ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«УНІВЕРСИТЕТ ЕКОНОМІКИ ТА ПРАВА «КРОК»

КВАЛІФІКАЦІЙНА РОБОТА

Тема: «Месенджер з використанням алгоритмів з шифруванням даних (комплексна робота)»

Ступінь вищої освіти – бакалавр
Спеціальність – 122 «Комп'ютерні науки»
Освітня програма «Комп'ютерні науки»

ПОЯСНЮВАЛЬНА ЗАПИСКА

Виконав: здобувач 4 курсу
групи КН-21
Максим СОРОКІН
Керівник: к.ф.-м.н, доцент, доцент кафедри
інформаційного менеджменту,
математики та статистики
Віра ТКАЧЕНКО

Засвідчую, що кваліфікаційна робота оформлена відповідно до ДСТУ 3008:2015 та не містить запозичень з праць інших авторів без відповідних посилань.
Здобувач: _____
(підпис)

м. Київ – 2025 рік

ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД

«УНІВЕРСИТЕТ ЕКОНОМІКИ ТА ПРАВА «КРОК»

ЗАТВЕРДЖУЮ:
завідувач кафедри
комп'ютерних наук
_____Сергій МІЧКІВСЬКИЙ
«____»____20___р

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Сорокін Максим Васильович

Тема роботи	Месенджер з використанням алгоритмів з шифруванням даних (комплексна робота)
Номер та дата наказу про затвердження теми	№121-7 від 24 грудня 2024 року
Коротка постановка завдання	Розробка корпоративно-орієнтованого месенджеру з використанням AES шифрування та Elliptic curve Diffie–Hellman. (Команда: Сорокін М.В., Орлов О.В.)
Посилання на джерела інформації (не більше п'яти найменувань, які рекомендує науковий керівник)	High performance AES256-GCM for secure communication over 100G Ethernet. // URL: https://www.linkedin.com/pulse/high-performance-aes256-gcm-secure-communication-over-100g (дата звернення 11.03.2024). 1. Exploring E2EE: Real-world Examples of End-to-End Encryption. // URL: https://www.kiteworks.com/secure-file-sharing/real-world-examples-of-end-to-end-encryption/ (дата звернення 11.03.2024).
Вимоги до кваліфікаційної роботи	Кваліфікаційна робота має містити теоретичне, системотехнічне або експериментальне дослідження за темою роботи, яку слід розглядати як складне спеціалізоване завдання або практичну проблему в галузі комп'ютерних наук, яка характеризується комплексністю та невизначеністю умов і потребує застосування теорій і методів інформаційних технологій.

Керівник Віра ТКАЧЕНКО

Здобувач освітнього ступеня бакалавра Максим СОРОКІН

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання	Примітка
	Підготовчий етап		
1	Вибір напряму дослідження	02.12.2024 р.	*виконано*
2	Формування теми та призначення керівника	16.12.2024 р.	*виконано*
3	Затвердження теми кваліфікаційної роботи	23.12.2024 р.	*виконано*
4	Затвердження завдання на кваліфікаційну роботу	27.12.2024 р.	*виконано*
	Основний етап		
5	Розробка концепції кваліфікаційної роботи	13.01.2025 р.	*виконано*
6	Підбір та вивчення джерел інформації з напряму дослідження. Огляд існуючих аналогів	20.01.2025 р.	*виконано*
7	Затвердження розширеної постановки завдання. Підготовка та подання керівникові розділу 1 кваліфікаційної роботи	10.03.2025 р.	*виконано*
8	Проєктування. Підготовка та подання керівникові розділу 2 кваліфікаційної роботи	24.03.2025 р.	*виконано*
9	Підготовка доповіді для експертизи стану виконання кваліфікаційної роботи (проміжний контроль)	31.03-04.04.2025 р.	*виконано*
10	Реалізація. Підготовка та подання керівникові розділу 3 кваліфікаційної роботи	07.04.2025 р.	*виконано*
11	Підготовка та подання керівнику першого варіанту всієї кваліфікаційної роботи	14.04.2025 р.	*виконано*
12	Доопрацювання кваліфікаційної роботи з урахуванням зауважень керівника та представлення керівникові доопрацьованого варіанту кваліфікаційної роботи	21.04.2025 р.	*виконано*
	Завершальний етап		
13	Представлення рукопису для перевірки на плагіат	28.04-04.05.2025 р.	*виконано*
14	Підготовка презентації та доповіді на передзахист	05.05-11.05.2025 р.	*виконано*
15	Передзахист кваліфікаційної роботи	12.05-16.05.2025 р.	*виконано*
16	Доопрацювання роботи за результатами передзахисту	19.05-06.06.2025 р.	*виконано*
17	Експертиза роботи керівником та зовнішнім експертом	09.06-15.06.2025 р.	*виконано*
18	Доопрацювання доповіді та презентації для захисту	09.06-15.06.2025 р.	*виконано*
19	Захист кваліфікаційної роботи	16.06-22.06.2025 р.	*виконано*

Керівник Віра ТКАЧЕНКО

Здобувач освітнього ступеня бакалавра Максим СОРОКІН

Сорокін М.В. Месенджер з використанням алгоритмів з шифруванням даних (комплексна робота).

Пояснювальна записка кваліфікаційної роботи за спеціальністю 122 – Комп'ютерні науки (освітня програма – Комп'ютерні науки) СО Бакалавр. – ВНЗ «Університет економіки та права «КРОК», Навчально-науковий інститут інформаційних та комунікаційних технологій, кафедра комп'ютерних наук, Київ, 2025.

Розглянуто проблему комунікації у сучасних умовах з точки зору кібербезпеки. Розроблено корпоративно орієнтований месенджер, з імплементацією AES шифрування.

Ключові слова: месенджер, кібербезпека, шифрування, AES, ECDH.

Табл. 1. Рис. 35. Бібліограф.: 1 найм.

Sorokin M.V. Messenger using data encryption algorithms (complex work). Project explanatory note by specialty 122 – Computer science. – «KROK» University, Educational and Scientific Institute of information and communication technologies, Department of Computer Science, Kyiv, 2025.

The problem of communication in modern conditions was considered from the point of view of cybersecurity. A corporate-oriented messenger with the implementation of AES encryption was developed.

Keywords: messenger, cybersecurity, encryption, AES, ECDH.

Tbl. 1. Fig. 35. Bibliography: 1 Items.

ЗМІСТ

ВСТУП

Актуальність теми: сучасний процес спілкування у колективах потребує швидких та оперативних комунікацій між співробітниками цього самого колективу.

На сьогоднішній день вкрай розповсюдженим інструментом комунікації як між співробітниками, так і керівництвом є месенджери, через їхню зручність і широкий функціонал, такий як можливість вести переписку (що є зручнішим за дзвінки, адже дає змогу відповісти в будь-який зручний час після отримання повідомлення), групові чати, можливість надсилати медіа, а також нерідкою є підтримка як звичайних дзівнків так і відеозв'язку[1].

Проблемна ситуація: як правило, через простоту та доступність месенджерів-сервісів, яким і віддають перевагу різні колективи, дані які циркулюють через нехай і локальні чати, можуть опинитися під загрозою, тому що вся інформація про переписку та її вміст знаходиться на серверах корпорації, що володіє месенджером.

Це не тільки ставить під загрозу корпоративну таємницю через недостатню поінформованість про правила користування та конфіденційності (terms of service, privacy policy) але й про можливий доступ державних органів до даних на запит від уряду недружньої країни (адже досить часто країни зобов'язують власників тих чи інших цифрових продуктів, розміщувати свої сервіси на серверах у тій країні, де вони зареєстровані як юридична особа).

Крім цього, як показує практика, навіть досить великі корпорації не особливо дбають про безпеку даних їхніх користувачів. практично щорічно можна знайти новини про витіки баз даних різних сервісів[2]. Отже, всі перераховані вище фактори підштовхують до того, що використання пропрієтарних рішень від інших надавачів послуг може бути досить небезпечним, а також позбавляє гарантії контролю над інформацією, що розповсюджується в закритому колективі (оскільки ця інформація фактично знаходиться в руках особи, яка надає послуги, відповідно до користувальницької угоди.)

Мета дослідження: метою роботи є розробка сучасного месенджера з відкритим вихідним кодом та наскрізним AES шифруванням, що забезпечило б достатній рівень безпеки даних за сучасними стандартами. Поточна реалізація дозволила б організаціям розгортати месенджер у себе локально на серверах, не покладаючись на представників послуг.

Завдання дослідження: завдання роботи полягає у вивченні існуючих аналогів та реалізацій різних месенджерів, вивченні їх сильних та слабких сторін, і на основі отриманих знань розробити більш досконалу свою реалізацію.

Об'єкт дослідження: системи обміну повідомленнями та механізми захисту даних у подібних системах

Предмет дослідження: засоби реалізації захисту та конфіденційності даних у месенджерах з відкритим вихідним кодом та наскрізним шифруванням.

Методи дослідження:

- аналіз літератури та існуючих рішень, що передбачає вивчення наукових статей, документації та реалізацій популярних месенджерів;

- аналіз конкурентів та аналогів, дослідження сильних і слабких сторін існуючих рішень для виявлення їхніх недоліків та переваг;

- моделювання та розробка архітектури безпечного месенджера;

- проєктування та створення програмної реалізації месенджера з локальним розгортанням та наскрізним AES-шифруванням.

Практичне значення: реалізований корпоративний месенджер дасть можливість різним організаціям мати зручний та гнучкий інструмент комунікації, що відповідає сучасним вимогам, а також є повністю підконтрольним цій організації, тим самим при його використанні зменшуючи ризик витоків конфіденційної інформації та підвищивши безпеку всередині компанії, інституту організації, тощо.

Структура роботи. кваліфікаційна робота складається зі вступу, трьох розділів, висновків та списку посилань (найменувань). Пояснювальна записка

містить 2 таблиці, 35 рисунків. Загальний обсяг пояснювальної записки складає 54 сторінок, основний зміст викладено на n сторінках.

РОЗДІЛ 1
ПОСТАНОВКА ЗАВДАННЯ НА РОЗРОБКУ

1.1 Опис предметної області

Сучасний світ наскрізь пронизаний технологіями. Сьогоднішню епоху справедливо називають інформаційною, адже наше життя тісно пов'язане з технологічними засобами, основна мета яких полягає у зберіганні, обробці та передачі інформації. Найбільшою з таких систем є Інтернет - всесвітня мережа, створена для об'єднання локальних і регіональних мереж в єдине середовище з можливістю вільного обміну інформацією.

Однак технології не стоять на місці, і формати спілкування в мережі теж постійно змінюються: комунікації через всесвітню павутину пройшли довгий шлях, від емейлів та веб-серфінгу різних сайтів та форумів і закінчуючи на сьогоднішній день нам звичним форматом соціальних мереж, та груп у месенджерах. Месенджери є досить зручним засобом, тому що надають можливість миттєвого обміну повідомленнями в реальному часі, що несе в собі великий перелік плюсів (починаючи від очевидних, таких як відсутність великих затримок, і закінчуючи зручністю текстового формату, який можна прочитати у будь-який зручний час, як тільки з'явиться така можливість).

Проте, в сучасному світі, інтернет комунікація це вже давно не просто зручність, а справжнісінька необхідність, інструмент яким користуються щодня мільйони людей по всьому світу щоб залишатися на зв'язку, комунікувати між собою, обмінюватися інформацією, як у локальних підгрупах, так і глобально, розповсюджуючи ідеї та інформацію. Популярність месенджерів також зростає через неймовірну поширеність і доступність інтернету в сьогоднішні дні[3], що призвело до того, що більшість традиційних засобів передачі інформації (радіо, газети, телебачення) також перейшли в інтернет (більше того, з появою інтернету та приходом можливості завдяки глобальній мережі набагато простіше поширювати інформацію, стали з'являтися також і перші незалежні інтернет ЗМІ), спочатку у формі своїх

власних веб-сайтів, і потім почали почали переходити і до інструментарію, що був більш популярним у колах простого населення, щоб мати можливість мовити на куди ширшу публіку.

Все це, безсумнівно, призвело до значного розширення функціоналу месенджерів, через їх популярність. Наприклад, у Telegram є система каналів (по суті блогів), з коментарями, реакціями, можливістю створення голосувань, тощо.

Однак, навіть у таких неймовірно великих, по-справжньому гіганстких платформ, все також існують і недоліки, властиві будь-яким централізованим системам. По-перше, потрібно розуміти, що вся інформація що проходить через месенджер, зберігається на серверах компанії, що володіє цим месенджером, і відповідно надає послуги у вигляді можливості користування своїм додатком. так як практично завжди, вихідний код клієнтської, що серверної частини є закритим, ми не можемо напевно знати як весь цей загальний механізм працює, яку інформацію збирає, зберігає, і ми як користувачі обмежені лише користувальницькою угодою та політикою конфіденційності на які ми погоджуємось при встановленні. в останніх, як правило коротко і досить розмито подається сам факт, що постачальник дійсно збирає інформацію, зберігає її у себе, і (як правило) може її використовувати у своїх цілях, навчати на ньому нейромережі, передавати третім особам, тощо.

Так, звісно більшість месенджерів турбуються про безпеку збереження даних, (адже ніяк в інформаційний вік - інформація це теж цінний товар або ресурс, більш того головний як зрозуміло з назви періоду в який ми живемо), проте навіть у таких умовах це не дає жодних гарантій безпеки та (або) секретності, коли та може бути необхідна. це стосується не тільки месенджерів, а й сучасного спільного ринку додатків та програм в цілому, а також це є однією з причин, чому офіси національної безпеки різних країн так наполягають, щоб на телефонах політиків та інших людей, що працюють у державних установах, були відсутні програми, що належать компаніям розташованим у недружніх, або потенційно недружніх країнах.

Тому витікаючи з сьогоднішніх реалій, досить розумним буде створення месенджера який був би набагато конфіденційнішим, і в той же час зручним, відповідним сучасним уявленням про дизайн та належний функціонал, і був би таким же простим і інтуїтивним для непідготовленого користувача, при цьому не втрачаючи своїх плюсів у вигляді конфіденційності та безпеки.

Забезпечення конфіденційності переданих даних є однією з ключових вимог до запропонованої системи. Зважаючи на потреби в безпечній корпоративній комунікації, у проєкті передбачено використання наскрізного шифрування на основі симетричного алгоритму AES (Advanced Encryption Standard). Цей алгоритм зарекомендував себе як ефективний стандарт для захисту повідомлень, файлів і сеансів зв’язку в реальному часі. В кожному окремому чаті передбачено генерацію унікального симетричного ключа, що дозволяє ізолювати канали обміну інформацією між учасниками.

1.2 Визначення потенційних конкурентних переваг програмного продукту

Ідея створення месенджера або схожого рішення, яке можна розгорнути локально в межах організації, не є новою. Подібні системи були досить поширеними у 2000-х роках і активно використовувалися для внутрішньої комунікації.

Першим конкурентом котрого можна розглянути, є IRC. за своєю суттю, IRC це просто протокол, який є досить простим у своїй реалізації, без нічого зайвого. він був створений аж у 1988 році, лише однією людиною, вченим та програмістом Ойкаріненом Яркко[4]. Коли користувач підключається до IRC серверу, йому доступний список каналів цього сервера, до кожного з яких він може підключитися. у кожному каналі може бути кілька користувачів. всі повідомлення, що надсилаються в канал, видно всім користувачам, що знаходяться в ньому. Також у кожного каналу є своя назва, в якій зазвичай і вказується тематика, що обговорюється (наприклад #issues, #support, тощо).

До явних недоліків IRC (окрім морального застарювання) можна віднести:

- неможливість відсилати медіа стандартними способами; (Деякі клієнти мають таку можливість, але це може призвести до проблем сумісності різних клієнтів, і сама реалізація є незграбною, оскільки сам протокол ніколи не мав та не планував такого функціоналу);

- неможливість бачити, що писали інші учасники до вашого приєднання до каналу; само собою це неймовірно незручно за сучасними мірками, тому що всі ваші листування не логуються (а навіть якщо це роблять на сервері сторонніми методами, в рамках протоколу або клієнта все одно отримати доступ до старого листування є досить проблематично);

- велика кількість клієнтів: непідготовленого користувача різноманітність клієнтів може запросто спантеличити, і він може випадково вибрати просто поганий або невідповідний клієнт (наприклад випадково вибрати клієнт призначений виключно для консолі, хоча набагато зручніше був би клієнт з графічним інтерфейсом);

- відсутність будь-якої безпеки за замовчуванням: багато серверів не використовують якогось шифрування, а передають простий текст. При атаці людина посередині (MitM), ця людина може легко підглядати за вмістом вашої переписки. IRC можна змусити шифрувати повідомлення, але цього потрібно встановити доповнення, що створює зайву мороку.

На рис. 1.1 наведено клієнт Irssi, як приклад середньостатистичного IRC клієнту з інтерфейсом командного рядка.

Наступним аналогом є протокол XMPP (стара назва – Jabber). Це протокол, заснований на XML, на відміну від IRC, XMPP виглядає набагато солідніше: він є набагато більш розширюваним, що забезпечено XEP доповненнями (наприклад XEP-0163 Personal Eventing Protocol, що доповнює розміщення фото профілю, передачу ключів OMEMO).

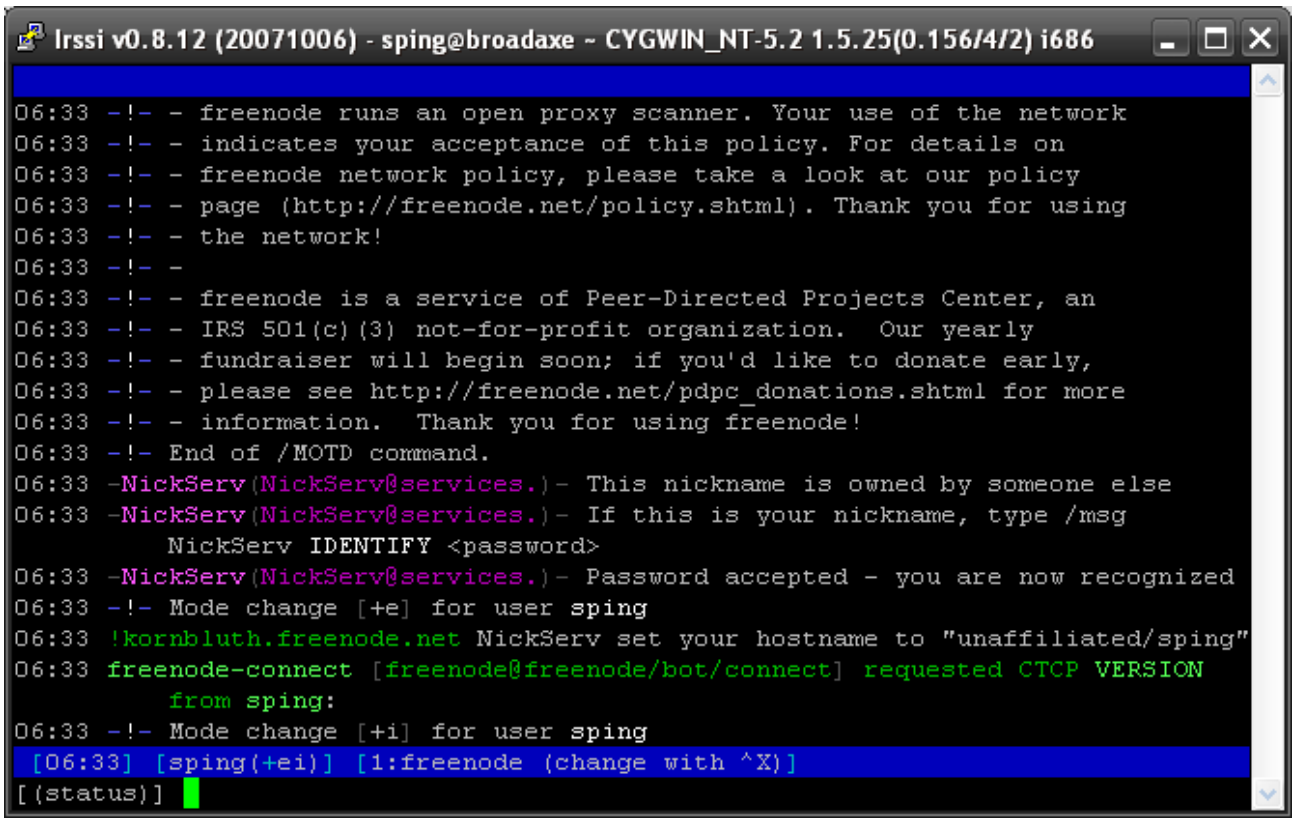

Рисунок 1.1 – Клієнт Kiwi IRC

Джерело: [5].

Більшість популярних XMPP клієнтів зазвичай мають всі необхідні розширення за умовчанням, і навіть незважаючи на те, що XMPP не отримав належного поширення, видно, що розробники та спільнота не забули про нього, і протокол весь цей час розвивався і доповнювався. таким чином на сьогоднішній день XMPP набагато більше відповідає сучасним стандартам спілкування ніж IRC, підтримує шифрування (OTR що засноване на PGP та OMEMO, на вибір), а також підтримує відправлення медіа. однак, навіть так він не позбавлений мінусів: XMPP все також залишається досить складним у налаштуванні серверної частини, а також існуючі навіть популярні клієнти, такі як Dino (приклад інтерфейсу наведено на рис. 1.2), Gajim та Converstations є неймовірно неінтуїтивними, заплутаними, і досі мають дещо застарілий UI дизайн.

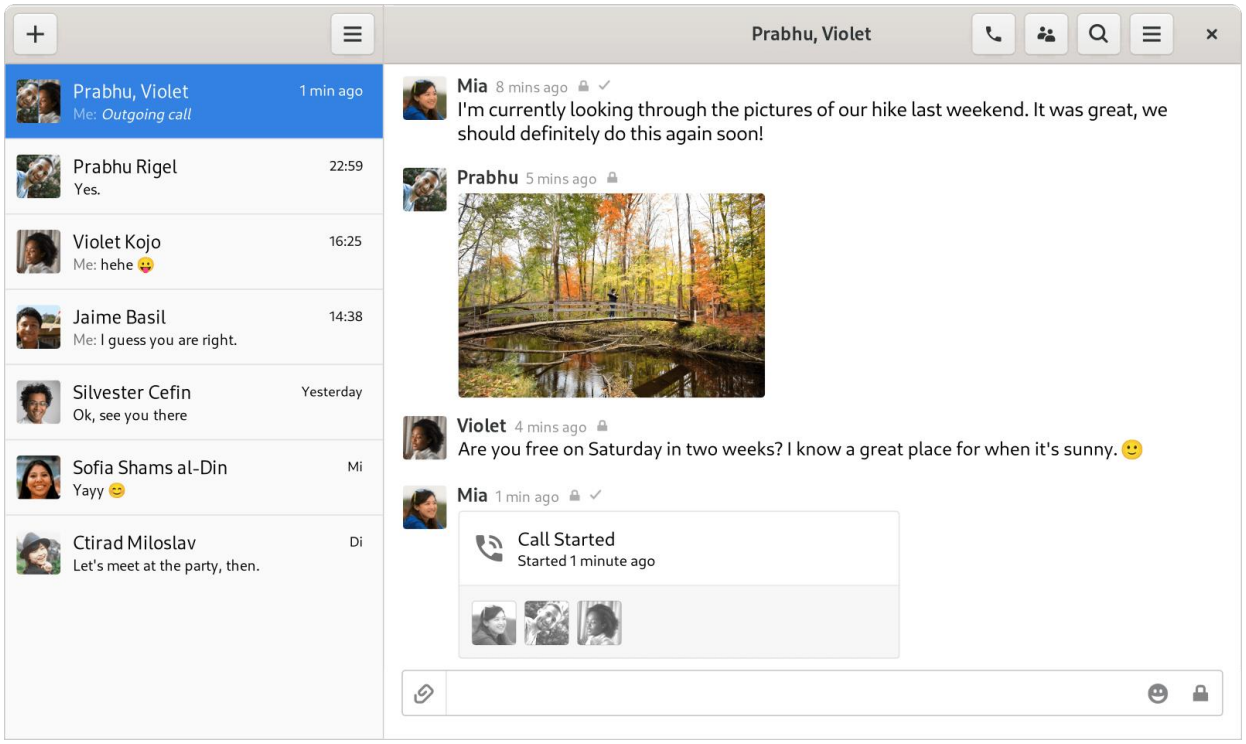

Рисунок 1.2 – XMPP клієнт Dino

Джерело: [6].

Ще одним можливим конкурентом можна назвати протокол matrix (іноді стилізується як [matrix]). на перший погляд це дійсно гідна на сьогоднішній день існуюча альтернатива: у популярних matrix-клієнтів таких як element (приклад інтерфейсу наведено на рис. 1.3), дуже приємний, сучасний і дійсно інтуїтивний та простий дизайн, а сам протокол має скрізне шифрування і на вигляд всі переваги децентралізованих протоколів як XMPP, при цьому зберігаючи елегантність та інтуїтивність сучасного UI дизайну, притаманного сучасним популярним месенджерам.

Однак, разом з цим у протоколу Matrix є й серйозні недоліки: у 2021 році, у ряді популярних Matrix клієнтів (Element (Web/Desktop/Android), FluffyChat, Nheko, Cinny, та SchildiChat) були виявлені критичні вразливості CVE-2021-40823 та CVE-2021-40824[8]. Вони дозволяють отримати відомості про ключі, використані для передачі повідомлень у чатах із наскрізним шифруванням (E2EE).

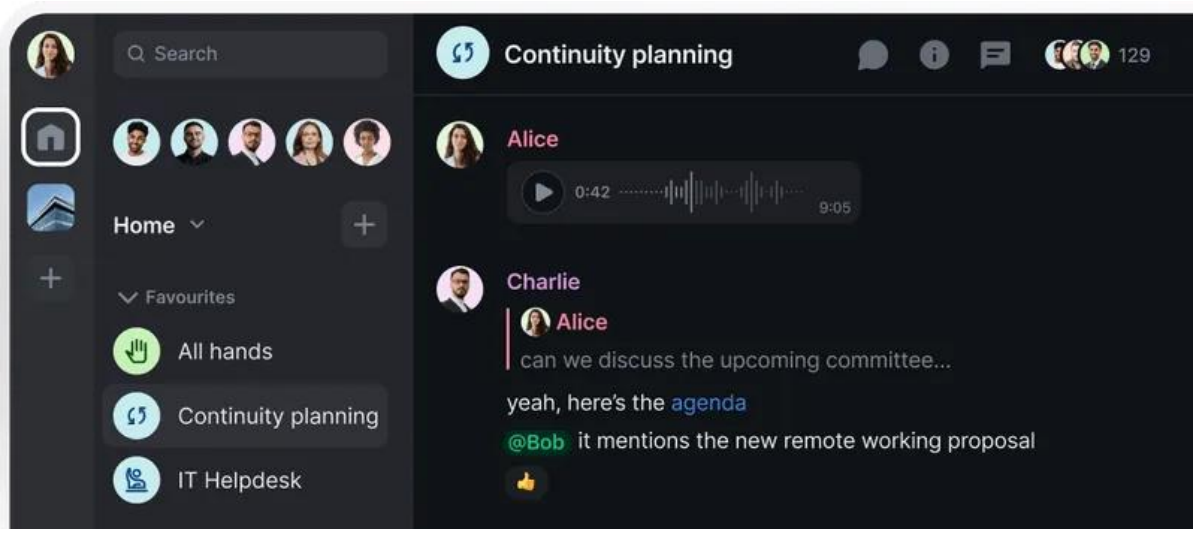

Рисунок 1.3 – Matrix клієнт Element (версія для ПК)

Джерело: [7].

Атакуючий, скомпрометувавши одного з користувачів чату, може розшифрувати повідомлення, раніше надіслані цьому користувачеві з вразливих клієнтських програм. Само собою цей випадок підірвав довіру до протоколу багатьох користувачів. після цього почали розповзатися чутки про можливий бекдор всередині протоколу (або клієнтів), проте подальші аудити безпеки нічого не виявили, хоча звісно певний слід в історії протоколу це наклало. Окрім цього, Matrix не позбавленний й інших недоліків:

- сервера Matrix надзвичайно незручні в управлінні. Стандартне серверне програмне забезпечення Matrix зовсім не гнучке. Спроба зробити щось просте, як наприклад видалити обліковий запис користувача за допомогою командного рядка, майже неможливе що досить розчаровує. Скоріш за все, вам доведеться самостійно відкривати бази даних і робити це вручну. Також у Matrix явно не вистачає параметрів конфігурації;

- Matrix — це «bloatware» тип програмного забезпечення. В середньому Matrix сервер розрахований на 1 клієнтський обліковий запис, може потребувати до 300 мегабайт оперативної пам'яті. в той час, як сервер XMPP, в середньому в таких самих умовах в 15(!) разів є більш економним;

- Matrix є неоптимальним рішенням з точки зору безпеки метаданих. Оскільки Matrix насправді не просто обмінюється окремими повідомленнями, а синхронізує цілі чати з усіма залученими серверами, це означає, що, хоча всі повідомлення можуть бути зашифровані наскрізно, метадані розмови відомі всім серверам, включаючи облікові записи, коли надсилаються повідомлення та іншу інформацію про обліковий запис, яка оприлюднюється (наприклад, користувачі можуть додавати свої електронні адреси та номери телефонів до своїх облікових записів).

Це означає, що всі сервери Matrix, особливо Matrix.org, по суті є величезним сховищем метаданих. Хоча чати є зашифрованими, все одно - зашифровані журнали чатів синхронізуються між усіма відповідними серверами, поширюючи метадані[9].

Для підведення підсумків, порівняння недоліків та переваг конкурентів було наведено у табл. 1.1.

Таблиця 1.1 – Порівняння недоліків та переваг конкурентів

Назва конкурента	Інтуїтивний UI	Наява шифрування	Простота розгортання та управління сервером	Відповідність функціоналу до сучасних стандартів	Оптимізація
IRC	Ні	Ні	Так	Ні	Так
XMPP	Ні	Так	Ні	Так	Так
Matrix	Так	Так	Ні	Так	Ні

Таким чином, поставлена ціль – розробити месенджер який буде враховувати всі недоліки аналогів, тим самим надавши кінцевим користувачам зручний, та інтуїтивний інструмент, який буде мати достатню оптимізацію, буде забезпечувати достатню безпеку, а також своїм функціоналом не буде поступатися сучасним конкурентам у сфері.

1.3 Постановка завдання на кваліфікаційну роботу

Месенджер, який заплановано розробити, спеціально розрахований на потреби корпорацій. У месенджері користувачі зможуть створювати як особисті чати для приватного спілкування, так і групові чати для обговорення навчальних проєктів, організації заходів або просто для спілкування з одногрупниками. Платформа підтримуватиме відправку не лише звичайних текстових повідомлень, а й медіа-файлів, таких як фото, відео, та інші формати файлів. Крім того, кожен користувач зможе налаштувати свій профіль, змінюючи фото та додаючи опис у розділ “біо”, що дозволить особисто налаштувати свій акаунт.

Також серед функціональних вимог до системи особливо виділено реалізацію наскрізного шифрування, що забезпечує захист повідомлень як під час передачі, так і в стані збереження. Алгоритм AES обрано не лише через його криптографічну стійкість, але й завдяки його підтримці в сучасних криптобібліотеках, що спрощує інтеграцію в програмну архітектуру. Такий підхід відповідає сучасним стандартам безпеки та підвищує довіру користувачів до цифрового продукту .

Основні переваги запропонованого месенджеру включають:

1) простий і сучасний дизайн – ми зробили все можливе, щоб інтерфейс був інтуїтивно зрозумілим та приємним для користувача. Кожен елемент дизайну продуманий до дрібниць, щоб забезпечити максимальний комфорт під час використання;

2) легкість розгортання на сервері – одна з ключових переваг нашого месенджеру полягає в тому, що він дуже простий у встановленні. Все, що потрібно зробити, – це запустити один файл, і система буде готова до роботи. Це значно спрощує процес впровадження для адміністраторів університету або будь-якої іншої організації;

3) надійне AES шифрування – безпека наших користувачів є нашим пріоритетом. Усі повідомлення, як текстові, так і медіа-файли, будуть надійно захищені за допомогою скрізного AES шифрування. Для кожного чату

генерується окремий AES ключ, що забезпечує максимальний рівень конфіденційності та захисту даних.

Висновки до розділу 1

Було розглянуто основні аспекти предметної області, проаналізовано сучасні тенденції в організації внутрішньої цифрової комунікації, а також здійснено огляд подібних реалізацій, що існували раніше. Особливу увагу зосереджено на загрозах, пов’язаних із централізованим зберіганням повідомлень і вразливістю протоколів шифрування.

Визначено ключові переваги, які має запропонований підхід — зокрема, простота розгортання, автономність роботи та орієнтація на корпоративне середовище.

На основі проведеного аналізу було сформульовано цілі проєкту й окреслено перелік вимог до програмного продукту. Одним із пріоритетів визначено безпеку переданої інформації, що передбачає використання наскрізного шифрування на основі алгоритму AES. Це дозволяє гарантувати конфіденційність комунікації між користувачами навіть у випадку компрометації серверної частини або каналів передачі даних.

РОЗДІЛ 2
ПРОЄКТУВАННЯ

2.1 Моделювання поведінки продукту

У попередньому розділі було проведено детальний аналіз предметної області, визначено основні вимоги до корпоративного месенджера, а також сформовано архітектурні рішення, що забезпечують його функціональність та безпеку. На цьому етапі дослідження необхідно перейти до моделювання поведінки програмного продукту, що дозволить наочно продемонструвати логіку взаємодії між користувачами та системою, а також визначити ключові процеси, які відбуваються під час його експлуатації.

Моделювання поведінки є критично важливим етапом проектування будь-якої інформаційної системи, оскільки воно дозволяє формалізувати динамічні аспекти її функціонування. У рамках даного підрозділу будуть розглянуті такі аспекти:

- діаграма використання (Use Case), яка відображає основні сценарії взаємодії користувачів із системою;
- діаграма діяльності (Activity), що ілюструє алгоритми виконання ключових процесів;
- діаграма послідовності (Sequence), яка демонструє хронологічний порядок обміну повідомленнями між компонентами системи;
- діаграма комунікації (Communication), що акцентує увагу на взаємодії об'єктів у рамках певного сценарію використання.

Кожна з цих діаграм відіграє важливу роль у проектуванні системи, оскільки дозволяє розробникам, тестувальникам та іншим зацікавленим сторонам краще зрозуміти логіку роботи месенджера, а також виявити потенційні слабкі місця ще на етапі проектування.

Одним із найбільш важливих інструментів візуалізації функціональних вимог є діаграма використання (Use Case Diagram). Вона дозволяє відобразити основні сценарії взаємодії користувачів із системою, визначити межі

програмного продукту та ідентифікувати ролі, які беруть участь у цих процесах.

У контексті корпоративного месенджера, що розробляється, діаграма використання дозволяє чітко окреслити можливості кожної з трьох ролей: Owner, Адміністратор та Користувач. Кожен із цих акторів має свій набір дій, які він може виконувати в системі, що відображає гнучкість та безпеку запропонованого рішення.

Наприклад, Owner має максимальний рівень доступу, включаючи можливість керування адміністраторами, тоді як звичайний користувач обмежений лише основними функціями, такими як обмін повідомленнями, редагування профілю та управління чатами. Адміністратор займає проміжне положення, оскільки йому дозволено додавати нових користувачів та блокувати їх, але він не може втручатися у роботу інших адміністраторів або Owner.

Діаграма використання також дозволяє виявити загальні сценарії, такі як авторизація в системі, зміна пароля, створення нового чату, які можуть бути доступні для всіх ролей, але реалізовані з урахуванням їхніх прав доступу.

На рис 2.1 зображена діаграма використання.

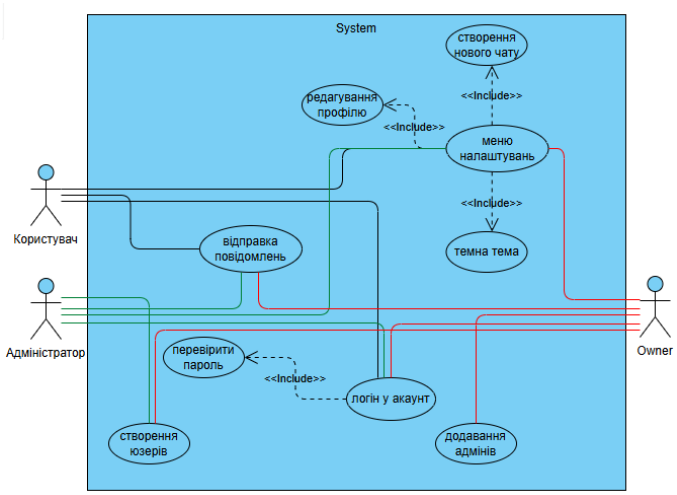

Рис 2.1 – Діаграма використання

Джерело: розроблено автором.

Далі необхідно детальніше розглянути, як саме відбуваються ключові процеси в системі, для чого буде використано діаграму діяльності (Activity Diagram).

Діаграма діяльності є одним із ключових інструментів візуалізації бізнес-процесів та алгоритмів у рамках об'єктно-орієнтованого проектування. Вона належить до категорії поведінкових діаграм UML (Unified Modeling Language) і використовується для моделювання послідовності дій, які виконуються в системі, з урахуванням умовних переходів, паралельних потоків та циклічних операцій.

Основна мета діаграми діяльності полягає в тому, щоб наочно відобразити логіку виконання складних процесів, які можуть включати декілька учасників або підсистем. На відміну від діаграми використання, яка фокусується на взаємодії користувача з системою на високому рівні абстракції, діаграма діяльності дозволяє деталізувати внутрішню структуру процесів, що відбуваються під час виконання того чи іншого сценарію.

У контексті корпоративного месенджера діаграми діяльності є особливо корисними для опису таких процесів, як:

- авторизація користувача (з урахуванням різних сценаріїв входу, включаючи первинну реєстрацію);
- обмін повідомленнями (від моменту створення до доставки та відображення в інтерфейсі);
- створення та управління чатами (включаючи додавання учасників, шифрування даних тощо);
- керування обліковими записами (зміна пароля, редагування профілю, блокування користувачів адміністратором).

Крім того, діаграми діяльності дозволяють виявити потенційні "вузькі місця" в алгоритмах, наприклад, надмірно складні умовні переходи або можливі конфлікти при паралельному виконанні операцій.

Одним із найбільш важливих процесів у корпоративному месенджері є авторизація користувача, оскільки саме на цьому етапі відбувається перевірка прав доступу та ініціалізація сеансу роботи з системою.

На рис 2.2 зображена діаграма діяльності.

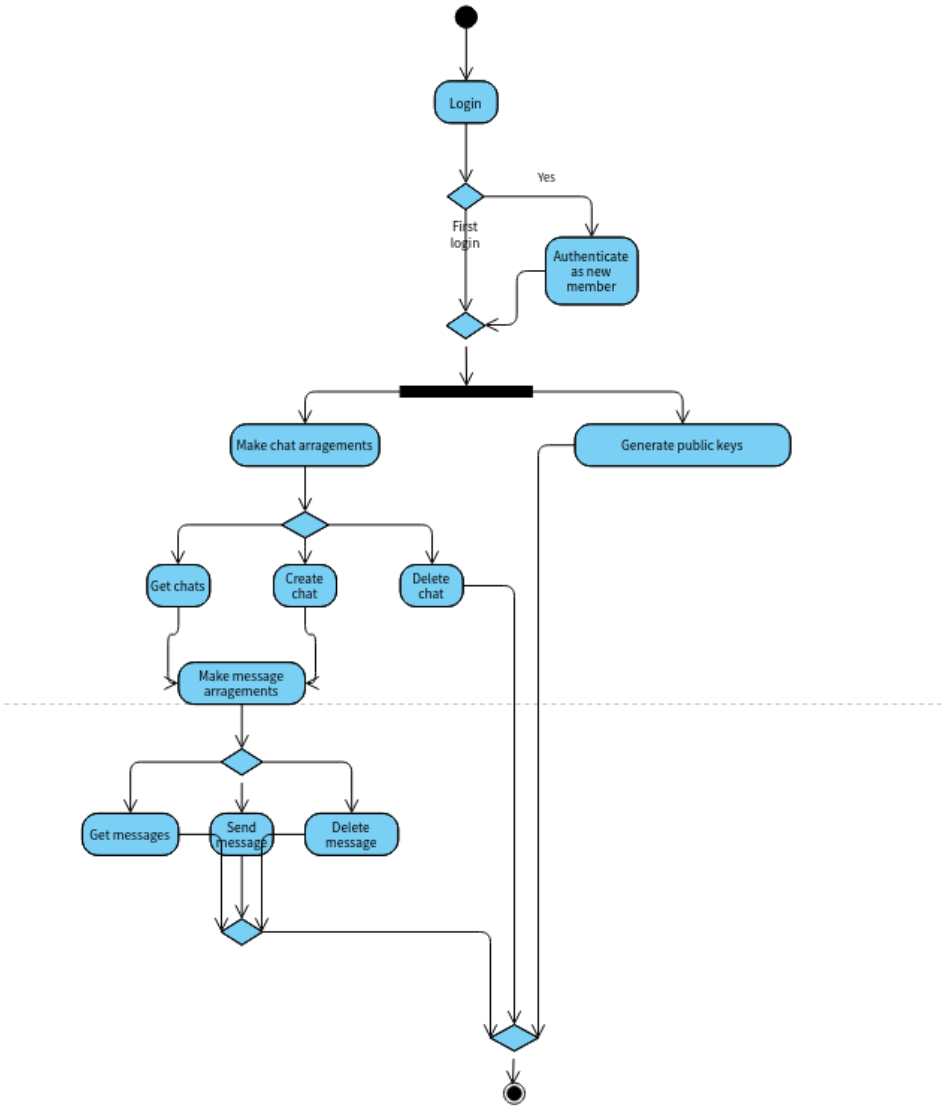

Рис 2.2 – Діаграма діяльності

Джерело: розроблено автором.

Таким чином, діаграма діяльності є незамінним інструментом для деталізації складних процесів, що відбуваються в корпоративному месенджері. Вона дозволяє не лише формалізувати алгоритми виконання

окремих операцій, але й виявити потенційні проблеми на етапі проектування, що значно підвищує якість кінцевого продукту.

Далі буде розглянуто діаграму послідовності (Sequence Diagram), яка доповнює діаграму діяльності, акцентуючи увагу на часовій послідовності взаємодії між об’єктами системи.

Діаграма послідовності займає особливе місце серед інструментів UML, оскільки дозволяє наочно відобразити динамічну взаємодію між об'єктами системи в часовому контексті. На відміну від статичних діаграм класів, які фіксують структуру системи, діаграми послідовності акцентують увагу на часовій розгортці подій, що є критично важливим для розуміння поведінки складних систем у реальному часі.

Методологічна цінність діаграм послідовності полягає в їхній здатності:

- відтворювати точний порядок взаємодій між компонентами;
- визначати часові залежності між операціями;
- виявляти потенційні проблеми синхронізації;
- моделювати альтернативні сценарії виконання операцій.

У контексті корпоративного месенджера діаграми послідовності стають особливо корисними для аналізу:

- процесів автентифікації та авторизації;
- механізмів обміну повідомленнями;
- процедур управління чатами;
- системи обробки помилок та виняткових ситуацій.

Процес авторизації в корпоративному месенджері представляє собою складну багаторівневу взаємодію між:

- клієнтським додатком;
- сервером автентифікації;
- базою даних користувачів;
- системою керування сесіями.

На рис 2.3 зображена діаграма послідовності.

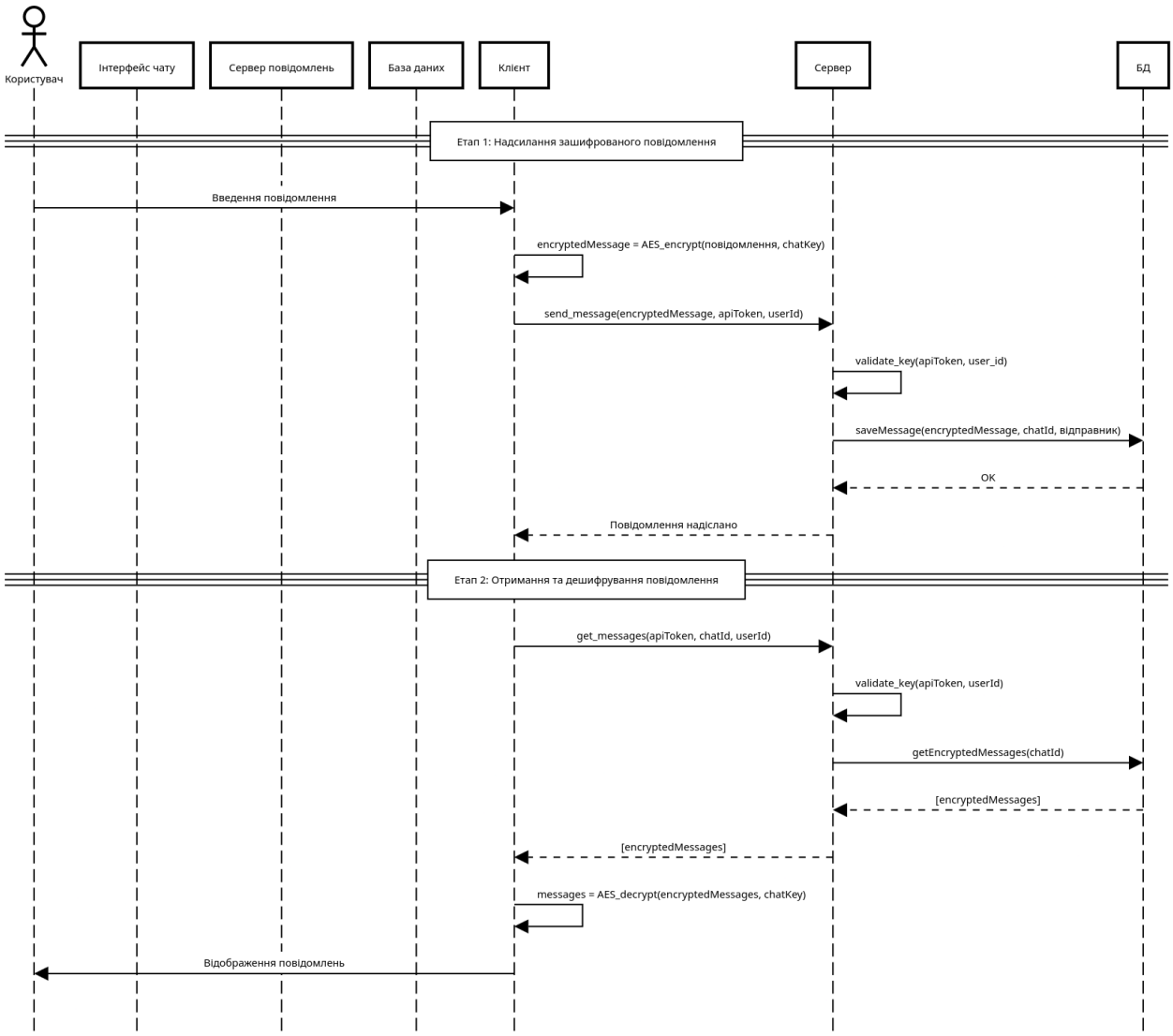

Рис 2.3 – Діаграма послідовності

Джерело: розроблено автором.

Процес відправлення повідомлення в корпоративному месенджері включає складний ланцюг взаємодій між численними компонентами системи.

Далі буде розглянута діаграма комунікацій, яка також є одним із ключових інструментів UML для візуалізації взаємодії об'єктів у рамках конкретного сценарію використання. На відміну від діаграми послідовності, яка акцентує увагу на часовій послідовності повідомлень, діаграма комунікації робить акцент на структурі взаємодії між об'єктами та організації їх зв'язків.

У контексті корпоративного месенджера діаграми комунікації особливо корисні для:

- аналізу складних багатокомпонентних взаємодій;
- проектування архітектури розподілених підсистем;
- оптимізації мережевих взаємодій;
- визначення точок інтеграції з зовнішніми системами.

На рис 2.4 зображена діаграма комунікації.

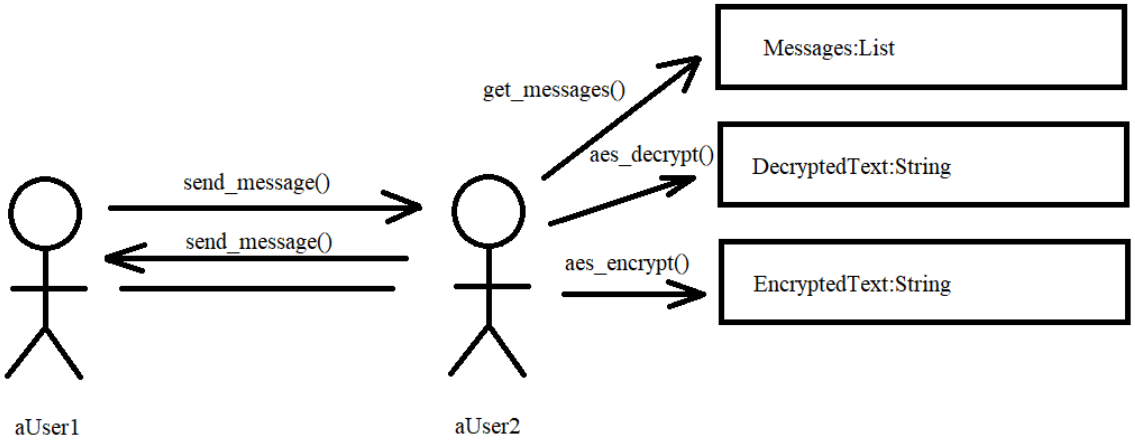

Рис 2.4 – Діаграма комунікації

Джерело: розроблено автором.

Діаграма комунікації дозволяє чітко відобразити роль кожного компонента в процесі, напрямки передачі повідомлень, умовні переходи, альтернативні сценарії виконання.

2.2 Моделювання структури продукту

На попередніх етапах дослідження було детально розглянуто динамічні аспекти функціонування корпоративного месенджера за допомогою поведінкових діаграм. Однак для повноцінного розуміння архітектури системи необхідно звернутись до структурного моделювання, яке дозволяє визначити базові елементи системи та зв'язки між ними.

Структурне моделювання є фундаментальним етапом проектування будь-якої програмної системи, оскільки воно:

- формалізує основні сутності предметної області;
- визначає принципи організації коду;
- забезпечує узгодженість між різними компонентами системи;
- слугує основою для подальшого вдосконалення архітектури.

У рамках даного підрозділу будуть розглянуті три ключові типи структурних діаграм:

- діаграма класів - відображає основні сутності системи та їх взаємозв'язки;
- діаграма об'єктів - демонструє конкретні екземпляри класів у певний момент часу;
- діаграма компонентів - показує організацію фізичних компонентів системи.

Кожен з цих інструментів має унікальне призначення та доповнює загальну картину архітектури системи.

Діаграма класів є основним інструментом структурного моделювання в UML і представляє собою статичний вигляд системи, який залишається незмінним протягом усього життєвого циклу розробки. На відміну від ER-діаграм, які використовуються переважно для моделювання баз даних, діаграми класів UML містять значно більше інформації про поведінку та стан об'єктів.

Ключові переваги діаграми класів включають:

- універсальність - можливість представлення як логічної, так і фізичної структури системи;
- гнучкість - підтримка різних рівнів абстракції;
- інтуїтивність - зручність сприйняття для всіх учасників проекту;
- комплексність - можливість відображення всіх аспектів класів (атрибути, методи, зв'язки).

На рис 2.5 зображена діаграма класів.

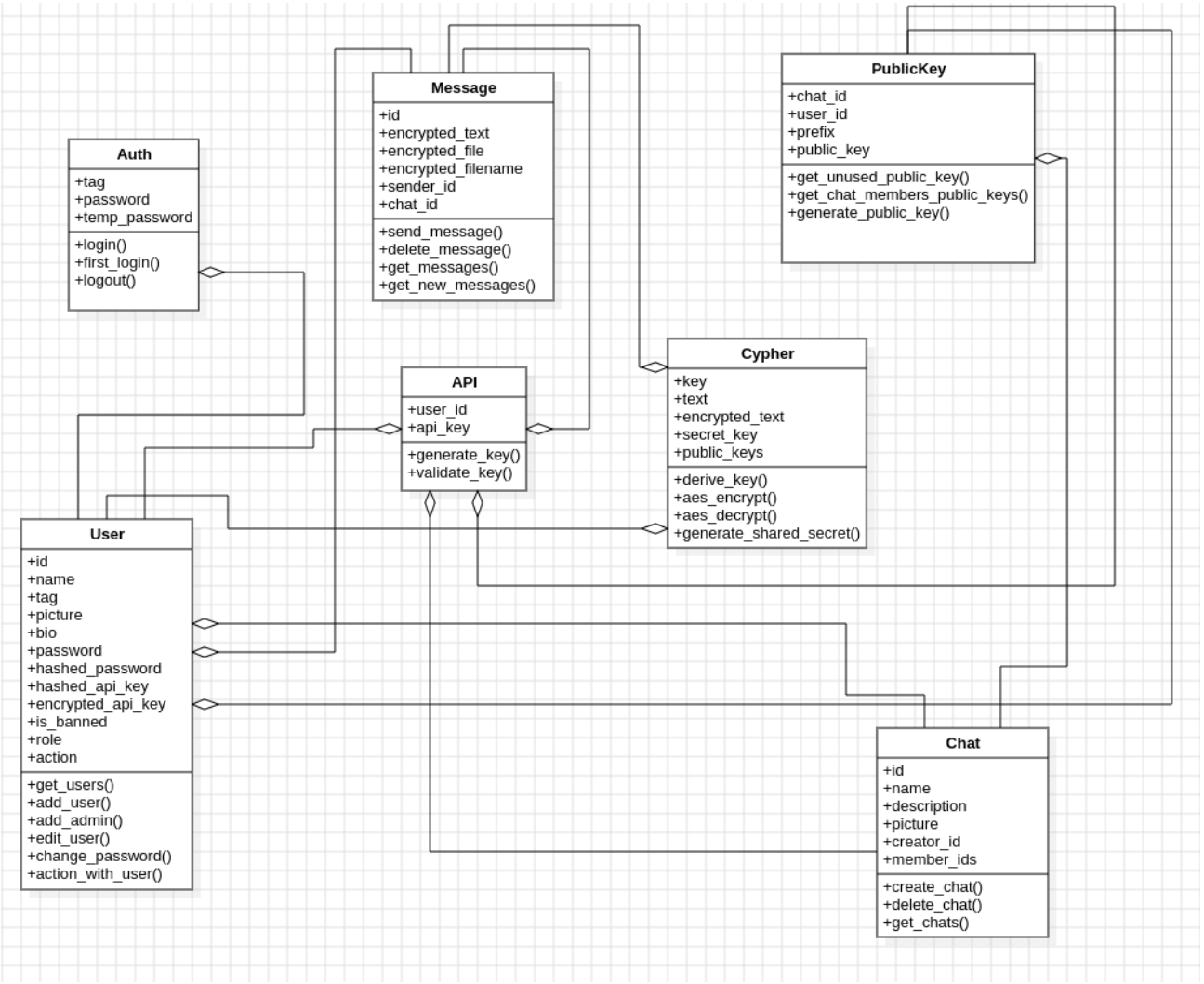

Рис 2.5 – Діаграма класів

Джерело: розроблено автором.

Діаграма об'єктів займає особливе місце в арсеналі інструментів структурного моделювання, оскільки слугує важливим проміжним ланкою між абстрактною концепцією класу та його конкретною реалізацією у вигляді програмного коду. На відміну від діаграми класів, яка оперує абстрактними категоріями, діаграма об'єктів працює з конкретними екземплярами цих категорій у певний момент часу.

Методологічна цінність діаграм об'єктів полягає в їхній здатності:

- конкретизувати абстрактні концепції діаграми класів;
- демонструвати стан системи у конкретний момент часу;

- валідувати коректність діаграми класів;
- слугувати мостом між проектуванням та реалізацією.

Для корпоративного месенджера цей зв'язок особливо важливий, оскільки дозволяє:

- конкретизувати складні відносини між учасниками чатів;
- відобразити реальні екземпляри повідомлень;
- продемонструвати роботу механізмів безпеки.

Така діаграма наочно демонструє, які конкретні об'єкти беруть участь у процесі та які значення вони містять у конкретний момент часу.

На рис 2.6 зображена діаграма об’єктів.

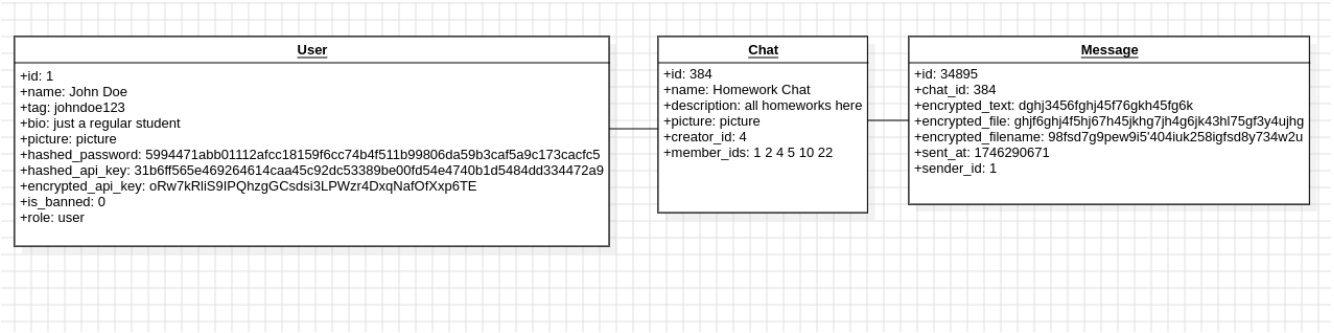

Рис 2.6 – Діаграма об’єктів

Джерело: розроблено автором.

Діаграма компонентів займає особливе місце в архітектурному проектуванні корпоративного месенджера, оскільки дозволяє перейти від абстрактних логічних моделей до фізичної організації системи. На відміну від діаграм класів та об'єктів, які описують логічну структуру, діаграма компонентів фокусується на фізичній реалізації системи у вигляді взаємопов'язаних модулів.

На рис 2.7 зображена діаграма компонентів.

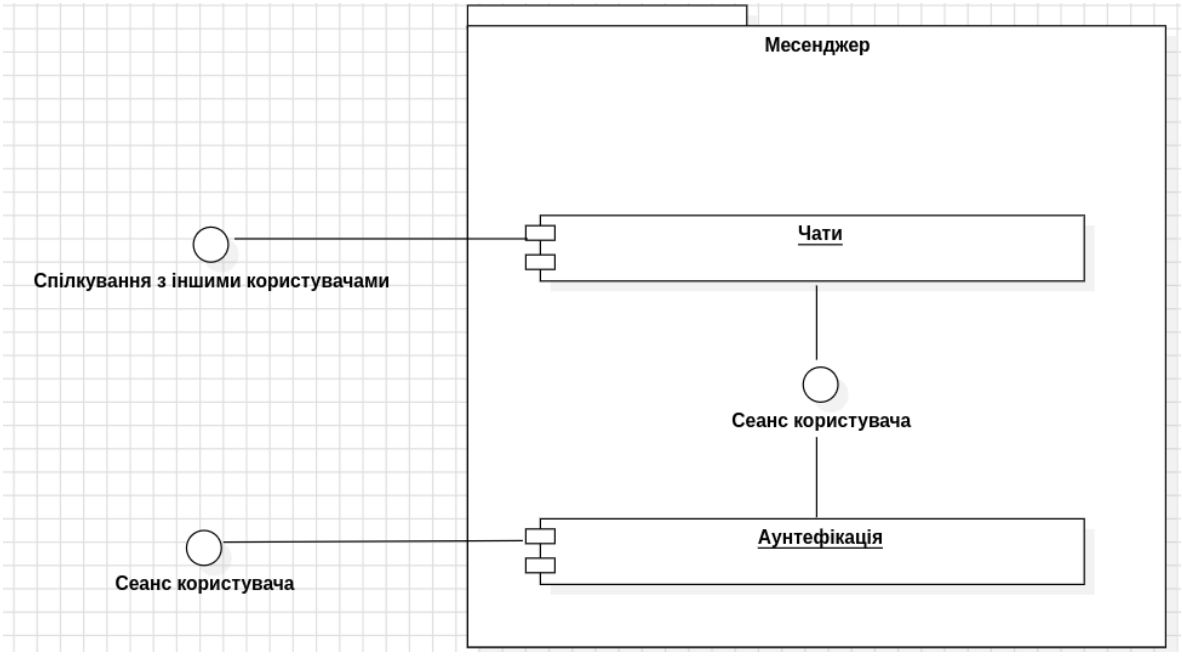

Рис 2.7 – Діаграма компонентів

Джерело: розроблено автором.

2.3 Опис архітектури продукту

Архітектура програмного продукту базується на модульному підході, що дозволяє ефективно розподіляти функціональність між різними компонентами. Система розроблена з використанням мови програмування Python та низки стандартних і сторонніх бібліотек для забезпечення безпеки, обробки запитів, роботи з даними та інших задач. Нижче наведено опис просторів імен класів і методів, що використовуються в програмному продукті.

Програмний продукт використовує такі бібліотеки:

- FastAPI – відповідає за обробку HTTP-запитів, надає API для взаємодії з клієнтською частиною;
- cryptography – забезпечує функціонал для шифрування та дешифрування даних (AES, ECDH);
- os – використовується для роботи з файловою системою (наприклад, зчитування конфігураційних файлів);

- bcrypt – призначений для безпечного хешування паролів перед збереженням у базі даних;
- time – відповідає за фіксацію часових міток (наприклад, час надсилання повідомлення);
- uuid – генерує унікальні ідентифікатори для користувачів, чатів, повідомлень тощо;
- sqlite3 – забезпечує взаємодію з базою даних SQLite для зберігання інформації;
- traceback – використовується для логування системних помилок у разі виникнення винятків;
- ecdsa – реалізує протокол ECDH (Elliptic Curve Diffie-Hellman) для обміну криптографічними ключами.

Програмний продукт містить такі основні класи, які групують логіку за функціональними модулями:

Клас User відповідає за управління користувачами системи.

- get_users() – отримання списку користувачів;
- change_password() – зміна пароля облікового запису;
- action_with_user() – виконання дій над користувачем (блокування, розблокування);
- edit_user() – редагування даних користувача.

Клас Message відповідає за роботу з повідомленнями.

- get_messages() – отримання повідомлень із чату;
- send_message() – надсилання нового повідомлення;
- delete_message() – видалення повідомлення.

Клас Chat забезпечує функціонал для роботи з чатами.

- get_chats() – отримання списку чатів користувача;
- create_chat() – створення нового чату;
- delete_chat() – видалення чату.

Клас PublicKey відповідає за управління публічними ключами.

- add_public_key() – додавання нового публічного ключа;

- get_chat_public_keys() – отримання ключів, пов’язаних із чатом;
- get_free_public_key() – отримання вільного ключа для нового чату.

Клас Auth забезпечує автентифікацію та авторизацію користувачів.

- login() – вхід у систему;
- first_login() – перший вхід із зміною тимчасового пароля;
- logout() – вихід із системи.

Клас Cypher відповідає за шифрування та дешифрування даних.

- aes_encrypt() – шифрування повідомлень за допомогою AES;
- aes_decrypt() – дешифрування повідомлень.

Клас API надає методи для роботи з API-ключами.

- generate_key() – генерація нового API-ключа;
- validate_key() – перевірка валідності API-ключа.

Архітектура програмного продукту побудована наступним чином:

- Клієнт взаємодіє з FastAPI через HTTP-запити;
- FastAPI маршрутизує запити до відповідних методів класів (User, Message, Chat тощо);
- для роботи з даними використовується SQLite, а для шифрування – Cryptography та ecdsa;
- кожен модуль має чітко визначені межі відповідальності, що спрощує підтримку та масштабування.

Така структура забезпечує гнучкість, безпеку та легкість у розширенні функціоналу.

Висновки до розділу 2

У другому розділі було розглянуто процес проєктування програмного продукту з використанням UML-діаграм. Кожна з них відіграла важливу роль у моделюванні різних аспектів системи.

Діаграма прецедентів допомогла визначити основні сценарії взаємодії користувача з месенджером, діаграма діяльності — структурувати логіку

виконання дій, а діаграма послідовності — проілюструвати динаміку взаємодії об’єктів у часі.

Для відображення архітектури програмного забезпечення використано діаграму компонентів, яка дозволила наочно структурувати зв’язки між модулями. Завдяки цим інструментам вдалося повноцінно описати функціональну та логічну структуру системи ще до етапу реалізації, що є важливою умовою її подальшої стабільної роботи.

Використання діаграм забезпечило цілісне бачення архітектури месенджера, включаючи поведінку системи, взаємодію користувачів та основні компоненти програмної реалізації. Це створює надійну основу для ефективного етапу розробки

РОЗДІЛ 3
РЕАЛІЗАЦІЯ

3.1 Реалізація та конструювання програмного продукту

Корпоративний месенджер було розроблено у середовищі Visual Studio Code з використанням мови програмування Python. Вибір даного інтегрованого середовища розробки обумовлений його низьким навантаженням на систему, широкою підтримкою розширень та зручністю використання.

Мова програмування Python була обрана через її гнучкість, велику кількість доступних бібліотек та наявність досвіду роботи з нею. У процесі розробки застосовувалися такі бібліотеки:

- FastAPI – для обробки HTTP-запитів;
- cryptography – для шифрування даних;
- os – для роботи з файловою системою;
- bcrypt – для хешування паролів;
- time – для фіксації часових міток;
- uuid – для генерації унікальних ідентифікаторів;
- sqlite3 – для взаємодії з базами даних;
- traceback – для логування системних помилок;
- ecdsa – для реалізації протоколу ECDH.

Система використовує три бази даних SQLite, що дозволяє ефективно керувати інформацією та забезпечує її розподілене зберігання.

Головна база даних (main.db) містить інформацію про користувачів та чати. Таблиця users включає наступні поля:

- id – унікальний ідентифікатор користувача;
- name – ім'я користувача;
- tag – текстовий ідентифікатор для авторизації;
- bio – опис профілю;
- temp_password – тимчасовий пароль, що видається адміністратором;

- hashed_password – хешований основний пароль;
- encrypted_api_key – зашифрований API-токен;
- hashed_api_key – хешований API-токен для валідації запитів;
- created_at – час створення облікового запису;
- role – роль користувача;
- is_banned – статус блокування.

Таблиця chats містить:

- id – унікальний ідентифікатор чату;
- name – назва чату;
- description – опис чату;
- picture – зображення чату;
- creator_id – ідентифікатор користувача, який створив чат.

Таблиця chat_members складається з:

- user_id – ідентифікатор учасника чату;
- chat_id – ідентифікатор відповідного чату.

База даних чатів (chats.db) призначена для зберігання повідомлень. Кожен чат має окрему таблицю chat_X, де X – ідентифікатор чату. Поля таблиці chat_X:

- id – унікальний ідентифікатор повідомлення;
- encrypted_message_text – зашифрований текст повідомлення;
- encrypted_file – зашифрований файл (якщо є);
- encrypted_filename – зашифрована назва файлу;
- sent_at – час відправки повідомлення (у форматі Unix timestamp);
- sender_id – ідентифікатор відправника.

База даних публічних ключів (public_keys.db) використовується для зберігання публчних ключів, необхідних для створення спільного секрету. Кожен користувач має окрему таблицю keys_X, де X – його ідентифікатор. Поля таблиці keys_X:

- prefix – префікс для генерації ключів;
- public_key – публічний ключ;

- chat_id – ідентифікатор чату, до якого прив'язаний ключ.

Така структура баз даних забезпечує ефективне управління даними, безпеку зберігання інформації та масштабованість системи.

3.2 Тестування програмного продукту

Функціональне тестування є одним із ключових етапів перевірки якості програмного забезпечення, основним завданням якого є верифікація відповідності системи до вимог та очікувань замовника.

Функціональне тестування не лише допомагає виявити помилки в логіці роботи програми, але й забезпечує її стабільність перед релізом, що робить його невід’ємною частиною сучасних процесів QA[10]. У подальшому матеріалі ми детально дослідимо кожен із цих елементів, а також наведемо практичні приклади їх застосування.

Далі будуть приведені результати функціонального тестування авторизації користувача. На рис. 3.1 наведено приклад помилки, коли користувач намагається увійти в обліковий запис, не заповнюючи поля для входу.

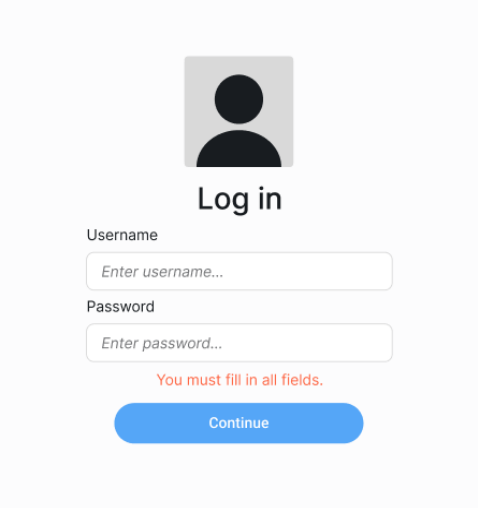

Рис 3.1 – Помилка при вході в обліковий запис при пустих полях

Джерело: розроблено автором.

На рис 3.2 наведено приклад помилки, при введенні неправильного логіну або паролю.

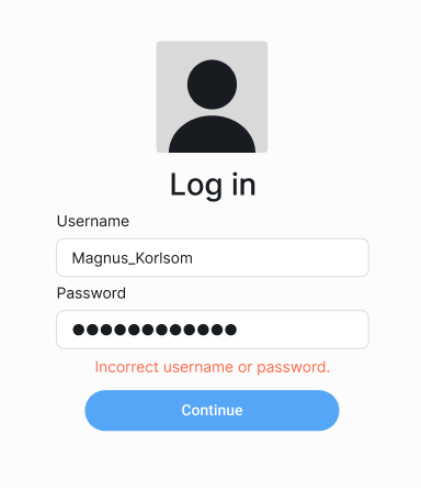

Рис 3.2 – Помилка при вході в обліковий запис при неправильному логіні або паролю

Джерело: розроблено автором.

На рис 3.3 наведено приклад помилки, коли користувач ввів правильні дані для входу, але його обліковий запис заблокований.

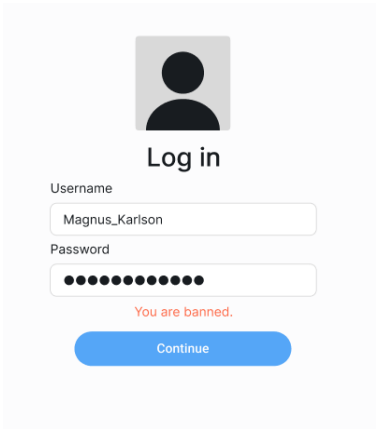

Рис 3.3 – Помилка при вході в обліковий запис з заблокованим обліковим записом

Джерело: розроблено автором.

Також було протестовано введення постійного паролю, коли користувач в перший раз заходив у свій обліковий запис по тимчасовому паролю, виданим адміністратором. Під час тестування перевірялися такі помилки, як:

- збіг паролів (перевірка, чи співпадають введені паролі);
- заповненість усіх обов'язкових полів.

Далі будуть приведені результати функціонального тестування редагування профіля користувача. На рис 3.4 зображено помилку, що виникає при редагуванні свого облікового запису із спробою виставити собі username, котрий вже використовується іншим користувачем.

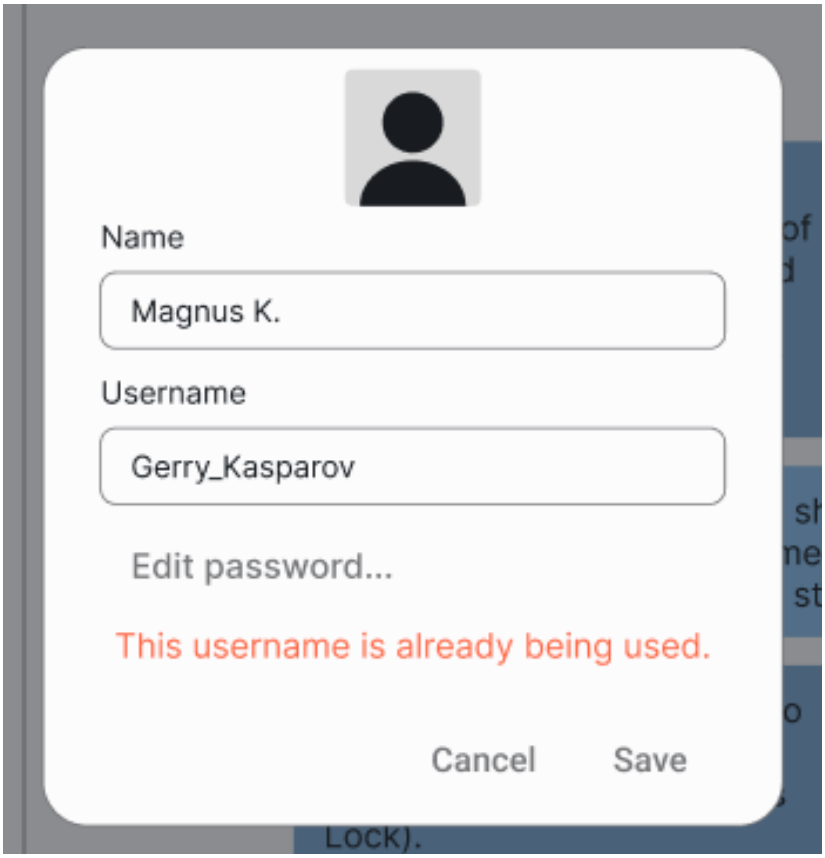

Рис 3.4 – Помилка, що виникає при редагуванні свого облікового запису із спробою виставити собі username, котрий вже використовується іншим користувачем.

Джерело: розроблено автором.

На рис 3.5 зображено помилку, що виникає при редагуванні свого профілю і одне з полів залишається пустим.

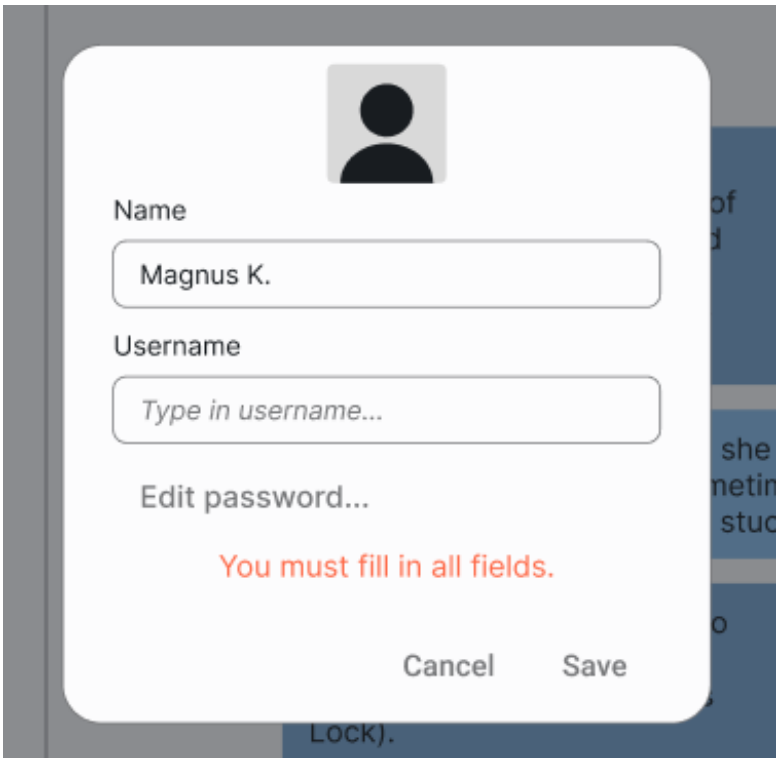

Рис 3.5 – Помилка, що виникає при редагуванні свого профілю і одне з полів залишається пустим.

Джерело: розроблено автором.

Було також протестовано зміну паролю користувачем. Під час тестування перевірялися такі помилки, як:

- збіг нових паролів (перевірка, чи співпадають введені нові паролі);
- заповненість усіх обов'язкових полів;
- правильність введення поточного паролю облікового запису.

Далі будуть приведені результати функціонального тестування створення користувачів. На рис 3.6 зображена помилка, що виникає під час створення користувача, але введений username вже зайнятий іншим користувачем.

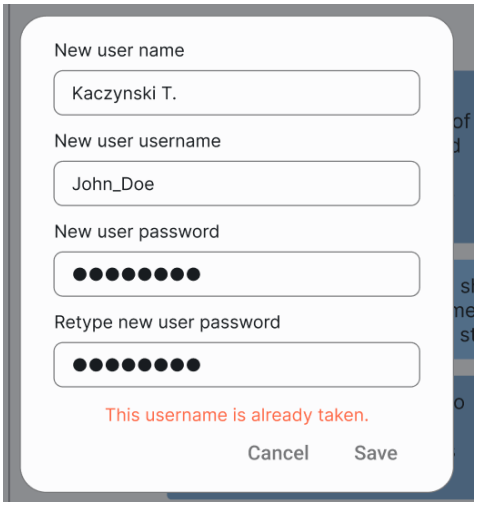

Рис 3.6 – Помилка, що виникає під час створення користувача, але введений username вже зайнятий іншим користувачем.
Джерело: розроблено автором.

На рис 3.7 зображена помилка, що виникає під час створення користувача, але два поля для вводу паролю не співпадають.

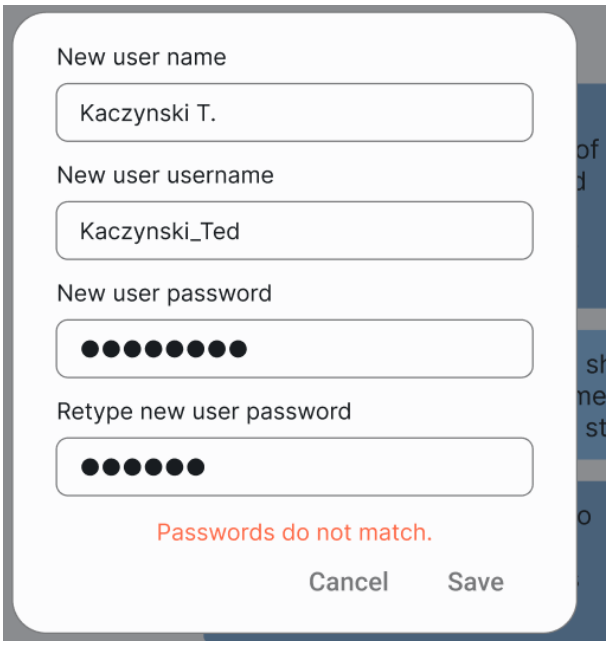

Рис 3.7 – Помилка, що виникає під час створення користувача, але два поля для вводу паролю не співпадають.

Джерело: розроблено автором.

Додавання адміністраторів тестувалося аналогічно додаванню користувачів.

Далі будуть приведені результати функціонального тестування створення чатів. На рис 3.8 зображена помилка, що виникає під час створення чату, а обраний користувач не може зайти в чат.

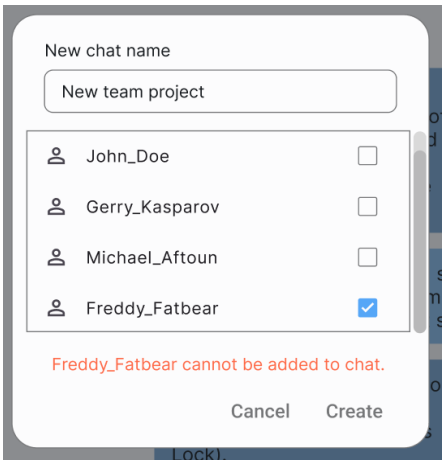

Рис 3.8 – Помилка, що виникає під час створення чату, а обраний користувач не може зайти в чат.

Джерело: розроблено автором.

Модульне тестування, або unit-тестування, є фундаментальним рівнем перевірки якості програмного забезпечення, основним завданням якого є тестування окремих компонентів системи (функцій, методів, класів) у ізоляції від решти коду. На відміну від інтеграційного чи системного тестування, unit-тести зосереджені на перевірці коректності роботи найменших логічних одиниць програми, що дозволяє виявляти помилки на ранніх етапах розробки[11].

На рис 3.9 зображено результати тестування класу Auth, де були протестовані методи login() і first_login().

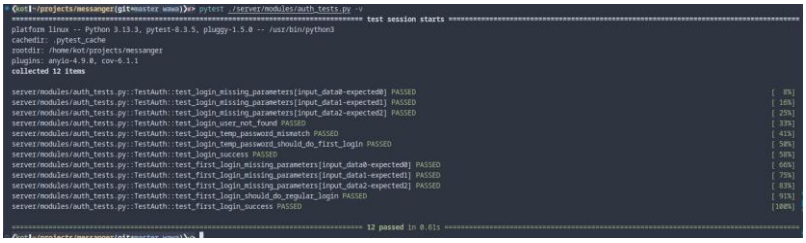

Рис 3.9 – Результати тестування класу Auth.

Джерело: розроблено автором.

На рис 3.10 зображено результати тестування класу User, де були протестовані методи change_password(), add_user(), add_admin() і action_with_user().

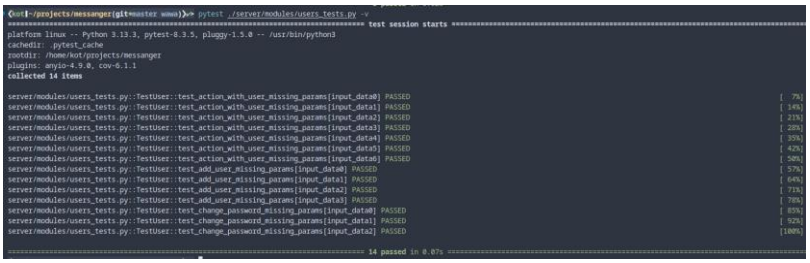

Рис 3.10 – Результати тестування класу User.

Джерело: розроблено автором.

На рис 3.11 зображено результати тестування класу API, де були протестовані методи generate_key() і validate_key().

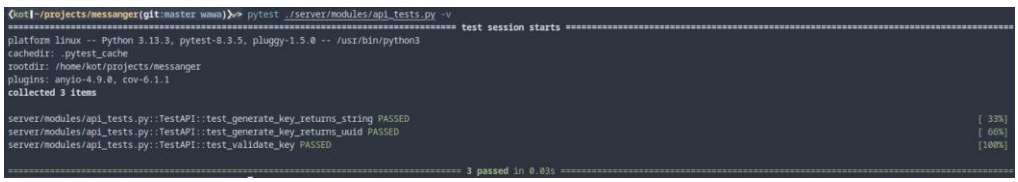

Рис 3.11 – Результати тестування класу API.

Джерело: розроблено автором.

На рис 3.12 зображено результати тестування класу Cypher, де були протестовані методи aes_encrypt() і aes_decrypt().

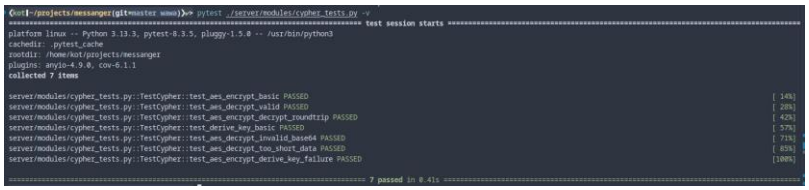

Рис 3.12 – Результати тестування класу Cypher.

Джерело: розроблено автором.

На рис 3.13 зображено результати тестування класу Chat, де були протестовані методи get_chats(), create_chat() і delete_chat().

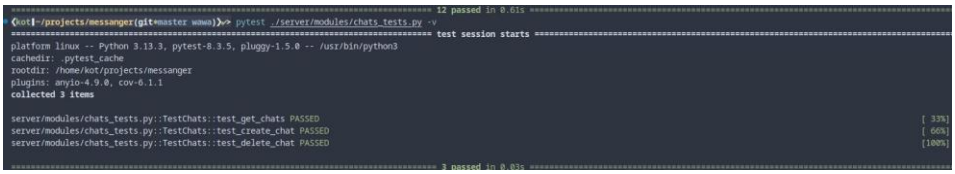

Рис 3.13 – Результати тестування класу Chat.

Джерело: розроблено автором.

На рис 3.14 зображено результати тестування класу Message, де були протестовані методи get_messages(), send_message() і delete_message().

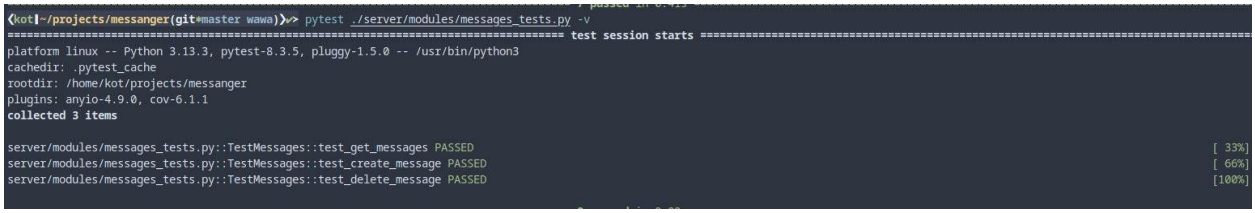

Рис 3.14 – Результати тестування класу Message.
Джерело: розроблено автором.

Проведений аналіз результатів модульного тестування демонструє його значну ефективність у забезпеченні якості програмного забезпечення. Тестування окремих класів підтвердило низку ключових переваг даного підходу.

3.3 Використання програмного продукту

У даному підрозділі розглядається процес використання програмного продукту – корпоративного месенджера. Основна увага приділяється налаштуванню серверної частини, її запуску та подальшій роботі з клієнтським інтерфейсом.

Спочатку буде висвітлено встановлення та конфігурування серверного середовища, включаючи необхідні залежності та параметри для коректної роботи системи. Далі розглянемо процедуру запуску сервера.

Після цього увага буде зосереджена на клієнтській частині: буде проаналізовано інтерфейс месенджера, його основні функціональні можливості та взаємодію з сервером. Це дозволить оцінити зручність користування та ефективність запропонованого рішення.

Для подальшої роботи потрібно буде виконати git clone репозиторію корпоративного месенджеру. Після чого відкриваємо термінал у директорії, де знаходиться репозиторій. Для підготовки і запуску серверної частини виконаємо наступні команди:

- pip3 install -r requirements.txt – встановлюємо необіхдні залежності для роботи з серверною частиною;
- cd server/ - переходимо у директорію серверної частини корпоративного месенджеру;

- python3 –m init – запускаємо файл, котрий створить необхідну файлову структуру, підготує бази даних і створить обліковий запис Owner. У процесі виконання він спитає, чи потрібно генерувати самопідписані SSL сертифікати;
- python3 –m main – після успішної підготовки серверної частини, запускаємо сам сервер корпоративного месенджеру.

Після успішного налаштування і запуску серверної частини переходимо до клієнтського інтерфейсу. Виконуємо наступні команди:

- pip3 install –r requirements.txt – встановлюємо необхідні залежності для роботи з клієнтською частиною;
- npm install – встановлюємо додаткові необхідні залежності для роботи з клієнтською частиною;
- npm start – після встановлення необхідних залежностей, запускаємо сам клієнт корпоративного месенджеру.

Першим чином відкривається вікно авторизації, де потрібно буде ввести свій логін і пароль (приклад на рис 3.15).

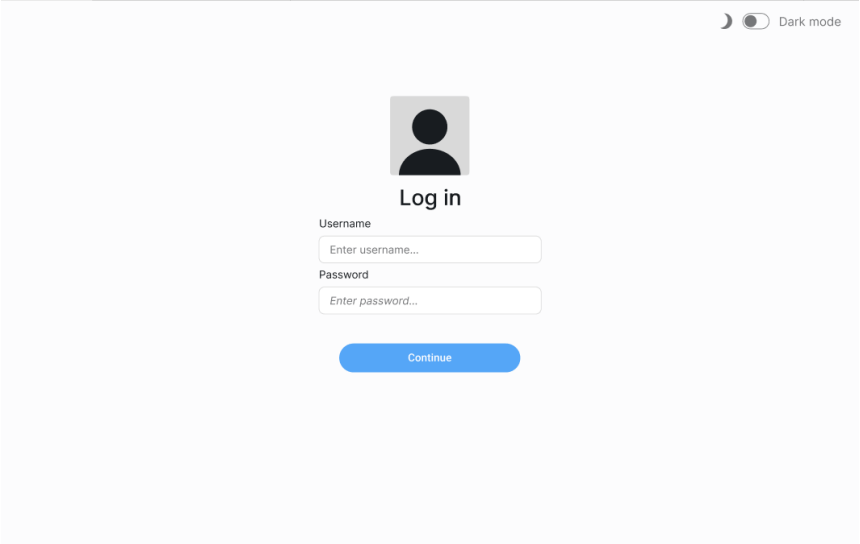

Рис 3.15 – Екран логіну з клієнтської сторони корпоративного месенджеру. Джерело: розроблено автором.

Після введення правильних даних для входу, якщо це перший вхід і роль користувача не Owner, то відкриється додаткове вікно (рис 3.16), де користувач повинен буде ввести свій основний пароль для подальшої роботи з месенджером.

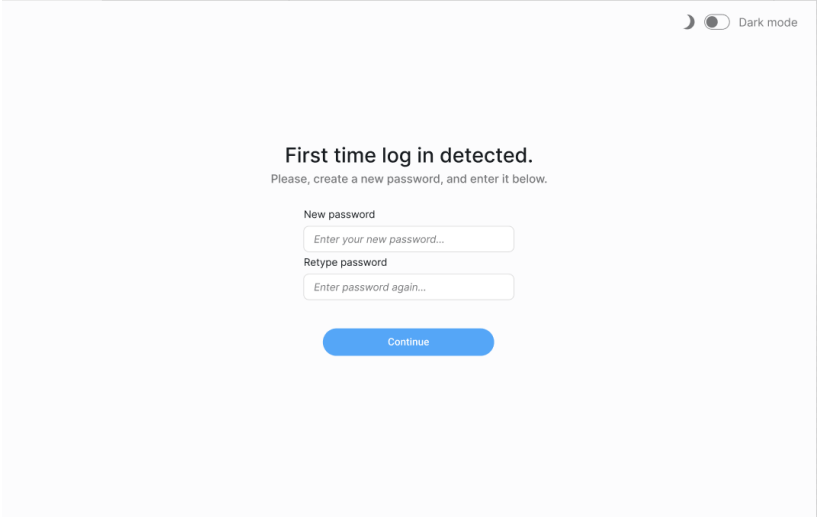

Рис 3.16 – Екран створення основного паролю при першому вході в обліковий запис з клієнтської сторони корпоративного месенджеру.

Джерело: розроблено автором.

Після успішного входу в обліковий запис користувача зустрічає головний екран (рис 3.17). На цьому екрані можна побачити список чатів, у котрих є користувач, список повідомлень обраного чату, кнопку налаштувань і кнопку створення нового чату.

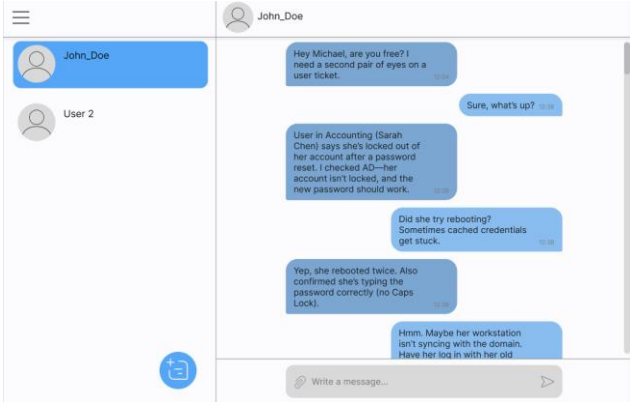

Рис 3.17 – Головний екран з клієнтської сторони корпоративного месенджеру.

Джерело: розроблено автором.

Якщо натиснути на кнопку налаштувань, відкриється випадаючий список (рис 3.18). У списку є такі кнопки як:

- перемикання світлої і темної теми;
- редагування свого облікового запису;
- створення користувача (якщо роль адміністратор і вище);
- створення адміністратора (якщо роль Owner);
- вихід з облікового запису.

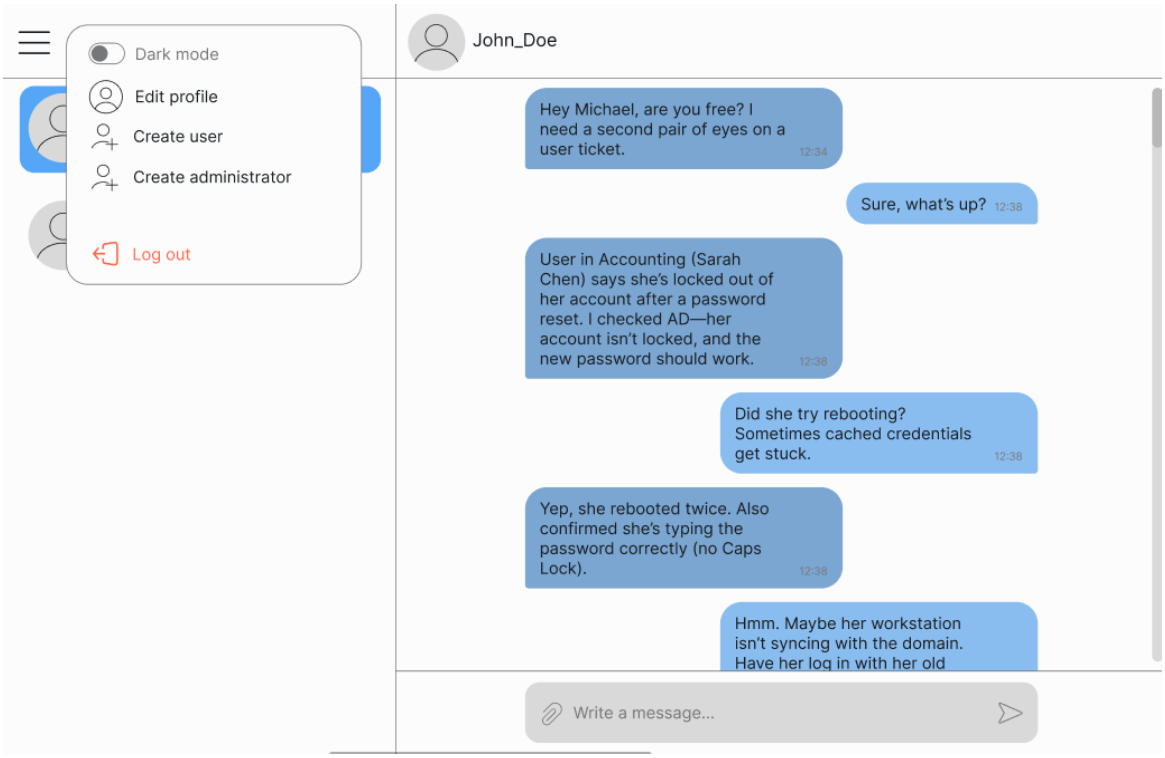

Рис 3.18 – Випадаючий список налаштувань з клієнтської сторони корпоративного месенджеру.

Джерело: розроблено автором.

У вікні редагування свого облікового запису (рис 3.19) користувач може змінити ім’я, логін, а також там є кнопка для зміни паролю, котра буде розглянута пізніше.

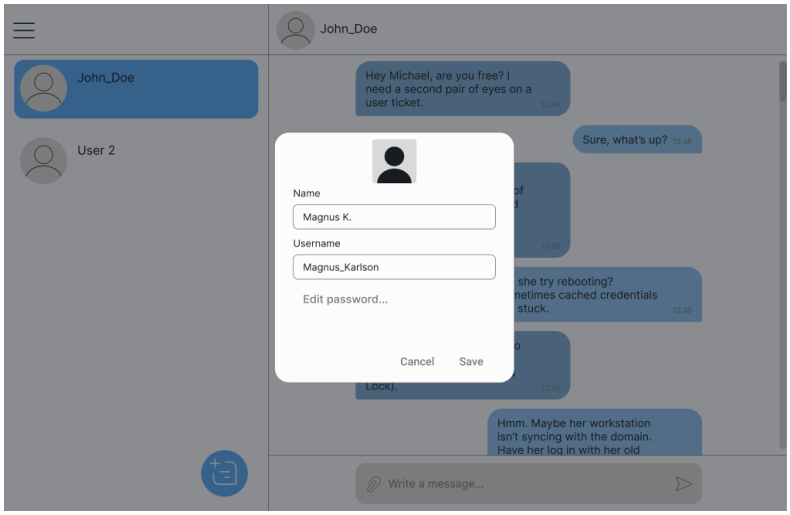

Рис 3.19 – Меню редагування облікового запису користувача з клієнтської сторони корпоративного месенджеру.

Джерело: розроблено автором.

Якщо натиснути на кнопку зміни пароля, відкриється нове відповідне меню (рис 3.20). Для зміни паролю користувачу потрібно буде ввести свій поточний пароль, новий пароль і підтвердити свій новий пароль.

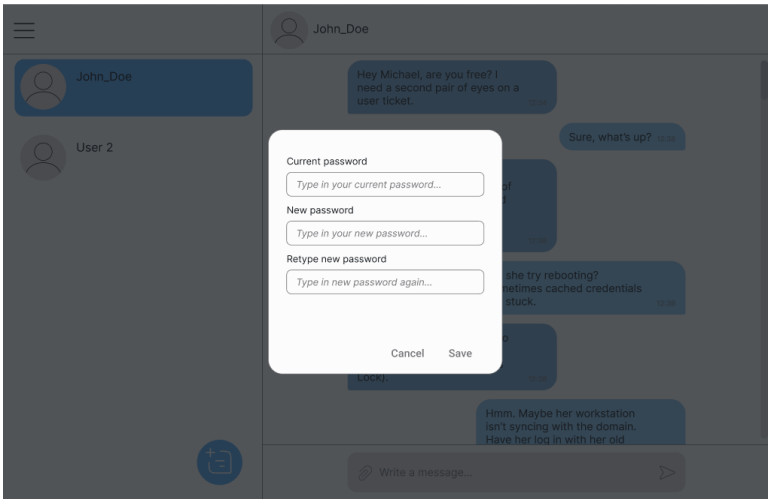

Рис 3.20 – Меню зміни пароля користувача з клієнтської сторони корпоративного месенджеру.

Джерело: розроблено автором.

Повертаючись до основного меню налаштувань, якщо роль користувача адміністратор або Owner, у меню буде кнопка для створення користувача. Якщо натиснути на неї, відкриється меню створення користувача (3.21). Для створення користувача адміністратору потрібно буде ввести ім’я, логін і тимчасовий пароль майбутнього користувача.

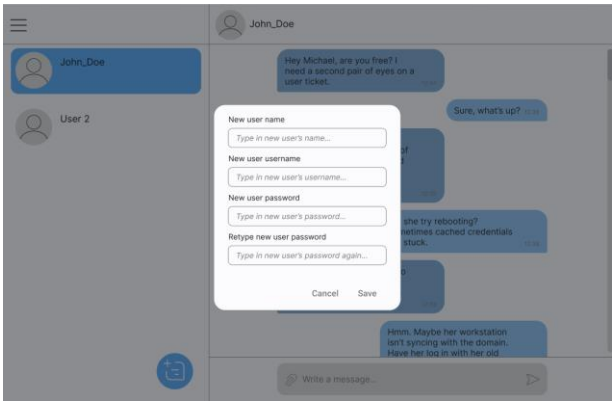

Рис 3.21 – Меню створення користувача з клієнтської сторони корпоративного месенджеру.

Джерело: розроблено автором.

Якщо роль користувача Owner, тоді у меню налаштувань буде ще кнопка для створення користувача з роллю адміністратора (рис 3.22). Користувачу з роллю Owner потрібно буде ввести ім’я, логін і тимчасовий пароль майбутнього адміністратора.

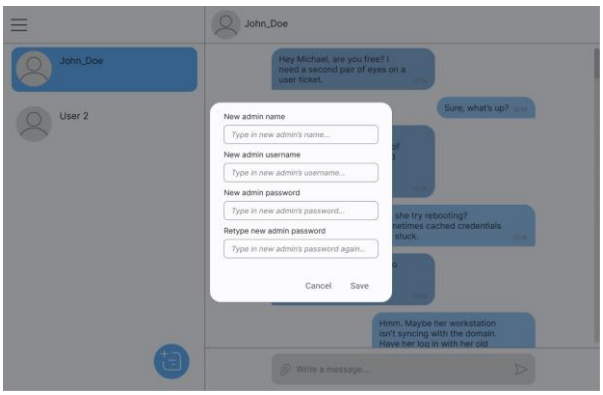

Рис 3.22 – Меню створення адміністратора з клієнтської сторони корпоративного месенджеру.

Джерело: розроблено автором.

В самому низу меню налаштувань знаходиться кнопка для виходу з облікового запису. При натисканні на неї, поточний сеанс призупиняється і користувача переносить на екран логіну.

Внизу, у списку чатів, є кнопка для створення нового чату. Натиснувши на неї, відкриється меню створення чату (рис 3.23). Для створення чату потрібно буде вказати назву чату і обрати із списку користувачів майбутніх учасників чату.

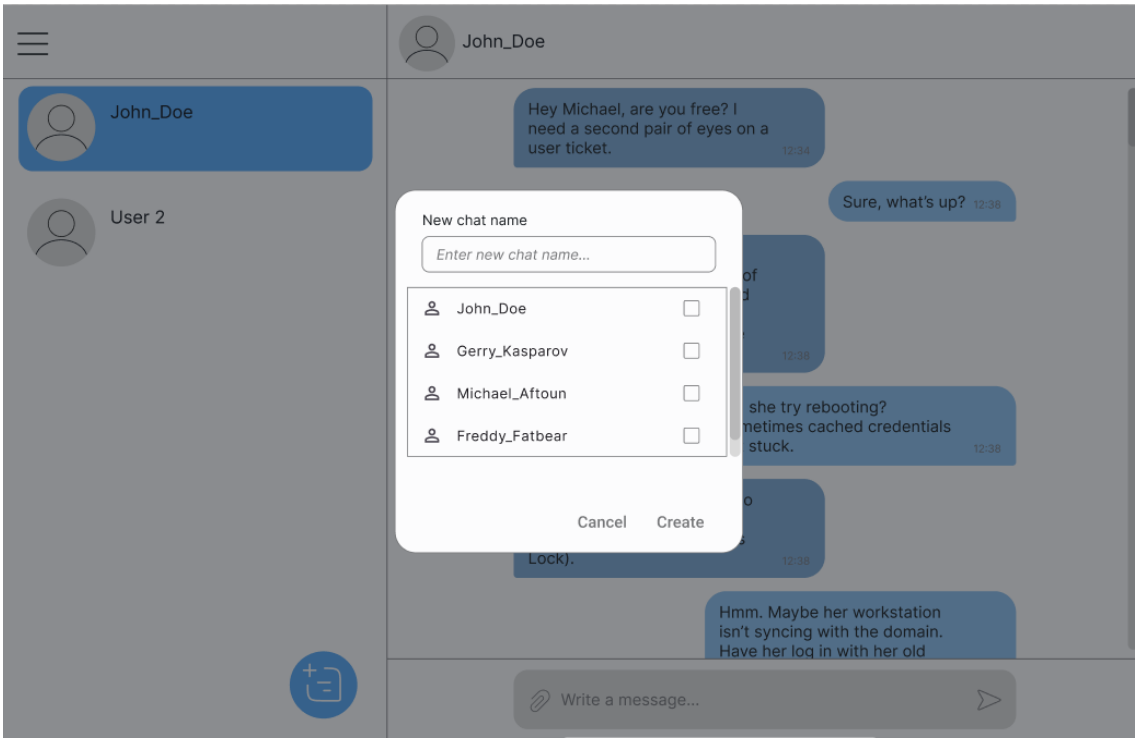

Рис 3.23 – Меню створення чату з клієнтської сторони корпоративного месенджеру.

Джерело: розроблено автором.

У чатах користувач може обмінюватися повідомленнями з іншими користувачами. Також користувач може видаляти свої чати і свої повідомлення в чатах. Для цього потрібно клікнути правою кнопкою миші і відкриється відповідне меню (рис 3.24, рис 3.25).

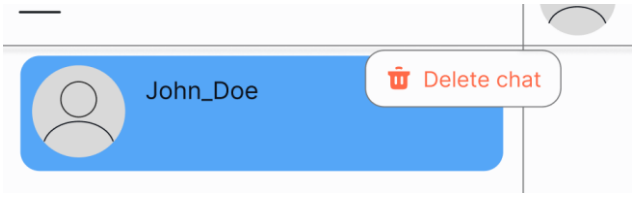

Рис 3.24 – Меню видалення чату з клієнтської сторони корпоративного месенджеру.

Джерело: розроблено автором.

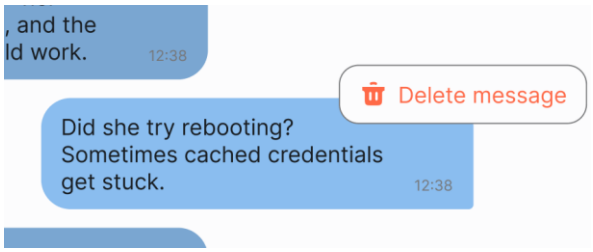

Рис 3.25 – Меню видалення повідомлення з клієнтської сторони корпоративного месенджеру.

Джерело: розроблено автором.

Таким чином було продемонстровано весь функціонал корпоративного месенджеру з клієнтської сторони.

Висновки до розділу 3

У процесі виконання кваліфікаційної роботи проведено комплексне вивчення та реалізацію механізмів шифрування повідомлень у месенджері з відкритим вихідним кодом.

Застосування алгоритму AES-256 у режимі CBC дозволило впровадити наскрізний захист даних і поєднати теоретичні знання з практичною реалізацією криптографічного захисту.

Реалізовано клієнтську та серверну частини програмного продукту з використанням мови Python і фреймворку FastAPI. Це дало змогу створити гнучку архітектуру, адаптовану для локального розгортання в організаціях без використання зовнішніх сервісів.

Для зберігання даних використано локальні бази SQLite, що забезпечили ефективне управління інформацією про користувачів, повідомлення, чати й публічні ключі. Такий підхід сприяє збереженню конфіденційності та підвищенню контролю над інформацією.

Інтеграція наскрізного шифрування передбачає динамічну генерацію AES-ключів для кожного чату, а також використання ECDH-протоколу для створення спільних секретів. Це підвищило рівень криптографічного захисту системи при обміні повідомленнями.

Проведено функціональне й модульне тестування основних класів системи. Виявлені помилки було усунуто, а тестові сценарії підтвердили відповідність функціоналу вимогам і коректність роботи всіх модулів.

Розроблений програмний продукт може бути рекомендований для внутрішнього використання в організаціях, що потребують захищеного спілкування з повним контролем над даними.

Результати роботи представлені на конференції “V наукова конференція «сучасний менеджмент організації: витоки, реалії та перспективи розвитку 2025»” у секції 4 «Проблематика інформаційного менеджменту, гнучких методологій управління та економіко-математичне моделювання»[12].

ВИСНОВКИ

У рамках проведеного дослідження було отримано низку ключових результатів, що стосуються реалізації безпечного корпоративного месенджера. Головним науковим досягненням став поглиблений аналіз механізмів шифрування даних, зокрема скрізного застосування алгоритму AES для забезпечення конфіденційності передачі інформації. Це дозволило не лише закріпити теоретичні знання, а й сформувати практичні навички роботи з криптографічними інструментами.

Практичним результатом дослідження є розробка функціонального месенджера на Python, який інтегрує AES-шифрування на всіх етапах комунікації. Система забезпечує безпечний обмін повідомленнями, зберігаючи дані у локальному сховищі організації, що виключає зовнішній доступ до конфіденційної інформації. Порівняльний аналіз існуючих рішень підтвердив конкурентоспроможність розробки у ніші текстових корпоративних комунікацій.

Серед перспектив розвитку проєкту – впровадження функціоналу для аудіо- та відеодзвінків із збереженням рівня шифрування, що розширить сферу застосування продукту. Наразі ж система може бути рекомендована компаніям, які потребують мінімалістичного, але надійного інструменту для внутрішнього текстового спілкування. Її перевагою є повний контроль над даними, відсутність залежності від зовнішніх сервісів та адаптивність до специфічних вимог організацій.

Таким чином, робота довела ефективність обраного підходу до захисту комунікацій та заклала основу для подальшого вдосконалення продукту у відповідності до сучасних викликів кібербезпеки.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Crawford J. 10 Modern Ways to Communicate Online in the Digital Age. Tech Research Online. URL: http://www.techresearchonline.com/blog/new-ways-to-communicate-digitally (дата звернення: 21.04.2025).
2. Popular Messenger Users' Data Leaked and Students' Personal Information Exposed – SearchInform. URL: https://searchinform.com/blog/2023/12/5/popular-messenger-users-data-leaked-and-students-personal-information-exposed (дата звернення: 21.04.2025).
3. Internet. Our World in Data. URL: https://ourworldindata.org/internet (дата звернення: 21.04.2025).
4. mIRC: History of IRC. mIRC: Internet Relay Chat client. URL: https://www.mirc.com/history.html (дата звернення: 21.04.2025).
5. Irssi – Windows Installer. Irssi. URL: https://irssi.org/2008/01/30/windows-installer (дата звернення: 13.03.2025).
6. Dino. Communicating happiness. URL: https://dino.im/ (дата звернення: 13.03.2025).
7. Productivity-boosting E2EE collaboration and messaging for enterprises. Element | Secure collaboration and messaging. URL: https://element.io/app (дата звернення: 13.03.2025).
8. Kasak D., Callahan D., Hodgson M. Disclosing CVE-2021-40823 and CVE-2021-40824: E2EE vulnerability in multiple Matrix clients. Matrix.org. URL: https://matrix.org/blog/2021/09/13/vulnerability-disclosure-key-sharing/ (дата звернення: 21.04.2025).
9. Smith L. Matrix vs. XMPP. Luke's Webpage. URL: https://lukesmith.xyz/articles/matrix-vs-xmpp/ (дата звернення: 21.04.2025).
10. What is Functional Testing? Types & Examples. URL: https://www.guru99.com/functional-testing.html (дата звернення: 25.04.2025)

11. Unit Testing Basics. URL: https://learn.microsoft.com/en-us/dotnet/core/testing/unit-testing-best-practices (дата звернення: 26.04.2025)
12. Конференція «Проблематика інформаційного менеджменту, гнучких методологій управління та економіко-математичне моделювання – 2025» [Електронний ресурс]. — Режим доступу: https://conf.krok.edu.ua/MMO/MMO-2025/paper/view/2785 (дата звернення 01.05.2025).