Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБКА БОТА В TELEGRAM ЗА ДОПОМОГОЮ FASTAPI ДЛЯ АНОНІМНОГО СПІЛКУВАННЯ У ФОРМАТІ ЧАТ РУЛЕТКИ
(тема)

Виконав:
здобувач 4 року навчання,
групи ІТІНФ-21-2
Лазаренко А. М.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Кобилін О. А.
(посада, прізвище, ініціали)

Допускається до захисту

Завідувач кафедри інформатики _____ Кобилін О. А.
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту

Кафедра Інформатики

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Лазаренка Артема Максимовича
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка бота в Telegram за допомогою FastAPI для анонімного спілкування у форматі чат рулетки

затверджена наказом університету від 19 травня 2025 року № 381Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 1 червня 2025 р.
3. Вихідні дані до роботи технічна література, відео матеріали з розробки телеграм ботів, матеріали конференцій, дані інтернет-мережі, бібліотека комп'ютерного зору.

4. Перелік питань, що потрібно опрацювати в роботі
1. Огляд методів розробки ботів та принцип роботи.
2. Аналіз роботи і структури існуючих чат-ботів.
3. Моделювання архітектури та схеми функцій боту.
4. Аналіз сучасних технологій для розробки боту.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Актуальність розробки чат-ботів, постановка задачі, варіанти використання чат-боту, принцип роботи боту, користь чат-боту для користувачів, зображення результату роботи.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	07.04.2025	
2	Аналіз завдання, підбір літератури	08.04.25-11.04.25	
3	Аналіз літератури з досліджуваної проблеми	12.04.25-14.04.25	
4	Аналіз розробки	15.04.25-19.04.25	
5	Розробка бота	20.04.25-28.04.25	
6	Програмна реалізація	29.04.25-10.05.25	
7	Оформлення пояснювальної записки	11.05.25-22.05.25	
8	Перевірка на нормоконтроль	21.05.25-01.06.25	
9	Перевірка на плагіат	21.05.25-01.06.25	
10	Рецензування	21.05.25-01.06.25	
11	Підготовка презентації та доповіді	21.05.25-18.06.25	
12	Занесення роботи в електронний архів	02.06.25-18.06.25	
	Попередній захист кваліфікаційної роботи	02.06.25-18.06.25	

Дата видачі завдання 7 квітня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ _____ доц. Кобилін О.А._____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 65 с., 2 табл., 12 рис., 31 джерела.

АНОНІМНИЙ ЧАТ, FASTAPI, JINJA2, UVICORN, SQLALCHEMY, HTML5, PYCHARM.

Об’єктом роботи є анонімний чат для професійного спілкування. Метою є створення застосунку, де користувачі можуть ставити питання, отримувати поради, думки експертів або підтримку в конфіденційному форматі, а також використовувати ці поради у повсякденному житті.

Клієнтська частина створена за допомогою HTML5, серверна – на базі FastAPI. Для рендерингу та роботи сервера використовуються Jinja2 і Uvicorn. SQLAlchemy для роботи з базою даних. Розробка здійснюється у середовищі PyCharm.

У результаті роботи буде зроблений вебзастосунок, що дозволяє користувачам залишати анонімні повідомлення, отримувати відповіді та зберігати історію спілкування з дотриманням конфіденційності та зручністю у використанні.

ANONYMOUS CHAT, FASTAPI, JINJA2, UVICORN, SQLALCHEMY, HTML5, PYCHARM.

The object of this work is an anonymous chat for professional communication.

The goal is to create an application where users can ask questions, receive advice, expert opinions, or support in a confidential format, and apply this advice in their everyday lives.

The client side was developed using HTML5, and the server side is based on FastAPI. Jinja2 and Uvicorn are used for rendering and server operation. SQLAlchemy is used for working with the database. Development is carried out in the PyCharm environment.

As a result of the work, a web application will be implemented, allowing users to leave anonymous messages, receive responses, and store the chat history, ensuring both confidentiality and ease of use.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

URL – Uniform Resource Locator (єдиний покажчик ресурсів)

API – Application Programming Interface (інтерфейс прикладного програмування)

FastAPI – швидкий інструмент для створення сайтів і сервісів

NPM – Node Package Manager (програма, яка допомагає встановлювати і оновлювати потрібні частини для сайтів)

PHP – Hypertext Preprocessor (препроцесор гіпертексту)

CRM – Customer Relationship Management (система управління взаємовідносинами з клієнтами)

OpenAI – Open Artificial Intelligence (відкритий штучний інтелект)

SQLite – Structured Query Language Lite (полегшена версія SQL-бази даних)

SQLAlchemy – Structured Query Language Alchemy (це «перекладач» між Python і базою даних)

ASGI – Asynchronous Server Gateway Interface (спосіб, за допомогою якого Python взаємодіє з інтернетом)

NPM-пакети – Node Package Manager (пакети програмного забезпечення, які використовуються в середовищі Node.js)

Фулстек – Fullstack (повний стек розробки).

ВСТУП

У сучасному цифровому світі онлайн-комунікація стала невід'ємною частиною життя мільйонів людей. З розвитком технологій зростає не лише зручність спілкування, а й усвідомлення користувачами важливості збереження своєї приватності. Все більше людей шукають можливість відкрито висловлювати свої думки, ставити запитання, отримувати підтримку чи консультацію, не розкриваючи при цьому свою особу. У багатьох випадках це пов'язано з бажанням уникнути осуду, зберегти анонімність при обговоренні особистих, психологічних чи професійно чутливих тем. Саме тому анонімні чати набувають дедалі більшої популярності, перетворюючись на ефективні інструменти комунікації в умовах інформаційної відкритості та соціального тиску.

Розробка анонімного чату є актуальною відповіддю на ці потреби. Такий інструмент дозволяє користувачам вільно спілкуватися без реєстрації особистих даних, що створює безпечне середовище для щирого діалогу. Це особливо важливо у випадках, коли традиційне публічне обговорення є складним або неможливим. Анонімне спілкування відкриває нові можливості для обміну досвідом, отримання порад чи емоційної підтримки від незнайомих людей або навіть експертів у різних галузях. Таким чином, розробка анонімного чату поєднує в собі актуальність соціального запиту на конфіденційність та відкритість з технічними можливостями сучасних вебтехнологій, що дозволяє створити доступну та ефективну платформу для комунікації в умовах цифрового простору.

Актуальність теми полягає в потребі створення таких цифрових рішень, які поєднують зручність, швидкість, конфіденційність та доступність. Анонімний чат, представлений у межах цієї роботи, демонструє можливості сучасних технологій у вирішенні соціально значущих задач, сприяючи формуванню безпечного простору для комунікації в інтернеті.

1 ОГЛЯД ТЕХНОЛОГІЙ ДЛЯ ТЕЛЕГРАМ БОТУ

1.1 Сучасне використання чат-ботів і анонімного спілкування в месенджерах

У XXI столітті стрімкий розвиток цифрових технологій призвів до кардинальної трансформації способів спілкування між людьми. Якщо ще кілька десятиліть тому основним засобом комунікації були особисті зустрічі та телефонні дзвінки, то сьогодні все більше людей віддають перевагу електронному обміну повідомленнями – через месенджери, соціальні мережі, форуми та інші онлайн-платформи.

Серед усіх сучасних засобів комунікації особливу роль відіграють чат-боти – програми, які автоматизують спілкування з користувачами, імітуючи реальні діалоги або виконуючи певні функції на основі команд. Вони використовуються в бізнесі, освіті, медицині, сфері обслуговування, розвагах та багатьох інших напрямках. За останні роки чат-боти значно еволюціонували: від простих скриптів, які відповідали на запити за шаблонами, до складних інтелектуальних систем, що базуються на машинному навчанні та природній обробці мови.

Однією з найпоширеніших платформ для розгортання чат-ботів є Telegram – месенджер, який дозволяє створювати ботів із відкритим API, підтримує асинхронний обмін повідомленнями, вбудовані кнопки, клавіатури, обробку команд і навіть інтерактивні сценарії. В Україні Telegram займає одне з провідних місць серед месенджерів, поступаючись хіба що Viber, WhatsApp і Facebook Messenger. За даними досліджень, аудиторія Telegram продовжує зростати, особливо серед молоді та IT-спільнот.

Однією з цікавих тенденцій останніх років стало зростання попиту на сервіси анонімного спілкування. Люди дедалі частіше шукають простір, де

можна висловити думку, поставити делікатне запитання, звернутись за порадою чи просто поспілкуватись без страху бути впізнаним або засудженим. Такі платформи популярні серед підлітків, людей із ментальними труднощами, пацієнтів, що шукають анонімну медичну або психологічну допомогу, а також серед професіоналів, які з різних причин не хочуть «світити» свої запити в публічному просторі.

Прикладами популярних платформ з елементами анонімності є:

Omegle – одна з перших чат-рулеток, яка дозволяла спілкування між випадковими користувачами без реєстрації. Попри простоту, Omegle часто критикували за відсутність модерації та зловживання з боку користувачів.

Chatroulette – аналог Omegle, який теж базувався на випадковому з’єднанні співрозмовників, але підтримував ще й відео-чат.

Yik Yak та Whisper – соціальні платформи, орієнтовані на обмін анонімними повідомленнями у стрічці.

Telegram-боти: наприклад, TalkBot, AnonTalkBot або ChatRouletteBot, які реалізують анонімний обмін повідомленнями між випадковими користувачами без зберігання особистих даних.

Інтерес до таких сервісів пов’язаний із психологічним аспектом анонімності. Люди почуваються вільніше, коли впевнені, що їх ніхто не ідентифікує. Це відкриває нові можливості для самовираження, запитів на підтримку, щирих обговорень і навіть професійних консультацій, які у відкритому просторі могли б бути складними або неможливими.

На цьому фоні Telegram особливо привабливий як платформа для реалізації анонімних чатів, адже:

– не потребує верифікації номера телефону для ботів, лише для користувачів;

– має розвинену інфраструктуру API з великою кількістю методів;

– підтримує одночасно простоту інтеграції й потужні інструменти для обробки логіки чатів;

– дозволяє використовуватиWebhook або Long Polling для реалізації динамічної взаємодії.

Крім Telegram, для реалізації анонімних чатів також використовуються вебплатформи, мобільні додатки, але Telegram вирізняється своєю простотою входу в середовище, високою продуктивністю, поширеністю і довірою серед користувачів.

Таким чином, актуальність розробки Telegram-бота для анонімного спілкування в форматі чат-рулетки зумовлена:

зростанням попиту на конфіденційне спілкування;

популярністю чат-рулеток як формату комунікації;

доступністю і гнучкістю інструментів Telegram Bot API;

зручністю використання таких сервісів у різних ситуаціях.

Усі ці фактори створюють міцне підґрунтя для подальшого аналізу технологічних рішень, підходів до реалізації та вибору оптимальної архітектури проєкту.

1.2 Розвиток сервісів типу чат-рулетки

Ідея випадкового з’єднання користувачів у чаті, які не знають нічого один про одного, має досить давнє коріння. Вона бере початок ще з перших текстових чатів у мережі Інтернет, проте справжній прорив стався у 2009 році з появою сервісу Chatroulette. Цей сайт запропонував новий формат спілкування – кожен користувач випадковим чином з’єднувався з іншим, і обидва мали змогу поспілкуватися у відеоформаті. Якщо розмова не задовольняла, достатньо було натиснути кнопку «Next», і система миттєво шукала іншого співрозмовника.

Простота, новизна та відчуття «рандомності» швидко зробили сервіс популярним. За лічені місяці Chatroulette зібрав мільйони користувачів по всьому світу та став предметом дослідження не лише для маркетологів, а й

для соціологів. Успіх цієї платформи став поштовхом для появи численних аналогів – Omegle, TinyChat, CooMeet, а згодом і мобільних застосунків, таких як Azar, HOLLA, Monkey, де формат чат-рулетки було адаптовано під нові технології та пристрої.

Попри очевидні переваги – легкість у використанні, новизну, швидкий доступ до спілкування – такі сервіси зіткнулися з цілим рядом проблем. Однією з головних стала велика кількість користувачів, які зловживали свободою та анонімністю. Чимало людей використовували ці сервіси для поширення неприйнятного або навіть незаконного контенту. Через це Chatroulette, наприклад, втратив частину аудиторії, а розробникам довелося терміново впроваджувати системи модерації та фільтрації.

Ще однією важливою проблемою стала відсутність ефективних інструментів боротьби з порушеннями. Більшість платформ не мали кнопок для скарг, систем оцінювання користувачів або механізмів блокування. Згодом ці недоліки було частково усунено: сучасні чат-рулетки зазвичай мають можливість поскаржитися на співрозмовника, використовують автоматичну модерацію за допомогою алгоритмів машинного навчання, запроваджують обмеження для порушників.

Також виникло питання з конфіденційністю. Хоча такі сервіси позиціонують себе як анонімні, деякі з них зберігали дані користувачів, записували IP-адреси, а іноді навіть дозволяли отримати особисту інформацію співрозмовника через уразливості в URL. Це поставило під загрозу безпеку користувачів, особливо молоді.

Попри всі труднощі, формат чат-рулетки продовжує розвиватися. З’явилися текстові версії, які не потребують використання камери – лише обмін повідомленнями між випадково обраними людьми. Вони стали альтернативою відеочатам, дозволяючи знизити ризик неприйнятного контенту та спростити процес модерації. Крім того, деякі сервіси запровадили можливість вказати теми, які цікавлять, аби покращити якість

розмов. Такий підхід підвищив релевантність спілкування та зменшив кількість випадкових непорозумінь.

Окремо варто звернути увагу на розвиток Telegram-ботів із функцією чат-рулетки. Telegram як платформа забезпечує хорошу анонімність – користувачі не бачать ні номери телефонів, ні справжніх імен. Це дозволило створювати прості текстові боти, де система з’єднує двох випадкових користувачів у приватній розмові. Такі боти реалізують базову логіку: приєднання до черги, очікування співрозмовника, початок діалогу, можливість завершити його та перейти до нового. Деякі з ботів також мають опції надсилання скарг, блокування небажаних контактів або навіть додавання «улюблених» користувачів до списку для повторного спілкування.

Популярність таких ботів підтверджує потребу в простому та конфіденційному способі спілкування. Вони активно використовуються молоддю, працівниками IT-сфери, волонтерами, навіть психологами – як спосіб отримати швидку розмову, пораду чи підтримку без розкриття особистості.

Таким чином, розвиток сервісів типу чат-рулетки демонструє як привабливість ідеї анонімного спілкування, так і необхідність чітко продуманої системи захисту, модерації та гнучкості. Саме ці принципи варто враховувати під час розробки власного Telegram-бота для анонімного спілкування, аби зробити сервіс не лише зручним, але й безпечним та етичним.

1.3 Психологічна і соціальна потреба в анонімному форматі комуцнікації

Анонімне спілкування в цифровому середовищі стало важливим соціальним явищем, яке має глибоке психологічне підґрунтя. У світі, де приватність дедалі більше обмежується, а соціальні мережі вимагають

відкритості, дедалі більше людей шукають простір для вільного, позбавленого тиску самовираження. Анонімність у цьому контексті виступає своєрідним цифровим захистом, що дозволяє відвертість, чесність і емоційну розрядку.

Важливим чинником, що зумовлює популярність анонімних платформ, є соціальний тиск. У багатьох випадках людина може соромитися поставити питання або поділитися проблемою, боячись осуду, непорозуміння або негативної реакції з боку інших. Це особливо стосується:

– студентів і школярів, які можуть мати питання, пов’язані з навчанням, особистими стосунками чи психологічним станом, але не хочуть звертатися до батьків або викладачів;

– фахівців, які хочуть обговорити професійні труднощі або поділитися невпевненістю без ризику втрати репутації;

– людей з ментальними або емоційними проблемами, які прагнуть підтримки, але побоюються осуду з боку суспільства;

– представників вразливих або маргіналізованих груп, яким складно знайти безпечне середовище для комунікації.

Анонімне спілкування дає змогу зняти бар’єри, які заважають комунікації у реальному житті. За відсутності персональної ідентифікації людина почувається вільнішою – вона може поділитися справжніми переживаннями, емоціями, думками без страху бути неправильно зрозумілою або засудженою.

Існують і психологічні причини, які стимулюють попит на анонімне спілкування:

– ефект розкриття: люди, залишаючись анонімними, частіше схильні ділитися особистим досвідом, навіть інтимного характеру. Це пояснюється зниженням рівня тривожності, яке виникає при усвідомленні того, що особистість залишиться невідомою;

– відчуття безпеки: анонімність створює ілюзію контролю над ситуацією. Користувач сам вирішує, що говорити, коли завершити розмову, чи повертатися знову;

– формування нових соціальних зв'язків: у чатах-рулетках користувачі можуть познайомитися з людьми, з якими б ніколи не перетнулися в реальному житті. Це розширює світогляд, дає змогу отримати підтримку від незнайомців і навіть знайти однодумців.

Особливу роль анонімність відіграє у психологічному консультуванні. Дослідження показують, що багато клієнтів вважають за краще отримати первинну консультацію в анонімному форматі. Це дозволяє «приміряти» формат терапії, відчути довіру до консультанта і лише потім (при бажанні) перейти до відкритого спілкування.

Існує кілька форматів, які реалізують анонімне спілкування:

– форуми та онлайн–платформи Reddit, Ask.fm, Quora у режимі анонімності – дозволяють ставити запитання без вказання імені, але спілкування відбувається публічно;

– анонімні мобільні застосунки Whisper, Sarahah, YikYak – спеціалізуються на обміні думками, враженнями, зізнаннями;

– чат–рулетки Omegle, Chatroulette – випадкове з'єднання двох незнайомців для спілкування в режимі 1 на 1;

– Telegram-боти – приватний формат, в якому немає відкритої стрічки чи збереженої історії, що гарантує додаткову конфіденційність.

Telegram як платформа має ряд переваг у контексті забезпечення анонімності:

– відсутність необхідності надавати персональні дані при взаємодії з ботом;

– можливість відображати лише ім'я користувача або взагалі уникати показу будь-якої інформації;

– простота відключення, блокування, зміни акаунта;

– роль анонімних чатів у подоланні соціальних бар'єрів.

Анонімні сервіси можуть виступати засобом боротьби з самотністю, ізоляцією, депресивними станами. Особливо це актуально під час кризових періодів – наприклад, пандемії COVID-19 або під час воєнних дій, коли багато людей опинились у стані невизначеності, тривоги і втрати соціальних контактів.

У таких умовах анонімне спілкування виконує не лише інформаційну, а й терапевтичну функцію – люди отримують полегшення просто від того, що їх хтось вислухав або підтримав.

1.4 Підхід до реалізації анонімного спілкування в Telegram-ботах та альтернативні технічні рішення

Розробка анонімного чату – це завдання, що поєднує в собі як технічні, так і соціальні аспекти. З одного боку, важливо забезпечити стабільність, швидкість і безпеку взаємодії користувачів, з іншого – гарантувати повну конфіденційність та простоту використання. Telegram, як одна з найпопулярніших платформ для розгортання ботів, надає широкі можливості для реалізації подібних сервісів, однак існують різні підходи до побудови архітектури таких ботів.

Telegram Bot API дозволяє створювати ботів, які можуть:

– отримувати повідомлення від користувачів, надсилати повідомлення, зображення, документи, відео, голосові повідомлення та відповідати на команди;

– обробляти Inline запити, Callback кнопки, Webhook запити.

Telegram дозволяє будувати peer to peer комунікацію, навіть якщо обидві сторони не знають ідентичності одна одної. Це відкриває шлях для реалізації анонімних чатів, в яких бот виступає «посередником», що з’єднує випадкових користувачів.

Основна логіка роботи чат-бота виглядає так: користувач натискає кнопку «Почати чат». Бот додає його в «чергу очікування». Коли з’являється другий користувач у черзі – бот з’єднує їх. Усі повідомлення, які надсилає кожен з них, бот ретранслює іншому користувачу.

Коли хтось натискає «Завершити чат», бот припиняє пересилання і повертає обох в початкове меню. Цей підхід можна реалізувати за допомогою різних стеків.

Варіанти реалізації: порівняння технологічних стеків:

– FastAPI – сучасний високопродуктивний фреймворк для розробки API на Python, який дозволяє легко працювати з Webhook-архітектурою, масштабувати ботів і інтегрувати додаткові сервіси;

– aiogram – асинхронна Telegram-бібліотека, яка чудово інтегрується з FastAPI. Вона дозволяє легко обробляти повідомлення, клавіатури, команди та управляти FSM для побудови складних логік.

Переваги:

– швидка розробка;

– асинхронність – це висока продуктивність;

– простота обслуговування коду;

– велика спільнота.

Недоліки:

– потрібне знання асинхронного програмування;

– обмеження Telegram API.

Другий варіант;

Telegraf.js – це популярна бібліотека для Telegram-ботів на JavaScript/Node.js. Підходить тим, хто працює у фронтенд або фулстек-розробці.

Переваги:

– висока продуктивність;

– велика кількість npm-пакетів для розширення функціоналу;

– добра підтримка WebSocket, що важливо для real-time чатів.

Недоліки:

– Node.js менш стабільний у довготривалих завданнях порівняно з Python;

– необхідність налаштування додаткових сервісів, наприклад: Redis для черг користувачів.

Третій варіант менш популярний для ботів, але також можливий. Це PHP з Laravel або Symfony.

Переваги:

– простота входу для веброзробників зі світу PHP;

– добре підходить для монолітних вебсистем, де бот є частиною CRM.

Недоліки:

– обмежена підтримка асинхронності;

– менша гнучкість при роботі з Telegram API.

Є декілька архітектурних підходів, це Long Polling та Webhook.

Long Polling – цей бот постійно запитує Telegram-сервер, чи є нові повідомлення. Простий у налаштуванні – не потребує SSL-сертифіката. Мінус: більше навантаження на сервер, менш ефективний у великих проєктах.

Webhook – Telegram надсилає нові повідомлення безпосередньо на вказаний URL. Потрібен SSL-сертифікат. Більш продуктивний варіант для масштабованих сервісів.

Зберігання даних. Для реалізації функцій анонімного чату необхідно зберігати інформацію про:

– стан користувача;

– ID співрозмовника;

– статистику.

Для цього на вибір можна обрати такий стек технологій:

– SQLite підходить для простих, локальних проєктів;

– PostgreSQL – стабільний вибір для великих проєктів;

– Redis – найкраще підходить для черг, тимчасових сесій і кешування.

Також, можна розглянути додаткові можливості та захист конфіденційності щоб покращити досвід користувача, а також підвищити рівень анонімності, можна впровадити:

– тимчасові ID користувачів замість Telegram ID;

– автоматичне видалення повідомлень після завершення чату;

– модерацію за допомогою ключових слів;

– кнопки для скарг на користувача;

– блокування при порушенні правил;

– інтеграцію з OpenAI чи GPT-ботом для ситуацій, коли немає співрозмовника – бот може підтримати розмову.

Попри зручність Telegram, є й інші платформи:

– Discord-боти можна реалізовувати приватні кімнати, але вимагає облікового запису;

– вебчати на WebSocket повна свобода дизайну, але більше роботи зі сторони фронтенду та бекенду;

– мобільні застосунки підходять для створення нативного UX, але потребують більше часу та ресурсів.

Таким чином, розробка Telegram-бота для анонімного чату – це не лише соціально затребуваний проект, а й цікаве технічне завдання, що дає простір для різних архітектурних і дизайнерських рішень. Кожен варіант реалізації має свої переваги та недоліки, і вибір конкретного стеку залежить від цілей, ресурсів і бажаної масштабованості сервісу.

1.5 Варіативність тем і форматів анонімного спілкування

У сфері анонімного спілкування важливу роль відіграє не лише технічна реалізація, але й тематика розмов. І хоча на перший погляд чат-рулетка – це випадкове спілкування без заданого сценарію, з часом

з’ясувалося, що наявність певних тематичних напрямів значно підвищує цінність такого сервісу для користувачів.

Звичайна чат-рулетка, де учасники просто шукають випадкових співрозмовників, має свою аудиторію. Проте вона часто виявляється обмеженою в плані змістовності діалогів. Користувачі не знають, чого очікувати, іноді не можуть знайти «спільну мову», розмова не виходить за рамки банального «Привіт – Як справи». З іншого боку, коли сервіс пропонує вибір теми перед початком спілкування, він автоматично підвищує ймовірність якіснішого контакту.

Так, існують чат-сервіси, де користувач може обрати категорію, наприклад:

- психологічна підтримка;
- питання з ІТ та програмування;
- поради щодо навчання;
- спілкування з людьми з інших країн;
- обговорення фільмів, книг, ігор.

Подібний підхід дозволяє привабити різні аудиторії – студентів, спеціалістів, інтровертів, людей, які шукають неформальне спілкування, або тих, хто не готовий звернутись по допомогу відкрито. Тематична варіативність дає користувачу контроль над типом спілкування, що, у свою чергу, підвищує довіру до сервісу.

Ще одним важливим напрямом є формат діалогу. Хоча більшість чатів мають формат один-на-один, існують варіанти групового чату, або ж «консультаційного» режиму, де одна сторона виступає в ролі експерта, а інша – запитувача. Це створює можливість для реалізації чат-рулетки у форматі запитань і відповідей, де користувач може звернутись із конкретною проблемою, а на іншому боці виявиться людина, яка має досвід чи бажання допомогти.

Цікавим є також варіант, коли спілкування відбувається з ботом або ШІ у випадку, якщо вільного співрозмовника немає. Такий підхід допомагає утримати користувача в системі навіть у години низької активності. Боти можуть підтримувати розмову, ставити питання, імітувати співчуття чи навіть давати поради на основі сценаріїв, підготовлених фахівцями. У поєднанні з реальними діалогами це створює відчуття постійної присутності.

Крім того, деякі сервіси дають змогу після завершення спілкування залишити оцінку, побажання або короткий відгук. Ці дані можна використовувати для подальшого аналізу і вдосконалення роботи бота. Залежно від цілей, платформи також можуть збирати анонімну статистику щодо найпопулярніших тем, часу активності користувачів, середньої тривалості сесії тощо.

Важливо зазначити, що варіативність не повинна порушувати принцип анонімності та простоти використання. Якщо система перевантажена категоріями, складними формами чи зайвими деталями, вона може втратити основну перевагу – миттєвий старт розмови. Саме тому, навіть додаючи варіанти тем чи типів спілкування, важливо залишити інтерфейс максимально інтуїтивним та ненав’язливим.

Отже, завдяки варіативності тематики і форматів можна значно підвищити якість спілкування в чат-рулетці, залучити ширшу аудиторію та створити платформу, яка буде корисною не лише для випадкових розмов, але й для пошуку підтримки, порад, нових ідей та професійного розвитку.

1.6 Мотивація користувачів та утримання аудиторії у сервісах анонімного спілкування

Оскільки сервіси чат-рулетки базуються на ідеї випадкових розмов, їхня ефективність безпосередньо залежить від кількості активних користувачів. Якщо в системі перебуває мало людей – очікування з’єднання

збільшується, користувачі швидко втрачають інтерес і покидають платформу. Тому завдання будь-якого подібного сервісу – не тільки залучити, а й удержати аудиторію, створивши достатньо мотивацій для регулярного використання.

У класичних чат-рулетках, особливо тих, що працюють на відеоформаті, утримання аудиторії будувалося переважно на ефекті «новизни» – людям було цікаво побачити, з ким їх з'єднає наступного разу. Проте цей ефект швидко вичерпується, особливо якщо якість розмов залишає бажати кращого. Тому сучасні сервіси активно впроваджують механіки, подібні до програм лояльності, які, хоч і не завжди в класичному вигляді, але служать тим самим цілям – стимулювати повернення.

Одним із таких способів є система «улюблених співрозмовників». Наприклад, після вдалої розмови користувач може додати іншого до умовного списку друзів або зберегти його ID, щоб за бажанням продовжити спілкування пізніше. У Telegram-ботах подібне реалізується шляхом створення спеціальних команд або меню, де зберігаються запам'ятані співрозмовники. Це дає змогу поєднати елемент анонімності з можливістю будувати довготривалі зв'язки.

Ще одним інструментом є мотивація у вигляді балів, досягнень або внутрішньої статистики. Наприклад, користувач може бачити:

- кількість проведених розмов;
- середню тривалість сесій;
- кількість позитивних оцінок від співрозмовників;
- рейтинги довіри.

Такі індикатори дають користувачу відчуття прогресу, а також стимулюють повертатися до чату знову, щоб підвищити показники або досягти нового «рівня».

Іншим популярним методом утримання є тематичні події або «розмовні марафони». Наприклад, сервіс може анонсувати «вечір психологічної підтримки» або «розмови про фільми суботнього вечора», де користувачі

випадково підбиратимуться, але всередині певної тематики. Це дозволяє залучати не лише випадкових відвідувачів, але й формувати спільноту навколо сервісу, створюючи емоційний зв’язок.

У випадку з анонімними сервісами, важливим елементом мотивації є безпека та конфіденційність. Якщо користувач знає, що його особисті дані не буде збережено, що він може в будь-який момент завершити діалог, що в системі є адекватна модерація – ймовірність повернення значно зростає. Такі невидимі фактори часто грають більшу роль.

Окремо варто згадати рекомендації та сарафанне радіо. Успішні анонімні сервіси часто розповсюджуються не через рекламу, а через особисті рекомендації – «там можна просто поговорити і ніхто тебе не осудить». Це можливо лише тоді, коли у користувача був позитивний досвід, а для цього важливо забезпечити мінімум фрустрації – зручний інтерфейс, швидке з’єднання, відсутність спаму чи токсичності.

Також іноді використовується так званий контентний фільтр або «рівень доступу». Наприклад, щоб мати можливість змінювати тему розмови, потрібна певна кількість активностей. Або ж користувач може «відкрити» нові функції після кількох завершених розмов. Такі методи створюють мотивацію залишатися у сервісі довше.

У випадку Telegram-бота, реалізація програми лояльності можлива за допомогою:

– повідомлень із нагадуваннями про нові функції або події;

– збереження інформації про користувача – наприклад, його тематика інтересів;

– додавання простих повідомлень подяки за використання сервісу, що теж має позитивний емоційний вплив.

Отже, мотивація користувачів у контексті чат-рулетки – це не лише заохочення залишатися в чаті довше, а й створення комфортного простору, де людині цікаво, безпечно та просто повертатися. Це особливо важливо для сервісів, орієнтованих на анонімне, чутливе або професійне спілкування.

1.7 Постановка задачі

Таким чином, розробка інформаційної системи для анонімного спілкування у форматі чат-рулетки є актуальним завданням. В епоху стрімкого розвитку засобів онлайн-комунікації, важливо забезпечити користувачам безпечну, зручну та сучасну платформу для знайомств та спілкування. Тому ставиться завдання створення телеграм-бота, що поєднує інтуїтивний інтерфейс, сучасні технології та ефективну систему підбору співрозмовників.

Об’єктом розробки є телеграм-бот, який дозволяє користувачам швидко знайти анонімного співрозмовника на основі заданих параметрів, а також вести реальний діалог у режимі реального часу. Користувач може обрати свої вподобання – це полегшує пошук відповідного партнера. Бот забезпечує автоматичний підбір пари, обробку повідомлень та завершення розмови у будь-який момент.

Сторона адміністратора представлена у вигляді бази даних, в якій зберігається інформація про користувачів, сесії та активні з’єднання, що дозволяє ефективно керувати логікою роботи бота.

Метою роботи є створення інформаційного застосунку – чат-бота в Telegram, що забезпечує користувачам:

– анонімне спілкування з випадковими співрозмовниками;

– налаштування параметрів пошуку;

– можливість швидко завершити розмову або перейти до нового партнера;

– сучасний адаптивний вебінтерфейс через Telegram WebApp.

Для досягнення мети необхідно вирішити такі завдання:

– визначити оптимальні критерії для підбору співрозмовників;

– обрати архітектуру системи з використанням WebSocket для реального часу;

– реалізувати Telegram WebApp для збору налаштувань користувача;

– реалізувати систему авторизації через Telegram;

– реалізувати алгоритм створення сесій з обмеженим терміном дії;

– реалізувати алгоритм пошуку відповідного співрозмовника;

– реалізувати обробку повідомлень між користувачами у форматі реального часу;

– реалізувати алгоритм завершення сесії та оповіщення іншого учасника;

– забезпечити можливість відновлення пошуку нового співрозмовника після завершення розмови;

– протестувати систему з різними сценаріями спілкування;

– забезпечити збереження конфіденційності користувачів.

2 МОДЕЛЮВАННЯ ПРОЦЕСІВ ОБРОБКИ ДАНИХ У ЧАТ-БОТІ

2.1 Вимоги до чат-боту в телеграмі

Сучасне цифрове середовище висуває високі вимоги до якості онлайн-комунікації. З одного боку, користувачі прагнуть відкритого діалогу, з іншого – все більше зростає запит на конфіденційність та безпеку спілкування. Ідея анонімного чат-бота базується на поєднанні цих двох потреб. Передбачається створення платформи, що надає змогу будь-якій людині розпочати діалог із випадковим співрозмовником без розкриття своєї особи.

Основна функція чат-бота – з’єднання двох анонімних користувачів для спілкування в режимі реального часу. Перед початком діалогу кожному користувачу пропонується вказати базову інформацію про себе: стать та вікову категорію. Додатково можна обрати, з ким саме хотілося б поспілкуватися – це дозволяє здійснювати персоналізований підбір партнера. Після підтвердження цих налаштувань запускається процес пошуку, де система намагається знайти співрозмовника, який максимально відповідає вказаним вподобанням.

Після успішного з’єднання між користувачами відкривається чат, у якому вони можуть надсилати один одному текстові повідомлення. Усе спілкування є анонімним – жодна особиста інформація, включно з ім’ям, фото чи контактами, не передається. Діалог триває до того моменту, поки один із користувачів не вирішить завершити розмову. У такому випадку чат закривається для обох учасників, і кожен із них може повернутися в головне меню, щоб розпочати новий пошук.

Передбачається, що користувач не потребує створення облікового запису або авторизації через зовнішні сервіси. Взаємодія з ботом повинна бути якомога швидшою та простішою: відкрив, обрав налаштування, почав спілкування. Такий підхід особливо важливий у випадках, коли користувачі

прагнуть негайної психологічної підтримки, хочуть висловити думку, поставити незручне запитання або просто поговорити з кимось у зручний момент – без зобов’язань і публічності.

Таким чином, чат-бот повинен забезпечити:

– простий інтерфейс налаштування перед початком діалогу;

– можливість вибору бажаного типу співрозмовника;

– швидке з’єднання з випадковим користувачем, який відповідає вподобанням;

– комфортне спілкування без розкриття особистості;

– можливість миттєвого завершення діалогу та повернення до пошуку.

У запропонованому анонімному чат-боті єдиним типом користувача є звичайний анонімний відвідувач, який має змогу скористатися всіма функціями платформи без реєстрації та передачі особистих даних. Система орієнтована на максимально просту та швидку взаємодію, тому всі дії користувача відбуваються у кілька послідовних етапів. Основні можливості користувача описані нижче:

– запуск чат-бота. Користувач відкриває чат-бот у Telegram. Після цього він отримує доступ до функціонального WebApp-інтерфейсу, який відкривається прямо з вікна месенджера;

– автоматична аутентифікація. Система визначає Telegram-ідентифікатор користувача за допомогою автоматичної аутентифікації через WebApp. Це дозволяє розпізнати кожного користувача при наступних сесіях без необхідності ручного входу чи реєстрації;

– налаштування профілю перед спілкуванням. Користувач має можливість вказати свою стать та вікову категорію. Ці дані є важливими для подальшого підбору співрозмовника. Всі вибрані параметри зберігаються автоматично;

– вибір уподобань щодо співрозмовника. Користувач може обрати бажану стать та вікову групу партнера для спілкування. Ці вподобання не є

обов’язковими, але дозволяють підвищити релевантність майбутньої бесіди. Наприклад, користувач може вказати, що хоче спілкуватися лише з жінками віком від 25 до 32 років;

– початок пошуку співрозмовника. Після налаштування параметрів користувач натискає кнопку «Почати чат» і запускає процес пошуку. Система поміщає його в чергу очікування та намагається знайти іншого користувача з відповідними параметрами. У випадку успішного підбору обом користувачам надсилається повідомлення про початок розмови;

– очікування співрозмовника. Якщо підходящого співрозмовника ще немає, користувач бачить анімований інтерфейс очікування. Він може в будь-який момент скасувати пошук і повернутися в меню;

– участь у чаті. Коли пара сформована, користувач переходить у вікно чату. Тут він може надсилати текстові повідомлення у реальному часі. Повідомлення автоматично відображаються у чаті співрозмовника. Спілкування триває, доки один із учасників не вирішить завершити діалог;

– завершення розмови. Користувач має змогу завершити розмову в будь-який момент, натиснувши кнопку «Завершити чат». У такому випадку чат закривається, а співрозмовник отримує повідомлення про відключення;

– повернення до головного меню. Після завершення чату користувач автоматично повертається до головного екрана WebApp, де може знову змінити налаштування або почати пошук нового співрозмовника;

– оновлення своїх вподобань у будь-який момент. Користувач має можливість змінювати свій профіль – тобто обирати іншу стать, вік або змінювати вподобання щодо співрозмовника. Це можна зробити до початку кожного нового чату. Бездіяльність або вихід із бота. Якщо користувач довго неактивний або просто закриває WebApp, система не зберігає жодних чутливих даних. При повторному відкритті бот знову виконує авторизацію через Telegram.

2.2 Застосування різних технологій та бібліотек у чат-боті

Для створення анонімного чат-бота у Telegram можуть бути застосовані різні технології та бібліотеки, які дозволяють забезпечити швидку роботу сервісу, реальне спілкування через WebSocket і зручний вебінтерфейс для користувачів.

В основі побудови серверної частини може використовуватись FastAPI, що є сучасним асинхронним фреймворком на Python. Він дозволяє обробляти як класичні HTTP-запити, так і WebSocket-з'єднання, які критично важливі для чат-системи в реальному часі. Однією з сильних сторін FastAPI є його продуктивність завдяки підтримці асинхронності на базі стандарту ASGI. Крім того, він має дуже просту інтеграцію з OpenAPI для автоматичної генерації документації, що може бути корисним у великих системах. Проте певним недоліком є те, що для початківців робота з асинхронним кодом може бути складнішою у порівнянні з синхронними фреймворками.

Альтернативним варіантом для серверної частини може стати використання Flask у зв'язці з Flask-SocketIO. Flask – це класичний мікрофреймворк на Python для створення вебзастосунків, який має дуже просту структуру, що зручно для швидкого створення невеликих проектів. Через Flask-SocketIO можна забезпечити реальний обмін повідомленнями через WebSocket. Його основною перевагою є простота і велика кількість навчальних матеріалів, але недоліком буде менша продуктивність на великій кількості одночасних з'єднань, порівняно з FastAPI.

Для асинхронної обробки великої кількості користувачів можна також розглянути використання фреймворку Sanic. Це асинхронний вебсервер та фреймворк на Python, орієнтований на максимально швидку обробку запитів. Sanic дозволяє легко працювати з WebSocket і підходить для побудови високонавантажених чат-систем. Його недоліком може бути менша популярність у порівнянні з FastAPI або Flask, що означає менше готових рішень і прикладів.

На стороні клієнта основною технологією має залишатися HTML, CSS і JavaScript, що дозволяють створити інтерактивний вебінтерфейс, який буде працювати всередині Telegram WebApp. Пряме використання чистого JavaScript надає високу гнучкість і мінімальний розмір сторінок, що важливо для швидкості завантаження у мобільних додатках. Проте підтримка складної логіки у чистому JavaScript може бути важкою для масштабування.

Для побудови більш динамічного інтерфейсу можна використовувати Preact – легку альтернативу React, яка сумісна з ним на рівні API, але має набагато менший розмір бандлу. Це дозволяє отримати всі переваги компонентної архітектури без втрати продуктивності. Перевага у тому, що можна створювати інтерактивні переходи між станами користувача – меню налаштувань, режим пошуку співрозмовника, чат без перезавантаження сторінки. Недоліком може бути те, що для налаштування складної інтеграції все одно доведеться створювати окрему логіку керування станами.

Ще однією легкою бібліотекою для клієнтської частини є Alpine.js. Вона дозволяє додавати інтерактивність у HTML без створення важкої архітектури. Alpine добре підходить для невеликих проєктів, де необхідна простота: наприклад, для перемикання між вікнами «Пошук», «Чат», «Завершення». Її головним плюсом є дуже мала вага та швидкість навчання. Однак у складних сценаріях, наприклад, при обробці великих потоків даних через WebSocket, Alpine може бути обмеженим.

На сервері для роботи з базою даних може бути обраний SQLAlchemy – популярна ORM для Python, яка дозволяє працювати з базами даних через моделі. SQLAlchemy є зрілим рішенням і добре підходить для невеликих і середніх проєктів, забезпечуючи зручність при створенні таблиць для користувачів та сесій. Але для дуже великих навантажень іноді рекомендується розглядати альтернативи Tortoise ORM, яка краще оптимізована для асинхронної роботи.

Для запуску серверної частини може бути застосовано Uvicorn – легкий сервер ASGI, який відомий своєю швидкістю старту і обробки запитів. Він

ідеально поєднується із FastAPI та Sanic і підтримує роботу в асинхронному режимі. Альтернативою може бути Daphne або Hypercorn, які також підтримують ASGI, але Uvicorn зазвичай має кращу продуктивність і ширше використовується в реальних проєктах.

Як система обміну повідомленнями в реальному часі, крім класичного WebSocket API, можна розглядати використання бібліотек на кшталт Socket.IO. Socket.IO забезпечує зручний рівень абстракції над WebSocket і автоматично обробляє перепідключення та сумісність із різними браузерами. Для простого чат-бота у Telegram WebApp, однак, класичний WebSocket може виявитися кращим через меншу вагу і більшу прямоту в реалізації.

Таким чином, для розробки клієнтської частини та серверної логіки анонімного чат-бота може бути застосовано декілька різних підходів, що будуть обиратись в залежності від того, на чому робиться акцент: мінімальна вага і швидкість завантаження, масштабованість для великої кількості користувачів, або легкість у розробці та підтримці інтерфейсу.

Нижче буде розташована секція, де користувачу буде запропоновано обрати інформацію про себе. На вибір користувача будуть доступні варіанти вибору статі, приклад відображення наведено на рисунку 2.2. Користувач може обрати один із варіантів, після чого вибраний елемент змінить свій зовнішній вигляд: підсвітиться фоном та отримає кольоровий контур, що надасть користувачу видиму інформацію про активність вибору. Після вибору статі користувач зможе перейти до вибору інших характеристик або до наступного етапу налаштувань для пошуку співрозмовника. Вибрані дані будуть використовуватися для підбору відповідного партнера у чаті.

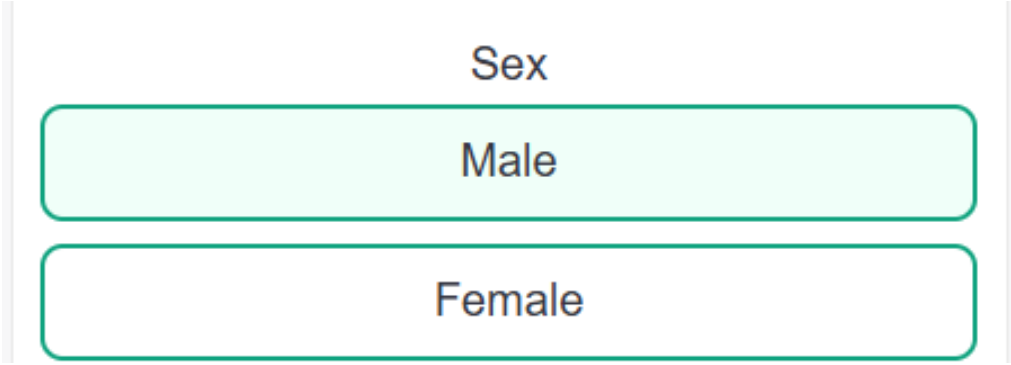

Рисунок 2.2 – Секція вибору статі

Після вибору власної статі користувач переходить до наступної секції, де необхідно вказати свою вікову категорію. На екрані буде відображено список доступних варіантів вікових груп, для прикладу якого наведено на рисунку 2.3. Користувач може обрати одну із запропонованих категорій, після чого обраний варіант буде візуально виділений: зміниться фон елемента та з’явиться кольоровий контур, що підтверджуватиме активний вибір. Вибрана вікова група надалі буде врахована при пошуку відповідного співрозмовника для чату.

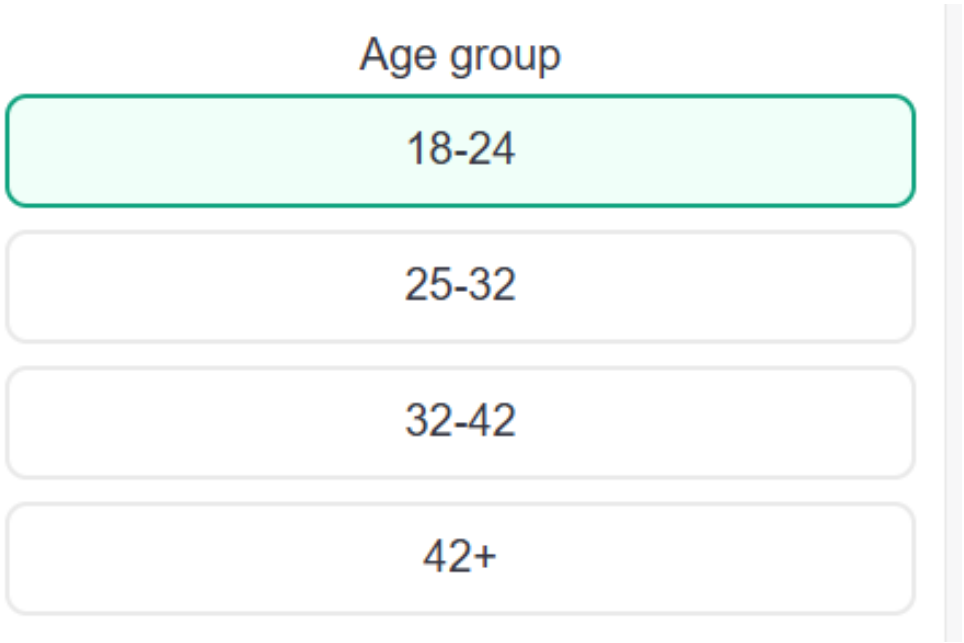

Рисунок 2.3 – Секція вибору віку користувача

Після того, як користувач обере свою власну вікову категорію, інтерфейс переходить до наступного етапу налаштувань – вибору бажаної вікової групи співрозмовника. Ця секція за структурою та оформленням буде аналогічною до попередньої.

На екрані буде представлено список доступних вікових категорій для майбутнього співрозмовника. Користувач зможе самостійно обрати одну або декілька вікових груп зі списку. При натисканні на варіант віку елемент змінить свій вигляд: активний вибір буде підсвічено іншим фоном і обведено кольоровим контуром, що дасть користувачу зрозумілий візуальний сигнал про зроблений вибір.

Це дозволить легко орієнтуватися в обраних параметрах і вносити зміни у разі потреби.

Користувач має змогу вибирати кілька вікових категорій одночасно, наприклад, якщо він бажає спілкуватися з людьми як у групі 18–24 роки, так і у групі 25–32 роки. Система надалі враховуватиме всі обрані категорії при здійсненні підбору співрозмовника, шукаючи відповідність між вподобаннями обох сторін.

Після завершення вибору вікових категорій для співрозмовника користувач може переходити до етапу запуску пошуку пари для спілкування. Усі вибрані параметри – стать та вік як користувача, так і бажаного співрозмовника – зберігаються і використовуються системою для побудови найбільш релевантного з’єднання.

Таким чином, процес налаштування профілю перед початком пошуку співрозмовника буде включати два ключові етапи: спочатку користувач вказує інформацію про себе: стать та вік, а потім задає параметри бажаного партнера для чату: стать та вікову категорію. Обидва етапи організовані у вигляді простого послідовного вибору, що робить інтерфейс інтуїтивним та зрозумілим навіть для нових користувачів.

2.3 Аналіз бази даних

У розробці сучасних чат-ботів бази даних відіграють одну з ключових ролей, забезпечуючи зберігання інформації про користувачів, їх налаштування, сесії спілкування, історії повідомлень, статистику активності та інші важливі компоненти системи. Успішна інтеграція баз даних дозволяє чат-боту бути не лише механізмом для обміну повідомленнями в реальному часі, але й інтелектуальною системою, яка адаптується до поведінки користувача, зберігає контекст розмов та підвищує загальний рівень сервісу.

Бази даних у чат-ботах можуть виконувати різні функції залежно від задачі. Наприклад, у простих ботах використовується мінімальне зберігання даних, таке як фіксація ID користувача для обробки повідомлень. У більш складних рішеннях бази даних містять розширені профілі користувачів із характеристиками, вподобаннями, станом підключення, інформацією про поточні або минулі діалоги, результатами взаємодій і навіть елементами персоналізації сервісу.

У сучасних чат-ботах найбільш поширеним є використання реляційних баз даних, таких як PostgreSQL або MySQL. Вони дозволяють створити структуровані таблиці, легко налаштовувати зв’язки між об’єктами наприклад, між користувачем і його сесіями та забезпечують високу надійність і сталість збережених даних. Для анонімного чат-бота реляційна база даних може зберігати такі таблиці, як «Користувачі», «Сесії», «Активні підключення», «Історія підбору співрозмовників», «Статистика діалогів».

Реляційні бази даних добре підходять для зберігання важливої та постійної інформації, яка потребує чіткої структури та можливості робити складні запити. Одним з основних переваг є простота валідації даних через встановлення обмежень на рівні схеми. Недоліком реляційних баз може бути складність горизонтального масштабування при зростанні навантаження, а також необхідність ретельної оптимізації при роботі з дуже великими обсягами інформації у реальному часі.

Окрім реляційних, у чат-ботах також можуть використовуватись нереляційні бази даних – наприклад, MongoDB або Redis. MongoDB підходить для зберігання неструктурованих або слабо структурованих даних у вигляді документів. Це може бути корисно, якщо бот передбачає вільний формат профілю користувача або гнучкі налаштування вподобань. Перевагою MongoDB є її висока гнучкість і масштабованість, однак недоліком може стати необхідність ручного контролю цілісності даних.

Redis використовується здебільшого для тимчасового зберігання інформації, що швидко змінюється, наприклад, активних сесій чатів або короткострокових токенів автентифікації. Завдяки роботі у пам’яті Redis забезпечує миттєвий доступ до даних із дуже низькими затримками, що ідеально підходить для систем, де швидкість реакції критично важлива. Недоліком Redis є обмеженість у використанні для довготривалого зберігання великих обсягів інформації, оскільки дані зберігаються в оперативній пам’яті.

У контексті моделювання анонімного чат-бота для Telegram WebApp використання баз даних може виглядати так.

Кожного разу, коли користувач відкриває чат-бот, його Telegram ID і базова інформація зберігаються у таблиці «Користувачі»; при вході у WebApp створюється нова сесія із унікальним токеном, який перевіряється перед початком спілкування; під час пошуку співрозмовника дані про активні очікування записуються у швидкодоступну базу таку як Redis; після підключення обох користувачів можна зберігати базову інформацію про діалог або оновлювати статус їхньої сесії.

Таким чином, бази даних дозволяють не лише реєструвати користувачів, а й керувати станами чатів, забезпечувати безперервність сесій, здійснювати швидкий пошук співрозмовників, фіксувати статистику та підвищувати загальну стабільність системи. Успішна інтеграція БД в архітектуру чат-бота безпосередньо впливає на швидкість роботи, якість підключення та зручність для користувачів.

Сучасні підходи дозволяють поєднувати різні типи баз даних: наприклад, використовувати реляційні бази для збереження основної структури користувачів і сесій, а нереляційні або in-memory бази для прискорення пошуку активних підключень або кешування проміжних станів. Такий комбінований підхід забезпечує оптимальний баланс між стабільністю, масштабованістю і швидкістю роботи чат-бота.

2.4 Моделювання архітектури алгоритмів серверної частини

У розробці серверної частини сучасних чат-ботів важливу роль відіграє правильний вибір технологій та архітектурних рішень. Серверна частина відповідає за обробку запитів користувачів, керування сесіями, взаємодію з базами даних, обробку повідомлень у реальному часі, а також за підтримку стабільної роботи всієї системи під час високих навантажень.

Одним із популярних підходів для побудови серверної логіки чат-ботів є використання асинхронних вебфреймворків, таких як FastAPI або Sanic. Асинхронна обробка запитів є особливо важливою для чат-сервісів, оскільки одночасно може бути підключено велика кількість користувачів, і система повинна швидко реагувати на їхні дії. FastAPI пропонує простий синтаксис для побудови як HTTP-ендпойнтів, так і WebSocket-з’єднань, що робить його ідеальним кандидатом для реалізації логіки спілкування у режимі реального часу. До переваг можна віднести високу продуктивність, просту інтеграцію з асинхронними бібліотеками та автоматичну генерацію документації. Водночас, для тих розробників, які мають менше досвіду з асинхронним програмуванням, робота з FastAPI може виявитися складнішою.

Альтернативним вибором може стати Flask у поєднанні з бібліотекою Flask-SocketIO. Flask є мікрофреймворком, який дозволяє швидко будувати невеликі серверні застосунки, а SocketIO дає змогу легко обробляти події обміну повідомленнями між клієнтом і сервером. Такий підхід може бути

зручним для невеликих чат-ботів або систем, де навантаження є помірним. Однак у великих проектах, де одночасно працюють тисячі користувачів, цей стек може вимагати додаткової оптимізації і горизонтального масштабування.

Для обробки WebSocket-з’єднань на сервері особливо важливо враховувати правильне розмежування станів підключень, обробку відключень та можливість швидкого реагування на події. Важливою частиною серверної архітектури є також наявність пулу очікування користувачів, обробка з’єднання пари співрозмовників, контроль за завершенням чат-сесій та безпечна робота із закриттям активних сесій.

Як сервер для розгортання серверної частини чат-бота доцільно використовувати Uvicorn або Daphne. Обидва рішення підтримують ASGI-протокол і добре працюють з асинхронними фреймворками. Uvicorn забезпечує високу продуктивність і мінімальні затримки при обробці запитів. Він добре масштабується за допомогою додаткових процесів через Gunicorn або інші подібні менеджери процесів. Недоліком такого підходу може бути складніше налаштування масштабування у порівнянні зі звичайними синхронними серверами.

У контексті анонімного чат-бота велика увага приділяється забезпеченню безперервності сеансу зв’язку між співрозмовниками. Тому необхідно моделювати механізми управління активними підключеннями, обробки скасування запитів на пошук пари, логіки повідомлення партнера про відключення співрозмовника, а також збереження мінімальної інформації про поточний стан чату для можливого подальшого аналізу чи статистики.

Також для серверної частини важливо передбачити взаємодію з базою даних. Сервер має забезпечувати безпечне зчитування та запис інформації про користувача, наприклад, його Telegram ID, статеву належність, вікову групу, створення сесій із терміном дії для підтримки автентифікації, оновлення статусу активних з’єднань та інші операції. Використання ORM,

наприклад SQLAlchemy, спрощує створення моделей даних і запитів до бази, підвищуючи безпеку і стабільність.

Для підвищення масштабованості також можна застосувати брокери повідомлень, наприклад Redis Pub/Sub або RabbitMQ, які дозволяють розділяти обробку подій між кількома серверними процесами, тим самим забезпечуючи безперебійну роботу навіть під час пікових навантажень. Проте впровадження таких рішень потребує додаткової складності в архітектурі, і їх доцільно використовувати в разі реальної необхідності масштабування.

Таким чином, серверна частина сучасного чат-бота повинна не лише ефективно обробляти запити і зберігати дані, а й забезпечувати швидке встановлення з’єднань між користувачами, підтримувати обмін повідомленнями у реальному часі без затримок, управляти станами чатів і гарантувати стабільність системи під час роботи з великою кількістю одночасних користувачів. Вибір технологій має базуватися на вимогах до продуктивності, стабільності, швидкості розробки та можливостей подальшого масштабування проєкту.

2.4.1 Алгоритми авторизації

Цей алгоритм буде забезпечувати безпечне підключення користувача до чат-бота через Telegram WebApp без необхідності ручної реєстрації. Авторизація проходить автоматично шляхом перевірки даних, що передаються Telegram при вході у WebApp:

– отримання об’єкта initData з Telegram при запуску WebApp;

– перевірка достовірності initData за допомогою розрахунку хешу та порівняння з переданим значенням;

– витягування ідентифікатора користувача Telegram ID з об’єкта initData;

– пошук користувача у базі даних за Telegram ID;

– якщо користувача не знайдено, створення нового запису користувача у базі даних;

– генерація унікального токену сесії session_token для користувача;

– створення нового об’єкта сесії у базі даних із встановленням терміну дії токену;

– встановлення токену сесії у cookie браузера користувача для подальшої авторизації.

2.4.2 Алгоритми створення або оновлення профілю користувача

Алгоритм відповідає за збереження налаштувань користувача, таких як стать, вік і критерії пошуку співрозмовника. Ці дані необхідні для правильної роботи механізму підбору пари в чаті.

– отримання ідентифікатора користувача через сесію;

– отримання даних з форми: стать, вік, бажана стать співрозмовника, бажаний вік співрозмовника;

– пошук користувача у базі даних за ідентифікатором;

– оновлення відповідних полів профілю користувача новими даними;

– збереження змін профілю користувача у базі даних;

– повернення повідомлення про успішне оновлення профілю.

2.4.3 Алгоритм перевірки авторизації

Перед кожним з’єднанням через WebSocket потрібно впевнитися, що користувач авторизований. Цей алгоритм перевіряє токен сесії у cookie та вирішує, чи дозволити підключення.

– отримання токену сесії session_token із cookie користувача під час спроби підключення до WebSocket;

– пошук сесії у базі даних за токеном сесії;

– перевірка, чи існує сесія і чи не закінчився термін її дії;

– якщо сесія валідна, дозвіл на підключення користувача до WebSocket;

– якщо сесія недійсна або прострочена, закриття підключення з відповідним повідомленням про помилку.

2.4.4 Алгоритм додавання користувача в чергу очікування

Для того, щоб знайти співрозмовника, користувача потрібно додати у чергу пошуку. Алгоритм додає користувача до списку тих, хто чекає на пару для спілкування:

– створення об’єкта підключення користувача UserConnection з його налаштуваннями профілю;

– отримання блокування доступу до списку очікування waiting_lock для синхронної роботи;

– додавання об’єкта користувача у список очікування waiting_users;

– зняття блокування після додавання користувача у чергу.

2.4.5 Алгоритм пошуку співрозмовника

Цей алгоритм автоматично підбирає співрозмовника серед користувачів, що знаходяться у черзі очікування, за їхніми налаштуваннями статі та віку:

– ініціація процесу пошуку при додаванні користувача у чергу;

– перебирання користувачів, що вже знаходяться у списку очікування;

– перевірка відповідності налаштувань обох користувачів: стать та вікова група.

– якщо знайдено підходящого співрозмовника то обидва користувача зі списку очікування видаляють, створюються пари у словнику активних з’єднань «active_pairs», надсилається обом користувачам повідомлення про початок чату.

2.4.6 Алгоритм обміну повідомленнями у реальному часі

Після створення пари між користувачами цей алгоритм забезпечує обмін повідомленнями через WebSocket без затримок:

– очікування надходження повідомлення від користувача через WebSocket;

– визначення партнера по чату через словник активних з’єднань active_pairs;

– надсилання отриманого повідомлення партнеру по чату через WebSocket;

– обробка виключень у випадку, якщо співрозмовник відключився або втратив з’єднання.

2.4.7 Алгоритм завершення сесії чату

Цей алгоритм закриває сесію чату, коли один із користувачів вирішує завершити розмову або при втраті з’єднання.

– визначення розмовника користувача за словником активних пар;

– надсилання співрозмовнику повідомлення про завершення чату;

– видалення обох користувачів із словника активних пар;

– закриття WebSocket-з’єднання для обох користувачів.

2.4.8 Алгоритм автоматичного очищення черги при відключенні

Коли користувач, який знаходиться у черзі пошуку, відключається від сервера, цей алгоритм видаляє його з черги, щоб уникнути помилок пошуку.

– виявлення факту відключення користувача із WebSocket;

– отримання блокування списку очікування waiting_lock;

– видалення користувача зі списку очікування waiting_users, якщо він там знаходиться;

– зняття блокування після очищення списку.

2.4.9 Алгоритм обробки тайм-аутів і помилок у з’єднанні

Якщо під час обміну повідомленнями трапляються помилки або користувач несподівано втрачає з’єднання, цей алгоритм обробляє такі ситуації.

– виявлення помилки під час обміну повідомленнями або при втраті з’єднання;

– виконання процедури завершення чату для обох користувачів;

– очистка даних про сесію з активних пар та списку очікування, якщо потрібно.

2.4.10 Алгоритм створення та перевірки сесії користувача

Цей алгоритм генерує унікальні токени для авторизації користувачів і перевіряє їхню дійсність при кожному запиті або підключенні.

– генерація унікального токену сесії session_token для користувача;

– створення запису про нову сесію у базі даних із встановленим терміном дії;

– збереження токену сесії у cookie користувача при відповіді від сервера;

– при кожному запиті через WebApp або WebSocket – перевірка наявності дійсної сесії за токеном;

– якщо сесія прострочена – відмова у доступі та вимога перевірити авторизацію.

3 РОЗРОБКА ЧАТ-БОТУ ДЛЯ АНОНІМНОГО СПІЛКУВАННЯ

3.1 Обґрунтування вибору середовища програмної реалізації

При розробці чат-бота для анонімного спілкування у середовищі Telegram WebApp важливим етапом стало визначення оптимального середовища розробки, яке забезпечує зручність у створенні як клієнтської, так і серверної частини системи. Основними критеріями для вибору середовища стали: зручність розгортання, підтримка сучасних бібліотек, стабільність, швидкодія, наявність інтегрованих інструментів відлагодження, а також сумісність із використовуваними технологіями.

Під час розробки чат-бота для анонімного спілкування постало питання вибору середовища розробки, яке б забезпечило максимальну зручність роботи з мовою програмування Python, технологією WebSocket, базами даних, а також створенням і редагуванням вебінтерфейсу. З-поміж багатьох доступних інструментів найкращим вибором виявився Visual Studio Code – безкоштовне, кросплатформне середовище розробки, створене компанією Microsoft.

Visual Studio Code є редактором вихідного коду, який підтримує велику кількість мов програмування, включно з Python та багатьма іншими. Це дозволило вести повноцінну розробку всієї архітектури чат-бота в одному інструменті – як серверної частини, так і клієнтської на HTML та JS.

Visual Studio Code має офіційне розширення для Python, яке надає підсвічування синтаксису, автодоповнення, перевірку типів, форматування коду black, yapf, а також вбудовану інтеграцію з середовищем виконання. Бібліотека FastAPI, яка використовується для створення API і обробки WebSocket-з’єднань, також дуже зручно розробляється у VS Code завдяки підтримці інтерактивних серверів і відлагодження запитів.

У VS Code доступний вбудований термінал, що дозволяє запускати сервери, створювати і активувати віртуальні середовища, встановлювати

залежності без потреби перемикатися на зовнішні інструменти. Це суттєво пришвидшує розробку та спрощує процес тестування сервера.

Visual Studio Code має багатий магазин розширень, серед яких були використані наступні:

– Python;

– Jinja для роботи з HTML-шаблонами;

– Live Server для попереднього перегляду вебінтерфейсу;

– REST Client або Thunder Client для тестування API-запитів без сторонніх програм;

– GitLens для зручної роботи з Git-версіями;

– Prettier, ESLint для форматування і перевірки JavaScript/HTML коду.

Для тестування WebSocket-з’єднань можна використовувати вбудовані розширення або інтегрувати з зовнішніми сервісами на зразок Postman чи wscat. VS Code дозволяє гнучко організувати логіку обробки повідомлень і запускати локальний сервер безпосередньо з інтерфейсу середовища.

Оскільки інтерфейс чат-бота створюється як Telegram WebApp, важливо мати зручний редактор для фронтенду. У VS Code чудова підтримка HTML, CSS і JavaScript: підказки, навігація по коду, перегляд структури DOM, швидке форматування та миттєве оновлення сторінки за допомогою Live Server.

Середовище має повноцінну інтеграцію з Git. Це дозволяє відслідковувати зміни в коді, створювати коміти, працювати з гілками, порівнювати редакції файлів і відновлювати історію. Усе це можна робити прямо з інтерфейсу VS Code без потреби відкривати консоль.

На відміну від важких IDE, таких як PyCharm, VS Code споживає менше ресурсів і запускається дуже швидко. Це робить його зручним навіть для слабших комп’ютерів або під час віддаленої роботи.

Для реалізації серверної частини чат-бота обирається мова програмування Python. Такий вибір був зроблений на основі її універсальності, простоти синтаксису, широкої підтримки сучасних

вебтехнологій, а також активної спільноти розробників. Python дозволяє швидко створювати надійні вебзастосунки, легко інтегрується з базами даних, підтримує асинхронне програмування та має потужні бібліотеки для роботи з WebSocket-з'єднаннями, що особливо актуально для реалізації чату в реальному часі.

Крім того, Python є оптимальним вибором для проєктів, де важливо зосередитись на логіці роботи системи, а не на технічних складностях мови. Його лаконічність дозволяє скоротити обсяг коду без втрати функціональності, що значно пришвидшує процес розробки. Завдяки використанню фреймворку FastAPI вдалося побудувати сучасну архітектуру серверної частини, яка працює з високою продуктивністю та добре масштабується.

Python також зручно поєднується з такими бібліотеками, як SQLAlchemy, Uvicorn, та Jinja2, що у сукупності дозволяє реалізувати повноцінний вебсервіс із підтримкою клієнтської частини, яка вбудовується у Telegram WebApp.

Саме поєднання простоти, широкого функціонального потенціалу та активного розвитку мови Python зробило її найбільш раціональним і зручним вибором для реалізації даного програмного продукту.

А також Python є однією з найпопулярніших мов програмування у сфері створення чат-ботів завдяки своїй гнучкості, простоті та великій кількості готових рішень. Його використання дозволяє реалізовувати як прості сценарії взаємодії з користувачем, так і складні високонавантажені системи з підтримкою обробки повідомлень у реальному часі.

Однією з ключових переваг Python є наявність великої кількості бібліотек та фреймворків, які спеціально орієнтовані на розробку ботів, вебзастосунків і API. Наприклад, FastAPI забезпечує високу продуктивність і зручність у створенні як HTTP-, так і WebSocket-інтерфейсів. Це критично важливо для чатів, де обмін повідомленнями має відбуватись миттєво й без затримок.

Також значною перевагою є ORM-бібліотеки, такі як SQLAlchemy, які дозволяють швидко і безпечно працювати з базами даних без потреби писати сирі SQL-запити. Це суттєво спрощує збереження інформації про користувачів, їх сесії, параметри, історію чату та інші дані.

Ще однією сильною стороною Python є простота синтаксису. Завдяки цьому розробка чат-ботів стає доступною навіть для розробників-початківців, а підтримка й масштабування таких систем не викликає труднощів у майбутньому. Python дозволяє зосередитись на логіці роботи бота, а не на низькорівневих деталях реалізації.

Окремо варто відзначити інтеграцію Python із Telegram API та Telegram WebApp. Для роботи з Telegram існують спеціальні бібліотеки, які полегшують обробку повідомлень і підключення до користувачів. Крім того, мова добре підтримує створення клієнтської частини через шаблонізатори на зразок Jinja2, що дозволяє легко реалізовувати WebApp без використання важких фронтенд-фреймворків.

Також Python чудово підходить для запуску на будь-яких платформах і має потужні можливості для розгортання в хмарних сервісах, на VPS або контейнерах Docker. Це дозволяє легко перенести бота з локального середовища на продуктивний сервер.

Таким чином, Python забезпечує весь необхідний набір інструментів для створення сучасного, стабільного та масштабованого чат-бота: від логіки спілкування і маршрутизації повідомлень — до зберігання даних та взаємодії з користувачем через Telegram WebApp. Це робить його одним з найкращих виборів для реалізації чат-систем будь-якої складності.

У процесі створення сучасного чат-бота важливу роль відіграє правильно організоване зберігання та обробка інформації. Саме тому використання бази даних є невіддільною частиною архітектури будь-якої стабільної та функціональної системи спілкування. База даних дозволяє зберігати, структурувати та швидко обробляти інформацію про користувачів, їх профілі, сесії, вподобання, а також історію з’єднань і повідомлень. Це дає

змогу забезпечити логіку чат-бота, яка не залежить від одного сеансу і здатна масштабуватись із зростанням кількості користувачів.

Однією з ключових переваг бази даних є можливість постійного зберігання даних навіть після завершення сеансу. Це означає, що користувачі можуть повертатись до чат-бота пізніше, а система буде «пам’ятати» їх попередні дії, налаштування та історію. У випадку анонімного чат-бота з фільтрацією співрозмовників за статтю та віком, саме база даних зберігає вибрані параметри та забезпечує коректний підбір пари на основі цієї інформації.

Ще однією перевагою є використання бази даних для управління сесіями. Після авторизації користувача через Telegram WebApp система створює сесію з унікальним токеном, яка зберігається у базі з обмеженим терміном дії. Це забезпечує захищений доступ до функціоналу WebApp, дозволяє відстежувати стан користувача та запобігає несанкціонованому доступу.

База даних також виконує роль центрального джерела інформації під час роботи алгоритму пошуку співрозмовника. Усі користувачі, які перебувають у черзі, мають збережені налаштування у базі. Це дозволяє серверу швидко зіставити параметри двох людей і прийняти рішення про створення пари. У разі, якщо користувач припинив з’єднання або покинув чат, база даних дозволяє коректно оновити його статус і уникнути помилок у логіці системи.

У розробці чат-бота ефективною є реалізація бази даних за допомогою ORM-технології – зокрема, бібліотеки SQLAlchemy. Вона дозволяє працювати з даними через об’єктно-орієнтовану модель, замість написання сирих SQL-запитів. ORM-структура спрощує моделювання сутностей: користувачів, сесій, параметрів, пар чату тощо. Це робить код більш читабельним, безпечним та зручним для масштабування.

Використання реляційної бази даних дозволяє чітко структурувати таблиці, встановлювати зв’язки між ними та забезпечувати цілісність даних.

У невеликих проєктах може бути застосована локальна база SQLite, яка не потребує окремого сервера і працює прямо у файловій системі. У більш складних або масштабованих проєктах база даних легко переноситься на сервер PostgreSQL або MySQL без зміни логіки програми.

Перевагою централізованого зберігання є також можливість збору статистики: скільки сесій було створено, скільки пар було сформовано, які параметри найчастіше обирають користувачі. Ці дані можуть бути використані для подальшої аналітики або вдосконалення логіки підбору співрозмовників.

Таким чином, використання бази даних у розробці чат-бота дозволяє забезпечити надійність, структурованість, безпеку й масштабованість проєкту. База даних є необхідною складовою будь-якого сервісу, що передбачає роботу з користувачами, їх профілями, станами підключення та взаємодією в реальному часі.

3.2 Створення моделей в SQLAlchemy

У процесі розробки серверної частини чат-бота для анонімного спілкування було використано бібліотеку SQLAlchemy для створення та керування моделями бази даних. SQLAlchemy є потужним інструментом, що дозволяє будувати зв’язки між об’єктами Python і таблицями у реляційній базі даних, що значно спрощує роботу з даними.

Було створено дві основні моделі: User та Session. Кожна з цих моделей представляє окрему таблицю в базі даних, з чітко визначеною структурою та логікою зберігання інформації.

Завдяки створеним моделям у SQLAlchemy структура зберігання даних є гнучкою та масштабованою, що дозволяє зручно додавати нові поля чи таблиці у майбутньому. Крім того, така організація коду дозволяє легко взаємодіяти з даними за допомогою об’єктно-орієнтованого підходу, що зменшує кількість помилок і підвищує зручність підтримки проєкту.

Таблиця 3.1 в базі даних – це таблиця користувачів. Її основне призначення – зберігати дані про кожного користувача Telegram-бота, які необхідні для коректного функціонування системи підбору співрозмовника. Через цю таблицю система ідентифікує користувача за його Telegram ID, визначає його особисті характеристики, такі як стать і вік, а також зберігає його побажання щодо майбутніх співрозмовників. Завдяки цьому бот може коректно фільтрувати потенційних кандидатів для діалогу згідно з заданими критеріями користувача. Таким чином, таблиця є критично важливою для персоналізації взаємодії та правильної логіки поєднання людей у чатах. Вона дозволяє зберігати налаштування, що визначають поведінку чат-бота під час кожної сесії спілкування.

Таблиця 3.1 – Таблиця моделі «users»

Назва полів	**Типи**
id	Integer
telegram_id	Integer
gender	Enum
age	Enum
target_genders	String
target_ages	String

Таблиці 3.2 в базі даних – це таблиця сесій. Її головне призначення полягає в тому, щоб відстежувати активні сесії користувачів, які взаємодіють із Telegram-ботом. Вона зберігає унікальний токен сесії, що дозволяє однозначно ідентифікувати поточну взаємодію користувача з ботом, а також пов’язує кожну сесію з конкретним користувачем за допомогою зовнішнього ключа. Таблиця також фіксує час завершення сесії, що дозволяє системі автоматично деактивувати неактуальні або застарілі сесії. Завдяки цьому таблиця забезпечує безперервність та логічну послідовність у роботі чат-бота. Бот може коректно реагувати на повідомлення, зберігати контекст розмови або завершувати її при необхідності. Це особливо важливо для

чатів-рулеток, де користувачі мають можливість швидко змінювати співрозмовників і мати окрему історію взаємодії в межах кожної сесії.

Таблиця 3.2 – Таблиця моделі «sessions»

Назва полів	**Типи**
id	Integer
session_token	String
user_id	Integer
expires_at	DateTime

3.3 Демонстрація роботи програми

Під час демонстрації роботи чат-бота буде послідовно показано основний функціонал, реалізований у межах цього проєкту. Демонстрація охоплює базовий сценарій взаємодії користувача з ботом – від початкового запуску до вибору параметрів пошуку співрозмовника та процесу з'єднання. Зокрема, буде продемонстровано, як користувач може обрати свою стать, вік, а також вподобання щодо статі й віку співрозмовника. На основі цих даних здійснюється пошук іншого користувача з відповідними параметрами.

У межах демонстрації будуть показані ключові етапи роботи бота: відображення інтерфейсів для вибору параметрів, повідомлення про очікування співрозмовника, надсилання повідомлень у сесії та можливість завершити розмову. Основна увага приділена логіці взаємодії та візуальному оформленню діалогу користувача з ботом, а також передачі інформації в рамках поточної сесії.

Демонстрація не охоплює внутрішню обробку помилок або логіку збереження даних у базу даних, оскільки ці процеси реалізовані на серверній частині програми та відбуваються у фоновому режимі без виведення додаткової інформації для користувача. Всі ілюстрації будуть подані у вигляді скріншотів, які наочно демонструють кроки взаємодії з ботом та його реакції на дії користувача. Завдяки цьому читач зможе оцінити загальну

логіку роботи чат-бота та ознайомитися з його функціональними можливостями з точки зору користувача.

Після того, як користувач переходить до чату з ботом у Telegram, на екрані з’являється кнопка «Launch». Натискання цієї кнопки ініціює запуск бота. У відповідь бот активується та надсилає привітальне повідомлення або пропозицію розпочати налаштування. Це дозволяє користувачеві швидко й інтуїтивно зрозуміти, як почати взаємодію. Кнопка «Launch» є стандартною функцією Telegram для ботів, що полегшує старт роботи без необхідності вводити команди вручну, як на рисунку 3.1.

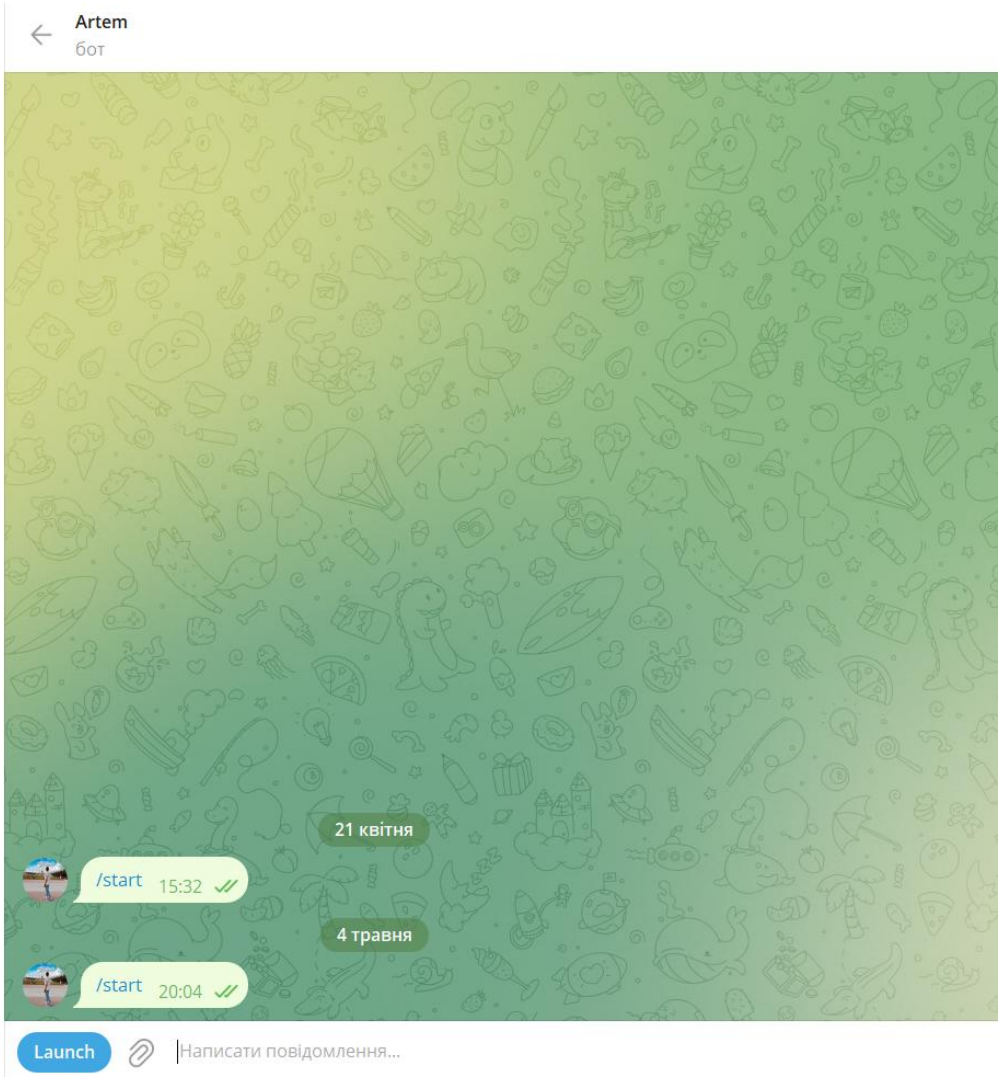

Рисунок 3.1 – Запуск чат-бота через кнопку «Launch» у Telegram

На зображенні показано, як користувач натискає кнопку «Launch» у Telegram, що ініціює запуск бота. Після цього бот готовий до взаємодії з користувачем, що є першим кроком у процесі роботи системи.

Після запуску чат-бота користувачеві автоматично надсилається меню для налаштування основної інформації про себе. В інтерактивному інтерфейсі користувач обирає свою стать та вікову категорію з попередньо заданих варіантів. Ці дані необхідні для формування профілю та коректного підбору співрозмовника згідно із зазначеними критеріями. Завдяки цьому бот має змогу враховувати побажання кожного користувача та забезпечити більш персоналізований досвід спілкування в рамках чат-рулетки.

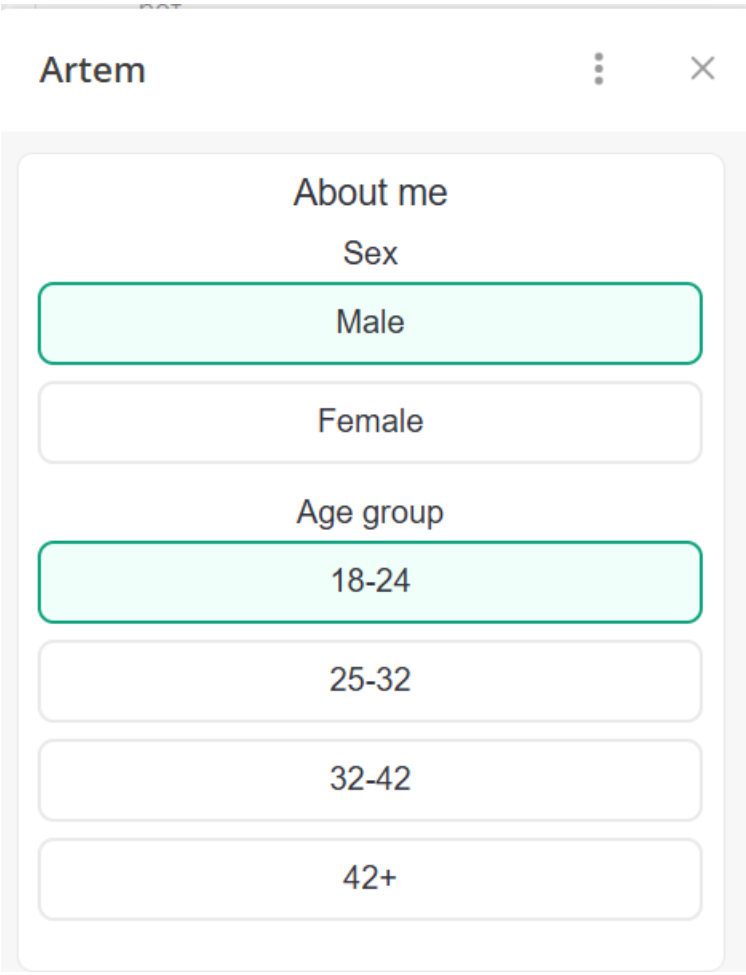

Рисунок 3.2 – Вибір статі та вікової категорії користувача

Після того як користувач зазначає інформацію про себе – обирає стать та вікову групу – бот переходить до наступного етапу налаштування. На

цьому етапі з’являється нове меню, в якому користувач вказує бажану стать співрозмовника. Це дозволяє системі врахувати переваги користувача при пошуку відповідного співрозмовника. Такий підхід підвищує ймовірність вдалого підбору та сприяє більш комфортному та релевантному спілкуванню в чаті, як продемонстровано на рисунку 3.3.

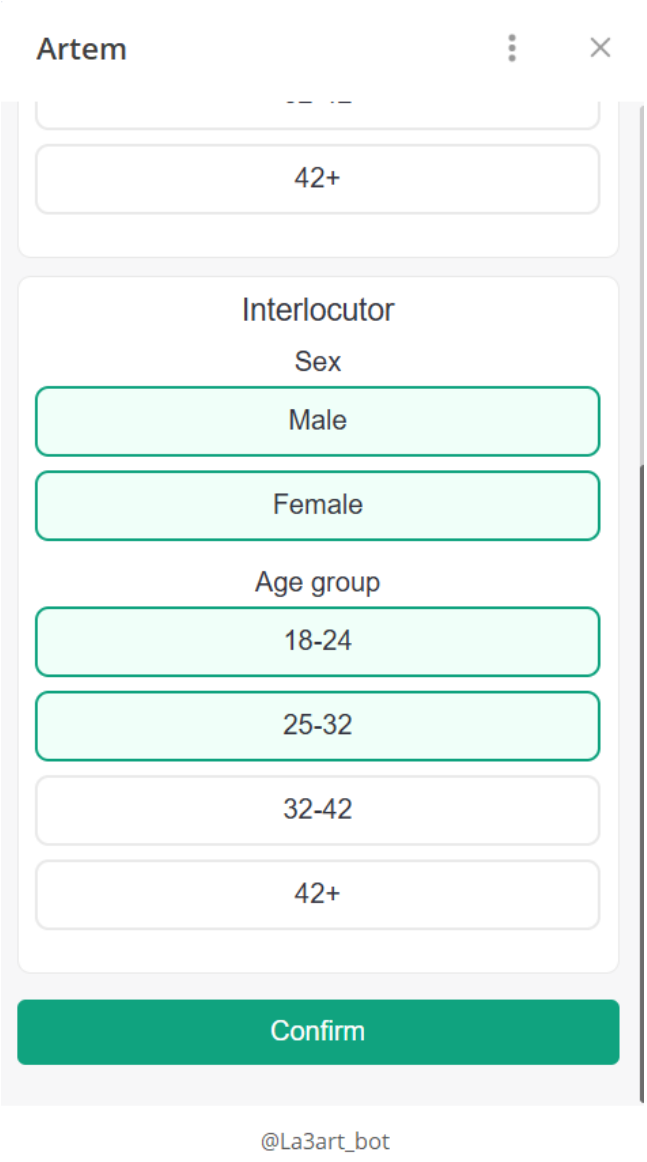

Рисунок 3.3 – Вибір статі та вікової категорії співрозмовника

Після того як користувач завершив налаштування свого профілю – обравши стать, вікову категорію, а також бажані параметри для співрозмовника – бот переходить до етапу пошуку. Система аналізує всі доступні профілі в базі даних, які наразі не зайняті у спілкуванні, та порівнює

їх характеристики з заданими критеріями користувача, як продемонстровано на рисунку 3.4.

Фільтрація відбувається на основі наступних параметрів: стать співрозмовника повинна збігатися з тим, що зазначив користувач у налаштуваннях, і його вік має входити до обраного діапазону. Аналогічно, також перевіряється, чи відповідає сам користувач критеріям, які вказав потенційний співрозмовник. Таким чином, підбір відбувається двосторонньо – обидві сторони повинні задовольняти вимоги одна одної.

Як тільки знаходиться відповідна пара, бот створює нову сесію та з’єднує користувачів у приватному чаті. Вони можуть почати спілкування одразу, а бот у фоновому режимі контролює статус сесії та за потреби дозволяє завершити розмову й перейти до пошуку нового співрозмовника.

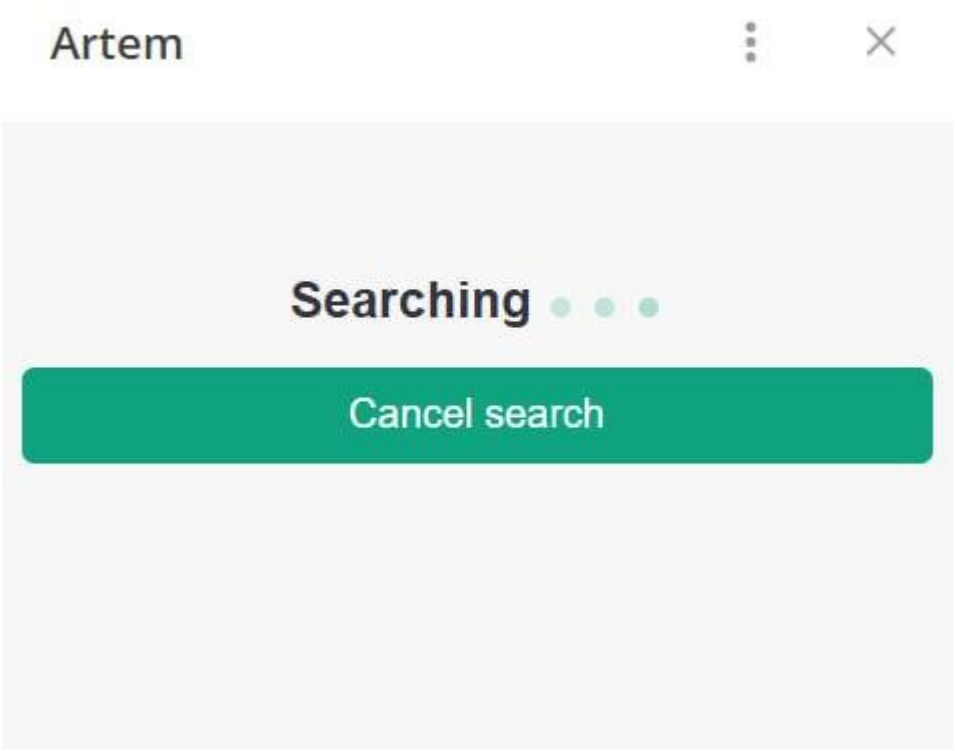

Рисунок 3.4 – Демонстрація етапу пошуку співрозмовника

Після того як бот знаходить відповідного співрозмовника відповідно до заданих параметрів, автоматично створюється нова сесія спілкування. Користувачу надходить повідомлення, що співрозмовника знайдено, і одразу

відкривається чат, у якому обидві сторони можуть розпочати анонімне спілкування. У цьому чаті відсутні будь-які персональні дані – повідомлення надсилаються через бота, що забезпечує повну конфіденційність.

Користувач може надсилати текстові повідомлення, а бот пересилає їх співрозмовнику. Таким чином, обидва учасники бачать лише повідомлення одне одного, не маючи доступу до жодної ідентифікуючої інформації. Спілкування триває доти, доки один із користувачів не вирішить завершити розмову, після чого бот знову запропонує можливість пошуку нового співрозмовника, це продемонстровано на рисунку 3.5.

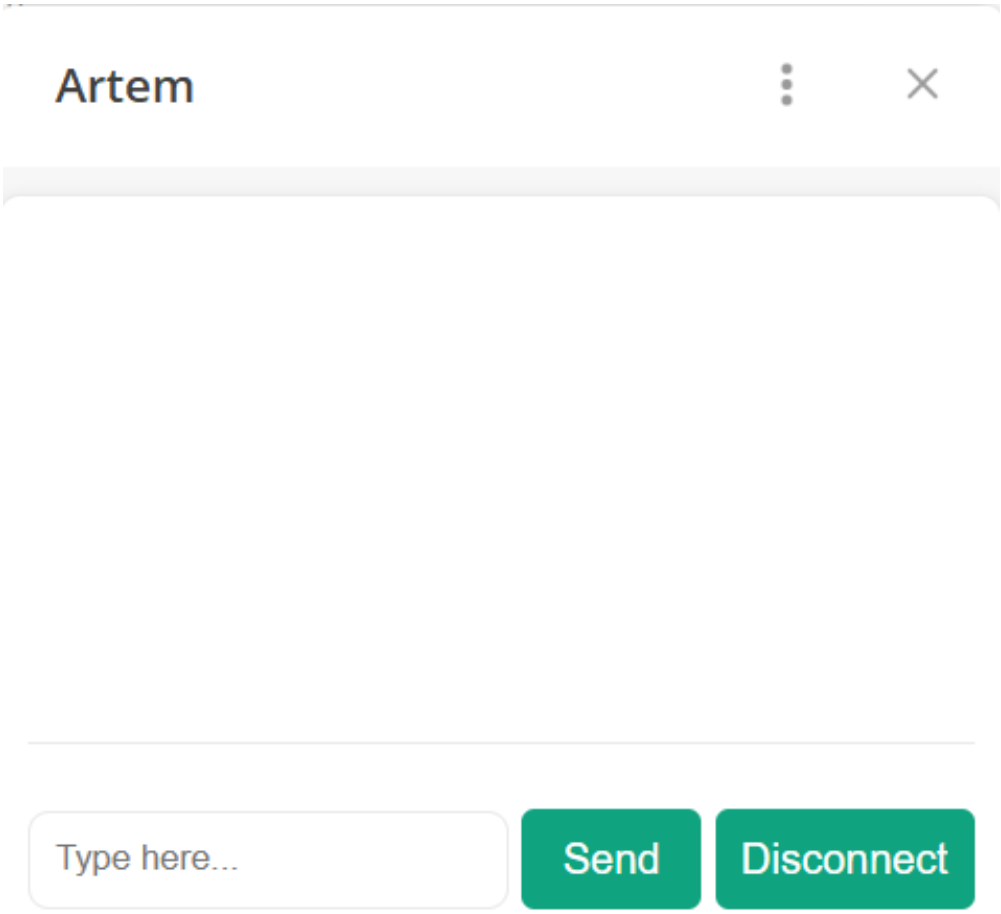

Рисунок 3.5 – Демонстрація чату з співрозмовником

Далі буде продемонстровано приклад безпосереднього спілкування між двома анонімними користувачами, на рисунку 3.6. Після підбору співрозмовника бот відкриває захищений чат, у якому обидва учасники

можуть вільно обмінюватися повідомленнями. Як видно на скріншоті, один із користувачів ставить питання, пов’язане з особистими переживаннями, а інший надає підтримку та відповідає – спілкування є щирим та конфіденційним.

Бот виступає посередником: він приймає повідомлення від одного учасника і надсилає іншому, не розкриваючи особистих даних. Кожен користувач може завершити розмову в будь-який момент, натиснувши кнопку «Відключитися». Таким чином забезпечується як емоційна безпека, так і технічна анонімність.

.

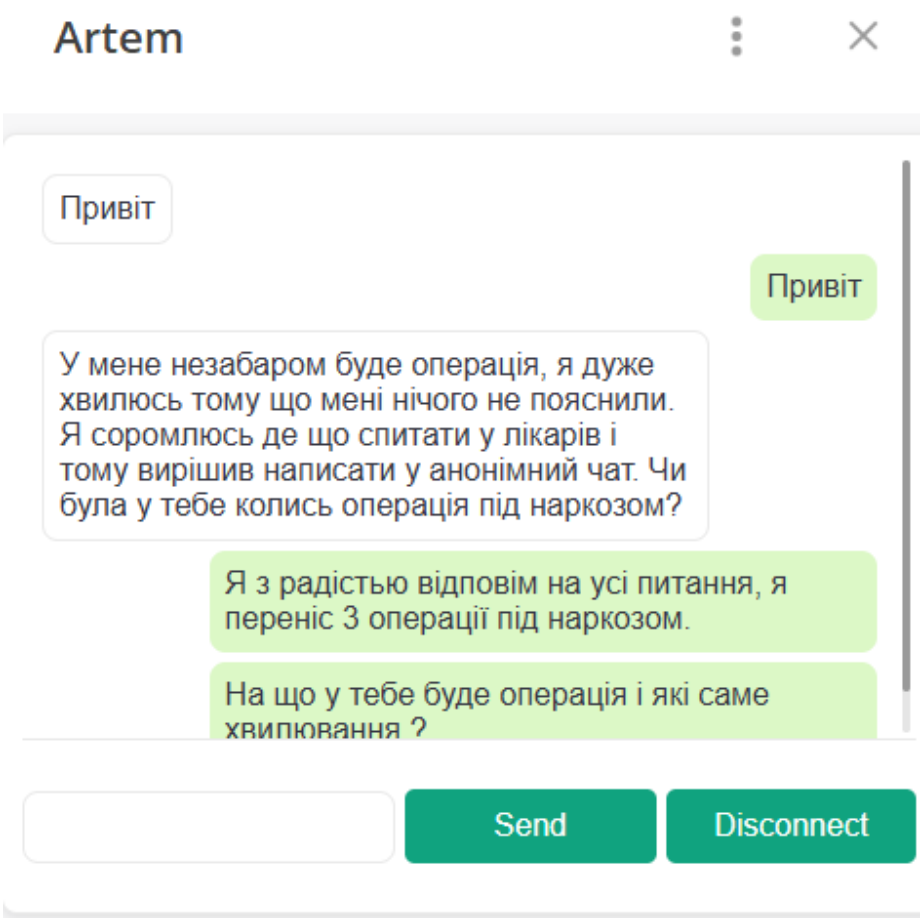

Рисунок 3.6 – Демонстрація спілкування чату двох співрозмовників

Окрім основного функціоналу анонімного спілкування, чат-бот також має додаткові можливості, які роблять використання зручнішим та гнучкішим. Зокрема, реалізована функція припинення пошуку

співрозмовника, як на рисунку 3.7. Якщо користувач більше не хоче продовжувати очікування на підбір, він може натиснути відповідну кнопку, і бот миттєво зупиняє процес пошуку. Це дозволяє уникнути зайвих очікувань і надає користувачу повний контроль над взаємодією з ботом у будь-який момент.

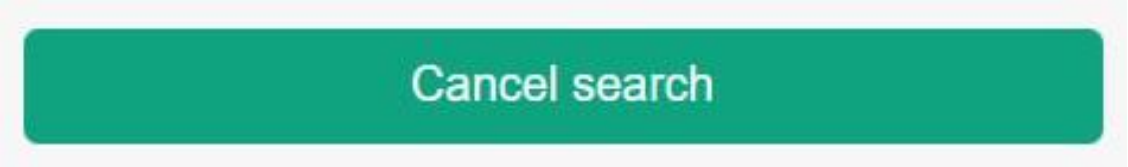

Рисунок 3.7 – Демонстрація кнопки припинення пошуку

Окрім основного функціоналу, який полягає у пошуку анонімного співрозмовника та організації спілкування, бот також надає користувачу доступ до трьох додаткових кнопок, що виконують допоміжні, але важливі функції, демонстрація на рисунку 3.8. Перша кнопка – це «Оновити сторінку». Вона дозволяє перезапустити інтерфейс бота у випадках, коли трапляється збій, кнопки не реагують або потрібно заново ініціалізувати взаємодію. Це зручно, оскільки дозволяє не чекати автоматичного оновлення, а самостійно повернутися до стабільного стану.

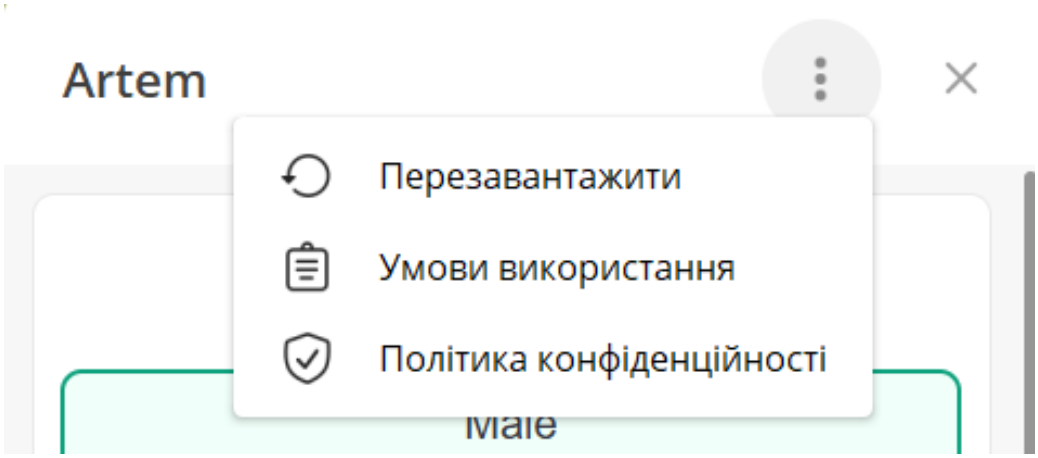

Рисунок 3.8 – Демонстрація трьох додаткових кнопок

Друга кнопка – «Правила користування». Натиснувши на неї, користувач відкриває перелік основних правил, які регулюють використання

бота. Там детально описано, як правильно поводитися в анонімному чаті, що заборонено (наприклад, образи, спам, реклама тощо), а також що варто знати перед початком спілкування. Це дозволяє зберігати безпечне, комфортне середовище для всіх учасників.

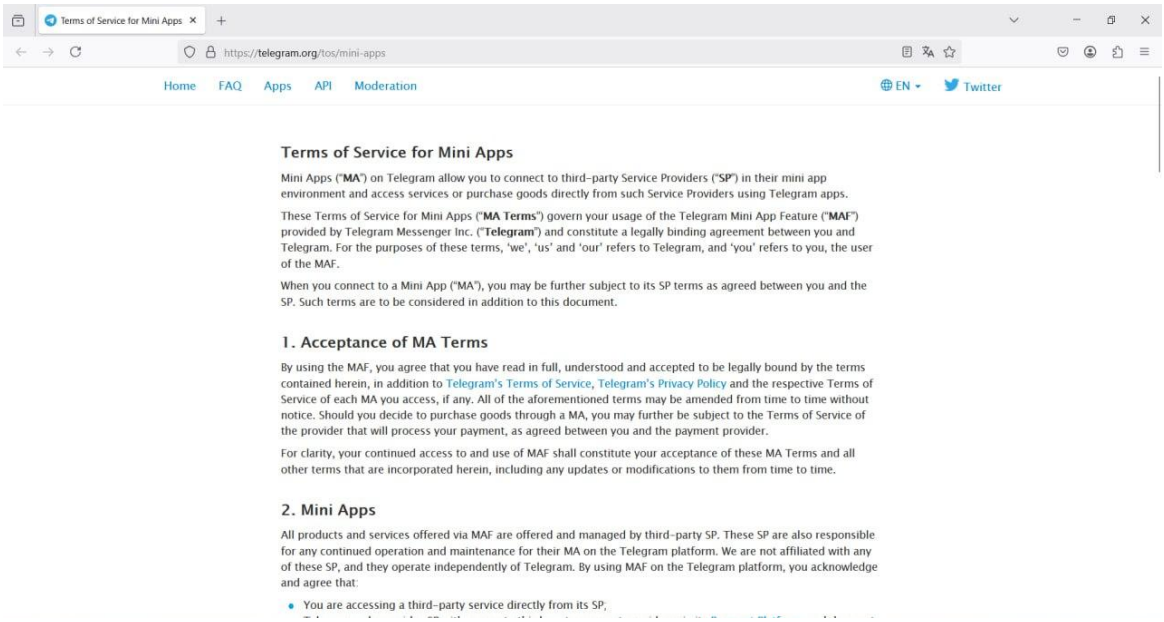

Рисунок 3.9 – Демонстрація переходу на правила користування

Третя кнопка – «Політика конфіденційності», яка продемонстрована на рисунку 3.10. Вона пояснює, які саме дані збираються, з якою метою, де зберігаються і хто має до них доступ. Це дуже важливо, оскільки користувачі можуть залишати особисту інформацію в процесі спілкування. Завдяки прозорій політиці конфіденційності бот демонструє відповідальне ставлення до захисту приватності та даних користувачів, підвищуючи рівень довіри до сервісу.

Загалом у чат-боті передбачено додаткові елементи функціональності, які не належать до основного процесу пошуку чи спілкування, проте суттєво покращують досвід користувача. До них належать інтерфейсні кнопки, що надають доступ до інформаційних розділів, таких як «Правила користування» та «Політика конфіденційності». Це дозволяє користувачу ознайомитися з умовами використання сервісу безпосередньо в інтерфейсі

бота, що є важливою складовою з точки зору прозорості та безпеки. Крім того, реалізовано кнопку оновлення, яка дає змогу перезапустити або оновити стан чату в разі технічних збоїв або затримок.

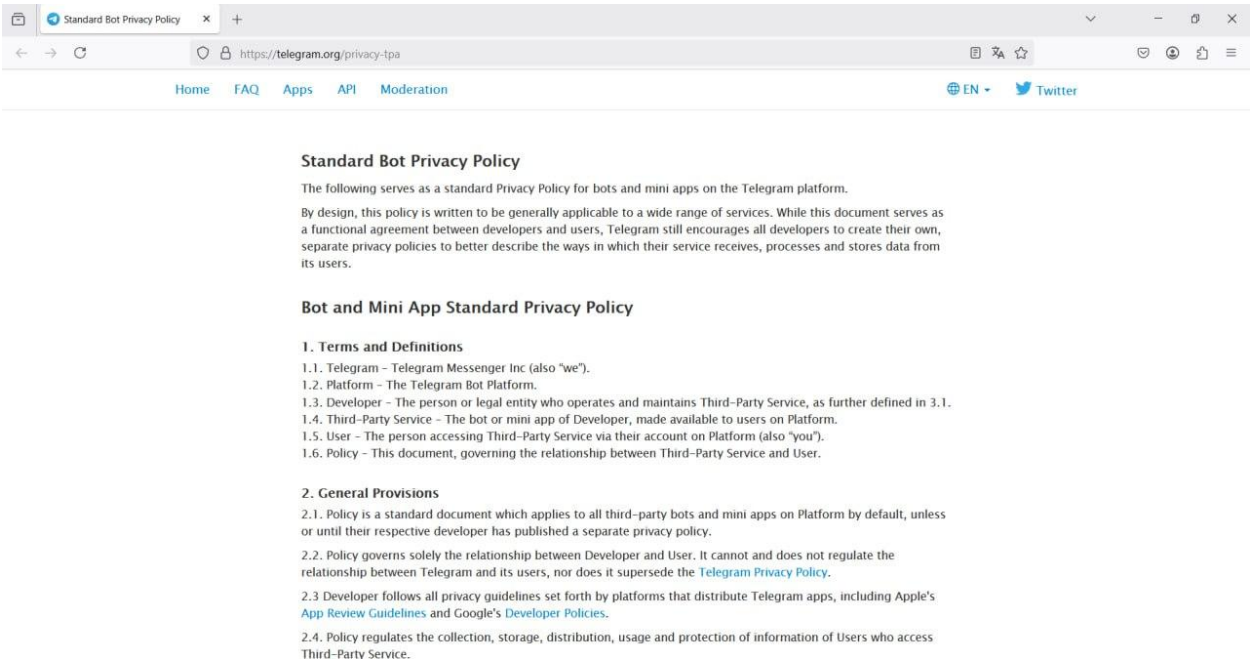

Рисунок 3.10 – Демонстрація переходу на політику конфіденційності

ВИСНОВКИ

У кваліфікаційній роботі розібралися і вивчили моделювання та аналітичну підготовку до створення Telegram чат-бота для анонімного спілкування у форматі чат-рулетки. Основною метою було вивчити як працюють системи, які технічні рішення та інструменти можна використовувати у чат-боті і які алгоритми забезпечують швидку роботу.

У роботі були зроблені всі поставлені аналітичні завдання, а саме:

- було розглянуто сучасний стан технологій у сфері чат-ботів у середовищі Telegram;
- використано приклади реалізації популярних анонімних чатів, зокрема з точки зору архітектури, логіки роботи, безпеки, та варіантів масштабування;
- обґрунтовано вибір мови програмування, середовища розробки, системи керування базами даних, а також серверної інфраструктури для майбутньої реалізації;
- описано потенційні алгоритми, які можуть бути застосовані у функціонуванні бота;
- виявлено переваги та недоліки використання різних технологій та бібліотек;
- створено схеми можливого інтерфейсу, алгоритмів взаємодії користувачів та логіки роботи бота.

Усі ці результати будуть гарною основою для наступного етапу – створення чат-бота, його тестування та оптимізації. Потрібно знати, що на цьому етапі все ще є можливість змінювати або доповнювати моделі та рішення. Вони можуть бути змінені або доповнені в процесі реалізації. Створена основа дає змогу ефективно перейти до етапу розробки, враховуючи аналітичні висновки та підготовлені технічні рішення.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Kobylin, O., Vyskrebentseva, S., & Petrova, R. (2019). Обробка даних, що містять пропуски в задачах кластеризації. *Системи управління, навігації та зв'язку. Збірник наукових праць*, *5*(57).

2. Mashtalir, V., Ruban, I., & Levashenko, V. (Eds.). (2019). Advances in Spatio-Temporal Segmentation of Visual Data (Vol. 876). Springer Nature.

3. Kobylin, O. A., Gorokhovatskyi, V. O., Tvoroshenko, I. S., & Peredrii, O. O. (2020). The application of non-parametric statistics methods in image classifiers based on structural description components. Telecommunications and Radio Engineering, 79(10).

4. Gorokhovatskyi, V., & Tvoroshenko, I. (2023). Identification of visual objects by the search request. *International scientific symposium «INTELLIGENT SOLUTIONS-S»* (pp. 25-27).

5. Daradkeh, Y. I., Gorokhovatskyi, V., Tvoroshenko, I., Gadetska, S., & Al-Dhaifallah, M. (2023). Statistical data analysis models for determining the relevance of structural image descriptions. IEEE Access, 11, 126938-126949.

6. Gorokhovatskyi V., Tvoroshenko I., Kobylin O., and Vlasenko N. (2023) Search for visual objects by request in the form of a cluster representation for the structural image description, Advances in Electrical and Electronic Engineering, 21(1), pp. 19-27.

7. Kobylin, O. A., Gorokhovatskyi, V. O., Tvoroshenko, I. S., & Peredrii, O. O. (2020). The application of non-parametric statistics methods in image classifiers based on structural description components. Telecommunications and Radio Engineering, 79(10).

8. Ibrahim, D. Y., Gorokhovatskyi, V., Tvoroshenko, I., & Mujahed, A. D. (2022). Classification of Images Based on a System of Hierarchical Features.

9. Rabotiahov, A., Kobylin, O., Dudar, Z., & Lyashenko, V. (2018, February). Bionic image segmentation of cytology samples method. In *2018 14th*

International Conference on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET) (pp. 665-670). IEEE.

10. Kobylin, O., & Lyashenko, V. (2016). Contrast Modification as a Tool to Study the Structure of Blood Components.

11. Kinoshenko, D., Kobylin, O., Mashtalir, S., & Stolbovyi, M. (2019, March). Metric video retrieval speedup by irrelevant data elimination. In Eleventh International Conference on Machine Vision (ICMV 2018) (Vol. 11041, pp. 176-183). SPIE.

12. Gorokhovatskyi, V. A., Rusakova, N., & Tvoroshenko, I. S. (2020). The application of image analysis methods and predicate logic in applied problems of magnetic monitoring. Telecommunications and Radio Engineering, 79(20).

13. Daradkeh, Y. I., & Tvoroshenko, I. (2020). Technologies for making reliable decisions on a variety of effective factors using fuzzy logic. International Journal of Advanced Computer Science and Applications, 11(5).

14. Gorokhovatskyi, V. O., Tvoroshenko, I. S., & Vlasenko, N. V. (2020). Using fuzzy clustering in structural methods of image classification. Telecommunications and Radio Engineering, 79(9).

15. Yakovleva, O., & Nikolaieva, K. (2020). Research of descriptor based image normalization and comparative analysis of SURF, SIFT, BRISK, ORB, KAZE, AKAZE descriptors. Advanced Information Systems, 4(4), 89-101.

16. Daradkeh, Y. I., & Tvoroshenko, I. (2020). Technologies for making reliable decisions on a variety of effective factors using fuzzy logic. International Journal of Advanced Computer Science and Applications, 11(5).

17. Pomazan, V., Tvoroshenko, I., & Gorokhovatskyi, V. (2023). Development of an application for recognizing emotions using convolutional neural networks. *International Journal of Academic Information Systems Research*, 7(7), (pp. 25-36).

18. Lyashenko V., Kobylin O., Selevko O. (2020) Wavelet Analysis and Contrast Modification in the Study of Cell Structures Images. International Journal of Advanced Trends in Computer Science and Engineering. 9(4). – 4701-4706.

19. Oleg, K., Sergii, M., & Mykhailo, S. (2017, October). Video Clustering via Multidimensional Time-Series Analysis. In *Proceedings of the 9th International Conference on Information Management and Engineering* (pp. 60-63). ACM.

20. Кобилін, О. А., & Творошенко, І. С. (2021). Методи цифрової обробки зображень.

21. Yakovleva, O., Kovtunenko, A., Liubchenko, V., Honcharenko, V., & Kobylin, O. (2023). Face Detection for Video Surveillance-based Security System. In COLINS (3) (pp. 69-86).

22. Yakovleva, O., Slyusar, V., Kushnir, O., & Sabovchyk, A. (2021). New trends in scientific and technological revolution (STR) and transformation of science and education systems in the paradigm of sustainable development. In E3S Web of Conferences (Vol. 277, p. 06006). EDP Sciences.

23. Bodyanskiy, Y., Vynokurova, O., Kobylin, I., & Kobylin, O. (2016). Adaptive fuzzy clustering of short time series with unevenly distributed observations in Data Stream Mining tasks. *Information Technology and Management Science*, *19*(1), 23-28.

24. Gorokhovatskyi, V., Tvoroshenko, I., & Olena, Y. (2024). Transforming image descriptions as a set of descriptors to construct classification features. *Indonesian Journal of Electrical Engineering and Computer Scienc*, 33 (1), (pp. 113-125)

25. Kuzminska, O., Mazorchuk, M., Morze, N., & Kobylin, O. (2019, June). Digital learning environment of ukrainian universities: The main components to influence the competence of students and teachers. In International Conference on Information and Communication Technologies in Education, Research, and Industrial Applications (pp. 210-230). Springer, Cham.

26. Кобилін, О. А., & Путятіна, О. Є. (2024). Знешумлення зображень, зіпсованих дробовим шумом, у реальному часі. Системи обробки інформації, (1 (176), 46-51.

27. Gorokhovatskyi , V., Chmutov , Y., Tvoroshenko , I., & Kobylin , O. (2025). Reducing computational costs by compressing the structural description in image classification methods. Advanced Information Systems, 9(1), 5–12.

28. Кобилін, О., Вечірська, І., Афанасьєв, А. (2024). Аналіз існуючих моделей глибинного навчання в задачах обробки природної мови. Information Technology: Computer Science, Software Engineering and Cyber Security, 3, 63–76,

29. Кобилін, О., Вечірська, І., Кравченко, О. (2024). Порівняння нейронних мереж типу RNN та LSTM. Information Technology: Computer Science, Software Engineering and Cyber Security, 3, 97–107,

30. Vechirska, I., Kobylin, O., Prokopiev , S., Vechirska, A., & Kucherenko, M. (2022). Building a logical network for solving the problem of car rental by means algebra of finite predicates. Computer Systems and Information Technologies, (2), 78–87.

31. Kobylin, O.; Putiatina, O. (2025). Some aspects of real-time image denoising influenced by shot noise and compound Poisson noise. CEUR Workshop Proceedings ,volume = 3943 , 109-117.