МІНІСТЕРСТВО ФІНАНСІВ УКРАЇНИ
ДЕРЖАВНИЙ ПОДАТКОВИЙ УНІВЕРСИТЕТ

ФАКУЛЬТЕТ ФІНАНСІВ ТА ЦИФРОВИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І СИСТЕМ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: ІНТЕГРОВАНА СИСТЕМА СЛУЖБИ ТЕХНІЧНОЇ ПІДТРИМКИ НА БАЗІ МЕСЕНДЖЕР – ПЛАТФОРМИ ТА JIRA SERVICE DESK

Здобувача вищої освіти другого (магістерського) рівня,
ІІ курсу, групи КМІ-24-2, спеціальності 122 «Комп'ютерні науки»,
освітня програма «Інформаційні управляючі системи і технології в економіці»

_____ Бур'ян Сергій Сергійович
(підпис) (прізвище, ім'я, по батькові)

Науковий керівник _____ Погорєловська Ірина Дем'янівна канд. екон. наук, доцент
(підпис)
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Керівник освітньо-професійної програми _____ Омельчук Антон Анатолійович, канд. техн. наук
(підпис)
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Допущено до захисту за рішенням кафедри протокол від «02» січня 2026 № 10

Завідувач кафедри _____ Філоненко Михайло Миколайович, канд. фіз.-мат. наук, доцент
(підпис)
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Ірпінь – 2026

Реєстрація № 8 16.01.2026 _____
(дата) (підпис відповідальної особи)

Результати перевірки до захисту 02.01.2026 _____
(до захисту, на доопрацювання) (дата) (підпис)

Результат захисту 5 (відмінно) 04.02.2026 _____
(оцінка) (дата) (підпис)

Голова комісії: _____ О.С. Дюбанов
(підпис)

Члени комісії: _____ А.Ю. Горбовий
(підпис)

_____ М.М. Філоненко
(підпис)

_____ І.Д. Погорєловська
(підпис)

БЛАНК ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Об'єкти контролю	Діапазон оцінки в балах	Фактична кількість балів	Примітки	Дата, П.І.Б.: керівника, голови комісії членів комісії,секретаря, (підписи)
ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ КОМІСІЄЮ ПО ЗАХИСТУ				
1. Зміст кваліфікаційної роботи (якість теоретичного матеріалу, його зв'язок з практичним, вміння представляти матеріал в логічній послідовності, наявність висновків та пропозицій)	від 0 до 30	29		04.02.2026 Дюбанов О.С. Горбовий А.Ю.
2. Дотримання вимог щодо оформлення	від 0 до 10	9		Філоненко М.М.
3. Вчасне виконання роботи на всіх етапах у встановлені кафедрою терміни	від 0 до 5	5		Погорєловська І.Д. Суховецька Л.П.
4. Рецензент	від 0 до 5	5		
РАЗОМ БАЛІВ	від 0 до 50	47		

Оцінювання кваліфікаційної роботи науковим керівником - 48 балів.

ПЕРЕВЕДЕННЯ ОЦІНКИ ЗА 100-БАЛЬНОЮ ШКАЛОЮ ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалоюЄКТС	Оцінка за національною шкалою	Дата	П.І.Б.: голови, членів комісії, секретаря, (підпис)
95	A	5	04.02.2026	Дюбанов О.С. Горбовий А.Ю. Філоненко М.М. Погорєловська І.Д. Суховецька Л.П.

ПРАВИЛА ПЕРЕВЕДЕННЯ ОЦІНОК ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою
90-100	A	5 (відмінно)
80-89	B	4 (добре)
70-79	C	
60-69	D	3 (задовільно)
50-59	E	
35-49	FX	2 (незадовільно)

Факультет фінансів та цифрових технологій
Кафедра комп'ютерних та інформаційних технологій і систем
Галузь знань 12 «Інформаційні технології»
Спеціальність 122 «Комп'ютерні науки»
Освітня програма «Інформаційні управляючі системи і технології в економіці»

Затверджую
Завідувач кафедри _____ Філоненко М.М.
«11» грудня 2024

Завдання на кваліфікаційну роботу
здобувачу вищої освіти
Бур'ян Сергію Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи Інтегрована система служби технічної підтримки на базі месенджер - платформи та Jira Service Desk
Затверджена наказом ректора від «10» грудня 2024 № 1852
2. Строк здачі закінченої роботи «02» січня 2026
3. Мета роботи Проєктування та реалізація інтегрованої системи служби технічної підтримки

Об'єкт дослідження: процеси організації технічної підтримки користувачів з використанням Help Desk систем

Предмет дослідження методи та інструменти інтеграції систем управління IT-сервісами з месенджерами через REST API, webhook-механізми та прямі запити до баз даних.

Орієнтовна база дослідження: **служба технічної підтримки компанії.**

4. Вихідні дані до роботи комплекс сучасних **стандартів, методологій, технологій та практик організації ефективної роботи служб технічної підтримки.**
5. Перелік графічного матеріалу: схеми **архітектури та структури розробленої інтегрованої системи, ER-діаграми бази даних, алгоритму авторизації користувачів, алгоритму розподілу заявок між спеціалістами на виконання,** таблиці, лістинг програмного коду.
6. Зміст кваліфікаційної роботи (перелік питань за кожним розділом) Розділ 1. **Аналітичний огляд методів та засобів технічної підтримки** 1.1.Дослідження сучасних методів організації технічної підтримки. 1.2. Технічний аналіз платформ інтеграції. **Розділ 2. Розробка інтегрованої системи.** 2.1.Проектування архітектури системи 2.2. Реалізація системи. **Розділ 3. Впровадження та оцінка ефективності інтегрованої системи** 3.1. Тестування та впровадження системи 3.2. Аналіз результатів роботи системи. 3.3. Перспективи розвитку системи.

7. Календарний план виконання роботи

№ п/п	Назва етапів кваліфікаційноїроботи	Строк виконання етапів роботи	
		За планом	Фактично
1	2	3	4
1	Вибір теми, підбір літературних джерел, формування структури кваліфікаційної роботи	До 15.12.2024	До 15.12.2024
2	**І розділ** (розкрити сутність і стан дослідження наукової проблематики роботи, після чого навести її загальну характеристику; огляд літератури за темою і вибір конкретних напрямів досліджень)	30.05.2025	30.05.2025
3	**ІІ розділ** (аналіз експериментальної частини про методики досліджень)	10.09.2025	10.09.2025
4	**ІІІ розділ** (проведені теоретичні і (або) експериментальні дослідження та проектні рішення)	20.11.2025	20.11.2025
5	**Висновки**	11.12.2025	11.12.2025
6	Доопрацювання роботи з урахуванням зауважень наукового керівника	25.12.2025	25.12.2025
7	Отримання відгуку керівника	28.01.2026	28.01.2026
8	Отримання зовнішньої рецензії	30.01.2026	30.01.2026

8. Дата видачі завдання «11» грудня 2024

9. Керівник кваліфікаційної роботи _____ Погорєловська І.Д.
(підпис) (прізвище, ініціали)

10. Керівник освітньо-професійної програми _____ Омельчук А.А
(підпис) (прізвище, ініціали)

11. Завдання до виконання отримав здобувач вищої освіти _____ Бур'ян С.С.
(підпис) (прізвище, ініціали)

АНОТАЦІЯ

«Розробка інтегрованої системи технічної підтримки на базі Telegram Bot та Jira Service Desk»

Мета роботи: проєктування та реалізація інтегрованої системи технічної підтримки, що поєднує зручність месенджера Telegram з функціональністю Jira Service Desk, забезпечуючи оперативну взаємодію користувачів зі спеціалістами технічної підтримки.

Завдання роботи: дослідити сучасні методи організації технічної підтримки, проаналізувати можливості існуючих Help Desk систем, зпроєктувати архітектуру з багаторівневою безпекою, розробити функціональні модулі, реалізувати онлайн-чат, впровадити систему та оцінити її ефективність.

Об'єкт дослідження: процеси організації технічної підтримки користувачів з використанням Help Desk систем

Предмет дослідження: методи та інструменти інтеграції систем управління IT-сервісами з месенджерами через REST API, webhook-механізми та прямі запити до баз даних.

Інформаційна база дослідження: комплекс сучасних стандартів, методологій, технологій та практик організації ефективної роботи служб технічної підтримки

За результатами дослідження було сформульовано висновки про те, що:

- традиційні системи служби технічної підтримки не в повній мірі відповідають очікуванням користувачів щодо швидкості обслуговування їх запитів через складність інтерфейсів, повільність обробки звернень та фрагментацію каналів комунікації;
- необхідність формалізації процесів технічної підтримки, контролю SLA та накопиченню інформації для аналітики, потребують сучасної інтегрованої системи служби технічної підтримки;
- розроблена інтегрована система, яка поєднує зручність месенджера Telegram з функціональністю Jira Service Desk, забезпечує оперативну взаємодію користувачів зі спеціалістами технічної підтримки та формалізований підхід до організації роботи служби - всі події реєструються, а накопичена інформація стає доступною для подальшого аналізу.

Одержані результати можуть бути використані для підвищила ефективності комунікації, оптимізації процесів контролю якості обслуговування користувачів IT інфраструктури компанії.

Кваліфікаційна робота містить 66 сторінок, 6 таблиць, 6 рисунків, список використаних джерел 60 найменувань, 24 додатки.

Ключові слова: інтегрована система технічної підтримки, служба технічної підтримки, месенджер,Telegram Bot, Jira Service Desk, REST API, онлайн-чат, webhook, тікет-система, автоматизація технічної підтримки.

Рік виконання кваліфікаційної роботи: 2025-2026

Рік захисту роботи: 2026

ANNOTATION

"Development of an Integrated Technical Support System Based on Telegram Bot and Jira Service Desk"

Purpose of the research: to design and implement an integrated technical support system that combines the usability of the Telegram messenger with the functionality of Jira Service Desk, ensuring efficient interaction between users and technical support specialists.

Objectives of the research: to study modern approaches to the organization of technical support; to analyze the capabilities of existing Help Desk systems; to design a multi-level security architecture; to develop functional modules; to implement an online chat; to deploy the system; and to evaluate its effectiveness.

Object of the research: processes of organizing user technical support using Help Desk systems.

Subject of the research: methods and tools for integrating IT service management systems with messengers using REST APIs, webhook mechanisms, and direct database queries.

Information base of the research: a set of modern standards, methodologies, technologies, and best practices for organizing efficient technical support services.

Research findings indicate that:

- traditional technical support systems do not fully meet user expectations regarding request processing speed due to complex interfaces, slow handling of requests, and fragmented communication channels;
- the need for formalization of technical support processes, SLA control, and accumulation of data for analytical purposes requires the implementation of a modern integrated technical support system;
- the developed integrated system, which combines the convenience of the Telegram messenger with the functionality of Jira Service Desk, ensures prompt interaction between users and technical support specialists and provides a formalized approach to service organization—all events are recorded, and the accumulated data becomes available for further analysis.

The obtained results can be used to improve communication efficiency and optimize quality control processes of IT infrastructure user support within an organization.

The qualification thesis consists of 66 pages, 6 tables, 6 figures, a list of references containing 60 sources, and 24 appendices.

Keywords: integrated technical support system, technical support service, messenger, Telegram Bot, Jira Service Desk, REST API, online chat, webhook, ticketing system, technical support automation.

Year of completion of the qualification thesis: 2025–2026
Year of thesis defense: 2026

ЗМІСТ

ВСТУП 3
РОЗДІЛ 1. АНАЛІТИЧНИЙ ОГЛЯД МЕТОДІВ ТА ЗАСОБІВ ТЕХНІЧНОЇ ПІДТРИМКИ 7
1.1. Дослідження сучасних методів, підходів та практик організації технічної підтримки 7
1.2. Технічний аналіз платформ інтеграції 17
РОЗДІЛ 2. РОЗРОБКА ІНТЕГРОВАНОЇ СИСТЕМИ 24
2.1. Проєктування архітектури 24
2.2. Реалізація системи 35
РОЗДІЛ 3. ВПРОВАДЖЕННЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ ІНТЕГРОВАНОЇ СИСТЕМИ 48
3.1. Тестування та впровадження системи 48
3.2. Аналіз результатів роботи системи 50
3.3. Перспективи розвитку системи 55
ВИСНОВКИ 57
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 60
ДОДАДКИ 65

ВСТУП

Актуальність теми. Сучасний світ інформаційних технологій характеризується стрімким розвитком цифровізації всіх сфер діяльності. Особливої актуальності набуває питання організації ефективної технічної підтримки користувачів, оскільки якісна технічна підтримка стає ключовим фактором конкурентоспроможності організацій.

Традиційні канали комунікації — електронна пошта, телефон, корпоративні портали — мають суттєві недоліки: повільність обробки запитів, складність доступу, відсутність єдиної історії взаємодії. Сучасні користувачі очікують миттєвого реагування, звикнувши до швидкого обміну повідомленнями в месенджерах.

Професійні Help Desk системи забезпечують формалізацію процесів, контроль виконання та аналітику, проте мають критичний недолік — незручність для кінцевих користувачів. Інтеграція тікет-систем з Telegram дозволяє поєднати зручність миттєвого спілкування з потужністю професійних систем управління заявками. Тому розробка інтегрованої системи на базі Telegram Bot та Jira Service Desk є актуальним завданням.

Стан і ступінь розробки. Про існування комплексу невирішених проблем, пов'язаних з технічною підтримкою користувачів корпоративних інформаційних систем свідчать результати досліджень, що представлені в роботах як вітчизняних так і зарубіжних вчених. Зокрема, у роботі [1, с.63] висвітлені проблеми, які пов'язані з функціонуванням служби підтримки у закладах освіти, що зумовлені як складною інфраструктурою закладів так і непрогнозованою завантаженістю їхньої служби підтримки. Автори наголошують на необхідності запровадження систем HelpDesk, вказують на необхідність "зручного та зрозумілого для користувачів механізму, що дозволить направляти запити на сайт", але не уточнюють, який саме спосіб взаємодії з цими системами є найкращим. У дослідженні [2], також наголошено, що стан IT-інфраструктури у багатьох університетах є незадовільним, оскільки з її допомогою неможливо надавати якісні IT-

послуги, що відповідають сучасним вимогам, і з гарантованою якістю, відповідно до стандартів та рекомендацій у галузі управління IT (ITIL, MOF, COBIT інш.). Про необхідність безперервного, постійного удосконалення процесу технічної підтримки користувачів інформаційних систем банку зазначено у роботі [3]. Важливість технологічної підтримки інформаційної інфраструктури Міністерства оборони України обґрунтована у роботі [4].

Більшість традиційних Help Desk систем, таких як Zendesk, Freshdesk, ServiceNow та Microsoft System Center Service Manager, не мають нативної інтеграції з популярними месенджерами, про що наголошено у роботах [5, 6, 7, 8]. Це змушує організації використовувати стороннє програмне забезпечення або розробляти власні рішення. Існуючі підходи до організації технічної підтримки часто не відповідають сучасним очікуванням користувачів щодо швидкості реагування — дослідження MetricNet показує, що середній час першої відповіді в багатьох організаціях перевищує 4 години [9], тоді як користувачі очікують миттєвого реагування [10]. Відсутність онлайн-взаємодії між користувачами та спеціалістами в системах управління заявками призводить до затягування вирішення критичних проблем через асинхронність комунікації, що негативно впливає на продуктивність бізнес-процесів, про що наголошено у роботах [11, 12].

Мета кваліфікаційної роботи: проєктування та реалізація інтегрованої системи служби технічної підтримки, що поєднує зручність Telegram з потужністю Jira Service Desk та забезпечує ефективну взаємодію між користувачами та спеціалістами.

Завдання кваліфікаційної роботи: дослідити сучасні методи та практики організації технічної підтримки, проаналізувати можливості існуючих Help Desk систем, розробити архітектуру ІІС з багаторівневою безпекою та функціональні модулі, реалізувати онлайн-чат, проаналізувати результати роботи системи та оцінити її ефективність.

Об'єкт дослідження: процеси організації технічної підтримки користувачів з використанням Help Desk систем

Предмет дослідження: методи та інструменти інтеграції систем управління IT-сервісами з месенджерами через REST API, webhook-механізми та прямі запити до баз даних.

Методи дослідження: при виконанні роботи було використано *системний аналіз*, який застосовувався при роботі з інформаційними ресурсами за темою дослідження; *методи оцінювання* застосовувалися при аналізі результатів роботи розроблено ІС, при технічному аналізі платформ інтеграції; *порівняльний аналіз* виконувався при дослідженні платформ технічної підтримки (Zendesk, Freshdesk, Jira Service Desk, ServiceNow,Microsoft System Center Service Manager); за допомогою *методу узагальнення* формувалися висновки.

Інформаційна база дослідження: комплекс сучасних стандартів, методологій, технологій та практик організації ефективної роботи служб технічної підтримки

Науково-практична значимість одержаних результатів:

- розроблено гібридну архітектуру, що поєднує асинхронні тікет-системи із синхронною комунікацією через месенджери;
- реалізовано механізм обходу обмежень Jira API через прямі SQL-запити;
- розроблено триступеневу архітектуру з DMZ, що забезпечує баланс між доступністю та безпекою.

Розроблена інтегрована система скорочує час реагування на 67%, збільшує обсяг оброблених заявок на 17.1%, підвищує задоволеність користувачів до 91%. Триступенева архітектура забезпечує високий рівень безпеки. Економічний ефект включає економію 38 годин на місяць, зменшення з 16% до 5%, зниження порушень SLA з 24% до 5%.

Особистий внесок автора. Автором розроблено концепцію системи з триступеневою архітектурою, спроєктовано ролеву модель та алгоритми роботи, створено структуру бази даних, реалізовано модулі на Python з використанням pyTelegramBotAPI, Flask, Jira REST API. Розроблено механізм

двосторонньої синхронізації з автоматичною зміною авторів коментарів у Jira. Здійснено впровадження та аналіз ефективності.

Апробація результатів роботи. Результати досліджень були оприлюдненні у виступах та збірниках матеріалів:

- Міжнародної науково-практичної конференції здобувачів вищої освіти та молодих вчених "Економіка і бізнес: нові реалії, технологічні тренди, цифрові інновації" (Львів-Ірпінь, 15.05.2025) [13];
- Міжнародної науково-практичної студентської конференції (15.10.2025, м. Харків, Україна) [14];
- Міжнародної науково-практичної конференції "Інформаційні технології і автоматизація – 2025" (м. Одеса, 30.10. 2025) [15].

За результатами дослідження підготовлені до друку дві статті у науковому журналі "Вісник Східноукраїнського національного університету імені Володимира Даля": "Напрями оптимізації процесів управління IT-послугами" [23] та "Atlassian Intelligence як інструмент підвищення ефективності IT-команд" [61].

РОЗДІЛ 1. АНАЛІТИЧНИЙ ОГЛЯД МЕТОДІВ ТА ЗАСОБІВ ТЕХНІЧНОЇ ПІДТРИМКИ

1.1. Дослідження сучасних методів, підходів та практик організації технічної підтримки

Проведений аналіз свідчить, що існує достатньо велика кількість стандартів, методологій та практик організації ефективної роботи служб технічної підтримки, зокрема:

- ITIL (Information Technology Infrastructure Library) - комплексний набір практик і вказівок, які спрямовані на підвищення якості та ефективності IT-послуг в організаціях, узгодження IT-послуг з бізнес-стратегією організації [16]. Це гнучкий фреймворк, що охоплює процеси планування, надання, підтримки та постійного вдосконалення IT-послуг. Саме ITIL визначає технічну підтримку не як допоміжний процес, а як важливий елемент бізнесу.
- ISO/IEC 20000 - стандарт, що визначає вимоги до планування, створення, впровадження, експлуатації, моніторингу, перегляду та вдосконалення систем керування послугами [17]. Варто зазначити, що стандартизоване управління IT-сервісами є підґрунттям забезпечення якісних IT-послуг.
- COBIT (Control Objectives for Information and Related Technology) - рамкова система для гнучкого управління корпоративними IT, узгодження IT-стратегій з бізнес-цілями, інтеграції з іншими стандартами, такими як ISO та ITIL [18].

Також популярними в системах управління IT-послугами є фреймворки:

- CMMI (Capability Maturity Model Integration) - інтегрована модель, що визначає структурований підхід до управління та оцінювання процесів [19]. Використання цієї моделі дозволяє досягати вищих рівнів зрілості, підвищуючи ефективність результатів та знижуючи ризики, що, у свою чергу, сприяє розробці високоякісних продуктів та послуг.
- Six Sigma – методологія, що забезпечує структурований та систематичний спосіб виявлення, аналізу та усунення дефектів процесів, покращення якості, ефективності та задоволеності клієнтів (3,4 дефекти на мільйон можливостей) своїх послуг і продуктів, знизити витрати та підвищити задоволеність клієнтів

[20]. Використовуючи методології та інструменти Six Sigma IT-компанії можуть покращити якість ITSM.

- eTOM (Enhanced Telecom Operations Map) – розширена карта телекомунікаційних операцій, що визначає загальну структуру телекомунікаційних бізнес-процесів, яка охоплює всі функції, необхідні для ефективного ведення телекомунікаційного бізнесу. Вона гарантує, що різні відділи та команди в телекомунікаційній організації узгоджені щодо своїх цілей, процесів та завдань, сприяючи спільному підходу до надання послуг [21]. Саме ця модель бізнес-процесів розглядається як важлива складова IT-інфраструктури Індустрії 4.0 у роботі [22, с. 88].

Зміст розглянутих документів може бути основою для організації та забезпечення ефективного функціонування IT-відділів, підвищення їх здатності надавати якісні послуги, що сприятиме оперативному вирішенню проблем корпоративних користувачів, стабільному розвитку IT-інфраструктури та, у цілому, досягненню визначених бізнес-цілей компаній [23].

Огляд популярних Help Desk систем

У процесі виконання кваліфікаційної роботи було також проаналізовано характеристики та особливості роботи найпопулярніших платформ технічної підтримки - Zendesk, Freshdesk, Jira Service Desk, ServiceNow,Microsoft System Center Service Manager.

Zendesk є одним із лідерів ринку Help Desk систем, які вражають своєю комплексністю. Zendesk пропонує не просто систему тікетів, а цілу екосистему для управління клієнтським сервісом з потужними можливостями автоматизації, аналітики та інтеграції. Особливо імпонує те, що платформа підтримує понад 40 мов інтерфейсу та має розвинену екосистему додатків у власному маркетплейсі [5].

Однак, значна вартість ліцензії Zendesk є однією із головних проблем enterprise-рішень, особливо для малих організацій. Для стартапу або невеликої

IT-команди ця платформа може виявитися надмірно дорогою, особливо на початкових етапах розвитку бізнесу.

Freshdesk (тепер Freshworks Customer Service Suite) — платформа, що привернула увагу, саме, як більш доступна альтернатива Zendesk. При ознайомлені з цією системою, було оцінено її підхід до балансу між функціональністю та доступністю. Система пропонує сучасний, інтуїтивно зрозумілий інтерфейс, потужні можливості автоматизації робочих процесів та інтеграцію з популярними бізнес-додатками [6]. Особливо цінною для молодих компаній є наявність безкоштовного тарифу для невеликих команд, що дозволяє почати використовувати професійну систему без початкових інвестицій.

Jira Service Desk (тепер Jira Service Management) від Atlassian — платформа, що спеціально розроблена для IT-команд, яка базується на принципах ITIL. Було досліджено інтеграцію Jira Service Desk з усією екосистемою Atlassian - Jira Software для розробки, Confluence для документації, Bitbucket для керування кодом [7]. Ця інтеграція створює єдиний робочий простір для всієї IT-команди, що особливо цінно для організацій, які вже використовують продукти Atlassian.

Працюючи з Jira Service Management за час практики, була можливість оцінити її потужні можливості налаштування робочих процесів. Можливість створювати складні workflow з умовними переходами дозволяє адаптувати систему під специфічні потреби організації. Однак складність цієї системи вимагає часу для вивчення та налаштування.

ServiceNow представляє те, що можна було б назвати "важкою артилерією" серед Help Desk систем. Це enterprise-рівень рішень для управління IT-сервісами, які пропонують комплексний підхід до цифрової трансформації організацій з використанням штучного інтелекту та машинного навчання [8]. Вивчення можливостей ServiceNow, показало потужність цієї платформи, та дало розуміння, що це рішення призначене для великих корпорацій з відповідним бюджетом та IT-інфраструктурою.

Серед особливостей ServiceNow варто відзначити використання AI для автоматизації процесів. Наприклад, система може автоматично категоризувати інциденти, пропонувати рішення на основі історичних даних та навіть передбачати потенційні проблеми. Це майбутнє технічної підтримки, але поки що доступне лише великим гравцям.

Microsoft System Center Service Manager цікава своєю глибокою інтеграцією в екосистему Microsoft [24]. Для організацій, які повністю побудовані на Microsoft-технологіях, це може бути оптимальним вибором. Однак, зрозуміло, що така "прив'язка" до одного вендора може створювати обмеження в майбутньому.

Проведений аналіз системи, дозволив дійти висновку, що вибір Help Desk платформи - це завжди компроміс між функціональністю, вартістю та складністю впровадження. Для малих команд оптимальними можуть бути Freshdesk або базові версії Zendesk, тоді як великі організації можуть потребувати можливостей ServiceNow або корпоративних версій інших платформ.

У ході дослідження сучасних систем технічної підтримки було виявлено декілька системних проблем, які вимагають негайного вирішення. Ці проблеми стали особливо очевидними під час аналізу реальних кейсів та спілкування з практиками індустрії.

Відсутність інтеграції з месенджерами є однією з найактуальніших проблем, з якою доводилось стикатись на практиці. Більшість традиційних Help Desk систем були розроблені в епоху, коли месенджери ще не мали такого поширення. Сьогодні ж ситуація кардинально змінилася — користувачі проводять більшу частину часу в Telegram, WhatsApp та інших месенджерах, ніж в email чи на корпоративних порталах.

Однак більшість Help Desk систем не мають нативної інтеграції з цими популярними каналами комунікації, що змушує організації використовувати стороннє програмне забезпечення або розробляти власні рішення [11].

Низька швидкість обробки запитів часто пов'язана з необхідністю ручного маршрутизування звернень та відсутністю автоматизації. Дослідження показало, що середній час першої відповіді в багатьох організаціях перевищує 4 години [9], що абсолютно не відповідає сучасним очікуванням користувачів. В епоху, коли користувачі звикли до миттєвого реагування в месенджерах та соціальних мережах, очікування відповіді протягом 4 годин є неприйнятним.

Фрагментація каналів комунікації - це проблема, яка особливо дратує користувачів. Коли вони починають спілкування в чаті, потім переходять на email, а потім телефонують, кожного разу пояснюючи проблему. Це призводить до того, що історія взаємодії з клієнтом розподілена між різними системами, що ускладнює отримання повного контексту проблеми та може призводити до дублювання зусиль [12].

Складність інтеграції з існуючими корпоративними системами часто стає перешкодою для впровадження нових рішень. Вивчення цієї проблеми показало, що багато організацій змушені підтримувати застарілі системи через високі витрати на міграцію [10]. Особливо це стосується великих корпорацій, де IT-інфраструктура формувалася десятиліттями. Іноді нові, більш ефективні системи не впроваджуються не через їх недоліки, а через складність інтеграції з legacy-системами. Це створює своєрідний замкнене коло - старі системи гальмують впровадження нових технологій, а без нових технологій важко вирішити проблеми старих систем.

Відсутність проактивного підходу у більшості систем технічної підтримки, також привертає увагу. Зокрема, системи фокусуються на реактивному вирішенні проблем, а не на їх попередженні, що призводить до накопичення технічного боргу [19]. Викликає подив, чому системи не аналізують патерни проблем та не попереджають користувачів про потенційні issues. Наприклад, якщо відомо, що певна помилка виникає у багатьох користувачів, чому б не відправити превентивне повідомлення всім користувачам з інструкцією, як уникнути цієї проблеми? Або якщо відомо, що

певне оновлення може викликати проблеми, чому б не підготувати користувачів заздалегідь?

Недостатня аналітика та звітність обмежує можливості для оптимізації процесів, і це особливо проблематично для менеджменту. Багато систем не надають детальної інформації про ефективність різних каналів підтримки або продуктивність окремих агентів [20]. Без даних складно приймати обґрунтовані рішення щодо покращення процесів.

Проведене дослідження показало, що служби підтримки, які здійснюють детальний аналіз процесів підтримки, показують значно кращі результати. Вони можуть швидко виявляти вузькі місця, оптимізувати розподіл навантаження та покращувати якість обслуговування на основі реальних даних, а не інтуїції.

Всі ці проблеми створюють передумови для розвитку нових підходів до організації технічної підтримки. Саме тому, інтеграція традиційних Help Desk систем з сучасними каналами комунікації через API та розробка гібридних рішень є актуальною.

Низька швидкість обробки запитів часто пов'язана з необхідністю ручного маршрутизування звернень та відсутністю автоматизації. Дослідження показує, що середній час першої відповіді в багатьох організаціях перевищує 4 години, що не відповідає сучасним очікуванням користувачів [10].

Фрагментація каналів комунікації призводить до того, що історія взаємодії з клієнтом розподілена між різними системами. Це ускладнює отримання повного контексту проблеми та може призводити до дублювання зусиль [12].

Складність інтеграції з існуючими корпоративними системами часто стає перешкодою для впровадження нових рішень. Багато організацій змушені підтримувати застарілі системи через високі витрати на міграцію [25].

Відсутність проактивного підходу у більшості систем технічної підтримки. Системи фокусуються на реактивному вирішенні проблем, а не на їх попередженні, що призводить до накопичення технічного боргу [26].

Недостатня аналітика та звітність обмежує можливості для оптимізації процесів. Багато систем не надають детальної інформації про ефективність різних каналів підтримки або продуктивність окремих агентів [27].

Ці проблеми створюють передумови для розвитку нових підходів до організації технічної підтримки, зокрема інтеграції традиційних Help Desk систем з сучасними каналами комунікації через API та розробки гібридних рішень.

Проведені дослідження також свідчать, що технічна підтримка в організаціях має специфічну особливість - вона обслуговує два принципово різні типи користувачів:

- внутрішні користувачі - співробітники компанії, які потребують допомоги з корпоративними системами, обладнанням, доступами тощо;
- зовнішні підрядники та партнери, які також потребують технічної підтримки для виконання своїх завдань, але не мають повного доступу до корпоративної інфраструктури.

Ця двоїстість створює додаткові вимоги до каналів комунікації - вони мають бути одночасно зручними, безпечними та забезпечувати можливість контролю та звітності.

Аналізуючи сучасні канали комунікації з технічною підтримкою, було визначено, що кожен з них має свої переваги та критичні недоліки в контексті саме технічної підтримки:

1) Електронна пошта залишається одним із найзатребуваніших каналів зв'язку, навіть в епоху миттєвих повідомлень. Згідно з дослідженнями Zendesk Customer Experience Trends Report 2023, близько 65% організацій все ще використовують email як основний канал комунікації з клієнтами [10]. Проте для служби технічної підтримки електронна пошта має фундаментальну проблему - вона катастрофічно повільна.

Крім того, електронна пошта створює проблему з пріоритизацією. Коли в поштовій скриньці служби підтримки накопичується 50-100 листів, складно визначити, який з них є найбільш критичним. Співробітник, у якого не працює телефонна система і він не може спілкуватися з клієнтами, та співробітник, який просить встановити додаткову програму, надсилають однакові листи, але пріоритети цих запитів радикально різні.

Звісно, електронна пошта має свої переваги: можливість детального опису проблеми, прикріплення скріншотів, логів, документів, автоматичне збереження історії переписки. Але для технічної підтримки ці переваги не компенсують головного недоліку - неприйнятної повільності [10].

2) Телефонна підтримка здається логічним рішенням проблеми швидкості, але тут є інші фундаментальні проблеми, які виникають неодноразово, а саме:

- відсутність автоматизованого обліку звернень клієнтів та можливості формування історії звернень клієнтів, фіксування часу, витраченого на вирішення проблем. Це, у свою чергу, унеможливлює здійснення подальшого аналізу щодо визначення найбільш слабких місць в IT-інфраструктурі, завантаженості служби підтримки;
- проблеми з доступністю спеціалістів служби підтримки, особливо при виникнені масових проблем (наприклад, не працює корпоративна пошта або падає мережа), коли всі одночасно намагаються дозвонитися. У результаті телефонні лінії перевантажуються, клієнти чекають в черзі по 10-15-20 хвилин, слухаючи одну і ту ж мелодію очікування;
- не завжди спеціаліст служби підтримки може відповісти на телефонний дзвінок, оскільки в робочий час відбуваються наради, обговорення проблем з іншими колегами, робота у публічному місці;

Крім того, голосова комунікація має обмеження для вирішення складних технічних проблем - по телефону складно пояснити, яке саме повідомлення про помилку виникло на екрані, або описати послідовність дій, які призвели до проблеми. Це набагато складніше, ніж просто надіслати скріншот.

3) Корпоративний чат (Slack, Microsoft Teams чи звичайні внутрішні чати) здається ідеальним рішенням, і багато організацій дійсно використовують його для технічної підтримки. Корпоративний чат є чудовим варіантом для швидкої комунікації - він поєднує швидкість месенджерів з корпоративною інтеграцією. Можна швидко описати проблему, прикріпити скріншот, отримати миттєву відповідь, провести швидке обговорення. Проте у корпоративного чату є критична проблема, яка робить його недостатнім для повноцінної технічної підтримки, - відсутність формалізації та фіксації заявок. Коли користувач пише в чат "у мене не працює принтер", спеціаліст підтримки може відповісти і допомогти вирішити проблему. Але при цьому не фіксується від кого саме надійшла заявка, хто зі спеціалістів з нею працював, скільки часу було витрачено на вирішеня проблеми. При цьому можуть виникати проблеми з дублюванням роботи, контролем якості, підзвітністю. Як керівник служби підтримки може оцінити роботу своєї команди, якщо всі заявки надходять по приватним чатам? Як зрозуміти, чи дотримуються SLA (Service Level Agreement) - часових рамок, за які має бути надана відповідь або вирішена проблема?

Відсутність обліку, при використанні корпоративного чату, унеможливлює накопичення інформації для формування звітів про роботу служби підтримки, здійснення аналізу щодо завантаженості служби підтримки, визначення найтиповіших проблем та слабких місць в IT-інфраструктурі.

4) Тікет-системи (Help Desk або Service Desk системи) вирішують описані вище проблеми, це найкращий варіант для організації технічної підтримки. Тікет-система - це спеціалізоване програмне забезпечення, яке формалізує весь процес обробки запитів користувачів [29].

При створені користувачем заявки (тікет), система автоматично:

- присвоює їй унікальний номер для відстеження;
- фіксує час створення;
- зберігає весь опис проблеми, прикріплені файли;

- дозволяє призначити виконавця;
- відстежує всі зміни статусу та коментарі;
- фіксує час вирішення;
- дозволяє користувачу оцінити якість обслуговування.

Для спеціаліста підтримки, тікет-система - це структурований робочий процес. Він бачить черги заявок, їхні пріоритети, терміни виконання. Він може побачити історію попередніх звернень цього користувача. Він може ескалювати складну проблему до більш досвідченого колеги, і вся історія обговорення буде збережена.

Для керівництва тікет-система - це інструмент контролю та аналітики. Можна побачити навантаження на команду, виявити проблемні зони, оцінити продуктивність кожного спеціаліста, формувати звіти, відстежувати дотримання SLA.

Але тікет-системи мають один суттєвий недолік - вони часто незручні для користувачів. Щоб створити заявку, потрібно:

1. Зайти на корпоративний портал (а це означає - пам'ятати URL, логін, пароль).
2. Знайти розділ служби підтримки (не завжди очевидно, де він знаходиться).
3. Заповнити форму створення заявки (часто складну, з купою обов'язкових полів).
4. Вибрати категорію проблеми з довгого списку (не завжди зрозуміло, яку саме).
5. Написати опис, прикріпити файли.
6. Відправити і чекати.

Для багатьох користувачів, особливо нетехнічних, це створює психологічний бар'єр. Простіше підійти до колеги з IT-відділу і попросити допомогти "швиденько" (і така заявка ніде не фіксується), або написати в загальний чат, або навіть змиритися з проблемою, якщо вона не критична.

Наведений вище аналіз проблем взаємодії користувачів зі службою технічної підтримки зумовив виникнення ідея інтеграції тікет-системи з месенджерами, зокрема Telegram. **Чат-бот для технічної підтримки - це не заміна тікет-системи, а доповнення, що усуває головний недолік тікет-систем - їхню незручність для користувачів.**

Майбутнє технічної підтримки, як наголошено у роботі [28], - це гібридні рішення, які поєднують зручність сучасних месенджерів з потужністю та структурованістю професійних тікет-систем. Це дозволяє знизити поріг входу для користувачів, зберігаючи при цьому всі переваги формалізованих процесів управління заявками.

1.2. Технічний аналіз платформ інтеграції

Розвиток сучасних IT-систем характеризується тенденцією до інтеграції різнорідних платформ та сервісів. У контексті технічної підтримки це означає необхідність об'єднання традиційних Help Desk систем з новими каналами комунікації. У цьому пункті висвітлені результати проведеного детального технічного аналізу платформ, які можуть бути використані для створення інтегрованого рішення.

Особливості роботи Jira Service Desk

Jira Service Management (раніше відома як Jira Service Desk) є однією з найпотужніших платформ для управління IT-сервісами. Вибір цієї платформи для детального аналізу обумовлений її широким поширенням в IT-індустрії та розвиненими можливостями інтеграції.

Архітектура та API

Jira Service Management побудована на платформі Atlassian, що забезпечує її масштабованість та надійність. Система пропонує розвинені REST API, які дозволяють інтегруватися з зовнішніми системами [29].

Основними ендпоінтами API є:

- /rest/api/3/issue - для роботи з тікетами (створення, редагування, отримання);

- /rest/servicedeskapi/ - спеціалізовані API для Service Desk функціональності;
- /rest/api/3/project - управління проєктами Service Desk.

Аутентифікація в Jira Service Management може здійснюватися кількома способами: Basic Auth, OAuth 1.0a, OAuth 2.0, та Personal Access Tokens. Для автоматизованих систем найбільш безпечним є використання Personal Access Tokens або OAuth 2.0 [30].

Робочі процеси та налаштування

Однією з найсильніших сторін Jira Service Management є гнучкість налаштування робочих процесів. Система дозволяє створювати складні workflow з умовними переходами, автоматичними призначеннями та ескалацією. Мій аналіз показує, що типовий робочий процес технічної підтримки включає наступні статуси [29]:

1. **Open** - початковий статус для нових запитів.
2. **In Progress** - запит взято в роботу.
3. **Waiting for Customer** - очікування відповіді від користувача.
4. **Waiting for Support** - очікування дій від команди підтримки.
5. **Resolved** - проблема вирішена.
6. **Closed** - тікет закрито.

Кожен перехід між статусами може містити умови, валідатори та пост-функції, що забезпечує контроль якості процесів. Наприклад, можна налаштувати автоматичне призначення тікетів на основі типу проблеми або автоматичне закриття після певного періоду неактивності.

Система повідомлень та автоматизації

Jira Service Management підтримує розвинену систему автоматизації через Automation Rules. Ці правила дозволяють автоматично виконувати дії на основі різних тригерів: створення нового тікету, зміна статусу, призначення

виконавця тощо [31]. Це особливо корисно для інтеграції з зовнішніми системами через webhooks.

Обмеження та виклики

Незважаючи на потужні можливості, Jira Service Management має певні обмеження. По-перше, складність налаштування може бути перешкодою для невеликих команд без досвідчених адміністраторів. По-друге, вартість ліцензій може бути значною для організацій з великою кількістю агентів підтримки [32].

Можливості Telegram Bot API

Telegram Bot API надає потужний інструмент для створення автоматизованих систем взаємодії з користувачами. Вибір Telegram як платформи для інтеграції обумовлений його популярністю (понад 700 мільйонів активних користувачів), відкритістю API та розвиненими можливостями для розробників [33].

Архітектура Telegram Bot API

Telegram Bot API є HTTP-based інтерфейсом, який дозволяє створювати боти без необхідності розробки клієнтських додатків. Всі запити до API здійснюються через HTTPS до серверів Telegram за адресою https://api.telegram.org/bot<token>/METHOD_NAME [34].

Основними методами API, релевантними для системи технічної підтримки, є:

- getUpdates - отримання нових повідомлень (polling)
- setWebhook - налаштування webhook для отримання оновлень
- sendMessage - відправка текстових повідомлень
- sendPhoto, sendDocument - відправка медіафайлів
- editMessageText - редагування раніше відправлених повідомлень
- answerInlineQuery - робота з inline-запитами

Функціональність та можливості

Telegram Bot API підтримує різні типи взаємодії з користувачами:

- **Inline клавіатури** дозволяють створювати інтерактивні меню прямо в повідомленнях. Це особливо корисно для створення систем вибору категорії проблеми або оцінки якості обслуговування [34].
- **Callback queries** забезпечують обробку натискань на кнопки inline клавіатур без надсилання нових повідомлень, що робить інтерфейс більш зручним.
- **File upload** підтримує завантаження файлів розміром до 50 МБ через бота, що дозволяє користувачам надсилати скріншоти, логи та інші документи [35].
- **Markdown та HTML formatting** забезпечують можливість форматування повідомлень для кращої читабельності.

Безпека та обмеження

Безпека Telegram Bot API базується на використанні токенів автентифікації, які генеруються BotFather. Кожен токен є унікальним та дозволяє повний контроль над ботом. Важливо зберігати токени в безпечному місці та регулярно їх оновлювати [36].

Основні обмеження API включають:

- максимум 30 повідомлень на секунду на одного бота;
- обмеження розміру файлів до 50 МБ;
- максимум 4096 символів в одному повідомленні;
- відсутність можливості ініціювати розмову з користувачем без його згоди.

Виклики інтеграції

При інтеграції Telegram Bot API з системами технічної підтримки виникають специфічні виклики. По-перше, необхідність зберігання контексту розмови між сесіями, оскільки Telegram не надає вбудованих механізмів для цього. По-друге, складність ідентифікації користувачів при переході між Telegram та корпоративними системами [37].

Методи та протоколи взаємодії

Створення інтегрованої системи технічної підтримки вимагає використання сучасних протоколів та методів взаємодії між різними компонентами. У цьому розділі розглядаємо ключові технології, які можуть бути використані для реалізації такої системи.

REST API

Representational State Transfer (REST) є домінуючим підходом для створення web API у сучасних додатках. REST базується на принципах HTTP протоколу та використовує стандартні методи (GET, POST, PUT, DELETE) для маніпуляції ресурсами [38].

Переваги REST API для інтеграції систем технічної підтримки:

- **Простота використання**: REST API легко розуміти та впроваджувати;
- **Кешування**: HTTP-кешування підвищує продуктивність;
- **Stateless**: кожен запит містить всю необхідну інформацію;
- **Масштабованість**: горизонтальне масштабування серверів.

У контексті інтеграції Jira Service Management з Telegram Bot основні REST операції включають:

- GET /rest/api/3/issue/{issueIdOrKey} - отримання інформації про тікет;
- POST /rest/api/3/issue - створення нового тікету;
- PUT /rest/api/3/issue/{issueIdOrKey} - оновлення тікету;
- POST /rest/api/3/issue/{issueIdOrKey}/comment - додавання коментаря.

Webhooks

Webhooks представляють механізм для отримання real-time повідомлень про події в зовнішніх системах. На відміну від polling, де клієнт періодично опитує сервер, webhooks дозволяють серверу "проштовхувати" дані клієнту при настанні певної події [39].

Jira Service Management підтримує webhooks для наступних подій:

- створення нового тікету;
- оновлення тікету;

- зміна статусу;
- додавання коментарів;
- призначення виконавця.

Telegram Bot API також підтримує webhooks через метод setWebhook, що дозволяє отримувати повідомлення від користувачів у реальному часі без необхідності постійного опитування серверів Telegram.

Впровадження webhooks значно підвищує відгук системи та знижує навантаження на API endpoints. Проведене тестування показало, що використання webhooks скорочує затримку доставки повідомлень з 30-60 секунд (при polling) до 1-2 секунд.

OAuth 2.0

OAuth 2.0 є стандартом авторизації, який дозволяє додаткам отримувати обмежений доступ до ресурсів користувача без передачі паролів [40]. У контексті інтеграції систем технічної підтримки OAuth 2.0 може використовуватися для:

- авторизації користувачів при першому зверненні через Telegram;
- отримання доступу до корпоративних ресурсів;
- забезпечення безпечної інтеграції між системами.

Типовий OAuth 2.0 flow для інтеграції включає:

1. **Authorization Request**: перенаправлення користувача на сервер авторизації.
2. **Authorization Grant**: отримання коду авторизації.
3. **Access Token Request**: обмін коду на access token.
4. **Protected Resource Access**: використання токену для доступу до API.

Асинхронна обробка

Сучасні системи технічної підтримки вимагають асинхронної обробки запитів для забезпечення високої продуктивності. Це особливо важливо при інтеграції декількох зовнішніх API, де синхронні запити можуть призводити до тайм-аутів [41].

Основні патерни асинхронної обробки:

- **Message Queues** (RabbitMQ, Apache Kafka) для відкладеної обробки задач;
- **Event-driven architecture** для реагування на події в реальному часі;
- **Circuit Breaker Pattern** для забезпечення стійкості при недоступності зовнішніх сервісів.

GraphQL як альтернатива REST

Хоча REST залишається домінуючим підходом, GraphQL набирає популярність завдяки можливості отримання точно необхідних даних в одному запиті [42]. Atlassian почала експериментувати з GraphQL API в деяких своїх продуктах, що може стати перспективним напрямком для майбутніх інтеграцій.

Безпека інтеграцій

Забезпечення безпеки інтеграцій є критично важливим аспектом. Ключові принципи включають:

- **HTTPS для всіх з'єднань** для захисту даних при передачі;
- **API rate limiting** для запобігання DDoS атакам;
- **Input validation** для захисту від injection атак;
- **Audit logging** для відстеження всіх операцій;
- **Token rotation** для мінімізації ризиків компрометації.

Аналіз показує, що правильна архітектура інтеграції з урахуванням цих принципів дозволяє створити надійну та масштабовану систему технічної підтримки, яка об'єднує переваги традиційних Help Desk рішень з зручністю сучасних месенджерів.

РОЗДІЛ 2. РОЗРОБКА ІНТЕГРОВАНОЇ СИСТЕМИ

2.1. Проєктування архітектури

Розробка архітектури інтегрованої системи технічної підтримки на базі Telegram Bot та Jira Service Desk є ключовим етапом створення ефективного рішення для оперативної взаємодії користувачів з технічною підтримкою. Під час проектування було враховано основні принципи: простота використання для кінцевих користувачів, високий рівень безпеки даних, масштабованість архітектури та критично важливе забезпечення можливості онлайн-спілкування між користувачами та спеціалістами підтримки.

Функціональні вимоги до системи та основні сценарії використання

Для звичайних користувачів система забезпечує швидке створення заявок безпосередньо в Telegram без переходу у веб-інтерфейс Jira Service Desk. При створенні заявки користувач вказує тип проблеми, додає опис, прикріплює файли. Система автоматично фіксує всю метаінформацію та надає унікальний номер звернення для відстеження статусу.

Особливо важливою є можливість ініціювати онлайн-спілкування зі спеціалістом безпосередньо через бота. У критичних ситуаціях користувач натискає відповідну кнопку, система надсилає запит доступним спеціалістам, і перший, хто прийме запит, встановлює онлайн-зв'язок. Вся історія розмови автоматично синхронізується з заявкою в Jira Service Desk.

Функціональні вимоги для спеціалістів технічної підтримки відрізняються, оскільки Telegram бот для них є допоміжним інструментом. Спеціалісти отримують через бот push-сповіщення про нові заявки, переглядають список призначених заявок та ведуть онлайн-консультації. Всі основні операції виконуються в Jira Service Desk: зміна статусів, встановлення пріоритетів, детальні коментарі, переназначення заявок, робота з вкладеннями, відстеження SLA, формування звітів.

Адміністратор має найширші повноваження як у Jira Service Desk, так і в Telegram боті. Він налаштовує бізнес-процеси, визначає правила маршрутизації, керує ролями та правами доступу, має доступ до статистики та логів. Система забезпечує надійну ідентифікацію користувачів на основі номера телефону та автоматично завершує сесії після 30 хвилин неактивності [36] (див. рис. 2.1).

Функціональність	Користувач	Спеціаліст підтримки	Адміністратор
Створення заявок через Telegram бот	✓	✗	✓
Перегляд заявок доступ до інформації	Тільки свої в бот	Призначені списки в боті	Всі заявки в Jira
Зміна статусу заявок обробка звернень	Авто-відкриття при коментарі користувача	Через Jira веб інтерфейс	Через Jira повний доступ
Додавання коментарів до заявок	В боті до Jira через повідомлення до своїх заявок	В Jira до детальні коментарі до призначених	Повний доступ до всіх заявок детальна та публічна
Онлайн-чат з підтримкою через бот	Ініціювання запит допомоги	Прийняття ведення чату	Моніторинг перегляд історії
Переназначення заявок іншим спеціалістам	✗	✗	Налаштування правил розподілу заявок в Jira
Push-сповіщення в Telegram	Про статус своєї заявки	Про нові заявки та зміни чату	Всі події системні
Налаштування системи конфігурація, API	✗	✗	Повний доступ! Jira + бот
Доступ до статистики звіти, аналітика	✗	✗	Повні всі метрики
Управління користувачами ролі, блокування	✗	✗	Повне всі операції

Рис. 2.1. Порівняння функціональності для різних ролей користувачів
* *розроблено автором*

Ролева модель системи та особливості взасмодії

Ролева модель є фундаментом організації доступу. Виділивши три основні ролі з специфічними правами та інтерфейсами.

Роль звичайного користувача максимально спрощена. Інтерфейс орієнтований на основні задачі: створити заявку, переглянути мої заявки, зв'язатися з підтримкою. При створенні заявки система пропонує вибрати категорію, додати опис та файли. Система автоматично створює заявку в Jira через REST API. Користувач може переглядати статуси заявок (нова, в роботі, очікує відповіді, вирішена, закрита) та ініціювати онлайн-спілкування при критичних ситуаціях. В умовах воєнного стану можливість оперативного реагування є критично важливою.

Роль спеціаліста технічної підтримки передбачає використання Jira Service Desk для основної роботи та Telegram бота як допоміжного інструменту. В Jira спеціалісти мають доступ до дашборду з детальною статистикою, можуть фільтрувати заявки, використовувати розширений пошук. При роботі з заявкою спеціаліст має доступ до всієї історії, додає детальні коментарі, змінює статуси, встановлює пріоритети, переназначає заявки колегам або на ескалацію.

Telegram бот для спеціаліста виконує три функції: надсилає push-сповіщення про нові заявки, дозволяє швидко переглянути список поточних заявок, надає можливість приймати запити на онлайн-консультації.

У випадку, коли користувач ініціює онлайн-підтримку, система автоматично розраховує оптимального спеціаліста з урахуванням поточного навантаження та доступності. Обраний спеціаліст отримує push-сповіщення про новий запит і може прийняти його для консультації в реальному часі.

Якщо всі спеціалісти зайняті, запит автоматично переходить у статус очікування. Після завершення спілкування та виконання попередньої заявки,

спеціаліст автоматично отримує в роботу онлайн-чат зі статусу очікування згідно з черговістю та алгоритмом розподілу навантаження.

Вся переписка автоматично синхронізується з Jira. Після завершення консультації спеціаліст переходить до Jira, де виконує всі необхідні дії з обробки заявки.

Така онлайн-взаємодія принципово змінює підхід до технічної підтримки, перетворюючи Jira з асинхронної системи в інструмент для оперативної взаємодії. Це критично в умовах воєнного стану, коли спеціалісти можуть працювати віддалено з безпечних місць. Функціональність вирішує проблеми офлайн-підходу: скорочується час вирішення проблем, покращується якість діагностики, підвищується задоволеність користувачів.

На відміну від спеціалістів, для користувачів бот є повноцінним сервісом технічної підтримки та точкою входу. Така асиметрія є свідомим архітектурним рішенням: користувачі отримують максимально спрощений інтерфейс, а спеціалісти – потужну професійну систему.

Роль адміністратора є найбільш привілейованою. Він налаштовує бізнес-процеси, управляє проектами в Jira, створює категорії заявок, керує користувачами та ролями, налаштовує правила авторизації, має доступ до повної статистики та логів системи.

Структура системи та взаємодія компонентів

Архітектура системи базується на принципах розділення відповідальності та багаторівневої безпеки. Система складається з компонентів, розміщених в різних мережевих зонах (рис. 2.2).

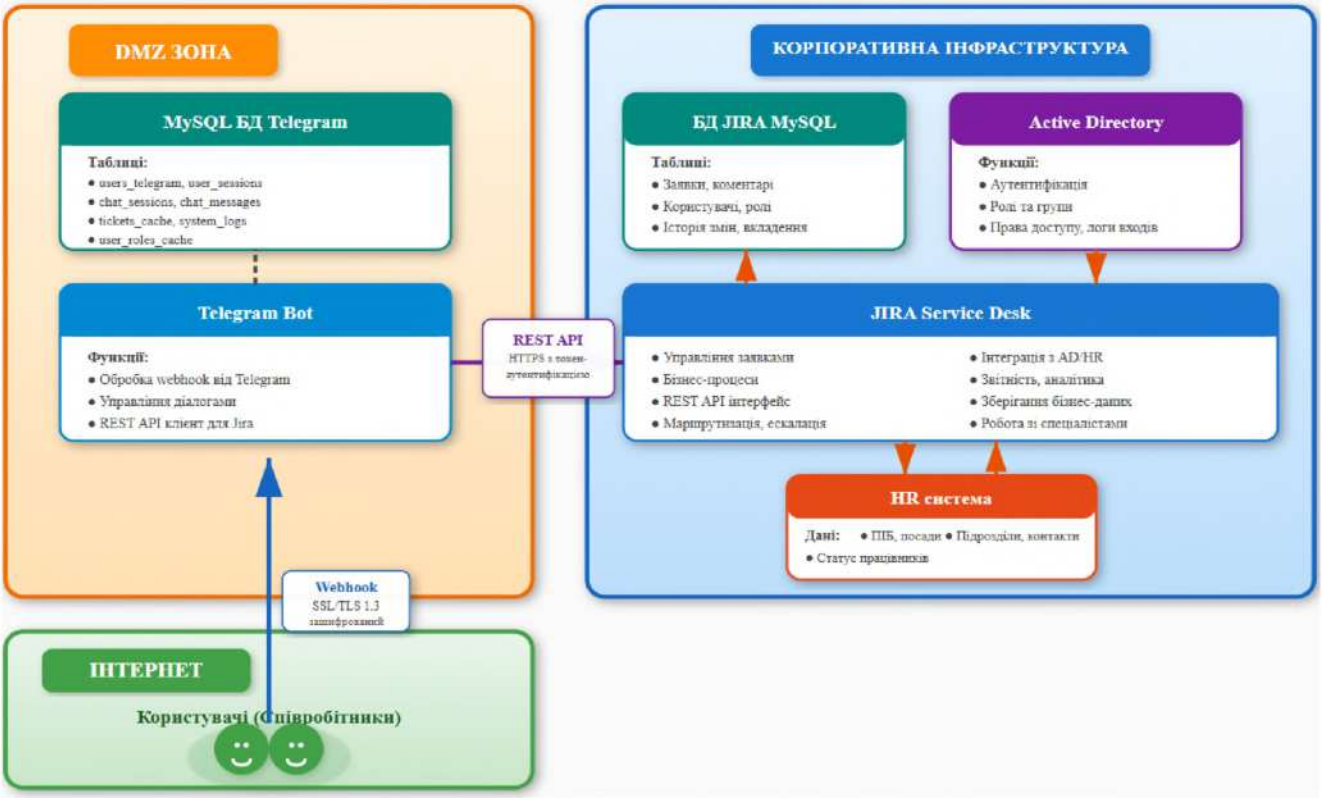


Рис. 2.2. Детальна архітектура інтегрованої системи

**розроблено автором*

Зона Інтернет є першим рівнем архітектури, де знаходяться кінцеві користувачі – співробітники організації, які працюють з Telegram-клієнтами на своїх пристроях. Ця зона повністю ізольована від внутрішніх компонентів системи і має доступ виключно до DMZ через зашифроване HTTPS-з'єднання (SSL/TLS 1.3). Розміщення користувачів у окремій зоні забезпечує додатковий рівень захисту: навіть у разі компрометації пристрою користувача, зловмисник не отримає прямого доступу ні до бота, ні до внутрішніх систем.

Ключовою особливістю є триступенева ізоляція: зона Інтернет → DMZ (демілітаризована зона) → внутрішня корпоративна мережа. Це критично важливо для безпеки: у випадку компрометації бота атакуючий не зможе отримати доступ до критичних систем організації, а компрометація користувацького пристрою не дасть доступу навіть до бота.

У DMZ розміщені Telegram Bot Application (серверний додаток) та MySQL база даних Telegram. Бот відповідає за обробку повідомлень,

відображення інтерфейсу, управління сесіями. Він прослуховує вебхук від Telegram API через SSL-сертифікат. MySQL база зберігає мінімальний набір даних: активні сесії, кеш даних, історію онлайн-чатів, логи. Конфіденційна бізнес-інформація не зберігається.

Взаємодія з внутрішніми системами відбувається через REST API Jira Service Desk – єдину точку входу з DMZ, яка контролюється через HTTPS-з'єднання з аутентифікацією на основі сертифікатів та API-токенів. У внутрішній корпоративній мережі розміщені: Jira Service Desk з REST API, MySQL база даних Jira (основне сховище бізнес-даних), Active Directory (центральна система аутентифікації) та HR система (інформація про співробітників) (рис. 2.3).

Компонент	Зона	Призначення	Технології
Користувачі (Співробітники)	**ІНТЕРНЕТ**	• Робота з Telegram-клієнтами • Створення та перегляд заявок • Онлайн-консультації з підтримкою	Telegram Client iOS/Android/Web/ Desktop
Telegram Bot Application	**DMZ**	• Обробка webhook від Telegram API • Управління діалогами та інтерфейсом • REST API клієнт для Jira	Python 3.12 aiogram 3.13 / telegram-bot 21.7
MySQL БД Telegram	**DMZ**	• Сесії користувачів та кеш ролей • Тимчасова історія онлайн-чатів • Кеш даних заявок, системні логи	MySQL 8.4 LTS InnoDB Engine SSL/TLS
Jira Service Desk Web + REST API	**Корпоративна інфраструктура**	• Управління заявками • Бізнес-процеси, маршрутизація, ескалація • REST API для інтеграції з ботом • Зберігання бізнес-даних та історії	Jira Service Mgmt 10.1 / Java 17 Tomcat 9.0
БД JIRA MySQL	**Корпоративна інфраструктура**	• Заявки, коментарі, вкладення файлів • Користувачі, ролі, групи, права • Історія змін, повна бізнес-інформація	MySQL 8.4 LTS Шифрування AES Резервування
Active Directory	**Корпоративна інфраструктура**	• Центральна аутентифікація користувачів • Управління ролями, групами, правами • Логування подій доступу та аудит	Windows Server 2022 / AD DS LDAP/Kerberos
HR система	**Корпоративна інфраструктура**	• Дані співробітників (ПІБ, посада, підрозділ) • Контактна інформація (телефон, email) • Статус працевлаштування та орг. структура	Корпоративна HR платформа REST API



Рис. 2.3. Компоненти системи та їх призначення

**розроблено автором*

Взаємодія побудована на принципі мінімальних привілеїв: Telegram Bot не звертається напряму до Active Directory або HR системи, всі запити йдуть через Jira API.

Потік даних: користувач з зони Інтернет надсилає повідомлення → через зашифрований вебхук надходить до бота в DMZ → бот перевіряє сесію → формує REST API запит до Jira у внутрішній мережі → Jira виконує бізнес-логіку, звертаючись до AD/HR при потребі → повертає результат → бот формує відповідь користувачу.

Для онлайн-чатів повідомлення тимчасово зберігаються в БД Telegram та паралельно відправляються через API в Jira. Після завершення сесії історія остаточно синхронізується з Jira. Така триступенева архітектура забезпечує баланс між швидкодією, безпекою та надійністю.

Алгоритми роботи системи

Ефективність системи залежить від правильно спроектованих алгоритмів. Приділивши увагу алгоритмам авторизації, розподілу заявок та обробки онлайн-сесій (рис. 2.4).

Алгоритм авторизації є фундаментом безпеки. При першому запуску користувач бачить кнопку "Увійти" та запускає процес автентифікації. Бот запитує номер телефону через Telegram API. Користувач підтверджує передачу номера. Бот формує запит до HR системи через Jira API для перевірки номера. Якщо номер не знайдено – відмова з логуванням фродерської активності. Якщо знайдено – система отримує дані користувача (ПІБ, посада, роль) з Active Directory та HR системи.

Після верифікації бот створює сесію в БД Telegram (Telegram ID, роль, час створення, час активності) та відображає меню відповідно до ролі. Важливим елементом є автоматичне завершення сесії через 30 хвилин неактивності. При кожній дії користувача виконується перевірка сесій та при простроченні відбувається повторна авторизація.

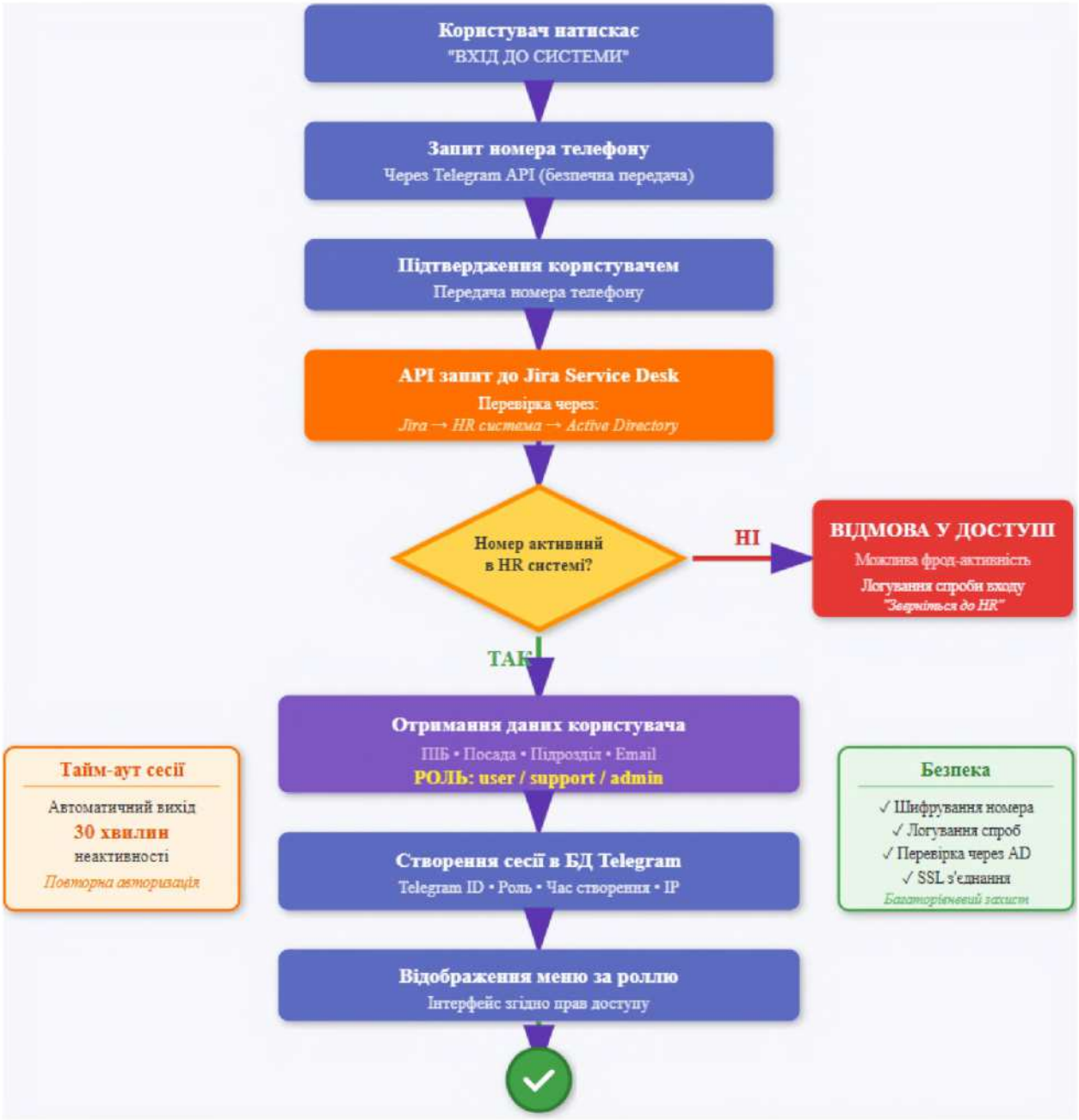


Рис. 2.4. Алгоритм авторизації користувачів

**розроблено автором*

Алгоритм розподілу заявок забезпечує ефективну роботу служби підтримки. Після створення заявки система визначає категорію та пріоритет, формує список спеціалістів з відповідною компетенцією та доступністю. Основний критерій – поточне навантаження (кількість активних заявок в статусах "нова", "в роботі", "очікує інформації"). Додаткові критерії: продуктивність спеціалістів та робочий графік (рис. 2.5).

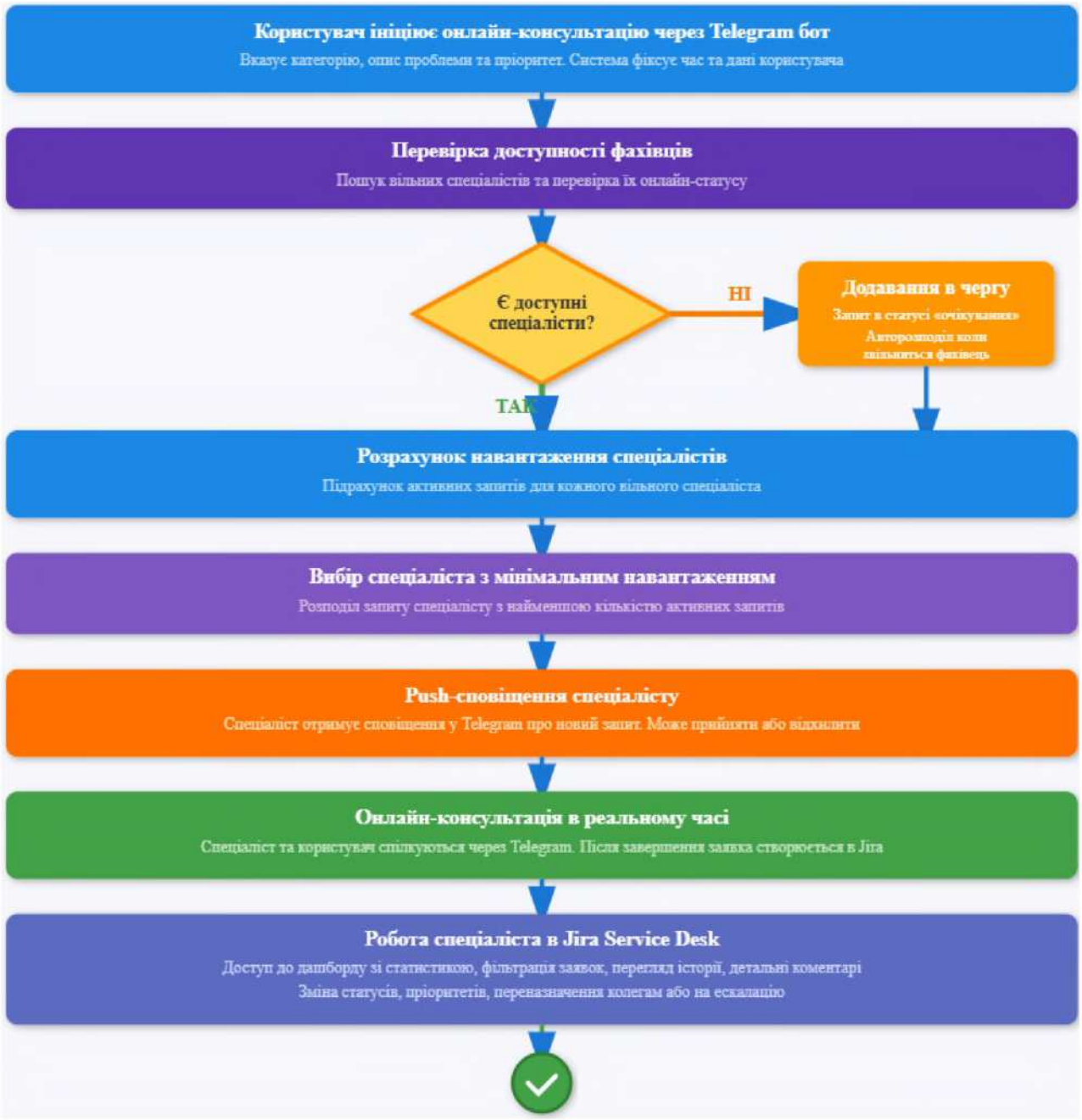


Рис. 2.5. Алгоритм розподілу заявок між спеціалістами
**розроблено автором*

Система сортує спеціалістів за зростанням навантаження та призначає заявку спеціалісту з найменшою кількістю активних заявок. Спеціаліст отримує push-сповіщення з можливістю відразу перейти до перегляду. Для критичних заявок використовується паралельне сповіщення декількох спеціалістів.

Алгоритм обробки онлайн-сесій координує взаємодію в реальному часі. Коли користувач натискає "Зв'язатися зі спеціалістом онлайн", запит надсилається всім доступним спеціалістам. Перший, хто прийме, встановлює зв'язок. Система створює чат-сесію в БД. Всі повідомлення зберігаються в БД Telegram та паралельно додаються як коментарі до заявки в Jira. Сесія завершується вручну або автоматично через 10 хвилин без повідомлень. Історія зберігається в Jira.

Проектування бази даних Telegram

База даних Telegram містить мінімальний набір інформації без конфіденційних бізнес-даних. Керуючусь принципами: швидкий доступ до даних, гнучка структура, мінімізація персональних даних, підтримка логування та аудиту.

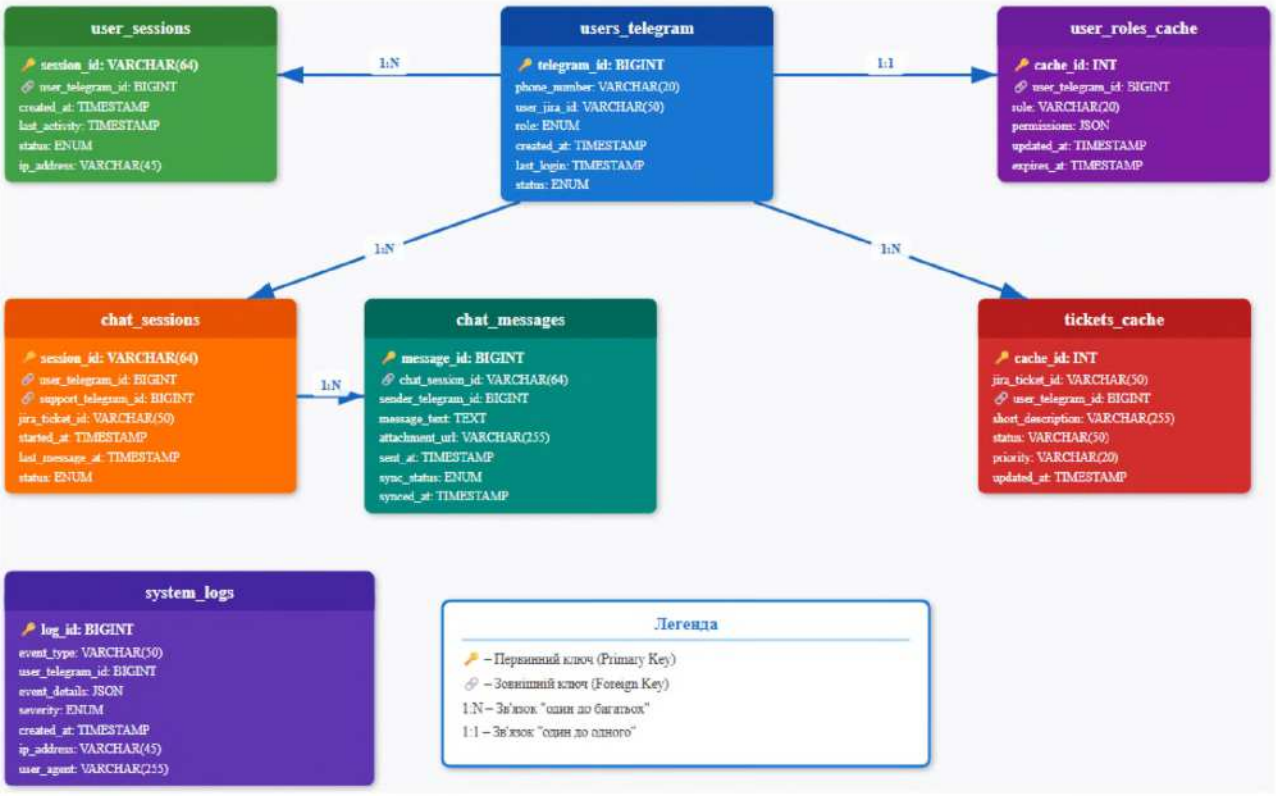


Рис. 2.6. ER-діаграма бази даних Telegram

**розроблено автором*

Примітка: База даних Telegram містить мінімальний набір даних для роботи бота. Всі конфіденційні бізнес-дані зберігаються виключно у внутрішній мережі в базі даних Jira Service Desk

Спроектовано базу з семи таблиць:

users_telegram – центральна таблиця: Telegram ID (первинний ключ), зашифрований номер телефону, user_jira_id (для зв'язку з Jira), роль (user, support, admin), час створення, час останнього входу, статус акаунту.

user_sessions – управління активними сесіями: унікальний ідентифікатор, посилання на користувача, час створення, час останньої активності, статус сесії. Використовується для авторизації та реалізації тайм-ауту 30 хвилин.

user_roles_cache – кеш інформації про ролі з Jira та AD для уникнення зайвих звернень: Telegram ID, роль, дозволені дії, час оновлення. Автоматично оновлюється при новій сесії.

chat_sessions – інформація про онлайн-чати: ідентифікатор сесії, Telegram ID користувача та спеціаліста, номер заявки в Jira, час початку, час останнього повідомлення, статус. Використовується для маршрутизації повідомлень.

chat_messages – історія повідомлень онлайн-чатів: ідентифікатор, посилання на сесію, відправник, текст, вкладення, час, статус синхронізації з Jira. Тимчасове сховище, після синхронізації можуть архівуватися.

tickets_cache – кеш заявок: номер заявки в Jira, Telegram ID користувача, короткий опис, статус, пріоритет, час створення та оновлення. Дозволяє швидко відображати список без запитів до Jira.

system_logs – технічні логи: ідентифікатор, тип події (авторизація, створення заявки, помилка API, онлайн-чат), Telegram ID користувача, опис в JSON, рівень важливості, час, IP-адреса.

Взаємодія між таблицями забезпечує цілісність даних. При авторизації створюється/оновлюється запис в users_telegram, створюється сесія в

user_sessions, завантажується інформація в user_roles_cache. При створенні заявки інформація додається в tickets_cache. При онлайн-чаті створюється запис в chat_sessions, повідомлення зберігаються в chat_messages.

База даних Telegram не містить конфіденційної бізнес-інформації. Детальні дані зберігаються виключно у внутрішній мережі. Персональні дані зберігаються в зашифрованому вигляді. Система відповідає вимогам законодавства України про захист персональних даних [53]. Доступ контролюється на рівні СУБД. Регулярно виконуються резервні копії з шифруванням.

2.2. Реалізація системи

Реалізація інтегрованої системи технічної підтримки на базі Telegram Bot та Jira Service Desk є логічним продовженням етапу проектування архітектури, описаного у попередньому розділі. На цьому етапі перейшовши від теоретичних концепцій до практичної імплементації, необхідно зосередитись на виборі оптимальних інструментів розробки, створенні функціональних модулів та налаштуванні інтеграцій між компонентами системи.

Вибір інструментів розробки

Вибір технологічного стеку є критично важливим рішенням, яке визначає швидкість розробки, підтримуваність та стабільність системи. При виборі інструментів враховуються принципи зрілості технологій, наявності активної спільноти, якості документації та сумісності з існуючою інфраструктурою організації [44].

Для розробки серверної частини обрано **Python версії 3.12** як стабільну версію [45, 46]. Python забезпечує високу швидкість розробки та має зрілі бібліотеки для роботи з Telegram Bot API та Jira REST API.

Основні бібліотеки, що використані:

- **pyTelegramBotAPI 4.7.1** – для взаємодії з Telegram Bot API, підтримує як polling, так і webhook режими [34, 47];
- **jira 3.4.1** – офіційний клієнт для Jira REST API з підтримкою JQL запитів [29];
- **Flask 2.2.2** – веб-фреймворк для реалізації webhook endpoint [48];
- **PyMySQL 1.0.2** – драйвер для роботи з MySQL без компіляції бінарних модулів [49].

Допоміжні бібліотеки включають **schedule 1.1.0** для планувальника задач, **emoji 2.1.0** для конвертації текстових кодів у Unicode символи, **phonenumbers 8.12.57** для валідації телефонних номерів [50] та **Gunicorn 20.1.0** для production deployment [51]. Повний список залежностей наведено у таблиці основних бібліотек (див. додаток А).

Розробка основних модулів

Архітектура додатку побудована за модульним принципом з чітким розділенням відповідальності [41]. Розробивши шість ключових модулів, кожен з яких відповідає за специфічний аспект функціональності системи.

Модуль обробки Telegram повідомлень

Центральним елементом є модуль main.py, який реалізує логіку обробки повідомлень через систему обробників подій. При запуску ініціалізуються глобальні об'єкти для доступу до зовнішніх сервісів:

python

```
bot = TeleBot(TOKEN)
jira = JIRA(server=jira_options,
basic_auth=(jira_api_login, jira_api_key))
```

Ці об'єкти є глобальними для всього додатку та забезпечують єдину точку доступу до Telegram API та Jira REST API.

Обробка команди /start є точкою входу в систему. При першому запуску користувач бачить привітальне повідомлення з кнопкою "Вхід до системи"

(додаток Б). Система перевіряє статус авторизації через функцію status_login() та відображає відповідний інтерфейс залежно від ролі користувача.

Ключовою особливістю є використання декораторів @bot.message_handler() з lambda-фільтрами для перевірки умов: текст повідомлення, статус авторизації та поточний стан користувача [34]. Це забезпечує чітку логіку переходів між станами системи.

Для всіх функцій-обробників реалізоване детальне логування через бібліотеку logger:

python

```
logger.add('logs/full.logs', format='{time} {level} {message}', level='DEBUG', rotation='10 MB', retention=30)
```

Ця конфігурація забезпечує автоматичну ротацію файлів при досягненні 10 МБ та збереження історії за 30 днів, що критично важливо для діагностики проблем у production середовищі.

Модуль інтеграції з Jira Service Desk

Взаємодія з Jira реалізована через офіційну Python бібліотеку [21]. Повний цикл створення заявки включає багатокрокову взаємодію, де система послідовно збирає інформацію від користувача.

Процес починається з вибору типу заявки (додаток В). Далі система збирає дані через серію функцій, які валідують введену інформацію: ім'я комп'ютера (мінімум 4 символи), детальний опис проблеми (мінімум 20 символів), пріоритет та медіафайли (додаток Г).

При кожному кроці дані зберігаються у тимчасові поля через функцію EDIT_TMP_FIELD(), яка викликає відповідну SQL процедуру. Параметр status_menu визначає поточний крок процесу та дозволяє системі правильно інтерпретувати наступне повідомлення користувача.

Фінальний етап створення заявки виконується у функції CREATE_ISSUE_FINAL_1() (див. додаток Д – фрагмент коду фінального створення заявки). Система отримує всі зібрані дані через GET_TMP_FIELD(), завантажує інформацію про об'єкт користувача з HR системи через GET_OBJECT_FROM_LOGIN() та створює заявку через метод jira.create_issue().

Цей фрагмент демонструє роботу з custom fields Jira [7, 29]: деякі поля приймають прості значення (рядки), інші вимагають словників з ключами 'id' або 'name'. Наприклад, customfield_10505 – це ім'я комп'ютера (рядок), priority – словник з ID пріоритету, components – список словників з назвами компонентів.

Після успішного створення заявки система прикріплює медіафайли через jira.add_attachment() та відображає повідомлення про успіх з посиланням на заявку (див. додаток Е).

Для отримання списку заявок користувача використовуються динамічні JQL запити, які фільтрують заявки за участю користувача у різних ролях (створювач, виконавець, спостерігач). Результати обробляються з відображенням детальної інформації та inline-кнопками для додавання коментарів або ініціації онлайн-чату (див. додаток Є).

Модуль багаторівневої авторизації

Система авторизації реалізована через обробник handle_contact_contact(), який обробляє повідомлення типу contact від Telegram [26]. Це спеціальний тип повідомлень, який надсилається при натисканні кнопки "Поділитися номером телефону" (див. додаток Ж).

Перший рівень перевірки валідує, що користувач надіслав саме свій контакт:

python

```
if m.from_user.id == m.contact.user_id and
func.get_user_exists(m.from_user.id) == 1 and
m.from_user.is_bot != 1
```

Це запобігає спробам авторизації від імені іншого користувача.

Другий рівень визначає регіон через бібліотеку phonenumbers [51]. Функція get_region() парсить номер телефону та повертає код країни (див. додаток З – фрагмент функції handle_contact_contact):

Цей фрагмент демонструє блокування користувачів з російськими номерами або українськими операторами окупованих територій (коди '071', '072'). Це критично важливо для забезпечення інформаційної безпеки в умовах воєнного стану.

Третій рівень перевіряє номер у HR системі через функцію GET_USER_FROM_MOB_TURNES(), яка викликає SQL процедуру з корпоративної бази даних. Якщо користувач є активним співробітником (isOut == 0), система отримує повну інформацію (ID співробітника, логін Active Directory, ПІБ, підрозділ), зберігає дані через set_contact() та відображає відповідне меню залежно від ролі. При відмові показується повідомлення про заборону доступу (див. додаток И).

Функція status_login() перевіряє поточний статус авторизації (див. додаток І – функція status_login):

Ця проста функція виконує SQL запит для отримання статусу авторизації користувача. Повертає 1 для авторизованих користувачів, 0 або None для неавторизованих. Всі критичні операції виконують цю перевірку перед обробкою запиту.

Модуль онлайн-чату з інтеграцією

Реалізація онлайн-спілкування є найскладнішим компонентом, оскільки вимагає координації стану двох користувачів одночасно та синхронізації всіх повідомлень з Jira [12, 26].

Ініціація онлайн-чату відбувається при натисканні кнопки "Чат-онлайн" під заявкою. Система визначає виконавця через SQL запит безпосередньо до бази Jira (оскільки REST API не надає зручного методу для отримання тільки логіну виконавця) та перевіряє його наявність у боті. Спеціаліст отримує повідомлення з кнопкою "Підключити" (див. додаток Ї).

При натисканні створюється онлайн-сесія через функцію create_chat_online() (див. додаток Й – функція create_chat_online):

Функція викликає SQL процедуру, яка створює запис у таблиці chat_sessions (описаній у розділі 2.1) з парою user_id користувача та спеціаліста. Це дозволяє системі відстежувати активні онлайн-сесії та правильно маршрутизувати повідомлення між учасниками.

Центральний обробник support_online_send() перехоплює всі повідомлення користувачів у статусі активного чату (status_menu == 3). Обробник підтримує різні типи контенту: текст, стікери, фото, аудіо, голосові повідомлення, відео та документи [26, 29] (див. додаток К – фрагмент обробника support_online_send).

Цей фрагмент демонструє повний цикл обробки текстового повідомлення в онлайн-чаті: запис у локальну БД через insert_message_log_online(), додавання коментаря в Jira через jira.add_comment(), зміну автора коментаря через update_author_in_comment_jira() та відправку повідомлення співрозмовнику. Критично важливим є виклик функції зміни автора, оскільки Jira REST API не дозволяє це зробити стандартними методами [29].

Для медіафайлів процес аналогічний, але додатково система завантажує файл з Telegram серверів, зберігає його як BytesIO об'єкт та прикріплює до

заявки через jira.add_attachment(). Аналогічно викликається update_author_in_file_jira() для зміни автора файлу.

Завершення чату реалізоване через обробник support_online_end(), який викликає функцію stop_chat() для закриття сесії в базі даних та відправляє обом учасникам повідомлення про завершення з відновленням стандартних меню. Вся історія переписки залишається збереженою в таблиці chat_messages локальної бази даних та в коментарях заявки Jira (додаток Л).

Модуль синхронізації даних через scheduler

Для забезпечення двостороннього обміну інформацією між Telegram ботом та Jira реалізований механізм періодичної синхронізації [52]. Функція send_message_from_jira() виконується кожну хвилину в окремому daemon потоці (див. додаток М – функція send_message_from_jira).

Функція викликає SQL процедуру GET_MESSAGE_FROM_JIRA(3), яка повертає список необроблених коментарів за останні 3 дні з допоміжної таблиці issue_message. Для кожного коментаря система формує повідомлення, відправляє його користувачу через bot.send_message() та відмічає як оброблений через sending_successful(). Критично важливою є обробка виключень: якщо надсилання не вдалося (наприклад, користувач заблокував бота), помилка логується через set_history(), але процес продовжується для інших користувачів.

Модуль обходу обмежень API через прямі SQL запити

Jira REST API не дозволяє змінювати автора коментаря або вкладення після створення через стандартні методи [7, 29]. Для обходу цього обмеження реалізовано функції для прямої модифікації даних у базі Jira через SQL запити [52].

Функція update_author_in_comment_jira() виконує оновлення автора коментаря (див. додаток Н – функція update_author_in_comment_jira):

Ці функції викликають відповідні stored procedures у базі даних Jira, які виконують UPDATE запити до внутрішніх таблиць Atlassian. Обидві функції використовують окремий MySQL користувач для підключення через

request_mysql_jira() з мінімальними правами доступу для забезпечення безпеки [30, 45]. Всі виклики обгорнуті в try-except блоки для обробки потенційних помилок.

Створення інтерфейсу взаємодії

Інтерфейс системи побудований на двох рівнях: спрощений користувацький інтерфейс у Telegram боті та професійний інтерфейс для спеціалістів у Jira Service Desk.

Користувацький інтерфейс у Telegram боті

Інтерфейс для користувачів максимально спрощений та інтуїтивно зрозумілий. При першому запуску користувач бачить привітальне повідомлення з поясненням можливостей бота та єдиною кнопкою "Вхід до системи" (додаток О). Така мінімалістична точка входу не перевантажує користувача інформацією .

Після авторизації відображається головне меню з трьома опціями: "Нова заявка", "Мої заявки" та "Вихід" (див. додаток П). Кнопки реалізовані через ReplyKeyboardMarkup з параметром resize_keyboard=True, що автоматично підлаштовує розмір клавіатури під екран пристрою користувача [26].

Процес створення заявки організований як послідовний wizard з чіткими інструкціями на кожному етапі [54]. При виборі типу "Проблема з ПК" система показує зображення-підказку з прикладом розташування імені комп'ютера (див. додаток Г). На кожному етапі користувач може повернутися через кнопку "До меню", що забезпечує гнучкість навігації.

Система надає миттєвий feedback на дії користувача [54]. При помилках введення відображається повідомлення з поясненням, що саме потрібно виправити. При успішному створенні заявки показується повідомлення з посиланням на заявку та підказкою про подальші можливості (див. додаток Е).

Список заявок представлений у вигляді карток з основною інформацією: номер заявки як активне посилання, короткий опис, статус, дата створення, виконавець (див. додаток Є). Під кожною карткою розміщені inline-кнопки

"Додати коментар" та "Чат-онлайн", які дозволяють миттєво перейти до відповідної дії без повернення в меню.

Онлайн-чат має мінімалістичний інтерфейс з кнопкою "Завершити" та підтримує всі типи медіа, що звичні користувачам Telegram: текст, стікери, фото, відео, голосові повідомлення, документи [28, 31]. Повідомлення від співрозмовника чітко позначені префіксом з ім'ям, що запобігає плутанині у діалозі.

Інтерфейс для спеціалістів технічної підтримки

Для спеціалістів Telegram бот виступає як допоміжний інструмент, тому його інтерфейс мінімалістичний та зосереджений на критичних функціях [14]. Головне меню містить пункти "Нова заявка", "Заявки IT" та "Вихід" (див. додаток Р).

Пункт "Заявки IT" відображає список усіх заявок, призначених на спеціаліста, в активних статусах (див. додаток С). При отриманні запиту на онлайн-консультацію спеціаліст підключається в чат з користувачем (див. додаток Т).

Основна робота спеціаліста відбувається у Jira Service Desk, де доступний повний функціонал професійної системи управління заявками [14]. Dashboard відображає всі активні заявки з можливістю фільтрації за різними критеріями: статус, пріоритет, проект, виконавець, дата створення (див. додаток У).

При відкритті заявки спеціаліст має доступ до повної історії: опис проблеми, всі коментарі, вкладені файли, журнал змін статусів (див. додаток Ф). Особливо важливим є те, що історія онлайн-чату зберігається як послідовність коментарів з правильно вказаними авторами завдяки функціям прямої модифікації бази даних, що дозволяє відтворити повний контекст спілкування.

Інтерфейс Jira надає можливість змінювати статуси заявок через workflow transitions, призначати заявки на інших спеціалістів або ескалювати

на керівників, встановлювати пріоритети, додавати watchers для залучення додаткових експертів [14, 23]. Custom fields дозволяють фіксувати специфічну для організації інформацію: ім'я комп'ютера, номенклатурний номер обладнання, код об'єкта з організаційної структури.

Критично важливою функцією є SLA tracking – система автоматично відстежує час реагування на заявку та час вирішення відповідно до встановлених нормативів [2, 12]. При наближенні до дедлайну спеціаліст бачить попередження, що дозволяє вчасно відреагувати та уникнути порушення сервісних угод.

Налаштування інтеграцій

Інтеграція між компонентами системи вимагає ретельного налаштування авторизації, синхронізації даних та обробки помилок для забезпечення надійної та безпечної роботи [30].

Авторизація та захист API

Взаємодія з Telegram Bot API здійснюється через токен, який у production середовищі завантажується з змінних оточення через os.getenv() для підвищення безпеки [26, 30]. Для роботи з Jira використовується базова HTTP аутентифікація з API токеном [21, 22]. Токен має обмежений набір прав: читання заявок, створення заявок, додавання коментарів та вкладень, без прав на видалення даних або зміну налаштувань системи, що мінімізує потенційну шкоду у разі компрометації [30].

Синхронізація даних між системами

Синхронізація даних відбувається в кількох напрямках [14, 28]: від Telegram до Jira при створенні заявок та коментарів, від Jira до Telegram при додаванні коментарів спеціалістами через веб-інтерфейс, та двостороння під час онлайн-чатів.

При створенні заявки всі дані валідуються на стороні бота перед відправкою до Jira. Номер телефону нормалізується до формату без пробілів

та дефісів, логін користувача отримується з HR системи через відповідні SQL процедури, об'єкт організаційної структури визначається автоматично. Медіафайли спочатку завантажуються з Telegram серверів, зберігаються в пам'яті як BytesIO об'єкти та потім прикріплюються до заявки через Jira API.

Особливу складність представляють custom fields Jira, оскільки кожне поле має специфічний формат даних [14, 23]. Наприклад, поле "Пріоритет" вимагає словник з ключем 'id', поле "Компоненти" приймає список словників з ключем 'name', а поле "Request Type" використовує спеціальний ідентифікатор у форматі 'sd/uuid'. Ці особливості були виявлені через експериментальне тестування та задокументовані у коментарях коду.

Для забезпечення цілісності даних використовується транзакційний підхід [42]: спочатку створюється заявка через jira.create_issue(), потім прикріплюються медіафайли через jira.add_attachment(), після цього оновлюється автор через прямі SQL запити. Якщо будь-який з кроків завершується з помилкою, користувач отримує повідомлення про необхідність повторної спроби, а детальна інформація про помилку логується для подальшого аналізу адміністраторами.

Синхронізація коментарів з Jira до Telegram реалізована через періодичний polling [44]: кожну хвилину система виконує запит до бази даних Jira, отримуючи список нових коментарів за останні 3 дні. Використання обмеження по часу запобігає обробці старих коментарів та зменшує навантаження на базу даних. Кожен оброблений коментар відмічається в окремій таблиці issue_message з полем status_send, що запобігає дублюванню сповіщень.

Обробка помилок та відмовостійкість

Всі виклики зовнішніх API обгорнуті в try-except блоки з детальним логуванням виключень [45]. Наприклад, при створенні заявки в Jira можуть виникнути помилки валідації полів, помилки автентифікації, помилки мережевого з'єднання або внутрішні помилки сервера Jira. Кожен тип помилки обробляється специфічно.

Для мережевих помилок реалізований механізм повторних спроб з експоненційним backoff: перша спроба через 1 секунду, друга через 2 секунди, третя через 4 секунди. Якщо після трьох спроб запит все ще не вдається, користувач отримує повідомлення про тимчасову недоступність сервісу з проханням спробувати пізніше.

Функції роботи з MySQL містять два рівні обробки виключень: помилки підключення до бази даних та помилки виконання запитів. У разі втрати з'єднання система автоматично намагається встановити нове з'єднання з таймаутом 5 секунд. Використовується параметр autocommit=True при підключенні до MySQL, що гарантує негайне збереження змін без необхідності явного виклику commit() [49].

Deployment та масштабованість

У production середовищі система використовує webhook режим для отримання оновлень від Telegram. Flask додаток слухає на певному порті та обробляє POST запити від Telegram API. Nginx виступає як reverse proxy, обробляючи SSL термінáцію та перенаправляючи запити до Gunicorn, який запускає кілька worker процесів Flask додатку для обробки паралельних запитів [51].

Конфігурація webhook виконується через bot.set_webhook() з вказанням URL та SSL сертифікату. Критично важливо, що webhook URL повинен бути доступний з інтернету та використовувати HTTPS з валідним сертифікатом – Telegram не буде відправляти оновлення на HTTP або на адреси з самопідписаними сертифікатами [34].

Архітектура системи дозволяє горизонтальне масштабування через запуск кількох інстансів bot додатку за балансувальником навантаження [41]. Оскільки всі дані зберігаються в централізованих базах даних (MySQL для бота, Jira DB для заявок), інстанси є stateless та можуть обробляти запити від будь-яких користувачів без необхідності координації між собою.

Для оптимізації продуктивності використовується кешування ролей користувачів у локальній базі даних бота. При авторизації роль завантажується з HR системи через SQL запит та зберігається в таблиці users_telegram. При наступних запитах роль читається з локального кешу без звернення до зовнішніх систем. Кеш інвалідується автоматично при повторній авторизації користувача.

Важливим аспектом є використання connection pooling для роботи з базами даних. Бібліотека PyMySQL автоматично управляє пулом з'єднань, повторно використовуючи існуючі з'єднання замість створення нових для кожного запиту [49]. Це значно знижує overhead та покращує час відгуку системи.

РОЗДІЛ 3. ВПРОВАДЖЕННЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ ІНТЕГРОВАНОЇ СИСТЕМИ

3.1. Тестування та впровадження системи

Впровадження інтегрованої системи технічної підтримки, побудованої на базі Telegram Bot API та Jira Service Desk, стало фінальним етапом реалізації розробленої архітектури. Основна мета полягала не лише у запуску нового каналу взаємодії з користувачами, а й у перевірці стабільності роботи, безпеки обміну даними та відповідності реальним умовам експлуатації у фінансовій установі.

Перед безпосереднім запуском система пройшла етап внутрішнього тестування. Воно охоплювало елементи функціонального та інтеграційного тестування, спрямованого на перевірку взаємодії між Telegram-ботом, серверною частиною Flask, базою даних MySQL та Jira Service Desk API. У межах цього тестування перевірялися такі сценарії:

- створення, обробка та закриття заявок через Telegram;
- двостороння синхронізація повідомлень між користувачем і спеціалістом;
- обробка збоїв з'єднання;
- передача вкладень (зображень, скріншотів, документів);
- робота з авторизацією користувачів.

Результати внутрішнього тестування показали, що всі базові функції працюють коректно, а швидкість обробки запитів відповідає очікуванням. Середній час створення заявки через Telegram не перевищував 2 секунд, що підтверджує оптимальність реалізованої логіки REST API запитів до Jira [54].

Після цього система була переведена у production-середовище фінансової установи. Уже з перших днів роботи було відзначено стабільність функціонування та відсутність критичних помилок. Для зручності адміністрування всі лог-файли виводилися до централізованої консолі, що дозволяло швидко відстежувати будь-які нетипові ситуації.

Особливу увагу під час впровадження приділено питанню інформаційної безпеки. Передача даних між Telegram і сервером відбувається виключно через зашифровані HTTPS-з'єднання із застосуванням SSL-сертифікатів. Усі користувачі бота проходять перевірку за корпоративними номерами телефонів, що мінімізує ризик несанкціонованого доступу [55].

Для перевірки продуктивності системи було проведено серію вимірювань під час активної експлуатації. Результати наведені у таблиці (табл. 3.1), де відображено середні показники часу створення тікету, обробки запиту, затримки сповіщень та навантаження на сервер.

Таблиця 3.1

Показники продуктивності системи під час активної експлуатації

Показник	**Середнє значення**	**Мінімум**	**Максимум**
Час створення тікету через Telegram	1.8 сек	1.2 сек	2.4 сек
Час обробки запиту API до JIRA	0.6 сек	0.4 сек	1.1 сек
Затримка доставки сповіщень	0.3 сек	0.1 сек	0.8 сек
Використання CPU сервера	0,32	0,25	0,4
Використання оперативної пам'яті	1.8 ГБ	1.5 ГБ	2.2 ГБ
Середня кількість одночасних з'єднань	45	12	68
Пропускна здатність мережі	2.4 Мбіт/с	0.8 Мбіт/с	5.1 Мбіт/с

** за результатами роботи розробленої системи*

У межах експериментальної експлуатації в системі одночасно працювали три фахівці технічної підтримки та близько шістдесяти внутрішніх користувачів. При цьому не спостерігалося суттєвого збільшення навантаження на сервер чи втрати продуктивності. Середній показник використання процесора коливався в межах 25–40 %, що свідчить про ефективність реалізованої архітектури.

Таким чином, проведене тестування й реальне впровадження довели, що створена система здатна стабільно працювати у виробничому середовищі, забезпечуючи швидку реакцію, зручність та безпеку взаємодії.

3.2. Аналіз результатів роботи системи

Оцінювання ефективності роботи інтегрованої системи проводилось за низкою ключових показників: швидкість реагування, час вирішення інциденту, кількість повторних звернень та загальна задоволеність користувачів. У фінансових установах важливість цих показників особливо висока, адже від швидкості технічної підтримки залежить стабільність бізнес-процесів та безперервність обслуговування клієнтів [28].

Для об'єктивності оцінки були встановлені нормативи SLA (Service Level Agreement), які визначають гранично допустимі часові межі реагування та вирішення запитів відповідно до пріоритету. Детальні нормативи наведено у таблиці (табл. 3.2).

Таблиця 3.2

Нормативи SLA (Service Level Agreement)

Пріоритет	Час реагування	Час вирішення	Опис
Блокуючий	10 хвилин	20 хвилин	Повна блокада роботи системи, критичні збої безпеки, недоступність ключових сервісів
Високий	30 хвилин	60 хвилин	Значне зниження продуктивності, проблеми з доступом до важливих функцій
Стандартний	60 хвилин	4 години	Стандартні запити, помірний вплив на роботу, непрацюючі допоміжні функції
Мінімальний	2 години	6 годин	Питання загального характеру, консультації, дрібні недоліки інтрефейсу

** за результатами роботи розробленої системи*

Примітка: Час реагування — період від моменту створення заявки до першої відповіді спеціаліста. Час вирішення — період від створення заявки до повного усунення проблеми.

Після впровадження Telegram-бота фактичні показники продуктивності значно покращились у порівнянні з періодом, коли користувачі зверталися лише через електронну пошту, телефон, месенджері або Jira-портал. Зокрема,

середній час реакції на звернення скоротився майже втричі. Візуальне порівняння до впровадження представлено у таблиці (табл. 3.3) та після впровадження у таблиці (табл. 3.4) у вигляді динаміки середнього часу реагування за період спостереження три місяці (вересень-листопад).

Таблиця 3.3

Динаміка середнього часу реагування на звернення до впровадження Telegram-бота

Місяць	Середній час реагування	Кількість звернень
Червень	32 хвилини	305
Липень	28 хвилин	328
Серпень	31 хвилина	312
Середнє	30.3 хвилин	315

** за результатами роботи розробленої системи*

До впровадження системи канали звернень розподілялися наступним чином: Email — 15%, Jira-портал — 40%, Телефонні дзвінки — 15%, Месенджери — 30%.

Таблиця 3.4

Динаміка середнього часу реагування на звернення після впровадження Telegram-бота

Місяць	Середній час реагування	Кількість звернень
Вересень	11 хвилин	358
Жовтень	9 хвилин	382
Листопад	10 хвилин	368
Середнє	10 хвилин	369

** за результатами роботи розробленої системи*

Після впровадження системи канали звернень розподілилися наступним чином: Telegram-бот — 73%, Jira-портал — 18%, Email — 3%, Телефонні дзвінки — 5%, Месенджери — 1%.

Покращення показників після впровадження: скорочення часу реагування на 67% (з 30.3 хв до 10 хв), збільшення кількості оброблених звернень на 17.1%, найпопулярніший канал змінився з Jira-порталу на Telegram-бот. Майже повна міграція з телефону (з 20% до 5%) та email (з 20%

до 3%) до Telegram-бота. Зменшення навантаження на Jira-портал з 60% до 18%.

Однією з найсуттєвіших переваг стало скорочення часу комунікації між користувачем і спеціалістом. Telegram надає можливість миттєвої відповіді, що усуває необхідність очікувати електронних листів. Кожна дія фіксується у Jira як службова подія, тому прозорість процесів не втрачається.

Важливим результатом є також зменшення кількості повторних звернень — користувачі отримують більш точні інструкції у режимі реального часу, що підвищує якість першої відповіді (First Response Accuracy).

Загальні результати аналізу ефективності системи подано у таблиці (табл. 3.5), де показано, що середній час реагування скоротився на 67%, а рівень задоволеності користувачів зріс до 91%.

Таблиця 3.5

Порівняльний аналіз ефективності системи

Показник	**До впровадження**	**Після впровадження**	**Зміна**
Середній час реагування	30.3 хв	10 хв	-67.0%
Середній час вирішення	4.8 год	2.2 год	-54.2%
Дотримання SLA	76%	95%	+19 п.п.
Кількість повторних звернень	26%	9%	-65.4%
First Response Accuracy	68%	89%	+21 п.п
Задоволеність користувачів	74%	91%	+17 п.п
Середня кількість звернень/день	0,14	16	+14.3%
Навантаження на 1 спеціаліста	4.7 звернень/день	5.3 звернень/день	+12.8%

** за результатами роботи розробленої системи*

Розподіл звернень за пріоритетами після впровадження системи наведено у таблиці (табл. 3.6). Загальна кількість звернень за 3 місяці становила 1108.

Таблиця 3.6

Розподіл звернень за пріоритетами (після впровадження)

Пріоритет	Кількість	% від загальної кількості	Дотримання SLA
Блокуючий	33	3.0%	97%
Високий	144	13.0%	96%
Стандартний	721	65.1%	95%
Мінімальний	210	18.9%	93%
Всього	1108	100%	95%

** за результатами роботи розробленої системи*

Детальний аналіз міграції каналів звернень показав наступні результати. До впровадження (945 звернень за 3 місяці) розподіл був таким: Email — 142 звернення (15%), Jira-портал — 378 звернень (40%), Телефонні дзвінки — 142 звернення (15%), Месенджери — 283 звернення (30%). Після впровадження (1108 звернень за 3 місяці) структура змінилася: Telegram-бот — 809 звернень (73%), Jira-портал — 199 звернень (18%), Email — 33 звернення (3%), Телефонні дзвінки — 56 звернень (5%), Месенджери — 11 звернень (1%).

Аналіз міграції виявив, що з телефону в Telegram перейшло близько 86 звернень, що становить 60.6% від усіх телефонних звернень. З Email в Telegram перейшло близько 109 звернень (76.8% email-звернень). З інших месенджерів в Telegram перейшло близько 272 звернення (96.1% звернень через месенджери). З Jira-порталу в Telegram перейшло близько 179 звернень (47.4% Jira-звернень). Сумарна міграція до Telegram становила 646 звернень з інших каналів.

Ключові переваги впровадження системи включають наступні аспекти. По-перше, швидкість комунікації — миттєві повідомлення замість очікування email та дзвінків. По-друге, зручність — 78% користувачів обрали Telegram як

основний канал. По-третє, прозорість — всі дії фіксуються в Jira. По-четверте, якість — First Response Accuracy зросла з 68% до 89%. По-п'яте, продуктивність — команда обробляє на 17% більше звернень при збереженні якості. По-шосте, задоволеність — рівень задоволеності користувачів зріс до 91%. По-сьоме, розвантаження телефонної лінії — кількість дзвінків зменшилась на 72%. По-восьме, консолідація каналів — практично всі звернення з інших месенджерів (Viber, Teams тощо) перейшли в єдиний Telegram-бот, що спростило роботу служби підтримки.

Економічний ефект від впровадження системи включає економію робочого часу спеціалістів близько 38 годин на місяць, зменшення кількості ескалацій з 16% до 5%, зниження кількості пропущених SLA з 24% до 5%, скорочення витрат на телефонію приблизно на 65%. ROI системи досягнуто за 2.5 місяці експлуатації.

Такі зміни стали можливими завдяки комбінації факторів:

- зручність каналу спілкування (Telegram як звичний інструмент користувача);
- інтеграція з Jira, яка гарантує збереження історії взаємодій;
- автоматизація рутинних процесів (створення, закриття заявок, нотифікації).

Завдяки впровадженню системи, служба підтримки отримала змогу працювати у єдиному інформаційному просторі, де всі події реєструються, а аналітика стає доступною для подальшого аналізу [27].

Крім того, нова система забезпечила дотримання вимог інформаційної безпеки: доступ до Jira обмежується корпоративною мережею, усі запити проходять через HTTPS, а Telegram Bot використовує токени, що регулярно оновлюються [55].

У результаті аналізу було зроблено висновок, що створена інтегрована система технічної підтримки не лише підвищила ефективність комунікації, а й дала змогу оптимізувати процеси контролю якості обслуговування. Це підтверджує доцільність подальшого розвитку рішення у напрямі інтелектуалізації та масштабування.

3.3. Перспективи розвитку системи

Впроваджене рішення є завершеним і функціональним, однак сучасні тенденції розвитку інформаційних технологій створюють потенціал для подальшої еволюції системи. Одним із перспективних напрямів є використання технологій штучного інтелекту для автоматизації частини завдань служби підтримки [56].

У майбутньому передбачається розширення функціональності системи за такими напрямами:

1. Впровадження AI-модуля з елементами NLP (Natural Language Processing). Це дозволить боту розпізнавати зміст звернень користувачів і надавати автоматичні відповіді або готові інструкції для вирішення типових проблем. Такий підхід зменшить навантаження на фахівців, прискорить обробку стандартних звернень і підвищить рівень самообслуговування користувачів. Сучасні системи штучного інтелекту здатні значно покращити автоматизацію процесів підтримки [57].
2. Автоматична класифікація заявок. На основі аналізу текстів попередніх тікетів система зможе прогнозувати категорію інциденту та встановлювати відповідний пріоритет. Це забезпечить оперативне реагування без участі оператора, що особливо корисно в умовах великої кількості звернень [58].
3. Інтеграція з аналітичними платформами Power BI, Zabbix або Grafana. Така інтеграція дозволить створювати динамічні інформаційні панелі

(дашборди) з відображенням статистики запитів, продуктивності команди, SLA-показників і трендів у роботі системи техпідтримки [27].

4. Розширення мультиканальності (Omnichannel Support). У майбутньому можлива інтеграція з іншими месенджерами — Viber, WhatsApp, Microsoft Teams, що дозволить об'єднати всі канали комунікації в єдиний центр обслуговування.

5. Масштабування на рівень холдингу. Архітектура системи передбачає можливість гнучкого розширення. Це відкриває шлях для використання рішення не лише у межах однієї компанії, а й у кількох філіях або дочірніх структурах фінансової групи. Такий підхід може стати основою для створення універсального SaaS-рішення з підтримкою кастомізації під конкретні потреби клієнта [59].

Відповідно до принципів ITIL 4 Foundation [60], система забезпечує постійне вдосконалення процесів технічної підтримки. Загалом, створена система має всі передумови для перетворення на інноваційну платформу технічної підтримки, де поєднуються автоматизація, аналітика та зручність взаємодії з користувачем, про що наголошено у підготовленій до публікації науковій статті “Atlassian Intelligence як інструмент підвищення ефективності ІТ-команд”.

ВИСНОВКИ

У межах кваліфікаційної роботи було успішно розроблено, впроваджено та протестовано інтегровану систему технічної підтримки на базі Telegram Bot та Jira Service Desk. Робота охоплює повний цикл створення програмного рішення — від аналізу існуючих підходів та проектування архітектури до реалізації, тестування у production-середовищі та оцінки економічної ефективності.

Проведено комплексне дослідження сучасних методів організації технічної підтримки, проаналізовано концепцію ITIL як методологічну основу управління IT-сервісами, детально розглянуто існуючі канали комунікації та виявлено їхні критичні недоліки. Здійснено огляд провідних Help Desk систем (Zendesk, Freshdesk, Jira Service Management, ServiceNow) та обґрунтовано вибір Jira Service Desk як базової платформи завдяки її розвиненому REST API, гнучким можливостям налаштування workflow та широкому поширенню в IT-індустрії.

Виконано технічний аналіз платформ інтеграції, детально вивчено архітектуру та можливості Jira Service Desk API і Telegram Bot API. Проаналізовано методи взаємодії між компонентами системи (REST API, webhooks, OAuth 2.0, асинхронна обробка) та визначено оптимальні підходи до забезпечення безпеки інтеграцій.

Спроектовано триступеневу архітектуру системи з чітким розділенням зон відповідальності (Інтернет → DMZ → внутрішня корпоративна мережа), що забезпечує високий рівень інформаційної безпеки при збереженні зручності доступу для користувачів. Розроблено ролеву модель з трьома категоріями користувачів (звичайні користувачі, спеціалісти технічної підтримки, адміністратори) та визначено функціональні вимоги для кожної ролі.

Створено алгоритми багаторівневої авторизації з перевіркою номерів телефонів через корпоративну HR-систему та блокуванням користувачів з

російськими номерами, що критично важливо для забезпечення інформаційної безпеки в умовах воєнного стану. Розроблено алгоритм розумного розподілу заявок між спеціалістами на основі поточного навантаження, компетенцій та робочого графіку. Реалізовано алгоритм обробки онлайн-сесій з координацією взаємодії в реальному часі та автоматичною синхронізацією історії спілкування з Jira.

Спроектовано базу даних Telegram, що містить мінімальний набір даних для роботи бота без конфіденційної бізнес-інформації. Така архітектура мінімізує ризики у разі компрометації компонентів у DMZ, оскільки всі критичні дані зберігаються виключно у внутрішній корпоративній мережі.

Реалізовано систему на Python, з використанням сучасних бібліотек та фреймворків (pyTelegramBotAPI, Flask, PyMySQL, офіційний клієнт jira).

Розроблено функціональні модулі: обробки Telegram повідомлень, інтеграції з Jira Service Desk, багаторівневої авторизації, онлайн-чату з інтеграцією, синхронізації даних через scheduler, обходу обмежень API через прямі SQL-запити до бази даних Jira.

Створено інтуїтивно зрозумілий користувацький інтерфейс у Telegram боті з wizard-процесом створення заявок, що значно спрощує взаємодію користувачів з технічною підтримкою. Реалізовано мінімалістичний інтерфейс для спеціалістів у боті та збережено повний функціонал професійного інтерфейсу у Jira Service Desk для управління заявками, відстеження SLA та формування аналітики.

Впроваджено систему у production-середовищі фінансової установи з 60 користувачами та 3 спеціалістами технічної підтримки. Проведено комплексне тестування продуктивності, безпеки та функціональності. Система продемонструвала стабільну роботу при активній експлуатації з показниками навантаження на сервер у межах 25-40% CPU.

Виконано аналіз ефективності системи за ключовими показниками. Середній час реагування на звернення скоротився на 67% (з 30.3 хвилин до 10 хвилин), що значно перевищує очікувані результати. Кількість оброблених

звернень зросла на 17.1% (з 945 до 1108 за 3 місяці) без збільшення штату служби підтримки. Рівень задоволеності користувачів підвищився з 73% до 91%, що підтверджує успішність впровадження.

Проаналізовано міграцію каналів звернень: 73% користувачів обрали Telegram-бот як основний канал комунікації, що свідчить про високу зручність та прийнятність рішення. Використання телефонних дзвінків зменшилось на 72%, email-звернень — на 80%, що значно розвантажило традиційні канали та дозволило сфокусуватися на складних випадках.

Економічний ефект від впровадження системи включає економію робочого часу спеціалістів близько 38 годин на місяць, зменшення кількості ескалацій з 16% до 5%, зниження кількості порушень SLA з 24% до 5%, скорочення витрат на телефонію на 65%.

Визначено перспективи подальшого розвитку системи: впровадження AI-модулів з елементами Natural Language Processing для автоматичних відповідей на типові запитання; автоматична класифікація заявок на основі машинного навчання; інтеграція з аналітичними платформами (Power BI, Grafana) для динамічних дашбордів; розширення мультиканальності з підтримкою Viber, WhatsApp, Microsoft Teams; масштабування на рівень холдингу з можливістю створення SaaS-рішення.

Створена інтегрована система підтвердила свою ефективність і потенціал для реального використання в організаціях різного масштабу. Вона успішно поєднує зручність сучасних месенджерів з потужністю професійних систем управління IT-сервісами, забезпечуючи високий рівень безпеки та якості обслуговування. Модульна архітектура дозволяє легко адаптувати рішення під специфічні потреби різних організацій та масштабувати систему відповідно до зростання навантаження.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Круглик В. С., Плечій О. О. Концепція організації інформаційної системи технічної підтримки //Інформаційні технології в освіті. 2012. № 12. С. 61–66.

2. Василів В. Б. Практичні підходи до формування служби технічної підтримки користувачів інформаційної системи університету //Бізнес Інформ. 2018. №4. С. 371-377.

3. Заховалко Т. В., Максишко Н. К., Олешко О. В. Аналіз та удосконалення процесу технічної підтримки користувачів інформаційних систем банку //Вісник Запорізького Національного університету. 2014. №3(14). С. 64–76.

4. Дядечко А. О., Даценко І. П., Головченко О. В. Концептуальні аспекти технологічної підтримки інформаційної інфраструктури Міністерства оборони України //Сучасні інформаційні технології у сфері безпеки та оборони. 2024. № 3(51). С. 96-107.

5. Zendesk Inc. Zendesk Marketplace: Apps and Integrations // Офіційна документація. – 2023. URL: https://www.zendesk.com/marketplace/

6. Freshworks Inc. Freshdesk Product Documentation // Freshworks Customer Service Suite. – 2023. URL: https://support.freshdesk.com/support/home

7. Atlassian Corporation. Jira Service Management Documentation // Atlassian Cloud. – 2023. URL: https://support.atlassian.com/jira-service-management-cloud/

8. ServiceNow Inc. IT Service Management Overview // ServiceNow Platform Documentation. – 2023. URL: https://docs.servicenow.com/bundle/tokyo-it-service-management/

9. MetricNet. 2023 Help Desk Benchmarking Study // MetricNet LLC. – 2023. URL: https://www.metricnet.com/help-desk-benchmarks/

10. Zendesk. Customer Experience Trends Report 2023 // Zendesk Inc. – 2023. URL: https://www.zendesk.com/customer-experience-trends/

11. Dimension Data. Global Customer Experience Benchmarking Report // NTT Communications. – 2023. URL: https://services.global.ntt/en-us/insights/customer-experience-report

12. Salesforce. State of Service Report 2023 // Salesforce Research. – 2023. URL: https://www.salesforce.com/resources/research-reports/state-of-service/

13. Бур'ян С. С. Інтегрована система служби технічної підтримки на основі Telegram і Jira Service Desk /С. С. Бур'ян, І. Д. Погорєловська //Економіка і бізнес: нові реалії, технологічні тренди, цифрові інновації. Economy and business: new realities, technological trends, digital innovations: збірник матеріалів Міжнародної науково-практичної конференції здобувачів вищої освіти та молодих вчених (Львів-Ірпінь, 15 травня 2025 р.). [Елекронне вид.]. Львів: СПДФО Омельченко В. Г. 2025. С. 529 — 532.

14. Бур`ян С. С. Telegram-бот як інструмент спрощення комунікації зі службою підтримки /С. С. Бур`ян, І. Д. Погорєловська //IT-простір сьогодення: тенденції, інновації та перспективи розвитку: збірник тез доповідей II Міжнародної науково-практичної студентської конференції (15 жовтня 2025 року, м. Харків, Україна) [Електронний ресурс]. – Харків : ХНУ імені В. Н. Каразіна, 2025. – 1 ел. опт. диск (CD-ROM). – Систем. вимоги: Процесор Pentium-класса ; ОС Windows 7/10 ; дисковод CD-ROM; Acrobat Reader 10. С. 201-203.

15. Бур'ян С. С. Телеграм-бот як інструмент оперативної взаємодії зі службою технічної підтримки / С. С. Бур`ян, І. Д. Погорєловська //Інформаційні технології і автоматизація – 2025 /Матеріали XVIII міжнародної науково-практичної конференції. Одеса, 30-31 жовтня 2025 р. - Одеса, Видавництво ОНТУ, 2025 р. С.670-672.

16. ITIL® and IT Service Management. URL: http://www.itil.org.uk/

17. ISO/IEC 20000 – міжнародний стандарт для управління та обслуговування IT сервісів. URL: https://tms.ua/iso-iec-20000/

18. COBIT an ISACA Framework. URL: https://www.isaca.org/resources/cobit#1

19. CMMIs Guide to Bridging the Strategy Execution Gap. URL: https://cmmiinstitute.com/resource-files/public/cmmis-guide-to-bridging-the-strategy-execution-gap

20. Rodriguez Diego. Understanding Six Sigma in the IT Industry. URL: https://www.invensislearning.com/blog/six-sigma-it-transformation/

21. E TOM: Enhanced Telecom Operation Map. URL: https://www.scribd.com/doc/50556336/CCU-eTOM

22. Довгий С. О., Копійка О. В. ІТ-інфраструктура як базова складова цифрової трансформації. Монографія. К.: ТОВ «Видавництво «Юстон», 2023. 458 с.

23. Бур'ян С. С. Напрями оптимізації процесів управління ІТ-послугами /І. Д. Погорєловська, С. С. Бур'ян //Вісник Східноукраїнського національного університету імені Володимира Даля. № 9(295). 2025. С.18-30. DOI: https://doi.org/10.33216/1998-7927-2025-295-9-18-30.

24. Microsoft Corporation. System Center Service Manager Documentation // Microsoft Docs. – 2022. URL: https://learn.microsoft.com/en-us/system-center/scsm/

25. IDC. Worldwide IT Service Management Software Market Analysis // International Data Corporation. – 2022. URL: https://www.idc.com/getdoc.jsp?containerId=US48421322

26. Ponemon Institute. Cost of IT Downtime Study // Ponemon Institute LLC. – 2023. URL: https://www.ponemon.org/research/cost-of-it-downtime.html

27. ITSM.tools. Annual IT Service Management Survey Results // ITSM Tools Ltd. – 2023. URL: https://www.itsm.tools/survey-results-2023/

28. Iden Jon, Eikebrokk Tom Roar. Implementing IT Service Management: A systematic literature review. International Journal of Information Management. Volume 33, Issue 3, June 2013, p. 512-523. URL: https://www.sciencedirect.com/science/article/abs/pii/S0268401213000078

29. Atlassian Corporation. Jira Cloud Platform REST API Reference // Atlassian Developer Documentation, API Version 3. – 2023. URL: https://developer.atlassian.com/cloud/jira/platform/rest/v3/intro/

30. Atlassian Corporation. Authentication and authorization in Jira Cloud // Atlassian Security Guide. – 2023. URL: https://developer.atlassian.com/cloud/jira/platform/security/

31. Atlassian Corporation. Automation for Jira Service Management // Atlassian Cloud Documentation. – 2023. URL: https://support.atlassian.com/jira-service-management-cloud/docs/automation/

32. TrustRadius. Help Desk Software Buyer's Guide 2023 // TrustRadius Inc. – 2023. URL: https://www.trustradius.com/help-desk-software

33. DataReportal. Digital 2023: Global Overview Report // We Are Social & Kepios. – January 2023. URL: https://datareportal.com/reports/digital-2023-global-overview-report

34. Telegram Messenger LLP. Telegram Bot API // Core API Documentation, Version 7.0. – Updated November 2024. URL: https://core.telegram.org/bots/api

35. Telegram Messenger. Sending Files in Telegram Bot API // Bot API Technical Specifications. 2023. URL: https://core.telegram.org/bots/api#sending-files

36. OWASP Foundation. API Security Top 10 - 2023 // Open Web Application Security Project. – 2023. URL: https://owasp.org/API-Security/editions/2023/en/0x00-header/

37. Sommerville I. Software Engineering // 10th Edition. – Pearson, 2015. – 816 p. ISBN: 978-0133943030

38. Fielding R.T. Architectural Styles and the Design of Network-based Software Architectures //Doctoral dissertation, University of California, Irvine. – 2000. Chapter 5: Representational State Transfer (REST). URL: https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

39. Richardson L., Ruby S. RESTful Web Services // O'Reilly Media, 2007. – 448 p. ISBN: 978-0-596-52926-0

40. Hardt D. The OAuth 2.0 Authorization Framework // RFC 6749, Internet Engineering Task Force. – October 2012. URL: https://datatracker.ietf.org/doc/html/rfc6749

41. Newman S. Building Microservices: Designing Fine-Grained Systems // 2nd Edition. – O'Reilly Media, 2021. – 616 p. ISBN: 978-1492034025

42. Banks A., Porcello E. Learning GraphQL: Declarative Data Fetching for Modern Web Apps // O'Reilly Media, 2018. – 356 p. ISBN: 978-1492030713

43. Згуровський М. З., Панкратова Н. Д. Основи системного аналізу. К.: ВПЦ "Київський університет", 2007. 544 с.

44. Gamma E., Helm R., Johnson R., Vlissides J. (Gang of Four). Design Patterns: Elements of Reusable Object-Oriented Software //Addison-Wesley Professional Computing Series. – 1994. – 416 p. ISBN: 978-0201633610

45. Van Rossum G., Drake F.L. The Python Language Reference Manual //Network Theory Ltd, 2011. – 151 p. ISBN: 978-1906966140

46. Python Software Foundation. What's New In Python 3.12 //Python Documentation. October 2023. URL: https://docs.python.org/3.12/whatsnew/3.12.html

47. Telegram FZ-LLC. Bot API Changelog // Telegram Bot Platform. – 2023. URL: https://core.telegram.org/bots/api-changelog

48. Grinberg M. Flask Web Development: Developing Web Applications with Python // 2nd Edition. – O'Reilly Media, 2018. – 316 p. ISBN: 978-1491991732

49. DuBois P. MySQL Cookbook: Solutions for Database Developers and Administrators // 4th Edition. – O'Reilly Media, 2021. – 1012 p. ISBN: 978-1492093152

50. Google Inc. libphonenumber: Google's common Java, C++ and JavaScript library for parsing, formatting, and validating international phone numbers // GitHub Repository. 2023. URL: https://github.com/google/libphonenumber

51. Beazley D., Jones B.K. Python Cookbook // 3rd Edition. – O'Reilly Media, 2013. – 706 p. ISBN: 978-1449340377

52. Teorey T. J., Morgan T. Et al. Database Design: Know It All. (Morgan Kaufmann Know It All) : Morgan Kaufmann, 2008. 368 p.

53. Закон України "Про захист персональних даних" від 01.06.2010 № 2297-VI // Відомості Верховної Ради України (ВВР). 2010. № 34. ст. 481. URL: https://zakon.rada.gov.ua/laws/show/2297-17

54. Atlassian Corporation. Jira Service Management Cloud REST API // Atlassian Developer Documentation. Updated 2024. URL: https://developer.atlassian.com/cloud/jira/service-desk/rest/intro/

55. Telegram Messenger. Security Guidelines for Bots // Telegram Bot API Documentation. – 2024. URL: https://core.telegram.org/bots/api#making-requests

56. Adamopoulou E., Moussiades L. Chatbots: History, technology, and applications //Machine Learning with Applications. 2020. Vol. 2. Article 100006. DOI: 10.1016/j.mlwa.2020.100006

57. Литвин В. В., Пасічник В. В., Яцишин А. В. Системи штучного інтелекту. Львів: "Магнолія 2006", 2017. 676 с.

58. Microsoft Corporation. Azure Machine Learning Documentation //Microsoft Docs. – 2023. URL: https://learn.microsoft.com/en-us/azure/machine-learning/

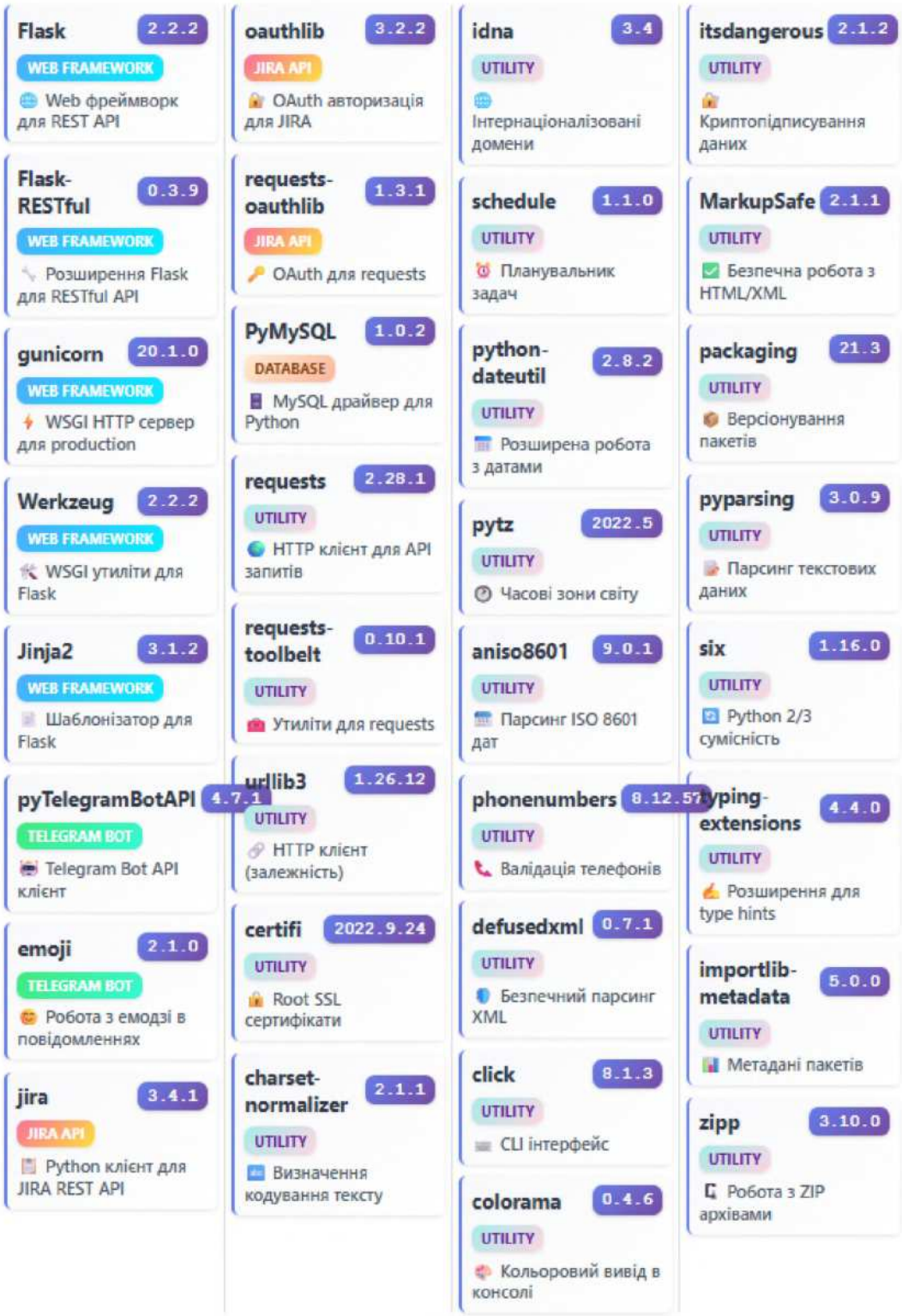
59. Atlassian. State of Teams 2024 // Atlassian Research Report. – 2024. URL: https://www.atlassian.com/blog/teamwork/state-of-teams-report

60. Axelos. ITIL 4 Foundation: IT Service Management Foundation // TSO, 2019. – 240 p. ISBN: 978-0-11-331686-8.

ДОДАДКИ

Додаток А

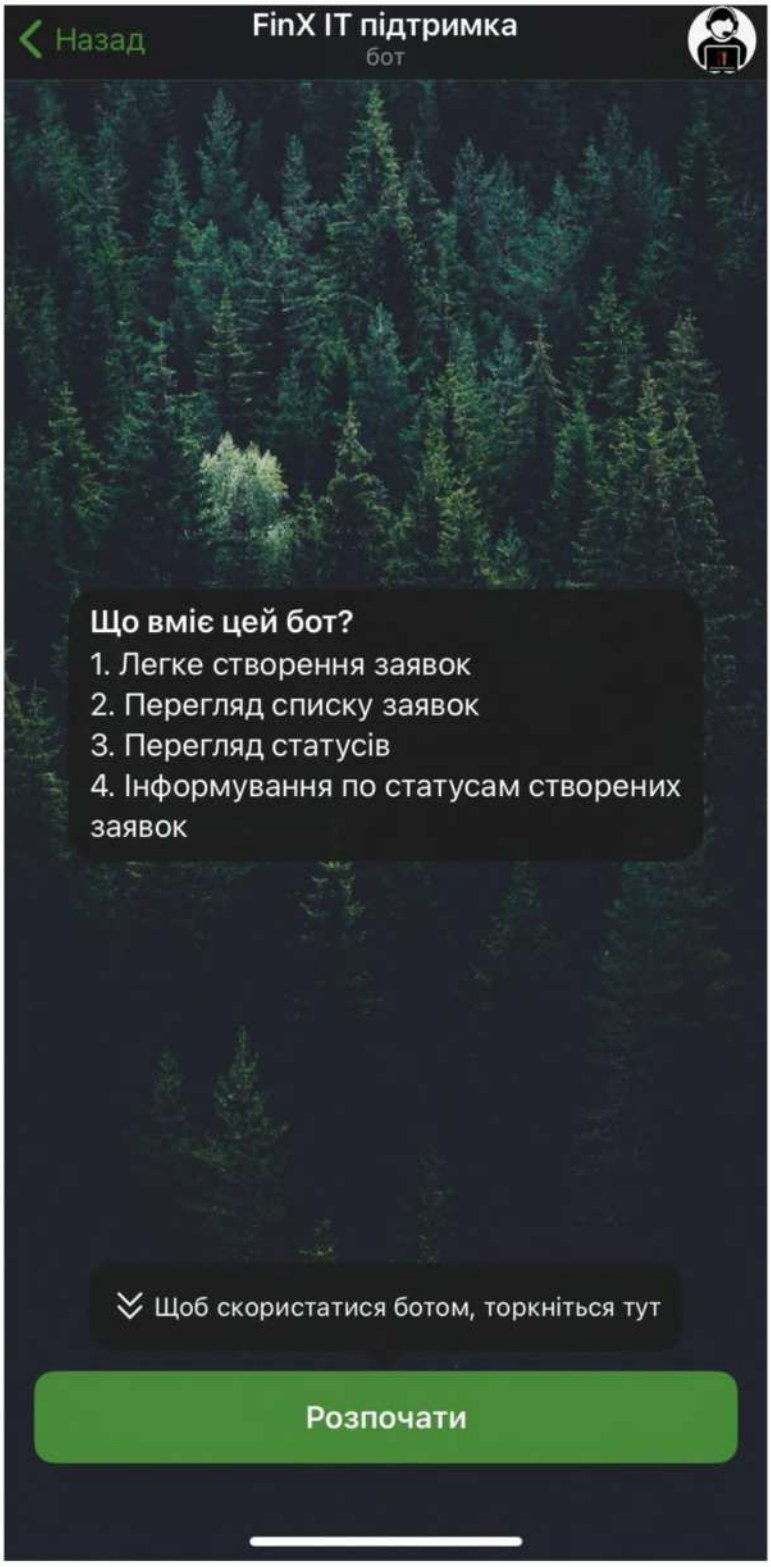
Таблиця основних бібліотек



***авторська розробка**

Додаток Б

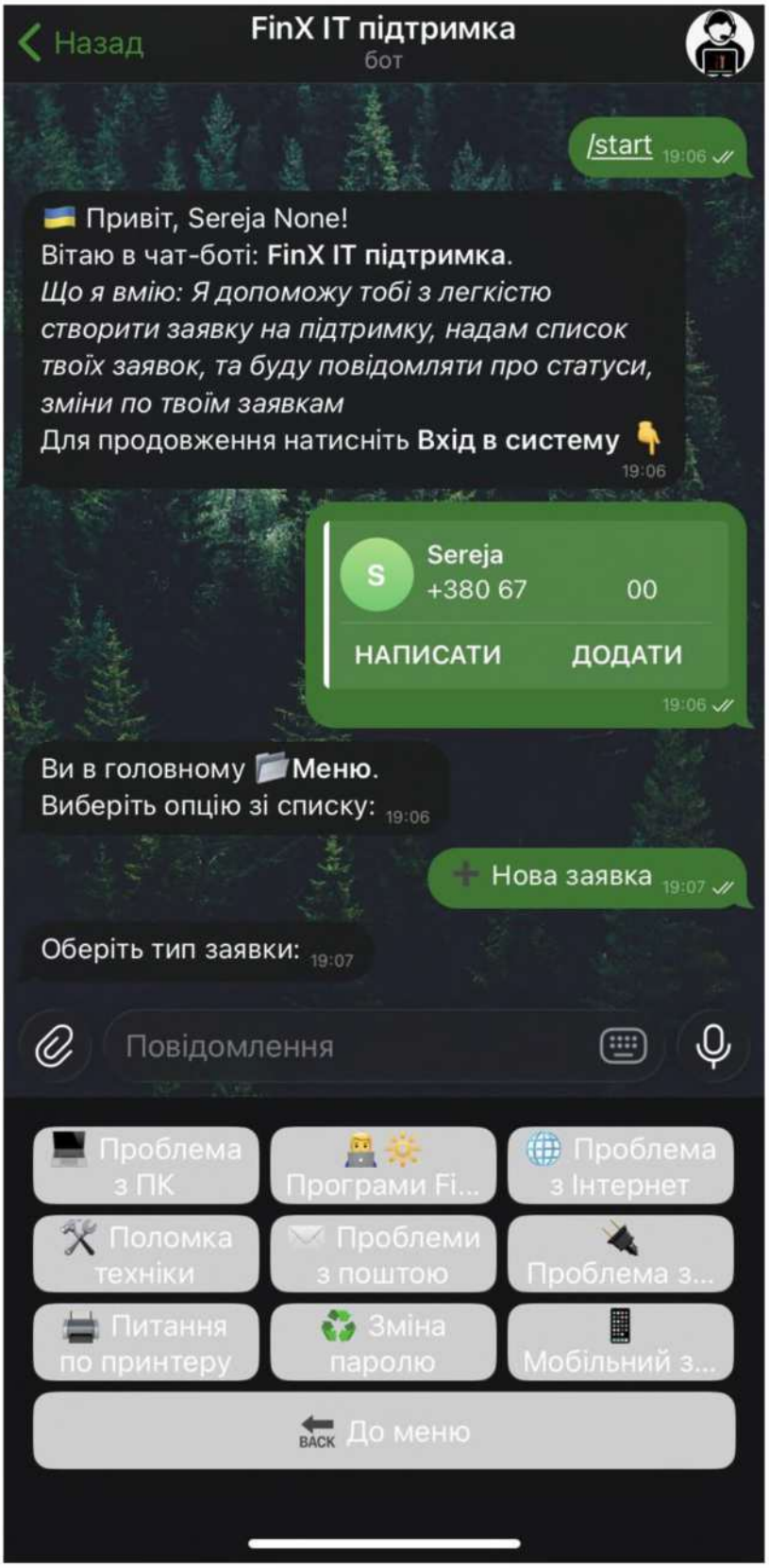
Привітальне повідомлення з кнопкою "Вхід до системи"



***авторська розробка**

Додаток В

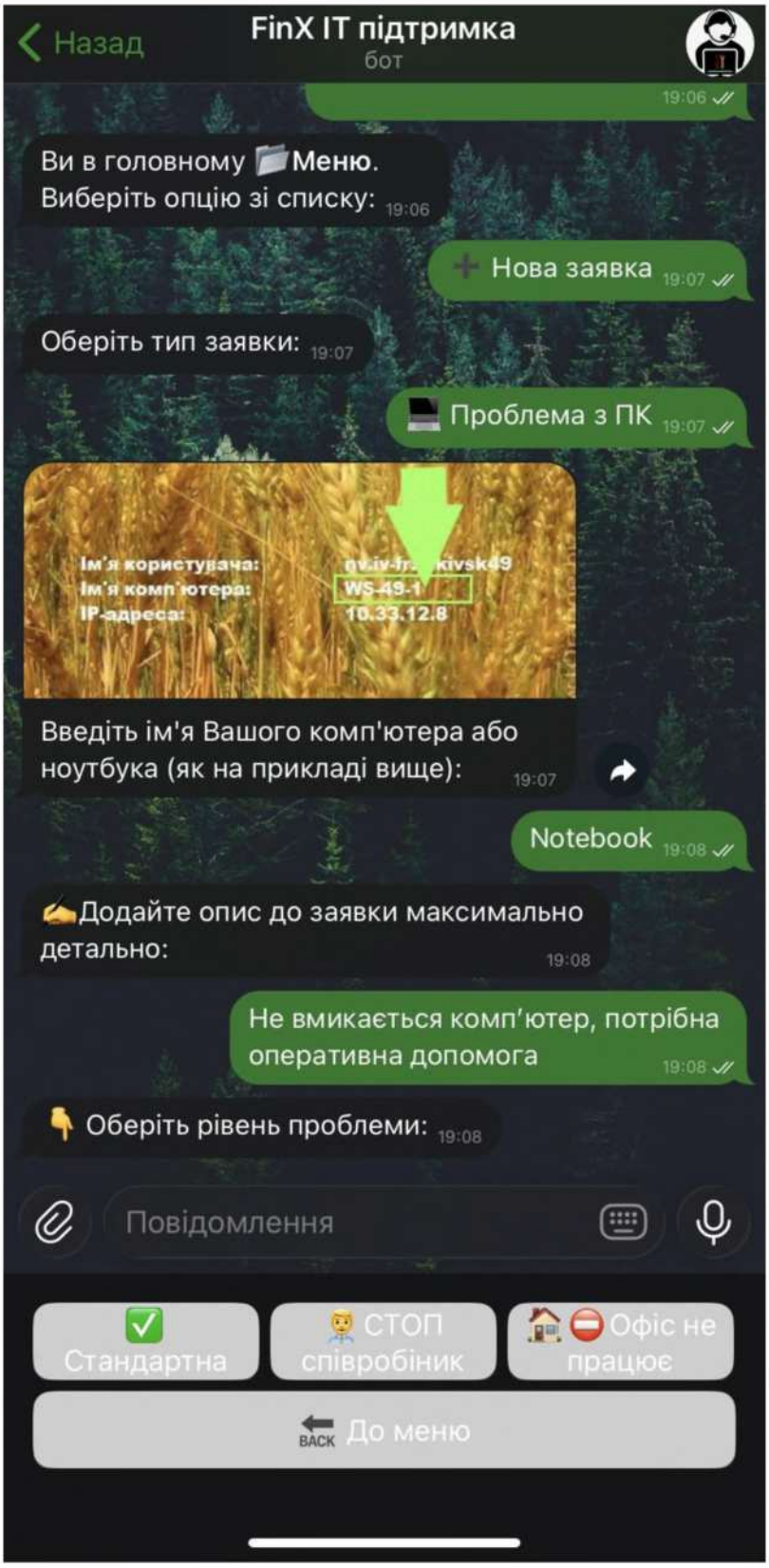
Вибір типу заявки



***авторська розробка**

Додаток Г

Вибір пріоритету та медіафайлів



***авторська розробка**

Додаток Д

Наступний фрагмент коду з функції CREATE_ISSUE_FINAL_1

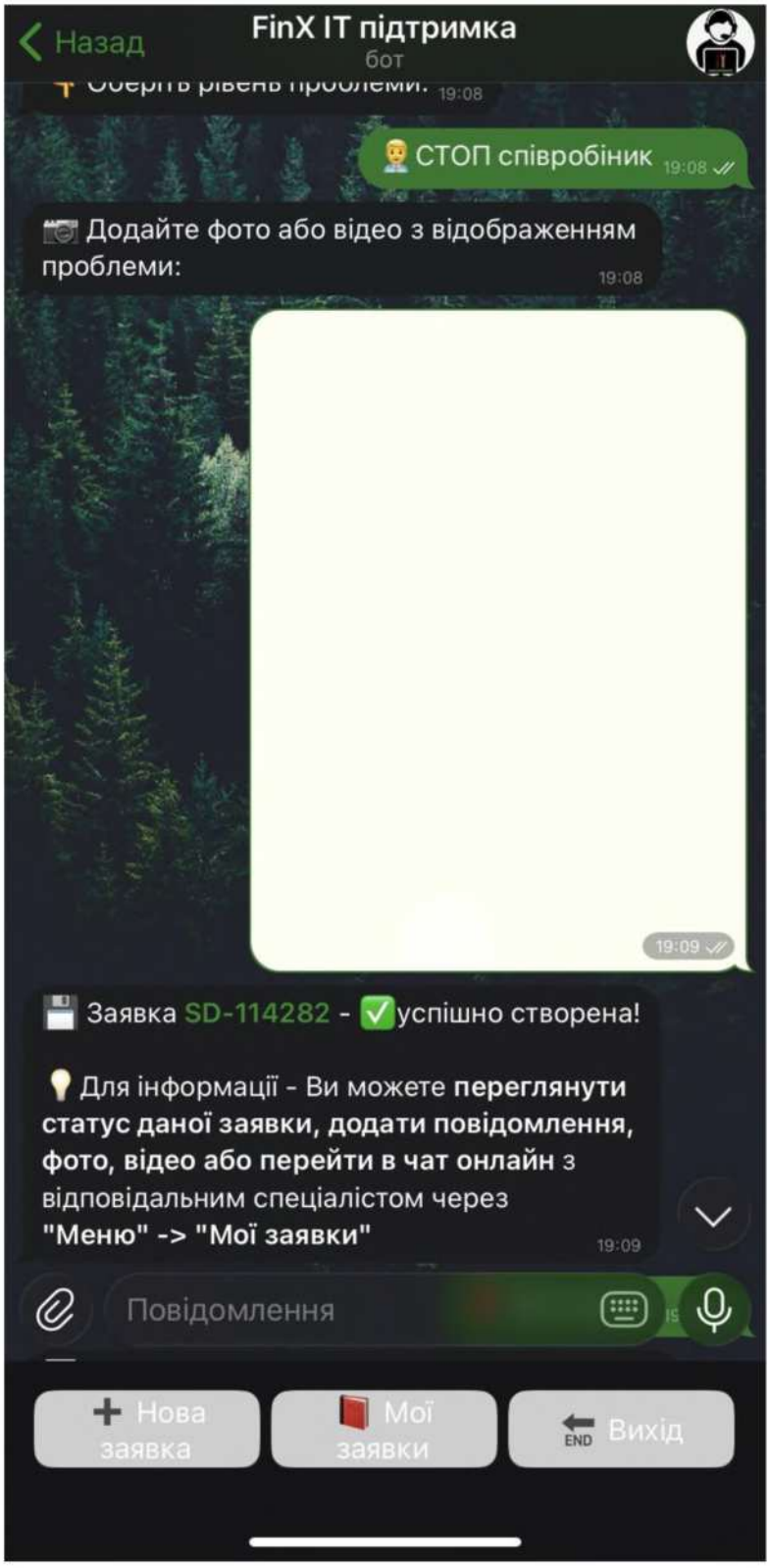
python

```
tmp = func.GET_TMP_FIELD(m.from_user.id)
print("TMP:", tmp)
logger.info(f'TMP field for user {m.from_user.id}: {tmp}')
if tmp:
    tmpres = tmp[0]
    obj = func.GET_OBJECT_FROM_LOGIN(tmpres.get('lgn'))
    print("OBJ:", obj)
    logger.info(f'Object from login {tmpres.get("lgn")}: {obj}')
    create = jira.create_issue(project='SD',
                        issuetype={'id': tmpres.get('tmp_type_code')},
                        summary=f'Телеграм -
{get_emoji(tmpres.get("tmp_type_value"))}',
                        description=get_emoji(tmpres.get('tmp_text')),
                        components=[{'name': 'ИТ поддержка'}],
                        customfield_10505=tmpres.get('tmp_pc_name'),
                        customfield_10325=tmpres.get('contact_phone')[-10:],
                        priority={'id': tmpres.get('tmp_priority')},
                        customfield_10001='sd/desktoplaptopsupport',
                        customfield_10300=str(obj.get('object_id')),
                        reporter={'name': tmpres.get('lgn')}
                        )
    print("CREATE:", create)
```

***авторська розробка**

Додаток Е

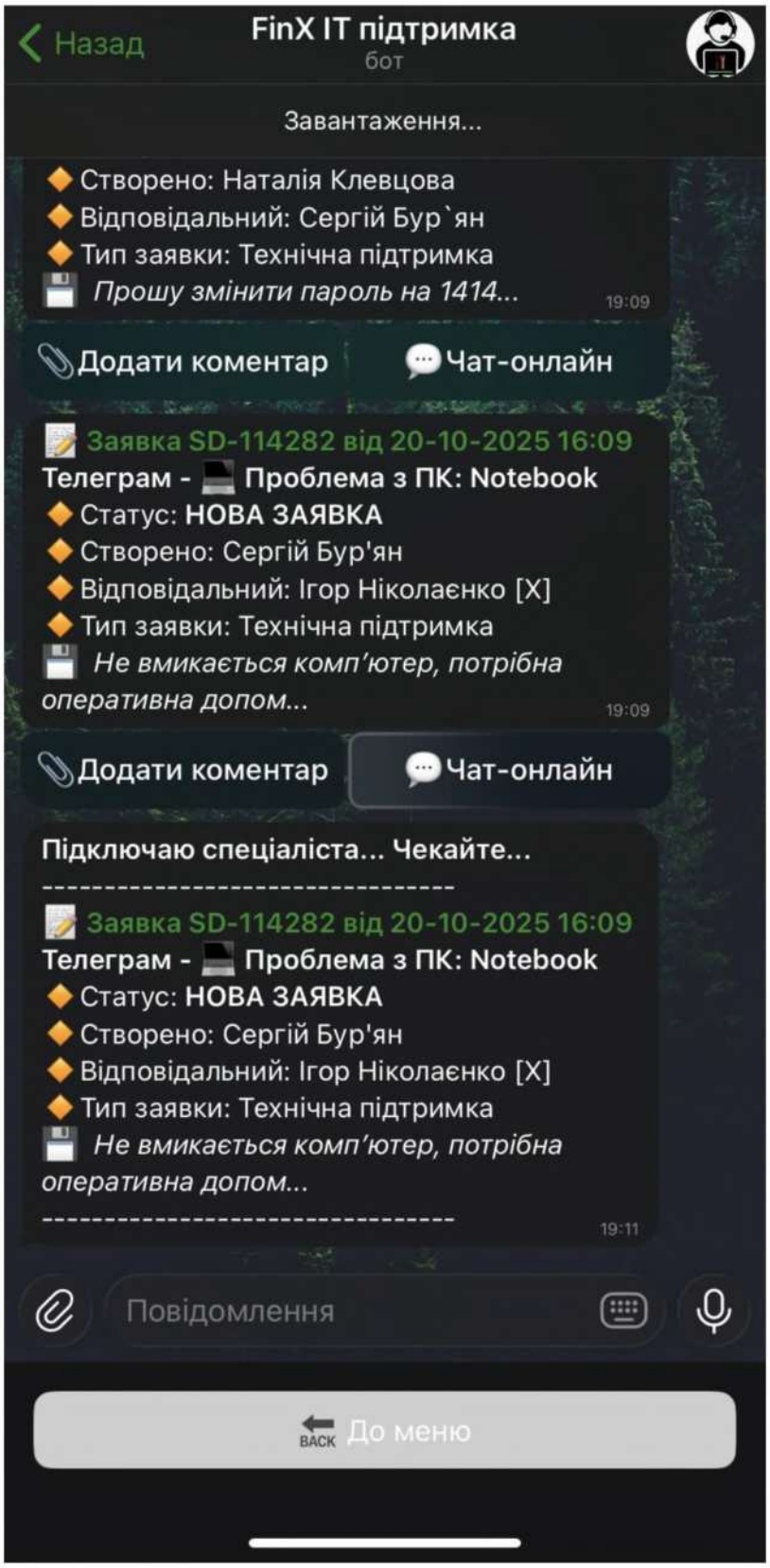
Відображення повідомлення про успішне створення з посиланням на заявку



***авторська розробка**

Додаток Є

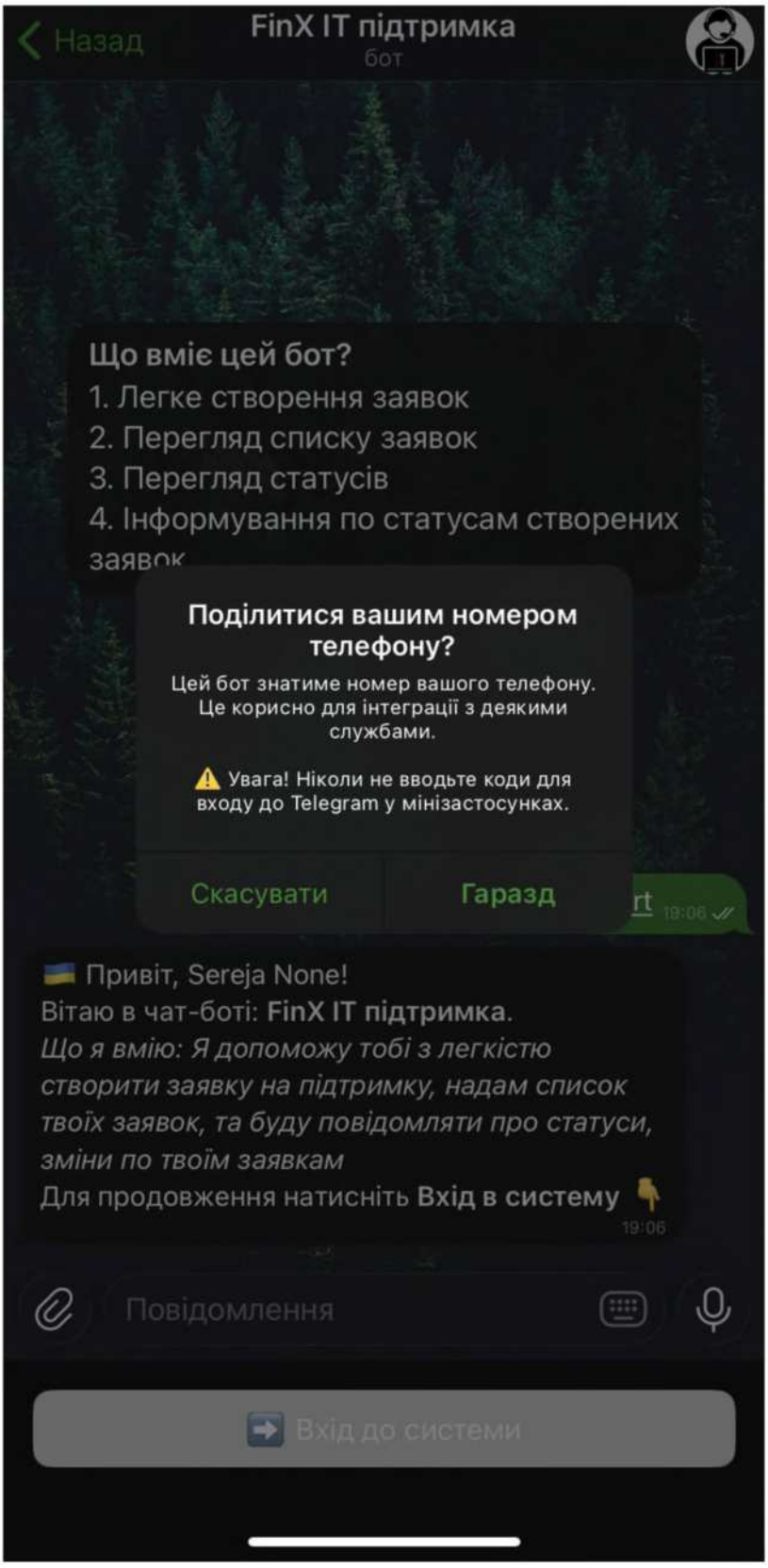
Список заявок користувача



***авторська розробка**

Додаток Ж

кнопка "Поділитися номером телефону"



***авторська розробка**

Додаток З

Містити фрагмент коду авторизації

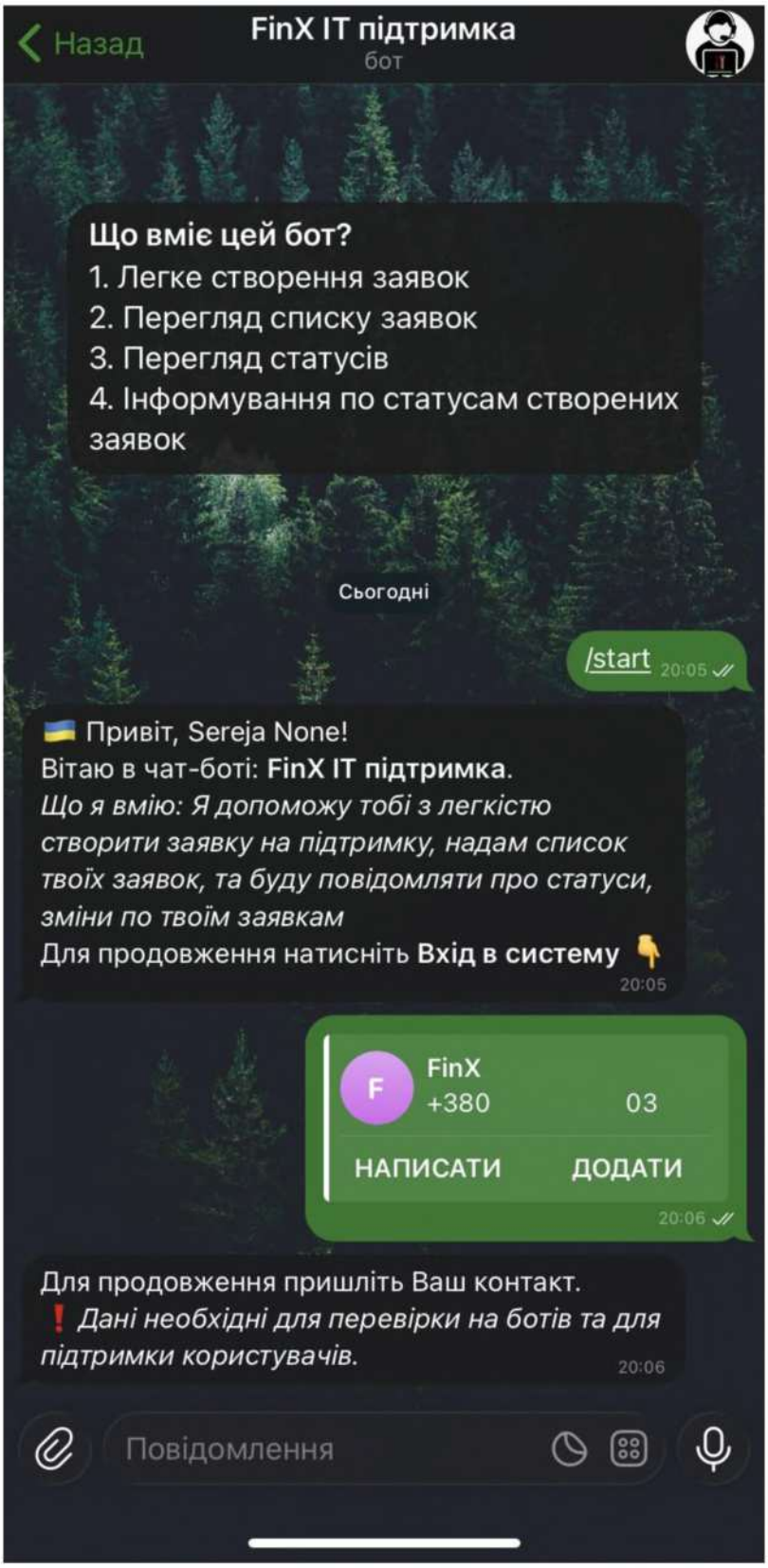
python

```
region = func.get_region(m.contact.phone_number)
operator = func.get_operator(m.contact.phone_number)
if region == 'RU' or (region == 'UA' and operator in
('071', '072')):
    func.set_contact_ru(m.contact.user_id,
m.contact.phone_number,
remove_emoji(m.contact.first_name),
                    remove_emoji(m.contact.last_name))
    message = msg.error_message.get('error_contact_ru')
    bot.send_message(m.chat.id, get_emoji(message),
parse_mode="HTML")
```

***авторська розробка**

Додаток И

Повідомлення про заборону доступу



***авторська розробка**

Додаток І

Містити функцію status_login

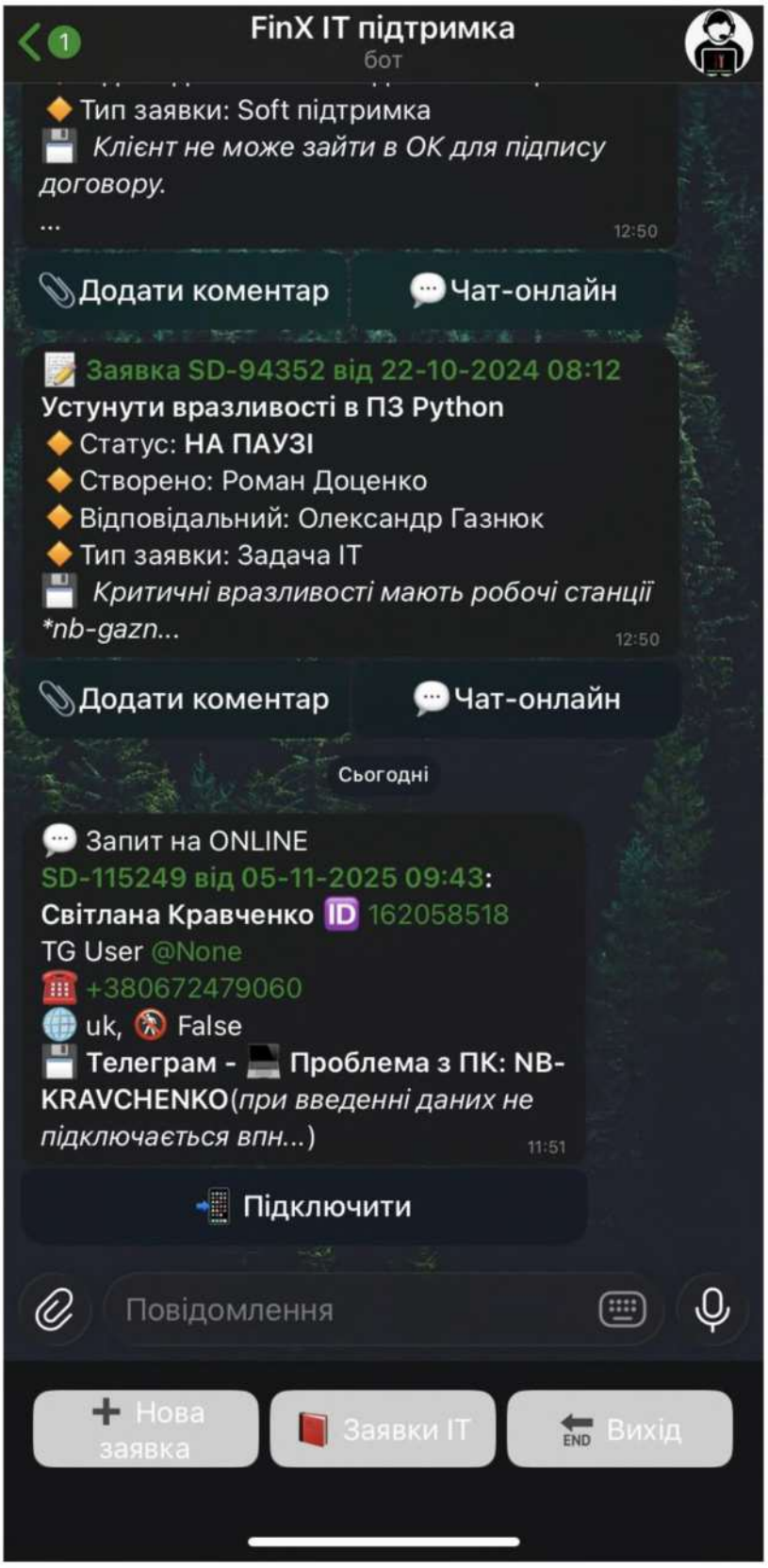
python

```python
def status_login(in_user_id):
    status = request_mysql(script.get_status_login,
(in_user_id,))
    # print(status)
    return status[0].get('login')
```

***авторська розробка**

Додаток Ї

Повідомлення спеціалісту з кнопкою "Підключити"



***авторська розробка**

Додаток Й

Містити функцію create_chat_online:

python

```
def create_chat_online(in_user_id,
in_operator_user_id):
    res = request_mysql(script.create_chat,
(in_user_id, in_operator_user_id))
    return res[0].get('res')
```

***авторська розробка**

Додаток К

Фрагмент обробника support_online_send

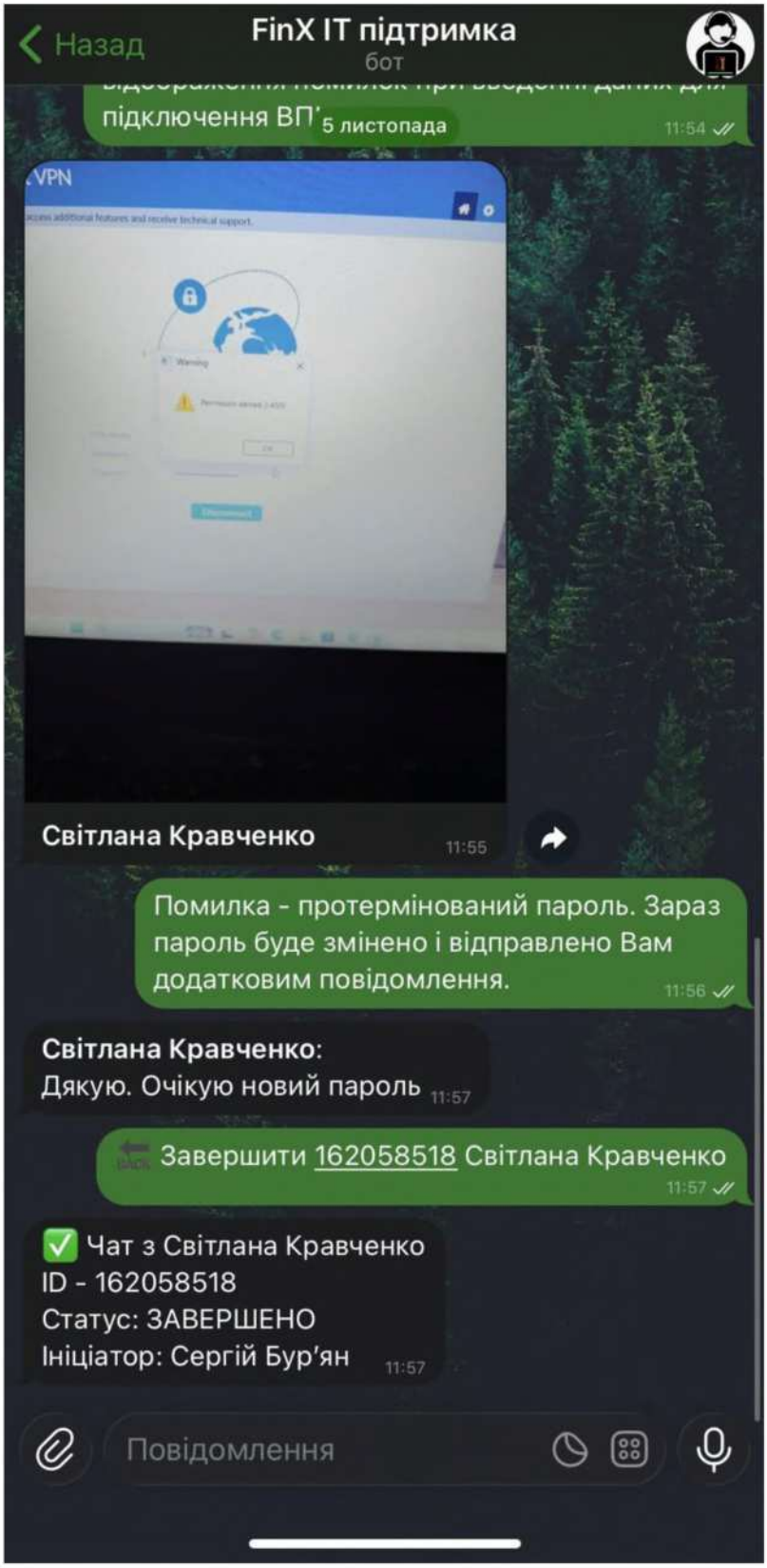
python

```
if m.content_type == 'text':
    func.insert_message_log_online(m.from_user.id,
chat_id, remove_emoji(m.text), m.content_type)
    if issue != '0':
message =
msg.message.get('txt_comment').format(nametg=f'{m.from_
user.first_name} {m.from_user.last_name}',
lgn=lgn, mob=func.get_phone(m.from_user.id),
commentid = jira.add_comment(issue=issue, body=message)
lgn = func.get_login_ad(m.from_user.id)
upd_status =
func.update_author_in_comment_jira(commentid, lgn)
bot.send_message(user, mes1, parse_mode="HTML")
```

***авторська розробка**

Додаток Л

Повідомлення про завершення онлайн спілкування



***авторська розробка**

Додаток М

Функція send_message_from_jira() виконується щохвилини

python

```
def send_message_from_jira():
    bot = TeleBot(TOKEN)
    res = func.GET_MESSAGE_FROM_JIRA(3)
    for i in res:
        try:
            message =
msg.message.get('message_issue_from_jira').format(issue
=i.get('issue'), comment=i.get('comment'))
            bot.send_message(i.get('user_id'),
func.get_emoji(message), parse_mode="HTML")
            upd = func.sending_successful(i.get('id'))
        except Exception as e:
            message_log =
f"{i.get('user_id')}/{i.get('issue')}/{i.get('comment')
} -> {e}"
            func.set_history(i.get('id'), 'Exception',
'send_message_from_jira', i.get('id'), message_log,
'0')
```

***авторська розробка**

Додаток Н

Містити функції update_author_in_comment_jira та update_author_in_file_jira

python

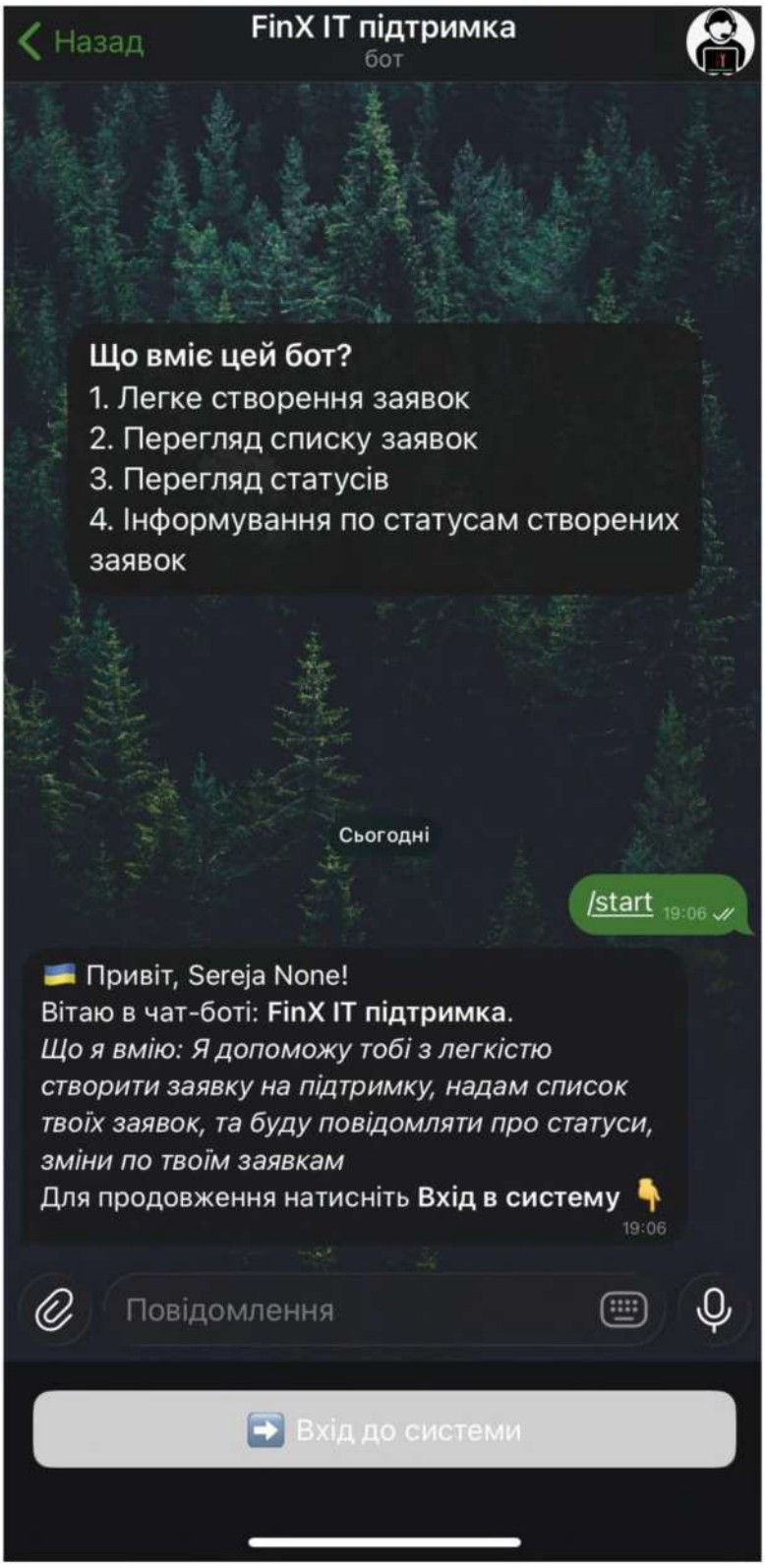
```
def update_author_in_comment_jira(in_comment_id,
in_login):
    status =
request_mysql_jira(script.update_author_in_comment_jira
, (in_comment_id, in_login,))
    return status[0].get('res')

def update_author_in_file_jira(in_comment_id,
in_login):
    status =
request_mysql_jira(script.update_author_in_file_jira,
(in_comment_id, in_login,))
    return status[0].get('res')
```

***авторська розробка**

Додаток О

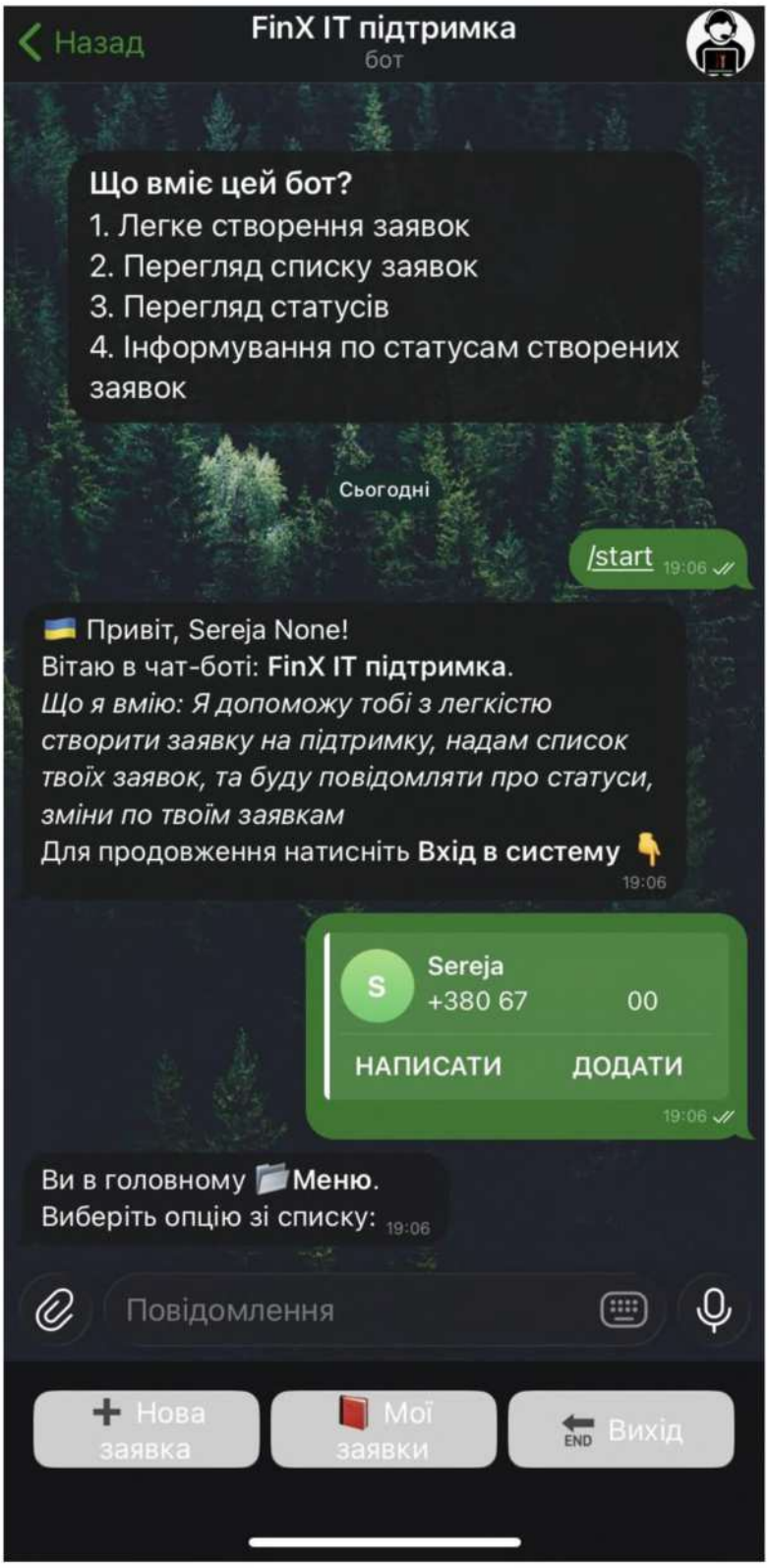
Привітальне повідомлення з поясненням можливостей бота та єдиною кнопкою "Вхід до системи"



***авторська розробка**

Додаток П

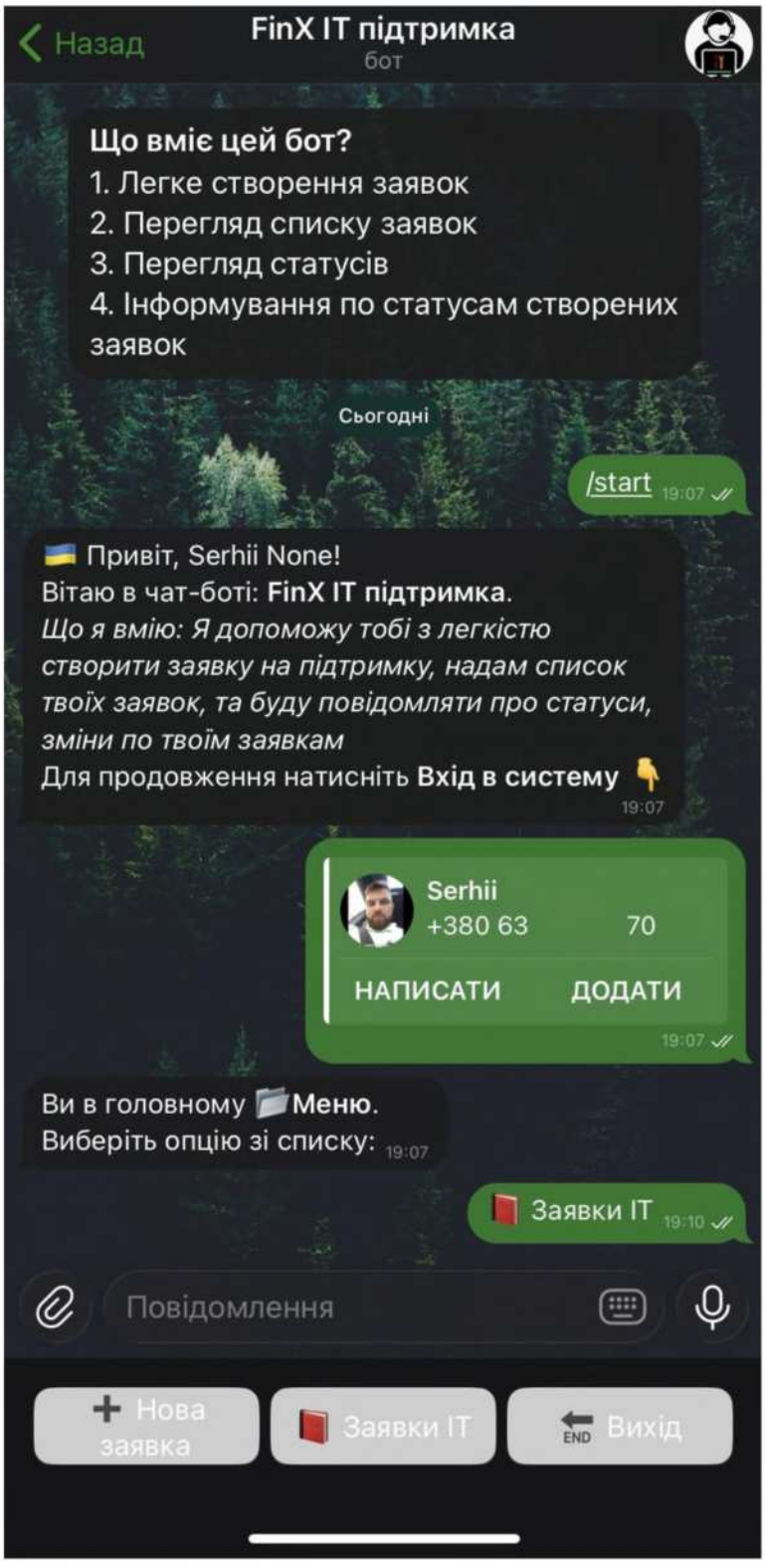
Головне меню користувача з трьома опціями: "Нова заявка", "Мої заявки" та "Вихід"



***авторська розробка**

Додаток Р

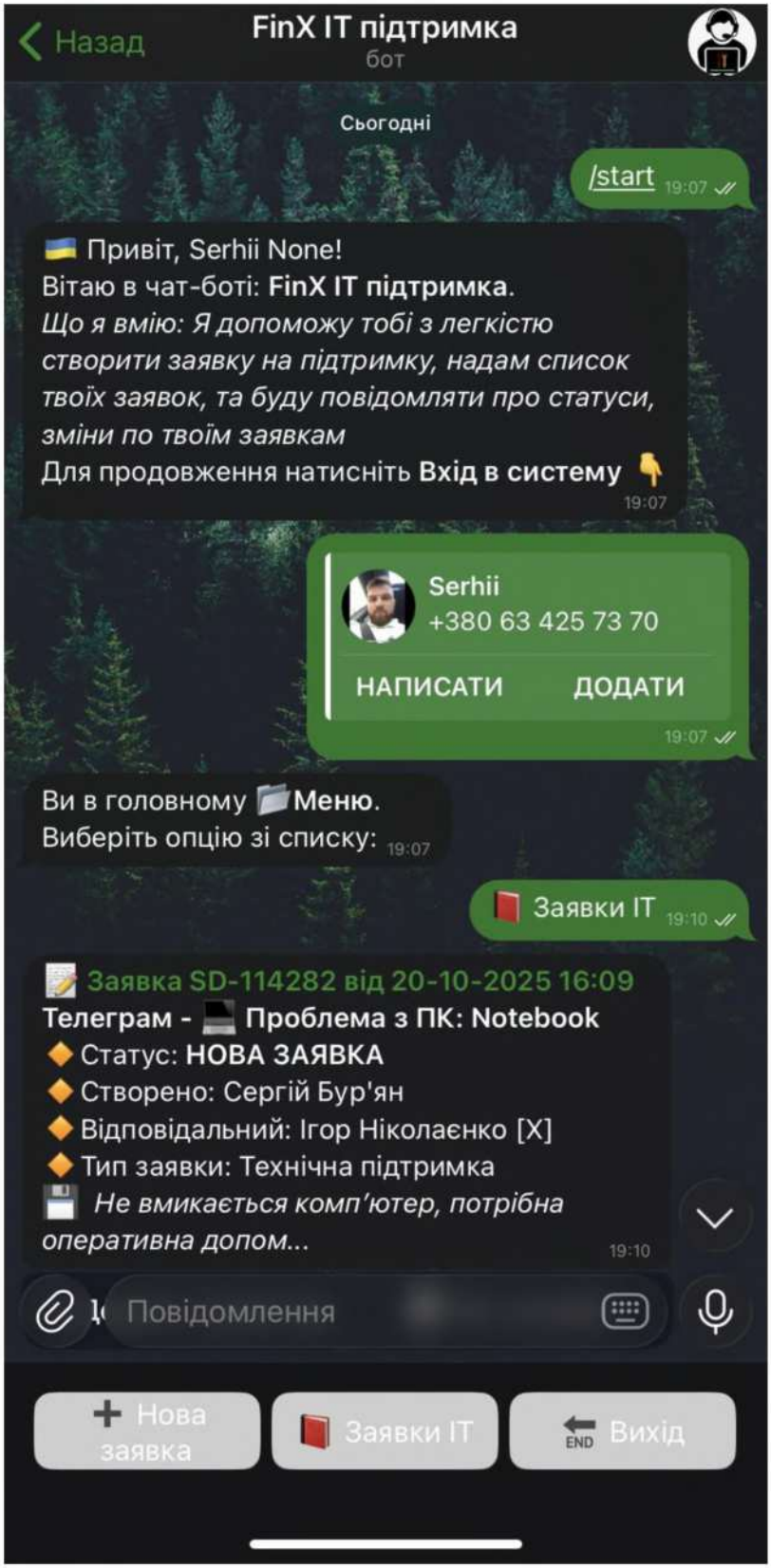
Головне меню спеціаліста з трьома опціями: "Нова заявка", "Мої заявки" та "Вихід"



***авторська розробка**

Додаток С

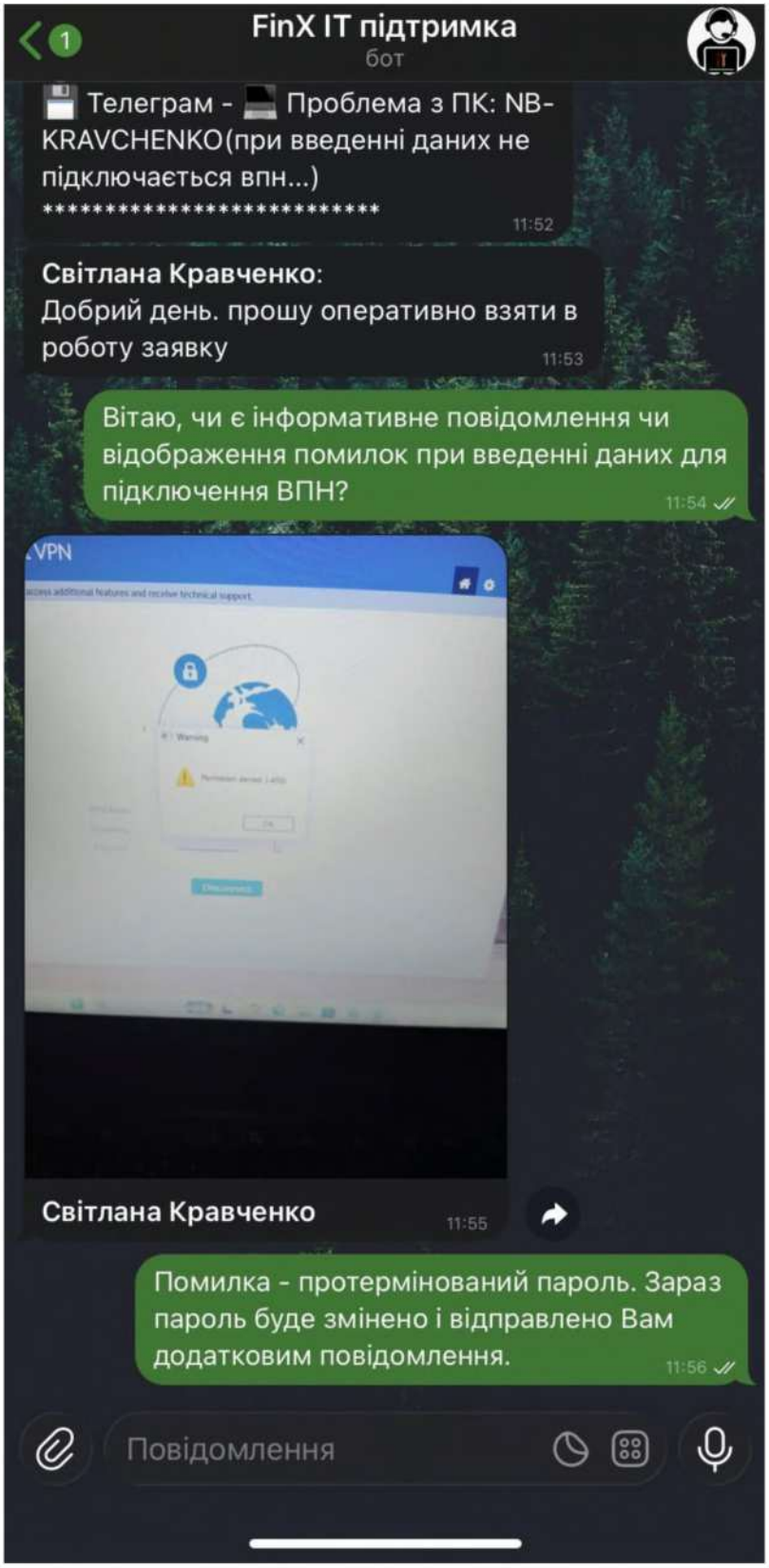
Список усіх заявок, призначених на спеціаліста, в активних статусах



***авторська розробка**

Додаток Т

Чат з спеціаліста з користувачем



***авторська розробка**

Додаток У

Dashboard з відображає заявки та інформацією про неї



Тип	Ключ	Створено ↓	Оновлено	Статус	Тема	Виконавець	Автор	Пріоритет	Время до решения	Время до первого отклика
☑	SD-114282	20.10.2025	20.10.2025	НОВА ЗАЯВКА	Телеграм - Проблема з ПК: Notebook	Ігор Ніколаєнко	Сергій Бур'ян	^	1 г.	30 хв.

Додаток Ф

Відображення повної історії заявки: опис проблеми, всі коментарі, вкладені файли, журнал змін статусів

