

Міністерство освіти і науки України
Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки

ПОЯСНЮВАЛЬНА ЗАПИСКА

до випускної кваліфікаційної роботи бакалавра
на тему: «Проект системи обміну повідомленням та
VoIP-зв'язку на основі Firebase та Asterisk»

Виконав: студент 4-го курсу групи КІ-21-2
спеціальності 123 «Комп'ютерна інженерія»

 Нікіта ЮХИМЧУК

Керівник: завідувач кафедри КН, Ph.D, доцент

 Марина ГРАФ

Рецензент: Доцент кафедри КІ та КБ, к.т.н., доцент

 Юрій БРОДСЬКИЙ

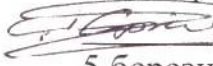
Житомир – 2025

Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки
Спеціальність 123 «Комп'ютерна інженерія»
Освітня програма «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ

Завідувач

кафедри комп'ютерної
інженерії та кібербезпеки

 Андрій ЄФІМЕНКО
5 березня 2025 р.

ЗАВДАННЯ

на випускню кваліфікаційну роботу бакалавра

Студент: Юхимчук Нікіта Олегович

Тема роботи: «Проект системи обміну повідомленням та VoIP-зв'язку на основі
Firebase та Asterisk»

затверджена Наказом університету від 3 березня 2025 р. №79/с

Термін здачі студентом закінченої роботи 13 червня 2025 р.

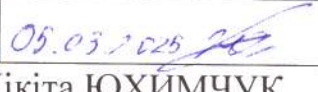
Вихідні дані для роботи:

- Процес функціонування системи захищеного обміну даними, текстовими повідомленнями та VoIP-зв'язку на основі Firebase та Asterisk.
- програмні та мережеві засоби для захищеного обміну текстовими повідомленнями та VoIP-зв'язком,
- хмарні сервіси Firebase та Asterisk,
- інші необхідні протоколи, бібліотеки та інструменти

Обладнання та програмне забезпечення:

1. Сервер: Windows Server 2022, Intel Core I5, NVidia GeForce GTX1050TI.
2. Ubuntu сервер: Asterisk 18, Python3.12, JetBrains Community edition
3. Комп'ютер: Intel Core I9 14900KF, Nvidia GeForce RTX4070TI.
4. Visual Studio, C#, .Net 6, Ozeki VoIP SDK.
5. Google Cloud Firebase

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1-3	Олексій ШЕЛУХА	05.03.2025 	05.03.2025 

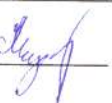
Студент  Нікіта ЮХИМЧУК

Керівник  Марина ГРАФ

Календарний план

№ з/п	Назва етапів випускної кваліфікаційної роботи	Терміни виконання етапів роботи	Примітка
1	Постановка задачі. Пошук, огляд та аналіз аналогічних розробок. Формування технічного завдання. Опрацювання літературних джерел.	2 березня 2025 – 5 квітня 2025	
2	Проектування структури системи	6 квітня 2025 – 17 квітня 2025	
3	Налаштування Asterisk	17 квітня 2025 – 20 квітня 2025	
4	Написання програмного коду	20 квітня 2025 – 30 квітня 2025	
5	Тестування	1 травня 2025 – 4 травня 2025	
6	Підготовка пояснювальної записки	5 травня 2025 – 30 травня 2025	

Студент  Нікіта ЮХИМЧУК

Керівник  Марина ГРАФ

АНОТАЦІЯ

Кваліфікаційна робота бакалавра, присвячена дослідженню методів безпечного обміну повідомленнями та VoIP-дзвінками за допомогою мережі інтернет сформованого в клієнт-серверну архітектуру. У роботі вирішується потреба у поєднанні розрізнених технологій та протоколів для об'єднання та спрощення використання користувачу в єдиний застосунок, та розробляється і впроваджується власне шифрування з ціллю забезпечення безпеки передачі даних.

Основні результати: створено проєкт клієнт-серверного застосунку з назвою "GenGo", для цього розроблено алгоритми для асинхронного обміну повідомленнями та VoIP зв'язку, алгоритм шифрування повідомлень, встановлено та налаштовано сервер VoIP зв'язку, реалізовано автоматизований скрипт створення та застосування авторизаційних даних нових Sip-акаунтів на серверній стороні за допомогою Python.

Практичне призначення кваліфікаційної роботи полягає у створенні проєкту системи обміну повідомленнями та VoIP-зв'язку на основі Firebase та Asterisk для можливості виключення самотійного локального створення Backend-застосунку, натомість розміщення його у хмарі для забезпечення процесу обміну текстовими та файловими повідомленнями, а також забезпечення VoIP-зв'язку на стороні клієнта. Створення серверу для забезпечення функціонування VoIP-дзвінків.

Результати роботи можуть використовуватись у особистих або корпоративних спілкуваннях між користувачами, а також слугувати основою для проведення подальших досліджень хмарних комунікаційних технологій.

Робота складається з вступу, трьох розділів, висновків, списку використаних джерел та додатків.

Кваліфікаційна робота бакалавра, має обсяг 99 сторінка, містить 48 рисунків, 6 таблиць, 3 додатки та список використаних джерел із 47 найменувань.

Ключові слова: проєкт системи, архітектура інтерфейсу прикладних програм, запити, шифрування, передача повідомлень, протоколи передачі даних.

					ККІКБ.КР.Б – 123 – 14 – ПЗ		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Юхимчук Н.О.		5.06.25	Проект системи обміну повідомленнями та VoIP-зв'язку на основі Firebase та Asterisk	Літ.	Арк.
Керівник		Граф М.С.		3.06.25			4
Реценз.		Броцький Ю.Б.		06.06.25			99
Н. контр.		Граф М.С.		5.06.25		ФІКТ, Гр. КІ-21-2	
Затверд.		Єфіменко А.А.		3.06.25			

ABSTRACT

Bachelor's Qualification Thesis is dedicated to the study of secure messaging and VoIP call methods over the Internet using a client-server architecture. The work addresses the need to integrate disparate technologies and protocols into a single user-friendly application, and it develops and implements a custom encryption mechanism to ensure the security of data transmission.

Main results: a client-server application project named "GenGo" was created. Algorithms for asynchronous message exchange and VoIP communication were developed, as well as a message encryption algorithm. A VoIP communication server was installed and configured, and an automated script was implemented using Python to generate and apply authorization data for new SIP accounts on the server side.

The practical purpose of this qualification thesis is to develop a messaging and VoIP communication system based on Firebase and Asterisk, enabling the replacement of a locally developed backend application with a cloud-hosted solution. This ensures the process of text and file message exchange, as well as VoIP communication on the client side. The server was developed to support the operation of VoIP calls.

The results of the work can be used for personal or corporate user communication and may serve as a foundation for further research in cloud communication technologies.

The thesis consists of an introduction, three chapters, conclusions, a list of references, and appendices.

The Bachelor's Qualification Thesis comprises 98 pages, includes 48 figures, 6 tables, 3 appendices, and a list of 47 references.

Keywords: system design, application program interface architecture, queries, encryption, message passing, data transfer protocols.

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		5

Зміст

ВСТУП	8
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ СИСТЕМИ ОБМІНУ ПОВІДОМЛЕННЯМИ ТА VoIP-ЗВ'ЯЗКУ	10
1.1 Аналіз існуючих методів та засобів обміну повідомлень та VoIP зв'язків....	10
1.2 Аналіз інструментів та технологій захищеного обміну даними.....	18
1.3 Аналіз мережевих протоколів для забезпечення обміну даними.	21
1.4. Аналіз технологій Firebase, Asterisk	23
1.5 Постановка технічного завдання.....	27
Висновки до 1 розділу	28
РОЗДІЛ 2. ПРОЄКТУВАННЯ СИСТЕМИ ОБМІНУ ПОВІДОМЛЕННЯМИ ТА VoIP-ЗВ'ЯЗКУ	30
2.1. Проєктування структури клієнт-серверної взаємодії та вимог до проєкту на основі UML-діаграм	30
2.2. Проєктування роботи застосунку.....	39
2.3. Математичне представлення шифрування повідомлень	45
Висновки до 2 розділу	48
РОЗДІЛ 3. РЕАЛІЗАЦІЯ, РОЗГОРТАННЯ ТА ТЕСТУВАННЯ СИСТЕМИ, ІНСТРУКЦІЯ З ПОРЯДКУ ВИКОРИСТАННЯ	49
3.1. Реалізація клієнт-серверної частини.....	49
3.1.1. Реалізація серверної частини	49
3.1.2. Реалізація клієнтської частини	58
3.2. Тестування клієнтської сторони.....	71
3.3. Тестування на фішингове вторгнення	78
3.4. Інструкція з експлуатації	80
Висновки до 3 розділу	87
ВИСНОВКИ	88
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	89
ДОДАТКИ	93

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ОС – операційна система

ПЗ – програмне забезпечення

IP- англ. Internet Protocol address, укр. інтернет протокол

VoIP - англ. Voice over IP, укр. Голос через IP

DHSP - англ. Dynamic Host Configuration Protocol, укр. Протокол динамічної конфігурації вузла

WPF - англ. Windows Presentation Foundation, укр. Основа презентації windows

CEF - англ. Chromium Embedded Framework, укр. Вбудований фреймворк Chromium

XAML - англ. eXtensible Application Markup Language, укр. Розширювана мова розмітки застосунків

SDK - англ. Software Development Kit, укр. Набір для розробки програмного забезпечення

SIP - англ. Session Initiation Protocol, укр. Протокол встановлення сесії

UDP - англ. User Datagram Protocol, укр. Протокол користувальницьких датаграм

TCP - англ. Transmission Control Protocol, укр. Протокол управління передачею

RTP - англ. Real-time Transport Protocol, укр. Протокол транспортування в реальному часі

SRTP - англ. Secure Real-time Transport Protocol, укр. Протокол безпечного транспортування в реальному часі

HTTPS - англ. HyperText Transfer Protocol Secure, укр. Безпечний протокол передачі гіпертексту

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				7
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Сьогодні обмін інформацією через мережу між людьми досягнув піку популярності. Все більше користувачів відмовляються від голосового зв'язку через стільникову мережу на користь цифрових способів спілкування. На ринку представлено безліч месенджерів, що працюють на різних платформах і використовують різноманітні технології. Хоча більшість із них пропонують подібний набір функцій, кожен застосунок має свої унікальні особливості – деякі забезпечують шифрування даних, інші дозволяють спілкування навіть без підключення до інтернету. Попри велику кількість доступних сервісів, потреба у зручних засобах комунікації залишається незадовільною.

Незважаючи на велику кількість доступних проєктів, існує об'єктивна потреба в розробці нових рішень, зважаючи на війну, за кошти, ворог може отримати доступ до ваших даних, якщо це іноземна розробка, тому потрібні рішення які б не тільки забезпечували надійність і безпеку обміну даними, але й були відкритими до масштабування та адаптації під конкретні потреби користувача або організацій.

Серед багатьох нових інструментів можна виокремити декілька інструментів, які мають велику перспективу для вирішення завдання. Firebase – у ролі хмарної платформи для обміну даними у режимі реального часу. Asterisk – система з відкритим кодом для з'єднання користувачів за VoIP методом. Комбінація цих інструментів створить потужну та гнучку систему, яка зв'яжеться за допомогою двох мов програмування C# для клієнтської сторони та мови Python на стороні серверу. Така комбінація дозволяє розробити інноваційне рішення, яке забезпечуватиме багатофункціональність, високу надійність, безпеку та зручність використання.

Об'єктом дослідження кваліфікаційної роботи є процес функціонування системи захищеного обміну даними, текстовими повідомленнями та VoIP-зв'язку на основі Firebase та Asterisk.

Предметом дослідження є програмні та мережеві засоби для захищеного обміну текстовими повідомленнями та VoIP-зв'язком, створення застосунку на

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		8

мові C#, з хмарними сервісами Firebase та сервером Asterisk, а також протоколи, бібліотеки та інструменти, які надають механізми взаємодії.

Метою кваліфікаційної роботи на тему “Проект системи обміну повідомленням та VoIP-зв’язку на основі Firebase та Asterisk” є повністю діючий проєкт системи, який здатний одночасно обслуговувати як текстову так і голосову комунікацію, приватні чати та групові з подальшою можливістю масштабування та кастомізацією.

Кваліфікаційна робота спрямована на створення проєкту системи обміну повідомленнями на основі інтеграції з хмарною базою даних та хмарним сховищем, налаштування обміном повідомленнями у режимі реального часу, впровадження VoIP-зв’язку з урахуванням сучасних вимог до якості та швидкості передачі повідомлень, конфіденційності та масштабованості.

В рамках цієї роботи також буде досліджено питання взаємодії між клієнтською частиною застосунку, серверною логікою, зовнішніми сервісами, що дозволить сформулювати цілісне враження від використання та аналізу побудови подібних систем у реальних умовах.

Увага приділятиметься практичним аспектам реалізації, створенню інтерфейсу застосунку на WPF C#, налаштуванню серверу Asterisk, інтеграції Firebase для передачі та синхронізації повідомлень. Предметом дослідження визначено саме як, технічну реалізацію системи цифрової комунікації, що поєднує в собі як традиційний обмін повідомлень, так і голосових дзвінків.

Виконання цієї роботи є важливим кроком до набуття практичного досвіду у сфері розробки складних, багатокомпонентних комунікаційних систем, а результати роботи можуть використовуватись як основа для створення спеціалізованих рішень, у сфері бізнесу, освіти чи внутрішньо-корпоративних контактів. При цьому є можливість використовувати роботи у якості звичайного месенджера для простих користувачів. Робота також демонструє, як на основі доступних інструментів з відкритим вихідним кодом можливо реалізувати ефективну альтернативу дорогим комерційним продуктам, що є особливо актуальним у реаліях обмежених бюджетів чи потреби в індивідуальних рішеннях.

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		9

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ СИСТЕМИ ОБМІНУ ПОВІДОМЛЕННЯМИ ТА VoIP-ЗВ'ЯЗКУ

1.1 Аналіз існуючих методів та засобів обміну повідомлень та VoIP зв'язків.

З появою Інтернету почалася нова ера цифрової комунікації. Текстові повідомлення, які раніше обмежувалися короткими SMS-повідомленнями у стільникових мережах, поступово трансформувалися у зручніші та масштабовані рішення, як месенджери, які не залежать від конкретного оператора зв'язку. Паралельно розвивались і технології передачі голосу через IP-протокол – VoIP.

Перші кроки у сфері VoIP були зроблені ще у 1990-х роках, коли з'явилися перші програми для голосових дзвінків через Інтернет, як-от VocalTec Internet Phone (1995). Тоді технологія ще мала обмежену якість зв'язку та потребувала значної пропускної здатності мережі, проте вона заклала основу для подальшого розвитку.

У той самий період почали з'являтися перші мережеві месенджери. Одним із перших був ICQ, створений у 1996 році ізраїльською компанією Mirabilis. Він запропонував новий підхід до спілкування: можливість надсилати миттєві текстові повідомлення безпосередньо через Інтернет. Згодом його концепцію перейняли такі популярні сервіси, як AIM (AOL Instant Messenger), MSN Messenger, Yahoo! Messenger та інші.

2000-і роки стали справжнім проривом у розвитку месенджерів. Вони набули нових функцій і з'явилася підтримка голосових та відео дзвінків, обміну файлами, створення чат-груп. Саме тоді світ побачив один із найвідоміших VoIP-сервісів – Skype (2003). Його унікальна архітектура peer-to-peer забезпечувала високу якість зв'язку навіть на повільних з'єднаннях, а простота використання зробила Skype надзвичайно популярним як серед приватних користувачів, так і в бізнес-середовищі [42].

З розвитком смартфонів і мобільного Інтернету у 2010-х роках спостерігався новий етап еволюції – месенджери стали мобільними. Такі застосунки як

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		10

WhatsApp, Telegram, Viber, Facebook Messenger та інші, почали витіснити традиційні SMS і голосовий зв'язок через мобільну мережу. Користувачі отримали змогу обмінюватися не лише текстом і голосом, але й мультимедійним контентом у будь-який момент, незалежно від географічного розташування.

Сучасні месенджери поєднують в собі численні сервіси: текстове спілкування, голосові і відеодзвінки, пересилання файлів, підтримку чат-ботів, інтеграцію з іншими системами, використання хмарного зберігання. Паралельно з цим зростає значення безпеки та конфіденційності – все частіше використовуються наскрізне шифрування (end-to-end encryption), контроль доступу та інші захисні механізми [42].

На технічному рівні VoIP також зазнала значної еволюції. З'явилися відкриті платформи на кшталт Asterisk, які дозволяють створювати власні телефонні системи, адаптовані до потреб бізнесу або окремих користувачів. Вони підтримують гнучку маршрутизацію дзвінків, запис розмов, IVR-меню, інтеграцію з CRM та іншими зовнішніми системами. Це дає змогу створювати повноцінні комунікаційні платформи, зберігаючи контроль над даними і функціональністю [32].

Розглянемо деякі аналоги таких програмних продуктів, створених професіоналами та мають немалу популярність:

Viber (рис. 1.1) – програма для дзвінків та повідомлень, яка об'єднує всіх нас, де б ми не знаходилися. Кожен місяць сотні мільйонів людей безкоштовно спілкуються з друзями та близькими за допомогою повідомлень, голосових та відео дзвінків, та інших функцій програми. Наскрізне шифрування всіх особистих чатів та дзвінків, а також групових чатів у Viber гарантує безпеку спілкування [46].



Рисунок 1.1 - Viber

Як каже сам вайбер “З Viber всі особисті чати завжди під захистом та назавжди залишаться тільки вашими. Наша місія – захищати приватність спілкування, щоб вам не довелося вибирати теми, які ви можете чи не можете обговорювати у Viber.”, але як відомо вайбер було створено білоруськими розробниками тому довіряти безпечності даних, не є доцільним.

Переваги:

- Поширеність платформи з великою базою користувачів;
- Підтримка голосових повідомлень та відео-дзвінків;
- Можливість створення спільнот;
- Інтеграція з десктопною версією.

Недоліки:

- Закрите програмне ядро;
- Важке навантаження на систему, особливо на старих пристроях;
- Часті рекламні повідомлення.

Facebook Messenger (рис. 1.2) — це також безкоштовний мобільний застосунок для обміну повідомленнями, який використовується для обміну миттєвими повідомленнями, обміну фотографіями, відео, аудіозаписами та для

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

групових чатів. Застосунок можна використовувати для спілкування зі своїми друзями у Facebook та з телефонними контактами [46].

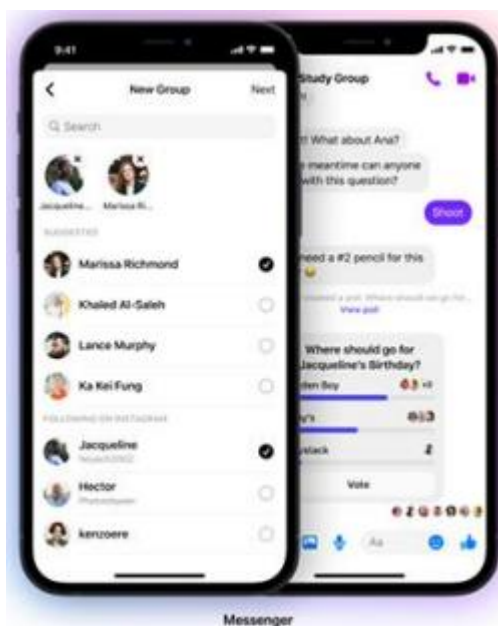


Рисунок 1.2 - Facebook Messenger

Якщо говорити про Facebook Messenger, варто згадати, що він позбавлений функції видалення переписки у всіх учасників переписки. Тобто ваші повідомлення назавжди залишаються у співрозмовника і можуть бути доступні зловмиснику, навіть якщо в себе ви їх видалили.

Переваги:

- Інтеграція з екосистемою Meta;
- Висока доступність;
- Підтримка аудіо та відео дзвінків;
- Підтримка обмінів файлами.

Недоліки:

- Немає наскрізного шифрування;
- Високе споживання ресурсів і трафіку;
- Залежність від облікового запису Facebook.

Telegram (рис. 1.3) – багатоплатформовий клауд-месенджер з функціями VoIP для смартфонів, планшетів та ПК, який дозволяє обмінюватися текстовими, голосовими та відео повідомленнями, наліпками та фотографіями, файлами багатьох форматів [46].



Рисунок 1.3 - Telegram

Мессенджер Telegram дозволяє за кілька хвилин вирахувати точні координати своїх користувачів, і для цього не потрібно бути хакером. Уязвимість скривається в сервісі «Люди поряд», вбудованому в месенджер. З його допомогою можна переходити користувачів Telegram, що знаходяться поблизу – їх місце дислокації визначається за GPS-координатами і відстежується в реальному часі. Сам месенджер не розкриває координати користувачів, а лише вказує точне розташування до них, але цього вже достатньо для вирахування їх місця розташування та навіть адреси проживання, якщо вони підключені до Telegram з дому. Для цього можна скористатися картою та геометричним методом, доступним в Інтернеті. Також повідомлення, надіслані через Telegram, зберігаються на серверах Telegram у вихідній формі або у вигляді звичайного тексту, без шифрування для захисту особистих даних користувачів [46].

Переваги:

- Швидкість роботи та висока стабільність;
- Підтримка великих груп та каналів;
- Відкрите API для розробників;
- Хмарна синхронізація;
- Платформи для комп'ютерної версії.

Недоліки:

- Стандартні чати не мають шифрування;

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				14
Змн.	Арк.	№ докум.	Підпис	Дата		

- Не підтримує VoIP;
- Обмежений контроль над повідомленнями.

3CX (рис. 1.4) – це комерційне програмне забезпечення, яке забезпечує всі функції для IP-телефонії, включаючи відеоконференції, чат, інтеграцію з CRM та мобільними застосунками. 3CX доступний як для локального розгортання, так і як хмарний сервіс. Ця платформа зручна в адмініструванні та пропонує простий інтерфейс, що робить її популярною серед компаній, які шукають готове рішення [47].

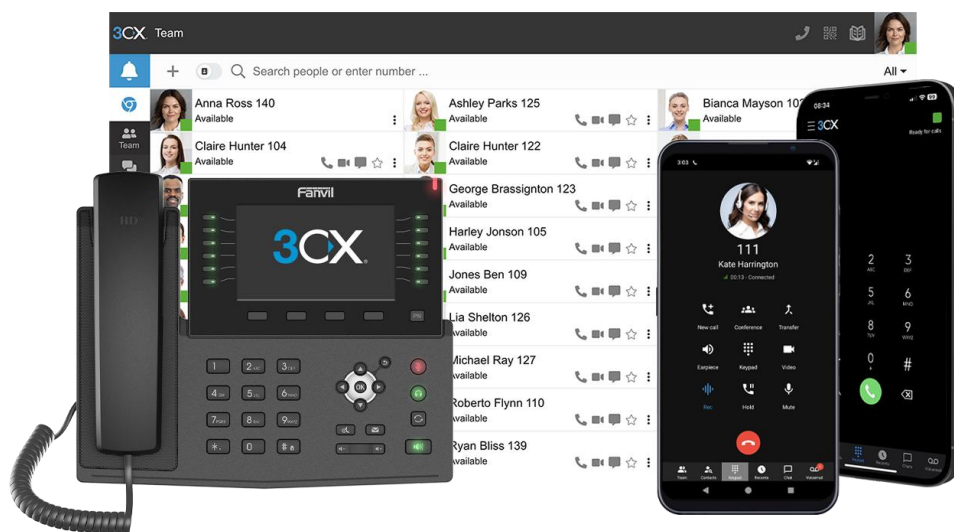


Рисунок 1.4 - 3CX

Переваги:

- Потужна VoIP-система з підтримкою SIP;
- Наявність мобільних та десктопних застосунків;
- Підтримка відео дзвінків та конференцій;
- Можливість локального та хмарного хостингу.

Недоліки:

- Комерційна модель, повна функціональність доступна лише в платних версіях;
- Вимагає роботу тільки в середині NAT/Firewall, за STUN дозволені лише текстові повідомлення;
- Менш зручний інтерфейс для звичайного користувача в порівнянні з месенджерами.

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		15

FreePBX (рис. 1.5) – це безкоштовне рішення для створення IP-АТС, побудоване на базі Asterisk. Вона має зручний веб-інтерфейс для управління конфігурацією телефонної системи, включаючи маршрутизацію дзвінків, записи та повідомлення. FreePBX дозволяє інтегруватися з багатьма сторонніми сервісами, що робить її вигідним вибором для малого та середнього бізнесу, який прагне зекономити на зв'язку [47].



Рисунок 1.5 - FreePBX

Переваги:

- Безкоштовна система з відкритим кодом на основі Asterisk;
- Повнофункціональна IP-телефонія;
- Велика спільнота, підтримка сторонніх модулів;
- Гнучкість у налаштуванні та масштабуванні.

Недоліки:

- Складність у початковому налаштуванні;
- Інтерфейс адміністратора не інтуїтивні;
- Частина модулів є платними;
- Вимагає linux-сервер.

Linphone (рис. 1.5) — це програмне забезпечення з відкритим кодом для VoIP, яке дозволяє здійснювати голосові та відеодзвінки через протокол SIP. Linphone доступний для різних операційних систем, включаючи Windows, macOS, Linux, Android та iOS. Цей клієнт орієнтований на кінцевих користувачів та невеликі компанії, пропонуючи простий інтерфейс, підтримку текстових повідомлень і HD-звуку. Linphone також відомий високим рівнем безпеки завдяки

підтримці шифрування дзвінків і повідомлень, що робить його популярним серед користувачів, які цінують приватність [27].

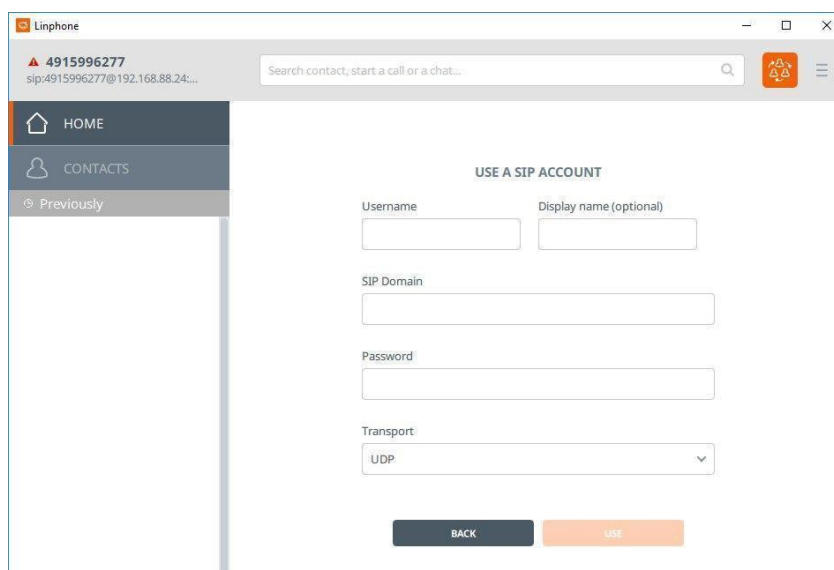


Рисунок 1.6 - Linphone

Переваги:

- Відкрите програмне забезпечення;
- Підтримує sip-протокол;
- Є мобільна та десктопна версія;
- Можна використовувати як основу для власних клієнтів.

Недоліки:

- Мінімалістичний інтерфейс;
- Відсутність чатів/груп/каналів, лише VoIP-функціональність;
- Вимагає сервер SIP і знань для налаштування;
- Обмежений функціонал без доповнень або кастомізації.

Порівняльна таблиця найвідоміших конкурентних застосунків

Назва ПД	Функціонал			
-//-	Безпечність	Зручність	Гарність інтерфейсу	Бистродія
Viber	-	±	+	+
Facebook Messenger	-	-	+	+
Telegram	-	+	+	+
3CX	+	+	+	+
FreePBX	+	-	-	+
Linphone	+	-	+	-

1.2 Аналіз інструментів та технологій захищеного обміну даними.

У сучасних умовах розвитку інформаційних технологій особливу увагу приділяють розробці інтерактивних, надійних і масштабованих комунікаційних систем, які здатні забезпечити як обмін повідомленнями, так і підтримку голосового зв'язку з використанням IP-телефонії. Одним із важливих етапів створення подібної системи є правильний вибір інструментальних засобів, які забезпечують реалізацію усіх функціональних вимог, а також оптимізують процес розробки, спростять подальшу підтримку проєкту і сприятимуть досягненню високого рівня стабільності та продуктивності. Обрана тема кваліфікаційної роботи «проєкт системи обміну повідомленнями та VoIP-зв'язку на основі технологій Firebase і Asterisk» обумовлює специфіку вибору засобів програмування, платформ, інтерфейсних технологій, бібліотек і серверного середовища.

У якості основної мови програмування для клієнтської частини було обрано C#. Цей вибір зумовлений, з одного боку, високою продуктивністю платформи .NET, а з іншого широким розповсюдженням середовища Windows як цільової ОС для десктопного застосунок [3]. У поєднанні з Windows Presentation Foundation

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		18

(WPF) та мовою розмітки XAML, C# надає змогу створювати зручний, адаптивний, масштабований графічний інтерфейс користувача, що відповідає сучасним вимогам до дизайну клієнтських програм. WPF дозволяє реалізувати поділ логіки інтерфейсу та бізнес-логіки програми, що значно спрощує розробку, тестування і подальший супровід програмного забезпечення. Крім того, XAML забезпечує можливість динамічної модифікації інтерфейсу та підключення шаблонів стилів, що особливо корисно для реалізації UI/UX-елементів у системі комунікацій [6].

Інтеграція веб-функціоналу здійснюється за допомогою CEF (Chromium Embedded Framework) браузерного компонента, який дозволяє вбудовувати повноцінний веб-оглядач всередину десктопного застосунка. Це дає змогу використовувати веб-технології для відображення певних елементів інтерфейсу, наприклад, повідомлень, профілів користувачів чи контенту з Firebase. Використання CEF також відкриває можливості глибокої інтеграції з сервісами, побудованими на основі JavaScript або інших клієнтських веб-технологій, що додає гнучкості до архітектури проєкту [21].

Для реалізації голосового зв'язку було обрано використання бібліотеки OZEKI VoIP SDK. Це програмне забезпечення забезпечує повноцінну підтримку протоколу SIP і дозволяє реалізувати функції прийому та здійснення дзвінків, передачі аудіо, а також взаємодії з сервером Asterisk. Головною перевагою OZEKI SDK є стабільність роботи, широкий набір функцій та зручна інтеграція з C#-застосунками, що дозволяє реалізувати VoIP-функціонал без необхідності створення низькорівневих реалізацій.

З метою забезпечення функціональності зберігання, обміну та синхронізації текстових повідомлень використовується хмарна платформа Firebase. Вибір Firebase продиктований тим, що цей сервіс забезпечує реальний час синхронізації даних між клієнтами, має зручне REST API, високу надійність і масштабованість, а також пропонує гнучку систему аутентифікації, що може бути корисною для майбутнього розширення системи. Для взаємодії клієнтської частини з Firebase застосовується бібліотека Firesharp, це C#-обгортка над REST API Firebase, яка спрощує процес відправки, отримання та оновлення даних. Це дозволяє розробнику

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		19

зосередитися на бізнес-логіці застосунку, не заглиблюючись у низькорівневу реалізацію запитів [11, 12, 13].

Для автоматизації процесів керування користувачами VoIP, зокрема додавання нових облікових записів SIP на сервер Asterisk, було використано мову Python. Використання Python на серверному рівні обумовлено її простотою, високою продуктивністю у сфері обробки конфігураційних файлів та взаємодії з операційною системою. Python-скрипти дозволяють створювати та оновлювати конфігураційні файли Asterisk, автоматично перезапускати необхідні служби та забезпечувати централізоване керування SIP-користувачами [35].

Серверна частина розгортається на базі операційної системи Ubuntu, що обумовлено її стабільністю, широкою підтримкою Asterisk, великою кількістю документації та спільноти. Ubuntu також має низькі вимоги до ресурсів і забезпечує високий рівень безпеки. Розміщення Asterisk на Linux-платформі є стандартною практикою в індустрії, яка забезпечує найкращу продуктивність та гнучкість у налаштуванні. У той час як клієнтська частина працює на Windows, серверна екосистема є ізольованою і незалежною, що підвищує стабільність та відмовостійкість всієї системи в цілому [26].

Розроблювана система передбачає використання сучасного апаратного забезпечення на стороні клієнта: персональний комп'ютер з операційною системою Windows 10 або вище, процесором не нижче Intel Core i3 або AMD Ryzen 3, оперативною пам'яттю обсягом від 4 ГБ, та наявністю стабільного підключення до мережі Інтернет. Такі вимоги є обґрунтованими, оскільки WPF-застосунки споживають помірну кількість ресурсів, але для стабільної роботи відео або голосового зв'язку потрібна певна апаратна база. Серверна частина, розгорнута на Ubuntu, не має високих вимог до ресурсів: достатньо віртуальної машини або фізичного сервера з процесором початкового рівня, 2 - 4 ГБ оперативної пам'яті та кількома гігабайтами місця на жорсткому диску. Asterisk не потребує графічного інтерфейсу і працює у фоновому режимі, що робить можливим його ефективне використання навіть на недорогих віртуальних приватних серверах чи локальних домашніх серверах.

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		20

1.3 Аналіз мережевих протоколів для забезпечення обміну даними.

В системах обміну повідомленнями та VoIP-зв'язку вибір мережевих протоколів є одним із ключових аспектів, оскільки від цього залежить якість, швидкість та безпека. Вибір протоколів має бути обґрунтованим, а їх комбінація повинна забезпечувати найкращий баланс між швидкістю передачі даних, зручністю використання та безпекою.

SIP - це протокол, що використовується для ініціалізації, підтримки та завершення сеансів зв'язку, таких як голосові дзвінки та відеоконференції, через IP-мережі. SIP є відкритим стандартом, і його широко підтримують усі основні платформи для VoIP, включаючи Asterisk. Він працює за принципом запит-відання (request-response), що дозволяє клієнтам ініціювати дзвінки, встановлювати з'єднання та передавати медіа потоки [33].

IAX - спеціально розроблений для Asterisk і використовується в основному для внутрішніх з'єднань між серверами Asterisk. Він пропонує більш ефективну передачу даних, оскільки використовує менше час затримок і не потребує додаткових портів для медіа потоків. Однак IAX має обмежену підтримку на стороні сторонніх пристроїв та менш розповсюджений, ніж SIP [27].

PJSIP - це бібліотека, яка реалізує SIP і призначена для роботи з мобільними пристроями. Вона має більшу функціональність і простоту інтеграції в мобільні застосунки, але в цьому проєкті перевага у SIP через його універсальність і підтримку більшістю пристроїв [33].

Тому зважаючи на переваги та недоліки у даному проєкті було обрано саме SIP-протокол.

Але йому потрібна допомога по обміну даними між користувачами, так званий транспорт, тому на транспорт потрібно також обрати протокол.

UDP – це протокол передачі даних, який працює без встановлення з'єднання між відправником і приймачем. Це означає, що він не гарантує доставку пакетів, їх порядок чи перевірку на помилки. Замість цього, UDP просто відправляє пакети даних і не чекає підтвердження від отримувача. Це робить протокол швидким, але менш надійним порівняно з іншими протоколами, такими як TCP. UDP

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		21

використовується в ситуаціях, де важлива швидкість передачі, а точність не є критичною, наприклад, в онлайн-іграх або при потоковій передачі аудіо та відео [28].

TCP – це протокол, що забезпечує надійну передачу даних. У порівнянні з UDP, TCP встановлює з'єднання між відправником і приймачем і гарантує, що всі пакети будуть доставлені в правильному порядку і без помилок. TCP здійснює перевірку на помилки, підтвердження отримання даних та повторну передачу пакетів, якщо вони не були доставлені. Це забезпечує високу надійність, але через всі ці додаткові перевірки TCP може бути повільнішим. Цей протокол часто використовують для передачі важливих даних, таких як веб-сторінки, електронні листи чи файли [29].

У VoIP-зв'язку швидкість передачі та мінімальна затримка є критичними, тому UDP є ідеальним вибором.

Але медіа потік, потрібно передавати по іншому, для цього не підійде голий UDP протокол, йому також потрібен додатковий помічник.

RTP – це протокол для транспортування мультимедійних даних в реальному часі через комп'ютерні мережі. Зазвичай використовується для передачі аудіо, відео, голосових викликів або інших медіа, де важливий час і затримка. RTP забезпечує синхронізацію і доставку пакетів даних у потрібному порядку, що є важливим для підтримки якості медіа потоку, особливо для таких застосунків, як VoIP, відеоконференції, потокове відео тощо [41].

SRTP – це розширення RTP, яке додає механізми шифрування та аутентифікації для захисту медіа потоків у реальному часі, таких як голосові і відеодзвінки. SRTP забезпечує конфіденційність переданих даних, що є надзвичайно важливим для забезпечення безпеки в VoIP-зв'язку. Використання SRTP дозволяє запобігти перехопленню або модифікації медіа потоків під час їх передачі через Інтернет [40].

ZRTP — це ще один протокол шифрування для медіа потоків, який працює без необхідності використання сертифікатів або централізованого серверу. ZRTP використовується для встановлення ключів шифрування безпосередньо між

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		22

кінцевими точками, що забезпечує високий рівень безпеки. Однак ZRTP не підтримується так широко, як SRTP, і його впровадження може бути складнішим [42].

Вибір SRTP для проєкту є більш доцільним через його широке поширення та підтримку у стандартних рішеннях для VoIP. Крім того, SRTP вже інтегровано в Asterisk і може бути легко налаштовано для забезпечення шифрування медіа потоків.

Тому, для створення проєкту системи було вибішено використовувати наступні протоколи: SIP, UDP, SRTP.

1.4. Аналіз технологій Firebase, Asterisk

Розробка сучасних систем, які орієнтуються на обмін повідомленнями та голосовим зв'язком через мережеві сервіси, вимагає надійну технологічну основу. Технологічна основа повинна забезпечити ефективну роботу систем в умовах багатокористувацького потоку даних, високого навантаження. Також задля запобігання перехоплення трафіку має бути безпечним.

У рамках роботи обрано дві ключові платформи технологій, навколо яких буде побудовано проєкт, а саме Firebase, як хмарний провайдер серверних послуг, що включає в себе документно-орієнтовану базу даних реального часу для обміну повідомленнями, сповіщень систем та файлове сховище для обміну файлами між користувачами. З іншого боку, оскільки Firebase не може задовольнити обмін голосовим потоком, обрано Asterisk, у якості SIP-сервера та провайдера, VoIP-зв'язку.

Такий вибір технологій обумовлений великою кількістю документації, практики безвідмовності інструментів, підтримкою сучасних протоколів міжмережевого зв'язку, гнучкістю конфігурації та активною спільнотою користувачів. Також варто зазначити що Asterisk поставляється, як продукт з відкритим вихідним кодом за GNU Public License version 2 (GPLv2), в той час як Firebase, це комерційний продукт від Google, який має безкоштовну версію для

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		23

розробки, та дозволяє зберігати до 10 GiB даних, але не більше ніж 1 GiB за 24 години, в іншому випадку потрібна оплачена ліцензія [15, 26].

Firebase дає набір інструментів для створення мобільних, веб та GameDev застосунки, до них є офіційні бібліотеки від Google, інші види застосунків вимагають використання сторонніх бібліотек, для зв'язку з ним. Він забезпечує серверну функціональність без необхідності розгортання та налаштування окремої серверної інфраструктури, це дасть змогу зменшити навантаження на сервер де встановлено Asterisk, а також забезпечує синхронізацію та зберігання даних між користувачами без великих затрат людської праці для досягнення цих рішень [15].

Також Firebase синхронізується з Google Analytics, що дає змогу моніторити та аналізувати години пікової активності користувачів, відсоток завантаження системи, кількість дзвінків та повідомлень [18].

Великою перевагою Firebase є його технічна можливість масштабуватись, у випадку зростання навантаження або стрибку активності користувачів, або активних користувачів, платформа автоматично розширює свої ресурси, що змушує усю систему працювати стабільно без необхідності ручного втручання. Це зменшить витрати часу розробників та прибере необхідність наявності On Call спеціалістів для адміністрування та підтримки інфраструктури [15].

Голосовий зв'язок забезпечується технологічним рішенням Asterisk, котра посідає одне з провідних місць серед Open Source (застосунки або рішення з відкритим вихідним кодом) систем VoIP. Asterisk забезпечує функціонал телефонної станції на програмному рівні. Він підтримує використання більшості сучасних протоколів сигналізації та передачі медіа-потоків, серед них можна виокремити: Sip, PJSIP, IAX, MGCP, RTP, SRTP, ZRTP протоколи [26].

Основна перевага Asterisk, це його модульність і можливість розгортання на невеликих локальних мережах, з можливістю масштабування на світову павутину, відкриттям необхідних транспортних портів.

Asterisk у цьому проєкті використовується, як єдиний VoIP-сервер на якому буде зав'язана реалізація функцій голосового зв'язку, маршрутизації медіа-пакетів,

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		24

реєстрація та сигналізація користувачів, кодування та декодування голосових потоків.

Завдяки підтримці SIP, Asterisk може взаємодіяти з різноманітними клієнтами, починаючи від програмних Soft-phone (надалі софтфон) до апаратних IP-телефонів, це забезпечує велику підтримку цього типу серверу у різних умовах.

В даному проєкті буде використовуватись протокол SIP, тому це одна з найбільших переваг Asterisk, для даної роботи.

Ще один плюс до Asterisk-у це його підтримка шифрування голосових потоків, через протокол SRTP та сигналізації через DTLS-SRTP. Це критично важливо в умовах кіберзагроз, які постійно зростають в умовах війни, це дозволить захистити не лише вміст розмов, а й сам факт встановлення виклику [42].

Самі по собі вони не є клієнтом, для забезпечення взаємодії потрібно використати сторонні інструменти які будуть створювати клієнтським застосунком та дозволять використовувати усі функції Firebase-у та Asterisk-у.

Ключове місце поєднання цих технологій є мова програмування C#, у поєднанні з Windows Presentation Foundation (надалі WPF), можна створити сучасний застосунок з гнучким інтерфейсом користувача та підтримкою багатопотокових і асинхронних операцій з екосистемою .NET, при необхідності використовуючи Xamarin, цей застосунок можна перенести і на мобільні пристрої. C# має велику кількість бібліотек і сервісів, включно з прямою взаємодією RESTful API, що дозволить створити інтеграцію з Firebase [6].

Для реалізації доступу до Firebase, у C# можна використати RESTful API, проте це не буде зручно, оскільки потрібно буде зробити велику кількість класів, методів-перевірок та методів-передатчиків даних, це все можна прибрати використавши одну з Open-Source бібліотек для взаємодії з Firebase.

Було обрано FireSharp, це Open-Source бібліотека що забезпечує готові інтерфейси, класи та методи для взаємодії з Firebase Realtime Database (База даних реального часу Firebase) за допомогою HTTPS протоколу. З'єднання безпечне оскільки використовується secure версія протоколу HTTP, проте дані які передаються попередньо шифруються також, щоб ніхто отримавши доступ до бази

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		25

даних не міг нічого прочитати. Firesharp гарно підійде до цієї системи, оскільки він дозволяє виконувати CRUD запити, мінімізуючи написання програмного коду, і не шкодить швидкості роботи, оскільки більшість його функціоналу це обгортка над RESTful API. Робота з ним спрощується додатково тим, що FireSharp підтримує об'єктну модель даних, що дозволяє економити час над проведенням серіалізації та десеріалізації даних, а також використовувати його в межах MVVM, що використовується в WPF [11].

З голосовим зв'язком вже складніше, оскільки нормальних робочих Open Source проєктів, які б працювали з .NET 5, або старше майже немає. Тому буде використовуватись комерційна SDK, від Японських розробників, OZEKI VoIP SDK.

OZEKI VoIP SDK - це набір інструментів який поставляється з усім необхідним для створення клієнту, з прикладом форми, спеціальним дебагером та бібліотеками для взаємодії з SIP серверами та створення софтфонів. Крім цього вона поставляє розробникам високо рівневі інтерфейси для створення, прийому, маршрутизації та управління дзвінками, а також можливості роботи з медіа-потоками.

Використання технології WPF у межах клієнтського застосунку обумовлено її перевагою над WinForms, оскільки має величезний спектр кастомізації клієнту, оскільки весь візуальний інтерфейс написаний на мові декларативної розмітки XAML, її можна стилізувати, а C# буде виступати обробником подій та моніторщиком даних. Це також дозволяє відокремити бізнес-логіку від побудови інтерфейсу, що дає великий приріст в масштабованість елементів, легке оновлення інтерфейсу у відповідь на зміни даних або подіям.

Обмін повідомленнями вимагає від користувача мати можливість читати ці повідомлення у простому та звичному форматі. Але дані які отримує клієнт передаються та конвертуються у структури JSON, що не є простим читабельним форматом для простої середньостатистичної людини, не пов'язаною з програмуванням або ІТ в загальному, тому для забезпечення читабельності дані необхідно конвертувати спочатку в HTML, а далі відобразити їх на внутрішньому

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		26

браузері. В даному етапі виникне проблема, вбудований браузер в C#, це Edge на нових комп'ютерах та Internet Explorer на більш старих, в даному випадку сторінка використовує HTML, але ігнорує каскадні таблиці стилів (надалі CSS), та це виглядає як сторінка формату А4 заповнена випадковим текстом, щоб виправити це та змусити C# працювати однаково в різних умовах буде імпортовано бібліотеку Chromium Embedded Framework Browser (надалі CEF Browser), і з планів використання стандартного веб-переглядача погляд плавно переїжджає на CEF Browser, що дозволить забезпечити ідентичну роботу на різних пристроях та комфортний перегляд даних.

При розробці цього проєкту буде активно використовуватись NuGet, стандартний менеджер пакетів, що забезпечує зручну доставку зовнішніх бібліотек та їх оновлень.

1.5 Постановка технічного завдання

Аналіз сучасного стану VoIP-телефонії та засобів обміну повідомленнями показало актуальну проблему створення застосунку що вирішить проблему незахищеного зв'язку між користувачами.

Основною метою проєктування є реалізація клієнтського застосунку на базі C# з використанням WPF, для створення графічного інтерфейсу. Обмін повідомленнями відбуватиметься через платформу Firebase Realtime Database, яка дозволить забезпечити швидку синхронізацію даних між клієнтами. Для реалізації голосового зв'язку використовується VoIP – сервер Asterisk, що підтримує стандарт SIP-протоколу та дозволяє налаштовувати користувацькі акаунти і маршрутизацію викликів. Створення та керування обліковими записами на Asterisk відбувається за допомогою окремого скрипта, написаного мовою Python, що автоматизує додавання нових користувачів без необхідності ручного редагування конфігурацій.

Для вказаного завдання, заплановані наступні кроки:

1. Розробка клієнтського застосунку на основі WPF C#, що надає користувачу змогу:
 - авторизуватись у системі;

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		27

- обмінюватись текстовим повідомленнями через Firebase;
 - обмінюватись файлами через Firebase Storage;
 - переглядати історію повідомлень;
 - здійснювати SIP–дзвінки;
 - отримувати windows/push сповіщення про нові повідомлення.
2. Налаштування хмарної бази даних Firebase:
- налаштування доступів для запису та читання даних;
 - створення ключів доступу для застосунку;
 - обробки подій у реальному часі;
 - увімкнення синхронізації.
3. Налаштування VoIP-сервер Asterisk:
- встановлення збірки на Ubuntu;
 - прийом і маршрутизація дзвінків;
 - налаштування портів.
4. Python-script:
- додавання нових користувачів у конфігураційні файли Asterisk;
 - перезапуск для оновлення конфігурації;
 - валідація даних для створення унікальних логінів.
5. Забезпечити інтеграцію між компонентами:
- передача та зчитування повідомлень між WPF-додатком та Firebase;
 - встановлення та керування SIP-сесіями між клієнтом і Asterisk;
 - логічне об'єднання текстового та голосового функціоналу в єдиній системі.

Висновки до 1 розділу

У першому розділі кваліфікаційної роботи було проведено аналіз існуючих методів та засобів обміну повідомлень та VoIP зв'язків, інструментів та технологій захищеного обміну даними, мережевих протоколів для забезпечення обміну даними, технологій Firebase, Asterisk.

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		28

Початково, була поставлена задача та визначено цілі роботи. Завдання включало в себе такі пункти як: розробка застосунку-клієнту, налаштування бази даних, розробка та налаштування серверної частини.

Проведено аналіз конкурентних програмних продуктів для детального вивчення існуючих систем, визначення їх сильних та слабких сторін. Цей аналіз дав зрозуміти, як можна покращити майбутній застосунок, яких помилок потрібно уникати, під час роботи над застосунком.

Крім цього було обґрунтовано вибір необхідних інструментів для розробки застосунку. Було використано мову програмування C#, яка є потужною та гнучкою, ідеально підходить для розробки десктопних застосунків, Asterisk як провайдер послуг SIP з відкритим кодом, та решту інструментів.

Було обрано протоколи які необхідні для забезпечення роботи VoIP частини застосунку.

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		29

РОЗДІЛ 2. ПРОЄКТУВАННЯ СИСТЕМИ ОБМІНУ ПОВІДОМЛЕННЯМИ ТА VoIP-ЗВ'ЯЗКУ

2.1. Проєктування структури клієнт-серверної взаємодії та вимог до проєкту на основі UML-діаграм

На початку побудови будь-якої системи необхідно визначити як користувач буде використовувати систему. Це необхідно для визначення необхідних функцій застосунку та проведення UML-моделювання. У мові UML давно існує спеціальний стандарт діаграми для отримання розуміння потреб користувачів - діаграма варіантів використання (рисунок 2.1). Актором у даному випадку буде виступати звичайний користувач системи. Провівши дослідження було визначено 11-ть потреб користувачів такого типу:

- Реєстрація;
- Вхід;
- Зміна пароллю;
- Зміна теми інтерфейсу;
- Пошук контактів;
- Створення пари контактів;
- Створення групових чатів;
- Відправка повідомлення;
- Отримання повідомлення;
- Дзвінок вхідний;
- Дзвінок Вихідний.

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		30

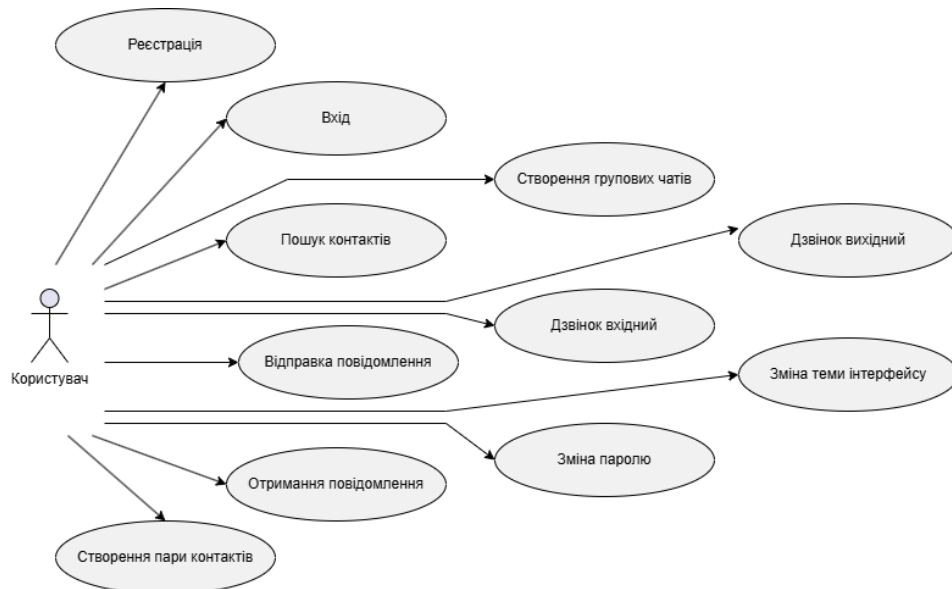


Рисунок 2.1 - Діаграма варіантів використання

До кожного варіанту використання потрібно поставити вимоги для отримання бажаного результату та зменшення кількості мануального тестування.

Вимоги до реєстрації:

- Унікальний логін (email/нікнейм);
- Валідний формат email;
- Надійний пароль (мінімум 8 символів, з великою літерою, цифрою тощо);
- Неможливість реєстрації з уже існуючим логіном.

Вимоги до входу:

- Валідні логін і пароль;
- Повідомлення про помилку при неправильних даних.

Вимоги до зміни паролю:

- Введення поточного паролю;
- Новий пароль повинен відповідати вимогам надійності.

Вимоги до зміни теми інтерфейсу:

- Наявність кількох тем;
- Миттєве застосування теми після зміни;
- Збереження вибору теми між сесіями користувача;
- Перевірка відображення основних елементів інтерфейсу при кожній темі.

Вимоги до пошуку контактів:

- Швидкий пошук за ім'ям, нікнеймом;

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		31

- Відображення збігів;
- Коректна робота пошуку при великій кількості контактів.

Вимоги до створення пари контактів:

- Обидва користувачі мають існувати в системі;
- Неможливість створити пару з самим собою;
- Перевірка, що пари ще не існує;
- Повідомлення обох користувачам про створення пари.

Вимоги до створення групових чатів:

- Мінімум 2 учасники окрім ініціатора;
- Можливість вибору назви для групи.

Вимоги до відправки повідомлення:

- Вибраний чат або користувач;
- Наявність тексту або медіавмісту;
- Шифрування вмісту повідомлення;
- Миттєве відображення повідомлення у чаті.

Вимоги до отримання повідомлення

- Користувач підключений до мережі;
- Повідомлення відображається у відповідному чаті;
- Нотифікації про нові повідомлення, якщо чат не відкритий.

Вимоги до вхідного дзвінка

- Користувач повинен бути онлайн;
- Повідомлення про вхідний дзвінок (UI + звук);
- Можливість відповісти або відхилити дзвінок;
- Логування пропущеного дзвінка.

Вимоги до вихідного дзвінка

- Вибраний користувач має бути доступний для дзвінка;
- Інтерфейс очікування відповіді;
- Завершення дзвінка з обох сторін.

Наведені вимоги важливо дотримуватись під час розробки системи оскільки вони забезпечують плавність роботи користувача, безпечність та якість.

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		32

Маючи діаграму активностей, та вимоги до них можна сформувати загальне представлення роботи клієнтського застосунку у вигляді діаграми класів (рисунок 2.2.)

На діаграмі класів маємо декілька класів, які будуть обслуговувати роботу клієнтського застосунку.

Клас **MainWindows.xaml.cs**: в своїй основі виконує комплексну роботу з автентифікації користувача. Якщо користувач раніше був авторизований, то з файлу збереження програми, дістається логін сесії користувача та виконується вхід у застосунок. Інакше користувачу необхідно ввести свої дані у поля входу користувача (нікнейм та пароль), після чого буде зроблено запит даних з Firebase, якщо користувача з таким нікнеймом буде знайдено, то перевіряється зміст хеш-пароллю. При повному співпадінні, користувача буде допущено у систему. При цьому з'єднання з базою даних буде збережено, та занесено у статичне поле класу, для зменшення кількості клієнтів зв'язку з базою даних. Це з'єднання буде використовуватись для інших подій у клієнтському застосунку, що зменшить ризики помилок Access Violation (спроба доступу до даних що ще не створені або використовуються в іншому потоці).

Клас **WindowsRegistration.xaml.cs**: виконує роль реєстрації даних на серверній частині. Користувач вводить три рядка даних. Електронна адреса, нікнейм та пароль, усі дані перевіряються на унікальність у базі даних, якщо вони унікальні, то акаунт створиться, в іншому випадку, на користувацький інтерфейс буде виведено помилку.

Клас **Aunt.cs**: використовується, як модель схеми бази даних користувача, який входить або реєструється в системі.

Клас **FileSystem.cs**: використовується для відправки файлів на віддалене сховище Firebase, файли на відміну від даних відправляються до Firebase Storage, це працює дещо інакше, в результаті, формується посилання на завантаження цього файлу, та передається у Firebase realtime database, як текст повідомлення.

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		33

Клас **MessageReciver.cs**: реалізує модель схеми даних діалогу між двома користувачами, його об'єкт представляє собою одне повідомлення, у якому міститься інформація, хто та кому відправив, а також порядковість повідомлення.

Клас **Utils.cs**: використовується для зберігання статичних методів, які не можна віднести ні до одного з інших класів, з принципу доцільності розміщення, цей клас не створює та не змінює жодних даних, що впливають на глобальну роботу застосунку.

Клас **Friend.cs**: є схемою моделі бази даних для створення пар користувачів, у які буде записуватись діалоги. Діалоги неможливі, якщо пара користувачів не створена, оскільки без присутності пари, користувачі не будуть відображені на панелі історії діалогів.

Клас **GroupModel.cs**: реалізує собою модель бази даних для створення груп користувачів, група може бути створена при наявності від трьох користувачів, з якими попередньо був діалог у людини, що створює групу.

Клас **Sip.cs**: реалізує собою softphone, який створюється за допомогою Ozeqi VoIP SDK, у цьому класі описані всі методи з початком, отриманням, відповіданням, втриманням, проходженням та завершенням виклику. Також тут налаштовується тип роботи, транспорт, адреса серверу-провайдеру та STUN-сервер.

Клас **CallWindow.xaml.cs**: це обгортка над sip.cs, використовують як інтерфейс виклику, створює взаємодію з медіа потоком, дозволяє вмикати та вимикати мікрофон, починати і завершувати виклик.

Клас **MainMessenger.xaml.cs**: це основний клас взаємодії з користувачем. При створенні клас кешує повідомлення користувача з іншими користувачами, заповнює бокову панель контактів, створює браузер, запускає на безкінечний асинхронний цикл перевірки нових повідомлень, наявності вхідного дзвінка, та наявності нової пари користувачів, відправляє повідомлення, файли, емоїї, створює діалогові вікна, вікна дзвінка, дозволяє змінювати дизайн-схему.

Схему діаграмів класу представлено в додатку Б

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		34

Маючи таку інформацію про майбутній застосунок, можна продумати декілька сценаріїв, які передбачають наперед найбільші та найважливіші пункти застосунку. Оскільки застосунок буде реалізовуваний на базі Ozeki Voip SDK та Firebase, було обрано декілька критичних сценаріїв:

- Вхід користувача;
- Відправлення повідомлення;
- Вхідний дзвінок.

Сценарій входу користувача в аккаунт починається з ініціалізації стартового вікна програми, яке здійснює перевірку наявності збережених даних сесії. Якщо попередній сеанс існував, то з файлової системи, завантажуються логін користувача, виконуються з'єднання з базою даних для перевірки автентичності.

У разі відсутності збереження облікових даних, користувачеві пропонується ввести нікнейм та пароль вручну. Після введення даних відбувається запит до бази даних, де здійснюється пошук користувача за нікнеймом, якщо дані знайдено, порівнюється хешований пароль збережений на базі даних та пароллю введеного користувачем та у разі успішного співпадіння, здійснюється вхід у систему (Рисунок 2.2).

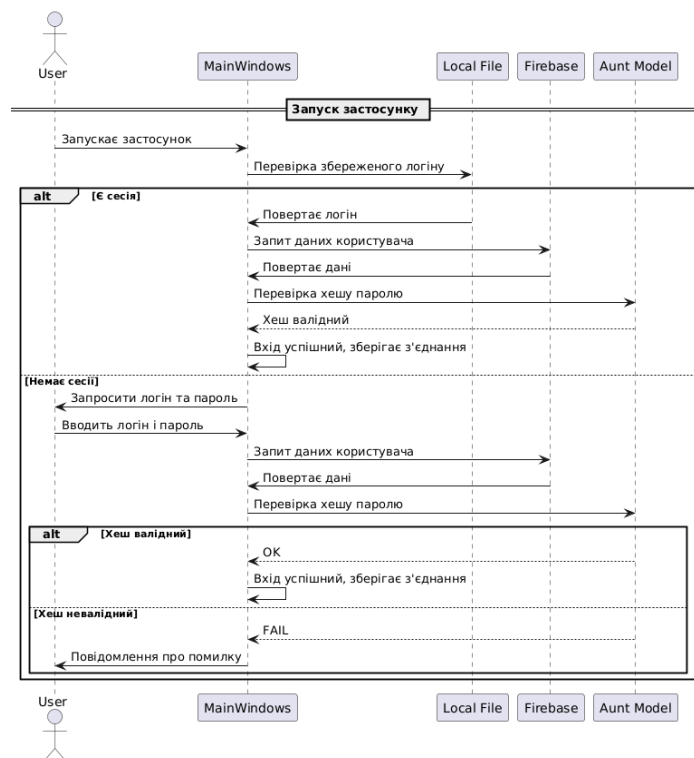


Рисунок 2.2 - Діаграма послідовності сценарію входу в акаунт

Механізм вхідного дзвінка базується на постійні перевірці змін у Firebase Realtime Database, Якщо виявлено запис про вхідний дзвінок, головний клас MainMessenger, створює екземпляр вікна дзвінку. Користувачеві надається можливість прийняти дзвінок, відхилити його або виконати додаткові дії, такі як вимкнення або ввімкнення мікрофону. Прийняття дзвінка ініціює CallWindow при натисканні на відповідну екранну кнопку, в той час Sip.cs, змінить статус та відправить клієнту з іншого боку статус про початок обміну медіапоток. У випадку якщо дзвінок не прийнято або скасовано, спеціалізованим методом, буде видалено запис про поточний дзвінок з Firebase, а на сервер буде відправлено статус Timeout або Discard відповідно (Рисунок 2.3).

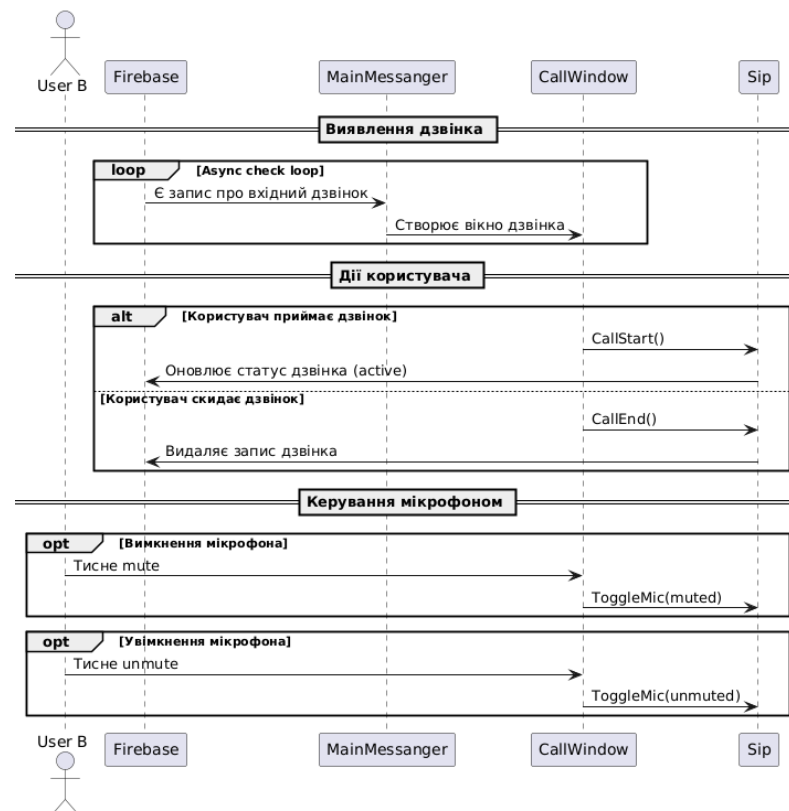


Рисунок 2.3 - Діаграма послідовності сценарію вхідного дзвінку

Головний механізм будь-якого месенджера - це обмін повідомленнями, тому це також критична частина застосунку. У процесі відправлення повідомлення, ініціатор комунікації обирає одержувача зі списку доступних контактів або шукає через пошук новий контакт та обирає його. Після чого вводить текст на клієнтському боці, після чого формується об'єкт повідомлення на базі моделі MessageResciever. Сформований об'єкт серіалізується та передається на базу даних

для зберігання та оновлення в отримувача. Отримувачу приходить нотифікація про нове повідомлення. У разі якщо відправляється файл, то перед передачею створюється запит до Firebase Storage, де відбувається вивантаження файлу, а повертається посилання на доступ до файлу, згенероване посилання передається в якості тексту у повідомлення. Система реалізується так, що в асинхронному режимі виконується перевірка оновлень на базі даних на наявність нових повідомлень, які при виявленні одразу кешуються та відображаються у вікні проєкту (Рисунок 2.4).

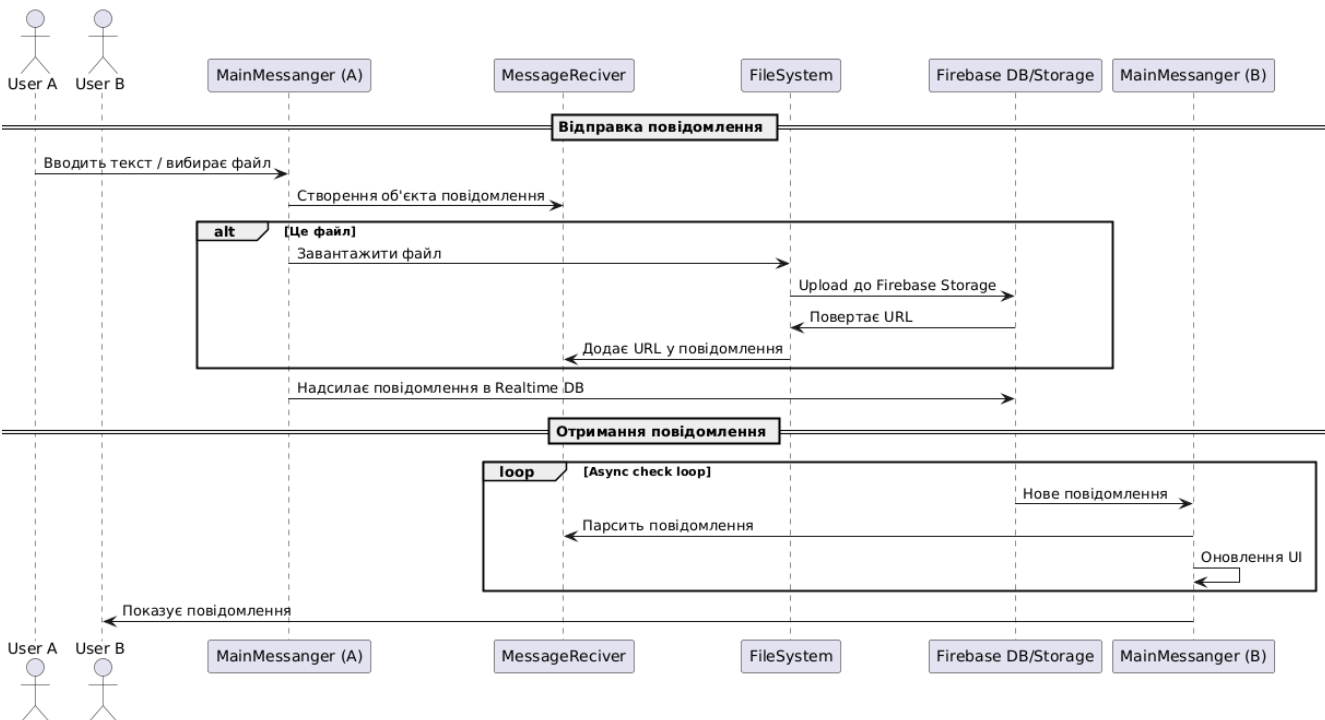


Рисунок 2.4 - Діаграма послідовності сценарію відправлення та отримання повідомлень

Для швидкого розуміння взаємодії інших компонентів у застосунку, необхідно скласти діаграму компонентів, на якій буде показано взаємодію, для правильного проєктування та виключення надлишкових декларацій компонентів.

Діаграма класів демонструє логічну структуру застосунку та його архітектуру. На діаграмі показано залежність між класами, модулями та зовнішніми сервісами.

У рамках цього застосунку MainWindows та WindowsRegistration реалізують функціональність автентифікації та реєстрації користувачів. Обидва взаємодіють з Firebase Realtime Database, для перевірки та зберігання облікових даних, а також з Aunt що відповідають за структуру даних користувача і утилітарний клас.

Центральною одиницею взаємодії є компонент MainMessenger, він забезпечує основний функціонал обміну повідомленнями, організації діалогів і групових чатів, а також обробку вхідних дзвінків. Цей компонент покладається на компоненти MessageReciever, Friend, GroupModels для представлення відповідних моделей даних, на FileSystem для керування файлами, на CallWindow для візуалізації викликів, а також використовує Utils. Взаємодія з Firebase Realtime Database виконуються у двосторонньому порядку, що відображає як надсилання так і отримання даних у асинхронному варіанті (позначається жирними стрілками), у реальному часі.

Компонент CallWindow, оболонка для управління дзвінками, здійснює взаємодію з Sip, який безпосередньо має зв'язок з Ozeki VoIP SDK. Це дозволяє ініціювати, приймати та завершувати дзвінки. В той самий час CallWindow, має взаємодію з Firebase Realtime Database для оновлення повідомлення про виклик.

Компонент FileSystem використовують для відправки файлів до Firebase Storage, а також для реєстрації посилань на файли у якості повідомлень з Firebase Realtime Database. Це дозволяє забезпечити асинхронний обмін мультимедійними даними в межах діалогів без додаткового перетворення файлів у біти та перебирання назад у файли, що займе значно більше часу процесору (Рисунок 2.5).

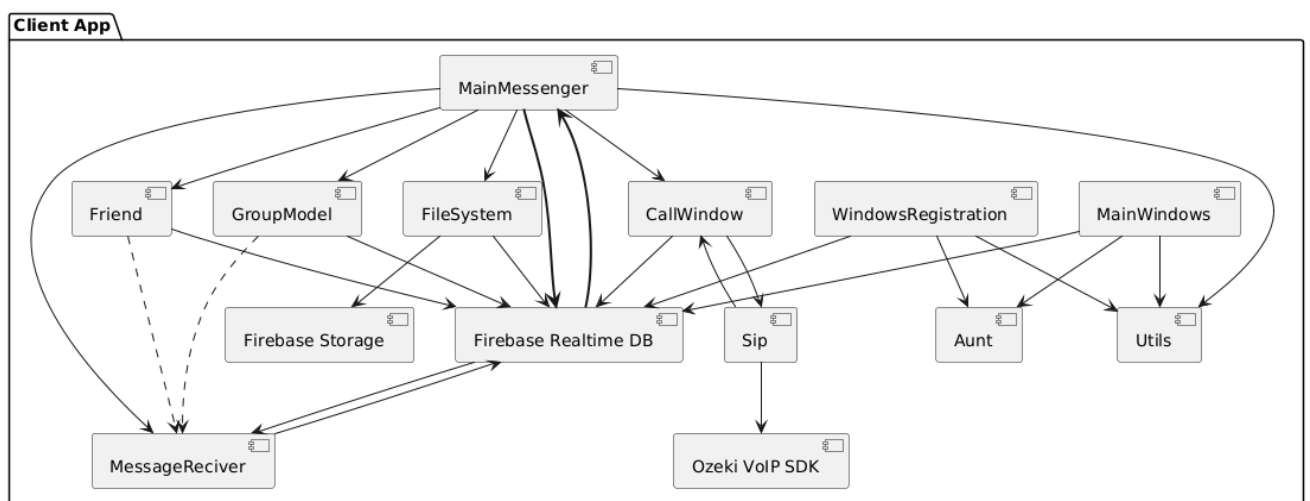


Рисунок 2.5 - UML-діаграма компонентів системи

Таким чином, можна зазначити що більша частина клієнтського застосунку, успішно спроектована, проектування дозволило встановити взаємозалежності між компонентами, а також забезпечити високий рівень модульності.

В загальному проєктування взаємодії з сервером можна зобразити на діаграмі пакетів. На діаграмі зображено розподілену систему, що складається з компонентів, розташованих в різних середовищах: Cloud, Linux і Windows. У середовищі Cloud знаходиться центральний компонент – Firebase, хмарна служба, що відповідає за обробку оновлень для інших компонентів системи. У середовищі Linux представлені два основних компоненти: Asterisk Server та Python Application. Python Application взаємодіє як з Firebase, так і з Asterisk Server. Він отримує оновлення з Firebase і надсилає команди на Asterisk Server для додавання користувачів або перезапуску системи. У середовищі "Windows" знаходиться SIP Client, реалізований на C#, який взаємодіє з Firebase і Asterisk Server. SIP Client отримує оновлення з Firebase і взаємодіє з Asterisk Server для керування SIP-з'єднаннями, такими як реєстрація та здійснення дзвінків.

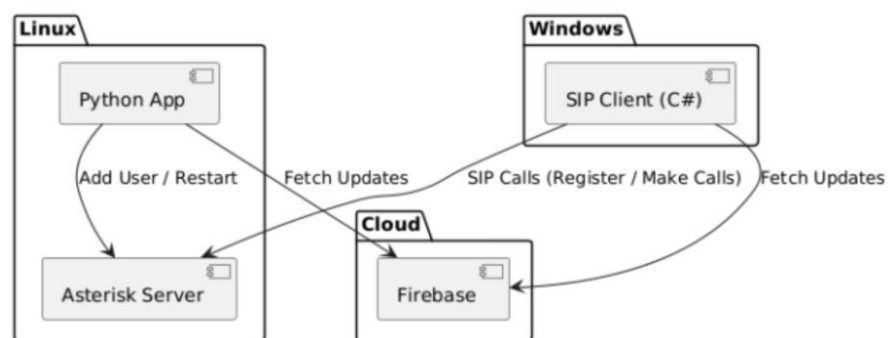


Рисунок 2.6 - Діаграма розгортання програмного комплексу

2.2. Проєктування роботи застосунку

У проєктуванні є сенс сформувати алгоритмічно-візуальне представлення роботи застосунку, для спрощення майбутньої розробки та протидії непередбачуваним результатам його роботи. Після аналізу ринку можна отримати інформацію про те, як користувач звик користуватись іншими застосунками конкурентів, за допомогою теплових мап (Рисунок 2.7).

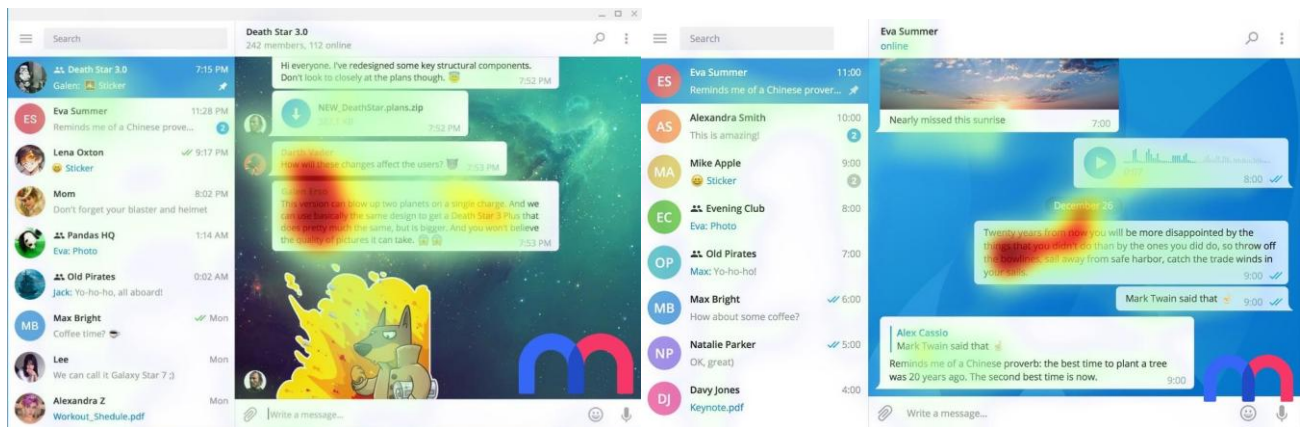


Рисунок 2.7 - Теплова карта додатків конкурентів

Судячи з теплових карт, можна вирішити, що по центру зі зсувом вправо має знаходитись поле основного контенту повідомлень, в лівій частині екрана має бути розміщено список чатів та контактів. Відповідно поля для вводу та відправки повідомлень, традиційно розміщують під полем виведення основного контенту повідомлень.

Також необхідно розробити алгоритми для спрощення реалізації критично-важливого функціоналу.

Реєстрація та вхід відбувається одразу після відкриття застосунку, вводяться дані у відповідні текстові поля, далі дані треба перевірити на коректність вводу за окремими параметрами та відповідність існуючих даних у базі даних та здійснити вибрану операцію. Відповідно в результаті необхідно відобразити наступне вікно. Цей алгоритм візуально можна побачити на рисунку 2.8.

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				40
Змн.	Арк.	№ докум.	Підпис	Дата		

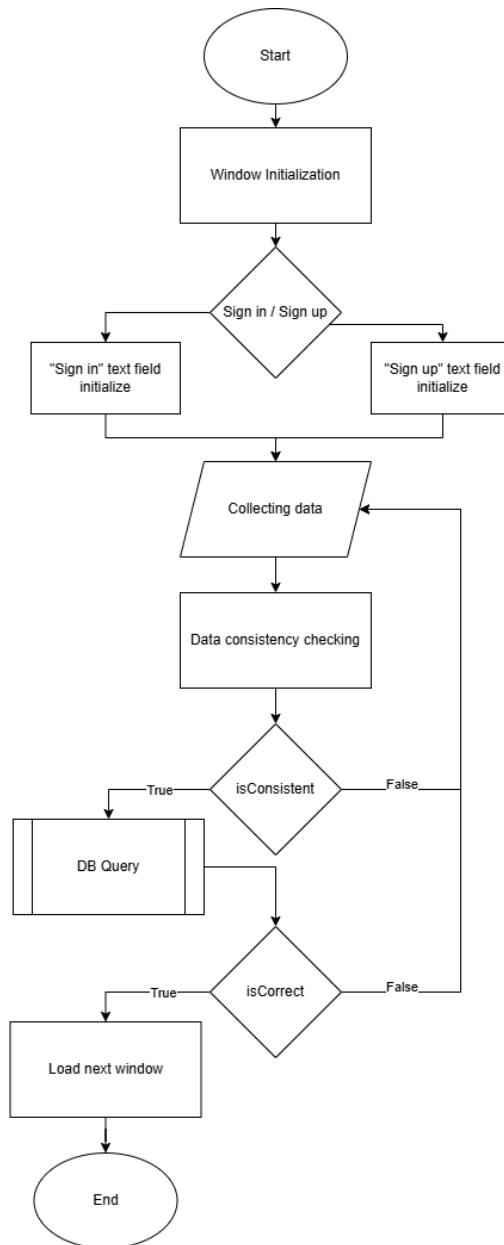


Рисунок 2.8 - Блок-схема алгоритму входу/реєстрації

Важливо продумати, процес відправки повідомлення, та очікування відповіді. Повідомлення може бути не тільки текстом, а й файлом або Емої. Процес відправки повідомлення можна описати так: користувач має авторизуватись у системі, отримати список контактів або знайти новий, обрати користувача у лівій частині застосунку, а у нижній частині застосунку потрібно написати текстове повідомлення або обрати Емої, або обрати файл, в залежності від обраного варіанту, це буде завантажувати різну логіку. Якщо це файл, то спочатку файл буде завантажено на віддалений сервер Firebase Storage, далі отримаємо посилання на файл, та відправимо як текстове повідомлення з

префіксом. Якщо це Емої то повідомлення обробляється як звичайне текстове повідомлення, оскільки Емої це частина Unicode, як і інші текстові символи.

Візуально це можна побачити на рисунку 2.9

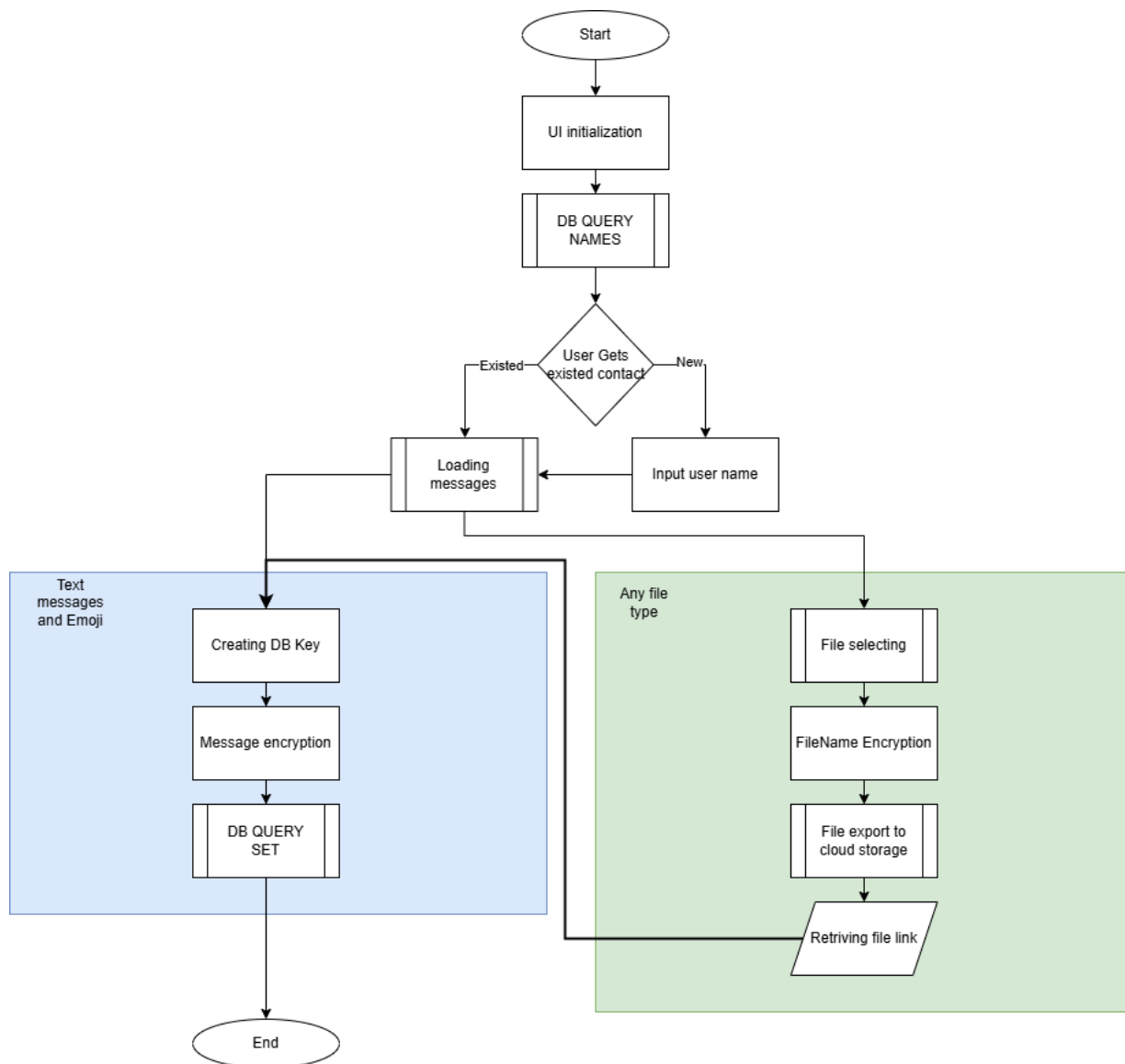


Рисунок 2.9 - Блок-схема алгоритму відправки повідомлення

Важливою частиною безпеки користувача це можливість зміни паролю, оскільки іноді можуть відбутись акти шахрайства методами “соціальної інженерії” та незаконного проникнення в акаунт користувача, необхідно дозволити користувачу змінювати пароль, для викиду інших користувачів хто увійшов під цим самим акаунтом в застосунок. На практиці така процедура, відбувається за наступним сценарієм. Користувач вводить поточний пароль та новий пароль, з повторенням, далі введені дані перетворюються на хеш (англ. Hash), та звіряються з поточним паролем на базі даних, у випадку якщо співпало, дані оновлюються на

новий хеш на базі даних, в іншому випадку користувачу пропонується ввести дані ще раз.

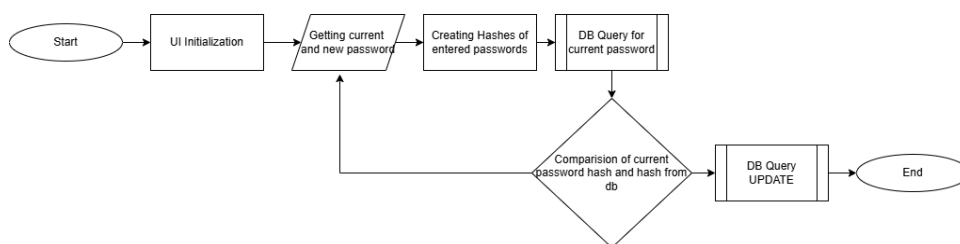


Рисунок 2.10 - Блок-схема алгоритму зміни паролю

Також потрібно продумати асинхронний алгоритм отримання повідомлень, у фоновому режимі для відображення їх на екрані, їх перетворення на формат виведення у область відображення контенту. Спочатку перевіряється чи задано, що обрано користувача діалогу, далі перевіряється чи це користувач, чи група користувачів в залежності від цього по різному отримується чи є зміни в діалозі, перевіряючи чи валідним є вибір діалогу, то робимо запит на базу і перевіряємо чи поточна версія діалогу відповідає тому що на базі даних, якщо ні, то заміняємо попередню версію діалогу на поточну та зберігаємо їх у HTML форматі, для зменшення затримки переформатовування, та виграємо зміни на CEF-браузер (Рисунок 2.11).

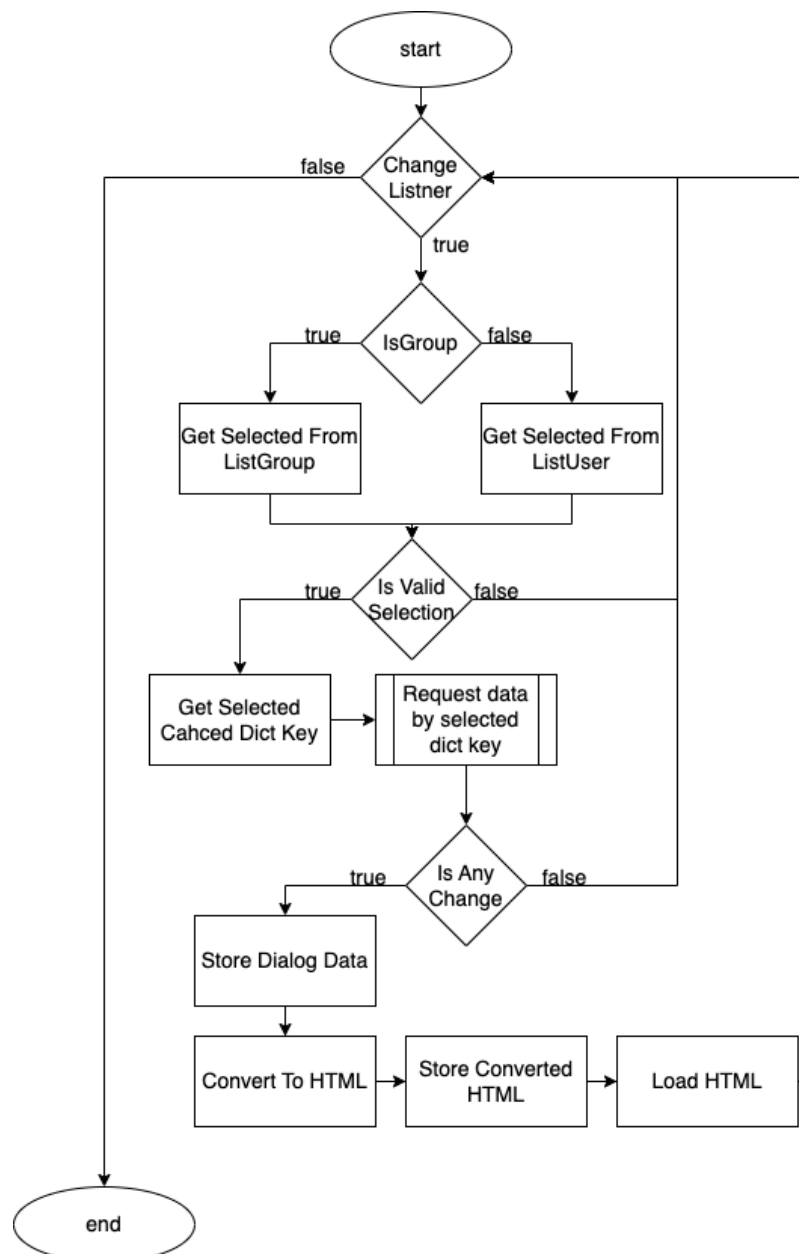


Рисунок 2.11 - Блок-схема алгоритму отримання нових повідомлень

Також здійснення дзвінку це один із критичних моментів такого типу проектів. Тому потрібно продумати простий та зрозумілий для користувача алгоритм створення дзвінку. Цю процедуру можна описати так: спочатку створюється вікно, завантажуються імена користувачів-контактів, користувач обирає контакт, відповідно до обраного користувача завантажуються повідомлення. Далі при натисканні на кнопку дзвінка, має відобразитись діалогове вікно дзвінку, на якому будуть відповідні клавіші керування, при натисканні на клавішу дзвінку, нікнейм обраного користувача перетворюється на номер контакту, перевіряється чи дзвінок буде створений поточним користувачем, та створюється

лінія за допомогою зовнішніх методів SDK. Вікно має залишатись сфокусованим, доки дзвінок триває та має мати можливість завершити фокус та закрити його після завершення дзвінку.

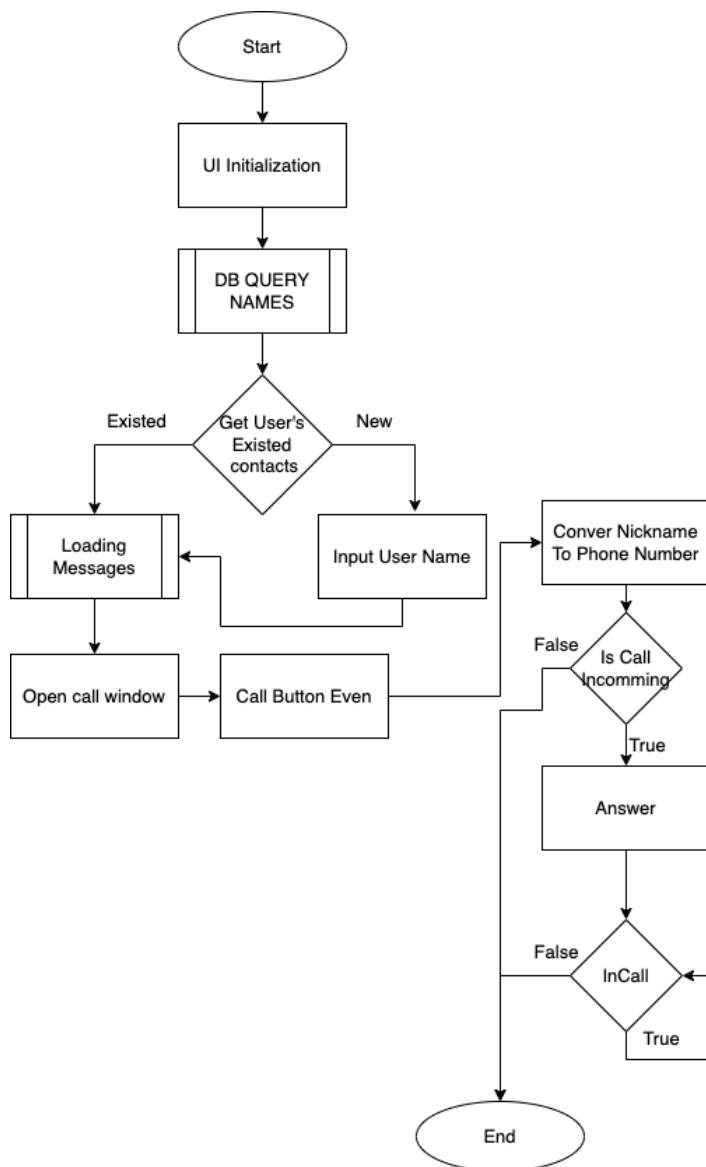


Рисунок 2.12 - Блок-схема алгоритму вихідного дзвінку

Отож маючи алгоритми критично важливих елементів застосунку, їх буде простіше розробити у подальшому.

2.3. Математичне представлення шифрування повідомлень

Усі повідомлення потребують шифрування у проєкті, для унеможливлення небажаного перехоплення пакетів. Шифрування даних є фундаментальним механізмом забезпечення інформаційної безпеки, а цілісність та автентичність

даних під час їх передавання або зберігання, є вимогою до будь-якого застосування з передачі конфіденційних даних.

Основною метою шифрування є захист інформації від несанкціонованого доступу. Шифрування додатково сприяє цілісності повідомлення, унеможливаючи непомітні зміни інформації під час передачі. Багато сучасних систем інтегрують механізми виявлення спотворень або несанкціонованої модифікації даних. [42]

Тому було розроблено окремий алгоритм шифрування під цей застосунок. Математично шифрування повідомлень можна описати так:

Нехай повідомлення складається з послідовності символів $M = (m_0, m_1 \dots m_{n-1})$, де кожен $m_i \in \{0, 1 \dots 65535\}$, тобто значення символу в межах двох байт.

Ключ формується об'єднанням основного ключа з псевдовипадковою "сіллю", тобто $K = k || s$, де s це випадкова послідовність довжиною у 8 символів.

Відкритий текст, додатково отримує сіль перед тим як почати шифруватись:

$$M' = s + M \quad (2.1)$$

Процедура шифрування символу будується, таким чином, що для кожного символу повідомлення m_i обчислюється зміщення за допомогою функції:

$$f(i, j, K, S) = |(K_j \oplus [\log(i + 2) * 17 + \sin((i + 1) * \pi) * \cos((S + 1) * \varphi * 255 + (S \bmod 31))]) \bmod 65536| \quad (2.2)$$

де " i " це індекс символу в повідомленні, " j " це індекс ключа, що циклічно оновлюється, " S " це акумульована сума всіх попередніх не зашифрованих символів, " φ " це число золотого перетину (приблизно 1,61803398875).

Використовуючи знайдене зміщення зашифрований символ C_i обраховується як:

$$c_i = (m_i \oplus f(i, j, K, S)) \bmod 65536 \quad (2.3)$$

Після кожної операції акумулятор " S " оновлюється:

$$S = S + m_i \quad (2.4)$$

А індекс ключа, змінюється циклічно

$$j = (j + 1) \bmod |K| \quad (2.5)$$

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		46

По результату шифрування, дані кодуються у формат який дозволяє передати дані без їх спотворення:

$$Base64(Encrypt(s||M', K||s)) \quad (2.6)$$

Процес шифрування можна зобразити на блок схемі у додатку В

Для забезпечення зворотності шифрування та можливості читання відкритого тексту на стороні одержувача, необхідно продумати і розшифрування шифро тексту.

Згідно з попередніми формулами шифрування, дешифрування виконуватиметься аналогічною процедурою, оскільки операція побітового або є самопервертаючою, тому дешифрування можна описати так:

$$m_i = (c_i \oplus f(i, j, K, s)) \bmod 65536$$

а, акумулятор "S" оновлюється за вже розшифрованими символами.

Крипостійкість такого алгоритма базується на декількох факторах.

Пункт перший, псевдо випадковість зміщення кожного символу залежить не лише від його позиції, а й від сумарного значення попередніх символів, що створює залежність від історичного контенту. Додавання синусо-косинусних хаотичних компонентів з ірраціональними базами π та ϕ підвищує непередбачуваність.

Пункт другий, випадкова сіль, кожне шифрування одного і того ж самого відкритого тексту генерує інший результат через зміну солі, що унеможливорює атаки типу “зіставлення відомих пар” (plaintext-chifertext pair).

Пункт третій, композиція з ключем, сіль вплітається в ключ, що ще більше заплутує залежність між відкритим текстом та шифро текстом.

Пункт четвертий, криптоаналітична стійкість, через динамічну зміну послідовності зсуву, для кожного символу, атаки типу “аналіз частот” (frequency analisys) неефективні, оскільки однакові символи на різних позиціях шифруються по різному.

Оцінити таке шифрування на крипостійкість найпростіше через оцінку стійкості до атак, оскільки деякі типи атак були відкинуті через тип шифрування, можна перевірити їх за допомогою інших атак.

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		47

Через найрозповсюдженішу атаку брутфорс, складність можна оцінити як пропорційну 2^b , де b це ефективна ентропія ключу та солі, наприклад якщо ключ має 128 біт та 8 байтів солі, це понад 2^{192} варіантів.

До атаки Chosen-plaintext attack складність підвищена оскільки шифр має компоненту хаотичного зміщення і акумуляцію суми S .

Атака Known-plaintext attack ускладнена через змінність ключа та використання сольових даних.

Висновки до 2 розділу

У другому розділі проведено велику роботу над дослідженням обраних технологій, обумовлено їх використання. Розписано причини вибору та принцип роботи оточення технологій. Описано допоміжні інструменти, аргументовано їх використання та оглянуто їх відмінності та можливості.

Проведено теоретичне проектування системи, сформовані у вигляді UML-діаграм. Теоретично оформлено варіанти використання системи для людини. Сформовано та описано основні сценарії критичних для додатку подій. Сформовано компонентну структуру застосунка-клієнта. Теоретизовано пакетну діаграму на якій видно взаємодію компонентів комплексу.

Докладено інформацію про тип шифрування повідомлень у системі, представлені алгоритми та математичні формули, проведено аналіз криптостійкості алгоритму шифрування.

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		48

РОЗДІЛ 3. РЕАЛІЗАЦІЯ, РОЗГОРТАННЯ ТА ТЕСТУВАННЯ СИСТЕМИ, ІНСТРУКЦІЯ З ПОРЯДКУ ВИКОРИСТАННЯ

3.1. Реалізація клієнт-серверної частини

3.1.1. Реалізація серверної частини

Для початку було встановлено операційну систему Ubuntu на базі Linux. З віддаленого репозиторія завантажено заархівований пакет інструкцій та команд для операційної системи та дистрибутив серверу з відкритим програмним кодом Asterisk. Asterisk-сервер: проєкт з відкритим кодом, до якого будь-хто може долучитись для розробки. Після його завантаження необхідно опрацювати команди серверу.

Для встановлення останньої стабільної версії, на вже існуючому Windows Server встановлюю віртуальну машину на дистрибутивів Linux Ubuntu.

Після встановлення необхідно оновити репозиторії. (Рисунок 3.1)

```
vboxuser@UBUNTU:~$ sudo apt update && sudo apt upgrade -y
[sudo] password for vboxuser:
Hit:1 http://ua.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://ua.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://ua.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
63 packages can be upgraded. Run 'apt list --upgradable' to see them.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
The following upgrades have been deferred due to phasing:
  python3-distupgrade ubuntu-release-upgrader-core ubuntu-release-upgrader-gtk
The following packages will be upgraded:
  apparmor cloud-init distro-info-data dmsetup fwupd gir1.2-gtk-3.0
  gir1.2-mutter-14 gnome-initial-setup gnome-shell gnome-shell-common
```

Рисунок 3.1 - Оновлення репозиторіїв

Для роботи Asterisk, необхідно налаштувати оточення, завантаживши необхідні інструменти. (Рисунок 3.2)

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				49
Змн.	Арк.	№ докум.	Підпис	Дата		

```
vboxuser@UBUNTU:~$ sudo apt install build-essential libssl-dev libncurses5-dev libnewt-dev libxml2-dev libsqlite3-dev uuid-dev libjansson-dev libksemel-dev -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Note, selecting 'libncurses-dev' instead of 'libncurses5-dev'
The following additional packages will be installed:
  binutils binutils-common binutils-x86-64-linux-gnu bzip2 dpkg-dev fakeroot g++ g++-13 g++-13-x86-64-linux-gnu
  g++-x86-64-linux-gnu gcc gcc-13 gcc-13-x86-64-linux-gnu gcc-x86-64-linux-gnu icu-devtools libalgorithm-diff-perl
  libalgorithm-diff-xs-perl libalgorithm-merge-perl libasan8 libbinutils libbct1-0 libctf-nobfd0 libctf0 libdpkg-perl
  libfakeroot libfile-fcntllock-perl libgcc-13-dev libgprofng0 libhwasan0 libicu-dev libksemel3 libitm1 liblsan0
  libpng-dev libpng-tools libquadmath0 libstdc++-13-dev libstdc++-13 libtsan2 libubsan1 lto-disabled-list
  make zlib1g-dev
Suggested packages:
  binutils-doc gprofng-gui bzip2-doc debconf debconf-archive debconf-i18n g++-multilib g++-13-multilib gcc-13-doc gcc-multilib autoconf
  automake libtool flex bison gcc-doc gcc-13-multilib gcc-13-locales gdb-x86-64-linux-gnu git bzr icu-doc ncurses-doc
  sqlite3-doc libssl-doc libstdc++-13-doc pkg-config make-doc
The following NEW packages will be installed:
  binutils binutils-common binutils-x86-64-linux-gnu build-essential bzip2 dpkg-dev fakeroot g++ g++-13
  g++-13-x86-64-linux-gnu gcc gcc-13 gcc-13-x86-64-linux-gnu gcc-x86-64-linux-gnu icu-devtools
  libalgorithm-diff-perl libalgorithm-diff-xs-perl libalgorithm-merge-perl libasan8 libbinutils libbct1-0 libctf-nobfd0
  libctf0 libdpkg-perl libfakeroot libfile-fcntllock-perl libgcc-13-dev libgprofng0 libhwasan0 libicu-dev
  libksemel-dev libksemel3 libitm1 libjansson-dev liblsan0 libncurses-dev libnewt-dev libpng-dev libpng-tools
  libquadmath0 libstdc++-13-dev libstdc++-13 libtsan2 libubsan1 libxml2-dev
  lto-disabled-list make uuid-dev zlib1g-dev
0 upgraded, 53 newly installed, 0 to remove and 3 not upgraded.
Need to get 74.9 MB of archives.
After this operation, 287 MB of additional disk space will be used.
Get:1 http://ua.archive.ubuntu.com/ubuntu noble-updates/main amd64 binutils-common amd64 2.42-4ubuntu2.3 [239 kB]
Get:2 http://ua.archive.ubuntu.com/ubuntu noble-updates/main amd64 libstdc++6 amd64 12.3.0-2ubuntu1 [613 kB]
```

Рисунок 3.2 - встановлення інструментів

Сам Asterisk-сервер неможливо завантажити за допомогою менеджера інструментів apt. Його остання стабільна версія завантажується за допомогою wget за посиланням з офіційного сайту. (Рисунок 3.3)

```
vboxuser@UBUNTU:~$ wget https://downloads.asterisk.org/pub/telephony/asterisk/asterisk-22-current.tar.gz
--2024-11-11 19:29:35-- https://downloads.asterisk.org/pub/telephony/asterisk/asterisk-22-current.tar.gz
Resolving downloads.asterisk.org (downloads.asterisk.org)... 165.22.184.19, 2604:a880:400:d0::14:9001
Connecting to downloads.asterisk.org (downloads.asterisk.org)[165.22.184.19]:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 26196468 (25M) [application/octet-stream]
Saving to: 'asterisk-22-current.tar.gz'

asterisk-22-current.tar.gz 100%[=====] 24.98M 9.29MB/s in 2.7s

2024-11-11 19:29:39 (9.29 MB/s) - 'asterisk-22-current.tar.gz' saved [26196468/26196468]
```

Рисунок 3.3 - встановлення інструментів

Після встановлення необхідно розпакувати архів. (Рисунок 3.4)

```
vboxuser@UBUNTU:~$ tar xvf asterisk-22-current.tar.gz
asterisk-22.0.0/
asterisk-22.0.0/.cleancount
asterisk-22.0.0/.gitignore
asterisk-22.0.0/.version
asterisk-22.0.0/BSDmakefile
asterisk-22.0.0/BUGS
asterisk-22.0.0/CHANGES.md
asterisk-22.0.0/COPYING
asterisk-22.0.0/CREDITS
asterisk-22.0.0/ChangeLogs/
asterisk-22.0.0/ChangeLogs/ChangeLog-22.0.0.md
asterisk-22.0.0/LICENSE
asterisk-22.0.0/Makefile
asterisk-22.0.0/Makefile.moddir_rules
asterisk-22.0.0/Makefile.rules
```

Рисунок 3.4 - Розпаковка Asterisk серверу

		Юхимчук Н.О.			Арк.
		Граф М.С.			50
Змн.	Арк.	№ докум.	Підпис	Дата	

Сам пакунок з сервером має у собі список ресурсів необхідних перед встановленням, їх також необхідно встановити перед початком збирання серверу. (Рисунок 3.5)

```
vboxuser@UBUNTU:~/asterisk-22.0.0$ sudo contrib/scripts/install_prereq install
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  aptitude-common libboost-iostreams1.83.0 libcwidget4 libsigc++-2.0-0v5 libxapian30
Suggested packages:
  apt-xapian-index aptitude-doc-en | aptitude-doc debtags tasksel libcwidget-dev xapian-tools
The following NEW packages will be installed:
  aptitude aptitude-common libboost-iostreams1.83.0 libcwidget4 libsigc++-2.0-0v5 libxapian30
0 upgraded, 6 newly installed, 0 to remove and 3 not upgraded.
Need to get 4,259 kB of archives.
After this operation, 19.2 MB of additional disk space will be used.
Get:1 http://ua.archive.ubuntu.com/ubuntu noble/universe amd64 aptitude-common all 0.8.13-5ubuntu5 [1,800 kB]
Get:2 http://ua.archive.ubuntu.com/ubuntu noble/main amd64 libboost-iostreams1.83.0 amd64 1.83.0-2.1ubuntu3 [259 kB]
Get:3 http://ua.archive.ubuntu.com/ubuntu noble/main amd64 libsigc++-2.0-0v5 amd64 2.12.1-2 [12.8 kB]
Get:4 http://ua.archive.ubuntu.com/ubuntu noble/universe amd64 libcwidget4 amd64 0.5.18-6build1 [241 kB]
Get:5 http://ua.archive.ubuntu.com/ubuntu noble/universe amd64 libxapian30 amd64 1.4.22-1build1 [716 kB]
```

Рисунок 3.5 - встановлення ресурсів

Далі потрібно налаштувати компілятор мови C++, для успішної збірки пакетів та скриптів. (Рисунок 3.6)

```
vboxuser@UBUNTU:~/asterisk-22.0.0$ ./configure
checking build system type... x86_64-pc-linux-gnu
checking host system type... x86_64-pc-linux-gnu
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether the compiler supports GNU C... yes
checking whether gcc accepts -g... yes
checking for gcc option to enable C11 features... none needed
checking for stdio.h... yes
checking for stdlib.h... yes
checking for string.h... yes
checking for inttypes.h... yes
```

Рисунок 3.6 - виконання конфігурації компілятора

Та примінити конфігурацію до компілятора (Рисунок 3.7)

```
vboxuser@UBUNTU:~/asterisk-22.0.0$ sudo make config
vboxuser@UBUNTU:~/asterisk-22.0.0$ sudo ldconfig
```

Рисунок 3.7 - конфігурація компілятора

Фінальним кроком стане початок компіляції збірки серверу Asterisk. (Рисунок 3.8)

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				51
Змн.	Арк.	№ докум.	Підпис	Дата		

```
vboxuser@UBUNTU:~/asterisk-22.0.0$ sudo make install
```

Рисунок 3.8 - Запуск компіляції

Налаштовуємо кінцеві точки, порти та користувача для перевірки роботи.

У файлі sip.conf налаштовується робота через «Білу» IP-адресу, що не змінюється та дозволяє підключитись до серверу з будь-якого пристрою за межами мережі, також стандартні порти для підключення різних користувачів.

У користувачів налаштовується тип підключення на динамічне, оскільки користувач може приєднатись до іншої мережі, але, знаючи дані для входу на сервер та IP-адресу, може приєднатись до дзвінку. (Рисунок 3.9)

```
GNU nano 7.2 /etc/asterisk/sip.conf
[general]
context=default
allowguest=no
allowoverlap=no
udpbindaddr=0.0.0.0
transport=udp
tcpenable=no
tcpbindaddr=0.0.0.0
disallow=all
allow=alaw

nat=force_rport,comedia
externaddr=185.249.161.189
localnet=192.168.0.113/24

rtpstart=10000
rtpend=20000

[1001]
type=friend
host=dynamic
secret=1234
nat=force_rport,comedia
```

Рисунок 3.9 - налаштування у sip.conf

Також необхідно додати користувачів у extentions.conf за допомогою команди exten налаштовуємо користувача

exten => <номер>,<пріоритет>,<дія> (3.1)

Якщо налаштувати користувача таким чином exten => 1001,1,Dial(SIP/1001,10), то 1001, це номер на який буде набиратися. 1, це пріоритет кроку виконання дзвінка, його можна залишити однаковим для усіх користувачів, тоді пріоритет буде визначатись по мілісекундам, від того хто перший дзвонить. У свою чергу

Dial(SIP/1001,10) буде вказуватись скільки секунд будуть йти гудки, до того як дзвінок буде скасовано, якщо на іншій стороні не взяли слухавку. (Рисунок 3.10)

```
[general]

[default]
exten => 1001,1,Dial(SIP/1001,10)
exten => 1002,1,Dial(SIP/1002,10)
exten => 116101115116,1,Dial(Sip/116101115116,10)
exten => 11610111511650,1,Dial(Sip/11610111511650,10)
```

Рисунок 3.10 - налаштування у extentions.conf

Оскільки сервер після компіляції та запуску запустився автоматично для закріплення щойно зроблених конфігурацій необхідно перезавантажити сервер. (Рисунок 3.11)

```
a@a-VirtualBox:~$ sudo asterisk -rx "reload"
```

Рисунок 3.11 - перезапуск серверу

Для того щоб віртуальна машина мала змогу отримувати дані з інтернету для неї потрібно зробити фіксацію IP-адреси на роутері, та відкрити порти. Оскільки SIP використовує зовнішні порти з діапазону 60001-84000, необхідно відкрити порти на роутері, для забезпечення переадресації їх на локальні порти віртуальної машини.

Для цього необхідно перейти, до налаштувань роутеру, наприклад 192.168.0.1, для роутерів компанії TP-Link. (Рисунок 3.12)

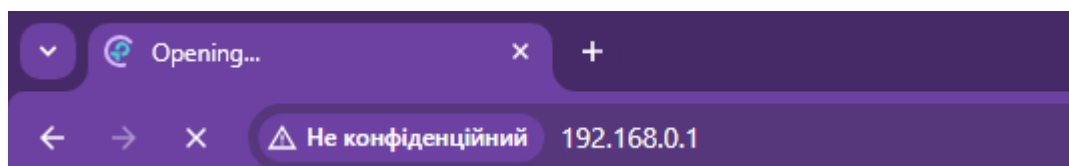


Рисунок 3.12 - сторінка налаштувань роутеру

Після авторизації необхідно перейти до розділу мережа, де обрати підпункт налаштувань DHCP-серверу (Рисунок 3.13)

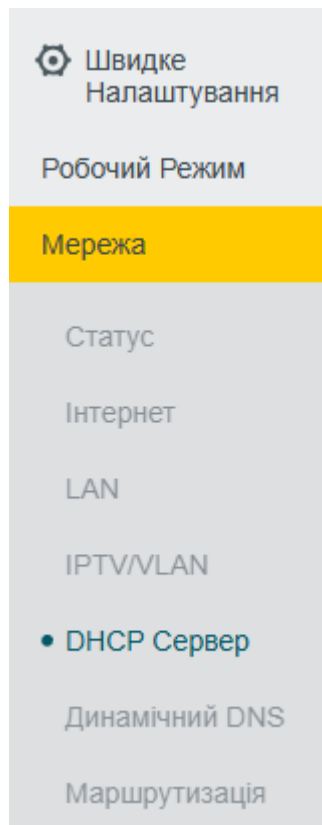


Рисунок 3.13 - Розділ мережа

Тепер резервуємо IP-адресу за mac-адресою віртуальної машини, задану IP-адресу. (Рисунок 3.14-3.17)

Змінити запис про резервування
×

MAC-адреса:

00 - D8 - 61 - 8A - A4 - 3D

ПЕРЕГЛЯНУТИ ПІДКЛЮЧЕНІ ПРИБОРИ

IP-адреса:

192.168.0.113

ВІДМІНИТИ

ЗБЕРЕГТИ

Рисунок 3.14 - Резервування IP-адреси

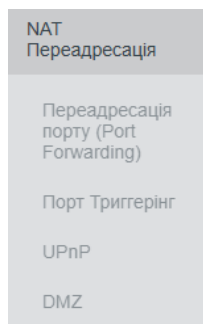


Рисунок 3.15 - Розділ переадресації порту

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				54
Змн.	Арк.	№ докум.	Підпис	Дата		

Aster	192.168.0.113	60001	5060	Vci		 
ast	192.168.0.170	60013	5060	Vci		 

Рисунок 3.16 - Активація переадресації портів

Редагувати

Ім'я служби:

Aster

ПЕРЕГЛЯНУТИ ЗАГАЛЬНІ ФУНКЦІЇ

IP-адреса пристрою:

192.168.0.113

ПЕРЕГЛЯНУТИ ПІДКЛЮЧЕНІ ПРИСТРОЇ

Зовнішній порт:

60001

Внутрішній порт:

5060

Протокол:

Vci

☒ Увімкнути цей запис

ВІДМІНИТИ

ЗБЕРЕГТИ

Рисунок 3.17 - Налаштування переадресації портів

На сервері запущений Python application який спілкується з базою для пошуку оновлень.

```
def check_for_new_entry(ref, last_value):
    current_value = ref.get() # Get the current data from the
    database
    if current_value != last_value:
        print("True: Data changed!")

    # Find the newly added entry by comparing the keys
    last_keys = set(last_value.keys()) if last_value else set()
    current_keys = set(current_value.keys())
    new_keys = current_keys - last_keys # Get the newly added
keys

    for new_key in new_keys:
        new_entry_data = current_value[new_key]
        # Handle nested structure within new entries
        for sub_key, sub_data in new_entry_data.items():
            if `NameSip` in sub_data and `PasswordSip` in
```

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				55
Змн.	Арк.	№ докум.	Підпис	Дата		

sub_data:

```
        sip_name = sub_data['NameSip']
        sip_password = sub_data['PasswordSip']
        print(f"New SIP Name: {sip_name}")
        print(f"New SIP Password: {sip_password}")

        # Update the files with the new SIP details
        update_files(sip_name, sip_password)

    return current_value
return last_value
```

Функція `check_for_new_entry` є частиною програми, яка виконує перевірку на зміни в базі даних, на наявність нових записів або оновлень існуючих даних. Її основною метою є виявлення нових даних, що з'явилися в системі, і подальша обробка цих даних, для оновлення конфігураційних файлів з новими обліковими даними користувачів SIP.

Для роботи функція отримує посилання на об'єкт `ref`, який відповідає за взаємодію з базою даних, та `last_value`, що є останнім збереженим станом даних. Спочатку виконується отримання поточних даних за допомогою методу `ref.get()`, який дозволяє отримати актуальний стан бази даних. Поточні дані зберігаються в змінній `current_value`.

Для виявлення нових записів функція порівнює множини ключів у двох об'єктах у поточному та попередньому значенні. Якщо в поточному значенні є нові ключі, які відсутні в попередньому, це свідчить про наявність нових записів. Множини ключів обчислюються за допомогою операцій над множинами. Від поточних ключів віднімаються ключі з попереднього значення, що дозволяє отримати набір нових ключів. Математично це можна записати так

$$\text{Update} = \text{NewData} \setminus \text{OldData}$$

Кожен новий запис, який містить ключі, не представлені в попередньому значенні, піддаються детальній обробці. Якщо новий запис містить вкладені дані, функція перевіряє наявність специфічних полів, таких як `NameSip` та `PasswordSip`, що є обліковими даними користувача для SIP системи. Якщо ці поля знайдені, їх значення виводяться в консоль, для логування, і передаються далі.

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		56

Отримані нові дані, зокрема, ім'я користувача SIP та пароль SIP, передаються в функцію `update_files`, яка відповідає за оновлення локальних файлів або інших конфігураційних ресурсів з новими обліковими даними. Після виконання всіх перевірок і обробки нових даних функція повертає поточне значення (`current_value`). Це дозволяє зберігати актуальні дані для порівняння з майбутніми змінами в базі. Якщо змін не було виявлено, повертається старе значення (`last_value`), що гарантує збереження стабільності роботи скрипту.

```
def update_files(sip_name, sip_password):
    # Format for a.txt
    a_txt_content = f"""
\n\n
[{sip_name}]
type=friend
host=dynamic
secret={sip_password}
nat=force_rport,comedian
"""

    with open(`/etc/asterisk/sip.conf`, `a`) as a_file: # Append
mode
        a_file.write(a_txt_content)
        print(f"a updated with SIP Name: {sip_name}")

    # Format for b.txt
    b_txt_content = f"exten =>
{sip_name},1,Dial(Sip/{sip_name},100)\n"
    with open(`/etc/asterisk/extensions.conf`, `a`) as b_file: #
Append mode
        b_file.write(b_txt_content)
        print(f"b updated with SIP Name: {sip_name}")

def run_linux_command():
    command = `sudo asterisk -rx "dialplan reload"`
    try:
        subprocess.run(command, shell=True, check=True)застосунок
    except:
        print("Failed to reload")
```

Функція `run_linux_command` призначена для перезавантаження окремих компонентів Asterisk-сервера за допомогою виклику команди в операційній системі Linux. Ця операція є необхідною для застосування змін у конфігурації Asterisk, що дозволяє оновлювати налаштування без необхідності повного перезапуску серверу.

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		57

У функції створюється командний рядок, що містить інструкцію для виконання на сервері. Команда `sudo asterisk -rx "dialplan reload"` є типовим способом перезавантаження конфігурації Asterisk через командний рядок. Використання команди `sudo` забезпечує надання прав адміністратора, необхідних для перезавантаження сервісу. Для виконання цієї команди використовуються можливості бібліотеки `subprocess` в Python. Виклик функції `subprocess.run(command, shell=True, check=True)` запускає команду через оболонку (`shell=True`) і перевіряє, чи не призводить її виконання до помилки (`check=True`). Якщо команда завершується з помилкою (наприклад, через відсутність прав доступу або проблеми з самим сервером Asterisk), буде викликана виключна ситуація. Якщо виникає помилка при виконанні команди, система виводить повідомлення "Failed to reload", що дозволяє користувачеві або адміністраторам відразу дізнатися про існування проблеми.

Функція `update_files` відповідає за оновлення двох текстових конфігураційних файлів. Ці файли використовуються для налаштування SIP-облікових записів і визначення правил маршрутизації для SIP-викликів. Обробка та оновлення цих файлів є важливим кроком у налаштуванні SIP-сервера для підключення нових користувачів або зміни існуючих налаштувань.

3.1.2. Реалізація клієнтської частини

При початку роботи над програмою необхідно створити можливість авторизації, реєстрації та автоматичного входу при наявності попередньої сесії у застосунок, тому було розроблено код, який допомагає це реалізувати.

`button1_Click`, це обробник подій натискання на кнопку Login. Вона реалізує авторизацію користувача через Firebase.

Спочатку ініціалізуються поля класу, в які записуються значення з текстових полів на користувацькому інтерфейсі, туди зберігається логін і пароль.

Після чого відбувається перевірка заповненості полів, якщо хоча б одне поле не заповнено, виводиться повідомлення "Заповніть всі поля". Генерируется ID для доступу до даних та за ними отримуються дані користувача, якщо користувач існує `ResUser` має його містити. Далі відбувається перевірка ідентичності користувачів,

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		58

якщо вони співпали, то відкривається головне вікно проєкту, в іншому випадку, виводиться повідомлення про помилку.

```
private void button1_Click(object sender, RoutedEventArgs e)
{
    current_User = Boxnick.Text;
    current_NameSip = Boxnick.Text;
    current_PasswordSip = Boxpassword.Text;
    if (string.IsNullOrEmpty(Boxnick.Text) &&
string.IsNullOrEmpty(Boxnick.Text))
    {
        MessageBox.Show("Заповніть всі поля");
        return;
    }
    var createIdFromUser = Encoding(Encoding(Boxnick.Text) +
Encoding(Boxpassword.Text));

    FirebaseResponse res = client.Get(@"users/" +
Boxnick.Text + "/" + createIdFromUser);
    Aunt ResUser = res.ResultAs<Aunt>();
    Aunt CurUser = new Aunt()
    {
        Name = Boxnick.Text,
        Password = Boxpassword.Text
    };
    if (Boxnick.Text != Name)
    {
        if (Aunt.IsEqual(ResUser, CurUser))
        {
            SaveCurrentUser(CurUser);
            MainMessenger authWindow = new
MainMessenger();
            Hide();
            authWindow.ShowDialog();
            Environment.Exit(0);
        }
        else errorLabel.Content = Aunt.Errormasseg();
    }
    else MessageBox.Show("T");
}
```

Дані користувача потрібно зберігати для того щоб не заставляти його авторизуватись при кожному запуску програми, для цього написана функція SaveCurrentUser. Метод є приватним для забезпечення безпеки дії, з іншого місця

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				59
Змн.	Арк.	№ докум.	Підпис	Дата		

до нього не дістатись. Файл з даними буде зберігатись у %AppData%, у бінарному форматі. У методі створено потік, для серіалізації об'єкта та бінарний формater, який перетворить дані у байти. Далі через Stream, запишеться у файл.

```
private void SaveCurrentUser(Aunt curUser)
{
    using (Stream stream =
        File.Open(Environment.GetFolderPath(Environment.SpecialFolder.Applic
            ationData) + "/loggedUser.bin", FileMode.Create))
    {
        var beanaryFormater = new
            System.Runtime.Serialization.Formatters.Binary.BinaryFormatter();
        beanaryFormater.Serialize(stream, curUser);
    }
}
```

Розроблено асинхронний обробник подій Button2_Click, котрий виконує декілька важливих функцій для забезпечення реєстрації користувача у системі. Метод робить запит, по шляху де розміщені поточні користувачі на базу даних, якщо в результаті запиту у змінній записано null, значить такого користувача ще немає, і можна його реєструвати. Далі виконується перевірка на правильність електронної адреси з використанням регулярного виразу після цього, якщо електронна адреса, правильна, скрипт заповнює об'єкт класу aunt, генерує для нього унікальний ID, та зберігає його у місце сховища користувачів. Якщо щось з введеного не підійшло, виводить помилку біля відповідного поля на користувацькому інтерфейсу.

```
private async void Button2_Click(object sender, EventArgs e)
{
    Aunt ReUser = null;
    client = new FireSharp.FirebaseClient(ifc);
    FirebaseResponse res = client.Get(@"users/" +
        Boxname.Text);
    if (res.Body == "null")
    {
        if (Regex.IsMatch(Boxmail.Text, "[a-z0-9]+@[a-z0-
            9]+\.[a-z]"))
        {
            client = new FireSharp.FirebaseClient(ifc);

            Aunt user = new Aunt()
```

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				60
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        {
            Name = Boxname.Text,
            Password = Boxpassword.Text,
            Mail = Boxmail.Text,
            NameSip = Boxname.Text,
            PasswordSip = Boxpassword.Text,
        };
        var createIdFromUser =
Encoding(Encoding(Boxname.Text) + Encoding(Boxpassword.Text) );
        SetResponse set = client.Set(@"users/" +
Boxname.Text + "/" + createIdFromUser , user);
        MainWindow authWindow = new MainWindow();
        authWindow.Show();
        Hide();
    }
    else
    {
        MessageBox.Show("Це email вже
використовується");
    }
}
else
{
    MessageBox.Show("Це і`мя вже використовується");
}
}

```

Після авторизації користувача, якщо це не перша його спроба користуватись застосунком, можуть бути контакти та діалоги та щоб це побачити необхідно загрузити пари користувачів та на основі існуючих пар користувачів та груп, треба вигрузити та закешувати існуючі діалоги, що зменшить час очікування завантаження, та зменшить навантаження на клієнт бази даних під час відкриття історії повідомлень.

Для цього було написано два методи PushNames, який завантажує пари та LoadAllAvialiableDialogData який завантажує історію повідомлень та хешує. Вони виконуються при запуску.

PushNames, при початку виконання, звертається за адресою UsersPriviousPairs та нікнеймом поточного користувача і викачує звідти все. За

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		61

допомогою десеріалізації з JSON формату відповіді витягуються усі екземпляри класу Friend, які формують словник, далі за допомогою циклу, якщо результат не пустий у listbox записуються імена людей з якими є пара.

```

        void PushNames()
        {
            FirebaseResponse res =
MainWindow.client.Get(@"UsersPreviousPairs/" + MainWindow.current_User +
"/");
            Dictionary<string, Friend> result =
JsonConvert.DeserializeObject<Dictionary<string,
Friend>>(res.Body.ToString());
            if (result != null)
            {
                foreach (var item in result)
                {
                    ListUser.Items.Add(item.Value.Name);
                }
            }
        }

```

Одразу після цього виконуються LoadAllAvialiableDialogData перебирає усі імена які було додано, робить запит на адресу де зберігаються повідомлення, та вивантажує їх, окремо зберігає відповідь, як дані, окремо зберігає переформатовану версію у html, для швидкого завантаження даних. Та проводить ту саму операцію з груповими діалогами.

```

private async void LoadAllAvialiableDialogData()
{
    foreach (string name in ListUser.Items)
    {
        var result = await MainWindow.client.GetAsync("message/" +
CreateKeyForDB(name, MainWindow.current_User) + "/");
        try
        {
            dialogData.Add(name, result.Body.ToString());
            dialogs.Add(name,
ConvertToHtml(JsonConvert.DeserializeObject<Dictionary<string,
MessageReciver>>(result.Body.ToString())));
        }
        catch { }
    }
    foreach (string name in ListGroupChats.Items)
    {
        var result = await
MainWindow.client.GetAsync("GroupMessages/" + name + "/");
        try

```

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				62
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        {
            dialogData.Add(name, result.Body.ToString());
            dialogs.Add(name,
ConvertToHtml(JsonConvert.DeserializeObject<Dictionary<string,
MessageReciver>>(result.Body.ToString()))));
        }
        catch { }
    }
}

```

Дані завантажено, але перевірки чи є нові повідомлення досі немає, тому це треба виправити. Для цього написано функцію checker, яка перевіряє чи є якийсь вибір діалогу, тобто якщо ніякий діалог не обрати перевірка оновлень не працюватиме. Далі проводяться перевірки валідності індексу вибору користувача або групи, якщо вони валідні тоді викачуються дані про діалог, якщо вони змінені, то тоді запускається довгий та затратний процес, дереалізації та перетворення даних на HTML, інакше дані отримані очищаються, і нічого не відбувається, бо змін на базі даних не знайдено, принаймні з активним діалогом.

```

async void checker()
{
    while (ChangeListener)
    {
        bool invalidSelection = false;
        if (!isGroup)                invalidSelection =
(CurentSelection < 0 || CurentSelection >= ListUser.Items.Count);
        }
        else
        {
            invalidSelection = (CurentSelection < 0 ||
CurentSelection >= ListGroupChats.Items.Count);
        }

        if (invalidSelection)
        {
            Console.WriteLine("Invalid selection index.");
            await Task.Delay(1000);
            continue;
        }

        var selectedKey = (!isGroup ?
ListUser.Items[CurentSelection].ToString() :
ListGroupChats.Items[CurentSelection].ToString());

        if (!dialogData.ContainsKey(selectedKey) ||
!dialogs.ContainsKey(selectedKey))
        {

```

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		63

```

        Console.WriteLine($"Key `{selectedKey}` not found in
dialogData or dialogs.");
        await Task.Delay(1000);
        continue;
    }

    var result = !isGroup ? await
MainWindow.client.GetAsync("message/" + CreateKeyForDB(selectedKey,
MainWindow.current_User) + "/" ) :
        await
MainWindow.client.GetAsync(@"GroupMessages/" +
ListGroupChats.SelectedItem.ToString() + "/");
    var newBody = result.Body?.ToString();

    if (newBody == null)
    {
        Console.WriteLine("Received null body from server.");
        await Task.Delay(1000);
        continue;
    }

    if (inScreenDialog == selectedKey &&
dialogData[selectedKey] == newBody)
    {
        await Task.Delay(1000);
        continue;
    }

    dialogData[selectedKey] = newBody;
    dialogs[selectedKey] =
ConvertToHtml(JsonConvert.DeserializeObject<Dictionary<string,
MessageReciver>>(newBody));

    web_view.LoadHtml(dialogs[selectedKey]);
    inScreenDialog = selectedKey;

    await Task.Delay(1000);
}
}

```

Аналогічно до того, як ми перевіряємо чи немає оновлень у повідомленнях, необхідно моніторити чи хтось не почав дзвінок до нас, але на відміну від того як ми моніторимо чи є зміни лише в поточному діалозі, ми маємо моніторити в принципі чи є дзвінок. Це допоможе зробити інший асинхронний метод callChecker, він працюватиме, так, що перевірятиме чи є в активних дзвінках наш нікнейм, і якщо знайде то в методі IsCallingToMe відкриє діалогове вікно дзвінку.

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		64

```

private async void callChecker()
{
    while (true)
    {
        var CallData = await
MainWindow.client.GetAsync("Call/");
        if (CallData != null)
        {
            var callDataString = CallData.Body.ToString();
            if (!string.IsNullOrEmpty(callDataString))
            {
                Dictionary<string, string> CallsData =
JsonConvert.DeserializeObject<Dictionary<string, string>>(callDataString);
                if (CallsData != null)
                {
                    IsCallingToMe(CallsData);
                }
            }
        }
    }
}

private async void IsCallingToMe(Dictionary<string, string> CallsData)
{
    foreach (KeyValuePair<string, string> call in CallsData)
    {
        if (call.Value == MainWindow.current_User)
        {
            if (CallInProgress == false)
            {
                isCalledToMe = true;
                string who = call.Key;
                Caller = who;
                callWindow = new Call_Window(true);
                callWindow.CurrentDispatcher =
Dispatcher.CurrentDispatcher;
                MainMessenger.CurentSelection_call = who;
                CallInProgress = true;
                callWindow.ShowDialog();
            }
        }
    }
}

```

Оскільки дані заплановано відображати у CEF браузері, дані які приходять необхідно конвертувати для цього реалізовано метод ConvertToHtml (Додаток А).

Спочатку створюються списки, які будуть використані для того, щоб сортувати дані, та створити структуру діалогу у форматі HTML. Після чого створюється рядок в який буде записуватись HTML, та в нього вивантажується CSS,

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		65

для створення стилів. Стили записані у окремому файлі. Після чого за допомогою ключових слів порядково перебирається увесь вміст діалогу, в залежності від того, що там написано, дані формуються або в текст, або в файл, або в PDF, для зручного відображення.

Оскільки ціль створити сучасний месенджер, потрібно мати можливість створювати діалог з групою людей, для цього потрібно мати групу відповідних методів `GetUsersForGroupDialog` що дозволить зібрати контакти у список, для того, щоб створити групу, оскільки групу можна створити від трьох людей, тут це також враховується. Метод `CreateRecordGroupDB` у свою чергу допоможе створити запис на базі даних про те, що створена нова група, туди ж запишуться усі учасники та за допомогою класу моделі групи, буде передано до бази даних. Протилежну дію буде виконувати метод `GetMyGroups` він завантажує групи в яких присутній користувач, і аналогічно до `PushName`, записує назви у свій `listbox`.

```
private void GetUsersForGroupDialog(object sender, RoutedEventArgs e)
{
    List<string> ParticipantsNames = new List<string>();
    foreach (var item in listBox.SelectedItems)
    {
        ParticipantsNames.Add(item.ToString());
    }
    ParticipantsNames.Add(MainWindow.current_User);
    groupDialog.Close();
    string GroupName = Utils.InputDialog("Group name", "Please enter group name");
    CreateRecordGroupDB(ParticipantsNames, GroupName);
}
private void CreateRecordGroupDB(List<string> Participants, string GroupName)
{
    string groupParticipants = "";
    foreach (var item in Participants)
    {
        groupParticipants += item + " ";
    }
    groupParticipants.Trim();
    GroupModel gm = new GroupModel
    {
        GroupName = GroupName,
        Users = groupParticipants
    };
};
```

```

        MainWindow.client.Set("/Groups/" + GroupName, gm);
    }

    private List<string> GetMyGroups()
    {
        List<string> userGroups = new List<string>();

        FirebaseResponse response = MainWindow.client.Get("/Groups");
        if (response.Body != "null")
        {
            var data = response.ResultAs<Dictionary<string,
GroupModel>>();

            foreach (var group in data)
            {
                var usersList = group.Value.Users.Split(new[] { ` ` },
StringSplitOptions.RemoveEmptyEntries);
                if (usersList.Contains(MainWindow.current_User))
                {
                    userGroups.Add(group.Key);
                }
            }
        }

        return userGroups;
    }

```

Також в житті людей іноді буває, що потрібно видалити переписку, щоб викинути зайвий контакт, або щоб закінчити спілкування з людиною, або з інших причин. Для цього також необхідно зробити спеціалізований метод. У методі DeleteHistory_Click вимикається слухач поточного діалогу, після чого видаляється історія повідомлень та пара з двох сторін, щоб ці контакти не світилися один у одного в боковій панелі.

```

    private void DeleteHistory_Click(object sender, RoutedEventArgs
e)
    {
        ChangeListener = false;
        var result = MainWindow.client.Delete("message/" +
CreateKeyForDB(ListUser.Items[CurentSelection].ToString()),
MainWindow.current_User) + "/");
        MainWindow.client.Delete(@"UsersPriviousPairs/" +
MainWindow.current_User + "/" +
ListUser.Items[ListUser.SelectedIndex].ToString());
        MainWindow.client.Delete(@"UsersPriviousPairs/" +
ListUser.Items[ListUser.SelectedIndex].ToString() + "/" +
MainWindow.current_User);
    }

```

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				67
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        ListUser.Items.Clear();
        PushNames();
        web_view.LoadHtml("");
    }

```

Велику роль буде виконувати дзвінок, треба забезпечити відповідну функціональність для початку дзвінку або відповіді на вхідний дзвінок, клас обгортка над Sip.cs пропонує функцію AnswerBtn_Click, котра починає аудіо-сповіщення яке змінюється в залежності від дії, та використовує вкладений у Sip.cs, функціонал.

```

private void AnswerBtn_Click(object sender, RoutedEventArgs e)
{
    axMusicPlayer.controls.play();
    MainMessenger.sip.btn_PickUp_Click();
    AnswerBtn.Visibility = Visibility.Collapsed;
}

```

Метод btn_PickUp_Click реалізовує логіку обробки події при натисканні кнопки для прийняття або ініціації телефонного дзвінку. На початку методу CurrentSelection_call, переводиться у числовий формат, за допомогою EncryptToNumbers, що виконує власний механізм кодування номерів, ця дія у свою чергу є додатковим захистом VoIP-ліній. Здійснюється перевірка inComingCall, якщо такий дзвінок, вхідний, то змінну встановлено false, після чого перевірка чи призначений об'єкт дзвінку. Після чого здійснюється прийом дзвінка та приводиться відповідне повідомлення до консольного журналу.

У разі, якщо всі перевірки пройдено, створюється виклик, виводиться повідомлення про початок дзвінка, викликається метод WireUpCallEvent, для підключення усіх обробників подій, та ініціалізується виклик через метод Start.

```

public void btn_PickUp_Click()
{
    string number = EncryptToNumbers(MainMessenger.CurentSelection_call);
    if (inComingCall)
    {
        inComingCall = false;
        if (call == null)
        {
            call = softPhone.CreateCallObject(phoneLine, number);
        }
        call.Answer();
        Console.WriteLine("Call accepted.");
    }
}

```

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		68

```

        return;
    }

    if (call != null)
    {
        return;
    }
    if (phoneLineInformation != RegState.RegistrationSucceeded)
    {
        Console.WriteLine("Registration Failed!");
        return;
    }
    call = softPhone.CreateCallObject(phoneLine, number);
    Console.WriteLine("Call");
    WireUpCallEvents();
    call.Start();
}

```

У свою чергу протилежну дію виконує btn_HangUp_Click який перевірить чи існує дзвінок, якщо він в стані вхідний або гудки то відмінить дзвінок, якщо в стані в розмові, то покладе трубку, очистить дзвінок і якщо дзвінок був вхідний, то очистить інформацію на базі даних.

```

public void btn_HangUp_Click()
{
    if (call != null)
    {
        if (inComingCall && call.CallState == CallState.Ringing)
        {
            call.Reject();
        }
        else
        {
            call.HangUp();
            inComingCall = false;
        }

        call = null;
    }
    if (MainMessenger.isCalledToMe)
    {
        MainWindow.client.Delete(@"Call/" + MainMessenger.Caller + "/");
    }
}

```

Але для коректної роботи, спочатку потрібно все ініціалізувати, цей метод InitializeSoftPhone виконується у конструкторі Sip.cs, на початку здійснюється шифрування Sip-name поточного користувача, встановлюється ліцензійні

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				69
Змн.	Арк.	№ докум.	Підпис	Дата		

налаштування шляхом передачі імені та ключа. Це обов'язковий крок для забезпечення повних функцій SDK.

Наступний етап, це створення екземпляру софтфону за фабричним патерном проєктування, де вказуються IP-адреса та діапазон портів (5000-20000). До створеного об'єкта софтфону підключають обробник подій `softPhone_inComingCall`. Для забезпечення коректної маршрутизації трафіку через NAT використовується конфігурація NAT типу STUN (Simple Traversal of UDP through NAT) із сервером `stun.l.google.com:19302`. Об'єкт `PhoneLineConfiguration` конфігурується з використанням зазначеного SIP-аккаунта та NAT-параметрів, після чого створюється телефонна лінія за допомогою методу `CreatePhoneLine`. Змінна `phoneLine` підписується на подію зміни стану реєстрації (`RegistrationStateChanged`), що дозволяє відстежувати, чи успішно відбулася реєстрація на SIP-сервері. Завершальним етапом є реєстрація телефонної лінії через `RegisterPhoneLine` та виклик допоміжного методу `ConnectMedia()`, який, відповідає за ініціалізацію аудіо-потoku мікрофону зокрема.

```
public void InitializeSoftPhone()
{
    try
    {
        string nameSip = EncryptToNumbers(MainWindow.current_NameSip);
        string PasswordSip = MainWindow.current_PasswordSip;
        var userName = "";
        var key = "";
        Ozeki.Common.LicenseManager.Instance.SetLicense(userName, key);
        string getMyIpAddress = "";
        WebRequest request =
        WebRequest.Create("http://checkip.dyndns.org/");
        using (WebResponse response = request.GetResponse())
        {
            using (StreamReader stream = new
            StreamReader(response.GetResponseStream()))
            {
                getMyIpAddress = stream.ReadToEnd();
            }
            int first = getMyIpAddress.IndexOf("Address: ") + 9;
            int last = getMyIpAddress.IndexOf("</body>");
            getMyIpAddress = getMyIpAddress.Substring(first, last - first);
        }
    }
}
```

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				70
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        softPhone =
SoftPhoneFactory.CreateSoftPhone(SoftPhoneFactory.GetLocalIP(), 5000,
20000);
        Console.WriteLine("Softphone created!");
        softPhone.IncomingCall += softPhone_inComingCall;
        NatConfiguration natconfig = new
NatConfiguration(NatTraversalMethod.STUN, "stun.l.google.com:19302",
false);
        sa = new SIPAccount(true, nameSip, nameSip, nameSip, PasswordSip,
"185.249.161.189", 60013);
        Console.WriteLine("SIP account created!");
        PhoneLineConfiguration config = new PhoneLineConfiguration(sa);
        config.NatConfig = natconfig;
        phoneLine = softPhone.CreatePhoneLine(config);
        phoneLine.RegistrationStateChanged +=
phoneLine_PhoneLineInformation;
        Console.WriteLine("Phoneline created.");
        softPhone.RegisterPhoneLine(phoneLine);
        ConnectMedia();
    }
    catch (Exception ex)
    {
        Console.WriteLine("Local IP error!");
    }
}

```

3.2. Тестування клієнтської сторони

Тестування застосунку, це важлива частина, яка дозволяє перевірити стабільність роботи системи. Тестування має багато варіацій, функціональне тестування, тестування продуктивності, тестування безпеки, тестування зручності використання та тестування приймання. Проте згідно до вимог та тематики кваліфікаційної роботи з цих варіантів потрібно тестувати лише функціональне тестування та тестування продуктивності, ці два типи можна об'єднати в один тип тест кейсів та проводити в мануальному стилі перевіряючи усі можливі варіанти, або напів-автоматизованому варіанті з використанням Unit-тестування.

Unit-тестування використовують у системах з великою кількістю варіантів даних в одному і тому місці використання, це не про месенджер, тут маємо тільки два варіанта даних, текстовий та файловий, тому тестування буде проводитись у мануальному стилі.

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		71

Оскільки проєкт написаний на .Net потрібно встановити його оточення, тому це попередня умова до кожного з тест-кейсів. Також для початку тестування програма має бути запущеною.

Таблиця 3.1.

Тест-кейс “відправка повідомлення”

Reporter:	Maryna Graf	Report date:	15.04.2025	Status:
Assined to:	Nikita Yukhymchuk	Test date:	20.04.2025	Fixed.
TestCase Name:		T0001: Send Message		
Prerequisites:		Install .Net 3.5, 4.5, 5. Open app		
#	Steps to reproduce	Expected Result		Anomaly
1	Find contact on left side or search for it	Displays names		✓
2	Click on contact name	Displays chat history or notification about awaiting first message		✓
3	Press send button	Nothing happens		✗ placeholder sended
4	Click textbox	Placeholder dissapears		✓
5	Type text	Letters appear with no delay		✓
6	Click send button	Text cleared. Send request via HTTPS		✓
7	Check for view update	Updating CEF View with text that was typed		✓

Під час тестування відправки повідомлення, користувач повинен мати змогу знайти контакт у списку зліва або за допомогою пошуку. Після натискання на контакт, користувача має побачити діалог на вбудованому браузері. При спробі відправити повідомлення яке ще не було написано, було виявлено проблему. Проблема усунуто. Наступний крок, це відправка повідомлення. Повідомлення відправляються та одразу без затримок оновлюється CEF-браузер. Тест кейс мав вияви помилки, які було виправлено шляхом перевірки чи перед цим було сфокусовано поле для вводу. (Таблиця 3.1.)

Таблиця 3.2.

Тест-кейс “Прикріплення та відправка файлу”

Reporter:	Maryna Graf	Report date:	16.04.2025	Status:
Assined to:	Nikita Yukhymchuk	Test date:	20.04.2025	Fixed.
TestCase Name:		T0002: Attach and Send file or image		
Prerequisites:		Install .Net 3.5, 4.5, 5. Open app		
#	Steps to reproduce	Expected Result	Anomaly detected	
1	Find contact on left side or search for it	Displays names	✓	
2	Click on contact name	Displays chat history or notification about awaiting first message	✓	
3	Click on pin icon	Opened system window “Select file”	✓	
4	Select file	Selected file should be send via https.put	✓	

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				73
Змн.	Арк.	№ докум.	Підпис	Дата		

Reporter:	Maryna Graf	Report date:	16.04.2025	Status:
Assined to:	Nikita Yukhymchuk	Test date:	20.04.2025	Fixed.
TestCase Name:		T0002: Attach and Send file or image		
Prerequisites:		Install .Net 3.5, 4.5, 5. Open app		
5	Update view	CEF view should display file as message type depends on file type		

Під час тестування прикріплення та відправки файлу, користувач має обрати контакт та натиснути на іконку “скріпка”, яка символізує прикріплення файлу, після чого має відкритись вбудоване в операційну систему, після пошуку та вибору файлу, файл повинен відправитись та в залежності від типу файлу по різному відобразитись у CEF-браузері. Тест пройшов перевірку, проблем не виявлено. (Таблиця 3.2)

Таблиця 3.3.

Тест-кейс “Підтримка Емоїї”

Reporter:	Maryna Graff	Report date:	16.04.2025	Status:
Assined to:	Nikita Yukhymchuk	Test date:	20.04.2025	Fixed.
TestCase Name:		T0003: Emoji support		
Prerequisites:		Install .Net 3.5, 4.5, 5. Open app		
#	Steps to reproduce	Expected Result		Anomaly
1	Find contact on left side or search	Displays names		

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				74
Змн.	Арк.	№ докум.	Підпис	Дата		

Reporter:	Maryna Graff	Report date:	16.04.2025	Status:
Assined to:	Nikita Yukhymchuk	Test date:	20.04.2025	Fixed.
TestCase Name:		T0003: Emoji support		
Prerequisites:		Install .Net 3.5, 4.5, 5. Open app		
	for it			
2	Click on contact name	Displays chat history or notification about awaiting first message		
3.	Click on the emoji icon	Should be displayed list of emoji		
4	Press win+. and select emoji from windows menu	Opened a list of emoji in window embedded menu		
5	Press send button	Message send and CEF view updated		

Під час тестування проекту, потрібно обов'язково враховувати та перевіряти підтримку unicode символів, та можливість за допомогою них виражати емоції. З цим добре справляються Емоїї, оскільки вони unicode символи та є невід'ємною частиною спілкування та будь-якого месенджера, ця підтримка має бути гарантована. Для тестування цих функцій, користувач має відкрити чат з контактом за вибором, після чого знайти іконку *смайлику* (сленгова назва іконки Емоїї) та обрати будь-який Емоїї з запропонованих, проте є інший шлях, користувачу може бути недостатньо запропонованих Емоїї, в такому випадку користувач повинен натиснути комбінацію клавіш "Win+.", та обрати з великого спектру емоцій підходящий Емоїї. Вибравши Емоїї будь-яким способом, він має відобразитись у текстовому полі. При натисканні на клавішу відправки, воно має відправити його

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		75

як звичайний текст та оновити CEF-браузер. Тест пройшов перевірку, проблем не виявлено. (таблиця 3.3)

Таблиця 3.4.

Тест-кейс “видалення історії повідомлень”

Reporter:	Maryna Graf	Report date:	16.04.2025	Status:
Assined to:	Nikita Yukhymchuk	Test date:	20.04.2025	Fixed.
TestCase Name:		T0004: Delete message history		
Prerequisites:		Install .Net 3.5, 4.5, 5. Open app		
#	Steps to reproduce	Expected Result		Anomaly
1	Find contact on left side or search for it	Displays names		✓
2	Click on contact name	Displays chat history or notification about awaiting first message		✓
3	Click on trash icon	Chat closed, history deleted, user pair deleted		✓

При тестуванні проєкту також важливо переконатись, що функція повного видалення історії повідомлень, працює коректно. Видалення історії є критично важливим для забезпечення приватності, безпеки користувача. Для цього користувач має знайти контакт який він хоче видалити, натиснувши на іконку смітника, має зникнути переписка, викинути користувача з поточного чату, та видалити контакт з бокової панелі, оскільки їхня історія закінчилась. (Таблиця 3.4)

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		76

Таблиця 3.5.

Тест-кейс “Дзвінок”

Reporter:	Maryna Graf	Report date:	16.04.2025	Status:
Assined to:	Nikita Yukhymchuk	Test date:	20.04.2025	Fixed.
TestCase Name:		T0004: Call		
Prerequisites:		Install .Net 3.5, 4.5, 5. Open app		
#	Steps to reproduce	Expected Result		Anomaly
1	Find contact on left side or search for it	Displays names		✓
2	Click on contact name	Displays chat history or notification about awaiting first message		✓
3.	Click on call button	Chat closed, history deleted, user pair deleted		✓
4.	Click on call button (call window)	Sound of beeping starts until companion answered		✓
5.	Sound of companion	Clear voice translated		✓
6.	Click mute button several time with interval	on each odd even times, mic is muted, on each even times, mic is unmuted		✓
7.	Click hangup	Call ended db record cleared		✓

Reporter:	Maryna Graf	Report date:	16.04.2025	Status:
Assined to:	Nikita Yukhymchuk	Test date:	20.04.2025	Fixed.
TestCase Name:		T0004: Call		
Prerequisites:		Install .Net 3.5, 4.5, 5. Open app		
	button			

Останній тест який має пройти, це тест дзвінка. Під час тестування функції дзвінків у проєкті необхідно перевірити повний цикл ініціації, проведення та завершення дзвінка, включаючи взаємодію з базовими елементами керування. Користувач має знайти контакт у списку або скористатися пошуком, після чого обрати його для відкриття чату. Натиснувши кнопку дзвінка, запускається вікно дзвінка, де розпочинається гудок, який триває до моменту відповіді співрозмовника. Після з'єднання має бути чутний голос співрозмовника з чіткою передачею звуку. Під час активного дзвінка користувач може натискати кнопку вимкнення мікрофона. При кожному непарному натисканні мікрофон має вимикатися, при кожному парному вмикатися. Після завершення дзвінка натисканням кнопки завершення дзвінок має очищатись та зв'язок пропадати, а дані пов'язані з дзвінком на базі даних зникати. Тест пройшов успішно. (Таблиця 3.5)

3.3. Тестування на фішингове вторгнення

Під час розробки, було передбачено велику кількість помилок, тому під час тестування було виявлено лише таку незначну кількість помилок, котрі впливали на роботу з застосунком, усі з вищеназваних помилок були успішно виправлені, застосунок є повністю функціональним.

Фішингові дзвінки, з зовнішнього SIP-клієнта це одна з найбільших загроз будь-якому застосунку який працює на базі Asterisk.

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		78

Щоб виконати захист від такого типу втручання треба реалізовувати свою логіку, оскільки сам Asterisk не поставляє жодного інструменту виявлення даних клієнту.

Сам алгоритм такого проникнення є простим, виявлення IP-адреси та порту серверу, після чого дізнатись логін та пароль користувача, що можна зробити Brute-force (надалі брутфорс), або іншими способами. Знаючи ці дані, вже можна підключити зовнішній SIP.

Щоб це зробити для початку необхідно створити дзвінок у базі даних, далі встигнути за 10 секунд, натиснути на виклик у зовнішній програмі та підняти слухавку, тоді можливий виклик у застосунку. Якщо не мати доступу до бази даних, такий дзвінок наразі неможливий, що захищає користувачів від настирливих спам ботів. (Рисунок 3.18)

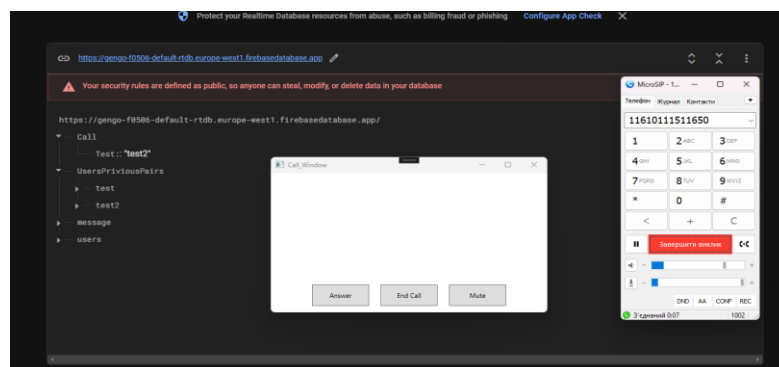


Рисунок 3.18 - Дзвінок зі стороннього застосунку.

Такий захист необхідно робити оскільки існують загрози та типи атак на VoIP, як Vishing це тип атаки при якій атакуюча сторона видає себе за працівника, друга, службу підтримки або банкіра, з метою змусити жертву повідомити конфіденційні дані, такі як паролі, пін-коди, дані банківських карт та інше. Caller ID Spoofing це тип атаки при якій атакуючий змінює ідентифікатор номеру, щоб при дзвінку у жертви відображався інший номер, наприклад номер банку або поліції, що підвищує довіру жертви до дзвінків. Така атака часто використовується у парі з Vishing-ом. Intrusion Attacks це атака, при якій атакуюча сторона намагається проникнути на VoIP мережу або сервер, використовуючи різні методи, серед яких і BruteForce, з метою отримати нелегальний доступ до прослуховування дзвінків, здійснення несанкціонованих дзвінків або повного контролю над VoIP

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		79

сервером. VoIP Phishing Attack атака яка поєднує фішинг та VoIP, де користувачу надсилають повідомлення із закликом зателефонувати на спеціальний номер, а коли жертва телефонує її автоматично вітає голос або оператор, який під виглядом реальної організації збирає дані. Інколи VoIP Phishing використовує автоматичні системи Interactive Voice Response, які симулюють банківські служби.

Усі ці загрози нейтралізуються простою перевіркою наявності дзвінка з клієнтського застосунка на номер іншого клієнту, через базу даних. Оскільки Firebase немає ніякого прямого контакту з Asterisk-ом, такий формат захисту нейтралізує кожен з цих методів атак на VoIP мережу.

3.4. Інструкція з експлуатації

Після того як було отримано готову до роботи програму, потрібно створити до неї інструкцію по користуванню. Як засвідчує практика, наявність інструкції може значно прискорити початок роботи з застосунком для користувача та допомогти уникнути небажаних помилок при використанні.

Після запуску програми буде виведене вікно входу (Рисунок 3.19), на цьому вікні можна увійти у свій акаунт або створити новий натиснувши клавішу реєстрації

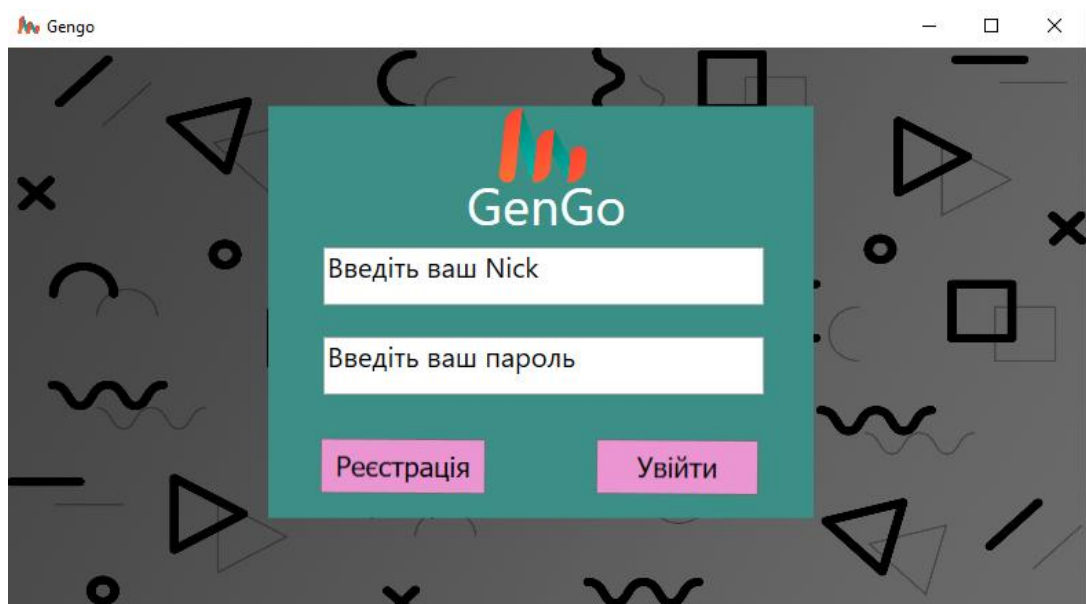


Рисунок 3.19 - Вікно програми після запуску

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		80

Для створення нового акаунту потрібно натиснути клавiшу «реєстрація» після чого з'явиться вікно реєстрації (Рисунок 3.20) у якому потрібно заповнити наступні поля:

- Nick;
- Email;
- Password.

Після введення цієї інформації потрібно натиснути клавiшу «зареєструватися» після чого вас відправить на вікно входу.

Рисунок 3.20 - Вікно реєстрації

Після входу/реєстрації вмикається основне вікно для листування та вибору контакту для дзвінків. (Рисунок 3.21)

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		81

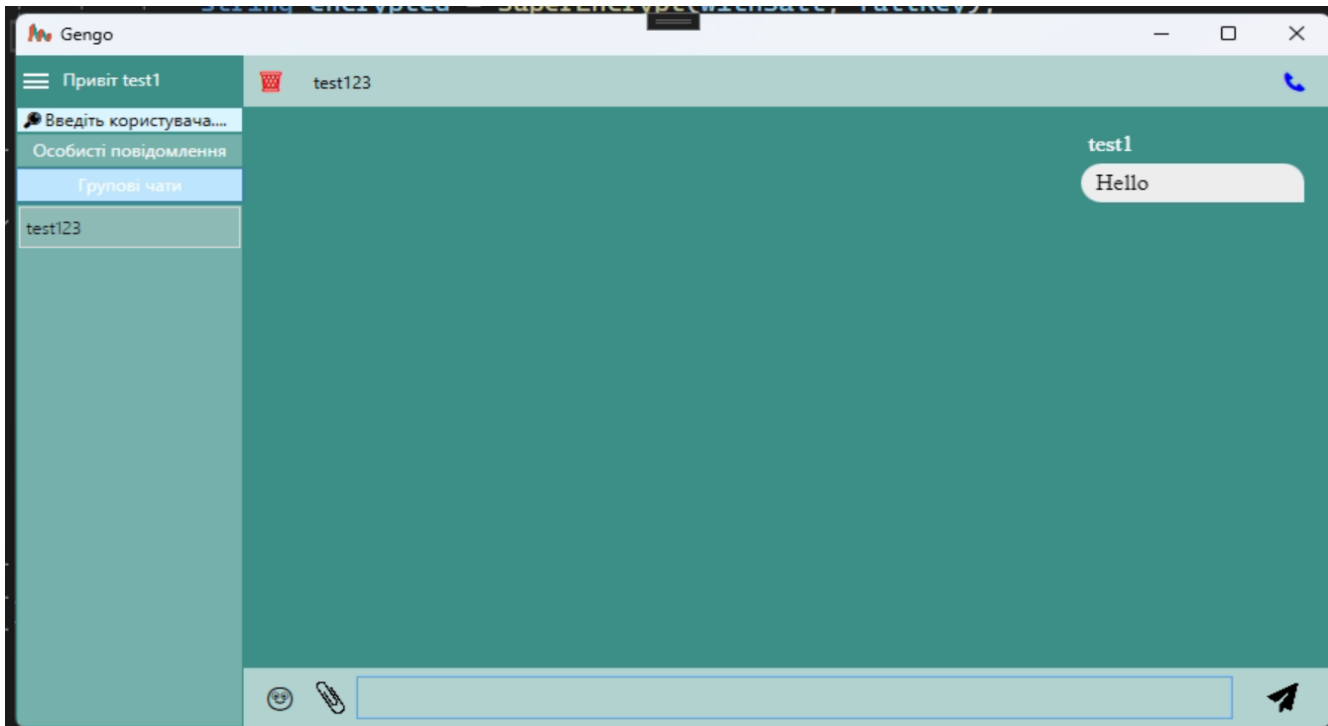


Рисунок 3.21 - Вікно нового акаунту

У вікні зверху зліва потрібно ввести ім'я людини з якою ви хочете зв'язатися, та натиснути клавішу "Enter" для створення пари. (Рисунок 3.22)

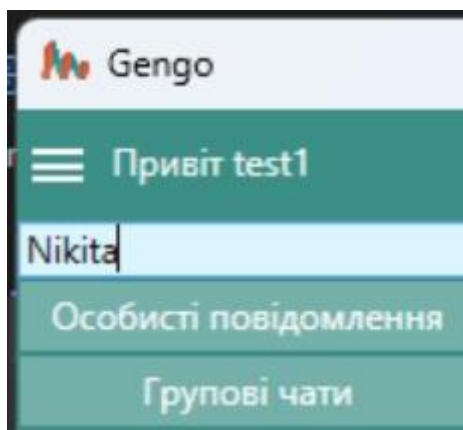


Рисунок 3.22 - Додавання людей

Після натискання клавіші з іменем людини, яке ви ввели, з'явиться у списку, що знаходиться під полем для вводу. (Рисунок 3.23)

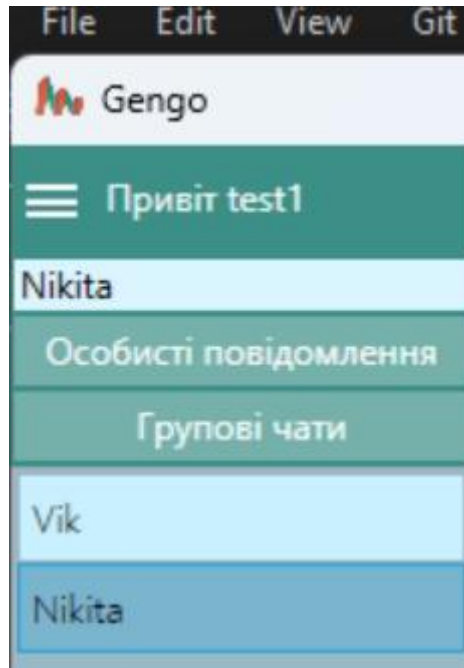


Рисунок 3.23 - Приклад як виглядає список друзів

Після натискання на ім'я ви можете ввести повідомлення знизу та відправити його. (Рисунок 3.24)

У верхньому правому куті ви можете натиснути на кнопку «телефон», для того щоб зв'язатися за допомогою голосового дзвінку. (Рисунок 3.25)

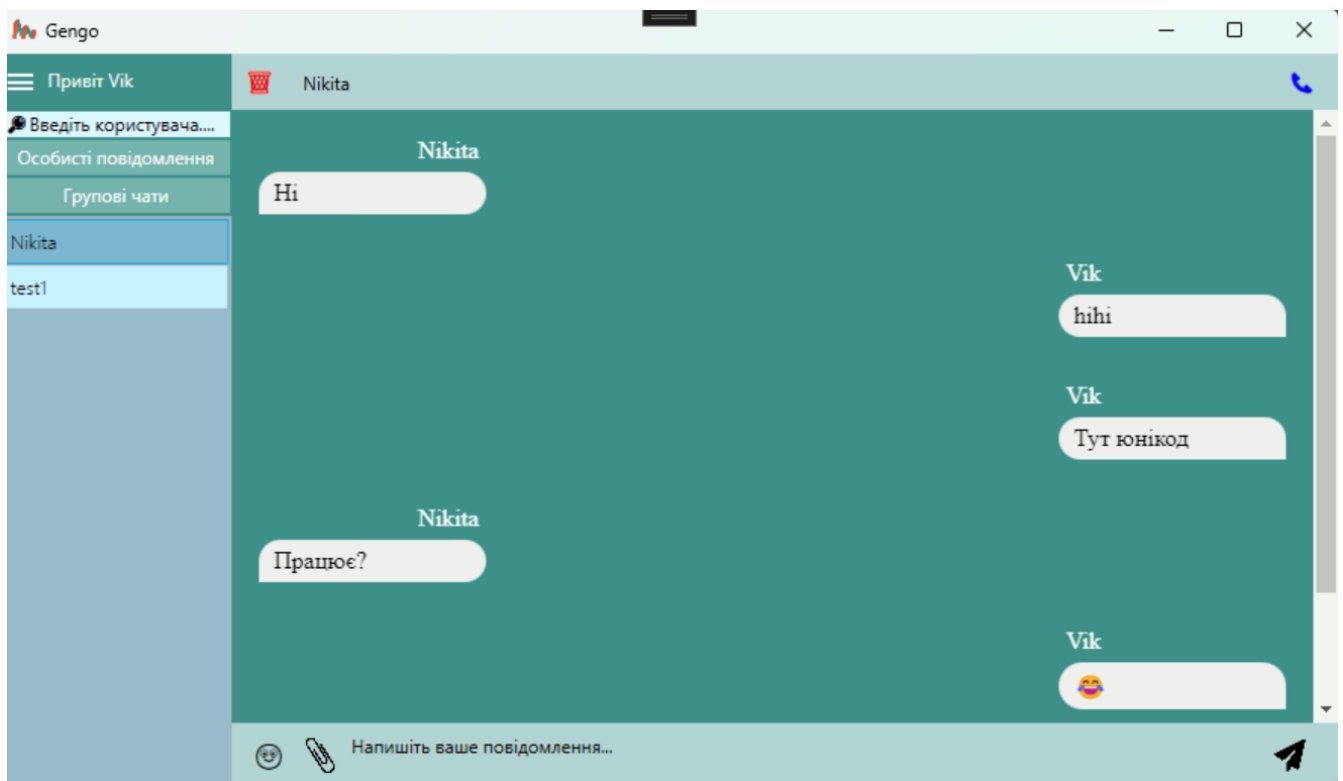


Рис 3.24 - Вигляд чату

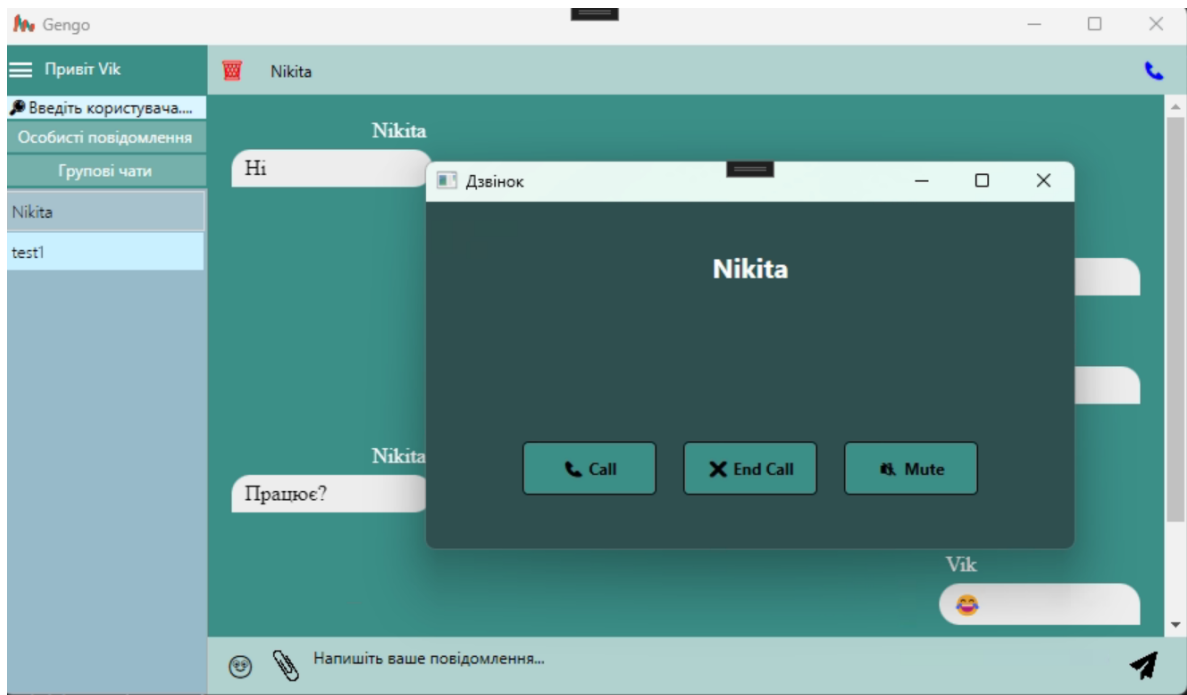


Рисунок 3.25 - Вікно дзвінку користувачу «Nikita»

Користувач тест, має запис на базі даних, у якому зберігається логін та пароль SIP користувача, тому з ним можливо зв'язатися, ці поля створюються при реєстрації, якщо він їх немає йому подзвонити не можна. (Рисунок 3.26)

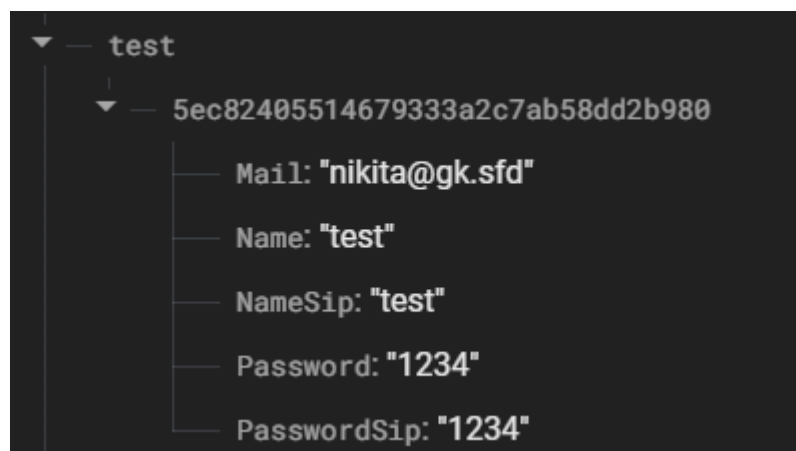


Рисунок 3.26 - Запис на базі даних

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				84
Змн.	Арк.	№ докум.	Підпис	Дата		

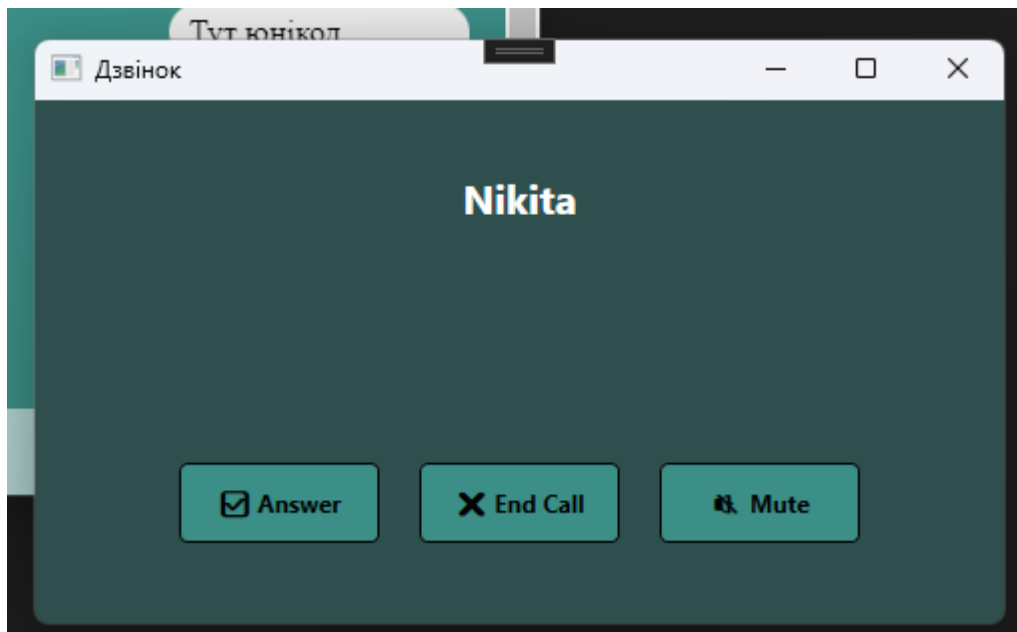


Рисунок 3.27 - Вхідний дзвінок

Для створення групового чату, необхідно перейти до налаштувань, та в відкритому віконці та натиснути на кнопку створення нового групового чату (“Create new group chat”) та у новому вікні вибору контактів, вибрати хоча б два користувача для створення групи, після чого натиснути на клавішу підтвердження (Рисунки, 3.28, 3.29, 3.30)

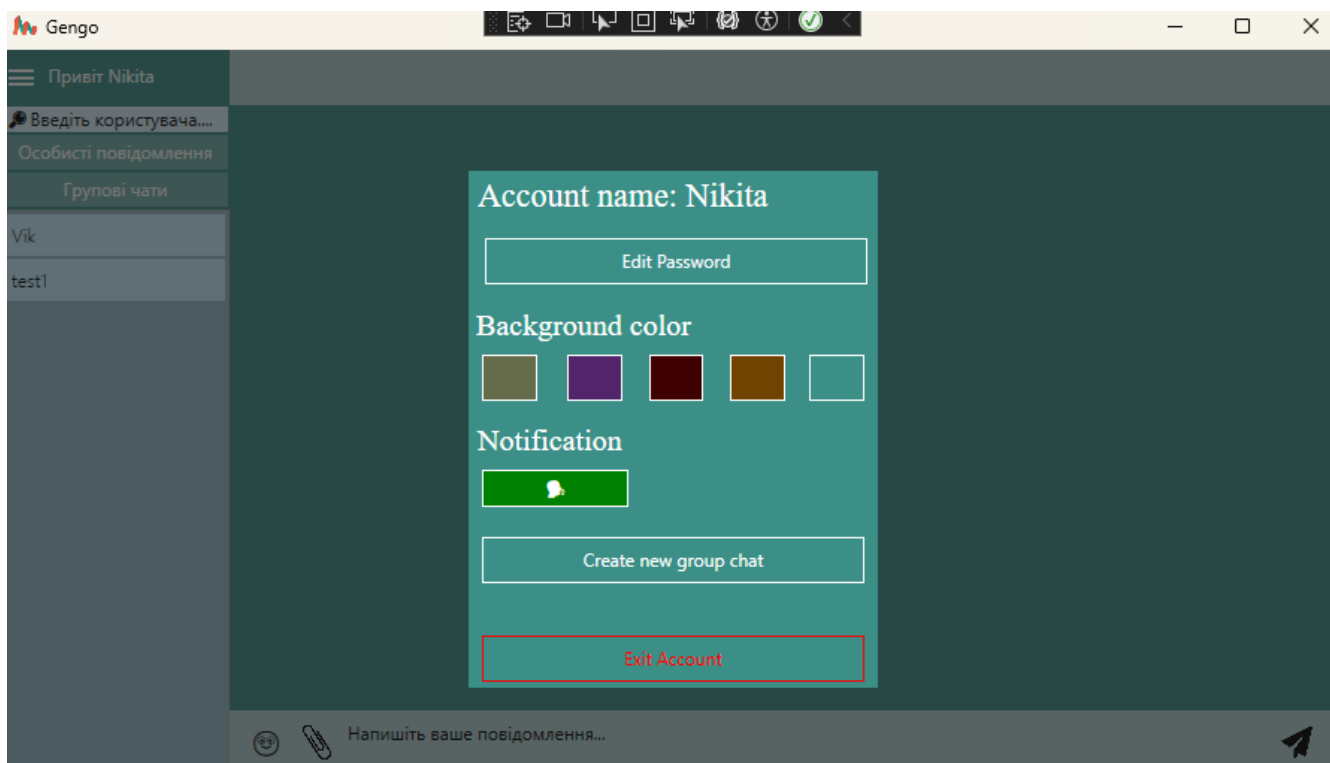


Рисунок 3.28 - Вікно налаштувань

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				85
Змн.	Арк.	№ докум.	Підпис	Дата		

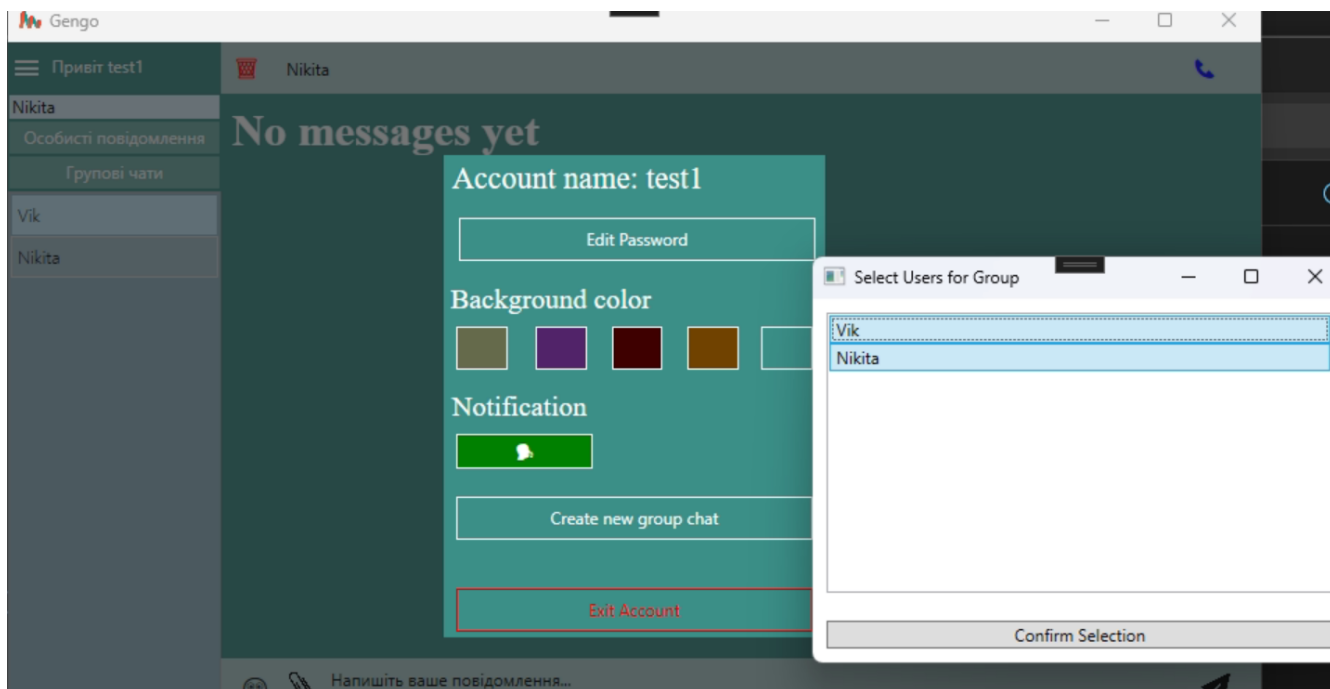


Рисунок 3.29 - Вікно вибору контактів

Далі потрібно перейти у список групових контактів та вибрати контакт. Подальше використання групових чатів співпадає зі звичайними чатами.

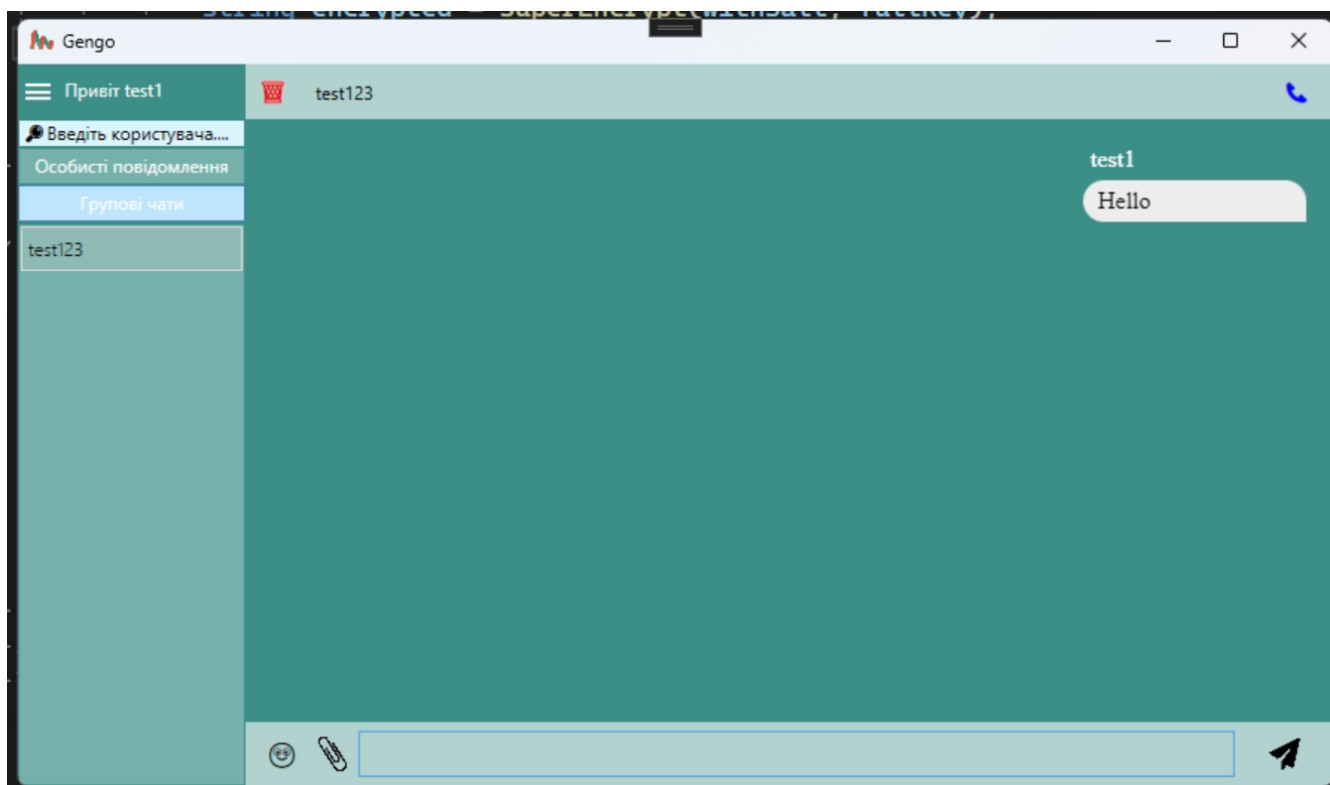


Рисунок 3.30 - Вікно повідомлень у групові чати

Також у вікні налаштувань можна змінити кольорову гаму застосунку.

До будь яких чатів можна прикріпити файл, для цього необхідно натиснути на скріпку та обрати файл у вікні вибору файлів, вбудовану в операційну систему. Вікно може варіюватися своїм виглядом в залежності від операційної системи.

Висновки до 3 розділу

У третьому розділі описано створення, встановлення та налаштування серверної частини, створено скрипт автоматизації реєстрації нових користувачів у системі, описано принцип встановлення Asterisk на Ubuntu та принцип роботи автоматизованого скрипту на мові Python.

Наведено порядок створення клієнтського застосунку, описано критичні методи класів, розписано принцип роботи методів. Описано підключення Firebase сервісів та Ozeki VoIP SDK до проєкту, налаштування їх роботи в проєкті.

Проведено мануальне тестування, та наведено тест кейси які дозволяють перевірити типові помилки, які можливі при необережному проєктуванні та створенні програмного коду до клієнтської частини. В результаті проведеного тестування було виявлено помилку відправлення допоміжного тексту яку було усунуто зміненням програмного коду клієнтського застосунку.

Проведено перевірку на стійкість до атаки на VoIP системи типу “Vishing”, “Caller ID Spoofing”, “Intrusion attacks”, “VoIP Phishing”, та доведено стійкість до них через використання додаткового захисту дзвінків через базу даних.

Створено мануал користувача системи.

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		87

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було створено проєкт системи обміну повідомленнями та VoIP-зв'язку на основі Firebase та Asterisk

У першому розділі кваліфікаційної роботи було розглянуто підготовчі етапи: аналіз ринку існуючих застосунків обміну повідомленнями та VoIP-клієнтів, інструментів та протоколів передачі інформації, проведено огляд літератури. Було сформульовано задачі, цілі роботи серед яких створення клієнтської частини, налаштування бази даних та розробка серверної складової системи. Описано переваги та недоліки конкурентних застосунків, що дозволило визначити напрям вдосконалення майбутнього застосунку. Обґрунтовано вибір інструментів, мову програмування C#, Asterisk для SIP-зв'язку та інших необхідних компонентів. Обґрунтовано вибір протоколів передачі даних.

У другому розділі досліджено обрані технології, обґрунтовано їх використання та описано принципи роботи середовища й допоміжних інструментів. Проведено теоретичне проєктування системи за допомогою UML-діаграм, описано варіанти використання, основні сценарії критичних подій та побудовано компонентну і пакетну діаграми клієнтського застосунку. Описано тип шифрування повідомлень, надано алгоритм, математичні формули та проведено аналіз криптостійкості.

У третьому розділі проведено мануальне тестування клієнтської частини, наведено тест кейси для перевірки типових помилок. Перевірено стійкість до атак типу “Vishing”, “Caller ID Spoofing”, “Intrusion attacks”, “VoIP Phishing”. Стійкість до них підтверджено завдяки використанню додаткового захисту через базу даних. Створено просту інструкцію для користувачів без технічної підготовки. Розгорнуто й налаштовано серверну частину, створено скрипт на Python для автоматизації реєстрації користувачів, описано встановлення Asterisk. Розроблено клієнтський застосунок, описано критичні методи класів та принципи їх роботи. Підключено й налаштовано сервіси Firebase та Ozeki VoIP SDK.

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		88

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. The C# Programming Language, Third Edition / A. Hejlsberg, M. Torgersen, S. Wiltamuth, P. Golde. — Boston : Addison-Wesley, 2008. — 784 с. — ISBN 9780321592255.
2. C# 10 and .NET 6 – Modern Cross-Platform Development: Build apps, websites, and services with ASP.NET Core 6, Blazor, and EF Core 6 using Visual Studio 2022 and Visual Studio Code, 6th Edition / M. J. Price. — Birmingham : Packt Publishing, 2021. — 752 с. — ISBN 9781801077361.
3. C# in Depth, Fourth Edition / J. Skeet. — Shelter Island : Manning Publications, 2019. — 528 с. — ISBN 9781617294532.
4. Programming C# 8.0: Build Cloud, Web, and Desktop Applications / I. Griffiths. — Sebastopol, USA : O'Reilly Media, 2020. — 797 с. — ISBN 9781492056812.
5. Learn C# Programming: A guide to building a solid foundation in C# language for writing efficient programs / M. Bancila, R. Rialdi, A. Sharma, D. Esposito. — Birmingham : Packt Publishing, 2020. — 636 с. — ISBN 9781789805864.
6. Professional WPF and C# Programming: Practical Software Development Using WPF and C# / A. Khang. — Independently Published, 2019. — 387 с. — ISBN 9781097122653.
7. NuGet CLI reference / @JonDouglas та ін. Microsoft Learn. 17.04.2023. URL: <https://learn.microsoft.com/en-us/nuget/reference/nuget-exe-cli-reference>.
8. NuGet documentation / @JonDouglas та ін. Microsoft Learn. 17.04.2023. URL: <https://learn.microsoft.com/en-us/nuget/>.
9. Manage NuGet packages with the NuGet CLI / @JonDouglas та ін. Microsoft Learn. 03.03.2025. URL: <https://learn.microsoft.com/en-us/nuget/consume-packages/install-use-packages-nuget-cli>.
10. What is NuGet and what does it do? / @JonDouglas та ін. Microsoft Learn. 17.04.2023. URL: <https://learn.microsoft.com/en-us/nuget/what-is-nuget>.
11. FireSharp: .NET client for Firebase API / BugTheSystem. — GitHub, 2023. URL: <https://github.com/bugthesystem/FireSharp>

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		89

12. FirebaseDatabase.net: Complex C# library for Firebase Realtime Database / NuGet Team. — NuGet Gallery, 2023. URL: <https://www.nuget.org/packages/FirebaseDatabase.net/>
13. FirebaseStorage-dotnet: C# library for Firebase Storage / Step-Up Labs. — GitHub, 2023. URL: <https://github.com/step-up-labs/firebase-storage-dotnet>
14. Firebase Authentication .NET SDK / Google. — Firebase Documentation, 2023. URL: <https://firebase.google.com/docs/auth/admin/verify-id-tokens?hl=uk>
15. Firebase Realtime Database REST API / Google. — Firebase Documentation, 2023. URL: <https://firebase.google.com/docs/database/rest/start>
16. Firebase Storage REST API / Google. — Firebase Documentation, 2023. URL: <https://firebase.google.com/docs/storage/web/start>
17. Using Firebase in .NET / Maksym Labutin. — Stack Overflow, 2019. URL: <https://stackoverflow.com/questions/41676714/using-firebase-in-net>
18. Integrating Firebase with WPF Applications / Timothy Bolorunduro. — Bolorunduro WB, 2023. URL: <https://bolorundurowb.dev/posts/31/using-the-firebase-realtime-database-with-net>
19. Making a Simple To-Do List with Firebase using FireSharp / Kode Snippets. — Kode Snippets, 2015. URL: <https://kodesnippets.wordpress.com/2015/09/05/making-a-simple-to-do-list-with-firebase-using-firesharp/>
20. Firebase using FireSharp with C# & WPF Session1: Connecting & Insert Data / YouTube. — YouTube, 2019. URL: <https://www.youtube.com/watch?v=pZ6YkFvL4Uo>
21. CefSharp.Wpf / @CefSharp Team. NuGet Gallery. 26.04.2025. URL: <https://www.nuget.org/packages/CefSharp.Wpf>
22. CefSharp: Open Source web browser for WinForms and WPF Apps / @CefSharp. CefSharp Project. 26.04.2025. URL: <https://cefsharp.github.io/>
23. The Official Ubuntu Book / Benjamin Mako Hill, Jono Bacon, Corey Burger, Jonathan Jesse, Ivan Krstić. — Prentice Hall PTR, 2007. 412 с. ISBN 9780132435942

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				90
Змн.	Арк.	№ докум.	Підпис	Дата		

24. The Official Ubuntu Book, Revised Edition / Matthew Helmke, Amber Graner, Benjamin Mako Hill, Corey Burger. — Addison-Wesley Professional, 2011. 432 c. ISBN 9780132748506
25. Ubuntu Linux Unleashed, 2021 Edition / Matthew Helmke et al. — Addison-Wesley Professional, 2020. 752 c. ISBN 9780136778851
26. Asterisk: The Definitive Guide, Third Edition / Leif Madsen, Jim Van Meggelen, Russell Bryant. — O'Reilly Media, 2011. 295 c. ISBN 9780596517342
27. Asterisk Cookbook: Solutions to Everyday Telephony Problems / Leif Madsen, Russell Bryant. — O'Reilly Media, 2011. 66 c. ISBN 9781449303822
28. How to configure static IP on the computer / TP-Link Support. 23.10.2023. URL: <https://www.tp-link.com/us/support/faq/14/>
29. How to Set up Static Routing on a TP-Link Router / TP-Link Support. 11.09.2023. URL: <https://www.tp-link.com/us/support/faq/3601/>
30. Setting up static ip with wireless N USB adapter (TL-WN723N) ubuntu 16.04.2 / @kevin janada. Ask Ubuntu. 01.06.2017. URL: <https://askubuntu.com/questions/920963/setting-up-static-ip-with-wireless-n-usb-adapter-tl-wn723n-ubuntu-16-04-2>
31. Asterisk For Dummies / S. P. Olejniczak, B. Kirby. — Hoboken, NJ : For Dummies, 2007. — 368 c. — ISBN 9780470098547
32. Asterisk: The Definitive Guide: Open Source Telephony for the Enterprise, 5th Edition / J. Van Meggelen, R. Bryant, L. Madsen. — Sebastopol : O'Reilly Media, 2019. — 412 c. — ISBN 9781492031604
33. Asterisk 1.6: Build feature-rich telephony systems with Asterisk / D. Merel, B. Dempster. — Birmingham : Packt Publishing, 2009. — 240 c. — ISBN 9781847198624
34. Think Python: How to Think Like a Computer Scientist / A. Downey. — Needham : Green Tea Press, 2012. — 304 c. — ISBN 9780994392903
35. Python Programming: An Introduction to Computer Science / J. Zelle. — Franklin, Beedle & Associates Inc., 2004. — 400 c. — ISBN 9781887902993

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		91

36. Programming Python: Powerful Object-Oriented Programming (4th Edition) / M. Lutz. — Sebastopol : O'Reilly Media, 2010. — 1600 c. — ISBN 9781449397529
37. Serious Cryptography: A Practical Introduction to Modern Encryption / J.-P. Aumasson. — San Francisco : No Starch Press, 2017. — 256 c. — ISBN 9781593278267
38. Applied Cryptography: Protocols, Algorithms, and Source Code in C / B. Schneier. — New York : Wiley, 1996. — 784 c. — ISBN 9781119096726
39. Modern Cryptography: Theory and Practice / W. Stallings. — Boston : Pearson, 2017. — 672 c. — ISBN 9780134444284
40. VoIP Security: Threats, Vulnerabilities, and Countermeasures / S. K. Sood, S. J. Murdoch. — Boca Raton : CRC Press, 2010. — 320 c. — ISBN 9781420088633
41. VoIP Security: Protecting Your VoIP Network / M. J. Baca. — Indianapolis : Cisco Press, 2005. — 304 c. — ISBN 9781587051952
42. VoIP Security Threats & Best Practices / @Tech.co. — Tech.co, 2025. URL: <https://tech.co/business-phone-systems/voip-security>
43. Cryptanalysis: A Study of Ciphers and Their Solution / H. F. Gaines. — New York : Dover Publications, 1956. — 160 c. — ISBN 9780486200974
44. Modern Cryptanalysis: Techniques for Advanced Code Breaking / W. Swenson. — Hoboken : Wiley-Interscience, 2002. — 352 c. — ISBN 9780471060145
45. Algebraic Cryptanalysis / A. Canteaut, F. Chabaud. — Berlin : Springer, 2009. — 200 c. — ISBN 9783540855974
46. Corezoid. Telegram vs Viber vs Facebook Messenger: The comparison of the messengers' capabilities. *Corezoid*. 06.06.2017. URL: <https://corezoid.com/blog/the-comparison-of-the-messengers-capabilities/>.
47. Liem D. 3CX Vs FreePBX: Which VoIP Solution is Right for You?. *Ringblaze*. 16.01.2024. URL: <https://www.ringblaze.com/blog/3cx-vs-freepbx/>.

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		92

ДОДАТКИ

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		93

```

private string ConvertToHtml(Dictionary<string, MessageReciver> Data)
{
    List<string> sender = new List<string>();
    List<string> senderList = new List<string>();
    List<string> message = new List<string>();
    List<int> index = new List<int>();
    string html = "";

    using (StreamReader reader = new StreamReader("yik.css"))
    {
        html = reader.ReadToEnd();

        html = $"<html lang=\"UA\" ><head></head><meta charset=\"UTF-8\"><body>
<style>{html}</style>";
    }

    html += "<div id = \"content\" > ";

    if (Data != null)
    {
        foreach (var item in Data)
        {
            item.Value.Decode_Message();
            sender.Add($"{item.Value.From}");
            message.Add($"{item.Value.Message}");
            index.Add(item.Value.Number);
            senderList.Add(item.Value.From);
        }

        int max = maxNumer(index);

        if (PreviousSelection < max)

```

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				94
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        {
            if (PreviousSelection != max && MainWindow.current_User !=
senderList[index.IndexOf(max)])
            {

                PreviousSelection = index.Count;

                if (NotificationCounter == true)
                {
                    notifier.ShowInformation(message[index.IndexOf(max)]);

                    SystemSounds.Beep.Play();

                }
            }
            else
            {
                PreviousSelection = index.Count;

            }
        }
        Sort(message, senderList, index);
        for (int i = 0; i < message.Count; i++)
        {
            if (message[i].Contains("src:"))
            {
                if (senderList[i] == MainWindow.current_User)
                {
                    if (message[i].Contains(".jpg") || message[i].Contains(".png") ||
message[i].Contains(".gif"))
                    {
                        html += $"<div class=\"box right\"><div
class=\"reciver\"><span>{senderList[i]}</span></div><div class=\"message\"><img
src=\"{message[i].Replace("src:", "")}\" width=\"100\" height=\"100\"/></div></div>";

                    }

                    else if (message[i].Contains(".pdf"))

```

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				95
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        {
            html += $"<div class=\"box right\"><div
class=\"reciver\"><span>{senderList[i]}</span></div><div
class=\"message\"><a
download=\"proposed_file_name\" href =\"{message[i].Replace(\"src:\", \"\")}\">View PDF
FILE</a></div></div>";
        }
    }
    else
    {
        html += $"<div class=\"box right\"><div
class=\"reciver\"><span>{senderList[i]}</span></div><div
class=\"message\"><a
download=\"proposed_file_name\" href =\"{message[i].Replace(\"src:\", \"\")}\">Download
file</a></div></div>";
    }
}
}
else
{
    if (message[i].Contains(".jpg") || message[i].Contains(".png") ||
message[i].Contains(".gif"))
    {
        html += $"<div class=\"box left\"><div
class=\"reciver\"><span>{senderList[i]}</span></div><div
class=\"message\"><img
src=\"{message[i].Replace(\"src:\", \"\")}\" width=\"100\" height=\"100\"/></div></div>";
    }
    else if (message[i].Contains(".pdf"))
    {
        html += $"<div class=\"box left\"><div
class=\"reciver\"><span>{senderList[i]}</span></div><div
class=\"message\"><a
download=\"proposed_file_name\" href =\"{message[i].Replace(\"src:\", \"\")}\">View PDF
FILE</a></div></div>";
    }
}
}
else
{
    html += $"<div class=\"box left\"><div
class=\"reciver\"><span>{senderList[i]}</span></div><div
class=\"message\"><a

```

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		96

```

download=\"proposed_file_name\" href =\"{message[i].Replace(\"src:\", \"\")}\">Download
file</a></div></div>;

    }

    }

    continue;

}

if (senderList[i] == MainWindow.current_User)

{

    html += $\"<div class=\"box right\"><div
class=\"reciver\"><span>{senderList[i]}</span></div><div
class=\"message\"><span>{message[i]}</span></div></div>\";

    }

    else

    {

        html += $\"<div class=\"box left\"><div
class=\"reciver\"><span>{senderList[i]}</span></div><div
class=\"message\"><span>{message[i]}</span></div></div>\";

        }

    }

    html +=
$\"</div><script>window.scrollTo(0,document.body.scrollHeight);</script><body></html>\";

    }

    else

    {

        html = \"<h1> No messages yet </h1> <style> h1{ color: white; align-text:center;}
<style>\";

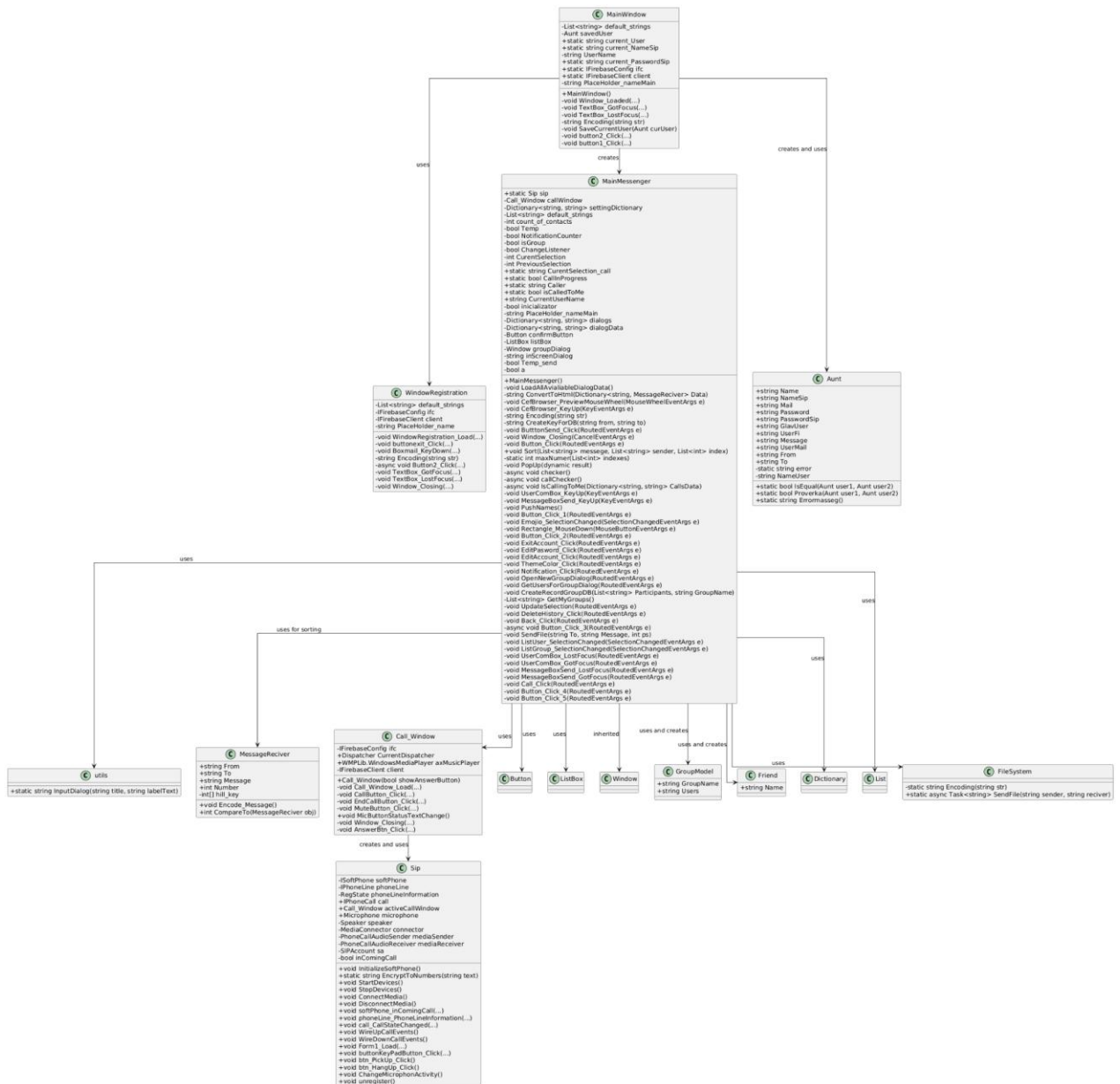
    }

    return html;

}

```

		Юхимчук Н.О.			ККІКБ.КР.Б – 123 – 14 – ПЗ	Арк.
		Граф М.С.				97
Змн.	Арк.	№ докум.	Підпис	Дата		



Додаток В Блок схема алгоритму шифрування повідомлення

