

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,
Факультет комп'ютерних наук і технологій

(повне найменування інституту, назва факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему Програмна реалізація Телеграм-бота служби технічної підтримки
Software Implementation of a Telegram Bot for the Technical Support Service

Виконав: студент(ка) 4 курсу, групи КНТ-128т
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

Вічева Г.Ю.

(прізвище та ініціали)

Керівник Каплієнко Т.І.

(прізвище та ініціали)

Рецензент Шитікова О.В.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ІРЕ, ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
“ ” 20 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Вічевої Ганни Юріївни

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Програмна реалізація Телеграм-бота служби технічної підтримки. Software Implementation of a Telegram Bot for the Technical Support Service

керівник проєкту (роботи) Каплієнко Тетяна Ігорівна, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “30” березня 2021 року № 103

2. Строк подання студентом проєкту (роботи) 17 травня 2021 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Опис програми. 3. Розробка програми. 4. Експлуатація, тестування та експериментальне дослідження програми. 5. Додатки.

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

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	Каплієнко Т.І., к.н.т, доцент		
Нормоконтролер	Камінська Ж.К., асистент		

7. Дата видачі завдання “ 22 ” лютого 2021 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання
2	Аналіз предметної області.	2 тиждень	Розділ 1
3	Розробка архітектури програми.	3 тиждень	Розділ 2
4	Розробка програми.	4–6 тижні	Розділ 3
5	Тестування та експериментальне дослідження програмного забезпечення.	7 тиждень	Розділ 4
6	Оформлення пояснювальної записки та документів до неї.	8-9 тижні	Додатки
7	Нормоконтроль та рецензування.	10–11 тижні	
8	Захист роботи.	12 тиждень	

Студент(ка)

(підпис)

Вічева Г.Ю.

(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис)

Каплієнко Т.І.

(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 165 с., 6 табл., 82 рис., 2 дод., 16 джерел.

ТЕХНІЧНА ПІДТРИМКА, КОМП'ЮТЕРНА ПРОГРАМА, TELEGRAM, БОТ, PYTHON, API, TELEBOT.

Об'єкт дослідження – внутрішні процеси роботи служби технічної підтримки підприємства.

Предмет дослідження – комп'ютерні програми для обробки заявок користувачів службою технічної підтримки.

Мета роботи – створення програмної системи для автоматизації та прискорення обробки заявок користувачів службою технічної підтримки підприємства.

Матеріали, методи та технічні засоби: структурне програмування, мова програмування Python, будь-який пристрій, на який можливо встановлення Telegram.

Результати. Створено програмний засіб, що вбудовано у месенджер Telegram, який автоматизує та систематизує подачу заявок користувачем до служби технічної підтримки підприємства.

Висновки. Розроблено програмний засіб для подачі заявок до служби технічної підтримки, що прискорює, автоматизує та систематизує цей процес.

Галузь використання – служба технічної підтримки підприємств, великих корпорацій або середніх масштабів компанії.

ABSTRACT

Explanatory note to the bachelor's thesis: 165 pages, 6 tables, 82 figures, 2 appendices, 16 sources.

TECHNICAL SUPPORT, COMPUTER PROGRAM, TELEGRAM, BOT, PYTHON, API, TELEBOT.

The object of research - the internal processes of the technical support service of the enterprise.

The subject of the research is computer programs for processing user requests by technical support service.

The purpose of the work is to create a software system for automation and speed up the processing of user requests by the technical support service of the enterprise.

Materials, methods and technical means: structural programming, Python programming language, any device on which Telegram can be installed.

Results. A software tool built into the Telegram messenger has been created, which automates and systematizes the submission of applications by the user to the technical support service of the enterprise.

Conclusions. A software tool for submitting applications to the technical support service has been developed, which accelerates, automates and systematizes this process.

Field of application - technical support service for enterprises, large corporations or medium-sized companies.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	7
Вступ.....	8
1 Аналіз предметної області.....	10
1.1 Поняття чат-бота	10
1.2 Застосування чат-ботів в бізнесі.....	11
1.2.1 Бот каршерінга YouDrive	12
1.2.2 Бот «Aviasales»	13
2 Опис програми.....	16
2.1 Порівняльний аналіз мов програмування.....	16
2.2 Порівняння платформ	19
2.3 Опитування зацікавлених осіб.....	22
2.4 Архітектура програми.....	24
2.5 Висновки з розділу 2.....	32
3 Розробка програми	33
3.1 Методи та функції	35
3.2 Команди.....	46
3.3 Інтерфейс.....	51
4 Експлуатація, тестування та експериментальне дослідження програми	76
4.1 Опис застосування	76
4.2 Керівництво адміністратора.....	76
4.3 Керівництво модератора.....	76
4.3 Керівництво користувача	78
4.4 Тестування	79
Висновки	82
Перелік джерел посилання	84
Додаток А Текст програми.....	86
Додаток Б Слайди презентації	152

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API	– Application Programming Interface;
IDE	– Integrated Development Environment;
INTERFACE	– об’єкт взаємодії користувача та програми;
Internet	– всесвітня система взаємополучених комп.мереж;
inline	– клавіатура, що прикріплена до повідомлення;
reply	– стандартна клавіатура під полем введення повідомлень;
IT	– інформаційні технології;
БД	– база даних;
БОТ	– скорочення від «робот», програма;
ВС	– вбудована система;
ПЗ	– програмне забезпечення;
ПП	– програмний продукт;
ТП	– технічна підтримка.

ВСТУП

На даний момент на підприємствах найчастіше функціонують ІТ-служби або відділи технічної підтримки. З ростом кількості співробітників підприємства збільшується кількість звернень до технічної підтримки [1].

Існує множина питань, які є приводом приблизно сімдесяти відсотків звернень до технічної підтримки. Відповідь на будь-яке питання з цієї множини у співробітника ІТ-служби в середньому займає від 5 до 15 хвилин, якщо відповідь на питання заздалегідь відома [1].

В цілому тривалість відповіді не велика, але зі зростанням кількості співробітників підприємства зростає кількість заявок, що надходять до ІТ-служби, відповідно, збільшується черга з питань, які необхідно з'ясувати. Чим більше часу заявка знаходиться без уваги, тим більший ризик простоїв підприємства та втрат робочого часу інших співробітників. Отже для технічної підтримки вкрай важливо скоротити чергу із заявок [1].

Існує декілька варіантів вирішення даної проблеми. Підприємство або розширює штат операторів, які відповідають на заявки до технічної підтримки, або автоматизує обробку певної частини заявок. Економічно доцільніше одноразове розроблення програмного продукту, ніж постійне утримання додаткової робочої сили [1].

Перед обранням технологій та засобів розробки важливо розуміти перелік необхідного функціоналу майбутнього програмного продукту. Коли сформовано ціль розробки, варто також врахувати технічні особливості підприємства (стабільність доступу до мережі Інтернет, тощо) [1].

Найчастіше автоматизується інформаційна функція (отримання інформації за запитом користувача) або інформаційна розсилка, та частково функція спілкування [1].

Через деякий проміжок часу роботи ІТ-служби накопичується перелік питань, на які заздалегідь відомі відповіді, що несуть виключно інформаційний характер. Обробку саме таких заявок було вирішено

автоматизувати, оскільки якщо питання повторюються, надання стандартних відповідей на такі питання не є ефективно використаним часом оператора. Також було вирішено реалізувати для користувача можливість залишити питання в певній категорії [1].

Для реалізації програмного продукту доцільно використовувати звичні пересічному користувачу засоби взаємодії, також варто забезпечити кросплатформність. Ідеально підходить варіант месенджера, оскільки важко знайти співробітника, який не користується останнім, також таким чином буде дотримано вимогу про кросплатформність, бо для більшості месенджерів реалізовано як мобільну, так і десктопну версії [1].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Поняття чат-бота

Термін «чат-бот» вперше був використаний в 1994 році і придуманий Майклом Молдіном, творцем «Verbot (Verbal Robot) Julia» [2].

Існує декілька визначень цього терміну, всі вони по різному описують сутність цього поняття.

Чат-боти - це механізми відповідей на основі даних в реальному часі, які говорять на природній мові і залежать від контексту [3].

Чат-боти - це спеціальні акаунти, за якими не закріплена людина, а повідомлення, відправлені з них або на них, оброблюються зовнішньою системою. Крім того, для користувача спілкування з ботом виглядає як звичайне листування з реальною людиною [4, 5].

Чат-бот дозволяє ефективно створювати інтерфейс обробки природної мови для програм і баз даних, надаючи відповіді користувачам, структуруючи мову відповідно до вимог і цілей розробки [6].

Чат-бот - це програма, яка виконується в месенджерах і виконує різні функції. Виділяють наступні основні функції бота [4, 7]:

- технічна підтримка;
- допомога у виборі продукту;
- консультація за типовими питаннями;
- збір клієнтів;
- прийом і оформлення замовлень;
- розвага користувачів;
- пошук інформації;
- підбір кандидатів або пошук роботи;
- розсилка повідомлень;
- нестандартні функції.

1.2 Застосування чат-ботів в бізнесі

Розробка чат-ботів зараз набагато простіше, ніж це було кілька років тому, але чат-боти існували і десятиліття тому, проте популярність чат-ботів за останні кілька років зростає в геометричній прогресії [8].

Згідно з дослідженням LiveWorld, три роки тому 58% маркетологів не хотіли вкладати кошти в месенджери і чат-ботів [4].

Facebook підняв кількість чат-ботів з нуля в лютому 2016 року до 18000 до липня того ж року [9].

У той же час аналітики зробили більш оптимістичні прогнози щодо автоматизованих процесів і штучного інтелекту - 85% назвали месенджер-маркетинг новим стандартом для бізнесу [4]. І вони мали рацію.

Середньостатистичні користувачі смартфонів перевіряють свої телефони 150 раз в день, відповідаючи на повідомлення в Facebook, електронною поштою або Twitter [10].

Соціальні мережі процвітають вже майже 10 років: вони є не тільки платформою для спілкування між людьми, а й для ведення бізнесу. Багато компаній зобов'язані своєю популярністю і фінансовим успіхом соціальним мережам [4].

Кардинальне зрушення відбулося в 2018 році. Близький родич соціальних мереж - Messenger - обігнав соціальні мережі за кількістю користувачів. За оцінками The Economist, люди витрачають більше двох годин в день на застосунки для обміну повідомленнями. Така активна аудиторія могла впливати тільки на нові тенденції в маркетингу [4].

Компанії використовували не тільки месенджери в якості повноцінної платформи для спілкування з клієнтами і рекламних послуг, але і для прямих продажів [4, 11].

Переваги месенджерів:

- конфіденційність. Месенджери WhatsApp, Facebook Messenger, Viber і Telegram – це канали, за якими інформація передається безпосередньо

одержувачу. Крім того, більшість програм обміну повідомленнями мають додаткову наскрізну систему шифрування, що також робить цю платформу безпечною для банківського сектора;

- управління репутацією. Спілкування тет-а-тет через месенджери набагато краще. По перше, проблеми клієнтів не стануть предметом обговорення для всієї аудиторії Facebook і ЗМІ. В особистому листуванні, можна виконувати запит більш якісно і ймовірність, що клієнт залишиться задоволеним, зростає. По-друге, при листуванні в месенджері можна виявити як позитивні якості компанії, так і негативні;

- зручне середовище для зв'язку. В основі месенджерів - асинхронність, швидкість і доступ до всіх діалогів в одному місці. Поєднання всіх цих властивостей робить їх основним цифровим засобом комунікації [4].

Згідно з новим дослідженням, глобальні продажі штучного інтелекту виростуть до 126 мільярдів доларів до 2025 року в порівнянні з 10,1 мільярдами доларів в 2018 році. Тепер штучний інтелект має своє продовження в чатах. З огляду на те, як бренди впроваджують віртуальних помічників, очікується, що вони стануть основними інвестиціями в обслуговування клієнтів протягом наступних кількох років [4].

Нижче наведено приклади ботів для бізнесу.

1.2.1 Бот каршерінга YouDrive

Приклад меню бота каршерінга YouDrive наведено на рисунку 1.1.

Платформою цього боту є Telegram.

Для бізнесу бот корисний тим, що на $\frac{2}{3}$ знизив число дзвінків в компанію.

Основна мета бота: зробити процес повідомлення про проблеми з автомобілем або прокатом простим і зручним для клієнта, а також підвищити швидкість обробки викликів співробітниками служби, що скорочує час очікування рішення [12, 13].

Функції перелічені нижче:

- бот пропонує готові миттєві рішення різних питань: від «як завести автомобіль» до «що робити, якщо я потрапив в ДТП»;
- розподіляє звернення від користувачів по відповідальним співробітникам: кожен тип звернення направляється до відповідного відділу для оперативного вирішення проблеми [4, 12].

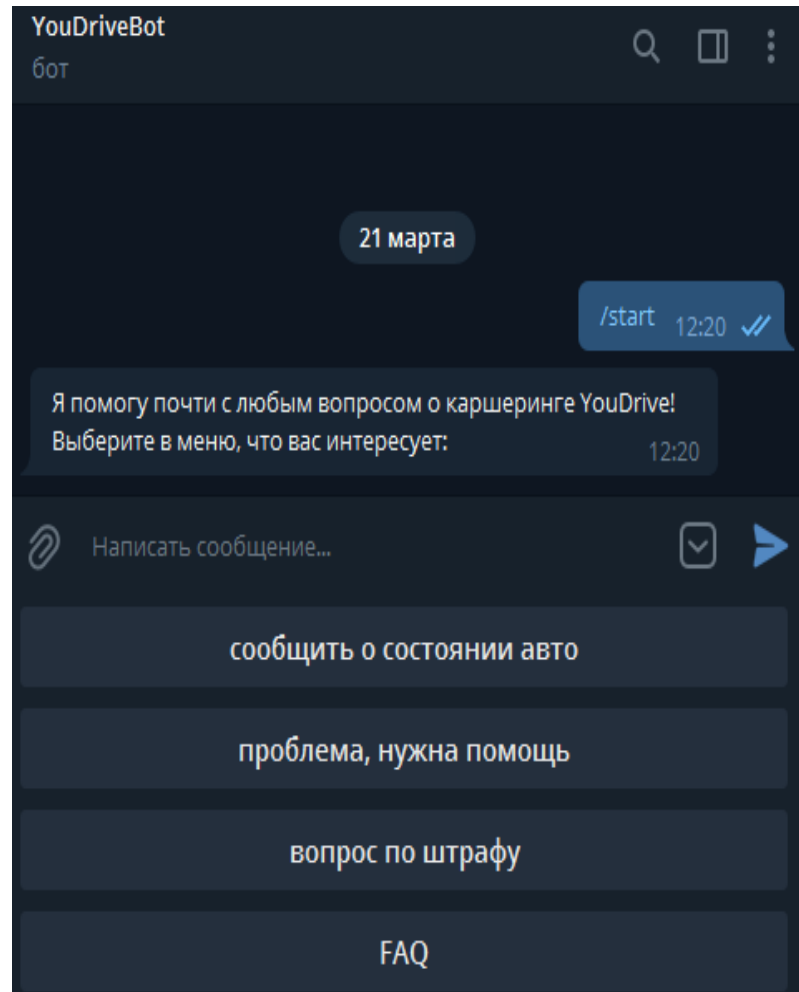


Рисунок 1.1 - Чат-бот каршеринга YouDrive

1.2.2 Бот «Aviasales»

Приклад меню бота «Aviasales» наведено на рисунку 1.2.

Платформою цього боту є Telegram.

Основним призначенням боту є замовлення та моніторинг авіабілетів, дозволяє слідкувати за знижками [4, 14].

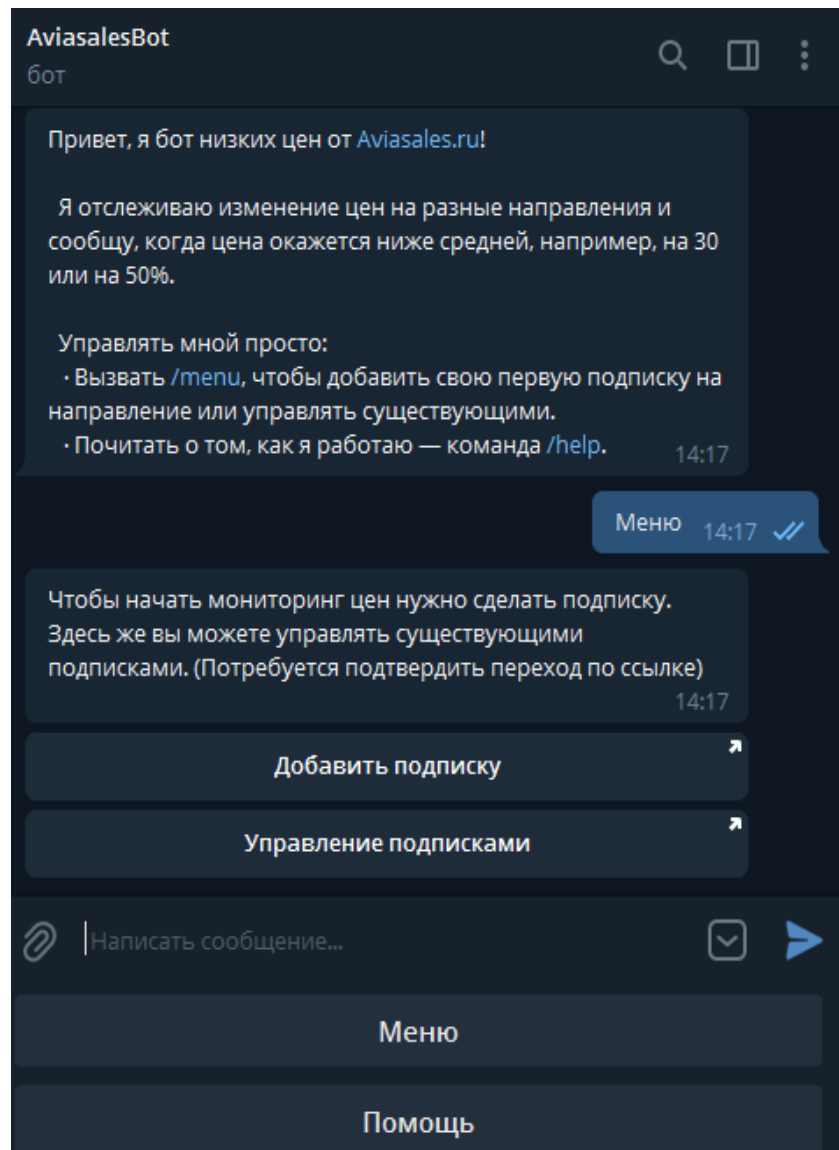


Рисунок 1.2 - Чат-бот Aviasales

Наразі боти застосовуються майже у всіх областях бізнесу, дуже розповсюдженими є технічні підтримки, оскільки це забезпечує високий рівень автоматизації обробки та розподілу заявок.

Відповідно до успішного досвіду застосування ботів та з огляду на той факт, що повного аналогу боту, що розроблюється, не існує, прийнято рішення про актуальність розробки даного програмного продукту.

1.3 Постановка завдання роботи

Для технічної підтримки будівельної компанії «АЛД ІНЖИНІРИНГ ТА БУДІВНИЦТВО» необхідно розробити бота.

Найменування продукту – бот «SupportUkBot».

Програмний продукт (ПП) призначений для прийняття заявок про технічні проблеми підприємства. Область застосування – підприємства, що мають на балансі офісну техніку, корпоративне програмне забезпечення.

Дана розробка дає можливість користувачу залишити заявку про технічні проблеми з корпоративної технікою та/або програмним забезпеченням. Також користувач має доступ до інструкцій до корпоративного програмного забезпечення (ПЗ), навчальних відео та застосунків корпоративного ПЗ.

2 ОПИС ПРОГРАМИ

2.1 Порівняльний аналіз мов програмування

Для написання програмного коду бота найчастіше використовуються наступні мови програмування: Python, Php, Java, C# [15]. Нижче буде наведено порівняльний аналіз найпопулярнішої мови для написання ботів – Python з іншими мовами програмування.

В таблиці 2.1 наведено порівняння мов Python і Php.

Таблиця 2.1 - Порівняння мов Python і Php [15]

	Python	Php
Популярність	Дуже популярний	Дуже популярний
Фреймворки	Мало фреймворків	Багато фреймворків
Вивчення	Легше вчити	Складніше вчити

Що стосується популярності, Python і PHP не поступаються один одному: вони обидва дуже широко застосовуються в розробці. Наприклад, PHP використовувався при створенні таких проєктів як Вікіпедія, Yahoo, WordPress та ін. А Python вибрали для створення YouTube, Instagram, Quora, Spotify і ін. [15].

Одним з важливих чинників при виборі мови програмування є число фреймворків. Вони суттєво спрощують роботу розробника, роблять її більш комфортною. До найпопулярніших фреймворків PHP відносяться Laravel, Symfony, CodeIgniter, але є й інші PHP-фреймворки. Звичайно, у Python фреймворків менше. Найпопулярніші з них - Django і Flask. Але зростаючі спільноти схильні створювати все нові фреймворки, так що і число фреймворків Python незабаром може змінитися [15].

Що стосується простоти в вивченні, Python безумовно є переможцем - його синтаксис простіше і легше, ніж у PHP [15].

В таблиці 2.2 наведено порівняння мов Python і Java.

Таблиця 2.2 - Порівняння мов Python і Java [15]

	Python	Java
Вивчення	Легше вчити	Складніше вчити
Кросплатформні застосунки	-	+
Сумісність з ОС	+	+
Мережеві застосунки	-	+

Java це статично типізована мова, так що змінні повинні оголошуватися із зазначенням типу. А Python - це мова з динамічною типізацією, в ньому вказівка типу для змінних не потрібна. Про переваги і недоліки динамічної та статичної типізації в різних мовах програмування постійно точаться дискусії [15].

Java дає можливість розробляти кросплатформні застосунки, але і Python сумісний з багатьма операційними системами. Втім, Java має одну суттєву перевагу: можливість створювати мережеві застосунки [15].

Якщо говорити про складність мови, Java набагато складніше Python [15].

В таблиці 2.3 наведено порівняння мов Python і C#.

Таблиця 2.3 - Порівняння мов Python і C# [15]

	Python	C#
Простота	+	-
Написання скриптів	В будь-якому середовищі	Тільки в IDE
Бібліотеки	Багато бібліотек	Мало бібліотек
Продуктивність	Низька	Висока

У Python простий синтаксис, так що його легко читати. Крім того, в ньому немає синтаксичних дужок, великої кількості модифікаторів, C-подібних конструкцій і т.д [15].

З іншого боку, C# досить схожий на Java і C++, що виражено в C-подібному синтаксисі. Синтаксис C# вимагає від розробника дотримуватися певних правил при написанні методів або успадкування класів [15].

Python простіше вчити, до того ж у нього багато бібліотек з відкритим кодом. Крім того, C# пропонує кращу продуктивність [15].

Нижче перераховані переваги мови програмування Python [15]:

- простота. Код, написаний на Python, легкий для читання і розуміння. Завдяки цьому в разі виникнення проблем розробник може зосередитися саме на них, не відволікаючись при цьому на складності мови;

- безкоштовність. Python це безкоштовна і відкрита технологія. Розробники можуть її поширювати, копіювати і змінювати. Як наслідок цього, Python швидко розвивається і має гарне співтовариство. Що стосується останнього, воно підтримує початківців розробників, ділячись з ними знаннями;

- сумісність. Python пропонує сумісність з різними платформами. Це позбавляє розробників, які пишуть на Python, від багатьох проблем, звичайних для інших мов;

- об'єктно-орієнтованість. Python підтримує як процедурно-орієнтоване, так і об'єктно-орієнтоване програмування (ООП). Перше дозволяє застосовувати різні частини коду повторно. А ООП дозволяє використовувати об'єкти, в основі яких лежать дані та функціонал;

- велика кількість бібліотек. Для Python створено дуже багато бібліотек. Завдяки цьому розробникам простіше працювати з документацією, базами даних і браузерами, проводити модульне тестування і т.д. Крім того, Python може використовуватися для самих різних завдань, наприклад, для розробки десктопних і вебзастосунків, складних обчислювальних систем, ігор, інтернету речей і багато чого іншого;

- код на Python добре читається і зрозумілий навіть розробникам, які ніколи не працювали з цією мовою. В результаті співтовариство Python

безперервно зростає. Крім того, ця мова переважає в сфері науки, так що серед його користувачів багато професорів і вчених;

- в даний час Python масово використовується в фінансових технологіях, науці про дані, машинному навчанні. В результаті такі гіганти як Google, Yandex і Dropbox підтримують цю технологію і створюють з її допомогою великі проєкти.

З огляду на порівняльну характеристику мов програмування та переваги розробки мовою Python прийнято рішення про використання останньої при розробці боту.

2.2 Порівняння платформ

Найбільш розповсюдженими месенджерами наразі є Telegram та Viber. Нижче у таблиці 2.4 наведено порівняння можливостей Viber і Telegram.

Таблиця 2.4 - Порівняння можливостей Viber і Telegram [16]

Можливості месенджерів	Viber	Telegram
Відправлення повідомлень і синхронізація з усіма пристроями	+	+
Застосунок для робочого столу	+	+
Використання ніка	-	+
Підключення декількох акаунтів	-	+
Створення групового чату	+	+
Розширення групового чату (для участі більшої кількості людей)	-	+
Можливість звертання до обраного співрозмовнику через "@"	-	+
Створення каналу	+	+
Хмарне зберігання даних	-	+
Безпека	+	++
Використання ботів	+	+

В таблиці 2.5 наведено особливості роботи Viber і Telegram.

Таблиця 2.5 - Особливості роботи Viber і Telegram [16]

Особливості	Viber	Telegram
1	2	3
Основна концепція	Чат-бот для оптимізації бізнесу та підтримки при взаємодії з публічними акаунтами	Інтеграція месенджера з будь-якими сервісами, виконання найрізноманітніших завдань в межах Telegram
Реалізація	Програми через інтерфейс Viber API відповідають користувачам, надаючи зокрема різні бізнес пропозиції і пов'язану інформацію	Особливі Telegram-акаунти без номерів телефону, контрольовані програмами і звертаються до користувачів в рамках можливостей Bot API
Атрибути ботів	Параметри рівня публічного аккаунта (бренд, опис, місцезнаходження)	Ім'я, фото, опис і адреса
Обмеження на створення	Всі вимоги до ботів і вимоги до публічного аккаунту	Практично відсутні
Підтримувані типи вмісту	Текст, зображення, відео, контакт, URL, каруселі, графічний зміст, а також місцезнаходження	Всі (що підтримуються Telegram), зокрема місцезнаходження та номер, опитування
Можливість монетизації	На рівні публічних акаунтів	Підтримується
Комунікація в режимі чату	Так, за допомогою повідомлень і гнучких клавіатур	Так, за допомогою повідомлень, команд і клавіатур
Комунікація через вбудований режим	Обмежена реалізація через розширення	Підтримується (відправка запитів і отримання результатів відповіді)
Взаємодія з повідомленнями ботів	Обмежена підтримка (через кнопки)	Через вбудовані клавіатури з підтримкою оновлення вмісту динамічних повідомлень
Додавання ботів до інших чатів	Не підтримується	Підтримується (для груп / каналів)

Продовження таблиці 2.5

1	2	3
Зупинка ботів	На рівні відписки від публічного аккаунта	Підтримується
Дозвіл на бесіду	Не підтримується	Підтримується (для відправки повідомлень)
Режим приватності	Не підтримується	Підтримується
Deep Linking	Підтримується (з орієнтацією на публічні чати)	Підтримується
Швидка веб-авторизація	Не підтримується	Підтримується (за посиланням)
Створення ботів	Через Viber REST API	Через бота @BotFather і Bot API
Варіанти використання	Отримання новин, інформації, реклами, зв'язок з публічним аккаунтом, підтримка	Отримання новин, інформації, повідомлень, покупка товарів і послуг, створення нових інструментів, ігор, соціальних сервісів, оптимізація користувацького досвіду (IV, попередній перегляд посилань), навчання, автоматизація, зворотний зв'язок, авторизація, вікторини

З дня свого заснування месенджер Telegram стабільно піднімається в рейтингах найпопулярніших застосунків. Про це в понеділок, 8 лютого 2021 року, повідомив засновник месенджера Павло Дуров. «З моменту запуску в 2013 році база користувачів Telegram щорічно зростала більш ніж на 40%», - написав Дуров у своєму каналі. Він зазначив, що Telegram став самим завантажуваним мобільним застосунком в світі в січні 2021 року [1].

У квітні 2016 було анонсовано перше масштабне оновлення – Bot Platform 2.0, підготовлене з урахуванням побажань розробників і потреб користувачів, яке значно розширює можливості роботів. Додана можливість обміну музикою і відео, варіанти швидких відповідей, ідентифікація місця розташування користувача, глибока інтеграція з іншими службами на основі телефонних номерів (при авторизації) і ряд інших корисних можливостей [1].

З огляду на статистичні дані про поточну популярність месенджеру Telegram та можливості і особливості двох месенджерів прийнято рішення про використання Telegram в якості платформи для розробки програмного продукту.

2.3 Опитування зацікавлених осіб

Для виявлення потреб користувачів було виконано опитування цільової аудиторії користувачів – співробітників компанії. Виявлено, що найчастіше виникає потреба в доступі до корпоративного ПЗ та технічні проблеми з технікою. Більше 70% співробітників використовує месенджер Telegram або знайомі з ним.

Враховано побажання модератора – співробітника технічної підтримки підприємства, що виконує обробку заявок, а саме структуризація меню користувача, точне розбиття на розділи, з якими пов'язана проблема користувача, наявність термінів на обробку заявки.

Також враховано програмні можливості корпоративного серверного простору.

Відповідно до результатів опитування було сформовано вимоги до функціональних характеристик, вимоги до надійності, вимоги до складу і параметрів технічних засобів, вимоги до інформаційної та програмної сумісності. Надалі Telegram-бот іменується системою (система взаємодій окремих функцій та методів).

Вимоги до функціональних характеристик:

- система має містити обробку стандартних для класу ботів команд: /start, /help;
- система повинна надавати доступ до реєстрації при першому запуску;
- система повинна надавати можливість повернутися до головного меню зареєстрованому користувачу;

- система повинна містити ієрархію розділів меню для визначення характеру проблеми користувача;
- система має надавати можливість залишення заявки – текстові повідомлення, фото, документи;
- система має містити окремі команди для модератора;
- система повинна містити авторизацію модератора;
- система має надавати можливість подати заявку на реєстрацію для користувача, для модератора – можливість схвалити та відхилити заявку на реєстрацію;
- система має містити обробку системних помилок, повідомляти про відтворення останніх адміністратору;
- система має надавати можливість відповіді на заявки модератору, адміністратору;
- система має зберігати дані про звернення у базі даних.

Вимоги до надійності:

- система повинна інформувати адміністратора про системні помилки;
- система має відповідати користувачу при введенні невірних даних;
- система повинна виконуватись в якості служби на сервері для забезпечення безперервної роботи;
- система повинна відповідати підтвердженням на дії користувача.

Вимоги до складу і параметрів технічних засобів — для функціонування системи необхідний сервер:

- налаштовано доступ до мережі Internet;
- наявність ПЗ для запуску системи в якості служби.

Вимоги до інформаційної та програмної сумісності — для функціонування системи необхідна наявність Python-інтерпретатора на сервері, для користувача виконується доступ на будь-якому пристрої, що підтримує запуск Telegram. Мова інтерфейсу — українська.

2.4 Архітектура програми

Для реалізації програмного продукту обрано структурний підхід, який полягає в декомпозиції системи –розбитті її на функції, що автоматизуються. Система розбивається на функціональні підсистеми, які в свою чергу діляться на підфункції і т. д. Таким чином за програмну реалізацію функцій відповідає окремий блок коду. На рисунку 2.1 наведено умовний поділ загального обсягу програмного коду на блоки реалізації інтерфейсу користувача, обробки функцій, реалізації запитів до бази даних (БД), взаємодії з таблицями БД.

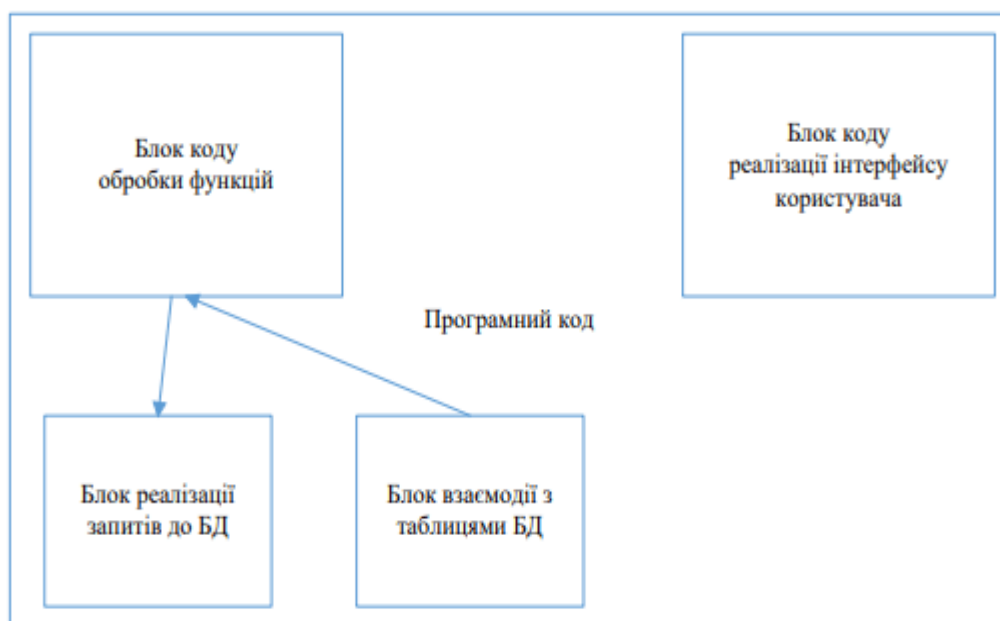


Рисунок 2.1 – Умовний поділ програмного коду на блоки

Також на рисунку 2.2 наведено концептуальну схему взаємодії програмного продукту з таблицями БД для зберігання даних, таблицями іншого ПП для коректного відтворення заявок у сторонньому ПП, з відділом кадрів підприємства.

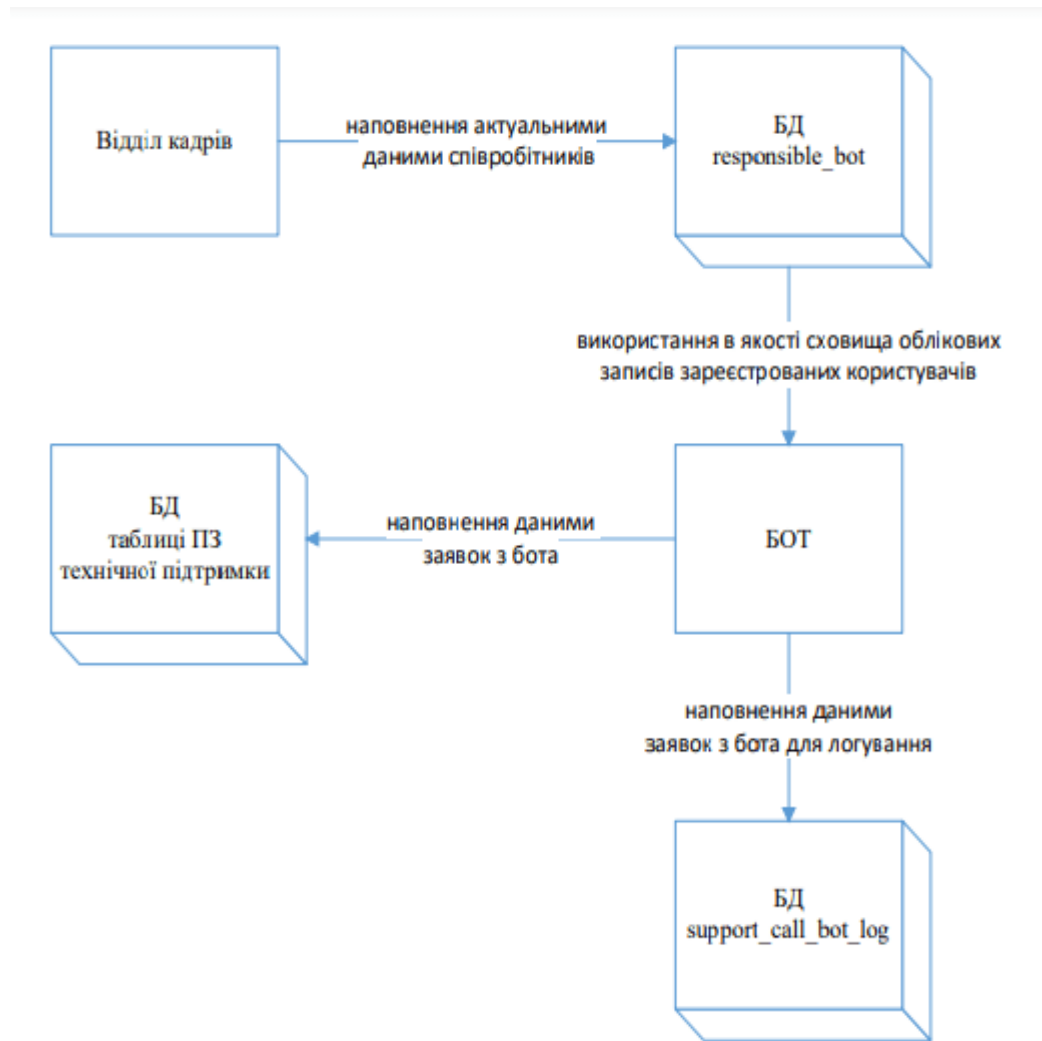


Рисунок 2.2 - Концептуальна схема взаємодії програмного продукту

Програмний продукт взаємодіє з таблицями БД, а саме наповнює таблицю support_call_bot_log та таблиці стороннього ПП даними про заявки, які надійшли. На основі роботи відділу кадрів даними наповнюється таблиця responsible_bot, яку в якості сховища облікових записів користувачів використовує бот.

Виділено три типи користувачів: користувач зареєстрований (користувач незареєстрований), модератор, адміністратор.

Відповідно до вимог всіх типів користувачів прийнято рішення про програмну реалізацію функцій, use-case діаграми яких наведені на рисунках 2.3-2.4.

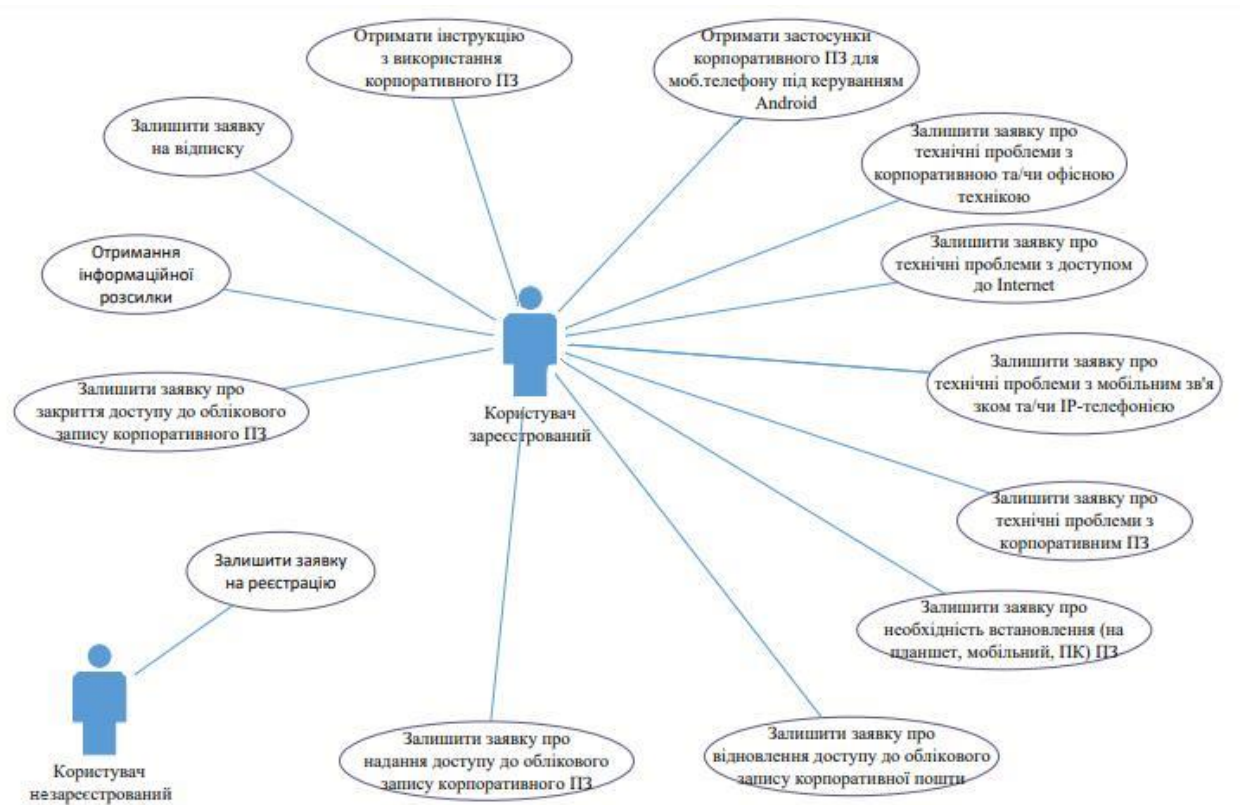


Рисунок 2.3 – Use-case діаграма функцій користувача

Прийнято рішення про програмну реалізацію наступних можливостей для користувача незареєстрованого: залишення заявки на реєстрацію.

Прийнято рішення про програмну реалізацію наступних можливостей для користувача зареєстрованого:

- отримання інструкції з використання корпоративного ПЗ;
- отримання застосунків корпоративного ПЗ для телефону під керуванням Android;
- залишення заявки про технічні проблеми з корпоративною та/чи офісною технікою;
- залишення заявки про технічні проблеми з доступом до Internet;
- залишення заявки про технічні проблеми з мобільним зв'язком та/чи IP-телефонією;
- залишення заявки про технічні проблеми з корпоративним ПЗ;

- залишення заявки про необхідність встановлення (на планшет, мобільний, персональний комп'ютер) ПЗ;
- залишення заявки про відновлення доступу до облікового запису корпоративної пошти;
- залишення заявки про надання доступу до облікового запису корпоративного ПЗ;
- отримання інформаційної розсилки;
- залишення заявки на відписку.

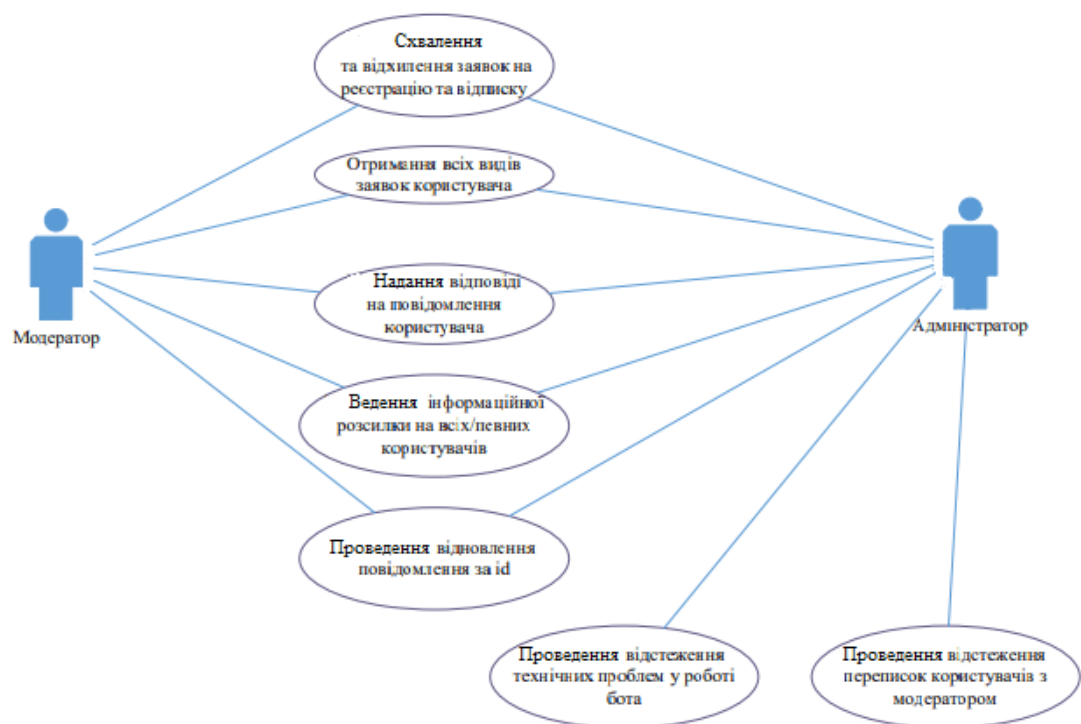


Рисунок 2.4 - Use-case діаграма функцій модератора та адміністратора

Прийнято рішення про програмну реалізацію наступних можливостей для модератора:

- можливість схвалення та відхилення заявок на реєстрацію та відписку;
- отримання всіх видів заявок користувача;
- можливість відповіді на повідомлення користувача;
- можливість інформаційної розсилки на всіх/певних користувачів;

- можливість відновлення повідомлення за id.

Прийнято рішення про програмну реалізацію наступних можливостей для адміністратора:

- можливість схвалення та відхилення заявок на реєстрацію та відписку;
- отримання всіх видів заявок користувача;
- можливість відповіді на повідомлення користувача;
- можливість інформаційної розсилки на всіх/певних користувачів;
- можливість відновлення повідомлення за id;
- можливість відстеження технічних проблем у роботі бота;
- можливість відстеження переписок користувачів з модератором.

Прийнято рішення про розробку структури меню користувача відповідно до схеми, яку наведено на рисунку 2.5.

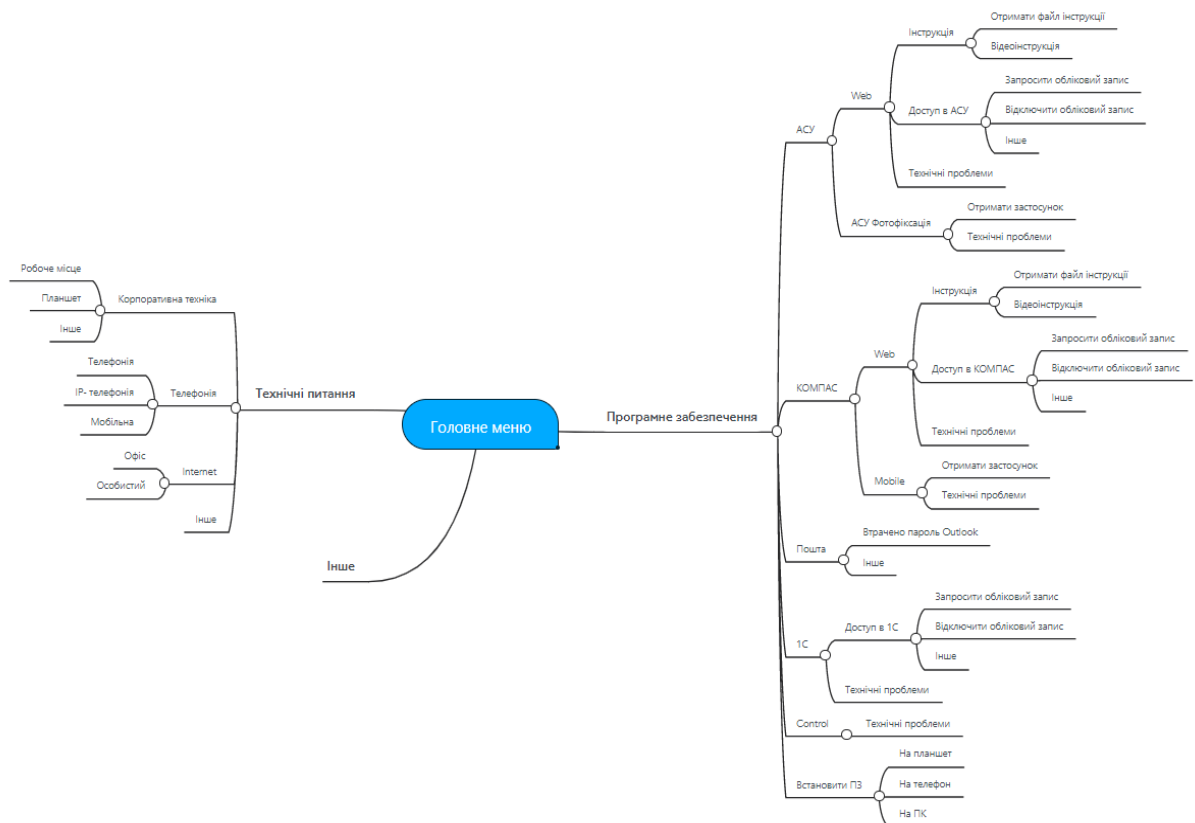


Рисунок 2.5 – Структура меню користувача

Відповідно до потреб у даних для реєстрації розроблено BPMN 2.0 діаграму реєстрації користувача, яку наведено на рисунку 2.6.

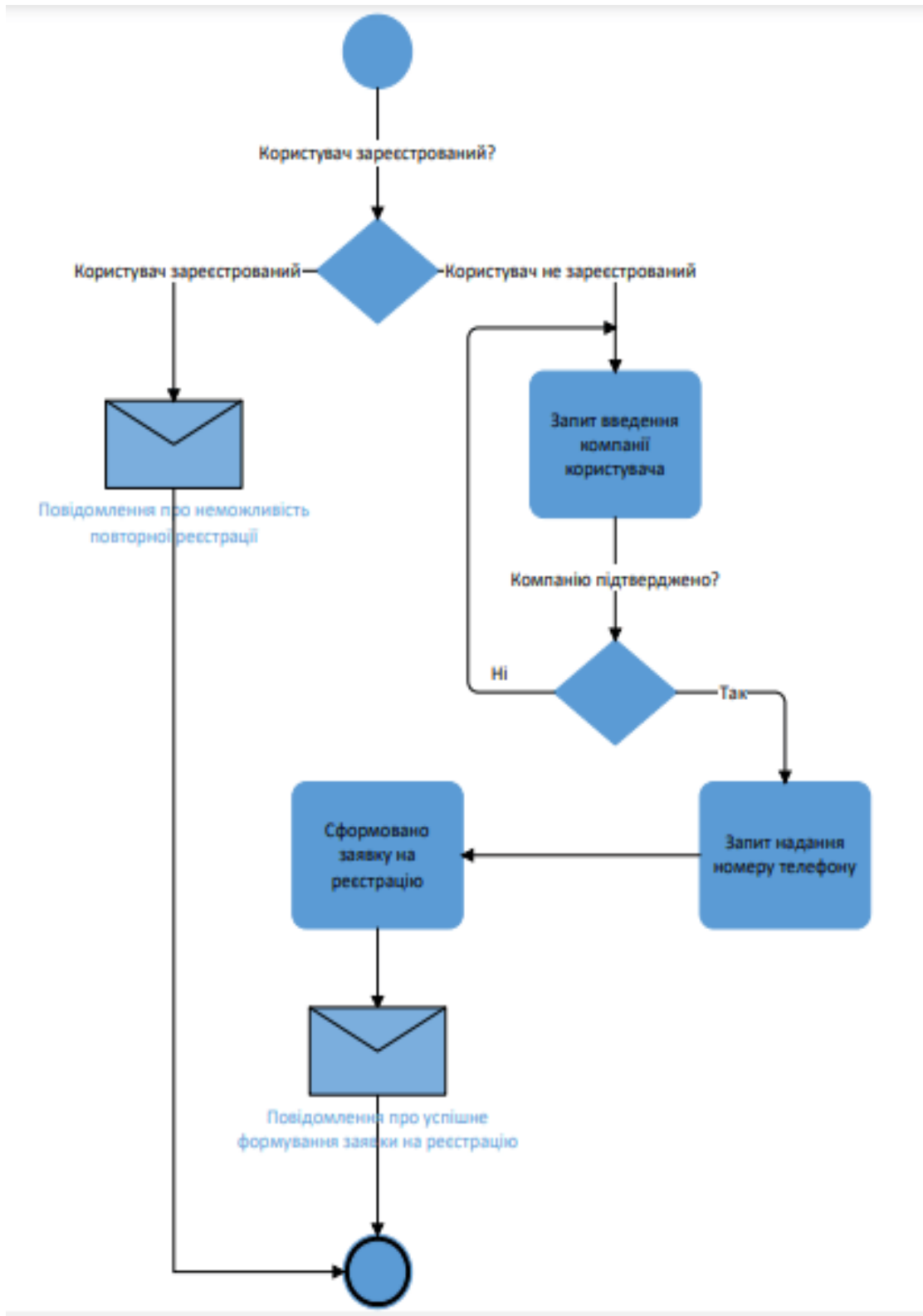


Рисунок 2.6 – Діаграма реєстрації користувача

Процес реєстрації користувача.

Крок 1. Користувач переходить до реєстрації.

Крок 2. Система пропонує обрати компанію, до якої належить користувач.

Крок 3. Користувач обирає компанію.

Крок 4. Система пропонує підтвердити вибір компанії.

Крок 5. Користувач підтверджує вибір компанії.

Крок 6. Система пропонує надати номер телефону користувача.

Крок 7. Користувач надає номер телефону.

Крок 8. Система формує заявку на реєстрацію.

Крок 9. Користувач отримує повідомлення про успішне формування заявки на реєстрацію.

Якщо користувач вже авторизований, система надає відповідне повідомлення при вже виконану реєстрацію.

Якщо компанію користувача не підтверджено, система надає можливість повторного вибору компанії користувача.

Якщо користувач не надає номер телефону, реєстрація неможлива.

Також розроблено BPMN 2.0 діаграму подій для загального випадку залишення заявки про проблему в боті, наведено на рисунку 2.7. Доступ до функції залишення заявки мають лише зареєстровані користувачі.

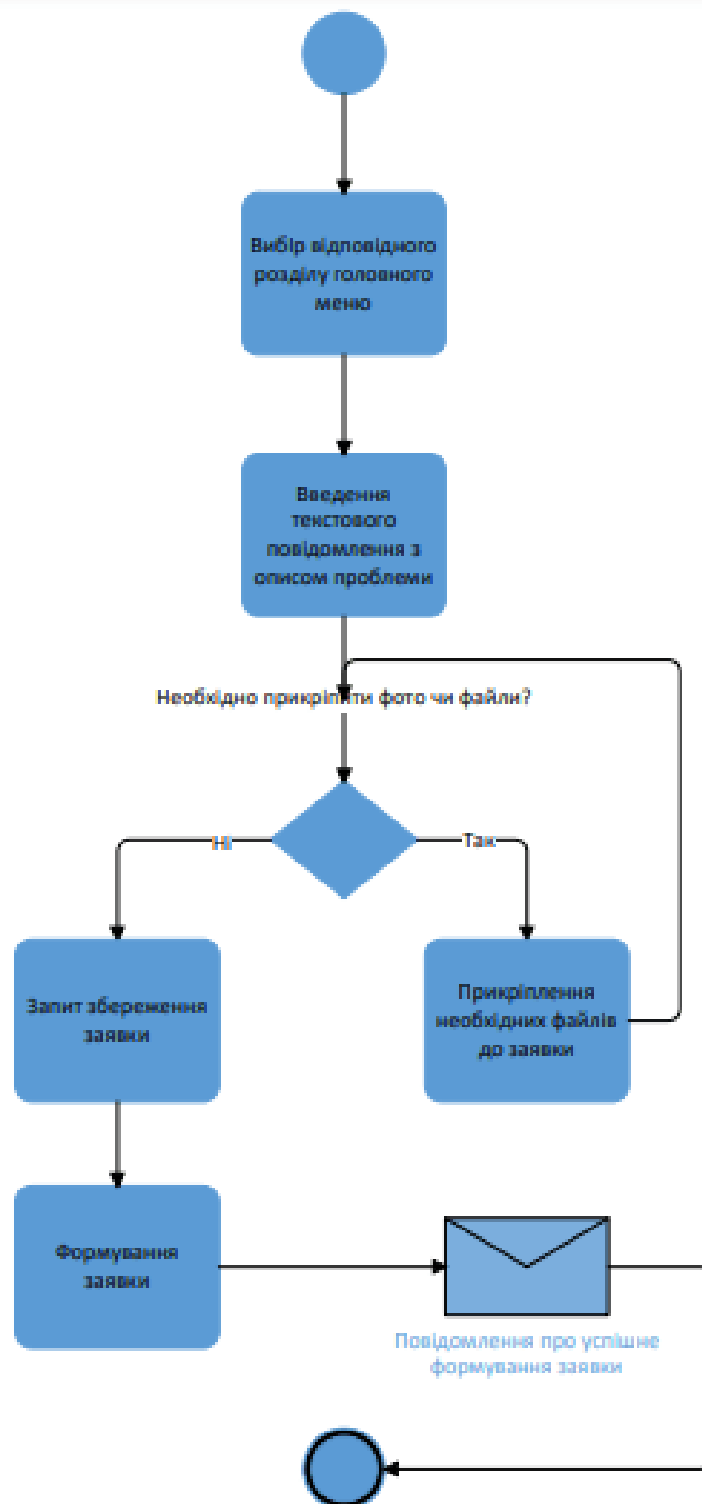


Рисунок 2.7 – Діаграма залишення заявки

Процес загального випадку залишення заявки.

Крок 1. Користувач переходить до розділу, який відповідає його проблемі.

Крок 2. Система пропонує описати текстом проблему користувача.

Крок 3. Користувач вводить текстове повідомлення з описом проблеми.

Крок 4. Система пропонує прикріпити до заявки фото чи файли.

Крок 5. Користувач прикріплює фото та файли.

Крок 6. Система робить запит на збереження заявки.

Крок 7. Користувач підтверджує збереження заявки.

Крок 8. Система формує заявку про проблему.

Крок 9. Користувач отримує повідомлення про успішне формування заявки про проблему.

Якщо користувач не має потреби у прикріпленні файлів чи фото, він переходить до підтвердження збереження заявки.

2.5 Висновки з розділу 2

У даному розділі проведено аналіз мов програмування. З огляду на порівняльну характеристику мов програмування, наведену у підрозділі 2.1, та переваги розробки мовою Python прийнято рішення про використання останньої при розробці боту.

Проведено порівняння платформ для розробки програмного продукту. З огляду на статистичні дані про поточну популярність месенджера Telegram та можливості і особливості двох месенджерів прийнято рішення про використання Telegram в якості платформи для розробки програмного продукту.

У підрозділі 2.3 наведено результати опитування зацікавлених осіб.

Для реалізації обрано структурний підхід, розроблено концептуальну схему взаємодії програмного продукту, use-case діаграми функцій для всіх типів користувачів, структуру меню користувача, BPMN 2.0 діаграми реєстрації користувача та залишення заявки.

З РОЗРОБКА ПРОГРАМИ

Для взаємодії з API Telegram боту надається токен доступу. Таким чином однозначно ідентифікується програмне керування певним ботом – сутністю месенджера.

Прийнято рішення зберігати токен доступу окремо від основного програмного коду з міркувань безпеки. Токен доступу зберігається у файлі config.py, основний програмний код зберігається у файлі main.py. До проєкту підключено додаткові бібліотеки:

- telebot – взаємодія з API Telegram;
- datetime – робота з датою та часом;
- mysql.connector – робота з базою даних та обробка помилок;
- os.path – робота з файловою системою пристрою;
- json – робота з типом даних json;
- pygame – затримка у часі.

На рисунку 3.1 наведено структуру проєкту та підключення бібліотек.

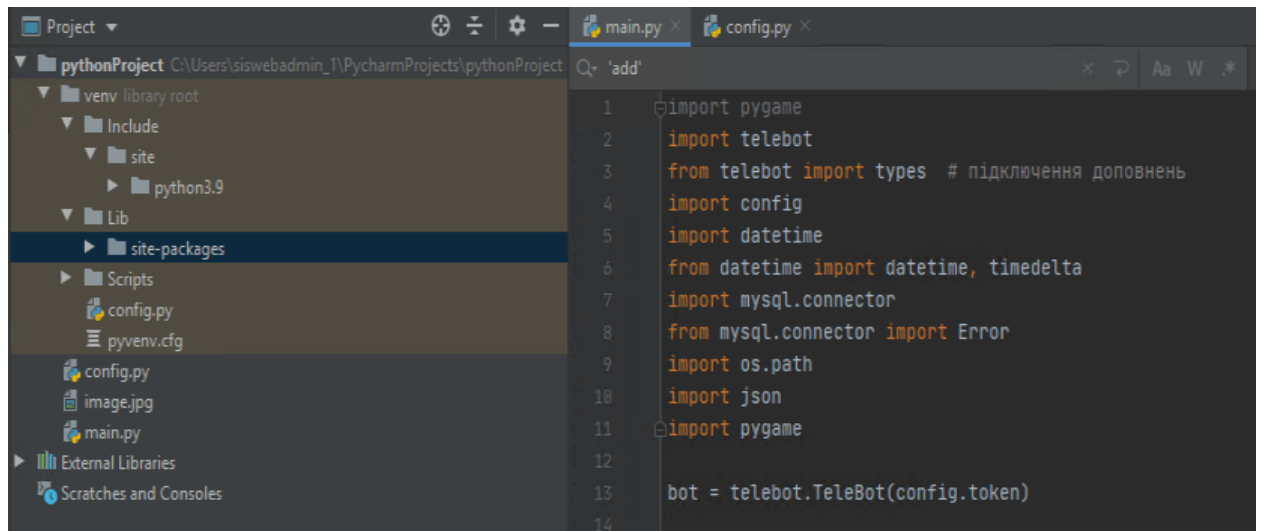


Рисунок 3.1 – Структура проєкту

Відповідно до потреб у зберіганні даних (логуванні) прийнято рішення про розробку структури таблиці support_call_bot_log, яку наведено на рисунку 3.2.

Name: support_call_bot_log		Database: support					
Engine: InnoDB							
☐	Name	Data Type	Unsigned	Auto Increment	Not Null	Default	Collation
☒	id	INT(11)	☐	☒	☒		
☐	telegram_id	INT(11)	☐	☐	☐	NULL	
☐	message_id	INT(11)	☐	☐	☐	NULL	
☐	topic_call	INT(11)	☐	☐	☐	NULL	
☐	date_telegram	INT(11)	☐	☐	☐	NULL	
☐	status	INT(11)	☐	☐	☐	NULL	
☐	once	INT(11)	☐	☐	☐	NULL	
☐	child	INT(11)	☐	☐	☐	NULL	
☐	mod_telegram_id	INT(11)	☐	☐	☐	NULL	
☐	mod_message_id	INT(11)	☐	☐	☐	NULL	
☐	support_call_id	VARCHAR(255)	☐	☐	☐	NULL	utf8_general_ci
☐			☐	☐	☐		

Рисунок 3.2 – Структура таблиці support_call_bot_log

Таблиця support_call_bot_log містить поля:

- telegram_id - автор повідомлення (заявки);
- message_id - унікальне в межах одного чату id повідомлення;
- topic_call - номер заявки в технічну підтримку;
- status – статус поточної заявки;
- once - ознака того, що модератор вже одного разу відповідав на це повідомлення;
- child - null якщо батьківська заявка, 1 якщо це продовження спілкування в рамках поточної заявки;
- mod_telegram_id - телеграм id модератора, який відповів на це повідомлення;
- mod_message_id - телеграм id повідомлення, яким відповів модератор;
- support_call_id - id заявки з таблиці технічної підтримки іншого ПП;
- date_telegram - поле для зберігання дати відправлення повідомлення, збігаються у користувача і модератора.

3.1 Методи та функції

Склад програмного коду можна умовно поділити на реалізацію інтерфейсу користувача та блок обробки функцій, який взаємодіє з блоком реалізації запитів до БД та блоком взаємодії з таблицями БД.

Нижче перераховані функції та методи блоку реалізації інтерфейсу користувача:

- `start_command(message)` – метод, що відправляє користувачу inline-клавіатуру основного меню користувача (доступний як зареєстрованому, так і незареєстрованому);
- `method_main_menu_user(message)` – метод, що відправляє користувачу inline-клавіатуру головного меню користувача (доступний зареєстрованому користувачу);
- `menu_21_user(message)` – метод, що перевіряє авторизованість користувача, у разі зареєстрованості користувача виконується виклик функції `method_menu_21_user(message)`;
- `method_menu_21_user(message)` – метод, що відправляє користувачу inline-клавіатуру меню підрозділу «Програмне забезпечення» (доступний зареєстрованому користувачу);
- `menu_22_user(message)` – метод, що перевіряє авторизованість користувача, у разі зареєстрованості користувача виконується виклик функції `method_menu_22_user(message)`;
- `method_menu_22_user(message)` – метод, що відправляє користувачу inline-клавіатуру меню підрозділу «Технічні питання» (доступний зареєстрованому користувачу);
- `menu_211_user(message)` – метод, що перевіряє авторизованість користувача, у разі зареєстрованості користувача виконується виклик функції `method_menu_211_user(message)`;

- `method_menu_211_user(message)` - метод, що відправляє користувачу inline-клавіатуру меню підрозділу «АСУ» (доступний зареєстрованому користувачу);

- `menu_212_user(message)` - метод, що перевіряє авторизованість користувача, у разі зареєстрованості користувача виконується виклик функції `method_menu_212_user(message)`;

- `method_menu_212_user(message)` - метод, що відправляє користувачу inline-клавіатуру меню підрозділу «КОМПАС» (доступний зареєстрованому користувачу);

- `menu_213_user(message)` - метод, що перевіряє авторизованість користувача, у разі зареєстрованості користувача виконується виклик функції `method_menu_213_user(message)`;

- `method_menu_213_user(message)` - метод, що відправляє користувачу inline-клавіатуру меню підрозділу «Пошта» (доступний зареєстрованому користувачу);

- `menu_217_user(message)` - метод, що перевіряє авторизованість користувача, у разі зареєстрованості користувача виконується виклик функції `method_menu_217_user(message)`;

- `method_menu_217_user(message)` - метод, що відправляє користувачу inline-клавіатуру меню підрозділу «Встановити ПЗ» (доступний зареєстрованому користувачу);

- `menu_214_user(message)` - метод, що перевіряє авторизованість користувача, у разі зареєстрованості користувача виконується виклик функції `method_menu_214_user(message)`;

- `method_menu_214_user(message)` - метод, що відправляє користувачу inline-клавіатуру меню підрозділу «ІС» (доступний зареєстрованому користувачу);

- `menu_215_user(message)` - метод, що перевіряє авторизованість користувача, у разі зареєстрованості користувача виконується виклик функції `method_menu_215_user(message)`;

- `method_menu_215_user(message)` - метод, що відправляє користувачу inline-клавіатуру меню підрозділу «Control» (доступний зареєстрованому користувачу);

- `menu_221_user(message)` - метод, що перевіряє авторизованість користувача, у разі зареєстрованості користувача виконується виклик функції `method_menu_221_user(message)`;

- `method_menu_221_user(message)` - метод, що відправляє користувачу inline-клавіатуру меню підрозділу «Корпоративна техніка» (доступний зареєстрованому користувачу);

- `menu_222_user(message)` - метод, що перевіряє авторизованість користувача, у разі зареєстрованості користувача виконується виклик функції `method_menu_222_user(message)`;

- `method_menu_222_user(message)` - метод, що відправляє користувачу inline-клавіатуру меню підрозділу «Телефонія» (доступний зареєстрованому користувачу);

- `menu_223_user(message)` - метод, що перевіряє авторизованість користувача, у разі зареєстрованості користувача виконується виклик функції `method_menu_223_user(message)`;

- `method_menu_223_user(message)` - метод, що відправляє користувачу inline-клавіатуру меню підрозділу «Internet» (доступний зареєстрованому користувачу);

- `menu_2111_user(message)` - метод, що перевіряє авторизованість користувача, у разі зареєстрованості користувача виконується виклик функції `method_menu_2111_user(message)`;

- `method_menu_2111_user(message)` - метод, що відправляє користувачу inline-клавіатуру меню підрозділу «АСУ Web» (доступний зареєстрованому користувачу);

- `menu_2112_user(message)` - метод, що перевіряє авторизованість користувача, у разі зареєстрованості користувача виконується виклик функції `method_menu_2112_user(message)`;

- `method_menu_2112_user(message)` - метод, що відправляє користувачу inline-клавіатуру меню підрозділу «АСУ Фотофіксація» (доступний зареєстрованому користувачу);
- `menu_2121_user(message)` - метод, що перевіряє авторизованість користувача, у разі зареєстрованості користувача виконується виклик функції `method_menu_2121_user(message)`;
- `method_menu_2121_user(message)` - метод, що відправляє користувачу inline-клавіатуру меню підрозділу «КОМПАС Web» (доступний зареєстрованому користувачу);
- `menu_2122_user(message)` - метод, що перевіряє авторизованість користувача, у разі зареєстрованості користувача виконується виклик функції `method_menu_2122_user(message)`;
- `method_menu_2122_user(message)` - метод, що відправляє користувачу inline-клавіатуру меню підрозділу «КОМПАС Mobile» (доступний зареєстрованому користувачу);
- `menu_2141_user(message)` - метод, що перевіряє авторизованість користувача, у разі зареєстрованості користувача виконується виклик функції `method_menu_2141_user(message)`;
- `method_menu_2141_user(message)` - метод, що відправляє користувачу inline-клавіатуру меню підрозділу «Доступ до 1С» (доступний зареєстрованому користувачу);
- `menu_21111_user(message)` - метод, що перевіряє авторизованість користувача, у разі зареєстрованості користувача виконується виклик функції `method_menu_21111_user(message)`;
- `method_menu_21111_user(message)` - метод, що відправляє користувачу inline-клавіатуру меню підрозділу «Інструкція» (доступний зареєстрованому користувачу);
- `menu_21112_user(message)` - метод, що перевіряє авторизованість користувача, у разі зареєстрованості користувача виконується виклик функції `method_menu_21112_user(message)`;

- `method_menu_21112_user(message)` - метод, що відправляє користувачу inline-клавіатуру меню підрозділу «Доступ до АСУ» (доступний зареєстрованому користувачу);

- `menu_21211_user(message)` - метод, що перевіряє авторизованість користувача, у разі зареєстрованості користувача виконується виклик функції `method_menu_21211_user(message)`;

- `method_menu_21211_user(message)` - метод, що відправляє користувачу inline-клавіатуру меню підрозділу «Інструкція» (доступний зареєстрованому користувачу);

- `menu_21212_user(message)` - метод, що перевіряє авторизованість користувача, у разі зареєстрованості користувача виконується виклик функції `method_menu_21212_user(message)`;

- `method_menu_21212_user(message)` - метод, що відправляє користувачу inline-клавіатуру меню підрозділу «Доступ до КОМПАС» (доступний зареєстрованому користувачу);

- `help(message)` – метод виконує обробку команди «/help» користувача;

- `method_main_menu_replay_user(message)` – метод закріплює кнопку головного меню в області клавіатури для користувача;

- `help_mod(message)` – метод виконує обробку команди «/help_mod» модератора;

- `start(message)` - метод виконує обробку команди «/start» користувача.

Нижче перераховані функції та методи блоку реалізації запитів:

- `close_connection(connection)` – метод закриття підключення до БД;

- `create_connection(host_name, user_name, user_password, db_name)` – функція створення підключення, повертає відповідний екземпляр;

- `execute_read_query(connection, query)` - метод виконання запиту на читання до БД;

- `execute_query(connection, query)` – метод виконання запиту на вставку, оновлення БД.

Нижче перераховані функції та методи блоку взаємодії з таблицями БД:

- `get_once_call_from_buffer_db(telegram_id, message_id, message_date)` - метод отримання статусу заявки (`once`) з проміжної таблиці `support_call_bot_log`;
- `update_status_call_to_support_db(topic_call)` - метод оновлення статусу заявки (`status`) в таблицях `support_call` та `support_topic`, статус 4 - закрита заявка;
- `update_status_call_to_buffer_db(telegram_id, topic_call)` - метод оновлення статусу заявки (`status`) в таблиці `support_call_bot_log`, статус 4 - закрита заявка;
- `notificate_close_call_for_mod (topic_call, user_chat)` - метод для повідомлення модератора про закриття заявки користувачем;
- `insert_comment_to_db(telegram_id, message_date, telegram_id_mod, message_id_mod)` - метод вставки відповіді модератора в таблиці `support_call`, `support_comment`, `telegram_id` - призначений для користувача `id` - від кого повідомлення в бот, `message_date` - дата повідомлення користувача, що запитали, `telegram_id_mod` - модераторський `id` - хто відповів на повідомлення в бот, `message_id_mod` - повідомлення модератора, що відповіли;
- `update_once_call_to_buffer_db(telegram_id, message_id, message_date, telegram_id_mod, message_id_mod)` - метод оновлення статусу заявки (`once`) в проміжній таблиці `support_call_bot_log`, `telegram_id` - користувацький `id` - від кого повідомлення в бот, `message_id` - `id` повідомлення користувача, що запитали, `message_date` - дата пересилання повідомлення щоб впізнати його в базі, `telegram_id_mod` - модераторське `id` - хто відповів на повідомлення в бот, `message_id_mod` - повідомлення модератора, що відповіли;
- `insert_call_to_buffer_db(telegram_id, topic_call, message_id, message_date, child_flag, status)` - метод вставки заявки в проміжну таблицю `support_call_bot_log`, коли користувач додав заявку;

- `insert_call_to_db_comm(topic, message)` - метод вставки заявки (вторинної) в таблиці техпідтримки;
- `insert_call_to_db(user, now, title, text)` - метод вставки заявки (первинної) в таблиці техпідтримки;
- `get_user_not_str_info_from_db_by_tel_id(message)` - повертає інформацію про користувача за телеграм id з бази структурою даних;
- `get_user_info_from_db_by_tel_id(message)` - повертає інформацію про користувача за телеграм id з бази рядком.

Нижче перераховані функції та методи блоку обробки функцій бота:

- `is_moderator(user)` – функція перевірки поточного користувача на авторизованість обліковим записом модератора;
- `get_user_problem(message)` - метод запиту введення проблеми користувача;
- `get_problem_user(message)` - метод обробки введеної користувачем проблеми;
- `default_command(message)` – метод обробки введення користувачем аудіофайлів, відеофайлів, місцезнаходження, стікерів, голосових повідомлень, вкладень до повідомлення;
- `get_message_user_to_forward_mod(message)` - метод ззовні викликається - обгортка для рекурсивної відправки повідомлень користувача після відповіді модератора на заявку, якщо користувач виявив бажання продовжити листування;
- `method_get_message_user_to_forward_mod(message)` - метод рекурсивної відправки повідомлень користувача після відповіді модератора на заявку, якщо користувач виявив бажання продовжити листування;
- `send_by_id(message)` – метод обробки запиту `telegram_id` користувача при виконанні модератором адресної розсилки;
- `get_id_for_send(message)` – метод обробки запиту повідомлення для адресної розсилки модератором;

- `get_text_for_send(message)` - метод обробки відправлення користувачу повідомлення в адресній розсилці модератора;
- `answer_by_id(message)` – метод обробки запиту `telegram_id` користувача при виконанні модератором відповіді за `id`;
- `get_id_for_answer(message)` – метод обробки запиту повідомлення для відповіді модератора за `id`;
- `get_text_for_answer(message)` - метод обробки відправлення користувачу повідомлення у відповіді модератора за `id`;
- `process_start(message)` – метод обробки запиту вибору повідомлення, на яке відповідає модератор;
- `send_continue_close_call(chat_id, flagok)` – метод обробки відправлення користувачу клавіатур для продовження спілкування або закриття заявки в залежності від статусу заявки;
- `process_mind(message)` – метод обробки відправлення повідомлення користувачу під час відповіді модератора на повідомлення, викликаються методи зберігання заявки в БД;
- `get_user_need_install_pc(message)` – метод обробки запиту переліку ПЗ, яке необхідно встановити користувачу;
- `get_po_need_install_user(message)` – метод обробки формування та зберігання заявки про необхідне встановлення ПЗ;
- `method_get_account(message)` – метод обробки запиту на відключення облікового запису користувача у корпоративному ПЗ;
- `method_get_account1(message)` – метод обробки запиту на надання доступу до облікового запису користувача у корпоративному ПЗ;
- `get_user_position(message)` – метод запиту введення посади користувача при запиті пароля від пошти;
- `send_mp4_file(message, flag_mp4)` – метод обробки відправлення користувачу відео інструкції до корпоративного ПЗ;
- `send_pdf_file(message, flag_pdf)` - метод обробки відправлення користувачу файлу інструкції до корпоративного ПЗ;

- `send_kompas_apk_url(message)` - метод обробки відправки apk-файлу корпоративного ПЗ з перевіркою на авторизацію;
- `send_kompas_apk_url_method(message)` - метод відправки кнопки з посиланням на apk-файл корпоративного ПЗ на гугл-диску;
- `send_asu_apk_url(message)` - метод обробки відправки apk-файлу корпоративного ПЗ з перевіркою на авторизацію;
- `send_asu_apk_url_method(message)` - метод відправки кнопки з посиланням на apk-файл корпоративного ПЗ на гугл-диску;
- `append_photo_doc_to_list(type1, name1)` – метод обробки додавання відомостей про вкладені файли та фото до відповідного поля запису про заявку до БД;
- `main_menu_user(message)` – метод обробки команди «Головне меню»;
- `method_save_photo_in_comm(message)` - метод збереження фото і повернення рядка для вставки в коментар користувача в базу;
- `method_save_document_in_comm(message)` - метод збереження документу і повернення рядка для вставки в коментар користувача в базу;
- `handle_docs_photo(message)` – метод обробки повідомлень типу фото (photo);
- `handle_docs_photo(message)` - метод обробки повідомлень типу документ (document);
- `main_menu_user(message)` - метод обробки команди «/menu»;
- `get_id(message)` – метод обробки команди «/id», «/ID», «/iD», «/Id», повертає telegram_id користувача;
- `get_company_method(message)` – метод обробки запиту компанії користувача;
- `get_company(message)` – метод обробки команди «/registration», містить виклик функції `get_company_method(message)`;
- `confirm_company(call, message_user)` – метод обробки запиту підтвердження вибору компанії користувача;

- `external_error(message)` - метод обробки помилкового введення тексту, зовнішньо-викликається;
- `error_company_method()` - метод містить обробку введення тексту, містить виклик методу `external_error(message)`;
- `remote_company_markup(call, current_message)` – метод виконує видалення клавіатури після вибору компанії користувача, відправляє підтвердження вибору користувача;
- `remote_company_confirm_markup(call)` – метод виконує видалення клавіатури з підтвердженням компанії після вибору поточної;
- `remote_main_menu_markup(call, text)` – метод виконує видалення клавіатури з головним меню;
- `remote_start_markup(call)` – метод виконує видалення клавіатури з реєстрацією / відпискою користувача після вибору;
- `remote_reauth_confirm_markup(call)` – метод виконує видалення клавіатури з підтвердженням / запереченням виходу з облікового запису модератора;
- `remote_exit_confirm_markup(call)` – метод виконує видалення клавіатури з підтвердженням / запереченням виходу з облікового запису користувача;
- `callback_inline(call)` - обробник натискання на кнопки inline-клавіатур;
- `get_number_topic_from_text(text, first_symb, second_symb)` - метод для отримання номера заявки з повідомлення;
- `update_call_by_photo_doc()` – метод доповнює запис заявки в БД даними про вкладені фото / документи;
- `send_confirm_send_call()` – метод виконує розсилку підтвердження збереження заявки (після натискання на відповідну кнопку);
- `phone(message)` – метод виконує відправку повідомлення та кнопки для надання користувачем свого номеру телефона;

- `contact(message)` – обробник повідомлень типу контакт (`contact`), виконується обробка надання користувачем номеру телефону;
- `get_message_mod(message)` – метод виконує запит `id` користувача та повідомлення при виконанні функції відтворення певного повідомлення;
- `method_get_message_mod(message)` – метод обробки введених модератором `id` користувача і `id` повідомлення через пробіл для отримання вмісту повідомлення;
- `send_all_mod(message)` – метод запиту повідомлення, яке буде розіслано модератором на всіх користувачів;
- `get_password(message)` - обробка введення повідомлення модератора для загальної розсилки;
- `get_all_id_user_method()` – функція повертає `telegram_id` всіх зареєстрованих користувачів;
- `auth_mod(message)` – метод містить перевірку авторизації модератора, вимагає пароль у разі відсутності успішної авторизації;
- `reauth_mod(message)` – метод містить запит підтвердження виходу з облікового запису модератора;
- `get_password(message)` – метод містить перевірку паролю модератора, у разі успішності перевірки надає повідомлення з переліком команд модератора;
- `show_mod(message)` – метод надає модератору перелік всіх зареєстрованих користувачів;
- `exit_user_method(message)` – метод запиту підтвердження відправлення заявки на відписку користувача;
- `exit_user(message)` – метод обробки команди «/exit», містить виклик функції `exit_user_method(message)`;
- `add_user_mod(message)` – метод містить запит у модератора прізвища або номера телефону при додаванні нового співробітника;
- `method_show_people_from_db(message)` – метод повертає співробітників з бази за прізвищем;

- `get_sur_user_add_mod(message)` – метод містить обробку введення прізвища користувача і запиту id для додавання модератором користувача;
- `get_id_user_add_mod(message)` – метод містить обробку введення id користувача для додавання модератором користувача;
- `get_tel_id_user_add_mod(message)` – метод містить обробку введення telegram_id користувача для додавання модератором;
- `get_comp_user_add_mod(message)` - обробка введення компанії користувача для додавання модератором;
- `remove_user_mod(message)` - видалення користувача модератором після заявки;
- `get_id_user_remove_mod(message)` - обробка введення id користувача для видалення модератором;
- `fail_user_mod(message)` - видалення користувача модератором після заявки;
- `get_id_user_del_mod(message)` - обробка введення id користувача для видалення модератором.

3.2 Команди

Для користувача передбачена обробка наступних команд:

- «/start» - вітання;
- «/help» - отримання довідки по командам;
- «/registration» - реєстрація користувачів;
- «/exit» - видалення зареєстрованого користувача з бота.

Обробку команди «start» виконує метод `start (message)`, який містить виклик методу `start_command (message)`.

Обробку команди «help» виконує метод `help (message)`, користувач отримує довідкове повідомлення зі списком доступних команд.

Обробку команди «registration» виконує метод `get_company (message)`, який містить виклик методу `get_company_method (message)`. У методі

`get_company_method (message)` відбувається перевірка користувача на авторизованість, в разі наявності реєстрації користувача виконується відправка клавіатури з вибором компанії, до якої належить користувач. При натисканні на кнопку з компанією відбувається обробка вмісту поля `call.data` в методі `callback_inline (call)` для даних `company1`, `company2`, `company3`, `company4`, `company5`, `company6`. Незалежно від вибору компанії користувача відбувається виклик методу `remote_company_markup (call, current_message)` - видалення клавіатури з текстовим підтвердженням вибору. Після відбувається виклик методу `confirm_company (call, current_message)`, який виконує відправку користувачеві клавіатури з кнопками «ТАК» і «НІ» і повідомленням з проханням підтвердити вибір або відмовитися від нього. Обробка натискання на кнопки «ТАК» і «НІ» відбувається в методі `callback_inline (call)` для даних `yes_company`, `no_company` відповідно. У разі натискання кнопки «НІ» користувач отримує повідомлення «Ви можете змінити свій вибір.», Клавіатура видаляється, відбувається виклик методу `get_company_method (message)`, що дозволяє вибрати компанію знову, якщо сталася помилка. У разі натискання кнопки «ТАК» відбувається виклик методу `remote_company_confirm_markup (call)` - відправка користувачеві повідомлення «Ви підтвердили приналежність до наступної організації: <назва організації користувача>», видалення клавіатури підтвердження компанії. Після викликається метод `phone (call.message)` - містить відправку повідомлення «Для реєстрації вам необхідно надати свій номер телефону. Він буде використаний виключно для реєстрації. » і клавіатуру з відповідною кнопкою надання номера телефону. Натискання цієї кнопки обробляється в методі `contact (message)`, який містить відправку заявки на реєстрацію модератору, користувач отримує повідомлення «Дякую за реєстрацію, ваша заявка буде оброблена протягом 1 робочого дня.».

Обробку команди «exit» виконує метод `exit_user (message)`, який містить виклик методу `exit_user_method (message)`. У разі незареєстрованого користувача - повідомлення про помилку, інакше останній пропонує

користувачеві підтвердити скасування реєстрації, відправляючи клавіатуру з кнопками «ТАК» і «НІ». Обробка натискання цих кнопок відбувається в методі `callback_inline (call)` для даних `yes_exit_user`, `no_exit_user` відповідно. У разі отримання підтвердження відписки в методі `callback_inline (call)` встановлюється значення `True` змінної `exit_confirm_user_flag`, інакше встановлюється значення `False`. В обох випадках виконується виклик методу `remote_exit_confirm_markup (call)`. Цей метод в залежності від значення змінної повідомляє користувача про скасування відписки (`exit_confirm_user_flag = False`), або відправляє користувачеві підтвердження формування заявки на відписку, формує останню для модератора (`exit_confirm_user_flag = True`).

Також існує службова команда «/id» (доступна всім типам користувачів), обробка якої виконується за такими ключовими словами: «id», «ID», «iD», «Id». За обробку відповідає метод `get_id (message)`, виконується повернення користувачеві інформації про його `telegram_id`.

Для модератора передбачена обробка наступних команд:

- «/help_mod» - довідка;
- «/auth» - авторизація модератора;
- «/reauth» - вихід з аккаунта модератора;
- «/add» - схвалити заявку, додати користувача;
- «/show» - перегляд всіх зареєстрованих користувачів;
- «/fail» - відмовити в реєстрації;
- «/remove» - видалити вже зареєстрованого користувача;
- «/answer» - відповісти на повідомлення користувача;
- «/answer_by_id» - відповісти на повідомлення користувача по id;
- «/send_all» - розсилка повідомлення всім користувачам;
- «/send_by_id» - адресна відправка листа за id;
- «/get_message» - отримати повідомлення за id чату і повідомлення.

Обробку команди «help_mod» виконує метод help_mod (message), який в залежності від авторизованості модератора надає список команд для модератора або просить авторизуватися.

Обробку команди «auth» виконує метод auth_mod (message), перевіряються права, в разі їх відсутності запитується логін, обробка якого відбувається в методі get_password (message), де в залежності від статусу авторизації відбувається відправка повідомлення про помилку або відправка переліку команд модератора.

Обробку команди «reauth» виконує метод reauth_mod (message), де користувачеві відправляється клавіатура з кнопками «ТАК» і «НІ» для підтвердження або відмови від виходу з облікового запису модератора. У разі відсутності прав модератора користувач отримує повідомлення з проханням реєстрації аккаунтом модератора. Обробка натискання цих кнопок відбувається в методі callback_inline (call) для даних yes_reauth_mod, no_reauth_mod відповідно. У разі підтвердження виходу з облікового запису модератора відбувається очищення даних облікового запису, незалежно від вибору викликається метод remote_reauth_confirm_markup (call), в якому в залежності від авторизованості аккаунтом модератора поточного облікового запису, користувач отримує підтвердження виходу з облікового запису або підтвердження відмови від виходу зі свого облікового запису.

Обробку команди «add» виконує метод add_user_mod (message), де запитується прізвище або номер телефону співробітника, якого необхідно зареєструвати, обробка введених даних відбувається в методі get_sur_user_add_mod (message), де за допомогою методу method_show_people_from_db (message.text) відбувається запит до БД, результат пошуку відправляється повідомленням модератору, відбувається запит id користувача з цього списку. Введене id (унікальний ідентифікатор співробітника в БД) обробляється в методі get_id_user_add_mod (message), де також запитується telegram_id, обробка якого відбувається в методі get_tel_id_user_add_mod (message), де в разі вторинної реєстрації виникає

повідомлення про помилку, інакше формується вітальне повідомлення для користувача, якого реєстрували, і повідомлення про підтвердження успішності реєстрації для модератора в разі, якщо введений коректний telegram_id, інакше формується повідомлення з помилкою. Також відбувається запит введення компанії користувача, обробка якої відбувається в методі get_comp_user_add_mod (message), модератор отримує підтвердження успішності реєстрації.

Обробку команди «show» виконує метод show_mod (message), де модератору повертається повідомлення зі списком зареєстрованих в боті користувачів.

Обробку команди «fail» виконує метод fail_user_mod (message), де у модератора запитується telegram_id користувача, якому необхідно відмовити в реєстрації. Обробка отриманого id відбувається в методі get_id_user_del_mod (message), якщо введений коректний id, користувач отримує повідомлення про відмову в реєстрації, модератор отримує підтвердження успішності операції, інакше виникає повідомлення про помилку для модератора.

Обробку команди «remove» виконує метод remove_user_mod (message), де запитується id з БД співробітника, обробка якого відбувається в методі get_id_user_remove_mod (message), де в разі успіху користувач і модератор отримують підтвердження успіху операції, інакше модератор отримує повідомлення про помилку.

Обробку команди «answer» виконує метод process_start (message), де модератору пропонують відповісти на цільове повідомлення користувача. Обробка відправки відповіді модератора на повідомлення користувача відбувається в методі process_mind (message), де модератор отримує підтвердження успішності операції, користувач отримує повідомлення модератора, відповідь модератора зберігається у відповідні таблиці БД.

Обробку команди «answer_by_id» виконує метод answer_by_id (message), де у модератора запитується telegram_id користувача, якому

необхідно відповісти. Далі в методі `get_id_for_answer (message)` запитується введення повідомлення, яке буде відправлено. У методі `get_text_for_answer (message)` відбувається адресна відповідь, в разі введення помилкового `telegram_id` модератор отримує повідомлення про помилку.

Обробку команди «`send_all`» виконує метод `send_all_mod (message)`, де запитується введення повідомлення для розсилки. У методі `get_password (message)` викликається функція `get_all_id_user_method()` для формування списку адресатів, відбувається розсилка повідомлення модератора.

Обробку команди «`send_by_id`» виконує метод `send_by_id (message)`, де запитується `telegram_id` користувача - адресанта. Метод `get_id_for_send (message)` запитує введення повідомлення для адресної відправки. Метод `get_text_for_send (message)` містить обробку адресної відправки повідомлення, передбачена обробка введення помилкового `telegram_id`.

Обробку команди «`get_message`» виконує метод `get_message_mod (message)`, де запитується введення `id` користувача і `id` повідомлення, яке необхідно отримати, через пробіл (" "). Метод `method_get_message_mod (message)` оброблює дані і повертає шукане повідомлення.

3.3 Інтерфейс

Відповідно до політики месенджеру Telegram боти не можуть починати переписку першими. Бот доступний в пошуку за нікнеймом `@uksupport_bot`. Після переходу до шуканого боту він все ще не має права на переписку з користувачем. Для користувача інтерфейс має вигляд, як на рисунку 3.3.

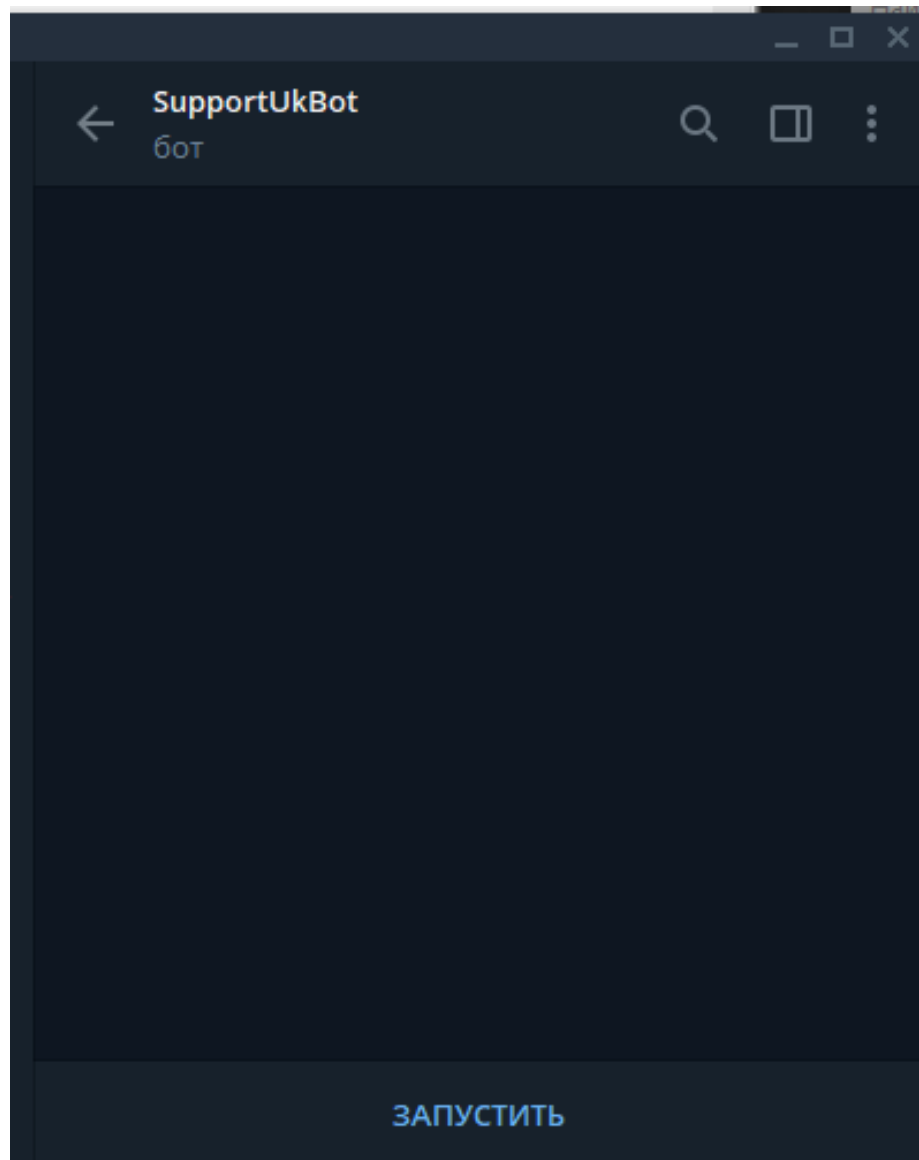


Рисунок 3.3 – Інтерфейс користувача перед перепискою

Лише після натискання стандартної кнопки про запуск бот має право на листування з користувачем.

З технічної точки зору відбувається обробка команди «/start».

Користувач не залежно від наявності реєстрації, або її відсутності, отримує вітальне повідомлення, яке наведено на рисунку 3.4. До повідомлення прикріплено inline-клавіатуру з кнопками переходу до відписки, реєстрації, головного меню.

Головне меню та відписка доступна зареєстрованому користувачу, реєстрація доступна незареєстрованому користувачу.

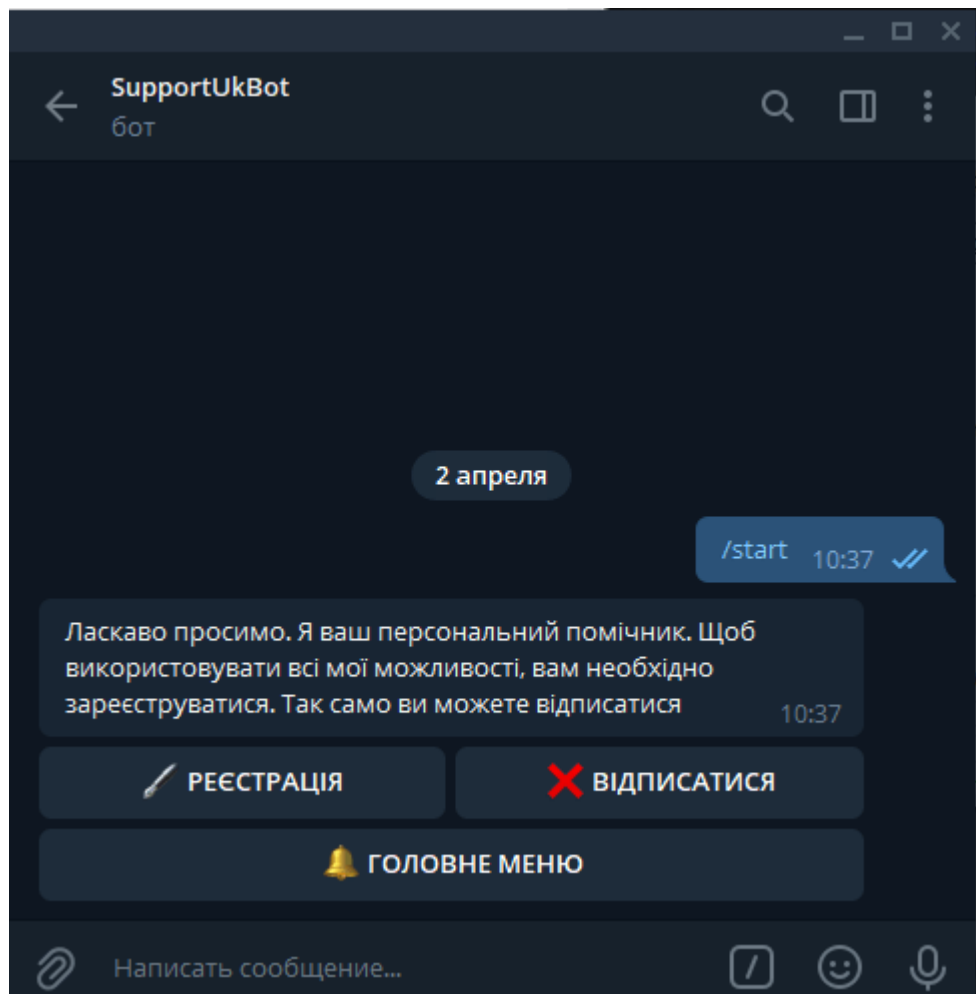


Рисунок 3.4 – Вітальне повідомлення

У випадку, якщо зареєстрований користувач переходить до реєстрації, бот повідомляє про авторизованість поточного облікового запису, повідомлення наведено на рисунку 3.5.

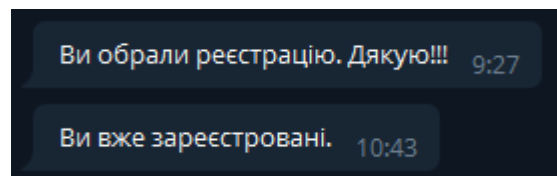


Рисунок 3.5 – Повідомлення про авторизованість користувача

Якщо користувача ще не зареєстровано, бот пропонує обрати компанію, до якої належить користувач. В переліку компаній доступні лише

активи цільової компанії, оскільки технічна підтримка є суворо корпоративною. Повідомлення наведено на рисунку 3.6.

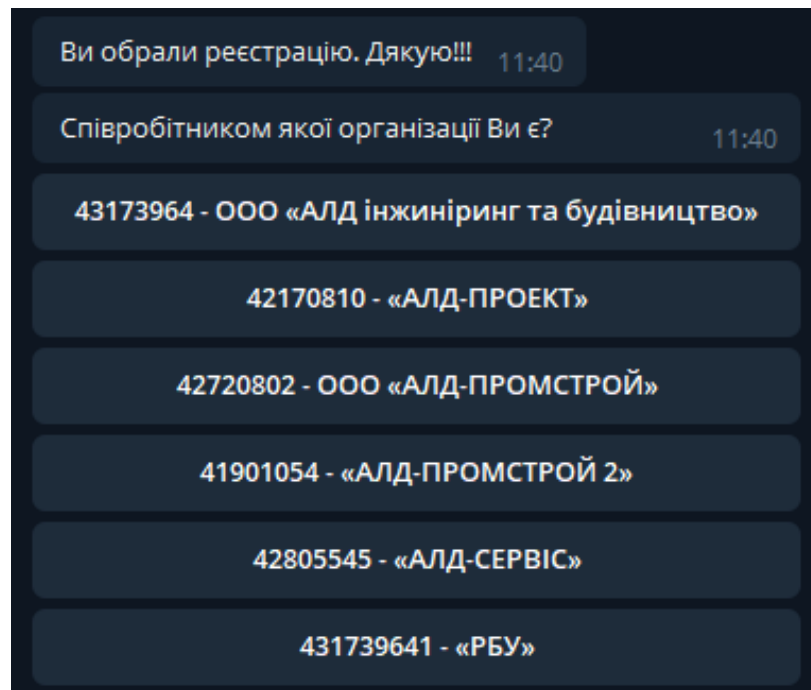


Рисунок 3.6 – Запит компанії користувача

Далі користувачу пропонується підтвердити обрано компанію, повідомлення наведено на рисунку 3.7.

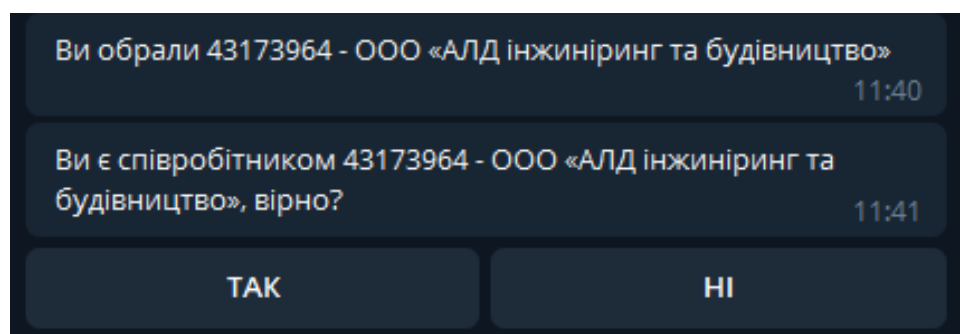


Рисунок 3.7 – Запит підтвердження обраної користувачем компанії

У випадку помилки користувача під час вибору компанії та натисканні на відмову (рисунок 3.7), користувач може обрати компанію знову, повідомлення наведено на рисунку 3.8.

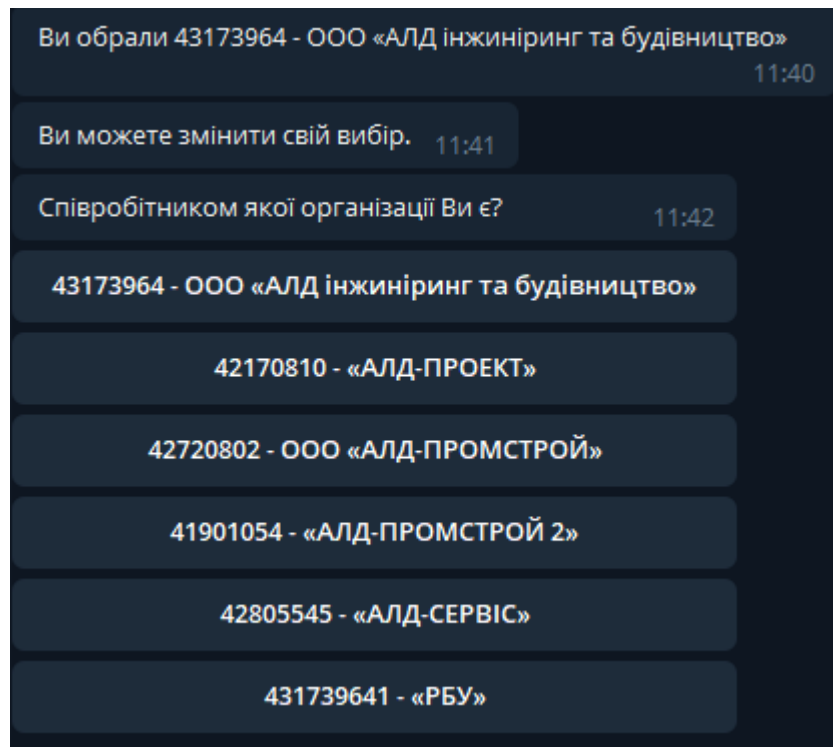


Рисунок 3.8 – Повторний запит компанії користувача

У разі підтвердження компанії (рисунок 3.8) користувачу пропонується надати номер телефону натисканням на reply-кнопку, запит наведено на рисунку 3.9.

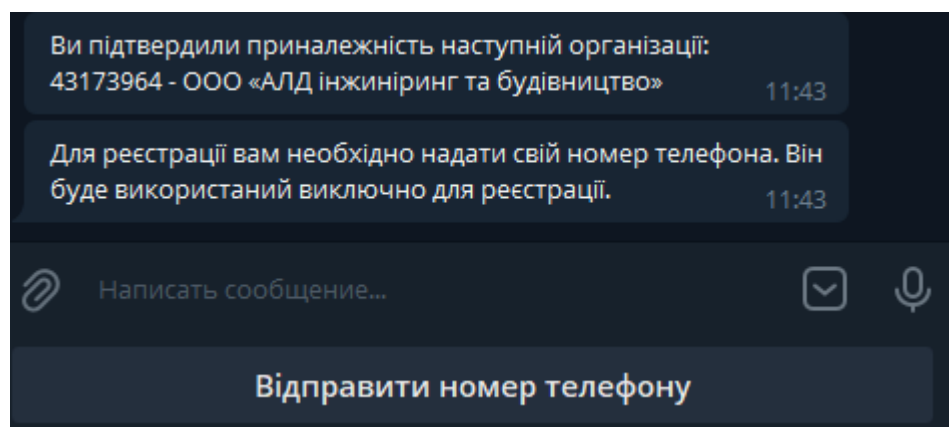


Рисунок 3.9 – Запит номеру телефону

Натискання на кнопку надання номеру викликає стандартне повідомлення Telegram про підтвердження надання номеру телефону боту, повідомлення наведено на рисунку 3.10.

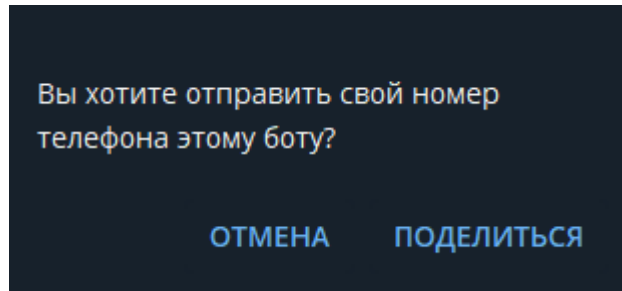


Рисунок 3.10 – Підтвердження надання номеру телефону

Після надання номеру телефону процес реєстрації для користувача завершено. Процес від запуску бота до повідомлення про успішне формування заявки про реєстрацію наведено на рисунку 3.11.

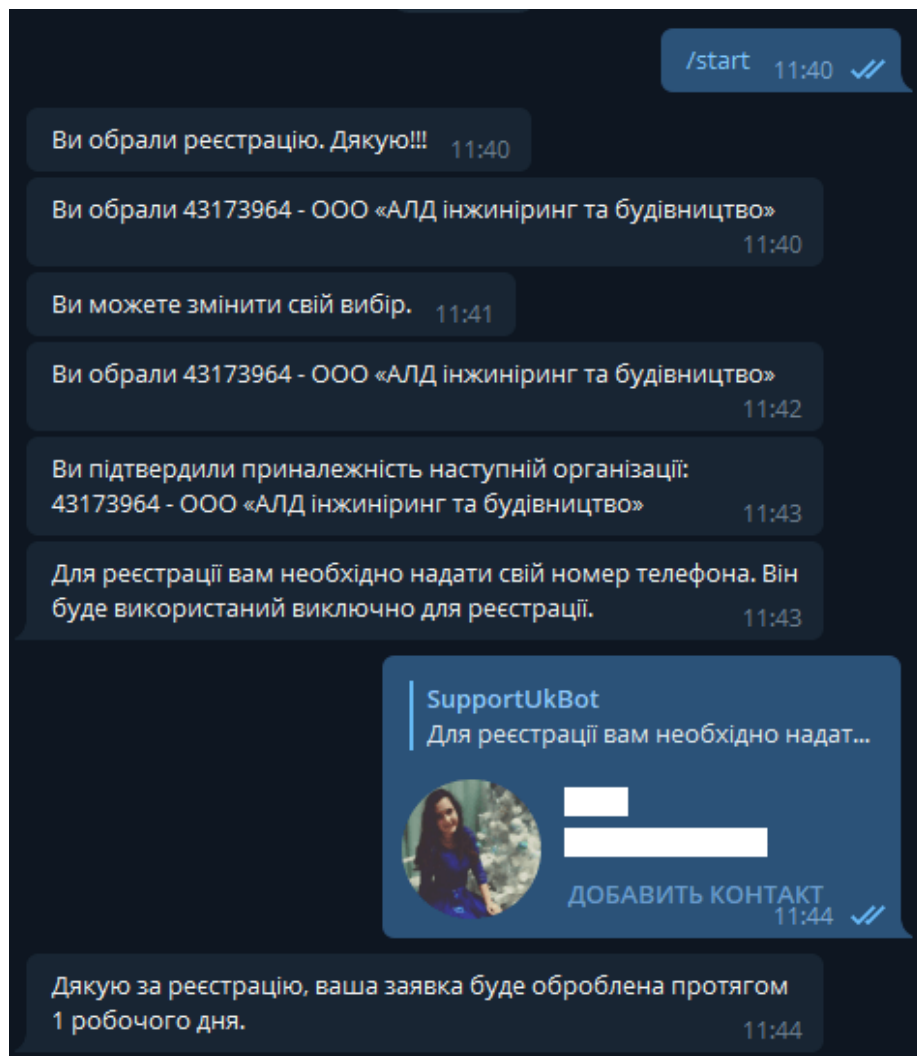


Рисунок 3.11 – Реєстрація користувача

Відповідно модератор отримує повідомлення про заявку на реєстрацію від користувача. Процес реєстрації користувача модератором (схвалення заявки на реєстрацію) наведено на рисунку 3.12.

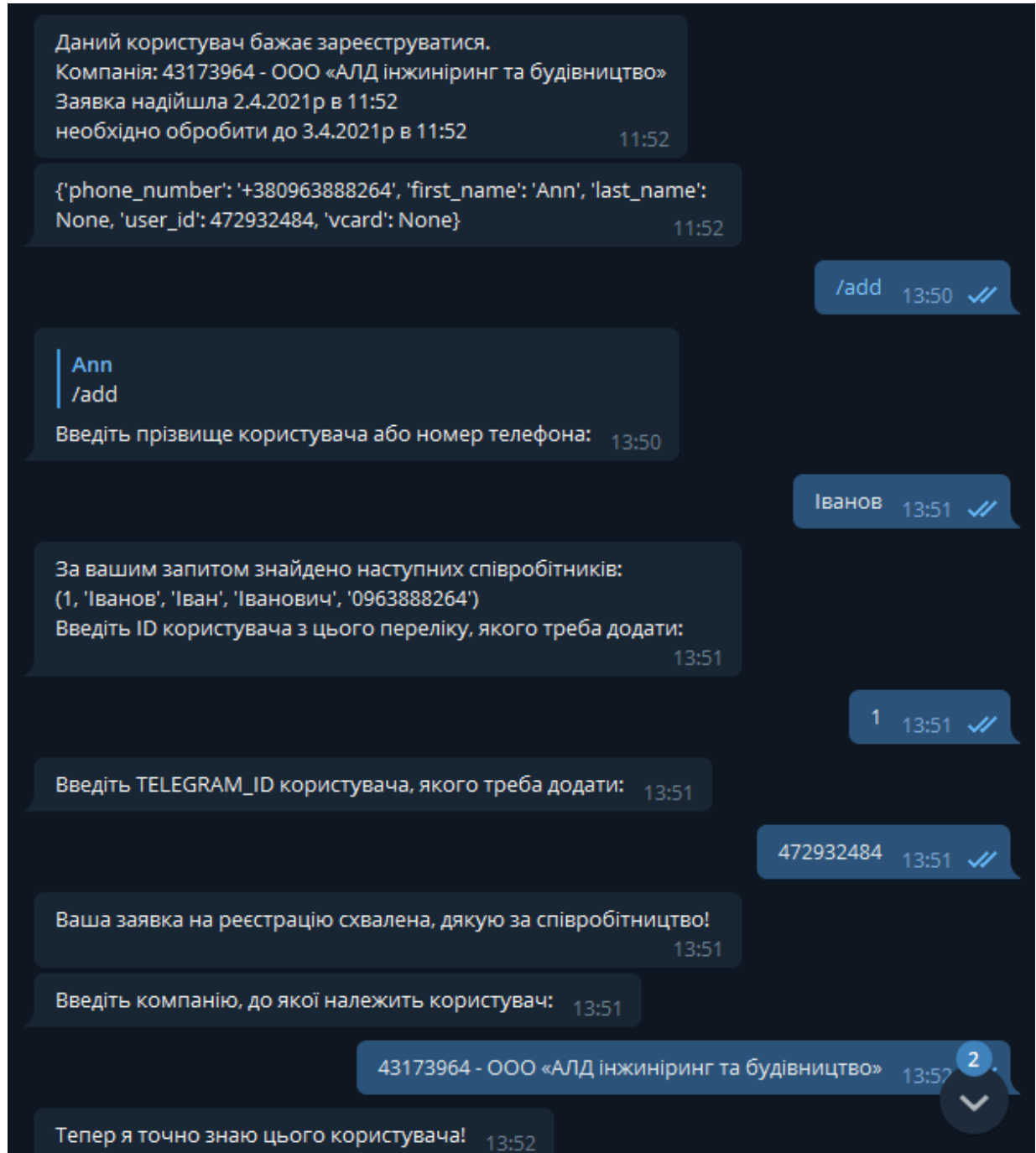


Рисунок 3.12 – Схвалення заявки на реєстрацію модератором

Пошук за прізвищем або номером телефону відбувається за таблицею БД, яка наповнюється даними з відділу кадрів підприємства. Таким чином

зберігається корпоративність боту. Модератор надає дані про telegram_id користувача для подальшої переписки бота та користувача, та дані про компанію користувача, бот надає повідомлення про успішність схвалення заявки.

На рисунку 3.13 наведено виконання команди «/show» – перегляду всіх зареєстрованих в боті користувачів.

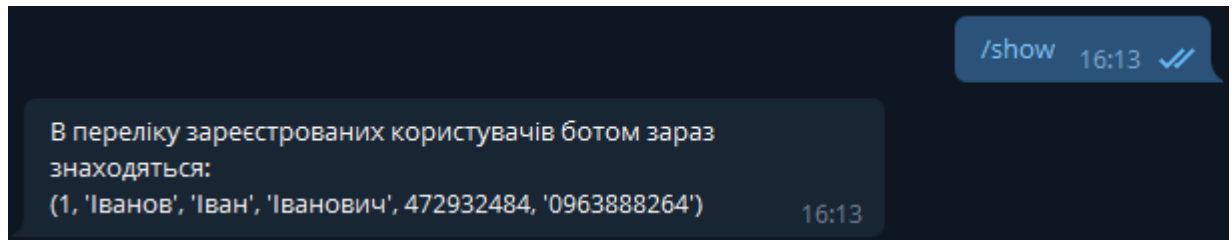


Рисунок 3.13 - Перегляд всіх зареєстрованих в боті користувачів

Після схвалення заявки на реєстрацію модератором користувач отримує повідомлення (рисунок 3.14) з кнопками головного меню та закріплюється герлу-кнопка під полем введення повідомлень, яка доступна у будь-який момент часу.

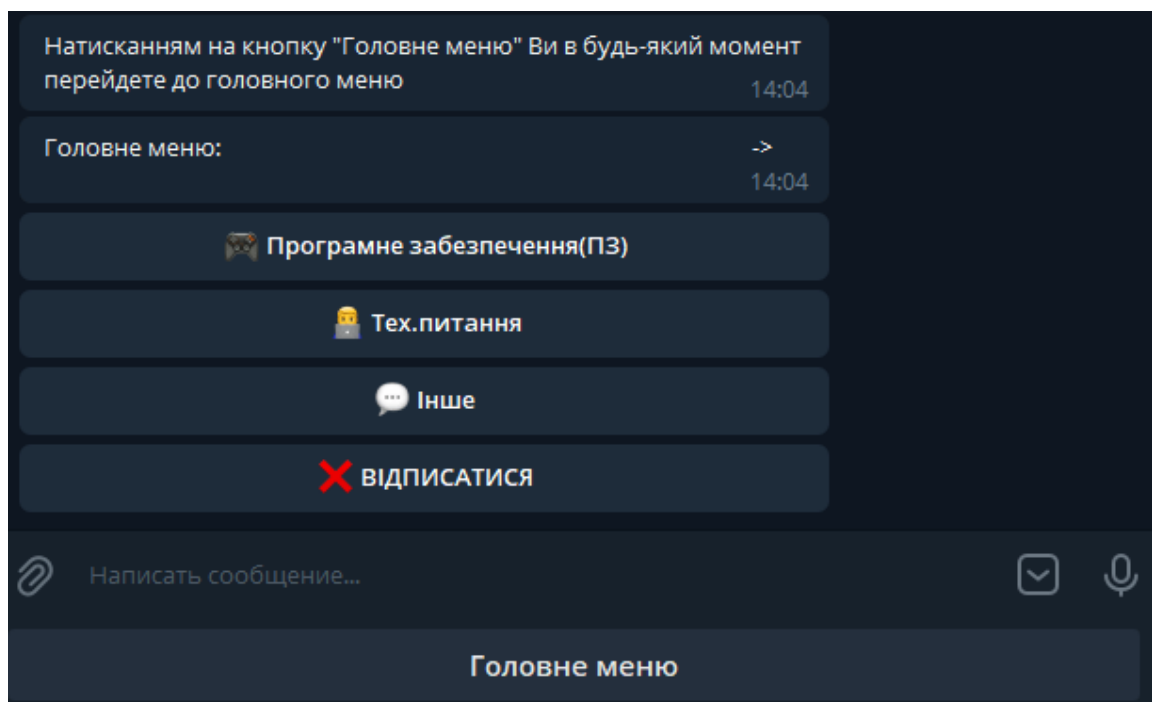


Рисунок 3.14 – Повідомлення з кнопками головного меню

Нижче буде наведено зовнішній вигляд inline-клавіатур всіх розділів меню. На рисунку 3.15 наведено клавіатуру розділу «Програмне забезпечення(ПЗ)».

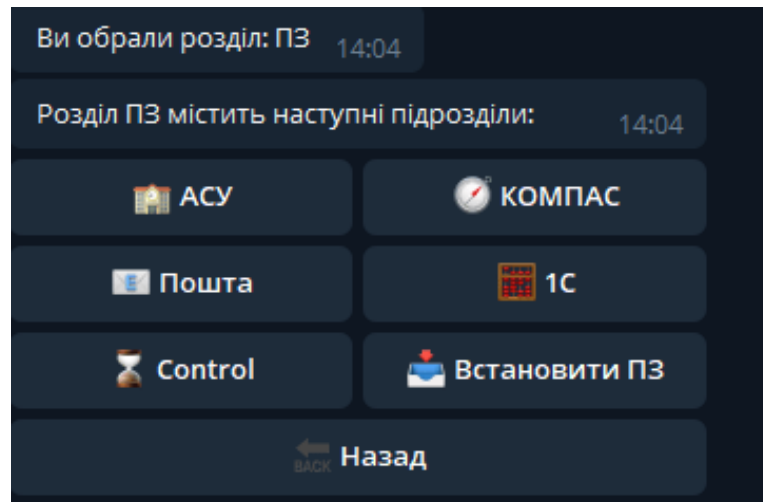


Рисунок 3.15 – Клавіатура розділу «Програмне забезпечення(ПЗ)»

На рисунку 3.16 наведено клавіатуру розділу «АСУ».

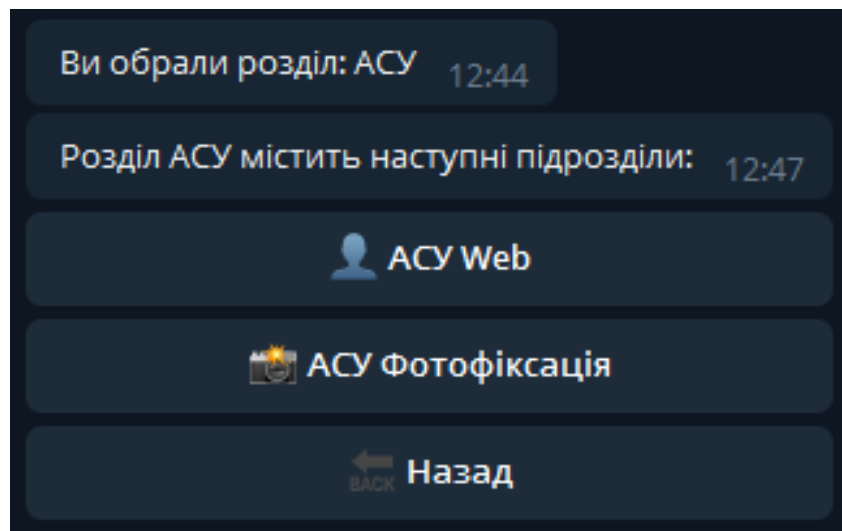


Рисунок 3.16 – Клавіатура розділу «АСУ»

На рисунку 3.17 наведено клавіатуру розділу «АСУ Web».

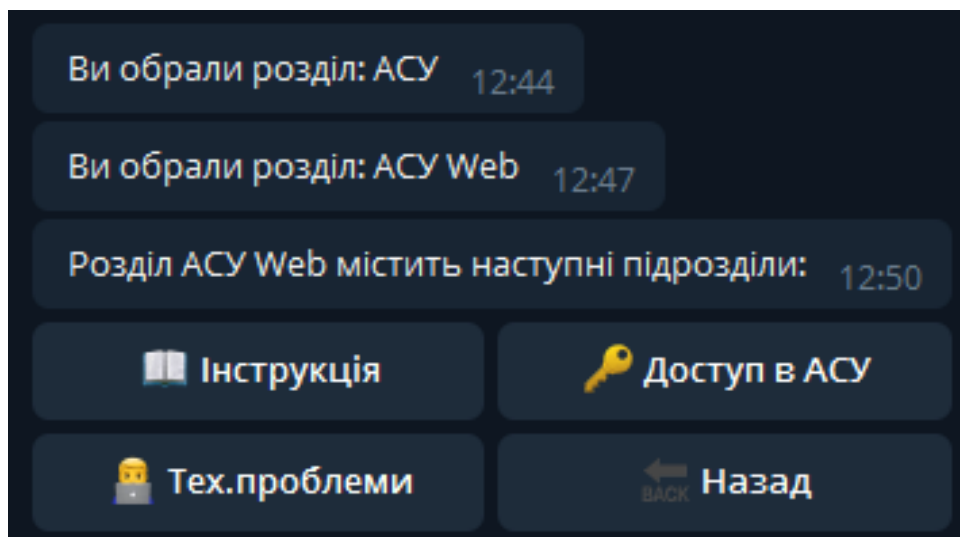


Рисунок 3.17 – Клавіатура розділу «АСУ Web»

На рисунку 3.18 наведено клавіатуру розділу «Інструкція».

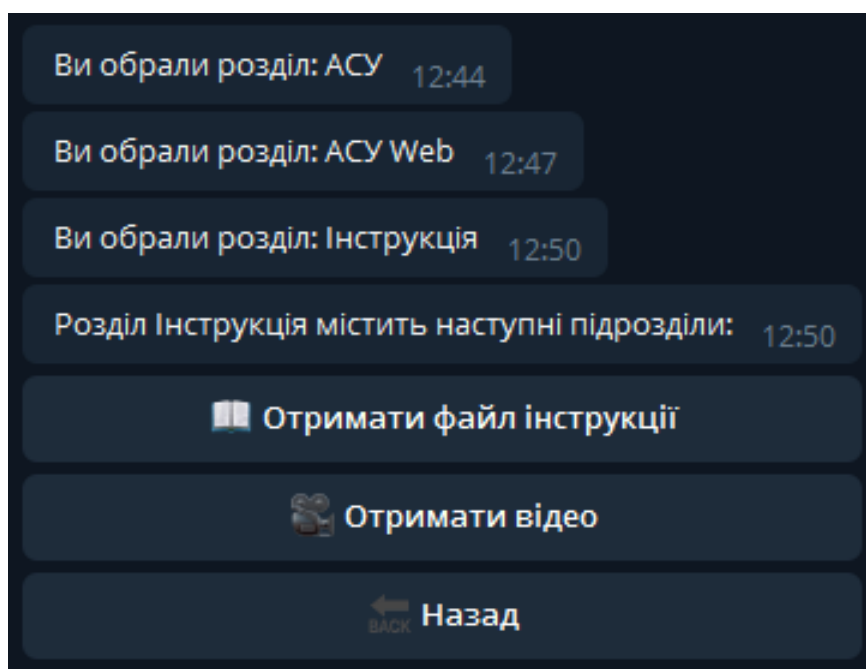


Рисунок 3.18 – Клавіатура розділу «Інструкція»

На рисунку 3.19 наведено результат натискання кнопки «Отримати файл інструкції» в розділі «АСУ Web». Користувач в чаті отримує файл інструкції до відповідного корпоративного програмного забезпечення.

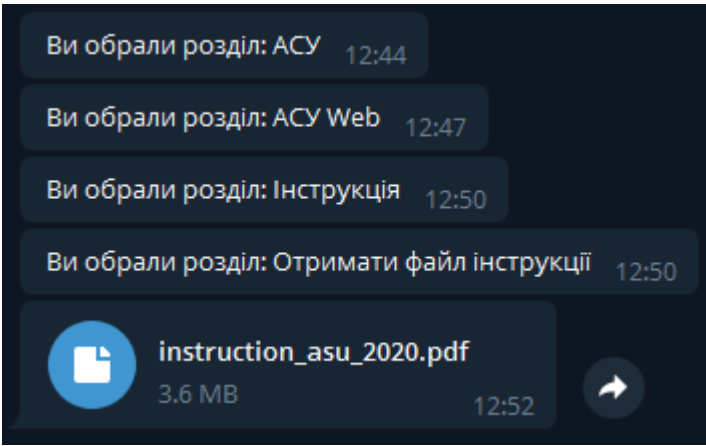


Рисунок 3.19 – Результат натискання кнопки «Отримати файл інструкції» в розділі «АСУ Web»

На рисунку 3.20 наведено результат натискання кнопки «Отримати відео» в розділі «АСУ Web». Користувач в чаті отримує відео-інструкцію до відповідного корпоративного програмного забезпечення.

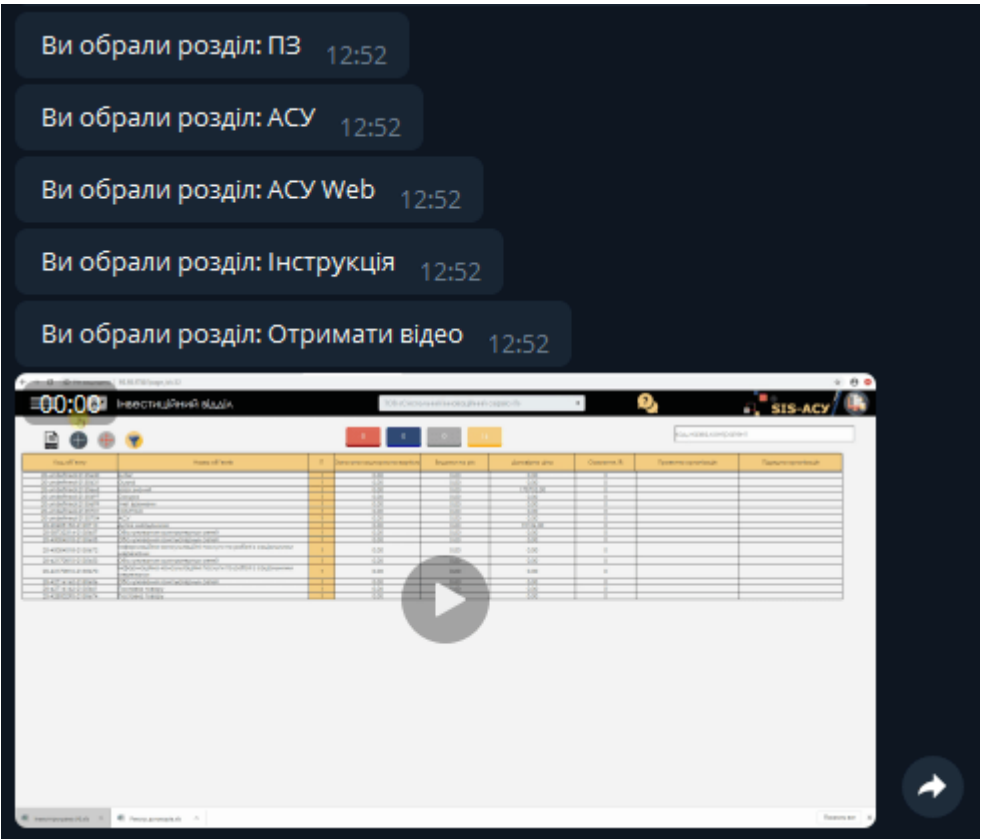


Рисунок 3.20 – Результат натискання кнопки «Отримати відео» в розділі «АСУ Web»

На рисунку 3.21 наведено клавіатуру розділу «Доступ в АСУ».

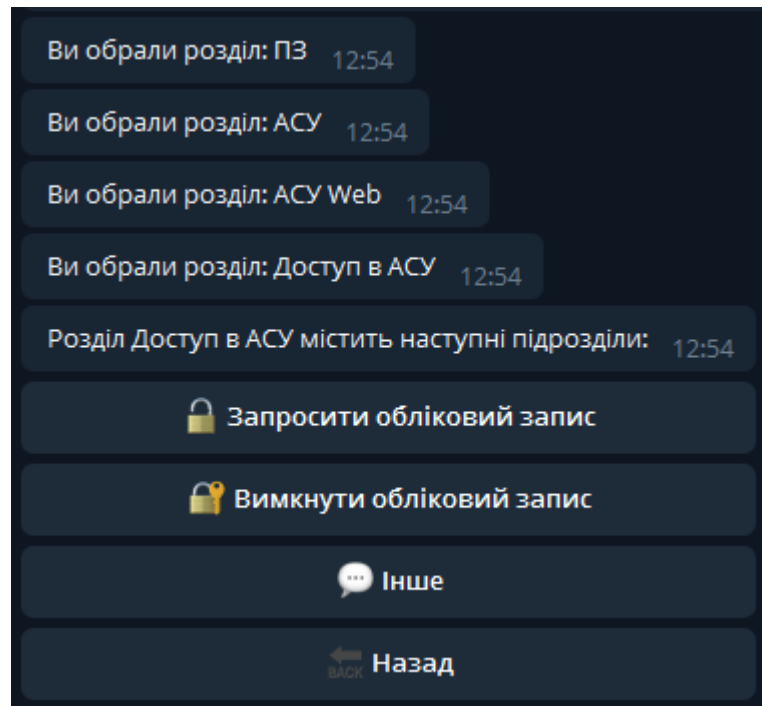


Рисунок 3.21 – Клавіатура розділу «Доступ в АСУ»

На рисунку 3.22 наведено клавіатуру розділу «АСУ Фотофіксація».

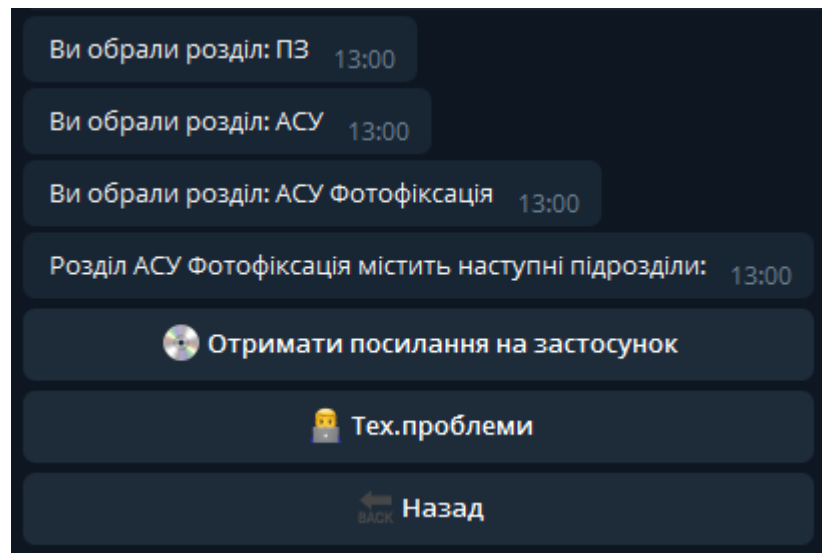


Рисунок 3.22 – Клавіатура розділу «АСУ Фотофіксація»

На рисунку 3.23 наведено результат натискання кнопки «Отримати посилання на застосунок» в розділі «АСУ Фотофіксація».

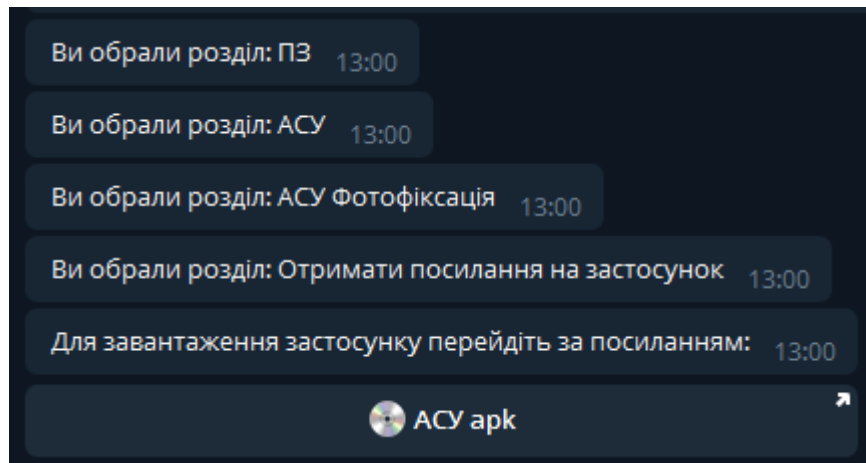


Рисунок 3.23 – Результат натискання кнопки «Отримати посилання на застосунок» в розділі «АСУ Фотофіксація»

На рисунку 3.24 наведено результат натискання кнопки з посиланням на застосунок в розділі «АСУ Фотофіксація». Користувач переходить за посиланням та починається завантаження відповідного застосунку.

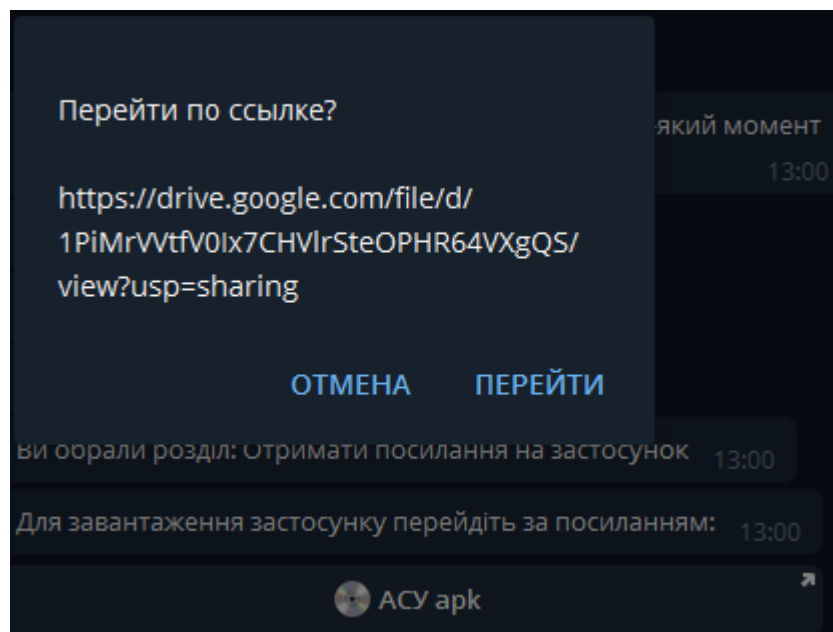


Рисунок 3.24 – Результат натискання кнопки з посиланням на застосунок в розділі «АСУ Фотофіксація»

На рисунку 3.25 наведено клавіатуру розділу «КОМПАС».

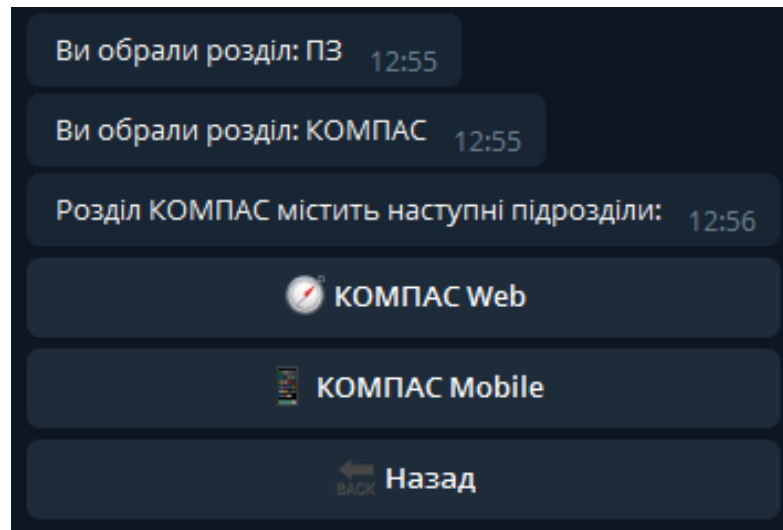


Рисунок 3.25 – Клавіатура розділу «КОМПАС»

На рисунку 3.26 наведено клавіатуру розділу «КОМПАС Web».

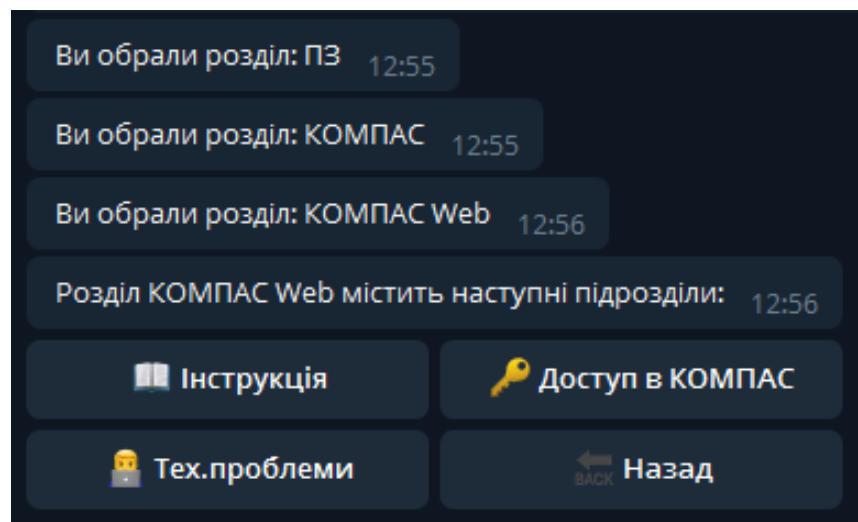


Рисунок 3.26 – Клавіатура розділу «КОМПАС Web»

На рисунку 3.27 наведено клавіатуру розділу «Інструкція».

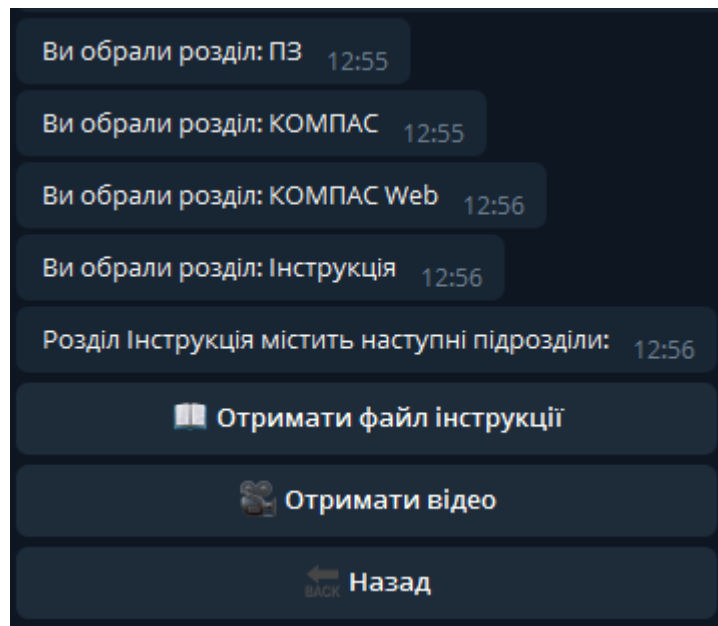


Рисунок 3.27 – Клавіатура розділу «Інструкція»

На рисунку 3.28 наведено результат натискання кнопки «Отримати файл інструкції» в розділі «КОМПАС Web». Користувач в чаті отримує файл інструкції до відповідного корпоративного програмного забезпечення.

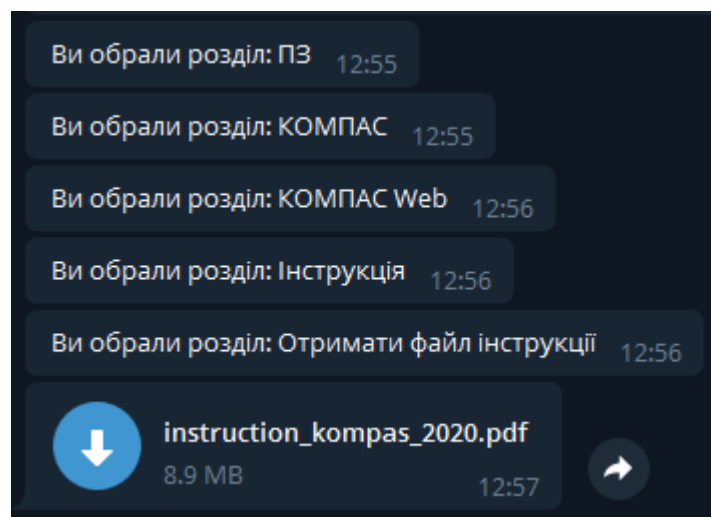


Рисунок 3.28 – Результат натискання кнопки «Отримати файл інструкції» в розділі «КОМПАС Web»

На рисунку 3.29 наведено результат натискання кнопки «Отримати відео» в розділі «КОМПАС Web». Користувач в чаті отримує відео-інструкцію до відповідного корпоративного програмного забезпечення.

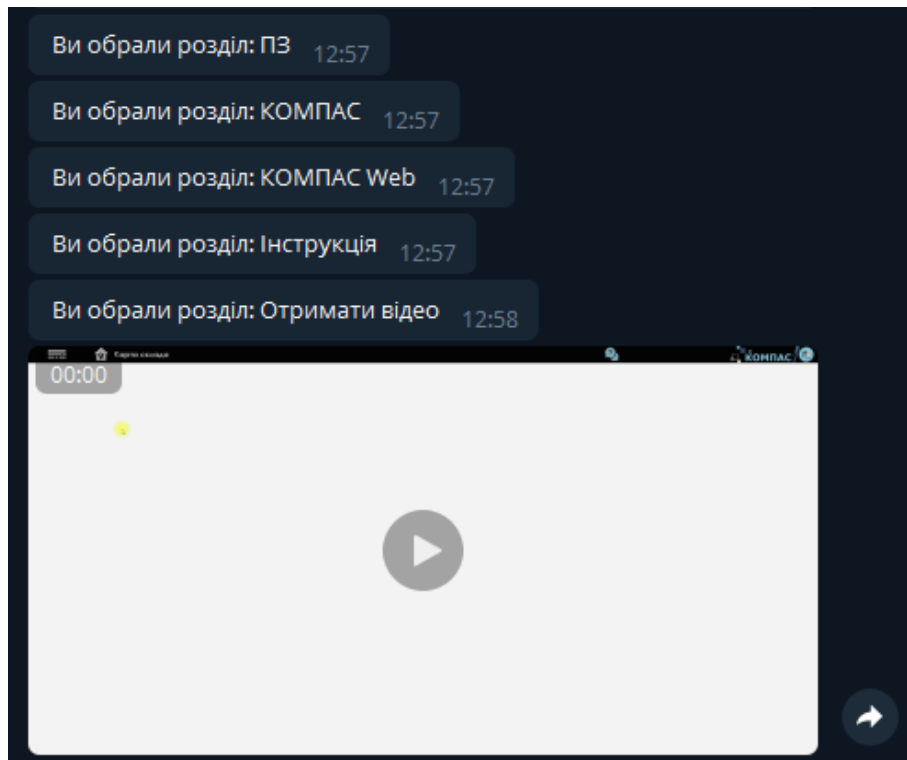


Рисунок 3.29 – Результат натискання кнопки «Отримати відео» в розділі «КОМПАС Web»

На рисунку 3.30 наведено клавіатуру розділу «Доступ в КОМПАС».

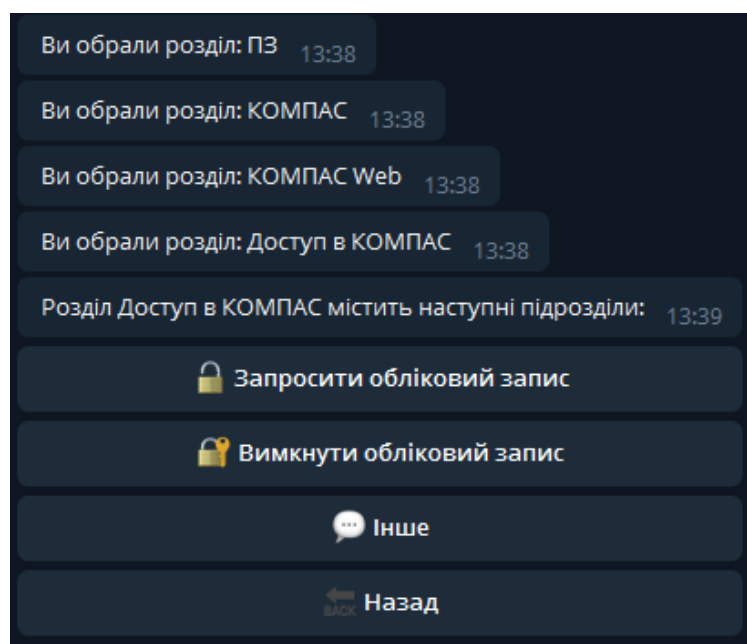


Рисунок 3.30 – Клавіатура розділу «Доступ в КОМПАС»

На рисунку 3.31 наведено клавіатуру розділу «КОМПАС Mobile».

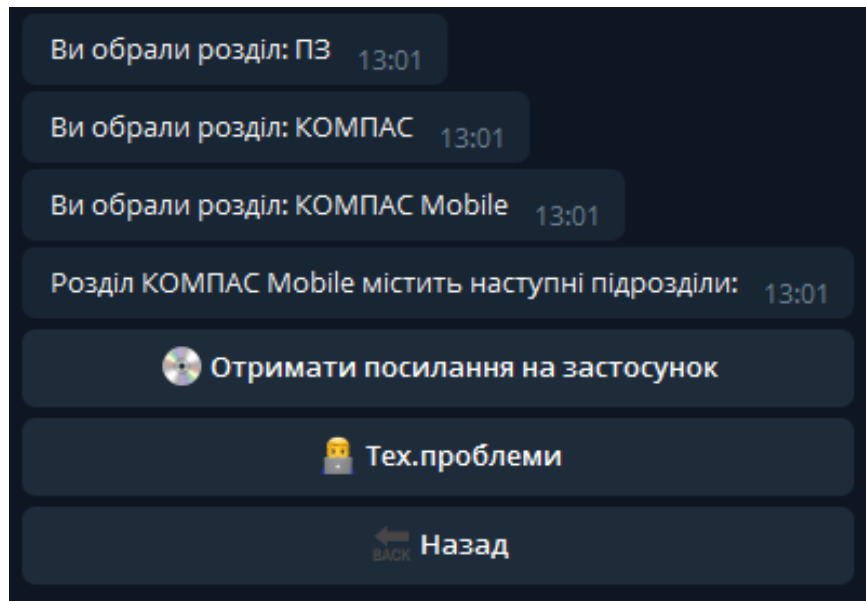


Рисунок 3.31 – Клавіатура розділу «КОМПАС Mobile»

На рисунку 3.32 наведено результат натискання кнопки «Отримати посилання на застосунок» в розділі «КОМПАС Mobile».

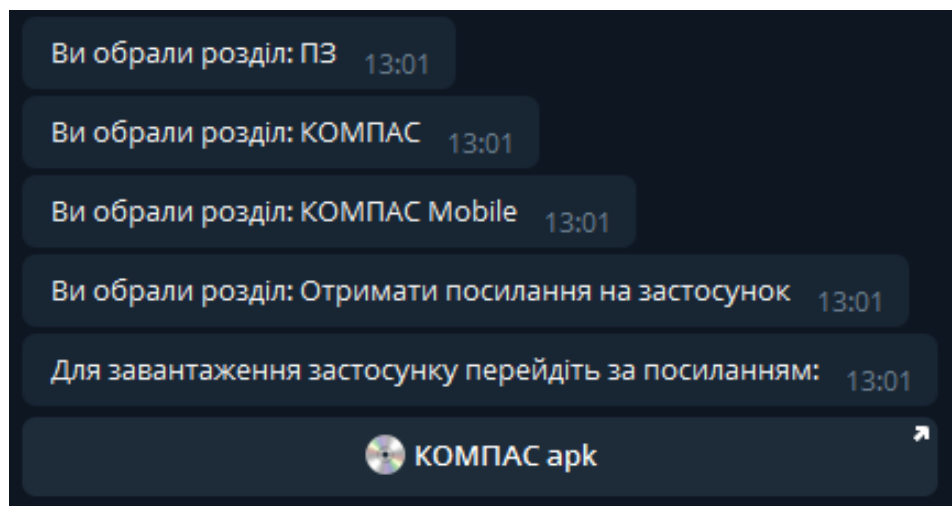


Рисунок 3.32 – Результат натискання кнопки «Отримати посилання на застосунок» в розділі «КОМПАС Mobile»

На рисунку 3.33 наведено результат натискання кнопки з посиланням на застосунок в розділі «КОМПАС Mobile». Користувач переходить за посиланням та починається завантаження відповідного застосунку.

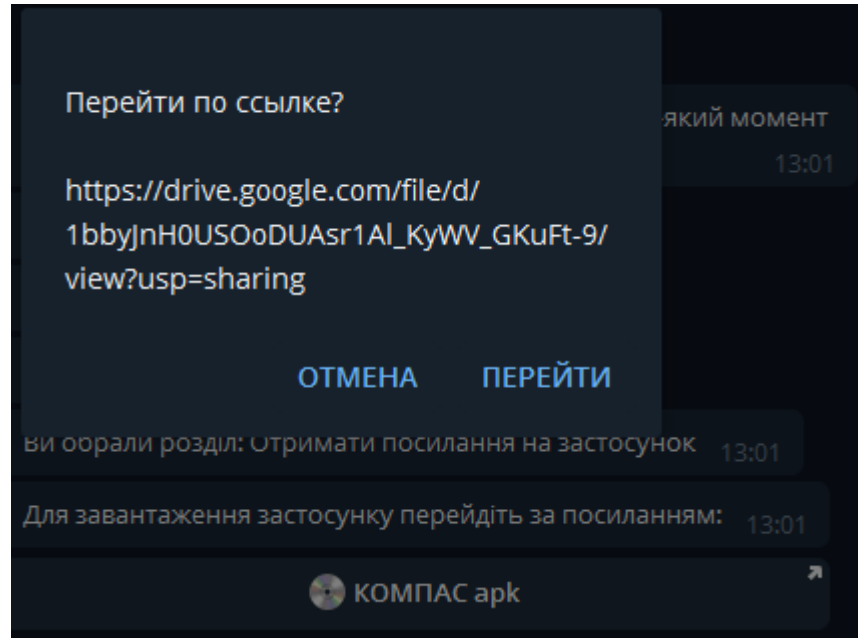


Рисунок 3.33 – Результат натискання кнопки з посиланням на застосунок в розділі «КОМПАС Mobile»

На рисунку 3.34 наведено клавіатуру розділу «Пошта».

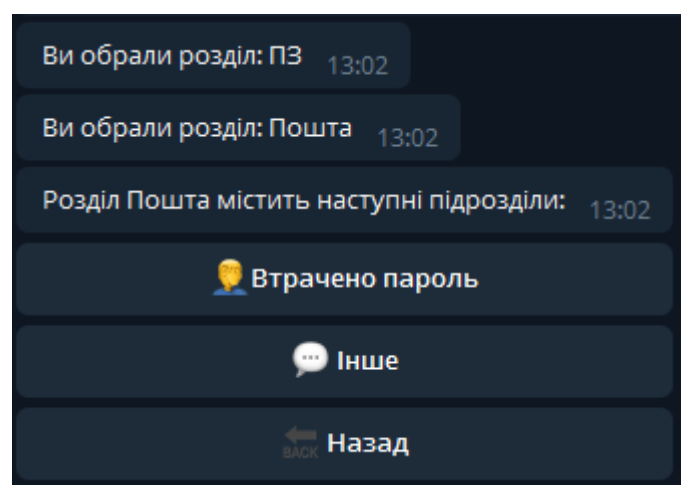


Рисунок 3.34 – Клавіатура розділу «Пошта»

На рисунку 3.35 наведено клавіатуру розділу «1С».

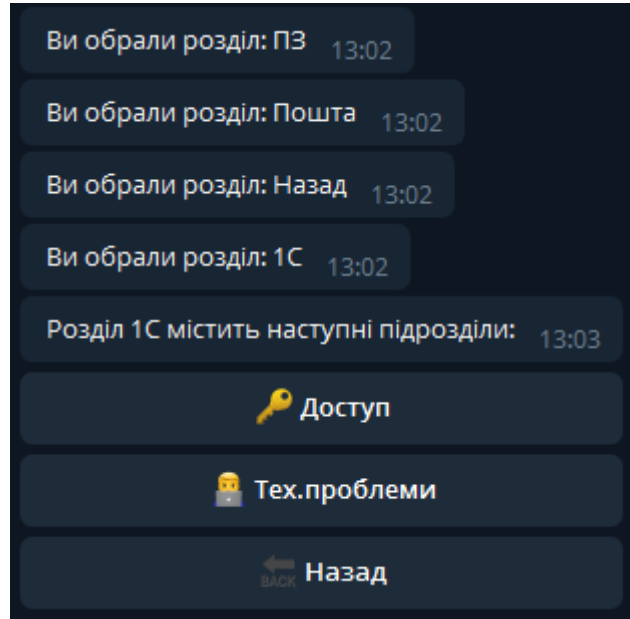


Рисунок 3.35 – Клавіатура розділу «1С»

На рисунку 3.36 наведено клавіатуру розділу «Доступ в 1С».

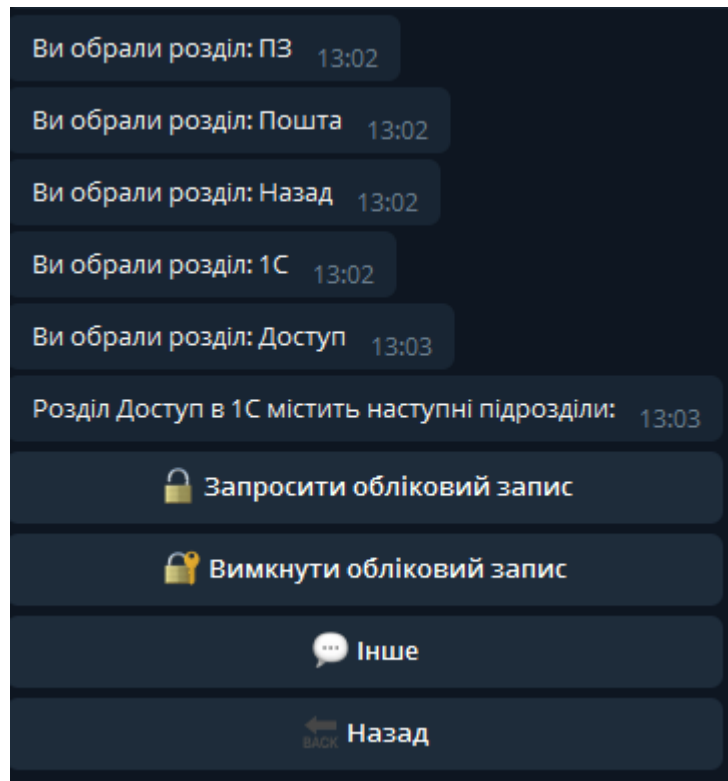


Рисунок 3.36 – Клавіатура розділу «Доступ в 1С»

На рисунку 3.37 наведено клавіатуру розділу «Control».

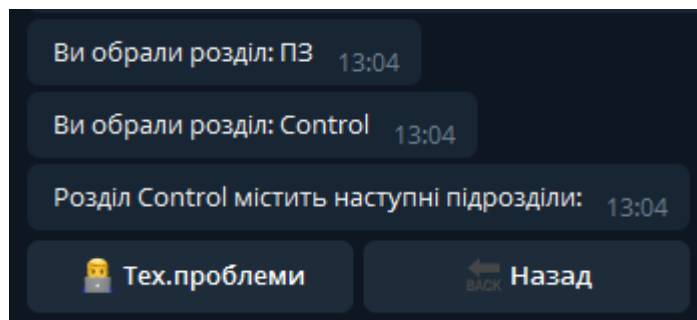


Рисунок 3.37 – Клавіатура розділу «Control»

На рисунку 3.38 наведено клавіатуру розділу «Встановити ПЗ».

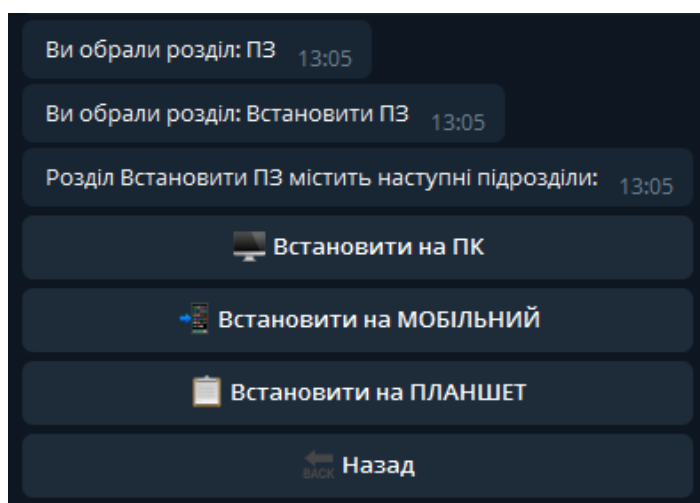


Рисунок 3.38 – Клавіатура розділу «Встановити ПЗ»

На рисунку 3.39 наведено клавіатуру розділу «Тех.питання».

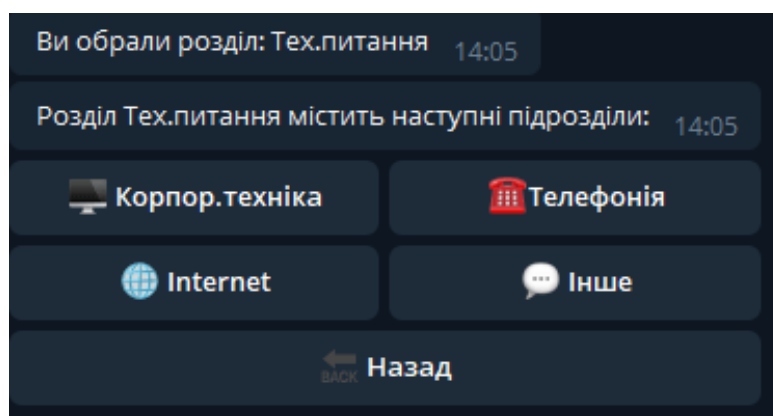


Рисунок 3.39 – Клавіатура розділу «Тех.питання»

На рисунку 3.40 наведено клавіатуру розділу «Корпоративна техніка».

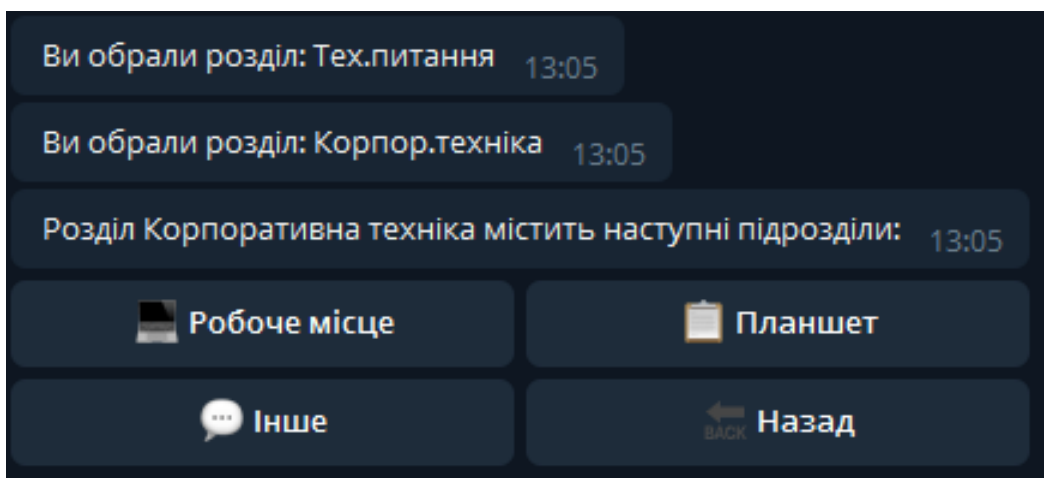


Рисунок 3.40 – Клавіатура розділу «Корпоративна техніка»

На рисунку 3.41 наведено клавіатуру розділу «Телефонія».

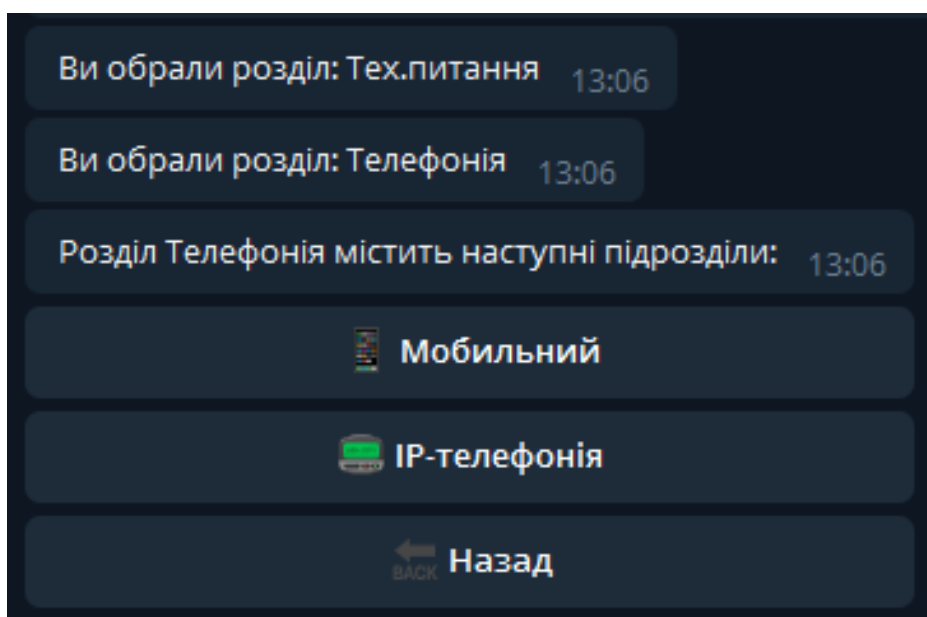


Рисунок 3.41 – Клавіатура розділу «Телефонія»

На рисунку 3.42 наведено клавіатуру розділу «Internet».

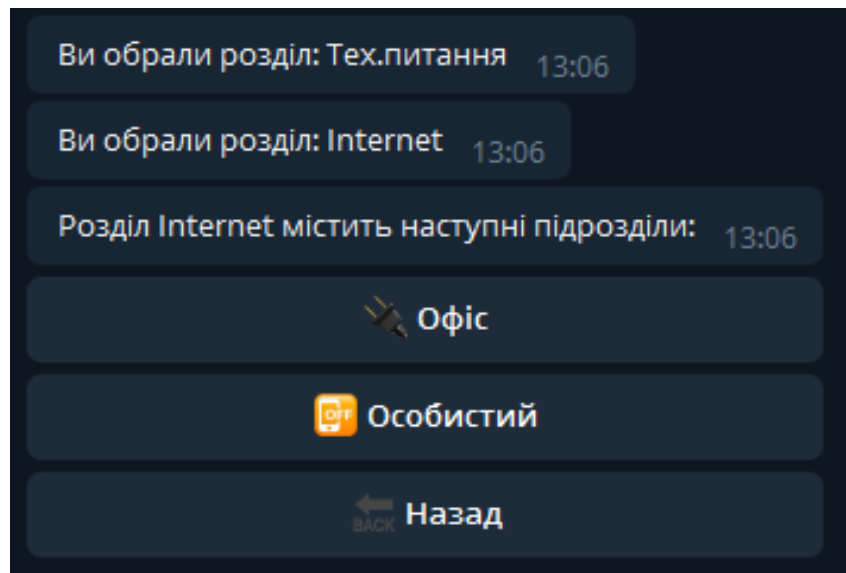


Рисунок 3.42 – Клавіатура розділу «Internet»

На рисунку 3.43 наведено початок залишення заявки в розділі «Інше». Бот запитує введення опису заявки текстовим повідомленням.

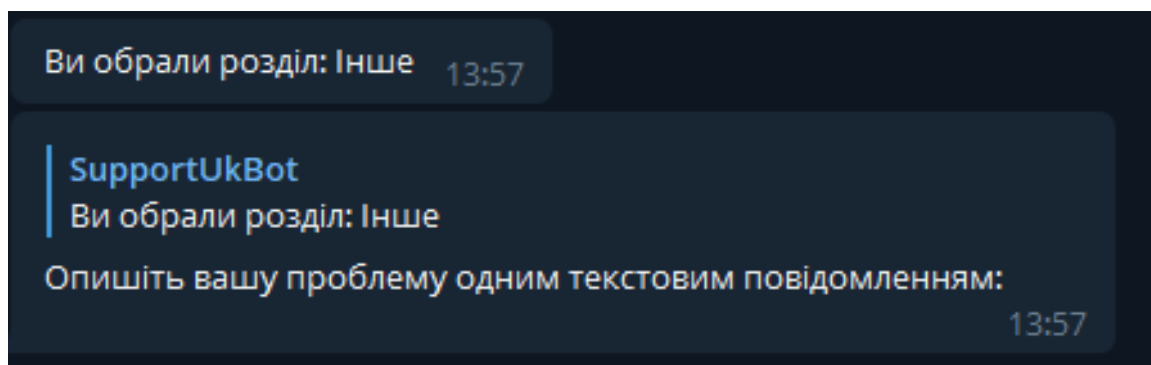


Рисунок 3.43 – Початок залишення заявки в розділі «Інше»

На рисунку 3.44 наведено процес загального випадку залишення заявки. Користувач вводить текстове повідомлення з описом проблеми, прикріплює за необхідності фото чи файл (у даному випадку фото). Також користувач отримує підтвердження про прикріплення вкладень до заявки.

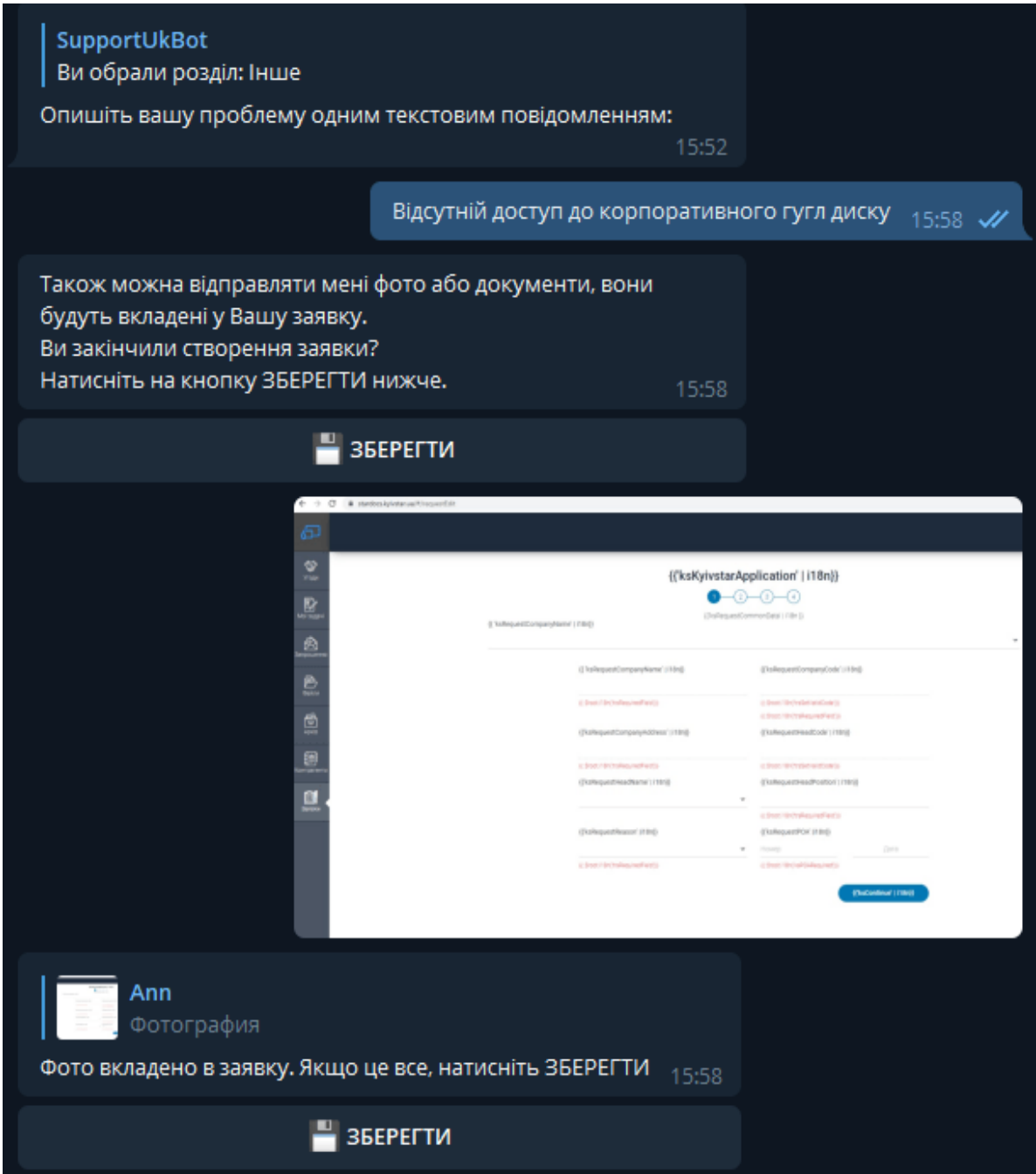


Рисунок 3.44 – Процес залишення заявки в розділі «Інше»

На рисунку 3.45 наведено підтвердження про успішне залишення заявки в розділі «Інше». Бот відповідає підтвердженням на останнє повідомлення користувача.

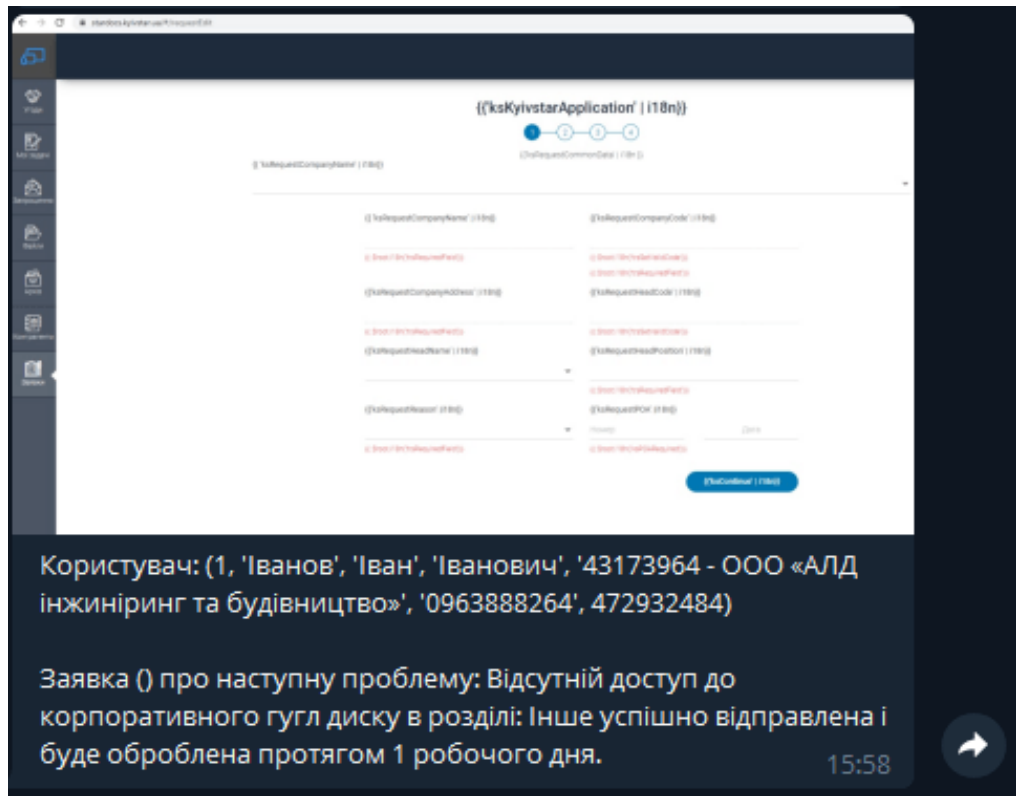


Рисунок 3.45 – Підтвердження успішності залишення заявки

На рисунку 3.46 наведено приклад повідомлень заявки про проблему користувача, які надходять модератору.

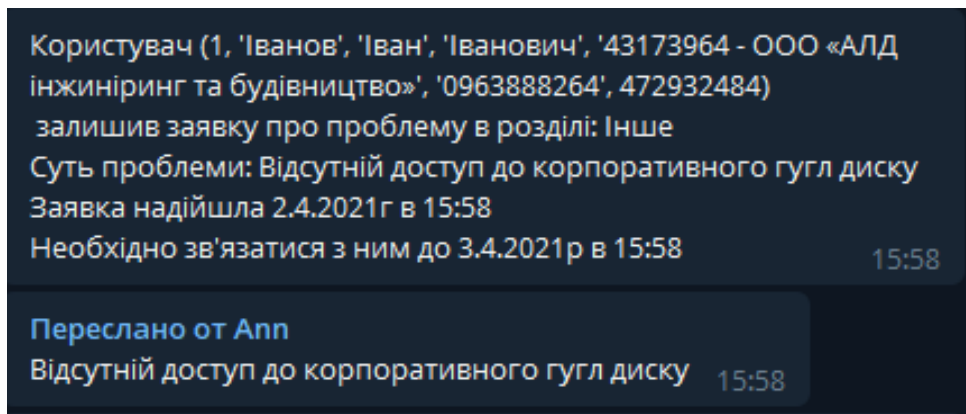


Рисунок 3.46 – Повідомлення про проблему користувача для модератора

Нижче на рисунку 3.47 наведено приклад залишеного користувачем для модератора документа.

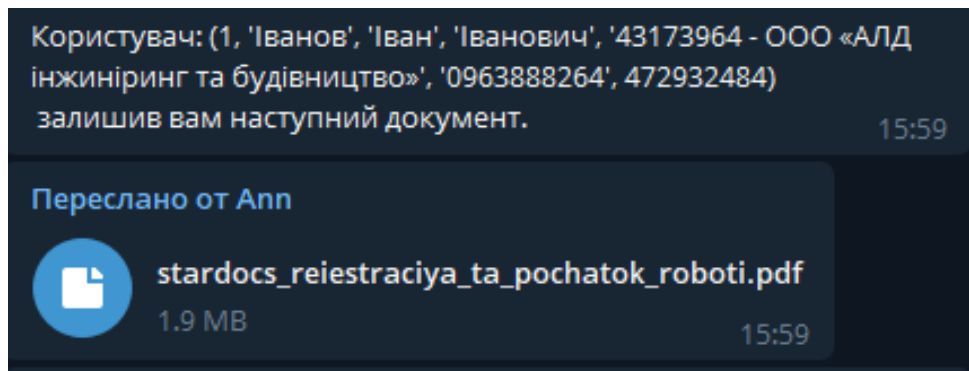


Рисунок 3.47 - Приклад залишеного користувачем для модератора документа

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Опис застосування

Найменування продукту – бот “SupportUkBot”.

Програмний продукт призначений для прийняття заявок про технічні проблеми підприємства. Область застосування – підприємства, що мають на балансі офісну техніку, корпоративне програмне забезпечення.

Дана розробка дає можливість користувачу залишити заявку про технічні проблеми з корпоративної технікою та/або програмним забезпеченням. Також користувач має доступ до інструкцій до корпоративного ПЗ, навчальних відео та застосунків корпоративного ПЗ.

4.2 Керівництво адміністратора

Адміністратору доступна функція відслідковування переписок користувачів з модератором. Також в чат з адміністратором надходять системні повідомлення про помилки. Адміністратору доступні всі команди модератора, що будуть описані в розділі 4.3.

4.3 Керівництво модератора

Команда «/help_mod». Для використання команди необхідно ввести «/help_mod» в чат з ботом та надіслати повідомлення. Бот надає наступним повідомленням інформацію про команди для поточного користувача.

Команда «/add». Для використання команди необхідно ввести «/add» в чат з ботом та надіслати повідомлення. Бот пропонує ввести прізвище користувача або номер телефону. Модератор надає інформацію в тестовому повідомленні. Бот надає результати пошуку та пропонує ввести номер шуканого користувача зі списку результатів. Модератор вводить номер

шуканого користувача зі списку результатів в текстовому повідомленні. Бот пропонує ввести telegram_id користувача, якого необхідно зареєструвати. Модератор вводить telegram_id користувача текстовим повідомленням, яке отримує з відповідної заявки на реєстрацію. Бот пропонує ввести компанію користувача. Модератор вводить компанію користувача текстовим повідомленням, яку отримує з відповідної заявки на реєстрацію. Бот надає підтвердження успішності виконання команди.

Команда «/show». Для використання команди необхідно ввести «/show» в чат з ботом та надіслати повідомлення. Бот надає наступним повідомленням інформацію про всіх зареєстрованих користувачів.

Команда «/fail». Для використання команди необхідно ввести «/fail» в чат з ботом та надіслати повідомлення. Бот пропонує ввести номер користувача, отриманий із заявки на реєстрацію. Модератор вводить номер користувача. Бот надає підтвердження успішності виконання команди.

Команда «/remove». Для використання команди необхідно ввести «/remove» в чат з ботом та надіслати повідомлення. Бот пропонує ввести номер користувача, отриманий із переліку зареєстрованих користувачів. Модератор вводить номер користувача. Бот надає підтвердження успішності виконання команди.

Команда «/answer». Для використання команди необхідно ввести «/answer» в чат з ботом та надіслати повідомлення. Бот пропонує обрати повідомлення користувача для відповіді, засобами Telegram необхідно натиснути «відповісти». Модератор вводить повідомлення для користувача та відправляє повідомлення. Бот надає підтвердження успішності виконання команди.

Команда «/answer_by_id». Для використання команди необхідно ввести «/answer_by_id» в чат з ботом та надіслати повідомлення. Бот пропонує ввести telegram_id користувача для відповіді. Модератор вводить telegram_id користувача та відправляє текстовим повідомленням. Бот пропонує ввести повідомлення для користувача. Модератор вводить

повідомлення та відправляє текстовим повідомленням. Бот надає підтвердження успішності виконання команди.

Команда «/send_all». Для використання команди необхідно ввести «/send_all» в чат з ботом та надіслати повідомлення. Бот пропонує ввести повідомлення. Модератор вводить текстове повідомлення та відправляє. Бот надає підтвердження успішності виконання команди.

Команда «/get_message». Для використання команди необхідно ввести «/get_message» в чат з ботом та надіслати повідомлення. Бот пропонує ввести id чату та id повідомлення в форматі «<id чату>пробіл<id повідомлення>», наприклад «472932484 1234». Модератор вводить шукану інформацію текстовим повідомленням та відправляє. Бот надає повідомлення, яке необхідно відновити.

4.3 Керівництво користувача

Команда «/start». Для використання команди необхідно ввести «/start» в чат з ботом та надіслати повідомлення. Бот надає повідомлення з основною клавіатурою.

Команда «/help». Для використання команди необхідно ввести «/help» в чат з ботом та надіслати повідомлення. Бот надає повідомлення з інформацією про команди для поточного користувача.

Команда «/registration». Для використання команди необхідно ввести «/registration» в чат з ботом та надіслати повідомлення. Бот пропонує обрати компанію, до якої належить користувач, натисканням на відповідну кнопку клавіатури. Користувач обирає компанію, натисканням на відповідну кнопку повідомлення. Бот пропонує підтвердити обрану компанію натисканням на кнопки «ТАК» чи «НІ» клавіатури. У разі помилки користувач не підтверджує компанію та може знову обрати компанію. У разі підтвердження бот пропонує надати номер телефону користувача, натисканням на кнопку в області клавіатури під полем введення повідомлень. Користувач натискає

кнопку надання номеру телефону. Telegram надає попередження, у якому пропонує підтвердити надання номеру телефону цьому боту. Користувач підтверджує надання номеру телефону, натисканням на кнопку «ПОДІЛИТИСЯ». Бот надає підтвердження успішності виконання команди. У разі схвалення заявки на реєстрацію модератором користувач отримує повідомлення з клавіатурою головного меню. У разі відхилення заявки користувач отримує відповідне повідомлення.

Команда «/exit». Для використання команди необхідно ввести «/exit» в чат з ботом та надіслати повідомлення. Бот надає підтвердження успішності виконання команди. У разі схвалення заявки на видалення користувач отримує відповідне повідомлення.

Натисканням на відповідні кнопки клавіатури головного меню, користувач обирає розділ, що відповідає характеру його проблеми. В загальному випадку бот пропонує описати проблему текстовим повідомленням. Користувач описує проблему текстовим повідомленням та надсилає його. Бот пропонує зберегти заявку натисканням на відповідну кнопку, або прикріпити до заявки фото чи документ. Користувач за потреби прикріплює фото чи документ, відправляючи останній в чат з ботом, бот надає підтвердження прикріплення фото чи документу до заявки користувачу. Натисканням на кнопку «ЗБЕРЕГТИ» користувач завершує створення заявки. Бот надає підтвердження успішності виконання команди.

4.4 Тестування

Прийнято рішення про тестування виконання всіх команд для користувача зареєстрованого, користувача незареєстрованого, модератора, адміністратора. Також протестовано функції залишення заявки, отримання повідомлень про помилки, відстеження переписок користувача та модератора. У таблиці 4.1 наведено результати тестування програмного продукту.

Таблиця 4.1 – Результати тестування програмного продукту

	Користувач незареєстрований	Користувач зареєстрований	Модератор	Адміністратор
/start	+	+	+	+
/help	+	+	+	+
/registration	+	+	+	+
/exit	+	+	+	+
/menu	+	+	+	+
/help_mod	+	+	+	+
/auth	+	+	+	+
/reauth	+	+	+	+
/add	+	+	+	+
/show	+	+	+	+
/fail	+	+	+	+
/remove	+	+	+	+
/answer	+	+	+	+
/answer_by_id	+	+	+	+
/send_all	+	+	+	+
/send_by_id	+	+	+	+
/get_message	+	+	+	+
Залишення заявки	+	+	+	+
Отримання повідомлень про помилки	+	+	+	+
Відстеження переписок користувача та модератора	+	+	+	+

Для користувача незареєстрованого вважається успішним результатом перевірки присутній доступ та коректне виконання команд «start», «help», «registration» та відсутність доступу до всіх інших команд та функцій.

Для користувача зареєстрованого вважається успішним результатом перевірки присутній доступ та коректне виконання команд «start», «help», «exit», «menu», можливість залишення заявки та відсутність доступу до всіх інших команд та функцій.

Для модератора вважається успішним результатом перевірки присутній доступ та коректне виконання всіх команд, окрім отримання

повідомлень про помилки та відстеження переписок користувача та модератора.

Для адміністратора вважається успішним результатом перевірки присутній доступ та коректне виконання всіх команд та функцій.

Відповідно до проведених тестів, програмне забезпечення працює коректно та стабільно, всі тести пройдені успішно.

ВИСНОВКИ

З метою автоматизації подання та обробки заявок про технічні проблеми до ІТ-служби підприємства прийнято рішення про розробку Telegram-боту.

Під час розробки Telegram-боту для дипломного проєкту були проаналізовані сучасні технології створення та розробки ботів для платформи.

Для проєктування була обрана платформа Telegram, бо відповідала вимогам до функціоналу, та є досить розповсюдженою серед цільової аудиторії. В якості мови програмування обрано Python, бо відповідала основним вимогам та можливостям, що висувалися до мови програмування.

Під час розробки програмного коду були вивчені та застосовані:

- обробники події введення команди;
- робота з inline та reply-клавіатурами;
- робота з динамічною пам'яттю;
- робота з розмежуванням прав відповідно до типів користувачів;
- запити до бази даних;
- взаємодія з іншим програмним продуктом.

Створений проєкт відповідає меті розробки, всім вимогам, що були пред'явлені до програмного засобу.

Повний код проєкту наведено у додатку А.

Також програмний продукт пройшов всі тести: для тестування було обрано команди для всіх типів користувачів, функції залишення заявки, відстеження переписок користувача та модератора, відстеження системних помилок.

Передбачено зручний інтерфейс користувача з набором клавіатур для обирання відповідного розділу, передбачено введення помилкових даних та невірних дій користувача.

За темою роботи є 1 наукова публікація, а саме тези доповіді на науково-практичній конференції «Тиждень науки», яка дистанційно проводилася у Національному університеті «Запорізька політехніка» у квітні 2021 року.

Шляхи майбутнього удосконалення:

- керування ботом з іншого програмного продукту технічної підтримки;
- автоматичне або автоматизоване розподілення заявок на відповідальних фахівців в залежності від категорії проблеми;
- впровадження штучного інтелекту в процес спілкування з користувачем;
- розширення кола вирішуваних проблем.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Вічева, Г. Програмна реалізація телеграм-бота служби технічної підтримки [Текст] /Г. Вічева, Т. Каплієнко // Тиждень науки-2021 : щорічна науково-практична конференція викладачів, науковців, молодих учених, аспірантів та студентів, Запоріжжя, 19-23 квітня 2021 р : proceedings. – Запоріжжя : подано до друку.
2. Khan, R. Build Better Chatbots [Text] / R. Khan, D. Anik. – Apress, 2018. – 109 p.
3. Janarthanam, S. Hands-On Chatbots and Conversational UI Development [Text] / S. Janarthanam. – Apress, 2019. – 394 p.
4. Рытикова, В. Разработка чат-бота помощника на языке Python для ускорения работы техподдержки компании СКБ Контур [Электрон. ресурс] / В. Рытикова. – Режим доступа: https://elar.urfu.ru/bitstream/10995/93466/1/m_th_v.o.rytikova_2020.pdf.
5. Чат-боты – кто они и что умеют? [Электрон. ресурс] / – Режим доступа: <https://efsol.ru/articles/messendzhery-i-chat-boty-dlya-biznesa-dostavki.html>.
6. McNeal, M, Streamlining Information Services Using Chatbots (Library Technology Reports) [Text] / M. McNeal, D. Newyear. - Amer Library Assn, 2014.
7. Как использовать чат-боты в бизнесе: 5 идей и 5 кейсов [Электрон. ресурс] / – Режим доступа: <https://vc.ru/services/93850-kak-ispolzovat-chat-boty-v-biznese-5-idey-i-5-keysov>.
8. Raj, S. Building Chatbots with Python: Using Natural Language Processing and Machine Learning [Text] / S. Raj. – Bangalore : Apress, 2018. – 205 p.
9. Mujassim, N. Chat bot Marketing Mastery II [Text] / N. Mujassim. 2020. – 13 p.
10. Krishna, G. The Best Interface Is No Interface [Text] / G. Krishna,

2015.

11. Не соцсети, а чат-бот: зачем бизнесу мессенджеры [Электрон. ресурс] / – Режим доступа: <https://igate.com.ua/news/24336-ne-sotsseti-a-chatbot-zachem-biznesu-messendzhery>.

12. Примеры использования чат-ботов в бизнесе [Электрон. ресурс] / – Режим доступа: <https://vc.ru/flood/25197-business-bot>.

13. Каршеринг-сервис YouDrive запустил бота-помощника в Telegram [Электрон. ресурс] / – Режим доступа: https://www.cnews.ru/news/line/2016-03-30_karsheringservis_youdrive_zapustil_botapomoshchnika.

14. Бот поиска аномальных цен на авиабилеты [Электрон. ресурс] / – Режим доступа: <https://bot.aviasales.ru/>.

15. Почему из всех языков программирования стоит выбрать именно Python [Электрон. ресурс] / – Режим доступа: <https://pythonist.ru/pochemu-iz-vseh-yazykov-programmirovaniya-stoit-vybrat-imenno-python/>.

16. Основные различия между Viber и Telegram и как тестировать боты для них [Электрон. ресурс] / – Режим доступа: <https://blog.ithillel.ua/articles/osnovnye-razlichija-mezhdu-viber-i-telegram-i-kak-testirovat-boty-dlja-nih>.

ДОДАТОК А
Текст програми

```

import pygame
import telebot
from telebot import types # підключення доповнень
import config
import datetime
from datetime import datetime, timedelta
import mysql.connector
from mysql.connector import Error
import os.path
import json
import pygame

bot = telebot.TeleBot(config.token)

TEAM_USER_LOGGING = 0
TEAM_USER_ACCEPTED = 1
ADD_USER_MOD_ACCEPT = 2
REMOVE_USER_MOD_ACCEPT = 3
FAIL_USER_MOD_ACCEPT = 4
ADD_USER_SUR_MOD_ACCEPT = 5
SEND_ALL_USER_MOD = 11
ADD_USER_ID_MOD_ACCEPT = 6
ADD_USER_TEL_ID_MOD_ACCEPT = 7
ADD_USER_COMPANY_MOD_ACCEPT = 10
GET_NEED_INSTALL_PC = 8
ADD_USER_PROBLEM = 9
ADD_USER_POSITION = 12
input_user_id_for_update_add_mod = ""
what_account = ""
action_account = ""
install_need_where = ""
current_title = ""

user_step = {}

# змінні для зберігання поточної заявки
current_call_code = ""
current_call_status = False
current_call_list = []
current_call_comment_list = []
current_call_chat_id = None
current_call_text = ""

current_id_call_from_support = "" # змінна для зберігання id заявки, яка була додана в
support_call
answer_topic_number = "" # змінна для зберігання номера поточної заявки, за якою
спілкується користувач
topic_for_insert_answer = None # змінна для зберігання номера поточної заявки, на яку
відповідає модератор
answer_telegram_id = "" # змінна для зберігання id користувача, якому відповідає
модератор при відповіді за id
send_telegram_id = "" # змінна для зберігання id користувача, якому пише модератор при
розсилці за id

name = ""
surname = ""

```

```

company = "
age = 0
once_replay_main_menu_flag = False
register_exit_flag = "
forward_message_to_call_flag = False
get_account_call_flag = False
confirm_company_flag = False
confirm_reauth_flag = False
confirm_add_user_flag = False
message_contact = None # номер телефону користувача
exit_confirm_user_flag = False
dont_five_status_flag = False # прапор, що означає заявку у вигляді картинки / документа
# при запиті доступу (статус не 5 при вставці в проміжну таблицю)
flag = 0
contact_flag = False
auth_login_mod_regular = "7" # example for auth_moderator and command
auth_password_mod_regular = "7" # example for auth_moderator
auth_password_mod_user = " # variable for data user for moderator auth
auth_login_mod_user = " # variable for data user for moderator auth
moderator_chat_id = [472932484] # 472932484 here is my/to be elena-924915753/253068519-
soboleva/here list moderation
expection = ['Головне меню', '/start', '/menu', '/help', '/id', '/ID', '/iD', '/Id', '/contact', '/answer',
            '/answer_by_id', '/send_all', '/registration', '/exit', '/help_mod', '/auth', '/reauth',
            '/add', '/show', '/fail', '/remove', '/get_message', '/send_by_id']

def start_command(message):
    # train_sql(message)
    markup = types.InlineKeyboardMarkup(row_width=2) # register/exit
    item1 = types.InlineKeyboardButton("✍ РЕЄСТРАЦІЯ",
callback_data='registr_user_command')
    item2 = types.InlineKeyboardButton("✕ ВІДПИСАТИСЯ",
callback_data='exit_user_command')
    item3 = types.InlineKeyboardButton("🔔 ГОЛОВНЕ МЕНЮ",
callback_data='main_menu_command')
    markup.add(item1, item2, item3)

    bot.send_message(message.chat.id, 'Ласкаво просимо. Я ваш персональний помічник. '
    'Щоб використовувати всі мої можливості, вам необхідно
зареєструватися. '
    'Так само ви можете відписатися', reply_markup=markup)
    global flag, register_exit_flag
    register_exit_flag = "
    flag = 0
    error_company_method()

def is_moderator(user):
    global moderator_chat_id, auth_login_mod_user, auth_password_mod_user
    global auth_login_mod_regular, auth_password_mod_regular
    if user in moderator_chat_id:
        return True
    else:
        if auth_login_mod_user == auth_login_mod_regular and auth_password_mod_user ==
auth_password_mod_regular:
            return True
        else:

```

```

return False

# метод запиту введення проблеми користувача
def get_user_problem(message):
    global flag, current_call_status
    if is_register_user(message.chat.id):
        current_call_status = True
        user_step[message.chat.id] = ADD_USER_PROBLEM
        bot.reply_to(message, "Опишіть вашу проблему одним текстовим повідомленням:")
    else:
        bot.send_message(message.chat.id, 'Для доступу до даної команди спочатку необхідно зареєструватися '
                                     'і дочекатися схвалення заявки.')
        flag = 0
        error_company_method()

# метод обробки введення користувачем проблеми
@bot.message_handler(func=lambda message: user_step.get(message.chat.id) ==
ADD_USER_PROBLEM)
def get_problem_user(message):
    global flag, current_title, exeption, current_call_chat_id, current_call_text, current_call_code
    # bot.send_message(message.chat.id, 'get_problem_user')
    if message.text != "":
        if message.text in exeption:
            # print("here888")
            user_step[message.chat.id] = TEAM_USER_ACCEPTED
            flag = 10
            external_error(message)
        else:
            now = datetime.now()
            period = timedelta(hours=24)
            before = now + period
            if get_user_info_from_db_by_tel_id(message) != "":
                for chat_id in moderator_chat_id:
                    bot.send_message(chat_id,
                                     'Користувач ' +
                                     get_user_info_from_db_by_tel_id(message) +
                                     ' залишив заявку про проблему в розділі: ' + current_title +
                                     # "\nКомпанія: " + company +
                                     "\nСуть проблеми: " + message.text +
                                     "\nЗаявка надійшла " + str(now.day) + '!' + str(now.month) + '!' +
                                     str(now.year)
                                     + 'Г в ' + str(now.hour) + ':' + str(now.minute) +
                                     "\nНеобхідно зв'язатися з ним до " + str(before.day) + '!' +
                                     str(before.month)
                                     + '!' + str(before.year) + 'П в ' + str(before.hour) + ':' + str(before.minute)
                                     )
                    bot.forward_message(chat_id, message.chat.id, message.message_id)
            markup = types.InlineKeyboardMarkup(row_width=1) # save call
            item1 = types.InlineKeyboardButton("📁 ЗБЕРЕГТИ", callback_data='save_call')
            markup.add(item1)
            bot.send_message(message.chat.id, 'Також можна відправляти мені фото або документи, '
                                     'вони будуть вкладені у Вашу заявку. \nВи закінчили створення заявки? '

```

```

        "\nНатисніть на кнопку ЗБЕРЕГТИ нижче.",
reply_markup=markup)

        current_call_chat_id = message.chat.id
        # bot.send_message(message.chat.id, current_call_text)
        # ++insert_call_to_db(get_user_not_str_info_from_db_by_tel_id(message), now,
current_title, message.text)
        insert_call_to_buffer_db(message.chat.id, current_call_code, message.message_id,
message.date, ", ")
        current_call_text = 'Заявка (' + str(current_call_code) + ') про наступну проблему: '
+ \
            message.text + ' в розділі: ' + current_title + \
            ' успішно відправлена і буде оброблена протягом 1 робочого дня.'
        # print("before insert_call_to_buffer_db message = " + str(message))
        user_step[message.chat.id] = TEAM_USER_ACCEPTED
    else:
        bot.send_message(message.chat.id, 'Помилка відправки заявки про проблему: ' +
message.text +
            ' в розділі: ' + current_title)
        for chat_id in moderator_chat_id:
            bot.send_message(chat_id, 'Виникла помилка для користувача ' +
str(message.chat.id) +
                '\nКомпанія: ' + company +
                '. Він намагався відправити заявку про проблему: ' + message.text +
                '\nВ розділі: ' + current_title +
                '\nПомилка заявки надійшла ' + str(now.day) + ' '
                + str(now.month) + ' ' + str(now.year)
                + ' г в ' + str(now.hour) + ':' + str(now.minute)
                )
            bot.forward_message(chat_id, message.chat.id, message.message_id)
        flag = 0
        current_title = ""

# Default command handler. A lambda expression which always returns True is used for this
purpose.
@bot.message_handler(content_types=['audio', 'video', 'location', 'sticker', 'voice', 'venue'])
def default_command(message):
    bot.send_message(message.chat.id, 'Припиніть мені відправляти таке різне!! Я всі покажу
модератору!!')
    for chat_id in moderator_chat_id:
        if is_register_user(message.chat.id):
            bot.send_message(chat_id, "Користувач " +
str(get_user_info_from_db_by_tel_id(message)) +
                " не впорався з ботом та залишив вам це:")
            bot.forward_message(chat_id, message.chat.id, message.message_id)

# метод зовн.викликаємий - обертка для рекурсивної відправки повідомлень користувача
після відповіді модератора на заявку,
# та користувач виявив бажання продовжити переписку
def get_message_user_to_forward_mod(message):
    global forward_message_to_call_flag, flag, exeption
    if forward_message_to_call_flag:
        # if message.text in exeption:
        # send_continue_close_call(message.chat.id)
        # else:

```

```

        bot.register_next_step_handler(message, method_get_message_user_to_forward_mod)
    else:
        flag = 0
        error_company_method()

# метод рекурсивної відправки повідомлень користувача після відповіді модератора на
заявку,
# та користувач виявив бажання продовжити переписку
def method_get_message_user_to_forward_mod(message):
    global forward_message_to_call_flag, flag, exeption, answer_topic_number
    if forward_message_to_call_flag:
        if message.text in exeption:
            send_continue_close_call(message.chat.id, False)
        else:
            for chat_id in moderator_chat_id:
                bot.send_message(chat_id, 'Користувач ' +
get_user_info_from_db_by_tel_id(message) +
                'залишив вам наступне повідомлення в рамках заявки')
                bot.forward_message(chat_id, message.chat.id, message.message_id)
            if message.content_type == 'photo':
                method_save_photo_in_comm(message)
                print('this photo')
            if message.content_type == 'document':
                method_save_document_in_comm(message)
                print('this is document')
            #++insert_call_to_db_comm(answer_topic_number, message)
            insert_call_to_buffer_db(message.chat.id, answer_topic_number,
                message.message_id, message.date, "1", ")
            bot.send_message(message.chat.id, 'Ваше повідомлення доставлено модератору,
очікуйте...'
                '\nУ вас буде можливість закрити заявку після відповіді
модератора.')
            bot.register_next_step_handler(message, method_get_message_user_to_forward_mod)
    else:
        flag = 0
        error_company_method()

@bot.message_handler(commands=['send_by_id'])
def send_by_id(message):
    global flag, auth_login_mod_user, auth_password_mod_user, moderator_chat_id,
send_telegram_id
    # if auth_login_mod_user != " and auth_password_mod_user != ":
    if is_moderator(message.chat.id):
        send_telegram_id = "
        bot.send_message(message.chat.id, 'Введіть telegram_id користувача, якому
напишемо:')
        bot.register_next_step_handler(message, get_id_for_send)
    else:
        bot.send_message(message.chat.id, 'Ви не є модератором для виконання цієї команди!')
        flag = 4

def get_id_for_send(message):
    global send_telegram_id
    if message.text != "":

```

```

        send_telegram_id = message.text
        bot.send_message(message.chat.id, 'Введіть повідомлення, яке напишемо
користувачу:')
        bot.register_next_step_handler(message, get_text_for_send)

def get_text_for_send(message):
    global send_telegram_id, flag
    if message.text != "":
        try: # якщо додається вірний id(вдається відправити на нього повідомлення) то
            відбувається обробка
            # інакше помилка(передбачено обробку зі спамом модератору)
            bot.send_message(send_telegram_id, message.text)
            bot.send_message(message.chat.id, 'Повідомлення *' + message.text + '* доставлено.')
            send_telegram_id = "
        except BaseException as e:
            bot.send_message(message.chat.id, 'Помилка - ' + str(e) + ' \nНе вдалося відправити
повідомлення '
                                'на вказаний id користувача. Перевірте, будь
ласка, '
                                'правильність наступного id: ' +
                                answer_telegram_id)
            flag = 4

@bot.message_handler(commands=['answer_by_id'])
def answer_by_id(message):
    global flag, auth_login_mod_user, auth_password_mod_user, moderator_chat_id,
answer_telegram_id
    # if auth_login_mod_user != " and auth_password_mod_user != "":
    if is_moderator(message.chat.id):
        answer_telegram_id = "
        bot.send_message(message.chat.id, 'Введіть telegram_id користувача, якому
напишемо:')
        bot.register_next_step_handler(message, get_id_for_answer)
    else:
        bot.send_message(message.chat.id, 'Ви не є модератором для виконання цієї команди!')
        flag = 4

def get_id_for_answer(message):
    global answer_telegram_id
    if message.text != "":
        answer_telegram_id = message.text
        bot.send_message(message.chat.id, 'Введіть повідомлення, яке напишемо
користувачу:')
        bot.register_next_step_handler(message, get_text_for_answer)

def get_text_for_answer(message):
    global answer_telegram_id, flag
    if message.text != "":
        try: # якщо додається вірний id(вдається відправити на нього повідомлення) то
            відбувається обробка
            # інакше помилка(передбачено обробку зі спамом модератору)
            bot.send_message(answer_telegram_id, message.text)
            bot.send_message(message.chat.id, 'Повідомлення *' + message.text + '* доставлено.')

```

```

# print(" id answer_telegram_id = " + str(answer_telegram_id))
# print(" id message.chat.id = " + str(message.chat.id))
# print(" id message.id = " + str(message.chat))
update_once_call_to_buffer_db(answer_telegram_id, ", ",
                               message.chat.id, message)
#++insert_comment_to_db(answer_telegram_id, ", message.chat.id, message)
# here maybe problem
send_continue_close_call(
    answer_telegram_id,
    get_once_call_from_buffer_db(
        answer_telegram_id,
        ", "
    )
)

answer_telegram_id = "
except BaseException as e:
    bot.send_message(message.chat.id, 'Помилка - ' + str(e) + ' \nНе вдалося відправити
повідомлення '
                                'на вказаний id користувача. Перевірте, будь
ласка, '
                                'правильність наступного id: ' +
answer_telegram_id)
    flag = 4

@bot.message_handler(commands=['answer'])
def process_start(message):
    global flag, auth_login_mod_user, auth_password_mod_user, moderator_chat_id
    # if auth_login_mod_user != " and auth_password_mod_user != "":
    if is_moderator(message.chat.id):
        try:
            bot.send_message(message.chat.id, 'Тепер відповідайте на повідомлення
користувача')
            # bot.forward_message(moderator_chat_id[0], message.chat.id, message.message_id)
            bot.register_next_step_handler(message, process_mind)
        except:
            bot.send_message(moderator_chat_id[0], "Помилка виникла в блоці коду:")

    "\n<code>@bot.message_handler(commands=['send_message'])</code>",
        parse_mode='HTML')
    else:
        bot.send_message(message.chat.id, 'Ви не є модератором для виконання цієї команди!')
        flag = 4

# метод обробки відправки повідомлення з клавіатурою для продовження спілкування з
модератором
# після подачі заявки та отримання відповіді або закриття заявки
# flagok - True - 2 buttons, False - 1 button
def send_continue_close_call(chat_id, flagok):
    global topic_for_insert_answer
    # print("after chat_id = " + str(chat_id))
    # print("after flagok = " + str(flagok))
    # print("after topic_for_insert_answer = " + str(topic_for_insert_answer))
    string2 = str(topic_for_insert_answer)
    # string1 = string.split("[")[1]

```

```

# string2 = string1.split(" ")[0]
# print("after string2 = " + str(string2))
if flagok:
    markup = types.InlineKeyboardMarkup(row_width=2)
    item1 = types.InlineKeyboardButton("🗨️ ПРОДОВЖИТИ",
callback_data='continue_call')
    item2 = types.InlineKeyboardButton("✅ ЗАКРИТИ ЗАЯВКУ",
callback_data='close_call')
    markup.add(item1, item2)
    bot.send_message(chat_id, 'Отримано відповідь модератора на вашу заявку (' + string2
+ '). '
                                'Залишилися питання? Натисніть
ПРОДОВЖИТИ та вводьте повідомлення для модератора в цей чат.'
                                '\nЯкщо Ваша проблема була
вирішена, '
                                'закрийте заявку відповідною
кнопкою. ',
                                reply_markup=markup)
else:
    markup = types.InlineKeyboardMarkup(row_width=1)
    item1 = types.InlineKeyboardButton("✅ ЗАКРИТИ ЗАЯВКУ",
callback_data='close_call')
    markup.add(item1)
    bot.send_message(chat_id, 'Залишилися питання? Вводьте повідомлення для
модератора в цей чат.'
                                '\nЯкщо Ваша проблема (' + string2 + ') була вирішена, '
                                'закрийте заявку відповідною кнопкою. ',
                                reply_markup=markup)
    topic_for_insert_answer = None

def process_mind(message):
    global flag, auth_login_mod_user, auth_password_mod_user, moderator_chat_id
    # if auth_login_mod_user != " and auth_password_mod_user != ":
    if is_moderator(message.chat.id):
        try:
            text = 'Повідомлення *' + message.text + '* було відправлено користувачу ' + \
                str(message.reply_to_message.forward_from.first_name)
            # bot.send_message(message.reply_to_message.forward_from.id, message.text)
            bot.forward_message(message.reply_to_message.forward_from.id, message.chat.id,
message.message_id)

            # print('before message.reply_to_message.forward_from.id = ' +
str(message.reply_to_message.forward_from.id))
            # print('before message.reply_to_message.id = ' + str(message.reply_to_message.id-4))
            # print('before message.chat.id = ' + str(message.chat.id))
            # print('before message.message_id = ' + str(message.message_id))
            # print('before update_once_call_to_buffer_db message = ' +
str(message.reply_to_message))
            update_once_call_to_buffer_db(message.reply_to_message.forward_from.id,
                # message.reply_to_message.id-4,
                message.reply_to_message.id,
                message.reply_to_message.forward_date,
                message.chat.id,
                message)
            #++insert_comment_to_db(message.reply_to_message.forward_from.id,
message.reply_to_message.forward_date,

```

```

    #++          message.chat.id, message)
    send_continue_close_call(
        message.reply_to_message.forward_from.id,
        get_once_call_from_buffer_db(
            message.reply_to_message.forward_from.id,
            message.reply_to_message.id,
            message.reply_to_message.forward_date
        )
    )
    for user in moderator_chat_id:
        bot.send_message(user, text)
    except Exception as e:
        bot.send_message(moderator_chat_id[0], 'Бот продовжив свою роботу.' +
            ' Помилка ' + str(e) + 'сталася в блоці коду:\n\n <code>def
process_mind(message)</code>',
            parse_mode='HTML')
    else:
        bot.send_message(message.chat.id, 'Ви не є модератором для виконання цієї команди!')
    flag = 4

# метод запиту введення ПЗ, яке необхідно встановити користувачу
def get_user_need_install_pc(message):
    global flag, current_call_status
    if is_register_user(message.chat.id):
        current_call_status = True
        user_step[message.chat.id] = GET_NEED_INSTALL_PC
        bot.reply_to(message, "Введіть програмне забезпечення, яке вам необхідно
встановити:")
    else:
        bot.send_message(message.chat.id, 'Для доступу до даної команди спочатку необхідно
zareestruvatisia '
            'та дочекатись схвалення заявки.')
        flag = 0
        error_company_method()

# метод отримання статусу заявки (once) з проміжної таблиці support_call_bot_log
def get_once_call_from_buffer_db(telegram_id, message_id, message_date):
    # print('telegram_id = ' + str(telegram_id))
    # print('message_id = ' + str(message_id))
    # print('message_date = ' + str(message_date))
    connection = create_connection()
    if connection is not None:
        # if message_id != "":
        #     select_once_query = "select once from support_call_bot_log where telegram_id=" +
str(telegram_id) \
        #         + " and topic_call=" \
        #         "(select topic_call FROM support_call_bot_log WHERE " \
        #         "telegram_id=" + str(telegram_id) + " AND date_telegram=" + \
        #         str(message_date) + " ORDER BY id DESC LIMIT 1) ORDER BY
id"
        # else:
        #     select_once_query = "select once from support_call_bot_log where telegram_id=" +
str(telegram_id) \
        #         + " and topic_call=" \
        #         "(select topic_call FROM support_call_bot_log WHERE " \

```

```

#             "telegram_id=" + str(telegram_id) + " AND message_id IS NULL " \
#             "ORDER BY id DESC LIMIT 1) ORDER BY id"
if message_id != "":
    select_once_query = "select once from support_call_bot_log where telegram_id=" +
str(telegram_id) \
        + "and (status IS NULL OR status=4) and topic_call=" \
        + "(select topic_call FROM support_call_bot_log WHERE " \
        + "telegram_id=" + str(telegram_id) + " AND date_telegram=" + \
        str(message_date) + " ORDER BY id DESC LIMIT 1) ORDER BY id"
else:
    select_once_query = "select once from support_call_bot_log where telegram_id=" +
str(telegram_id) \
        + "and (status IS NULL OR status=4) and topic_call=" \
        + "(select topic_call FROM support_call_bot_log WHERE " \
        + "telegram_id=" + str(telegram_id) + " AND message_id IS NULL " \
        + "ORDER BY id DESC LIMIT 1) ORDER BY id"
support_call_topic = execute_read_query(connection, select_once_query)
support_call_topic_new = [] # костыльный переход(промежуточный массив та проход
за списку)
# від списку вида [(472932484,), (472932484,)]
# до списку вида [253068519, 472932484]
for e in support_call_topic:
    support_call_topic_new.append(e[0]) # тут останній номер заявки/теми
support_call_topic_next = support_call_topic_new[0]
# print('support_call_topic_new.lenght=' + str(len(support_call_topic_new)))
# + "\nsupport_call_topic_next=" + str(support_call_topic_next)
# + "\nsupport_call_topic_new mass=" + str(support_call_topic_new))
if support_call_topic_next == 1:
    if len(support_call_topic_new) > 1:
        # print("return False1")
        return False
    else:
        # print("return True1")
        return True
else:
    # print("return True2")
    return True

# метод оновлення статусу заявки (status) в таблиці support_call/support_topic
# статус 4 - закрыта заявка
def update_status_call_to_support_db(topic_call):
    connection = create_connection()
    if connection is not None:
        # print('topic_call = ' + str(topic_call))
        update_call = "UPDATE support_call SET support_call.type='4' \
            " WHERE support_call.topic=" + str(topic_call) + ""
        execute_query(connection, update_call)
        update_topic = "UPDATE support_topic SET support_topic.status='4' \
            " WHERE support_topic.code=" + str(topic_call) + ""
        execute_query(connection, update_topic)

# метод оновлення статусу заявки (status) в проміжній таблиці support_call_bot_log
# статус 4 - закрыта заявка
def update_status_call_to_buffer_db(telegram_id, topic_call):
    connection = create_connection()

```

```

if connection is not None:
    # print('telegram_id = ' + str(telegram_id))
    # print('topic_call = ' + str(topic_call))
    update_call = "UPDATE support_call_bot_log SET support_call_bot_log.status='4' \
        " WHERE support_call_bot_log.topic_call='" + str(topic_call) + \
        "' AND support_call_bot_log.telegram_id='" + str(telegram_id) + "'"
    execute_query(connection, update_call)

# метод повідомлення модератора про закриття заявки користувачем
def notificate_close_call_for_mod(topic_call, user_chat):
    connection = create_connection()
    if connection is not None:
        support_call_resp_text_query = "SELECT text, responsible FROM support_call WHERE
topic='" + str(topic_call) + "' ORDER BY id DESC LIMIT 1"
        support_resp_text = execute_read_query(connection, support_call_resp_text_query)
        support_text_new = []
        support_resp_new = []
        for e in support_resp_text:
            support_text_new.append(e[0])
            support_resp_new.append(e[1])
        support_text_next = support_text_new[0] # останній text
        support_resp_next = support_resp_new[0] # останній responsible
        # print('support_text_next = ' + str(support_text_next))
        # print('support_resp_next = ' + str(support_resp_next))
        support_author_query = "SELECT full_name FROM author WHERE id='" +
str(support_resp_next) + "' LIMIT 1"
        support_resp_name_text = execute_read_query(connection, support_author_query)
        support_user_new = []
        for e in support_resp_name_text:
            support_user_new.append(e[0])
        support_user_next = support_user_new[0] # останній full_name user`a
        support_topic_query = "SELECT title FROM support_topic WHERE code='" +
str(topic_call) + "' LIMIT 1"
        support_topic_text = execute_read_query(connection, support_topic_query)
        support_topic_new = []
        for e in support_topic_text:
            support_topic_new.append(e[0])
        support_topic_next = support_topic_new[0] # останній title topic
        for user in moderator_chat_id:
            bot.send_message(user, "Користувач " + str(support_user_next) + " закрив заявку (" \
                + str(topic_call) + ") в розділі " + str(support_topic_next) + \
                ". Зміст заявки: " + str(support_text_next) + ".")
            bot.send_message(user_chat, str(support_user_next) + ", ваша заявка (" \
                + str(topic_call) + ") " + str(support_text_next) + " в розділі " \
                + str(support_topic_next) + " була закрита. Дякуємо за звернення!")
        # update_call = "UPDATE support_call_bot_log SET support_call_bot_log.once='1' \
        # " WHERE support_call_bot_log.topic_call='" + str(support_call_topic_next)
        + "'"

        # execute_query(connection, update_call)
        # insert_query = "INSERT INTO `support_call_bot_log`(`telegram_id`, `topic_call`,
`once`) VALUES ('" + \
            str(telegram_id) + "', '" + str(topic_call) + "', '1')"
```

метод вставки відповіді модератора в таблицю support_call, support_comment

```

# telegram_id - користувацький id - від кого повідомлення в бот
# message_date - дата повідомлення користувача, що запитали
# telegram_id_mod - модераторський id - хто відповів на повідомлення в бот
# message_id_mod - повідомлення модератора, що відповіли
def insert_comment_to_db(telegram_id, message_date, telegram_id_mod, message_id_mod):
    # print('after telegram_id = ' + str(telegram_id))
    # print('after message_date = ' + str(message_date))
    # print('after telegram_id_mod = ' + str(telegram_id_mod))
    # print('after message_id_mod = ' + str(message_id_mod.message_id))
    global topic_for_insert_answer
    connection = create_connection()
    if connection is not None:
        if message_date != "":
            support_call_id_query = "SELECT support_call_id, topic_call FROM
support_call_bot_log WHERE telegram_id=" + str(telegram_id) + " AND date_telegram=" + str(message_date) + \
            " AND mod_telegram_id=" + str(telegram_id_mod) + \
            " AND mod_message_id=" + str(message_id_mod.message_id) + "
LIMIT 1"
        else:
            support_call_id_query = "SELECT support_call_id, topic_call FROM
support_call_bot_log WHERE telegram_id=" + str(telegram_id) + " AND mod_telegram_id=" + str(telegram_id_mod)
+ \
            " AND mod_message_id=" + str(message_id_mod.message_id) + "
LIMIT 1"
        support_call_id = execute_read_query(connection, support_call_id_query)
        support_call_id_new = []
        support_call_topic_new = []
        for e in support_call_id:
            support_call_id_new.append(e[0])
            support_call_topic_new.append(e[1])
        support_call_id_next = support_call_id_new[0] # останній номер id
        support_call_topic_next = support_call_topic_new[0] # останній номер заявки
        now = datetime.now()
        insert_comment_query = "INSERT INTO `support_comment`(`topic`, `comment`,
`create_at`) " +
            "VALUES (" + str(support_call_topic_next) + ", " +
str(message_id_mod.text) +
            + ", " + now.strftime('%Y-%m-%d %H:%M:%S') + ")"
        execute_query(connection, insert_comment_query)
        select_comment_id_query = "SELECT id FROM support_comment WHERE topic=" + \
            str(support_call_topic_next) + " AND comment=" + str(
            message_id_mod.text) + " LIMIT 1"
        support_comment_id = execute_read_query(connection, select_comment_id_query)
        support_comment_id_new = []
        for e in support_comment_id:
            support_comment_id_new.append(e[0])
        support_comment_id_next = support_comment_id_new[0] # останній номер id comment
        update_call = "UPDATE support_call SET support_call.comment=" +
str(support_comment_id_next) \
            + " WHERE support_call.id=" + str(support_call_id_next) + "
        execute_query(connection, update_call)

# метод оновлення статусу заявки (once) в проміжній таблиці support_call_bot_log
# telegram_id - користувацький id - від кого повідомлення в бот

```

```

# message_id - id повідомлення користувача, що запитали
# message_date - дата форварда повідомлення щоб впізнати його в базі
# telegram_id_mod - модераторський id - хто відповів на повідомлення в бот
# message_id_mod - повідомлення модератора, що відповіли
def update_once_call_to_buffer_db(telegram_id, message_id, message_date, telegram_id_mod,
message_id_mod):
    global topic_for_insert_answer
    connection = create_connection()
    if connection is not None:
        if message_id != "":
            support_call_topic_query = "SELECT topic_call FROM support_call_bot_log WHERE
telegram_id=" + str(telegram_id) + " AND date_telegram=" + str(message_date) + "
ORDER BY topic_call DESC LIMIT 1"
        else:
            support_call_topic_query = "SELECT topic_call FROM support_call_bot_log WHERE
telegram_id=" + str(telegram_id) + " ORDER BY topic_call DESC LIMIT 1"
        support_call_topic = execute_read_query(connection, support_call_topic_query)
        support_call_topic_new = []
        for e in support_call_topic:
            support_call_topic_new.append(e[0]) # останній номер заявки/теми
        support_call_topic_next = support_call_topic_new[0]
        topic_for_insert_answer = support_call_topic_next
        if message_id != "":
            update_call = "UPDATE support_call_bot_log SET support_call_bot_log.once='1', " +
"support_call_bot_log.mod_telegram_id=" + str(telegram_id_mod) +
"support_call_bot_log.mod_message_id=" +
str(message_id_mod.message_id) +
" WHERE support_call_bot_log.topic_call=" + str(support_call_topic_next)
+ \
" AND support_call_bot_log.date_telegram=" + str(message_date) + ""
        else:
            update_call = "UPDATE support_call_bot_log SET support_call_bot_log.once='1', " +
"support_call_bot_log.mod_telegram_id=" + str(telegram_id_mod) +
"support_call_bot_log.mod_message_id=" +
str(message_id_mod.message_id) +
" WHERE support_call_bot_log.topic_call=" + str(support_call_topic_next)
+ \
" AND support_call_bot_log.once IS NULL"
            execute_query(connection, update_call)

# метод вставки заявки в проміжну таблицю support_call_bot_log коли користувач додав
заявку
def insert_call_to_buffer_db(telegram_id, topic_call, message_id, message_date, child_flag,
status):
    global current_id_call_from_support
    connection = create_connection()
    if connection is not None:
        if message_id != "":
            if child_flag != "":
                if status != "":
                    insert_query = "INSERT INTO `support_call_bot_log`(`telegram_id`, `topic_call`,
`message_id`,`child`,`support_call_id`,`date_telegram`,`status`) VALUES (" + str(
telegram_id) + ", " + str(topic_call) + ", "

```

```

        + "", "" + str(message_id) + "", '1', "" + str(current_id_call_from_support) +
\
        "", "" + str(message_date) + "", "" + str(status) + "")"
    else:
        insert_query = "INSERT INTO `support_call_bot_log`(`telegram_id`, `topic_call`,
`message_id`, " \
            "`child`, `support_call_id`, `date_telegram`) VALUES (" + str(
            telegram_id) + "", "" + str(topic_call) \
            + "", "" + str(message_id) + "", '1', "" + str(current_id_call_from_support) +
\
            "", "" + str(message_date) + "")"
    else:
        if status != "":
            insert_query = "INSERT INTO `support_call_bot_log`(`telegram_id`, `topic_call`,
`message_id`, " \
                "`support_call_id`, `date_telegram`, `status`) VALUES (" + str(
                telegram_id) + "", "" + str(topic_call) + \
                "", "" + str(message_id) + "", "" + str(current_id_call_from_support) + \
                "", "" + str(message_date) + "", "" + str(status) + ")"
        else:
            insert_query = "INSERT INTO `support_call_bot_log`(`telegram_id`, `topic_call`,
`message_id`, " \
                "`support_call_id`, `date_telegram`) VALUES (" + str(telegram_id) + "",
"" + str(
                topic_call) + \
                "", "" + str(message_id) + "", "" + str(current_id_call_from_support) + \
                "", "" + str(message_date) + ")"
    else:
        if child_flag != "":
            if status != "":
                insert_query = "INSERT INTO `support_call_bot_log`(`telegram_id`, `topic_call`,
`child`, " \
                    "`support_call_id`, `status`) VALUES (" + str(telegram_id) + "", "" + str(
                    topic_call) + \
                    "", '1', "" + str(current_id_call_from_support) + "", "" + str(status) + ")"
            else:
                insert_query = "INSERT INTO `support_call_bot_log`(`telegram_id`, `topic_call`,
`child`, " \
                    "`support_call_id`) VALUES (" + str(telegram_id) + "", "" +
str(topic_call) + \
                    "", '1', "" + str(current_id_call_from_support) + "")"
        else:
            if status != "":
                insert_query = "INSERT INTO `support_call_bot_log`(`telegram_id`, `topic_call`,
" \
                    "`support_call_id`, `status`) VALUES (" + str(telegram_id) + "", "" + str(
                    topic_call) + \
                    "", "" + str(current_id_call_from_support) + "", "" + str(status) + ")"
            else:
                insert_query = "INSERT INTO `support_call_bot_log`(`telegram_id`, `topic_call`,
" \
                    "`support_call_id`) VALUES (" + str(telegram_id) + "", "" +
str(topic_call) + \
                    "", "" + str(current_id_call_from_support) + ")"
    execute_query(connection, insert_query)

```

```

# метод вставки заявки(вторичної) в базу даних техпідтримки
def insert_call_to_db_comm(topic, message):
    print("insert_call_to_db_comm message = " + str(message))
    global current_call_list, current_call_code, current_id_call_from_support,
    current_call_comment_list
    print("insert_call_to_db_comm current_call_comment_list = " +
    str(current_call_comment_list))
    current_id_call_from_support = "
    connection = create_connection()
    if connection is not None:
        support_call_id_resp_query = "SELECT id, responsible FROM support_call WHERE
topic="" \
                                + str(topic) + "" ORDER BY id LIMIT 1"
        support_resp_id = execute_read_query(connection, support_call_id_resp_query)
        support_id_new = []
        support_resp_new = []
        for e in support_resp_id:
            support_id_new.append(e[0])
            support_resp_new.append(e[1])
        support_id_next = support_id_new[0] # останній id заявки з support_call
        support_resp_next = support_resp_new[0] # останній responsible
        author_query = "SELECT full_name FROM author WHERE id="" +
str(support_resp_next) + "" LIMIT 1"
        result_author = execute_read_query(connection, author_query)
        result_author_new = []
        for e in result_author:
            result_author_new.append(e[0])
        author_name = result_author_new[0] # ПІБ автора
        insert_author = "INSERT INTO `author`(`full_name`) VALUES (" + str(author_name) +
")"

        execute_query(connection, insert_author)
        author_query = "SELECT id FROM author ORDER BY id DESC LIMIT 1"
        result_author = execute_read_query(connection, author_query)
        result_author_new = []
        for e in result_author:
            result_author_new.append(e[0])
        author_code = result_author_new[0] # останній номер автора для обробки
        now = datetime.now()
        print("before current_call_comment_list = " + str(current_call_comment_list))
        print("not current_call_comment_list = " + str(not current_call_comment_list))
        if not current_call_comment_list:
            insert_support_call = "INSERT INTO
`support_call`(`parent_id`,`type`,`topic`,`text`,`create_at`,`
responsible`) VALUES (" + str(support_id_next) + ", '1', " + str(topic) +
", " + \
str(message.text) + ", " + now.strftime("%Y-%m-%d %H:%M:%S") + ", " +
+ str(author_code) + ")"
        else:
            jsonData = json.dumps(current_call_comment_list)
            print("jsonData = " + str(jsonData))
            insert_support_call = "INSERT INTO
`support_call`(`parent_id`,`type`,`topic`,`text`,`create_at`,`
files`,`responsible`) VALUES (" + str(support_id_next) + ", '1', " + str(
topic) + ", " + \
str(message.text) + ", " + now.strftime("%Y-%m-%d %H:%M:%S") + ", " +
+ str(jsonData) + \
+ ", " + str(author_code) + ")"

```

```

        execute_query(connection, insert_support_call)
        # нижче блок коду для запам'ятовування id заявки з support для подальшої вставки в
support_call_bot_log
        select_id_support_call = "SELECT id FROM support_call WHERE parent_id=" +
str(support_id_next) + \
            "" AND topic=" + str(topic) + "" AND text=" + str(message.text) \
            + "" AND responsible=" + str(author_code) + "" LIMIT 1"
        result_id_support_call = execute_read_query(connection, select_id_support_call)
        support_call_id_new = []
        for e in result_id_support_call:
            support_call_id_new.append(e[0])
        current_id_call_from_support = support_call_id_new[0] # id доданої заявки

# метод вставки заявки(первісної) в базу даних техпідтримки
def insert_call_to_db(user, now, title, text):
    global current_call_list, current_call_code, current_id_call_from_support
    current_id_call_from_support = "
    connection = create_connection()
    if connection is not None:
        fio = user[0]
        id, surname, name, patr = fio
        fio_str = ""
        fio_str = surname + " " + name + " " + patr
        # нижче блок коду - вставка автора в таблицю-author
        insert_author = "INSERT INTO `author`(`full_name`) VALUES ('" + fio_str + "')"
        execute_query(connection, insert_author)
        author_query = "SELECT id FROM author ORDER BY id DESC LIMIT 1"
        result_author = execute_read_query(connection, author_query)
        result_author_new = []
        for e in result_author:
            result_author_new.append(e[0])
        author_code = 0
        author_code = result_author_new[0] # останній номер автора
        support_call_topic_query = "SELECT topic FROM support_call ORDER BY topic DESC
LIMIT 1"
        support_call_topic = execute_read_query(connection, support_call_topic_query)
        support_call_topic_new = []
        for e in support_call_topic:
            support_call_topic_new.append(e[0]) # останній номер заявки/теми
        support_call_topic_next = 0
        support_call_topic_next = support_call_topic_new[0] + 1 # доданий номер заявки/теми
        # зберігаємо значення коду заявки в глобальну змінну на випадок прикріплення
файлів
        current_call_code = support_call_topic_next
        # нижче блок коду - вставка запису в таблицю заявок-support_call
        if not current_call_list:
            jsonData = json.dumps(current_call_list)
            insert_support_call = "INSERT INTO
`sUPPORT_CALL`(`parent_id`,`type`,`topic`,`text`,`files`,`create_at`,`
responsible`) VALUES ('0','1'," + str(support_call_topic_next) + "," + \
text + "," + str(jsonData) + "," + now.strftime("%Y-%m-%d
%H:%M:%S") + "," +
            + str(author_code) + ")"
        else:
            insert_support_call = "INSERT INTO
`sUPPORT_CALL`(`parent_id`,`type`,`topic`,`text`,`create_at`,`"

```

```

        "`responsible`) VALUES ('0','1','" + str(support_call_topic_next) + "',' + \
        text + "',' + now.strftime("%Y-%m-%d %H:%M:%S") + "',' + \
        + str(author_code) + "')"
    execute_query(connection, insert_support_call)
    # нижче блок коду - вставка запису в таблицю тем-support_topic
    insert_support_call1 = "INSERT INTO
`support_topic`(`code`,`status`,`title`,`responsible`,`\
        "`create_at`,`server`,`program`,`priority`) VALUES ('" \
        + str(support_call_topic_next) + "','1','" + title + "','\
        + str(author_code) + "',' + str(now.strftime("%Y-%m-%d %H:%M:%S")) +
        "','17','16','0')\"
    execute_query(connection, insert_support_call1)
    # нижче блок коду для запам'ятовування id заявки з support для подальшої вставки в
support_call_bot_log
    select_id_support_call = "SELECT id FROM support_call WHERE parent_id='0' AND
topic=\"" \
        + str(support_call_topic_next) + "\" AND text=\"" + str(text) + "\" AND
responsible=\"" \
        + str(author_code) + "\" LIMIT 1\"
    result_id_support_call = execute_read_query(connection, select_id_support_call)
    support_call_id_new = []
    for e in result_id_support_call:
        support_call_id_new.append(e[0])
    current_id_call_from_support = support_call_id_new[0] # id доданої заявки

    # метод обробки введенного ПЗ, яке необхідно встановити користувачу
    @bot.message_handler(func=lambda message: user_step.get(message.chat.id) ==
GET_NEED_INSTALL_PC)
    def get_po_need_install_user(message):
        global flag, install_need_where, current_call_code
        if message.text != "":
            if message.text in exeption:
                user_step[message.chat.id] = TEAM_USER_ACCEPTED
                flag = 10
                external_error(message)
            else:
                now = datetime.now()
                period = timedelta(hours=24)
                before = now + period
                if get_user_info_from_db_by_tel_id(message) != "":
                    for chat_id in moderator_chat_id:
                        bot.send_message(chat_id, 'Користувач ' +
get_user_info_from_db_by_tel_id(message) +
                            ' запросив установку ' + install_need_where +
                            ' наступного програмного забезпечення: ' + message.text +
                            "\nЗаявка надійшла " + str(now.day) + '.' + str(now.month) + '.' +
str(now.year)
                            + ' в ' + str(now.hour) + ':' + str(now.minute) +
                            "\nНеобхідно обробити заявку до " + str(before.day) + '.' +
str(before.month) + '.' +
                            + str(before.year) + ' в ' + str(before.hour) + ':' + str(before.minute)
                            )
                        bot.forward_message(chat_id, message.chat.id, message.message_id)
        #

```

```

        insert_call_to_buffer_db(message.chat.id, current_call_code, message.message_id,
message.date, ", ")
        #
        bot.send_message(message.chat.id, 'Заявка на установку ' + install_need_where +
            ' наступного програмного забезпечення: '
            + message.text + ' успішно відправлена та буде оброблена протягом 1
робочого дня.')
        user_step[message.chat.id] = TEAM_USER_ACCEPTED
    else:
        bot.send_message(message.chat.id, 'Помилка відправки заявки на установку ' +
install_need_where +
            ' наступного програмного забезпечення:' + message.text)
        for chat_id in moderator_chat_id:
            bot.send_message(chat_id, 'Виникла помилка для користувача ' +
str(message.chat.id) +
                ' . Він намагався запросити установку ' + install_need_where + '
наступного '
                    ' програмного забезпечення: ' +
message.text +
                        # "\nКомпанія: " + company +
                        "\nПомилка заявки надійшла " + str(now.day) + ' '
                        + str(now.month) + ' ' + str(now.year)
                        + 'р в ' + str(now.hour) + ' ' + str(now.minute)
                        )
            bot.forward_message(chat_id, message.chat.id, message.message_id)
        flag = 0

# метод повертає інформацію про користувача за телеграм id з бази не рядком
def get_user_not_str_info_from_db_by_tel_id(message):
    connection = create_connection()
    if connection is not None:
        select_users = "SELECT id, surname, name, patronymic FROM responsible_train where
delete1 is null and telegram_id =" \
            + str(message.chat.id)
        users = execute_read_query(connection, select_users)
        return users

# метод повертає інформацію про користувача за телеграм id з бази
def get_user_info_from_db_by_tel_id(message):
    connection = create_connection()
    if connection is not None:
        select_users = "SELECT id, surname, name, patronymic, company, phone_number,
telegram_id FROM responsible_train where delete1 is null and telegram_id =" \
            + str(message.chat.id)
        users = execute_read_query(connection, select_users)
        new_string = ""
        for row in users:
            new_string += str(row) + "\n"
        return new_string

# метод обробки відключення облікового запису користувачем
# what_account-ПЗ, до якого вимикається обліковий запис
# action_account - відключення
# moderator_chat_id-перелік модераторів, яким відправляється запит

```

```

def method_get_account(message):
    global moderator_chat_id, company, what_account, action_account, flag, current_call_status,
current_call_code
    now = datetime.now()
    period = timedelta(hours=24)
    before = now + period
    if get_user_info_from_db_by_tel_id(message) != "":
        current_call_status = True
        for chat_id in moderator_chat_id:
            bot.send_message(chat_id, 'Користувач ' + get_user_info_from_db_by_tel_id(message)
+
                                ' запросив ' + action_account + ' облікового запису для ПЗ ' + what_account
+
                                '\nЗаявка надійшла ' + str(now.day) + '.' + str(now.month) + '.' +
str(now.year)
                                + ' в ' + str(now.hour) + ':' + str(now.minute) +
                                '\nНеобхідно обробити заявку до ' + str(before.day) + '.' + str(before.month)
+ '.'
                                + str(before.year) + ' в ' + str(before.hour) + ':' + str(before.minute)
                                )
            title = action_account + " облікового запису"
            text = "Запрошено " + action_account + " облікового запису для ПЗ " + what_account
            #++insert_call_to_db(get_user_not_str_info_from_db_by_tel_id(message), now, title, text)
            insert_call_to_buffer_db(message.chat.id, current_call_code, ", ", ", ")
            bot.send_message(message.chat.id, 'Заявка на ' + action_account + ' облікового запису
ПЗ ' + what_account +
                                ' успішно відправлена та буде оброблена протягом 1 робочого дня.')
        else:
            bot.send_message(message.chat.id, 'Помилка відправки заявки на ' + action_account
                                + ' облікового запису ПЗ ' + what_account)
            for chat_id in moderator_chat_id:
                bot.send_message(chat_id, 'Виникла помилка для користувача ' + str(message.chat.id)
+
                                ' . Він намагався запросити ' + action_account + ' облікового запису для ПЗ
' + what_account +
                                '\nПомилка заявки надійшла ' + str(now.day) + '.' + str(now.month) + '.' +
str(now.year)
                                + ' в ' + str(now.hour) + ':' + str(now.minute)
                                )
            flag = 0
            action_account = ""
            what_account = ""

# метод обробки запиту доступу облікового запису користувачем
# what_account-ПЗ, до якого запрошується обліковий запис
# action_account - запит доступу(доступ до)
# moderator_chat_id-перелік модераторів, яким відправлен запит
def method_get_account1(message):
    global moderator_chat_id, company, what_account, action_account, flag, current_call_code
    global current_call_status, current_call_text, current_call_chat_id, get_account_call_flag,
dont_five_status_flag
    now = datetime.now()
    period = timedelta(hours=24)
    before = now + period
    if get_user_info_from_db_by_tel_id(message) != "":
        current_call_status = True

```



```

else:
    bot.send_message(message.chat.id, 'Для доступу до даної команди спочатку необхідно
зареєструватися '
                                'та дочекатись схвалення заявки.')
    flag = 0
    error_company_method()

# метод обробки запиту паролю від пошти користувачем
# what_account-ПЗ, до якого запитується обліковий запис
# action_account - запит паролю
# moderator_chat_id-перелік модераторів, яким відправляється запит
@bot.message_handler(func=lambda message: user_step.get(message.chat.id) ==
ADD_USER_POSITION)
def method_get_account2(message):
    global moderator_chat_id, company, what_account, action_account, flag, current_call_code
    global current_call_status, current_call_text, current_call_chat_id, get_account_call_flag
    now = datetime.now()
    period = timedelta(hours=24)
    before = now + period
    if get_user_info_from_db_by_tel_id(message) != "":
        current_call_status = True
        for chat_id in moderator_chat_id:
            bot.send_message(chat_id, 'Користувач ' + get_user_info_from_db_by_tel_id(message)
+
                                ' запросив ' + action_account + ' облікового запису для ПЗ ' + what_account
+
                                '\nЗаявка надійшла ' + str(now.day) + '!' + str(now.month) + '!' +
str(now.year)
                                + 'p в ' + str(now.hour) + ':' + str(now.minute) +
                                '\nНеобхідно обробити заявку до ' + str(before.day) + '!' + str(before.month)
+ '!'
                                + str(before.year) + 'p в ' + str(before.hour) + ':' + str(before.minute)
                                )
            bot.forward_message(chat_id, message.chat.id, message.message_id)
            title = action_account + " облікового запису"
            text = "Запрошено " + action_account + " облікового запису для ПЗ " + what_account
            #++insert_call_to_db(get_user_not_str_info_from_db_by_tel_id(message), now, title, text)
            insert_call_to_buffer_db(message.chat.id, current_call_code, message.message_id,
message.date, ", ")
            bot.send_message(message.chat.id, 'Заявка на ' + action_account + ' облікового запису
ПЗ ' + what_account +
                                ' успішно відправлена та буде оброблена протягом 1 робочого дня.')
        else:
            bot.send_message(message.chat.id, 'Помилка відправки заявки на ' + action_account
                                + ' облікового запису ПЗ ' + what_account)
            for chat_id in moderator_chat_id:
                bot.send_message(chat_id, 'Виникла помилка для користувача ' + str(message.chat.id)
+
                                ' . Він намагався запросити ' + action_account + ' облікового запису для ПЗ
' + what_account +
                                '\nПомилка заявки надійшла ' + str(now.day) + '!' + str(now.month) + '!' +
str(now.year)
                                + 'p в ' + str(now.hour) + ':' + str(now.minute)
                                )
            user_step[message.chat.id] = TEAM_USER_ACCEPTED
            flag = 0

```

```

error_company_method()
action_account = ""
what_account = ""

@bot.message_handler(commands=['start'])
def start(message):
    start_command(message)

# відправка відео-інструкцій для асу/компас
def send_mp4_file(message, flag_mp4):
    global flag
    flag = 9
    if flag_mp4 == 0: # асу
        mp4 = open('C:/USR/www/support_bot/pythonProject/instruction_asu.mp4', 'rb')
    if flag_mp4 == 1: # компас
        mp4 = open('C:/USR/www/support_bot/pythonProject/instruction_kompas.mp4', 'rb')
    bot.send_video(message.chat.id, mp4)
    flag = 0

# відправка pdf-інструкцій для асу/компас
def send_pdf_file(message, flag_pdf):
    global flag
    flag = 8
    if flag_pdf == 0: # асу
        pdf = open('C:/USR/www/support_bot/pythonProject/instruction_asu_2020.pdf', 'rb')
    if flag_pdf == 1: # компас
        pdf = open('C:/USR/www/support_bot/pythonProject/instruction_kompas_2020.pdf', 'rb')
    bot.send_document(message.chat.id, pdf)
    flag = 0

# метод обробки відправки apk компас з перевіркою на авторизацію
def send_kompas_apk_url(message):
    global flag
    if is_register_user(message.chat.id):
        send_kompas_apk_url_method(message)
    else:
        bot.send_message(message.chat.id, 'Для доступу до даної команди спочатку необхідно
зареєструватися '
                                'та дочекатись схвалення заявки.')
        flag = 0
        error_company_method()

# метод відправки кнопки з посиланням на apk компас на гугл-диску
def send_kompas_apk_url_method(message):
    markup = types.InlineKeyboardMarkup(row_width=1)
    item1 = types.InlineKeyboardButton("📄 КОМПАС apk",
url='https://drive.google.com/file/d/1bbyJnH0USOoDUAsr1Al_KyWV_GKuFt-9/view?usp=sharing')
    markup.add(item1)
    bot.send_message(message.chat.id, 'Для завантаження застосунку перейдіть за
посиланням:', reply_markup=markup)

```

```

# метод обробки відправки apk асу з перевіркою на авторизацію
def send_asu_apk_url(message):
    global flag
    if is_register_user(message.chat.id):
        send_asu_apk_url_method(message)
    else:
        bot.send_message(message.chat.id, 'Для доступу до даної команди спочатку необхідно
зареєструватися '
                                'та дочекатись схвалення заявки.')
        flag = 0
        error_company_method()

# метод відправки кнопки з посиланням на apk асу на гугл-диску
def send_asu_apk_url_method(message):
    markup = types.InlineKeyboardMarkup(row_width=1)
    item1 = types.InlineKeyboardButton("🔗 ACU apk",
url='https://drive.google.com/file/d/1PiMrVVtfV0Ix7CHVlrSteOPHR64VXgQS/view?usp=sharing')
    markup.add(item1)
    bot.send_message(message.chat.id, 'Для завантаження застосунку перейдіть за
посиланням:', reply_markup=markup)

# метод перевірки авторизації користувача
def is_register_user(id_user):
    connection = create_connection()
    if connection is not None:
        select_users = "SELECT telegram_id FROM responsible_train where delete1 is null and
telegram_id is not null"
        users = execute_read_query(connection, select_users)
        new_user = []
        for e in users:
            new_user.append(e[0])
        if int(id_user) in new_user:
            return True
        else:
            return False

# обробка команди меню 2.1.2.1.2 користувача зареєстрованого
@bot.message_handler(commands=['menu21212'])
def menu_21212_user(message):
    global flag
    if is_register_user(message.chat.id):
        method_menu_21212_user(message)
    else:
        bot.send_message(message.chat.id, 'Для доступу до даної команди спочатку необхідно
зареєструватися '
                                'та дочекатись схвалення заявки.')
        flag = 0
        error_company_method()

def method_menu_21212_user(message):
    markup = types.InlineKeyboardMarkup(row_width=1)

```

```

        item1 = types.InlineKeyboardButton("🔒 Запросити обліковий запис",
callback_data='get_account_kompas_21212_level')
        item2 = types.InlineKeyboardButton("🔓 Вимкнути обліковий запис",
callback_data='off_account_kompas_21212_level')
        item3 = types.InlineKeyboardButton("💬 Інше", callback_data='other_21212_level')
        item4 = types.InlineKeyboardButton("⬅️ Назад", callback_data='back_21211_level')
        markup.add(item1, item2, item3, item4)
        bot.send_message(message.chat.id, 'Розділ Доступ в КОМПАС містить наступні
підрозділи:', reply_markup=markup)
        global flag
        flag = 0
        error_company_method()

```

```

# обробка команди меню 2.1.2.1.1 користувача зареєстрованого
@bot.message_handler(commands=['menu21211'])
def menu_21211_user(message):
    global flag
    if is_register_user(message.chat.id):
        method_menu_21211_user(message)
    else:
        bot.send_message(message.chat.id, 'Для доступу до даної команди спочатку необхідно
зареєструватися '
                                'та дочекатись схвалення заявки.')
        flag = 0
        error_company_method()

```

```

def method_menu_21211_user(message):
    markup = types.InlineKeyboardMarkup(row_width=1)
    item1 = types.InlineKeyboardButton("📄 Отримати файл інструкції",
callback_data='get_instruct_kompas_21211_level')
    item2 = types.InlineKeyboardButton("📺 Отримати відео",
callback_data='get_video_kompas_21211_level')
    item3 = types.InlineKeyboardButton("⬅️ Назад", callback_data='back_21211_level')
    markup.add(item1, item2, item3)
    bot.send_message(message.chat.id, 'Розділ Інструкція містить наступні підрозділи:',
reply_markup=markup)
    global flag
    flag = 0
    error_company_method()

```

```

# обробка команди меню 2.1.1.1.2 користувача зареєстрованого
@bot.message_handler(commands=['menu21112'])
def menu_21112_user(message):
    global flag
    if is_register_user(message.chat.id):
        # bot.send_message(message.chat.id, 'Working')
        method_menu_21112_user(message)
        # start_command(message)
    else:
        bot.send_message(message.chat.id, 'Для доступу до даної команди спочатку необхідно
зареєструватися '
                                'та дочекатись схвалення заявки.')
        flag = 0

```

```

error_company_method()

def method_menu_21112_user(message):
    markup = types.InlineKeyboardMarkup(row_width=1)
    item1 = types.InlineKeyboardButton("🔒 Запросити обліковий запис",
callback_data='get_account_asu_21112_level')
    item2 = types.InlineKeyboardButton("🔓 Вимкнути обліковий запис",
callback_data='off_account_asu_21112_level')
    item3 = types.InlineKeyboardButton("💬 Інше", callback_data='other_21112_level')
    item4 = types.InlineKeyboardButton("⬅️ Назад", callback_data='back_21111_level')
    markup.add(item1, item2, item3, item4)
    bot.send_message(message.chat.id, 'Розділ Доступ в АСУ містить наступні підрозділи:',
reply_markup=markup)
    global flag
    flag = 0
    error_company_method()

# обробка команди меню 2.1.1.1.1 користувача зареєстрованого
@bot.message_handler(commands=['menu21111'])
def menu_21111_user(message):
    global flag
    if is_register_user(message.chat.id):
        # bot.send_message(message.chat.id, 'Working')
        method_menu_21111_user(message)
        # start_command(message)
    else:
        bot.send_message(message.chat.id, 'Для доступу до даної команди спочатку необхідно
зареєструватися '
                                'та дочекатись схвалення заявки.')
        flag = 0
        error_company_method()

def method_menu_21111_user(message):
    markup = types.InlineKeyboardMarkup(row_width=1)
    item1 = types.InlineKeyboardButton("📄 Отримати файл інструкції",
callback_data='get_instruct_asu_21111_level')
    item2 = types.InlineKeyboardButton("📺 Отримати відео",
callback_data='get_video_asu_21111_level')
    item3 = types.InlineKeyboardButton("⬅️ Назад", callback_data='back_21111_level')
    markup.add(item1, item2, item3)
    bot.send_message(message.chat.id, 'Розділ Інструкція містить наступні підрозділи:',
reply_markup=markup)
    global flag
    flag = 0
    error_company_method()

# обробка команди меню 2.1.4.1 користувача зареєстрованого
@bot.message_handler(commands=['menu2141'])
def menu_2141_user(message):
    global flag
    if is_register_user(message.chat.id):
        # bot.send_message(message.chat.id, 'Working')

```

```

        method_menu_2141_user(message)
        # start_command(message)
    else:
        bot.send_message(message.chat.id, 'Для доступу до даної команди спочатку необхідно
зареєструватися '
                                'та дочекатись схвалення заявки.')
        flag = 0
        error_company_method()

def method_menu_2141_user(message):
    markup = types.InlineKeyboardMarkup(row_width=1)
    item1 = types.InlineKeyboardButton("🔒 Запросити обліковий запис",
callback_data='get_account_1s_2141_level')
    item2 = types.InlineKeyboardButton("🔒? Вимкнути обліковий запис",
callback_data='off_account_1s_2141_level')
    item3 = types.InlineKeyboardButton("🗨 Інше", callback_data='other_2141_level')
    item4 = types.InlineKeyboardButton("⬅️ Назад", callback_data='back_2141_level')
    markup.add(item1, item2, item3, item4)
    bot.send_message(message.chat.id, 'Розділ Доступ в 1С містить наступні підрозділи:',
reply_markup=markup)
    global flag
    flag = 0
    error_company_method()

# обробка команди меню 2.1.2.2 користувача зареєстрованого
@bot.message_handler(commands=['menu2122'])
def menu_2122_user(message):
    global flag
    if is_register_user(message.chat.id):
        # bot.send_message(message.chat.id, 'Working')
        method_menu_2122_user(message)
        # start_command(message)
    else:
        bot.send_message(message.chat.id, 'Для доступу до даної команди спочатку необхідно
зареєструватися '
                                'та дочекатись схвалення заявки.')
        flag = 0
        error_company_method()

def method_menu_2122_user(message):
    markup = types.InlineKeyboardMarkup(row_width=1)
    item1 = types.InlineKeyboardButton("📍 Отримати посилання на застосунок",
callback_data='get_kompas_apk_2122_level')
    item2 = types.InlineKeyboardButton("👤💻 Тех.проблеми",
callback_data='tech_problem_2122_level')
    item3 = types.InlineKeyboardButton("⬅️ Назад", callback_data='back_2121_level')
    markup.add(item1, item2, item3)
    bot.send_message(message.chat.id, 'Розділ КОМПАС Mobile містить наступні
підрозділи:', reply_markup=markup)
    global flag
    flag = 0
    error_company_method()

```

```

# обробка команди меню 2.1.2.1 користувача зареєстрованого
@bot.message_handler(commands=['menu2121'])
def menu_2111_user(message):
    global flag
    if is_register_user(message.chat.id):
        # bot.send_message(message.chat.id, 'Working')
        method_menu_2121_user(message)
        # start_command(message)
    else:
        bot.send_message(message.chat.id, 'Для доступу до даної команди спочатку необхідно зареєструватися '
                                           'та дочекатись схвалення заявки.')
        flag = 0
        error_company_method()

def method_menu_2121_user(message):
    markup = types.InlineKeyboardMarkup(row_width=2)
    item1 = types.InlineKeyboardButton("📄 Інструкція", callback_data='instr_2121_level')
    item2 = types.InlineKeyboardButton("🔑 Доступ в КОМПАС",
callback_data='access_kompas_2121_level')
    item3 = types.InlineKeyboardButton("👤💻 Тех.проблеми",
callback_data='tech_problem_2121_level')
    item4 = types.InlineKeyboardButton("⬅️ Назад", callback_data='back_2121_level')
    markup.add(item1, item2, item3, item4)
    bot.send_message(message.chat.id, 'Розділ КОМПАС Web містить наступні підрозділи:',
reply_markup=markup)
    global flag
    flag = 0
    error_company_method()

# обробка команди меню 2.1.1.2 користувача зареєстрованого
@bot.message_handler(commands=['menu2112'])
def menu_2112_user(message):
    global flag
    if is_register_user(message.chat.id):
        # bot.send_message(message.chat.id, 'Working')
        method_menu_2112_user(message)
        # start_command(message)
    else:
        bot.send_message(message.chat.id, 'Для доступу до даної команди спочатку необхідно зареєструватися '
                                           'та дочекатись схвалення заявки.')
        flag = 0
        error_company_method()

def method_menu_2112_user(message):
    markup = types.InlineKeyboardMarkup(row_width=1)
    item1 = types.InlineKeyboardButton("📄 Отримати посилання на застосунок",
callback_data='get_asu_apk_2112_level')
    item2 = types.InlineKeyboardButton("👤💻 Тех.проблеми",
callback_data='tech_problem_2112_level')
    item3 = types.InlineKeyboardButton("⬅️ Назад", callback_data='back_2111_level')

```

```

markup.add(item1, item2, item3)
bot.send_message(message.chat.id, 'Розділ АСУ Фотофіксація містить наступні
підрозділи:', reply_markup=markup)
global flag
flag = 0
error_company_method()

# обробка команди меню 2.1.1.1 користувача зареєстрованого
@bot.message_handler(commands=['menu2111'])
def menu_2111_user(message):
    global flag
    if is_register_user(message.chat.id):
        # bot.send_message(message.chat.id, 'Working')
        method_menu_2111_user(message)
        # start_command(message)
    else:
        bot.send_message(message.chat.id, 'Для доступу до даної команди спочатку необхідно
зареєструватися '
                                'та дочекатись схвалення заявки.')
        flag = 0
        error_company_method()

def method_menu_2111_user(message):
    markup = types.InlineKeyboardMarkup(row_width=2)
    item1 = types.InlineKeyboardButton("📄 Інструкція", callback_data='instr_2111_level')
    item2 = types.InlineKeyboardButton("🔑 Доступ в АСУ",
callback_data='access_asu_2111_level')
    item3 = types.InlineKeyboardButton("🔧 Тех.проблеми",
callback_data='tech_problem_2111_level')
    item4 = types.InlineKeyboardButton("⬅️ Назад", callback_data='back_2111_level')
    markup.add(item1, item2, item3, item4)
    bot.send_message(message.chat.id, 'Розділ АСУ Web містить наступні підрозділи:',
reply_markup=markup)
    global flag
    flag = 0
    error_company_method()

# обробка команди меню 2.2.3 користувача зареєстрованого
@bot.message_handler(commands=['menu223'])
def menu_223_user(message):
    global flag
    if is_register_user(message.chat.id):
        # bot.send_message(message.chat.id, 'Working')
        method_menu_223_user(message)
        # start_command(message)
    else:
        bot.send_message(message.chat.id, 'Для доступу до даної команди спочатку необхідно
зареєструватися '
                                'та дочекатись схвалення заявки.')
        flag = 0
        error_company_method()

```

```

def method_menu_223_user(message):
    markup = types.InlineKeyboardMarkup(row_width=1)
    item1 = types.InlineKeyboardButton("📁 Офіс", callback_data='office_223_level')
    item2 = types.InlineKeyboardButton("👤 Особистий", callback_data='private_223_level')
    item3 = types.InlineKeyboardButton("⬅️ Назад", callback_data='back_221_level')
    markup.add(item1, item2, item3)
    bot.send_message(message.chat.id, 'Розділ Internet містить наступні підрозділи:',
reply_markup=markup)
    global flag
    flag = 0
    error_company_method()

# обробка команди меню 2.2.2 користувача зареєстрованого
@bot.message_handler(commands=['menu222'])
def menu_222_user(message):
    global flag
    if is_register_user(message.chat.id):
        # bot.send_message(message.chat.id, 'Working')
        method_menu_222_user(message)
        # start_command(message)
    else:
        bot.send_message(message.chat.id, 'Для доступу до даної команди спочатку необхідно
зареєструватися '
                                'та дочекатись схвалення заявки.')
        flag = 0
        error_company_method()

def method_menu_222_user(message):
    markup = types.InlineKeyboardMarkup(row_width=1)
    item1 = types.InlineKeyboardButton("📱 Мобильний", callback_data='mobile_222_level')
    item2 = types.InlineKeyboardButton("☎️ IP-телефонія", callback_data='ip_tel_222_level')
    item3 = types.InlineKeyboardButton("⬅️ Назад", callback_data='back_221_level')
    markup.add(item1, item2, item3)
    bot.send_message(message.chat.id, 'Розділ Телефонія містить наступні підрозділи:',
reply_markup=markup)
    global flag
    flag = 0
    error_company_method()

def method_menu_221_user(message):
    markup = types.InlineKeyboardMarkup(row_width=2)
    item1 = types.InlineKeyboardButton("💻 Робоче місце",
callback_data='work_place_221_level')
    item2 = types.InlineKeyboardButton("📱 Планшет", callback_data='planshet_221_level')
    item3 = types.InlineKeyboardButton("💬 Інше", callback_data='other_221_level')
    item4 = types.InlineKeyboardButton("⬅️ Назад", callback_data='back_221_level')
    markup.add(item1, item2, item3, item4)
    bot.send_message(message.chat.id, 'Розділ Корпоративна техніка містить наступні
підрозділи:', reply_markup=markup)
    global flag
    flag = 0
    error_company_method()

```

```

# обробка команди меню 2.2.1 користувача зареєстрованого
@bot.message_handler(commands=['menu221'])
def menu_221_user(message):
    global flag
    if is_register_user(message.chat.id):
        # bot.send_message(message.chat.id, 'Working')
        method_menu_221_user(message)
        # start_command(message)
    else:
        bot.send_message(message.chat.id, 'Для доступу до даної команди спочатку необхідно зареєструватися '
                                           'та дочекатись схвалення заявки.')
        flag = 0
        error_company_method()

def method_menu_215_user(message):
    markup = types.InlineKeyboardMarkup(row_width=2)
    item1 = types.InlineKeyboardButton("👤💻 Тех.проблеми",
callback_data='tech_problem_215_level')
    item2 = types.InlineKeyboardButton("⬅️ Назад", callback_data='back_211_level')
    markup.add(item1, item2)
    bot.send_message(message.chat.id, 'Розділ Control містить наступні підрозділи:',
reply_markup=markup)
    global flag
    flag = 0
    error_company_method()

# обробка команди меню 2.1.5 користувача зареєстрованого
@bot.message_handler(commands=['menu215'])
def menu_215_user(message):
    global flag
    if is_register_user(message.chat.id):
        # bot.send_message(message.chat.id, 'Working')
        method_menu_215_user(message)
        # start_command(message)
    else:
        bot.send_message(message.chat.id, 'Для доступу до даної команди спочатку необхідно зареєструватися '
                                           'та дочекатись схвалення заявки.')
        flag = 0
        error_company_method()

def method_menu_214_user(message):
    markup = types.InlineKeyboardMarkup(row_width=1)
    item1 = types.InlineKeyboardButton("🔑 Доступ", callback_data='access_214_level')
    item2 = types.InlineKeyboardButton("👤💻 Тех.проблеми",
callback_data='tech_problem_214_level')
    item3 = types.InlineKeyboardButton("⬅️ Назад", callback_data='back_211_level')
    markup.add(item1, item2, item3)
    bot.send_message(message.chat.id, 'Розділ 1C містить наступні підрозділи:',
reply_markup=markup)
    global flag

```

```

flag = 0
error_company_method()

# обробка команди меню 2.1.4 користувача зареєстрованого
@bot.message_handler(commands=['menu214'])
def menu_214_user(message):
    global flag
    if is_register_user(message.chat.id):
        # bot.send_message(message.chat.id, 'Working')
        method_menu_214_user(message)
        # start_command(message)
    else:
        bot.send_message(message.chat.id, 'Для доступу до даної команди спочатку необхідно
зареєструватися '
                                'та дочекатись схвалення заявки.')
        flag = 0
        error_company_method()

def method_menu_217_user(message):
    markup = types.InlineKeyboardMarkup(row_width=1)
    item1 = types.InlineKeyboardButton("🖨 Встановити на ПК",
callback_data='install_pc_217_level')
    item2 = types.InlineKeyboardButton("➡️ Встановити на МОБІЛЬНИЙ",
callback_data='install_mob_217_level')
    item3 = types.InlineKeyboardButton("📱 Встановити на ПЛАНШЕТ",
callback_data='install_plan_217_level')
    item4 = types.InlineKeyboardButton("⬅️ Назад", callback_data='back_211_level')
    markup.add(item1, item2, item3, item4)
    bot.send_message(message.chat.id, 'Розділ Встановити ПЗ містить наступні підрозділи:',
reply_markup=markup)
    global flag
    flag = 0
    error_company_method()

# обробка команди меню 2.1.7 користувача зареєстрованого
@bot.message_handler(commands=['menu217'])
def menu_217_user(message):
    global flag
    if is_register_user(message.chat.id):
        # bot.send_message(message.chat.id, 'Working')
        method_menu_217_user(message)
        # start_command(message)
    else:
        bot.send_message(message.chat.id, 'Для доступу до даної команди спочатку необхідно
зареєструватися '
                                'та дочекатись схвалення заявки.')
        flag = 0
        error_company_method()

def method_menu_213_user(message):
    markup = types.InlineKeyboardMarkup(row_width=1)

```

```

        item1 = types.InlineKeyboardButton("👤 Втрачено пароль",
callback_data='lost_passw_213_level')
        # item2 = types.InlineKeyboardButton("Встановити на ПК",
callback_data='install_pc_213_level')
        item3 = types.InlineKeyboardButton("🗨 Інше", callback_data='other_213_level')
        item4 = types.InlineKeyboardButton("⬅️ Назад", callback_data='back_211_level')
        markup.add(item1, item3, item4)
        bot.send_message(message.chat.id, 'Розділ Пошта містить наступні підрозділи:',
reply_markup=markup)
        global flag
        flag = 0
        error_company_method()

# обробка команди меню 2.1.3 користувача зареєстрованого
@bot.message_handler(commands=['menu213'])
def menu_213_user(message):
    global flag
    if is_register_user(message.chat.id):
        # bot.send_message(message.chat.id, 'Working')
        method_menu_213_user(message)
        # start_command(message)
    else:
        bot.send_message(message.chat.id, 'Для доступу до даної команди спочатку необхідно
зареєструватися '
                                'та дочекатись схвалення заявки.')
        flag = 0
        error_company_method()

def method_menu_212_user(message):
    markup = types.InlineKeyboardMarkup(row_width=1)
    item1 = types.InlineKeyboardButton("🌐 КОМПАС Web",
callback_data='kompas_web_212_level')
    item2 = types.InlineKeyboardButton("📱 КОМПАС Mobile",
callback_data='kompas_mob_212_level')
    item3 = types.InlineKeyboardButton("⬅️ Назад", callback_data='back_211_level')
    markup.add(item1, item2, item3)
    bot.send_message(message.chat.id, 'Розділ КОМПАС містить наступні підрозділи:',
reply_markup=markup)
    global flag
    flag = 0
    error_company_method()

# обробка команди меню 2.1.2 користувача зареєстрованого
@bot.message_handler(commands=['menu212'])
def menu_212_user(message):
    global flag
    if is_register_user(message.chat.id):
        # bot.send_message(message.chat.id, 'Working')
        method_menu_212_user(message)
        # start_command(message)
    else:
        bot.send_message(message.chat.id, 'Для доступу до даної команди спочатку необхідно
зареєструватися '

```

```

        'та дочекатись схвалення заявки.')
    flag = 0
    error_company_method()

def method_menu_211_user(message):
    markup = types.InlineKeyboardMarkup(row_width=1)
    item1 = types.InlineKeyboardButton("👤 ACY Web", callback_data='asu_web_211_level')
    item2 = types.InlineKeyboardButton("📷 ACY Фотофіксація",
callback_data='asu_photo_211_level')
    item3 = types.InlineKeyboardButton("⬅️ Назад", callback_data='back_211_level')
    markup.add(item1, item2, item3)
    bot.send_message(message.chat.id, 'Розділ ACY містить наступні підрозділи:',
reply_markup=markup)
    global flag
    flag = 0
    error_company_method()

# обробка команди меню 2.1.1 користувача зареєстрованого
@bot.message_handler(commands=['menu211'])
def menu_211_user(message):
    global flag
    if is_register_user(message.chat.id):
        # bot.send_message(message.chat.id, 'Working')
        method_menu_211_user(message)
        # start_command(message)
    else:
        bot.send_message(message.chat.id, 'Для доступу до даної команди спочатку необхідно
зареєструватися '
        'та дочекатись схвалення заявки.')
    flag = 0
    error_company_method()

def method_menu_22_user(message):
    markup = types.InlineKeyboardMarkup(row_width=2)
    item1 = types.InlineKeyboardButton("📁 Корпор.техніка",
callback_data='korp_teh_22_level')
    item2 = types.InlineKeyboardButton("☎️ Телефонія", callback_data='telephony_22_level')
    item3 = types.InlineKeyboardButton("🌐 Internet", callback_data='internet_22_level')
    item4 = types.InlineKeyboardButton("💬 Інше", callback_data='other_22_level')
    item5 = types.InlineKeyboardButton("⬅️ Назад", callback_data='back_21_level')
    markup.add(item1, item2, item3, item4, item5)
    bot.send_message(message.chat.id, 'Розділ Тех.питання містить наступні підрозділи:',
reply_markup=markup)
    global flag
    flag = 0
    error_company_method()

# обробка команди меню 2.2 користувача зареєстрованого
@bot.message_handler(commands=['menu22'])
def menu_22_user(message):
    global flag
    if is_register_user(message.chat.id):

```

```

        # bot.send_message(message.chat.id, 'Working')
        method_menu_22_user(message)
        # start_command(message)
    else:
        bot.send_message(message.chat.id, 'Для доступу до даної команди спочатку необхідно
зареєструватися '
                                'та дочекатись схвалення заявки.')
        flag = 0
        error_company_method()

def method_menu_21_user(message):
    markup = types.InlineKeyboardMarkup(row_width=2)
    item1 = types.InlineKeyboardButton("🔌 АСУ", callback_data='asu_21_level')
    item2 = types.InlineKeyboardButton("☐ КОМПАС", callback_data='kompas_21_level')
    item3 = types.InlineKeyboardButton("✉️ Пошта", callback_data='poshta_21_level')
    item4 = types.InlineKeyboardButton("☐ 1С", callback_data='1s_21_level')
    item5 = types.InlineKeyboardButton("⌘ Control", callback_data='control_21_level')
    item6 = types.InlineKeyboardButton("💡 Інше", callback_data='other_21_level')
    item6 = types.InlineKeyboardButton("⬇️ Встановити ПЗ",
callback_data='install_po_21_level')
    item7 = types.InlineKeyboardButton("⬅️ Назад", callback_data='back_21_level')
    markup.add(item1, item2, item3, item4, item5, item6, item7)
    bot.send_message(message.chat.id, 'Розділ ПЗ містить наступні підрозділи:',
reply_markup=markup)
    global flag
    flag = 0
    error_company_method()

# обробка команди меню 2.1 користувача зареєстрованого
@bot.message_handler(commands=['menu21'])
def menu_21_user(message):
    global flag
    if is_register_user(message.chat.id):
        # bot.send_message(message.chat.id, 'Working')
        method_menu_21_user(message)
        # start_command(message)
    else:
        bot.send_message(message.chat.id, 'Для доступу до даної команди спочатку необхідно
зареєструватися '
                                'та дочекатись схвалення заявки.')
        flag = 0
        error_company_method()

# метод інлайн-клавіатури головного меню
def method_main_menu_user(message):
    global current_call_status
    current_call_status = False
    if is_register_user(message.chat.id):
        method_main_menu_replay_user(message.chat.id)
        markup = types.InlineKeyboardMarkup(row_width=1)
        item1 = types.InlineKeyboardButton("🛡️ Програмне забезпечення(ПЗ)",
callback_data='po_1_level')

```

```

        item2 = types.InlineKeyboardButton("💻 Тех.питання",
callback_data='tech_vopr_1_level')
        item3 = types.InlineKeyboardButton("💡 Інше", callback_data='other_1_level')
        item4 = types.InlineKeyboardButton("✕ ВІДПИСАТИСЯ",
callback_data='exit_user_command')
        markup.add(item1, item2, item3, item4)
        bot.send_message(message.chat.id, 'Головне меню: ',
                           reply_markup=markup)
    else:
        bot.send_message(message.chat.id, 'Для доступу до даної команди спочатку необхідно
зареєструватися '
                           'та дочекатись схвалення заявки.')

    global flag
    flag = 0
    # error_company_method()

def append_photo_doc_to_list(type1, name1):
    global current_call_list
    dict = {"type": type1, "name": name1}
    current_call_list.append(dict)
    # print(current_call_list)

@bot.message_handler(commands=['Головне меню'])
def main_menu_user(message):
    method_main_menu_user(message)

# метод збереже фото та поверне рядок для вставки в коментар користувача в базу
def method_save_photo_in_comm(message):
    try:
        file_info = bot.get_file(message.photo[len(message.photo) - 1].file_id)
        downloaded_file = bot.download_file(file_info.file_path)
        src1 = 'C:/USR/www/support-2/uploads/call/' + str(str(file_info.file_path).split('/')[1])
        if os.path.isfile(src1):
            count = 0
            while os.path.isfile(src1):
                src = src1.split('.')[0] + str(count) + '.' + src1.split('.')[1]
                src1 = src
                count = count + 1
            else:
                src = src1
            with open(src, 'wb') as new_file:
                new_file.write(downloaded_file)
            global current_call_comment_list
            current_call_comment_list = []
            dict = {"type": "image", "name": src.split('/')[6]}
            current_call_comment_list.append(dict)
        except Exception as e:
            bot.send_message(moderator_chat_id[0], "Помилка " + str(e) + " в блоці коду
method_save_photo_in_comm(message)")

# метод збереже документ та вернет строку для вставки в коментарий користувача в
базу

```

```

def method_save_document_in_comm(message):
    try:
        file_info = bot.get_file(message.document.file_id)
        downloaded_file = bot.download_file(file_info.file_path)
        src1 = 'C:/USR/www/support-2/uploads/call/' + message.document.file_name # path file
        if os.path.isfile(src1):
            count = 0
            while os.path.isfile(src1):
                src = src1.split('.')[0] + str(count) + '.' + src1.split('.')[1]
                src1 = src
                count = count + 1
            else:
                src = src1
        with open(src, 'wb') as new_file:
            new_file.write(downloaded_file)
        global current_call_comment_list
        current_call_comment_list = []
        dict = {"type": "application", "name": src.split('/')[6]}
        current_call_comment_list.append(dict)
    except Exception as e:
        # bot.reply_to(message, e)
        bot.send_message(moderator_chat_id[0],
            "Помилка " + str(e) + " в блоці коду
method_save_document_in_comm(message)")

```

```

@bot.message_handler(content_types=['photo'])
def handle_docs_photo(message):
    global current_call_status, moderator_chat_id, current_call_text, get_account_call_flag
    global current_call_code, dont_five_status_flag
    if current_call_status:
        try:
            file_info = bot.get_file(message.photo[len(message.photo) - 1].file_id)
            downloaded_file = bot.download_file(file_info.file_path)
            src1 = 'C:/USR/www/support-2/uploads/call/' + str(str(file_info.file_path).split('/')[1])
            if os.path.isfile(src1):
                count = 0
                while os.path.isfile(src1):
                    src = src1.split('.')[0] + str(count) + '.' + src1.split('.')[1]
                    src1 = src
                    count = count + 1
                else:
                    src = src1
            with open(src, 'wb') as new_file:
                new_file.write(downloaded_file)
            markup = types.InlineKeyboardMarkup(row_width=1) # save call
            item1 = types.InlineKeyboardButton("📁 ЗБЕРЕГТИ", callback_data='save_call')
            markup.add(item1)
            bot.reply_to(message, "Фото вкладено в заявку. Якщо це все, натисніть ЗБЕРЕГТИ",
reply_markup=markup)
            append_photo_doc_to_list("image", src.split('/')[6])
        except Exception as e:
            # bot.reply_to(message, e)
            bot.send_message(moderator_chat_id[0], "Помилка " + str(e) + " в блоці коду
@bot.message_handler"

```

"(content_types=['photo']) def "
 "handle_docs_photo(message), "

```

"current_call_status = True")
else:
    try:
        file_info = bot.get_file(message.photo[len(message.photo) - 1].file_id)
        downloaded_file = bot.download_file(file_info.file_path)
        src1 = 'C:/USR/www/support_bot/' + str(str(file_info.file_path).split('/')[1])
        if os.path.isfile(src1):
            count = 0
            while os.path.isfile(src1):
                src = src1.split('.')[0] + str(count) + '.' + src1.split('.')[1]
                src1 = src
                count = count + 1
            else:
                src = src1
            with open(src, 'wb') as new_file:
                new_file.write(downloaded_file)
            bot.reply_to(message, "Я збережу це та покажу модератору, але не розумію, навіщо
Ви мені це відправили.")
            # append_photo_doc_to_list("image", src.split('/')[4])
        except Exception as e:
            # bot.reply_to(message, e)
            bot.send_message(moderator_chat_id[0], "Помилка " + str(e) + " в блоці коду
@bot.message_handler")

"(content_types=['photo']) def "
"handle_docs_photo(message), "
"current_call_status = False")

for chat_id in moderator_chat_id:
    if get_account_call_flag:
        bot.send_message(chat_id, 'Користувач: ' +
get_user_info_from_db_by_tel_id(message) +
' залишив вам наступний документ.')
        bot.forward_message(chat_id, message.chat.id, message.message_id)
    else:
        bot.send_photo(chat_id, photo=open(src, 'rb'), caption='Користувач: ' +
get_user_info_from_db_by_tel_id(
message) + '\n' + current_call_text)

if current_call_status:
    # print("before photo message.chat.id = " + str(message.chat.id))
    # print("before photo current_call_code = " + str(current_call_code))
    # print("before photo message.message_id = " + str(message.message_id))
    print("in photo dont_five_status_flag = " + str(dont_five_status_flag))
    if dont_five_status_flag:
        insert_call_to_buffer_db(message.chat.id, current_call_code, message.message_id,
message.date, ", ")
    else:
        insert_call_to_buffer_db(message.chat.id, current_call_code, message.message_id,
message.date, ", '5')
        dont_five_status_flag = False
        get_account_call_flag = False

@bot.message_handler(content_types=['document'])
def handle_docs_photo(message):
    global current_call_status, moderator_chat_id, current_call_text, get_account_call_flag
    global current_call_code, dont_five_status_flag
    if current_call_status:
        try:

```

```

file_info = bot.get_file(message.document.file_id)
downloaded_file = bot.download_file(file_info.file_path)
src1 = 'C:/USR/www/support-2/uploads/call/' + message.document.file_name # path file
if os.path.isfile(src1):
    count = 0
    while os.path.isfile(src1):
        src = src1.split('.')[0] + str(count) + '.' + src1.split('.')[1]
        src1 = src
        count = count + 1
else:
    src = src1
with open(src, 'wb') as new_file:
    new_file.write(download_file)
markup = types.InlineKeyboardMarkup(row_width=1) # save call
item1 = types.InlineKeyboardButton("📁 ЗБЕРЕГТИ", callback_data='save_call')
markup.add(item1)
bot.reply_to(message, "вкладено в заявку. Якщо це все, натисніть ЗБЕРЕГТИ",
reply_markup=markup)
append_photo_doc_to_list("application", src.split('/')[6])
except Exception as e:
    # bot.reply_to(message, e)
    bot.send_message(moderator_chat_id[0], "Помилка " + e + " в блоці коду
@bot.message_handler"

"(content_types=['document']) def "
"handle_docs_photo(message), "
"current_call_status = True")

else:
    try:
        file_info = bot.get_file(message.document.file_id)
        downloaded_file = bot.download_file(file_info.file_path)
        src1 = 'C:/USR/www/support_bot/' + str(str(file_info.file_path).split('/')[1])
        if os.path.isfile(src1):
            count = 0
            while os.path.isfile(src1):
                src = src1.split('.')[0] + str(count) + '.' + src1.split('.')[1]
                src1 = src
                count = count + 1
            else:
                src = src1
            with open(src, 'wb') as new_file:
                new_file.write(download_file)
            bot.reply_to(message, "Я збережу це та покажу модератору, але не розумію, навіщо
Ви мені це відправили.")
            # append_photo_doc_to_list("image", src.split('/')[4])
        except Exception as e:
            # bot.reply_to(message, e)
            bot.send_message(moderator_chat_id[0], "Помилка " + str(e) + " в блоці коду
@bot.message_handler"

"(content_types=['document']) def "
"handle_docs_photo(message), "
"current_call_status = False")

document = open(src, 'rb')
for chat_id in moderator_chat_id:
    # bot.send_document(chat_id, document, caption='Користувач: ' +
get_user_info_from_db_by_tel_id(message)
# + '\n' + current_call_text)
bot.send_message(chat_id, 'Користувач: ' + get_user_info_from_db_by_tel_id(message) +

```

```

        ' залишив вам наступний документ.')
    bot.forward_message(chat_id, message.chat.id, message.message_id)
    # bot.forward_message(chat_id, 472932484, 2708)
    if current_call_status:
        print("in document dont_five_status_flag = " + str(dont_five_status_flag))
        if dont_five_status_flag:
            insert_call_to_buffer_db(message.chat.id, current_call_code, message.message_id,
message.date, ", ")
        else:
            insert_call_to_buffer_db(message.chat.id, current_call_code, message.message_id,
message.date, ", '5')
            dont_five_status_flag = False
            get_account_call_flag = False

    # try:
    # chat_id = message.chat.id
    # file_info = bot.get_file(message.document.file_id)
    # downloaded_file = bot.download_file(file_info.file_path)
    # src = 'C:/USR/www/support_bot/' + message.document.file_name # here need path file
    # with open(src, 'wb') as new_file:
    #     new_file.write(downloaded_file)

    # bot.reply_to(message, "Пожалуй, я сохраню это")
    # except Exception as e:
    # bot.reply_to(message, e)

# метод реплай-клавіатури головного меню
# , resize_keyboard=True
# тут message - message.chat.id
def method_main_menu_replay_user(message):
    keyboard = types.ReplyKeyboardMarkup(row_width=1, resize_keyboard=True)
    if is_register_user(message):
        button_phone = types.KeyboardButton(text="Головне меню")
        keyboard.add(button_phone)
        bot.send_message(message,
            'Натисканням на кнопку "Головне меню" Ви в будь-який момент перейдете
до головного меню',
            reply_markup=keyboard)
    else:
        bot.send_message(message, 'Для доступу до даної команди спочатку необхідно
zareestruvatisia '
            'та дочекатись схвалення заявки.')
        flag = 0
        error_company_method()
        flag = 999

# обробка команди меню користувача зареєстрованого
@bot.message_handler(commands=['menu'])
def main_menu_user(message):
    global flag, once_replay_main_menu_flag
    if is_register_user(message.chat.id):
        # bot.send_message(message.chat.id, 'Working')
        if once_replay_main_menu_flag is False:
            method_main_menu_replay_user(message.chat.id)
            once_replay_main_menu_flag = True

```

```

        method_main_menu_user(message)
        # start_command(message)
    else:
        bot.send_message(message.chat.id, 'Для доступу до даної команди спочатку необхідно
зареєструватися '
                                'та дочекатись схвалення заявки.')
        flag = 0
        error_company_method()

@bot.message_handler(commands=['help'])
def help(message):
    bot.send_message(message.chat.id, '/start - вітання'
                                '\n/help - отримання справки по командам'
                                '\n/registration - реєстрація користувачів'
                                '\n/exit - видалення зареєстрованого користувача')

    global flag
    flag = 0
    error_company_method()

# повертає id чата користувачу
@bot.message_handler(func=lambda message:
message.text == "/id" or
message.text == "/ID" or
message.text == "/iD" or
message.text == "/Id")
def get_id(message):
    user = {}
    name = message.chat.first_name
    user[str(name)] = message.chat.id
    for name in user:
        bot.send_message(message.chat.id, '%s: %s' % (name, user[name]))

def get_company_method(message):
    global company, confirm_company_flag, flag, contact_flag
    if is_register_user(message.chat.id):
        bot.send_message(message.chat.id, 'Ви вже зареєстровані.')
    else:
        company = "
        confirm_company_flag = False
        contact_flag = False
        markup = types.InlineKeyboardMarkup(row_width=1) # company
        item1 = types.InlineKeyboardButton("43173964 - ООО «АЛД інжиніринг та
будівництво)", callback_data='company1')
        item2 = types.InlineKeyboardButton("42170810 - «АЛД-ПРОЕКТ»",
callback_data='company2')
        item3 = types.InlineKeyboardButton("42720802 - ООО «АЛД-ПРОМСТРОЙ»",
callback_data='company3')
        item4 = types.InlineKeyboardButton("41901054 - «АЛД-ПРОМСТРОЙ 2»",
callback_data='company4')
        item5 = types.InlineKeyboardButton("42805545 - «АЛД-СЕРВІС»",
callback_data='company5')
        item6 = types.InlineKeyboardButton("431739641 - «РБУ»", callback_data='company6')

        markup.add(item1, item2, item3, item4, item5, item6)

```

```

        bot.send_message(message.chat.id, 'Співробітником якої організації Ви є?',
reply_markup=markup)
        flag = 1
        error_company_method()
        # print('here1')

@bot.message_handler(commands=['registration'])
def get_company(message):
    get_company_method(message)

# видалення клавіатури підтвердження компанії
def confirm_company(call, message_user):
    markup = types.InlineKeyboardMarkup(row_width=2) # yes/no
    item1 = types.InlineKeyboardButton("ТАК", callback_data='yes_company')
    item2 = types.InlineKeyboardButton("НІ", callback_data='no_company')
    markup.add(item1, item2)
    bot.send_message(call.message.chat.id, 'Ви є співробітником ' + message_user + ', вірно?',
reply_markup=markup)
    global company, flag
    company = message_user
    flag = 2
    error_company_method()

# метод обробки введення текста-зовн.виклик
def external_error(message):
    global flag, confirm_company_flag, company, message_contact
    global auth_login_mod_user, auth_login_mod_regular, auth_password_mod_user,
auth_password_mod_regular
    # print(flag)
    if message.text == "Головне меню":
        method_main_menu_user(message)
        flag = 999
    if message.text == "/start":
        start_command(message)
        flag = 999
    if flag == 0: # error /start /help
        bot.send_message(message.chat.id,
            'Я вас не розумію, скористайтесь командою /help для ознайомлення з
моїми командами')
    if flag == 2: # error confirm company
        if not confirm_company_flag:
            bot.send_message(message.chat.id, 'Вам необхідно підтвердити організацію або
відмовитися від неї')
    if flag == 1: # error order company
        if company == "":
            bot.send_message(message.chat.id, 'Вам необхідно вибрати організацію з наданих
вище')
    if flag == 3: # error get phone number
        if not contact_flag:
            bot.send_message(message.chat.id,
                'Вам необхідно натиснути на відповідну кнопку, а не друкувати вручну')
    if flag == 4: # error /start /help_mod
        bot.send_message(message.chat.id,

```

```

        'Я вас не розумію, скористайтесь командою /help_mod для ознайомлення з
        моїми командами')
    if flag == 5: # input login moderator auth
        if message.text != "":
            if message.text == auth_login_mod_regular:
                auth_login_mod_user = message.text
                user_step[message.chat.id] = TEAM_USER_LOGGING
                bot.reply_to(message, "Введіть пароль:")
            else:
                bot.send_message(message.chat.id,
                                'Помилка авторизації. Введено невірний логін модератора. Спробуйте
                                знову:')
    if flag == 6: # error confirm reauth moderator
        if not confirm_reauth_flag:
            bot.send_message(message.chat.id,
                              'Вам необхідно підтвердити вихід з аккаунта модератора '
                              'або відмовитися від даної ідеї')
    if flag == 7: # error confirm exit user
        if message_contact is not None:
            bot.send_message(message.chat.id,
                              'Вам необхідно підтвердити вихід з аккаунта користувача '
                              'або відмовитися від даної ідеї')
    if flag == 8: # error send file pdf
        bot.send_message(message.chat.id,
                          'Якщо файл інструкції не отримано, будь ласка, зачекайте. Йде
                          відправка...')
    if flag == 9: # error send file video
        bot.send_message(message.chat.id,
                          'Якщо видео-інструкція не отримана, будь ласка, зачекайте. Йде
                          відправка...')
    if flag == 10: # error command in text
        bot.send_message(message.chat.id,
                          'Ви не можете використовувати команди бота в описі.')
        flag = 0
    if flag == 11: # error call, second text in call
        bot.send_message(message.chat.id,
                          'необхідно викласти текст проблеми одним повідомленням.')
    if flag == 999:
        flag = 0

# метод на випадок введення тексту
def error_company_method():
    @bot.message_handler(content_types=['text'])
    def error_company(message):
        external_error(message)

# видалення клавіатури з компаніями після вибору поточної
def remote_company_markup(call, current_message):
    bot.edit_message_text(chat_id=call.message.chat.id, message_id=call.message.message_id,
                          text="Ви обрали " + current_message,
                          reply_markup=None)

# видалення клавіатури з підтвердженням компанії після вибору поточної
def remote_company_confirm_markup(call):

```

```

global company
bot.edit_message_text(chat_id=call.message.chat.id, message_id=call.message.message_id,
                      text="Ви підтвердили приналежність наступній організації: " + company,
                      reply_markup=None)

# видалення клавіатури з головним меню(по/тех.питання/Інше)
def remote_main_menu_markup(call, text):
    bot.edit_message_text(chat_id=call.message.chat.id, message_id=call.message.message_id,
                          text="Ви обрали розділ: " + text,
                          reply_markup=None)

# видалення клавіатури з реєстрацією/відпискою користувача після вибору
def remote_start_markup(call):
    global register_exit_flag
    if register_exit_flag != "":
        if register_exit_flag == 'register':
            bot.edit_message_text(chat_id=call.message.chat.id,
message_id=call.message.message_id,
                                text="Ви обрали реєстрацію. Дякую!!!",
                                reply_markup=None)
        if register_exit_flag == 'exit':
            bot.edit_message_text(chat_id=call.message.chat.id,
message_id=call.message.message_id,
                                text="Ви обрали відписку. Дуже шкода(",
                                reply_markup=None)
        if register_exit_flag == 'main_menu':
            bot.edit_message_text(chat_id=call.message.chat.id,
message_id=call.message.message_id,
                                text="Ви обрали Головне меню.",
                                reply_markup=None)

# видалення клавіатури з підтвердженням/відмовою виходу з аккаунта модератора
def remote_reauth_confirm_markup(call):
    global auth_login_mod_user, auth_password_mod_user, flag
    if auth_login_mod_user == "" and auth_password_mod_user == "":
        # if is_moderator(call.message.chat.id):
            bot.edit_message_text(chat_id=call.message.chat.id,
message_id=call.message.message_id,
                                text="Виконано вихід з аккаунта модератора, до нових зустрічей!",
                                reply_markup=None)

        flag = 0
        error_company_method()
    else:
        bot.edit_message_text(chat_id=call.message.chat.id,
message_id=call.message.message_id,
                                text="Ви відмовились від виходу з аккаунта модератора, будьте
уважніше",
                                reply_markup=None)

        flag = 4
        error_company_method()

# видалення клавіатури з підтвердженням/відмовою виходу з аккаунта користувача
def remote_exit_confirm_markup(call):

```

```

        global message_contact, flag, moderator_chat_id, exit_confirm_user_flag, company
        if not exit_confirm_user_flag:
            bot.edit_message_text(chat_id=call.message.chat.id,
                                message_id=call.message.message_id,
                                text="Ви відмовились від виходу з аккаунта користувача, будьте
уважніше",
                                reply_markup=None)
            flag = 0
            error_company_method()

        else:
            bot.edit_message_text(chat_id=call.message.chat.id,
                                message_id=call.message.message_id,
                                text="Виконано вихід з аккаунта користувача, "
                                "заявка на видалення залишена модератору. До нових зустрічей!",
                                reply_markup=None)
            now = datetime.now()
            period = timedelta(hours=24)
            before = now + period
            for chat_id in moderator_chat_id:
                bot.send_message(chat_id, get_user_info_from_db_by_tel_id(call.message) +
                                'Даний користувач бажає видалити свій аккаунт. '
                                # '\nКомпанія: ' + company +
                                '\nЗаявка надійшла ' + str(now.day) + '!' + str(now.month) + '!' +
str(now.year)
                                + 'Г в ' + str(now.hour) + ':' + str(now.minute) +
                                '\nнеобхідно виконати видалення до ' + str(before.day) + '!' +
str(before.month) + '!'
                                + str(before.year) + 'Г в ' + str(before.hour) + ':' + str(before.minute)
                                )
                # bot.forward_message(chat_id, message.chat.id, message.message_id)
                # bot.send_message(chat_id, message_contact)

            flag = 0
            error_company_method()

# def update_company_by_telegram_id(message):
# global company
# if company != "":
# connection = create_connection()
# update_users = "UPDATE responsible_bot SET responsible_bot.company="
# + str(company) + " WHERE responsible_bot.telegram_id=" + str(message.chat.id)
# execute_query(connection, update_users)
# close_connection(connection)

# обробник натискання на кнопки inline-клавіатур
@bot.callback_query_handler(func=lambda call: True)
def callback_inline(call):
    global confirm_company_flag, confirm_reauth_flag, auth_login_mod_user,
auth_password_mod_user
    global exit_confirm_user_flag, register_exit_flag, install_need_where, flag,
answer_topic_number
    global what_account, action_account, current_title, forward_message_to_call_flag
    # print('here2')
    try:

```

```

if call.message:
    if call.data == 'company1':
        current_message = "43173964 - ООО «АЛД інжиніринг та будівництво»"
        remote_company_markup(call, current_message)
        confirm_company(call, current_message)
    elif call.data == 'company2':
        current_message = "42170810 - «АЛД-ПРОЕКТ»"
        remote_company_markup(call, current_message)
        confirm_company(call, current_message)
    elif call.data == 'company3':
        current_message = "42720802 - ООО «АЛД-ПРОМСТРОЙ»"
        remote_company_markup(call, current_message)
        confirm_company(call, current_message)
    elif call.data == 'company4':
        current_message = "41901054 - «АЛД-ПРОМСТРОЙ 2»"
        remote_company_markup(call, current_message)
        confirm_company(call, current_message)
    elif call.data == 'company5':
        current_message = "42805545 - «АЛД-СЕРВИС»"
        remote_company_markup(call, current_message)
        confirm_company(call, current_message)
    elif call.data == 'company6':
        current_message = "431739641 - «РБУ»"
        remote_company_markup(call, current_message)
        confirm_company(call, current_message)
    elif call.data == 'yes_company':
        confirm_company_flag = True

        remote_company_confirm_markup(call)
        phone(call.message)
    elif call.data == 'no_company':
        confirm_company_flag = True
        bot.edit_message_text(chat_id=call.message.chat.id,
                              message_id=call.message.message_id,
                              text="Ви можете змінити свій вибір.",
                              reply_markup=None)
        get_company_method(call.message)
    elif call.data == 'yes_reauth_mod':
        auth_login_mod_user = "
        auth_password_mod_user = "
        remote_reauth_confirm_markup(call)
    elif call.data == 'no_reauth_mod':
        remote_reauth_confirm_markup(call)
    elif call.data == 'yes_exit_user':
        exit_confirm_user_flag = True
        # print(message_contact)
        remote_exit_confirm_markup(call)
    elif call.data == 'no_exit_user':
        exit_confirm_user_flag = False
        remote_exit_confirm_markup(call)
    elif call.data == 'registr_user_command':
        register_exit_flag = 'register'
        remote_start_markup(call)
        get_company_method(call.message)
    elif call.data == 'exit_user_command':
        register_exit_flag = 'exit'
        remote_start_markup(call)

```

```

        exit_user_method(call.message)
    elif call.data == 'main_menu_command':
        register_exit_flag = 'main_menu'
        remote_start_markup(call)
        method_main_menu_user(call.message)

    elif call.data == 'po_1_level':
        text = "ПЗ"
        remote_main_menu_markup(call, text)
        method_menu_21_user(call.message) # ready
    elif call.data == 'tech_vopr_1_level':
        text = "Тех.питання"
        remote_main_menu_markup(call, text)
        method_menu_22_user(call.message) # ready
    elif call.data == 'other_1_level':
        text = "Інше"
        remote_main_menu_markup(call, text)
        current_title = "Інше"
        get_user_problem(call.message) # ready

    elif call.data == 'asu_21_level':
        text = "АСУ"
        remote_main_menu_markup(call, text)
        method_menu_211_user(call.message) # ready
    elif call.data == 'kompas_21_level':
        text = "КОМПАС"
        remote_main_menu_markup(call, text)
        method_menu_212_user(call.message) # ready
    elif call.data == 'poshta_21_level':
        text = "Пошта"
        remote_main_menu_markup(call, text)
        method_menu_213_user(call.message) # ready
    elif call.data == '1s_21_level':
        text = "1С"
        remote_main_menu_markup(call, text)
        method_menu_214_user(call.message) # ready
    elif call.data == 'control_21_level':
        text = "Control"
        remote_main_menu_markup(call, text)
        method_menu_215_user(call.message) # ready
    elif call.data == 'install_po_21_level':
        text = "Встановити ПЗ"
        remote_main_menu_markup(call, text) # ready
        method_menu_217_user(call.message)
    elif call.data == 'other_21_level':
        text = "Інше"
        remote_main_menu_markup(call, text)
        current_title = "ПЗ-Інше"
        get_user_problem(call.message) # ready
    elif call.data == 'back_21_level':
        text = "Назад"
        remote_main_menu_markup(call, text)
        method_main_menu_user(call.message)

    elif call.data == 'korp_teh_22_level':
        text = "Корпор.техніка"
        remote_main_menu_markup(call, text)

```

```

        method_menu_221_user(call.message) # ready
    elif call.data == 'telephony_22_level':
        text = "Телефонія"
        remote_main_menu_markup(call, text)
        method_menu_222_user(call.message) # ready
    elif call.data == 'other_22_level':
        text = "Інше"
        remote_main_menu_markup(call, text)
        current_title = "Тех.питання-Інше"
        get_user_problem(call.message) # ready
    elif call.data == 'internet_22_level':
        text = "Internet"
        remote_main_menu_markup(call, text)
        method_menu_223_user(call.message) # ready

    elif call.data == 'asu_web_211_level':
        text = "АСУ Web"
        remote_main_menu_markup(call, text)
        method_menu_2111_user(call.message) # ready
    elif call.data == 'asu_photo_211_level':
        text = "АСУ Фотофіксація"
        remote_main_menu_markup(call, text)
        method_menu_2112_user(call.message) # ready
    elif call.data == 'back_211_level':
        text = "Назад"
        remote_main_menu_markup(call, text)
        method_menu_21_user(call.message)

    elif call.data == 'kompas_web_212_level':
        text = "КОМПАС Web"
        remote_main_menu_markup(call, text)
        method_menu_2121_user(call.message) # ready
    elif call.data == 'kompas_mob_212_level':
        text = "КОМПАС Mobile"
        remote_main_menu_markup(call, text)
        method_menu_2122_user(call.message) # ready

    elif call.data == 'lost_passw_213_level':
        text = "Втрачено пароль"
        remote_main_menu_markup(call, text) # ready
        what_account = "Outlook"
        action_account = "отримання даних"
        get_user_position(call.message)
    elif call.data == 'other_213_level':
        text = "Інше"
        remote_main_menu_markup(call, text)
        current_title = "ПЗ-Пошта-Інше"
        get_user_problem(call.message) # ready

    elif call.data == 'access_214_level':
        text = "Доступ"
        remote_main_menu_markup(call, text)
        method_menu_2141_user(call.message) # ready
    elif call.data == 'tech_problem_214_level':
        text = "Тех.проблеми"
        remote_main_menu_markup(call, text)
        current_title = "ПЗ-1С-Технічні проблеми"

```

```

get_user_problem(call.message) # ready

elif call.data == 'tech_problem_215_level':
    text = "Тех.проблеми"
    remote_main_menu_markup(call, text)
    current_title = "ПЗ-Control-Технічні проблеми"
    get_user_problem(call.message) # ready

elif call.data == 'install_pc_217_level':
    text = "Встановити на ПК"
    remote_main_menu_markup(call, text) # ready
    install_need_where = "на ПК"
    get_user_need_install_pc(call.message)
elif call.data == 'install_mob_217_level':
    text = "Встановити на МОБІЛЬНИЙ"
    remote_main_menu_markup(call, text) # ready
    install_need_where = "на МОБІЛЬНИЙ"
    get_user_need_install_pc(call.message)
elif call.data == 'install_plan_217_level':
    text = "Встановити на ПЛАНШЕТ"
    remote_main_menu_markup(call, text) # ready
    install_need_where = "на ПЛАНШЕТ"
    get_user_need_install_pc(call.message)

elif call.data == 'work_place_221_level':
    text = "Робоче місце"
    remote_main_menu_markup(call, text)
    current_title = "Тех.питання-Корпоративна техніка-Робоче місце"
    get_user_problem(call.message) # ready
elif call.data == 'planshet_221_level':
    text = "Планшет"
    remote_main_menu_markup(call, text)
    current_title = "Тех.питання-Корпоративна техніка-Планшет"
    get_user_problem(call.message) # ready
elif call.data == 'other_221_level':
    text = "Інше"
    remote_main_menu_markup(call, text)
    current_title = "Тех.питання-Корпоративна техніка-Інше"
    get_user_problem(call.message) # ready
elif call.data == 'back_221_level':
    text = "Назад"
    remote_main_menu_markup(call, text)
    method_menu_22_user(call.message)

elif call.data == 'mobile_222_level':
    text = "Мобільна"
    remote_main_menu_markup(call, text)
    current_title = "Тех.питання-Телефонія-Мобільна"
    get_user_problem(call.message) # ready
elif call.data == 'ip_tel_222_level':
    text = "ІР-телефонія"
    remote_main_menu_markup(call, text)
    current_title = "Тех.питання-Телефонія-ІР-телефонія"
    get_user_problem(call.message) # ready

elif call.data == 'office_223_level':
    text = "Офіс"

```

```

remote_main_menu_markup(call, text)
current_title = "Тех.питання-Internet-Офіс"
get_user_problem(call.message) # ready
elif call.data == 'private_223_level':
    text = "Особистий"
    remote_main_menu_markup(call, text)
    current_title = "Тех.питання-Internet-Особистий"
    get_user_problem(call.message) # ready

elif call.data == 'instr_2111_level':
    text = "Інструкція"
    remote_main_menu_markup(call, text)
    method_menu_21111_user(call.message) # ready
elif call.data == 'access_asu_2111_level':
    text = "Доступ в АСУ"
    remote_main_menu_markup(call, text)
    method_menu_21112_user(call.message) # ready
elif call.data == 'tech_problem_2111_level':
    text = "Тех.проблеми"
    remote_main_menu_markup(call, text) # ready
    current_title = "ПЗ-АСУ-Web-Технічні проблеми"
    get_user_problem(call.message)
elif call.data == 'back_2111_level':
    text = "Назад"
    remote_main_menu_markup(call, text)
    method_menu_211_user(call.message)

elif call.data == 'get_asu_apk_2112_level':
    text = "Отримати посилання на застосунок"
    remote_main_menu_markup(call, text)
    send_asu_apk_url(call.message) # ready
elif call.data == 'tech_problem_2112_level':
    text = "Тех.проблеми"
    remote_main_menu_markup(call, text)
    current_title = "ПЗ-АСУ-АСУ Фотофіксація-Технічні проблеми"
    get_user_problem(call.message) # ready

elif call.data == 'instr_2121_level':
    text = "Інструкція"
    remote_main_menu_markup(call, text)
    method_menu_21211_user(call.message) # ready
elif call.data == 'access_kompas_2121_level':
    text = "Доступ в КОМПАС"
    remote_main_menu_markup(call, text)
    method_menu_21212_user(call.message) # ready
elif call.data == 'tech_problem_2121_level':
    text = "Тех.проблеми"
    remote_main_menu_markup(call, text)
    current_title = "ПЗ-КОМПАС-Web-Технічні проблеми"
    get_user_problem(call.message) # ready
elif call.data == 'back_2121_level':
    text = "Назад"
    remote_main_menu_markup(call, text)
    method_menu_212_user(call.message)

elif call.data == 'get_kompas_apk_2122_level':
    text = "Отримати посилання на застосунок"

```

```

        remote_main_menu_markup(call, text)
        send_kompas_apk_url(call.message) # ready
    elif call.data == 'tech_problem_2122_level':
        text = "Тех.проблеми"
        remote_main_menu_markup(call, text)
        current_title = "ПЗ-КОМПАС-Mobile-Технічні проблеми"
        get_user_problem(call.message) # ready

elif call.data == 'get_account_1s_2141_level':
    text = "Запросити обліковий запис"
    remote_main_menu_markup(call, text) # ready
    what_account = "1С"
    action_account = "доступ до"
    method_get_account1(call.message)
elif call.data == 'off_account_1s_2141_level':
    text = "Вимкнути обліковий запис"
    remote_main_menu_markup(call, text) # ready
    what_account = "1С"
    action_account = "вимикання"
    method_get_account(call.message)
elif call.data == 'other_2141_level':
    text = "Інше"
    remote_main_menu_markup(call, text)
    current_title = "ПЗ-1С-Доступ в 1С-Інше"
    get_user_problem(call.message) # ready
elif call.data == 'back_2141_level':
    text = "Назад"
    remote_main_menu_markup(call, text)
    method_menu_214_user(call.message)

elif call.data == 'get_instruct_asu_21111_level':
    text = "Отримати файл інструкції"
    remote_main_menu_markup(call, text) # ready
    send_pdf_file(call.message, 0)
elif call.data == 'get_video_asu_21111_level':
    text = "Отримати відео"
    remote_main_menu_markup(call, text) # ready
    send_mp4_file(call.message, 0)
elif call.data == 'back_21111_level':
    text = "Назад"
    remote_main_menu_markup(call, text)
    method_menu_2111_user(call.message)

elif call.data == 'get_account_asu_21112_level':
    text = "Запросити обліковий запис"
    remote_main_menu_markup(call, text) # ready
    what_account = "АСУ"
    action_account = "доступ до"
    method_get_account1(call.message)
elif call.data == 'off_account_asu_21112_level':
    text = "Вимкнути обліковий запис"
    remote_main_menu_markup(call, text) # ready
    what_account = "АСУ"
    action_account = "вимикання"
    method_get_account(call.message)
elif call.data == 'other_21112_level':
    text = "Інше"

```

```

remote_main_menu_markup(call, text)
current_title = "ПЗ-АСУ-Web-Доступ в АСУ-Інше"
get_user_problem(call.message) # ready

elif call.data == 'get_instruct_kompas_21211_level':
    text = "Отримати файл інструкції"
    remote_main_menu_markup(call, text) # ready
    send_pdf_file(call.message, 1)
elif call.data == 'get_video_kompas_21211_level':
    text = "Отримати відео"
    remote_main_menu_markup(call, text) # ready
    send_mp4_file(call.message, 1)
elif call.data == 'back_21211_level':
    text = "Назад"
    remote_main_menu_markup(call, text)
    method_menu_2121_user(call.message)

elif call.data == 'get_account_kompas_21212_level':
    text = "Запросити обліковий запис"
    remote_main_menu_markup(call, text) # ready
    what_account = "КОМПАС"
    action_account = "доступ до"
    method_get_account1(call.message)
elif call.data == 'off_account_kompas_21212_level':
    text = "Вимкнути обліковий запис"
    remote_main_menu_markup(call, text) # ready
    what_account = "КОМПАС"
    action_account = "вимикання"
    method_get_account(call.message)
elif call.data == 'other_21212_level':
    text = "Інше"
    remote_main_menu_markup(call, text)
    current_title = "ПЗ-КОМПАС-Web-Доступ в КОМПАС-Інше"
    get_user_problem(call.message) # ready

elif call.data == 'save_call':
    text = "Зберегти заявку"
    remote_main_menu_markup(call, text)
    update_call_by_photo_doc()
    send_confirm_send_call() # ready

elif call.data == 'continue_call':
    text = "ЗБЕРЕГТИ"
    remote_main_menu_markup(call, text)
    bot.send_message(call.message.chat.id, "Тепер всі повідомлення, які будуть
написані в цей чат, "
                                "я перешлю модератору. Продовжуйте спілкування!!")
    forward_message_to_call_flag = True
    answer_topic_number = "
    answer_topic_number = get_number_topic_from_text(call.message.text, "(", ")")
    # print('answer_topic_number = ' + answer_topic_number)
    get_message_user_to_forward_mod(call.message)
    # print(call.message.text)
    # update_call_by_photo_doc()
    # send_confirm_send_call() # ready
elif call.data == 'close_call':
    text = "ЗАКРИТИ ЗАЯВКУ"

```

```

remote_main_menu_markup(call, text)
forward_message_to_call_flag = False
answer_topic_number = "
answer_topic_number = get_number_topic_from_text(call.message.text, "(", ")")
# print('answer_topic_number = ' + answer_topic_number)
update_status_call_to_buffer_db(call.message.chat.id, answer_topic_number)
update_status_call_to_support_db(answer_topic_number)
notificate_close_call_for_mod(answer_topic_number, call.message.chat.id)

flag = 0

# update_call_by_photo_doc()
# send_confirm_send_call() # ready

except Exception as e:
    for user in moderator_chat_id:
        bot.send_message(user, "Error: " + str(e) + "в блоці коду callback_inline")
    # print(repr(e))

# метод для отримання номера заявки з повідомлення
# відбувається обрізка між двома символами - first_symb, second_symb
def get_number_topic_from_text(text, first_symb, second_symb):
    string = text.split(first_symb)[1]
    result = string.split(second_symb)[0]
    return result

def update_call_by_photo_doc():
    global current_call_list, current_call_code
    json_data = json.dumps(current_call_list)
    # print("here last list: " + str(current_call_list))
    connection = create_connection()
    if connection is not None:
        update_call = "UPDATE support_call SET support_call.files=\"" \
            + json_data + "\" WHERE support_call.topic=\"" + str(current_call_code)
        execute_query(connection, update_call)
        close_connection(connection)
        current_call_list.clear()
        current_call_code = None

# розсилка підтвердження збереження заявки(після натискання на ЗБЕРЕГТИ)
def send_confirm_send_call():
    global current_call_chat_id, current_call_text
    bot.send_message(current_call_chat_id, current_call_text)
    current_call_chat_id = None
    current_call_text = "

def phone(message):
    global flag
    keyboard = types.ReplyKeyboardMarkup(row_width=1, resize_keyboard=True,
one_time_keyboard=True)
    button_phone = types.KeyboardButton(text="Відправити номер телефону",
request_contact=True)
    keyboard.add(button_phone)

```

```

bot.send_message(message.chat.id,
                  'Для реєстрації вам необхідно надати свій номер телефона. '
                  'Він буде використаний виключно для реєстрації.',
                  reply_markup=keyboard)
flag = 3
error_company_method()

@bot.message_handler(content_types=['contact'])
def contact(message):
    global message_contact, moderator_chat_id, company
    if message.contact is not None:
        message_contact = message.contact # store phone number user
        global contact_flag
        contact_flag = True
        # print(message.contact)
        now = datetime.now()
        period = timedelta(hours=24)
        before = now + period
        for chat_id in moderator_chat_id:
            bot.send_message(chat_id, 'Даний користувач бажає зареєструватися.' +
                              '\nКомпанія: ' + company +
                              '\nЗаявка надійшла ' + str(now.day) + '.' + str(now.month) + '.' +
                              str(now.year)
                              + 'в ' + str(now.hour) + ':' + str(now.minute) +
                              '\nнеобхідно обробити до ' + str(before.day) + '.' + str(before.month) + '.'
                              + str(before.year) + 'в ' + str(before.hour) + ':' + str(before.minute)
                              )
            bot.send_message(chat_id, message.contact)
            markup = types.ReplyKeyboardRemove(selective=False)
            bot.send_message(message.chat.id, 'Дякую за реєстрацію, ваша заявка буде оброблена
            протягом 1 робочого дня.',
                              reply_markup=markup)
            # bot.send_message(chat_id, message, reply_markup=markup)

# допомога в аккаунті модератора
@bot.message_handler(commands=['help_mod'])
def help_mod(message):
    global flag, auth_login_mod_user, auth_password_mod_user
    # if auth_login_mod_user == " and auth_password_mod_user == "":
    if is_moderator(message.chat.id):
        bot.send_message(message.chat.id, 'Ось мої команди для модератора:'
                              '\n/help_mod - довідка'
                              '\n/auth - авторизація модератора'
                              '\n/reauth - вихід з аккаунта модератора'
                              '\n/add - схвалити заявку, додати користувача'
                              '\n/show - перегляд всіх зареєстрованих користувачів'
                              '\n/fail - відмовити в реєстрації'
                              '\n/remove - видалити вже зареєстрованого користувача'
                              '\n/answer - відповісти на повідомлення користувача'
                              '\n/answer_by_id - відповісти на повідомлення користувача по id'
                              '\n/send_all - розсилка повідомлення всім користувачам'
                              '\n/send_by_id - адресна відправка листа за id'
                              '\n/get_message - отримати повідомлення за id чата та
повідомлення')
    else:

```

```

        bot.send_message(message.chat.id, 'Ось мої команди:'
                                '\n/help_mod - довідка'
                                '\n/auth - авторизація модератора'
                                '\nПісля авторизації стануть доступні деякі інші команди.')

    flag = 4
    error_company_method()

# обробка команди отримання повідомлення за id користувача та id повідомлення
@bot.message_handler(commands=['get_message'])
def get_message_mod(message):
    global flag, auth_login_mod_user, auth_password_mod_user
    if is_moderator(message.chat.id):
        bot.register_next_step_handler(message, method_get_message_mod)
        bot.send_message(message.chat.id, 'Введіть id користувача та id повідомлення, '
                                           'яке бажаєте отримати, через пробіл:')

        flag = 5
    else:
        bot.send_message(message.chat.id,
                        'Для доступу до даної команди спочатку необхідно авторизуватися'
                        'аккаунтом модератора.')
        flag = 4
        error_company_method()

# метод обробки введених модератором id користувача та id повідомлення через пробіл
для отримання вмісту повідомлення
def method_get_message_mod(message):
    global flag
    try:
        id_user = "
        id_message = "
        id_user = message.text.split(" ")[0].strip()
        id_message = message.text.split(" ")[1].strip()
        bot.forward_message(message.chat.id, id_user, id_message)
        flag = 5
        error_company_method()
    except BaseException as e:
        bot.send_message(message.chat.id, 'Помилка: ' + str(e))

# розсилка модератором повідомлення на всіх зареєстрованих користувачів
@bot.message_handler(commands=['send_all'])
def send_all_mod(message):
    global flag, auth_login_mod_user, auth_password_mod_user
    # if auth_login_mod_user == " and auth_password_mod_user == ":
    if is_moderator(message.chat.id):
        user_step[message.chat.id] = SEND_ALL_USER_MOD
        bot.send_message(message.chat.id, 'Введіть повідомлення, яке необхідно відправити'
                                           'всім:')

        flag = 5
    else:
        bot.send_message(message.chat.id,
                        'Для доступу до даної команди спочатку необхідно авторизуватися'
                        'аккаунтом модератора.')
        flag = 4
        error_company_method()

```

```

# обробка введення повідомлення модератора для загальної розсилки
@bot.message_handler(func=lambda message: user_step.get(message.chat.id) ==
SEND_ALL_USER_MOD)
def get_password(message):
    global flag, exception
    if message.text != "":
        if message.text in exception:
            # print("here888")
            user_step[message.chat.id] = TEAM_USER_ACCEPTED
            flag = 10
            external_error(message)
        else:
            user_step[message.chat.id] = TEAM_USER_ACCEPTED # finish
            list_user = get_all_id_user_method()
            for user in list_user:
                bot.send_message(user, message.text)
                bot.send_message(message.chat.id, 'Всі користувачі отримали ваше повідомлення.')
                flag = 4
                error_company_method()
            else:
                flag = 4
                error_company_method()

def get_all_id_user_method():
    connection = create_connection()
    list_user = []
    if connection is not None:
        select_users = "SELECT telegram_id FROM responsible_train where delete1 is null and
telegram_id is not null"
        new_user = execute_read_query(connection, select_users)
        for e in new_user:
            list_user.append(e[0])
    return list_user

# авторизація модератора
@bot.message_handler(commands=['auth'])
def auth_mod(message):
    global flag, auth_login_mod_user, auth_password_mod_user
    # if auth_login_mod_user == " and auth_password_mod_user == "":
    if is_moderator(message.chat.id):
        bot.send_message(message.chat.id, 'Ви вже авторизовані аккаунтом модератора.')
        flag = 4
        error_company_method()
    else:
        bot.send_message(message.chat.id, 'Введіть логін:')
        flag = 5
        auth_login_mod_user = "
        error_company_method()
        # print("here auth return")

# вихід з аккаунта модератора
@bot.message_handler(commands=['reauth'])

```

```

def reauth_mod(message):
    global flag, auth_login_mod_user, auth_password_mod_user
    # if auth_login_mod_user != " and auth_password_mod_user != "":
    if is_moderator(message.chat.id):
        markup = types.InlineKeyboardMarkup(row_width=2) # yes/no
        item1 = types.InlineKeyboardButton("ТАК", callback_data='yes_reauth_mod')
        item2 = types.InlineKeyboardButton("НІ", callback_data='no_reauth_mod')
        markup.add(item1, item2)
        bot.send_message(message.chat.id, 'Ви впевнені, що бажаєте вийти з аккаунта
модератора?', reply_markup=markup)
        flag = 6
        error_company_method()
    else:
        bot.send_message(message.chat.id, 'Для виходу з аккаунта модератора в ньому
спочатку необхідно авторизуватися.')
        flag = 4
        error_company_method()

# обробка введення пароля модератора
@bot.message_handler(func=lambda message: user_step.get(message.chat.id) ==
TEAM_USER_LOGGING)
def get_password(message):
    global flag, auth_password_mod_user, auth_password_mod_regular
    # print(message.text)
    if message.text != "":
        if message.text == auth_password_mod_regular:
            auth_password_mod_user = message.text
            user_step[message.chat.id] = TEAM_USER_ACCEPTED
            bot.send_message(message.chat.id, 'Ось мої команди для модератора:'
                '\n/help_mod - довідка'
                '\n/auth - авторизація модератора'
                '\n/reauth - вихід з аккаунта модератора'
                '\n/add - схвалити заявку, додати користувача'
                '\n/show - перегляд всіх зареєстрованих користувачів'
                '\n/fail - відмовити в реєстрації'
                '\n/remove - видалити вже зареєстрованого користувача'
                '\n/answer - відповісти на повідомлення користувача'
                '\n/answer_by_id - відповісти на повідомлення користувача за
id'
                '\n/send_all - розсилка повідомлення всім користувачам'
                '\n/send_by_id - адресна відправка листа за id'
                '\n/get_message - отримати повідомлення за id чата та
повідомлення')
            flag = 4
            error_company_method()
        else:
            bot.send_message(message.chat.id,
                'Помилка авторизації. Введено невірний пароль модератора. Спробуйте
знову:')

# показати всіх користувачів модератору
@bot.message_handler(commands=['show'])
def show_mod(message):
    global flag, auth_login_mod_user, auth_password_mod_user
    # if auth_login_mod_user != " and auth_password_mod_user != "":

```

```

if is_moderator(message.chat.id):
    # C:\USR\www\pythonProject2 --in saas01
    # C:/Users/User/PycharmProjects/pythonProject2/user.txt --in local PK
    connection = create_connection()
    if connection is not None:
        select_users = "SELECT id, surname, name, patronymic, telegram_id, phone_number
FROM responsible_train where delete1 is null and telegram_id is not null"
        users = execute_read_query(connection, select_users)
        new_string = ""
        for row in users:
            new_string += str(row) + "\n"
        # print(str(new_user))
        bot.send_message(message.chat.id, 'В переліку зареєстрованих користувачів ботом
зараз знаходяться: \n'
                                + new_string)
        close_connection(connection)
        flag = 4
        error_company_method()
    else:
        bot.send_message(message.chat.id,
                        'Для доступу до даної команди спочатку необхідно авторизуватися
аккаунтом модератора.')
        flag = 0
        error_company_method()

def exit_user_method(message):
    global flag, message_contact
    if is_register_user(message.chat.id):
        markup = types.InlineKeyboardMarkup(row_width=2) # yes/no
        item1 = types.InlineKeyboardButton("ТАК", callback_data='yes_exit_user')
        item2 = types.InlineKeyboardButton("НІ", callback_data='no_exit_user')
        markup.add(item1, item2)
        bot.send_message(message.chat.id, 'Ви дійсно бажаєте вийти з аккаунта?',
                        reply_markup=markup)
        flag = 7
        error_company_method()
    else:
        bot.send_message(message.chat.id, 'Для доступу до даної команди спочатку необхідно
зареєструватися '
                                'та дочекатись схвалення заявки.')
        flag = 0
        error_company_method()

# видалити аккаунт користувача(залишити заявку на видалення)
@bot.message_handler(commands=['exit'])
def exit_user(message):
    exit_user_method(message)

# додавання користувача модератором після заявки
@bot.message_handler(commands=['add'])
def add_user_mod(message):
    global flag, auth_login_mod_user, auth_password_mod_user
    # if auth_login_mod_user != " and auth_password_mod_user != "
    if is_moderator(message.chat.id):

```

```

user_step[message.chat.id] = ADD_USER_SUR_MOD_ACCEPT
bot.reply_to(message, "Введіть прізвище користувача або номер телефона:")
flag = 4
error_company_method()
else:
    bot.send_message(message.chat.id,
        'Для доступу до даної команди спочатку необхідно авторизуватися
аккаунтом модератора.')
    flag = 0
    error_company_method()

# метод повертає співробітників з бази за прізвищем
def method_show_people_from_db(message):
    new_string = ""
    global flag
    connection = create_connection()
    if connection is not None:
        select_users = 'SELECT id, surname, name, patronymic, phone_number ' \
            'FROM responsible_train ' \
            'where delete1 is null and telegram_id is null and surname LIKE "%' + message +
'%" ' \
            'OR phone_number LIKE "%' +
message + '%" '
        users = execute_read_query(connection, select_users)
        if users:
            # print(users)
            for row in users:
                new_string += str(row) + "\n"
            # print(new_string)
            flag = 4
        return new_string

# обробка введення прізвища користувача/запроса id для додавання модератором
@bot.message_handler(func=lambda message: user_step.get(message.chat.id) ==
ADD_USER_SUR_MOD_ACCEPT)
def get_sur_user_add_mod(message):
    global flag
    # print(message.text)
    if message.text != "":
        result = method_show_people_from_db(message.text)
        if result != "":
            user_step[message.chat.id] = ADD_USER_ID_MOD_ACCEPT
            bot.send_message(message.chat.id, 'За вашим запитом знайдено наступних
співробітників:\n'
            + result + "Введіть ID користувача з цього переліку, якого треба
дати:")
            # flag = 4
        else:
            bot.send_message(message.chat.id, 'Нічого не знайдено за вашим запитом.
Спробуйте знову..')
            # flag = 4

# обробка введення id користувача для додавання модератором

```

```

@bot.message_handler(func=lambda message: user_step.get(message.chat.id) ==
ADD_USER_ID_MOD_ACCEPT)
def get_id_user_add_mod(message):
    global flag, input_user_id_for_update_add_mod
    # print(message.text+"here")
    if message.text != "":
        connection = create_connection()
        if connection is not None:
            select_users = "SELECT id FROM responsible_train where delete1 is null and
telegram_id is null"
            new_user = execute_read_query(connection, select_users)
            list_user = []
            for e in new_user:
                list_user.append(e[0])
            # print(list_user)
            if int(message.text) in list_user:

                input_user_id_for_update_add_mod = message.text
                # print("input_user_id_for_update_add_mod="
                +
input_user_id_for_update_add_mod)
                bot.send_message(message.chat.id, "Введіть TELEGRAM_ID користувача, якого
треба додати:")
                user_step[message.chat.id] = ADD_USER_TEL_ID_MOD_ACCEPT
                flag = 4
            else:
                bot.send_message(message.chat.id, 'користувача з таким id не існує. Спробуйте
знову..')
                flag = 4

# обробка введення telegram_id користувача для додавання модератором
@bot.message_handler(func=lambda message: user_step.get(message.chat.id) ==
ADD_USER_TEL_ID_MOD_ACCEPT)
def get_tel_id_user_add_mod(message):
    global flag, input_user_id_for_update_add_mod
    # print(message.text)
    if message.text != "":
        connection = create_connection()
        if connection is not None:
            select_users = "SELECT telegram_id FROM responsible_train where delete1 is null and
telegram_id is not null"
            new_user = execute_read_query(connection, select_users)
            list_user = []
            for e in new_user:
                list_user.append(e[0])
            # print(list_user)
            if message.text in list_user:
                bot.send_message(message.chat.id, 'Даний telegram_id вже зареєстровано. '
                'Я його знаю..Спробуйте знову')
            else:
                try:
                    bot.send_message(message.text, 'Ваша заявка на реєстрацію схвалена, дякую за
співробітництво!')

                    connection = create_connection()
                    if connection is not None:

```

```

        update_users = "UPDATE responsible_train SET
responsible_train.telegram_id=" \
        + message.text + " WHERE responsible_train.id=" +
input_user_id_for_update_add_mod
        execute_query(connection, update_users)
        close_connection(connection)
        bot.send_message(message.chat.id,
            "Введіть компанію, до якої належить користувач:")
        user_step[message.chat.id] = ADD_USER_COMPANY_MOD_ACCEPT
        flag = 4
        flag = 4
        error_company_method()
    except BaseException as e:
        bot.send_message(message.chat.id, 'Помилка!!!Не вдалось відправити вітальне
повідомлення '
                                'на вказаний id користувача. Перевірте, будь ласка, '
                                'правильність наступного id: ' + message.text)

    # print(e)
    flag = 4
    error_company_method()

# обробка введення компанії користувача для додавання модератором
@bot.message_handler(func=lambda message: user_step.get(message.chat.id) ==
ADD_USER_COMPANY_MOD_ACCEPT)
def get_comp_user_add_mod(message):
    global flag, input_user_id_for_update_add_mod
    if message.text != "":
        connection = create_connection()
        if connection is not None:
            update_users = "UPDATE responsible_train SET responsible_train.company=" \
                + message.text + " WHERE responsible_train.id=" +
input_user_id_for_update_add_mod
            execute_query(connection, update_users)
            close_connection(connection)
            bot.send_message(message.chat.id, 'Тепер я точно знаю цього користувача!')
            method_main_menu_user(message)
            method_main_menu_replay_user(message.text)
            input_user_id_for_update_add_mod = ""
            user_step[message.chat.id] = TEAM_USER_ACCEPTED
            flag = 4

# видалення користувача модератором після заявки
@bot.message_handler(commands=['remove'])
def remove_user_mod(message):
    global flag, auth_login_mod_user, auth_password_mod_user
    # if auth_login_mod_user != " and auth_password_mod_user != " :
    if is_moderator(message.chat.id):
        user_step[message.chat.id] = REMOVE_USER_MOD_ACCEPT
        bot.reply_to(message, "Введіть ID користувача, якого мені необхідно видалити:")
        flag = 4
        error_company_method()
    else:
        bot.send_message(message.chat.id,
            'Для доступу до даної команди спочатку необхідно авторизуватися
аккаунтом модератора.')

```

```

flag = 0
error_company_method()

# обробка введення id користувача для видалення модератором
@bot.message_handler(func=lambda message: user_step.get(message.chat.id) ==
REMOVE_USER_MOD_ACCEPT)
def get_id_user_remove_mod(message):
    global flag
    # print(message.text)
    if message.text != "":
        connection = create_connection()
        if connection is not None:
            select_users = "SELECT telegram_id FROM responsible_train where delete1 is null and
telegram_id is not null"
            new_user = execute_read_query(connection, select_users)
            list_user = []
            for e in new_user:
                list_user.append(e[0])
            close_connection(connection)
            if int(message.text) in list_user:
                connection = create_connection()
                if connection is not None:
                    update_users = "UPDATE responsible_train SET responsible_train.delete1=1
WHERE responsible_train.telegram_id=" \
+ message.text
                    execute_query(connection, update_users)
                    close_connection(connection)
                try:
                    bot.send_message(message.text, 'Ви були видалені, дякую за
співробітництво!')
                    bot.send_message(message.chat.id, 'Я більше не знаю цього користувача!')
                    user_step[message.chat.id] = TEAM_USER_ACCEPTED
                    flag = 4
                    error_company_method()
                except BaseException as e:
                    bot.send_message(message.chat.id, 'Помилка!!!Не удалось відправити
повідомлення про видалення '
                                'на вказанный id користувача. id було видалено, але '
                                'перевірте, будь ласка, правильність '
                                'наступного id: ' + message.text)
                    # print(e)
                    flag = 4
                    error_company_method()
            else:
                bot.send_message(message.chat.id, 'Даний користувач незареєстрований...')
                flag = 4
                error_company_method()

# видалення користувача модератором після заявки
@bot.message_handler(commands=['fail'])
def fail_user_mod(message):
    global flag, auth_login_mod_user, auth_password_mod_user
    # if auth_login_mod_user != " and auth_password_mod_user != "":
    if is_moderator(message.chat.id):
        user_step[message.chat.id] = FAIL_USER_MOD_ACCEPT

```

```

        bot.reply_to(message, "Введіть ID користувача, якому я повинен відмовити в
реєстрації:")
        flag = 4
        error_company_method()
    else:
        bot.send_message(message.chat.id, 'Для доступу до даної команди спочатку '
                                          'необхідно авторизуватися аккаунтом модератора.')
        flag = 0
        error_company_method()

# обробка введення id користувача для додавання модератором
@bot.message_handler(func=lambda message: user_step.get(message.chat.id) ==
FAIL_USER_MOD_ACCEPT)
def get_id_user_add_mod(message):
    global flag
    # print(message.text)
    if message.text != "":
        try: # если Введено валидный id(удається відправити на него повідомлення) то
відбувається обробка
            # иначе выпадем в ошибку(предусмотрена обробка со спамом модератору)
            bot.send_message(message.text, 'Ваша заявка на реєстрацію відхилена. '
                                          '\nМабуть, наш модератор не зміг з вами поспілкуватися, '
                                          'або не було встановлено вашу приналежність до обраної вами
компанії. '
                                          '\nДякую за співробітництво!')
            bot.send_message(message.chat.id, 'Я відхилив заявку на реєстрацію цього
користувача!')
            user_step[message.chat.id] = TEAM_USER_ACCEPTED
            flag = 4
            error_company_method()
        except BaseException as e:
            bot.send_message(message.chat.id,
                              'Помилка!!! ' + str(e) + ' Не вдалось відправити повідомлення з
відхиленням заявки '
                              'на вказанный id користувача. Перевірте, будь ласка,
правильність '
                              'наступного id: ' + message.text)
            # print(e)
            flag = 4
            error_company_method()

def train_sql(message):
    connection = create_connection()
    if connection is not None:
        select_users = "SELECT telegram_id FROM responsible_train"
        users = execute_read_query(connection, select_users)
        # for user in users:
        # print(user)

# метод закриття підключення до бд
def close_connection(connection):
    connection.close()

```

```

# метод створення підключення до бд
def create_connection(host_name='localhost', user_name='root', user_password='root',
db_name='support'):
    connection = None
    try:
        connection = mysql.connector.connect(
            host=host_name,
            user=user_name,
            passwd=user_password,
            database=db_name)
        # print("Connection to MySQL DB successful")
    except Error as e:
        bot.send_message(moderator_chat_id[0], "Error: " + str(e) + "в блоці коду
create_connection")
        # print(f"The error '{e}' occurred")
    return connection

# метод виконання запросу на читання до бд
def execute_read_query(connection, query):
    cursor = connection.cursor()
    result = None
    try:
        cursor.execute(query)
        result = cursor.fetchall()
    except Error as e:
        bot.send_message(moderator_chat_id[0], "Error: " + str(e) + "в блоці коду
execute_read_query")
        # print(f"The error '{e}' occurred")
    return result

def execute_query(connection, query):
    cursor = connection.cursor()
    try:
        cursor.execute(query)
        connection.commit()
        # print("Query executed successfully")
    except Error as e:
        bot.send_message(moderator_chat_id[0], "Error: " + str(e) + "в блоці коду
execute_query")
        # print(f"The error '{e}' occurred")

def create_connection(host_name='localhost', user_name='root', user_password='root',
db_name='support'):
    connection = None
    try:
        connection = mysql.connector.connect(
            host=host_name,
            user=user_name,
            passwd=user_password,
            database=db_name)
        # print("Connection to MySQL DB successful")
    except Error as e:
        bot.send_message(moderator_chat_id[0], "Error: " + str(e) + "в блоці коду
create_connection")

```

```

        # print(f"The error '{e}' occurred")
    return connection

# метод виконання запроса на читання до бд
def execute_read_query(connection, query):
    cursor = connection.cursor()
    result = None
    try:
        cursor.execute(query)
        result = cursor.fetchall()
    except Error as e:
        bot.send_message(moderator_chat_id[0], "Error: " + str(e) + "в блоці коду
execute_read_query")
        # print(f"The error '{e}' occurred")
    return result

def execute_query(connection, query):
    cursor = connection.cursor()
    try:
        cursor.execute(query)
        connection.commit()
        # print("Query executed successfully")
    except Error as e:
        bot.send_message(moderator_chat_id[0], "Error: " + str(e) + "в блоці коду
execute_query")
        # print(f"The error '{e}' occurred")

def create_connection(host_name='localhost', user_name='root', user_password='root',
db_name='support'):
    connection = None
    try:
        connection = mysql.connector.connect(
            host=host_name,
            user=user_name,
            passwd=user_password,
            database=db_name)
        # print("Connection to MySQL DB successful")
    except Error as e:
        bot.send_message(moderator_chat_id[0], "Error: " + str(e) + "в блоке кода
create_connection")
        # print(f"The error '{e}' occurred")
    return connection

# метод виконання запроса на читання до бд
def execute_read_query(connection, query):
    cursor = connection.cursor()
    result = None
    try:
        cursor.execute(query)
        result = cursor.fetchall()
    except Error as e:
        bot.send_message(moderator_chat_id[0], "Error: " + str(e) + "в блоке кода
execute_read_query")

```

```

        # print(f"The error '{e}' occurred")
    return result

def execute_query(connection, query):
    cursor = connection.cursor()
    try:
        cursor.execute(query)
        connection.commit()
        # print("Query executed successfully")
    except Error as e:
        bot.send_message(moderator_chat_id[0], "Error: " + str(e) + "в блоке кода
execute_query")

while True:
    try:
        bot.polling(none_stop=True, interval=0, timeout=100) # изначально была только эта
строка
    except Exception as e:
        bot.send_message(moderator_chat_id[0], "Error: " + str(e))
        pygame.time.wait(15000)
        # time.sleep(15)

```

ДОДАТОК Б
Слайди презентації

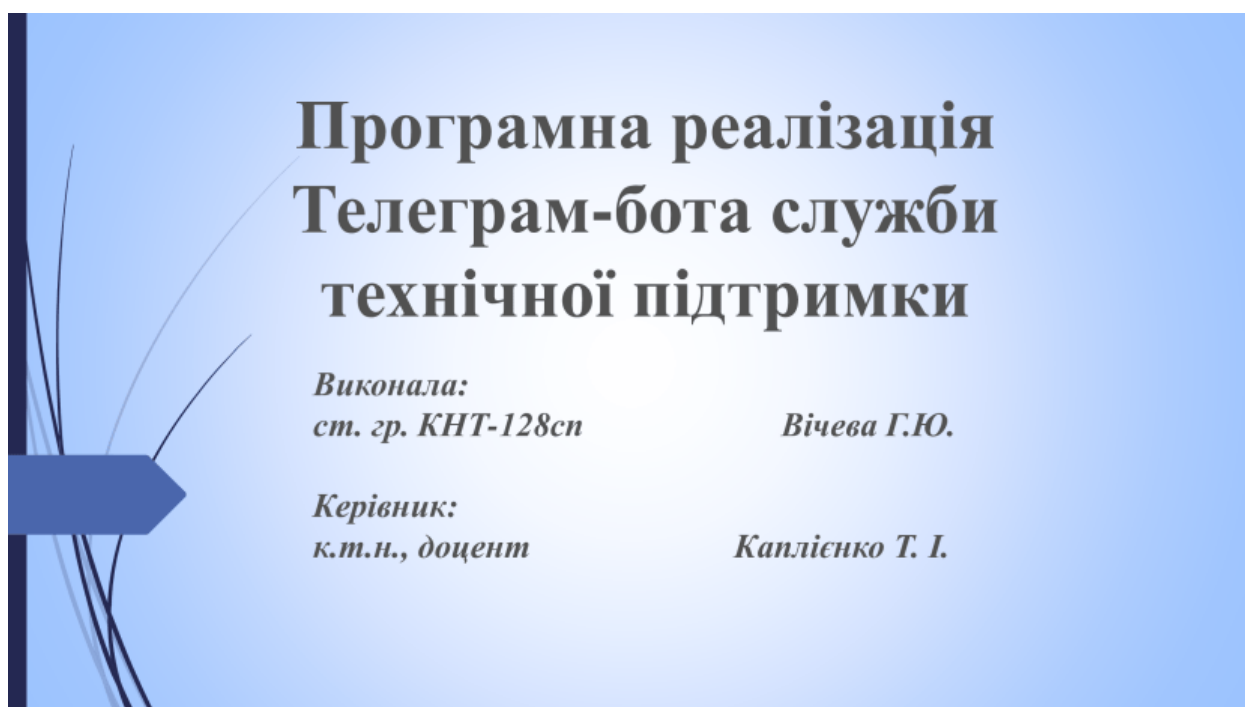


Рисунок Б.1 – Слайд 1 презентації

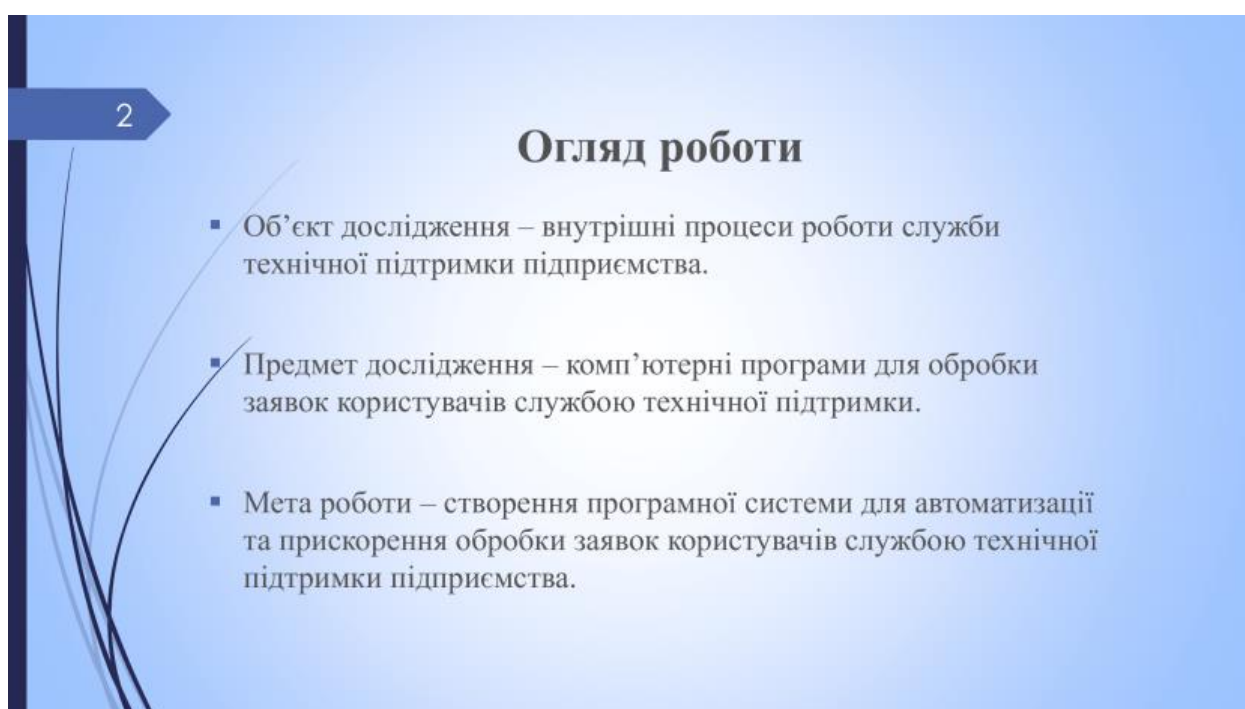


Рисунок Б.2 – Слайд 2 презентації

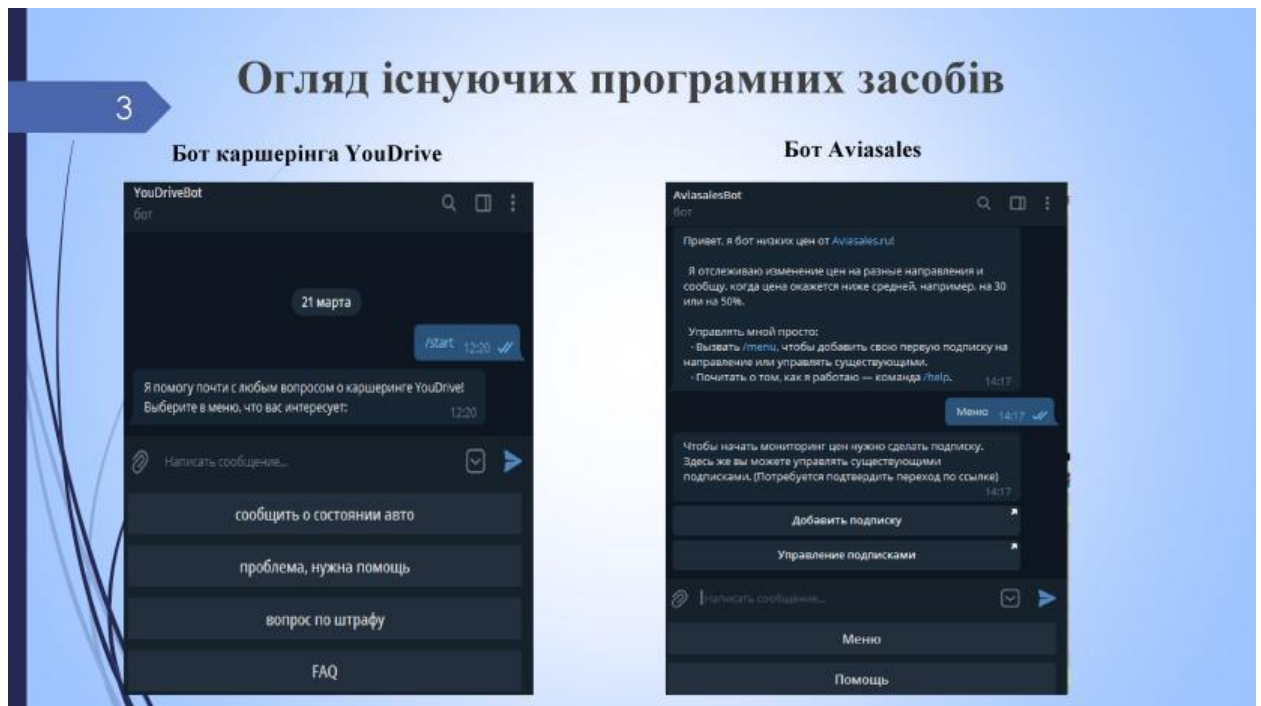


Рисунок Б.3 – Слайд 3 презентації



Рисунок Б.4 – Слайд 4 презентації

Структура меню користувача

6

```

graph LR
    HM[Головне меню] --> TP[Технічні питання]
    HM --> PZ[Програми забезпечення]
    HM --> I[Інше]

    TP --> RM[Робочі місця]
    TP --> T[Телефони]
    TP --> I2[Інше]

    RM --> P[Підприємство]
    RM --> I3[Інше]

    P --> CM[Корпоративна мережа]
    P --> I4[Інше]

    T --> T2[Телефони]
    T --> IP[IP-телефони]
    T --> M[Мобільні]
    T --> S[Світ]
    T --> O[Особисті]

    I2 --> I5[Інтернет]
    I2 --> I6[Інше]

    PZ --> ASU[АСУ]
    PZ --> KOMPAS[КОМПАС]
    PZ --> P2[Політи]
    PZ --> TS[ТС]
    PZ --> C[Control]
    PZ --> VSPZ[Встановити ПЗ]

    ASU --> I7[Інструкція]
    ASU --> DA[Доступ в АСУ]
    ASU --> TP3[Технічні проблеми]

    I7 --> OFI[Оформити файл інструкції]
    I7 --> VI[Відкрити інструкцію]

    DA --> ZOBZ[Запросити обліковий запис]
    DA --> VOBZ[Відновити обліковий запис]
    DA --> I8[Інше]

    KOMPAS --> I9[Інструкція]
    KOMPAS --> DKOMPAS[Доступ в КОМПАС]
    KOMPAS --> TP4[Технічні проблеми]

    I9 --> OFI2[Оформити файл інструкції]
    I9 --> VI2[Відкрити інструкцію]

    DKOMPAS --> ZOBZ2[Запросити обліковий запис]
    DKOMPAS --> VOBZ2[Відновити обліковий запис]
    DKOMPAS --> I9a[Інше]

    P2 --> VPO[Враховано партію Outlook]
    P2 --> I10[Інше]

    TS --> DTS[Доступ в ТС]
    TS --> TP5[Технічні проблеми]
    TS --> I11[Інше]

    DTS --> ZOBZ3[Запросити обліковий запис]
    DTS --> VOBZ3[Відновити обліковий запис]

    C --> TP6[Технічні проблеми]

    VSPZ --> NP[На планшет]
    VSPZ --> NT[На телефон]
    VSPZ --> NPK[На ПК]
  
```

Рисунок Б.6 – Слайд 6 презентації

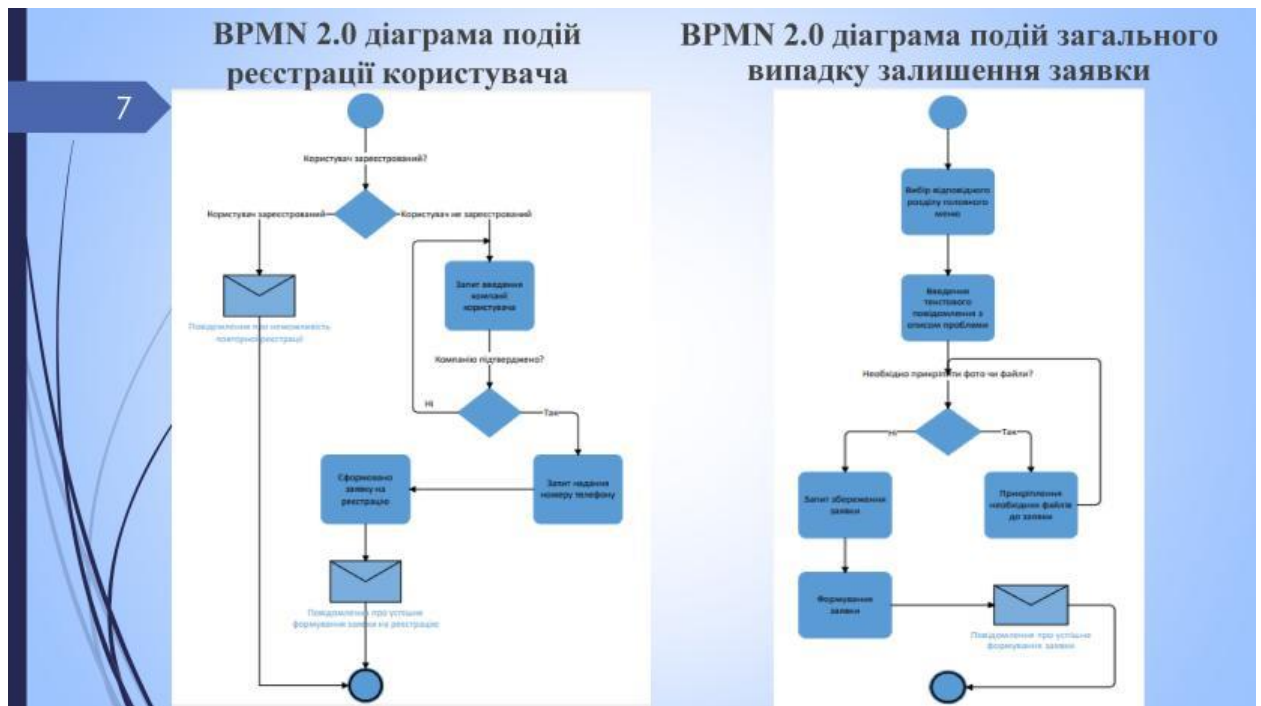


Рисунок Б.7 – Слайд 7 презентації



Рисунок Б.8 – Слайд 8 презентації

Обґрунтування вибору платформи

9

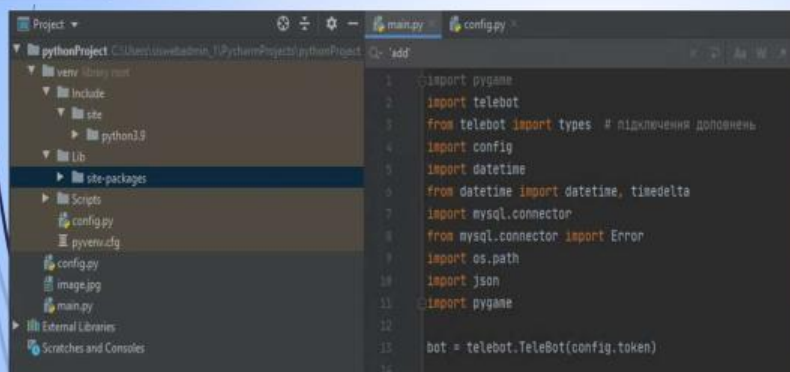
Порівняння можливостей Viber і Telegram [16]

Можливості месенджерів	Viber	Telegram
Відправлення повідомлень і синхронізація з усіма пристроями	+	+
Застосунок для робочого столу	+	+
Використання ніка	-	+
Підключення декількох акаунтів	-	+
Створення групового чату	+	+
Розширення групового чату (для участі більшої кількості людей)	-	+
Можливість звертання до обраного співрозмовнику через "@"	-	+
Створення каналу	+	+
Хмарне зберігання даних	-	+
Безпека	+	++
Використання ботів	+	+

Рисунок Б.9 – Слайд 9 презентації

Структура проєкту

10



До проєкту підключено додаткові бібліотеки:

- telebot – взаємодія з API Telegram;
- datetime – робота з датою та часом;
- mysql.connector – робота з базою даних та обробка помилок;
- os.path – робота з файловою системою пристрою;
- json – робота з типом даних json;
- pygame – затримка у часі.

Рисунок Б.10 – Слайд 10 презентації

11

Структура таблиці support_call_bot_log бази даних

Таблиця support_call_bot_log містить поля:

- telegram_id - автор повідомлення (заявки);
- message_id - унікальне в межах одного чату id повідомлення;
- topic_call - номер заявки в технічну підтримку;
- status – статус поточної заявки;
- once - ознака того, що модератор вже одного разу відповідав на це повідомлення;
- child - null якщо батьківська заявка, 1 якщо це продовження спілкування в рамках поточної заявки;
- mod_telegram_id - телеграм id модератора, який відповів на це повідомлення;
- mod_message_id - телеграм id повідомлення, яким відповів модератор;
- support_call_id - id заявки з таблиці технічної підтримки іншого ПП;
- date_telegram - поле для зберігання дати відправлення повідомлення, збігаються у користувача і модератора.

Рисунок Б.11 – Слайд 11 презентації

12

Команди

Для користувача передбачена обробка наступних команд:

- «/start» - вітання;
- «/help» - отримання довідки по командам;
- «/registration» - реєстрація користувачів;
- «/exit» - видалення зареєстрованого користувача з бота.

Для модератора передбачена обробка наступних команд:

- «/help_mod» - довідка;
- «/auth» - авторизація модератора;
- «/reauth» - вихід з аккаунта модератора;
- «/add» - схвалити заявку, додати користувача;
- «/show» - перегляд всіх зареєстрованих користувачів;
- «/fail» - відмовити в реєстрації;
- «/remove» - видалити вже зареєстрованого користувача;
- «/answer» - відповісти на повідомлення користувача;
- «/answer_by_id» - відповісти на повідомлення користувача по id;
- «/send_all» - розсилка повідомлення всім користувачам;
- «/send_by_id» - адресна відправка листа за id;
- «/get_message» - отримати повідомлення за id чату і повідомлення.

Рисунок Б.12 – Слайд 12 презентації

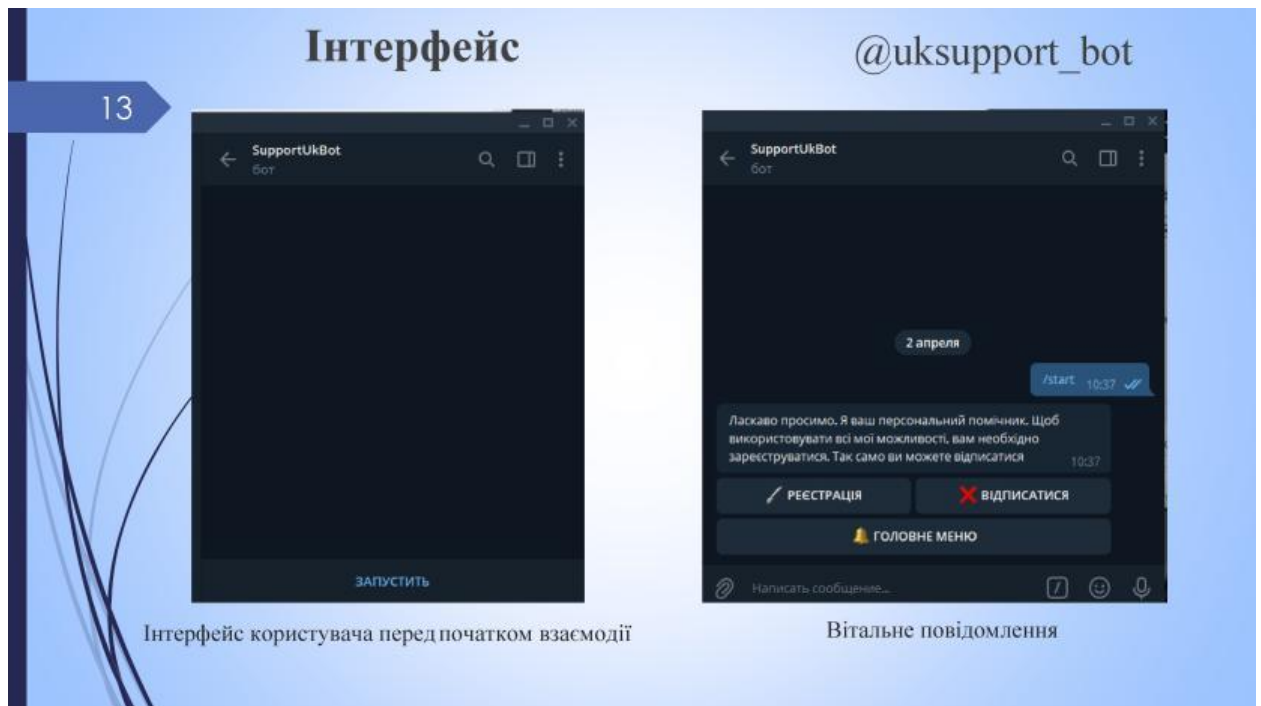


Рисунок Б.13 – Слайд 13 презентації

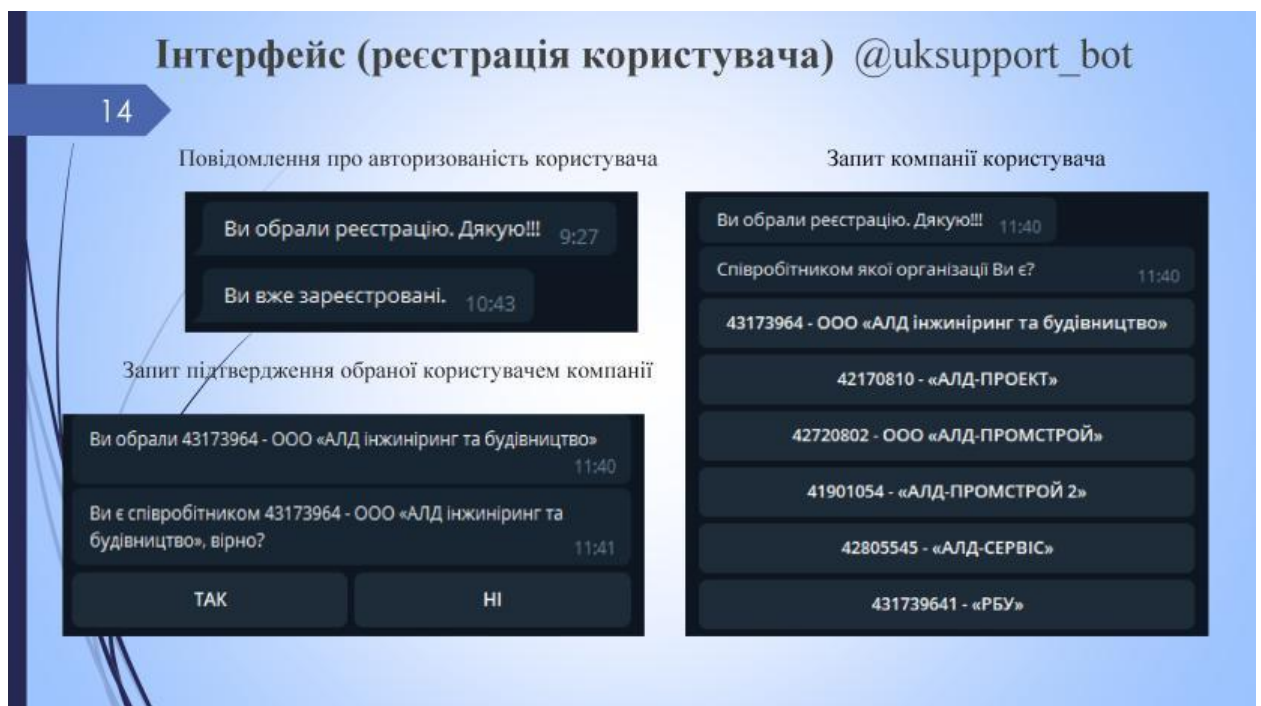


Рисунок Б.14 – Слайд 14 презентації

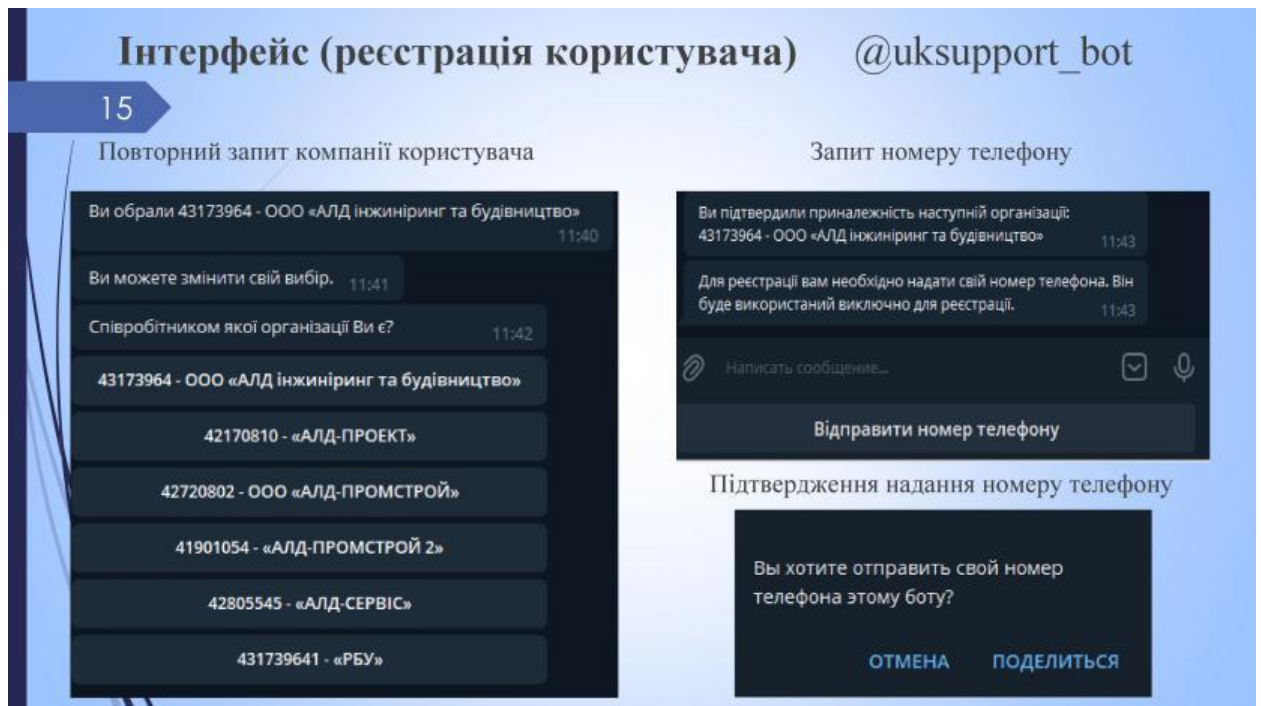


Рисунок Б.15 – Слайд 15 презентації

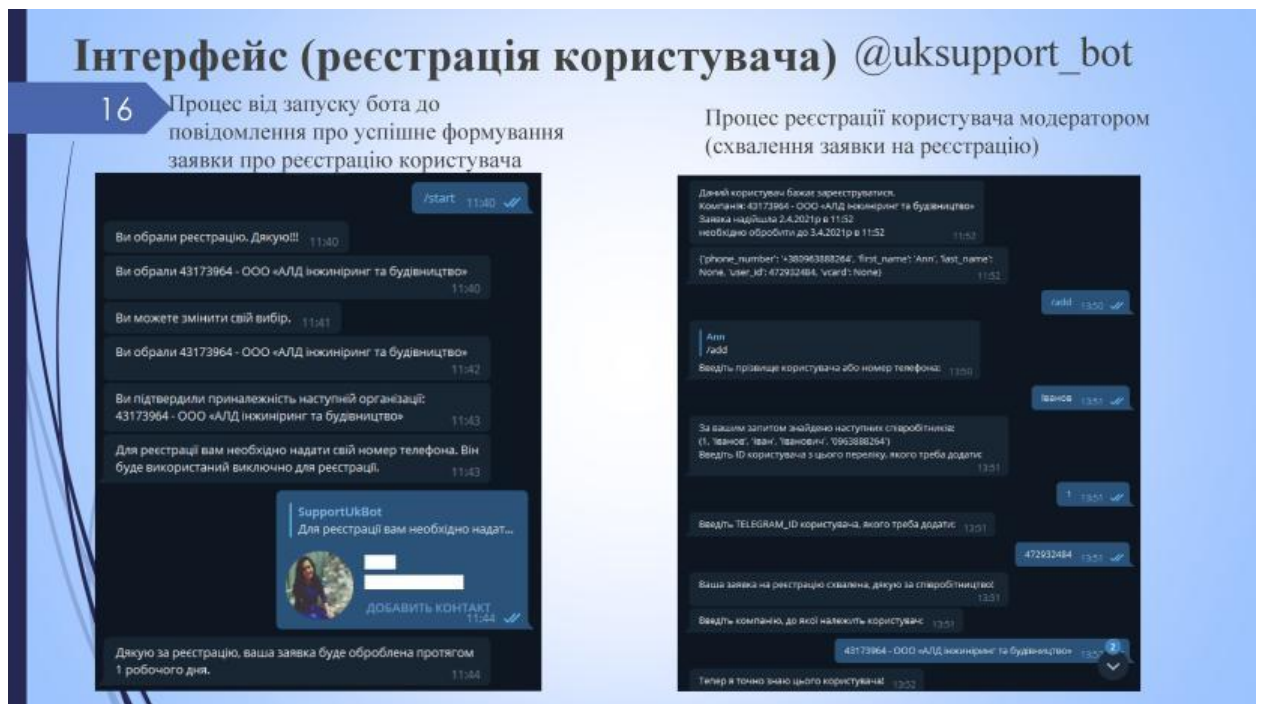


Рисунок Б.16 – Слайд 16 презентації

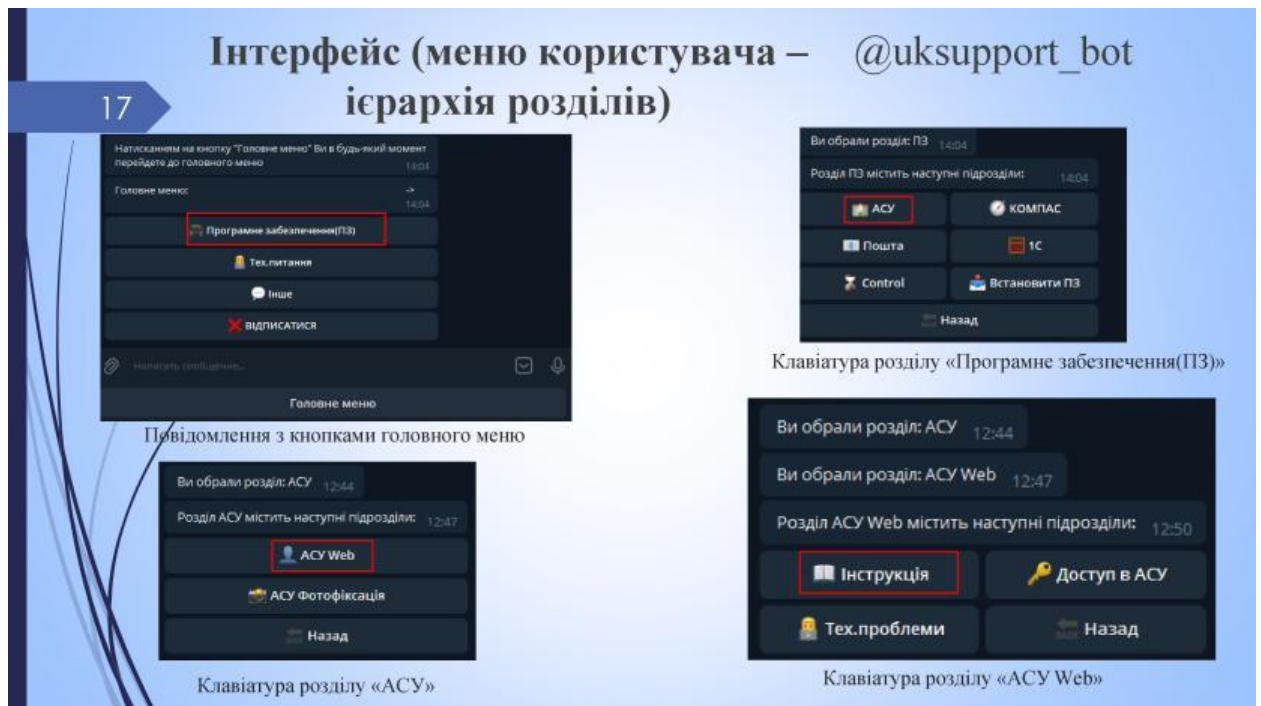


Рисунок Б.17 – Слайд 17 презентації

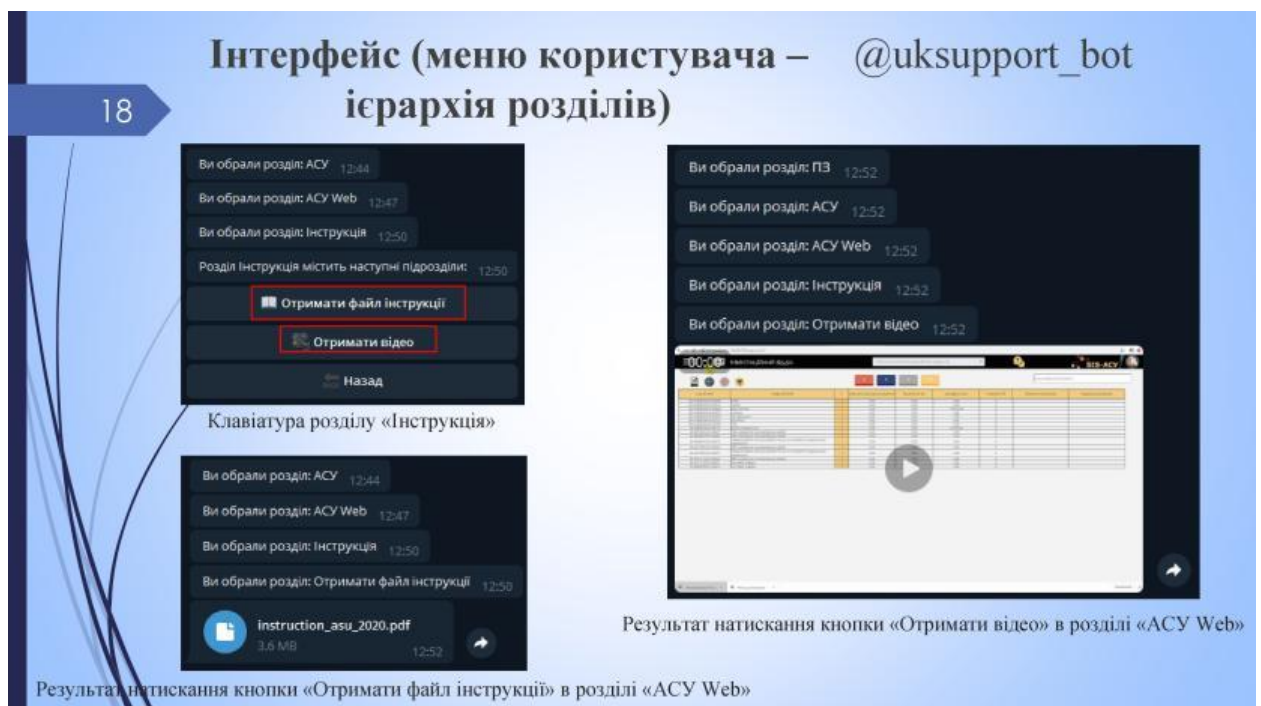


Рисунок Б.18 – Слайд 18 презентації

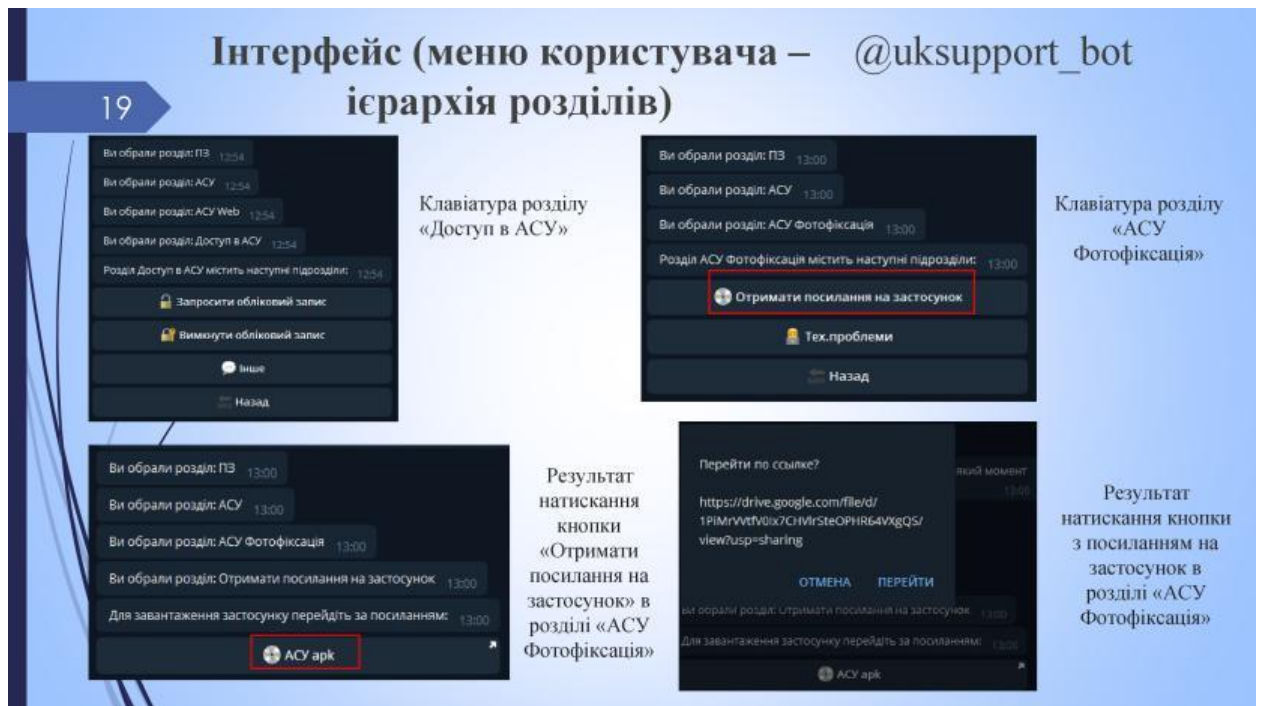


Рисунок Б.19 – Слайд 19 презентації

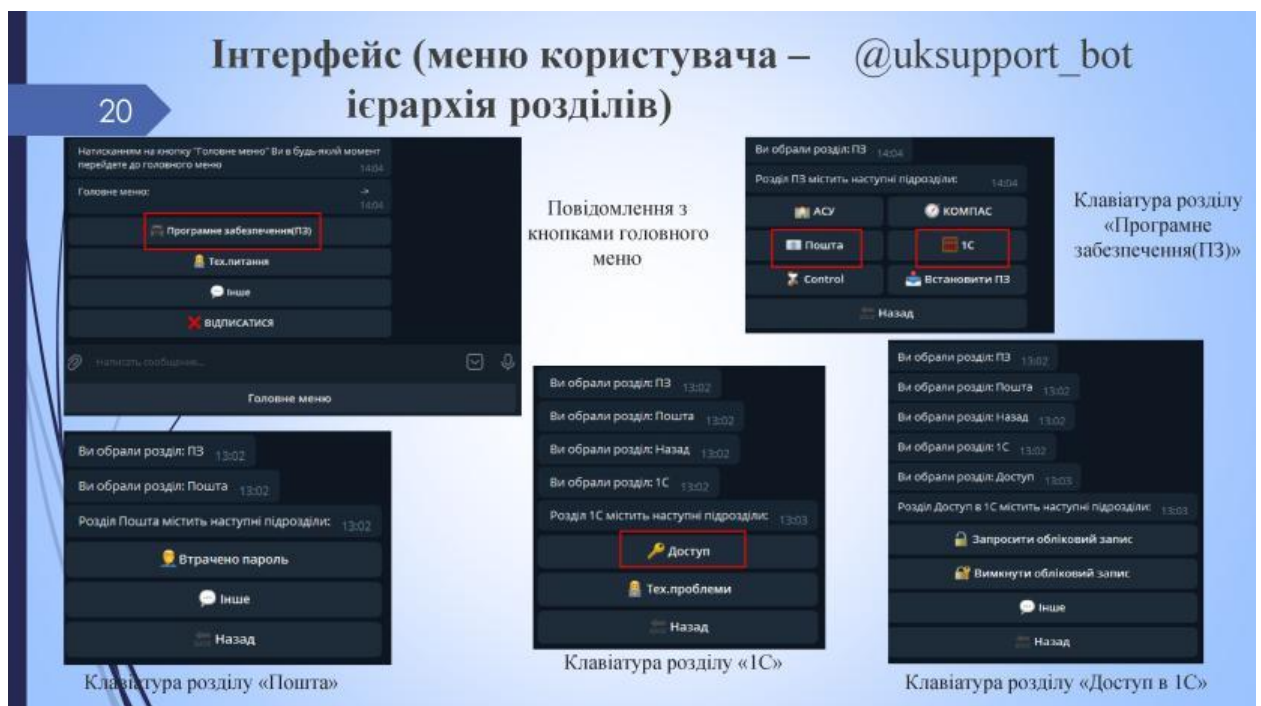
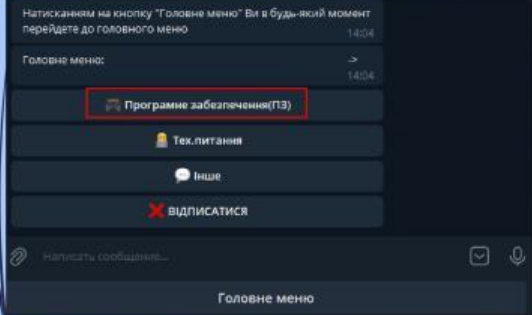


Рисунок Б.20 – Слайд 20 презентації

@uksupport_bot

Інтерфейс (меню користувача – ієрархія розділів)

21



Натисканням на кнопку "Головне меню" Ви в будь-який момент перейдете до головного меню. 14:04

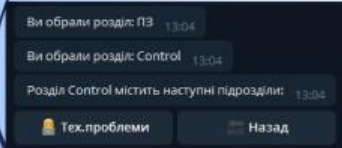
Головне меню: 14:04

- Програмне забезпечення(ПЗ)
- Тех.питання
- Інше
- ВІДПИСАТИСЯ

Написати повідомлення... 14:04

Головне меню

Повідомлення з кнопками головного меню



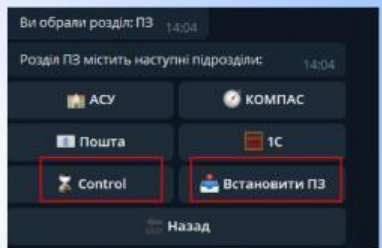
Ви обрали розділ: ПЗ 13:04

Ви обрали розділ: Control 13:04

Розділ Control містить наступні підрозділи: 13:04

- Тех.проблеми
- Назад

Клавіатура розділу «Control»

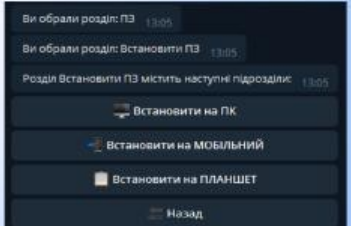


Ви обрали розділ: ПЗ 14:04

Розділ ПЗ містить наступні підрозділи: 14:04

- АСУ
- КОМПАС
- Пошта
- 1С
- Control
- Встановити ПЗ
- Назад

Клавіатура розділу «Програмне забезпечення(ПЗ)»



Ви обрали розділ: ПЗ 13:05

Ви обрали розділ: Встановити ПЗ 13:05

Розділ Встановити ПЗ містить наступні підрозділи: 13:05

- Встановити на ПК
- Встановити на МОБІЛЬНИЙ
- Встановити на ПЛАНШЕТ
- Назад

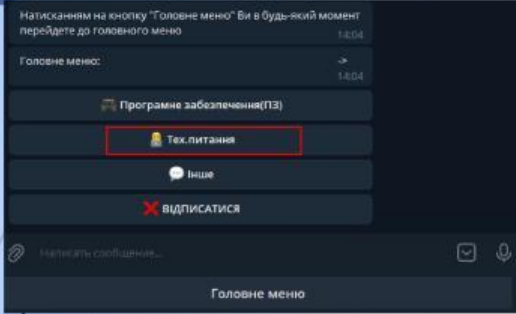
Клавіатура розділу «Встановити ПЗ»

Рисунок Б.21 – Слайд 21 презентації

@uksupport_bot

Інтерфейс (меню користувача – ієрархія розділів)

22



Натисканням на кнопку "Головне меню" Ви в будь-який момент перейдете до головного меню. 14:04

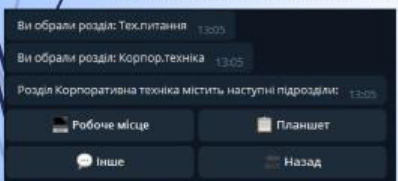
Головне меню: 14:04

- Програмне забезпечення(ПЗ)
- Тех.питання
- Інше
- ВІДПИСАТИСЯ

Написати повідомлення... 14:04

Головне меню

Повідомлення з кнопками головного меню



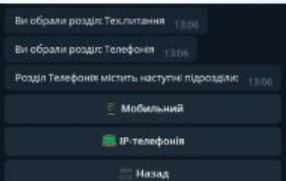
Ви обрали розділ: Тех.питання 13:05

Ви обрали розділ: Корпор.техніка 13:05

Розділ Корпоративна техніка містить наступні підрозділи: 13:05

- Робоче місце
- Планшет
- Інше
- Назад

Клавіатура розділу «Корпоративна техніка»



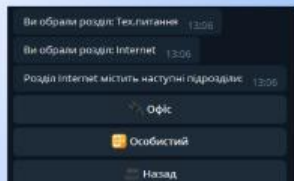
Ви обрали розділ: Тех.питання 13:06

Ви обрали розділ: Телефонія 13:06

Розділ Телефонія містить наступні підрозділи: 13:06

- Мобільний
- ІР-телефонія
- Назад

Клавіатура розділу «Телефонія»



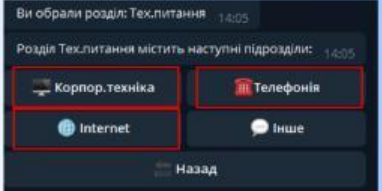
Ви обрали розділ: Тех.питання 13:06

Ви обрали розділ: Internet 13:06

Розділ Internet містить наступні підрозділи: 13:06

- Офіс
- Особистий
- Назад

Клавіатура розділу «Internet»



Ви обрали розділ: Тех.питання 14:05

Розділ Тех.питання містить наступні підрозділи: 14:05

- Корпор.техніка
- Телефонія
- Internet
- Інше
- Назад

Клавіатура розділу «Тех.питання»

Рисунок Б.22 – Слайд 22 презентації

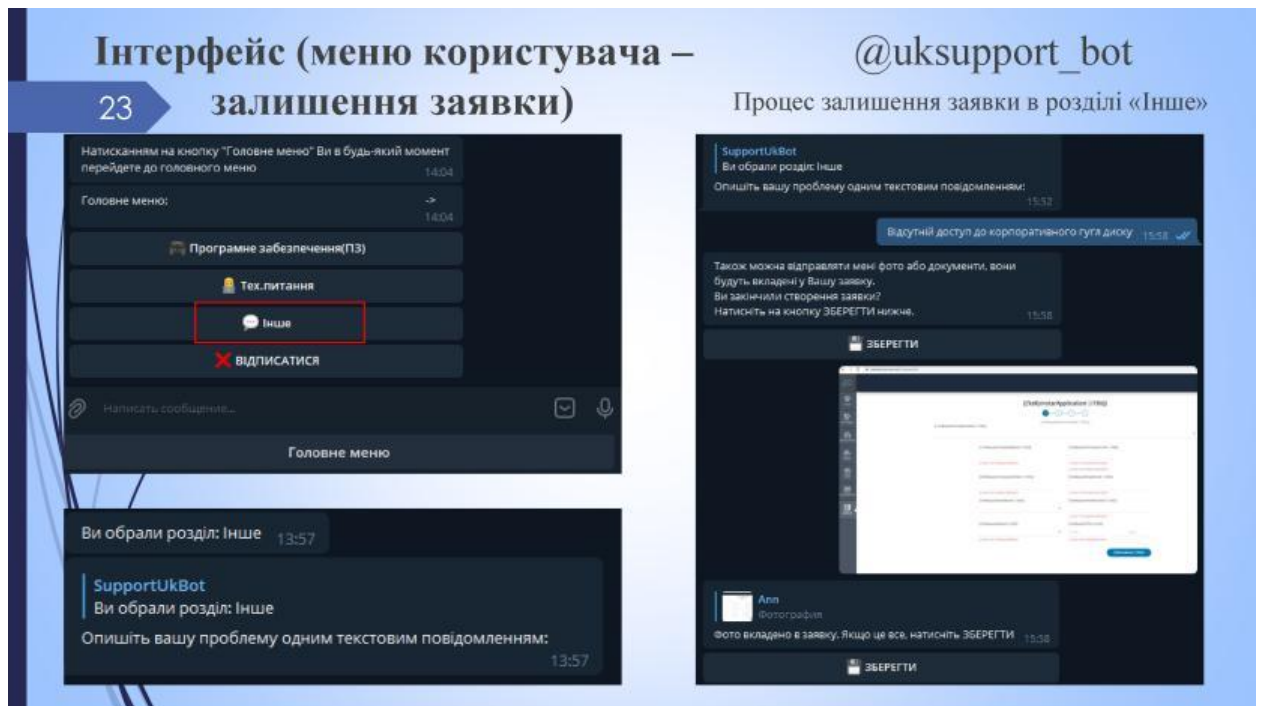


Рисунок Б.23 – Слайд 23 презентації

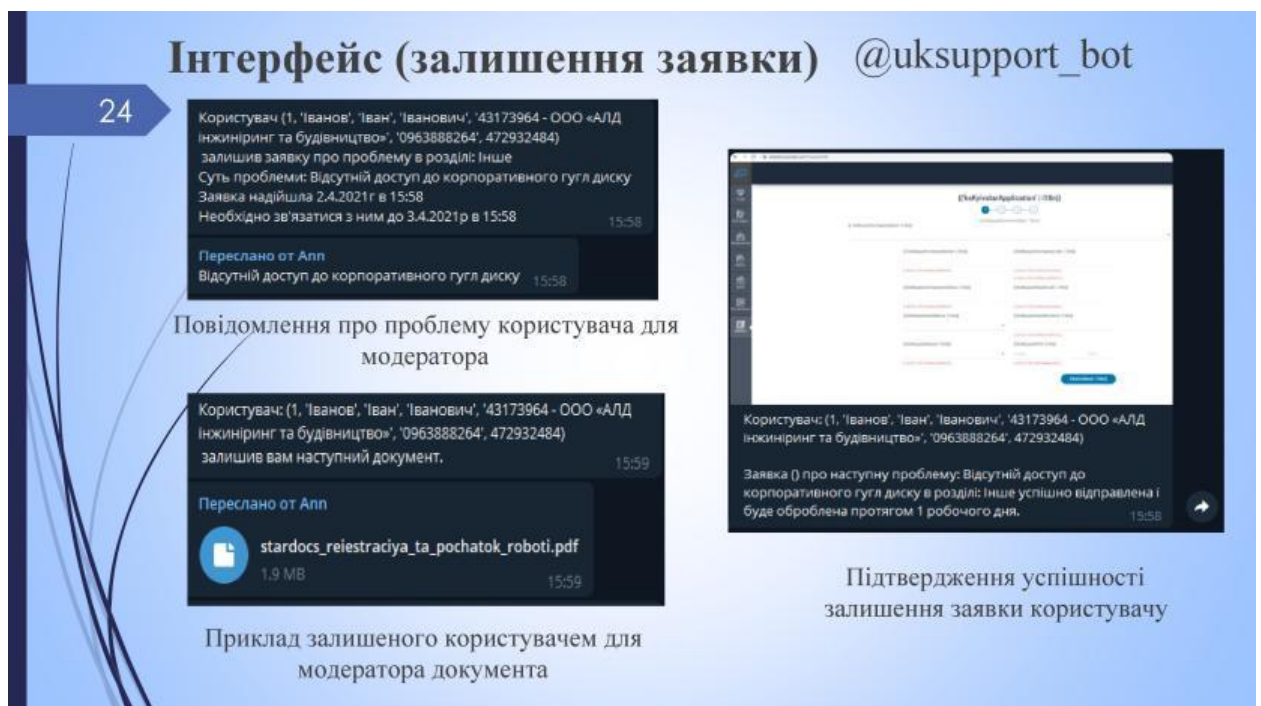


Рисунок Б.24 – Слайд 24 презентації

Чек-лист тестування					
25	Користувач незареєстрований	Користувач зареєстрований	Модератор	Адміністратор	
/start	+	+	+	+	
/help	+	+	+	+	
/registration	+	+	+	+	
/exit	+	+	+	+	
/menu	+	+	+	+	
/help_mod	+	+	+	+	
/auth	+	+	+	+	
/reauth	+	+	+	+	
/add	+	+	+	+	
/show	+	+	+	+	
/fail	+	+	+	+	
/remove	+	+	+	+	
/answer	+	+	+	+	
/answer_by_id	+	+	+	+	
/send_all	+	+	+	+	
/send_by_id	+	+	+	+	
/get_message	+	+	+	+	
Залишення заявки	+	+	+	+	
Отримання повідомлень про помилки	+	+	+	+	
Відстеження переписок користувача та модератора	+	+	+	+	

Рисунок Б.25 – Слайд 25 презентації

26

ВИСНОВКИ

- Створений проєкт відповідає меті розробки, всім вимогам, що були пред'явлені до програмного засобу.
- Програмний продукт пройшов всі тести: для тестування було обрано команди для всіх типів користувачів, функції залишення заявки, відстеження переписок користувача та модератора, відстеження системних помилок.
- Передбачено зручний інтерфейс користувача з набором клавіатур для обирання відповідного розділу, передбачено введення помилкових даних та невірних дій користувача.
- За темою роботи є 1 наукова публікація, а саме тези доповіді на науково-практичній конференції «Тиждень науки», яка дистанційно проводилася у Національному університеті «Запорізька політехніка» у квітні 2021 року.

Шляхи майбутнього удосконалення:

- Керування ботом з іншого програмного продукту технічної підтримки.
- Автоматичне або автоматизоване розподілення заявок на відповідальних фахівців в залежності від категорії проблеми.
- Впровадження штучного інтелекту в процес спілкування з користувачем.
- Розширення кола вирішуваних проблем.

Рисунок Б.26 – Слайд 26 презентації