МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій

«До захисту допущено»
В.о. завідувача кафедри

_____ Світлана ВАЩЕНКО

____ _____ 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня магістр

зі спеціальності 122 «Комп'ютерні науки»,

освітньо-наукової програми «Інформаційні технології проектування»

на тему: Інформаційна технологія проектування месенджера з використанням біометричної автентифікації

Здобувача групи ІТ.м-21н Нагорного Євгенія Миколайовича

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідні джерела.

_____ Євгеній НАГОРНИЙ
(підпис) (Ім'я та ПРІЗВИЩЕ здобувача)

Керівник к.т.н., доцент, Володимир НАГОРНИЙ _____
(посада, науковий ступінь, вчене звання, Ім'я та ПРІЗВИЩЕ) (підпис)

Суми - 2024

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій
Спеціальність 122 «Комп'ютерні науки»
Освітньо-наукова програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри ІТ

Світлана ВАЩЕНКО

«_____» _____ 2024 р.

ЗАВДАННЯ

на кваліфікаційну роботу магістра студентові

Нагорного Євгенія Миколайовича
(прізвище, ім'я, по батькові)

1 Тема кваліфікаційної роботи *Інформаційна технологія проектування месенджера з використанням біометричної автентифікації*
затверджена наказом по університету від «01» лютого 2024 р. № 0096-VI

2 Термін здачі студентом кваліфікаційної роботи «10» __травня___ 2024 р.

3 Вхідні дані до кваліфікаційної роботи *Літературні джерела з питань проектування месенджера та використання біометричної автентифікації.*

4 Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *Аналіз предметної області; Постановка задачі та аналіз методів дослідження; Проектування інформаційної технології; Розробка інформаційної технології проектування месенджера з використанням біометричної автентифікації.*

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових слайдів презентації) *Обґрунтування актуальності, Постановка задачі, Функціональні вимоги, Структурно-функціональне моделювання, Моделювання варіантів використання, Моделювання бази даних, Розробка інформаційної технології, Висновки, Апробація результатів роботи.*

6. Консультанти випускної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

Дата видачі завдання _____.
Керівник _____
(підпис)
Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної області	1.11.23-29.12.23	
2	Огляд існуючих технологій	10.12.23-29.12.23	
3	Аналіз існуючих технології	22.12.23-29.21.23	
4	Мета та задачі дослідження розробки	03.01.24-10.01.24	
5	Моделювання інформаційної технології	11.01.24-29.01.24	
6	Реалізація технології	01.02.24-31.03.24	
7	Впровадження біометричної автентифікації	29.02.24-24.03.24	
8	Оформлення документації	23.04.24-19.05.24	

Магістрант _____ Євгеній НАГОРНИЙ

Керівник роботи _____ к.т.н., доц. Володимир НАГОРНИЙ

АНОТАЦІЯ

Тема кваліфікаційної роботи магістра «Інформаційна технологія проектування месенджера з використанням біометричної автентифікації».

Пояснювальна записка складається зі вступу, 4 розділів, висновків, списку використаних джерел із 27 найменувань, додатків. Загальний обсяг роботи − 81 сторінка, у тому числі 43 сторінок основного тексту, 3 сторінок списку використаних джерел, 27 сторінок додатків.

Перший розділ присвячено аналізу предметної області. Розглянуто поняття інформаційна технологія, наведено їх основні типи та проблеми. Наведено аналіз біометричної автентифікації, та з'ясовані у порівнянні кращі методи. Розглянуто сучасні месенджери стосовно їх переваг та недоліків, та методів їх розробки.

У другому розділі показані методи дослідження що включають в себе кілька компонентів, таких як моделювання мережевого трафіку, алгоритми маршрутизації повідомлень, механізми шифрування та автентифікації, а також інтерфейси користувача. До цього розділу ще додані методи дослідження біометричної автентифікації.

Третій розділ містить результати проектування інформаційної технології у вигляді діаграм IDEF0 та варіантів використання. Також наведено частину логічну модель бази даних месенджера, з таблицями, необхідними для роботи інформаційної технології.

У четвертому розділі представлено архітектуру інформаційної технології. Наведено опис розробленого месенджера, приклад інтеграції та використання біометричної автентифікації.

Результатом роботи є інформаційна технологія проектування месенджера, розроблена у вигляді програмного модуля, в яку інтегрований метод біометричної автентифікації.

Ключові слова: інформаційна технологія, біометрична автентифікація, месенджер.

ЗМІСТ

ВСТУП

Захист інформаційних технологій надає можливості без обмежень для розвитку і вдосконалення інструментів комунікації в онлайн суспільстві. Через постійне зростання популярності даних програм, потреба у модернізації нинішніх методів автентифікації, та розробкою сучасних методів стає більш важливою.

Актуальність. Месенджери стали невід'ємною частиною нашої щоденної комунікації, і захист особистої інформації стає все більш важливим. Використання біометричної автентифікації у месенджерах може забезпечити вищий рівень безпеки, оскільки це базується на унікальних фізичних характеристиках користувача. Розробка таких систем дозволить забезпечити надійний захист даних та зробить процес автентифікації зручнішим для користувачів. Таким чином, створення інформаційної технології месенджера з біометричною автентифікацією є актуальною та важливою задачею.

Об'єкт дослідження. Проектування месенджера з використанням біометричної автентифікації.

Предмет дослідження. Інформаційна технологія проектування месенджера з використанням біометричної автентифікації.

Новизна результатів. Наукова новизна дослідження є у розробці нової інформаційної технології проектування месенджера, яка інтегрує біометричну автентифікацію для забезпечення високого рівня безпеки і зручності користувачів.

Мета. Розробити інформаційну технологію проектування месенджера з використанням біометричної автентифікації.

Задачі дослідження.

- Провести аналіз предметної області;
- Поставити задачі та проаналізувати методи дослідження;
- Спроектувати інформаційну технологію;

- Розробити інформаційну технологію проектування месенджера з використанням біометричної автентифікації.

Практичне значення. Розроблена інформаційна технологія дозволяє спроектувати месенджер із використанням біометричної автентифікації, що дозволить з зменшити ризик отримати дані користувача, та запобігти шахрайським діям.

Апробація результатів. По результатам роботи було опубліковано тези на конференції «Інформатика, Математика, Автоматика 2024».

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд популярних месенджерів та методів автентифікації

Месенджер - це додаток або програма, що завантажують та інсталюють на смартфон або комп'ютер. Основною метою є миттєвий обмін повідомленнями, фото, картинками, відео, документами з друзями, родичами, знайомими, колегами. Ще є можливість телефонувати через аудіо та відео зв’язок[1]. Обмін повідомленнями відбувається миттєво в режимі реального часу між користувачами. Повідомлення відправляється співрозмовнику відразу після того, як відправник завершить введення даних, редагування на відправить дані. При цьому користувач, який отримує повідомлення має бути на зв’язку, в інакшому випадку повідомлення буде очікувати, коли він також зайде до месенджеру і прочитає його[2].

Цікаво те, що месенджери є не винаходом 21 століття. Перші месенджери розроблені наприкінці 1980-х у США. Це були Talk (1982 рік) і Zephyr (1987). На пострадянському просторі першим месенджером був додаток ICQ, який розробили у середині 1990-х і швидко стала популярною серед користувачів. На сьогоднішній день розроблені більше сотні різноманітних аналогів [3-4]. На рисунку 1.1 зображено результати опитування в 2024 році на квітень місяць щодо найбільш популярного месенджера в світі.

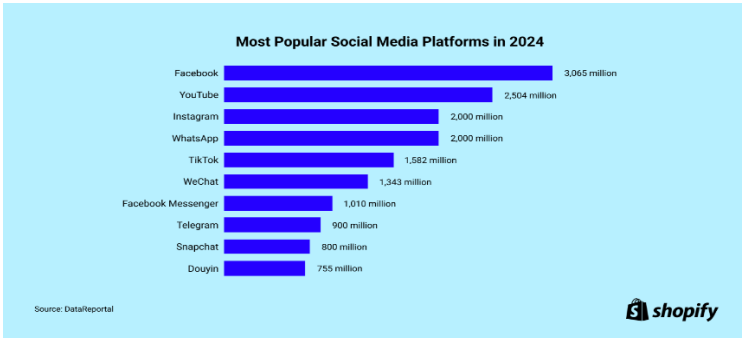

Рисунок 1.1 – Гістограма найпопулярніших месенджерів у світі

Джерело:[5]

Розвиток схожих додатків в подальшому визначався їх функціональністю, оптимізацією. ICQ поступився першим місцем зі Skype, через те, що підтримував аудіо та відео дзвінки [6]. На даний час лідерами є WhatsApp, Telegram та Viber.

Надалі суспільство дійшло до інформаційного середовища, де існує маса видів комунікації, що спричинило стрибок у вдосконаленні месенджерів та популяризації месенджерів Facebook Messenger, Instagram Direct Telegram, Signal, Viber, WhatsApp, та інші.

Упродовж початку осені 2021-го року, Київський міжнародний інститут соціології проводив опитування громадської думки «Омнібус». Методом телефонних інтерв'ю опитування проводилося тільки на території, що контролюється українською владою. Найпоширенішим засобом є Viber, яким користується 73,6% опитаних. Другим є Месенджер Фейсбуку – 42,7%. Третє та четверте місце зайняли Телеграм (31,6%) та WhatsApp (25,3%). Найменшою популярністю є Signal, використовують його 3,8%. Майже третина українців не обмінюється повідомленнями з мобільного телефону в жодному з вищезгаданих застосунків (18,7%). Динаміка використання зображає, що більше стали користуватись Viber-ом (з 65,8% до 73,6%), Facebook Messenger-ом (з 34,9% до 42,7%) та Telegram (24,2% до 31,6%). За розподілом статі, трохи більше чоловіків, ніж жінок використовують Telegram (34,9 та 28,9% відповідно)[7]. У таблиці 1.1 зображено використання можливостей для обміну повідомленнями.

Таблиця 1.1 – Популярність месенджерів для обміну повідомленнями

Месенджер	**%**
Viber	73,6
WhatsApp	25,3
Facebook Messenger	42,7

Продовження таблиці 1.1

Месенджер	**%**
Telegram	31,6
Twitter (X)	4,9
Signal	3,8
Skype	11,3
Тік Ток	9,1
Instagram	26,4
Інше	2,6
Нічим з перерахованого	18,7
Важко сказати	0,8

Джерело: побудовано автором на основі даних зі статті [7]

Інформаційна технологія – цілеспрямована організована сукупність інформаційних процесів з використанням комп'ютерної та мобільної техніки, які надають високу швидкість обробки даних, пришвидшує пошук, зосередження даних, доступ до інформації незалежно від місця розташування [8].

Найбільшою загрозою для інформації, яка оброблюється та зберігається за допомогою інформаційних систем, є несанкціонований до неї доступ. Відповідно з

цього, головною задачею захисту інформації являється ідентифікація та автентифікація для забезпечення розмежовування доступу та конфіденційності (Рис. 1.2).

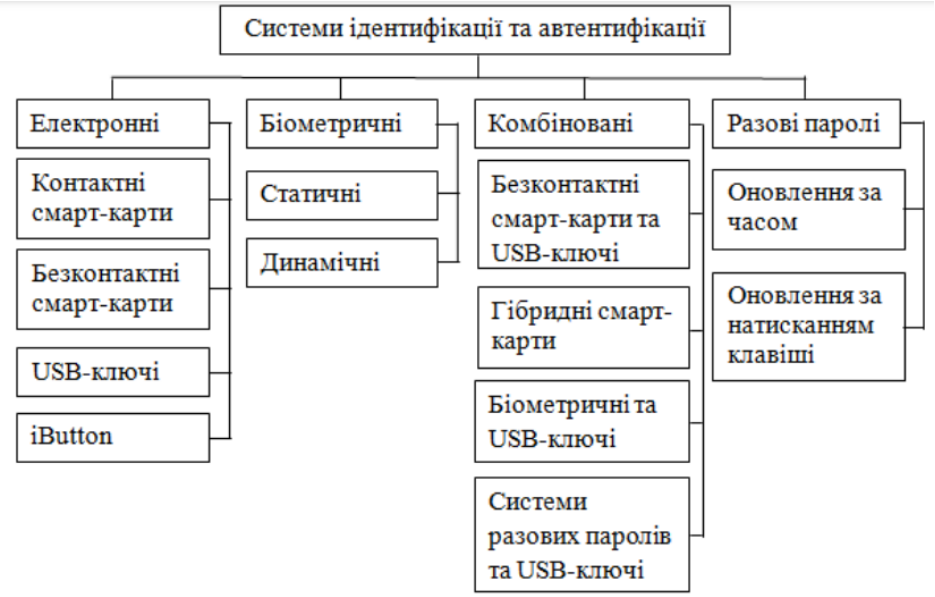

Рисунок 1.2 – Класифікація систем автентифікації та ідентифікації.

Джерело: [8]

Ідентифікація – це розпізнавання суб'єкта. Нею є можливість досягти за допомогою ідентифікатора (наприклад, логін або електронна пошта), ідентифікатора процесу тощо. Дуже важливо, щоб заявлені облікові дані були унікальними, щоб можна було розрізняти різні суб’єкти в системі. У технології не може бути зареєстровано два однакових ідентифікатора, дві однакові електронні пошти і т.д.. Наприклад, при раєстрації користувач вводить свої дані в поле, а технологія видає вікно з надписом «Користувач вже зареєстрований» [8].

Потім, як користувач ідентифікується, його потрібно автентифікувати, тобто користувач має підтвердити, що він насправді той, хто є насправді . Доказом ідентичності є надання облікових даних механізму контролю доступу. Потім йде перевірка дійсності наданих облікових даних перед тим, як підтвердити запит. Іншими словами, автентифікація означає, що користувач як володіє, так і контролює надані дані. Деякими

прикладами які можна використати для підтвердження, є паролі, PIN-коди, цифрові підписи, біометричні дані [9].

У різних варіантів інформаційних технологій виявляються три види автентифікації:, що використовують для розпізнавання користувачів: щось, що відомо користувачу; чим володіє користувач; що притаманне для користувача. Для дослідження та використання потрібно обрати метод автентифікації, що використовує щось притаманне користувачу.

Біометрична автентифікація — це автентифікація, що використовує інформацію біометрики. Розроблена автентифікація, така як відбитки пальців, скан сітківки ока, автентифікація за голосом, і т.д.. Вона складається з 8-ми пунктів про біометричну автентифікацію [10].

На даний час можна виділити чотири відмінні шаблони автентифікації: пряма; непряма; локальна; автономна [11].

До локальної автентифікації можна віднести те, що люди працюють з системою безпосередньо на місці, а не віддалено від нього. Приклади включають в себе портативні пристрої, але сюди можна віднести і робочі станції, які працюють на місці (Рис. 1.3).

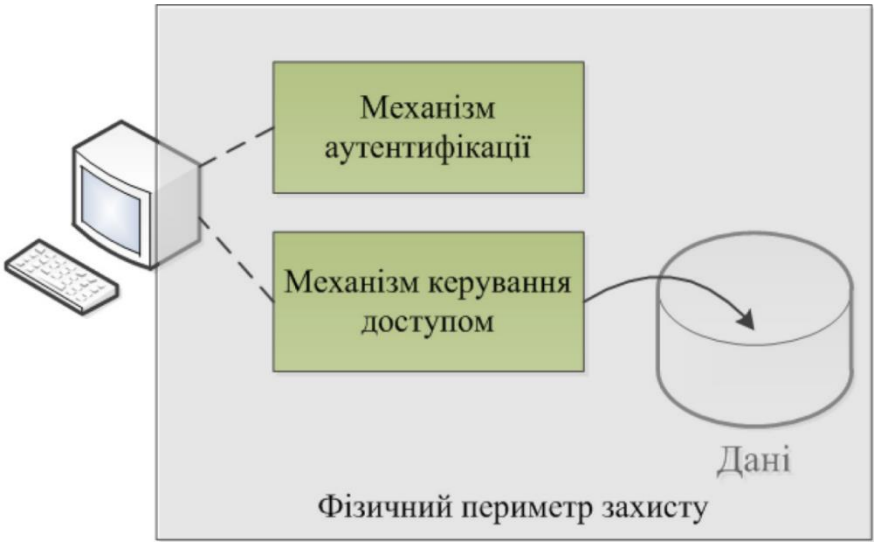

Рисунок 1.3 – Графічна модель для локальної автентифікації

Джерело: [12]

Пряма автентифікація, яку можна бачити в старих серверних системах, що стоять в локальних мережах з розподілом часу. Серверною системою мають змогу користуватися віддаленим чином багато користувачів. Механізми автентифікації і контролю доступом системи розміщені всередині одного фізичного периметра. Власник проводить підтримку бази даних автентифікації всередині системи [12]. На рисунку 1.4 зображено модель розгортання системи для прямої автентифікації.

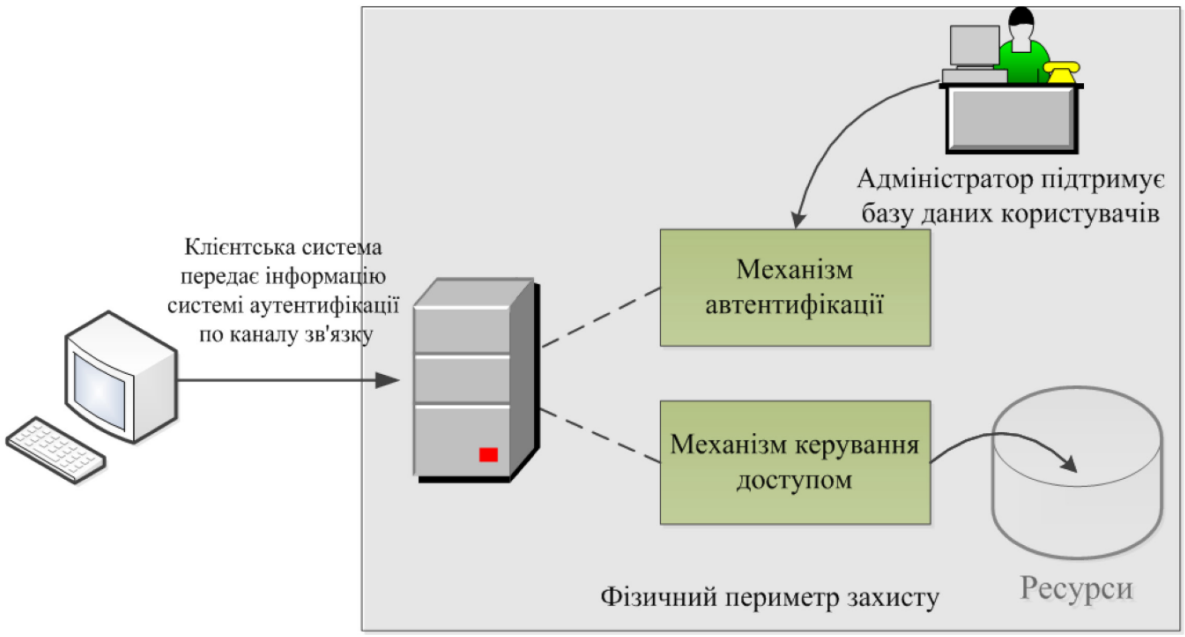

Рисунок 1.4 – Графічна модель розгортання системи для прямої автентифікації

Джерело: [12]

Рішення з урахуванням непрямої автентифікації вирішують завдання масштабованості на обчислювальних центрах, що має одна група користувачів, але кілька точок для їх обслуговування. Навіть на місці всього з двома серверами важко підтримувати сумісність двох окремих баз даних автентифікації. Якщо два попередні методи передбачають об'єднання механізмів автентифікації та управління доступом, то

в даній моделі механізм автентифікації з точки обслуговування переходить на сервер автентифікації, що колективно використовується, з єдиною базою даних записів про користувачів. Всі інші компоненти надають послуги чи керують доступом до ресурсів, але не ухвалюють рішення про автентифікацію. Непряма автентифікація знайшла застосування в багатьох додатках, що керують трафіком на межах мереж/підмереж і використовують одноразові паролі та ресурси мережної операційної системи [12]. На рисунку 1.5 зображено модель непрямої автентифікації.

Рисунок 1.5 - Графічна модель розгортання системи для непрямої автентифікації

Джерело: [12]

Автономна автентифікація зустрічається в системах з відкритими ключами, в яких безліч компонентів, які можуть приймати рішення по управлінню доступом навіть тоді коли вони не можуть зв'язуватися з іншими системами для отримання рішень. Власник погоджується з ризиком того, що дані рішення можуть прийматися з використанням застарілих даних з управління доступом або автентифікації, а отже, можуть давати результат некоректно [13].

1.2 Аналіз існуючих месенджерів та захисту повідомлень

Viber – це месенджер, що дозволяє дзвонити та надсилати СМС іншим користувачам безкоштовно, та за кошти користувачам, що незареєстровані. Месенджер був запущений у 2010 році. Використовувати його можна за допомогою Wi-Fi або мобільного інтернету. На зараз використовують месенджер більше 1 млрд людей по всьому світу.

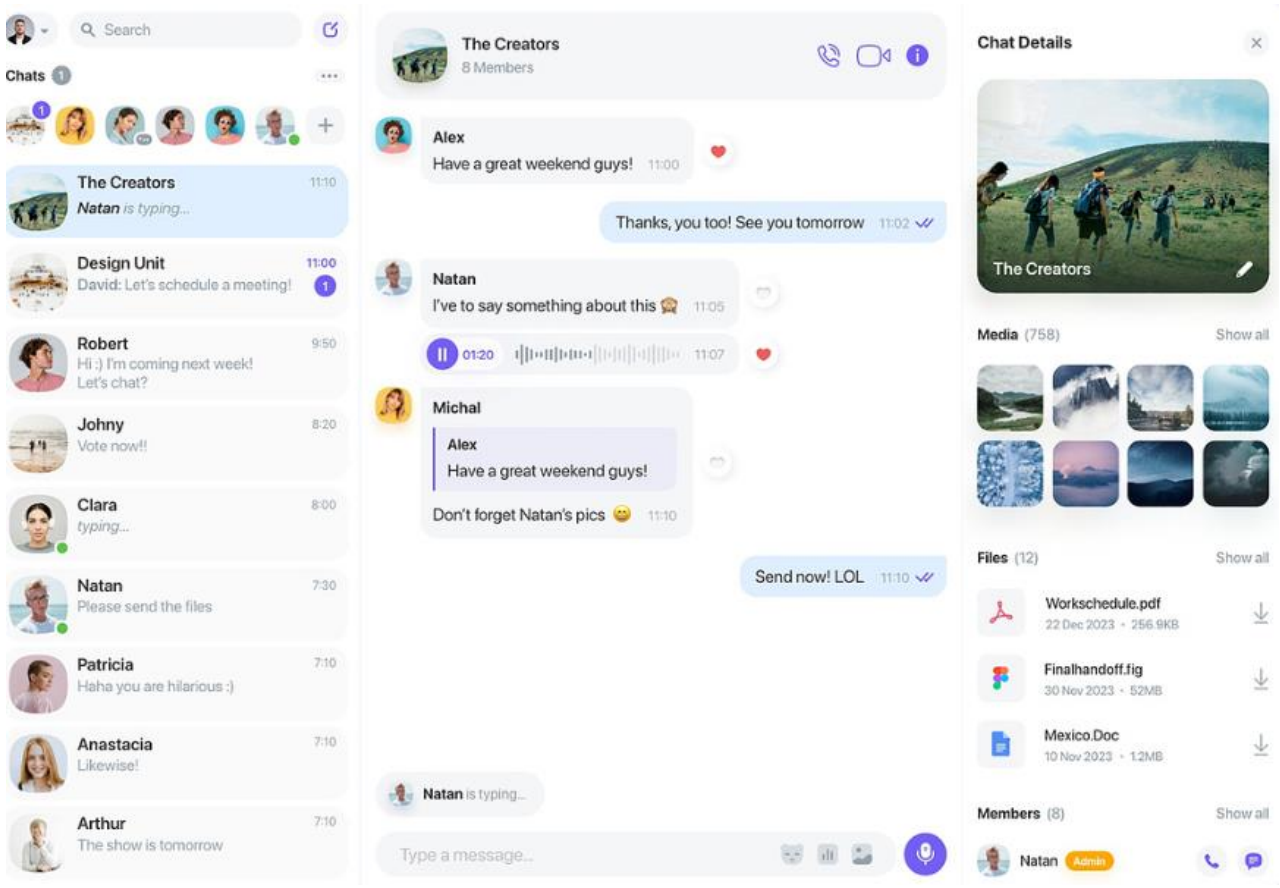

Рисунок 1.6 – Інтерфейс середовища месенджеру Viber

Джерело: [14]

Переваги:

- Обмін фото, а також відео та аудіо смс.
- Безкоштовні дзвінки за кордон.

- Шифрування повідомлень, як і текстових повідомлень, так і відео і аудіо.
- Перевірка достовірності кориистувачів вручну, щоб впевнитись, що спілкуєтеся саме з тим, з ким потрібно.
- Можливість видалення повідомлення навіть після надсилання.
- Можна за кошти дзвонити на мобільні телефони, якщо на них не встановлено програму Viber.

Недоліки:

- Багато реклами.
- Немає вебверсії, тільки додаток.
- Великий об’єм самого додатку.
- Поганий зв'язок при дзвінках, незалежно від якості з’єднання.
- Можливість злому. Головний мінус додатку є відсутність тестів на безпеку.
- Застарілий дизайн.
- Немає зберігання історії з користувачами.

Telegram запущений у 2013-ому році, використовують його більше 500 мільйонів користувачів. Додаток створений тим, хто надає перевагу швидкому та надійному зв’язку повідомленнями і дзвінками. Telegram можна встановлювати на телефон, планшет, комп'ютер або використовувати веб-інтерфейс. Він є зручним, безпечним та безкоштовний, не враховуючи підписки на більшість функцій у месенджері.

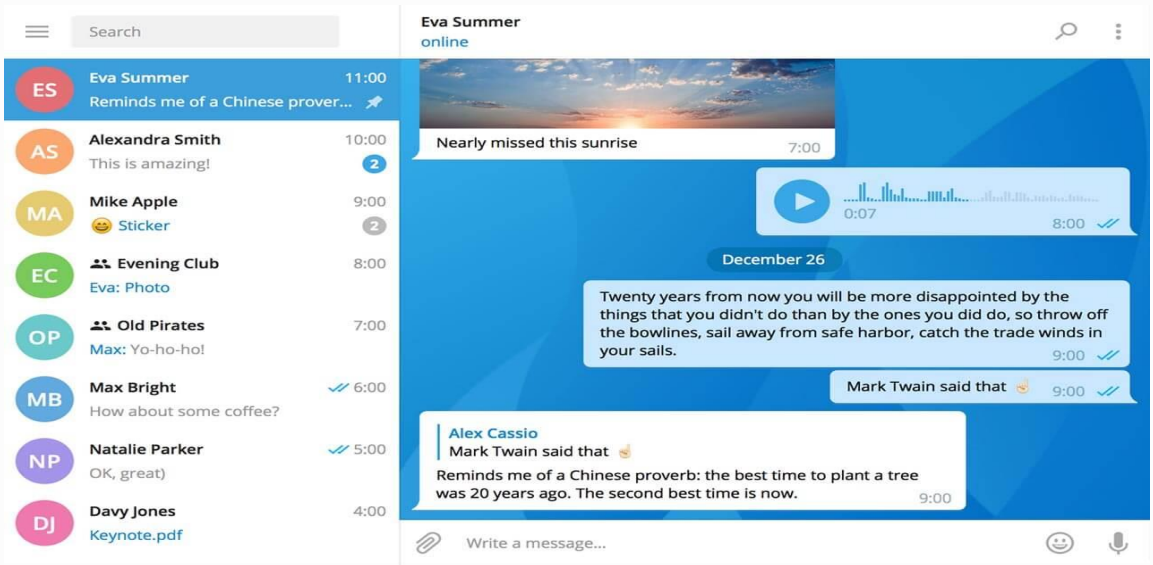

Рисунок 1.7 – Інтерфейс месенджеру Telegram

Джерело: [15]

Переваги:

- Використання аудіо та відео.
- Надсилати медіафайли без обмежень за розміром чи типом.
- Історія листування зберігається у хмарі і не займає зайвого місця на пристроях.
- Не надає доступу до даних іншим особам.
- Вхід на декількох пристроях в один час. Обмежень на кількість сесій немає.
- Месенджер групує дані у мінімально можливу кількість байтів, що дає змогу надсилати та отримувати повідомлення зі слабким з'єднанням.
- Підходить для створення онлайн чатів.

Недоліками є:

- Збої у додатку, через що Telegram може не працювати.
- Прив'язка до номера телефону. Навіть якщо в Telegram ви дотримуєтеся статусу інкогніто, порівняння бази номерів і бази месенджера покаже номер. Також можна ввести підтвердження входу в додаток за одноразовим кодом іншим користувачем.
- Повідомлення у Telegram не зашифровані. Вони передаютьсяу вигляді зашифрованих протоколів, але на серверах зберігаються у відкритому вигляді.

WhatsApp — Ним користуються понад 2 млрд людей у 180 країнах. Заупск додатку був у 2009-ому році.

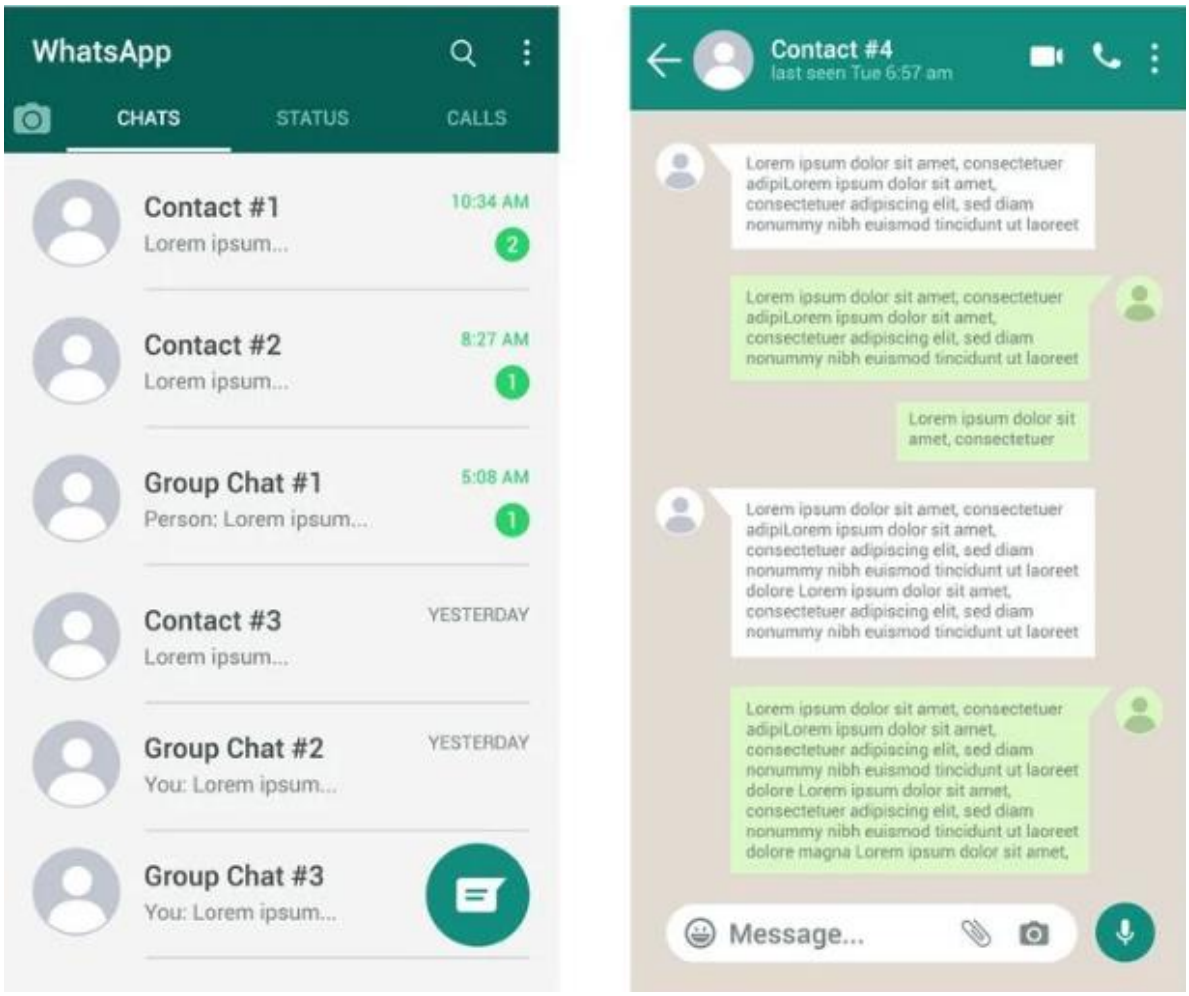

Рисунок 1.8 – Інтерфейс месенджеру WhatsApp

Джерело: [16]

Переваги:

- Проста форма реєстрації користувачів.
- Легкий у використанні.
- Обмін фото, відео, документів та голосовими повідомленнями, та безкоштовні дзвінки за кордон.
- Автоматичне імпортування контактів із телефону до додатку.
- Відсутність реклами
- Швидка робота програми.
- Зміна закріплених за обліковим записом номеру телефону без втрати даних.

- Повідомлення можна надсилати та отримувати у браузері.

Недоліки:

- Відсутність можливості скасувати надсилання повідомлення.
- WhatsApp є тільки мобільним додатком.
- Висока можливість злому.
- Займає багато місця у пам'яті через накопичення різних файлів.
- Може стягувати плату за користування інтернетом.
- Багато спаму.

Signal — це месенджер для обміну повідомленнями на телефоні або комп'ютерах. Додаток здобуває все більше популярності через конфіденційності спілкування. Функціонал схожий на Facebook Messenger – можна обмінюватися фото, відео, GIF-файлами, голосовими повідомленнями з користувачами, вести відеочат та спілкуватися у чатах. Signal запущений у 2010-ому році.

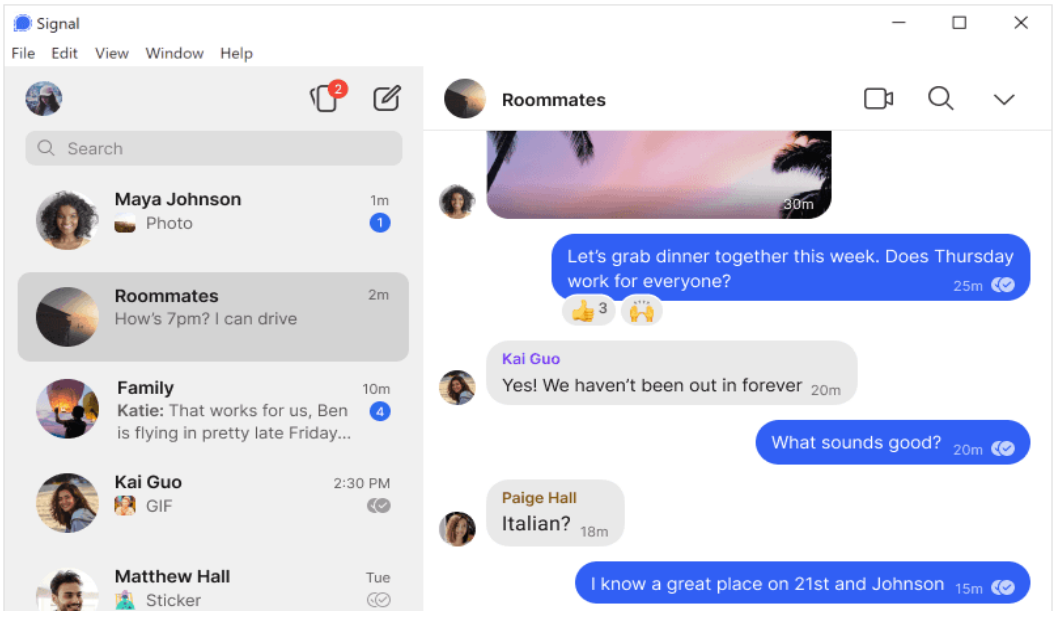

Рисунок 1.9 – Інтерфейс месенджеру Signal

Джерело: [17]

Переваги:

- Є шифрування, що забезпечує безпеку для користувачів.

- Швидкість відправки та отримань повідомлень висока навіть у повільних мережах.
- Відсутній спам та реклама.
- Додаток не збирає даних про користувачів.
- Під одним номером месенджер можна встановити на п'ятьох пристроях одночасно. Обмеження розробили для безпеки.

Недоліки:

- Під час установки месенджера на інший пристрій або при зміні номера, чати, файли та інші дані втрачаються. Програма зберігає їх на пристрій і не синхронізує зі сервером.
- Не відправляються файли, що займають більше 100 МБ.
- Груповий чат може містити не більше 1000 осіб[18].

Таблиця 1.2 – Порівняльна характеристика месенджерів

Характеристика	**Viber**	**Telegram**	**WhatsApp**	**Signal**	**Власна розробка**
Шифрування даних	E2E	E2E	E2E	E2E	E2E
Відкритість коду	-	Частково	-	+	+
Груповий чат	+	+	+	+	-
Біометрична автентифікація	+	-	+	+	+

Джерело: побудовано автором

2 ПОСТАНОВКА ЗАДАЧІ ТА МЕТОДИ ДОСЛІДЖЕННЯ

2.1 Мета та задачі дослідження розробки інформаційної технології

Мета**.** Розробити інформаційну технологію проектування месенджера з використанням біометричної автентифікації.

Для досягнення поставленої мети потрібно виконати наступні задачі:

- Провести аналіз предметної області;
- Поставити задачі та проаналізувати методи дослідження;
- Спроектувати інформаційну технологію;
- Розробити інформаційну технологію проектування месенджера з використанням біометричної автентифікації.
- **Провести аналіз предметної області**

Необхідно детально дослідити всі аспекти функціонування месенджера з біометричною автентифікацією. Це включає вивчення потреб користувачів, вимог безпеки, існуючих технологій біометричної автентифікації та аналогічних рішень на ринку. Результатом буде повне розуміння контексту, в якому працюватиме система, і виявлення ключових вимог та обмежень.

- **Поставити задачі та проаналізувати методи дослідження**

Слід визначити конкретні задачі, які необхідно вирішити для розробки системи, такі як забезпечення надійності автентифікації, швидкість роботи месенджера, і захист даних користувачів. Після цього потрібно розглянути можливі методи дослідження для кожної задачі, включаючи експериментальні, аналітичні та моделювальні підходи. Результатом буде чіткий план дій і обрані методи, які найкраще підходять для вирішення поставлених задач.

- **Спроектувати інформаційну технологію**

На цьому етапі необхідно створити архітектурний дизайн системи, визначити основні компоненти та їх взаємодію. Важливо також розробити логічну та фізичну моделі бази даних, які будуть підтримувати зберігання та обробку даних в системі.

Результатом буде детальний проект інформаційної технології, що забезпечує всі необхідні функціональні та нефункціональні вимоги.

- **Розробити інформаційну технологію проектування месенджера з використанням біометричної автентифікації**

Цей етап включає безпосередню реалізацію запроектованої технології, що включає написання коду, інтеграцію біометричних методів автентифікації, та налаштування бази даних. Також необхідно провести тестування системи для виявлення та виправлення помилок. Результатом буде працююча система месенджера з біометричною автентифікацією, готова до розгортання та використання.

2.2 Методи дослідження

Математична модель месенджера включає кілька компонентів, таких як моделювання мережевого трафіку, алгоритми маршрутизації повідомлень, механізми шифрування та автентифікації, а також інтерфейси користувача.

Розглянемо детальніше математичну модель месенджера з врахуванням усіх важливих аспектів: мережевий трафік, алгоритми маршрутизації, механізми шифрування та автентифікації, а також інтерфейси користувача.

Моделювання мережевого трафіку

Мережевий трафік в месенджері включає передачу повідомлень між клієнтами через сервери. Основою для моделювання трафіку є теорія черг[19].

Модель черги

Модель черги описує процес прибуття повідомлень на сервер і їх обробку. Для простоти, розглянемо серверну систему (M/M/1).

- Інтенсивність вхідного трафіку (λ): кількість повідомлень, що надходять на сервер в одиницю часу.

- Інтенсивність обслуговування **(μ)**: кількість повідомлень, що можуть бути оброблені сервером в одиницю часу.

Середній час очікування в черзі (W_q) (формула 2.1):

$$Wq=\mu/(\mu-\lambda)\lambda \quad (2.1)$$

Де λ — інтенсивність вхідного трафіку, тобто середня кількість повідомлень, що надходять на сервер за одиницю часу; $\mu\mu$ — інтенсивність обслуговування, тобто середня кількість повідомлень, що можуть бути оброблені сервером за одиницю часу; W_q — середній час, який повідомлення очікує в черзі перед обробкою.

Середня кількість повідомлень в черзі (L_q) (формула 2.2):

$$Lq=\mu/(\mu-\lambda)\lambda 2 \quad (2.2)$$

Де λ — інтенсивність вхідного трафіку, тобто середня кількість повідомлень, що надходять на сервер за одиницю часу; μ — інтенсивність обслуговування, L_q — середній кількість повідомлень, що очікують в черзі.

Для складніших систем з багатьма серверами або різними пріоритетами повідомлень можуть використовуватися моделі типу M/M/c або M/G/1.

Алгоритми маршрутизації повідомлень

Алгоритми маршрутизації визначають, як повідомлення переміщаються мережею для досягнення кінцевого одержувача. Вони забезпечують оптимальний та ефективний шлях доставки повідомлень, що сприяє швидкій та надійній комунікації між користувачами месенджера.

Алгоритм Дейкстри:

Одним з популярних алгоритмів для знаходження найкоротшого шляху в мережі є алгоритм Дейкстри[20]. Він визначає найкоротший шлях від одного вузла до всіх інших у графі з невід'ємними вагами ребер.

- Ініціалізація: Встановити відстань до початкового вузла *s* рівною 0, а до всіх інших вузлів — нескінченність.
- Встановити початковий вузол як поточний.
- Для поточного вузла оновити відстані до його сусідів, якщо новий шлях коротший.
- Вибрати невідвіданий вузол з найменшою відстанню і зробити його поточним.
- Повторити кроки 3-4, поки всі вузли не будуть відвідані[20].

Механізми шифрування

Безпека в месенджерах досягається за допомогою шифрування повідомлень[21].

Симетричне шифрування (AES):

- Шифрування (формула 2.3):

$$C=Ek(P)C=Ek(P) \tag{2.3}$$

де P – відкритий текст повідомлення, E_k – функція шифрування з ключем k

- Розшифрування (формула 2.4):

$$P=Dk(C)P=Dk(C) \tag{2.4}$$

Де D_k –функція розшифрування з ключем k

- Шифрування (формула 2.5):

$$C=Me\text{mod}\ nC=Me\text{mod}n \tag{2.5}$$

Де C – шифротекст, d – приватний ключ, n – модуль, M – відкритий текст повідомлення.

- Розшифрування(формула 2.6):

$$M=Cd\text{mod}\ nM=Cd\text{mod}n. \tag{2.6}$$

Де C – шифротекст, d – приватний ключ, n – модуль, M – відкритий текст повідомлення.

Інтерфейси користувача

Цей аспект включає дизайн і взаємодію користувача з месенджером

Модель взаємодії

Модель взаємодії користувача може бути описана через кількість кроків, необхідних для виконання задач[22].

Кількість кроків (K) (формула 2.8):

$$K=\sum i=1nSi \tag{2.8}$$

де $SiSi$ — кількість кроків для кожної операції.

Отже, використані джерела надають надійні теоретичні основи для моделювання трафіку, розробки алгоритмів маршрутизації, забезпечення безпеки шифрування та автентифікації, а також покращення інтерфейсів користувача в месенджерах. Цей підхід дозволяє створювати ефективні та безпечні комунікаційні засоби, що відповідають вимогам сучасних користувачів.

2.3 Методи дослідження біометричної автентифікації

Метод біометричної автентифікації ідентифікаціє користувача за ознаками, що пов'язані з фізіологічними особливостями, які в один час проводять ідентифікацію особи. До них можна віднести: геометричну будову руки, відбитки пальців, особливості малюнка сітківки ока, райдужну оболонку ока, характеристики мови, рукописний почерк, клавіатурний почерк, комп’ютерний почерк, та інші фізіологічні особливості людини, що робить її унікальною. Особливість автентифікації за біометричними параметрами з’ясовується на їх винятковості. Ймовірність, що будуть дві людини з однаковими ознаками майже нульова. Основними характеристиками перерахованих вище методів ідентифікації наведені в таблиці 1.2[23].

Таблиця 1.2 – Основні характеристики методів біометричної автентифікації

Метод отримання біометричних параметрів	**Ймовірність відмови у доступі %**	**Ймовірність помилкової ідентифікації «чужого» (без використання муляжу) %**	**Ймовірність помилкової ідентифікації «чужого» %**	**Збереження таємниці образу у процесі ідентифікації абонента**	**Вартість технічної реалізації в грошовому еквіваленті, у.о.**
Геометрична будова руки	0,2…4	0,2….1	10…75	неможливо приховати	Від 600 до 3000
Відбитки пальців	2...6	0,0001	10...70	неможливо приховати	Від 60 до 600
Клавіатурний почерк	3…9	3...9	_____	6-10…10-12	_____

Продовження таблиці 1.2

Метод отримання біометричних параметрів	**Ймовірність відмови у доступі %**	**Ймовірність помилкової ідентифікації «чужого» (без використання муляжу) %**	**Ймовірність помилкової ідентифікації «чужого» %**	**Збереження таємниці образу у процесі ідентифікації абонента**	**Вартість технічної реалізації в грошовому еквіваленті, у.о.**
Особливості малюнка сітківки ока	0,4	6...10	_____	неможливо приховати	приблизно 4000
Портрет обличчя	1…9	_____	_____	неможливо приховати	55000
Рукописний почерк	0,5…5	0,5…5	0,5...5	8-10…10-40	_____

Джерело: побудовано автором на основі даних зі статей [23]

Статичні та динамічні методи біометричної ідентифікації є взаємопов'язаними та взаємодоповнюючими напрямками. Основна перевага статичних методів полягає в їхній

відносній незалежності від психологічного стану користувача та незначних зусиллях з його боку, що дозволяє ефективно застосовувати їх для ідентифікації великих потоків людей. [24].

Біометрична ідентифікація на основі динамічних характеристик зазвичай простіша у реалізації, оскільки не потребує дорогого обладнання і може обмежуватися лише програмним забезпеченням, яке вимагає мінімальної підтримки фахівця під час експлуатації.

3 МОДЕЛЮВАННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ

3.1 Структурно-функціональне моделювання месенджера

IDEF0 – це визначення інтеграції для моделювання процесів, методологія загальнодоступного домену, яка використовується для моделювання бізнесу та їхніх процесів, щоб їх можна було зрозуміти та вдосконалити. Це тип блок-схеми[25]. Контекстна діаграма процесу біометричної автентифікації в месенджері, на якій показано основні взаємодії представлена на рисунку 3.1.

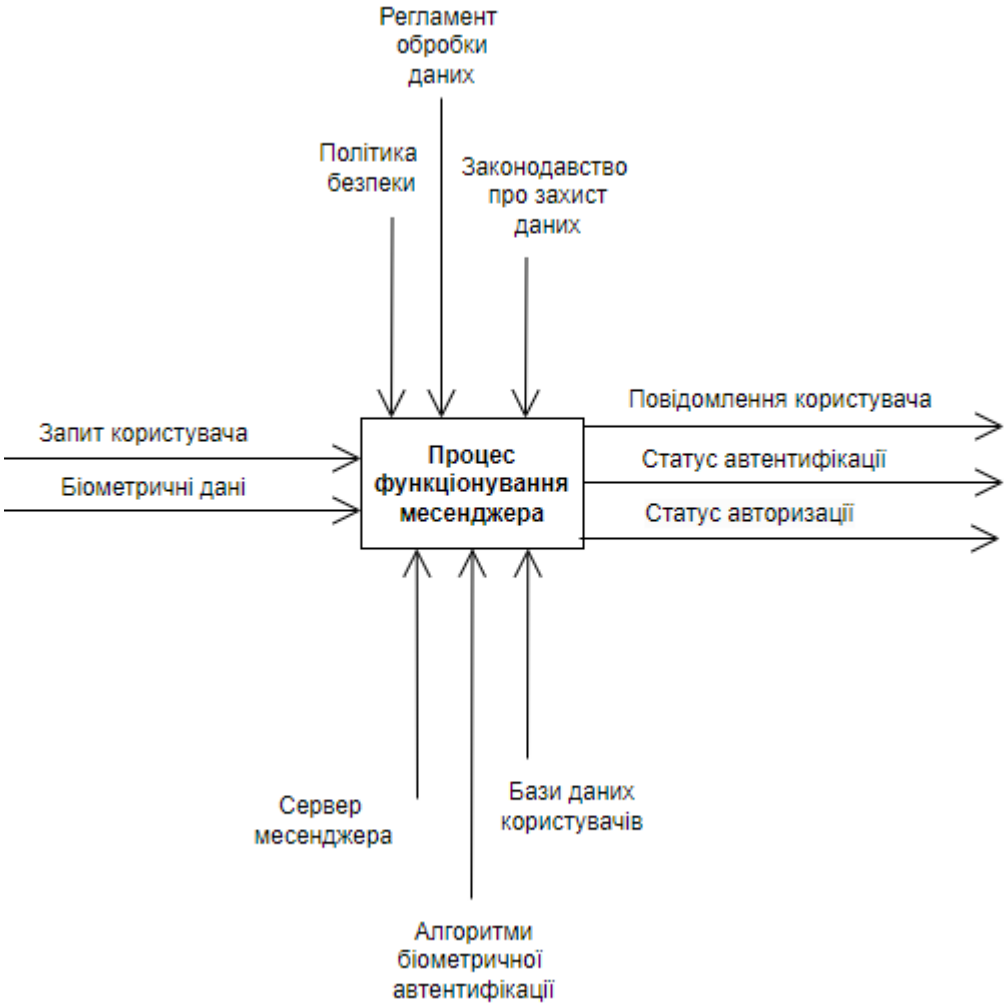

Рисунок 3.1 – Контекстна діаграма інформаційної технології в нотації IDEF0

Джерело: побудовано автором

Входи:

- Запит користувача - дані, які надходять від користувача, такі як текстові повідомлення, медіа-файли або запити на автентифікацію.
- Біометричні дані - біометрична інформація, що використовується для автентифікації користувача (наприклад, відбитки пальців, обличчя, голос).

Виходи:

- Повідомлення користувача - надіслані повідомлення, які передаються між користувачами через месенджер.
- Статус автентифікації - результат перевірки автентичності користувача, що визначає доступ до месенджера.
- Статус авторизації – результат авторизації користувача, що визначає дозволити, чи відхилити вхід.

Механізми управління:

- Сервер месенджера - основна платформа, на якій функціонує месенджер, забезпечує обробку даних, зберігання та передавання повідомлень.
- Алгоритми біометричної автентифікації - алгоритми, що використовуються для обробки та перевірки біометричних даних користувачів.
- Бази даних користувачів - зберігають інформацію про користувачів, їхні облікові записи та біометричні дані.

Контролі:

- Політика безпеки - правила та протоколи, які визначають вимоги до безпеки, зокрема використання біометричних даних.
- Регламент обробки даних - вимоги та стандарти щодо зберігання та обробки користувацьких даних.
- Законодавство про захист даних - нормативно-правова база, що регулює обробку персональних та біометричних даних.

Для детальнішого розуміння процесів оцінки, яка виконується, сама діаграма була декомпозована. Декомпозиція системи месенджера з біометричною автентифікацією зображено на рисунку 3.2.

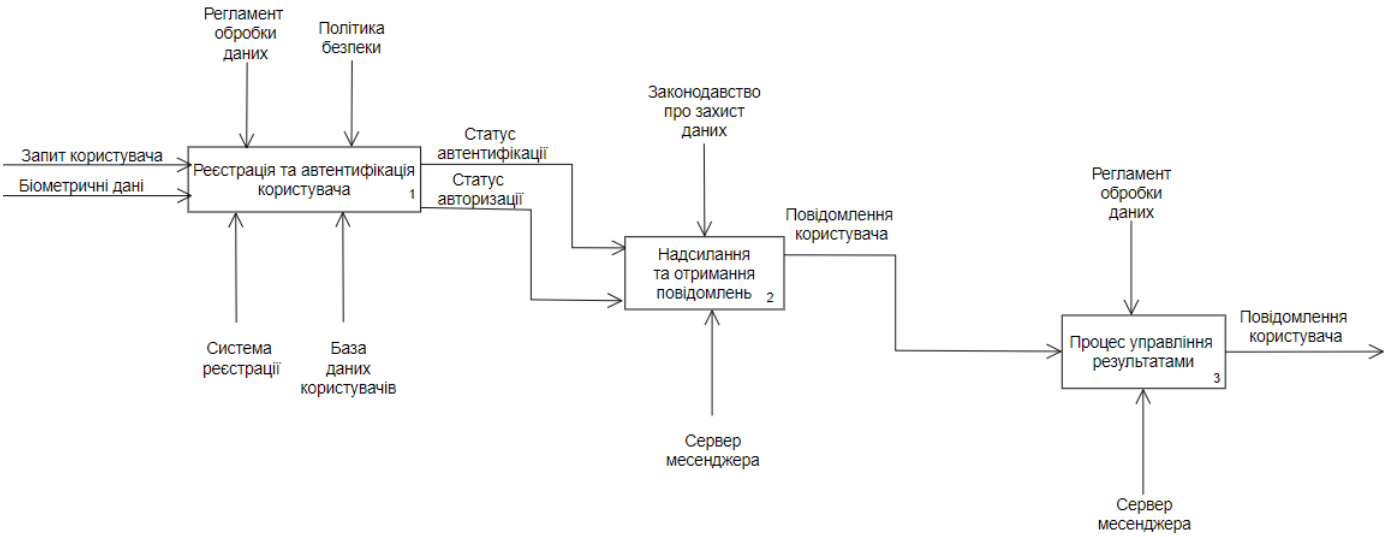

Рисунок 3.2 – Декомпозиція діаграми інформаційної технології IDEF0

Джерело: побудовано автором

Діаграма була декомпозована на такі блоки:

Реєстрація та автентифікація користувача – є процесом створення облікового запису для нового користувача (реєстрація) і перевірку особи користувача при спробі входу до системи (автентифікація) з використанням біометричних даних для підвищення рівня безпеки.

Вхідні потоки:

- Запит користувача - дані користувача, що реєструється.
- Біометричні дані - біометрична інформація, зібрана під час реєстрації.

Вихідні потоки:

- Статус автентифікації - результат перевірки автентичності користувача.

Механізми:

- Система реєстрації - програмне забезпечення, що забезпечує реєстрацію нових користувачів.
- База даних користувачів - обробляє та зберігає дані користувачів.

Контроль:

- Регламент обробки даних - правила реєстрації користувачів.
- Політика безпеки - заходи для захисту реєстраційних даних.

Надсилання та отримання повідомлень забезпечує обмін повідомленнями між користувачами, включаючи надсилання даних від одного користувача до іншого та отримання нових повідомлень на запит користувача.

Вхідні потоки:

- Статус автентифікації - результат перевірки автентичності користувача.

Вихідні потоки:

- Повідомлення користувача - повідомлення, яке надсилається іншому користувачеві.

Механізми:

- Сервер месенджера - забезпечує відправку та отримання повідомлень між користувачами.

Контроль:

- Законодавство про захист даних - правила захисту даних під час передачі повідомлень.

Блок процесу управління результатами - забезпечує перевірку, обробку та збереження результатів взаємодії користувача з системою, включаючи реєстрацію, автентифікацію та обмін повідомленнями.

Вхідні потоки:

- Повідомлення користувача - повідомлення, яке надсилається іншому користувачеві.

Вихідні потоки:

- Повідомлення користувача - повідомлення, яке надсилається іншому користувачеві.

Механізми:

- Сервер месенджера - забезпечує відправку та отримання повідомлень між користувачами.

Контроль:

- Регламент обробки даних - правила реєстрації користувачів.

3.2Моделювання діаграми Use Case месенджера

Для визначення вимог системи використовують діаграму Use Case. Саме ця діаграма показує, як технологія взаємодіє із об’єктами, що будуть її використовувати.

У діаграмі Use Case є актори

1. Користувач - це особа, яка використовує месенджер.
2. Сервіс біометричної автентифікації – система, що використовується для автентифікації користувачів на основі їх стилю набору тексту.
3. Сервер месенджера – система, що оброблює введені дані.

Саме за такими об’єктами була складена діаграма Use Case, що зображено на рисунку 3.3.

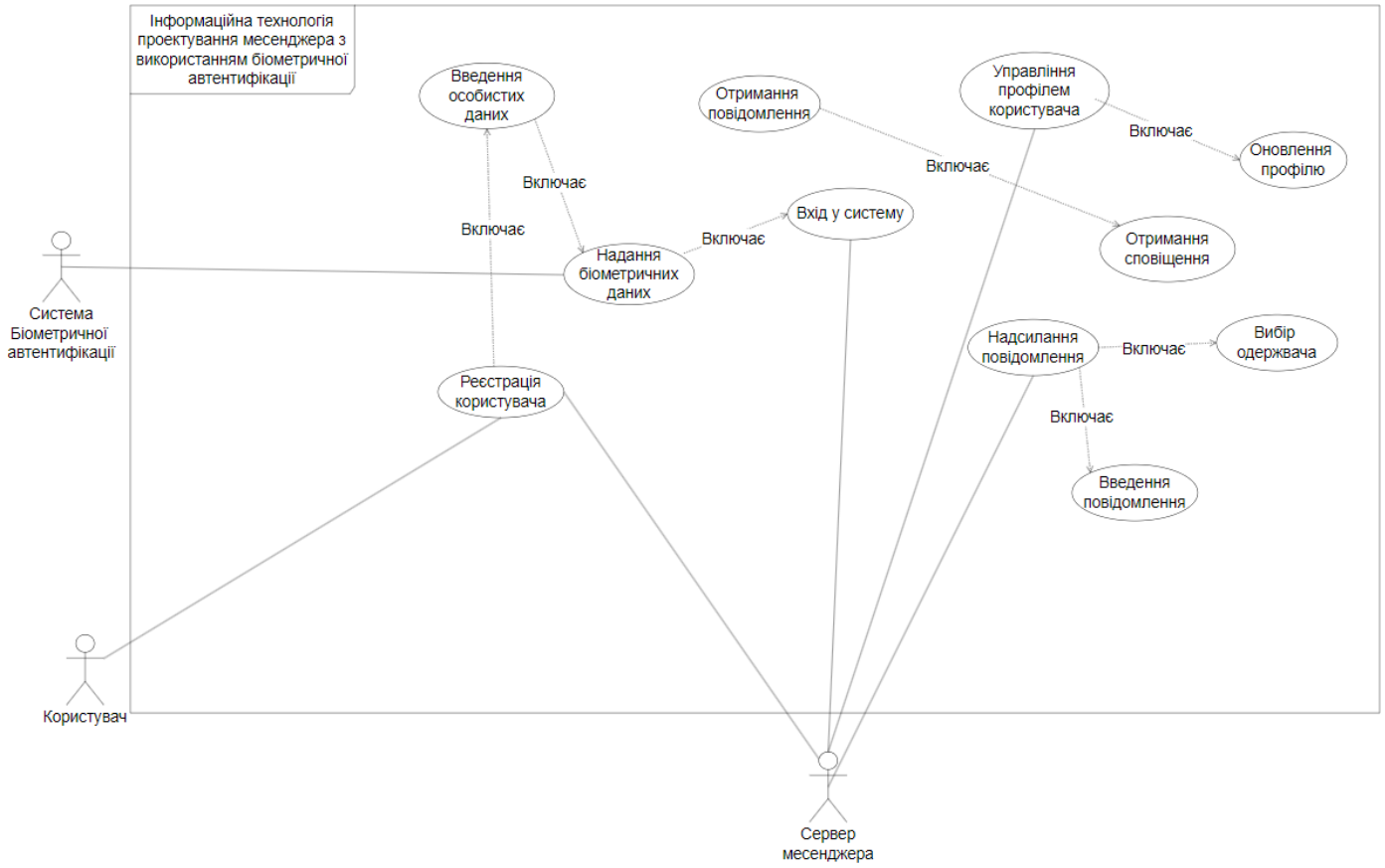

Рисунок 3.3 – Діаграма Use Case

Джерело: побудовано автором

Використання include в кожному з випадків означає, що:

- Залежність (реєстрація користувача до введення особистих даних) означає, що процес реєстрації користувача обов'язково включає введення особистих даних;
- Залежність (реєстрація користувача до надання біометричних даних) означає, що процес реєстрації користувача обов'язково включає надання біометричних даних;
- Залежність (вхід у систему надання біометричних даних) означає, що процес входу в систему обов'язково включає надання біометричних даних;
- Залежність (надсилання повідомлення до введення повідомлення) означає, що процес надсилання повідомлення обов'язково включає введення повідомлення або вибір медіа-файлу.

- Залежність (надсилання повідомлення вибір одержувача) означає, що процес надсилання повідомлення обов'язково включає вибір одержувача;
- Залежність (отримання повідомлення отримання сповіщення) означає, що процес отримання повідомлення обов'язково включає отримання сповіщення;
- Залежність (управління профілем користувача оновлення профілю) означає, що процес управління профілем обов'язково включає оновлення профілю.

3.3 Проектування моделі бази даних месенджера

Для представлення даних, що дотримуються нормативних вимог щодо даних потрібно сформувати модель даних, що будуть зберігатись в базі даних додатку. Саме дані моделі дають забезпечення умов іменування, забезпечуючи їх сумісність. Вони також полегшують процес інтеграції даних з різних джерел, забезпечуючи їх сумісність. Крім того, правильно розроблені моделі даних допомагають в оптимізації продуктивності бази даних та покращенні загальної ефективності технології.

Один з видів є логічна модель даних, яка робить опис як саме технологія буде реалізована за допомогою бази даних. Дана модель створюється розробниками. Вона слугує основою для фізичної моделі даних, яка визначає деталі зберігання даних на фізичному рівні.

Для створення логічної моделі бази даних месенджера з біометричною автентифікацією, було розглянуто основні сутності та атрибути. Рисунок 3.4 представляє собою модель бази даних вебдодатку месенджера з використанням біометричної автентифікації. Ця модель включає таблиці для зберігання даних користувачів, повідомлень та біометричних шаблонів.

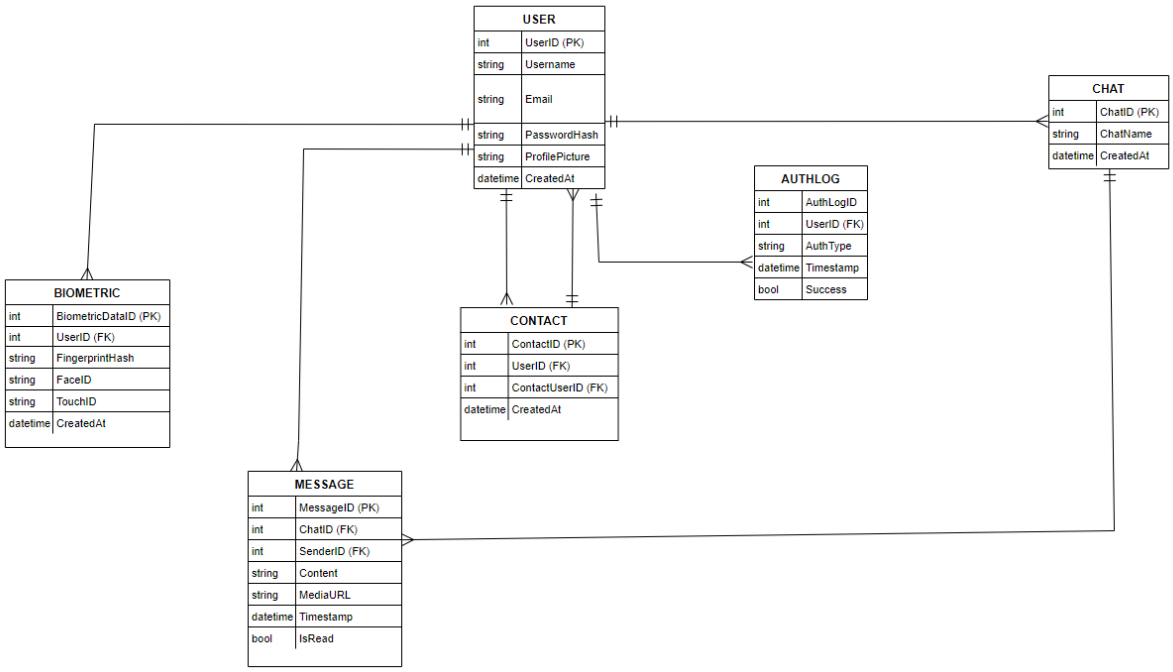

Рисунок 3.4 – Логічна модель даних

Джерело: побудовано автором

Під час моделювання були виділені наступні сутності:

- Користувачі (Users):

  UserID (PK) – унікальний ідентифікатор користувача;

  Username – ім'я користувача;

  Email – електронна пошта користувача;

  PasswordHash – хеш пароля;

  ProfilePicture – посилання на аватар користувача;

  CreatedAt – дата та час створення облікового запису.

- Біометричні дані (BiometricData):

  UserID (FK) – посилання на користувача;

FingerprintHash – хеш відбитка пальця;

FaceID – біометричні дані обличчя;

TouchID – біометричні дані відбитку пальця;

CreatedAt – дата та час створення біометричних даних.

- Повідомлення (Messages):

MessageID (PK) – унікальний ідентифікатор повідомлення;

ChatID (FK) – посилання на чат;

SenderID (FK) – посилання на відправника (користувача);

Content – текст повідомлення;

MediaURL – посилання на мультимедійний контент;

Timestamp – дата та час відправлення повідомлення;

IsRead – статус прочитання повідомлення.

- Чати (Chats):

ChatID (PK) – унікальний ідентифікатор чату;

ChatName – назва чату (для групових чатів);

CreatedAt – дата та час створення чату.

- Контакти (Contact):

ContactID (PK) – унікальний ідентифікатор контакту;

UserID (FK) – посилання на користувача;

ContactUserID (FK) – посилання на контактного користувача;

CreatedAt – дата та час додавання контакту.

- Журнали автентифікації (AuthLogs):

AuthLogID (PK) – унікальний ідентифікатор журналу автентифікації;

UserID (FK) – посилання на користувача;

AuthType – тип автентифікації (пароль, біометрія);

Timestamp – дата та час автентифікації;

Success – статус успішності автентифікації.

Зв'язками між сутностями являються ті, що користувачі можуть мати кілька біометричних даних, можуть брати участь у кількох чатах, а кожен чат може містити кілька повідомлень. Самі користувачі можуть мати кілька контактів, та кожен з них має журнали автентифікації.

4 ПРАКТИЧНА РЕАЛІЗАЦІЯ МЕСЕНДЖЕРА

4.1 Архітектура месенджера з біометричною автентифікацією

Архітектура месенджера складається з кількох основних компонентів, які взаємодіють між собою для забезпечення функціональності системи. Основні компоненти архітектури включають клієнтську частину, серверну частину, базу даних, а також інтеграцію з біометричними сервісами.

Основні компоненти архітектури:

1. Клієнтська частина (Front-end);
2. Серверна частина (Back-end);
3. База даних (Database);
4. Сервіси біометричної автентифікації (Biometric Authentication Services);

Клієнтські додатки включають вебдодаток, який використовує 1password для біометричної автентифікації.

Серверна частина (Back-end) складається з модуля, реалізованих на Python, які обробляють запити від клієнтських додатків. Вона також забезпечує авторизацію та автентифікацію користувачів.

Система безпеки включає локальне зберігання біометричних даних на пристроях користувачів, TLS шифрування для забезпечення безпечної передачі даних між клієнтом і сервером, а також End-to-End шифрування, яке гарантує, що лише відправник і отримувач можуть прочитати вміст повідомлень.

Бази даних включають нереляційні бази даних для зберігання неструктурованих даних або великих обсягів даних.

Хмарні сервіси використовуються для обробки даних і розміщення серверної частини додатка на платформах, а також для зберігання даних додатка.

Додаткові сервіси включають аналітичні інструменти для відстеження активності користувачів та аналітики використання додатка, а також системи моніторингу та логування для моніторингу стану додатка. Ці сервіси допомагають виявляти та вирішувати потенційні проблеми в реальному часі. Вони також сприяють покращенню продуктивності та стабільності додатка.

Клієнтські додатки взаємодіють із серверною частиною для відправлення та отримання повідомлень, а також для автентифікації та авторизації користувачів. Серверна частина використовує нереляційні бази даних для зберігання даних користувачів та повідомлень, а також взаємодіє з хмарними сервісами для обробки та зберігання даних. Ця архітектура дозволяє забезпечити швидкодіючу та масштабовану обробку даних. Вона також забезпечує надійність та безпеку обміну інформацією між клієнтськими додатками та серверною частиною.

Система безпеки забезпечує автентифікацію користувачів за допомогою біометричних даних, а також шифрування даних під час передачі та зберігання. Системи сповіщень надсилають push-сповіщення користувачам про нові повідомлення або події. Ця комплексна система безпеки та сповіщень забезпечує високий рівень конфіденційності та зручності взаємодії з месенджером для кожного користувача.

Дана архітектура, що зображена на рисунку 4.1 забезпечує безпечне та ефективне функціонування месенджера з біометричною автентифікацією, інтегруючи різні компоненти для забезпечення зручності та безпеки користувачів. Вона забезпечує гнучкість і легкість в розширенні функціоналу месенджера, що дозволяє відповідати зростаючим потребам користувачів та впроваджувати нові технології без перешкод для використання.

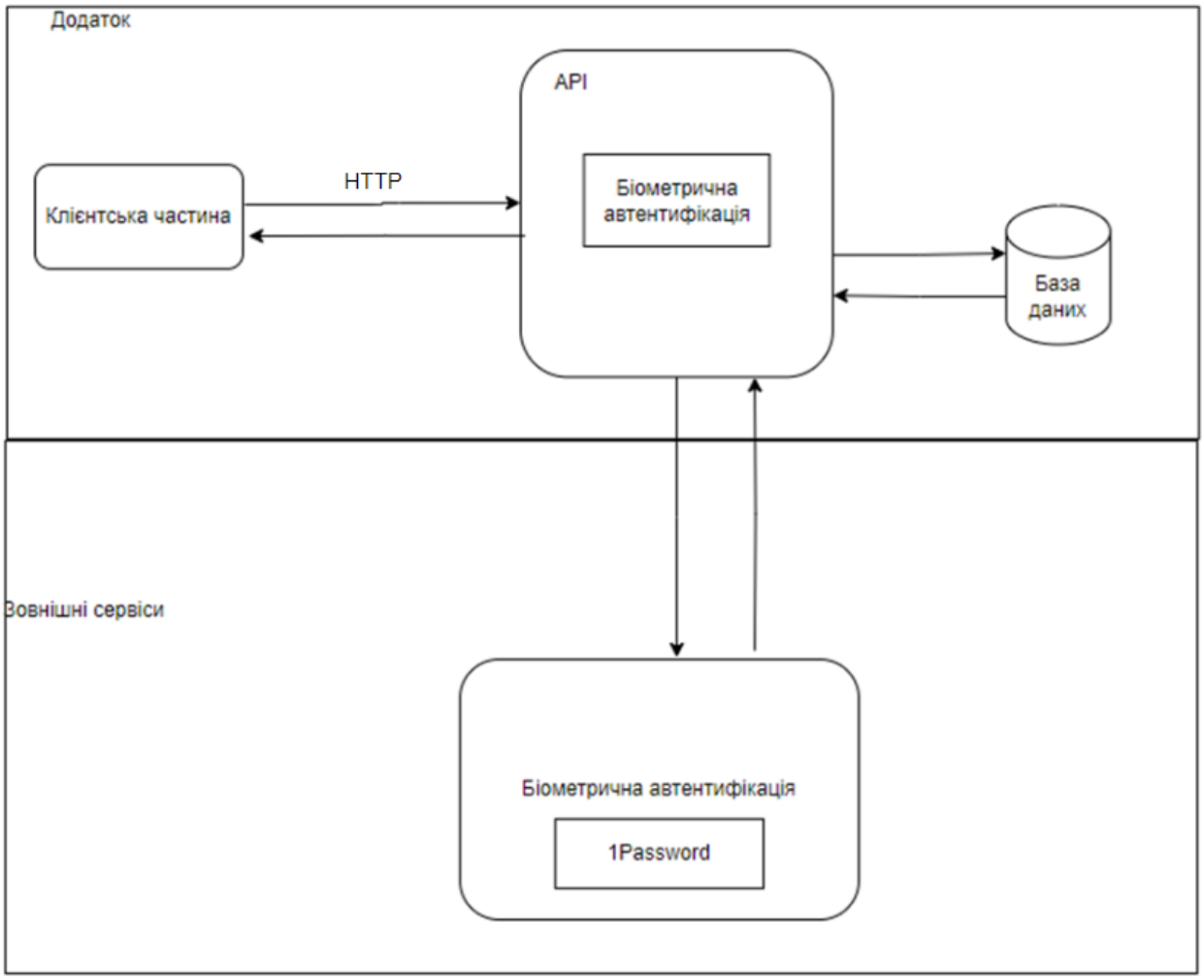

Рисунок 4.1 – Архітектура месенджера з біометричною автентифікацією

Джерело: побудовано автором

Основою для розробки стала платформа Visual Studio Code, призначена для створення вебдодатків. Це спрощений, але потужний редактор вихідного коду, який працює на комп'ютерах під керуванням Windows, macOS і Linux. Visual Studio Code підтримує JavaScript, TypeScript і Node.js, а також має розширену екосистему доповнень для інших мов (таких як C++, C#, Java, Python, PHP і Go) та середовищ виконання (наприклад, .NET і Unity). [25].

Дана інформаційна технологія складається з файлів та директорій, що зображені на рисунку 4.1

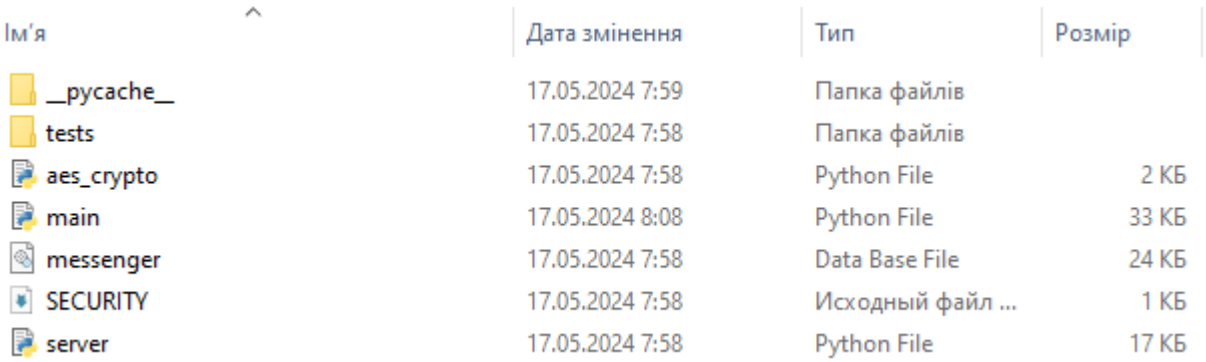

Рисунок 4.2 – Склад файлів програми

Джерело: побудовано автором (знімок з екрану)

1. **pycache**: Це папка, яка автоматично створюється Python для зберігання компільованих байт-кодів файлів Python.
2. **tests**: Це папка, містить тестові файли для проекту.
3. **aes_crypto.py**: Це файл Python, який, ймовірно, містить код для шифрування та дешифрування даних за допомогою алгоритму AES.
4. **main.py**: Це файл Python, який, містить основний код проекту.
5. **messenger.db**: Це файл бази даних, який використовується для зберігання даних для месенджера.
6. **SECURITY.md**: Це файл Markdown, який містить інформацію про безпеку проекту.
7. **server.py**: Це файл Python, який, містить код для сервера.

Під час запуску програми відкривається вікно з реєстрацією або входом користувача, в якому вносимо дані для реєстрації або входу. Рисунок входу в вебдодаток зображений на рисунку 4.2.

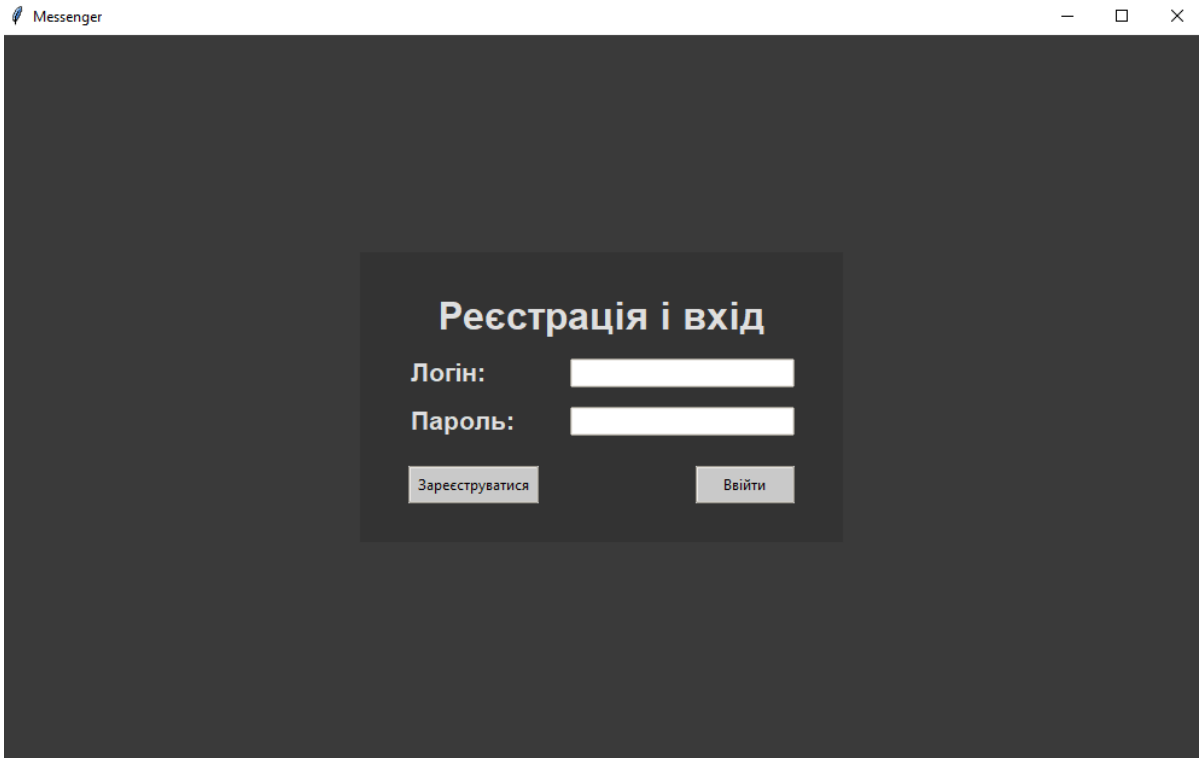

Рисунок 4.3 – Вікно реєстрації або входу користувачів

Джерело: побудовано автором (знімок з екрану)

4.2 Реалізація біометричної автентифікації

У робочому каталозі програми виконайте наведені нижче команди, щоб налаштувати віртуальне середовище Python:

python3 -m venv venv

source venv/bin/activate

Для цієї програми ви створите програму з двома сторінками, сторінкою входу та сторінкою інформаційної панелі, для доступу до яких потрібна автентифікація. Ми швидко налаштуємо базову програму. Встановіть пару ключових залежностей:

pip install flask python-dotenv requests

Після введення даних, починає працювати сервіс 1Password, який підключається до вебдодатку через команду pip install flask python-dotenv requests в терміналі Visual Studio Code. Для успішної реалізації даного методу біометричної автентифікації необхідно додати файл зі авторизованими маршрутами в програмі __init__.py з даним кодом:

```
from flask import Flask
def create_app():
  app = Flask(__name__)
  from .main import auth as auth_blueprint
  app.register_blueprint(auth_blueprint)
  from .main import main as main_blueprint
  app.register_blueprint(main_blueprint)
  return app
```

Наступний фрагмент коду повинен бути реалізований в main.py:

```
from flask import Blueprint, g, render_template, request, jsonify
main = Blueprint('main', __name__)
auth = Blueprint('auth', __name__)
@main.route('/')
def index():
  return render_template('index.html')
@auth.route('/dashboard', methods=['GET'])
def dashboard():
```

```
return render_template('dashboard.html')
```

Після даних кроків необхідно створити три файли HTML для домашньої, інформаційної панелі та неавторизованих сторінок, та додати CSS static/style.css.

Після цього надається вибір для автентифікації у випадку операційної системи Windows, а саме Windows Hello, або за допомогою телефону, відсканувавши QR-код надається доступ до біометричної автентиіфкації. На рисунку 4.5 зображений QR-код за допомогою якого можна використовувати біометричної автентифікації.

Рисунок 4.4 – Створений QR-код до доступу біометричної автентифікації

Джерело: побудовано автором (знімок з екрану)

Після сканування QR-коду на рисунку 4.6 зображено процес підключення власного смартфону до сервісу.

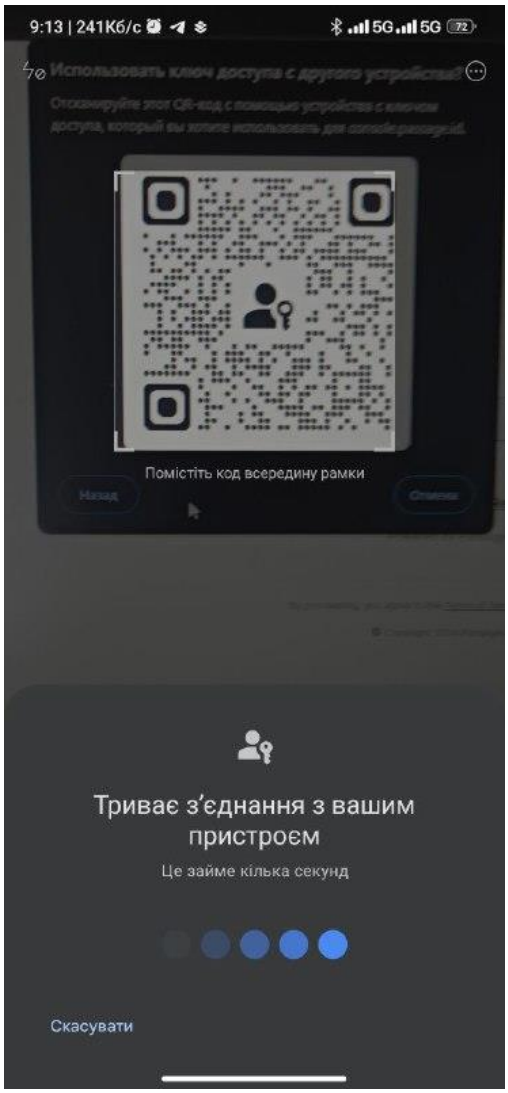

Рисунок 4.5 – Процес з'єднання смартфону зі сервісом біометричної автентифікації

Джерело: побудовано автором (знімок з екрану)

Щоб додати автентифікацію на домашню сторінку, вам потрібно буде додати елемент Passage Auth до шаблону index.html[27]. Для цього потрібно спочатку отримати свій ідентифікатор програми Passage її консолі. Потрібно створити обліковий запис та створити додаток.

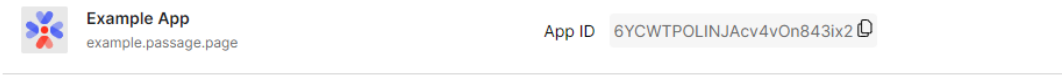

Рисунок 4.6 – Створений App ID для біометричної автентифікації

Джерело: побудовано автором (знімок з екрану)

Після застосування створеного App ID у файлі api.env, та проходження біометричної автентифікації користувача даних успішних кроків процес автентифікації вважається завершеним.

4.3 Тестування месенджеру

Щоб перевірити результати роботи інформаційної технології, необхідно здійснити вхід в нього, та надіслати повідомлення іншому користувачеві. На рисунку 4.7 зображено робочу область поточного користувача.

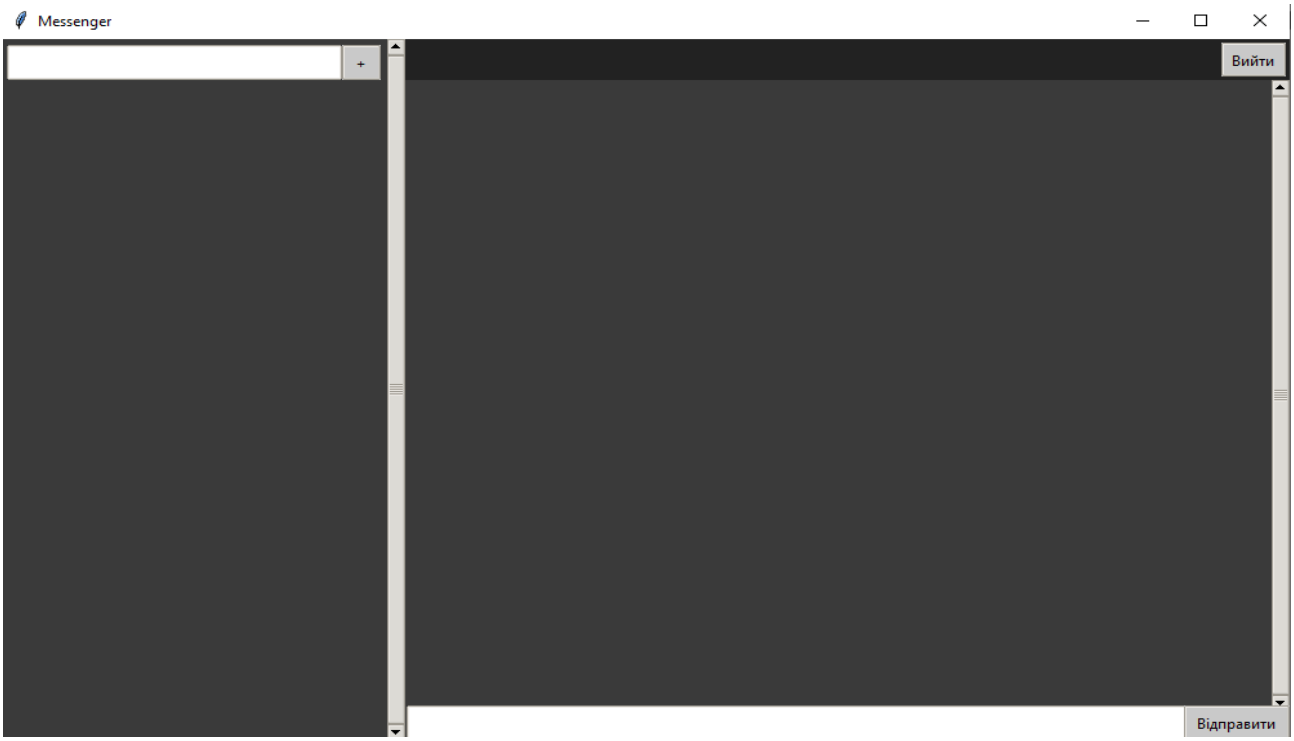

Рисунок 4.7 – Робоча область користувача

Джерело: побудовано автором (знімок з екрану)

На рисунку 4.8 зображено приклад відправки повідомлення іншому користувачу.

Рисунок 4.8 – Приклад відправки повідомлення

Джерело: побудовано автором (знімок з екрану)

На рисунку 4.9 можна побачити у лівій колонці діалоги з користувачами, а в основному блоці повідомлення.

Рисунок 4.9 – Приклад отримання повідомлення

Джерело: побудовано автором (знімок з екрану)

Після того, як користувач прочитав повідомлення, то у іншого користувача змінюється статус повідомлення на Прочитано, що зображено на рисунку 4.10.

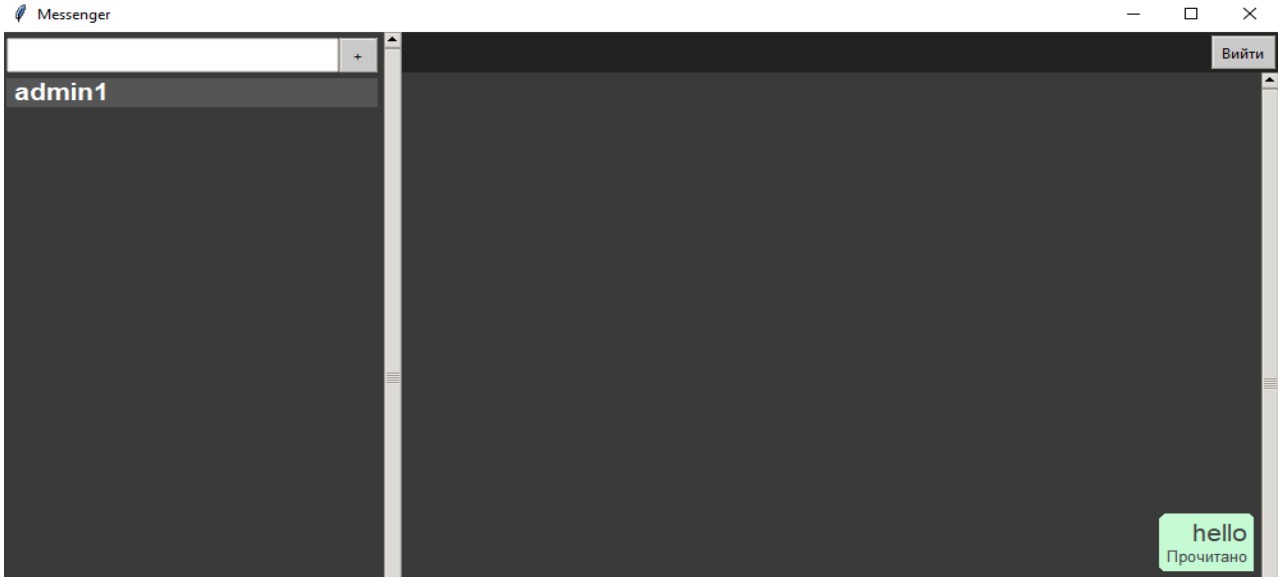

Рисунок 4.10 – Статус повідомлення користувача

Джерело: побудовано автором (знімок з екрану)

Отже, проектування месенджера з використанням біометричної автентифікації включає впровадження кількох ключових функцій для забезпечення зручності та безпеки користувачів. Користувачі можуть надсилати повідомлення іншим зареєстрованим користувачам. Якщо користувача, якому адресовано повідомлення, немає в списку зареєстрованих, система видає відповідну помилку про відсутність користувача. Впроваджено систему сповіщень про статус повідомлень, де користувач може бачити статус повідомлення: "відправлено", "доставлено" та "прочитано". Для входу в месенджер використовується біометрична автентифікація, що забезпечує високий рівень безпеки та захисту персональних даних користувачів.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи магістра було розроблено інформаційну технологію проектування месенджера з використанням біометричної автентифікації за допомогою мови Python. Для цього було проведено дослідження предметної області та проаналізовано основні типи розробки проектування месенджера та способи біометричної автентифікації.

Для інформаційної технології проектування месенджера було критично використовувати надійні та перевірені методи і алгоритми. Використання протоколів маршрутизації, таких як алгоритм Дейкстри, дозволило забезпечити оптимальну доставку повідомлень в мережі месенджера. Проаналізовані методи біометричної автентифікації в даному розділі показали ефективність кожного. На наступному етапі виконання роботи було здійснено проектування технології. У рамках проектування було розроблено контекстну IDEF0 діаграму та діаграму варіантів використання. Наведено частину логічної моделі бази даних месенджера, з таблицями, необхідними для роботи інформаційної технології. Була спроектована архітектура інформаційної технології месенджера з використанням біометричної автентифікації. Після проектування інформаційна технологія була реалізована у вигляді вебдодатку.

До заздалегідь розробленої інформаційної технології проектування месенджера було інтегровано метод біометричної автентифікації. Програмний модуль впроваджено у інформаційну технологію, та успішно виконує свою функцію.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Audio and video calls. [Електронний ресурс]. – Режим доступу: https://www.facebook.com/help/ messenger-app/1673374996287506
2. Messenger[Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/ Messenger_(software)
3. Zephyr the Roaming Dinosaur. [Електронний ресурс]. – Режим доступу: https://www.gazettedrouot.com/en/article/zephyr-the-roaming-dinosaur/38628
4. Як месенджери змінили спілкування. [Електронний ресурс]. – Режим доступу: https://www.imena.ua/blog/the-messenger-changed-humanity/
5. 10 найпопулярніших платформ соціальних мереж. [Електронний ресурс]. – Режим доступу: https://www.shopify.com/my/blog/most-popular-social-media-platforms
6. Які мобільні додатки є найбільш популярними? [Електронний ресурс]. – Режим до ступу: https://www.kiis.com.ua/?lang=ukr&cat=reports&id=1072&page=1
7. Закон України "Про захист інформації в інформаційно-телекомунікаційних системах", 1994.
8. Автентифікації, авторизація та ідентифікація: як не сплутати [Електронний ресурс.].- Режим доступу: https://training.qatestlab.com/blog/technical-articles/authentication-authorization-and-identification/
9. Taj Dini, M. (2022). СИСТЕМИ БІОМЕТРИЧНОЇ АУТЕНТИКАЦІЇ З ВИКОРИСТАННЯМ ЕЛЕКТРОЕНЦЕФАЛОГРАФІЇ. Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка», 3(15), 196–215. [Електронний ресурс.]. – Режим доступу: https://doi.org/10.28925/2663-4023.2022.15.196215
10. Захаров В. П., Рудешко В. І. Біометричні технології в XXI столітті та їх використання правоохоронними органами: посібник. – 2-ге вид., доп. / В. П. Захаров, В. І. Рудешко. – Львів: ЛьвДУВС, 2015. – 492 с.

11. Чунарьова А.В., Чунарьов А.В., Системи автентифікації в інформаційно – комунікаційних системах та мережах. [Електронний ресурс.]. – Режим доступу: https://www.rusnauka.com/29_NIOXXI_2012/Informatica/4_118674.doc.htm
12. Чунарьова А.В., Чунарьов А.В., АНАЛІЗ ІСНУЮЧИХ ШАБЛОНІВ СИСТЕМ АВТЕНТИФІКАЦІЇ В ІНФОРМАЦІЙНОКОМУНІКАЦІЙНИХ СИСТЕМАХ ТА МЕРЕЖАХ [Електронний ресурс.]. - Режим доступу: https://jrnl.nau.edu.ua/index.php/Infosecurity/article/view/3451/3426
13. Резніков А.О., АВТЕНТИФІКАЦІЯ КОРИСТУВАЧІВ НА ОСНОВІ СТІЙКОГО КЛАВІАТУРНОГО ПОЧЕРКУ. [Електронний ресурс.]. - Режим доступу: https://dspace.library.khai.edu/xmlui/bitstream/handle/123456789/1062/Reznikov.pdf?sequence=1&isAllowed=y
14. Viber Desktop App. [Електронний ресурс.]. - Режим доступу: https://dribbble.com/shots/17214182-Viber-Desktop-App
15. Web Telegram Online. [Електронний ресурс.]. - Режим доступу: https://addons.mozilla.org/uk/firefox/addon/web-telegram-online/
16. WhatsApp Chat Vector. [Електронний ресурс.]. - Режим доступу: https://www.template.net/editable/78746/whatsapp-chat-vector
17. Signal для комп’ютера. [Електронний ресурс.]. - Режим доступу: https://signal.org/uk/download/
18. Telegram, Viber, чи Signal – який месенджер найбезпечніший? [Електронний ресурс]. – Режим доступу: https://informator.ua/uk/telegram-viber-chi-signal-yakiy-mesendzher-naybezpechnishiy
19. Kleinrock, L. (1975). "Queueing Systems. Volume 1: Theory." Wiley-Interscience.
20. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). "Introduction to Algorithms." MIT Press.
21. Schneier, B. (1996). "Applied Cryptography: Protocols, Algorithms, and Source Code in C." Wiley.
22. Nielsen, J. (1993). "Usability Engineering." Academic Press.

23. Бідюк П. І., Сучасні методи біометричної автентифікації. [Електронний ресурс]. – Режим доступу: https://ela.kpi.ua/server/api/core/bitstreams/7f1251ba-7156-4730-8a08-3ae82ddbc1f3/content
24. Голубєв Г. А., Габрієлян Б. А. Сучасний стан та перспективи розвитку біометричних технологій// Нейрокомп'ютери. Розробка. Застосування. № 10, 2004, - С. 39 - 46.
25. Створення схем IDEF0. [Електронний ресурс]. – Режим доступу: https://support.microsoft.com/uk-ua/topic/%D1%81%D1%82%D0%B2%D0%BE%D1%80%D0%B5%D0%BD%D0%BD%D1%8F-%D1%81%D1%85%D0%B5%D0%BC-idef0-ea7a9289-96e0-4df8-bb26-a62ea86417fc
26. Використання розширення Visual Studio Code. [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/uk-ua/power-apps/maker/portals/vs-code-extension
27. Створення програми Python Flask із біометричною автентифікацією. [Електронний ресурс]. – Режим доступу: https://passage.1password.com/post/building-a-flask-app-with-biometric-authentication

ДОДАТОК А

ПЛАНУВАННЯ РОБІТ

А.1. Деталізація мети проекту методом SMART. Продуктом дипломного проекту є вебдодаток месенджера, що реалізує біометричну автентифікацію за допомогою клавіатурного почерку. Результати деталізації SMART - методом розміщено в табл. А.1.

Таблиця А.1 − Деталізація мети методом SMART

Specific (конкретна)	Створити інформаційну технологію месенджера та застосувати метод біометричної автентифікації для демонстрації роботи представленої технології.
Measurable (вимірювана)	Результатом роботи є розроблений месенджер, що захищений методом біометричної автентифікації за клавіатурним почерком.
Achievable (досяжна)	Для виконання проекту наявні потрібні знання технологій оцінки ефективності, знання HTML, CSS, мови програмування Python, баз даних MongoDB та навичок написання документації. Враховуючи доступні ресурсні можливості та обмеження мета є такою, яку можливо досягти.
Relevant (реалістична)	Дана технологія дозволить захистити дані користувачів месенджера від несанкціонованого входу в систему та запобігатиме до доступу персональних даних.
Time-framed (обмежена у часі)	Розробити інформаційну технологію месенджера з використанням біометричної автентифікації на основі сформованого календарного плану проекту.

А.2. Планування змісту структури робіт IT-проекту (WBS). Структура поділу робіт (WBS) є фундаментальним інструментом управління проектами, що дозволяє систематично розбивати проект на керовані компоненти. Він працює за ієрархічною моделлю, починаючи від загальних цілей і поступово розбиваючи їх на більш дрібні, більш визначені завдання. Цей розподіл включає фази, результати та робочі пакети, кожен з яких сприяє досягненню кінцевої мети проекту.

Добре розроблена структура поділу виконує кілька важливих функцій:

- надає дорожню карту для планування та контролю проекту, що дозволяє командам ефективно розподіляти ресурси та контролювати прогрес.
- полегшує оцінку витрат і бюджетування шляхом агрегування витрат на різних рівнях ієрархії. Крім того, це допомагає у розподілі завдань і роз'ясненні обов'язків, гарантуючи, що кожен член команди розуміє свою роль у досягненні цілей проекту.

Крім того, WBS служить основою для створення описів завдань, технічних специфікацій і звітів про хід. Це допомагає узгодити діяльність проекту з бажаними результатами, зосереджуючись на результатах, а не на окремих завданнях. Такий підхід сприяє чіткості та ефективності виконання проекту.

Структура поділу робіт відіграє вирішальну роль в управлінні проектами, організовуючи складні проекти в керовані частини, полегшуючи планування та контроль, а також забезпечуючи узгодження з цілями та завданнями проекту.

WBS діаграма проекту зображена на рисунку А.1.

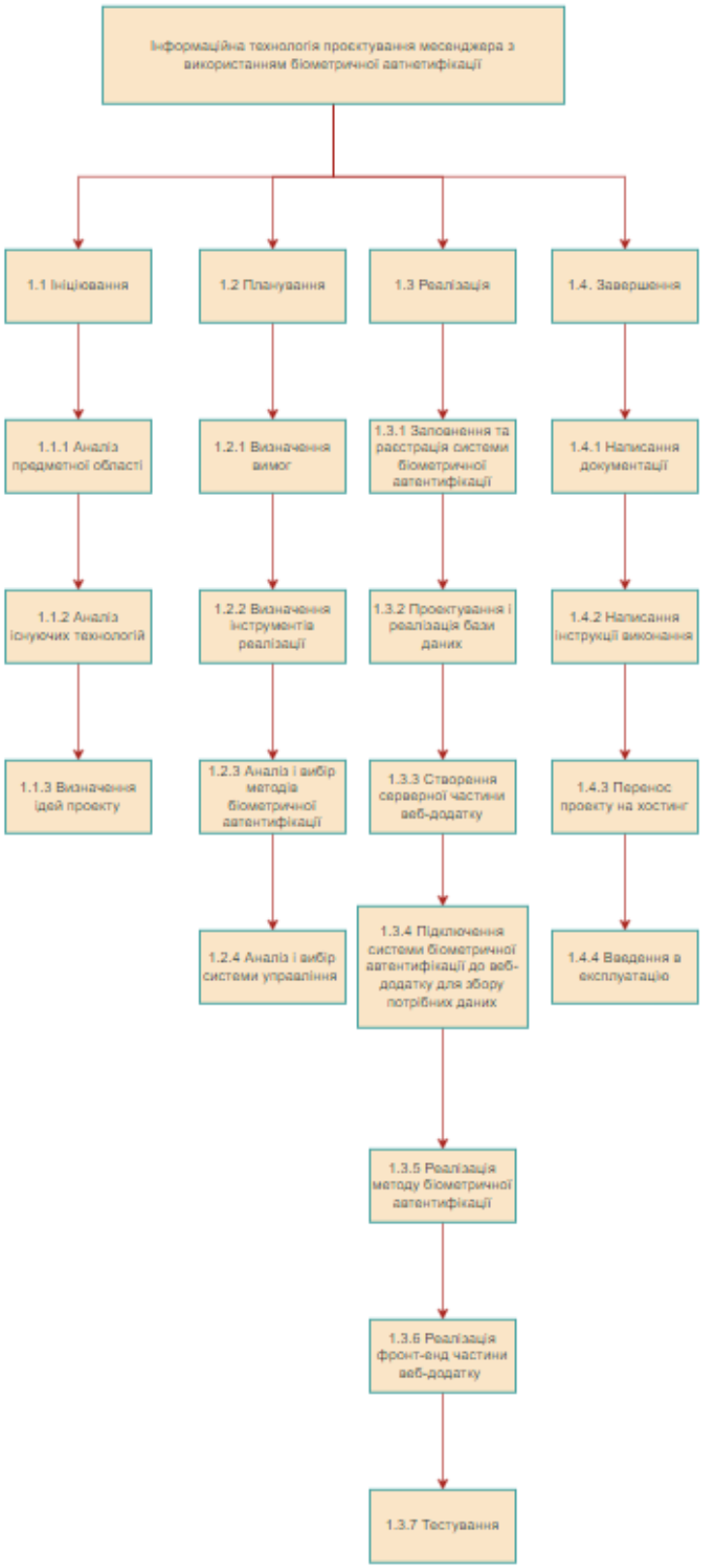

Рисунок А.1 - WBS структура проекту

А.3 Планування структури організації

Виконавці проекту:

- Менеджер проекту (Керівник дипломної роботи)
- Виконавець проекту (Студент)

На основі даної структури була складена таблиця А 2.

Таблиця А 2 – Виконавці проекту

Роль	Ім’я	Проектна роль
Розробник -тестувальник	Нагорний Є.М.	Створює продукт проекту. Перевіряє функціональні вимоги проекту.
Менеджер проекту (дипломної роботи)	Нагорний В.В.	Відповідає за виконання термінів, підтримує розробника – тестувальника проекту в питаннях виконання продукту проекту.

А.4. Організаційна структура проекту (OBS). Структура розподілу організацій (OBS) —інструмент управління проектом, що ілюструє ієрархічну структуру організації, яка бере участь в проекті, тим самим спрощує складну структуру організації, класифікуючи її на окремі рівні та групуючи пов’язані функціональні області зі спільними цілями. Це допомагає керівникам проектів отримати уявлення про організаційну структуру та визначити осіб, відповідальних за кожен аспект проекту. Окрім цього, OBS полегшує розподіл витрат і управління між різними рівнями організації. Він представлений схематично, як показано на рисунку А.2. Використання

OBS сприяє ефективній комунікації та співпраці між різними відділами та командами, тим самим покращуючи виконання проекту та потенційно знижуючи витрати.

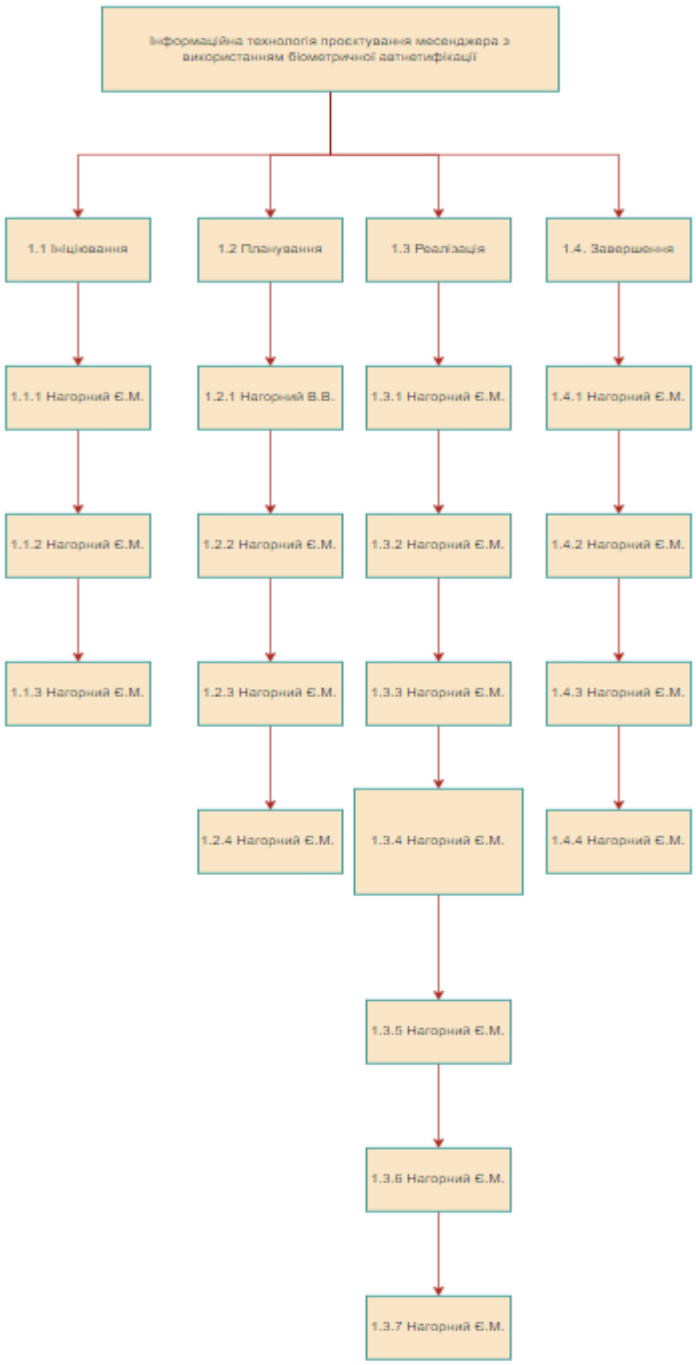

Рисунок А.2. – OBS структура проекту

А.5. Побудова календарного графіка виконання ІТ-проекту. Щоб мати реальне уявлення про тривалість виконання робіт зі врахуванням обмеженості у використанні

ресурсів, на підставі часткових мережевих моделей, та проекту в цілому з урахуванням вихідних і святкових днів, будують календарний графік робіт. Він є реальним розподілом робіт по пакету по календарними датами, тобто своєрідним розкладом виконання робіт. Діаграма Ганта в даному випадку є оптимальним варіантом вирішення даного питання. Діаграма Ганта представляє собою відрізки, що розміщені на шкалі часу, що знаходиться горизонтально. Кожен відрізок відповідає окремому завданню або задачі. Завдання і задачі, як складова, розміщуються вертикально. Початок, кінець та довжина відрізків на шкалі часу відповідають початку, кінцю та тривалості завдання. На рисунку А.3. представлена діаграма Ганта стосовно роботи.

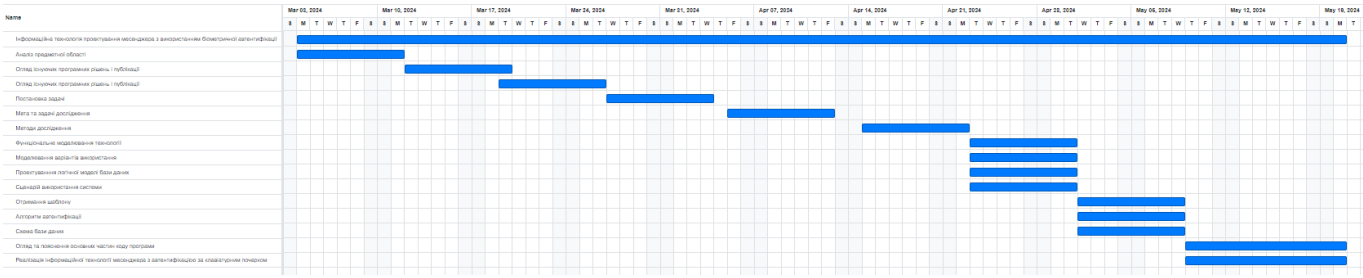

Рисунок А.3. – Діаграма Ганта

А.6. Управління ризиками. Під час реалізації проекту можуть виникнути небажані умови, ситуації та непередбачувані наслідки, які називаються ризиками. Управління ризиком полягає у відповіді на ці події під час виконання проекту шляхом моніторингу та контролю за ними. Ефективне управління ризиками включає виявлення потенційних загроз на ранніх етапах і розробку стратегій для їх мінімізації або усунення. Це дозволяє команді проекту бути підготовленою до непередбачених обставин і забезпечує успішне досягнення цілей проекту, знижуючи можливі негативні впливи на його виконання.

Для врахування ймовірності появи ризикових подій, які можуть мати негативні наслідки, можна створити таблицю класифікації ризиків. Потім на основі цієї таблиці створюється матриця ризиків. На таблиці можна побачити таблицю А.3, в якій написані ризики, виникнення і втрати.

Таблиця А. 3 - Ймовірність виникнення і величина ризику

№	Ризики	Виникнення	Втрати
1	Відсутність досвіду	6	4
2	Зміна строків виконання роботи	4	6
3	Неправильний матеріал для підключення біометричної автентифікації	2	5
4	Не чітко визначені задачі та підзадачі проекту	4	7
5	Зростання вимог до проекту	1	4

Таблиця А.4 – Матриця впливу

Вірогідність виникнення	Оцінка				
5			3	5	
4		4			
3		6	2		
2				1	
1					
Ступінь впливу	1	2	3	4	5

ДОДАТОК Б

Фрагменти коду реалізації інформаційної технології.

Main.py:

```
import tkinter as tk

from json import dumps
from json import loads
from math import ceil
from math import floor
from os import path as opath
from os import system
from socket import AF_INET
from socket import SOCK_DGRAM
from socket import socket
from threading import Thread
from time import sleep
from tkinter import ttk

from aes_crypto import acrypt

KEY_EXTRA = "LX@$wmd3l8Yt9zxj9WH8yp@DOzNrDk2^flJzzNU!%oYy3EUoXabyGF~k%5TiJBH*"
KEY_LOGIN_FILE =
"rW9M8%KphnA*Jt1rCNG*8ANo51$h*8TRI&c@mPnD)8$*EMUzxq0Kr(B5M~qd\
C)QM"


def absolute(path):

    return opath.join(opath.dirname(opath.realpath(__file__)), path)


class Window:


    def __init__(self, tk_window: tk.Tk) -> None:

        self.tk_window: tk.Tk = tk_window
        self.elements: dict = {}

    def place(self, id_: str, element, *args, **kwargs) -> None:

        if id_ in self.elements:
            raise ValueError
```

```
        self.elements[id_] = element
        element.place(*args, **kwargs)

    def pack(self, id_: str, element, *args, **kwargs) -> None:

        if id_ in self.elements:
            raise ValueError

        self.elements[id_] = element
        element.pack(*args, **kwargs)

    def clear(self) -> None:

        for child in self.tk_window.winfo_children():
            child.destroy()

        self.elements = {}

    def __getattribute__(self, id_: str):

        if id_ in ["pack", "place", "tk_window", "elements", "clear"]:
            return object.__getattribute__(self, id_)

        return object.__getattribute__(self, "elements")[id_]


class MessengerClient:


    MAIN_BACKGROUND = "#3a3a3a"
    SELECT_FOREGROUND = "#555"
    MAIN_FOREGROUND = "#dfdfdf"
    SECOND_BACKGROUND = "#c9c9c9"
    THIRD_BACKGROUND = "#bfbfbf"
    MESSAGE_BACK_COLOR = "#c5fad5"
    MESSAGE_FORE_COLOR = "#444"
    MESSAGE_BACK_COLOR2 = "#eee"
    MESSAGE_FORE_COLOR2 = "#444"
    FRAME_BG_COLOR = "#333"
    PANEL_BACKGROUND = "#222"
    _RECEIVE_SLEEP_TIME = 1 / 60
    _IDLE_SLEEP_TIME = 1 / 3

    def __init__(self) -> None:

        self.root = None
```

```
        self.win: Window = Window(self.root)
        self._sock: socket = socket(AF_INET, SOCK_DGRAM)
        self._sock.connect(("127.0.0.1", 7505))
        self.__sended: list = []
        self.__received: list = []
        self._logins: dict = {}
        self._userid_selected: int = -1
        self.last_height: int = -1
        self._is_on_main_tab: bool = False
        self.__temp_messages: list = []
        self.__key = None
        self.__aes = None
        self.destroyed = False
        self.__queued_requests = []
        self.__last_login = None
        self.__last_password = None

        Thread(target=self.receive, daemon=True).start()
        Thread(target=self.send_idle, daemon=True).start()

        self.main()

        if not self.destroyed:
            self._sock.send(b"\x05\x03\xff\x01")

    @staticmethod
    def show_error(title: str, message: str) -> None:

        title = f"Помилка: {title}"
        ltitle = len(title)
        lmessage = len(message)
        mlen = max(ltitle, lmessage)
        dtitle = (mlen - ltitle) / 2
        dmessage = (mlen - lmessage) / 2
        print("-" * (mlen + 4))
        print(f"|{' ' * (floor(dtitle) + 1)}{title}\
{' ' * (ceil(dtitle) + 1)}|")
        print("-" * (mlen + 4))
        print(f"|{' ' * (floor(dmessage) + 1)}{message}\
{' ' * (ceil(dmessage) + 1)}|")
        print("-" * (mlen + 4))

    def __encode_message(self, message) -> bytes:

        if self.__key is None and not self.destroyed:
            self._sock.send(b"\x05\x03\xff\x01")
            self.__queued_requests.append(message)
```

```
            return None

        return self.__aes.encrypt(dumps(
            message,
            separators=(",", ":"),
            ensure_ascii=False
        ))

    def __decode_message(self, message: bytes):

        if self.__key is None:
            self.__key = message.decode("ascii")
            self.__aes = acrypt(KEY_EXTRA + self.__key)

            for req in self.__queued_requests:
                self.send(req)

            return False

        return loads(self.__aes.decrypt(message))

    def send(self, message) -> None:

        msg = self.__encode_message(message)

        if msg is not None:
            self._sock.send(msg)

    def send_register(self, login, password):

        self.__last_login = login
        self.__last_password = password
        self.send(["register", login, password])

    def send_login(self, login, password):

        self.__last_login = login
        self.__last_password = password
        self.send(["login", login, password])

    def remember_login(self):

        if self.__last_login is None or self.__last_password is None:
            return False

        login = str(self.__last_login)
        password = str(self.__last_password)
```

```
    with open(absolute(".logindata"), "wb") as lfile:
        lfile.write(
            acrypt(KEY_LOGIN_FILE).encrypt("\n".join([login, password]))
        )

    return True

def restore_login(self):

    if self.__last_login is not None and self.__last_password is not None:
        return [self.__last_login, self.__last_password]

    try:
        with open(absolute(".logindata"), "rb") as lfile:
            data = acrypt(KEY_LOGIN_FILE).decrypt(lfile.read()).split("\n")
    except FileNotFoundError:
        return None

    return data

def forget_login(self):

    system(f'rm {absolute(".logindata")}')
    self.__aes = None
    self.__key = None
    self._is_on_main_tab = False
    self._logins = {}
    self.__sended = []
    self.__received = []
    self.login_tab()

@staticmethod
def create_round_rectangle(
    cnv,
    px1,
    py1,
    px2,
    py2,
    radius,
    ign1=False,
    ign2=False,
    **kwargs
) -> int:

    points = [
        px1 + radius, py1,
```

```
            px1 + radius, py1,
            px2 - radius, py1,
            px2 - radius, py1,
            px2, py1,
            px2, py1 + radius,
            px2, py1 + radius,
            px2, py2 - radius,
            px2, py2 - radius,
            *((px2, py2, px2, py2) if ign1 else (px2, py2)),
            px2 - radius, py2,
            px2 - radius, py2,
            px1 + radius, py2,
            px1 + radius, py2,
            *((px1, py2, px1, py2) if ign2 else (px1, py2)),
            px1, py2 - radius,
            px1, py2 - radius,
            px1, py1 + radius,
            px1, py1 + radius,
            px1, py1
        ]

        return cnv.create_polygon(points, **kwargs, smooth=True)

    def user_selected(self) -> None:

        try:
            listbox = self.win.userlist
        except KeyError:
            return

        sel = listbox.curselection()

        if len(sel) == 0:
            return

        inv_logins = {val: key for key, val in self._logins.items()}
        user_id = int(inv_logins[listbox.get(sel[0])[1:]])

        if user_id == self._userid_selected:
            return

        self._userid_selected = user_id

        messages = []

        for smsg in self.__sended:
            if smsg[3] != self._userid_selected:
```

```
            continue

        messages.append(smsg)

    for rmsg in self.__received:
        if rmsg[1] != self._userid_selected:
            continue

        messages.append(rmsg)

    messages.sort(key=lambda message: message[0])

    cnv = self.win.messages
    cwh = cnv.winfo_width()
    chg = cnv.winfo_height()

    cnv.delete("all")

    offset = chg

    for msg in self.__temp_messages[::-1]:
        if msg[1] != self._userid_selected:
            continue

        text = cnv.create_text(
            cwh - 5,
            offset,
            text=msg[0],
            anchor=tk.NE,
            fill=self.MESSAGE_FORE_COLOR,
            font="Arial 16",
            width=cwh - 20
        )
        bbox_text = cnv.bbox(text)

        diff = bbox_text[1] - bbox_text[3] - 20
        cnv.move(text, 0, diff)
        text2 = cnv.create_text(
            cwh - 5,
            cnv.bbox(text)[3],
            text="Відправлено",
            anchor=tk.NE,
            fill=self.MESSAGE_FORE_COLOR,
            font="Arial 10",
            width=cwh - 20
        )
        bbox_text = cnv.bbox(text)
```

```
        bbox_text2 = cnv.bbox(text2)

        text_bbox = [
            min(bbox_text[0], bbox_text2[0]),
            bbox_text[1],
            bbox_text[2],
            bbox_text2[3]
        ]

        rect = self.create_round_rectangle(
            cnv,
            text_bbox[0] - 5,
            text_bbox[1] - 5,
            text_bbox[2] + 5,
            text_bbox[3] + 5,
            15,
            fill=self.MESSAGE_BACK_COLOR,
            width=0,
            ign1=True
        )
        cnv.tag_lower(rect)

        offset += diff

    for msg in messages[::-1]:
        sended = msg[3] == self._userid_selected

        text = cnv.create_text(
            cwh - 5 if sended else 5,
            offset,
            text=msg[2],
            anchor=tk.NE if sended else tk.NW,
            fill=self.MESSAGE_FORE_COLOR if sended else self.
            MESSAGE_FORE_COLOR2,
            font="Arial 16",
            width=cwh - 20
        )
        bbox_text = cnv.bbox(text)

        diff = bbox_text[1] - bbox_text[3] - 20

        if sended:
            text2 = cnv.create_text(
                cwh - 5,
                cnv.bbox(text)[3],
                text=["Доставлено", "Отримано", "Прочитано"][msg[4]],
                anchor=tk.NE,
```

```
                fill=self.MESSAGE_FORE_COLOR,
                font="Arial 10",
                width=cwh - 20
            )
            bbox = cnv.bbox(text2)
            diff += bbox[1] - bbox[3]

        cnv.move(text, 0, diff)

        if sended:
            cnv.move(text2, 0, diff)

        if sended:
            bbox_text = cnv.bbox(text)
            bbox_text2 = cnv.bbox(text2)

            text_bbox = [
                min(bbox_text[0], bbox_text2[0]),
                bbox_text[1],
                bbox_text[2],
                bbox_text2[3]
            ]
        else:
            text_bbox = cnv.bbox(text)

        rect = self.create_round_rectangle(
            cnv,
            text_bbox[0] - 5,
            text_bbox[1] - 5,
            text_bbox[2] + 5,
            text_bbox[3] + 5,
            15,
            fill=self.MESSAGE_BACK_COLOR if sended else self.
            MESSAGE_BACK_COLOR2,
            width=0,
            ign1=sended,
            ign2=not sended
        )
        cnv.tag_lower(rect)

        offset += diff

    cnv.configure(scrollregion=cnv.bbox("all"))

def resize(self, event) -> None:

    if event.height == self.last_height:
```

```
            return

        self.last_height = event.height

        try:

            listbox = self.win.userlist

            if len(listbox.curselection()) > 0:
                self._userid_selected = -1
                listbox.event_generate("<<ListboxSelect>>")
        except KeyError:
            pass

    def send_message(self, message: str) -> None:

        if self._userid_selected == -1:
            return

        message = message[:65535]

        self.win.messages_input.delete(0, tk.END)
        self.__temp_messages.append([message, self._userid_selected])

        listbox = self.win.userlist

        if self._userid_selected != -1:
            listbox.select_set(
                list(
                    self._logins.keys()
                ).index(str(self._userid_selected))
            )
            self._userid_selected = -1
        else:
            listbox.select_set(0)

        listbox.event_generate("<<ListboxSelect>>")

        self.send(["send_message", message, self._userid_selected])

    def add_user(self, username: str) -> None:

        for uid, username2 in enumerate(self._logins.values()):
            if username == username2:
                listbox = self.win.userlist
                listbox.select_clear(0, tk.END)
                listbox.select_set(uid)
```

```
            listbox.event_generate("<<ListboxSelect>>")
            return

        self.send(["find_user", username])

    def receive(self) -> None:

        while True:
            try:
                jdata = self._sock.recv(70000)
            except ConnectionResetError:
                self.login_tab()
                self.show_error(
                    "Сервер відключений",
                    "Сервер примусово розірвав підключення"
                )

            data = self.__decode_message(jdata)

            if data is False:
                continue

            com = data[0]

            if com == "register_status":
                status = data[1]

                if status == 0:
                    self.remember_login()
                    self.send(["get_account_data"])
                elif status == 1:
                    self.show_error(
                        "Невірний логін",
                        "Логін занадто короткий"
                    )
                elif status == 2:
                    self.show_error(
                        "Невірний логін",
                        "Логін занадто короткий"
                    )
                elif status == 3:
                    self.show_error(
                        "Невірний пароль",
                        "Пароль слишком короткий"
                    )
                elif status == 4:
                    self.show_error(
```

```
            "Невірний логін",
            "Акаунт з вказаний логіном уже існує"
        )
    elif com == "login_status":
        status = data[1]

        if status == 0:
            self.remember_login()
            self.send(["get_account_data"])
        elif status == 1:
            self.show_error(
                "Невірний логін",
                "Логін занадто короткий"
            )
        elif status == 2:
            self.show_error(
                "Невірний логін",
                "Логін занадто довгий"
            )
        elif status == 3:
            self.show_error(
                "Невірний пароль",
                "Пароль занадто короткий"
            )
        elif status == 4:
            self.show_error(
                "Неверный логин",
                "Акаунт з вказаний логіном не існує"
            )
        elif status == 5:
            self.show_error(
                "Невірний пароль",
                "Пароль від акаунту не підходить"
            )

        if status != 0:
            self.login_tab()
    elif com == "account_data":
        adata = data[1]
        self.__sended = adata[0]
        self.__received = adata[1]
        self._logins = adata[2]

        main_tab = self._is_on_main_tab

        self._is_on_main_tab = True
```

```
if not main_tab:
    for element in self.win.elements.values():
        element.destroy()

    frame = tk.Frame(background=self.MAIN_BACKGROUND)
    self.win.place(
        "messages_frame",
        frame,
        relx=0.3,
        relw=0.7,
        relh=1
    )
    cnv = tk.Canvas(
        frame,
        background=self.MAIN_BACKGROUND,
        bd=0,
        highlightthickness=0
    )
    top_panel = ttk.Frame(frame, style="Panel.TFrame")
    self.win.place(
        "top_panel",
        top_panel,
        h=35,
        relw=1

    )
    self.win.place(
        "top_panel_logout",
        ttk.Button(
            top_panel,
            text="Вийти",
            command=self.forget_login
        ),
        anchor=tk.NE,
        relx=1,
        x=-3,
        w=50,
        h=29,
        y=3
    )
    cnv_sbar = ttk.Scrollbar(frame)
    self.win.place(
        "messages_scrollbar",
        cnv_sbar,
        y=35,
        relx=1,
        anchor=tk.NE,
```

```
    relh=1,
    h=-60
)
cnv_sbar.configure(command=cnv.yview)
cnv.configure(yscrollcommand=cnv_sbar.set)
self.win.place(
    "messages",
    cnv,
    relw=1,
    relh=1,
    x=20,
    y=35,
    w=-40,
    h=-5
)
msg_input = ttk.Entry(font="Arial 16")
self.win.place(
    "messages_input",
    msg_input,
    x=15,
    relx=0.3,
    rely=1,
    relw=0.7,
    anchor=tk.SW,
    w=-95,
    h=30
)
msg_input.bind("<Return>", lambda _: self.send_message(
    self.win.messages_input.get()
))
self.win.place(
    "message_send_btn",
    ttk.Button(
        text="Відправити",
        command=lambda: self.send_message(
            self.win.messages_input.get()
        )
    ),
    relx=1,
    rely=1,
    anchor=tk.SE,
    h=30
)

listbox = tk.Listbox(
    bg=self.MAIN_BACKGROUND,
    bd=0,
```

```
    font="Arial 16 bold",
    fg=self.MAIN_FOREGROUND,
    selectbackground=self.SELECT_FOREGROUND,
    selectmode=tk.SINGLE,
    activestyle=tk.NONE,
    highlightthickness=0
)
listbox.bind(
    "<<ListboxSelect>>",
    lambda _: self.user_selected()
)
self.root.bind("<Configure>", self.resize)
self.win.place(
    "add_user_name",
    ttk.Entry(font="Arial 16"),
    x=5,
    y=5,
    h=30,
    relw=0.3,
    w=-40,
)
self.win.place(
    "add_user_button",
    ttk.Button(text="+", command=lambda:
                    self.add_user(
                        self.win.add_user_name.get())
                ),
    relx=0.3,
    x=-35,
    w=30,
    y=5,
    h=30
)
self.win.place(
    "userlist",
    listbox,
    relw=0.3,
    relh=1,
    anchor=tk.NW,
    x=5,
    y=40,
    w=-10,
    h=-45
)
scrollbar = ttk.Scrollbar(command=listbox.yview)
self.win.place(
    "userlist_scrollbar",
```

```
            scrollbar,
            relx=0.3,
            anchor=tk.NW,
            relh=1
        )
        listbox.config(yscrollcommand=scrollbar.set)
    else:
        listbox = self.win.userlist
        listbox.delete(0, tk.END)

        for i, temp_msg in enumerate(self.__temp_messages):
            if temp_msg[0] == self.__sended[-1][2]:
                self.__temp_messages = self.__temp_messages[i + 1:]
                break

    for user in self._logins.values():
        listbox.insert(tk.END, f" {user}")

    if self._userid_selected != -1:
        listbox.select_set(
            list(
                self._logins.keys()
            ).index(str(self._userid_selected))
        )
        self._userid_selected = -1
    else:
        listbox.select_set(0)

    listbox.event_generate("<<ListboxSelect>>")
elif com == "find_user_result":
    self.win.add_user_name.delete(0, tk.END)

    if data[1] is False:
        self.show_error(
            "Не знайдено",
            "Не вдалось знайти користувача"
        )
    else:
        arr = data[1]

        self._logins[str(arr[0])] = arr[1]
        listbox = self.win.userlist
        listbox.insert(tk.END, f" {arr[1]}")
        listbox.select_clear(0)
        listbox.select_set(len(self._logins) - 1)
        listbox.event_generate("<<ListboxSelect>>")
elif com == "not_logged":
```

```
            if self._userid_selected == -1:
                self.show_error(
                    "Потрібен вхід",
                    "Потрібен вхід в акаунт"
                )
            else:
                self.login_tab()
                self.show_error(
                    "Ви вишли з акаунту",
                    "Потрібно знову зайти в акаунт для виконання даної \
операції"
                )

        sleep(self._RECEIVE_SLEEP_TIME)

    def send_idle(self) -> None:
        """Отправляет сообщение серверу о том, что клиент до сих пор открыт."""
        while True:
            if self.__key is not None and self._is_on_main_tab:
                self.send(["client_alive"])

            sleep(self._IDLE_SLEEP_TIME)

    def login_tab(self, clear=True) -> None:

        self.__sended = []
        self.__received = []
        self._logins = {}
        self._userid_selected = -1
        self._is_on_main_tab = False
        self.__temp_messages = []
        self.__key = None
        self.__aes = None

        if clear:
            self.win.clear()

        self.win.place(
            "loadscreen_background",
            ttk.Frame(),
            relx=0.5,
            rely=0.5,
            w=400,
            h=240,
            anchor=tk.CENTER,
        )
```

```
self.win.place(
    "loadscreen_registration",
    ttk.Label(
        text="Реєстрація і вхід",
        font="Arial 24 bold",
        background=self.FRAME_BG_COLOR
    ),
    relx=0.5,
    rely=0.5,
    anchor=tk.N,
    y=-88
)
self.win.place(
    "loadscreen_registration_login",
    ttk.Label(
        text="Логін:",
        font="Arial 16 bold",
        background=self.FRAME_BG_COLOR
    ),
    relx=0.5,
    rely=0.5,
    x=-160,
    y=-20,
    anchor=tk.W
)
self.win.place(
    "loadscreen_registration_password",
    ttk.Label(
        text="Пароль:",
        font="Arial 16 bold",
        background=self.FRAME_BG_COLOR
    ),
    relx=0.5,
    rely=0.5,
    x=-160,
    y=20,
    anchor=tk.W
)
self.win.place(
    "loadscreen_registration_login_field",
    ttk.Entry(font="Arial 12"),
    relx=0.5,
    rely=0.5,
    x=160,
    y=-20,
    anchor=tk.E
)
```

```
    self.win.place(
        "loadscreen_registration_password_field",
        ttk.Entry(font="Arial 12", show="•"),
        relx=0.5,
        rely=0.5,
        x=160,
        y=20,
        anchor=tk.E
    )
    self.win.place(
        "loadscreen_registration_button",
        ttk.Button(
            text="Зареєструватися",
            command=lambda: self.send_register(
                self.win.loadscreen_registration_login_field.get(),
                self.win.loadscreen_registration_password_field.get()
            )
        ),
        relx=0.5,
        rely=0.5,
        anchor=tk.SW,
        y=88,
        x=-160
    )
    self.win.place(
        "loadscreen_login_button",
        ttk.Button(
            text="Ввійти",
            command=lambda: self.send_login(
                self.win.loadscreen_registration_login_field.get(),
                self.win.loadscreen_registration_password_field.get()
            )
        ),
        relx=0.5,
        rely=0.5,
        anchor=tk.SE,
        y=88,
        x=160
    )

def on_destroy(self):

    if not self.destroyed and self.__key is not None:
        self.send(["disconnect"])

    self.destroyed = True
    self.root.destroy()
```

```
def main(self):

    self.root = tk.Tk()
    self.root.wm_title("Messenger")
    self.root.wm_geometry("1000x600")
    self.root.minsize(500, 340)
    self.win = Window(self.root)

    self.last_height = self.root.winfo_height()

    style = ttk.Style(self.root)
    style.theme_use("clam")
    self.root.configure(bg=self.MAIN_BACKGROUND)
    style.configure(
        "TLabel",
        background=self.MAIN_BACKGROUND,
        foreground=self.MAIN_FOREGROUND
    )
    style.configure(
        "TEntry",
        background=self.MAIN_BACKGROUND
    )
    style.configure(
        "TButton",
        background=self.SECOND_BACKGROUND,
        activebackground=self.THIRD_BACKGROUND
    )
    style.configure(
        "TFrame",
        background=self.FRAME_BG_COLOR,
        borderwidth=0,
        highlightthickness=0,
        relief=tk.SUNKEN
    )
    style.configure(
        "Panel.TFrame",
        background=self.PANEL_BACKGROUND,
        borderwidth=0,
        highlightthickness=0,
        relief=tk.SUNKEN
    )

    self.root.protocol("WM_DELETE_WINDOW", self.on_destroy)

    data = self.restore_login()
```

```
        if data is None:
            self.login_tab(False)
        else:
            self.win.place(
                "autologin.label",
                ttk.Label(
                    self.root,
                    font="Arial 24 bold",
                    text="Вхід в акаунт..."
                ),
                relx=0.5,
                rely=0.5,
                anchor=tk.CENTER
            )
            self.send(["login", data[0], data[1]])

        self.root.mainloop()


if __name__ == "__main__":
    MessengerClient()
    from passageidentity import Passage, PassageError

# Passage setup
PASSAGE_API_KEY = os.environ.get("PASSAGE_API_KEY")
PASSAGE_APP_ID = os.environ.get("PASSAGE_APP_ID")
try:
    psg = Passage(PASSAGE_APP_ID, PASSAGE_API_KEY)
except PassageError as e:
    print(e)
    exit()
# decorator that will run before every route in the auth blueprint
@auth.before_request
def before_request():
    try:
        g.user = psg.authenticateRequest(request)
    except PassageError as e:
        # this is an issue with the auth check, return 401
        return render_template('unauthorized.html')

@auth.route('/dashboard', methods=['GET'])
def dashboard():
    # g.user will be set here.
      # use Passage to get the user information and add it to the dashboard
    psg_user = psg.getUser(g.user)

    return render_template('dashboard.html', email=psg_user.email)
```