МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет

Факультет електроніки та інформаційних технологій

Кафедра комп'ютерних наук

«До захисту допущено»

В.о. завідувача кафедри

_____ Ігор ШЕЛЕХОВ

(підпис)

09 червня 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

зі спеціальності 122 – Комп'ютерних наук,

освітньо- професійної програми «Інформатика»

на тему: «Інформаційна система управління освітнім фріланс-сервісом на базі месенджера Telegram»

здобувача групи ІН-92 Подвези Сергія Юрійовича

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

_____ Сергій ПОДВЕЗА

(підпис)

Керівник
заступник декана з інформатизації,
доцент,
кандидат технічних наук Сергій ПЕТРОВ _____

(підпис)

Суми – 2023

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавра

зі спеціальності 122 - Комп'ютерних наук, освітньо-професійної програми «Інформатика»
здобувача групи ІН-92 Подвези Сергія Юрійовича

1. Тема роботи: «Інформаційна система управління освітнім фріланс-сервісом на базі месенджера Telegram»
затверджую наказом по СумДУ від *«01» червня 2023 р. № 0475-VI*
2. Термін здачі здобувачем кваліфікаційної роботи *до 09 червня 2023 року*
3. Вхідні дані до кваліфікаційної роботи _____
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1) Аналіз проблеми предметної області, постановка й формування завдань дослідження. 2) Огляд сервісів-аналогів та технологій, що використовуються для створення Telegram-ботів. 3) Розробка інформаційної системи управління освітнім фріланс-сервісом на базі месенджера Telegram. 4) Аналіз результату.*
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)_____
6. Консультанти до проєкту (роботи), із значенням розділів проєкту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «____» _____ 20 ___ р.

Завдання прийняв до виконання _____ Керівник _____
(підпис) (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Аналіз проблеми предметної області, постановка й формування завдань дослідження*		
2	*Огляд сервісів-аналогів та технологій, що використовуються для створення Telegram-ботів*		
3	*Розробка інформаційної системи управління освітнім фріланс-сервісом на базі месенджера Telegram*		
4	*Аналіз результату*		
5	*Оформлення пояснювальної записки до кваліфікаційної роботи*		

Здобувач вищої освіти _____ Керівник _____
(підпис) (підпис)

АНОТАЦІЯ

Записка: 46 стр., 37 рис., 1 додаток, 33 використаних джерела.

Обґрунтування актуальності теми роботи – тема кваліфікаційної роботи є актуальною, оскільки присвячена вирішенню проблеми з відсутністю зручного інструменту для ефективної комунікації між здобувачами освіти. Робота зосереджена на врахуванні недоліків наявних фріланс-платформ, таких як обов'язкова реєстрація та верифікація, платні публікації, перевірка особистих повідомлень та способів оплати. Розробка інформаційної системи на базі Telegram, що складається з бота, каналу і груп, дозволяє позбутися цих обмежень і забезпечує більшу свободу користувачам.

Об’єкт дослідження – процес управління освітнім фріланс-сервісом на базі месенджера Telegram.

Мета роботи – розробка інформаційної системи управління освітнім фріланс-сервісом на базі месенджера Telegram.

Методи дослідження – аналіз існуючих фріланс-платформ, їх функціональності, обмежень та проблем, пошук доступних і оптимальних інструментів для розробки Telegram-ботів.

Результати – розроблено інформаційну систему, що дозволяє користувачам створювати власні завдання та оголошення, а також відгукуватися на завдання та оголошення інших. Оголошення – пост у вільній формі, створений для реклами послуг, які надає виконавець. Завдання – пост, що обов’язково складається з назви предмету, опису та ціни, опціонально можна додати фото або файл. Система пропонує два види завдань: захищене і звичайне. Захищені завдання були впроваджені з метою зниження ризику шахрайства, оскільки виконання таких завдань дозволено лише верифікованим виконавцям. Кожен пост також

проходить верифікацію. В рамках розробки системи було проведено тестування, в якому брали участь кілька користувачів одночасно, що дозволило перевірити функціональність та ефективність системи в реальних умовах використання.

ІНФОРМАЦІЙНА СИСТЕМА, ФРІЛАНС, TELEGRAM-БОТ, PYTON, AIOGRAM, АСИНХРОННІСТЬ, SQLITE

ЗМІСТ

ВСТУП

Актуальність. Актуальність роботи обумовлена потребою в сервісі з відсутністю в ньому певних обмежень вже існуючих фріланс-платформ, таких як обов’язкова реєстрація та верифікація, платні публікації, перевірка особистих повідомлень та способів оплати. Також розробка такої ІС є перспективною, оскільки рішення, які використовуються в даному проекті, можна буде використати при створені подібної системи, але орієнтованої на більшу користувацьку базу, з можливістю монетизації.

Мета дослідження. Розробка інформаційної системи, що допоможе цільовому користувачу отримати оперативну допомогу у вирішенні освітніх задач. Основний акцент роботи спрямовується на спрощення та прискорення процесу комунікації між замовником та виконавцем.

Об’єкт дослідження. Процес управління освітнім фріланс-сервісом на базі месенджера Telegram.

Предмет дослідження. Функціональні можливості системи, взаємодія користувачів, процес створення та виконання завдань, механізми верифікації, комунікаційні процеси, аналіз ефективності системи.

Гіпотеза. Розроблена інформаційна система на базі месенджера Telegram буде забезпечувати зручну та ефективну комунікацію між користувачами, сприятиме зменшенню обмежень, які присутні в існуючих фріланс-платформах, та покращить процес навчання шляхом створення оптимальних умов для виконання завдань та взаємодії між виконавцями та замовниками.

Наукова новизна. Описане у даній роботі програмне рішення дозволить досягти більшої ефективності при комунікації між здобувачами освіти. Також робота відкриває нові можливості для використання месенджера Telegram як платформи для освітніх цілей.

Структура. Дана робота складається зі вступу, інформаційного огляду, вибору методу розв’язання задачі, програмної реалізації, висновків, списку використаних джерел та додатків.

1 ІНФОРМАЦІЙНИЙ ОГЛЯД

1.1 Аналіз проблеми

Розвиток технологій та глобалізація створили нові можливості для освіти і праці на відстані. Концепція фрілансу стала популярною, особливо серед молодих людей, які шукають гнучкі робочі умови та можливість працювати будь-де.

Зростання кількості фрілансерів свідчить про зростаючий інтерес до цієї форми зайнятості. Статистика свідчить, що в останні роки кількість фрілансерів значно збільшилась, особливо через пандемію коронавірусу (див. Рисунок 1.1). Люди віддають перевагу гнучкості та свободі, яку надає цей вид зайнятості, і обирають роботу, що відповідає їхнім навичкам та інтересам. Серед фрілансерів багато тих, хто пропонує допомогу в освітньому процесі, наприклад репетиторство. Попит на дані послуги є завжди, тому створення інформаційної системи для керування освітнім фріланс-сервісом може допомогти замовникам і виконавцям ефективніше співпрацювати та задовольняти потреби одне одного.

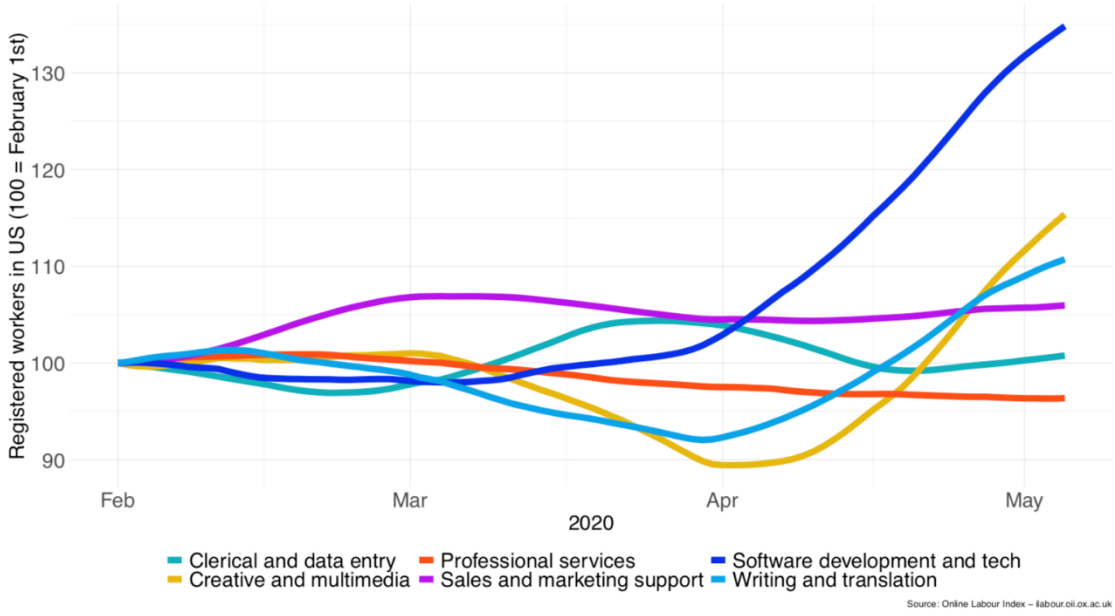

Рисунок 1.1 – Графік кількості зареєстрованих профілів на фріланс-платформах

Часто користувачі фріланс-платформ не використовують вбудовані чати, а спілкуються через месенджери, такі як WhatsApp, Viber або Telegram, навіть

якщо це заборонено правилами платформи. Тому сервіс, побудований на базі месенджера, дозволить в суттєво спростити та пришвидшити процес взаємодії між замовником і виконавцем.

Месенджер обов’язково повинен мати інструмент для автоматизації роботи інформаційної системи, наприклад можливість створення чат-ботів. Загалом, чат-боти є універсальним інструментом, який використовується в різних сферах діяльності. Вони забезпечують автоматизовану комунікацію з користувачами, що полегшує взаємодію та надає швидку та ефективну підтримку. Чат-боти виявляються особливо корисними в ситуаціях, коли потрібно взаємодіяти з великою кількістю користувачів одночасно або коли завдання можна автоматизувати. Часто цих віртуальних помічників використовують великі компанії для здійснення зворотного зв’язку або виконання специфічних функцій, наприклад пошуку працюючих відділень для поштової служби або розрахування прибутку від депозиту, якщо це бот банку. Їх часто інтегрують в різні месенджери, наприклад Telegram. В ньому боти вперше з’являються в 2015 році і швидко набирають популярності завдяки своїй доступності і відносній простоті Telegram API.

Також Telegram є широко поширеним месенджером зі значною кількістю користувачів, який надає зручну платформу для комунікації та обміну інформацією. Даний месенджер використовується мільйонами людей по всьому світу. Згідно з офіційною статистикою Telegram, на кінець 2022 року кількість активних щомісячних користувачів досягла 700 мільйонів (див. Рисунок 1.2).

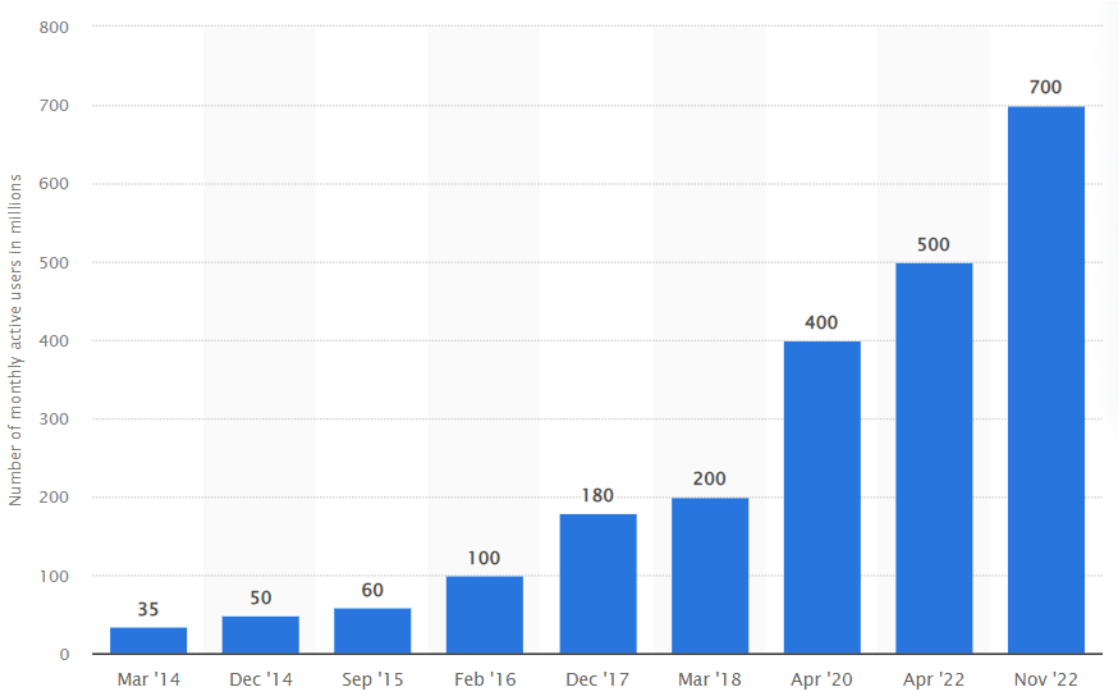

Рисунок 1.2 – Кількість активних щомісячних користувачів Telegram

Наявність ботів та великої бази користувачів робить Telegram привабливим середовищем для реалізації фріланс-сервісу.

1.2 Аналіз існуючих фріланс-сервісів

При виборі фріланс-платформи, необхідно ретельно аналізувати їх переваги та недоліки. Таблиця 1.1 описує деякі з найпопулярніших фріланс-сервісів та їх характеристики.

Таблиця 1.1 – Порівняння найпопулярніших фріланс-сервісів

Фріланс-сервіс	**Переваги**	**Недоліки**
Upwork	Велика база фрілансерів з різними навичками	Висока конкуренція
	Можливість пошуку професіоналів у різних сферах	Висока комісія
		Обмеження на спілкування в сторонніх сервісах

Продовження табл. 1.1

Фріланс-сервіс	**Переваги**	**Недоліки**
Freelancer	Велика кількість проектів	Висока конкуренція
	Система застав для забезпечення виконання проекту	Висока комісія
		Якість виконання проектів не контролюється адміністрацією
Fiverr	Широкий вибір послуг та цінових пропозицій	Менша гнучкість у виборі проектів та фрілансерів
	Простий процес замовлення та оплати	Обмежені можливості налаштування профілю
	Висока якість виконання послуг	Відсутність контролю з боку платформи над виконанням проектів

На основі аналізу наведених переваг і недоліків, можна зазначити, що основною перевагою системи на базі Telegram є відсутність певних обмежень, які характерні для інших фріланс-платформ. Наприклад, у порівнянні з відомими сервісами, де часто потрібні обов'язкова верифікація, платні публікації, перевірка особистих повідомлень та обмежені способи оплати, система на базі Telegram надає більшу свободу користувачам.

1.3 Аналіз аудиторії та її потреб

Аналіз аудиторії є ключовим етапом для розуміння потреб та очікувань користувачів. Для забезпечення успішної розробки та впровадження системи, необхідно мати вичерпну інформацію про цільову аудиторію, її розміри та поведінку.

Цільова аудиторія ІС в більшості буде складатися з студентів та випускників, тобто людей віком 17–25 років. Це вимагає уваги до деяких особливостей, що враховують їх потреби та вподобання. Ось кілька ключових особливостей, які варто врахувати:

1) **Мобільність**. Середній час, витрачений на мобільні додатки, зростає кожного року. Тому важливо забезпечити зручний інтерфейс для використання сервісу на смартфонах.
2) **Варіативність.** Кожен користувач має свої унікальні потреби та навчальні цілі. Створення системи, яка дає вибір, буде сприяти збільшенню потенційної аудиторії і відповідно забезпечувати більший шанс виконання задачі замовника.
3) **Візуальна привабливість**. Молоді люди частіше звертають увагу на різні моменти в дизайні. Хоча Telegram не надає можливість редагувати основні елементи, але привабливість системи можна створити за допомогою використання емодзі і комбінації різних варіантів шрифтів.
4) **Функціональність та мінімалізм.** Користувач повинен чітко розуміти всі можливості системи і як вона працює. Важливо передбачити хід думок користувача і створити ефективний інтерактивний дизайн, який не буде викликати розгубленість.

1.4 Аналіз етапів створення Telegram-бота

Так як найважливішою та найскладнішою частиною ІС є Telegram-бот, то його створення потребує аналізу та систематичного підходу для досягнення максимальної якості та продуктивності результатів.

Перейдемо до детального розгляду етапів створення Telegram-бота:

1) **Аналіз вимог та формулювання цілей**. На цьому етапі важливо визначити функціональні та нефункціональні вимоги до бота, його основні функції та мету. Також потрібно враховувати специфіку цільової аудиторії, її потреби та очікування. Чітко сформульовані вимоги є основою подальшої розробки.
2) **Проектування архітектури**. На цьому етапі вирішуються питання структури та організації бота. Розробляється схема взаємодії між

користувачем та ботом, визначаються компоненти системи та їх функціональність. Також важливо врахувати можливості платформи Telegram та її API для ефективної інтеграції.

3) **Розробка та реалізація функціоналу**. На цьому етапі реалізуються функції взаємодії з користувачем, обробка повідомлень та виконання запитів, які були сформовані в попередніх пунктах. Також можуть бути розроблені додаткові модулі, які розширюють функціональність бота.
4) **Тестування та налагодження**. Під час цього етапу перевіряється відповідність розробленого бота вимогам, виявляються та виправляються помилки. Тестування проводиться на різних сценаріях взаємодії та з різними типами користувачів.
5) **Впровадження та підтримка**. На цьому етапі запускається бот та надається доступ користувачам. Проводяться додаткові заходи з підтримки, вдосконалення та відладки. Забезпечується моніторинг роботи бота, а також, при потребі, впроваджуються зміни та оновлення.

Узагальнюючи, розбиття процесу створення ІС на етапи дозволяє систематизувати та якісно планувати роботу над проектом. Цей підхід дозволяє з більшою вірогідністю досягти успішного результату та забезпечити ефективну роботу сервісу.

1.5 Інструменти для розробки Telegram-ботів

Серед інструментів для розробки Telegram-ботів існує багато вже готових конструкторів. Вони мають свої переваги, свої недоліки та свою вартість. Конструктори чат-ботів дозволяють створювати простих ботів. Чат-боти можуть приймати заявки, видавати відповіді на основі шаблонів, розсилати листи та обробляти платежі. Ознайомившись з документацією конструкторів, можна

підібрати відповідний інструмент, або обрати один із найдоступніших і найзручніших.

Бота зі складною логікою краще створювати власноруч або з допомогою досвідченого розробника. Можна використовувати будь-яку мову програмування — від PHP та Node.js до Java та Ruby, але найпопулярнішою мовою для створення ботів є Python. У його екосистемі багато відкритих бібліотек та готових рішень, також він легкий у вивченні. Для створення Telegram-ботів на Python існує кілька десятків бібліотек (див Таблиця 1.2). Вони відрізняються популярністю, розміром ком'юніті та функціональністю.

Таблиця 1.2 – Характеристики найпопулярніших бібліотек для розробки Telegram-ботів

Бібліотека	**Характеристика**
aiogram	Сучасна бібліотека, що набирає популярності. Працює з асинхронним підходом виконання коду. Це дозволяє не зупиняти роботу робота при очікуванні відповіді користувача. aiogram має детальну документація та велике ком'юніті.
python-telegram-bot	Одна з перших бібліотек для створення роботів. Відрізняється від aiogram синхронним підходом до роботи, тобто при очікуванні відповіді від користувача виконання коду зупиняється.
TeleBot	Бібліотека для створення простих ботів, що дозволяє працювати з асинхронним та синхронним підходом на вибір. Підходить для невеликих проектів.

Бібліотеки, як і конструктори, мають різні можливості та різну складність опанування. Тому обирати бібліотеку потрібно в залежності від поставлених задач.

Для створення ІС була обрана комбінація мови Python та бібліотеки aiogram, адже aiogram надає широкі можливості для роботи з Telegram API. Вона підтримує всі основні функції, такі як обробка повідомлень, звичайних та інлайн-клавіатур. Бібліотека також надає інструменти для роботи зі стікерами, фотографіями, відео та аудіо, має зрозумілий синтаксис, що спрощує процес розробки ІС. Чітка документація та активна спільнота користувачів допоможуть зрозуміти та вирішити будь-які проблеми, які виникають під час розробки. Бібліотека постійно оновлюється та підтримується активною командою розробників, що гарантує безпеку та стабільну роботу інформаційної системи.

При розробці ІС, найважливішим параметром є асинхронність бібліотеки, що дозволяє обробляти багато запитів одночасно і досягати високої продуктивності. Це особливо важливо у випадку великої кількості користувачів або потенційно великого навантаження на сервер.

Загалом, бібліотека aiogram має весь необхідний функціонал для реалізації проекту.

1.6 Постановка задачі

Створення фріланс-сервісу на основі месенджера Telegram з функціоналом керування через Telegram-бота можна розділити на наступні етапи:

1) Аналіз потреб ринку:
    a. Дослідити потреби та очікування споживачів.
    b. Дізнатися про існуючі аналогічні сервіси та їхні можливості.
2) Визначення можливостей сервісу на основі даних з минулого пункту:
    a. Визначити потенційні ризики та проблеми для проекту.
    b. Визначити необхідний функціонал сервісу.

c. Розробити технічне завдання на розробку сервісу та бота.

3) Розробка Telegram-бота:
   a. Підготувати базу для створення сервісу (завантажити потрібне програмне середовище, бібліотеки, створити потрібні канали, чати і т.д.).
   b. Написати код для Telegram-бота згідно технічного завдання.
   c. Протестувати та налаштувати сервіс.
   d. Провести тестування сервісу в реальних умовах з добровольцями.
   e. Виправити всі помилки, які знайшлися на етапі тестування добровольцями.
4) Запуск проекту:
   a. Підготувати місце для хостингу.
   b. Розгорнути сервіс на сервері.
5) Підтримка та розвиток проекту:
   a. Забезпечувати надійність та безпеку сервісу.
   b. Взаємодіяти з користувачами для отримання фідбеку щодо роботи сервісу та бота.
   c. Розробити плани на майбутнє розширення та вдосконалення сервісу.
   d. Проводити постійний моніторинг функціоналу та роботи сервісу для запобігання можливих помилок або несправностей.
   e. В разі потреби виправляти баги, вдосконалювати та доповнювати функціонал бота.

2 ВИБІР МЕТОДУ РОЗВ'ЯЗАННЯ ЗАДАЧІ

2.1 Структура проекту

Вибір правильної структури проекту є важливим етапом розробки. При розробці нескладних Telegram-ботів однією людиною, не варто продумувати складну структуру, адже це тільки ускладнить простий проект. Достатньо залишити весь код в одному файлі. Але при розробці складних програм, в команді розробників, завжди використовується модульний підхід. Ось декілька ключових переваг модульного підходу над однофайловим:

- **Модульність та розширюваність**. В модульному проекті бота функціональність розділена на окремі модулі або компоненти. Це дозволяє зручно розробляти, тестувати та підтримувати кожну частину окремо. Крім того, модульна архітектура дозволяє легко додавати нові функції та розширювати функціонал без змін у загальній структурі проекту.
- **Повторне використання**. Завдяки модульній структурі, окремі модулі можуть бути легко використовувані в інших проектах або розширюватись для виконання подібних завдань. Це сприяє ефективному використанню ресурсів та зменшенню зусиль для розробки нових функцій.
- **Тестування та дебагінг**. Модульна структура спрощує процес тестування та дебаггінгу. Оскільки кожен модуль може бути тестований окремо, виявлення та виправлення помилок стає більш простим і швидким.
- **Командна розробка**. Модульний проект дозволяє команді розробників працювати над різними модулями паралельно. Кожен член команди може відповідати за розробку, тестування або підтримку певного модуля, що покращує продуктивність та сприяє швидкому впровадженню нових функцій.

Враховуючи ці переваги, модульний підхід до створення Telegram-бота є більш ефективним та масштабованим в порівнянні з однофайловим рішенням. Він забезпечує кращу організацію коду, розширюваність, повторне використання та полегшує процес тестування та командну розробку.

2.2 Проектування бази даних

База даних є невід'ємною частиною даного проекту. По-перше, нам потрібно вирішити, яку базу даних ми будемо використовувати. Існує багато систем управління базами даних, серед яких MySQL, PostgreSQL, Oracle, SQLite та інші. В нашому випадку буде використовуватись SQLite, як найзручніша та найлегша СУБД для Python. В ній будуть зберігатися дані про верифікованих виконавців, завдання та оголошення користувачів та відгуки на них (див. Рисунок 2.1).

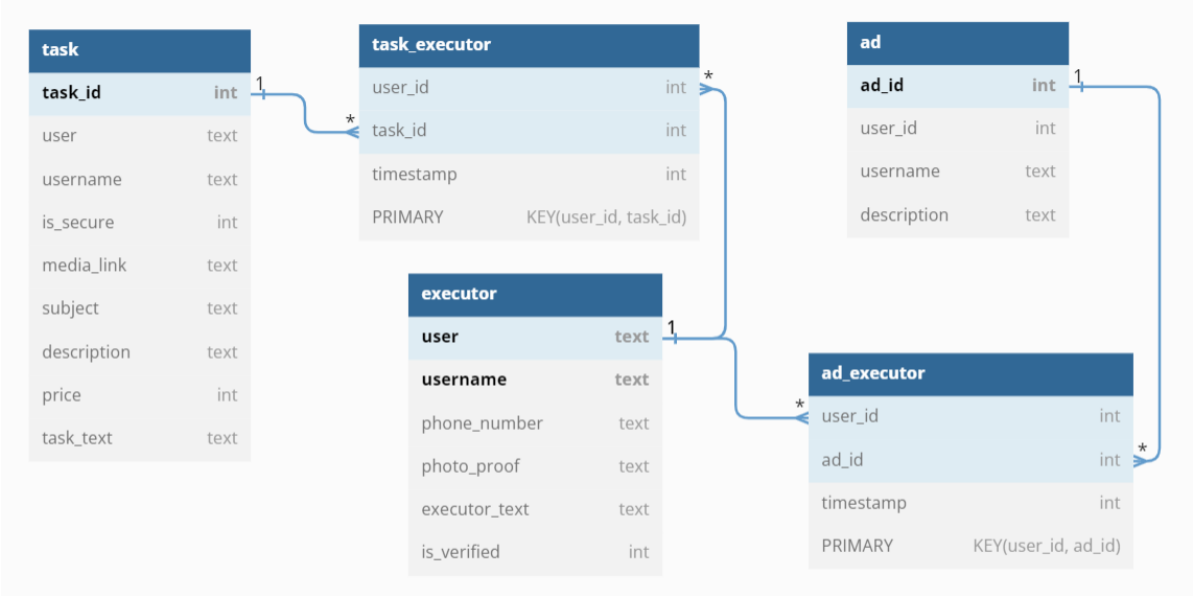

Рисунок 2.1 – Структура бази даних

2.3 Огляд інструментів для розробки

Для розробки програмного забезпечення на мові програмування Python можна використовувати майже будь-який редактор коду, а для зручного адміністрування бази даних можна використати DBeaver (див. Таблиця 2.1).

Таблиця 2.1 – Характеристики використаних для розробки інструментів

Інструмент	**Характеристика**
VS Code	В даному середовищі розробки Python IDE працює через розширення Python Tools for Visual Studio. Має велику базу розширень для роботи, що впливають як на функціонал, так і на дизайн. IDE гарно оптимізована та має зрозумілий інтерфейс.
DBeaver	Зручний засіб для взаємодії з системою керування базами даних SQLite.

Вибір наведених інструментів обумовлений їх зручним та зрозумілим інтерфейсом. Крім зазначених додатків, під час написання коду будуть використані модульна система та бібліотека aiogram для взаємодії з Telegram API – набором програмного коду для передачі даних від бота на сервер Telegram і назад.

2.4 Клієнтська частина системи

User-friendly дизайн безпосередньо впливає на враження від користування певним продуктом. Зазначений розроблюваний додаток буде характеризуватися простим і функціональним дизайном, що сприятиме легкому доступу користувача до всіх необхідних функцій. Замість ручного вводу команд, будуть використовуватися інтерактивні кнопки (див. Рисунок 2.2). Інформація буде подаватися максимально стисло.

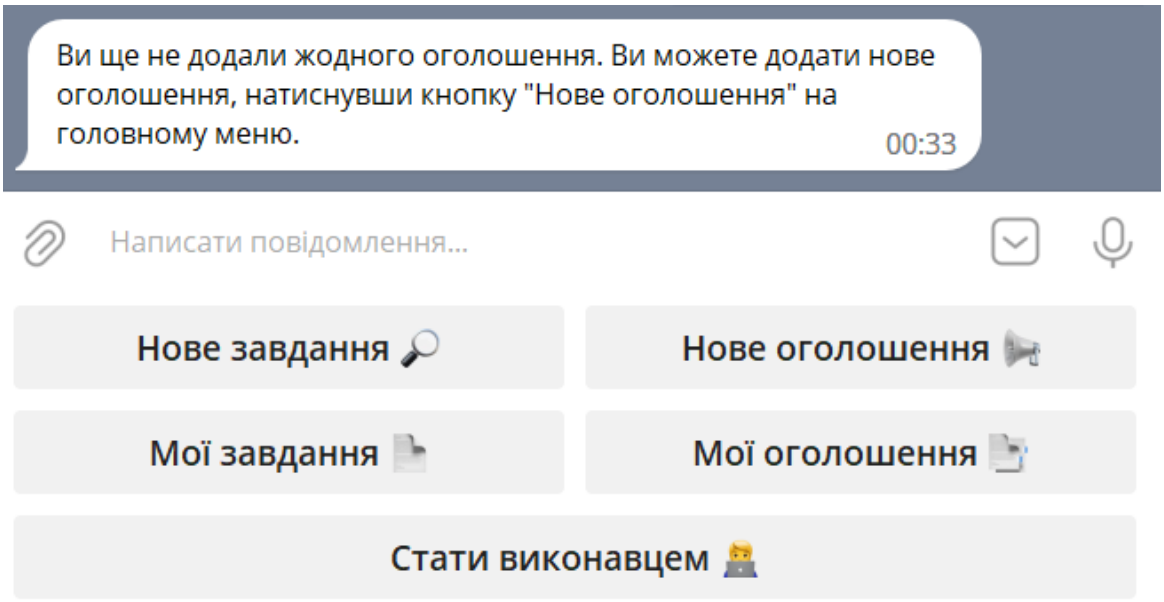

Рисунок 2.2 – Клавіатура головного меню

Інформування користувача про можливості системи буде відбуватися через відповідні повідомлення (див. Рисунок 2.3).

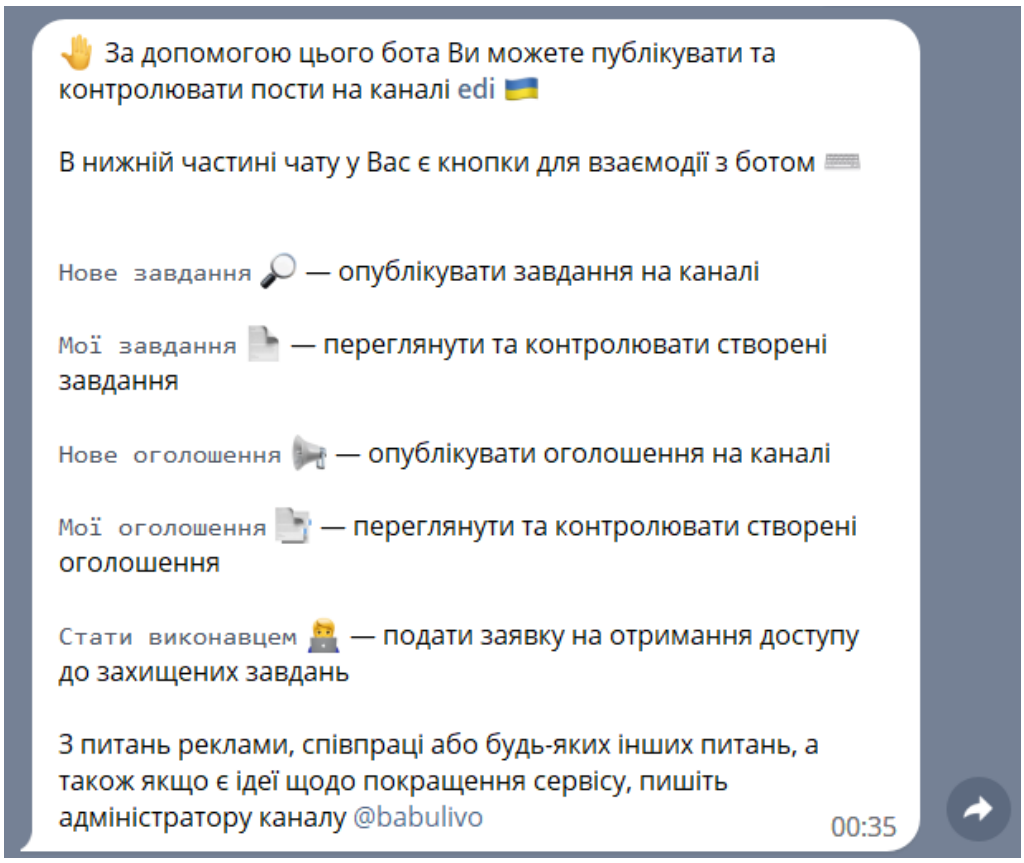

Рисунок 2.3 – Приклад інформаційного повідомлення

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Визначення технічного завдання

У процесі розробки будь-якої програмної системи, ключовим етапом є формування технічного завдання (див. Додаток А). Цей документ визначає вимоги та специфікації, що визначають функціональні та нефункціональні характеристики системи. Технічне завдання є основою для подальшої розробки, оскільки містить опис функціональності, взаємодії з користувачем, архітектурні рішення, вимоги до безпеки та продуктивності, а також інші важливі параметри. Успішна розробка програмної системи залежить від якісного формулювання технічного завдання, яке уникне непорозумінь і забезпечить успішну інтеграцію та функціонування системи.

3.2 Налаштування середовища розробки

Перед початком реалізації бота, потрібно завантажити інтерпретатор мови Python та додаток для роботи з цією мовою у середовищі VS Code. Додаток є обов’язковою частиною для комфортної роботи, адже надає розширену підтримку Python, включаючи підказки, валідацію синтаксису, автодоповнення, відлагодження та інструменти для управління середовищем віртуальних середовищ.

Крім того, у VS Code доступні розширення для роботи з популярними фреймворками та інструментами Python, такими як Django, Flask, Pytest, Jupyter Notebook і багато інших. Ці розширення надають додаткові функціональні можливості та інструменти, що спрощують розробку та підтримку проектів Python.

VS Code також має вбудовану підтримку системи керування версіями Git, що дозволяє легко працювати з репозиторіями коду та використовувати всі переваги розподіленої системи керування версіями.

3.3 Визначення архітектури додатку

Організація проекту та визначення архітектури додатку є важливими етапами у процесі розробки програмного продукту. Вони визначають структуру модулів та зв'язки між ними, що впливає на ефективність, розширюваність та підтримку проекту. Визначення архітектури додатку полягає у розробці плану, який визначає, як компоненти системи будуть співпрацювати між собою. Це включає в себе визначення структури модулів, функціональних блоків, їх залежностей та взаємодій. Вірно побудована архітектура сприяє чіткості, організованості та кращому розумінню системи, що полегшує розробку, тестування та підтримку.

Одним з ключових аспектів організації проекту є структура модулів (див. Рисунок 3.1). Вона визначає, як компоненти проекту будуть організовані та розподілені. Часто використовується модульна структура, де різні функціональності розміщені в окремих модулях. Це дозволяє тримати код добре організованим, зменшує його залежності та зменшує дублювання компонентів.

Зв'язки між модулями також мають важливе значення. Це визначає, які дані та повідомлення будуть передаватись між модулями та як вони взаємодіятимуть. Встановлення правильних зв'язків дозволяє досягти високої зв'язності модулів та низької залежності, що сприяє зручності модифікації та розширення проекту.

Основними при створенні бота є модулі хендлерів користувача (див. Рисунок 3.2) та адміністратора (див. Рисунок 3.3). Кожен модуль містить декілька файлів, але головним є файл з назвою __init__, який автоматично викликається при імпорті модуля. Також за допомогою цього модуля можна обмежити використання певних змінних, функцій або хендлерів, імпортуючи не весь модуль, а тільки потрібний функціонал.

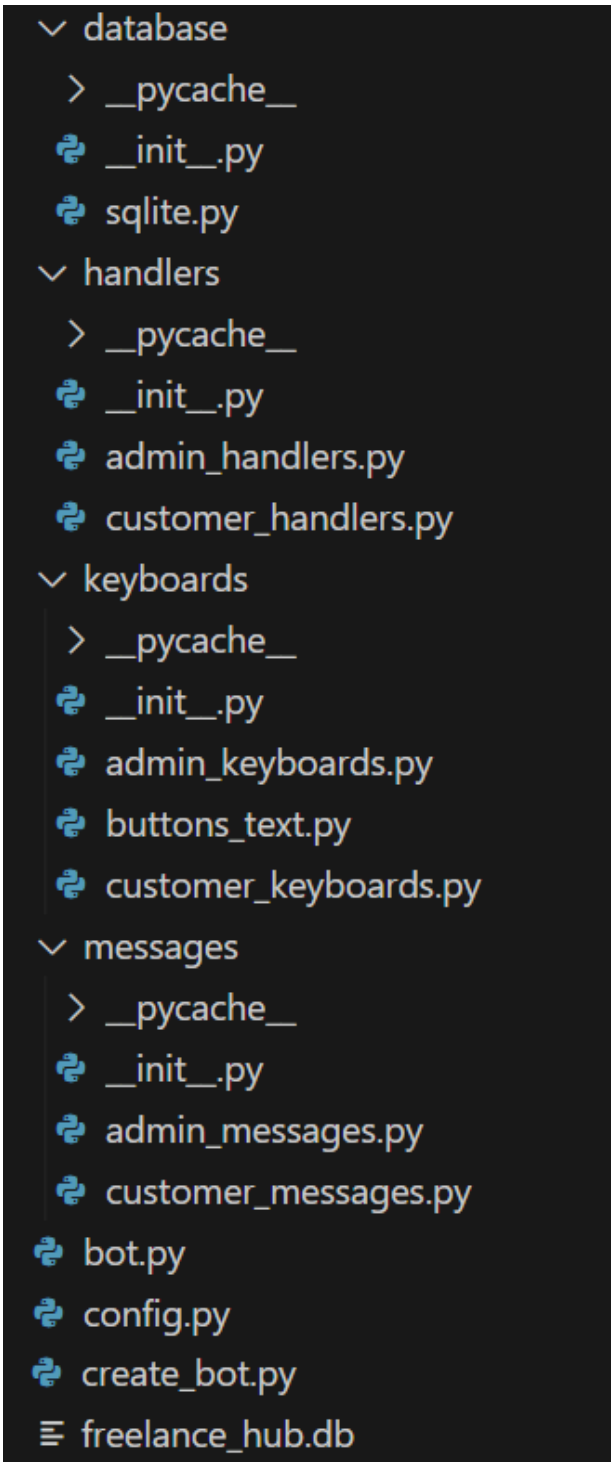

Рисунок 3.1 – Архітектура проекту

```python
def register_handlers_client(dp : Dispatcher):
    dp.register_message_handler(start_command, commands=['start'])
    dp.register_callback_query_handler(subscribed_callback, text=CUSTOMER_BUTTONS_CB['subscribed_cb'])
    dp.register_message_handler(cancel_action, text=CUSTOMER_BUTTONS['cancel'], state='*')
    dp.register_message_handler(become_executor, text=CUSTOMER_BUTTONS['become_executor'])
    dp.register_message_handler(get_contact, content_types=types.ContentType.CONTACT, state=ExecutorVerificationStatesGroup.contact)
    dp.register_message_handler(check_photo, lambda message: not message.photo, state=ExecutorVerificationStatesGroup.photo)
    dp.register_message_handler(load_photo_proof, content_types=['photo'], state=ExecutorVerificationStatesGroup.photo)
    dp.register_message_handler(confirm_ver, text=CUSTOMER_BUTTONS['confirm'], state=ExecutorVerificationStatesGroup.confirmation)
    dp.register_message_handler(new_task, text=CUSTOMER_BUTTONS['new_task'])
    dp.register_message_handler(new_ad, text=CUSTOMER_BUTTONS['new_ad'])
    dp.register_message_handler(load_ad_description, state=NewAdStatesGroup.description)
    dp.register_message_handler(confirm_ad, text=CUSTOMER_BUTTONS['confirm'], state=NewAdStatesGroup.confirmation)
    dp.register_message_handler(my_ads, text=CUSTOMER_BUTTONS['my_ads'])
    dp.register_message_handler(is_secure_task, text=[CUSTOMER_BUTTONS['secured_task'], CUSTOMER_BUTTONS['unsecured_task']], state=Ne
    dp.register_message_handler(skip_step, text=CUSTOMER_BUTTONS['skip'], state=NewTaskStatesGroup.media_link)
    dp.register_message_handler(check_link, lambda message: not re.match(fr"{MEDIA_CHAT_LINK}\d+$", message.text), state=NewTaskState
    dp.register_message_handler(load_link, lambda message: re.match(fr"{MEDIA_CHAT_LINK}\d+$", message.text), state=NewTaskStatesGrou
    dp.register_message_handler(load_subject, state=NewTaskStatesGroup.subject)
    dp.register_message_handler(check_prohibited_characters, lambda message: any(char in message.text for char in ['&', '<', '>', '\"
    dp.register_message_handler(load_description, state=NewTaskStatesGroup.description)
    dp.register_message_handler(check_price_not_digit, lambda message: not message.text.isdigit(), state=NewTaskStatesGroup.price)
    dp.register_message_handler(check_price_above_limit, lambda message: float(message.text) > 50000, state=NewTaskStatesGroup.price)
    dp.register_message_handler(load_price, state=NewTaskStatesGroup.price)
    dp.register_message_handler(confirm_task, text=CUSTOMER_BUTTONS['confirm'], state=NewTaskStatesGroup.confirmation)
    dp.register_message_handler(my_tasks, text=CUSTOMER_BUTTONS['my_tasks'])
    dp.register_message_handler(resend_files, content_types=[types.ContentType.PHOTO, types.ContentType.DOCUMENT])
    dp.register_callback_query_handler(get_order_callback, text=CUSTOMER_BUTTONS_CB['get_order_cb'])
    dp.register_callback_query_handler(respond_to_add_callback, text=CUSTOMER_BUTTONS_CB['respond_to_ad_cb'])
```

Рисунок 3.2 – Список хендлерів користувача

```python
def register_handlers_admin(dp : Dispatcher):
    dp.register_callback_query_handler(publish_task_cb, text=ADMIN_BUTTONS_CB['publish_task_cb'])
    dp.register_callback_query_handler(revoke_task_cb, text=ADMIN_BUTTONS_CB['revoke_task_cb'])
    dp.register_callback_query_handler(publish_ad_cb, text=ADMIN_BUTTONS_CB['publish_ad_cb'])
    dp.register_callback_query_handler(revoke_ad_cb, text=ADMIN_BUTTONS_CB['revoke_ad_cb'])
    dp.register_callback_query_handler(verify_executor_cb, text=ADMIN_BUTTONS_CB['verify_executor_cb'])
    dp.register_callback_query_handler(revoke_executor_cb, text=ADMIN_BUTTONS_CB['revoke_executor_cb'])
```

Рисунок 3.3 – Список хендлерів адміністратора

3.4 Налаштування підключення до бази даних

Для забезпечення підключення до бази даних, необхідно створити відповідний модуль. Цей модуль є важливою складовою реалізації бота, де зберігаються функції, що дозволяють встановити з'єднання з базою даних та створити відповідні таблиці, якщо їх не існує (див. Рисунок 3.4). Розміщення відповідної інформації у файлі sqlite.py забезпечує зручний та швидкий доступ до керування налаштуваннями бази даних та її функціями у контексті програмного додатку.

```
def sql_start():
    global base, cur
    base = sq.connect('freelance_hub.db')
    cur = base.cursor()
    if base:
        print('\nConnected to freelance_hub.db!')

    base.execute('''
        CREATE TABLE IF NOT EXISTS task (
            task_id INTEGER PRIMARY KEY,
            user TEXT,
            username TEXT,
            is_secure INT,
            media_link TEXT,
            subject TEXT,
            description TEXT,
            price INTEGER,
```

Рисунок 3.4 – З’єднання з базою даних та створення таблиць

3.5 Налаштування основних функцій

Можливість створення, перегляду та виконання завдань – головні функції для фріланс-сервісу. Інформаційна система складається не тільки з бота, а й каналу, в якому публікуються завдання. Процес створення завдання передбачає використання машини станів, тобто йде покроково. Це забезпечує чітке дотримання сценарію створення завдання, так як бот не реагує на сторонні команди. Створення оголошень та верифікація виконавці відбувається за ідентичним сценарієм. Реалізація клієнтської частини зберігається в окремому модулі.

Важливими складовими системи управління є групи адміністраторів, в які надходять створені завдання, оголошення та заявки на верифікацію. Даний функціонал реалізований через модуль адміністратора, що включає в себе набір відповідних клавіатур і хендлерів.

3.6 Використання системи користувачем.

Щоб користуватися сервісом, потрібно запустити бота, який спочатку перевіряє чи підписаний користувач на канал, в якому будуть публікуватися завдання та оголошення. Якщо користувач не підписаний, він отримує повідомлення з посиланням на канал (див. Рисунок 3.5). Після підписки потрібно натиснути кнопку «Вже підписався».

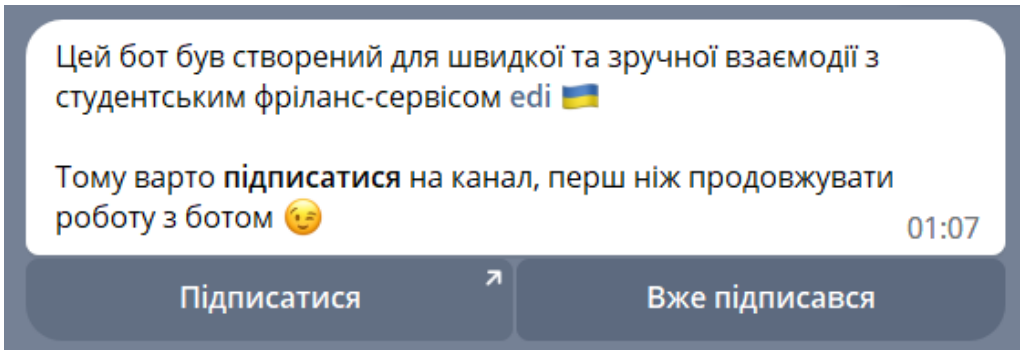

Рисунок 3.5 – Повідомлення для непідписаних користувачів

Якщо користувач не підписався на канал, але тисне кнопку «Вже підписався», з’явиться повідомлення про перевірку підписки (див. Рисунок 3.6).

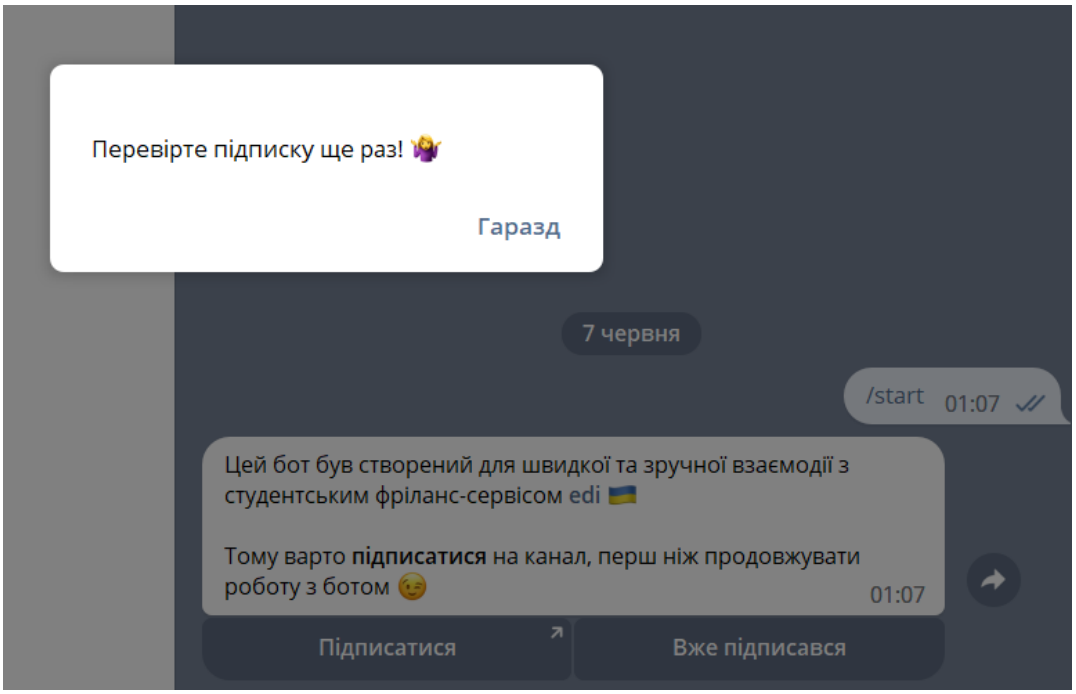

Рисунок 3.6 – Повідомлення про відсутність підписки

Після підписки на канал і натискання кнопки «Вже підписався» з’явиться інформаційне повідомлення про основні функції бота та клавіатура для керування ботом (див. Рисунок 3.7).

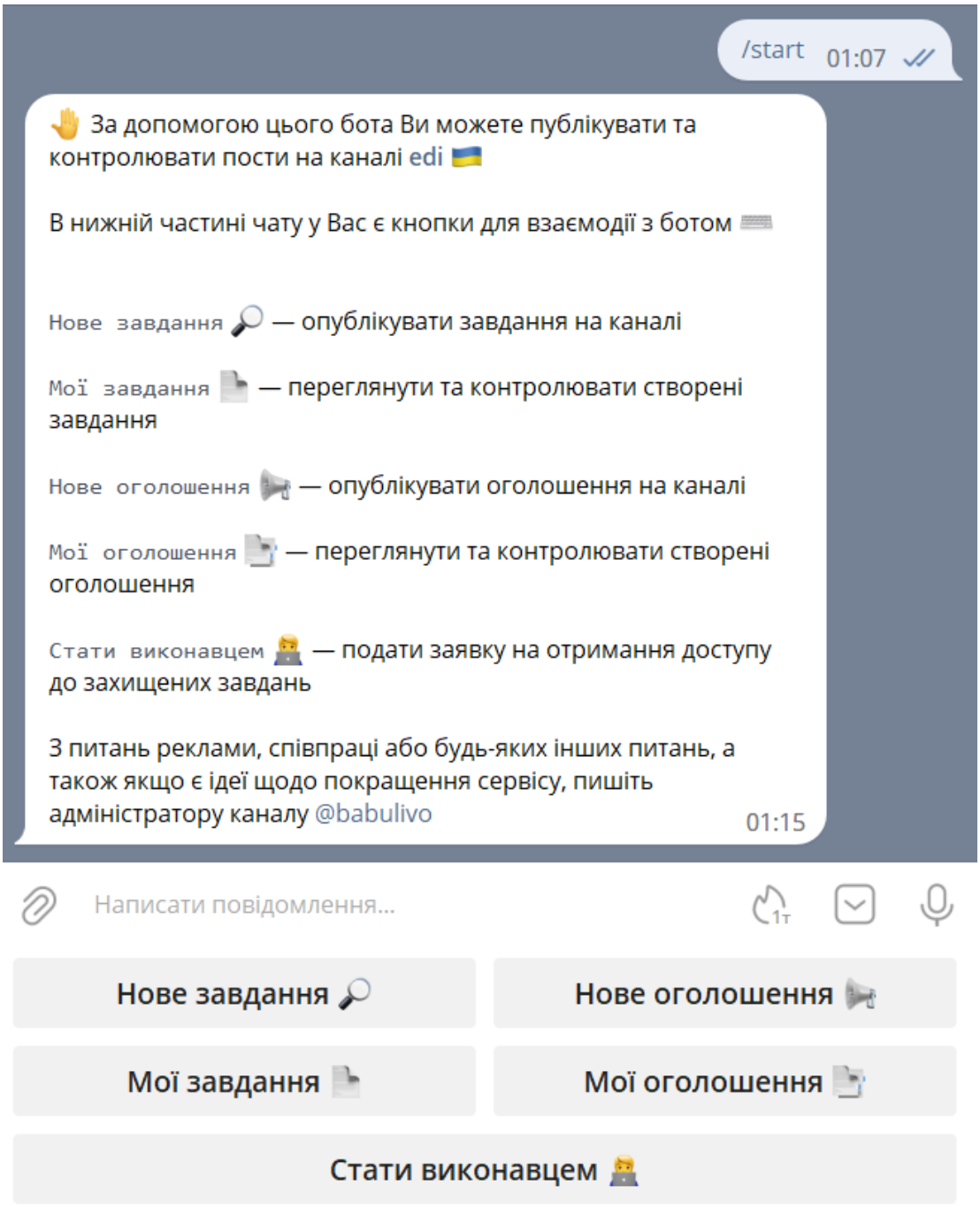

Рисунок 3.7 – Повідомлення і меню для підписаних користувачів

Для створення завдання потрібно натиснути відповідну кнопку, після чого бот відправить повідомлення про типи завдань та інструкцію для подальших дій. Також буде відправлена клавіатура для вибору типу завдання (див. Рисунок 3.8).

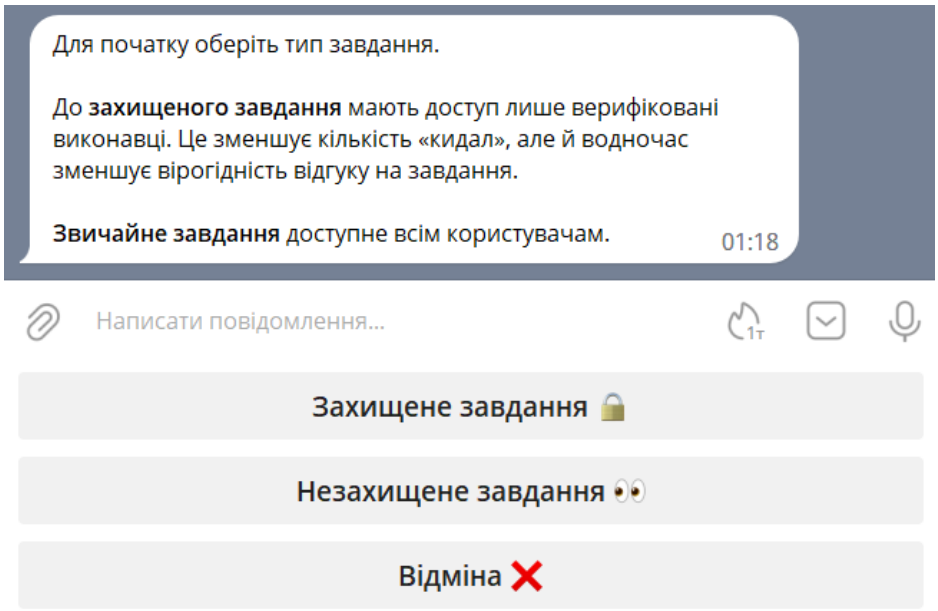

Рисунок 3.8 – Повідомлення та клавіатура для вибору типу завдання

Після вибору типу завдання з’явиться повідомлення про можливість додавання фото або файлу до завдання. Цей пункт можна пропустити, тому в меню є відповідна кнопка (див. Рисунок 3.9).

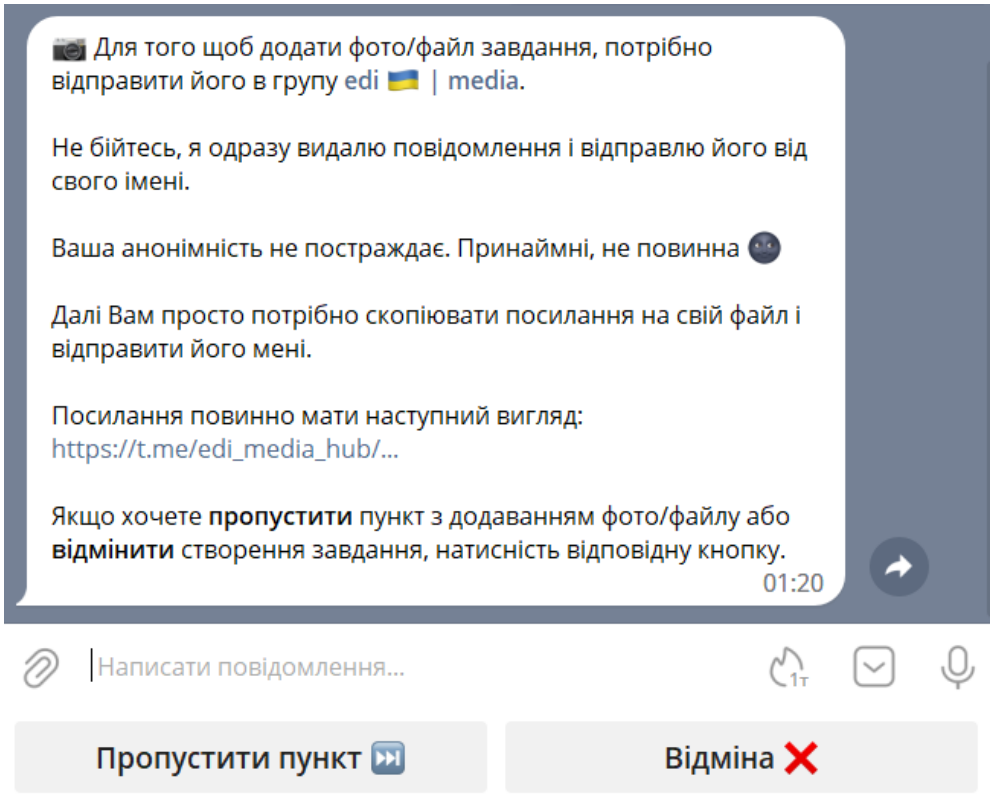

Рисунок 3.9 – Повідомлення та клавіатура для додавання фото або файлу

Важливо звернути увагу, що користувач повинен відправити саме посилання на фото або файл, розміщений в спеціальній групі з медіафайлами.

В даному випадку пропускаємо цей пункт і отримуємо повідомлення з попередженням (див. Рисунок 3.10).

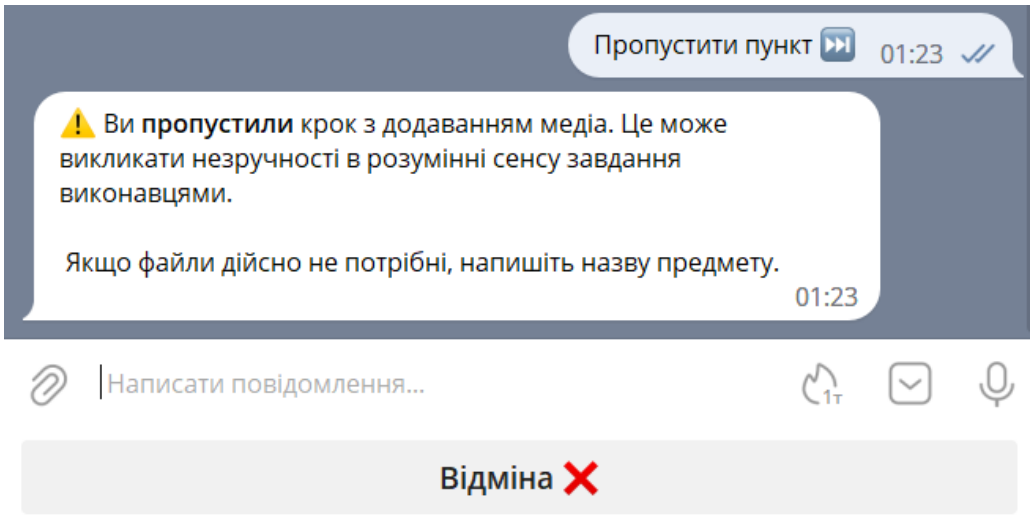

Рисунок 3.10 – Повідомлення та клавіатура після пропуску пункту

Відправляємо назву предмету і отримуємо повідомлення з проханням дати опис завдання (див. Рисунок 3.11).

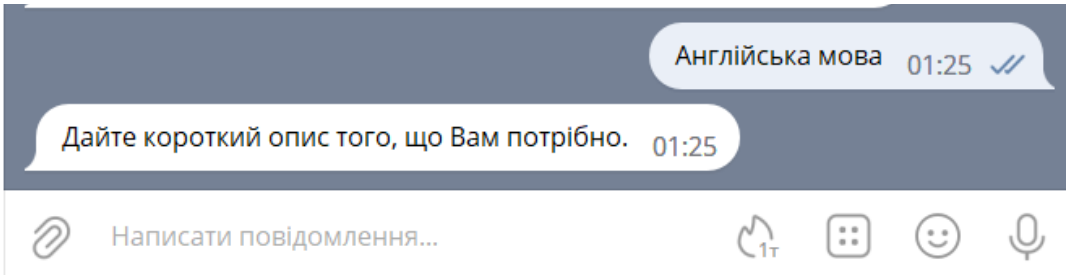

Рисунок 3.11 – Повідомлення для опису завдання

Вводимо опис і отримуємо повідомлення про те, що потрібно вказати ціну. Вказуємо нуль, що означає, що ціна договірна (див. Рисунок 3.12).

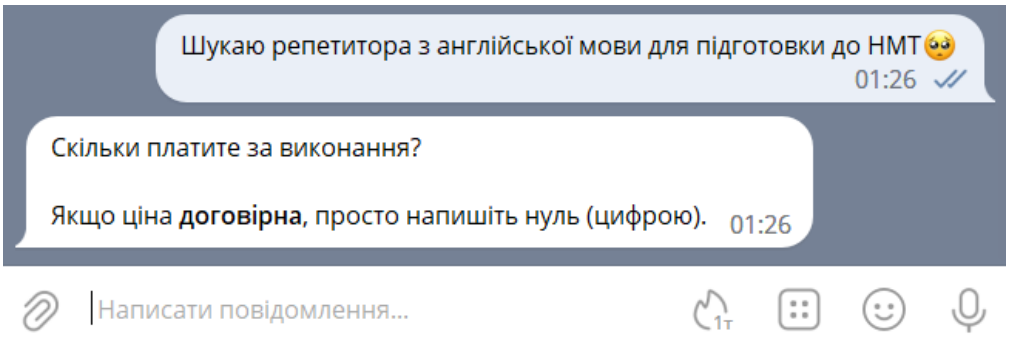

Рисунок 3.12 – Повідомлення для ціни

Створення завдання закінчено. Бот присилає повідомлення в тому вигляді, яке воно матиме при публікації на каналі (див. Рисунок 3.13). Якщо все влаштовує, підтвердимо створення завдання.

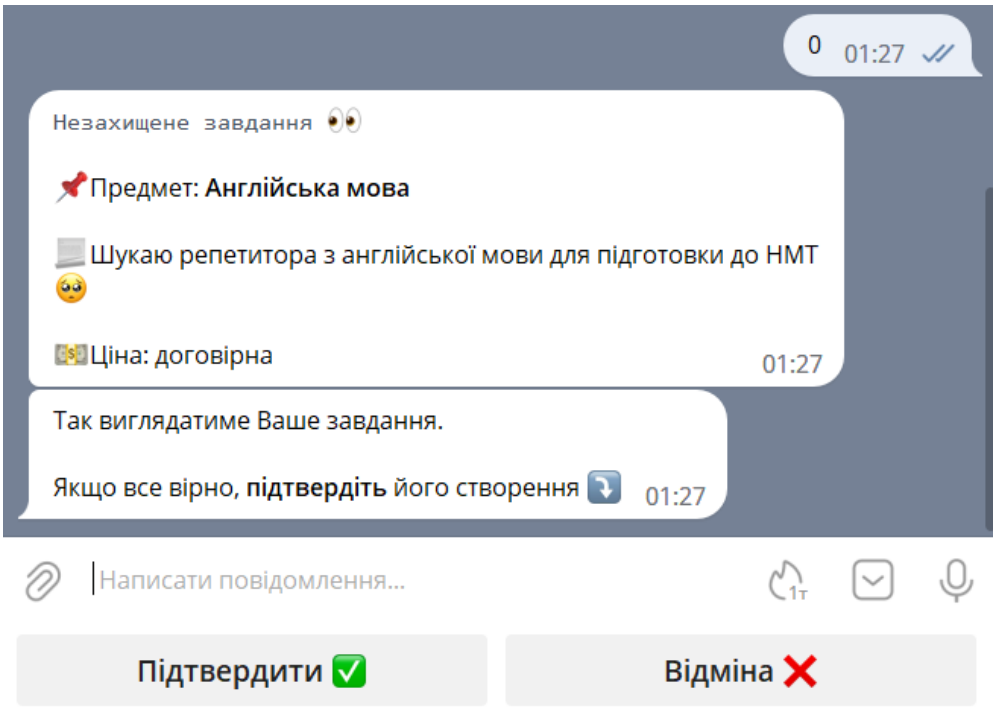

Рисунок 3.13 – Повідомлення для підтвердження завдання

Після підтвердження отримуємо відповідне повідомлення (див. Рисунок 3.14). Бот повертає клавіатуру головного меню.

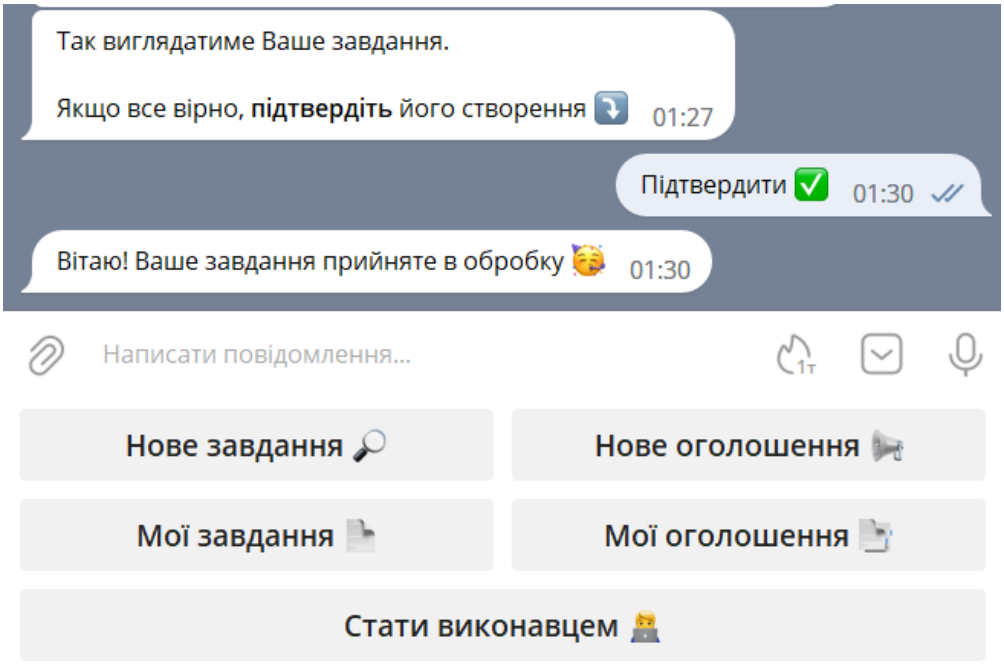

Рисунок 3.14 – Повідомлення після підтвердження завдання

Якщо натиснути кнопку «Мої завдання», бот виведе список завдань, створених користувачем (див. Рисунок 3.15).

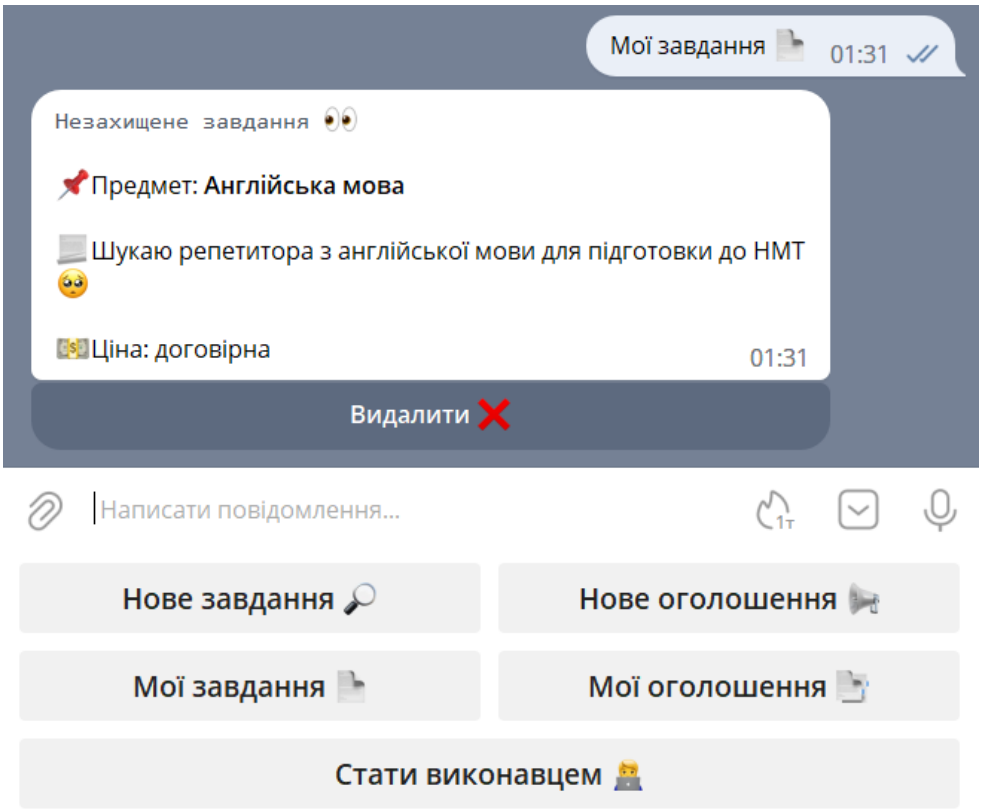

Рисунок 3.15 – Перегляд створених завдань

Якщо користувач хоче видалити завдання, він натискає відповідну кнопку, після чого отримує повідомлення (див. Рисунок 3.16).

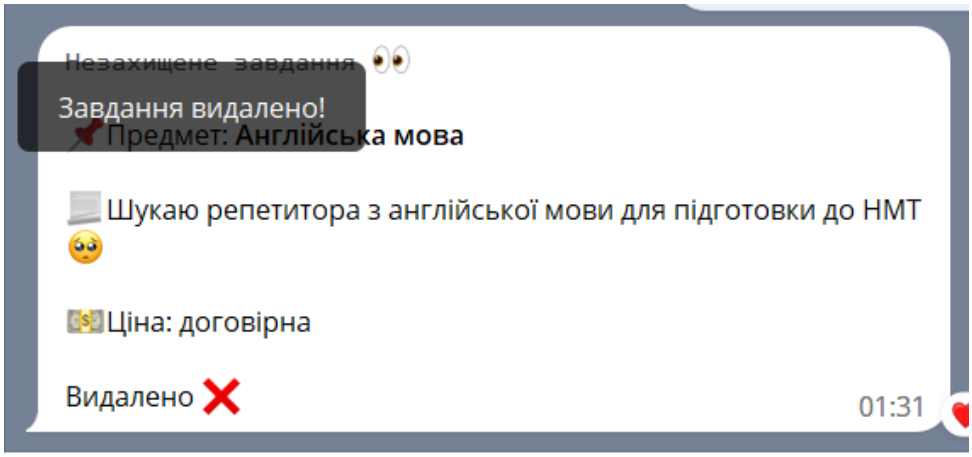

Рисунок 3.16 – Видалення завдання користувачем

При повторному натисканні кнопки «Мої завдання», бот виведе повідомлення, що користувач ще не додав жодного завдання (див. Рисунок 3.17).

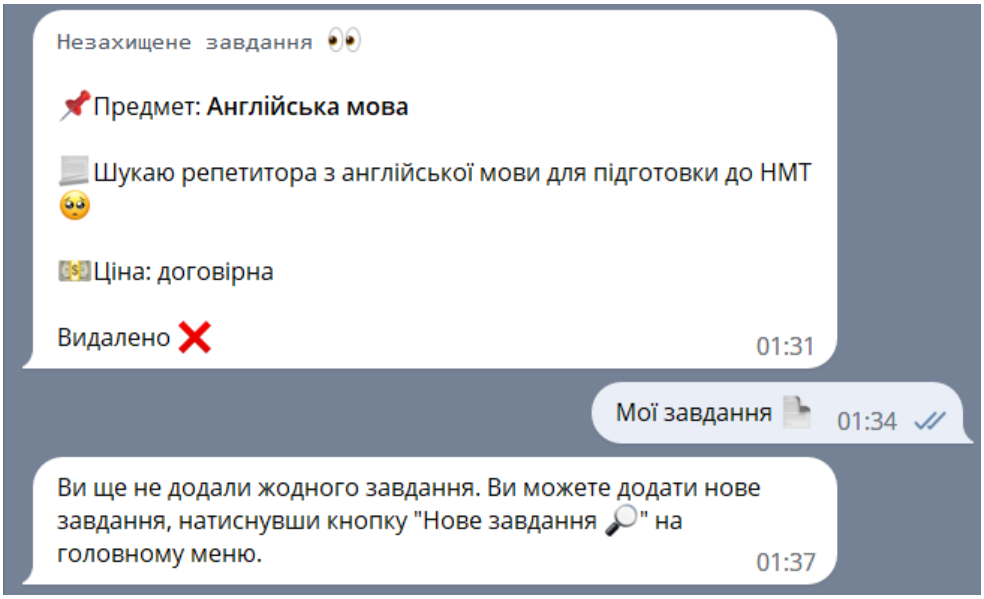

Рисунок 3.17 – Повідомлення про відсутність створених завдань

Створення, перегляд та видалення оголошень має майже ідентичний сценарій (див. Рисунок 3.18, Рисунок 3.19, Рисунок 3.20).

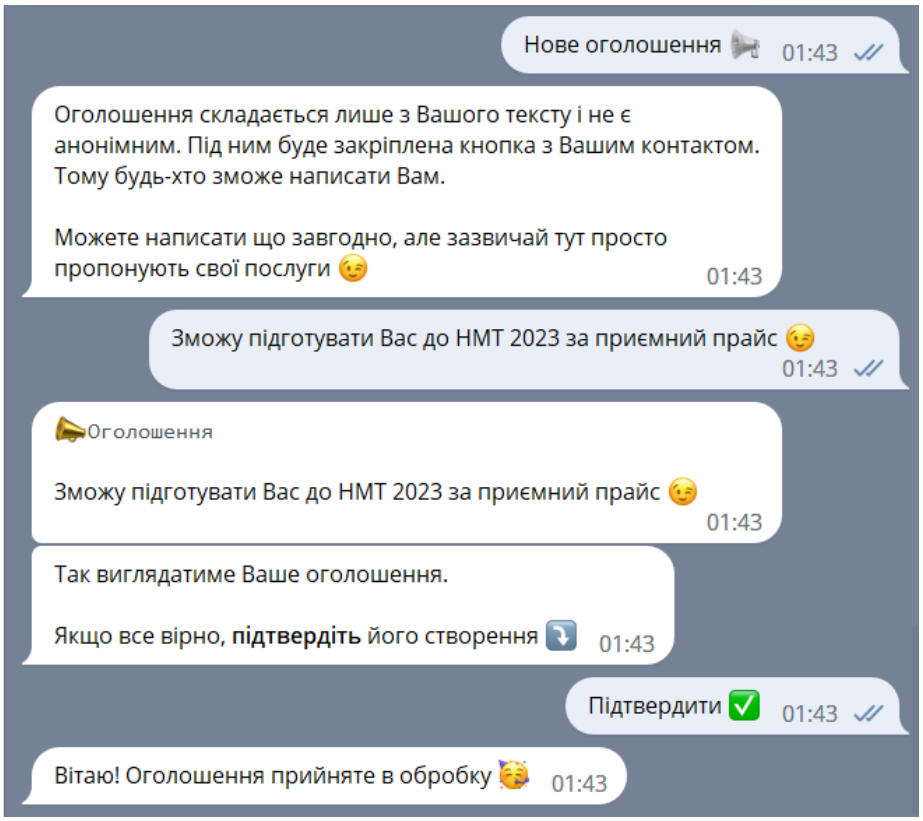

Рисунок 3.18 – Створення оголошення

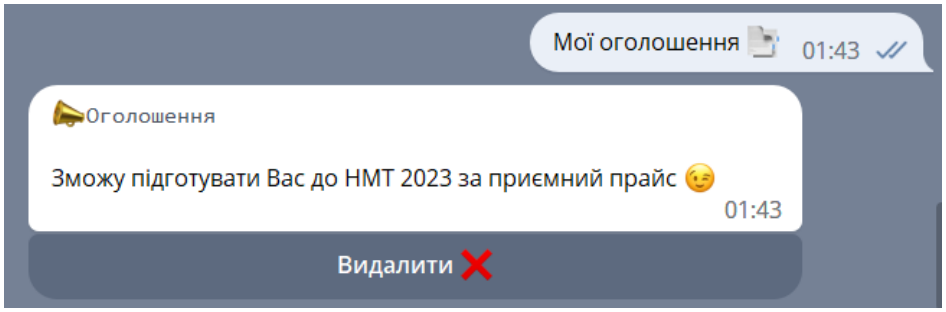

Рисунок 3.19 – Перегляд оголошень

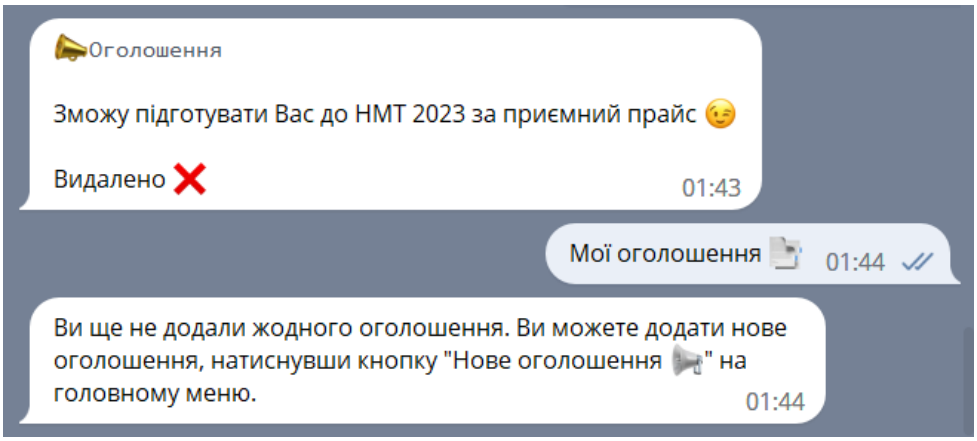

Рисунок 3.20 – Видалення оголошень

Після проходження модерації, завдання та оголошення публікуються на каналі. Під кожним постом розміщена відповідна кнопка, при натисканні якої користувач отримає повідомлення (див. Рисунок 3.21) та контакти автора (див. Рисунок 3.22).

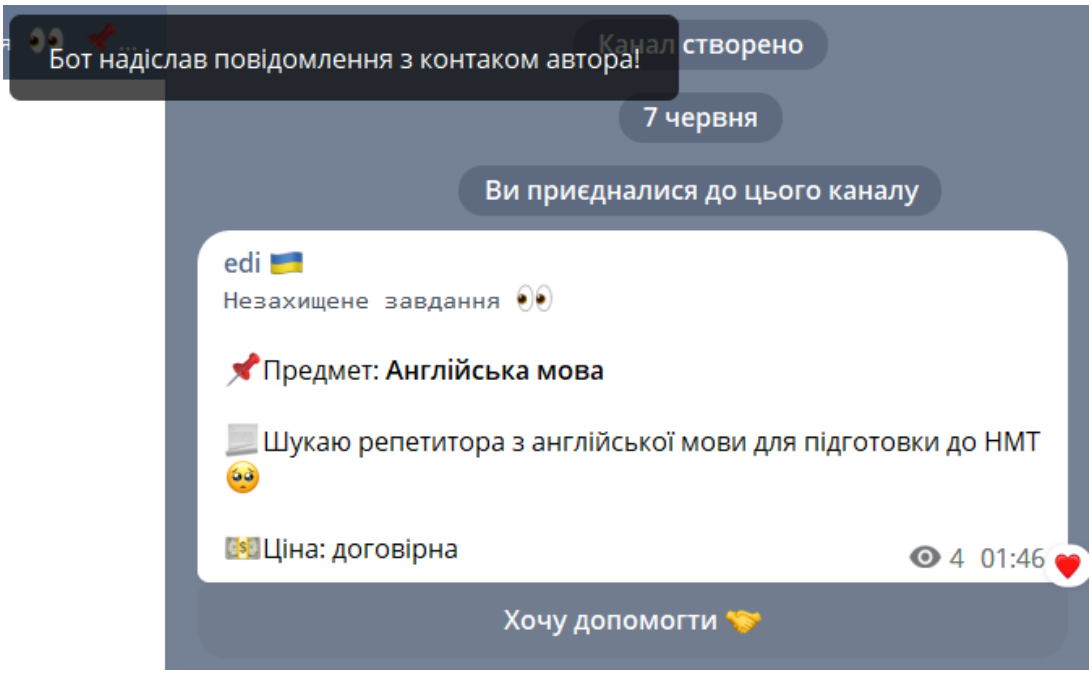

Рисунок 3.21 – Повідомлення про отримання контакту автора

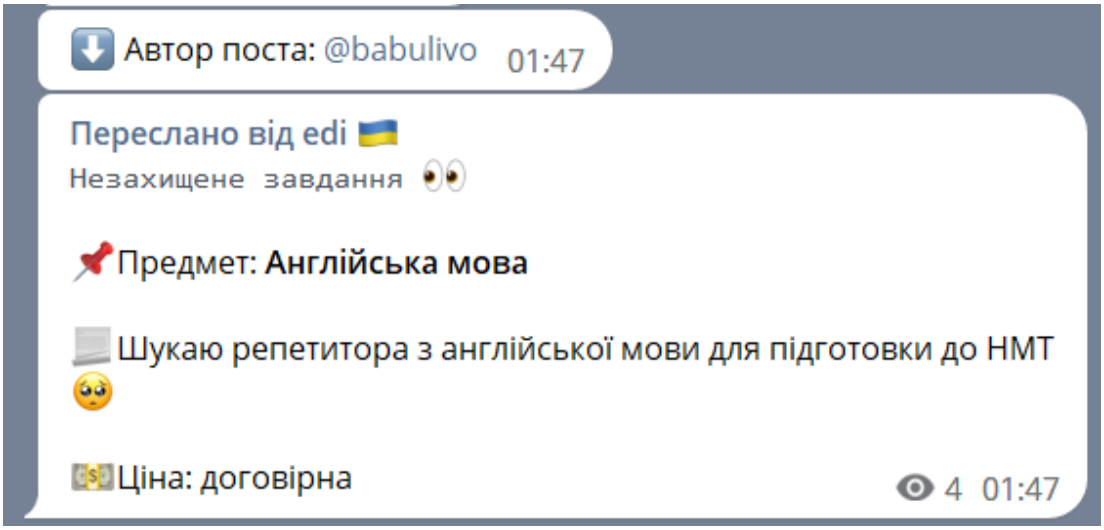

Рисунок 3.22 – Повідомлення з контактом автора

Доступ до виконання захищених завдань мають лише верифіковані користувачі. Якщо звичайний користувач намагатиметься отримати контакти

автора захищеного завдання, йому буде надіслане відповідне повідомлення (див. Рисунок 3.23).

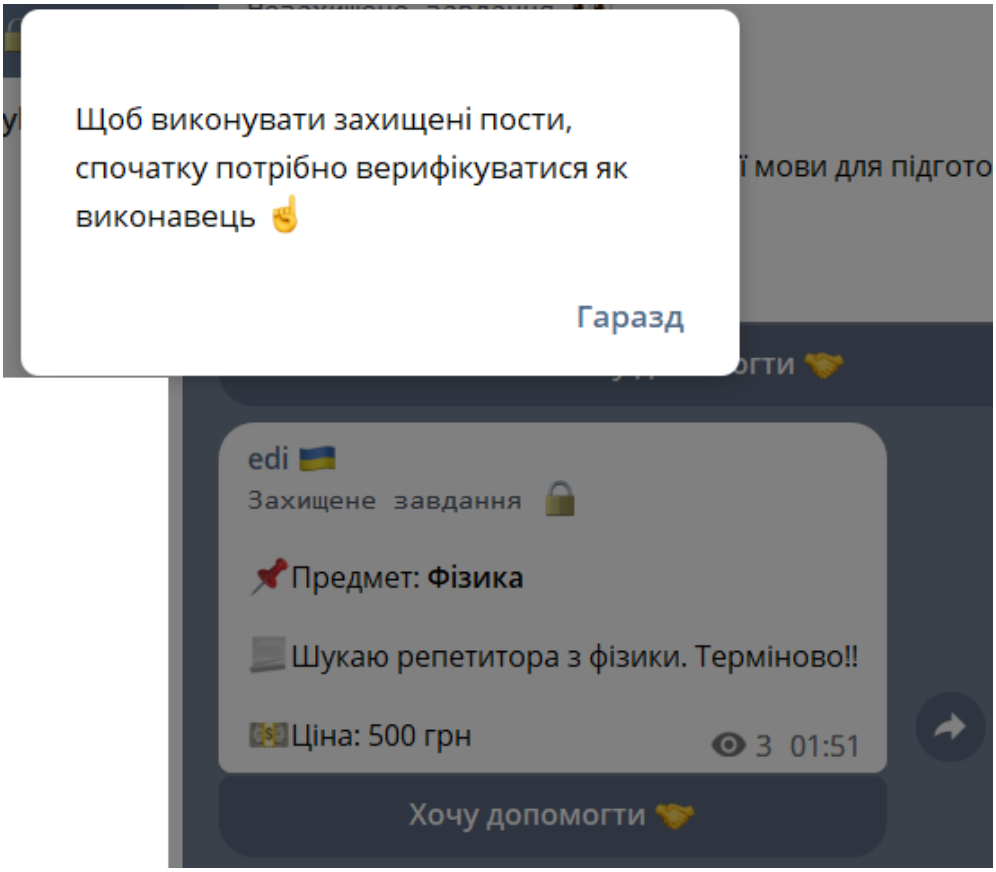

Рисунок 3.23 – Повідомлення для неверифікованих користувачів

Верифікуватися можна в головному меню бота, натиснувши на кнопку «Стати виконавцем». Бот запросить контакти користувача (див. Рисунок 3.24). і будь-який фото-доказ особистості (див. Рисунок 3.25).

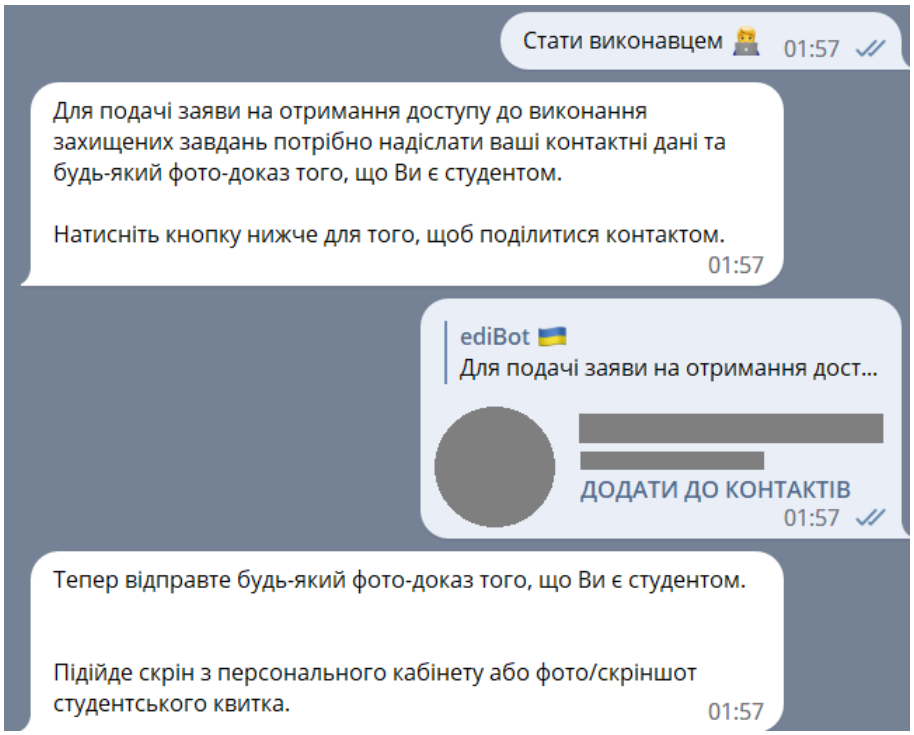

Рисунок 3.24 – Верифікація: відправка контакту

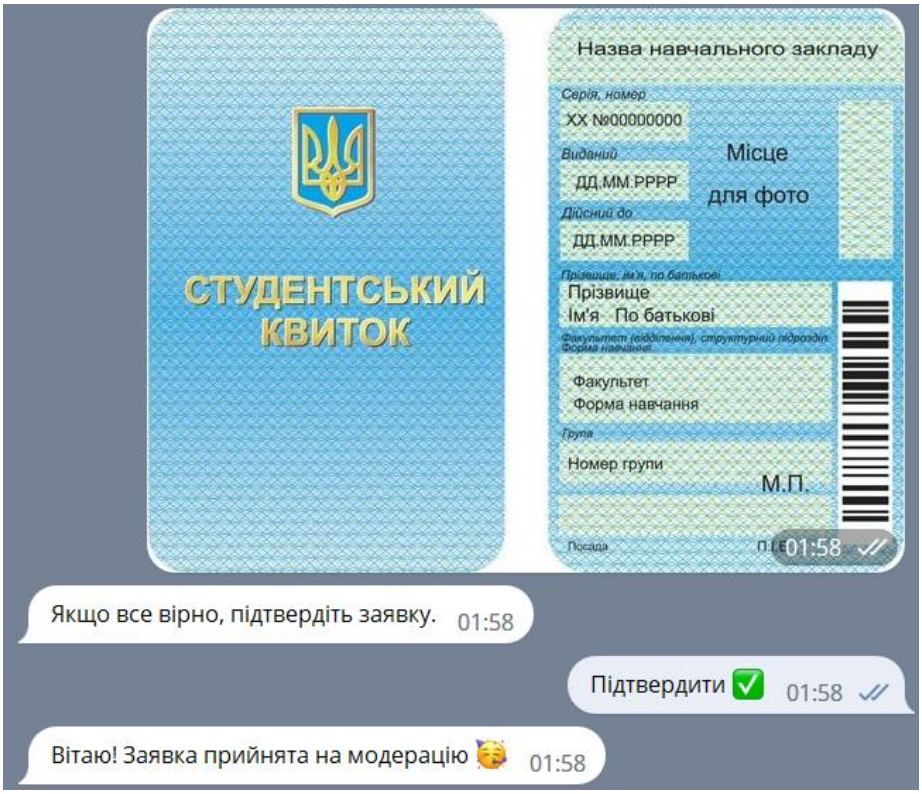

Рисунок 3.25 – Верифікація: відправка фото-доказу

Після успішної верифікації, при натисканні кнопки «Стати виконавцем», користувач отримає повідомлення про те, що він вже верифікований (див.

Рисунок 3.26). З цього моменту користувач має доступ до захищених завдань (див. Рисунок 3.27).

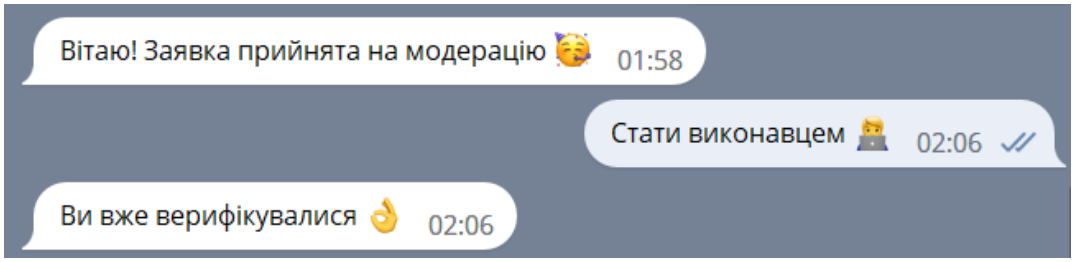

Рисунок 3.26 – Повідомлення про успішну верифікацію

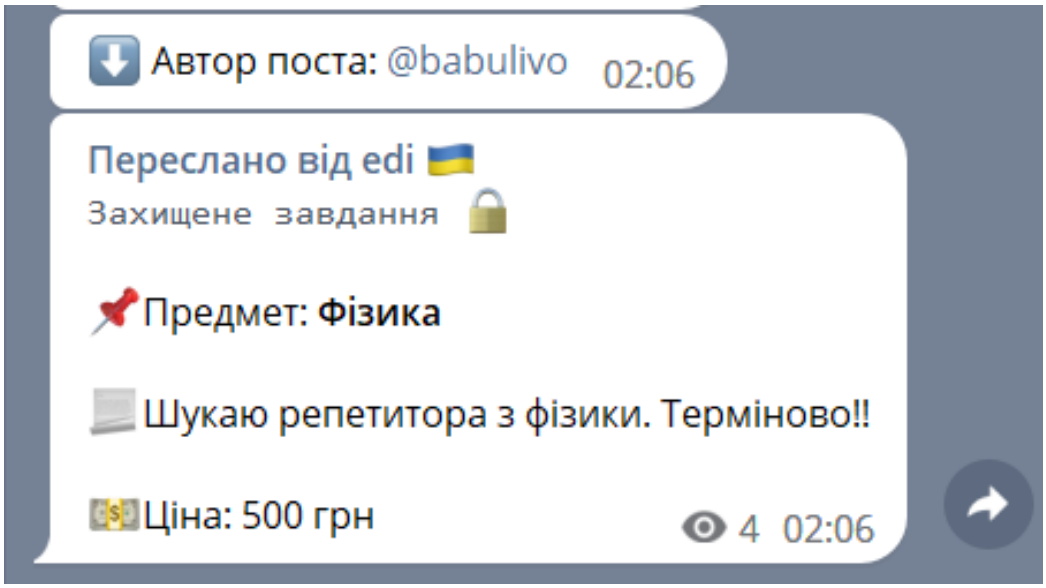

Рисунок 3.27 – Повідомлення з контактом автора захищеного завдання

Отримати контакти автора можна лише раз на годину. Якщо користувач вже має контакти автора, він отримає відповідне повідомлення (див. Рисунок 3.28).

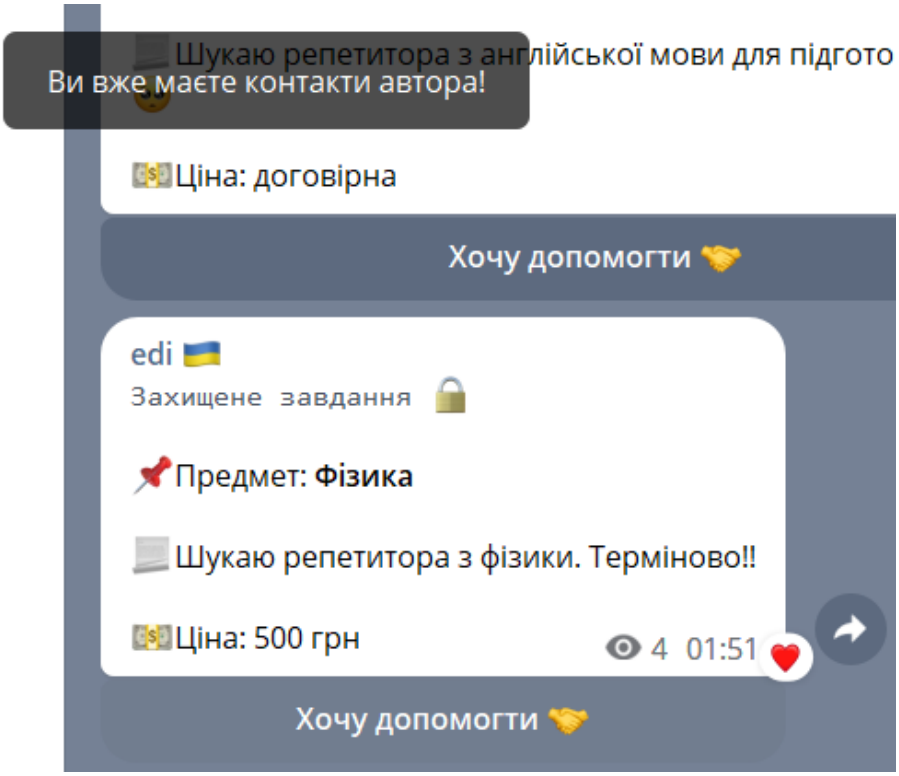

Рисунок 3.28 – Повідомлення про те, що користувач вже має контакти автора

3.7 Використання системи адміністратором

Основна задача адміністрації – верифікація виконавців, модерація завдань і оголошень, що відбувається через відповідні групи (див. Рисунок 3.29).

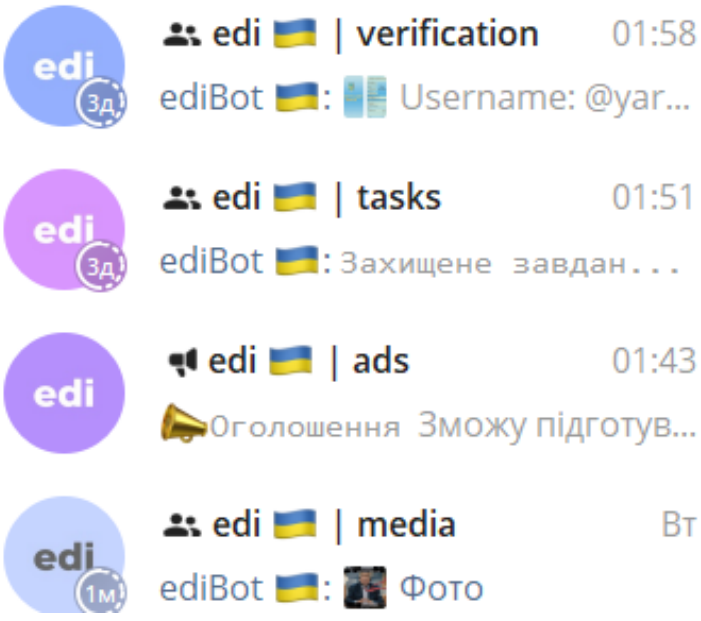

Рисунок 3.29 – Список груп для адміністрування

Після підтвердження користувачем, заявка на доступ до захищених завдань відправляється в групу верифікації (див. Рисунок 3.30). Переглянувши її, адміністратор верифікує користувача або ж видаляє заявку.

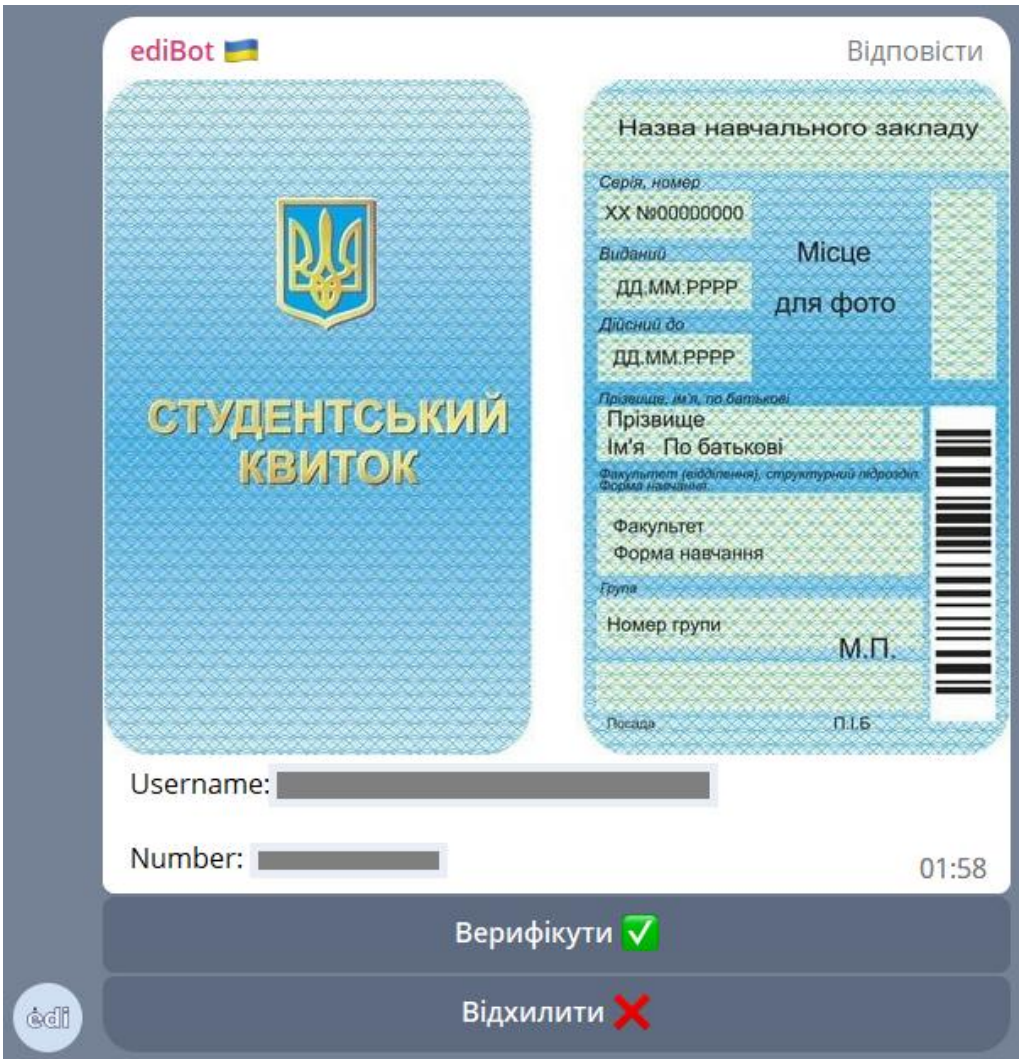

Рисунок 3.30 – Повідомлення з верифікацією виконавця

Після натискання кнопки, клавіатура зникає і до повідомлення додається інформацію про виконану дію (див. Рисунок 3.31).

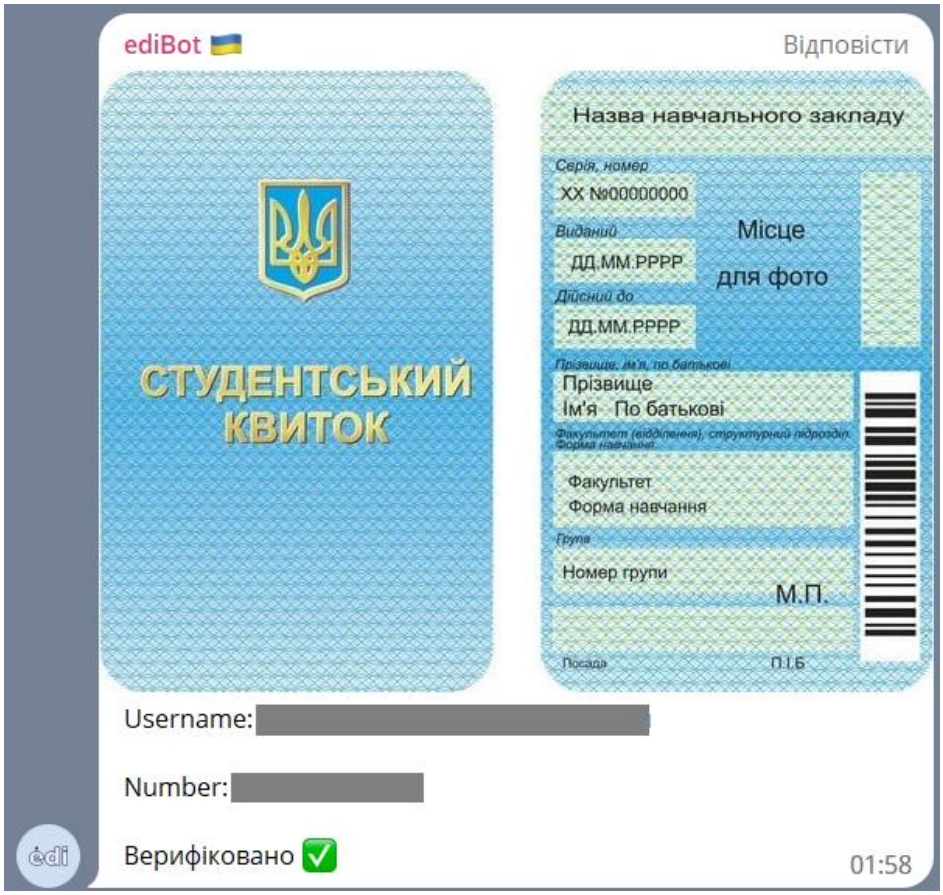

Рисунок 3.31 – Повідомлення про те, що користувача було верифіковано

При перевірці завдань та оголошень, адміністрація виконує аналогічні дії. Єдина різниця – ім’я користувача приходить окремим повідомленням (див. Рисунок 3.32).

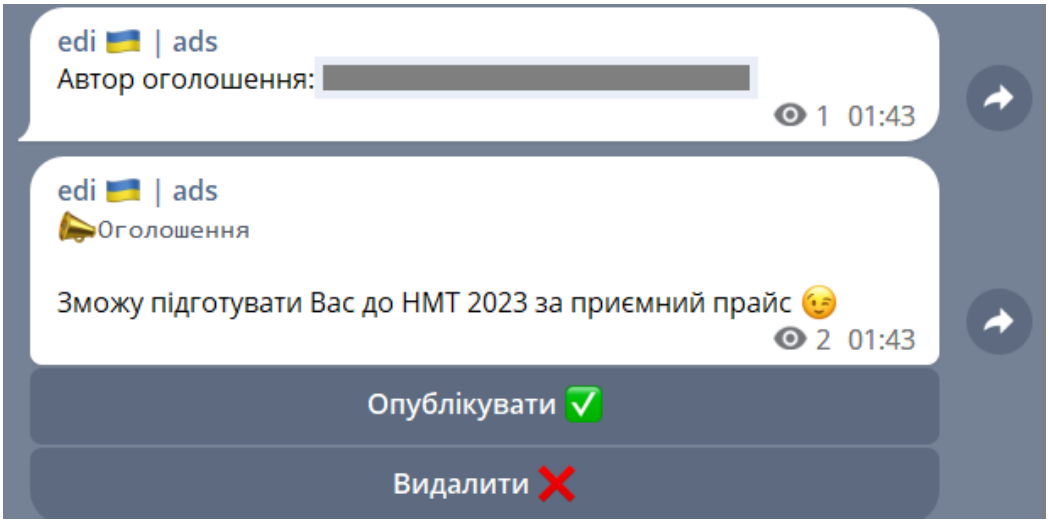

Рисунок 3.32 – Повідомлення для модерації оголошення

ВИСНОВКИ

У ході виконання кваліфікаційної роботи були виконані такі завдання:

1) Проведений аналіз найпопулярніших ресурсів-аналогів, на основі чого були розроблені вимоги до інформаційної системи.
2) Проаналізовано сучасні джерела, присвячені розробці Telegram-ботів, обрано найоптимальнішу бібліотеку та підхід до реалізації бота.
3) Розглянуто доступні і обрано оптимальні інструменти для розробки, що допомогли досягти максимальної зручності та продуктивності при написанні коду.
4) Отримано навички та знання необхідні для створення Telegram-ботів.
5) Розроблено інформаційну система управління освітнім фріланс-сервісом на базі месенджера Telegram. Реалізовано базовий функціонал, такий як створення, перегляд та модерація завдань і оголошень, реалізована функція захищених завдань та верифікації виконавців.

При збільшенні кількості активних користувачів, планується перехід на швидшу і зручнішу базу даних (наприклад PostgreSQL) та монетизація проекту. Наприклад, додавання захищених угод, в яких адміністрація буде виступати посередником між замовником і виконавцем. Функція не буде обов’язковою, щоб не порушувати ідею створення сервісу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Turing A. M. Computing Machinery and Intelligence. Mind, New Series. 1950. Vol. 59, no. 236. P. 433–460.

2. The History Of Chatbots – From ELIZA to ChatGPT. AI-chatbot software for complex requirements. URL: https://onlim.com/en/the-history-of-chatbots/#:~:text=The%20first%20chatbot%20ever%20was,20th%20century%20later%20on. (date of access: 05.01.2022).

3. Telegram Bot Platform. Telegram. URL: https://telegram.org/blog/bot-revolution (date of access: 18.05.2023).

4. Telegram Bot API. Telegram APIs. URL: https://core.telegram.org/bots/api (date of access: 18.05.2023).

5. Нова пошта запустила чат-бот для пошуку працюючих відділень і уточнення графіку їх роботи | «Нова пошта». URL: https://novaposhta.ua/news/rubric/2/id/10192 (дата звернення: 18.05.2023).

6. Чат-боти Telegram, Viber, Facebook Messenger для проведення фінансових операцій. privatbank.ua. URL: https://privatbank.ua/udalenniy-banking/chat-boty (дата звернення: 18.05.2023).

7. Telegram global MAU 2022 | Statista. Statista. URL: https://www.statista.com/statistics/234038/telegram-messenger-mau-users/ (date of access: 18.05.2023).

8. Terms of Service. Upwork. URL: https://www.upwork.com/legal (date of access: 18.05.2023).

9. Python Documentation. Python.org. URL: https://www.python.org/doc/ (date of access: 18.05.2023).

10. aiogram’s documentation. URL: https://docs.aiogram.dev/en/latest/ (date of access: 18.05.2023).

11. Pandemic-proof jobs? IT freelancers in high demand. The iLabour Project. URL: https://ilabour.oii.ox.ac.uk/pandemic-proof-jobs/ (date of access: 18.05.2023).

12. Економічна правда. Репетиторство в Україні: аналітичний огляд вартості додаткових занять та сучасних тенденцій. Економічна правда. URL: https://www.epravda.com.ua/publications/2021/09/1/677307/ (дата звернення: 18.05.2023).

13. Потреби цільової аудиторії: як зрозуміти свого споживача. Wizart. URL: https://wizart.agency/potreby_spozhyvachiv/ (дата звернення: 18.05.2023).

14. Етапи життєвого циклу розробки ПЗ. Веб-студія розробки програмного забезпечення - IC Studio, Україна. URL: https://icstudio.online/post/etapi-zhittyevogo-ciklu-rozrobki-pz (дата звернення: 18.05.2023).

15. Top 10 Telegram Chatbot Builders | Botpress Blog. Botpress | the Generative AI platform for ChatGPT Chatbots. URL: https://botpress.com/blog/top-10-telegram-chatbot-builders (date of access: 18.05.2023).

16. GitHub - php-telegram-bot/core: PHP Telegram Bot based on the official Telegram Bot API. GitHub. URL: https://github.com/php-telegram-bot/core (date of access: 18.05.2023).

17. GitHub - yagop/node-telegram-bot-api: Telegram Bot API for NodeJS. GitHub. URL: https://github.com/yagop/node-telegram-bot-api (date of access: 18.05.2023).

18. GitHub - pengrad/java-telegram-bot-api: Telegram Bot API for Java. GitHub. URL: https://github.com/pengrad/java-telegram-bot-api (date of access: 18.05.2023).

19. GitHub - atipugin/telegram-bot-ruby: Ruby wrapper for Telegram's Bot API. GitHub. URL: https://github.com/atipugin/telegram-bot-ruby (date of access: 18.05.2023).

20. 14 best Python Telegram Bot libraries in 2023 | kandi. kandi: find open source libraries, code snippets. URL: https://kandi.openweaver.com/collections/python/python-telegram-bot (date of access: 18.05.2023).

21. Asynchronous Programming in Python. Super Fast Python. URL: https://superfastpython.com/python-asynchronous-programming/ (date of access: 18.05.2023).

22. Morrison R. Модульне програмування: особливості, приклади, переваги, програми - Наука - 2023. warbletoncouncil. URL: https://uk.warbletoncouncil.org/programacion-modular-11773 (дата звернення: 18.05.2023).

23. Integrate the Telegram Bot API with the MySQL API - Pipedream. Pipedream. URL: https://pipedream.com/apps/telegram-bot-api/integrations/mysql (date of access: 18.05.2023).

24. Integrate the Telegram Bot API with the PostgreSQL API - Pipedream. Pipedream. URL: https://pipedream.com/apps/telegram-bot-api/integrations/postgresql (date of access: 18.05.2023).

25. Easy and safe Telegram Bot + Oracle DB integration | Onlizer. Onlizer. URL: https://onlizer.com/telegram_bot/oracle_db (date of access: 18.05.2023).

26. aiosqlite: Sqlite for AsyncIO – aiosqlite documentation. aiosqlite: Sqlite for AsyncIO – aiosqlite documentation. URL: https://aiosqlite.omnilib.dev/en/stable/ (date of access: 18.05.2023).

27. Microsoft. Documentation for Visual Studio Code. Visual Studio Code - Code Editing. Redefined. URL: https://code.visualstudio.com/docs (date of access: 18.05.2023).

28. Visual Studio Python IDE - Python Development Tools for Windows. Visual Studio. URL: https://visualstudio.microsoft.com/vs/features/python/ (date of access: 18.05.2023).

29. DBeaver Documentation. DBeaver. URL: https://dbeaver.com/docs/wiki/ (date of access: 18.05.2023).

30. User-friendly Design for Business Success | Koru UX Design Blog. Koru UX. URL: https://www.koruux.com/blog/user-friendly-design-is-the-driving-force-behind-innovation-and-business-success/#:~:text=A%20user-

friendly%20design%20enhances,and%20reducing%20the%20learning%20curve. (date of access: 18.05.2023).

31. Microsoft. Source Control with Git in Visual Studio Code. Visual Studio Code - Code Editing. Redefined. URL: https://code.visualstudio.com/docs/sourcecontrol/overview (date of access: 18.05.2023).

32. Архітектура ПЗ: що це таке, навіщо вона потрібна, стилі та моделі архітектури програмного забезпечення – Wezom. URL: https://wezom.com.ua/ua/blog/arhitektura-programmnogo-obespecheniya (дата звернення: 18.05.2023).

33. Modules. Python documentation. URL: https://docs.python.org/3/tutorial/modules.html (date of access: 18.05.2023).

ДОДАТОК А
ТЕХНІЧНЕ ЗАВДАННЯ

Створити Telegram-бота для освітнього фріланс-сервісу з використанням бібліотеки aiogram та бази даних SQLite. Сама система складається з бота, чатів адміністраторів та каналу, в якому публікуються пости користувачів. Пости та дані про виконавців зберігаються в базі даних. Пости поділяються на завдання та оголошення. Завдання можуть бути звичайними та захищеними. До захищених завдань мають доступ лише верифіковані виконавці. Завдання складається з назви предмету, опису, ціни та медіафайлу (опціонально). Оголошення – пост у вільній формі, складається лише з тексту користувача і не є анонімним.

Після того, як користувач вводить команду «/start», бот має перевіряти підписку користувача на канал і, після успішної перевірки, давати можливість створювати нові/переглядати власні пости, верифікуватися як виконавець.

Після створення завдання/оголошення/заявки, вона повинна надсилатися в відповідну групу адміністраторів для проходження верифікації.

Проект повинен бути розділений на модулі.

Нижче наведений приблизний сценарій взаємодії:

1) Перевірка підписки на канал:
    a. Перевірити, чи користувач підписаний на канал.
    b. Якщо ні, вивести відповідне повідомлення та кнопки «Підписатись» і «Вже підписався».
    c. При натисканні кнопки «Підписатись», переходити на канал за посиланням. При натисканні «Вже підписався», перевірити підписку і вивести стартове повідомлення.
    d. Якщо користувач досі не підписаний, вивести відповідне повідомлення.
2) Головне меню:
    a. Вивести стартове повідомлення з текстом та клавіатурою головного меню.

b. Головне меню повинно містити кнопки «Нове завдання», «Мої завдання», «Нове оголошення», «Мої оголошення» і «Стати виконавцем».

3) Створення нового завдання (створення нового оголошення матиме подібний сценарій):
   a. При натисканні кнопки «Нове завдання», розпочати процес створення поста.
   b. Запитати у користувача тип поста: захищений чи незахищений.
   c. Запитати назву предмету.
   d. Запитати посилання на медіафайл з завданням та вивести додаткову кнопку «Пропустити пункт», якщо замовнику не потрібно додавати медіафайл. Після натискання цієї кнопки, замовник отримує повідомлення, що він пропустив пункт.
   e. Запитати опис завдання.
   f. Запитати ціну. Перевіряти чи є надіслане повідомлення числом щоб повідомлення не було пустим, не було мінусовим або числом з плаваючою крапкою. При відправці «0», ставити ціну як «договірна».
   g. Вивести дані завдання користувачеві та дві кнопки, що дозволяють підтвердити відправку або відхилити завдання.
4) Відправка та модерація нового поста:
   a. При отриманні всіх необхідних даних для створення поста та підтвердження користувача, зберегти пост в відповідну таблицю.
   b. Відправити пост в відповідну групу адміністраторів на модерацію, прикріпити до повідомлення дві інлайн кнопки: «Опублікувати» та «Видалити».

c. Після натискання кнопки «Опублікувати», опубліковувати пост на каналі. Після натискання кнопки «Видалити», видаляти пост з бази даних.

5) Перегляд користувацьких постів:
   a. При натисканні кнопки «Мої пости», вивести список постів, які створив користувач.
   b. Додати кнопку «Видалити», яка дозволяє користувачеві видалити свій пост.
6) Процес верифікації виконавця:
   a. При натисканні кнопки «Стати виконавцем», розпочати процес верифікації виконавця.
   b. Запросити у користувача контакт, та будь-яке фото, що підтверджує особу виконавця.
   c. Зберегти дані в базу даних та відправити їх на модерацію в групу адмінів.
   d. Прикріити до заявки дві кнопки: «Верифікути» та «Відхилити».
   e. Зміни статус користувача в базі даних, залежно від рішення верифікації.

Після цього верифікований користувач матиме змогу відгукнутися на захищені завдання.