сервіс віддаленого керування IoT пристроями через сервер для впроваджених підсистем відеоспостереження та пожежної небезпеки.

Крім того, було налаштовано можливість функціонування та доступу через мережу до власного веб-сайту організації та взаємозв’язок між основною організацією та її віддаленим підрозділом завдяки протоколу NAT.

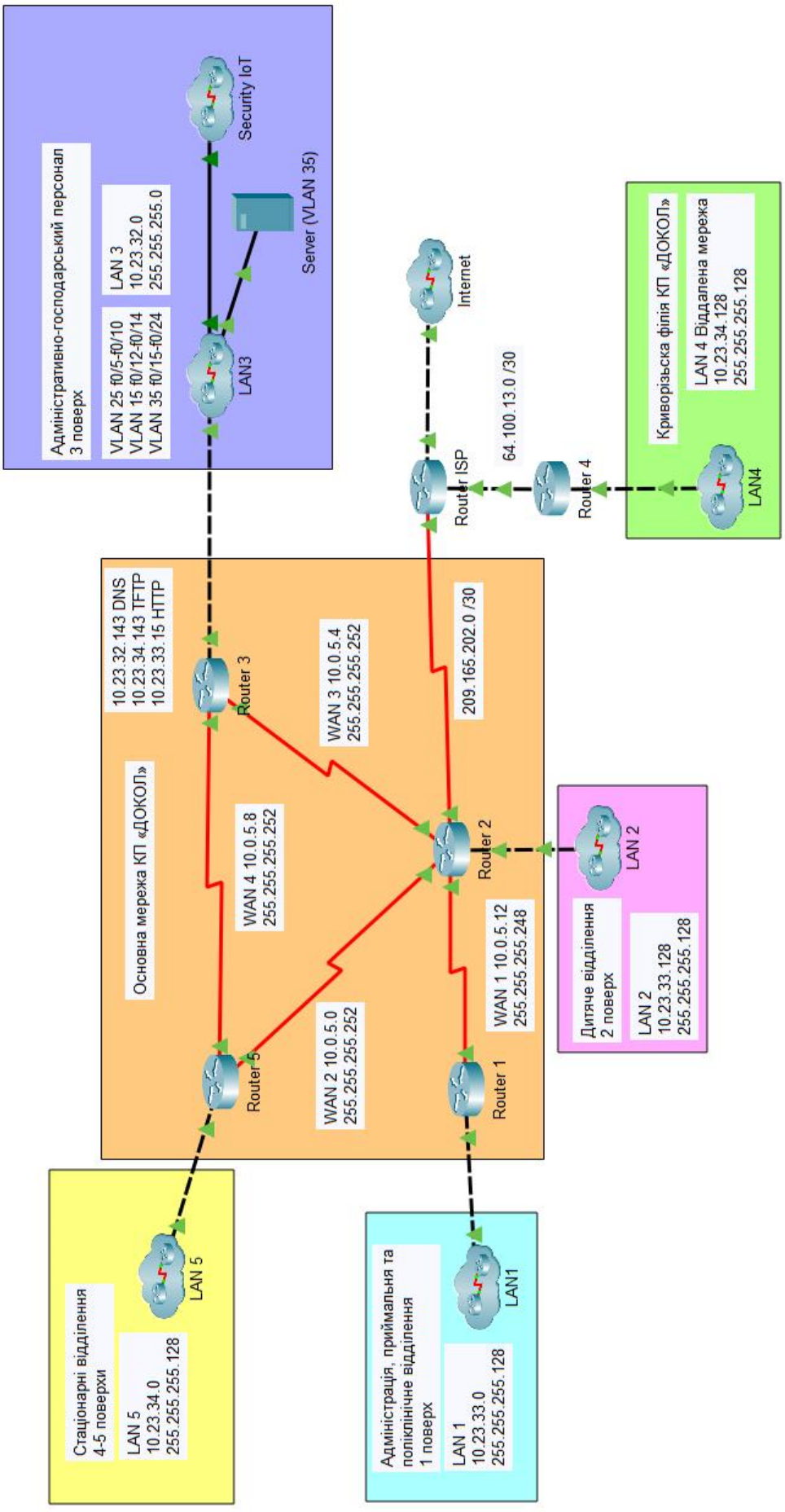

Рисунок 3.1 – Логічна топологія мережі КП «ДОКОЛ»

3.4 Структура взаємодій суб'єктів комплексу

Ґрунтуючись на цілях та меті дослідження, а також висновках та даних, отриманих у минулих розділах, нами було побудовано логічну структуру функціонування комплексу онлайн-консультацій на базі підприємства. Згідно з нею передбачається взаємодія між трьома умовними групами користувачів та базою даних підприємства, посередником в якому виступає чат-бот Telegram (рисунок 3.2).

Групи користувачів представлені пацієнтами, фахівцями-консультантами та фахівцями вищого рівня, що надають консультації в межах спілки офтальмологів. Представники кожної з цих груп звертаються до чат-боту за для проведення взаємодії з представниками іншої групи, а також в процесі цієї взаємодії за допомогою чат-боту надсилають різноманітні запити до бази даних.

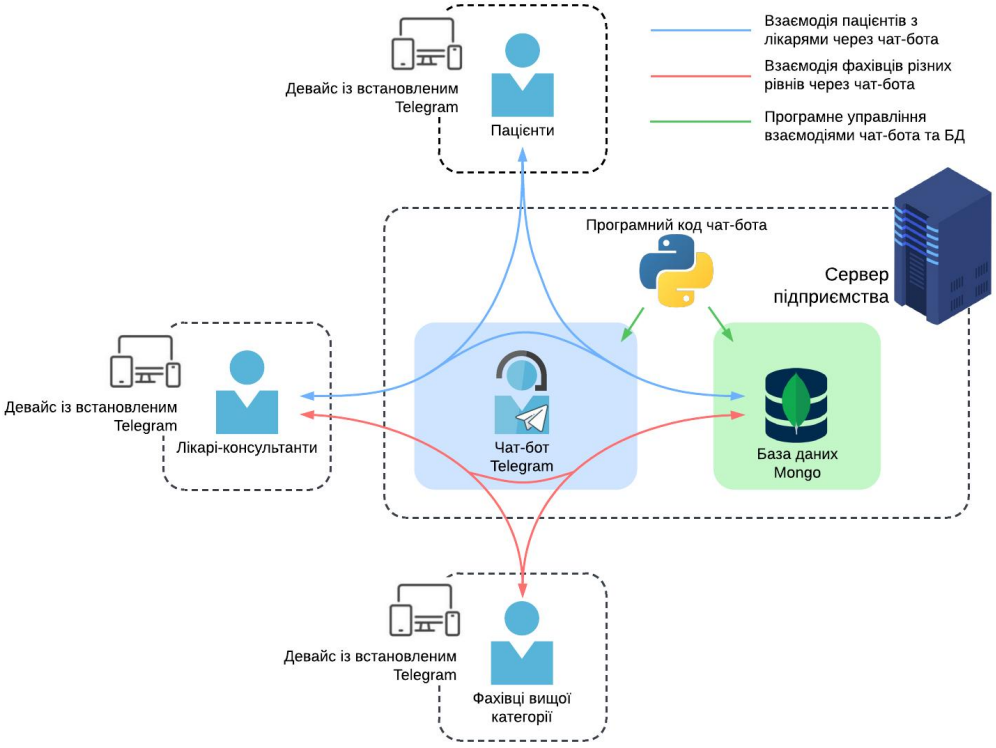

Рисунок 3.2 – Схема функціональної структури комплексу

Таким чином програма чат-боту керує потоками інформації між суб'єктами комплексу та виконує "диспетчерські" функції.

3.5 Специфікації апаратних засобів системи

В підприємстві КП «ДОКОЛ» вже налаштовано та функціонує комп’ютерна система із певними апаратними засобами. Досліджуваний комплекс онлайн-консультацій передбачає застосування цієї системи для своєї роботи та урахування можливостей його апаратного забезпечення.

Так, за для проведення подальшого дослідження, нами було отримано та проаналізовано ряд специфікацій пристроїв, що входять до комп’ютерної системи підприємства та прийматимуть участь в роботі комплексу онлайн-консультацій, а також пристроїв, що є складовими підсистем, які завдають навантаження на мережу та сервер системи (таблиця 3.1).

Таблиця 3.1 – Орієнтовні технічні характеристики пристроїв, використовуваних в комп’ютерній системі підприємства КП «ДОКОЛ»

№ п.	Найменування пристрою	Орієнтовні технічні характеристики пристрою
1.	Персональні комп’ютери (PC)	ОП: 8 ГБ Процесор: 4-6 ядерний, 3.7 - 4.2 ГГц Графічний адаптер: AMD Radeon Vega 7 Тип пам’яті: SSD, 240 ГБ або більше Підтримка LAN ОС: Windows 11
2.	Маршрутизатори	Вбудовані порти зі швидкістю: 10/100/1000 Мбіт/с Тип портів: RJ45, Serial Стандарт: 802.11 b/g/n
3.	Комутатори	Керований тип Наявні 24 фіксовані порти 10/100 Fast Ethernet та 2 – 10/100/1000 Gigabit Ethernet Встановлене ПЗ – LAN Base Розширені можливості управління QoS Вбудовані функції безпеки із контролем доступу в мережу
4.	Сервер	ОП: 16 GB Тип ОП: DDR3 (2 x 8 GB) Два процесори Intel Xeon E5-2650 v2 Мережевий контролер: 1Gb Ethernet, 2 порти

Існуюче технічне забезпечення об’єкта КП «ДОКОЛ» в цілому задовільняє потребам впровадження комплексу онлайн-консультацій. Оскільки робочі місця лікарів-консультантів підключені до мережі лікарні та мають вихід в Інтернет (рисунок 3.1), а в самій мережі лікарні закладено резерв для підключення нових пристроїв до комп’ютерної системи, нагальною потребою є лише обладнання робочих місць лікарів, що будуть надавати консультації, персональними комп’ютерами з встановленим додатком Telegram.

Для роботи самого комплексу передбачається його впровадження на один з фунціонуючих в організації серверів за умови його спроможності підпримувати нове навантаження без шкоди для вже існуючих процесів. В процесі подальшого аналізу навантажень від роботи комплексу, нами буде зроблено висновки про можливість подальшого використання існуючого серверу підприємства або доцільність додавання нових потужностей.

Технічні характеристики пристроїв, необхідних в ході подальшого дослідження впроваджуваного комплексу наведено у таблиці 3.2.

Таблиця 3.2 – Технічні характеристики пристроїв необхідних для впровадження комплексу онлайн-консультацій

№ п.	Найменування пристрою	Орієнтовні технічні характеристики пристрою
1.	Персональні комп’ютери (PC)	ОП: 8 ГБ Процесор: 4-6 ядерний, 3.7 - 4.2 ГГц Графічний адаптер: AMD Radeon Vega 7 Тип пам’яті: SSD, 240 ГБ або більше Підтримка LAN ОС: Windows 10 або вище
2.	Сервер	Об’єм оперативної пам’яті: 16 GB або більше Тип ОП: DDR3 (2 x 8 GB) Два процесори Intel Xeon E5-2650 v2 Мережевий контролер: 1Gb Ethernet, 2 порти

3.6 Висновки розділу синтезу системи

По завершенню даного етапу дослідження нами було встановлено детальні вимоги до функціонування, структури та забезпечення комплексу онлайн-консультацій.

Було також проаналізовано та обґрунтовано мережеву структуру та апаратне забезпечення комп’ютерної системи підприємства КП «ДОКОЛ», що є об’єктом нашого дослідження.

За результатами цього аналізу та згідно із сформульованими вимогами, нами було створено логічну структуру функціонування системи онлайн-консультацій та взаємодії між її суб’єктами.

4 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМПЛЕКСУ ОНЛАЙН КОНСУЛЬТАЦІЙ

4.1 Функціональне призначення програмного забезпечення

Програмне забезпечення комплексу онлайн-консультацій комп'ютерної системи КП «ДОКОЛ» розробляється з метою створення програмних засобів для проведення консультацій через розповсюджені сервіси із можливістю безпечного обміну конфіденційними даними та файлами, їх обробці та зберіганню на сервері підприємства у базі даних, та отриманням онлайн фідбеку.

4.2 Технічні характеристики програмного забезпечення

4.2.1 Постановка завдання на розробку програми

Розробити програму для чат-боту, що функціюватиме на базі розповсюдженого месенджеру Telegram, та буде забезпечувати функції передачі даних та файлів між користувачами, їх обробку, завантаження та відвантаження з бази даних серверу підприємства.

4.2.2 Алгоритм функціонування програми

Оскільки серед вимог до комп'ютерної системи підприємства, до складу якої також входить комплекс онлайн-консультацій, є безперервне функціонування протягом робочого дня, программа повинна бути побудованою таким чином, щоб забезпечувати її циклічну роботу весь цей час.

4.2.3 Опис організації вхідних та вихідних даних

Отримувані від користувачів дані зберігаються у відповідні колекції документами, шаблони яких прописані у коді програми та використовуються у відповідності із алгоритмами, заданими для кожного конкретного випадку.

Для отримання даних з БД використовується пошук за значеннями окремих полей серед документів відповідних колекцій по запиту програми.

4.2.4 Рішення з використання програмних засобів

В процесі розробки програмного забезпечення для функціонування комплексу онлайн-консультацій комп'ютерної системи підприємства КП «ДОКОЛ» нами було обрано мову програмування Python, яка належним чином підходить для розробки чат-ботів на базі Telegram API та має достатньо великий набір бібліотек, що ставатимуть у нагоді в процесі розробки.

Середовищем, що застосовується при розробці коду програми, було обрано Visual Studio Code від компанії Microsoft завдяки його зручності та безкоштовній моделі розповсюдження.

Для збереження та отримання даних програмою чат-боту, нами було використано MongoDB – документно-орієнтовану базу даних NoSQL, що зберігає дані у форматі JSON і забезпечує гнучке та масштабоване зберігання.

Для зручного відображення внесених у БД даних, нами було використано також MongoDB Compas – графічний інтерфейс для керування MongoDB, що дозволяє переглядати, аналізувати та взаємодіяти з даними в зручному візуальному форматі.

Маючи на меті скоротити час розробки та оптимізувати програмне забезпечення, при розробці програми нами було використано наступний набор бібліотек:

1. Asyncio – бібліотека для налаштування асинхронної роботи функцій програми, що дозволяє паралельно виконувати різні задачі.

2. Logging – фіксація подій програми, що допомагає в процесі її налагодження.

3. Aiogram (версія 3.13) – одна з доступних бібліотек для створення телеграм-ботів на Python із підтримкою зручного керування подіями та діалогами, та асинхронного формату роботи.

4. Pymongo – бібліотека для роботи з базою даних MongoDB, яка дозволяє взаємодіяти програмі, написаній мовою Python, із цим типом бази даних.

5. GridFS – спеціальний інструмент для завантаження та відвантаження великих файлів у MongoDB шляхом розбивання їх на частини (чанки) для зручного зберігання.

6. UUID – бібліотека для генерації унікальних ідентифікаторів, що корисно для ідентифікації даних або користувачів у системі та було застосовано на одному з етапів розробки.

4.3 Опис розробленої програми

4.3.1 Відомості про структуру програми

Структурно, програма чат-боту Telegram складається з декількох модулів, кожен з яких виконує свої задачі, такі як запуск чат-боту, робота його роутерів та вбудованих функцій, зовнішній вигляд та функціонал кнопок клавіатури, тощо, а також допоміжних директорій, необхідних для коректного функціонування програми:

- **kpdokolBot.py –** основний файл програми, що відповідає за запуск та неперервну роботу бота та отримує інструкції для роботи з файлів handlers.py та config.py;
- **handlers.py** – файл програми, що містить коди класів, функцій та хендлерів, що необхідні для роботи програми, та отримує інструкції щодо формування клавіатур з файлу keyboards.py;
- **keyboards.py –** файл програми, що містить набір клавіатур, необхідних для функціонування боту та навігації між його розділами;

– **config.py –** файл, що зберігає токен та, у перспективі, деякі інші налаштування боту, які не мають потрапити у сторонні руки у випадку, наприклад, відкритого поширення коду програми бота;
– Директорія **app** – організаційна директорія, що містить в собі системні файли, необхідні для функціонування програми, такі як **handlers.py та keyboards.py;**
– Директорія **tmp** – директорія, що створюється в процесі функціонування програми та використовується у якості тимчасового місцезнаходження файлів.

4.3.2 Функціональне призначення програми

Програма призначена для надання можливості чат-боту підприємства вести діалог із користувачем, приймати та надсилати повідомлення із файловими даними в процесі цього діалогу та взаємодіяти із базою даних підприємства, що було застосовано для подальших експериментальних досліджень комп’ютерної системи підприємства.

4.3.3 Опис алгоритму роботи програми чат-боту

На схемах, що зображено на рисунках 4.1 – 4.8, наведено алгоритм роботи програми, який починає свою роботу після введення користувачем команди /start. Програма складається з декількох гілок, кожна з яких відповідає за свій сегмент роботи програми та функції.

Нижче наведено перелік умовних позначень на цих схемах:

1. ГМ – етап переходу назад у початок програми (в головне меню);
2. П – перехід у гілку для пацієнта, в якій відбувається подання ним інформаційних та файлових даних;
3. Ф1 – перехід у гілку фахівця, що стосується надання ним консультації пацієнту;

4. Ф2 – перехід у гілку фахівця, що стосується створення звернення фахівцем на отримання консультації від спілки офтальмологів;
5. Ф3 – перехід до гілки з обмеженим доступом фахівців для призначення дати та часу проведення наступної конференції спілки.

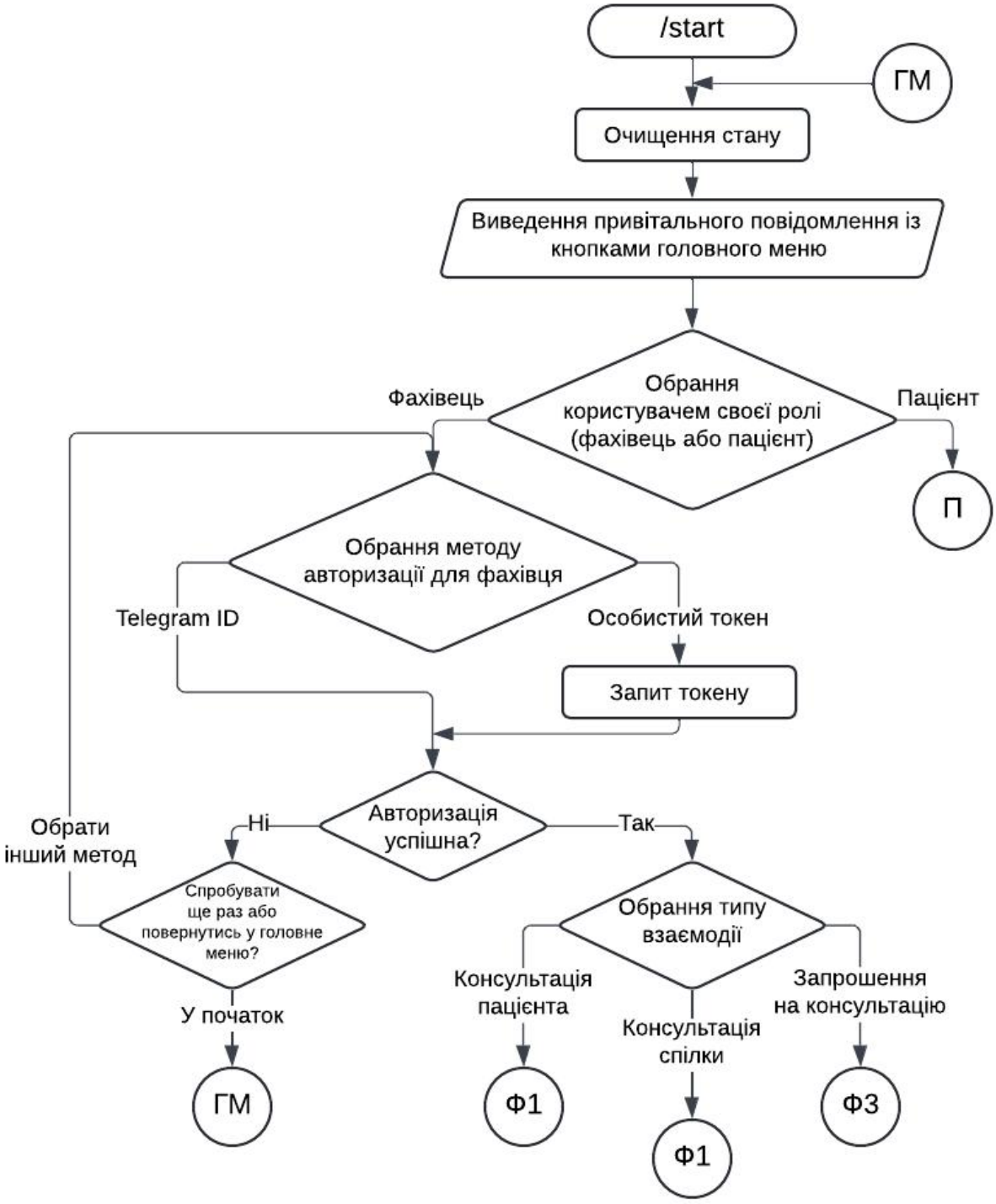

Рисунок 4.1 – Основна гілка програми, що бере початок з головного меню

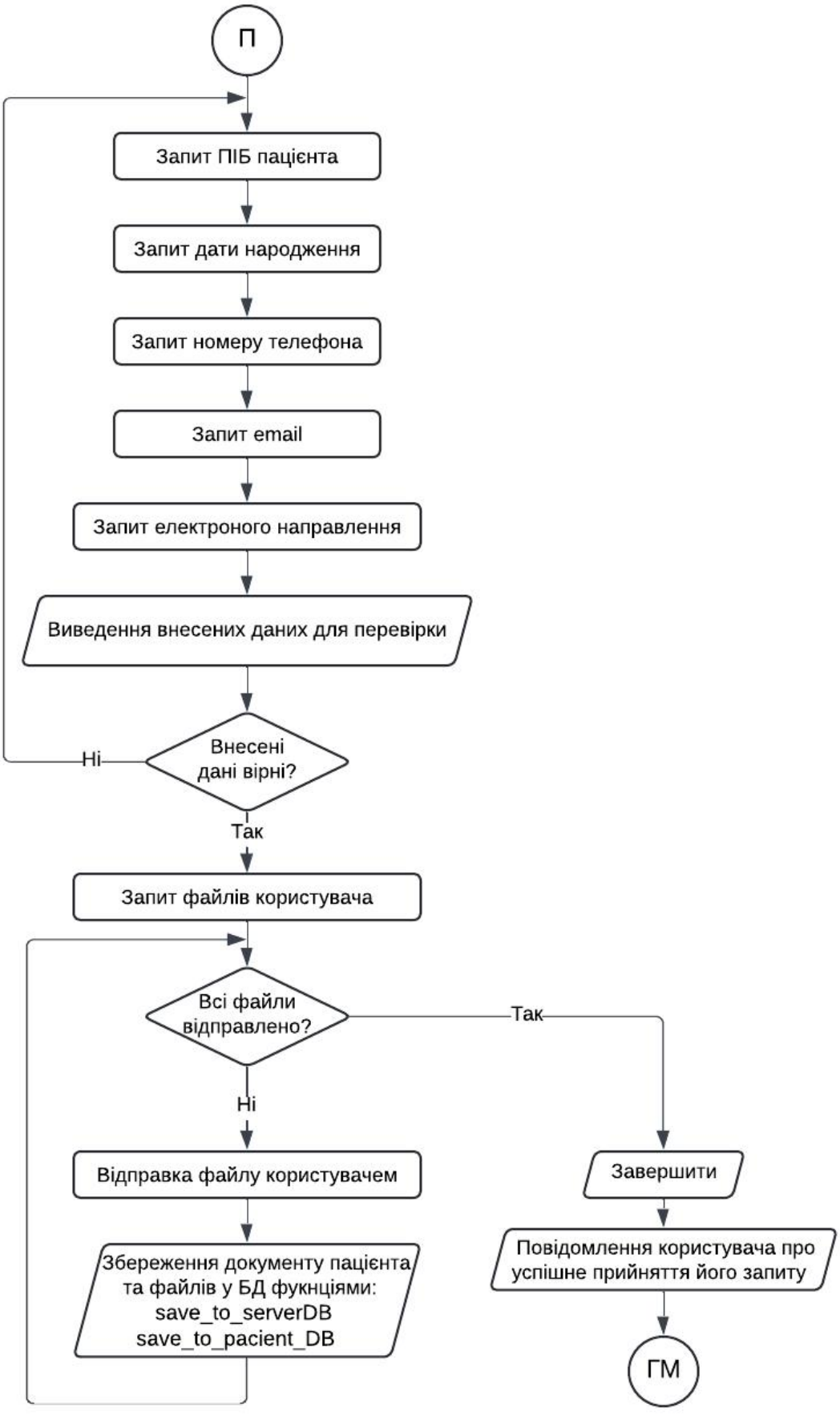

Рисунок 4.2 – Гілка запису пацієнта

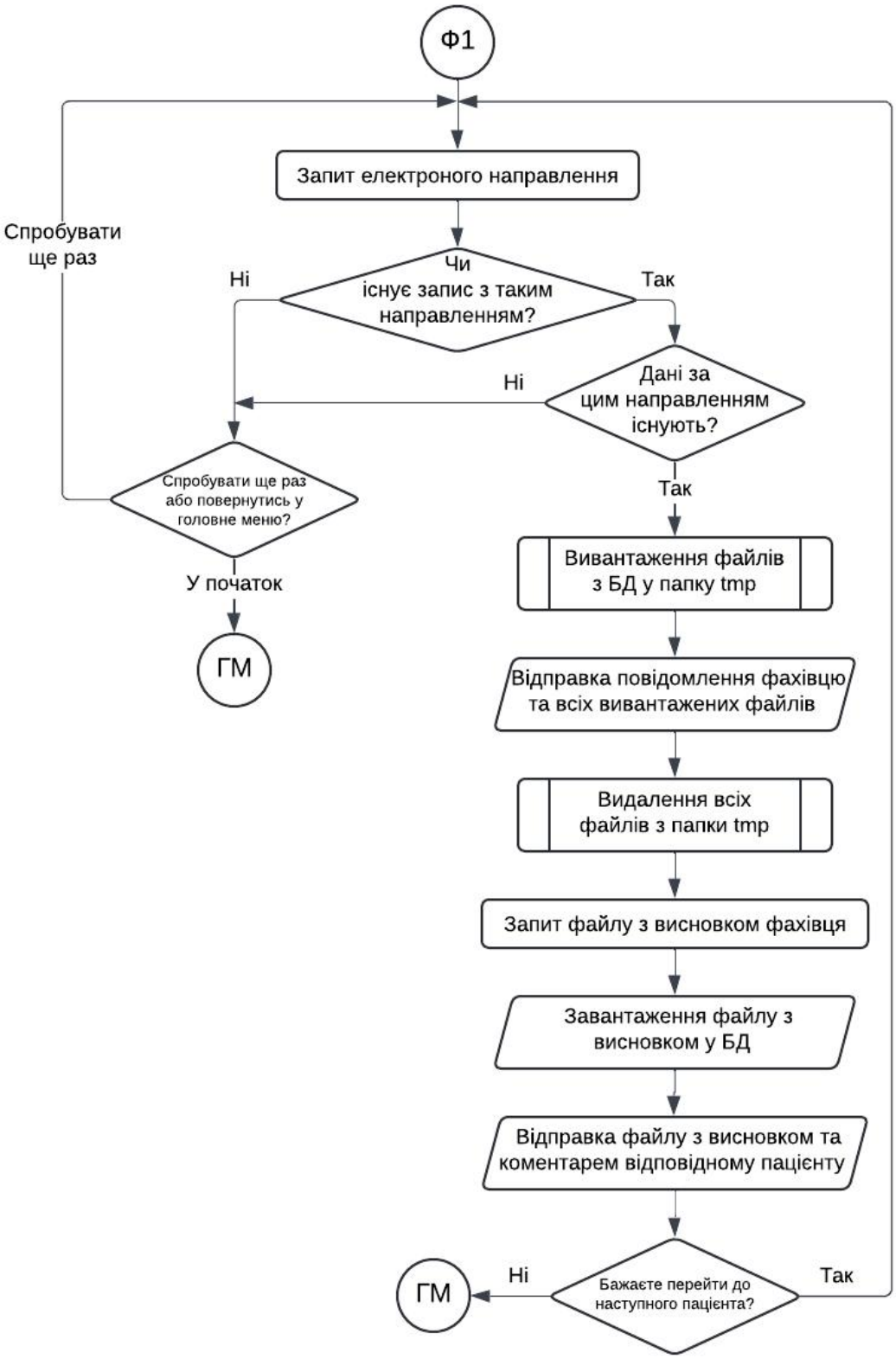

Рисунок 4.3 – Гілка консультації пацієнта фахівцем

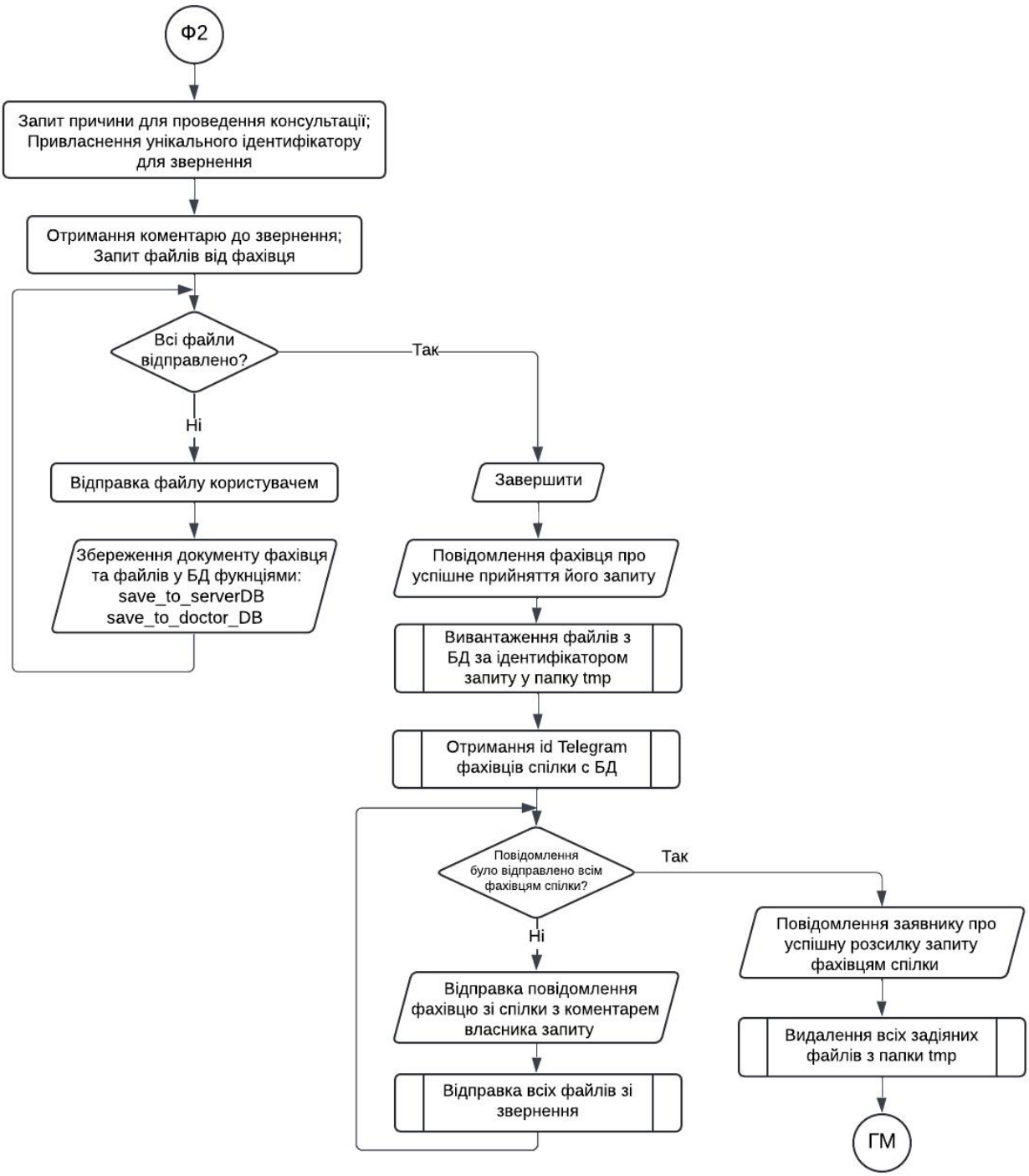

Рисунок 4.4 – Гілка звернення фахівця про проведення конференції спілки

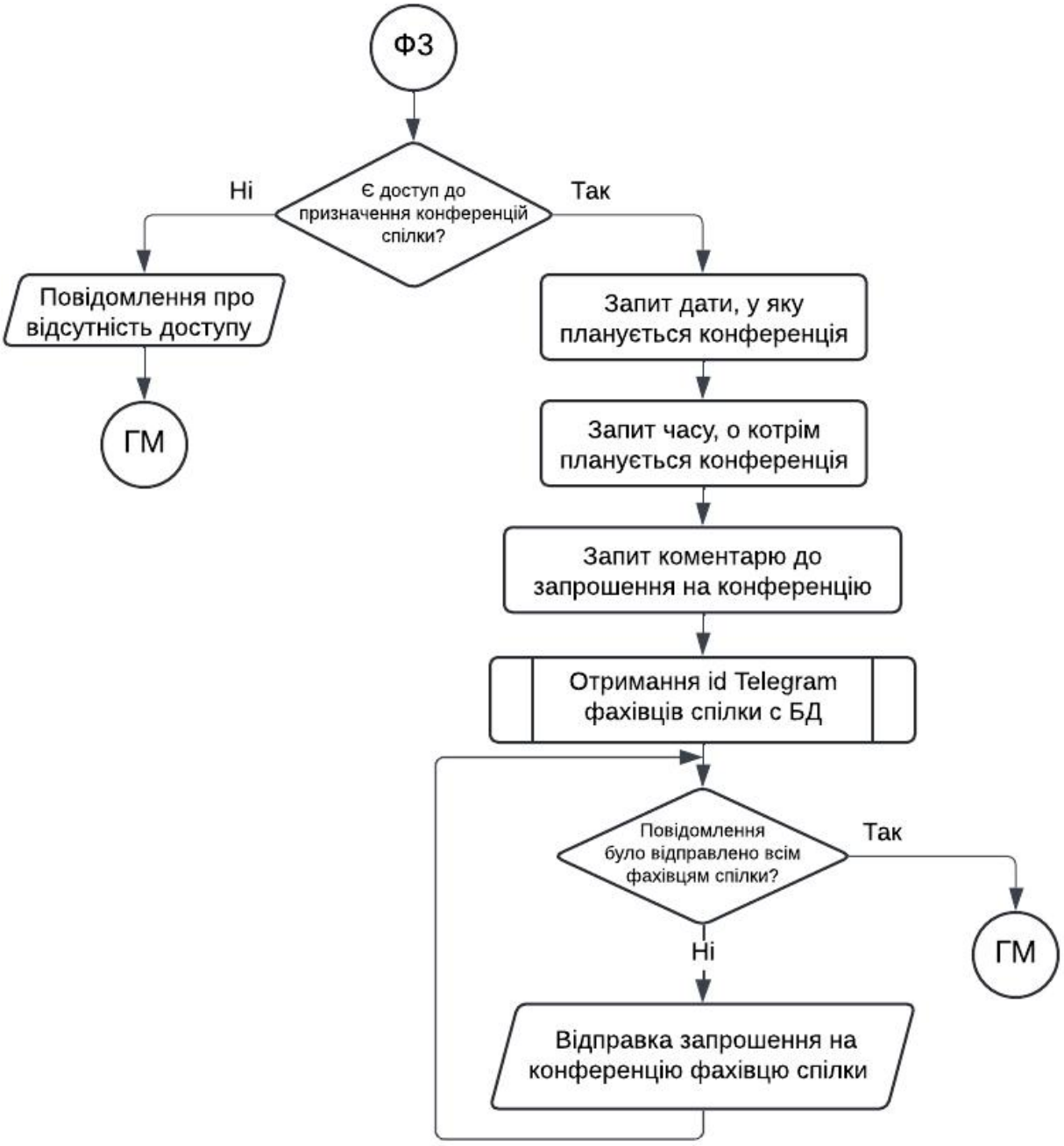

Рисунок 4.5 – Гілка призначення та розсилання запрошення на конференцію

Рисунок 4.6 – Функція запису даних пацієнта в БД

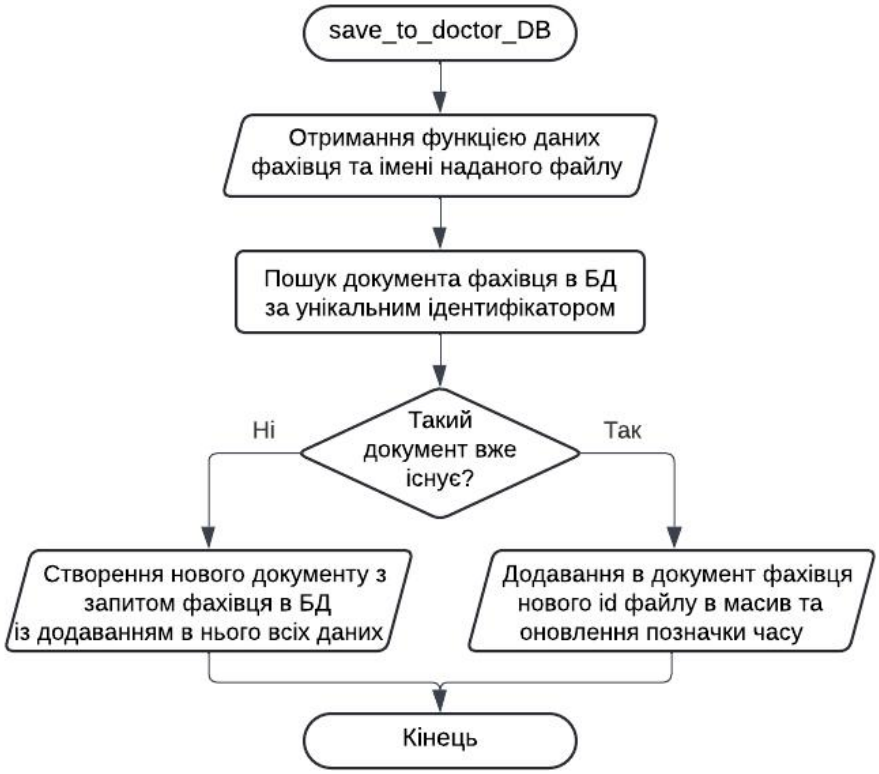

Рисунок 4.7 – Функція запису даних фахівця в БД

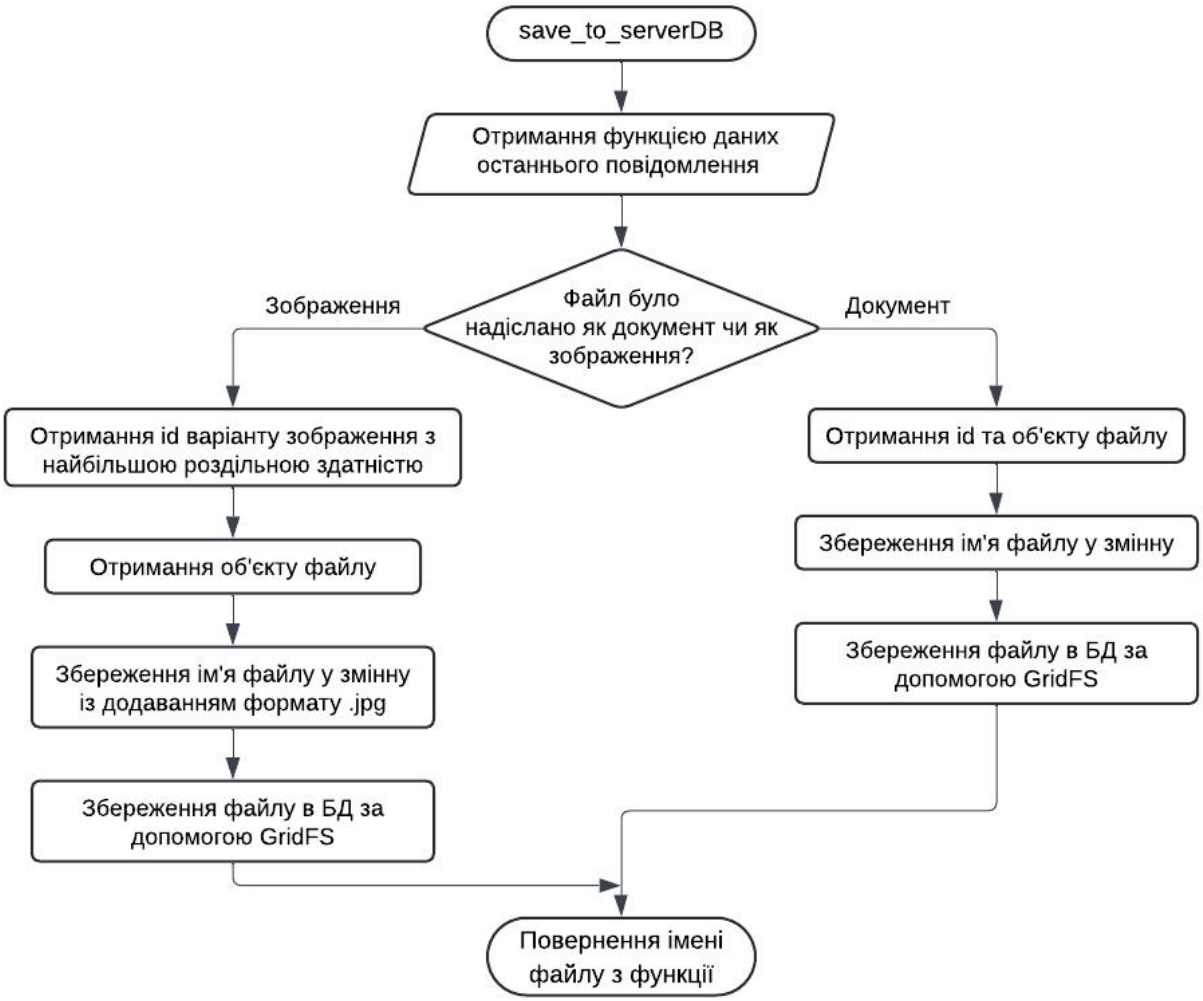

Рисунок 4.8 – Функція запису файлів в БД в залежності від типу

4.3.4 Використані технічні засоби

Для запуску роботи та тестування програми чат-боту та бази даних в рамках цієї роботи використовується сервер на базі ноутбуку з заздалегідь встановленою та розгорнутою БД Mongo та наступними системними характеристиками:

– ОС: Windows 10;

− Процесор: Intel Core i5-8300H;

− Оперативна пам’ять: 8 Гб;

− Необхідний дисковий простір для розгортки програми: 100 Мб;

Для використання функцій програми необхідний пристрій з встановленим додатком Telegram Android або Desktop версії та можливістю інтернет-з’єднання.

4.3.5 Виклик і завантаження

Для запуску програми на сервері використовується файл kpdokolBot.exe, що запускає роботу чат-боту. Також, паралельно з запуском програми чат-бот, на сервері підприємства слід запустити роботу БД Mongo.

Для початку роботи з програмою з боку користувача, слід запустити діалог з чат-ботом в додатку Telegram та ввести команду /start. Його можна знайти за адресою «@kpdokolBot».

4.3.6 Вхідні дані програми чат-боту

Для початку роботи з ботом користувачем використовується команда /start.

Під час створення запиту користувачем, программа отримує з діалогу текстові повідомлення від користувача з його даними та файли, оброблює та зберігає їх за його згодою. Крім того, на певних етапах роботи, програма автоматично отримує інформацію про id користувача Telegram.

Для надсилання відповідей, що містять в собі дані та файли користувачів, бот використовує в якості джерела цих даних базу даних, в якій ця інформація та файли зберігаються.

При отриманні файлових даних, програма автоматично з’ясовує тип надісланого користувачем файлу та оброблює його відповідним чином для подальшого збереження у БД за допомогою спеціалізованих бібліотек.

4.3.7 Вихідні дані програми чат-боту

Взаємодія з програмою чат-боту та навігація за його гілками здійснюється шляхом натискання користувачем виведених програмою

кнопок клавіатури, що розміщуються ботом під рядком для введення повідомлень або в середині текстових повідомлень від бота.

Програма чат-боту надає текстову та файлову інформацію користувачам в процесі своєї роботи, як у вигляді відповідей в процесі конкретного діалогу, так і в якості розсилки після виконання певних вимог. Повний перелік текстових відповідей чат-боту наведено в підрозділі 4.4 у вигляді демонстраційних матеріалів.

Програма зберігає та надсилає інформаційні дані та файли у базу даних серверу з метою їх подальшого використання, у випадку з файлами оброблюючи їх у доступний для подальшого запису в БД вигляд.

4.4 Опис та демонстраційні матеріали роботи функціонування ПЗ

Чат-бот, створений нами для забезпечення функціонування онлайн консультацій та програмний код якого наведено у додатку А, має формат діалогу з користувачем.

Потрапляючи у початок діалогу з чат-ботом, у нас є можливість обрати дві гілки для діалогу (рисунок 4.9).

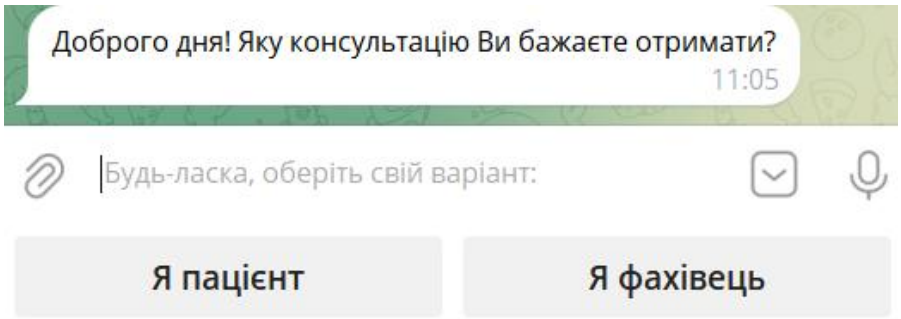

Рисунок 4.9 – Початок діалогу з користувачем

4.4.1 Функції чат-боту для пацієнта

Спочатку розглянемо гілку для пацієнта. Доступ до неї не обмежується. При обранні варіанту «Я пацієнт», починається діалог, в процесі якого пацієнту пропонується ввести власні особисті дані, а саме: ПІБ, дату

народження, номер телефону, пошту (за бажанням) та, найважливіше – електронне направлення, за яким далі буде здійснюватись консультація пацієнта фахівцем лікарні (рисунок 4.10).

Після надання всіх необхідних особистих даних, пацієнту буде запропоновано перевірити їх та продовжити у випадку, якщо всі дані було внесено вірно (рисунок 4.11). В іншому випадку, пацієнт матиме можливість внести дані повторно.

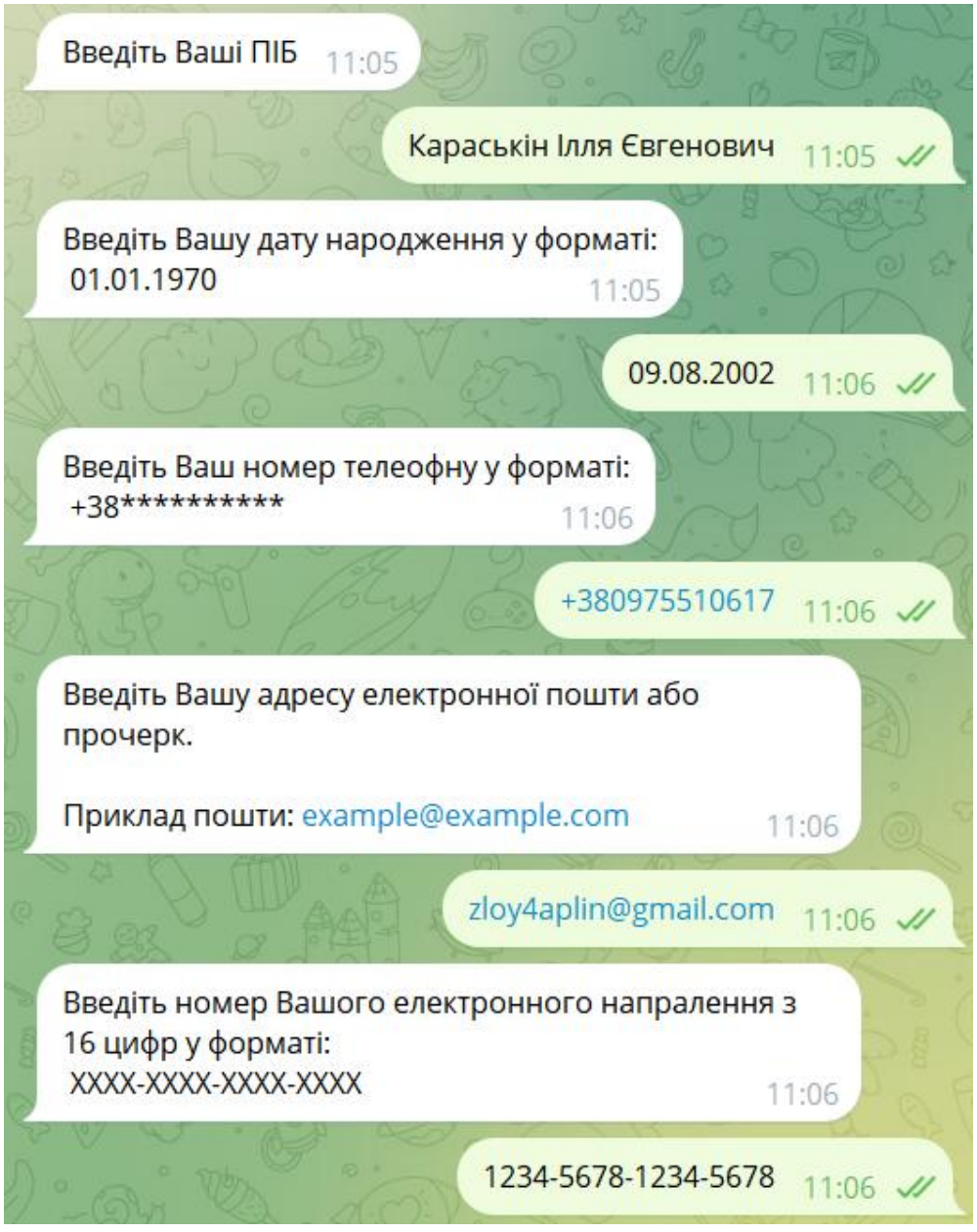

Рисунок 4.10 – Процес надання особистих даних пацієнтом

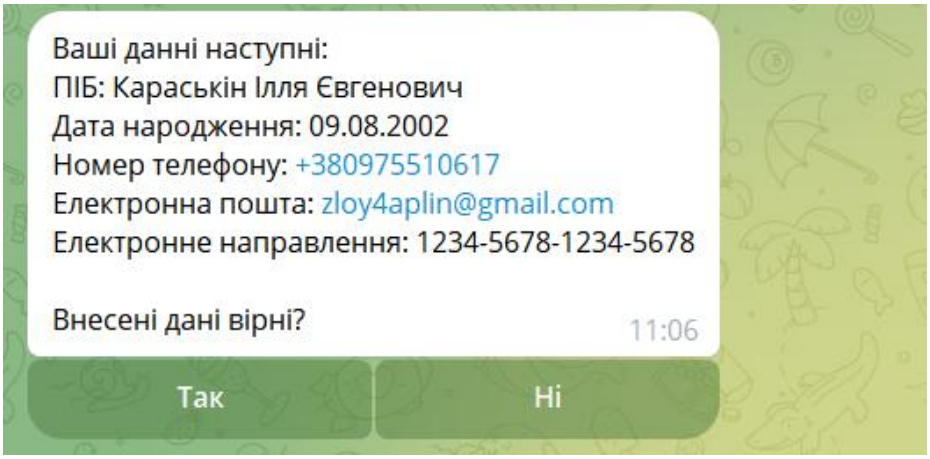

Рисунок 4.11 – Перевірка внесених даних пацієнтом

Після підтвердження внесених пацієнтом даних, йому буде запропоновано внести файли з результатами своїх обстежень, такі як знімки, висновки та ін.. Пацієнт може відправляти стільки файлів, скільки йому необхідно для отримання консультації. Після завершення надсилання файлів, пацієнт може припинити цю операцію, натиснувши на відповідну кнопку (рисунок 4.12).

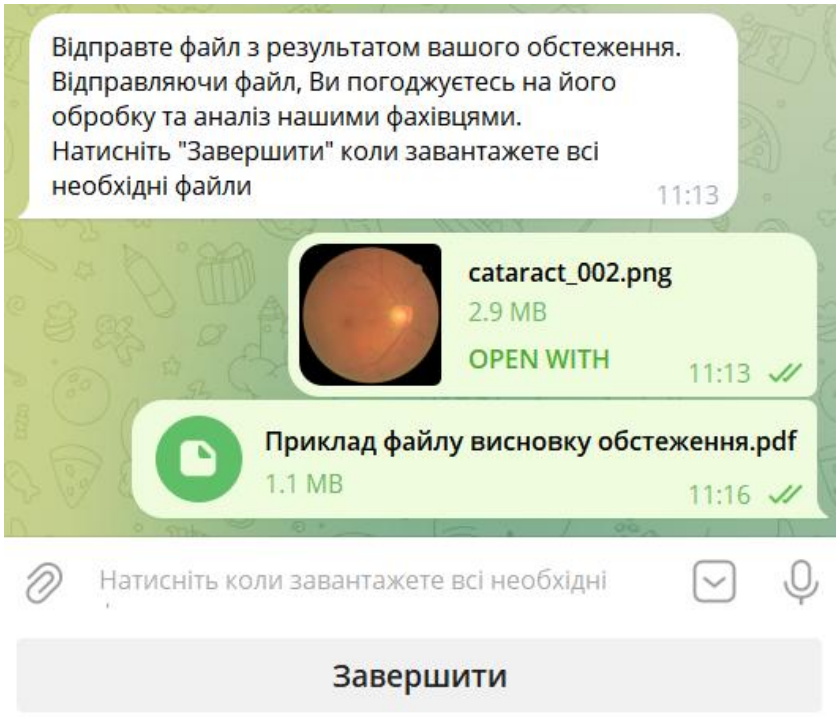

Рисунок 4.12 – Запит та процес надсилання різних типів файлів пацієнтом

В процесі надсилання, файли та їх імена, разом із даними пацієнта зберігатимуться в в відповідних колекціях в базі даних на сервері підприємства (рисунок 4.13 – 4.15). При збереженні файлів використовуються автоматично створювані GridFS колекції: fs.files та fs.chunks. В першій зберігається інформація про завантажений файл, в другій – файл у бінарному вигляді, розбитий на чанки та розподілений за декількома документами в колекції. Для демонстрації запису даних у БД тут і далі використано інтерфейс MongoDB Compas.

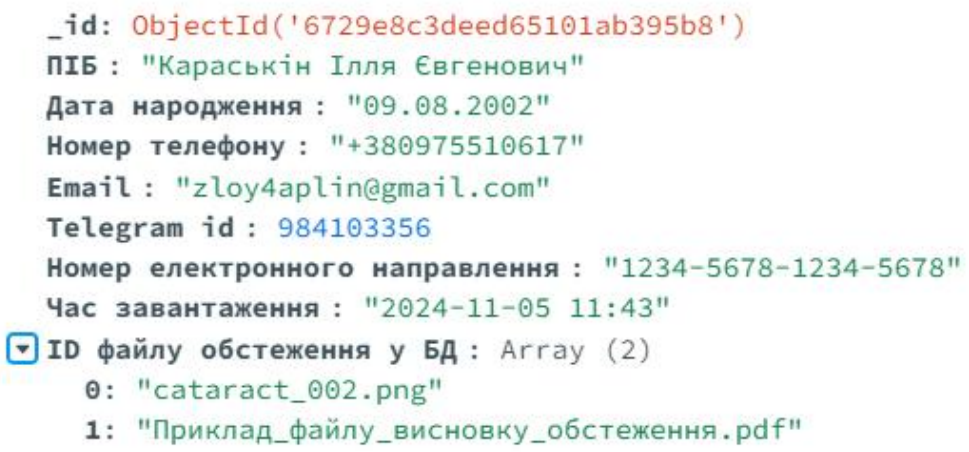

Рисунок 4.13 – Створений за запитом пацієнта документ у БД

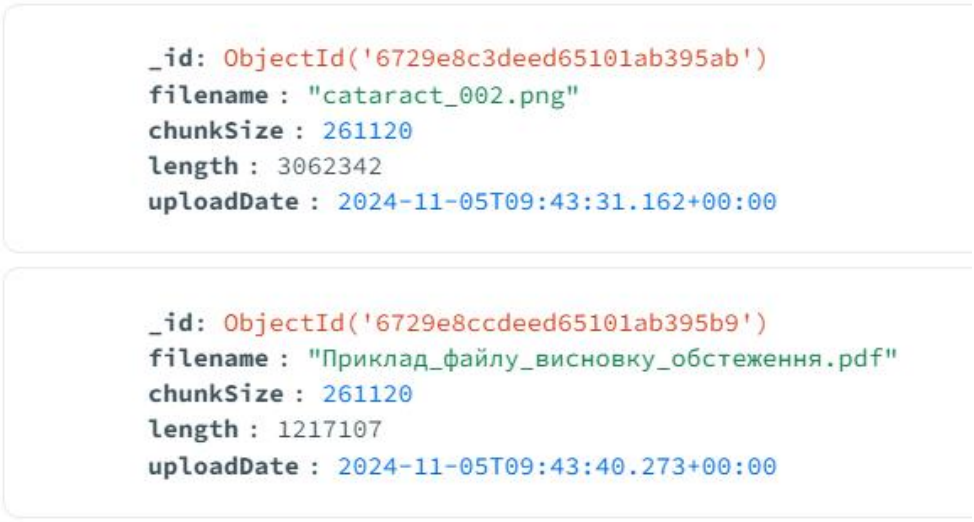

Рисунок 4.14 – Запис із завантаженими файлами в колекції fs.files

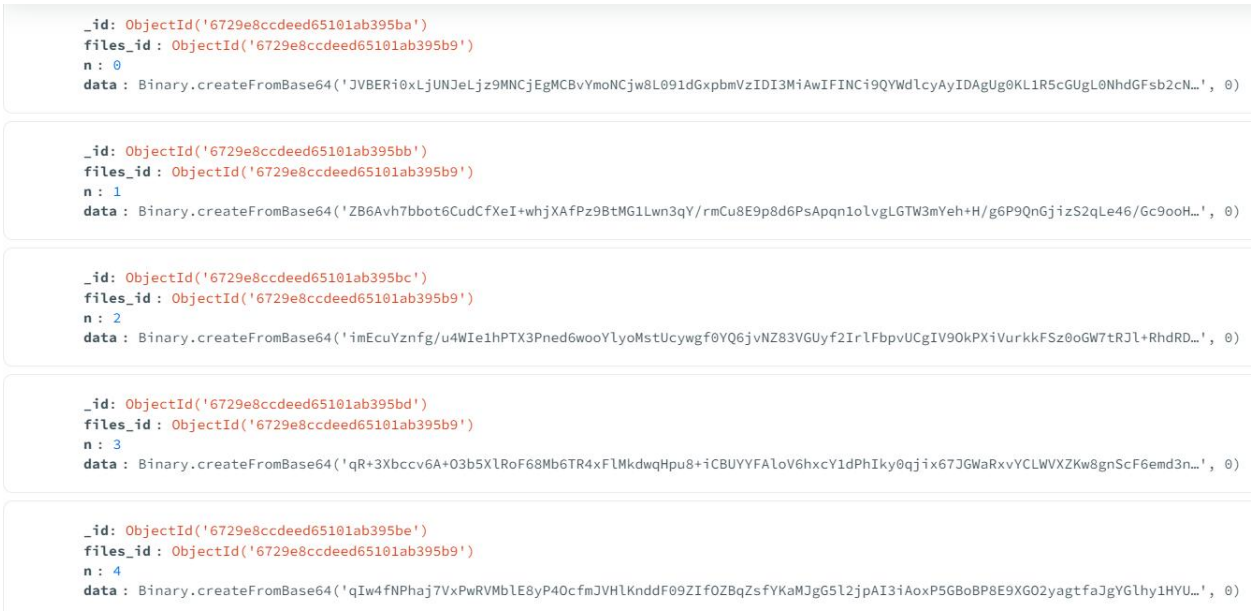

Рисунок 4.15 – Приклад запису єдиного файлу у БД в колекцію fs.chunks

Після завершення операції надсилання файлів пацієнтом, йому буде надіслано відповідне повідомлення про прийняття заявки на консультацію та короткі інструкції щодо його наступних дій (рисунок 4.16)

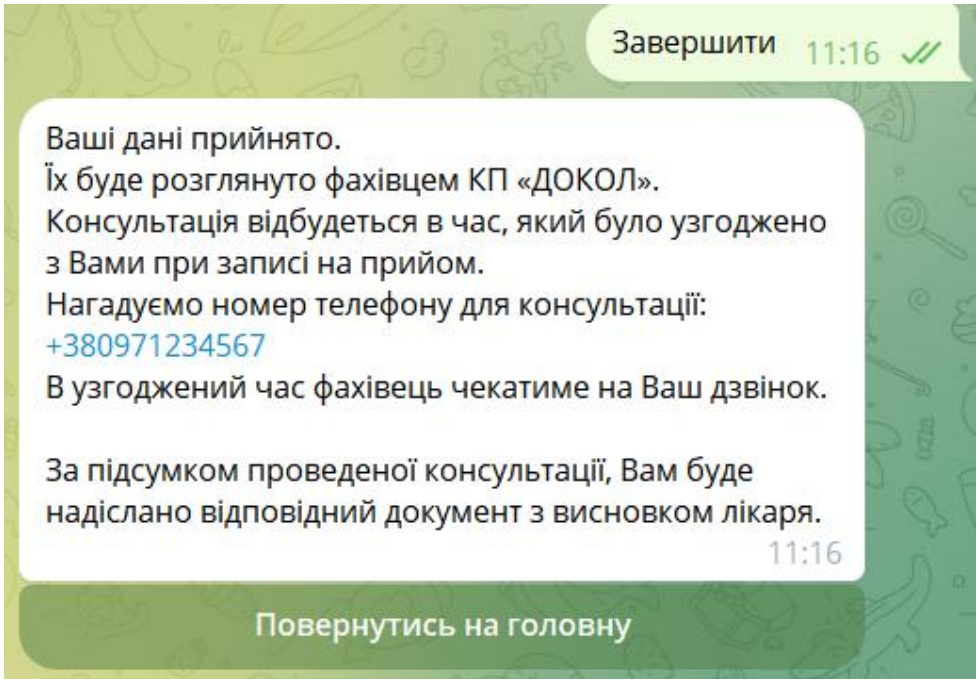

Рисунок 4.16 – Повідомлення про прийняття запиту пацієнта

4.4.2 Функції чат-боту для фахівця

Тепер розглянемо гілку для фахівця. Доступ до функцій, які стосуються фахівця спілки офтальмологів, обмежується за допомогою білого списку, що знаходиться у БД на сервері підприємства (рисунок 4.17)

Рисунок 4.17 – Приклад структури білого списку у БД підприємства

Авторизація за білим списком через чат-бот є вимогою для доступу до функцій фахівця та здійснюється одразу при переході до цієї частини чат-боту (рисунок 4.18). Вона може здійснюватись як автоматично – за id Telegram користувача (рисунок 4.19), що спілкується з чат-ботом, так і за допомогою введення особистого токену користувача – наприклад у разі необхідності зайти через власний обліковий запис з іншого акаунту Telegram (рисунок 4.20). У випадку, якщо авторизація не буде успішною, чат-ботом

буде запропоновано виконати ряд дій по усуненню проблеми з авторизацією (рисунок 4.21).

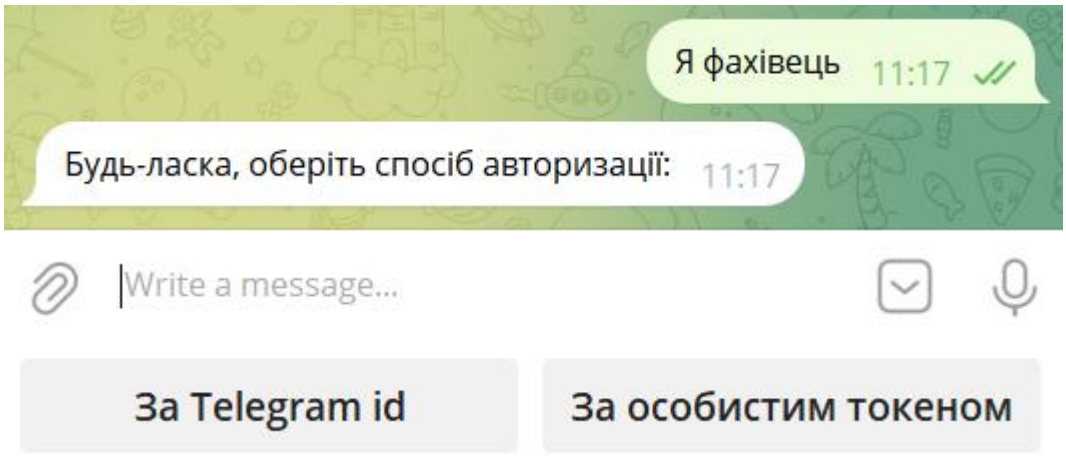

Рисунок 4.18 – Запит авторизації для доступу до функцій фахівця

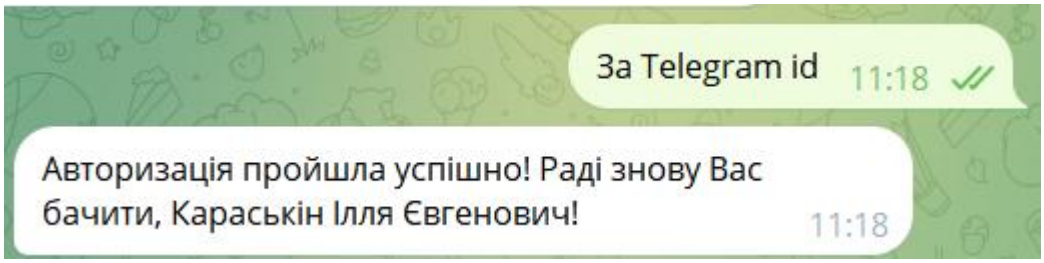

Рисунок 4.19 – Успішна авторизація за Telegram id користувача

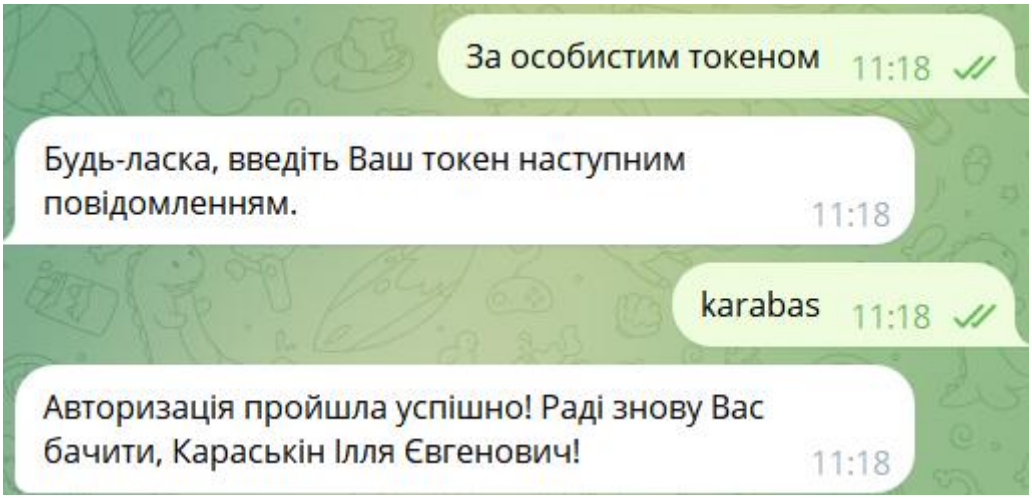

Рисунок 4.20 – Успішна авторизація за особистим токеном користувача

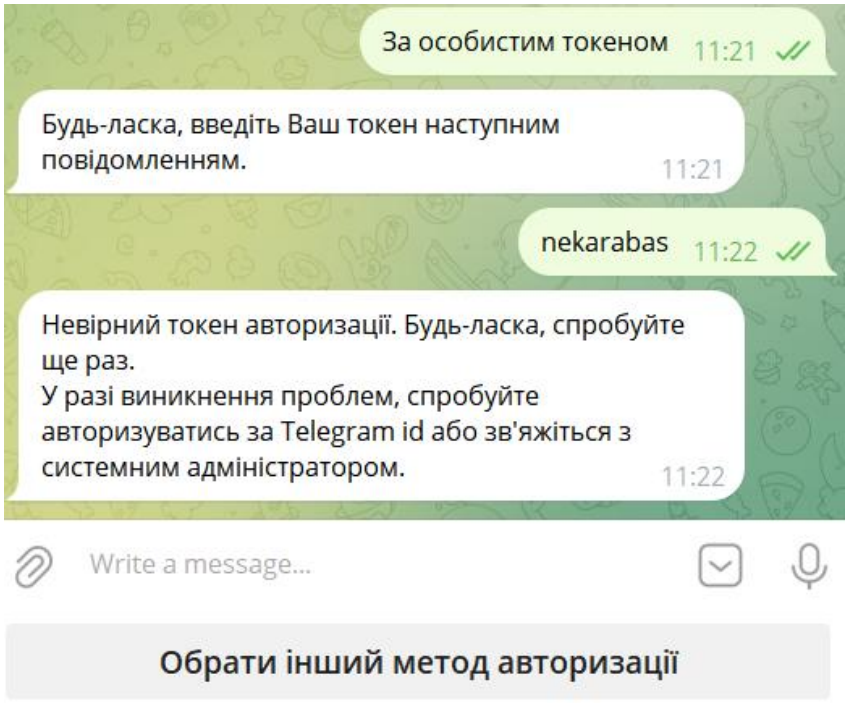

Рисунок 4.21 – Помилка авторизації на прикладі невірного токену

В разі успішної авторизації, автоматично встановленому фахівцю буде запропоновано обрати з трьох можливих варіантів для взаємодії з чат-ботом (рисунок 4.22).

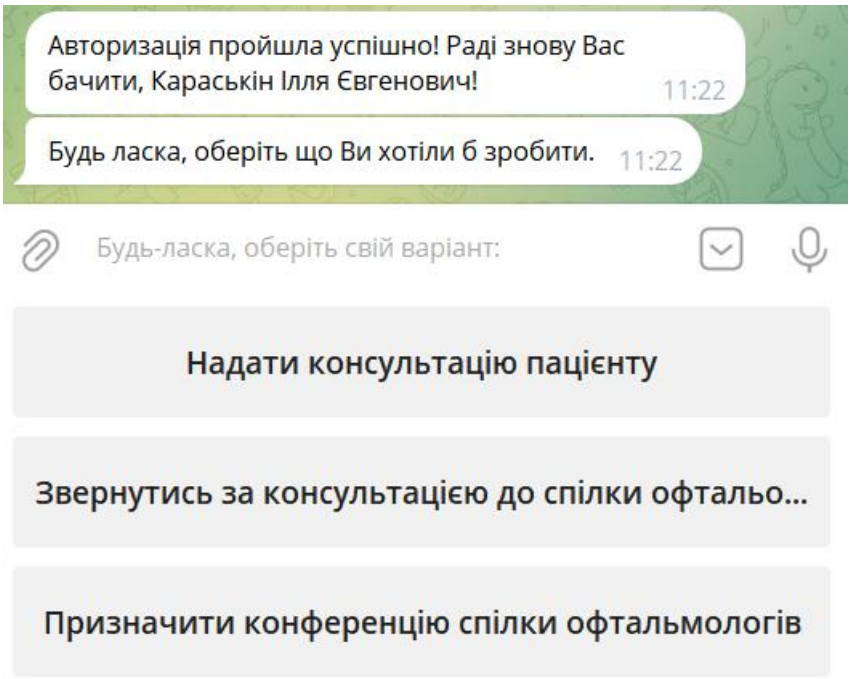

Рисунок 4.22 – Меню фахівця

4.4.2.1 Надання консультації пацієнтам

Якщо фахівець обирає варіант надання консультації пацієнту, йому буде запропоновано внести електронне направлення пацієнта, за яким його було записано на прийом до цього лікаря та яке було внесено пацієнтом при заповненні форми для пацієнтів у чат-боті (рисунок 4.23). Для демонстрації було використано форму, заповнену пункті 4.4.1.

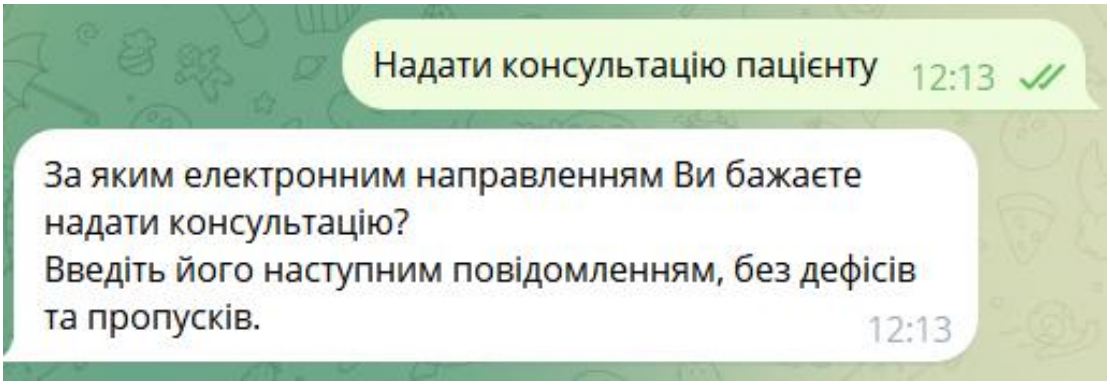

Рисунок 4.23 – Запит електронного направлення для надання консультації

Після внесення направлення, у разі наявності запису з таким номером, лікарю буде надіслано відповідні файли, отримані з цього запису, та запропоновано по завершенню онлайн-консультації надати у діалог з чат-ботом файл з висновком, який зазвичай створюється та заповнюється лікарем (рисунок 4.24). Цей файл із відповідним повідомленням, буде автоматично надіслано пацієнту, який отримав онлайн-консультацію (рисунок 4.25).

Після надсилання файлу з висновком лікаря, йому буде запропоновано обрати наступного пацієнта, що перенесе його на етап внесення наступного електронного направлення, або повернутись у головне меню, якщо він завершив свою взаємодію з чат-ботом.

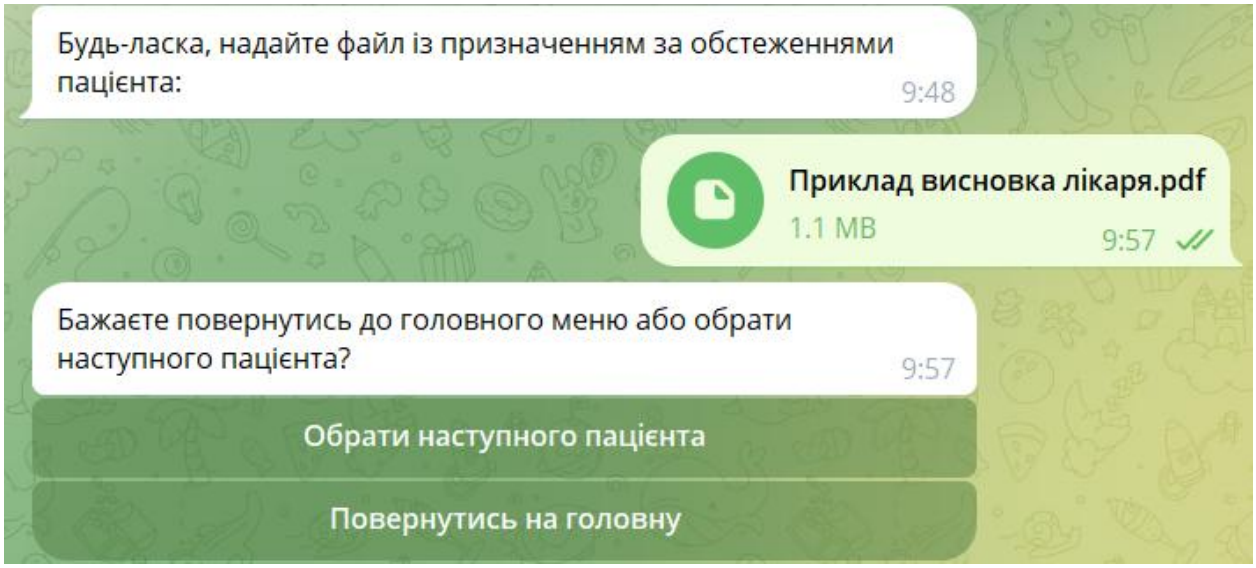

Рисунок 4.24 – Надсилання висновку лікаря з подальшою можливістю перейти до наступного пацієнта

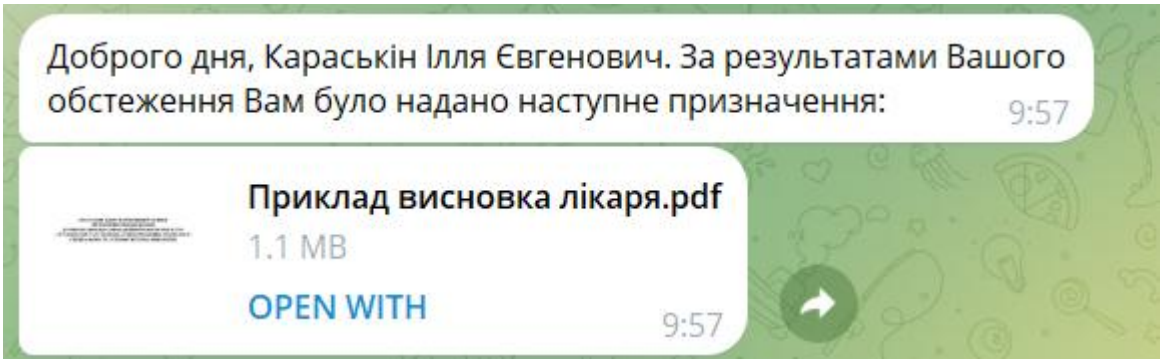

Рисунок 4.25 – Отримання висновку лікаря пацієнтом

4.4.2.2 Звернення за консультацією до спілки офтальмологів

У разі якщо фахівець сам бажає отримати консультацію в інших фахівців, що входять до спілки офтальмологів, він матиме також внести деякі дані, а саме причину запиту та файли з медичними даним, за якими він бажає отримати консультацію (рисунок 4.26).

Рисунок 4.26 – Створення звернення на проведення консультації зі спілкою офтальмологів

По завершенню формування звернення, фахівець отримує два повідомлення. Перше – у разі успішного формування запиту про консультацію, друге – в момент успішної розсилки чат-ботом цього запиту фахівцям спілки (рисунок 4.26). Приклад отриманого іншим фахівцем запиту можна побачити на рисунку 4.27.

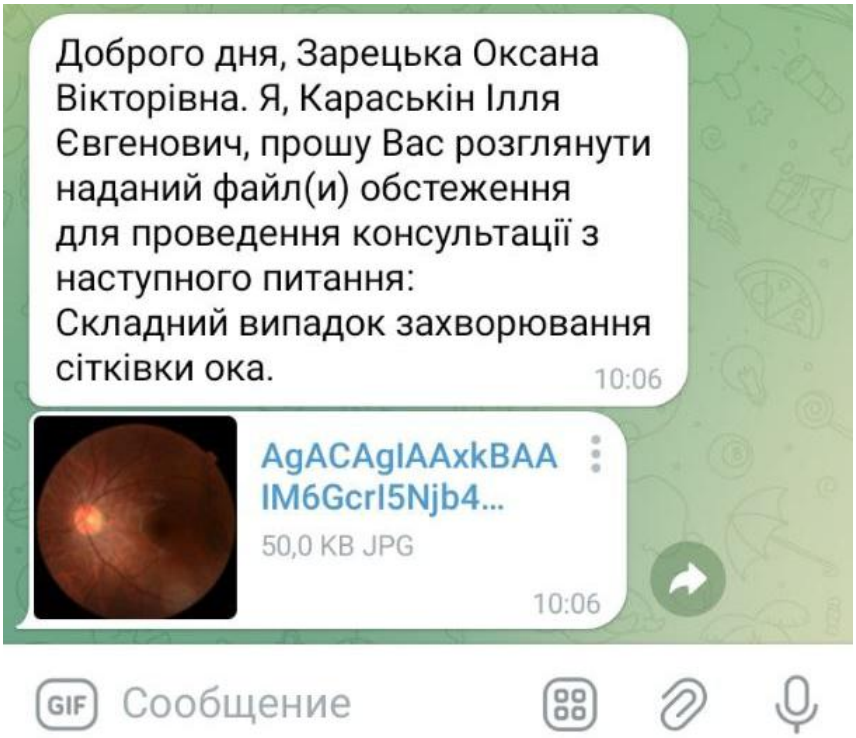

Рисунок 4.27 – Форма запиту консультації з розсилки чат-боту для членів спілки офтальмологів

4.4.2.3 Призначення онлайн консультації спілки офтальмологів

Останній варіант, який може обрати фахівець при взаємодії з чат-ботом – це призначити конференцію щодо питання або рядку питань для членів спілки. Доступ до цієї функції відбувається за наявності у конкретного члену спілки відповідної ролі в білому списку. Так, на прикладі рисунку 4.17, доступ до цієї функції матиме тільки фахівець Караськін Ілля Євгенович, оскільки він має роль «professor», а інші наявні на рисунку фахівці – роль «doctor».

При переході до цієї гілки, фахівцю, що має доступ, буде запропоновано призначити день та час проведення онлайн конференції, що відбудеться на базі одного з сервісів, таких як MS Teams або Zoom, а також надати власний коментар щодо тем, які будуть обговорюватись на цій конференції. Приклад створення призначення конференції та розсилки запрошення наведено на рисунках 4.28 та 4.29 відповідно.

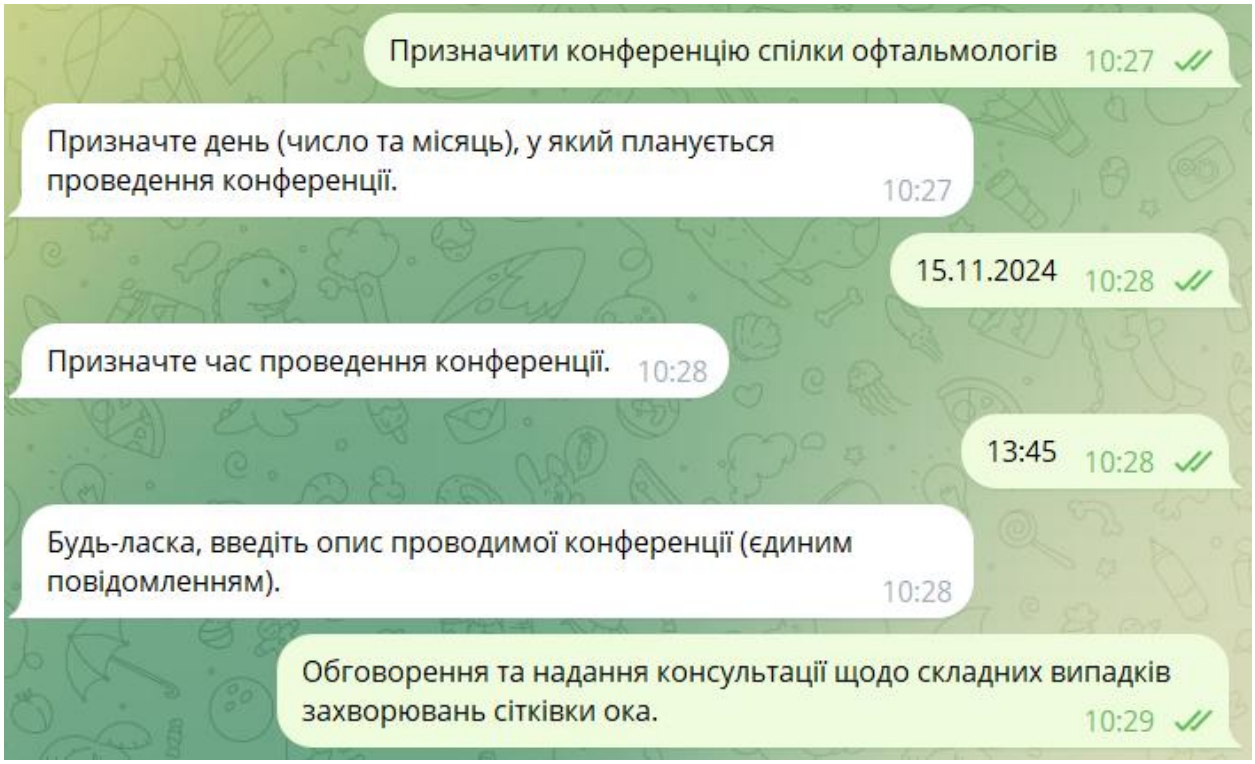

Рисунок 4.28 – Створення запрошення на конференцію спілки офтальмологів

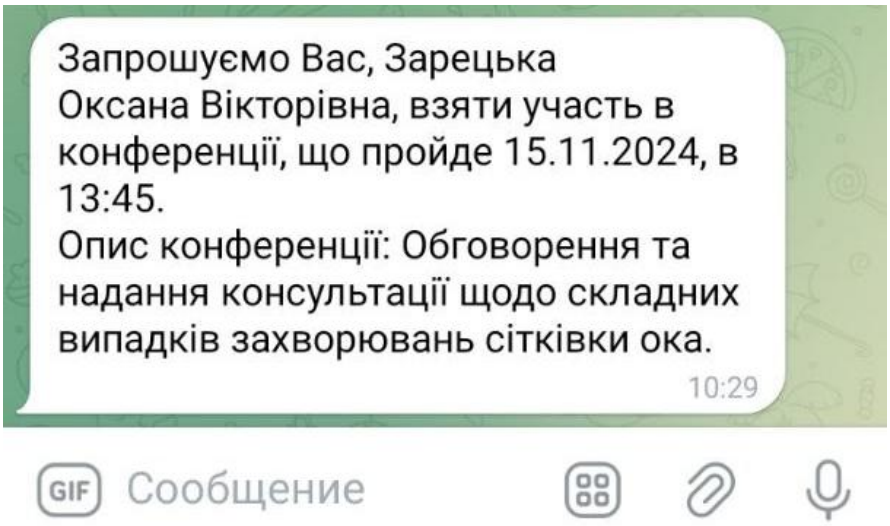

Рисунок 4.29 – Приклад розсилки запорошення чат-ботом

4.5 Висновки до розділу розробки програмного забезпечення

Таким чином було розроблено та обґрунтовано функціональні та технічні характеристики програмного забезпечення комплексу та

опрацьовано логічну структуру взаємодії з ним різних груп користувачів, згідно з якою було побудовано ряд структурних схем.

Спираючись на отримані та опрацьовані в ході дослідження програмного забезпечення дані, було написано програмний код чат-боту, що забезпечує взаємодію між користувачами та БД.

Всі функції програмного забезпечення було експериментально протестовано та встановлено їх працездатність для можливості проведення подальших експериментальних досліджень щодо впровадження комплексу онлайн-консультацій на об’єкті нашого дослідження.

5 ЕКСПЕРИМЕНТАЛЬНИЙ РОЗДІЛ

5.1 Постановка завдання експерименту

Завданням для нашого експерименту є дослідження навантаження на сервер підприємства від роботи комплексу онлайн-консультацій, а саме роботи чат-боту Telegram та пов’язаної з ним бази даних в залежності від виконуваних операцій та навантажень в моменти обробки, завантаження та відвантаження файлів та документів.

5.2 Методика проведення експериментальних досліджень

Для проведення експерименту, нами було застосовано спеціалізоване програмне забезпечення для отримання поточних даних про навантаження на БД та технічні компоненти системи, а саме:

1. MongoDB Compass – крім графічного інтерфейсу для взаємодію з БД, також має інтрефейс для перегляду базових метрік роботи БД у вигляді графіків та статистик, що стосується роботи самої БД, її колекцій та документів;
2. Mongostat – один з інструментів, що входять до набору MongoTools та призначений для виведення докладної інформації про работу БД та її вплив на систему у вигляді оновлюваної в режимі реального часу таблиці в консоль;
3. Mongo Exporter – ще одна утиліта для сбору інформації про функціонування MongoDB, що включає в себе набагато ширший перелік показників та можливість експорту цієї інформації в інші програми;
4. Prometheus – система моніторінгу та збору метрик, що використовується нами для отримання даних від MongoDB Exporter та моніторінгу навантаження на базу даних та інші технічні компоненти системи в режимі реального часу;

5. Grafana – завдяки можливості інтеграції з Prometheus та Mongo Exporter, використовується нами в рамках есперименту в якості інструменту для візуалізації даних метрик роботи БД та технічних компонентів у вигляді графіків і статистик, що спрощує аналіз ефективності роботи системи.
6. Крім того, для наближення до реалій та умов, в яких передбачається функціонування чат-боту, нами було використано файлові дані з реальними результатами обстеження (такими як знімки ока при різних захворюваннях), отримані з ресурсу Kaggle – онлайн-платформа для роботи з наборами даних, яка надає доступ до реальних даних для досліджень і навчання.

Для симуляції навантаження на чат-бот та базу даних, нами було використано декілька облікових записів телеграм, що одночасно вестимуть діалог із чат-ботом та будуть взаємодіяти з базою даних підприємства. Кількість облікових записів, використовуваних для тестування, було розраховано спираючись на статистичні дані підприємства щодо кількості звернень пацієнтів до медичних реєстраторів протягом робочого дня.

При симуляції навантаження, участниками експерименту буде здійснюватись надання файлів в тих форматах та кількості, які зазвичай надаються пацієнтами для консультацій з лікарями-офтальмологами. Взаємодія з ботом користувачами відбуватиметься одночасно для симуляції максимально можливого одномоментного навантаження на систему.

5.3 Вимоги до проведення експерименту

КП «ДОКОЛ» здійснює близько 50000 консультаційних прийомів щорічно. Виходячи з того, що кількість робочих днів на рік складає в середньому 250, на день здійснюється в середньому 200 прийомів. Спираючись на статистичні дані КП «ДОКОЛ», очікувана кількість пацієнтів, що можуть виявити бажання скористатися можливістю отримати онлайн

консультацию може з часом досягти 50 відсотків від загальної кількості. Тобто кількість активних користувачів комплексом онлайн консультацій може скласти близько 100 осіб на добу.

Крім того система має забезпечити можливість взаємного спілкування та співпраці фахівців зі спілки офтальмологів Дніпропетровщини, в яку входять 30 фахівців. Тобто, якщо враховувати максимально можливе потенційне навантаження на комплекс онлайн-консультацій, система має забезпечити можливість одночасного активного спілкування та розсилки даних для 130 користувачів.

Експериментальна група, яку було задіяно під час проведення експерименту, становить 10 осіб, що відповідає очікуваної кількості користувачів протягом 30 хвилин робочого часу.

Для проведення експерименту, всіх користувачів експериментальної групи було додано до білого списку фахівців спілки для можливості доступу до всіх функцій комплексу онлайн-консультацій.

Протягом експерименту, всім користувачам групи було надано розпорядження проводити активний діалог з наданням інформаційних даних та файлів необхідного виду та обсягу. По завершенню діалогу, користувач матиме одразу починати наступний, за для імітації неперервної роботи комплексу.

5.4 Отримані результати в ході експериментальних досліджень

В ході експерименту нами через консоль за допомогою утиліти mongostat безперервно отримувались ряд метричних даних про навантаження на БД та сервер за проміжок часу в пів години, під час якого відбувалась одночасна взаємодія десяти користувачів із чат-ботом та базою даних, що з ним пов’язана. Користувачами було забезпечено всі можливі варіанти навантаження на систему. На рисунках 5.1 – 5.3 наведено вибірку цих метрик на початку, середині та наприкінці тестування.

```
C:\Users\user>mongostat
insert query update delete getmore command dirty  used flushes vsize   res qrw arw net_in net_out conn               time
    *0    *0     *0     *0       0     1|0  0.0% 10.9%       0 5.03G 42.0M 0|0 0|0   162b   83.2k   28 Nov 10 16:15:35.986
    *0    *0     *0     *0       0     1|0  0.0% 10.9%       0 5.03G 42.0M 0|0 0|0   112b   83.9k   28 Nov 10 16:15:36.984
    *0    *0     *0     *0       0     2|0  0.0% 10.9%       0 5.03G 42.0M 0|0 0|0   246b   84.2k   28 Nov 10 16:15:37.982
    *0    *0     *0     *0       0     1|0  0.0% 10.9%       0 5.03G 42.0M 0|0 0|0   174b   83.0k   28 Nov 10 16:15:38.994
    *0    *0     *0     *0       0     1|0  0.0% 10.9%       0 5.03G 42.0M 0|0 0|0   113b   85.0k   28 Nov 10 16:15:39.979
    *0    *0     *0     *0       0     0|0  0.0% 10.9%       0 5.03G 42.0M 0|0 0|0   111b   83.5k   28 Nov 10 16:15:40.982
    *0    *0     *0     *0       0     0|0  0.0% 10.9%       0 5.03G 42.0M 0|0 0|0   111b   83.4k   28 Nov 10 16:15:41.986
    *0    *0     *0     *0       0     1|0  0.0% 10.9%       0 5.03G 42.0M 0|0 0|0   112b   84.0k   28 Nov 10 16:15:42.983
    *0    *0     *0     *0       0     3|0  0.0% 10.9%       0 5.03G 42.0M 0|0 0|0   451b   83.8k   28 Nov 10 16:15:43.992
    *0    *0     *0     *0       0     5|0  0.0% 10.9%       0 5.03G 42.0M 0|0 0|0   517b   86.6k   28 Nov 10 16:15:44.973
insert query update delete getmore command dirty  used flushes vsize   res qrw arw net_in net_out conn               time
    *0    *0     *0     *0       0     3|0  0.0% 10.9%       0 5.03G 42.0M 0|0 0|0   528b    179k   28 Nov 10 16:15:45.982
    *0    *0     *0     *0       0     2|0  0.0% 10.9%       0 5.03G 42.0M 0|0 0|0   470b    176k   28 Nov 10 16:15:47.006
    *0     1     *0     *0       0     4|0  0.0% 10.9%       0 5.03G 42.0M 0|0 0|0   629b    185k   28 Nov 10 16:15:47.982
    *0     2     *0     *0       0    24|0  0.0% 10.9%       0 5.03G 42.0M 0|0 0|0  3.56k    172k   29 Nov 10 16:15:48.993
    *0     1     *0     *0       0     3|0  0.0% 10.9%       0 5.03G 42.0M 0|0 0|0   486b    182k   28 Nov 10 16:15:49.982
    *0    *0     *0     *0       0     2|0  0.0% 10.9%       0 5.03G 42.0M 0|0 0|0   480b    180k   28 Nov 10 16:15:50.982
    *0     1     *0     *0       0     3|0  0.0% 10.9%       0 5.03G 42.0M 0|0 0|0   481b    180k   28 Nov 10 16:15:51.981
    *0    *0     *0     *0       0     2|0  0.0% 10.9%       0 5.03G 42.0M 0|0 0|0   479b    180k   28 Nov 10 16:15:52.985
    *0    *0     *0     *0       0     5|0  0.0% 10.9%       0 5.03G 42.0M 0|0 0|0   819b    180k   28 Nov 10 16:15:53.992
    *0     1     *0     *0       0     6|0  0.0% 10.9%       0 5.03G 42.0M 0|0 0|0   821b    183k   28 Nov 10 16:15:54.984
insert query update delete getmore command dirty  used flushes vsize   res qrw arw net_in net_out conn               time
    *0    *0     *0     *0       0     4|0  0.0% 10.9%       0 5.03G 42.0M 0|0 0|0   584b    178k   28 Nov 10 16:15:56.001
    *0     2      5     20       0     5|0  0.0% 10.9%       0 5.03G 42.0M 0|0 0|0   487b    183k   28 Nov 10 16:15:56.988
    *0    *0     *0     *0       0     3|0  0.0% 10.9%       0 5.03G 42.0M 0|0 0|0   611b    180k   28 Nov 10 16:15:57.993
    *0     1     *0     *0       0     3|0  0.0% 10.9%       0 5.03G 42.0M 0|0 0|0   481b    181k   28 Nov 10 16:15:58.991
    *0     1     *0     *0       0     4|0  0.0% 10.9%       0 5.03G 42.0M 0|0 0|0   551b    182k   28 Nov 10 16:15:59.982
    *0    *0     *0     *0       0     2|0  0.0% 10.9%       0 5.03G 42.0M 0|0 0|0   472b    177k   28 Nov 10 16:16:01.000
    *0     1     *0     *0       0     3|0  0.0% 10.9%       0 5.03G 42.0M 0|0 0|0   486b    182k   28 Nov 10 16:16:01.988
    *0     1     *0     *0       0     3|0  0.0% 10.9%       0 5.03G 42.0M 0|0 0|0   482b    181k   28 Nov 10 16:16:02.984
    *0     3     *0     *0       0    28|0  0.0% 10.9%       0 5.03G 42.0M 0|0 0|0  4.26k    272k   29 Nov 10 16:16:03.985
    *0     1     *0     *0       0     7|0  0.0% 10.9%       0 5.03G 42.0M 0|0 0|0   878b    182k   28 Nov 10 16:16:04.983
```

Рисунок 5.1 – Початок сбору метрик

```
insert query update delete getmore command dirty  used flushes vsize   res qrw arw net_in net_out conn               time
    *0    *0     *0     *0       0     2|0  0.2% 11.0%       0 5.04G 55.0M 0|0 0|0   391b   83.9k   33 Nov 10 16:28:15.990
     1     6     *0     *0       0     1|0  0.2% 11.0%       0 5.04G 56.0M 0|0 0|0  56.2k    139k   33 Nov 10 16:28:17.007
     3    24     *0     *0       0     1|0  0.2% 11.0%       0 5.04G 56.0M 0|0 0|0   316k   1.59m   33 Nov 10 16:28:17.979
    *0     2     *0     *0       0    24|0  0.2% 11.0%       0 5.04G 56.0M 0|0 0|0  3.61k    171k   34 Nov 10 16:28:18.999
    *0    *0     *0     *0       0     1|0  0.2% 11.0%       0 5.04G 56.0M 0|0 0|0   114b   85.6k   33 Nov 10 16:28:19.978
    *0    63     *0     *0       0     2|0  0.2% 11.0%       0 5.04G 56.0M 0|0 0|0  11.2k   3.55m   33 Nov 10 16:28:20.977
     1     4     *0     *0       0     0|0  0.2% 11.0%       0 5.04G 56.0M 0|0 0|0  98.3k    181k   33 Nov 10 16:28:21.997
    *0    *0     *0     *0       0     0|0  0.2% 11.0%       0 5.04G 56.0M 0|0 0|0   107b   80.6k   33 Nov 10 16:28:23.035
    *0    *0     *0     *0       0     4|0  0.2% 11.0%       0 5.04G 56.0M 0|0 0|0   480b   89.3k   33 Nov 10 16:28:23.983
    *0    21     *0     *0       0     2|0  0.2% 11.0%       0 5.04G 56.0M 0|0 0|0  3.86k   1.24m   33 Nov 10 16:28:24.978
insert query update delete getmore command dirty  used flushes vsize   res qrw arw net_in net_out conn               time
    *0    *0     *0     *0       0     3|0  0.2% 11.0%       0 5.04G 56.0M 0|0 0|0   451b   83.8k   33 Nov 10 16:28:25.989
    *0    *0     *0     *0       0     1|0  0.2% 11.0%       0 5.04G 56.0M 0|0 0|0   163b   84.0k   33 Nov 10 16:28:26.989
     2     5     *0     *0       0     2|0  0.2% 11.0%       0 5.04G 57.0M 0|0 0|0   214k    299k   33 Nov 10 16:28:27.984
    *0    *0     *0     *0       0     0|0  0.2% 11.0%       0 5.04G 57.0M 0|0 0|0   111b   83.4k   33 Nov 10 16:28:28.988
    *0    *0     *0     *0       0     0|0  0.2% 11.0%       0 5.04G 57.0M 0|0 0|0   110b   82.3k   33 Nov 10 16:28:30.005
    *0    *0     *0     *0       0     2|0  0.2% 11.0%       0 5.04G 57.0M 0|0 0|0   181b   86.0k   33 Nov 10 16:28:30.982
    *0    *0     *0     *0       0     0|0  0.2% 11.0%       0 5.04G 57.0M 0|0 0|0   271b   83.0k   33 Nov 10 16:28:31.993
    *0     1     *0     *0       0     3|0  0.2% 11.0%       0 5.04G 57.0M 0|0 0|0   483b    181k   33 Nov 10 16:28:32.989
    *0     3     *0     *0       0    27|0  0.2% 11.0%       0 5.04G 57.0M 0|0 0|0  3.95k    177k   34 Nov 10 16:28:33.975
    *0    *0     *0     *0       0     4|0  0.2% 11.0%       0 5.04G 57.0M 0|0 0|0   589b    179k   33 Nov 10 16:28:34.985
insert query update delete getmore command dirty  used flushes vsize   res qrw arw net_in net_out conn               time
    *0     1     *0     *0       0     5|0  0.2% 11.0%       0 5.04G 57.0M 0|0 0|0   764b    181k   33 Nov 10 16:28:35.983
    *0    *0     *0     *0       0     1|0  0.2% 11.0%       0 5.04G 57.0M 0|0 0|0   162b   83.1k   33 Nov 10 16:28:36.994
    *0     1     *0     *0       0     3|0  0.2% 11.0%       0 5.04G 57.0M 0|0 0|0   487b    183k   33 Nov 10 16:28:37.980
    *0     1     *0     *0       0     3|0  0.2% 11.0%       0 5.04G 57.0M 0|0 0|0   756b    176k   33 Nov 10 16:28:39.007
    *0    43     *0     *0       0     3|0  0.2% 11.0%       0 5.04G 57.0M 0|0 0|0  8.00k   2.54m   33 Nov 10 16:28:39.985
    *0    *0     *0     *0       0     3|0  0.2% 11.0%       0 5.04G 57.0M 0|0 0|0   525b    178k   33 Nov 10 16:28:40.999
    *0     1     *0     *0       0     3|0  0.2% 11.0%       0 5.04G 57.0M 0|0 0|0   488b    183k   33 Nov 10 16:28:41.984
    *0     1     *0     *0       0     3|0  0.2% 11.0%       0 5.04G 57.0M 0|0 0|0   481b    180k   33 Nov 10 16:28:42.983
     6    16     *0     *0       0     6|0  0.2% 11.0%       0 5.04G 57.0M 0|0 0|0   230k    412k   33 Nov 10 16:28:43.983
    *0    *0     *0     *0       0     4|0  0.2% 11.0%       0 5.04G 57.0M 0|0 0|0   592b    180k   33 Nov 10 16:28:44.987
```

Рисунок 5.2 – Метрики після п’ятнадцяти хвилин роботи

```
insert query update delete getmore command dirty  used flushes vsize   res qrw arw net_in net_out conn               time
    *0     1     *0     *0       0     6|0  0.2% 11.8%       0 5.07G 85.0M 0|0 0|0   818b    182k   33 Nov 10 16:38:55.978
    *0     4     *0     *0       0     2|0  0.2% 11.8%       0 5.07G 85.0M 0|0 0|0  1.16k    183k   33 Nov 10 16:38:56.987
    *0     1     *0     *0       0     5|0  0.2% 11.8%       0 5.07G 85.0M 0|0 0|0   601b    182k   33 Nov 10 16:38:57.982
    *0     1     *0     *0       0     3|0  0.2% 11.8%       0 5.07G 85.0M 0|0 0|0   481b    180k   33 Nov 10 16:38:58.982
    *0    *0     *0     *0       0     2|0  0.2% 11.8%       0 5.07G 85.0M 0|0 0|0   480b    180k   33 Nov 10 16:38:59.982
    *0     1     *0     *0       0     3|0  0.2% 11.8%       0 5.07G 85.0M 0|0 0|0   481b    180k   33 Nov 10 16:39:00.980
     2     4     *0     *0       0     4|0  0.2% 11.8%       0 5.07G 85.0M 0|0 0|0  51.1k    181k   33 Nov 10 16:39:01.978
    *0    *0     *0     *0       0     2|0  0.2% 11.8%       0 5.07G 85.0M 0|0 0|0   476b    178k   33 Nov 10 16:39:02.987
    *0     4     *0     *0       0    27|0  0.2% 11.8%       0 5.07G 85.0M 0|0 0|0  4.26k    271k   34 Nov 10 16:39:03.990
     2     4     *0     *0       0     5|0  0.2% 11.8%       0 5.07G 85.0M 0|0 0|0  29.4k    183k   33 Nov 10 16:39:04.980
insert query update delete getmore command dirty  used flushes vsize   res qrw arw net_in net_out conn               time
    *0     1     *0     *0       0     6|0  0.2% 11.8%       0 5.07G 85.0M 0|0 0|0   817b    182k   33 Nov 10 16:39:05.977
     1     3     *0     *0       0     2|0  0.2% 11.8%       0 5.07G 85.0M 0|0 0|0  80.8k    177k   33 Nov 10 16:39:06.998
    *0     1     *0     *0       0     4|0  0.2% 11.8%       0 5.07G 85.0M 0|0 0|0   538b    182k   33 Nov 10 16:39:07.987
    *0     1     *0     *0       0     4|0  0.2% 11.8%       0 5.07G 85.0M 0|0 0|0   534b    181k   33 Nov 10 16:39:08.984
    *0    *0     *0     *0       0     2|0  0.2% 11.8%       0 5.07G 85.0M 0|0 0|0   480b    180k   33 Nov 10 16:39:09.985
    *0    *0     *0     *0       0     2|0  0.2% 11.8%       0 5.07G 85.0M 0|0 0|0   480b    180k   33 Nov 10 16:39:10.986
     1     3     *0     *0       0     3|0  0.2% 11.8%       0 5.07G 85.0M 0|0 0|0  83.9k    181k   33 Nov 10 16:39:11.987
    *0    *0     *0     *0       0     2|0  0.2% 11.8%       0 5.07G 85.0M 0|0 0|0   480b    180k   33 Nov 10 16:39:12.988
    *0    *0     *0     *0       0     4|0  0.2% 11.8%       0 5.07G 85.0M 0|0 0|0   676b    181k   33 Nov 10 16:39:13.989
     1     3     *0     *0       0     4|0  0.3% 11.8%       0 5.07G 85.0M 0|0 0|0  86.0k    181k   33 Nov 10 16:39:14.991
insert query update delete getmore command dirty  used flushes vsize   res qrw arw net_in net_out conn               time
     4     7     *0     *0       0     6|0  0.3% 11.8%       0 5.07G 85.0M 0|0 0|0   174k    185k   33 Nov 10 16:39:15.977
     5     9     *0     *0       0     2|0  0.3% 11.8%       0 5.07G 85.0M 0|0 0|0   255k    181k   33 Nov 10 16:39:16.983
     6    10     *0     *0       0     4|0  0.0% 11.8%       0 5.07G 86.0M 0|0 0|0   258k    182k   33 Nov 10 16:39:17.980
     7    15     *0     *0       0    26|0  0.0% 11.8%       0 5.07G 86.0M 0|0 0|0   345k    273k   34 Nov 10 16:39:18.983
     5     9     *0     *0       0     2|0  0.0% 11.8%       0 5.07G 87.0M 0|0 0|0   256k    181k   33 Nov 10 16:39:19.988
     6    10     *0     *0       0     3|0  0.1% 11.8%       0 5.07G 87.0M 0|0 0|0   257k    182k   33 Nov 10 16:39:20.988
     2     4     *0     *0       0     4|0  0.1% 11.8%       0 5.07G 87.0M 0|0 0|0  87.2k    183k   33 Nov 10 16:39:21.975
    *0    *0     *0     *0       0     2|0  0.1% 11.8%       0 5.07G 87.0M 0|0 0|0   474b    178k   33 Nov 10 16:39:22.989
    *0     1     *0     *0       0     5|0  0.1% 11.8%       0 5.07G 87.0M 0|0 0|0   677b    181k   33 Nov 10 16:39:23.988
    *0     1     *0     *0       0     5|0  0.1% 11.8%       0 5.07G 87.0M 0|0 0|0   691b    181k   33 Nov 10 16:39:24.988
insert query update delete getmore command dirty  used flushes vsize   res qrw arw net_in net_out conn               time
    *0     1     *0     *0       0     6|0  0.1% 11.8%       0 5.07G 87.0M 0|0 0|0   815b    181k   33 Nov 10 16:39:25.987
    *0     1     *0     *0       0     3|0  0.1% 11.8%       0 5.07G 87.0M 0|0 0|0   486b    182k   33 Nov 10 16:39:26.976
     4     3     *0     *0       0     3|0  0.1% 11.8%       0 5.07G 89.0M 0|0 0|0   521k    178k   33 Nov 10 16:39:27.993
    *0     2     *0     *0       0     4|0  0.1% 11.8%       0 5.07G 89.0M 0|0 0|0   793b    183k   33 Nov 10 16:39:28.979
    *0    *0     *0     *0       0     2|0  0.1% 11.8%       0 5.07G 89.0M 0|0 0|0   475b    178k   33 Nov 10 16:39:29.991
    *0     1     *0     *0       0     3|0  0.1% 11.8%       0 5.07G 89.0M 0|0 0|0   487b    183k   33 Nov 10 16:39:30.978
    *0    *0     *0     *0       0     3|0  0.1% 11.8%       0 5.07G 89.0M 0|0 0|0   604b    178k   33 Nov 10 16:39:31.995
    *0     2     *0     *0       0     3|0  0.1% 11.8%       0 5.07G 89.0M 0|0 0|0   699b    181k   33 Nov 10 16:39:32.991
    *0     1     *0     *0       0     8|0  0.1% 11.8%       0 5.07G 89.0M 0|0 0|0  1.26k    130k   33 Nov 10 16:39:38.062
    *0     5     *0     *0       0     4|0  0.1% 11.8%       0 5.07G 89.0M 0|0 0|0  1.33k    771k   33 Nov 10 16:39:38.984
```

Рисунок 5.3 – Метрики наприкінці тестування

Крім того, нами було отримано проміжні графіки навантаження на саму БД (рисунок 5.4 – 5.6) та мережу від роботи БД (рисунок 5.7 – 5.9) в різні моменти часу протягом проведення експерименту.

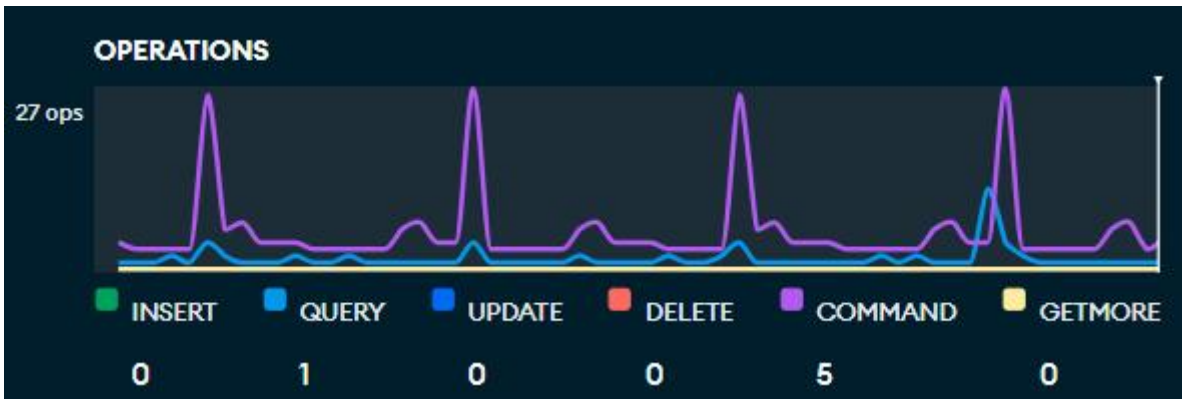

Рисунок 5.4 – Навантаження на БД на початку експерименту

Рисунок 5.5 – Навантаження на БД в середині експерименту

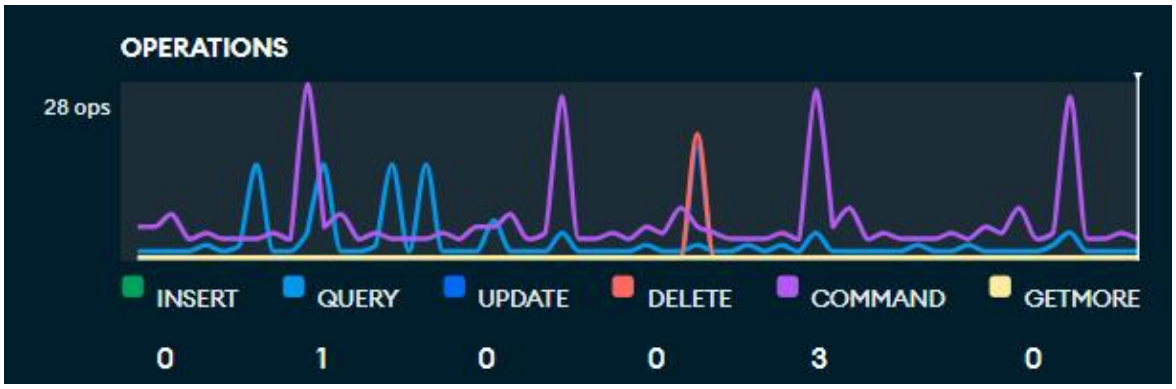

Рисунок 5.6 – Навантаження на БД незадовго до завершення експерименту

Рисунок 5.7 – Навантаження на мережу на початку експерименту

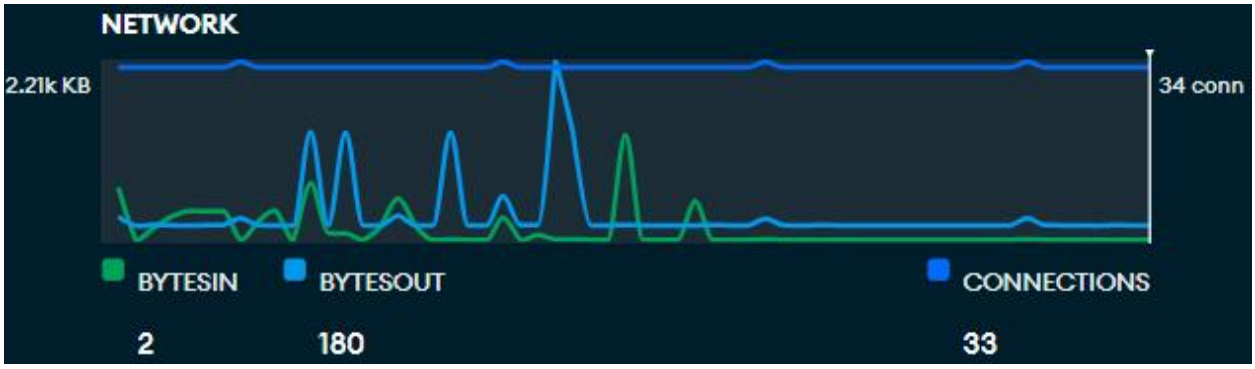

Рисунок 5.8 – Навантаження на мережу в середині експерименту

Рисунок 5.9 – Навантаження на мережу перед завершенням експерименту

Спираючись на ці метричні дані, а також на додаткові дані, отримані нами за допомогою роботи інструменту Mongo Exporter, нами було побудовано загальні графіки зміни показників навантаження за різними критеріями, що наведено на рисунках 5.10 – 5.15. Графіки було побудовано та отримано за допомогою додаткового підключеного до Mongo Exporter програмного забезпечення, до якого увійшли Prometheus та Grafana.

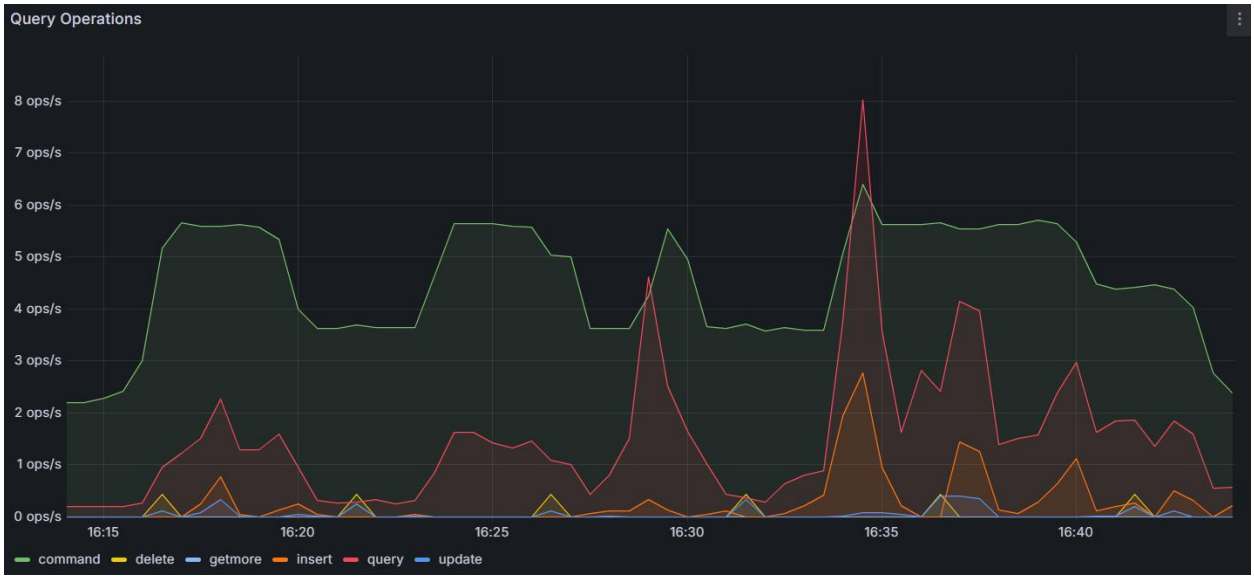

Рисунок 5.10 – Загальний графік кількості операцій запитів до БД

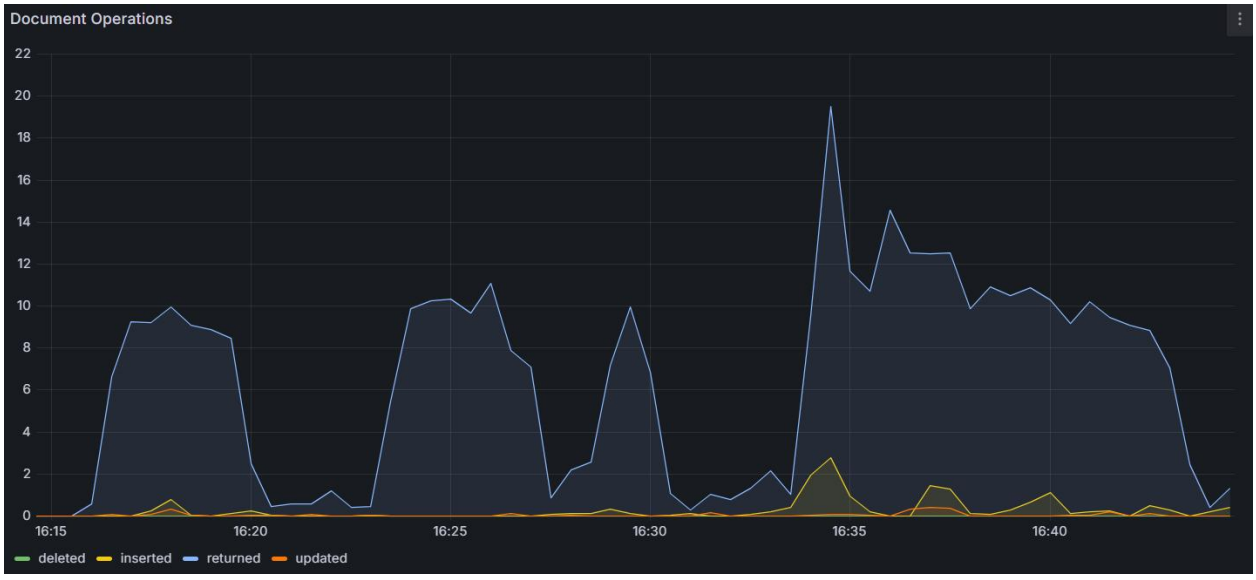

Рисунок 5.11 – Загальний графік кількості операцій з документами у БД

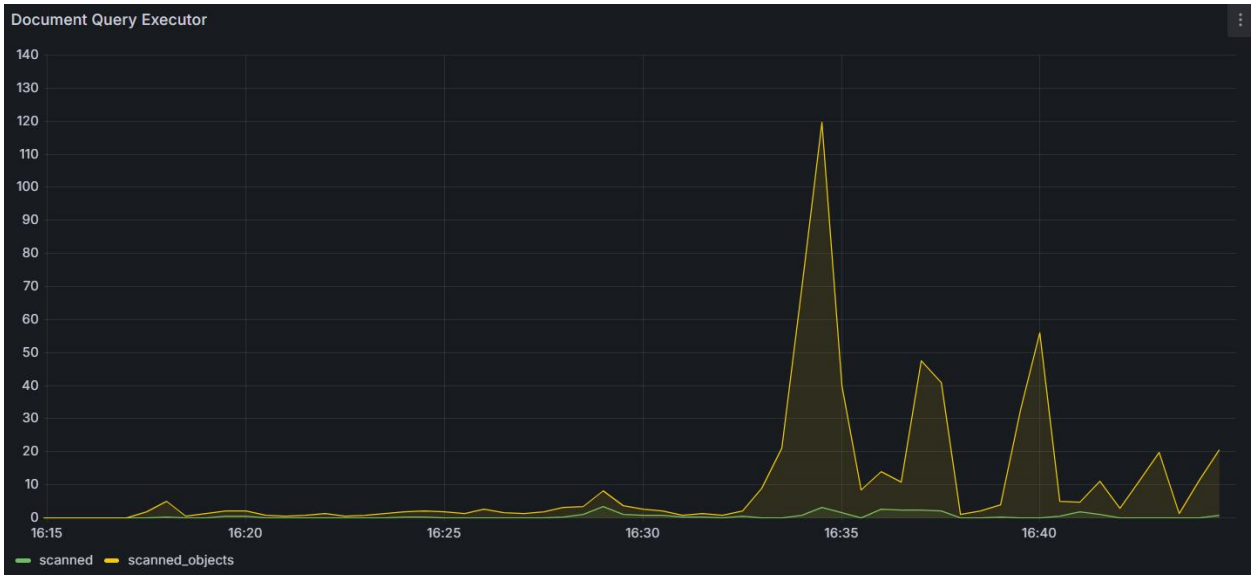

Рисунок 5.12 – Загальний графік операцій пошуку по документам у БД

Рисунок 5.13 – Графік та середнє значення кількості з’єднань з БД протягом експерименту

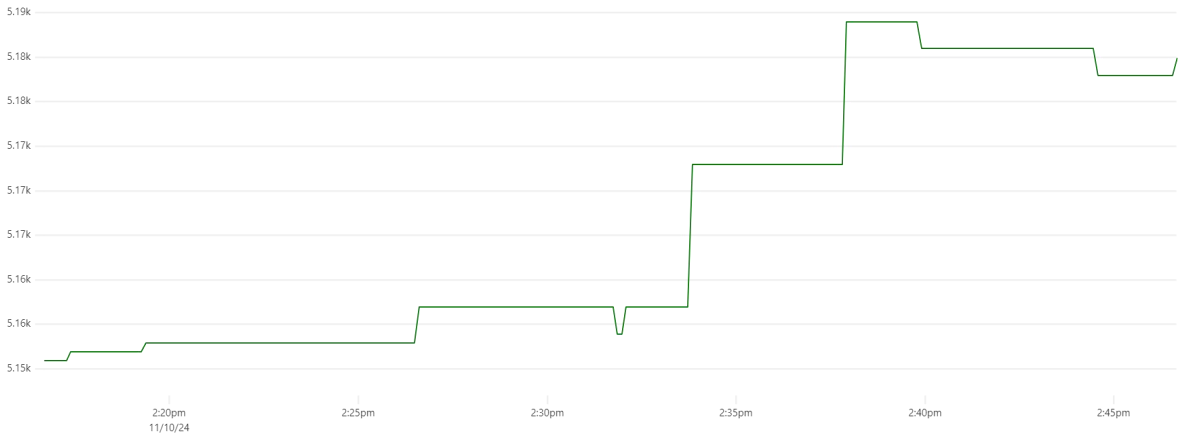

Рисунок 5.14 – Графік використання виділеної віртуальної пам’яті

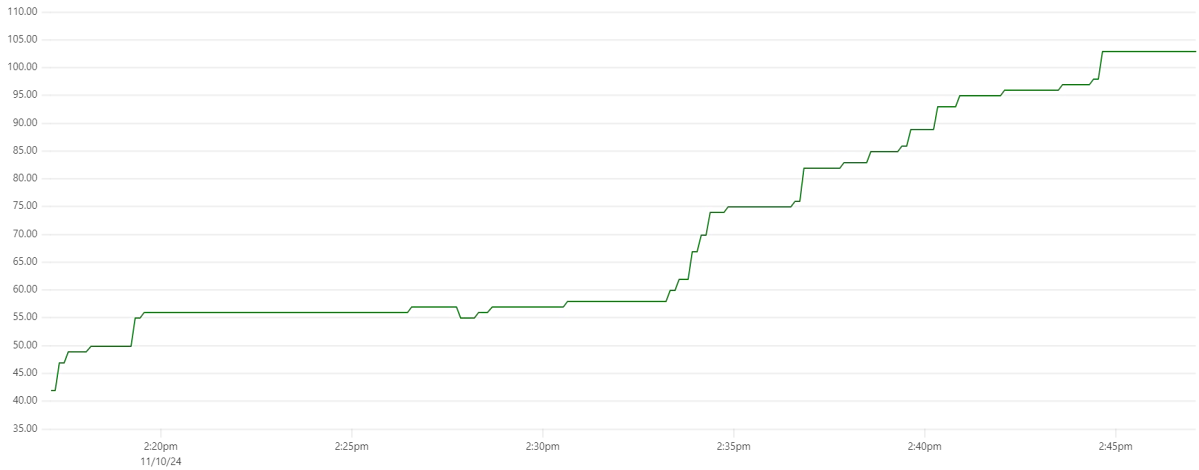

Рисунок 5.15 – Графік використання резидентної пам’яті базою даних

5.5 Аналіз та оцінка отриманих результатів

В процесі аналізу метрик, зібраних протягом проведення експерименту, та враховуючи відносно невеликий розмір групи тестування, нами було зроблено ряд проміжних висновків щодо різних показників роботи БД та серверу.

Так, показники dirty та used, що відповідають за відсоток брудних сторінок у кеші, які ще не вказані на диску, та використання кешу MongoDB, тримаються на низькому рівні (близько 1% та 11% відповідно) протягом всього експерименту. Це свідчить про те, що система оперативно скидає дані на диск, а також про достатню кількість ресурсів системи для виконання запитів з мінімальним залученням кешу.

Проаналізувавши дані графіків кількості операцій запитів до БД на секунду (рисунок 5.10) та зіставивши їх з графіками кількості операцій з самими документами у БД (рисунок 5.11) та кількості операцій пошуку по документам та індексам у БД (рисунок 5.12), можемо прослідкувати прямо пропорційну залежність між показниками кількості виконуваних запитів до БД, поверненню з неї документів та скануванням по документам у БД.

Аналізуючи отримані нами значення метрик інтернет трафіку, нами було встановлено максимальні значення 345 kb/s (рисунок 5.3) та 2.4 mb/s (рисунок 5.2) для вхідного та вихідного трафіку відповідно. Це свідчить про відносно невелике навантаження на мережу під час проведення експерименту, піки якого випадають на етапи розсилки великої кількості даних та файлів між фахівцями спілки.

Оскільки в процесі експерименту всі 10 користувачів вважались членами спілки, розсилка також відбувалась серед 10-и користувачів. Очікувана кількість членів спілки становить приблизно 30 фахівців. Виходячи з цього можемо припустити, що реальне пікове значення вихідного трафіку під час розсилки буде в три рази більше та становитиме ~7.2 mb/s.

Показники графіку резидентної пам’яті (рисунок 5.15), що використовує БД для своєї роботи, дещо відрізняються від показників таблиці mongostat (рисунок 5.1 – 5.3), та становлять ~40 Mb на початку експерименту та ~100 Mb наприкінці, демонструючи тим самим стрибкоподібне зростання протягом всього часу роботи. Стрибки використання резидентної пам’яті також прийшлись на моменти здійснення розсилок. Крім того, таке зростання характерне для кешування даних, що часто запитуються.

Показники використання віртуальної пам’яті (рисунок 5.14) також продемонстрували стрибкоподібне зростання, стабілізувавшись наприкінці експерименту. Зміна значень показників протягом експерименту склала ~5,15 Gb на початку експерименту та ~5,19 Gb – наприкінці. Тобто зростання під

час активної роботи БД складало не більше ~0,05 Gb. Таке невелике зростання віртуальної пам'яті свідчить про те, що MongoDB виділяє собі додаткові ресурси для нових операцій та при збільшенні кількості запитів в процесі експерименту, але гострої потреби в виділені додаткової пам’яті перед системою не постає.

5.6 Висновки до експериментального розділу

Спираючись на характеристики серверного обладнання, що представлено на підприємстві та описано в підрозділі 3.4, а саме сервер з встановленими процесорами Intel Xeon E5-2650 v2, 16 Gb оперативної пам’яті та можливістю підключення до інтернету через гігабітні мережеві інтерфейси, можемо зробити наступні висновки щодо аналітичних даних, отриманих нами за результатами проведеного експерименту:

1. Показники роботи бази даних, зокрема низький рівень використання кешу та ефективна взаємодія з хранилищем, свідчать про високу швидкість обробки запитів та мінімальне навантаження на систему.

2. Аналіз навантаження на ОП продемонстрував стабільний невисокий відсоток використання ресурсу сервера. І хоча, згідно прогнозу до пікових значень мережевого трафіку, в окремі моменти часу він може підійматись до достатньо високих значень – майже ~60 Mbit/s (чи 7,2 Mb/s), – це також не має завдати критичного навантаження на роботу серверу та мережі підприємства в цілому в силу короткої тривалості цих піків.

Таким чином, з огляду на технічні характеристики серверу та результати експерименту, можемо зробити висновок, що підприємство технічно спроможне розгорнути та підтримувати стабільну роботу комплексу онлайн-консультацій на базі свого обладнання, зокрема серверу, без завдання шкоди іншим технічним процесам, що відбуваються в рамках комп’ютерної системи організації.

ВИСНОВКИ

Кваліфікаційна робота є завершеною науковою роботою, в якій вирішена науково-практична задача створення та обґрунтування комплексу онлайн-консультацій в медичній галузі, шляхом використання існуючих програмних рішень та методів з розробкою на їх основі додаткового програмного забезпечення для реалізації проекту, а також проведений комплексний аналіз можливості реалізації та технічної доцільності даного проекту з урахуванням заданих параметрів існуючої комп'ютерної системи підприємства.

Основні висновки і результати роботи полягають у наступному:

1. Проведено дослідження основних тенденцій розвитку цифрової медицини, напрямків її використання та наголошено на необхідності створення відповідних систем у галузі охорони здоров'я нашої країни, а зокрема на підприємствах комунального типу.

2. Відповідно до цього, було зроблено та проаналізовано вибірку наявних систем для здійснення телекомунікації на спроможність до реалізації даного проекту.

3. Проаналізовано та обрано ряд оптимальних програмних рішень за для реалізації проекту комплексу онлайн-консультацій та проведення подальшого дослідження щодо показників його функціонування та технічної доцільності впровадження на підприємстві галузі охорони здоров'я.

4. Проведено аналіз та обґрунтування мережевого та апаратного забезпечення комп'ютерної системи підприємства, що є об'єктом цього дослідження, з метою встановлення технічних показників цієї системи та її обладнання.

5. Розроблено модель ефективного фукціонування комплексу онлайн-консультацій в рамках системи підприємства з урахуванням потреб всіх груп користувачів та реалізацією оптимального процесу здійснення комунікації.

6. На базі опрацьованої моделі функціонування комплексу, створено відповідне програмне забезпечення, що виступатиме в якості ланки, що пов’язує користувачів різних груп та рівнів між собою та сервером організації. Програмне забезпечення має циклічний формат роботи та передбачає можливість авторизації та ідентифікації користувачів, організації вхідних та вихідних даних, створення та роботу з документами у базі даних та взаємодію з тимчасовими файлами у директорії серверу.

7. Здійснено налагодження та вдосконалення програмного забезпечення. Доведено працездатність комплексу онлайн-консультацій та програмного забезпечення, що входить до його складу, шляхом проведення експерименту щодо поведінки чат-боту та бази даних під час всіх типів взаємодії з користувачами та зібрано і представлено демонстраційні матеріали.

8. Створено експериментальну групу користувачів та за допомогою додаткового програмного забезпечення, використаного для зняття метрик про роботу бази даних та серверу, проведено експеримент щодо симуляції навантаження, яке здатен завдавати комплекс онлайн-консультацій під час своєї активної роботи на базу даних, сервер на мережу об’єкту дослідження та, на основі зібраних в процесі експерименту метрик, побудовано ряд графіків, що за низкою критеріїв та показників характеризують ці навантаження за встановлений експериментом проміжок часу.

9. Спираючись на отримані в ході експерименту метричні та графічні результати, проведено аналіз та обґрунтування спроможності комп’ютерної системи об’єкту дослідження забезпечити підтримку безперебійної роботи комплексу онлайн-консультацій.

10. Таким чином була аналітично та експериментально підтверджена можливість впровадження комплексу онлайн-консультацій на базі комунальних підприємств галузі охорони здоров’я, зокрема об’єкту дослідження.

ПЕРЕЛІК ПОСИЛАНЬ

1. ДСТУ 3008-2015. Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання. –К.: Держстандарт, 2015. – 37 с.

2. Закон України «Про внесення змін до деяких законодавчих актів України щодо функціонування телемедицини» від 09.08.2023, документ 3301-IX (чинний) [Електронний ресурс] – Режим доступу: https://zakon.rada.gov.ua/laws/show/3301-20#n6

3. Національна служба здоров'я України. Що таке Телемедицина? [Електронний ресурс] – Режим доступу: https://ehealth.gov.ua/2021/11/15/shho-take-telemedytsyna/

4. Стратегії розвитку кіберофтальмології Ю. Г. Даник, д-р тех. наук, професор; О. В. Зборовська, д-р мед. наук; Н.В. Пасєчнікова, д-р мед. наук, член-кор. НАМН України, професор [Електронний ресурс] – Режим доступу: https://www.ozhurnal.com/sites/default/files/2020-5-ru12.pdf

5. Сутність і основи використання API [Електронний ресурс] – Режим доступу:https://learn.ztu.edu.ua/pluginfile.php/320400/mod_resource/content/1/%D0%9B%D0%B5%D0%BA%D1%86%D1%96%D1%8F%203.%20API.pptx

6. Цвіркун Л.І. Атестація здобувачів вищої освіти. Методичні рекомендації до виконання кваліфікаційної роботи магістра здобувачами галузі знань 12 Інформаційні технології спеціальності 123 Комп'ютерна інженерія / 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

7. Яковенко, А. В. Основи програмування. Python. Частина 1 : підручник для студ. спеціальності 122 "Комп'ютерні науки", спеціалізації "Інформаційні технології в біології та медицині" / А. В. Яковенко. – Київ : КПІ ім. Ігоря Сікорського, 2018. – 195 с.

8. Aiogram documentation [Електронний ресурс] – Режим доступу: https://docs.aiogram.dev/

9. Grafana documentation [Електронний ресурс] – Режим доступу: https://grafana.com/docs/grafana/latest/

10. GridFS for Self-Managed Deployment [Електронний ресурс] – Режим доступу: https://www.mongodb.com/docs/manual/core/gridfs/

11. Kaggle [Електронний ресурс] – Режим доступу: https://www.kaggle.com

12. Precedence Research. Digital Health Market Size, Share, and Trends 2024 to 2034 [Електронний ресурс] – Режим доступу: https://www.precedenceresearch.com/digital-health-market

13. Prometheus documentation [Електронний ресурс] – Режим доступу: https://prometheus.io/docs/prometheus/latest/getting_started/

14. Telegram APIs [Електронний ресурс] – Режим доступу: https://core.telegram.org/api#bot-api

15. The MongoDB Database Tools Documentation [Електронний ресурс] – Режим доступу: https://www.mongodb.com/docs/database-tools/

16. The MongoDB Documentation [Електронний ресурс] – Режим доступу: https://www.mongodb.com/docs/

17. Ultimate Guide to Server Load Testing and Monitoring - DNSstuff [Електронний ресурс] – Режим доступу: https://www-dnsstuff-com.translate.goog/server-load?_x_tr_sl=en&_x_tr_tl=ru&_x_tr_hl=ru&_x_tr_pto=rq

ДОДАТОК А

Текст програми чат-боту

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

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ЧАТ-БОТУ КОМПЛЕКСУ ОНЛАЙН-КОНСУЛЬТАЦІЙ КП «ДОКОЛ»

Текст програми

804.02070743.24007-01 12 01

Листів 29

2024

АНОТАЦІЯ

Дана програма містить в собі програмний код чат-боту Telegram, що э складовою комплексу онлайн-консультацій комп’ютерної системи КП «ДОКОЛ».

Програма призначена для ведення діалогу із користувачем, обробки запитів та файлів, та взаємодії з базою даних, що знаходиться на сервері підприємства.

Програма написана мовою Python з використанням ряду відповідних бібліотек у середовищі Visual Studio Code.

ЗМІСТ

1 МОДУЛЬ KPDOKOLBOT

```
# Імпорт бібліотек для асинхронної роботи та логування
import asyncio
import logging
# Імпорт необхідних елементів з бібліотеки aiogram
from aiogram import Bot, Dispatcher
# Імпорт токену
from app.config import TOKEN
# Імпорт хендлерів
from app.handlers import router
# Увімкнення логуваня
logging.basicConfig(level=logging.INFO)
# Привласнення токену об'єкту бота
bot = Bot(token  = TOKEN)
# Створення диспетчеру
dp = Dispatcher()
# Головна функція, що запускатиме роботу телеграм бота
async def main():
    dp.include_router(router)
    await bot.delete_webhook(drop_pending_updates=True)
    await dp.start_polling(bot)
# Запуск головної функції
if __name__ == "__main__":
    try:
        asyncio.run(main())
    except KeyboardInterrupt:
        print("Exit")
```

2 МОДУЛЬ HANDLERS

2.1 Завантаження бібліотек

```
#Імпорт необхідних елементів з бібліотеки aiogram
import os
from aiogram import Bot, Dispatcher, types
from aiogram import Router, F
from aiogram.filters import CommandStart, Command
from aiogram.types import Message, CallbackQuery
from aiogram.fsm.state import StatesGroup, State
from aiogram.fsm.context import FSMContext
from aiogram.types import FSInputFile
# Генерація випадкових унікальних id
import uuid
# Імпорт бібліотек для роботи із часом
from datetime import datetime
#Імпорт бібліотек для роботи із MongoDB
from pymongo import MongoClient
# Бібліотека для файлової взаємодії з БД
import gridfs
# Імпорт токену із захищеного файлу
from app.config import TOKEN
# Імпорт клавіатур для боту
import app.keyboards as kb
```

2.2 Додавання роутеру, БД та токену

```
# Створення роутеру
router = Router()
```

```
# Налаштування MongoDB
client = MongoClient('mongodb://localhost:27017/')
db = client['telegram_bot_db']  # Ім'я бази даних
collection = db['pacient_data']  # Ім'я колекції інформації пацієнтів
whitelist = db['whitelist'] # Ім'я колекції із списком фахівців спілки
doctorfiles = db['doctorfiles'] # Ім'я колекції для запитів консультації
fs = gridfs.GridFS(db) # Підключення GridFS

# Імпорт токену
bot = Bot(token=TOKEN)
```

2.3 Створення класів

```
# Створення класу для збору даних користувача пацієнта
class PacientReg(StatesGroup):
    name = State()
    birthdate = State()
    number = State()
    email = State()
    eHealth = State()
    telegram_id = State()
    # type_of_review = State()
    review = State()

# Створення класу для авторизації та взаємодії фахівців
class AuthStates(StatesGroup):
    waiting_for_auth_method = State()
    waiting_for_token = State()
    waiting_for_name = State()
```

```
    phantom_state = State() # Не видаляти ні в якому разі! (використовується для виходу зі стану авторизації)
    upload_id = State()
    waiting_for_comment = State()
    waiting_for_file_upload = State()
    request_consultation = State()
    waiting_for_eHealth = State()
    waiting_for_pacient_request = State()
    # test_state = State() # Використовується для тестування нових роутерів

# Створення класу для розсилки запрошення на конференцію
class Conference(StatesGroup):
    set_date = State()
    set_time = State()
    set_comment = State()
```

2.4 Створення універсальних функцій

2.4.1 Функція обробки успішної авторизації

```
async def handle_successful_auth(user, message, state: FSMContext):
    await message.answer(f"Авторизація пройшла успішно! Раді знову Вас бачити, {user['ПІБ']}!")
    await message.answer("Будь ласка, оберіть що Ви хотіли б зробити.",
                         reply_markup=kb.doctor_choice)
    await state.update_data(waiting_for_name=user['ПІБ'])
    await state.set_state(AuthStates.phantom_state)  # Перехід в новий стан
```

2.4.2 Функція запису даних користувача у БД

```
async def save_to_pacient_DB(data, file_id):
    # Намагаємось знайти існуючий запис по eHealth
```

```
    existing_patient = collection.find_one({
        'Номер електронного направлення': data['eHealth']
    })
    # Якщо документ вже існує, додаємо ім'я файлу у масив
    if existing_patient:
        collection.update_one(
            {'_id': existing_patient['_id']},
            {
                "$push": {
                    'ID файлу обстеження у БД': file_id,
                },
                "$set": {
                                'Час завантаження': datetime.now().strftime("%Y-%m-
%d %H:%M")  # Оновлюємо час завантаження
                }
            }
        )
    # Якщо такого документа ще не існує
    else:
        collection.insert_one({
            'ПІБ': data['name'],
            'Дата народження': data['birthdate'],
            'Номер телефону': data['number'],
            'Email': data['email'],
            'Telegram id': data['telegram_id'],
            'Номер електронного направлення': data['eHealth'],
            'Час завантаження': datetime.now().strftime("%Y-%m-%d %H:%M"),
            'ID файлу обстеження у БД': [file_id],
        })
```

2.4.3 Функція для збереження даних про звернення фахівця

```
async def save_to_doctor_db(data, file_id):
    existing_doctor = doctorfiles.find_one({
        'Унікальний ідентифікатор запиту': data['upload_id']
    })
    # Якщо документ вже існує, додаємо новий файл в масив
    if existing_doctor:
        doctorfiles.update_one(
            {'_id': existing_doctor['_id']},
            {
                "$push": {
                    'ID файлу у БД': file_id
                },
                "$set": {
                                    'Час завантаження': datetime.now().strftime("%Y-%m-%d %H:%M")  # Оновлюємо час завантаження
                }
            }
        )
    # Якщо документа ще не існує, створюємо новий запис
    else:
        doctorfiles.insert_one({
            'ПІБ': data['waiting_for_name'],
            'Час завантаження': datetime.now().strftime("%Y-%m-%d %H:%M"),
            'Унікальний ідентифікатор запиту': data['upload_id'],
            'ID файлу у БД': [file_id],
        })
```

2.4.4 Функція збереження файлових даних у БД

```
async def save_to_serverDB(data, message: types.Message):
    # Якщо це документ
    if message.document:
        file_id = message.document.file_id  # Отримуємо ID файлу
        file = await bot.get_file(file_id)  # Отримуємо об'єкт файлу
        filename=message.document.file_name # Зберігаємо ім'я файлу у зміну для повернення з функції
        # Зберігаємо файл в GridFS
            file_data = await bot.download_file(file.file_path)  # Завантажуємо зміст файлу
        fs.put(file_data, filename=filename)  # Зберігаємо в GridFS
            # Тестове повідомлення у боті у разі успішного збереження файлу. Використовувати для тестування
                # await message.answer(f"Файл сохранен в базе данных с именем {message.document.file_name}")

    # Якщо це зображення
    elif message.photo:
        file_id = message.photo[-1].file_id  # Отримуємо ID найбільшого (високої роздільної здатності) зображення
        file = await bot.get_file(file_id) # Отримуємо об'єкт файлу
        filename=f'{file_id}.jpg' # Зберігаємо ім'я файлу у зміну для повернення з функції
        # Зберігаємо зображення в GridFS
            file_data = await bot.download_file(file.file_path)  # Завантажуємо зміст зображення
        fs.put(file_data, filename=filename)  # Зберігаємо в GridFS
```

Тестове повідомлення у боті у разі успішного збереження файлу. Використовувати для тестування

await message.answer(f"Изображение сохранено в базе данных с именем {file_id}.jpg")

return file_id # Використовувати за необхідності повертати лише id зображення

return filename # Замість id (ім'я без формату у випадку із зображенням), повернення повного ім'я файлу із форматом

2.4.5 Функція для вивантаження файлу з БД

```
async def upload_from_serverDB(fileid):
    file_data = fs.find_one({"filename": fileid}) # Пошук файлу за його ім'ям
    # Якщо файл з таким ім'ям існує
    if file_data:
        file_bytes = file_data.read()  # Зчитуємо байти файлу
        # Створюємо тимчасовий файл для надсилання його через бот
            temp_file_path = f"tmp/{fileid}"  # Шлях до тимчасової директорії (використовується для сумісності бібліотек)
        with open(temp_file_path, 'wb') as temp_file:
            temp_file.write(file_bytes)  # Зберігаємо байти у тимчасовий файл
    else:
        print(f"Ошибка загрузки из базы данных.")
    return FSInputFile(temp_file_path)
```

2.5 Хендлери чат-боту

```
# Хендлер команди /start для початку роботи
@router.message(CommandStart())
async def start(message: Message):
```

await message.answer("Доброго дня! Яку консультацію Ви бажаєте отримати?",
reply_markup=kb.main)

2.5.1 Хендлери діалогу з пацієнтом

```
# Запит ПІБ
@router.message(F.text == "Я пацієнт")
async def step_one(message: Message, state: FSMContext):
    await state.clear()
    await state.set_state(PacientReg.name)
    await message.answer("Введіть Ваші ПІБ")

# Запит дати народження
@router.message(PacientReg.name)
async def name(message: Message, state: FSMContext):
    await state.update_data(name=message.text)
    await state.set_state(PacientReg.birthdate)
        await message.answer("Введіть Вашу дату народження у форматі: \n 01.01.1970")

# Запит номеру телефону
@router.message(PacientReg.birthdate)
async def birthdate(message: Message, state: FSMContext):
    await state.update_data(birthdate=message.text)
    await state.set_state(PacientReg.number)
        await message.answer("Введіть Ваш номер телеофну у форматі: \n +38**********")
```

```
# Запит електронної пошти
@router.message(PacientReg.number)
async def number(message: Message, state: FSMContext):
    await state.update_data(number=message.text)
    await state.set_state(PacientReg.email)
        await message.answer("Введіть Вашу адресу електронної пошти або прочерк. \n\nПриклад пошти: example@example.com")

# Запит електронного направлення
@router.message(PacientReg.email)
async def email(message: Message, state: FSMContext):
    await state.update_data(email=message.text)
    await state.set_state(PacientReg.eHealth)
    await message.answer("Введіть номер Вашого електронного напралення з 16 цифр у форматі:\n XXXX-XXXX-XXXX-XXXX")

# Виклик перевірки внесених даних
@router.message(PacientReg.eHealth)
async def eHealth(message: Message, state: FSMContext):
    await state.update_data(eHealth=message.text)
        await  state.update_data(telegram_id=message.from_user.id) # Отримання також Telegram ID
    data = await state.get_data()
       await message.answer(f'Ваші данні наступні:\nПІБ: {data["name"]}\nДата народження: {data["birthdate"]}\nНомер телефону: {data["number"]}\nЕлектронна пошта: {data["email"]}\nЕлектронне направлення: {data["eHealth"]} \n\nВнесені дані вірні?',
                            reply_markup=kb.back_to_step_one)
```

```
# Повернення у початок внесення даних пацієнтом
@router.callback_query(F.data == "step_one")
async def back_to_step_one(callback: CallbackQuery, state: FSMContext):
    await state.clear()
    await state.set_state(PacientReg.name)
    await callback.message.answer("Введіть Ваші ПІБ")

# Запит файів обстеження пацієнта
@router.callback_query(F.data == "next_step")
async def review(callback: CallbackQuery, state: FSMContext):
    await state.update_data(review=None)
    await state.set_state(PacientReg.review)
        await callback.message.answer('Відправте файл з результатом вашого обстеження. \nВідправляючи файл, Ви погоджуєтесь на його обробку та аналіз нашими фахівцями. \nНатисніть "Завершити" коли завантажете всі необхідні файли',
                      reply_markup=kb.End_upload)

# Цикл внесення файлів пацієнтом
@router.message(PacientReg.review)
async def handle_file_or_photo(message: Message, state: FSMContext):

    if message.text == "Завершити":
        data = await state.get_data()
            await message.answer(f'Ваші дані прийнято.\nЇх буде розглянуто фахівцем КП «ДОКОЛ».\nКонсультація відбудеться в час, який було узгоджено з Вами при записі на прийом.\nНагадуємо номер телефону для консультації: +380971234567\nВ узгоджений час фахівець чекатиме на Ваш
```

```
дзвінок.  \n\nЗа підсумком проведеної консультації, Вам буде надіслано відповідний документ з висновком лікаря.',
                    reply_markup=kb.back_to_main)
    await state.clear()
  elif not (message.document or message.photo):
        await message.answer("Будь-ласка, надсилайте файли, без текстових коментарів та повідомлень.")

  else:
    data = await state.get_data()
    # Виклик функції зберігання файлів користувача
      # filepath = await save_to_server(data, message) # Використовувати за необхідності запису файлу у директорію
    fileid =  await save_to_serverDB(data, message)
    if fileid:
            # await save_to_DB(data, filepath, fileid) # Використовувати за необхідності запису файлу у директорію
      await save_to_pacient_DB(data, fileid)
```

2.5.2 Хендлери діалогу з фахівцем

2.5.2.1 Процес авторизації

```
# Обрання методу авторизації для фахівця
@router.message(F.text.in_(["Я фахівець", "Обрати інший метод авторизації"]))
async def auth_handler(message: types.Message, state: FSMContext):
  await state.clear()
            # reply_markup = types.ReplyKeyboardMarkup(keyboard, resize_keyboard=True)
      await message.answer("Будь-ласка, оберіть спосіб авторизації:", reply_markup=kb.Auth)
```

```
    # Переходимо у стан вибору методу авторизації
    await state.set_state(AuthStates.waiting_for_auth_method)

# Хендлер для вибору способу авторизації
@router.message(AuthStates.waiting_for_auth_method)
async def auth_method_choice(message: types.Message, state: FSMContext):
    # Дії в залежності від обраного методу авторизації
    if message.text == "За Telegram id":
        # Отримуємо Telegram id користувача та здійснюємо пошук його у білому списку
        user_id = message.from_user.id
        user = whitelist.find_one({"user_id": user_id})
        # Дії в залежності від успіху перевірки за білим списком
        if user:
            await handle_successful_auth(user, message, state)
        else:
            await message.answer("Помилка! У Вас немає доступу. \nБудь-ласка, спробуйте авторизуватись за токеном або зв'яжіться з системним адміністратором.",
                                 reply_markup=kb.back_to_main_auth)
            # Очищення стану
            await state.clear()

    elif message.text == "За особистим токеном":
        await message.answer("Будь-ласка, введіть Ваш токен наступним повідомленням.")
        # Перехід у стан очікування токену
        await state.set_state(AuthStates.waiting_for_token)
    else:
```

```
        await message.answer("Будь-ласка, оберіть один з запропонованних методів авторизації.")

# Хендлер для обробки введеного токена
@router.message(AuthStates.waiting_for_token)
async def token_auth(message: types.Message, state: FSMContext):
    # Отримуємо токен користувача та здійснюємо пошук його у білому списку
    token = message.text
    user = whitelist.find_one({"token": token})
    # Дії в залежності від успіху перевірки за білим списком
    if user:
        await handle_successful_auth(user, message, state)
    elif message.text == "Обрати інший метод авторизації":
        await state.clear()
    else:
            await message.answer("Невірний токен авторизації. Будь-ласка, спробуйте ще раз. \nУ разі виникнення проблем, спробуйте авторизуватись за Telegram id або зв'яжіться з системним адміністратором.",
                                 reply_markup=kb.back_to_main_auth)
        await state.set_state(AuthStates.waiting_for_token)
```

2.5.2.2 Звернення на проведення консультації спілки

```
# Обробка обрання варіанту звернення за консультацією
@router.message(F.text == "Звернутись за консультацією до спілки офтальологів")
async def doctor_main(message: Message, state: FSMContext):
    await state.set_state(AuthStates.waiting_for_comment)
    await state.update_data(upload_id = str(uuid.uuid4()))
```

```
    await message.answer("Будь-ласка, введіть причину запиту та відправте єдиним текстовим повідомленням.")

# Надання коментарю до звернення
@router.message(AuthStates.waiting_for_comment)
async def birthdate(message: Message, state: FSMContext):
    await state.update_data(waiting_for_comment=message.text)
    await state.set_state(AuthStates.waiting_for_file_upload)
    await message.answer('Будь-ласка, надішліть файл(и) обстеження.\nНатисніть "Завершити", коли надішлете всі файли.',
                         reply_markup=kb.End_upload)

# Обробка та зберігання файлів від спеціаліста з подальшим надсиланням запиту із файлами членам спілки
@router.message(AuthStates.waiting_for_file_upload)
async def handle_doctors_file_or_photo(message: Message, state: FSMContext):
    # Цикл запиту файлів поки користувач не обере "Завершити"
    if message.text == "Завершити":
        # Розсилка запиту на консультацію з файлами
        data = await state.get_data()
          await message.answer(f'Ваш файл(и) прийнято.\nЗапит на консультацію спілки буде відправлено')
        # Пошук наданих файлів за ідентифікаторами у БД
         document = doctorfiles.find_one({'Унікальний ідентифікатор запиту': data['upload_id']})
        # Записуємо всі імена файлів із запиту на консультацію
        file_ids = document.get('ID файлу у БД', [])
        # Створюємо масив із файлами на відправку
        files = []
```

```
        for file_id in file_ids:
            file = await upload_from_serverDB(file_id) # Вивантаження файлів з БД у тимчасову директорію
            files.append(file)  # Додаємо кожен завантажений файл у масив
        # Якщо у запиті є імена файлів, виконуємо такі дії
        if file_ids:
            # Вилучення користувачів із білого списку для розсилки
            all_whitelist_users = whitelist.find()
            # whitelist_users = whitelist.find({"Role": "professor"}) # Підключити у разі необхідності обмежети розсилку
            # Цикл розсилки всім користувачам з отриманого списку
            for user in all_whitelist_users:
                try:
                    telegram_id = user['user_id']  # Отримуємо Telegram ID людей із списку
                    # Надсилання універсального привітального повідомлення
                    if user['user_id'] != message.from_user.id:
                        await bot.send_message(chat_id=telegram_id, text=f"Доброго дня, {user['ПІБ']}. Я, {data['waiting_for_name']}, прошу Вас розглянути наданий файл(и) обстеження для проведення консультації з наступного питання:\n{data['waiting_for_comment']}")
                        # Цикл відправки всіх файлів із запиту кожному конкретному члену спілки
                        for file in files:
                            await bot.send_document(chat_id=telegram_id, document=file) # Отправляем как документ
                        print(f"Сообщение отправлено пользователю с Telegram ID: {telegram_id}") # Вывод в консоль для тестировки
                except Exception as e:
```

```
                print(f"Ошибка отправки сообщения пользователю с Telegram ID: {telegram_id}. Ошибка: {e}")
        # У разі успіху надіслання запитів
            await message.answer(f"Запит на проведення консультації успішно відправлено.",
                                 reply_markup=kb.back_to_main)
        # Видалення тимчасових файлів
        for file_id in file_ids:
            os.remove(f"tmp/{file_id}") # Видалення тимчасового файлу(ів) з використання функції
    else:
            print(f"Файл з ім'ям {file_ids} не знайдено для користувача {telegram_id}.")
    # Очищення стану в разі успіху розсилки
    await state.clear()

elif not (message.document or message.photo):
        await message.answer("Будь-ласка, надсилайте файли, без текстових коментарів та повідомлень.")

# Цикл завантаження файлів фахівцем
else:
    data = await state.get_data()
    # Виклик функції зберігання файлів користувача
        # filepath = await save_to_server(data, message) # Увімкнути при необхідності зберікати файли у директорію
    fileid =  await save_to_serverDB(data, message)
    if fileid:
        await save_to_doctor_db(data, fileid)
```

2.5.2.3 Проведення консультацій пацієнтів

```
# Обробка запиту користувача про надання консультації пацієнту
@router.message(F.text == "Надати консультацію пацієнту")
async def doctor_main(message: Message, state: FSMContext):
    await state.set_state(AuthStates.waiting_for_eHealth)
        await message.answer('За яким електронним направленням Ви бажаєте надати консультацію?\nВведіть його наступним повідомленням, без дефісів та пропусків.')

# Хендлер переходу до консультації наступного пацієнта
@router.callback_query(F.data == "back_to_pacients")
async def back(callback: CallbackQuery, state: FSMContext):
    await state.set_state(AuthStates.waiting_for_eHealth)
        await callback.message.answer('За яким електронним направленням Ви бажаєте надати консультацію?\nВведіть його наступним повідомленням, без дефісів та пропусків.')

# Отримання електронного направлення пацієнта
@router.message(AuthStates.waiting_for_eHealth)
async def doctor_main(message: Message, state: FSMContext):
    await state.update_data(waiting_for_eHealth = message.text)
    data = await state.get_data()
        document = collection.find_one({'Номер електронного направлення': data['waiting_for_eHealth']})
    # Дії в залежності від наявності запису за таким електроним направленням
    if document:
        # Записуємо всі імена файлів із запиту на консультацію
        file_ids = document.get('ID файлу обстеження у БД', [])
        # Створюємо масив із файлами на відправку
```

```
        files = []
        for file_id in file_ids:
            file = await upload_from_serverDB(file_id)
            files.append(file)  # Додаємо кожен завантажений файл у масив
        await message.answer('Файли обстеження пацієнта:')
        # Відправка всіх отриманих файлів фахівцю
        for file in files:
            try:
                    await bot.send_document(chat_id=message.from_user.id, document=
file) # Відправляємо як документ
                        print(f"Сообщение отправлено пользователю с Telegram ID:
{message.from_user.id}") # Виведення в консоль для тестування
            except Exception as e:
                print(f"Ошибка отправки сообщения пользователю с Telegram ID:
{message.from_user.id}. Ошибка: {e}")
        for file_id in file_ids:
            os.remove(f"tmp/{file_id}")
        await state.set_state(AuthStates.waiting_for_pacient_request)
            await message.answer('Будь-ласка, надайте файл із призначенням за
обстеженнями пацієнта:')
    else:
            await message.answer('Запису за таким електроним направленням не
знайдено!\nБудь-ласка, перевірте коректність та спробуйте ввести
електронне направлення знову або поверніться у головне меню.',
                                 reply_markup=kb.back_to_patient_choice)
                print(f"Файл з ім'ям {file_ids} не знайдено для користувача
{message.from_user.id}.")
```

```
# Хендлер надання висновку лікаря
@router.message(AuthStates.waiting_for_pacient_request)
async def doctor_main(message: Message, state: FSMContext):
    # Перевірка на коректність наданого файлу висновку
    if not (message.document):
            await message.answer("Будь-ласка, надсилайте файл, без текстових коментарів та повідомлень.")

    else:
        data = await state.get_data()
        # Виклик функції зберігання файлу висновку у БД
            # filepath = await save_to_server(data, message) # Увімкнути за необзідності зебереження у директорію
        fileid =  await save_to_serverDB(data, message)
            # Етап надсилання відповіді та висновку пацієнту, що отримав консультацію
        # Вивантаження файлу у тимчасову директорію
        file = await upload_from_serverDB(fileid)
            # Пошук пацієнта за номером елетроного направлення та отримання його Telegram id
            pacient = collection.find_one({'Номер електронного направлення': data['waiting_for_eHealth']})
        telegram_id = pacient['Telegram id']
        # Відправка повідомлення та файлу висновку пацієнту
            await bot.send_message(chat_id=telegram_id, text=f"Доброго дня, {pacient['ПІБ']}. За результатами Вашого обстеження Вам було надано наступне призначення:")
        await bot.send_document(chat_id=telegram_id, document=file)
        # Очищення стану та тимчасової директорії
```

```
    os.remove(f"tmp/{fileid}")
    await state.clear()
    # Запит подальших дій фахівця
        await message.answer("Бажаєте повернутись до головного меню або обрати наступного пацієнта?",
                             reply_markup=kb.back_to_patient_choice)
```

2.5.2.4 Створення запрошення на конференцію спілки

```
# Призначення дати конференції
@router.message(F.text == "Призначити конференцію спілки офтальмологів")
async def broadcast(message: types.Message, state: FSMContext):
    # Отримуємо Telegram ID користувача
    user_id = message.from_user.id
    # Шукаємо користувача в базі даних за участю "professor"
    user = whitelist.find_one({"user_id": user_id, "Role": "professor"})
    await state.clear()
    # Дії в залежності від того чи має користувач доступ
    if user:
        await state.set_state(Conference.set_date)
            await message.answer("Призначте день (число та місяць), у який планується проведення конференції.")
    else:
        # Якщо не professor, повідомляємо про відсутність прав доступу
          await message.answer("У вас немає прав для призначення конференції. \nЯкщо Ви важаєте, що це помилка, будь-ласка зверніться до системного адміністратора.",
                             reply_markup=kb.back_to_main)
```

```
# Призначення часу конференції
@router.message(Conference.set_date)
async def birthdate(message: Message, state: FSMContext):
    await state.update_data(set_date=message.text)
    await state.set_state(Conference.set_time)
    await message.answer("Призначте час проведення конференції.")

# Надання коментарю до конференції
@router.message(Conference.set_time)
async def birthdate(message: Message, state: FSMContext):
    await state.update_data(set_time=message.text)
    await state.set_state(Conference.set_comment)
      await message.answer("Будь-ласка, введіть опис проводимої конференції (єдиним повідомленням).")

# Розсилка запрошення на конференцію
@router.message(Conference.set_comment)
async def birthdate(message: Message, state: FSMContext):
    await state.update_data(set_comment=message.text)
    data = await state.get_data()
    # Вилучення даних користувачів із білого списку
    whitelist_users = whitelist.find()
    # Цикл розсилки запрошення
    for user in whitelist_users:
        try:
            telegram_id = user['user_id']  # Отримуємо Telegram ID
            # Надсилаємо повідомлення про конференцію
```

```
            await bot.send_message(chat_id=telegram_id, text=f"Запрошуємо Вас, {user['ПІБ']}, взяти участь в конференції, що пройде {data['set_date']}, в {data['set_time']}.\nОпис конференції: {data['set_comment']}")
                print(f"Сообщение отправлено пользователю с Telegram ID: {telegram_id}")
        except Exception as e:
            print(f"Ошибка отправки сообщения пользователю с Telegram ID: {telegram_id}. Ошибка: {e}")
```

2.6 Загальний хендлер повернення до головного меню

```
@router.callback_query(F.data == "back")
async def back(callback: CallbackQuery, state: FSMContext):
    await state.clear()
    await callback.message.answer("Доброго дня! Яку консультацію Ви бажаєте отримати?",
                                  reply_markup=kb.main)
```

3 МОДУЛЬ KEYBOARDS

```
# Імпорт необхідних елементів з бібліотеки aiogram
from aiogram.types import ReplyKeyboardMarkup, KeyboardButton, InlineKeyboardMarkup, InlineKeyboardButton
```

3.1 Клавіатури типу reply

```
# Клавіатура Головного меню
main = ReplyKeyboardMarkup(keyboard=[
   [KeyboardButton(text="Я пацієнт"), KeyboardButton(text="Я фахівець")]
],
    input_field_placeholder="Будь-ласка, оберіть свій варіант:",
    one_time_keyboard= True,
    resize_keyboard= True,
)

# Клавіатура методу авторизації
Auth = ReplyKeyboardMarkup(keyboard=[
            [KeyboardButton(text="За Telegram id"), KeyboardButton(text="За особистим токеном")]
],
    one_time_keyboard= True,
    resize_keyboard= True,
)

# Клавіатура меню фахівця
doctor_choice = ReplyKeyboardMarkup(keyboard=[
    [KeyboardButton(text="Надати консультацію пацієнту")],
```

```
            [KeyboardButton(text="Звернутись за консультацією до спілки офтальмологів")],
        [KeyboardButton(text="Призначити конференцію спілки офтальмологів")]
],
        input_field_placeholder="Будь-ласка, оберіть свій варіант:",
        one_time_keyboard= True,
)

# Навігація на етапі авторизації для повернення до вибору методу авторизації
back_to_main_auth = ReplyKeyboardMarkup(keyboard=[
    [KeyboardButton(text="Обрати інший метод авторизації")]

],
        resize_keyboard= True,
)

# Клавіатура для завершення завантаження файлів
End_upload = ReplyKeyboardMarkup(keyboard=[
    [KeyboardButton(text="Завершити")]
],
        input_field_placeholder='Натисніть коли завантажете всі необхідні файли',
        one_time_keyboard= True,
        resize_keyboard= True,
```

3.2 Клавіатури типу inline

```
# Кнопки навігації
# Навігація при перевірці внесених даних пацієнтом
back_to_step_one = InlineKeyboardMarkup(inline_keyboard=[
    [InlineKeyboardButton(text="Так", callback_data="next_step"),
```

```
    InlineKeyboardButton(text="Hi", callback_data="step_one")]
])

# Навігація фахівця при проведенні консультації
back_to_patient_choice = InlineKeyboardMarkup(inline_keyboard=[
                [InlineKeyboardButton(text="Обрати наступного пацієнта", callback_data="back_to_pacients")],
                    [InlineKeyboardButton(text="Повернутись на головну", callback_data="back")]

])

# Загальна кнопка навікації для повернення до головного меню
back_to_main = InlineKeyboardMarkup(inline_keyboard=[
                    [InlineKeyboardButton(text="Повернутись на головну", callback_data="back")]
])
```