МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ ФАХОВИЙ БІЗНЕС-КОЛЕДЖ
Цифрова комісія (кафедра) комп’ютерної інженерії та інформаційних технологій

КВАЛІФІКАЦІЙНА РОБОТА

на тему

АВТОМАТИЗАЦІЯ ВЗАЄМОДІЇ З TELEGRAM-ГРУПАМИ

Виконав: студент групи 2К-21
Спеціальності
123 Комп’ютерна інженерія
Ян ЯНЧИШЕН

Керівник:
Маргарита МЕДОЛИЗ

Черкаси 2025

АНОТАЦІЯ

У сучасних умовах активного використання месенджера Telegram як платформи для спілкування, маркетингу й адміністрування, виникає необхідність автоматизації рутинних завдань, таких як приєднання до груп і розсилка повідомлень. Ці дії є особливо актуальними для адміністраторів великої кількості груп, де ручне виконання займає значний обсяг часу.

У кваліфікаційній роботі розроблено програмний засіб автоматизації взаємодії з Telegram-групами, що реалізує приєднання до публічних каналів, масову розсилку повідомлень, обробку винятків і логування результатів. Для реалізації функціоналу використано мову Python, бібліотеку Telethon для взаємодії з Telegram API та локальну базу даних SQLite для збереження логів.

Під час реалізації враховано обмеження Telegram щодо надсилання повідомлень, що дозволяє уникнути блокувань. Результати тестування засвідчили стабільність роботи системи, коректність приєднання до груп та успішне надсилання текстових і медіа-повідомлень.

Розроблена система може бути адаптована під потреби маркетингових кампаній, новинних каналів, громадських ініціатив і будь-яких Telegram-спільнот, де важлива оперативність, масштабованість та ефективне адміністрування.

ABSTRACT

In modern digital communication, Telegram is increasingly used not only as a messenger but also as a platform for marketing, administration, and information distribution. This creates a demand for automation of routine tasks such as group joining and mass messaging, especially for administrators handling multiple communities.

This qualification paper presents the development of a software tool for automating Telegram group interactions, enabling automatic joining of public channels, mass message delivery, exception handling, and logging. The tool is implemented in Python using the Telethon library for working with the Telegram API and SQLite as a local database for logging operations.

Telegram API limitations were taken into account to avoid blocking, and testing showed that the system performs reliably when joining groups and sending both text and media messages.

The developed solution is suitable for use in marketing campaigns, news distribution, public initiatives, and other Telegram communities where efficiency, automation, and scalability are crucial.

ЗМІСТ

ВСТУП 3
РОЗДІЛ 1 АВТОМАТИЗОВАНА ВЗАЄМОДІЯ ТА КЕРУВАННЯ ПРОЦЕСАМИ В TELEGRAM-ГРУПАХ ЗА ДОПОМОГОЮ ПРОГРАМНИХ ЗАСОБІВ 5
1.1 Telegram як середовище для автоматизації 5
1.2 Огляд інструментів для роботи з Telegram API 6
1.3 Бібліотека Telethon: можливості та приклади використання 7
1.4 Аналіз схожих систем 9
РОЗДІЛ 2 ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ TELEGRAM-БОТА 11
2.1 Аналіз вимог до програмного забезпечення, що розробляється 11
2.2 Загальна структура програмного рішення 12
2.3 Опис функціонального модуля Invite.py 14
2.4 Опис функціонального модуля Send.py 15
2.5 Взаємодія з Telegram API та обробка виключень 16
2.6 Збереження даних і журналювання 18
2.7 Реалізація авторизації та підключення клієнта 19
2.8 Алгоритм приєднання до Telegram-груп 21
2.9 Масове надсилання повідомлень із затримкою 23
2.10 Логування результатів роботи 25
2.11 Обробка винятків та стійкість до помилок 26
РОЗДІЛ 3 ТЕСТУВАННЯ ТА СУПРОВОДЖЕННЯ 29
3.1 Обґрунтування обраного методу тестування 29
3.2 План тестування 30
3.3 Результати тестування 31
ВИСНОВКИ 34
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 36
ДОДАТОК А - ЛІСТИНГ ПРОГРАМНОГО КОДУ 38

ВСТУП

Сучасні інформаційні технології активно проникають у всі сфери людської діяльності, зокрема в комунікацію, адміністрування та маркетинг. Одним із найпопулярніших засобів спілкування є месенджер Telegram, який з кожним роком набуває дедалі більшого поширення не тільки як інструмент особистих повідомлень, а й як потужна платформа для створення груп, каналів і ботів.

Зі зростанням кількості Telegram-спільнот виникає проблема ефективного управління цими ресурсами. Ручне приєднання до великої кількості груп, а також надсилання повторюваних повідомлень займає багато часу й вимагає значних зусиль. Саме тому особливу актуальність набуває автоматизація таких процесів за допомогою інструментів програмування.

Актуальність теми обумовлена потребою у зменшенні навантаження на адміністраторів Telegram-груп, підвищенні ефективності розсилок, уникненні рутинної ручної праці та забезпеченні стабільності роботи з Telegram API. З огляду на це, актуальним є створення програмного засобу, що дозволяє автоматизувати взаємодію з Telegram-групами.

Об’єкт дослідження. Процеси автоматизації взаємодії з Telegram-групами та каналами.

Предмет дослідження. Програмні методи й засоби автоматизації взаємодії з Telegram API за допомогою бібліотеки Telethon на мові Python.

Мета роботи. Розробити програмний засіб, що забезпечує автоматичне приєднання до Telegram-груп і масове надсилання повідомлень із функцією логування.

Для досягнення мети у роботі було поставлено такі завдання:

- проаналізувати можливості Telegram API та інструменти взаємодії з ним;
- дослідити принципи роботи бібліотеки Telethon;
- спроєктувати архітектуру Telegram-бота;

- реалізувати функціонал авторизації, приєднання до груп, масової розсилки та логування;
- протестувати стабільність роботи програмного забезпечення;
- оцінити ефективність реалізованої системи автоматизації.

Методи дослідження*:* Аналіз документації Telegram API; практичне програмування мовою Python; тестування розробленого функціоналу в реальних умовах; логування та оцінка результатів.

Запропоноване рішення має прикладне значення та може бути використане для організації масових повідомлень, модерування груп, запуску інформаційних кампаній, що охоплюють десятки і сотні Telegram-спільнот. Такі програмні засоби корисні як для малого бізнесу, так і для великих інформаційних проєктів.

РОЗДІЛ 1
АВТОМАТИЗОВАНА ВЗАЄМОДІЯ ТА КЕРУВАННЯ ПРОЦЕСАМИ В TELEGRAM-ГРУПАХ ЗА ДОПОМОГОЮ ПРОГРАМНИХ ЗАСОБІВ

1.1 Telegram як середовище для автоматизації

У сучасних умовах стрімкого розвитку цифрових комунікаційних технологій все більшої популярності набувають платформи, що забезпечують не лише обмін повідомленнями між користувачами, а й підтримку розширених інструментів автоматизації процесів взаємодії. Однією з таких платформ є Telegram – багатофункціональний месенджер, який поєднує можливості текстового обміну, створення групових чатів і каналів, а також підтримує відкритий інтерфейс прикладного програмування (API), що робить його придатним для інтеграції з програмними рішеннями різного рівня складності.

Telegram вирізняється наявністю двох рівнів доступу до API: Bot API, орієнтованого на створення ботів із обмеженими правами, та Core API, що забезпечує повноцінну взаємодію з інфраструктурою Telegram від імені реального користувача. Останній надає змогу здійснювати авторизацію через номер телефону, приєднуватися до груп, переглядати історію повідомлень, надсилати текстові й медіа-файли, а також фіксувати активність за допомогою журналювання (логування).

Широкі функціональні можливості Telegram Core API обумовили появу численних інструментів для автоматизації адміністрування спільнот, реалізації інформаційних кампаній, масового розсилання повідомлень, модерації чатів тощо. Особливої актуальності набуває використання Telegram як середовища для автоматизації у випадках, коли йдеться про управління великою кількістю груп або каналів, що потребують однотипних повторюваних дій.

Ручне виконання таких дій (наприклад, приєднання до декількох десятків груп, періодичне інформування учасників про певні події, технічне обслуговування спільнот) є неефективним, трудомістким та ресурсоємним. З

огляду на це, застосування спеціалізованих скриптів, що працюють на основі Telegram API, дозволяє зменшити вплив людського чинника, підвищити продуктивність і забезпечити контрольованість процесу шляхом фіксації результатів.

Окремо слід зазначити, що Telegram надає високий рівень стабільності передачі даних, підтримує асинхронну обробку запитів та має велику активну спільноту розробників. Це створює сприятливі умови для впровадження програмних рішень, орієнтованих на автоматизацію взаємодії з платформою як у бізнес-середовищі, так і в соціально-комунікаційній діяльності.

1.2 Огляд інструментів для роботи з Telegram API

Відкритість Telegram як цифрової платформи зумовила створення широкого спектра інструментів для програмної взаємодії з її функціоналом. Основу цієї взаємодії становить Telegram API, який реалізований у двох основних формах: Bot API та Telegram Core API (TDLib або Raw API). Обидва варіанти мають різний рівень доступу до функціоналу Telegram і застосовуються залежно від поставлених завдань.

Bot API використовується переважно для створення ботів – спеціальних облікових записів, які діють у межах строго визначених можливостей. Зокрема, вони не можуть ініціювати спілкування з користувачем без його дії, приєднуватися до груп без запрошення, а також не мають доступу до повної історії повідомлень. Попри це, Bot API дозволяє ефективно реалізовувати інтерфейси взаємодії з користувачем, проводити опитування, керувати чатами через Webhooks та HTTP-запити.

Натомість Telegram Core API, що реалізований через такі клієнтські бібліотеки, як Telethon, Pyrogram, MadelineProto, надає розробнику змогу здійснювати повноцінне керування акаунтом Telegram. Зокрема, мова йде про підтримку входу в систему за допомогою номера телефону, приєднання до груп і каналів, надсилання повідомлень від імені реального користувача, обробку

подій, зчитування чатів, а також перегляд профілів та списку учасників. Доступ до такого рівня взаємодії відкриває можливості для розробки клієнтів Telegram або реалізації рішень з автоматизації взаємодії з великою кількістю груп.

Серед бібліотек для роботи з Telegram Core API особливої уваги заслуговує Telethon, написана мовою програмування Python. Вона вирізняється високою стабільністю, підтримкою асинхронного виконання операцій (що особливо актуально при роботі з великою кількістю чатів), зручною структурою класів і методів, а також детальною документацією. Telethon дозволяє реалізовувати як елементарні сценарії, пов’язані з надсиланням повідомлень, так і складні автоматизовані процеси з керування обліковими записами, обробкою помилок і логуванням.

Інші бібліотеки, зокрема Pyrogram, також мають активну спільноту підтримки, однак Telethon залишається однією з найпоширеніших у задачах, що вимагають асинхронної обробки, масштабованості та повного контролю над взаємодією з Telegram API.

1.3 Бібліотека Telethon: можливості та приклади використання

Серед інструментів для програмної взаємодії з Telegram Core API, особливе місце посідає бібліотека Telethon, що є асинхронною клієнтською реалізацією API, написаною мовою Python. Вона забезпечує повноцінний доступ до функціоналу Telegram, аналогічний можливостям офіційного десктопного клієнта. Відкрита архітектура Telethon, підтримка асинхронного виконання задач та зручна об’єктно-орієнтована структура визначають її ефективність у розробці програмних рішень для автоматизації взаємодії з Telegram-групами.

Основною перевагою Telethon є можливість авторизації через акаунт звичайного користувача, що відкриває розширені сценарії автоматизації, недоступні через Bot API. Серед таких функцій: приєднання до груп або каналів за допомогою JoinChannelRequest, надсилання повідомлень як до публічних, так

і до приватних чатів, передавання медіафайлів, зчитування історії листування, обробка подій, отримання інформації про учасників, а також адміністрування чатів.

Бібліотека Telethon підтримує два основні режими роботи: асинхронний, що базується на використанні asyncio, і синхронний, який реалізується через стандартні виклики методів. Для задач високого навантаження (наприклад, масове розсилання повідомлень із затримкою, обробка великої кількості групових чатів) перевагу надають саме асинхронним підходам, які забезпечують ефективне управління ресурсами та мінімізують затримки.

Окрім базового обміну даними, Telethon надає підтримку обробки винятків, що дозволяє реалізовувати стійкі до помилок системи. Зокрема, бібліотека підтримує автоматичне відновлення з’єднання, повторні спроби надсилання повідомлень, а також генерацію логів, що фіксують усі події, пов’язані з помилками або успішними діями клієнта.

У контексті автоматизації приєднання до груп Telethon дозволяє зчитувати посилання із зовнішнього файлу, формувати запити на вступ до кожного каналу, а у разі помилок – зберігати інформацію про тип винятку (наприклад, обмеження з боку Telegram на частоту дій, недійсне посилання або блокування акаунта). Таким чином забезпечується не лише виконання необхідних дій, а й зворотний контроль за їх успішністю.

Для задач масового надсилання повідомлень Telethon підтримує передавання як текстових, так і медіафайлів. Наприклад, за допомогою методу send_message можна організувати поширення контенту одразу до декількох груп або каналів, дотримуючись встановлених затримок між повідомленнями, що критично важливо для запобігання блокуванням з боку Telegram.

Завдяки своїй функціональності та гнучкості, бібліотека Telethon стала стандартом де-факто для розробки систем автоматизації Telegram-клієнтів мовою Python. Її використання дозволяє створювати рішення, які не поступаються за можливостями офіційним додаткам, але при цьому

відкривають шлях до повної автоматизації повторюваних дій у Telegram-середовищі.

1.4 Аналіз схожих систем

Аналіз сучасних підходів до автоматизації взаємодії з Telegram-групами свідчить про наявність великої кількості публічних рішень, реалізованих переважно у вигляді скриптів, ботів або десктопних утиліт. Відкритий код багатьох із них розміщено на платформах на кшталт GitHub, що дозволяє досліджувати принципи їх роботи, архітектуру та обмеження. Проте переважна більшість таких засобів має ряд суттєвих недоліків, які унеможливлюють їх ефективне використання в умовах стабільного інформаційного середовища.

Зокрема, низка існуючих рішень реалізована із порушенням базових принципів безпеки, не враховує обмежень Telegram щодо частоти дій з боку клієнта, не підтримує налаштування затримок між повідомленнями, не має модулів логування, а також не забезпечує належної обробки виключень. Як наслідок, використання таких систем часто призводить до блокування облікового запису, втрати контролю над процесом або порушення цілісності комунікацій.

Ще одним поширеним недоліком є відсутність гнучкості. Значна частина готових скриптів або жорстко прив'язана до конкретних умов (наприклад, фіксованих ID чатів), або потребує ручного редагування коду для зміни конфігурації. Водночас для використання у реальному середовищі необхідно мати інструмент, який дозволяє легко змінювати списки груп, повідомлень або медіа-файлів без повторного втручання в логіку програми.

На основі проведеного огляду можна сформулювати чітке завдання для дослідження та реалізації власного рішення. Необхідно розробити програмний засіб, який би забезпечував:

- авторизацію облікового запису Telegram через номер телефону;

- автоматичне приєднання до груп на основі зовнішнього файлу з посиланнями;
- масове надсилання текстових повідомлень і медіа-контенту до списку груп або каналів;
- підтримку конфігурованих інтервалів між повідомленнями для уникнення блокувань;
- повноцінне логування усіх дій з можливістю відстеження результатів (успіхів і помилок);
- стійкість до помилок, викликаних мережею, обмеженнями API або некоректними вхідними даними.

Вибір інструментів для реалізації обумовлений потребою у стабільній роботі, розширюваності та підтримці асинхронної взаємодії з API. З цією метою доцільним є використання мови програмування Python у поєднанні з бібліотекою Telethon, яка зарекомендувала себе як потужний інструмент у розробці автоматизованих клієнтів для Telegram.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ TELEGRAM-БОТА

2.1 Аналіз вимог до програмного забезпечення, що розробляється

Програмний засіб створено з метою автоматизації процесів приєднання до Telegram-груп та надсилання повідомлень без участі користувача. Основне завдання – скоротити час на виконання однотипних дій, які вручну потребують значних зусиль, особливо коли йдеться про десятки або сотні груп. Передбачено, що користувач запускає готовий скрипт, попередньо налаштувавши необхідні параметри у текстових файлах, без необхідності змінювати код.

Робота програми повинна бути стійкою до помилок, працювати без збоїв навіть у разі обмежень з боку Telegram API. Усі ключові параметри, включно з повідомленням, списками груп і зображеннями, мають зчитуватися із зовнішніх файлів. Це дозволяє легко змінювати вміст розсилки без редагування вихідного коду.

Програмне забезпечення повинно підтримувати авторизацію Telegram-користувача за номером телефону, зберігати сесію для подальших запусків без повторного входу, дозволяти автоматичне приєднання до груп і надсилати повідомлення у вказані чати. Передбачено, що повідомлення можуть містити як текст, так і графічні вкладення.

Важливо також забезпечити фіксацію результатів роботи. Після кожного запуску повинна створюватися історія виконаних дій, що зберігається у лог-файлі. У ньому зазначається час надсилання, цільова група, статус (успішно або помилка) та, за можливості, посилання на відправлене повідомлення.

Вимоги до системи також охоплюють стійкість до виняткових ситуацій. У разі помилок Telegram (наприклад, перевищення лімітів чи відсутності доступу до групи) програма має обробляти ці ситуації і продовжувати

виконання. Надсилання повідомлень повинно відбуватися з випадковими затримками, щоб уникнути автоматичного блокування акаунта.

Усі ці вимоги спрямовані на створення надійного, простого у використанні і гнучкого програмного інструмента для Telegram-автоматизації, який може бути застосований у повсякденній діяльності без глибоких технічних знань користувача.

2.2 Загальна структура програмного рішення

Розроблене програмне забезпечення для автоматизації взаємодії з Telegram-групами побудоване на принципах модульності, масштабованості та незалежності окремих компонентів. Архітектура рішення охоплює два основних програмних модулі, які реалізують ключові функціональні напрями: автоматичне приєднання до груп та масове надсилання повідомлень. Обидва модулі реалізовані у вигляді окремих асинхронних скриптів мовою програмування Python із використанням бібліотеки Telethon.

Програмна система структурована таким чином, щоб забезпечити мінімальну взаємозалежність між частинами логіки, легке оновлення даних (списків груп, повідомлень, файлів) без необхідності змін у програмному коді, а також безперервність роботи навіть у разі виникнення часткових помилок. Детальну структуру проєкту з основними компонентами та файловою організацією наведено в Додатку А.

Загальна логічна структура системи охоплює такі компоненти:

- Скрипт Invite.py – відповідає за реалізацію механізму авторизації користувача та приєднання до Telegram-груп або каналів за заданим списком посилань.
- Скрипт Send.py – забезпечує надсилання повідомлень (текстових та медіа) до заданих груп з фіксацією результатів у лог-файлах.
- Конфігураційні файли – текстові файли у папці Change, які містять список посилань на групи (groupinvite.txt), список ID або юзернеймів чатів

(groups.txt), текст повідомлення (message.txt), а також зображення у відповідній підпапці Images/.

– Файли сесій Telegram – автоматично створюються при авторизації користувача, використовуються для збереження облікового запису Telegram і повторного доступу без потреби повторного введення коду.

– Система логування – реалізована як окрема підсистема, яка створює файл із часовою міткою та фіксує результати виконання кожної дії (успішне надсилання, помилки, згенеровані посилання на повідомлення тощо).

На рисунку 2.1 представлено узагальнену логічну структуру проєкту, яка відображає взаємозв'язки між файлами конфігурації, основними скриптами та зовнішніми даними.

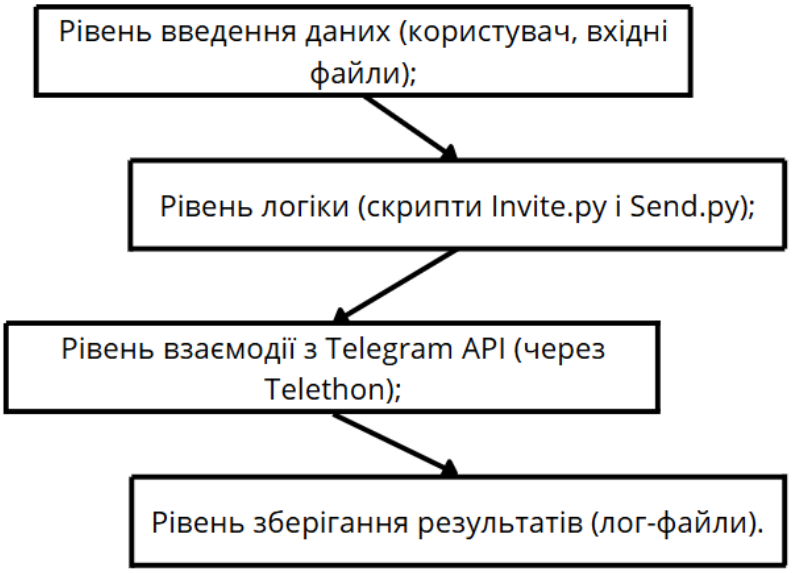

Рисунок 2.1 – Загальна структура програмного рішення з автоматизації Telegram-взаємодії

Така модульна організація забезпечує розділення відповідальності між компонентами, спрощує тестування, обслуговування та подальший розвиток системи, зокрема шляхом розширення функціоналу або інтеграції з іншими платформами.

2.3 Опис функціонального модуля Invite.py

Модуль Invite.py реалізує процес автоматичного приєднання облікового запису користувача до Telegram-груп або каналів. Робота скрипта організована у вигляді послідовного алгоритму, що включає авторизацію Telegram-клієнта, зчитування посилань на групи та обробку запитів приєднання.

На першому етапі, після запуску скрипта, система запитує номер телефону. Якщо відповідна Telegram-сесія вже існує, ініціалізується клієнт без додаткової авторизації. У протилежному випадку користувачеві надсилається код входу до Telegram, після введення якого створюється сесійний .session-файл для подальшого використання без повторного введення даних.

Після авторизації модуль зчитує список посилань на групи з файлу groupinvite.txt, розташованого у відповідній директорії. Кожен запис обробляється через запит JoinChannelRequest. У випадку успішного приєднання до групи Telegram підтверджує виконання дії, і скрипт переходить до наступного запису. Фрагмент коду, що реалізує приєднання до груп за допомогою Telethon, подано в Додатку А. У разі помилки (наприклад, недійсне посилання, закритий доступ до групи або блокування акаунта) програма обробляє виняток і продовжує виконання без зупинки.

Архітектура модуля включає базову систему обробки виключень (try/except), що дозволяє уникнути аварійного завершення роботи при виникненні типової помилки, зокрема FloodWaitError, ChannelPrivate, UserBannedInChannel, InviteHashExpired тощо.

Загальний алгоритм функціонування скрипта відображено на рисунку 2.2, який ілюструє всі основні етапи – від введення номера телефону до завершення приєднання до Telegram-груп.

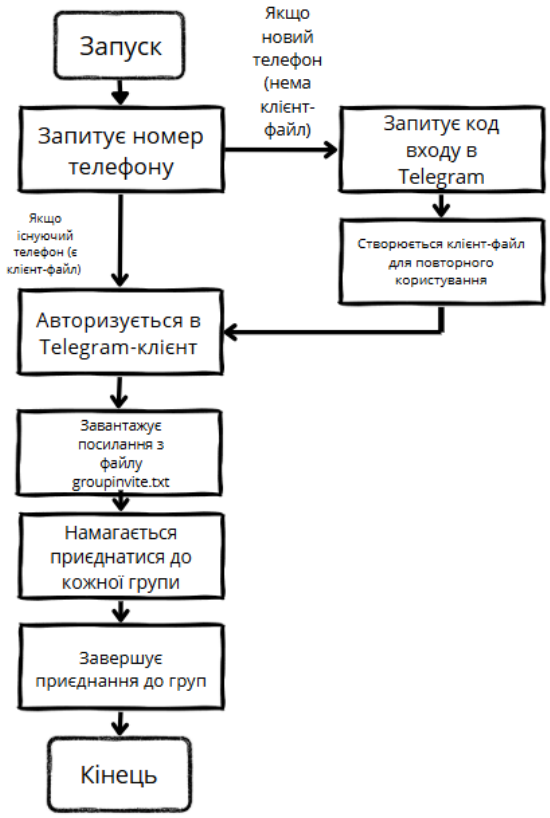

Рисунок 2.2 – Блок-схема алгоритму роботи модуля Invite.py

2.4 Опис функціонального модуля Send.py

Модуль Send.py реалізує автоматизовану розсилку повідомлень до попередньо визначеного списку Telegram-груп або каналів. Скрипт забезпечує надсилання як текстових повідомлень, так і повідомлень із медіафайлами. Робота модуля структурована у вигляді алгоритму, що поєднує авторизацію, зчитування конфігураційних файлів, обробку повідомлень, надсилання та логування результатів.

На початку виконання скрипт запитує номер телефону. У разі, якщо відповідний сесійний файл Telegram уже існує, авторизація виконується автоматично. Інакше користувач має ввести код, отриманий через Telegram, після чого створюється новий сесійний файл.

Після підключення Telegram-клієнта здійснюється зчитування вхідних файлів: message.txt (текст повідомлення) та groups.txt (список ID або юзернеймів груп). Якщо у папці Images/ виявлено зображення, воно буде

приєднане до повідомлення. У протилежному випадку надсилається лише текст. Алгоритм масової розсилки повідомлень, реалізований у модулі Send.py, проілюстровано у Додатку А.

Користувач встановлює мінімальний та максимальний інтервал між повідомленнями. Скрипт генерує випадкову паузу у заданому діапазоні перед кожною наступною відправкою, що дозволяє обійти обмеження Telegram щодо частоти повідомлень (anti-spam політика).

Кожен результат надсилання фіксується в лог-файлі з міткою часу. Якщо група має username, створюється пряме посилання на надіслане повідомлення. У випадку помилки (наприклад, PeerIdInvalid, FloodWaitError, ChatWriteForbiddenError) вона обробляється відповідним блоком except, без припинення роботи всієї програми.

Схематичний алгоритм роботи скрипта наведено на рисунку 2.3, який ілюструє послідовність усіх основних дій: авторизації, зчитування, перевірки медіа, надсилання повідомлень та логування результатів.

Рисунок 2.3 – Блок-схема алгоритму роботи модуля Send.py

2.5 Взаємодія з Telegram API та обробка виключень

Реалізація автоматизованої взаємодії з Telegram-групами базується на використанні Telegram Core API, доступ до якого забезпечується через

бібліотеку Telethon. Ця бібліотека надає низькорівневі засоби для повноцінного управління обліковим записом Telegram, включаючи авторизацію, приєднання до груп, надсилання повідомлень, роботу з медіа та обробку подій.

Авторизація користувача

Telegram Core API потребує первинної авторизації, яка здійснюється за допомогою номера телефону. Якщо сесія користувача раніше не була створена, система запитує одноразовий код підтвердження, що надходить у Telegram-додаток або через SMS. Після успішної авторизації створюється .session-файл, який дозволяє повторно використовувати сесію без потреби нової автентифікації.

У випадку, якщо акаунт захищений двофакторною аутентифікацією, викликається виключення SessionPasswordNeededError, яке обробляється окремо, з додатковим запитом пароля.

Скрипти використовують API-методи Telethon для виконання основних функцій:

- JoinChannelRequest(link) – приєднання до групи або каналу;
- send_message(entity, message, file=...) – надсилання повідомлення з опціональним медіа-вкладенням;
- UploadFile, GetEntityRequest, GetParticipantsRequest – допоміжні методи для ідентифікації груп, передачі файлів і отримання інформації.

Уся взаємодія виконується в асинхронному режимі з використанням async/await, що дозволяє ефективно керувати мережею і ресурсами клієнта без блокувань.

Telegram API містить вбудовані захисти від перевантаження, спаму, неправильного формату запитів або несанкціонованих дій. Приклади обробки типових винятків під час взаємодії з Telegram API наведено в Додатку А. У зв’язку з цим важливо забезпечити обробку виключень, що виникають під час виконання скриптів. У програмному рішенні реалізована системац винятків, яка обробляє такі ключові ситуації:

- FloodWaitError – перевищення допустимої частоти запитів. Скрипт зчитує параметр e.seconds і виконує паузу на вказаний час.
- PeerIdInvalidError – недійсний ID групи або помилка в імені користувача.
- ChannelPrivate – спроба доступу до закритої або приватної групи.
- UserBannedInChannel – обліковий запис заблоковано в певному каналі.
- MessageTooLongError або MediaEmptyError – некоректне або занадто велике повідомлення, порожній медіафайл.
- Інші загальні помилки: ValueError, TypeError, RPCError, які виникають при некоректному виклику API або помилках параметрів.

Усі операції, потенційно схильні до виключень, обгорнуті в конструкцію try/except. Завдяки цьому скрипт не перериває виконання навіть при серії помилок, а лише фіксує винятки в лог-файл і переходить до наступного об’єкта (групи або повідомлення).

Крім обробки помилок, система реалізує динамічне регулювання затримок між повідомленнями. Інтервал очікування задається у вигляді діапазону часу (мінімальне і максимальне значення), з якого випадковим чином обирається пауза перед кожним наступним API-запитом. Це дозволяє імітувати поведінку «людського» користувача і мінімізувати ймовірність блокування.

2.6 Збереження даних і журналювання

З метою забезпечення стабільності та контрольованості функціонування програмного засобу реалізовано механізм збереження вхідних даних і фіксації результатів виконання у вигляді журналів. Це дозволяє відстежувати роботу системи, виявляти помилки та формувати звітність без втручання користувача.

Конфігураційні дані зберігаються у текстових файлах, що розміщуються в каталозі Change/. Для кожного скрипта передбачено окремі вхідні файли: groupinvite.txt містить посилання на Telegram-групи для приєднання, groups.txt

– список ідентифікаторів чатів для розсилки, а message.txt – текст повідомлення. Додатково, у папці Images/ можуть зберігатися графічні файли, що додаються до повідомлень у разі виявлення вмісту.

Результати роботи кожного запуску скрипта Send.py зберігаються в окремому лог-файлі, який автоматично створюється у папці Logs/. Назва лог-файлу містить дату і час запуску, що дає змогу легко ідентифікувати конкретну сесію. У журналі фіксуються назви або ідентифікатори груп, до яких здійснювалася розсилка, результати спроб (успішне або неуспішне надсилання), а також повідомлення про помилки або винятки. За наявності публічного імені групи генерується пряме посилання на відповідне повідомлення.

Окрім логів, система зберігає Telegram-сесію у файлі з розширенням .session. Цей файл створюється під час першої авторизації і дозволяє обійти повторне введення коду при наступних запусках скриптів. Завдяки цьому забезпечується безперервність роботи без зайвих дій з боку користувача.

Журналювання дає змогу проаналізувати ефективність автоматизації, визначити проблемні моменти (наприклад, через обмеження Telegram API) і сформувати основу для вдосконалення процесу. Таким чином, збереження конфігураційних даних та результатів у логах є невіддільною частиною архітектури запропонованого рішення. Створення та запис у лог-файли проілюстровано у фрагменті коду, поданому в Додатку А.

2.7 Реалізація авторизації та підключення клієнта

Реалізація авторизації - фундаментальний етап побудови автоматизованої системи, що працює з Telegram. Без надійного механізму підключення до клієнта жодна з наступних дій - приєднання до груп, розсилка повідомлень, обробка подій - не може бути виконана. У межах цього проєкту процес авторизації реалізовано на основі бібліотеки Telethon, яка дозволяє повноцінно взаємодіяти з Telegram API від імені реального користувача, а не бота.

На першому кроці користувач вводить номер телефону, що використовується для ініціалізації Telegram-клієнта. Система перевіряє наявність локального *.session-файлу. Якщо файл існує, клієнт автоматично авторизується і переходить до виконання основних завдань. Якщо ж сесія відсутня, програма надсилає код підтвердження - через SMS або безпосередньо в Telegram-додаток. Користувач вводить цей код вручну, після чого здійснюється перевірка.

Якщо код введено вірно, формується Telegram-сесія й створюється відповідний .session-файл, який містить зашифровану інформацію про обліковий запис. Процедуру авторизації Telegram-клієнта реалізовано згідно з кодом у Додатку А. Це дозволяє надалі запускати систему без потреби повторної авторизації. У разі введення неправильного коду користувачеві надається можливість ввести його повторно.

Описаний процес передбачає обробку всіх стандартних сценаріїв, включно з активацією двофакторної автентифікації. У такому випадку після введення одноразового коду система додатково запитує пароль облікового запису, що додатково захищає доступ до клієнта.

На рисунку 2.4 подано візуалізацію алгоритму авторизації. Схема демонструє гілкування логіки залежно від наявності попередньої сесії, правильності введеного коду й створення нової сесії при першому запуску. Завдяки цьому процес є стійким, надійним і гнучким щодо повторного використання.

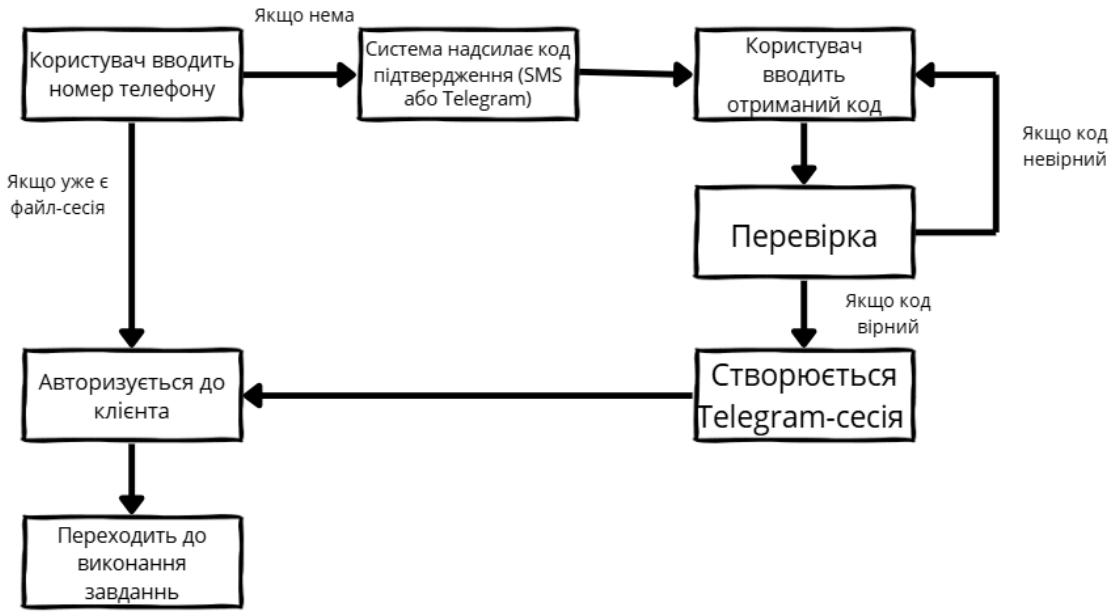

Рисунок 2.4 – Алгоритм авторизації користувача через TelegramClient

Така реалізація забезпечує мінімальну потребу у взаємодії з боку користувача й дозволяє запускати скрипти (наприклад, Invite.py, Send.py) в автоматичному режимі. Це зменшує кількість точок потенційного збою, пов’язаних із повторним введенням даних, втратами сесії або некоректною ідентифікацією користувача. У разі необхідності сесію можна вручну видалити, і авторизація буде ініціалізована повторно з повним відновленням.

Загалом, застосований підхід відповідає вимогам безпеки Telegram, зберігає баланс між зручністю користування і надійністю, і є стабільною основою для наступних етапів автоматизації.

2.8 Алгоритм приєднання до Telegram-груп

Автоматичне приєднання до Telegram-груп – один із базових етапів у реалізації автоматизованої взаємодії. Такий функціонал дозволяє заощадити значний обсяг часу, який зазвичай витрачається на ручне долучення акаунта до численних публічних каналів і спільнот. Це особливо актуально для інформаційних кампаній, технічного моніторингу, сповіщення великої кількості цільових груп або адміністрування Telegram-мереж.

У розробленому програмному забезпеченні реалізація приєднання відбувається за допомогою методу JoinChannelRequest з бібліотеки Telethon. Цей метод дає змогу надсилати запити на вступ до груп за вказаними URL-посиланнями, які попередньо зберігаються у текстовому файлі groupinvite.txt. Всі посилання обробляються у циклі, що дозволяє опрацювати навіть велику кількість записів автоматично. Послідовність приєднання до груп за списком посилань реалізована у вигляді коду в Додатку А.

Для забезпечення стабільності системи передбачено перевірку на порожні або дубльовані рядки. Кожне посилання проходить базову валідацію. У разі виявлення типових помилок, таких як неправильний формат запиту, відсутність доступу до групи, блокування акаунта чи обмеження Telegram, система не перериває виконання. Замість цього виняток перехоплюється, реєструється у консолі, і програма продовжує роботу з наступним елементом.

У процесі виконання результат приєднання кожної окремої групи виводиться в консоль: якщо приєднання успішне, відображається позитивне повідомлення, якщо виникла помилка – текст винятку. Таким чином користувач має змогу в реальному часі спостерігати за перебігом виконання й фіксувати потенційно проблемні посилання.

Для кращого розуміння загальна логіка приєднання до Telegram-груп наведена на рисунку 2.5. Схема відображає ключові кроки: зчитування посилань, фільтрацію, послідовне оброблення запитів, обробку типових винятків та вивід результатів у консоль.

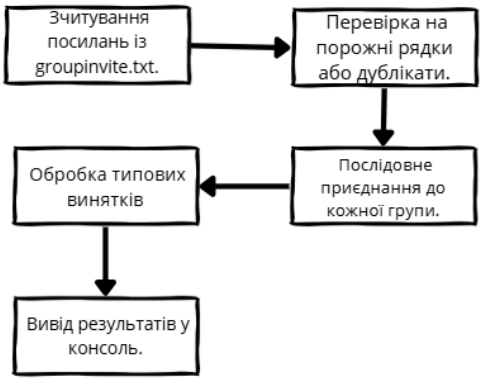

Рисунок 2.5 – Алгоритм автоматизованого приєднання до Telegram-груп

Реалізований підхід забезпечує високу надійність, масштабованість і стійкість до помилок, що особливо важливо для систем, які працюють із великою кількістю Telegram-груп у межах одного сеансу або розсилки.

2.9 Масове надсилання повідомлень із затримкою

Масове надсилання повідомлень є одним із ключових компонентів автоматизації взаємодії з Telegram-групами. У межах розробленого програмного засобу ця функція реалізована у скрипті Send.py, який забезпечує розсилку як текстових, так і мультимедійних повідомлень до великої кількості цільових чатів, визначених у зовнішніх конфігураційних файлах.

Перед початком розсилки відбувається авторизація користувача через Telegram API, після чого скрипт зчитує конфігураційні файли:
– groups.txt – список ID або username-ів Telegram-груп;
– message.txt – текст повідомлення, яке буде надіслано;
– папка Images/ – зображення, які можуть бути приєднані до повідомлень, якщо такі виявлено.

Після цього запускається цикл, що обробляє кожну групу окремо. Для кожної групи здійснюється ідентифікація через метод get_entity(), після чого виконується надсилання повідомлення. Якщо вхідний каталог містить зображення, воно автоматично додається до тексту.

Однією з важливих особливостей є реалізація затримок між повідомленнями. Ці затримки формуються випадковим чином у межах заданого користувачем інтервалу min_delay–max_delay. Реалізація випадкової затримки між повідомленнями для уникнення блокувань наведена в Додатку А. Такий підхід дозволяє моделювати природну поведінку людини й уникати блокування з боку Telegram за підозру у спам-діяльності.

Уся логіка роботи побудована з урахуванням асинхронності. Для реалізації затримки застосовується конструкція await asyncio.sleep(delay), де delay - випадкове значення з заданого діапазону.

Під час розсилки всі дії огорнуті в блок try-except, що дозволяє перехоплювати помилки, не зупиняючи виконання скрипта. Усі помилки (наприклад, FloodWaitError, PeerIdInvalid, ChatWriteForbiddenError) фіксуються у лог-файлі разом із позначенням групи, у якій сталася помилка.

Для кращого розуміння реалізованої логіки на рисунку 2.6 подано блок-схему алгоритму розсилки, яка відображає послідовність етапів: від авторизації й зчитування вхідних файлів – до надсилання повідомлень із медіа, логування результатів, затримки та завершення.

Під час тестування програмне забезпечення продемонструвало стабільну роботу навіть за умови обробки великої кількості груп. Розсилка відбувалася без збоїв, з коректним формуванням логів і дотриманням інтервалів, що підтверджує ефективність обраної моделі асинхронної обробки. Такий підхід дозволяє адаптувати систему під будь-яку кількість одержувачів, зберігаючи при цьому надійність та відповідність політикам Telegram API.

Рисунок 2.6 – Алгоритм масового надсилання повідомлень із затримкою

2.10 Логування результатів роботи

Логування є ключовим елементом програмного засобу, оскільки забезпечує фіксацію всіх подій, що відбуваються під час виконання автоматизованих дій. Це дає змогу користувачеві не лише контролювати роботу системи, а й виявляти помилки, зберігати історію запусків, а також забезпечувати прозору й достовірну звітність.

Механізм журналювання реалізовано у скрипті розсилки. При кожному запуску програма автоматично створює новий лог-файл, у якому фіксуються спроби надсилання повідомлень, результати виконання дій, час обробки, а також типи можливих винятків. Усі ці дії записуються у структурованому форматі з часовими мітками, що дозволяє легко ідентифікувати як успішно оброблені групи, так і ті, в яких виникли проблеми.

Кожен запис містить назву або ID групи, результат взаємодії з нею (успішно чи з помилкою), а в разі успіху – також посилання на надіслане повідомлення. У випадках помилок система не припиняє роботу, а перехоплює виняток, реєструє його у лог і переходить до наступного об'єкта. Формування структурованих логів про результати розсилки проілюстровано в Додатку А. Такий підхід гарантує стійкість до збоїв і забезпечує безперервне виконання сценарію навіть за наявності проблем із мережею, API або правами доступу.

Затримки між повідомленнями, які вводяться випадковим чином у заданому діапазоні, теж фіксуються. Це дозволяє моделювати поведінку реального користувача, а лог-файли відображають не лише результат, а й динаміку виконання.

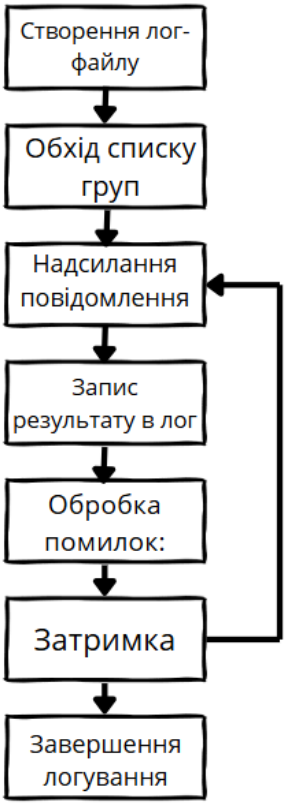

Рисунок 2.7 – Блок-схема алгоритму роботи логування

Загальна логіка логування проілюстрована на рисунку 2.7. Як видно зі схеми, процес охоплює етапи створення файлу, обробки груп, надсилання, реєстрації подій, обробки винятків, затримок і завершення сесії. Такий підхід дозволяє досягти високого рівня надійності та контролю над автоматизованим процесом.

2.11 Обробка винятків та стійкість до помилок

Telegram API має низку обмежень і політик безпеки, що можуть призвести до зупинки виконання скрипта у разі неправильної або надмірно частотної взаємодії. У зв’язку з цим, під час розробки модуля Send.py особливу увагу було приділено реалізації механізмів обробки винятків і підвищення стійкості програми до типових помилок, пов’язаних з Telegram-інфраструктурою.

Кожна операція, яка передбачає надсилання повідомлення, потенційно може викликати виняткову ситуацію (exception). З метою збереження працездатності системи реалізовано конструкцію try-except, яка перехоплює помилки, не допускає аварійного завершення роботи скрипта і дозволяє перейти до наступної цільової групи.

У межах реалізації обробляються наступні ключові помилки:

1. FloodWaitError – виняток, який виникає у випадку перевищення ліміту запитів до Telegram API за одиницю часу. Скрипт зчитує атрибут e.seconds, який вказує, скільки часу необхідно очікувати перед наступною дією. Ця інформація використовується для асинхронної паузи:

2. PeerIdInvalid – означає, що об'єкт групи не знайдено або ID вказаний некоректно. У таких випадках скрипт пропускає групу

3. ChatWriteForbiddenError – користувач не має прав писати в зазначену групу або його було обмежено. Така ситуація реєструється в лог

4. Загальні виключення (Exception) – слугують для фіксації всіх інших непередбачених ситуацій, включно з мережею або внутрішніми помилками Telegram:

Окрім виключень, у коді також реалізовано перевірку на наявність сесії, наявність зображень, правильність формату вхідних даних, а також автоматичне формування лог-файлів для кожної сесії запуску. Основні конструкції для обробки помилок у програмному забезпеченні подані в Додатку А.

Використання асинхронного програмування (async/await) забезпечує стабільне керування затримками і дає змогу адаптувати систему під політики Telegram без блокування потоку. Це дозволяє створити високонадійний і масштабований інструмент для розсилки, який не перериває виконання через окремі помилки.

Реалізація подібного підходу забезпечує ключову властивість - стійкість до збоїв. Навіть у випадку непередбаченої помилки або блокування одного з

каналів, скрипт продовжує роботу з іншими цільовими групами, що є критично важливим для масових розсилок.

РОЗДІЛ 3
ТЕСТУВАННЯ ТА СУПРОВОДЖЕННЯ

3.1 Обґрунтування обраного методу тестування

Для перевірки працездатності програмного засобу було застосовано ручне функціональне тестування. Цей підхід дав змогу оцінити роботу системи в реальному часі, враховуючи особливості Telegram API, які не завжди передбачувані або стабільні для автоматичних тестів.

Функціональне тестування дозволило перевірити основні можливості програми: авторизацію користувача, приєднання до Telegram-груп за списком посилань, надсилання повідомлень із текстом та медіа, а також обробку помилок і логування результатів. Ручне спостереження за виконанням скриптів допомогло оперативно виявляти помилки та оцінити загальну стабільність системи.

Окрему увагу було приділено тестуванню обробки винятків. Імітувалися ситуації з недійсними посиланнями, відсутністю доступу до груп, перевищенням лімітів Telegram, щоб переконатися в стійкості системи. Усі помилки фіксувались у логах, що підтверджувало надійність реалізованого механізму обробки.

Також програму було протестовано в умовах реального використання на базі Управління стратегічних розслідувань в Черкаській області ДСР Національної поліції України. У цьому середовищі програма пройшла практичну перевірку: виконувала масову розсилку службових повідомлень, правильно логувала результати та не зупиняла роботу у разі помилок.

Вибір саме ручного тестування був зумовлений необхідністю контролю над усіма етапами виконання та можливістю відслідковувати нестандартні ситуації, які складно заздалегідь спрогнозувати або автоматизувати. Завдяки цьому вдалося переконатися в стабільності та ефективності програмного рішення в умовах реального використання.

3.2 План тестування

План тестування був сформований таким чином, щоб охопити всі ключові функції програмного забезпечення і перевірити їхню працездатність у різних умовах. Тестування починалося з перевірки авторизації користувача через Telegram API. Важливо було впевнитися, що система правильно обробляє номер телефону, надсилає код підтвердження та створює сесію для подальшого використання.

Після цього перевірялося приєднання до Telegram-груп. У текстовому файлі groupinvite.txt були зібрані як дійсні, так і спеціально зіпсовані посилання. Це дозволило переконатися, що програма коректно приєднується до відкритих груп і не зупиняється у випадку помилки, а лише реєструє її.

Окремим етапом стало тестування функції розсилки. Перевірка проводилась із різними типами повідомлень – лише текстові та текст з прикріпленим зображенням. Тестувалося, як система реагує на наявність файлів у папці Images, як формує повідомлення і як веде себе при відсутності доступу до групи або помилковому ID.

Також було змодельовано ситуації, які могли викликати винятки, наприклад, обмеження за частотою надсилання повідомлень (FloodWait), недійсні або закриті чати, а також помилки доступу. Усі ці ситуації були успішно оброблені, програма продовжувала працювати, не перериваючи виконання.

Після завершення кожного етапу тестування вручну перевірялися створені лог-файли. У них фіксувалися час, адресат, статус повідомлення і, якщо можливо, пряме посилання на результат. Це дозволяло аналізувати хід виконання і оцінити успішність кожної дії.

3.3 Результати тестування

У ході тестування було підтверджено, що всі основні функції програмного засобу працюють стабільно та відповідають заявленим вимогам. Авторизація користувача виконувалась коректно, сесія Telegram створювалась після введення коду, і надалі використовувалась автоматично без повторного входу. Поведінку скрипта під час авторизації та обробки чатів можна побачити на рисунку 3.1, де зображено консольний вивід після запуску програми Send.py.

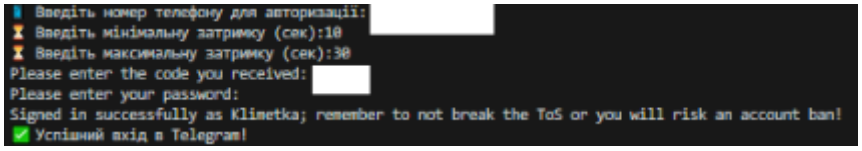

Рисунок 3.1 – Авторизація користувача в консольному режимі

Після авторизації система приєднувалась до Telegram-груп, зчитуючи посилання з файлу. Успішне приєднання до нової групи відображалося як у самому Telegram (через повідомлення «You joined the group»), так і в терміналі. Приклад виконання цього етапу наведено на рисунку 3.2.

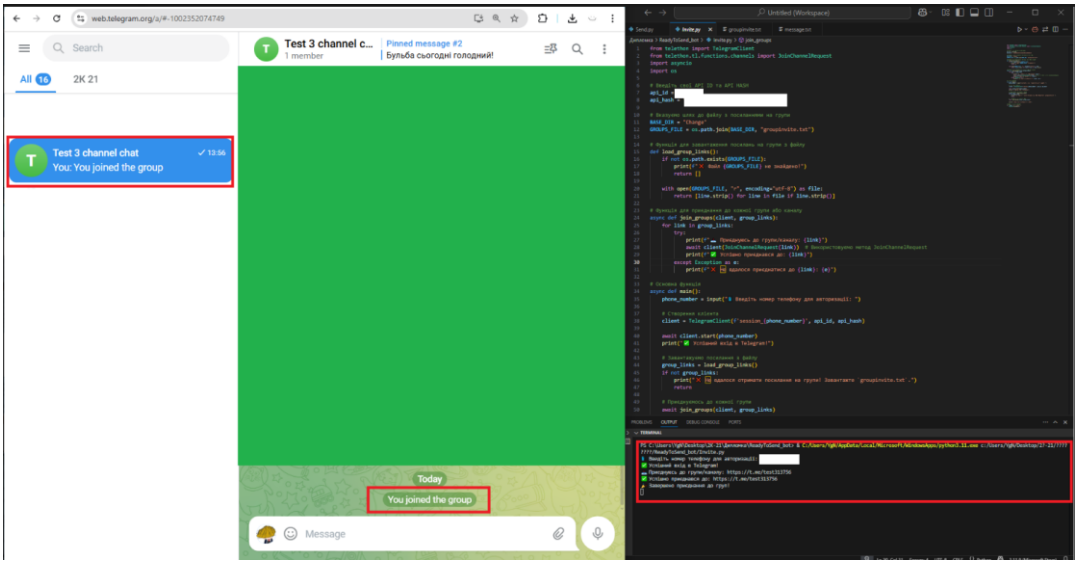

Рисунок 3.2 – Приєднання до групи в Telegram

Фінальним підтвердженням результатів тестування є скріншот безпосередньо з Telegram, де видно отримані повідомлення з прикріпленим зображенням (див. рисунок 3.3). Повідомлення було успішно доставлено в обидві групи одночасно, що підтверджує правильну роботу функціоналу надсилання та обробки медіа.

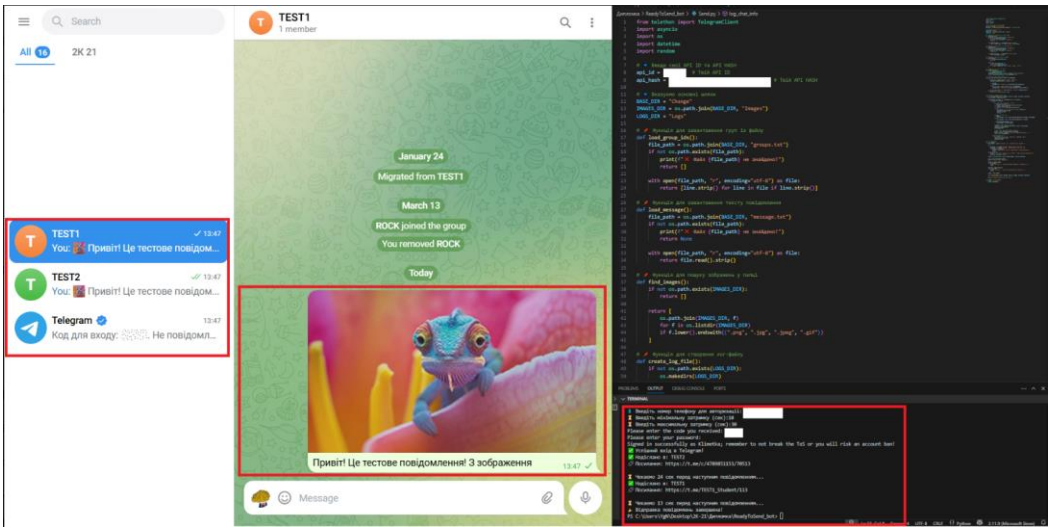

Рисунок 3.3 – Отримане повідомлення в Telegram-групах із зображенням

Фінальним підтвердженням результатів тестування є перевірка функції масової розсилки. Система доставляла повідомлення до всіх зазначених груп, а результати фіксувались у лог-файлі. У випадку успіху запис включав назву групи та пряме посилання на повідомлення в Telegram. Приклад такого лог-файлу показано на рисунку 3.4, де видно, що повідомлення було доставлено в групи TEST1 і TEST2.

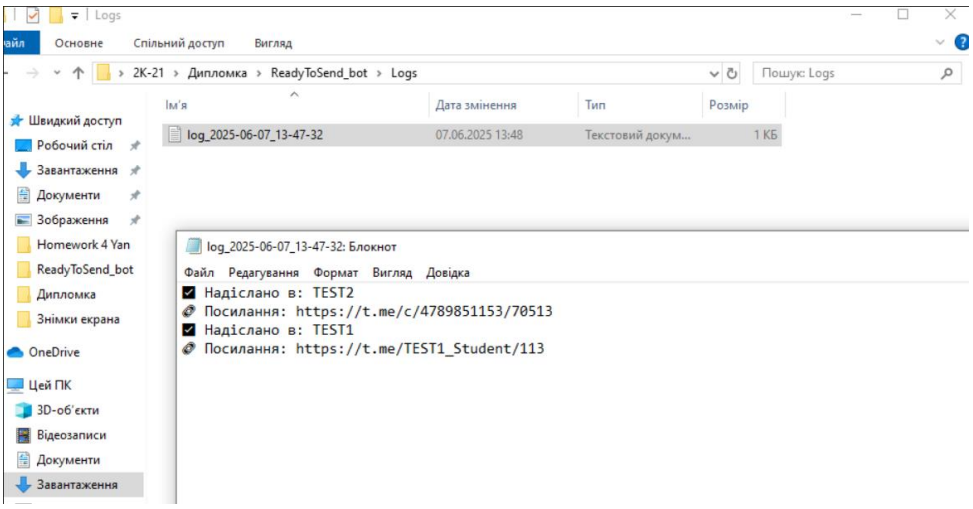

Рисунок 3.4 – Приклад лог-файлу з результатами розсилки

Система повністю витримала перевірку. Усі помилки, які виникали (наприклад, FloodWait), оброблялись без зупинки скрипта, а результат дій зберігався для подальшого аналізу. Розроблений інструмент підтвердив свою стабільність як у лабораторних умовах, так і під час реального використання в Управлінні стратегічних розслідувань в Черкаській області ДСР Національної поліції України.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було розроблено програмний засіб для автоматизації взаємодії з Telegram-групами. Реалізоване рішення забезпечує автоматичне приєднання до груп за списком посилань, масову розсилку повідомлень з підтримкою графічних вкладень, ведення журналів результатів і обробку можливих помилок. Усе програмне забезпечення створене мовою Python із використанням бібліотеки Telethon, що дозволило працювати безпосередньо з Telegram API від імені користувача.

Функціонал програми був розділений на два основних модулі: Invite.py - для приєднання до груп, та Send.py - для надсилання повідомлень. Завдяки використанню зовнішніх конфігураційних файлів і динамічної генерації затримок, система адаптована до обмежень Telegram та зберігає гнучкість у налаштуванні.

Тестування показало високу стабільність і надійність роботи системи. Програма успішно виконувала всі функції навіть при наявності помилок або непередбачених ситуацій, не перериваючи загальне виконання. Всі дії супроводжувалися створенням детальних логів, які можуть бути використані для аналізу. Практичне тестування в реальному середовищі – зокрема в Управлінні стратегічних розслідувань в Черкаській області ДСР Національної поліції України – підтвердило ефективність, швидкодію та придатність рішення для службового використання.

Відповідно до структури проєкту, у Додатку А наведено ключові фрагменти коду, що демонструють реалізацію основних функцій програмного засобу.

Розроблене рішення можна масштабувати, адаптувати під інші сценарії або об’єднати з додатковими функціями – наприклад, розкладом розсилки, інтеграцією із зовнішніми базами даних або підтримкою багатьох акаунтів. Таким чином, запропонований інструмент може бути корисним для

адміністраторів Telegram-груп, маркетологів, службових структур і всіх, хто потребує автоматизації взаємодії з великою кількістю Telegram-спільнот.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1 Telegram APIs. Telegram APIs. URL: https://core.telegram.org/ (дата звернення: 13.02.2025)

2 Telethon’s Documentation – Telethon 1.40.0 documentation. Telethon’s Documentation – Telethon 1.40.0 documentation. URL: https://docs.telethon.dev/en/stable/ (дата звернення: 18.02.2025)

3 Logging HOWTO. Python documentation. URL: https://docs.python.org/3/howto/logging.html (дата звернення: 20.02.2025)

4 SQLite documentation. SQLite Home Page. URL: https://www.sqlite.org/docs.html (дата звернення:23.02.2025)

5 GitHub - LonamiWebs/Telethon: Pure Python 3 MTProto API Telegram client library, for bots too!. GitHub. URL: https://github.com/LonamiWebs/Telethon (дата звернення: 03.03.2025)

6 ECMA-262 - ecma international. Ecma International. URL: https://ecma-international.org/publications-and-standards/standards/ecma-262/ (дата звернення: 05.03.2025)

7 Python 3.13 documentation. Python documentation. URL: https://docs.python.org/3 (дата звернення: 06.03.2025)

8 GitHub - LonamiWebs/Telethon: Pure Python 3 MTProto API Telegram client library, for bots too!. GitHub. URL: https://github.com/LonamiWebs/Telethon (дата звернення: 09.03.2025)

9 Asyncio a synchronous I/O. Python documentation. URL: https://docs.python.org/3/library/asyncio.html (дата звернення: 14.03.2025)

10 Sqlite3 DB-API 2.0 interface for SQLite databases. Python documentation. URL: https://docs.python.org/3/library/sqlite3.html (дата звернення: 20.03.2025)

11 Telegram bot API. Telegram APIs. URL: https://core.telegram.org/bots/api (дата звернення: 01.04.2025)

12 PEP 8 – style guide for python code | peps.python.org. Python Enhancement Proposals (PEPs). URL: https://peps.python.org/pep-0008/ (дата звернення: 05.04.2025)

13 PEP 257 – docstring conventions | peps.python.org. Python Enhancement Proposals (PEPs). URL: https://peps.python.org/pep-0257/ (дата звернення: 08.04.2025)

14 8. errors and exceptions. Python documentation. URL: https://docs.python.org/3/tutorial/errors.html (дата звернення: 14.04.2025)

15 W3Schools.com. W3Schools Online Web Tutorials. URL: https://www.w3schools.com/python/module_random.asp (дата звернення: 20.04.2025)

16 W3Schools.com. W3Schools Online Web Tutorials. URL: https://www.w3schools.com/python/python_file_handling.asp (дата звернення: 20.04.2025)

17 Said van de Klundert. Python exceptions: an introduction – real python. Python Tutorials – Real Python. URL: https://realpython.com/python-exceptions/ (дата звернення: 20.04.2025)

ДОДАТОК А - ЛІСТИНГ ПРОГРАМНОГО КОДУ

Фрагмент А.1 – Загальна структура проекту

```
# Структура проєкту
# Change/
# ├── groupinvite.txt – посилання на групи
# ├── groups.txt – ID або username чатів
# ├── message.txt – текст повідомлення
# └── Images/ – зображення для розсилки

# Основні скрипти:
# Invite.py – приєднання до груп
# Send.py – масова розсилка
# session_*.session – збереження сесії користувача
# Logs/ – лог-файли з результатами
```

Фрагмент А.2 - Функція приєднання до Telegram-груп

```
async def join_groups(client, group_links):
    for link in group_links:
        try:
            print(f"📥 Приєднуюсь до: {link}")
            await client(JoinChannelRequest(link))
        except Exception as e:
            print(f"❌ Помилка приєднання до {link}: {e}")
```

Фрагмент А.3 - Цикл масової розсилки повідомлень

```
async def send_messages(client, message, chat_ids, images, min_delay, max_delay):
    for chat_id in chat_ids:
        try:
            entity = await client.get_entity(chat_id)
            sent_message = await client.send_message(entity, message, file=images if images else None)
            await asyncio.sleep(random.randint(min_delay, max_delay))
        except Exception as e:
            print(f"❌ Не вдалося надіслати: {chat_id} → {e}")
```

Фрагмент А.4 - Обробка виключень Telegram API

```
try:
    await client(JoinChannelRequest(link))
except FloodWaitError as e:
    print(f"⚠ Затримка {e.seconds} сек")
    await asyncio.sleep(e.seconds)
except UserBannedInChannelError:
    print("🚫 Користувач заблокований у каналі")
```

Фрагмент А.5 - Формування лог-файлу

```
def create_log_file():
    timestamp = datetime.datetime.now().strftime("%Y-%m-%d_%H-%M-%S")
    return os.path.join("Logs", f"log_{timestamp}.txt")

with open(log_file_path, "a", encoding="utf-8") as log_file:
    log_file.write("✅ Надіслано в: ExampleGroup\n")
```

Фрагмент А.6 - Авторизація користувача

```
phone_number = input("📱 Введіть номер телефону: ")
client = TelegramClient(f'session_{phone_number}', api_id, api_hash)
await client.start(phone_number)
print("✅ Успішний вхід у Telegram!")
```

Фрагмент А.7 - Основний цикл приєднання до груп

```
group_links = load_group_links()
await join_groups(client, group_links)
print("🎉 Завершено приєднання до всіх груп!")
```

Фрагмент А.8 - Генерація випадкової затримки

```
delay = random.randint(min_delay, max_delay)
print(f"⏳ Чекаємо {delay} сек...")
await asyncio.sleep(delay)
```

Фрагмент А.9 - Запис інформації у лог-файл

```
with open(log_file_path, "a", encoding="utf-8") as log_file:
    log_file.write(f"✅ Надіслано в: {chat_title}\n🔗 {message_link}\n")
```

Фрагмент А.10 - Обробка помилок під час надсилання

```
except PeerIdInvalid:
    print(f"❌ Недійсний ID чату: {chat_id}")
except ChatWriteForbiddenError:
    print(f"🚫 Немає прав на надсилання в: {chat_id}")
except Exception as e:
    print(f"❗ Невідома помилка: {e}")
```