Вінницький національний технічний університет
Факультет менеджменту та інформаційної безпеки
Кафедра менеджменту та безпеки інформаційних систем

БАКАЛАВРСЬКА ДИПЛОМНА РОБОТА

на тему:

Розробка захищеного чат-месенджера

Виконав: студент 4 курсу, гр.1КІТС-20б
спеціальності 125– Кібербезпека
Освітня програма – Кібербезпека
інформаційних технологій та систем
(шифр і назва напряму підготовки, спеціальності)

Швець В. В.
(прізвище та ініціали)

Керівник: к.т.н., доц., каф. МБІС
Яремчук Ю. Є.
(прізвище та ініціали)
«6» червня 2024 р.

Рецензент: к.т.н., доц., каф. ОТ
Снігур А. В.
(прізвище та ініціали)
«6» червня 2024 р.

Допущено до захисту
Голова секції УБ кафедри МБІС
д.т.н., проф. Яремчук Ю.Є.
"6" червня 2024р.

Вінниця ВНТУ – 2024

Факультет менеджменту та інформаційної безпеки
Кафедра менеджменту та безпеки інформаційних систем

Рівень вищої освіти І-й (бакалаврський)
Галузь знань 12 – Інформаційні технології
Спеціальність 125 – Кібербезпека
Освітньо-професійна програма - Кібербезпека інформаційних технологій та систем

ЗАТВЕРДЖУЮ
Голова секції УБ, кафедра МБІС
д.т.н., проф. Яремчук Ю.Є.
"22" березня 2024 р.

З А В Д А Н Н Я
на бакалаврську дипломну роботу студенту
Швецю Вадиму Вікторовичу
(прізвище, ім'я, по-батькові)

1. Тема роботи Розробка захищеного чат-месенджера
Керівник роботи Яремчук Юрій Євгенович
(прізвище, ім'я, по-батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від "11" березня 2024року
№ 80
2. Термін подання студентом роботи за тиждень до захисту
3. Вихідні дані до роботи Електронні джерела, наукові статті, книги та документи, пов'язані з темою. Існуюче програмне забезпечення, технічна документація, вимоги та обмеження до програмного забезпечення
4. Зміст текстової частини:
Для досягнення мети, було поставлено наступні задачі: здійснити аналіз існуючих чат месенджерів, здійснити аналіз існуючих методів криптографічного шифрування, побудувати запропоновані алгоритми роботи системи, здійснити проектування бази даних, здійснити проектування інтерфейсу, обрати мову програмування, та середовище розробки, програмно реалізувати захищений чат-месенджер, провести тестування розробленого чат месенджера
Блок-схема алгоритму реєстрації, блок-схема алгоритму авторизації, блок-схема алгоритму роботи чату

5. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Розділ I	Яремчук Ю.Є., д.т.н., проф. каф. МБІС		
Розділ II	Яремчук Ю.Є., д.т.н., проф. каф. МБІС		
Розділ III	Яремчук Ю.Є., д.т.н., проф. каф. МБІС		

6. Дата видачі завдання 22 березня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1.	Визначення напрямку бакалаврської дипломної роботи, формулювання теми	23.03.2024	28.03.2024	Виконано
2.	Аналіз предметної області обраної теми	02.04.2024	08.04.2024	Виконано
3.	Розробка запропонованих алгоритмів роботи системи	09.04.2024	15.04.2024	Виконано
4.	Проектування бази даних	17.04.2024	21.04.2024	Виконано
5.	Проектування користувацького інтерфейсу	26.04.2024	29.04.2024	Виконано
6.	Програмна реалізація захищеного корпоративного файлообмінника з двофакторною автентифікацією	01.05.2024	14.05.2024	Виконано
7.	Тестування розробленого захищеного корпоративного файлообмінника з двофакторною автентифікацією	15.05.2024	19.05.2024	Виконано
8.	Попередній захист бакалаврської дипломної роботи	23.05.2024	23.05.2024	Виконано
9.	Виправлення, уточнення, корегування бакалаврської дипломної роботи	24.05.2024	29.06.2024	Виконано
10.	Захист бакалаврської дипломної роботи	12.06.2024	17.06.2024	Виконано

Студент _____ Швець В. В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Яремчук Ю. Є.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Дана дипломна робота присвячена розробці чат-месенджера. Метою дослідження і розробки є забезпечення безпечної передачі приватних повідомлень.

Під час розробки захищеного чат-месенджера, було розроблено запропоновані алгоритми його роботи, та побудовано відповідні блок-схеми.

Захищений чат месенджер був реалізований в середовищі розробки Visual Studio Code, за допомогою мови програмування Python, для розробки серверу, було використано програмний каркас PostgreSQL. Під час програмної реалізації було також використано додаткові модулі, фреймворки, та бібліотеки, такі як Django, Speakeasy, React, Cryptography, та Pytest.

В ході роботи, було проведено аналіз існуючих аналогівчат-месенджерів, проаналізовано їх переваги та недоліки, також було здійснено аналіз методів шифруваннф.

В результаті виконаної роботи, було проведено різноманітні тестування працездатності, і захищеності розробленого чат-месенджера.

Ключові слова: чат-месенджер, штфрування, безпечний обмін повідомленнями, захищений чат-месенджер.

ABSTRACT

This thesis is dedicated to the development of a chat messenger. The aim of the research and development is to ensure the secure transmission of private messages.

During the development of the secure chat messenger, proposed algorithms for its operation were developed, and corresponding block diagrams were constructed. The secure chat messenger was implemented in the Visual Studio Code development environment, using the Python programming language for server development, and the PostgreSQL framework was utilized. During the software implementation, additional modules, frameworks, and libraries such as Django, Speakeasy, React, Cryptography, and Pytest were also used.

In the course of the work, an analysis of existing chat messenger analogs was conducted, their advantages and disadvantages were analyzed, and encryption methods were also examined. As a result of the work performed, various tests were conducted to verify the functionality and security of the developed chat messenger.

Keywords: chat messenger, encryption, secure message exchange, secure chat messenger.

ЗМІСТ

ВСТУП

Актуальність теми. Давньогрецький філософ Аристотель вважав, що людина є істотою соціальною, тому що вона належить до групи, яка взаємодіє з іншими людьми. Людині від природи потрібно спілкування з іншими людьми для задоволення свої потреб та розвитку її як особистості.

З розвитком технологій і спілкування стало значно легшим, основну роль у сучасному спілкуванні відіграють системи обміну миттєвими повідомленнями – месенджері.

Месенджери займають центральне місце в сучасній світовій комунікації, та суттєво змінюють спосіб, яким люди взаємодіють один з одним. Від персонального спілкування до ділових комунікацій, ці платформи стали незамінними інструментами, забезпечуючи миттєвий обмін повідомленнями, медіа файлами та іншими даними. З розвитком технологій та зростанням інтернет-покриття, месенджери швидко еволюціонували від простих текстових сервісів до багатофункціональних платформ.

Особливої актуальності месенджери набули під час пандемії COVID-19, коли більшість людей перейшли на дистанційну роботу та навчання, а можливість залишатися на зв'язку з колегами, друзями та родичами стала можливою лише на відстані в цілях безпеки.

Окрім зручності, яку надають месенджери, вони також виступають потужним інструментом для бізнесу та політики Вони дозволяють компаніям швидко реагувати на запити клієнтів. А державі швидко передавати комерційні дані у інші держави та всередені держави, важливо необхідну або конфеденційну інформацію.

І важливо забезпечувати безпеку цих даних які передаються, щоб вони не пошкодились чи не перехопились і не були пошкоджені чи використані в непризначених цілях.

Мета роботи. Метою роботи є розробка чат-месенджера, який буде сприяти підвищенню конфіденційності, цілісності даних.

Для досягнення даної мети, необхідно виконати наступні завдання:

- здійснити аналіз існуючих аналогівчат-месенджерів;
- здійснити аналіз існуючих методів шифрування;

- розробити запропоновані алгоритми роботичат месенджера;
- обрати, та спроектувати базу даних;
- здійснити проектування користувацького інтерфейсу;
- здійснити вибір мови програмування, фреймворків, та бібліотек, здійснити вибір середовища розробки;
- здійснити програмну реалізацію спроектованого захищеного чат-месенджера;
- здійснити тестування роботи розробленого захищеного чат-месенджера;

Об’єктом дослідження є захищений чат-месенджео.

Предметом дослідження є розробка та ефективність застосування захищеного чат месенджера.

Новизна роботи полягає в розробці та дослідженні чат-месенджера, що буде забезпечувати шифрування повідомлень автоматично, і буде призначеним для комерційних та політичних цілей.

Практична цінність роботи, полягає в підвищені захищеності обміну повідомленнями. Розроблене рішення, дозволить не уникнути фінансових, репутаційних втрат, та крадіжки конфіденційних повідомлень, через можливі витоки, завдяки несанкціонованому доступу

1. АНАЛІЗ ІСНУЮЧИХ ЧАТ-МЕСЕНДЖЕРІВ ТА МЕТОДІВ ШИФРУВАННЯ

У цьому розділі пропонується розгляд та аналіз найвідоміших чат-месенджерів, проведено дослідження їх рівня захищеності та зручності для користувачів. Також в поданому розділі наведено розбір на аналіз основних понять, які використовуються в криптографічному шифруванні, а також основні методи шифрування.

1.1. Аналіз існуючих чат-месенджерів

Чат-месенджери служать програмами обміну миттєвими повідомленнями, які полегшують спілкування в реальному часі. Користувачі можуть брати участь у розмовах з окремими особами чи групами, миттєво обмінюватися повідомленнями та отримувати відповіді за лічені секунди. Ця безпосередність не тільки спрощує спілкування, але й забезпечує швидке прийняття рішень і співпрацю. Крім того, чат-месенджери підтримують різні форми спілкування, включаючи текст, голос і відео, що робить їх універсальними інструментами для різних комунікаційних уподобань. Крім того, такі функції, як емодзі та наклейки, додають розмовам виразності та персоналізації, покращуючи загальну взаємодію з користувачем.

Чат-месенджери перетворилися на універсальні комунікаційні платформи, які об’єднують широкий спектр функцій, щоб задовольнити різноманітні потреби своїх користувачів. Основним фактором їхньої привабливості є зручний інтерфейс, яким володіють більшість месенджерів, який зазвичай інтуїтивно зрозумілий, що полегшує новим користувачам навігацію різними функціями без крутого навчання [1]. По суті, чат-месенджери надають базові, але важливі послуги, такі як увімкнення текстового чату, обмін зображеннями та здійснення голосових чи відеодзвінків, які підтримують фундаментальну потребу в людському зв’язку в епоху цифрових технологій [1]. Окрім цих основних функцій, чат-месенджери розширили свою корисність, включивши розширені функції, такі як голос через Інтернет-протокол (VoIP) для високоякісних голосових і відеодзвінків, можливість обмінюватися файлами різних типів та інтеграцію чат-ботів, які можуть допомогти користувачам з діапазоном запитів, від обслуговування клієнтів до споживання

інтерактивного контенту [2]. Ця багатофункціональність у поєднанні зі зручністю обміну повідомленнями через комп’ютерні мережі дозволяє користувачам ефективно спілкуватися та обмінюватися різноманітним цифровим вмістом, що робить месенджери чатів незамінним інструментом у сучасному взаємопов’язаному світі [2].

Чат-месенджери набули величезної популярності як для особистого, так і для професійного використання. В особистих налаштуваннях люди часто використовують чат-месенджери для соціальної взаємодії, залишаючись на зв’язку з друзями та членами родини незалежно від фізичної відстані. З іншого боку, у професійному середовищі чат-месенджери стали важливими інструментами для спілкування, пов’язаного з роботою, що дозволяє швидко обмінюватися інформацією, оновлювати проекти та координувати роботу між членами команди. Крім того, доступність чат-месенджерів на різних пристроях, таких як смартфони, планшети та настільні комп’ютери, гарантує, що користувачі можуть залишатися на зв’язку в будь-який час і в будь-якому місці.

Останнім періодом Чат-месенджери значно розвинулися, ставши невід'ємною частиною нашого цифрового життя. Вони пропонують широкий спектр функцій, окрім широкого обміну текстовими повідомленнями, включаючи такі функції, як голосові та відеодзвінки, обмін файлами та навіть платіжні послуги. Найпопулярніші чат-месенджери мають безліч характеристик, які сприяли їх широкому програмному забезпеченню та залученню користувачів.

Використання чат-месенджерів викликає проблеми з конфіденційністю, оскільки ці платформи можуть збирати дані користувачів для цільової реклами та інших цілей. Інформація, яка надсилається через месенджери чату, включно з повідомленнями, мультимедійними файлами та особистими даними, може бути вразливою до зломів або несанкціонованого доступу, що викликає питання щодо безпеки та конфіденційності даних. Крім того, політика конфіденційності чат-месенджерів часто є складною та може не давати користувачам чіткого уявлення про те, як використовуються їхні дані та як вони передаються. Цей брак прозорості може підірвати довіру та викликати тривогу щодо захисту інформації користувачів.

Наразі найкращим та найзахищенішим месенджером в Україні є Signal. Далі

йдуть WhatsApp, Facebook Messenger і Telegram.[4]

WhatsApp є одним із найпоширеніших чат-месенджерів у всьому світі, володіє своїми комплексними функціями, які забезпечують різноманітні комунікаційні потреби. Його популярність пояснюється поєднанням зручного дизайну, надійної безпеки та різноманітних заходів з можливістю зв'язку.

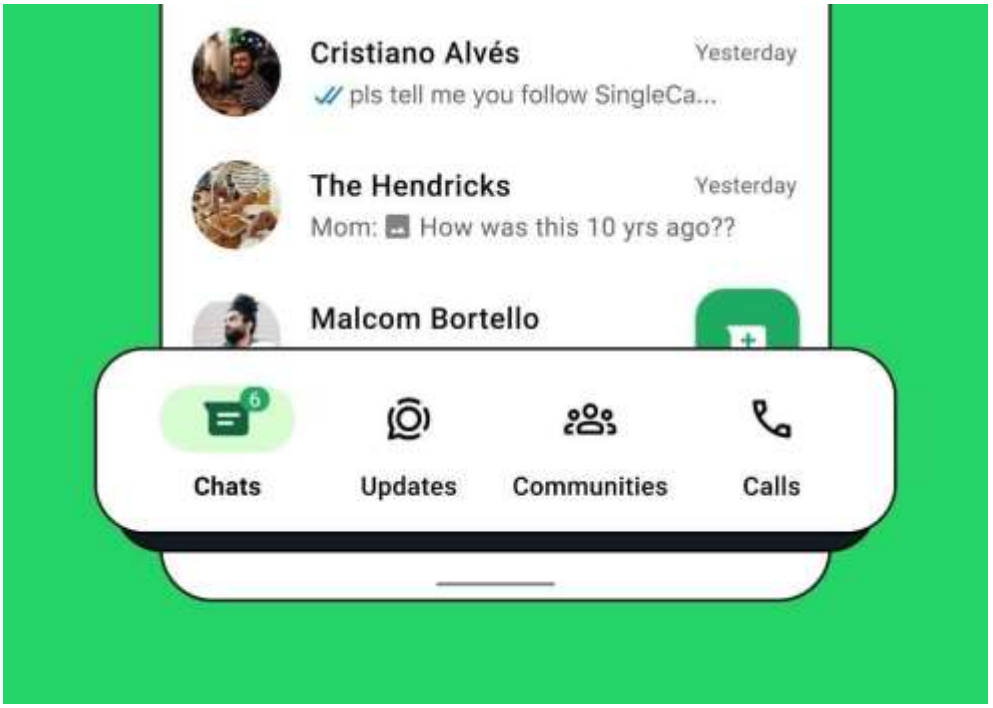


Рисунок 1.1 – «Інтерфейс месенджера WhatsApp»

WhatsApp має простий та інтуїтивно зрозумілий інтерфейс, який полегшує навігацію користувачам будь-якого віку. Його чистий дизайн гарантує, що користувачі можуть легко отримувати доступ до чатів, здійснювати дзвінки та керувати налаштуваннями, не перевантажуючи складними функціями.

Доступний на платформах Android, iOS і настільних ПК, гарантує, що користувачі можуть залишитися на зв'язку незалежно від пристрою, який вони вимагають. Додаток плавно синхронізує повідомлення на всіх пристроях, створюючи користувачам зручне перемикання між телефоном, планшетом і комп'ютером.

Наскрізне шифрування є основною функцією WhatsApp із наскрізним шифруванням для всіх форм спілкування, включаючи повідомлення, дзвінки та обмін медіафайлами. Це означає, що лише відправник і одержувач може читати повідомлення або слухати дзвінки, забезпечуючи конфіденційність і безпеку.

Підтримка Rich Media. Користувачі WhatsApp можуть надсилати й отримувати широкий спектр медіа-файлів, таких як фотографії, відео, голосові повідомлення та документи. Додаток підтримує різні формати файлів, що робить його універсальним інструментом для обміну інформацією та медіа.

Групові чати є важливою функцією WhatsApp, яка дозволяє користувачам спілкуватися з кількома людьми одночасно. WhatsApp також пропонує списки трансляцій, які дозволяють користувачам надсилати повідомлення кільком контактам одночасно без створення групового чату.

Додаток містить високоякісні функції голосових та відеодзвінків, що дозволяє користувачам використовувати особливості та групові дзвінки через Інтернет. Ця функція була особливо корисною для підтримки зв’язку з друзями та родиною по всьому світу, пропонуючи економічно ефективну альтернативу традиційним міжнародним дзвінкам.

Функція статусу дозволяє користувачам ділитися текстом, фотографіями, відео та GIF-файлами, які зникають через 24 роки, подібно до функції історії на інших платформах соціальних мереж. Це додає програмі динамічний та інтерактивний елемент, що дозволяє користувачам ділитися оновленнями про свій день або висловлювати себе творчо.

WhatsApp надає користувачам широкий спектр налаштувань конфіденційності та безпеки, включно з можливістю контролювати, хто може переглядати їхні фотографії, профіль, встановлення в мережу та статус оновлення. Користувачі також можуть блокувати контакти, повідомляти про спам і вмикати двоетапну перевірку для додаткової безпеки

Для компаній WhatsApp пропонує окрему програму, призначену для зв'язку з клієнтами. WhatsApp Business містить такі функції, як автоматичне повідомлення, швидкі відповіді та мітки для організації контактів, що полегшує компанії керування спілкуванням із клієнтами.

По суті, повний набір функцій WhatsApp — починаючи від зручного інтерфейсу та доступності на кількох платформах до сильного акценту на безпеці та конфіденційності — робить його улюбленим месенджером для мільйонів користувачів у всьому світі. Його здатність адаптуватися та запроваджувати нові

функції, зберігаючи при цьому простоту та легкість у використанні, значно сприяла його широкій популярності.

Signal – це програма для обміну повідомленнями, яка приділяє значну увагу конфіденційності та безпеці. Він привернув увагу та похвалу прихильників конфіденційності та експертів з кібербезпеки в усьому світі.

Додаток пропонує кілька помітних функцій, спрямованих на підвищення конфіденційності користувачів і забезпечення бездоганного обміну повідомленнями:

1. Наскрізне шифрування: в основі набору функцій Signal лежить найсучасніше наскрізне шифрування на основі протоколу Signal з відкритим кодом. Це гарантує, що всі форми спілкування на платформі, включаючи текстові повідомлення, голосові та відеодзвінки та обмін медіафайлами, є безпечними та доступними лише для відправника та одержувача.

2. Повідомлення, що зникають: користувачі можуть налаштувати повідомлення так, щоб вони зникали через певний період, підвищуючи конфіденційність і безпеку, особливо для конфіденційної інформації.

3. Групові дзвінки та повідомлення: Signal підтримує груповий обмін повідомленнями та нещодавно представлені зашифровані групові відеодзвінки, що робить його життєздатним варіантом як для особистого, так і для професійного спілкування.

4. Без реклами чи трекерів: Signal зобов’язується не показувати рекламу, не відстежувати користувачів і не продавати дані користувачів. Це зобов’язання є ключовим у філософії «користувач першочергово».

Заходи безпеки Signal є одними з найнадійніших серед додатків для обміну повідомленнями:

1. Signal Protocol вважається золотим стандартом шифрування. Він був прийнятий іншими службами обміну повідомленнями, включаючи WhatsApp, для обміну повідомленнями з наскрізним шифруванням.

2. Відкритий вихідний код: кодова база Signal є відкритою, що дозволяє незалежним експертам переглядати та перевіряти її безпеку. Ця прозорість зміцнює довіру та гарантує, що будь-яка вразливість може бути виявлена та усунена швидко.

3. Закритий відправник: ця функція приховує особу відправника від серверів Signal, додаючи до спілкування ще один рівень конфіденційності.

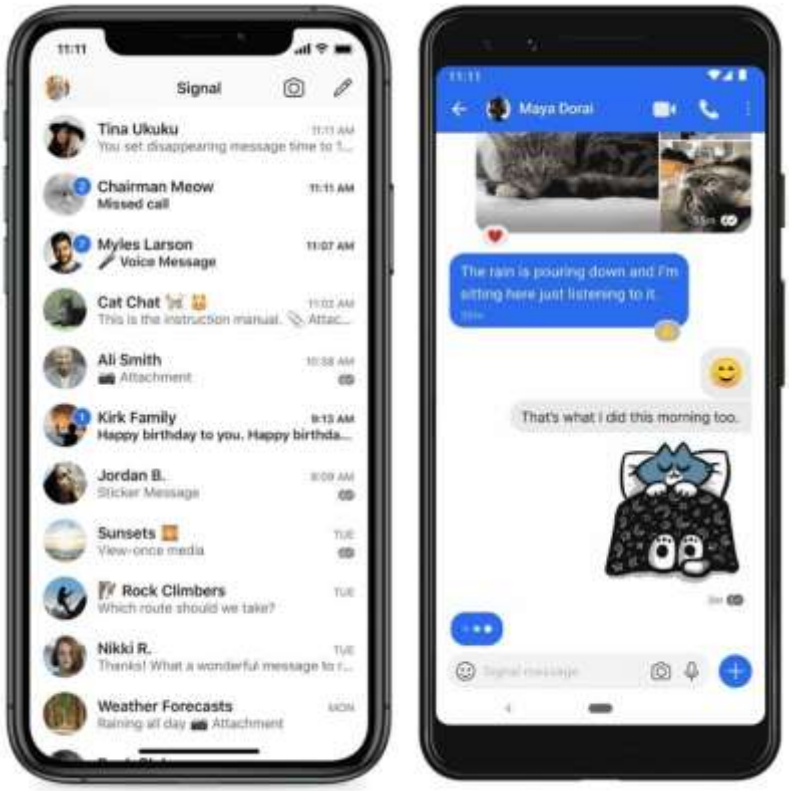


Рисунок 1.2 – «Інтерфейс додатку Signal»

Користувацький інтерфейс Signal є простим та інтуїтивно зрозумілим, зосередженим на простоті використання. Користувачі можуть персоналізувати чати за допомогою спеціальних шпалер і налаштувань темного режиму.

Незважаючи на безпеку, Signal не поступається функціональністю, пропонуючи повний набір функцій обміну повідомленнями, які користувачі очікують від сучасних програм. Хоча Signal може не пропонувати стільки функцій, скільки Telegram або соціальні інтеграції Facebook Messenger, він надає всі основні можливості обміну повідомленнями з великим акцентом на безпеку.[5. Чому саме Signal? https://signal.org/uk/]

Підсумовуючи, Signal представляє еталон безпечного обміну повідомленнями, пропонуючи надійне шифрування, прихильність конфіденційності та зручність для користувача. Він приваблює тих, хто надає перевагу конфіденційності над широкими можливостями чи соціальною інтеграцією, що робить його кращим вибором для осіб та організацій, які піклуються про безпеку.

Хоча такі програми, як Signal, пропонують чудове наскрізне шифрування як

стандарт для всіх повідомлень, Telegram надає користувачам можливість використовувати його вибірково через секретні чати.

За статистикою, Telegram входить у п’ятірку найпопулярніших додатків в Україні. Користуються ним широкі верстви населення. Загальна кількість користувачів по всьому світу перевалила за пів мільярда. На перший погляд, це просто цифри. А для когось Telegram – серйозний інструмент електронної розвідки та соціальної інженерії. Чимало функцій у ньому є шпигунськими.[3]

Telegram – це хмарний додаток для обміну повідомленнями, відомий своїм акцентом на швидкості та безпеці, що робить його популярним вибором серед користувачів у всьому світі. Його детальний аналіз можна розділити на кілька аспектів, включаючи функції, безпеку, інтерфейс користувача та порівняння з іншими програмами обміну повідомленнями.

Telegram пропонує широкий спектр функцій, які покращують взаємодію з користувачем:

1. Хмара: на відміну від багатьох інших програм для обміну повідомленнями, повідомлення Telegram працюють у хмарі, що дозволяє користувачам отримувати доступ до своїх повідомлень з кількох пристроїв одночасно. Ця функція особливо корисна для тих, хто використовує кілька пристроїв, таких як телефон, планшет і комп’ютер.

2. Групи та канали: Telegram підтримує великі групи до 200 000 учасників, що робить його придатним для великих спільнот. Канали - це унікальна функція, яка дозволяє користувачам транслювати повідомлення необмеженій кількості передплатників. Ці функції широко використовуються для створення спільноти, маркетингу та трансляції новин.

3. Боти: Боти Telegram – це програми сторонніх розробників, які працюють у Telegram. Користувачі можуть взаємодіяти з ботами, щоб отримати різноманітні послуги або виконувати завдання, наприклад отримувати оновлення новин, інтегруватися з іншими службами або навіть грати в ігри.

4. Секретні чати: для тих, хто турбується про конфіденційність, Telegram пропонує секретні чати, які використовують наскрізне шифрування, щоб гарантувати, що повідомлення можуть бути прочитані лише одержувачем. Крім

того, секретні чати мають функцію самознищення, що дозволяє повідомленням зникати через встановлений час.

Безпека є однією з головних переваг Telegram. Telegram використовує свій власний протокол шифрування під назвою MTProto. У той час як стандартні чати використовують шифрування клієнт-сервер/сервер-клієнт, секретні чати використовують наскрізне шифрування, забезпечуючи додатковий рівень безпеки.

Додаток збирає мінімум даних користувачів і пишається своєю відданістю конфіденційності користувачів. Користувачі мають можливість створювати облікові записи за допомогою віртуального номера телефону, додатково захищаючи свою особистість.

Рисунок 1.3 – «Інтерфейс Telegram»

Інтерфейс користувача Telegram простий, інтуїтивно зрозумілий і зручний. Він пропонує параметри налаштування, такі як фон і теми чату, що дозволяє користувачам персоналізувати свій досвід. Навігація проста, завдяки чітким значкам і меню можна легко отримати доступ до ключових функцій, таких як чати, групи, канали та налаштування.[5]

Підсумовуючи, поєднання швидкості, безпеки та багатого набору функцій сприяє зростанню популярності Telegram. Для індивідуального використання, створення спільноти чи бізнес-цілей він пропонує універсальну платформу, яка

задовольняє широкий спектр комунікаційних потреб. Його постійний розвиток і оновлення продовжують розширювати його можливості, гарантуючи, що він залишається сильним конкурентом на переповненому ринку додатків для обміну повідомленнями.

Facebook Messenger – це широко використовуваний чат-месенджер, який полегшує спілкування між користувачами Facebook. Він значно вдосконалився з моменту свого першого випуску, ставши автономним додатком із різними функціями, крім простого обміну повідомленнями. Ось поглиблений аналіз Facebook Messenger, зосереджений на його функціях, безпеці, інтерфейсі користувача та його позиції в порівнянні з іншими чат-месенджерами.[6]

Facebook Messenger пропонує широкий спектр функцій, що робить його не просто програмою для обміну повідомленнями. За своєю суттю Messenger дозволяє користувачам надсилати текстові повідомлення, ділитися зображеннями, відео та посиланнями. Він підтримує голосові та відеоповідомлення, забезпечуючи багатий досвід спілкування. Також користувачі можуть здійснювати голосові та відеодзвінки через Інтернет, що робить його універсальним інструментом для особистого та професійного спілкування без потреби в окремому додатку для викликів. Messenger підтримує групові чати, які включають такі функції, як опитування, групові голосові та відеодзвінки, що робить його зручним для координації з друзями, родиною чи робочими командами.

Запозичуючи успіх таких платформ, як Snapchat, Messenger включає функцію Stories, що дозволяє користувачам ділитися оновленнями, які зникають через 24 години. Він також пропонує цікаві фільтри та ефекти для фотографій і відео.

Безпека є важливим аспектом будь-якої програми обміну повідомленнями. Messenger пропонує наскрізне шифрування, але воно є необов’язковим і його потрібно ввімкнути, розпочавши «Секретну розмову». Це шифрує повідомлення так, що їх можуть прочитати лише відправник і отримувач, навіть не Facebook.

Як частина Facebook, методи Messenger щодо даних користувачів і конфіденційності пов’язані із загальною політикою Facebook. Це викликало занепокоєння серед користувачів, які піклуються про конфіденційність.

Рисунок 1.4 – «Зовнішній вигляд додатка Facebook Messenger»

Facebook Messenger має яскравий і зручний інтерфейс. У додатку є нижня панель навігації для легкого доступу до чатів, людей і пошуку розділів. Розділ виявлення дозволяє користувачам знаходити ботів, ігри та компанії. Користувачі можуть налаштовувати чати за допомогою різних кольорів, псевдонімів і реакцій емодзі. Він також підтримує темний режим, який легше впливає на очі в умовах слабкого освітлення.

Таким чином, Facebook Messenger — це багатофункціональна програма обміну повідомленнями, створена для соціальної взаємодії та зручності. Він вирізняється широким набором функцій та інтеграцією з ширшою екосистемою Facebook. Однак користувачі, які стурбовані конфіденційністю, можуть вважати необов’язковий характер його наскрізного шифрування та методи обробки даних є менш привабливими. Незважаючи на ці занепокоєння, простота використання та велика база користувачів роблять його популярним вибором для цифрового спілкування.

Цифрова ера започаткувала підвищене усвідомлення важливості конфіденційності та безпеки в онлайн-спілкуванні. У відповідь на цю потребу українці розробили революційний захищений месенджер Dober, який встановив новий стандарт для зашифрованих програм обміну повідомленнями.

Захищений месенджер Dober вирізняється своїми надійними технічними функціями, зокрема протоколами шифрування. Dober використовує наскрізне шифрування, гарантуючи, що лише відправник і одержувач можуть отримати доступ до вмісту своїх повідомлень. Крім того, програма пропонує передові методи

шифрування, такі як RSA та AES, що ще більше підвищує безпеку спілкування користувачів. Що стосується налаштувань конфіденційності, Dober надає користувачам низку опцій для налаштування їхніх параметрів безпеки, включаючи можливість установити термін дії повідомлень і ввімкнути двофакторну автентифікацію. Порівняльний аналіз показує, що функції безпеки Dober перевершують характеристики популярних програм для обміну повідомленнями, таких як WhatsApp і Telegram, що робить його найкращим вибором для користувачів, які надають перевагу конфіденційності та конфіденційності. [7]

Рисунок 1.5 – «Інтерфейс месенджера Dober»

Окрім технічної майстерності, захищений месенджер Dober вирізняється враженнями від користувача та дизайном інтерфейсу. Додаток може похвалитися інтуїтивно зрозумілим інтерфейсом, який надає перевагу простоті навігації, що дозволяє користувачам легко отримувати доступ до функцій і функцій. Крім того, Dober пропонує широкі можливості налаштування, що дозволяє користувачам персоналізувати обмін повідомленнями за допомогою тем, шрифтів і фону чату. Відгуки користувачів були виключно позитивними, і багато хто підкреслював простоту та естетичну привабливість програми. Такі орієнтовані на користувача елементи дизайну сприяють позитивному загальному досвіду та відрізняють Dober від конкурентів на ринку безпечних повідомлень.

Поява захищеного месенджера Dober справила значний вплив на ринок додатків для обміну повідомленнями, викликавши сильну реакцію як у користувачів, так і у конкурентів. Після запуску Dober привернув широку увагу та визнання за інноваційний підхід до безпеки та конфіденційності користувачів. Конкуруючі програми обміну повідомленнями звернули увагу на успіх Dober і внесли корективи у власні функції безпеки у відповідь на підвищений попит на безпечні комунікаційні платформи. Заглядаючи вперед, можна сказати, що вплив Dober на майбутній розвиток програм безпечного обміну повідомленнями відчутний, і галузеві тенденції, ймовірно, зміщуватимуться в бік надання пріоритету шифруванню та заходам конфіденційності відповідно до стандартів, встановлених цим новаторським українським витвором.

Підсумовуючи, захищений месенджер Dober є свідченням українських інновацій та технологічної експертизи у сфері безпечного спілкування. Завдяки вдосконаленим протоколам шифрування, зручному інтерфейсу та впливу на ринок Dober встановив новий стандарт для програм безпечного обміну повідомленнями. Оскільки цифровий ландшафт продовжує розвиватися, спадщина Dober готова сформувати майбутнє онлайн-комунікації, підкреслюючи першочергову важливість конфіденційності та безпеки в епоху цифрових технологій.

Отож, чат-месенджери пропонують низку додаткових функцій, які підвищують зручність і функціональність користувачів. Багато чат-месенджерів надають пріоритет конфіденційності користувачів, використовуючи наскрізне шифрування, гарантуючи безпеку розмов і захист від зовнішніх загроз. Крім того, ці платформи часто надають можливості обміну файлами, дозволяючи користувачам безперешкодно обмінюватися документами, зображеннями та іншими файлами під час розмов. Крім того, чат-месенджери можуть інтегруватися з іншими програмами та службами, пропонуючи цілісне спілкування, дозволяючи виконувати такі завдання, як планування зустрічей, здійснення платежів або доступ до вмісту, не виходячи з інтерфейсу обміну повідомленнями.

1.2. Загальні поняття криптографічного шифрування

Найпоширенішим методом захисту передачі даних у веб-додатках або проектах

з інформатики є сучасна криптографія. Це як мова секретного коду, яка допомагає зберігати інформацію в безпеці.[8]

Криптографія має багато практичних застосувань, які можуть бути цінними навичками:

1) Можливість використовувати криптографію для захисту зв’язку шляхом шифрування повідомлень і електронних листів.

2) По-друге, можемо використовувати його для захисту наших даних у програмах, захищаючи дані користувачів, як-от паролі та особисту інформацію.

3) Також є можливість забезпечити зберігання файлів, захистивши конфіденційні файли та документи.

4) Далі ми також можемо використовувати криптографію для захисту наших платформ електронної комерції, захищаючи онлайн-транзакції та платіжну інформацію.

5) Можемо створювати технологію блокчейну, забезпечуючи безпеку та цілісність транзакцій у системах на основі блокчейну.

6) Криптографія також може бути використана для захисту паролем для безпечного зберігання та керування паролями.

7) І найголовніше – це цифрові підписи для підтвердження автентичності цифрових повідомлень чи документів.[8]

Людські істоти споконвіку мали дві притаманні потреби: спілкуватися та ділитися інформацією та спілкуватися вибірково. Ці дві потреби породили мистецтво кодування повідомлень таким чином, щоб лише призначені люди мали доступ до інформації. Сторонні люди не могли витягти жодної інформації, навіть якщо зашифровані повідомлення потрапили їм у руки.

Мистецтво і наука приховування повідомлень для введення секретності в інформаційну безпеку визнано криптографією.

Слово «криптографія» виникло шляхом поєднання двох грецьких слів: «крипто», що означає прихований, і «графен», що означає письмо.

Криптографія існує дуже давно, ще до появи комп’ютерів та Інтернету. Люди завжди хотіли приховати свої повідомлення, тому знайшли хитрі способи це зробити.

Вдосконалені методи кодування, такі як кодування Віженера, з’явилися в 15 столітті, які пропонували переміщення літер у повідомленні з кількома змінними місцями замість того, щоб переміщати їх на однакову кількість місць.

Лише після 19 століття криптографія еволюціонувала від спеціальних підходів до шифрування до більш складного мистецтва та науки інформаційної безпеки.

На початку 20 століття винахід механічних і електромеханічних машин, таких як роторна машина Енігма, забезпечив більш досконалі та ефективні засоби кодування інформації.[8]

Щодня через Інтернет надсилається величезна кількість особистої інформації, включаючи паролі, введені на екранах входу, електронні листи з особистою інформацією та податкові документи, завантажені на сервери.

Приватні дані передаються через Інтернет пакетами за допомогою тих самих шляхів, що й публічні дані. На жаль, зловмисники знайшли методи перехоплення цих даних під час їх подорожі Інтернетом

Шифрування — це метод перетворення даних, щоб їх могли читати лише авторизовані особи. У процесі шифрування відкритий текст перетворюється в зашифрований за допомогою криптографічного ключа. Криптографічний ключ — це набір відомих математичних величин, погоджених як відправником, так і одержувачем.[9]

Цей процес забезпечує конфіденційність та безпеку обміну інформацією шляхом ускладнення доступу до даних без правильного ключа. Деякі загальні поняття, які часто використовуються в криптографічному шифруванні, включають:

1) Ключ - це параметр, що використовується у шифрувальному алгоритмі для шифрування або розшифрування даних. Ключ визначає, як саме буде перетворюватися вхідна інформація.

2) Симетричне шифрування - це метод шифрування, де один і той же ключ використовується як для шифрування, так і для розшифрування даних.

3) Асиметричне шифрування - це метод, де для шифрування та розшифрування використовуються різні ключі - публічний та приватний. Публічний ключ використовується для шифрування повідомлення, тоді як приватний ключ використовується для його розшифрування.

4) Цифровий підпис - це криптографічний метод, який дозволяє підтвердити автентичність та цілісність повідомлення за допомогою приватного ключа власника.

5) Хеш-функція - це криптографічне перетворення, яке призначене для отримання короткого унікального "відбитка" від будь-якого вхідного повідомлення. Хеш-функції використовуються для перевірки цілісності даних та підтвердження підпису.

Перший тип шифрування — це шифр підстановки, який використовує дуже простий метод заміни інших одиниць (символів або груп символів) на інші одиниці на основі різних правил підстановки, наприклад, A=N, B=O тощо. До комп'ютерів відправника та одержувача Безпека шифрування забезпечувалася встановленням ключів шифру поруч. Таким чином, навіть якщо месенджер був захоплений, зміст повідомлення залишався неясним.

Шифрування має вирішальне значення для захисту різних IT-активів і особистих ідентифікованих даних. У зв'язку з цим шифрування відповідає чотирьом важливим принципам

1) Конфіденційність – шифрування гарантує, що лише авторизовані сторони можуть отримати доступ і зрозуміти зашифровані дані. Кодує дані, щоб запобігти їх розумінню, якщо їх перехопить третя сторона. Змінюючи відкритий текст на зашифрований за допомогою криптографічних алгоритмів для захисту приватних даних. Наприклад: це як мати секретну мову, яку розумієте лише ви та ваші друзі.

2) Автентифікація – це процес приналежності частини даних, на які претендує користувач. Він перевіряє походження даних, які були зашифровані. Цифрові сертифікати та криптографічні протоколи автентифікують користувачів і пристрої, перевіряючи їхні облікові дані та підтверджуючи їхню особу перед встановленням безпечних каналів зв'язку. Автентифікація запобігає атакам уособлення та неавторизованому доступу, гарантуючи, що лише довірені об'єкти можуть розшифрувати та отримати доступ до зашифрованих даних.

3) Цілісність – шифрування допомагає підтримувати цілісність даних шляхом виявлення будь-яких неавторизованих змін або модифікацій.

Використовуючи такі методи, як цифрові підписи та коди автентифікації повідомлень (MAC), шифрування гарантує, що зашифровані дані залишаються незмінними під час передачі чи зберігання. Тож, можна сказати, це як накласти особливу печатку на лист. Якщо хтось спробує відкрити його до того, як він потрапить до вас, ви дізнаєтеся про це, оскільки пломба буде зламана.

Невідмова – функція відмови не дозволяє відправникам заперечувати, що вони надіслали зашифровані дані. Шифрування забезпечує неспростовність, гарантуючи, що сторони не можуть заперечувати свої дії чи транзакції. Наприклад, це схоже на отримання квитанції, коли ви щось купуєте. Якщо пізніше ви спробуєте сказати, що не купували його, квитанція підтвердить, що ви це зробили.

Примітиви криптографії — це не що інше, як інструменти та методи в криптографії, які можна вибірково використовувати для надання бажаних послуг безпеки:

- Шифрування
- Хеш-функції
- Коди автентифікації повідомлень (MAC)
- Цифрові підписи

У наведеній нижче таблиці показано примітиви, які можуть бути створені для певної служби безпеки самостійно.

Примітивні послуга	**Шифрування**	**Хеш-функція**	**MAC**	**Цифровий підис**
Конфіденційність	Так	Немає	Немає	Немає
Цілісність	Немає	Іноді	Так	Так
Аутентифікація	Немає	Немає	Так	Так
Не відмова	Немає	Немає	Іноді	Так

Таблиця 1.1 – Примітиви для служби безпеки

У цьому розділі було представлено основні поняття криптографічного шифрування, такі як ключ, симетричне та асиметричне шифрування, цифровий підпис і хеш-функція. Ці концепції грають важливу роль у забезпеченні безпеки та конфіденційності обміну даними в інтернеті. Шифрування дозволяє захистити інформацію від несанкціонованого доступу, забезпечити конфіденційність повідомлень та підтвердити їх автентичність. Розуміння та правильне використання

криптографічних методів є ключем до створення безпечного цифрового середовища.

1.3. Аналіз існуючих методів криптографічного захисту інформації

Криптографія – наука про способи перетворення інформації з метою її захисту від несанкціонованого доступу. Одним із видів такого перетворення є шифрування, яке забезпечує практичну неможливість читання або модифікації інформації зловмисниками. Існує цілий ряд алгоритмів шифрування даних, які можна розбити на дві великі групи.[10]

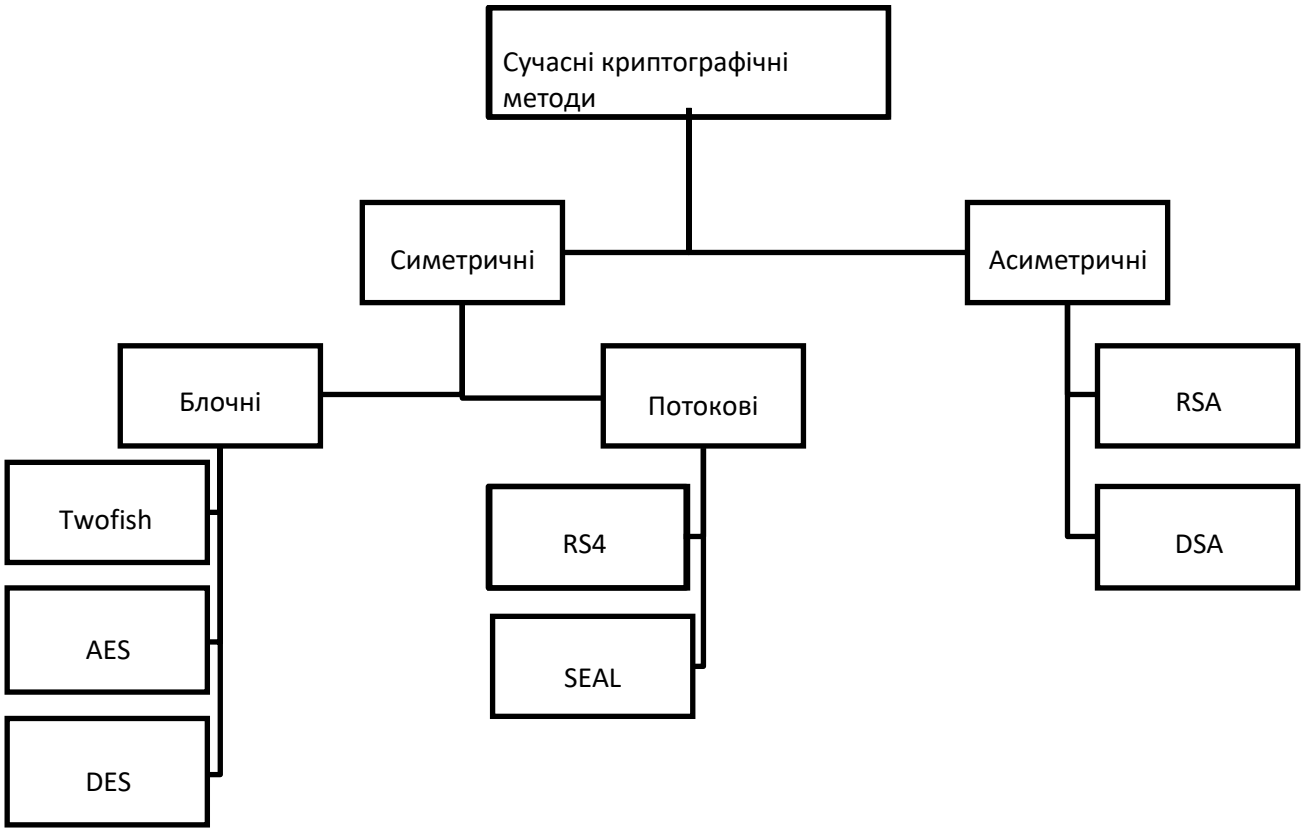


Рисунок 1.6 – «Сучасні криптографічні методи»

Шифрування даних захищає інформацію від втрати, зміни або зламу. Однак ключ дешифрування має бути закритим і захищеним від небажаного доступу, щоб гарантувати безпеку даних.

Будь-який тип даних, як ті, що передаються (наприклад, обмінюються через мережу), так і перебувають у стані спокою (наприклад, зберігаються на жорсткому диску), можуть бути зашифровані.

Є два методи шифрування, які широко використовуються:

Симетричне шифрування: той самий ключ використовується як для шифрування, так і для дешифрування в симетричному шифруванні.

Стандарт шифрування даних (DES) — це тип шифрування, який захищає цифрові дані за допомогою одного ключа. Хоча він може не забезпечувати таку безпеку, як поточні методи, через його коротку довжину ключа в 56 біт, він відіграв життєво важливу роль в еволюції криптографії.

Advanced Encryption Standard (AES), цей симетричний блоковий шифр використовується урядом США для захисту секретних даних. AES широко використовується в програмному та апаратному забезпеченні в усьому світі для шифрування цінної інформації. Він відіграє важливу роль у державній комп'ютерній безпеці, кібербезпеці та захисті електронних даних.

Стандарт потрійного шифрування даних (Triple DES) — це стандарт у криптографії, який використовує ключі фіксованої довжини та включає три проходи алгоритму DES. Це метод шифрування на основі симетричного блочного шифру, який означає, що і відправник, і одержувач мають однакові секретні ключі для шифрування та дешифрування.

Blowfish — це криптографічний метод, метою якого є заміна DES. Він розділяє повідомлення на 64-розрядні блоки та шифрує їх відповідно. Blowfish відомий своєю швидкістю, гнучкістю та чудовою безпекою, що робить його популярним вибором для захисту сайтів електронної комерції, банківських транзакцій та програм керування паролями. Велика популярність Blowfish серед розробників частково пояснюється його статусом суспільного надбання та безкоштовним використанням.

Twofish, наступник Blowfish, також використовує симетричне шифрування для дешифрування 128-бітних блоків даних без потреби ліцензії. На відміну від інших алгоритмів, Twofish завжди шифрує дані за 16 циклів незалежно від розміру ключа. Це робить його придатним як для програмного, так і для апаратного середовища та відомий своєю високою продуктивністю. Багато організацій використовують Twofish для безпечної передачі та захисту даних.

RC4 — це потоковий шифр із алгоритмом ключа змінної довжини. Цей

алгоритм шифрує один байт за раз (або більші одиниці даних). Ключовий вхід — це генератор псевдовипадкових бітів, який створює потік 8-бітних чисел, які неможливо передбачити без знання ключа введення. Вихід генератора, який називається потоком ключів, змішується по одному байту з шифром потоку відкритого тексту за допомогою операції X-OR.

Міжнародний алгоритм шифрування даних (IDEA) є прикладом шифрування симетричним ключем блокового шифру. IDEA використовує 128-бітний ключ і працює з 64-бітними блоками. По суті, він перетворює 64-бітний блок відкритого тексту на 64-бітний блок зашифрованого тексту.

Асиметричне шифрування – воно використовує відкритий ключ, який надається одержувачу даних, і закритий ключ, яким володіє власник даних. Оскільки асиметричне шифрування не потребує обміну закритим ключем, воно вважається більш безпечним.[8]

Асиметричне шифрування, яке часто називають криптографією з відкритим ключем, має два різні ключі для шифрування та декодування

RSA — це асиметрична система шифрування з відкритим ключем, яка працює як стандарт шифрування в Інтернеті. Шифрування RSA є надійним і безпечним. Це тип шифрування, який використовує пару ключів: відкритий ключ і закритий ключ для шифрування та дешифрування відповідно.

Криптографія еліптичної кривої (ECC) − Криптографія еліптичної кривої (ECC) — це метод шифрування, подібний до RSA, який дозволяє шифрувати відкритим ключем. ECC, як випливає з назви, є технікою асиметричного шифрування, заснованою на алгебраїчній структурі еліптичних кривих зі скінченними полями.

Обмін ключами Діффі-Хеллмана використовується для створення спільного секрету, який можна використовувати для приватного спілкування під час надсилання даних через загальнодоступну мережу. Еліптична крива використовується для створення точок, а секретний ключ отримується за допомогою параметрів.

Схема шифрування ElGamal – це метод шифрування з асиметричним ключем для криптографії з відкритим ключем, заснований на обміні ключами Діффі-Хеллмана. Алгоритм ElGamal є альтернативою RSA для шифрування відкритим

ключем. Конфіденційність RSA залежить від складності розкладання великих цілих чисел. Безпека алгоритму ElGamal полягає в складності обчислення дискретних даних у великому простому масштабі.

Алгоритм цифрового підпису (DSA). Він використовується в цифрових підписах і для їх перевірки. Він заснований на математичних принципах модульного піднесення до степеня та дискретного логарифмування. Національний інститут стандартів і технологій (NIST) створив його в 1991 році.[8]

1.4. Висновки та постановка задач

Ринок переповнений численними чат-месенджерами, які дозволяють користувачам спілкуватися в режимі реального часу. Одними з месенджерів, які зараз мають найбільшу базу користувачів, є WhatsApp, Telegram, Viber, Signal, Facebook Messenger та багато інших.

Багато з цих месенджерів намагаються наскрізне шифрування для методів шифрування, які спрямовані на забезпечення конфіденційності користувача та безпеки його розміщення. Суть нарізного шифрування виникає в тому, що повідомлення шифруються на стороні відправника та розшифровуються на стороні одержувача, що унеможливлює прослуховування переданих даних між двома сторонами: таким чином забезпечується захист від прослуховування або перехоплення даних третіми сторонами. партії. Безпека месенджерів не залежить тільки від методів шифрування. Є й інші аспекти: правильне керування доступом до облікових записів, використання надійних паролів і відповідна політика безпеки в цілому також змінюють важливу роль.

Виходячи з проведеного аналізу можна сформувати тему роботи «Розробка захищеного чат-месенджера.

Для досягнення визначеної мети необхідно виконати наступні завдання:

- побудувати запропоновані алгоритми роботи системи;
- обрати мову програмування, та середовище розробки;
- програмно реалізувати захищений чат-месенджер;
- провести тестування чат-месенджера.

2. ПРОЕКТУВАННЯ ЗАХИЩЕНОГО КОРПОРАТИВНОГО ФАЙЛООБМІННИКА З ДВОФАКТОРНОЮ АВТЕНТИФІКАЦІЄЮ

В даному розділі буде розроблено запропоновані алгоритми роботи чат месенджера, та побудовано відповідні блок-схеми даних алгоритмів, буде обрано і спроектовано базу даних. Також буде спроектовано користувацький інтерфейс, для чат месенджера.

2.1. Розробка запропонованих алгоритмів роботи захищеного чат-месенджера

Пропонується розробити веб додаток який буде мати 3 основні сторінки. А саме, сторінка реєстрації нових користувачів, сторінка авторизації раніше зареєстрованих користувачів, і захищена сторінка самого чат месенджера. Чат-месенджер буде розроблено для використання в корпоративних, комерційних та політичних цілях, а саме для захисту приватних повідомлень. Тому його користувачами є високопосадовці, військові та політики, вони матимуть змогу обмінюватися конфеденційною інформацією, збільшуючи їх захист за рахунок двофакторної автентифікації та шифрування.

Виходячи з цього в даному підрозділі буде розроблено запропоновані алгоритми роботи системи захищеного чат-месенджера.

Алгоритм роботи системи захищеного чат-месенджера

Алгоритм реєстрації користувачів

Крок 1 – Початок.

Користувач розпочинає процес реєстрації, відкривши відповідну сторінку.

Крок 2 – Введення даних для реєстрації.

Користувач вводить свої дані на сторінці для реєстрації (електронна адреса, пароль).

Крок 3 – Надсилання запиту на сервер.

Після заповнення всіх обов’язкових полів для реєстрації, користувач натискає кнопку «Зареєструватись», таким чином його дані відправляються на сервер.

Крок 4 – Отримання запиту.

Сервер отримує запит на реєстрацію.

Крок 5 – Перевірка чи email дозволений для реєстрації.

Сервер перевіряє, чи введений клієнтом email дозволений для реєстрації, щоб уникнути несанкціонованої реєстрації користувачів, яким реєстрація заборонена.

Крок 6 – Email дозволений?

Крок 6.1 – Ні. Надсилання помилки про заборону реєстрації даного користувача.

Якщо даного email немає в списку дозволених, користувачу надсилається помилка з забороною реєстрації.

Крок 6.1.1 – Отримання відповіді.

Клієнт отримує відповідь.

Крок 6.1.2 – Виведення помилки користувачу.

Виводиться помилка користувачу про заборону реєстрації даного email.

Крок 6.2 – Так.

Якщо email є в списку дозволених для реєстрації, виконується крок .

Крок 7 – Перевірка чи користувач з таким email ще не зареєстрований.

Сервер перевіряє, чи ще не зареєстрований користувач з таким email.

Крок 8 – Email вже зареєстрований?

Крок 8.1 – Так. Надсилання помилки про те, що користувач з таким email уже зареєстрований.

Якщо користувач з таким email уже зареєстрований, клієнту надсилається помилка про це.

Крок 8.1.1 – Отримання відповіді.

Клієнт отримує відповідь.

Крок 8.1.2 – Виведення помилки користувачу.

Виводиться помилка користувачу про те, що користувач з таким email уже зареєстрований.

Крок 8.2 – Ні.

Якщо email ще не зареєстрований, виконується крок 9.

Крок 9 – Хешування паролю.

Сервер хешує введений пароль.

Крок 10 – Генерація секретного ключа.

Сервер генерує секретний ключ для двофакторної автентифікації.

Крок 11 – Створення нового користувача в базі даних.

Сервер створює нового користувача в базі даних.

Крок 12 – Надсилання відповіді клієнту.

Сервер надсилає відповідь клієнту.

Крок 13 – Отримання відповіді.

Клієнт отримує відповідь.

Крок 14 – Вивід повідомлення користувачу.

Клієнту виводиться секретний ключ для двофакторної автентифікації і повідомлення про успішну реєстрацію.

Крок 15 – Кінець.

Таким чином відбувається процес реєстрації з забороною несанкціонованої реєстрації користувачів.

Також було розроблено блок-схему запропонованого розробленого алгоритму реєстрації (рис. 2.1.1).

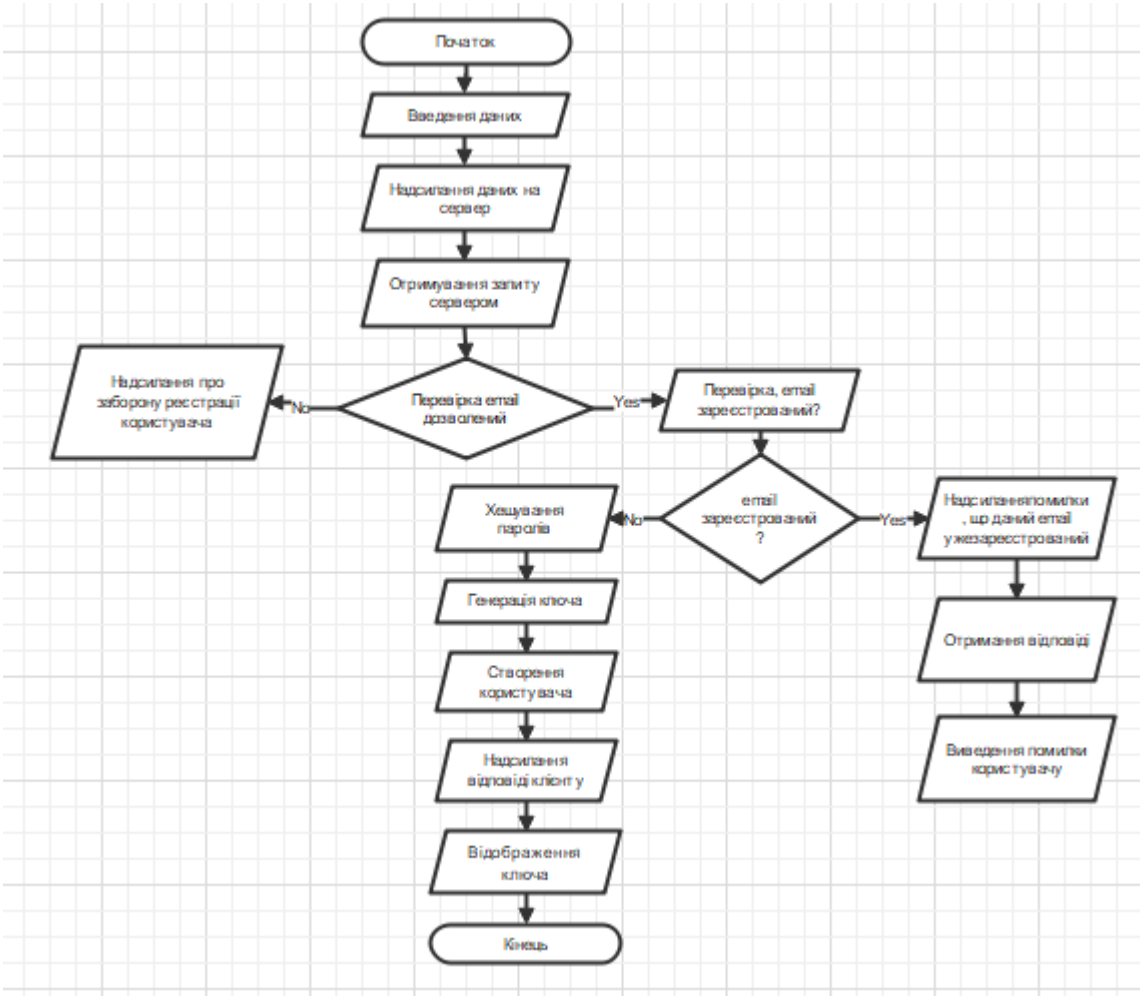


Рисунок 2.1 – розроблена блок-схема запропонованого алгоритму реєстрації користувачів

На блок схемі продемонстровано алгоритм реєстрації, з усіма необхідними перевірками, для того щоб уникнути несанкціонованої реєстрації користувачів, яким не дозволено доступ, до захищеного чат месенджера.

Далі розглянемо алгоритм авторизації користувачів захищеного чат месенджера.

Крок 1 – Початок.

Користувач розпочинає процес авторизації, відкривши відповідну сторінку.

Крок 2 – Введення даних для авторизації.

Користувач вводить свої дані (електронна адреса, пароль) на сторінці для авторизації.

Крок 3 – Надсилання запиту на сервер.

Після заповнення полів для авторизації користувач натискає кнопку «Увійти», таким чином його дані відправляються на сервер.

Крок 4 – Отримання запиту.

Сервер отримує запит на авторизацію.

Крок 5 – Перевірка даних користувача.

Сервер перевіряє, чи існує користувач з таким email і чи правильний пароль.

Крок 6 – Дані користувача правильні?

Крок 6.1 – Ні. Надсилання помилки про невірні дані.

Якщо дані неправильні, користувачу надсилається помилка про невірні дані.

Крок 6.1.1 – Отримання відповіді.

Клієнт отримує відповідь.

Крок 6.1.2 – Виведення помилки користувачу.

Виводиться помилка користувачу про невірні дані.

Крок 6.2 – Так.

Якщо дані правильні, виконується крок 7.

Крок 7 – Генерація тимчасового паролю для двофакторної автентифікації.

Сервер генерує тимчасовий пароль для двофакторної автентифікації і надсилає його користувачу (на email).

Крок 8 – Введення коду двофакторної автентифікації.

Користувач вводить отриманий код на сторінці авторизації.

Крок 9 – Надсилання паролю на сервер.

Пароль відправляється на сервер для перевірки.

Крок 10 – Перевірка паролю двофакторної автентифікації.

Сервер перевіряє правильність введеного паролю.

Крок 11 – Пароль правильний?

Крок 11.1 – Ні. Надсилання помилки про невірний пароль.

Якщо пароль неправильний, користувачу надсилається помилка про невірний пароль.

Крок 11.1.1 – Отримання відповіді.

Клієнт отримує відповідь.

Крок 11.1.2 – Виведення помилки користувачу.

Виводиться помилка користувачу про невірний пароль.

Крок 11.2 – Так.

Якщо пароль правильний, виконується крок 12.

Крок 12 – Авторизація успішна.

Користувач успішно авторизується і отримує доступ до захищеної сторінки чат-месенджера.

Крок 13 – Кінець.

Таким чином відбувається процес авторизації з двофакторною автентифікацією. Для авторизації, також було побудовано блок-схему.

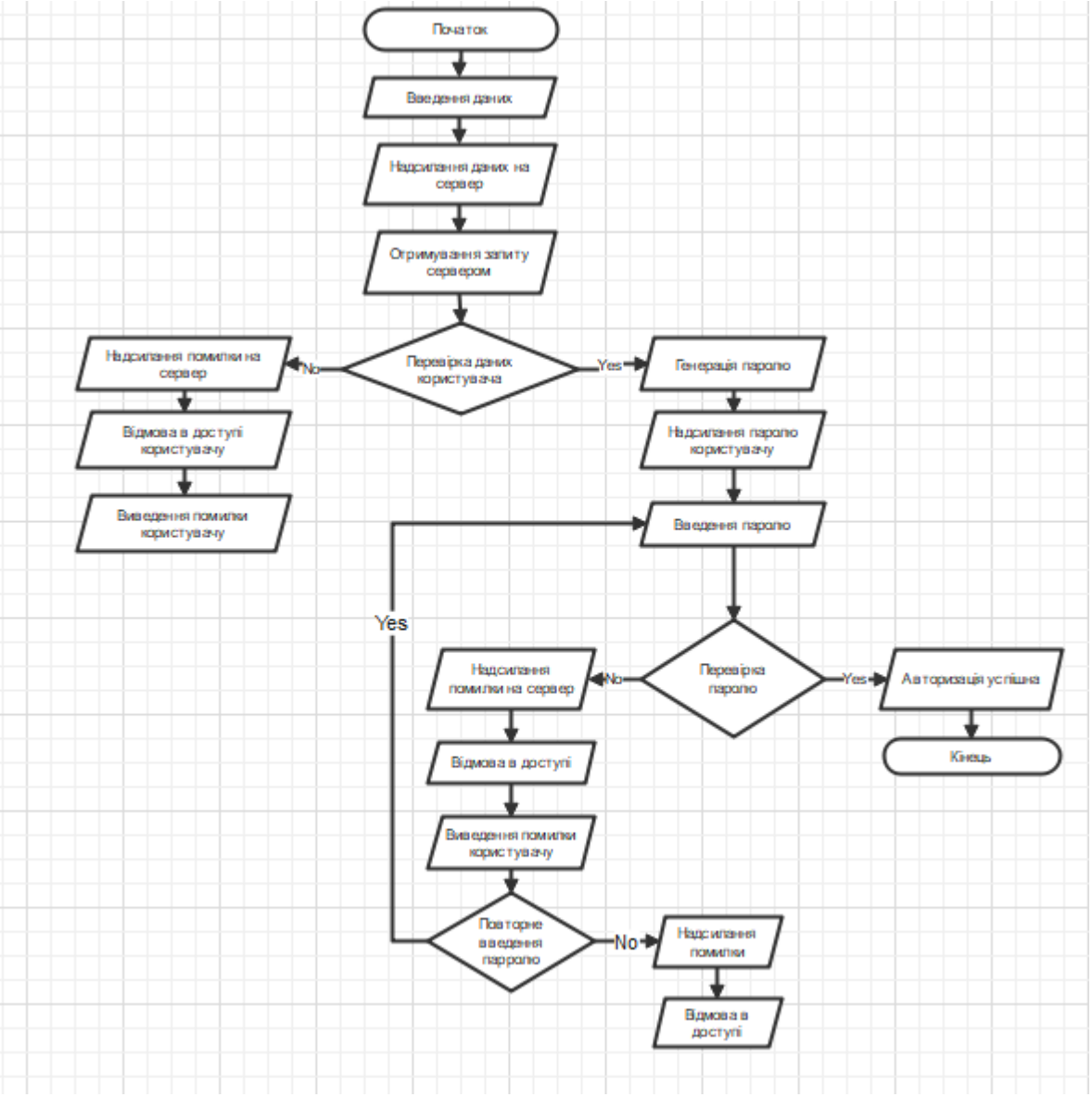


Рисунок 2.2 – розроблена блок-схема запропонованого алгоритму перевірки сесій користувачів

Також розглянемо алгоритм обміну повідомленнями

Крок 1 – Початок.

Користувач відкриває сторінку чат-месенджера.

Крок 2 – Введення повідомлення.

Користувач вводить повідомлення у відповідне поле.

Крок 3 – Шифрування повідомлення.

Повідомлення шифрується за допомогою криптографічного алгоритму з використанням ключа отримувача.

Крок 4 – Надсилання повідомлення.

Зашифроване повідомлення надсилається на сервер.

Крок 5 – Зберігання повідомлення на сервері.

Сервер зберігає зашифроване повідомлення в базі даних.

Крок 6 – Отримання повідомлення отримувачем.

Отримувач отримує зашифроване повідомлення з сервера.

Крок 7 – Введення ключа.

Отримувач вводить свій ключ для розшифрування повідомлення.

Крок 8 – Розшифрування повідомлення.

Повідомлення розшифровується і відображається у чаті.

Крок 9 – Зашифроване зберігання.

Повідомлення зберігаються на сервері у зашифрованому вигляді.

Крок 10 – Кінець.

Таким чином відбувається процес обміну повідомленнями з використанням шифрування. Також було розроблено блок-схему длязапропонованого алгоритму завантаження файлів рис.

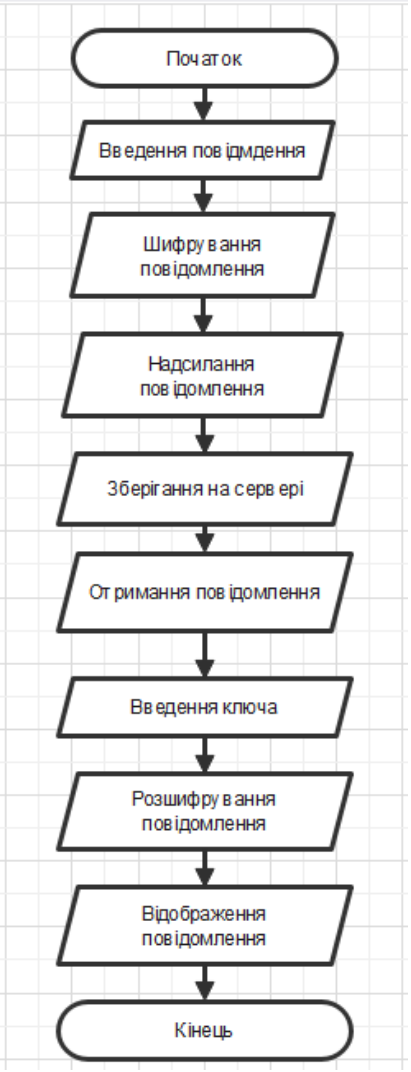


Рисунок 2.3 – розроблена блок-схема запропонованого алгоритму роботи чат-месенджера

Ці алгоритми забезпечують безпечну реєстрацію, авторизацію та обмін повідомленнями у чат-месенджері, з використанням сучасних криптографічних методів і двофакторної автентифікації. Таким чином відбувається процес реєстрації, з забороною несанкціонованої реєстрації користувачів.

2.2. Проектування бази даних для захищеного чат-месенджера

Для побудови чат месенджера, необхідновикористовувати не менш надійну і захищену базу даних. Так як в ній буде зберігатись як персональні дані зареєстрованих користувачів, так і зашифровані повідомлення нвдіслані користувачами. Тому для цього було обранобазу даних PostgreSQL.

PostgreSQL — це потужна об’єктно-реляційна база даних з відкритим вихідним кодом, активна розробка якої триває понад 35 років, завдяки чому вона заслужила

міцну репутацію надійності, надійності функцій і продуктивності. PostgreSQL має багато функцій, які допомагають розробникам щоб вони створювали додатки, адміністраторам — захищати цілісність даних і створювати відмовостійке середовище, а також допомагають вам керувати своїми даними, незалежно від того, наскільки великий чи малий набір даних. Приємна перевага, що PostgreSQL є безкоштовним і відкритим вихідним кодом , він дуже розширюваний. PostgreSQL притримується більшості стандартів стандарту SQL , якщо така відповідність не суперечить традиційним функціям або може призвести до невдалих архітектурних рішень. Багато функцій, необхідних стандарту SQL, підтримуються, хоча іноді з дещо відмінним синтаксисом або функціями. З часом можна очікувати подальших кроків у напрямку відповідності.

Для реалізації захищеного чат-месенджера, який включає функції реєстрації користувачів, авторизації з двофакторною автентифікацією та обмін зашифрованими повідомленнями, PostgreSQL надає потужний набір функцій, які забезпечать безпечну та ефективну роботу цього проекту.

PostgreSQL підтримує різноманітні типи даних, що є ключовими для зберігання та обробки інформації в чат-месенджері. Примітиви, такі як ціле, числове, рядкове та логічне, використовуються для базових полів, включаючи ідентифікатори користувачів, статуси та повідомлення. Структуровані типи, такі як дата і час, необхідні для збереження часу відправлення та отримання повідомлень. Крім того, підтримка документів у форматі JSON/JSONB дозволяє зберігати метадані повідомлень та налаштування користувачів у зручному і гнучкому форматі.

Забезпечення цілісності даних є критично важливим аспектом для будь-якого додатку, і PostgreSQL надає всі необхідні інструменти для цього. Використання обмежень унікальності та обов’язковості (UNIQUE, NOT NULL) гарантує, що важливі поля, такі як email та ідентифікатори повідомлень, будуть унікальними та заповненими. Первинні та іноземні ключі встановлюють зв’язки між таблицями, наприклад, між користувачами та повідомленнями, що забезпечує структурованість даних. Обмеження виключення використовуються для реалізації складних бізнес-правил, таких як обмеження на кількість спроб входу. Продуктивність і одночасність доступу до даних також є важливими факторами для чат-месенджера.

PostgreSQL підтримує різні типи індексування, включаючи B-дерево, багатостовпцеве, вирази та часткове індексування, що підвищує продуктивність пошуку та доступу до даних. Розширене індексування, таке як GIN та BRIN, полегшує роботу з великими обсягами даних, зокрема для пошуку повідомлень та повнотекстового пошуку. Multi-Version Concurrency Control (MVCC) забезпечує одночасний доступ багатьох користувачів без блокувань, що важливо для підтримки багатокористувацького середовища чату. Транзакції гарантують цілісність даних, забезпечуючи успішність або відкат всіх операцій при відправці та отриманні повідомлень.

Надійність і можливість аварійного відновлення є критичними для забезпечення безперебійної роботи чат-месенджера. Журналування з випереджальним записом (WAL) забезпечує збереження даних та можливість відновлення в разі збою. Реплікація, як асинхронна, так і синхронна, забезпечує безперебійну роботу і доступність даних, що важливо для безперервного доступу до чату. Відновлення на певний момент часу (PITR) дає можливість відновити базу даних до певного моменту, що важливо для захисту даних у випадку помилок або атак.

Безпека даних є одним з найважливіших аспектів для чат-месенджера. PostgreSQL забезпечує різні методи автентифікації користувачів, включаючи сучасні стандарти безпеки, такі як GSSAPI, SSPI, LDAP та SCRAM-SHA-256. Надійна система контролю доступу контролює доступ до даних на рівні таблиць і стовпців, а безпека на рівні стовпців і рядків забезпечує детальний контроль доступу до даних на рівні окремих полів і записів, що є важливим для захисту чутливої інформації. Багатофакторна аутентифікація підвищує безпеку, додаючи додаткові рівні автентифікації.

Розширюваність PostgreSQL дозволяє адаптувати систему під специфічні потреби проекту. Збережені функції та процедури використовуються для реалізації логіки на стороні сервера, зокрема для обробки та шифрування повідомлень. Підтримка процедурних мов, таких як PL/pgSQL, Perl та Python, дозволяє писати збережені процедури та тригери на різних мовах програмування. Іноземні оболонки даних дозволяють підключатися до інших баз даних або потоків даних, що корисно

для інтеграції з іншими системами.

Нарешті, підтримка інтернаціоналізації та текстового пошуку забезпечує коректне відображення та пошук тексту на різних мовах. Підтримка міжнародних наборів символів та сортування без урахування регістру та наголосу дозволяє забезпечити багатомовну підтримку. Повнотекстовий пошук дозволяє здійснювати швидкий і ефективний пошук повідомлень за текстом, що є важливим для користувачів чату.

Таким чином, PostgreSQL надає всі необхідні функції для створення безпечного та ефективного чат-месенджера, забезпечуючи високу продуктивність, надійність, безпеку та гнучкість.

Для реалізації захищеного чат-месенджера на основі PostgreSQL, булоприйнято рішення створити дві основні таблиці для зберігання даних:
«Користувачі» та «Повідомлення».

Таблиця «Користувачі»

В таблиці «Користувачі» було створено наступні поля:

- id

-email

-пароль

-секретний ключ

Таблиця «Повідомлення»

В таблиці «Повідомлення» було створено наступні поля:

-id

-відправник_id

- одержувач_id

-зашифроване_повідомлення

-ключ_шифрування

Опис полів

«Id» в таблиці «Користувачі» автоматично генерується базою даних PostgreSQL та є унікальним ідентифікатором для кожного користувача, що успішно зареєструвався. Поле «email» містить електронну адресу користувача, яку він ввів при реєстрації. Поле «пароль» зберігає захешований пароль користувача для

безпечного зберігання. Останнє поле, «секретний ключ», містить секретний ключ, згенерований під час реєстрації, який використовується для двофакторної автентифікації.

В таблиці «Повідомлення» поле «id» автоматично генерується PostgreSQL та є унікальним ідентифікатором для кожного повідомлення. Поле «відправник_id» містить ідентифікатор користувача, який відправив повідомлення, а поле «одержувач_id» - ідентифікатор користувача, який отримав повідомлення. Поле «зашифроване_повідомлення» зберігає саме повідомлення у зашифрованому вигляді, забезпечуючи конфіденційність інформації. Поле «ключ_шифрування» містить ключ, який використовується для шифрування та дешифрування повідомлення

2.3 Проектування інтерфейсу для чат-месенджера

В попередніх підрозділах були розроблені запропоновані алгоритми роботи системи, виходячи з того, чат-месенджер, буде складатися з 3-х основних сторінок, такихяк:

- сторінка реєстрації;
- сторінка авторизації;
- сторінка чату.

На сторінці реєстрації, користувач повинен мати змогу ввести особисті данідля реєстрації, а саме:

- email;
- пароль.
- ключ

Після введених даних, користувач повинен натиснути на кнопку «Зареєструватись», щоб розпочати процес реєстрації, тому дана кнопка також повинна бути в інтерфейсі користувача. Але якщо в користувача вже є обліковий запис, тому він повинен мати змогу «Авторизуватися».

Виходячи з попереднього аналізу було спроектовано макет інтерфейсу сторінки реєстрації (рис. 2.3.1).

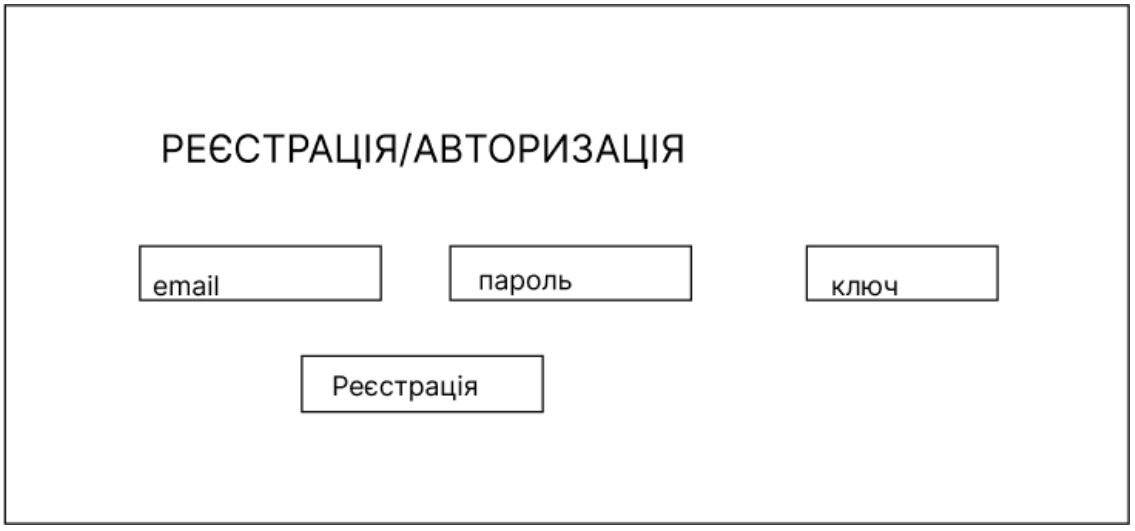


Рисунок 2.4 – спроектований макет інтерфейсу сторінки реєстрації/авторизації

Для реалізації захищеного чат-месенджера користувачі повинні мати можливість обирати співрозмовника та вводити свої повідомлення в текстове поле.

Після введення повідомлення користувач натискає кнопку «Відправити», яка запускає процес автоматичного шифрування повідомлення перед його відправкою.

Інтерфейс захищеного чат-месенджера має складатися з кількох ключових компонентів. Перш за все, список співрозмовників дозволяє користувачеві вибрати контакт для обміну повідомленнями. Це може бути список з іменами або псевдонімами користувачів, з якими можна вести діалог.

Центральним елементом є текстове поле для введення повідомлень. Тут користувач вводить текст свого повідомлення. Поруч з текстовим полем розташовується кнопка «Відправити», натискання на яку ініціює процес шифрування повідомлення. Зашифроване повідомлення потім надсилається обраному співрозмовнику. Важливим аспектом є те, що саме шифрування виконується автоматично і прозоро для користувача, забезпечуючи високий рівень безпеки та конфіденційності переданої інформації.

Для відображення повідомлень в інтерфейсі має бути спеціальне поле, яке показує як зашифровані, так і розшифровані повідомлення. Наприклад,зашифроване повідомлення "Швець В. В." виглядає так: `4c4ff5dd61db6332538ec0a406a5bab63d99b058618dd9a4`. Це повідомлення буде

відображатися в чаті до моменту його розшифрування.

На додаток до основних функцій, інтерфейс має включати кнопку «Ключ», яка дозволяє користувачу розшифрувати повідомлення. Це забезпечує додатковий рівень безпеки, гарантуючи, що дані користувача залишаться захищеними навіть після завершення роботи з месенджером.

Таким чином, макет інтерфейсу захищеного чат-месенджера виглядає наступним чином. Список співрозмовників розташований зліва або зверху, дозволяючи легко перемикатися між контактами. Поле повідомлень показує зашифровані повідомлення з зазначенням, хто їх надіслав. Нижче розташоване текстове поле для введення нових повідомлень і кнопка «Відправити». Завершує інтерфейс кнопка «Ключ», розташована в зручному місці для швидкого доступу.

Цей підхід до розробки інтерфейсу забезпечує простоту використання, захищеність переданих даних і зручність управління обліковими записами, роблячи чат-месенджер ефективним інструментом для безпечної комунікації.

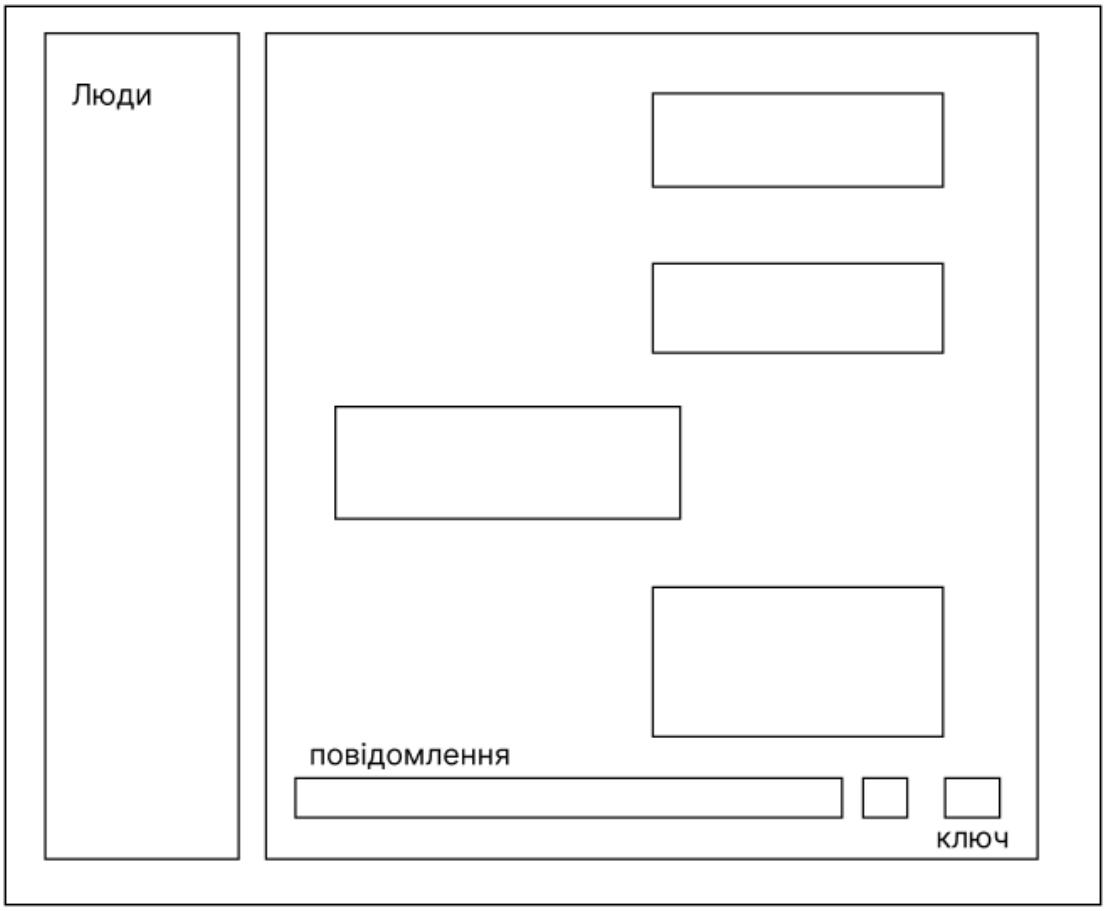


Рисунок 2.5 – спроектований макет інтерфейсу сторінки чату

2.4. Висновки до розділу 2

В даному розділі були розроблені блок-схеми алгоритмів роботи системи захищеного чат-месенджера. Основна увага приділена безпеці та захищеності комунікацій, включаючи алгоритми реєстрації, авторизації та управління сесіями користувачів. Алгоритми реєстрації забезпечують унікальність користувачів і захист їх даних, авторизація зберігає сесії, а алгоритм перевірки сесій гарантує доступ лише авторизованим користувачам. Алгоритм виходу забезпечує безпечне завершення сесії.

Також були розроблені алгоритми шифрування повідомлень для безпечної передачі даних. Результатом цього розділу є розроблені алгоритми функціонування захищеного чат-месенджера, спроектована база даних для зберігання інформації про користувачів і повідомлення, а також користувацький інтерфейс для зручного доступу до функцій месенджера, забезпечуючи його ефективність і безпеку.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ РОЗРОБЛЕНОГО ЧАТ-МЕСЕНДЖЕРА

В даному розділі буде обрано і обґрунтовано мову програмування, фреймворки, бібліотеки та середовище розробки. Також буде практично реалізовано чат месенджер. В результаті розробки, буде виконано тестування розробленого чат-месенджера,особливу увагу буде зосереджено на тестуванні його захищеності.

3.1. Обґрунтування вибору мови програмування, фреймворків, бібліотек та середовища розробки

Після розробки алгоритмів роботи чат-месенджера, та вибору і проектування бази даних, необхідно визначити за допомогою якої мови програмування буде реалізований весь функціонал програми який був спроектований раніше.

Python є ідеальним вибором для розробки захищеного чат-месенджера завдяки своїй простоті, гнучкості та широкому набору бібліотек і фреймворків. Його читабельний синтаксис робить його доступним для розробників будь-якого рівня, що сприяє швидкій розробці та легкому обслуговуванню коду. Крім того, Python має велику активну спільноту, яка надає численні ресурси для підтримки та навчання, що робить його ще привабливішим для проектів будь-якого масштабу.

Django — це високорівневий веб-фреймворк Python, який заохочує швидку розробку та чистий, прагматичний дизайн. Створений досвідченими розробниками, він впорається з більшою частиною клопоту веб-розробки, тому ви можете зосередитися на написанні свого додатка без потреби винаходити колесо. Це безкоштовно та з відкритим кодом.

Одним з найкращих фреймворків для створення захищеного чат-месенджера є Django. Django є потужним і універсальним фреймворком для розробки веб-додатків, який забезпечує високу продуктивність, безпеку та масштабованість. Він має вбудовану систему автентифікації та авторизації, яка включає засоби захисту

від найпоширеніших веб-загроз, таких як CSRF та SQL injection, що є критично важливим для будь-якого захищеного додатку.

Django забезпечує безпеку даних завдяки своїм вбудованим механізмам захисту. Він автоматично захищає від багатьох поширених вразливостей, таких як міжсайтові підробки запитів (CSRF), міжсайтові скриптові атаки (XSS) та SQL-ін'єкції. Ці функції є надзвичайно важливими для захищеного чат-месенджера, оскільки вони допомагають захистити конфіденційні дані користувачів.

Продуктивність і масштабованість є ще однією важливою перевагою Django. Він використовує концепцію "батареї включені", надаючи широкий набір інструментів та функцій, що дозволяє розробникам зосередитися на логіці додатку, а не на налаштуванні базової інфраструктури. Django ORM (Object-Relational Mapping) забезпечує легку роботу з базами даних, такими як PostgreSQL, що дозволяє швидко і ефективно виконувати запити та обробляти дані.

Завдяки своїй модульній архітектурі, Django дозволяє швидко створювати та розширювати функціональність додатку. Вбудована адміністративна панель дозволяє легко керувати користувачами та контентом, що спрощує обслуговування та адміністрування чат-месенджера. Велика спільнота та багата документація допомагають швидко знаходити рішення для будь-яких технічних питань.

Отже, Django є найкращим вибором для розробки захищеного чат-месенджера завдяки своїм потужним інструментам для забезпечення безпеки, продуктивності та масштабованості. Він надає всі необхідні засоби для швидкої розробки та ефективного управління даними, що робить його ідеальним рішенням для проектів будь-якої складності. Завдяки Django можна створити надійний та безпечний чат-месенджер, який відповідає найвищим стандартам сучасних веб-додатків.

Для реалізації двофакторної автентифікації, було надано перевагу бібліотеці Speakeasy, так як вона є досить одним із найкращих варіантів для генерації OTP (One-Time Password) в застосунках, які використовують двофакторну аутентифікацію.

Ще однією перевагою Speakeasy є його простота у використанні та бібліотека надає зручний API для генерації та перевірки OTP, що дозволяє впроваджувати двофакторну автентифікацію.

Для клієнтської частини було використано бібліотеку React. React - це популярна JavaScript бібліотека для створення інтерфейсів користувача. Вона дозволяє створювати швидкі та інтерактивні веб-додатки.

Задля безпеки було обрано бібліотеку Cryptography. Ця бібліотека призначена для забезпечення криптографічного захисту даних. Cryptography включає алгоритми для шифрування, цифрового підпису та інших криптографічних операцій.

Cryptography включає як високорівневі recipes, так і низькорівневі інтерфейси для загальних криптографічних алгоритмів, таких як симетричні шифри, дайджести повідомлень і функції виведення ключів. Наприклад, щоб зашифрувати щось за допомогою високорівневого recipes симетричного шифрування криптографії.

Для тестування та розгортання було обрано Pytest. Це потужний інструмент для тестування Python-додатків. Він забезпечує зручний спосіб написання і запуску тестів. Pytest є найкращим вибором для тестування. Тому, що pytest є одним з найпопулярніших інструментів для тестування в Python, і він ідеально підходить для розробки вашого захищеного чат-месенджера з кількох причин. По-перше, pytest має простий і зрозумілий синтаксис, що дозволяє швидко писати тести навіть новачкам у програмуванні. Це робить процес тестування доступним і зрозумілим для широкого кола розробників, що є важливим для підтримки якості коду. Основними перевагами pytest є те, щлпідтримує потужні фічі, такі як фікстури, параметризовані тести та плагіни, які спрощують тестування складних сценаріїв.

Pytest є ідеальним вибором для тестування захищеного чат-месенджера завдяки своєму простому синтаксису, потужним можливостям та гнучкості. Використовуючи pytest разом з pytest-django, можна легко створювати та запускати тести, що забезпечить високу якість додатку. Це дозволить виявляти та виправляти помилки на ранніх етапах розробки, що в свою чергу підвищує надійність та безпеку чат-месенджера.

Для програмної реалізації захищеного корпоративного файлообмінника з двофакторною автентифікацією, було обрано Visual Studio Code, як середовище розробки.

Visual Studio Code (VS Code) — це спрощений, але потужний редактор вихідного коду, який запускається на комп'ютері й доступний для Windows, macOS

і Linux. Вона підтримує JavaScript, TypeScript і Node.js і має багату екосистему розширень для інших мов (наприклад, C++, C#, Java, Python, PHP і Go) і середовищ виконання (наприклад, .NET і Unity). Для отримання додаткових відомостей див. Початок роботи з VS Code

VS Code надає додаткові можливості завдяки розширенням. Розширення VS Code можуть вносити додаткові функції до наявного набору. Після випуску цієї функції з’явилася можливість використовувати розширення VS Code для роботи із порталами Power Apps.

Перевагами Visual Studio Code є те що він безкоштовний для будь яких користувачів які працюють на операційних системахWindows, Linux та MacOS. Ще однією перевагою є те що він підтримує безліч мов програмування, таких як JavaScript і всі його варіації, Python, Java, C#, PHP. Крім цих основних мов, VS Code також підтримує множину інших мов програмування, включаючи Ruby, Go,Swift, Kotlin та багато інших. Багатофункціональність та гнучкість VS Code роблять його відмінним інструментом для розробки.

Також перевагою Visual Studio Code є те що в ньому є безліч розширень під різноманітні завдання, які дозволяють розробникам налаштувати середовище розробки під себе. Можна встановити розширення для роботи з конкретними мовами програмування, фреймворками, системами контролю версій та багато іншого.

Крім того, VS Code має вбудовану підтримку Git, що робить спільну роботунад проектами більш зручною. Завдяки інтеграції з Git можна легко відстежуватизміни проекту, та співпрацювати з іншими розробниками безпосередньо з середовища розробки.

Таким чином було повністю підготовлено все до програмної реалізації захищеного корпоративного файлообмінника, який буде розроблено в наступному розділі даної дипломної роботи.

3.2. Програмна реалізація захищеного корпоративного файлообмінника з двофакторною автентифікацією

Отже потрібно налаштувати середовище захищеного чат-месенджера з шифруванням повідомлень та ключів для зашифрованих повідомлень. Використовувати Python для серверної частини з фреймворком Django та Django Channels для WebSocket зв'язку, а також React для клієнтської частини.

Перш за все було встановлено Python, PostgreSQL, Redis. Та створено і активовано віртуальне середовище Python для проекту. Встановлено Django та необхідні пакети.

Створення Django проекту та додатку:

```
Bash
django-admin startproject chat_project
cd chat_project
django-admin startapp chat
```

Налаштування бази даних та конфігурація PostgreSQL. Встановлюємо PostgreSQL і створюємо базу даних. Також налаштовуємо «settings.py» для використання PostgreSQL.

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'chat_db',
        'USER': 'your_db_user',
        'PASSWORD': 'your_db_password',
        'HOST': 'localhost',
        'PORT': '',
    }
}
```

Проводиться налаштування Django Channels. Розпочнемо з додавання Channels до проекту.

```
INSTALLED_APPS = [
    ...
    'channels',
    'chat',
]
ASGI_APPLICATION = 'chat_project.asgi.application'
CHANNEL_LAYERS = {
    'default': {
        'BACKEND': 'channels_redis.core.RedisChannelLayer',
        'CONFIG': {
```

```
            "hosts": [('127.0.0.1', 6379)],
        },
    },
}
```

Створення «asgi.py`»

```
import os
from  django.core.asgi  import  get_asgi_application
from channels.routing import ProtocolTypeRouter, URLRouter
from channels.auth import AuthMiddlewareStack
import chat.routing

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'chat_project.settings')

application = ProtocolTypeRouter({
    "http": get_asgi_application(),
    "websocket":  AuthMiddlewareStack(
        URLRouter(
            chat.routing.websocket_urlpatterns
        )
    ),
})
```

Налаштування «routing.py» у додатку chat

```
from django.urls import re_path
from . import consumers

websocket_urlpatterns = [
    re_path(r'ws/chat/(?P<room_name>\w+)/$', consumers.ChatConsumer.as_asgi()),
]
```

Далі Створюємо Consumer для WebSockets з шифруванням повідомлень. За допомогою бібліотеки «cryptography» для шифрування та розшифрування повідомлень в вашому «ChatConsumer». У цьому компоненті React ChatRoom.js ми підключаємося до WebSocket сервера Django через socket.io-client бібліотеку. Під час підключення до WebSocket, ми отримуємо ключ шифрування з сервера через REST API. Повідомлення шифруються перед відправкою та розшифровуються при отриманні, використовуючи бібліотеку CryptoJS.

```
import json
from channels.generic.websocket import AsyncWebsocketConsumer
from cryptography.fernet import Fernet
```

```
class ChatConsumer(AsyncWebsocketConsumer):
    def init (self, *args, **kwargs):
        super(). init (*args, **kwargs)
        self.room_name = None
        self.room_group_name = None
        self.key = Fernet.generate_key()
        self.cipher_suite = Fernet(self.key)

    async def connect(self):
        self.room_name = self.scope['url_route']['kwargs']['room_name']
        self.room_group_name = f'chat_{self.room_name}'

        await self.channel_layer.group_add(
            self.room_group_name,
            self.channel_name
        )

        await self.accept()

    async def disconnect(self, close_code):
        await  self.channel_layer.group_discard(
            self.room_group_name,
            self.channel_name
        )

    async def receive(self, text_data):
        encrypted_message = json.loads(text_data)['message']
        decrypted_message = self.cipher_suite.decrypt(bytes(encrypted_message, 'utf-8')).decode('utf-8')

        await self.channel_layer.group_send(
            self.room_group_name,
            {
                'type': 'chat_message',
                'message': decrypted_message
            }
        )

    async def chat_message(self, event):
        message = event['message']
        encrypted_message = self.cipher_suite.encrypt(bytes(message, 'utf-8')).decode('utf-8')
```

```
        await self.send(text_data=json.dumps({
            'message': encrypted_message
        }))
```

Тут повідомлення шифрується перед відправкою через WebSocket і розшифровується при отриманні. Ключ для шифрування «self.key» генерується при підключенні користувача.

Наступним етапом є налаштування REST API. Створення моделі та серіалізаторів, а ткож зберігання повідомлень і серіалізатор для взаємодії з ними через REST API.

```
from django.db import models

class Message(models.Model):
    content = models.TextField()
    timestamp = models.DateTimeField(auto_now_add=True)
    sender = models.ForeignKey(User, on_delete=models.CASCADE)

# chat/serializers.py
from rest_framework import serializers
from .models import Message

class MessageSerializer(serializers.ModelSerializer):
    class Meta:
        model = Message
        fields = ['id', 'content', 'timestamp', 'sender']
```

Далі розписана клієнтська частина чат месенджера а саме робота створення «React» додатку. Далі прописано код ініціалізації проекту

```
npx create-react-app chat-client
cd chat-client
```

Далі відбувається встановлення залежностей та налаштування WebSockets та шифрування. Також створення з'єднання та компонентів React.

```
bash
npm install axios redux react-redux socket.io-client
// src/components/ChatRoom.js
import React, { useState, useEffect } from 'react';
import socketIOClient from 'socket.io-client';
```

```
import CryptoJS from 'crypto-js';

const ENDPOINT = 'http://localhost:8000'; // URL сервера Django

const ChatRoom = ({ roomName }) => {
  const [messages, setMessages] = useState([]);
  const [message, setMessage] = useState('');
  const [socket, setSocket] = useState(null);
  const [key, setKey] = useState(null); // ключ шифрування

  useEffect(() => {
    const fetchKey = async () => {
      const response = await fetch(`${ENDPOINT}/api/chat/key/`);
      const data = await response.json();
      setKey(data.key);
    };

    fetchKey();
  }, []);

  useEffect(() => {
    if (!key) return;

    const newSocket = socketIOClient(ENDPOINT, {
      query: { roomName },
      transports: ['websocket'],
    });

    newSocket.on('connect', () => {
      console.log('Connected to WebSocket');
    });

    newSocket.on('disconnect', () => {
      console.log('Disconnected from WebSocket');
    });

    newSocket.on('message', (encryptedMessage) => {
      const decryptedBytes = CryptoJS.AES.decrypt(encryptedMessage, key);
      const decryptedMessage = decryptedBytes.toString(CryptoJS.enc.Utf8);
      setMessages(prevMessages => [...prevMessages, decryptedMessage]);
    });
```

```
    setSocket(newSocket);

    return () => newSocket.disconnect();
  }, [key, roomName]);

  const sendMessage = (event) => {
    event.preventDefault();
    if (!message.trim()) return;

    const encryptedMessage = CryptoJS.AES.encrypt(message, key).toString();
    socket.emit('message', encryptedMessage);
    setMessage('');
  };

  return (
    <div>
      <h2>Chat Room: {roomName}</h2>
      <div>
        {messages.map((msg, index) => (
          <p key={index}>{msg}</p>
        ))}
      </div>
      <form onSubmit={sendMessage}>
        <input
          type="text"
          value={message}
          onChange={(e) => setMessage(e.target.value)}
        />
        <button type="submit">Send</button>
      </form>
    </div>
  );
};
export default ChatRoom;
```

Далі відбувається шифрування повідомлень та ключів. Використання бібліотеки CryptoJS для шифрування/розшифрування. У веб-додатку React використовується CryptoJS для AES шифрування/розшифрування. Ключ шифрування отримується з сервера за допомогою REST API, а потім використовується для зашифрування та розшифрування повідомлень, які

передаються через WebSocket.

Після чого проводиться інтеграція з сервером Django та з'єднання з сервером через REST API та WebSocket. REST API: Використовується як axios для звернень до REST API Django для отримання ключа шифрування і для взаємодії з базою даних. А WebSocket: Встановлюється з'єднання через socket.io-client з WebSocket сервером Django Channels для реального часу обміну повідомленнями.

3.3 Тестування розробленого чат-месенджера

Заключним етапом в розробці захищеного чат-месенджера, є його тестування.

Для початку проведення тестування розробленого чат-месенджера, було запущено сервер. Сервер працює і базу даних було підключено правильно.

Доступ неавторизованим користувачам до захищеної сторінки заборонено,при натиснені на кнопку «ОК», автоматично відкривається сторінка авторизації. Після чого було проведено тестування реєстрації, для цього на відповідній сторінці, було введено дані для реєстрації з електронною адресою, якої немає в списку дозволених для реєстрації

Алгоритм перевірки дозволених електронних адрес, спрацював чудово, і не дозволив зареєструватися даному користувачу.

Після цього було проведено тестування реєстрації з електронною адресоюяка є в списку дозволених.

Користувач який дозволений для реєстрації, успішно зареєстрований, йомубуло надане відповідне повідомлення з токеном для двофакторної автентифікації.

Для перевірки чи дійсно користувач зареєструвався, було перевірено базу даних зареєстрований користувач, збереження в базі даних користувача дійсно було успішно зареєстровано, з збереженням його імені,

електронної адреси, захешованого пароля, і токеном для двофакторної автентифікації.

Після чого було проведено тестування на повторну реєстрацію зареєстрованого користувача, для цього було введено ту ж саму електронну адресу яка вже зареєстрована.

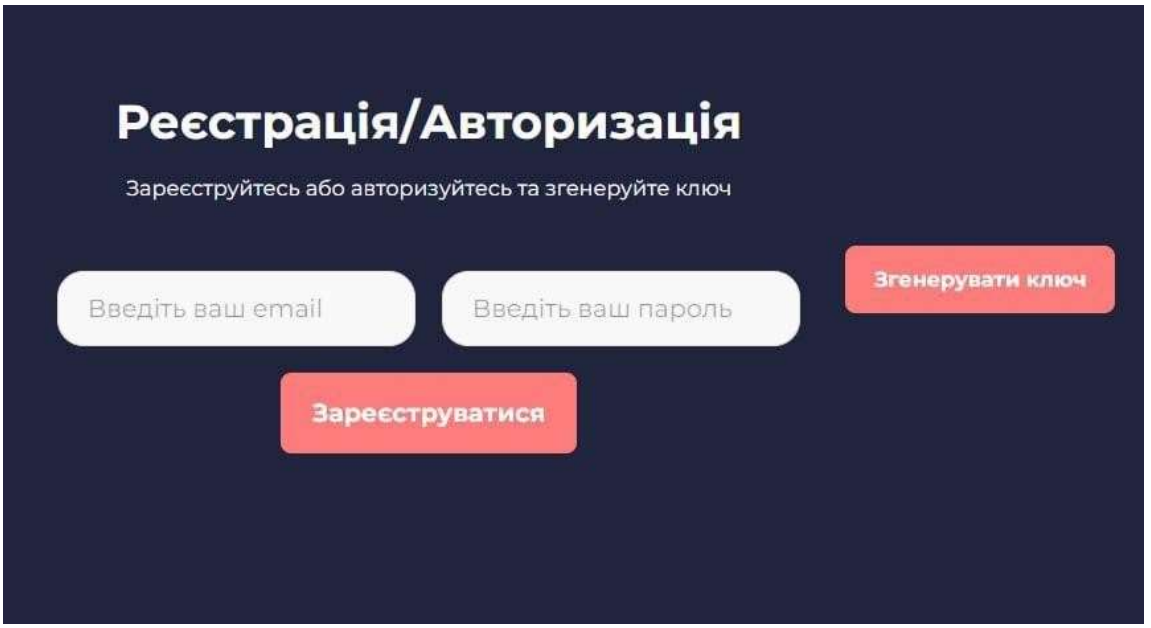


Рисунок 3.1 – зовнішній вигляд розробленого захищеного корпоративного файлообмінника з двофакторною автентифікацією

Користувача який уже зареєстрований не допущено до повторної реєстрації, йому виведено відповідне повідомлення.

Після чого було здійснено спробу авторизації раніше зареєстрованого користувача, з використанням даного ключа

В результаті користувач успішно авторизувався, також йому було виведено відповідне повідомлення. Після успішної авторизації користувачу було створено сесію.

Також було здійснено спробу генерації ключа, яка також хавершилась успішно.

Після успішної авторизації автоматично відкривається захищена сторінка чат месенджера, де зберігаються самі файли.

Рисунок 3.2 – Зовнішній вигляд розробленого чат-месенджера

Наступним було проведено тестування надсилання повідомлень. Повідомлення добре друкуються та відображаються на екрані. А після їх відправлення автоматично шифруються.

З іншої сторони було проведено тестування розшифрування повідомлень отримувачем з іншого облікового запису, яке теж завершилось успішно.

Аналізуючи результати проведених тестувань, можна дійти висновку що розробленийчат-месенжер, працює добре, забезпечуючи безпечний обмін повідомленями..

3.4. Висновок до розділу 3

В даному розділі було обрано і обґрунтовано мову програмування, фреймворки, бібліотеки та середовище розробки. Було програмно реалізовано чат месенджер, а саме було розроблено як клієнтську, так і серверну частини. Також було проведено тестування розробленого чат-месенджера, та сформовано висновок за результатами тестувань.

В результаті було отримано хороший продукт, для обміну повідомленнями.

Який буде забезпечувати безпеку як персональних даних так і самих повідомленнь.

.

ВИСНОВКИ

Отже, в ході написання даної бакалаврської роботи, було здійснено огляд і аналіз існуючих на ринку, сучасних чат-месенджерів, проаналізовано їх переваги та недоліки, сформовано відповідні висновки. Також було визначено і сформовано поняття шифрування.

Було розроблено запропоновані алгоритми роботи системи захищеного чат-месенджера, а саме алгоритм реєстрації, авторизації та роботи самого чату. Також було обрано і спроектовано базуданих для шифрування повідомлеень користувачів. Крім того було спроектовано користувацький інтерфейс, на основі аналізу розроблених алгоритмів роботи захищеногочат-месенджера.

Також було обрано і обґрунтовано мову програмування, фреймворки, бібліотеки, та середовище розробки, за допомогою яких буде програмно реалізовано корпоративний файлообмінник. Крім того, було програмно реалізовано захищений корпоративний файлообмінник з двофакторною автентифікацією, а саме було розроблено як клієнтську, так і серверну частини, з зручним інтерфейсом. Також було проведено тестування розробленогозахищеного чат-месенджера. В результаті проведених тестувань, було сформовано відповідні висновки.

В результаті виконаної роботи було досягнуто запланованого результату, а саме, отримано повноцінний і готовий до роботи, захищений чат-месенджер, для забезпечення захищеного обміну повідомленями.

ПЕРЕЛІК ПОСИЛАНЬ

1. Square. How 5 InnovativeBusinessesAreUsingChatbots. [Електронний ресурс] – Режим доступу. – URL: https://squareup.com/townsquare/how–5–innovative–businesses–are–using–chatbots (Дата звернення: 15.03.2024).

2. *У пошуках безпечного месенджера*. (n.d.) retrieved June 13, 2024 URL:https://kr-labs.com.ua/blog/top-secure-and-privacy-messaging-apps/(Дата звернення: 15.03.2024).

3. Джерело: URL:https://kr-labs.com.ua/blog/bezpeka-i-anonimnist-v-telegram(Дата звернення: 15.03.2024).

4. ФЕДОРОВ: НАЙКРАЩИЙ ТА НАЙЗАХИЩЕНІШИЙ МЕСЕНДЖЕР В УКРАЇНІ – SIGNAL 30.10.2023, 12:29 ВАЛЕНТИНА ТРОЯН URL:https://imi.org.ua/news/fedorov-najkrashhyj-ta-najzahyshhenishyj-mesendzher-v-ukrayini-signal-i56478(Дата звернення: 18.03.2024).

5. Застосунки Телеграм URL:https://telegram.org/apps?setln=uk(Дата звернення: 18.03.2024).

6. ЗастосунокFacebook_MessengerURL: https://uk.wikipedia.org/wiki/Facebook_Messenger(Дата звернення: 19.04.2024).

7. Марія Бровінська “Стартап” 26 жовтня 2022, 08:08 URL:https://dev.ua/news/dober(Дата звернення: 15.03.2024).

8. Підручник з криптографії URL:https://www.tutorialspoint.com/cryptography/index.htm(Дата звернення: 15.03.2024).

9. Що таке шифрування та як воно працює травень 2023 URL:https://www.kingston.com/ua/blog/data-security/what-is-encryption(Дата звернення: 15.03.2024).

10. АНАЛІЗ МЕТОДІВ КРИПТОГРАФІЧНОГО ЗАХИСТУ ІНФОРМАЦІЇ Савич Ю.О., к.т.н., доцент Вовк Р.Б., к.т.н., доцент Пасєка М.С.

11. Основні поняття авторизації. URL: https://auth0.com/intro-to-iam/what-is-authorization (Дата звернення: 15.03.2024).

12. Відмінність між авторизацією і автентифікацією. URL: https://www.sailpoint.com/identity-library/difference-between-authentication-and-

authorization/#:~:text=Simply%20put%2C%20authentication%20is%20the,people%20can%20come%20on%20board (Дата звернення: 08.04.2024).

13. Контроль доступу. URL: https://www.ibm.com/docs/en/wca/3.0.0?topic=security-authentication-versus-access-control (Дата звернення: 08.04.2024).

14. Важливість використання автентифікації в організаціях. URL: https://nordvpn.com/uk/blog/what-is-user-authentication/#:~:text=User%20authentication%20is%20important%20because,both%20private%20and%20business%20accounts (Дата звернення: 09.04.2024).

15. Основні завдання автентифікації. URL: https://www.ibm.com/docs/en/atlas-policy-suite/6.0.3?topic=tasks-authentication-task (Дата звернення: 03.04.2024).

16. Основні поняття, і визначення двофакторної автентифікації. URL: https://www.techtarget.com/searchsecurity/definition/two-factor-authentication (Дата звернення: 03.05.2024).

17. Фактор знань, в двофакторній автентифікації. URL: https://www.ibm.com/topics/2fa#:~:text=In%20most%202FA%20implementations%2C%20a,security%20questions%20are%20also%20typical (Дата звернення: 22.04.2024).

18. Фактор володінь в двофакторній автентифікації. URL: https://www.techtarget.com/searchsecurity/definition/two-factor-authentication#:~:text=A%20possession%20factor%20is%20something,in%20the%20user's%20physical%20self (Дата звернення: 22.04.2024).

19. Фактор приналежності в двофакторній автентифікації. URL: https://www.paidmembershipspro.com/two-factor-authentication/ (Дата звернення: 18.03.2024).

20. Фактор часу в двофакторній автентифікації. URL: https://help.peopleforce.io/en/articles/6628189-two-factor-authentication (Дата звернення: 12.04.2024).

21. Метод двофакторної автентифікації на основі одноразових кодів підтвердження. URL: https://www.kaspersky.com/blog/types-of-two-factor-authentication/48446/ (Дата звернення: 03.04.2024).

22. Метод двофакторної автентифікації за допомогою спеціального тимчасового, одноразового коду з програми автентифікації. URL: https://www.simplilearn.com/tutorials/cyber-security-tutorial/what-is-two-factor-authentication (Дата звернення: 26.04.2024).

23. Бібліотека React URL: https://uk.legacy.reactjs.org/ (Дата звернення: 26.04.2024).

24. Бібліотека cryptographi URL: https://cryptography.io/en/latest/ (Дата звернення: 28.04.2024).

25. Розробка інтерфейсу месенджера URL: https://merehead.com/ua/blog/messenger-development/ (Дата звернення: 28.04.2024).

26. Розробка месенджера на Python URL: https://drukarnia.com.ua/articles/rozrobka-primitivnogo-mesendzhera-na-python-za-dopomogoyu-biblioteki-socket-oF5_P (Дата звернення: 28.04.2024).

27. Тестування чат месенджера на Python URL: https://foxminded.ua/chat-boty-na-python/ (Дата звернення: 28.04.2024).

28. Django — це високорівневий веб-фреймворк Python. URL: https://www.djangoproject.com (Дата звернення: 28.04.2024).

29. PostgreSQL: найдосконаліша у світі реляційна база даних з відкритим кодом. URL: https://www.postgresql.org/ (Дата звернення: 26.04.2024).

30. Метод двофакторної автентифікації на основі біометричних даних. URL: https://jumpcloud.com/blog/biometric-totp-2fa#:~:text=Biometric%202FA%2C%20or%20biometric%20authentication,depresses%20keys%20on%20their%20keyboard (Дата звернення: 13.04.2024).

ДОДАТКИ

Додаток А. Технічне завдання

Вінницький національний технічний університет
Факультет менеджменту та інформаційної безпеки
Кафедра менеджменту та безпеки інформаційних систем

ЗАТВЕРДЖУЮ
Голова секції "Управління інформаційною безпекою" кафедри МБІС
д.т.н., професор
_____Юрій ЯРЕМЧУК
" 22 " березня 2024 р.

ТЕХНІЧНЕ ЗАВДАННЯ

до бакалаврської дипломної роботи на тему:
Розробка захищеного корпоративного файлообмінника з двофакторною автентифікацією

08-72.БДР.012.00.070 ТЗ

Керівник бакалаврської дипломної роботи
к.т.н., доцент
Яремчук Ю.Є.
"____" _____

Вінниця – 2024 р.

1. Найменування та область застосування

Розробка захищеного чат-месенджера

Область застосування: захист інформаційних ресурсів від несанкціонованого доступу.

2.Підстава для розробки

Розробка виконується на основі наказу ректора ВНТУ № 80 від 11.03.2024р.

3. Мета та призначення розробки

3.1 Мета розробки: розробка захищеного чат-месенджера, з метою захисту інформації від несанкціонованого доступу.

3.2 Призначення: розроблений програмний засіб виконує захист від несанкціонованого доступу.

4. Джерела розробки

4.1. Ахрамович В. М. Ідентифікація й аутентифікація, керування доступом // Сучасний захист інформації. – 2016. №4.– С. 47-51.

4.2. Бурячок В.Л. Політика інформаційної безпеки: підручник. / В.Л.Бурячок, Р.В.Грищук, В.О.Хорошко / За заг. ред. докт. техн. наук, проф. В.О. Хорошка. – К.: ПВП «Задруга», 2014. – 222 с.

4.3. Єсін В.І. Безпека інформаційних систем і технологій / В.І.Єсін, О.О. Кузнецов, Л.С. Сорока. – Харків: ХНУ імені В.Н. Каразіна, 2013. – 632 с.

5. Вимоги до програми

5.1 Вимоги до функціональних характеристик:

5.1.1 Програмний засіб повинен мати зручний, легкий у використанні інтерфейс користувача;

5.1.2 Реалізація методу не повинна вимагати спеціальних ліцензійних програмних додатків;

5.1.3 Програмний засіб повинен виконувати процес автентифікації користувачів у системі.

5.2 Вимоги до надійності:

5.2.1 Програмний засіб повинен працювати без помилок, у випадку виникнення критичних ситуацій необхідно передбачити виведення відповідних повідомлень;

5.2.3 Програмний засіб повинен виконувати свої функції.

5.3 Вимоги до складу і параметрів технічних засобів:

– процесор – Pentium 1500 МГц і подібні до них;

– оперативна пам’ять – не менше 512 Mb;

– середовище функціонування – операційна система сімейство Windows і MacOS;

– вимоги до техніки безпеки при роботі з програмою повинні відповідати існуючим вимогам та стандартам з техніки безпеки при користуванні комп’ютерною технікою.

6. Вимоги до програмної документації

6.1 Обов’язкова поетапна інструкція для майбутніх користувачів і тестування розробленого продукту, наведена у пункті 3.3

7. Вимоги до технічного захисту інформації

7.1 Необхідно забезпечити захист розроблюваного програмного засобу від несанкціонованого використання.

7.2 Неможливість отримання доступу незареєстрованих користувачів до інформаційних ресурсів.

8. Техніко-економічні показники

8.1 Цінність результатів використання даного проекту повинна перевищувати витрати на його реалізацію.

8.2 Має бути реалізований таким чином, щоб підходити для використання широкого загалу.

9. Стадії та етапи розробки

№ з/п	Назва етапів бакалаврської дипломної роботи	Початок	Закінчення
1.	Визначення напрямку бакалаврської роботи, формулювання теми	23.03.2024	28.03.2024
2.	Аналіз предметної області обраної теми	02.04.2024	08.04.2024
3.	Розробка запропонованих алгоритмів роботи системи	09.04.2024	15.04.2024
4.	Проектування бази даних	17.04.2024	21.04.2024
5.	Проектування користувацького інтерфейсу	26.04.2024	29.04.2024
6.	Програмна реалізація захищеного корпоративного файлообмінника з двофакторною автентифікацією	01.05.2024	14.05.2024
7.	Тестування розробленого захищеного корпоративного файлообмінника з двофакторною автентифікацією	15.05.2024	19.05.2024
8.	Попередній захист бакалаврської дипломної роботи	23.05.2024	23.05.2024
9.	Виправлення, уточнення, корегування бакалаврської дипломної роботи	24.05.2024	29.05.2024
10.	Захист бакалаврської дипломної роботи	12.06.2024	17.06.2024

10. Порядок контролю та прийому

10.1 До приймання бакалаврської дипломної роботи надається:

– ПЗ до бакалаврської дипломної роботи;

- демонстрація результату бакалаврської дипломної роботи;
- презентація;
- відзив керівника роботи;
- відзив рецензента

Технічне завдання до виконання прийняв_____ Швець В.В.

Додаток Б. Серверний код

```
# secure_chat/settings.py

INSTALLED_APPS = [
    ...
    'rest_framework',
    'channels',
    'chat',
]


MIDDLEWARE = [
    ...
]

ROOT_URLCONF = 'secure_chat.urls'

TEMPLATES = [
    ...
]

WSGI_APPLICATION = 'secure_chat.wsgi.application'
ASGI_APPLICATION = 'secure_chat.asgi.application'

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'secure_chat_db',
        'USER': 'your_db_user',
        'PASSWORD': 'your_db_password',
        'HOST': 'localhost',
        'PORT': '5432',
    }
}

AUTH_PASSWORD_VALIDATORS = [
    ...
]

LANGUAGE_CODE = 'en-us'
TIME_ZONE = 'UTC'
USE_I18N = True
USE_L10N = True
USE_TZ = True

STATIC_URL = '/static/'

# Channels specific settings
CHANNEL_LAYERS = {
    'default': {
        'BACKEND': 'channels_redis.core.RedisChannelLayer',
        'CONFIG': {
            "hosts": [('127.0.0.1', 6379)],
        },
    },
}

# secure_chat/asgi.py

import os
from channels.auth import AuthMiddlewareStack
```

```
from channels.routing import ProtocolTypeRouter, URLRouter
from django.core.asgi import get_asgi_application
import chat.routing

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'secure_chat.settings')

application = ProtocolTypeRouter({
  "http": get_asgi_application(),
  "websocket":  AuthMiddlewareStack(
    URLRouter(


      chat.routing.websocket_urlpatterns
    )
  ),
})

# chat/routing.py

from django.urls import path
from . import consumers

websocket_urlpatterns = [
  path('ws/chat/<str:room_name>/', consumers.ChatConsumer.as_asgi()),
]

# chat/consumers.py

import json
from channels.generic.websocket import AsyncWebsocketConsumer
from cryptography.fernet import Fernet

class ChatConsumer(AsyncWebsocketConsumer):
  async def connect(self):
    self.room_name = self.scope['url_route']['kwargs']['room_name']
    self.room_group_name = f'chat_{self.room_name}'

    await self.channel_layer.group_add(
      self.room_group_name,
      self.channel_name
    )

    await self.accept()

  async def disconnect(self, close_code):
    await  self.channel_layer.group_discard(
      self.room_group_name,
      self.channel_name
    )

  async def receive(self, text_data):
    text_data_json = json.loads(text_data)
    encrypted_message = text_data_json['message']

    await self.channel_layer.group_send(
      self.room_group_name,
      {
        'type': 'chat_message',
        'message': encrypted_message
      }
    )

  async def chat_message(self, event):
    message = event['message']
```

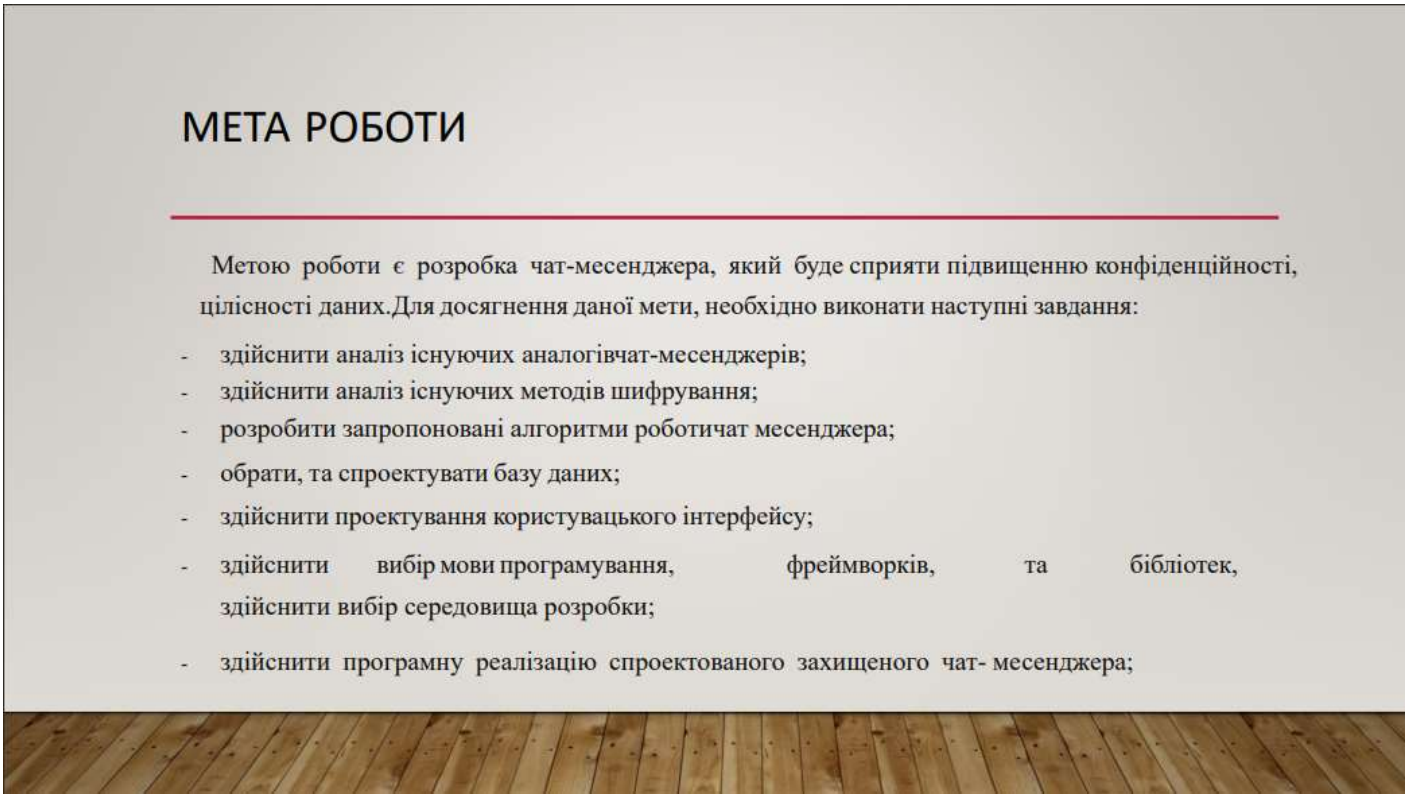
```
        await self.send(text_data=json.dumps({
            'message': message
        }))

# secure_chat/urls.py

from django.contrib import admin
from django.urls import path, include


urlpatterns = [
    path('admin/', admin.site.urls),
    path('api/chat/', include('chat.urls')),
]

# chat/views.py

from rest_framework.views import APIView
from rest_framework.response import Response
from rest_framework import status
from cryptography.fernet import Fernet

 class KeyView(APIView):
     def get(self, request):
        key = Fernet.generate_key().decode()
        return Response({'key': key}, status=status.HTTP_200_OK)

# chat/urls.py

from django.urls import path
from .views import KeyView

urlpatterns = [
    path('key/', KeyView.as_view(), name='key'),
]
```

Додаток В. Клієнтський код завантаження файлів

```
// src/components/ChatRoom.js

import React, { useState, useEffect } from 'react';
import socketIOClient from 'socket.io-client';
import CryptoJS from 'crypto-js';

const ENDPOINT = 'http://localhost:8000'; // URL сервера Django

const ChatRoom = ({ roomName }) => {
  const [messages, setMessages] = useState([]);
  const [message, setMessage] = useState('');
  const [socket, setSocket] = useState(null);
  const [key, setKey] = useState(null); // ключ шифрування

  useEffect(() => {
    const fetchKey = async () => {
      const response = await fetch(`${ENDPOINT}/api/chat/key/`);
      const data = await response.json();
      setKey(data.key);
    };

    fetchKey();
  }, []);

  useEffect(() => {
    if (!key) return;

    const newSocket = socketIOClient(ENDPOINT, {
      query: { roomName },
      transports: ['websocket'],
    });

    newSocket.on('connect', () => {
      console.log('Connected to WebSocket');
    });

    newSocket.on('disconnect', () => {
      console.log('Disconnected from WebSocket');
    });

    newSocket.on('message', (encryptedMessage) => {
      const decryptedBytes = CryptoJS.AES.decrypt(encryptedMessage, key);
      const decryptedMessage = decryptedBytes.toString(CryptoJS.enc.Utf8);
      setMessages(prevMessages => [...prevMessages, decryptedMessage]);
    });

    setSocket(newSocket);

    return () => newSocket.disconnect();
  }, [key, roomName]);

  const sendMessage = (event) => {
    event.preventDefault();
    if (!message.trim()) return;

    const encryptedMessage = CryptoJS.AES.encrypt(message, key).toString();
    socket.emit('message', encryptedMessage);
    setMessage('');
  };
```

```
    return (
      <div>
        <h2>Chat Room: {roomName}</h2>
        <div>
          {messages.map((msg, index) => (
            <p key={index}>{msg}</p>
          ))}
        </div>
        <form onSubmit={sendMessage}>
          <input
            type="text"
            value={message}
            onChange={(e) => setMessage(e.target.value)}
          />
          <button type="submit">Send</button>
        </form>
      </div>
    );
};

export default ChatRoom;

// src/App.js

import React, { useState } from 'react';
import ChatRoom from './components/ChatRoom';

const App = () => {
    const [roomName, setRoomName] = useState('');
    const [joined, setJoined] = useState(false);

    const joinRoom = (e) => {
      e.preventDefault();
      if (roomName.trim()) {
        setJoined(true);
      }
    };

    return (
      <div>
        {joined ? (
          <ChatRoom roomName={roomName} />
        ) : (
          <form onSubmit={joinRoom}>
            <input
              type="text"
              value={roomName}
              onChange={(e)  => setRoomName(e.target.value)}
            />
            <button type="submit">Join Chat Room</button>
          </form>
        )}
      </div>
    );
};

export default App;
```

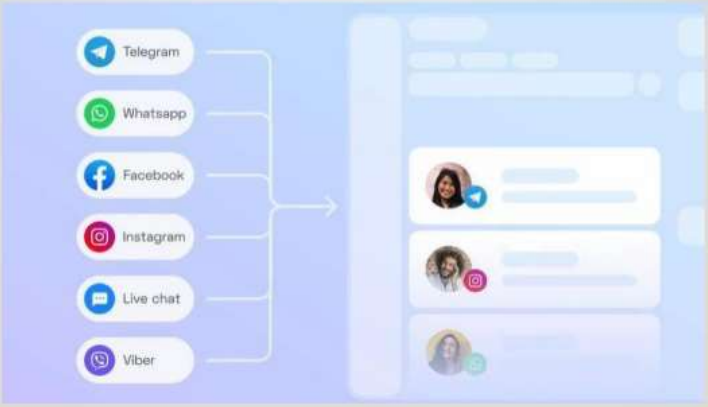
Додаток Г. Ілюстрований матеріал



ПРАКТИЧНА ЦІННІСТЬ

Практична цінність роботи, полягає в підвищені захищеності обміну повідомленнями. Розроблене рішення, дозволить не уникнути фінансових, репутаційних втрат, та крадіжки конфіденційних повідомлень, через можливі витоки, завдяки несанкціонованому доступу

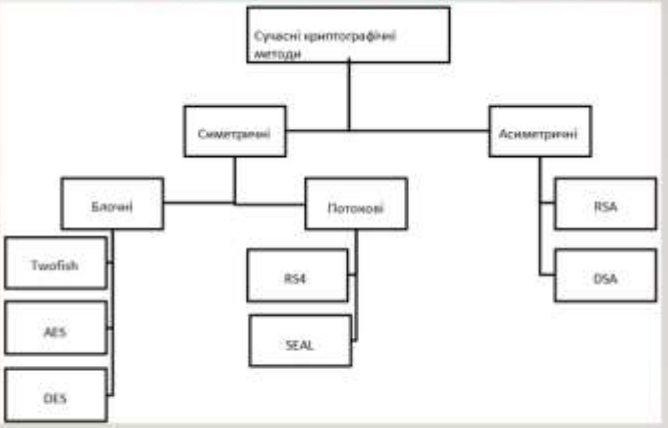
АНАЛІЗ ІСНУЮЧИХ ЧАТ-МЕСЕНДЖЕРІВ



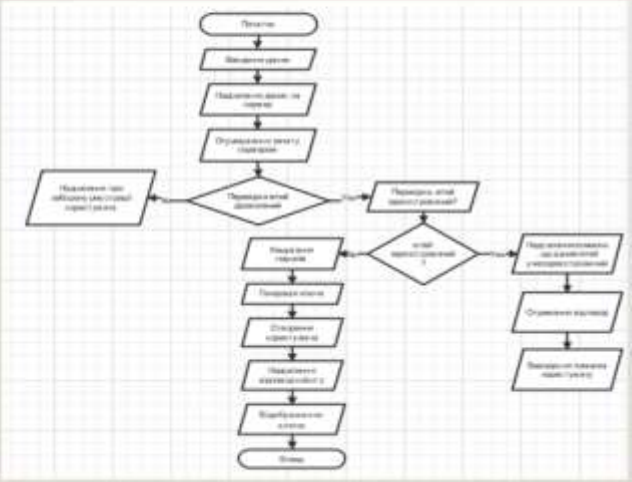
КРИПТОГРАФІЧНІ МЕТОДИ

ШИФРУВАННЯ ДАНИХ ЗАХИЩАЄ ІНФОРМАЦІЮ ВІД ВТРАТИ, ЗМІНИ АБО ЗЛАМУ. ОДНАК КЛЮЧ ДЕШИФРУВАННЯ МАЄ БУТИ ЗАКРИТИМ І ЗАХИЩЕНИМ ВІД НЕБАЖАНОГО ДОСТУПУ, ЩОБ ГАРАНТУВАТИ БЕЗПЕКУ ДАНИХ.

СУЧАСНІ КРИПТОГРАФІЧНІ МЕТОДИ



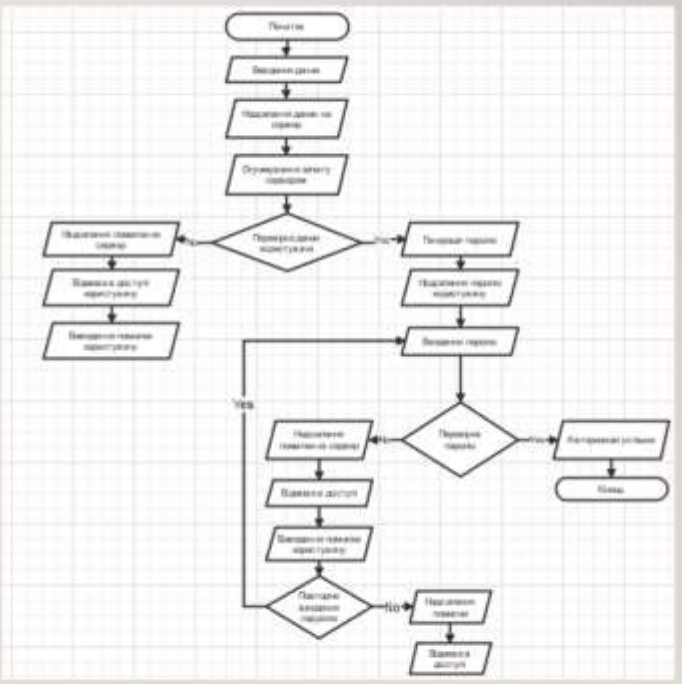
РОЗРОБКА ЗАПРОПОНОВАНИХ АЛГОРІТМІВ



розроблена блок-схема запропонованого алгоритму реєстрації користувачів

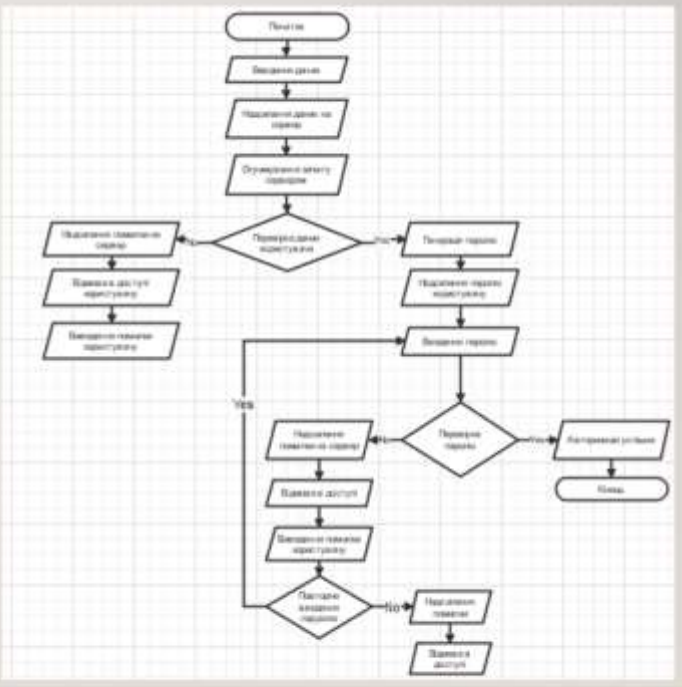
РОЗРОБКА ЗАПРОПОНОВАНИХ АЛГОРИТМІВ

розроблена блок-схема запропонованого алгоритму перевірки сесій користувачів



РОЗРОБКА ЗАПРОПОНОВАНИХ АЛГОРИТМІВ

розроблена блок-схема запропонованого алгоритму перевірки сесій користувачів



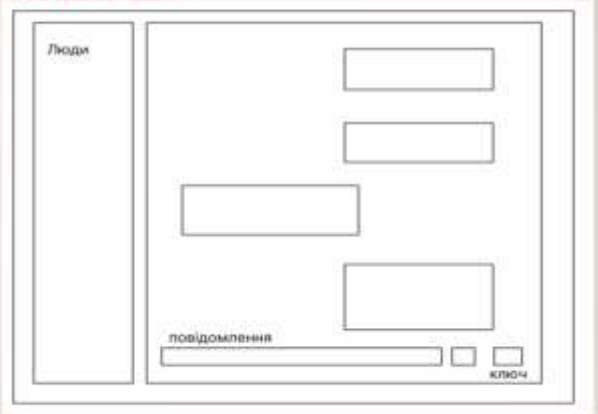
РОЗРОБКА ЗАПРОПОНОВАНИХ АЛГОРИТМІВ

– розроблена блок-схема запропонованого алгоритму роботи чат-месенджера

ПРОЕКТУВАННЯ ІНТЕРФЕЙСУ ДЛЯ ЧАТ-МЕСЕНДЖЕРА

СПРОЕКТОВАНИЙ МАКЕТ ІНТЕРФЕЙСУ СТОРІНКИ РЕЄСТРАЦІЇ/АВТОРИЗАЦІЇ

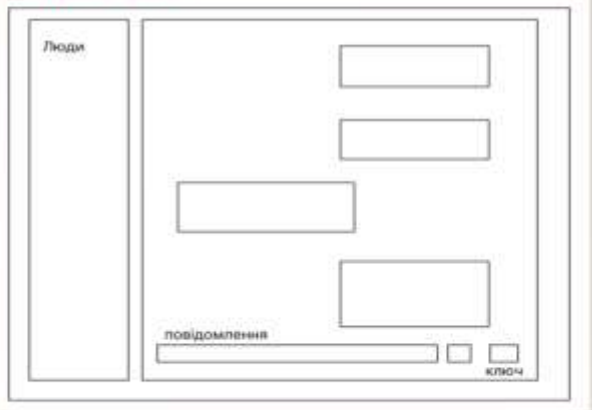
СПРОЕКТОВАНИЙ МАКЕТ ІНТЕРФЕЙСУ СТОРІНКИ ЧАТУ



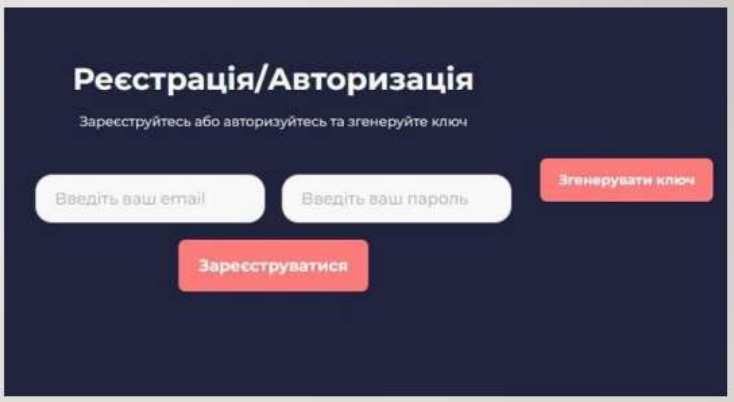
ПРОЕКТУВАННЯ ІНТЕРФЕЙСУ ДЛЯ ЧАТ-МЕСЕНДЖЕРА

СПРОЕКТОВАНИЙ МАКЕТ ІНТЕРФЕЙСУ СТОРІНКИ РЕЄСТРАЦІЇ/АВТОРИЗАЦІЇ

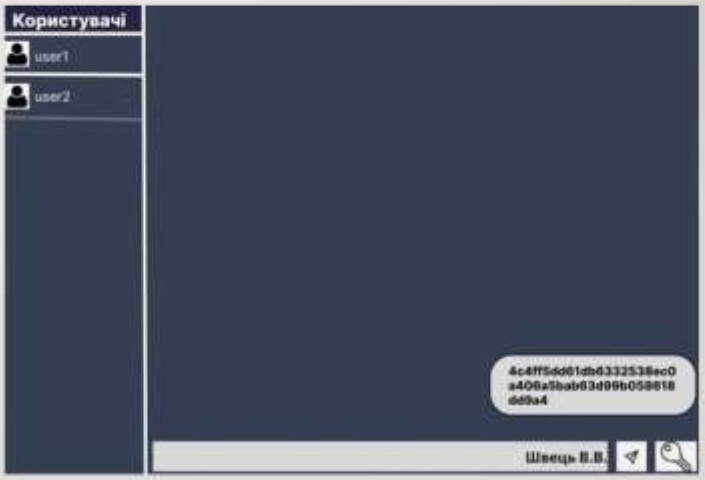
СПРОЕКТОВАНИЙ МАКЕТ ІНТЕРФЕЙСУ СТОРІНКИ ЧАТУ



ЗОВНІШНІЙ ВИГЛЯД РОЗРОБЛЕНОГО ЗАХИЩЕНОГО КОРПОРАТИВНОГО ФАЙЛООБМІННИКА З ДВОФАКТОРНОЮ АВТЕНТИФІКАЦІЄЮ



ЗОВНІШНІЙ ВИГЛЯД РОЗРОБЛЕНОГО ЧАТ-МЕСЕНДЖЕРА



ДЯКУЮ ЗА УВАГУ

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ
ЗАПОЗИЧЕНЬ

Назва роботи: Розробка захищеного чат-месенджера

Тип роботи: бакалаврська дипломна робота
(БДР, МКР)

Підрозділ: Кафедра менеджменту та безпеки інформаційних систем
Факультет менеджменту та інформаційної безпеки
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 93 % Схожість 7 %

Аналіз звіту подібності (відмітити потрібне):
1. Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.
2. Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.
3. Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Коваль Н.П.
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Швець В.В.
(підпис) (прізвище, ініціали)

Керівник роботи _____ Яремчук Ю.Є.
(підпис) (прізвище, ініціали)