МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

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

«До захисту допущено»
В.о. завідувача кафедри
_____ Ігор Шелехов
____ грудня 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня магістр

зі спеціальності 122 - Комп'ютерних наук,
освітньо-професійної програми «Інформатика»
на тему: «Інформаційна технологія підбору та пошуку інформаційного контенту для емігрантів»
здобувача групи ІН.м -24 Назаренка Олексія Васильовича

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

_____ Олексій Назаренко

Науковий керівник,
кандидат технічних наук _____ Олег Берест

Суми – 2023

Сумський державний університет

Факультет електроніки та інформаційних технологій

Кафедра комп'ютерних наук

«Затверджую»

В.о. завідувача кафедри

_____ Ігор ШЕЛЕХОВ

(підпис)

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня магістра

зі спеціальності 122 - Комп'ютерних наук, освітньо-професійної програми «Інформатика»
здобувача групи ІН.м- 24 Назаренко Олексія Васильовича

1. Тема роботи: «Інформаційна технологія підбору та пошуку інформаційного контенту для емігрантів»
затверджую наказом по СумДУ від *«06» грудня 2023 року № 1412-VI*

2. Термін здачі здобувачем кваліфікаційної роботи *до 22 друдня 2023 року*

3. Вхідні дані до кваліфікаційної роботи

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)
1) Аналіз проблеми предметної області, постановка і формулювання завдань дослідження. 2) Огляд технології підбору та пошуку інформаційного контенту для емігрантів 3) Розробка технології підбору та пошуку інформаційного контенту для емігрантів 4) Аналіз результатів.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Консультанти до проекту (роботи), із зазначенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

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

Завдання прийняв до виконання _____ Керівник _____

(підпис) (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів дипломного проекту (роботи)	Термін виконання проекту (роботи)	Приміт ка
1.	*Аналіз проблеми предметної області, постановка і формулювання завдань дослідження*		
2.	*Огляд технологій та вибір програмних засобів що використовуються*		
3.	*Розробка інформаційної технології підбору та пошуку інформаційного контенту для емігрантів*		
4.	*Аналіз отриманих результатів*		
5.	*Оформлення пояснювальної записки до кваліфікаційної роботи*		

Здобувач вищої освіти _____
(підпис)

Керівник проекту _____
(підпис)

АНОТАЦІЯ

Записка: 64 стор., 20 рис., 3 додатків, 20 джерел.

Об'єкт дослідження — процес підбору та пошуку інформаційного контенту для емігрантів з України.

Предмет дослідження — моделі та засоби інформаційної технології підбору та пошуку інформаційного контенту для емігрантів

Мета роботи — розробка та впровадження інформаційної технології, яка здатна забезпечити доступ користувачів до інформації щодо умов еміграції, документів, трудових можливостей, освітніх закладів та інших питань.

Результати — розроблений і впроваджений чат-бот Telegram на основі мови Python. Система збору та обробки основних даних користувачів для ефективної взаємодії. Впроваджена генерація промптів для chatGPT для надання коректних та більш точних відповідей. Представлено функціонал для вибору країни еміграції та доступ до різних категорій питань.

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ, TELEGRAM BOT API, CHAT GPT, ІНФОРМАЦІЯ ДЛЯ ЕМІГРАНТІВ

ЗМІСТ

ВСТУП

В епоху стрімкого технологічного розвитку та глобалізації, еміграція стає неодмінною частиною життя багатьох громадян України, які вирушають у світ у пошуках нових можливостей, безпеки чи просто для реалізації своїх амбіцій. За останні роки, а саме 2022-2023 р., цей процес набуває особливої актуальності та важливості, обумовленої рядом глобальних подій, одним з яких є широкомасштабні воєнні дії на території України.

Вторгнення Росії в Україну стало, мабуть, визначальною гео-політичною подією 2022 року, яка змусила мільйони людей емігрувати по всьому світу. Але це був лише один із багатьох факторів, які допомогли збільшити кількість переміщених людей у всьому світі до рекордно високого рівня, поряд із відсутністю продовольчої безпеки, природними катаклізмами, пов’язаними зі зміною клімату, і затяжними конфліктами. Тим часом зміна попиту на робочу силу та зміна економічних перспектив після пандемії COVID-19 спонукали багатьох мігрантів до нових місць призначення.

За цих умов, інформаційні технології стають важливим інструментом для сприяння успішній адаптації емігрантів у новому оточенні. Інформаційна технологія підбору та пошуку інформаційного контенту для емігрантів стає ключовою складовою цього процесу, забезпечуючи доступ до необхідної інформації та сприяючи швидкій інтеграції в нове соціокультурне середовище.

З моменту вирушення українців у світ, виникає проблема з орієнтування у великому потоці інформації. Постає безліч питань:

● Як знайти найбільш актуальні та корисні дані про житло, роботу, освіту, правові аспекти, важливі адреси та контакти, мовні курси, гуманітарну чи психологічну допомогу, та інші питання, що стосуються нового життя?

● Як зробити цей процес максимально ефективним та приємним для емігрантів?

І саме на ці питання спрямована дана робота, яка розглядатиме

інформаційні технології, призначені для полегшення пошуку та підбору інформаційного контенту для емігрантів з України.

Новизна та основна перевага даної інформаційної технології полягатиме у інтеграції з державними сайтами та заздалегідь визначеними офіційними джерелами країн, що приймають мігрантів та інтеграція з ChatGPT.

Тому ця робота присвячена визначенню актуальних потреб емігрантів, аналізу сучасного стану інформаційного середовища для них, а також розробці та впровадженню інформаційної технології, що врахує всі аспекти цього процесу. Надаючи інструменти для ефективної взаємодії з інформаційним простором, ці технології сприятимуть швидкій адаптації емігрантів, допомагаючи їм знаходити необхідну інформацію вчасно та ефективно.

1 ІНФОРМАЦІЙНИЙ ОГЛЯД

1.1 Дослідження предметної області

Перед розробкою інформаційної системи необхідно визначити цільову аудиторію, провести детальний аналіз актуальних потреб емігрантів, проблем з якими стикаються та країн їх перебування, провести аналіз сучасного стану інформаційного середовища для них.

Станом на червень 2023 року за даними українського МЗС за кордоном перебуває 8 млн 177 тис. громадян України, тобто близько 20% населення нашої держави до 24 лютого 2022 року [1].

З даними УВКБ ООН, станом на початок квітня 2023 року у Польщі було зареєстровано 1,58 млн. українців. На другому місці за кількістю українських біженців – Німеччина (923 тис.).

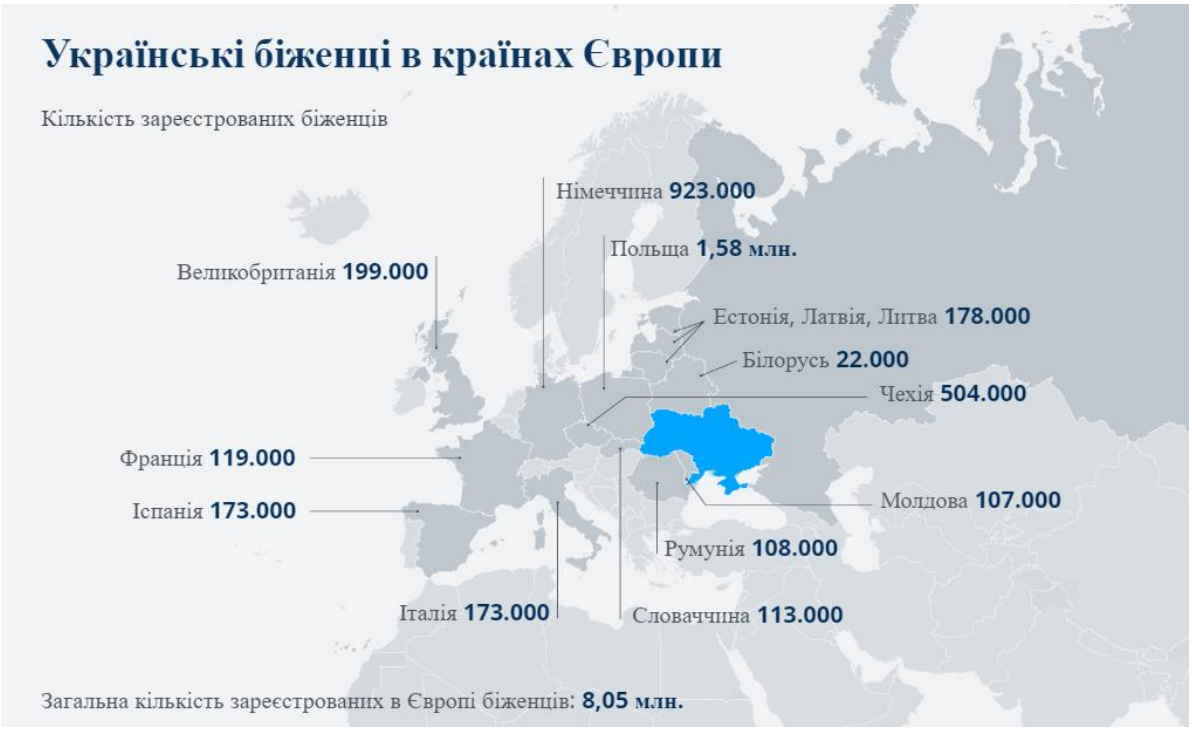

Рисунок 1.1 — Карта розміщення біженців українців у Європі

На третьому місці за кількістю українських біженців – Чехія, де зареєстровано трохи більше ніж 455 тис. наших громадян. До десятки країн, в які від війни поїхали найбільше українців, також увійшли: Італія (171,5 тис.),

Іспанія (150,4 тис.), Туреччина (145 тис.), Британія (141,5 тис.), Франція (118, 9 тис.), Словаччина (99,3 тис.) та Молдова (95,4 тис.).

Менше 10 тисяч українських біженців зареєструвалися у Словенії, Люксембурзі, Північній Македонії, Азербайджані, Албанії, Ісландії та Мальті.

Переважна більшість біженців – це жінки (найбільшою є частка жінок у віці 35–49 років – 18%) та діти [2].

За допомогою кластерного аналізу експерти виокремили чотири групи біженців.

Перша група (25% від усіх біженців) – це класичні біженці: переважно жінки середнього віку з дітьми, які виїхали до Польщі. Вони не є дуже адаптованими до життя за кордоном, оскільки 41% українців з цієї групи ніколи раніше не були за кордоном. До того ж вони переважно жили в населених пунктах, які перебували поза зоною бойових дій, але потерпали від ракетних обстрілів. Тож основною причиною виїзду за кордон були побоювання за власну безпеку [3].

Друга група (29% від усіх біженців) – це квазі трудові мігранти, які виїхали за кордон не тільки через війну, а й для роботи. Вони є найбільш адаптованими до життя за кордоном, скільки 25% людей цієї групи вже мали досвід роботи за кордоном.

Третя група (29% від усіх біженців) – це професіонали, особи, які переважно працюють за спеціальністю та менше готові працювати не за нею. Також, до війни вони часто мали власний бізнес. Ця група є більш лояльною до України та частіше, ніж інші, планує повернутися в Україну.

Четверта група (16% від усіх біженців) – це люди із зони бойових дій, українці, які найбільше постраждали від війни. Люди з цього сегменту найбільш готові робити кроки для адаптації за кордоном. Водночас вони також висловлюють найбільшу готовність до повернення в інший регіон України, якщо повернутися до їхнього рідного буде неможливо. Їхнє рішення щодо повернення залежатиме від створених умов для цього.

У багатьох країнах були проведені дослідження як інтегрувалися українці та з якими труднощами стикаються.

Наприклад, 77% українців зазначили, що стикаються у Литві із проблемою довгих черг до лікарень. Опитані акцентували увагу на проблемах із доступом до курсів мови, лікування, а також на певному відчуженні з боку місцевого населення та відчуття соціального ризику. Майже 90% українців планують вивчати литовську мову і більше половини мотивовані робити це на знак подяки за те, що країна їх прийняла [4].

Міжнародний комітет порятунку (IRC), який стурбований можливою відмовою у допомозі українцям з боку Польщі у небезпечний зимовий період, назвав основні проблеми, на які скаржаться біженці. «Серед українців, опитаних IRC, найбільші виклики інклюзії пов'язані з проблемами з житлом, що збільшує ризик передчасного повернення (до України). Ми також визначили напруженість всередині самої української громади, питання, пов'язані зі статусом біженця та експлуатацією на ринку праці, як сфери, що викликають значне занепокоєння», — зазначив директор IRC у Польщі Алан Мозлі. Для аналізу ризиків і тенденцій, пов'язаних із захистом і допомогою, що надаються українським біженцям у Польщі, IRC вивів основні проблеми: житлове питання; напруга в українській спільноті, спричинена тривалим стресом і відсутністю стабільності (щодо доходу, зайнятості, житла), особливо в місцях колективного розміщення біженців; робота в Польщі (експлуатація на ринку праці й дискримінація) [5]

Знання мови та рівень інтеграції багато в чому визначають шанси на ринку праці в Німеччині. Проте рівень зайнятості українських біженців порівняно з кінцем літа 2022 року лише незначно зріс: цієї весни 18% осіб віком від 18 до 64 років були зайняті, наприкінці літа 2022 року – 17%. Оскільки під час другого опитування близько двох третин опитаних все ще відвідували курси німецької мови, вони були лише обмеженою мірою або ж взагалі не представлені на ринку праці. Більше двох третин безробітних

сказали, що хотіли б зайнятися пошуком роботи негайно або протягом наступного року. «Зокрема жінки-біженки з маленькими дітьми мають дуже низький рівень зайнятості – 3%», – зазначила Юлія Косякова з Інституту досліджень ринку праці та професійної підготовки (IAB) у Нюрнберзі. «Зазвичай вони проживають у Німеччині без партнера. Водночас рівень зайнятості чоловіків з маленькими дітьми значно вищий і становить 23%, більшість з них живе у Німеччині з партнеркою». «Для великої групи українських біженців у Німеччині важливо, щоби було достатньо місць у дитячих садочках», – наголошує Андреас Етте (Andreas Ette) з Федерального інституту досліджень народонаселення у Вісбадені. «Це важливо для того, щоб батьки могли відвідувати мовні курси та влаштовуватися на роботу, а діти вивчали мову, мали структуровані будні та знаходили друзів». Дослідники висунули наступні пропозиції: «Насамперед оперативно продовжити право на проживання біженців після березня 2024 року або знайти рівну альтернативу; інвестиції в соціальну активність та працевлаштування одночасно вимагають довгострокового планування та правової визначеності, а також і надійних перспектив на право проживання» [6].

За даними МОН, в інших країнах зараз перебувають 672 тис. українських школярів. Найбільше таких дітей у Польщі, Німеччині, Чехії, Молдові, Угорщині, Словаччині, Румунії. Зокрема, у Польщі проживає 200 тисяч українських школярів, з яких 75 тисяч навчаються в місцевих школах. Загалом учні разом із батьками виїхали до 44 європейських країн, а також до США, Канади, Австралії, Японії. Утім, точної статистики щодо школярів за кордоном немає.

Аналітичний центр CEDOS провів дослідження проблем, з якими стикаються за кордоном українські школярі-біженці. На основі цих даних фахівці дали свої пропозиції.

Проблеми українських школярів [7]:

1. Мовний бар’єр. Є країни, де в школах можна обрати англомовні

програми, і це стає виходом для частини українських дітей. Але якщо навіть вони добре знають англійську та успішно навчаються, незнання державної мови країни перебування може заважати їм інтегруватися. Вирішення: державам доведеться подбати про вивчення місцевої мови дітьми-біженцями.

2. Академічна різниця. Освітні програми та стандарти можуть відрізнятися в Україні та країнах, куди потрапили наші школярі. Вирішення: спочатку провести оцінку освітніх втрат та академічної різниці. Варто створити можливості (курси, додаткові роки навчання) для того, щоб діти з України мали змогу наздогнати однолітків з країни перебування.

3. Дефіцит підготовлених до роботи з біженцями вчителів. Не всі країни мають достатній досвід інтеграції іншомовних дітей у систему освіти. Там, де найбільше біженців з України, інколи не вистачає педагогів. Окрім того, не всі вчителі вміють навчати дітей, які спілкуються іншою мовою, або викладати державну мову як іноземну. Вирішення: підвищувати відповідну кваліфікацію педагогів. Міністерства освіти Франції, Австрії, Чехії, Данії, Німеччини, Греції надають освітянам матеріали, як подолати мовний бар'єр, як розмовляти з дітьми про війну та як надавати психологічну підтримку. У Словаччині на цю тему для вчителів провели ще й вебінари та подкасти.

Одними з найкращих, на сьогодні, рішень в галузі штучного інтелекту, які здатні вирішувати широкий спектр задач є GPT (Generative Pre-trained Transformer) — генеративні передбачені трансформери, є сімейством моделей нейронних мереж, що використовують архітектуру трансформерів і є ключовим досягненням у галузі штучного інтелекту, за допомогою якого працюють генеративні додатки, такі як Chat GPT. Моделі GPT дають додаткам можливість генерувати текст і контент (зображення, музику та багато іншого), схожий на створений людиною, та відповідати на запитання у розмовній манері. Організації з різних галузей використовують моделі GPT та генеративний штучний інтелект для роботів питань та відповідей, короткого викладу тексту, генерації контенту та пошуку.

Chat GPT prompt – це фраза або вказівка, яку ви надаєте моделі Chat GPT AI для створення відповіді. Підказками можуть бути будь-які запитання, твердження чи інші слова, призначені для стимулювання творчості, «думок» або залучення.

Розробка підказок Chat GPT — це практика створення підказок, які ефективно інструктують Chat GPT генерувати результати. Щоб правильно керувати відповідями Chat GPT, потрібно зрозуміти його поведінку та відповідно скоригувати свої формулювання.

Створюючи найкращі підказки ChatGPT, необхідно зосередитись на таких принципах [8]:

1. Чіткість: підказки Chat GPT мають бути однозначними та точними. Уникайте розпливчастих слів, щоб забезпечити найкращі результати.

2. Простота: робіть підказки короткими, але потужними. Від одного до трьох речень – ідеальна довжина підказки.

3. Контекст: додайте якомога більше контекстної інформації. Наприклад, якщо ваша підказка пов’язана з роботою, вкажіть галузь, посаду та конкретне завдання чи мету.

4. Конкретність: додайте деталі та специфіку, щоб персоналізувати результат.

5. Рольові ігри: використовуйте методи рольових ігор, щоб встановити діапазон відповідей моделі AI. Наприклад, ви можете сказати: «Уявіть, що ви інженер Microsoft, а я — співбесіда на посаду молодшого інженера-програміста. Створіть три запитання, які ви можете поставити під час співбесіди зі мною».

Реалізація інформаційної технології підбору та пошуку інформаційного контенту для емігрантів з України може мати різні форми в залежності від потреб цільової аудиторії, специфіки завдань та зручності використання. Ось декілька можливих форм:

- Мобільний додаток

- Веб-додаток
- Чат-бот
- Локальне програмне забезпечення

Що таке чат-бот?

Чат-бот – це комп’ютерна програма, яка імітує розмову людини з кінцевим користувачем. Хоча не всі чат-боти оснащені штучним інтелектом (ШІ), сучасні чат-боти все частіше використовують розмовні методи ШІ, такі як обробка природної мови (NLP), щоб зрозуміти запитання користувача та автоматизувати відповіді на них [9].

Значення чат-ботів.

Чат-боти можуть полегшити користувачам пошук інформації, миттєво відповідаючи на запитання та запити — за допомогою введення тексту, введення аудіо або обох — без потреби втручання людини чи ручного дослідження.

Технологія чат-ботів тепер є звичайною справою, її можна знайти скрізь: від розумних динаміків удома до SMS-повідомлень, WhatsApp і Facebook Messenger для користувачів та ін. Остання еволюція чат-ботів штучного інтелекту, які часто називають «розумними віртуальними помічниками» або «віртуальними агентами», може не тільки розуміти вільну розмову за допомогою складних мовних моделей, але навіть автоматизувати відповідні завдання. Поряд із добре відомими інтелектуальними віртуальними помічниками, орієнтованими на споживачів, такими як Apple Siri та Amazon Alexa, віртуальні агенти також все частіше використовуються в корпоративному контексті для допомоги клієнтам і співробітникам.

Найкраще для реалізації інформаційної технології підбору та пошуку інформаційного контенту для емігрантів підійде чат-бот з багатьох причин:

- Доступність 24/7. Чат-боти можуть бути доступні цілодобово, що особливо важливо для емігрантів, які можуть знаходитися у різних часових

зонах. Це дозволяє їм отримувати необхідну інформацію у зручний час для них.

● Миттєве реагування. Чат-бот може миттєво відповісти на тисячі запитань клієнтів, це допомагає покращити середній час відповіді.

● Послідовні відповіді. Чат-боти допомагають компаніям підтримувати високий рівень узгодженості у своїх відповідях, що ще більше сприяє покращенню взаємодії клієнтів із брендом.

● Багатоканальний. Чат-боти на основі штучного інтелекту включають функцію підтримки багатоканального обміну повідомленнями, що допомагає клієнтам взаємодіяти з компаніями через різні канали, такі як Facebook, веб-сайти тощо [10].

● Багатомовність. Чат-боти підтримують кілька мов, ця функція особливо корисна для світових брендів. Ви можете запрограмувати чат-бота, щоб відповідати на запити мовою, яку віддають перевагу клієнти.

● Персоналізація. Чат-боти можуть надати більш персоналізований досвід, ніж професіонали з обслуговування клієнтів, залучаючи клієнтів до розмов один на один. Чат-бот може дізнатися історію вашої взаємодії з компанією, щоб забезпечити більш персоналізований досвід.

● Адаптованість до різних платформ. Чат-боти можуть бути інтегровані в різні месенджери та платформи, що дозволяє емігрантам використовувати їх на будь-якому пристрої за власним вибором. Це підтримує гнучкість та доступність.

● Зручність для різних вікових груп. Емігранти варіюються за віком, але багато з них, особливо молодші покоління, вже звикли до використання чат-подібних інтерфейсів. Це робить чат-боти природним та зручним інструментом для взаємодії з цільовою аудиторією.

Месенджери надійно вкоренилися в нашому повсякденному житті, ставши невід'ємною частиною засобів комунікації як в особистій, так і в професійній сфері. У 2023 році спостерігається подальше зростання популярності месенджерів, і на ринку виникають нові цікаві рішення [11].

За даними компанії Statista, у 2023 році найпопулярнішими месенджерами у світі є:

1. WhatsApp (2 млрд користувачів)
2. Facebook Messenger (1,3 млрд користувачів)
3. WeChat (1,2 млрд користувачів)
4. QQ (648 млн користувачів)
5. Telegram (500 млн користувачів)

Ці месенджери є лідерами на ринку вже протягом кількох років, завдяки широкому функціоналу та зручному інтерфейсу, що робить їх популярними серед користувачів по всьому світу.

WhatsApp, який наразі є найбільш популярним месенджером у світі, пропонує обмін повідомленнями, дзвінки, відеодзвінки, групові чати, обмін файлами та інші функції. Його доступність на всіх основних платформах робить його універсальним інструментом для спілкування.

Facebook Messenger, який інтегрований у соціальну мережу Facebook, надає ті ж самі функції, що і WhatsApp, а також можливість взаємодії з користувачами Facebook, які не використовують Messenger окремо.

WeChat є лідером серед месенджерів у Китаї, пропонуючи широкий функціонал, такий як обмін повідомленнями, дзвінки, відеодзвінки, групові чати, обмін файлами та платежі.

QQ, що є другим за популярністю месенджером у Китаї, пропонує аналогічний функціонал до WeChat.

Telegram, визначений як один із найбільш швидкозростаючих месенджерів у світі, відзначається розширеним набором функцій, таких як

обмін повідомленнями, дзвінки, відеодзвінки, групові чати, обмін файлами, секретні чати та інше.

В Україні ситуація з популярністю месенджерів трохи відрізняється від глобальної. На рисунку 1.2 представлені найпопулярніші месенджери в Україні, згідно з даними компанії InMind, у 2023 році.

Рисунок 1.2 — Найпопулярніші месенджери в Україні в 2023р.

Viber утримує свою лідерську позицію серед месенджерів в Україні протягом декількох років. Він приваблює користувачів зручним і зрозумілим інтерфейсом, який підходить будь-якому віку, та обширним функціоналом.

Telegram, в свою чергу, є одним із найбільш стрімко зростаючих месенджерів в Україні. Він пропонує підтримку каналів з новинами що дуже вплинуло на його популярність з початком війни.

Таблиця 1.1 — Порівняння месенджерів на 2023 р.

Критерії	Telegram	WhatsApp	Viber	Facebook Messenger

Зручність інтеграції	+ Надає відкрите API для створення чат-ботів та інтеграції сторонніх додатків	– Обмежені можливості для бізнесу через WhatsApp Business API	+/– Взаємодія обмежена через Viber REST API	+/– Взаємодія через власну систему Messenger API

Продовження таблиці 1.1 — Порівняння месенджерів на 2023 р.

Система безпеки	+ Сильне шифрування повідомлень, режим конфіденційності	+ Забезпечує криптографічне шифрування, включає функцію двофакторної автентифікації	+ Прийнятна система безпеки, але може бути менш надійною порівняно з іншими	– Шифрування присутнє, але не завжди вистачає для вимог деяких користувач
Функціональні можливості	+ Широкий набір функцій, включаючи групові чати, канали, боти та інші	+ Великий функціонал, включаючи відеодзвінки, групові чати, статуси та інше	+/– Непоганий функціонал, але менший у порівнянні з Telegram та WhatsApp	+ Великий функціонал, але часто вимагає встановлення додатків для розширення можливостей
Мовна підтримка	+ Підтримує багато мов, глобальна аудиторія	+ Має розширений перелік мов, але може бути менш різноманітним порівняно з Telegram	+ Забезпечує підтримку декількох мов, але не такий широкий як у Telegram та WhatsApp	+ Підтримка багатьох мов, з фокусом на англійській та основних європейських мовах
Інтерактивність	+ Велика кількість інтерактивних елементів, таких як кнопки та відповіді, підтримка	+ Забезпечує можливості відправлення фото, відео, голосових повідомлень та стікерів	+ Включає інтерактивні елементи, але обмежується в порівнянні з Telegram	– Обмежені можливості для інтерактивних елементів та ботів

	ботів та каналів			

Продовження таблиці 1.1 — Порівняння месенджерів на 2023 р.

Зручність використання	+ Інтуїтивний інтерфейс та висока гнучкість в управлінні чатами та групами	+ Простий та доступний інтерфейс, зручне додавання контактів та використання стандартних функцій	+ Зрозумілий інтерфейс, але менш гнучкий в порівнянні з Telegram і WhatsApp	+ Зручний дизайн, але інтерфейс може здаватися перевантаженим для новачків

Найкраще для реалізації інформаційної технології підбору та пошуку інформаційного контенту для емігрантів підійде чат-бот Telegram з наступних причин:

● Широка популярність Telegram: Telegram визначається великою кількістю користувачів та активною спільнотою. За умови такої популярності, чат-бот в Telegram отримує велику аудиторію серед емігрантів, забезпечуючи максимальний охоплення цільової аудиторії.

● Зручність та легкість використання: чат-бот в Telegram забезпечує зручний та інтуїтивно зрозумілий інтерфейс, який є особливо важливим для емігрантів, які можуть бути не обізнані з складними технічними рішеннями.

● Інтерактивність: чат-бот в Telegram може використовувати різноманітні інтерактивні елементи, такі як кнопки, швидкі відповіді та багато іншого. Це полегшує взаємодію користувачів з ботом, що важливо для ефективного отримання необхідної інформації.

● Широкі функціональні можливості: Telegram дозволяє інтегрувати

різноманітні функції у чат-бот, такі як відправка повідомлень, зображень, документів, аудіо та відео. Це дозволяє створювати багатогранний та інформативний інтерфейс для користувачів.

● Система безпеки: Telegram використовує ефективну систему шифрування, що гарантує конфіденційність інформації користувачів. Для емігрантів, які можуть шукати конфіденційну або чутливу інформацію, це є важливим аспектом.

● Підтримка мовних функцій: Telegram підтримує багато мов, що дозволяє адаптувати чат-бот до різних мовних спільнот та полегшує спілкування для емігрантів із різних країн.

Отже, розробка чат-боту має великий потенціал для надання ефективної та зручної інформаційної підтримки емігрантам. Вона враховує їхні індивідуальні потреби, надаючи персоналізовані відповіді та створюючи ефективний механізм обміну необхідною інформацією.

1.2 Огляд аналогів розроблюваного програмного продукту

МЗС України спільно з Institute of Cognitive Modeling та командою Romanuke створили сайт UA HELP (https://uahelp.today/) (рис. 1.3), завдяки якому громадяни України, які через війну були вимушені або ще планують виїхати до Польщі, можуть отримувати усю необхідну інформацію про виїзд, документи, безкоштовну допомогу. За допомогою нового веб-сайту, громадяни України матимуть можливість отримати актуальну інформацію щодо:

● Правил перетину кордону для дорослих та дітей

● Умов перебування в Польщі

● Процедур отримання документів та відповідного статусу — біженця або тимчасового захисту

Сайт надає можливість:

- Знаходження роботи
- Отримання роз’яснень щодо медичної та фінансової допомоги
- Пошук дитячих садочків і шкіл для дітей

На веб-сайті та в чат-боті також доступні всі необхідні контакти гарячих ліній, центрів допомоги та фондів, які забезпечують підтримку українцям у Польщі.

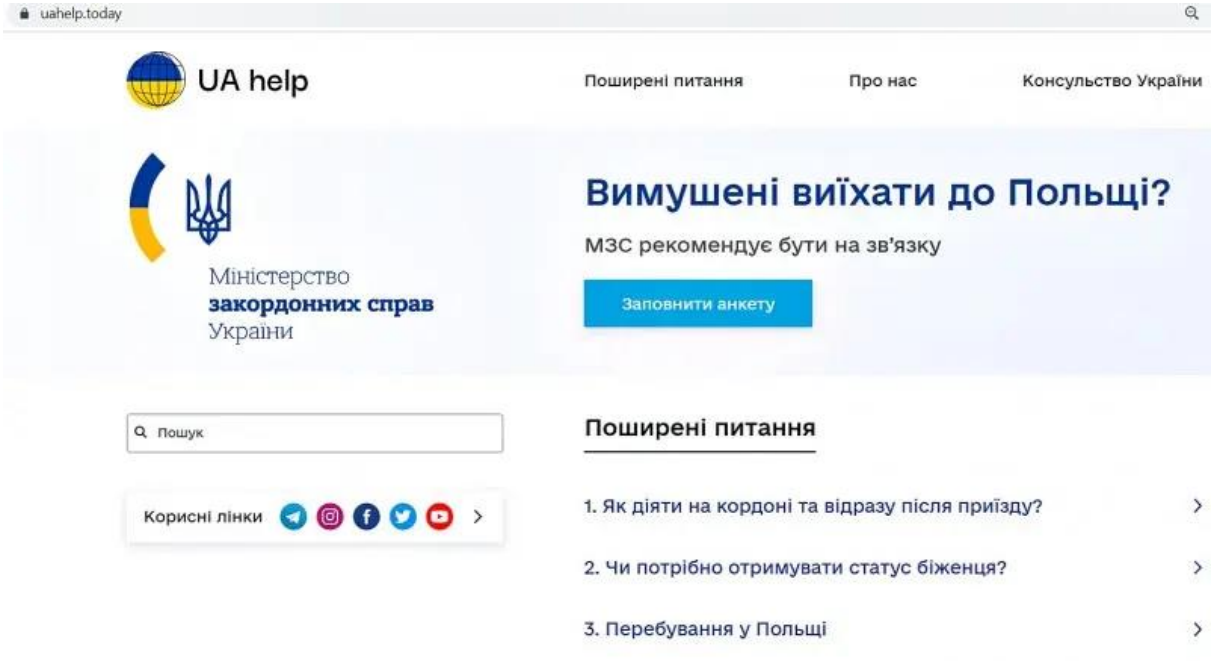

Рисунок 1.3 — Головна сторінка uahelp.today

Державний чеський сайт (https://pomahej-ukrajine.cz/) по допомозі Україні місцем для посередництва пропозицій допомоги громадян біженцям з України та організацій, що допомагають (рис. 1.4). Там можна розміщувати свою пропозицію допомоги, а також можна знайти допомогу за наступними категоріями:

- Матеріальна допомога
- Переклад документів
- Транспорт

- Вільне нежитлове приміщення
- Юридична допомога
- Психологічна допомога та терапія
- Волонтерська допомога
- Репетиторство
- У вільний час
- Курси чеської мови
- Групи адаптації
- Підручники та методичні матеріали
- Охорона здоров'я та соціальна допомога
- Ветеринарне обслуговування та догляд за тваринами

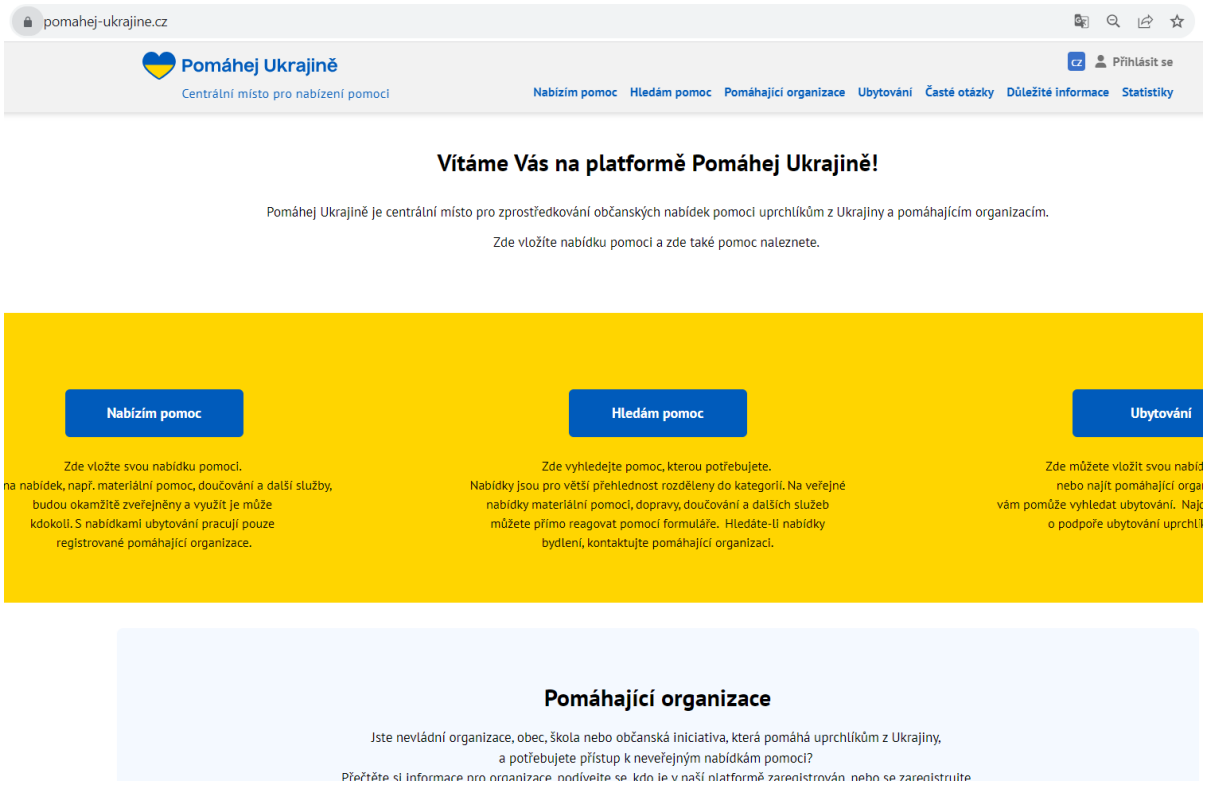

Рисунок 1.4 — Головна сторінка pomahej-ukrajine

Ukrainian Refugees UAPT (https://helpua.pt/) прагне відповісти на гуманітарну кризу, спричинену війною Росії проти України, допомагаючи біженцям з житлом, роботою, психологічною, медичною, економічною та юридичною підтримкою, а також соціальною інтеграцією (рис. 1.5).

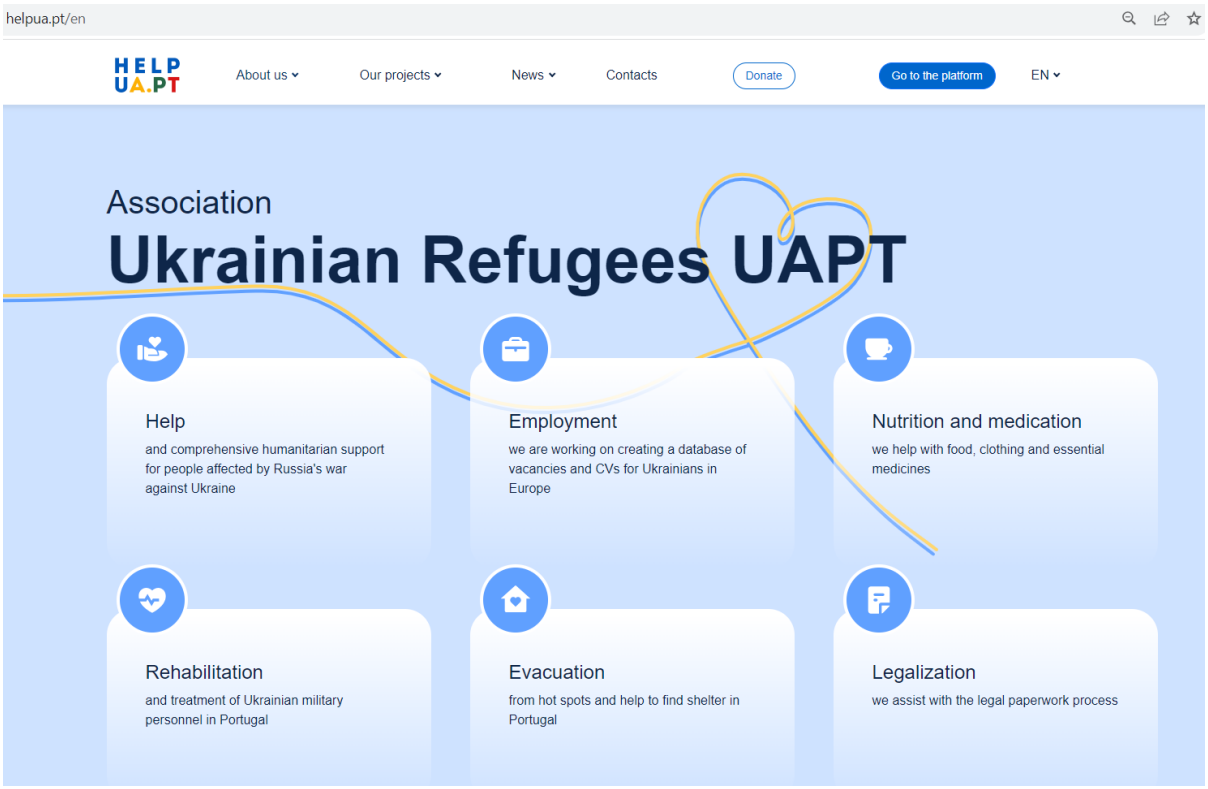

Рисунок 1.5 — Головна сторінка helpua.pt

RefAid є мобільним додатком, призначений для вразливих груп населення, включаючи переміщених осіб, мігрантів та біженців, людей, які постраждали від війн та стихійних лих, а також для волонтерів та організацій, які їм допомагають (рис. 1.6-1.7). На карті показано розташування та види допомоги, а також інформація про дні та години роботи. Вся допомога, надана в додатку, надходить від перевірених офіційних організацій з надання допомоги. Допомога класифікується за типами, включаючи: стихійні лиха та війни; Здоров'я; Їжа; Притулок; Вода; Непродовольчі товари; Юридичні питання/Адміністратор/Інформація; особлива допомога батькам та дітям,

несупроводжуваним дітям, жінкам та чоловікам; Здоров'я; Освіта; та туалети та душові.

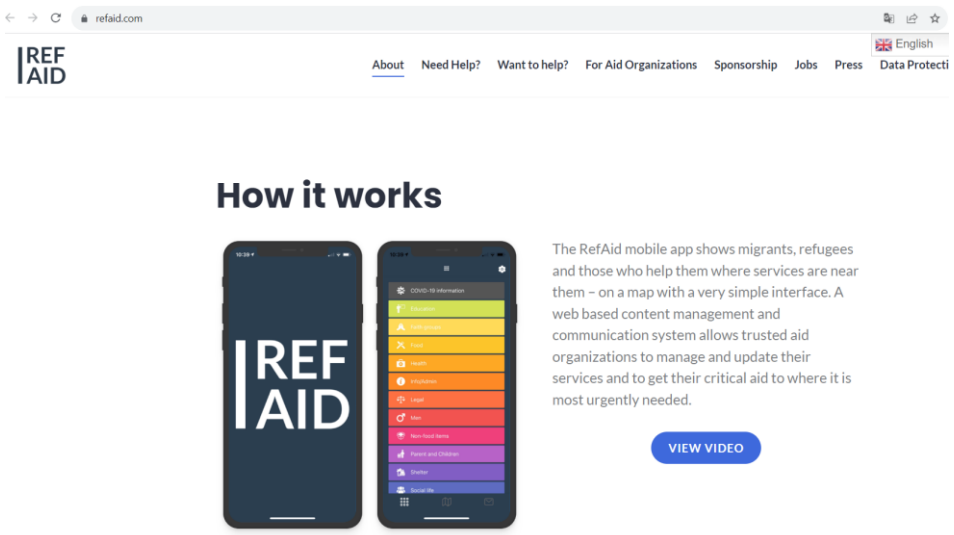

Рисунок 1.6 — Головна сторінка refaid.com

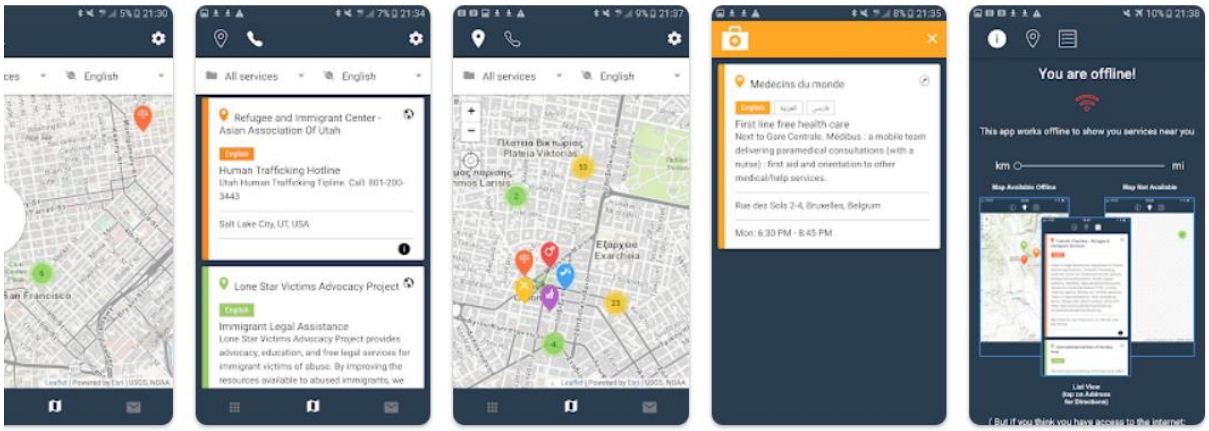

Рисунок 1.7 — Приклад використання RefAid

Refugee.Info — це онлайн-платформа, яка забезпечує біженців та переміщених осіб інформацією про доступні послуги, законодавство та місцеві ресурси в країнах, де вони перебувають (рис. 1.8).

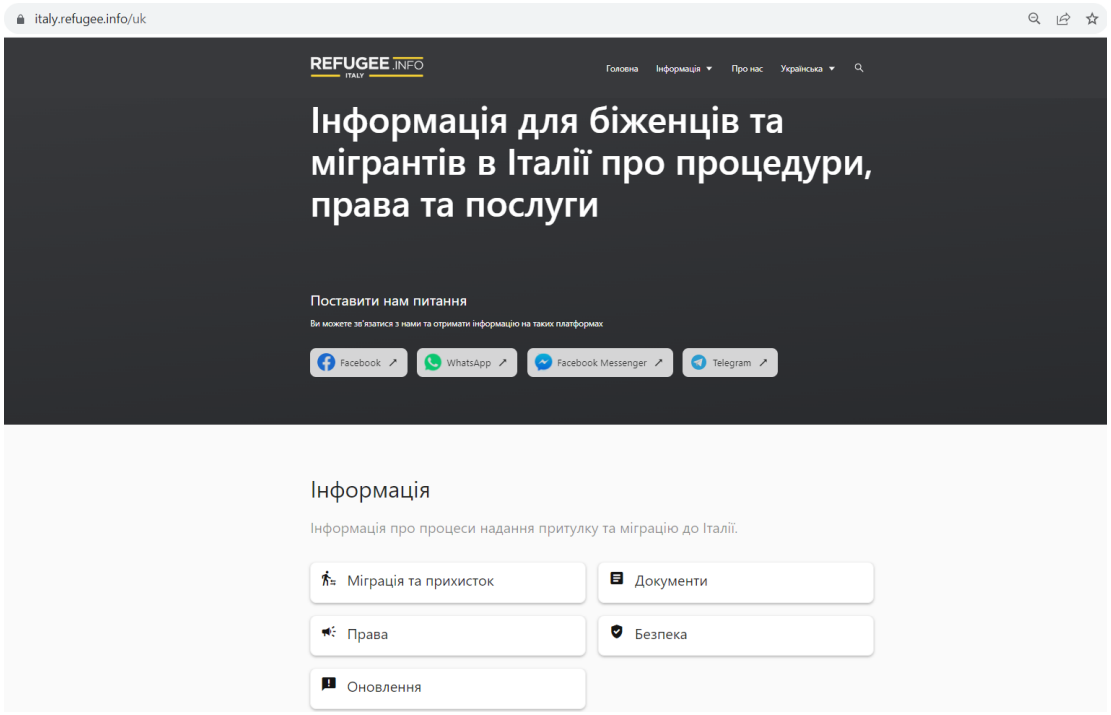

Рисунок 1.8 — Головна сторінка Refugee.Info на прикладі Італії

Недоліки Refugee.Info:

1. Оперативність оновлень. Інформація на платформі може стати застарілою через неоперативність оновлень. Тому біженці можуть отримувати неактуальну інформацію, що може вплинути на їхні рішення та плани.

2. Орієнтованість на деякі регіони. Деякі регіони можуть мати обмежену кількість інформації та ресурсів порівняно з іншими. Тому біженці в менш розроблених областях можуть зіткнутися з нестачею ресурсів та інформації.

3. Обмежена мовна підтримка. Інтерфейс та інформація можуть бути представлені обмеженою кількістю мов. Це заважає взаємодії з користувачами, які не володіють підтримуваними мовами.

Недоліки RefAid:

1. Неоднаковий рівень доступу. Рівень доступу до допомоги та ресурсів може варіюватися в залежності від регіону та партнерів.

2. Потенційні труднощі з навігацією. Інтерфейс може бути складним для користувачів з обмеженим досвідом використання технологій. Це може обмежити доступність ресурсів для тих, хто не знає, як ефективно

користуватися платформою.

3. Відсутність Персоналізації Рекомендацій. Відсутність персоналізованих рекомендацій може робити інформацію менш адаптованою до конкретних потреб користувача. Може бути складніше для користувачів знаходити точну інформацію, яка відповідає їхнім потребам.

Завантаження нових додатків може викликати дискомфорт. Це призводить до переповнення головного екрану смартфона та скорочення часу автономної роботи пристрою. Зазвичай користувачі не мають бажання встановлювати додаток, якщо вони не планують використовувати його щоденно. На сьогоднішній день багато людей віддають перевагу месенджерам та чат-ботам як зручним інструментом для вирішення своїх потреб.

Оскільки не було знайдено аналогів інформаційних систем для емігрантів у вигляді чат-бота, далі буде розглянутий бот Telegram для допомоги ВПО, що надає цілодобову безоплатну правову допомогу для українців, які постраждали від війни (рис. 1.9).

Бот надає відповіді на питання, пов’язані з:

- Воєнним станом
- Пенсією, соціальною допомогою
- Трудовими питаннями, ФОП
- Кредитами, грошима
- Виїздом за кордон
- Особистими документами
- Житлом, проживанням
- Сім’єю, освітою, захистом
- Юридичною, психологічною, медичною, ветеринарною допомогою, продуктами

Також можна переглянути актуальні новини та отримати консультацію.

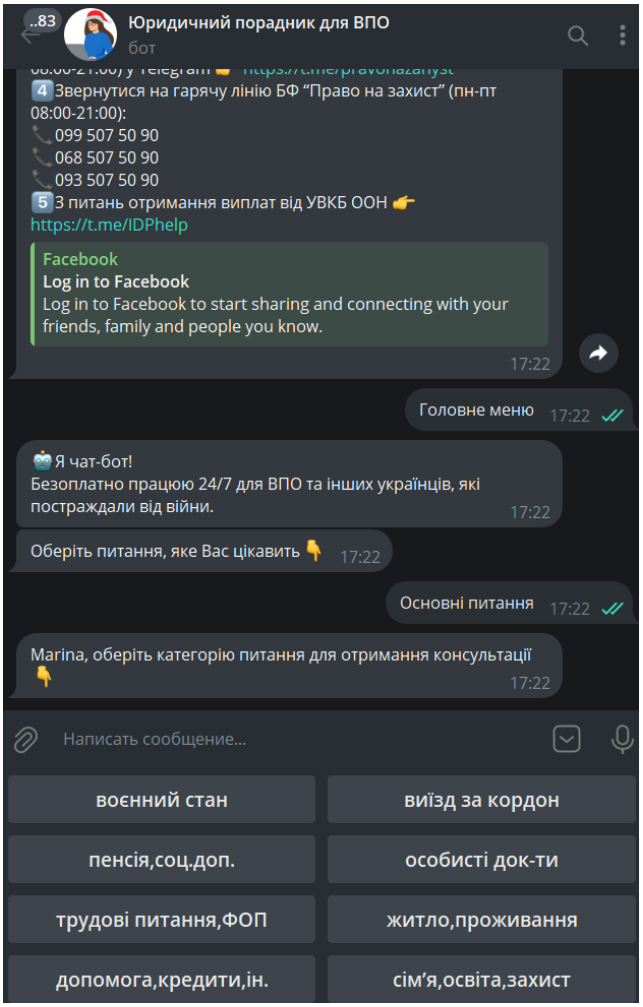

Рисунок 1.9 — Юридичний порадник для ВПО

Кожен з представлених продуктів має свої переваги, але водночас існують недоліки, які можуть впливати на їхню ефективність та доступність для користувачів.

1.3 Постановка задачі

Отже, метою роботи є розробка чат-боту Телеграм, який допоможе користувачам отримувати інформацію, що сприятиме швидкій адаптації емігрантів, допомагаючи їм знаходити необхідну інформацію вчасно та ефективно.

Функціональні вимоги:

1. Авторизація та основні налаштування

− Авторизація: користувач повинен мати можливість авторизації в системі через Telegram.

− Введення основних даних: користувач вводить основні дані: вік, стать, рід діяльності до еміграції та наявність неповнолітніх дітей.

2. Вибір країни міграції

Відображення списку країн для вибору або можливість введення власної країни.

2. Категорії та питання

− Відображення категорій: представлення користувачу категорій з питаннями про еміграцію.

− Вибір пункту категорії: можливість користувача обирати пункти категорії, які відповідають його потребам.

3. Генерація промпту для chat GPT

− Збір даних для генерації: збір введених користувачем даних та вибору пункту для генерації промпту.

− Генерація промпту: автоматичне створення промпту для chat GPT на основі обраних пунктів та введених даних користувача.

4. Запитання та відповіді

− Відправка промпту до chat GPT: передача сформованого промпту chat GPT для отримання відповіді.

− Відображення відповіді: показ отриманої відповіді в користувальницькому інтерфейсі.

6. Системні функції

Завершення сеансу: можливість користувача завершити сесію та заново вводити дані

Ці функціональні вимоги забезпечать користувачам зручний та ефективний інструмент для отримання інформації та порад щодо еміграції.

2 ВИБІР ПРОГРАМНИХ ЗАСОБІВ. ПРОЄКТУВАННЯ ТЕХНОЛОГІЇ

2.1 Вибір засобів програмування

Python є популярним інструментом для створення телеграм-ботів завдяки багатій екосистемі бібліотек для роботи з Telegram API. Один з найпопулярніших фреймворків для розробки телеграм-ботів на Python — це Telebot [12]. Його можна використовувати для створення інтерфейсу телеграм-боту, який прийматиме введені дані від користувачів та передаватиме їх до моделі аналізу та обробки даних для генерації та персоналізації промптів**.**

Згідно з офіційним веб-сайтом, Python є «інтерпретованою, об’єктно-орієнтованою мовою програмування високого рівня з динамічною семантикою». Python є найшвидше зростаючою мовою програмування у світі. Його високорівнева, інтерпретована та об’єктно-орієнтована архітектура робить його ідеальним для всіх типів програмних рішень. Більше того, наголос мови на читабельності синтаксису, модульності програми та повторному використанні коду значно збільшує швидкість розробки та зменшує вартість обслуговування [13].

Python є переважною мовою для проектів даних, проектів машинного навчання та проектів чат-ботів. Він має простий синтаксис, який легко читати та розуміти навіть розробникам-початківцям. Python також має великий вибір бібліотек для машинного навчання та обробки природної мови (NLP), включаючи потужний інструментарій Natural Language Toolkit (NLTK), який багато розробників вважають найкращою бібліотекою NLP. Facebook, Google, Dropbox, Spotify, Quora, Wikipedia, Netflix, Yahoo!, NASA та багато інших компаній використовують Python завдяки численним перевагам, які він надає.

Отже, Python є гнучким і потужним вибором для цієї задачі, особливо коли необхідно поєднати її з інтерфейсом Телеграм-боту для взаємодії з користувачами.

2.2 Аналіз та вибір алгоритмів розроблення

Що таке API Chat GPT?

ChatGPT API — це інтерфейс, наданий OpenAI, який дозволяє розробникам інтегрувати модель Chat GPT у власні програми, програмне забезпечення або платформи. Це дозволяє розробникам здійснювати виклики API до моделі Chat GPT, надаючи підказку та отримуючи відповідь, згенеровану моделлю. Використовуючи Chat GPT API, розробники можуть використовувати розмовні можливості Chat GPT, щоб покращити свої продукти, послуги чи програми за допомогою функцій обробки та генерації природної мови [14].

Як використовувати API Chat GPT?

Далі будуть наведені кроки, необхідні для впровадження API Chat GPT у власний проект на Python. Інтеграція з Python дає користувачам доступ до функцій Chat GPT, усуваючи необхідність відвідувати веб-сайт Chat GPT, щоб задавати питання.

1. Створення ключа API

Першим кроком для доступу до API Chat GPT є створення ключа API. Щоб створити ключ API, необхідно виконати такі дії [15]:

- Перейти на сторінку https://platform.openai.com/account/api-keys.
- Натиснути кнопку «Створити новий секретний ключ».

2. Встановлення бібліотеки OpenAI

Завантаження та налаштування необхідного програмного пакету для інтеграції OpenAI. Щоб використовувати API ChatGPT, потрібно встановити бібліотеку openai на Python. Можна виконати таку команду в Jupyter Notebook, щоб встановити цей пакет:

```
!pip install openai==1.3.7
```

Використання Chat GPT API.

Тепер, коли встановлена бібліотека openai і згенерований ключ API, можна використовувати API. Ось як це можна зробити у своєму проекті Python.

Імпорт необхідних бібліотек:

```
import openai
import os
import pandas as pd
import time
```

Присвоєння ключа API:

```
openai.api_key = '<YOUR API KEY>'
```

Створення функції, яку можна використовувати для отримання відповіді від Chat GPT:

```
def get_completion(prompt, model="gpt-3.5-turbo"):
messages = [{"role": "user", "content": prompt}]
response = openai.ChatCompletion.create(
model=model,
messages=messages,
temperature=0,
)
return response.choices[0].message["content"]
```

У наведеному вище коді була використана модель «gpt-3.5-turbo». У цій моделі використовується GPT 3.5, який є більш потужною версією GPT-3. Можна використати будь-яку іншу модель на вибір. Щоб переглянути різні доступні моделі, відвідайте цю сторінку: https://platform.openai.com/docs/models/gpt-4

Надсилання запиту по API:

```
prompt = "<YOUR QUERY>"
response = get_completion(prompt)
print(response)
```

Скільки коштує Chat GPT API?

У дописі в блозі від березня 2023 року OpenAI оголосив про значне зниження витрат на ChatGPT. Завдяки оптимізації системи їм вдалося знизити вартість на 90% порівняно з груднем 2022 року. Нещодавно випущена модель gpt-3.5-turbo, спеціально розроблена для діалогу, відображає це зниження витрат. З 0,002 долари за 1000 токенів модель gpt-3.5-turbo в 10 разів дешевша за оригінальну модель text-davinci-003 [16].

Ця економія коштів робить розробникам доступнішим інтегрувати API Chat GPT у свої програми навіть з обмеженим бюджетом. Це приваблива можливість дослідити вражаючі можливості Chat GPT.

Використовувати Chat GPT API легко та просто. Дотримуючись кроків, описаних вище, можна отримати доступ до потужності Chat GPT зі своїх сценаріїв Python і зробити своїх чат-ботів розумнішими та привабливими.

Порівняно з іншими програмами для обміну повідомленнями, такими як WhatsApp, Signal та подібними програмами, API та інтерфейс користувача Telegram є кращими. Кожен може створити бота, просто використовуючи бота Bot Father, який забезпечує повний контроль над створеними користувачем ботами.

Далі розглянемо бібліотеки Python, які потужно спрощують створення ботів:

1. python-telegram-bot — це потужна та зручна бібліотека для розробки телеграм-ботів на мові програмування Python. Вона надає простий та ефективний інтерфейс для взаємодії з Telegram API, дозволяючи швидко створювати різноманітні боти з різними функціональностями [17].

Її легко встановити:

```
pip install python-telegram-bot --upgrade
```

Ключові особливості:

- Підтримка тривалого опитування та веб-хуків для оновлень.
- Контекстні обробники зворотного виклику, що полегшує обробку

оновлень від користувачів.

● Надає інструменти для створення користувацьких клавіатур і вбудованих відповідей на запити.

● Робота з асинхронністю: підтримка асинхронних запитів, що робить бібліотеку ефективною для великих та швидких ботів.

Ця бібліотека також містить розширені функції, такі як обробка завантажень і завантажень файлів, які можуть додати розширені функції вашому Telegram-боту.

Широка підтримка спільноти та велика кількість документації роблять його найкращим вибором як для початківців, так і для досвідчених розробників.

2. Telethon — це бібліотека Python 3 для API Telegram. На відміну від python-telegram-bot, який призначений виключно для створення ботів, Telethon корисний для повноцінних клієнтських програм Telegram. Це означає, що окрім створення ботів, він дозволяє розробникам керувати каналами, надсилати та отримувати повідомлення та виконувати інші завдання [18].

Її легко встановити:

```
pip3 install telethon
```

Ключові особливості телемарафону:

● Доступ до нативних функцій Telegram API, що дозволяє легко настроїти ботів.

● Автоматична обробка повторного підключення та тайм-аути очікування.

● Архітектура, керована подіями, зручна для створення ботів, які реагують на оновлення користувача.

● Криптографічно безпечний, автоматично шифрує надіслані та отримані дані.

Для розробників, які шукають більш просунуту бібліотеку, яка може обробляти складні програми, крім створення ботів, Telethon є чудовим вибором.

3. Aiogram — це асинхронний фреймворк для розробки ботів Telegram. Це бібліотека високого рівня, створена на основі бібліотек python-telegram-bot і Telethon. Його головна перевага полягає в обробці одночасних завдань і управлінні кількома примірниками ботів, що може значно підвищити швидкість і продуктивність бота.

Ключові особливості Aiogram:

● Повна підтримка Telegram Bot API із завантаженням файлів.

● Проміжне програмне забезпечення для створення складних сценаріїв користувача.

● Розширені фільтри повідомлень і можливість створювати власні фільтри.

● Сильна підтримка спільноти з прикладами, стартовими проектами та додатковими утилітами [19].

Aiogram найкраще підходить для досвідчених розробників, які бажають створити складних ботів Telegram із розширеною взаємодією з користувачем.

4. Pyrogram — це потужний і простий у використанні фреймворк Telegram для Python. Він дозволяє розробникам взаємодіяти з API Telegram як користувач або через обліковий запис бота.

Основні функції Pyrogram включають:

● Підтримка API користувачів і ботів, що дозволяє використовувати більш широкий спектр програм Telegram.

● Обробка сеансу, обробка помилок і автоматичне повторне підключення.

● Розумна система плагінів для кращої організації та масштабування

ваших проектів.

- Підтримує різні типи медіа (фото, аудіо, відео, голос і документи).
- Сильна підтримка спільноти та обширна документація.

Pyrogram ідеально підходить як для початківців, які хочуть легкого старту, так і для професіоналів, яким потрібно більше контролю та простоти.

5. pyTelegramBotAPI — бібліотека, яка надає оболонку Python для API Telegram Bot. Він має простий для розуміння синтаксис і підтримує як довге опитування, так і веб-хуки, що робить його дуже універсальним для розробки ботів.

Встановити його так само просто:

```
pip install pyTelegramBotAPI
```

Ключові особливості:

- Підтримка всіх типів методів і типів API Telegram Bot.
- Розширена обробка повідомлень за допомогою команд і декораторів повідомлень.
- Надає інструменти для роботи у вбудованому режимі.

6. TGramBot — унікальний і мінімальний фреймворк для створення ботів Telegram. Він частково автоматично згенерований, що гарантує, що він залишається в курсі останніх змін в API Telegram Bot. Ключові особливості:

- Асинхронна та подійно-керована архітектура.
- Підтримує обробку повідомлень на основі команд і регулярних виразів.
- Вбудована підтримка журналювання та обробки помилок.

7. OrigamiBot — бібліотека, яка забезпечує чистий та інтуїтивно зрозумілий спосіб створення ботів Telegram. Його мета — зробити розробку ботів якомога більш приємною. Ключові особливості:

- Зручний і пітонічний синтаксис.
- Підтримка вбудованих запитів, запитів зворотного виклику та дій у чаті.
- Велика документація з прикладами.

Давайте подивимося на приклад коду після запуску `pip install tgrambot`:

```
import asyncio

from tgrambot import Bot
from tgrambot.filters import Filters
from tgrambot.types import Message
from tgrambot.text import Italic


bot = Bot("token", workers=50, parse_mode='MarkdownV2')

@bot.on_message(Filters.command('start'))
async def start_bot(c: Bot, m: Message):
    await c.send_message(m.chat.id, Italic("Hola Amigo!"))

async def main():
    await bot.run()

if __name__ == '__main__':
    loop = asyncio.get_event_loop()
    loop.run_until_complete(main())
```

8. Pytgbot — це повна обгортка API для ботів Telegram, яка підходить для складніших додатків ботів. Він забезпечує легкий доступ до API бота Telegram із додатковими можливостями.

Ключові особливості:

- Широке покриття API, включно з менш використовуваними API.
- Підтримка завантаження та завантаження файлів.
- Вбудовані функції журналювання та налагодження.

9. Teleflask — це унікальний фреймворк, який поєднує можливості

Flask і бібліотеки pytgbot, щоб забезпечити потужну платформу розробки ботів. Він найкраще підходить для розробників, які знайомі з веб-фреймворком Flask.

Ключові особливості:

- Автоматична обробка оновлень за допомогою маршрутів Flask.
- Простий у використанні інтерфейс для роботи з різними типами повідомлень.
- Надає допоміжні функції для надсилання повідомлень, фотографій тощо.

2.3 Структурно-функціональне моделювання

IDEF0 (Integration Definition for Function Modeling) та IDEF1 (Integration Definition for Information Modeling) — це стандартизовані методи моделювання для аналізу та проектування функціональних та інформаційних систем. Для даної задачі ми можемо створити діаграми IDEF0 та IDEF1 для технології підбору та пошуку інформаційного контенту для емігрантів [20].

Діаграма IDEF0 дає узагальнений огляд функціональної структури системи та показує взаємозв'язки між її основними функціями та підсистемами.

Діаграми IDEF0 та IDEF1 можуть бути досить складними та обширними, тому я створю загальний опис для обох. На рисунку 2.1 наведена спрощена функціональна модель для технології підбору та пошуку інформаційного контенту.

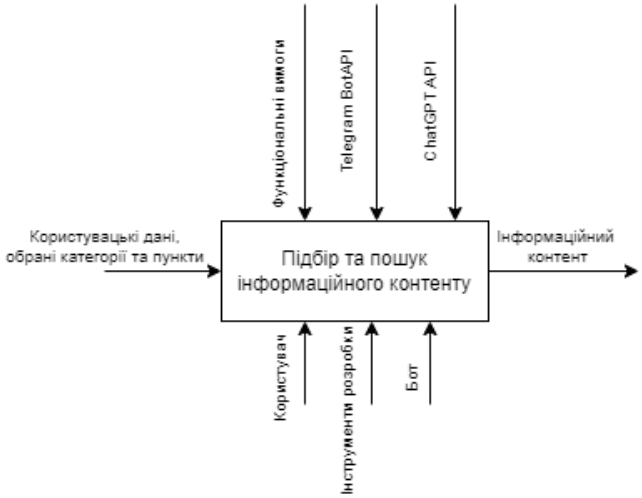

Рисунок 2.1 — IDEF0 діаграма

Діаграма IDEF1 моделює потік інформації через систему та показує, як дані обробляються та передаються між різними компонентами системи.

На діаграмі IDEF1 показано, як інформація пересилається від користувача до чат-бота та моделі для підбору та пошуку інформаційного контенту, а також як результати повертаються до користувача.

Ці діаграми IDEF0 та IDEF1 надають візуальне уявлення про функціональну та інформаційну структуру технології для підбору та пошуку інформаційного контенту та взаємодію з користувачем (див рис. 2.2).

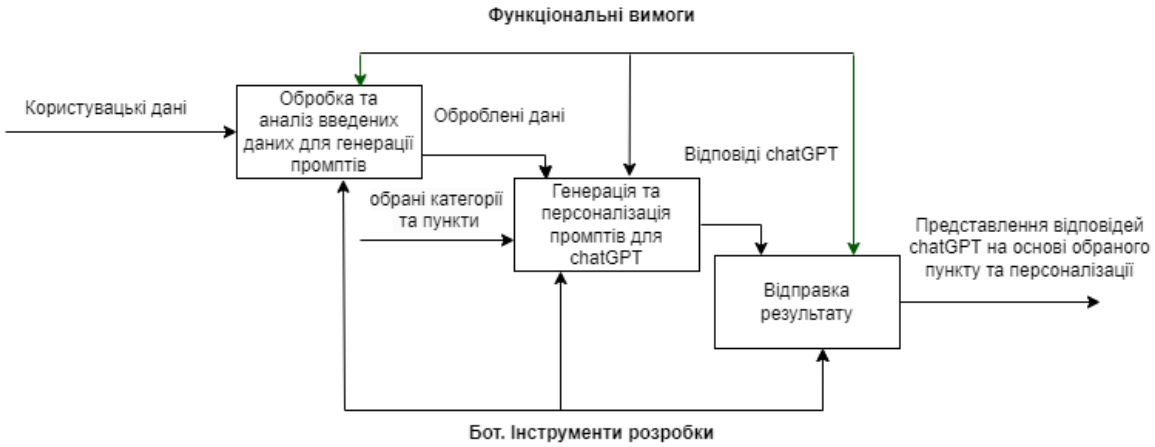

Рисунок 2.2 — IDEF1 діаграма

3 РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ

3.1 Реалізація Telegram-бота

Для простоти інтегрування і взаємодії з користувачем, підбір та пошук інформаційного контенту буде відбуватися шляхом взаємодії з Телеграм-ботом. Ось загальний план для цього:

1. Створення Telegram-бота:

Почнемо зі створення Telegram-бота, використовуючи Telegram @BotFather (рис. 3.1). Потім потрібно отримати доступ до API, використовуючи команду «/newbot». Це дозволить зареєструвати нового бота та визначити його ім'я та ідентифікатор username (рис. 3.2).

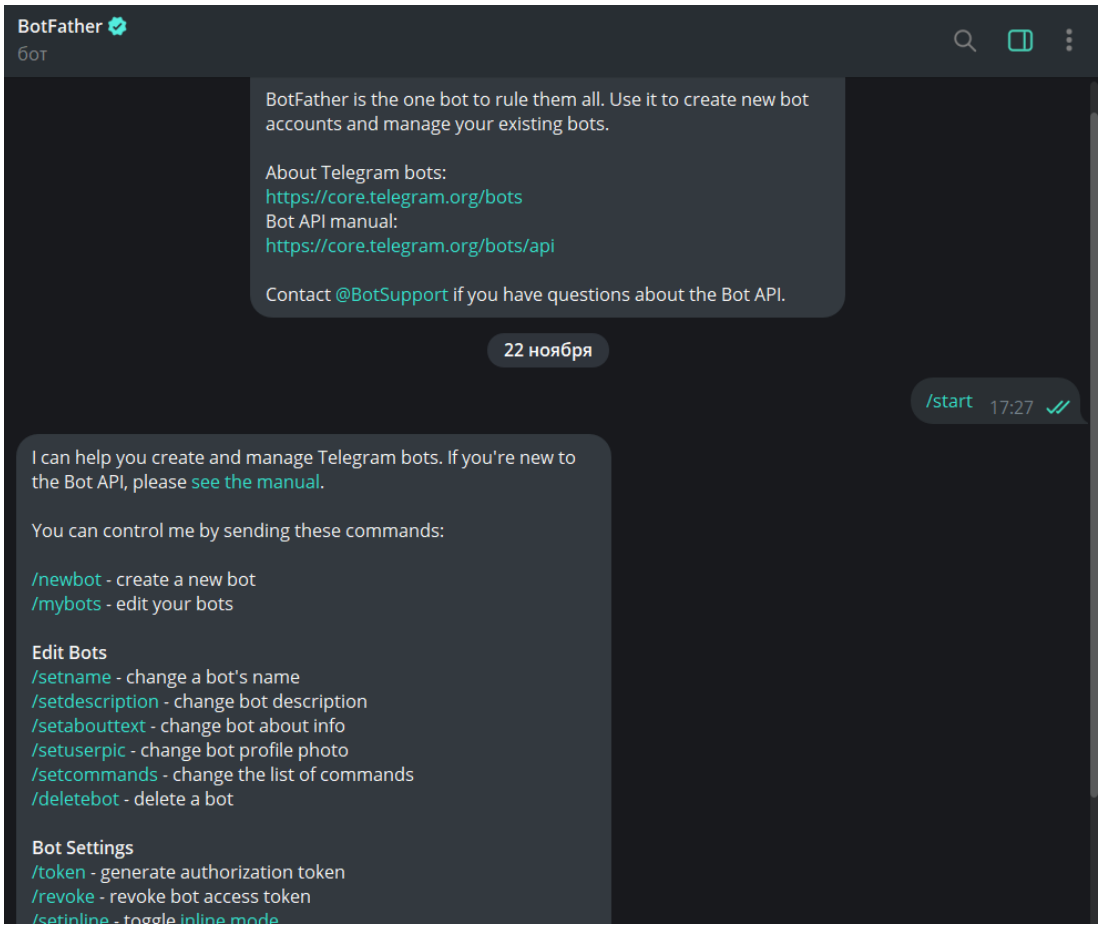

Рисунок 3.1 — Початок реєстрації бота

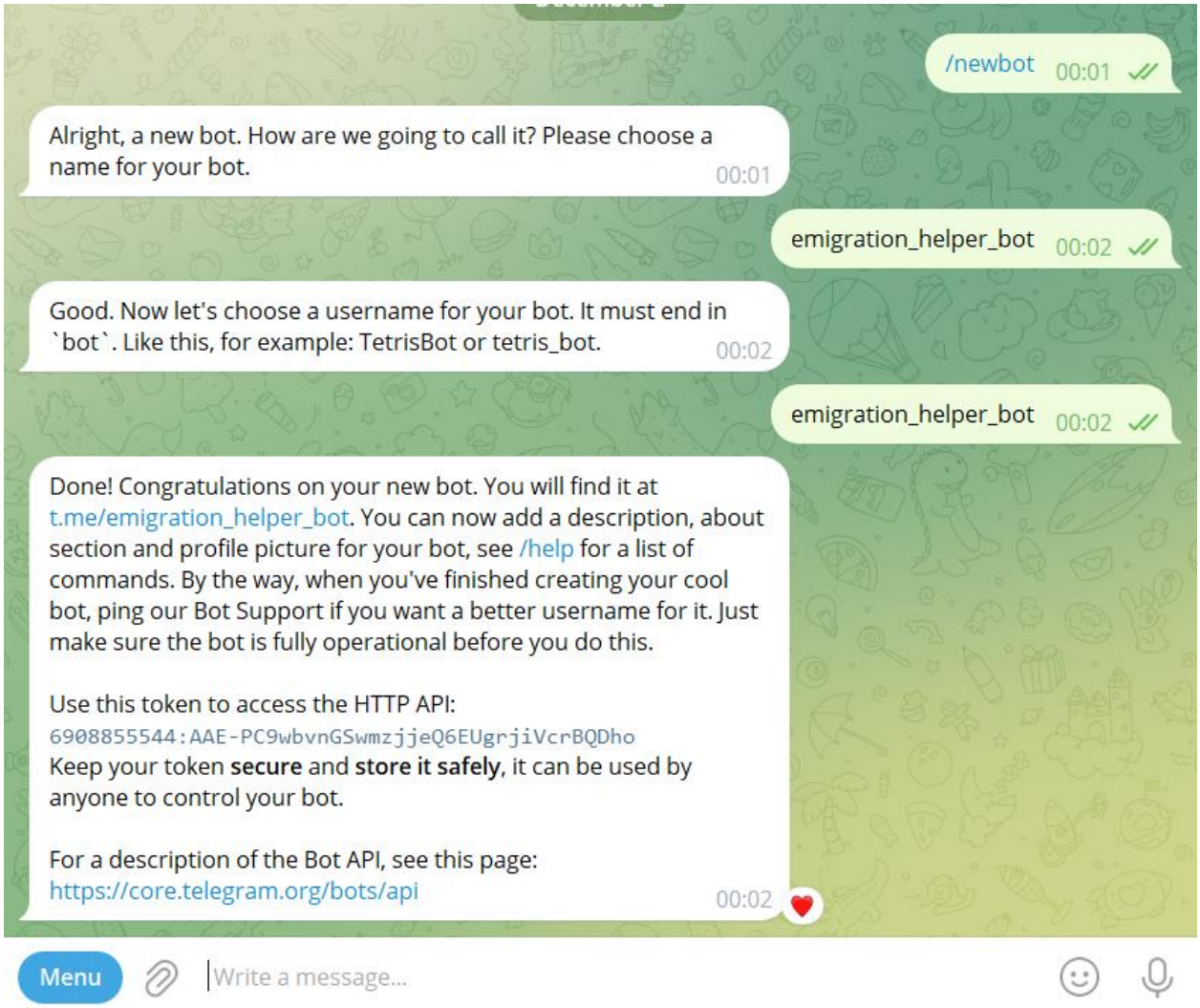

Рисунок 3.2 — Команди для створення бота

3.2 Реалізація основних компонентів

У цьому розділі будуть розглянуті ключові аспекти, такі як реєстрація користувача, обробка введених даних, вибір країни еміграції, інтеграція з мовною моделлю GPT-3 для генерації промптів, а також реалізація логіки чат-бота для ефективної комунікації з користувачем.

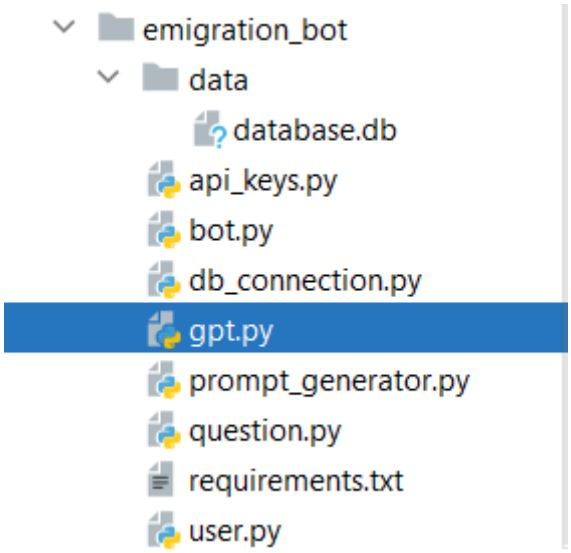

Рисунок 3.3 — Структура проекту

- data/database.db – файл з SQLlite базою даних, для зберігання інформації
- bot.py: Головний файл програми, який містить основний код для запуску та взаємодії з чат-ботом, який відповідає за взаємодію з API Telegram, обробку повідомлень та реалізацію основної логіки чат-бота.
- api_keys.py: Файл з конфігураційними параметрами, такими як токен для доступу до API Telegram, chat GPT та інші налаштування.
- user.py: Файл з імплементацією об’єкту юзера, в якому зберігається інформація про юзера:
- self.id – унікальний id юзера,
- self.additional_info – додаткова інформація про юзера,
- self.age – вік користувача,
- self.gender – стать юзера,

  - self.profession – професія юзера,
  - self.country – країна до якої юзер емігрував,
  - self.current_question – поточне питання юзера,
  - self.previous_question – попереднє питання юзера,
  - self.children – наявність дітей

- gpt.py: Файл, що містить логіку інтеграції з OpenAI GPT-4, включаючи генерацію промптів та взаємодію з моделлю.
- prompt_generator.py: Утиліта для генерації промптів на основі введеної користувачем інформації та уточнюючих фраз.
- question.py: Файл з класом який імплементує об'єкт «Питання», та має такі атрибути:
  - self.id - унікальний id питання,
  - self.question – текст самого питання
  - self.answers – список з можливими відповідями на це питання,
  - self.asked – поле яке вказує на те чи було це питання вже задане юзеру.

- requirements.txt: Файл, що містить усі залежності необхідні для роботи проекту
- db_connection.py: Файл, що містить методи для підє'днання та роботи з базою даних SQLlite

Налаштування середовища та інсталяція бібліотек.

Використовуючи одну із бібліотек Telegram Bot API (python-telegram-bot) необхідно розробити основні команди, логіку бота та інтерфейс керування, а openai – для роботи з chatGPT.

```
!pip install pyTelegramBotAPI
!pip install openai
```

Імпорт та налаштування бота в коді:

```
# Імпорт бібліотек
import telebot
@bot.message_handler(commands=['start'])
def send_start_question(message):
    current_user = get_user_from_db(message.chat.id)
```

```
    if current_user is  None:
        insert_user_info(current_user)
        current_user = get_user_from_db(message.chat.id)
    current_user = User(message.chat.id)

    dt_obj = datetime.fromtimestamp(message.date)

    mess= "Доброго ранку " if  4 <dt_obj.hour < 10 else "Доброго дня "
if  10 <dt_obj.hour < 16  else "Добрий вечір "  if 22 <dt_obj.hour <4
else "Доброї ночі "

    bot.send_message(message.chat.id, mess + message.chat.first_name +
",\n" + "Вас вітає бот для підбору та пошуку інформаційного контенту для
емігрантів.")
    current_user.current_question = questions_list_all[0]
    time.sleep(1)
    if current_user.current_question.answers:
        markup = telebot.types.InlineKeyboardMarkup()
        for i in range(len(current_user.current_question.answers)):
            button =
telebot.types.InlineKeyboardButton(text=current_user.current_question.an
swers[i],

callback_data=str(current_user.current_question.id) + "_" + str(i))
            markup.add(button)
        bot.send_message(message.chat.id,
current_user.current_question.question, reply_markup=markup)
    else:
        bot.send_message(message.chat.id,
current_user.current_question.question)

bot = telebot.TeleBot(TG_TOKEN)
bot.polling(none_stop=True)
```

Реєстрація та збір базової інформації:

```
# Додавання обробника для текстових повідомлень
@bot.message_handler(content_types=['text'])
def handle_text_message(message):
    user_c = get_user_from_db(message.chat.id)
    if user_c.current_question.id == 2:
        try:
            age = int(message.text)
            user_c.age = age
            user c.previous question = user c.current question
            user_c.current_question = questions_list_all[3]
            time.sleep(1)
            bot.send_message(user_c.id,
user_c.current_question.question)
        except:
            bot.send_message(user_c.id,
user_c.current_question.question)
```

Повний код наведено в додатках

Інтеграція з GPT моделлю: OpenAI's GPT-4

– Отримання ключа API від OpenAI (за інструкцією розділу 2.2).

– Використання ключа API в коді:

```
from openai import OpenAI
from emigration_bot.api_keys import GPT_KEY
# Встановлення ключа API
client = OpenAI(api_key=GPT_KEY)
```

Використання OpenAI GPT-4 API.

Створення функції для виклику OpenAI API та генерації тексту.

```
from openai import OpenAI
from emigration_bot.api_keys import GPT_KEY

client = OpenAI(api_key=GPT_KEY)

def get_response(prompts):
  system_promt, user_promt = prompts
  print(system_promt)
  print(user_promt)
  try:
    completion = client.chat.completions.create(
      model="gpt-4",
      messages=[
        {"role": "system", "content": system_promt},
        {"role": "user", "content": user_promt}
      ],
    max_tokens = 500
    )

    return completion.choices[0].message
  except Exception as e:
    print(e)
    return None
```

Використання функції для генерації тексту:

```
response = get_response(create_prompt(user,
                                      "Опиши випадки в яких мене можуть
депортувати з країни, та дії які можуть запобігти цьому."))
send_response(user, response)
bot.send_message(user_id, response, reply_markup=markup)
```

Генерація промптів для chatGPT:

```
def create_prompt(user, main_question):
    system_prompt = f"Ти соціальний працівник з країни під назвою
{user.country}, асистент людини, яка емігрувала з України і повинен
допомогти їй з питаннями у різних сферах життя для адаптації та
нормального життя в цій країні."
    add_info  = additional_info(user)
    client_prompt = f"{add_info} {main_question} Надай актуальну та
стислу інформацію по цьому питанню, намагайся знаходити інформацію з
офіційних джерел. Відповідь на нього дуже важлива для мене, оскільки я
знаходжусь і іншій країні."
    return (system_prompt, client_prompt)
```

```
def additional_info(user):
    additional_info= ""
    if user.gender is not None:
        if user.age is not None:
            if user.age < 30:
                additional_info+= "Я хлопець з України." if user.gender
== "M" else  "Я дівчина з України."
            else:
                additional_info += "Я чловік з України." if user.gender
== "M" else "Я жінка з України."
        else:
            additional_info+="Я чловік з України." if user.gender == "M"
else "Я жінка з України."
    if user.age is not None:
        additional_info += f" Мені {str(user.age)} років."
    if user.children:
        additional_info += f" Маю неповнолітніх дітей."
    if user.profession is not None:
        additional_info += f" Маю досвід в таких видах робіт як
{user.profession} ."
    return additional_info
```

Виведення результатів користувачу:

```
response = get_response(create_prompt(user,
                                      "Опиши процедуру оренди житла в
даній країні, які є особливості аренди, які документи необхідні, та де
можна знайти аредодавців?"))
send_response(user, response)
```

4 ТЕСТУВАННЯ РОБОТИ ПРОГРАМНОГО ПРОДУКТУ

Тестування бота є важливим етапом розробки для забезпечення високої якості та ефективності розробленого чат-боту. Першим етапом для формування більш точних запитів є питання щодо того чи бажає поточний користувач надати додаткову інформацію про себе, якщо він бажає то задаються питання які покривають досить значну невизначеність щодо того хто користується ботом в даний момент. До такої інформації належить: стать, вік, наявність неповнолітніх дітей, та чим людина займалась до еміграції (рис. 3.4). Знаючи це, бот видає набагато більш релевантні відповіді, в порівнянні без них.

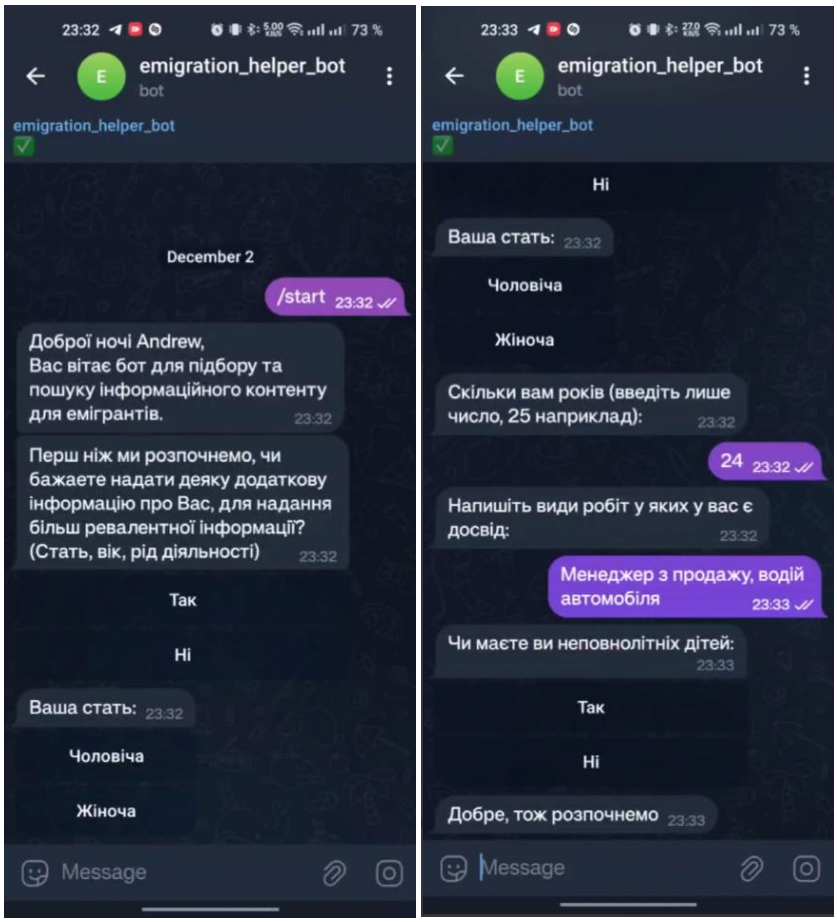

Рисунок 3.4 – Приклад додаткових питань, які задаються для формування більш точного запиту

Далі одним з важливих питань є те куди саме людина емігрувала, країни розташовані в порядку популярності вибору країни для еміграції. Якщо країна відсутня в списку, користувач може ввести її вручну (рис. 3.5)

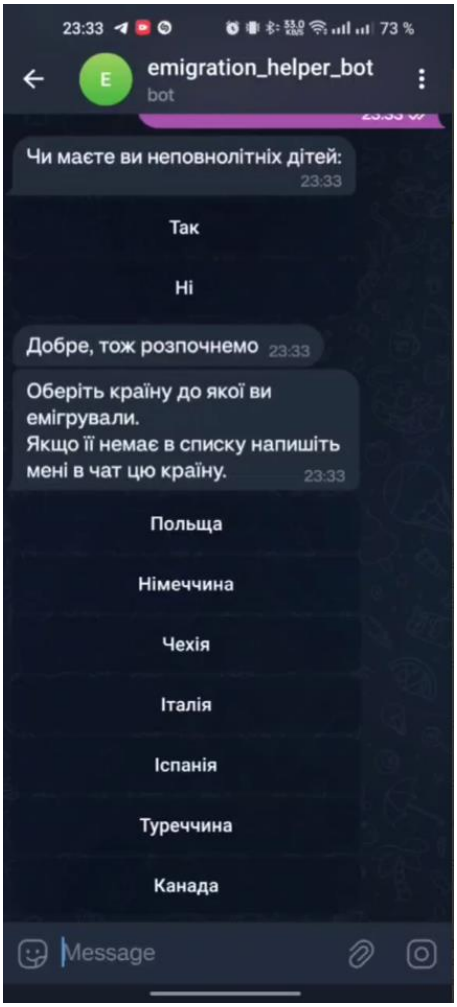

Рисунок 3.5 – Вибір країни еміграції зі списку або введення з клавіатури у разі відсутності в списку

Далі на рисунку 3.6 показано основне меню бота з категоріями та обрано підкатегорію «Юридичні питання». Де користувач може обрати зі списку категорій та підкатегорії. Категорії та підкатегорії побудовані таким чином щоб покрити основні питання які виникають у емігрантів.

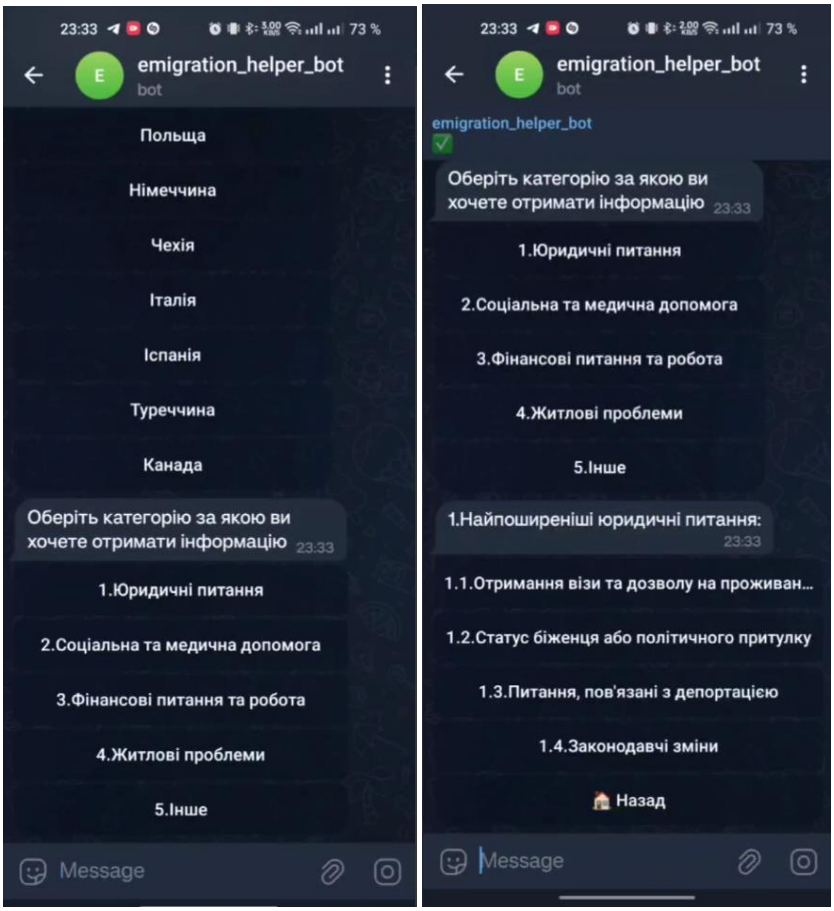

Рисунок 3.6 – Вибір основної категорії «Юридичні питання», та підкатегорію питання «Статус біженця або політичного притулку»

Далі з'являється повідомлення з інформацією про те що запит оброблюється, щоб користувач отримував зворотній зв’язок, оскільки кожен запит займає різний час, але в середньому 10-15 секунд (рис. 3.7).

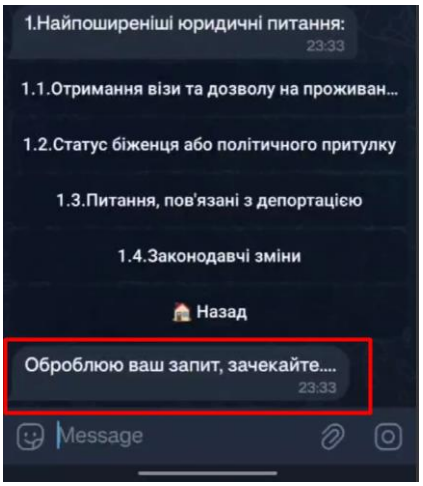

Рисунок 3.7 – Повідомлення про обробку запиту

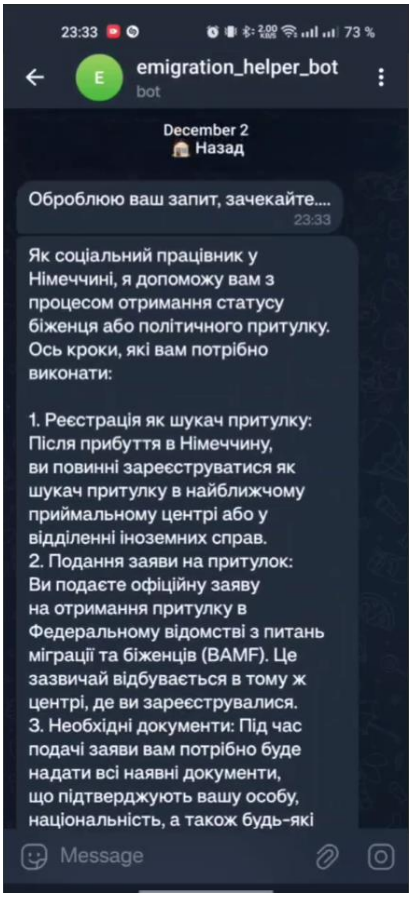

Рисунок 3.8 – Отримана відповідь на поставлене питання з відповідної категорії

Ось промпт який був згенерований щоб отримати відповідь на дане питання.

Роль чату GPT4: Ти соціальний працівник з країни під назвою Німеччина, асистент людини, яка емігрувала з України і повинен допомогти їй з питаннями у різних сферах життя для адаптації та нормального життя в цій країні.

Роль юзера: Я хлопець з України. Мені 24 років. Маю досвід в таких видах робіт як Менеджер з продажу, водій автомобіля . Опиши послідовність дій щоб отримати статус біженця або політичний притулок. Які документи та дії для цього необхідні Надати актуальну та стислу інформацію по цьому питанню, намагайся знаходити інформацію з офіційних джерел. Відповідь на нього дуже важлива для мене, оскільки я знаходжусь і іншій країні.

Також користувач може задавати власні питання, вони також будуть модифіковані з додаванням додаткової інформації. Додаткові питання які користувач вводить заносяться до бази, для подальшого аналізу з метою оптимізації поточних категорій та додавання нових.

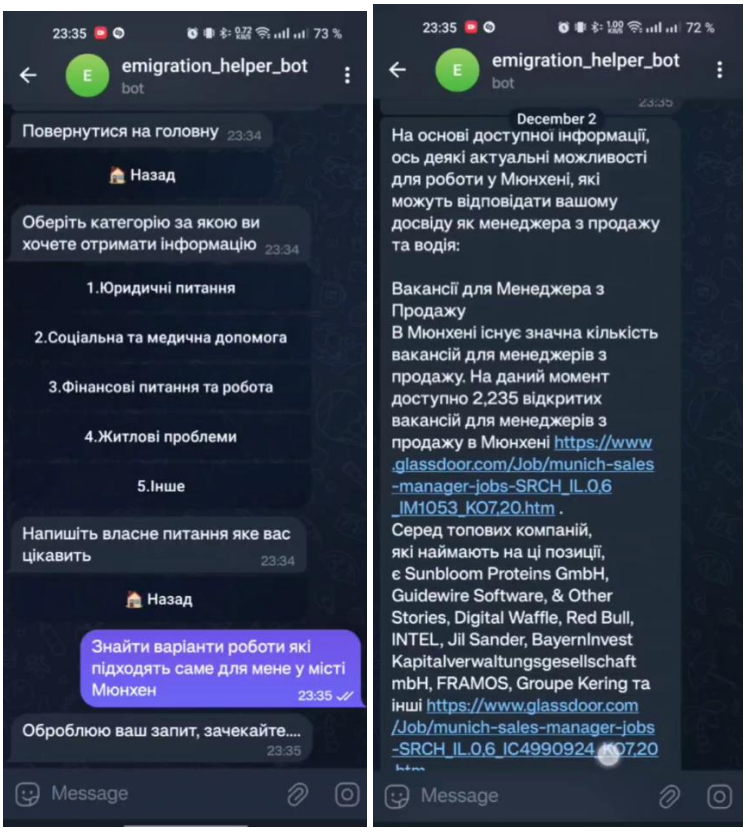

Рисунок 3.9 – Власне запитання від юзера та відповідь на нього

ВИСНОВКИ

Ця робота присвячена розробці та впровадженню інформаційної технології у вигляді чат-бота Telegram для надання інформаційної підтримки емігрантам з України. Досліджено та реалізовано функціонал, що включає в себе збір основних даних від користувачів, вибір країни еміграції, генерацію промптів для chatGPT та інші ключові елементи.

Результатом роботи став створений чат-бот, який дозволяє емігрантам легко та ефективно отримувати інформацію про умови еміграції, трудові можливості, освітні заклади та інші питання, що стосуються життя за кордоном. Використання промптів для chat GPT забезпечує коректні та індивідуалізовані відповіді, що робить взаємодію з ботом більш ефективною та приємною для користувачів. Реалізовано механізм зберігання питань які задають безпосередньо користувачі для подальшого аналізу і покращення структури категорій та підкатегорій.

Проведене тестування та отримані результати свідчать про успішність та ефективність розробленого чат-бота. Користувачі, які скористалися ботом, відзначили його зручність та корисність.

Враховуючи актуальність теми та позитивні результати впровадження, можна зазначити, що створений чат-бот є значущим внеском у полі інформаційної підтримки емігрантів та є потенційно важливим інструментом для подальших досліджень та розвитку в галузі інформаційних технологій для міграції.

СПИСОК ЛІТЕРАТУРИ

1. The role of information technology in the food industry [Електронний ресурс]. — Режим доступу: https://www.oporaua.org/viyna/cherez-povnomasshtabne-vtorgnennia-rosiyi-za-kordonom-perebuvaie-blizko-20-ukrayintsiv-doslidzhennia-opori-24792 – (дата звернення 01.12.2023) — Назва з титулу екрана.

2. В яких країнах Європи найбільше українських біженців [Електронний ресурс]. — Режим доступу: https://www.slovoidilo.ua/2022/11/08/infografika/suspilstvo/yakyx-krayinax-yevropy-najbilshe-ukrayinskyx-bizhencziv – (дата звернення 01.12.2023) — Назва з титулу екрана.

3. Скільки українських біженців перебуває за кордоном і як їх повернути, – дослідження [Електронний ресурс]. — Режим доступу: https://texty.org.ua/fragments/110574/skilky-ukrayinskyh-bizhenciv-perebuvaye-za-kordonom-i-yak-yih-povernuty-doslidzhennya/– (дата звернення 01.12.2023) — Назва з титулу екрана.

4. Як почуваються українці у Литві: дослідження виявило три проблеми [Електронний ресурс]. — Режим доступу: https://glavcom.ua/world/observe/jak-pochuvajutsja-ukrajintsi-u-litvi-doslidzhennja-vijavilo-tri-nahalni-problemi--971482.html – (дата звернення 01.12.2023) — Назва з титулу екрана.

5. Як почувають себе в Польщі українці: опитування [Електронний ресурс]. — Режим доступу: https://usp-ltd.org/iak-pochuvaiut-sebe-v-polshchi-ukraintsi-opytuvannia/ – (дата звернення 01.12.2023) — Назва з титулу екрана.

6. Як живеться у Німеччині біженцям з України - дослідження [Електронний ресурс]. — Режим доступу: https://www.dw.com/uk/ak-zivetsa-u-nimeccini-bizencam-z-ukraini-doslidzenna/a-66237029 – (дата звернення 01.12.2023) — Назва з титулу екрана.

7. Українські учні-біженці: скільки їх за кордоном та що заважає їм

навчатись? [Електронний ресурс]. — Режим доступу: https://osvitoria.media/opinions/ukrayinski-uchni-bizhentsi-skilky-yih-za-kordonom-ta-shho-zavazhaye-yim-navchatys/ – (дата звернення 01.12.2023) — Назва з титулу екрана.

8. 202 Awesome ChatGPT Prompts to Boost Productivity [Електронний ресурс]. — Режим доступу: https://blog.hootsuite.com/chatgpt-prompts/– (дата звернення 01.12.2023) — Назва з титулу екрана.

9. What is a chatbot? [Електронний ресурс]. — Режим доступу: https://www.ibm.com/topics/chatbots – (дата звернення 01.12.2023) — Назва з титулу екрана.

10. How to Master Chatbots in 60-Minutes: Get Your Time Freedom Back and Grow Your Business, Fast/ Sadler, Tarna — London: Independently published, 2018. — 75 с.

11. Python Crash Course, 2nd Edition: A Hands-On, Project-Based Introduction to Programming/ Eric Matthes — London: No Starch Press, 2019. — 544 с.

12. Effective Python: 90 Specific Ways to Write Better Python (Effective Software Development Series)/ Brett Slatkin — London: Addison-Wesley Professional, 2019. — 480 с.

13. Why Is Python Considered the Top Language? [Електронний ресурс]. — Режим доступу: https://www.bairesdev.com/blog/why-is-python-top-language/ – (дата звернення 01.12.2023) — Назва з титулу екрана.

14. Python API Development Fundamentals: Develop a full-stack web application with Python and Flask/ Jack Chan — London: Packt Publishing, 2019. — 372 с.

15. Crafting Applications with ChatGPT API: Using Python / Michael Gold — London: Green Belt Book LLC, 2023. — 180 с.

16. Developing Apps with GPT-4 and ChatGPT: Build Intelligent Chatbots, Content Generators, and More / Olivier Caelen — London: O'Reilly Media, 2023.

— 155 с.

17. Data Structures & Algorithms in Python (Developer's Library)/ John Canning — London: Addison-Wesley Professional, 2022. — 920 с.

18. Building Telegram Bots: Develop Bots in 12 Programming Languages using the Telegram Bot API/ Nicolas Modrzyk — London: Apress, 2018. — 319 с.

19. The Quick Python Book/ Naomi Ceder — London: Manning, 2019. — 472 с.

20. Компоненти синтаксису IDEF0 [Електронний ресурс]. — Режим доступу: https://elib.lntu.edu.ua/sites/default/files/elib_upload/page9.html – (дата звернення 21.11.2023) — Назва з титулу екрана.Df

ДОДАТКИ

Додаток А

Код файлу bot.py

```
import threading
import time
import telebot
from datetime import datetime

from emigration_bot.api_keys import TG_TOKEN
from emigration_bot.gpt import get_response
from emigration_bot.question import Question
from emigration_bot.emigration_bot.user import User
from emigration_bot.db_connection import get_user_from_db, insert_user_info,
update_user_info, insert_user_additional_questions
from prompt_generator import *
TO_MAIN  = "Повернутися на головну"
WAIT = "Оброблюю ваш запит, зачекайте...."

bot = telebot.TeleBot(TG_TOKEN)
questions_list_all  = [Question(0, "Перш ніж ми розпочнемо, чи бажаете надати
деяку додаткову інформацію про Вас, для надання більш ревалентної інформації?
(Стать, вік, рід діяльності)",
                                ["Так", "Ні"]),
                       Question(1,
                                "Ваша стать:",
                                ["Чоловіча", "Жіноча"]),
                       Question(2,
                                "Скільки вам років (введіть лише число, 25
наприклад):",
                                []),
                        Question(3,
                                "Напишіть види робіт у яких у вас є досвід:",
                                []),
                        Question(4,
                                "Чи маєте ви неповнолітніх дітей:",
                                ["Так", "Ні"]),
                       Question(5,
                                "Оберіть країну до якої ви емігрували.\nЯкщо
її немає в списку напишіть мені в чат цю країну.",
                                ["Польща",  "Німеччина", "Чехія", "Італія",
"Іспанія", "Туреччина", "Канада"]),
                        Question(6,
                                "Оберіть категорію за якою ви хочете отримати
інформацію",
                                ["1.Юридичні питання", "2.Соціальна та
медична допомога", "3.Фінансові питання та робота", "4.Житлові проблеми",
"5.Інше"]),
                        Question(7,
                                "1.Найпоширеніші юридичні питання:",
                                ["1.1.Отримання візи та дозволу на
проживання:",
                                 "1.2.Статус біженця або політичного
притулку",
                                 "1.3.Питання, пов'язані з депортацією",
                                 "1.4.Законодавчі зміни"]),
                       Question(8,
                                "2.Соціальна та медична допомога:",
                                ["2.1.Інтеграція в нове суспільство",
```

```
                                "2.2.Мовні бар'єри",
                                "2.3.Доступ до освіти",
                                "2.4.Доступ до медичних послуг"]),
                        Question(9,
                                "3.Фінансові питання та робота:",
                                ["3.1.Фінансова підтримка",
                                 "3.2.Банківські послуги",
                                 "3.3.Податкові обов'язки",
                                 "3.4.Працевлаштування"
                                 ]),
                        Question(10,
                                "4.Житлові проблеми",
                                ["4.1.Ореда житла",
                                 "4.2.Соціальне житло"]),
                         Question(11,
                                "Напишіть власне питання яке вас цікавить",
                                [])
                        ]


def generate_next_question(user, answer=None):
    user.previous_question = user.current_question
    if answer is not None:
        if int(answer.split('_')[1]) == -2:
            user.current_question = questions_list_all[6]
            if answer is not None and user.previous_question.id == 5:
                user.country =
user.previous_question.answers[int(answer.split('_')[1])]

            markup = telebot.types.InlineKeyboardMarkup()
            for i in range(len(user.current_question.answers)):
                button =
telebot.types.InlineKeyboardButton(text=user.current_question.answers[i],

callback_data=str(

user.current_question.id) + "_" + str(i))
                markup.add(button)
            send_message_to_user(user.id, user.current_question.question,
markup)

    if user.current_question.id == 0:
        if int(answer.split('_')[1]) == 0:
            user.additional_info = True
            user.current_question = questions_list_all[1]
            markup = telebot.types.InlineKeyboardMarkup()
            for i in range(len(user.current_question.answers)):
                button =
telebot.types.InlineKeyboardButton(text=user.current_question.answers[i],

callback_data=str(

user.current_question.id) + "_" + str(i))
                markup.add(button)
            send_message_to_user(user.id, user.current_question.question,
markup)
        else:
            user.additional_info = False
            user.current_question = questions_list_all[5]
            bot.send_message(user.id, "Добре, тож розпочнемо")
            time.sleep(1.5)
```

```
            markup = telebot.types.InlineKeyboardMarkup()
            for i in range(len(user.current_question.answers)):
                button =
telebot.types.InlineKeyboardButton(text=user.current_question.answers[i],

callback_data=str(

user.current_question.id) + "_" + str(i))
                markup.add(button)
            send_message_to_user(user.id, user.current_question.question,
markup)
    elif user.current_question.id == 1:
        user.current_question = questions_list_all[2]
        if int(answer.split('_')[1]) == 0:
            user.gender = "M"
        else:
            user.gender = "W"
        send_message_to_user(user.id, user.current_question.question)

    elif user.current_question.id == 4:
        user.current_question = questions_list_all[5]
        bot.send_message(user.id, "Добре, тож розпочнемо")
        if int(answer.split('_')[1]) == 0:
            user.children = True

        markup = telebot.types.InlineKeyboardMarkup()
        for i in range(len(user.current_question.answers)):
            button =
telebot.types.InlineKeyboardButton(text=user.current_question.answers[i],
                                                    callback_data=str(

user.current_question.id) + "_" + str(i))
            markup.add(button)
        send_message_to_user(user.id, user.current_question.question, markup,
delay=1.5)

    elif user.current_question.id == 5:
        user.current_question = questions_list_all[6]
        if answer is not None:
            user.country =
user.previous_question.answers[int(answer.split('_')[1])]

        markup = telebot.types.InlineKeyboardMarkup()
        for i in range(len(user.current_question.answers)):
            button =
telebot.types.InlineKeyboardButton(text=user.current_question.answers[i],
                                                    callback_data=str(

user.current_question.id) + "_" + str(i))
            markup.add(button)
        send_message_to_user(user.id, user.current_question.question, markup,
delay=1)

    elif user.current_question.id == 7:
        if int(answer.split('_')[1]) == 0:
            send_wait_message_in_thread(user)
            response = get_response(create_prompt(user, "Напиши яким чином
можна отримати візу та дозвіл на проживання. Які документи та дії для цього
необхідні?"))
            send_response(user, response)
        elif int(answer.split('_')[1]) == 1:
            send_wait_message_in_thread(user)
```

```
            response = get_response(create_prompt(user,
                                                  "Опиши послідовність дій
щоб отримати статус біженця або політичний притулок. Які документи та дії для
цього необхідні"))
            send_response(user, response)
        elif int(answer.split('_')[1]) == 2:
            send_wait_message_in_thread(user)
            response = get_response(create_prompt(user,
                                                  "Опиши випадки в яких мене
можуть депортувати з країни, та дії які можуть запобігти цьому."))
            send_response(user, response)
        elif int(answer.split('_')[1]) == 3:
            send_wait_message_in_thread(user)
            response = get_response(create_prompt(user,
                                                  "Опиши які зміни в
законодавстві відбулися в останній час та стосуються емігрантів, та як це
вплине на мене."))
            send_response(user, response)
    elif user.current_question.id == 8:
        if int(answer.split('_')[1]) == 0:
            send_wait_message_in_thread(user)
            response = get_response(create_prompt(user,
                                                  "Напиши які є варіанти
соціальної інтеграції в нове суспільство, та який варіант може підійти саме
для мене."))
            send_response(user, response)
        elif int(answer.split('_')[1]) == 1:
            send_wait_message_in_thread(user)
            response = get_response(create_prompt(user,
                                                  "Опиши як можна в
найкоротші терміни подолати мовні бар'єри в незнайомій країні. Які соціальні
програми наявні в цій країні?"))
            send_response(user, response)
        elif int(answer.split('_')[1]) == 2:
            send_wait_message_in_thread(user)
            response = get_response(create_prompt(user,
                                                  "Напиши чи є в цій країні
можливіть емігрантам отримати доступ до освіти, отрмати нову спеціальність.
Напиши які тут є програми то що потрібно зробити щоб ними скоритстатися.?"))
            send_response(user, response)
        elif int(answer.split('_')[1]) == 3:
            send_wait_message_in_thread(user)
            response = get_response(create_prompt(user,
                                                  "Як мені отримати доступ до
медичних послуг? Опиши варіанти отримання медичних послуг в даній країні"))
            send_response(user, response)
    elif user.current_question.id == 9:
        if int(answer.split('_')[1]) == 0:
            send_wait_message_in_thread(user)
            response = get_response(create_prompt(user,
                                                  "Опиши які є наявні
варіанти та програми в даній країні для отримання фінасової пдтримки для
емігрантів. Та куди потрібно звертатися за такою допомогою, та які документи
мати?"))
            send_response(user, response)
        elif int(answer.split('_')[1]) == 1:
            send wait message in thread(user)
            response = get_response(create_prompt(user,
                                                  "Як відкрити банківський
рахунок в даній країні, які документи необхідно мати та скільки приблизно
часу це займе?"))
            send_response(user, response)
```

```
        elif int(answer.split('_')[1]) == 2:
            send_wait_message_in_thread(user)
            response = get_response(create_prompt(user,
                                                  "знайди та напиши які
податкові зобов'язання є на даний момент в цій країні для емігрантів, згідно
чинного законодавства."))
            send_response(user, response)
        elif int(answer.split('_')[1]) == 3:
            send_wait_message_in_thread(user)
            response = get_response(create_prompt(user,
                                                  "Напиши які є варіанти для
пошуку офіційної роботи в даній країні з та без знання місцевої мови?"))
            send_response(user, response)
    elif user.current_question.id == 10:
        if int(answer.split('_')[1]) == 0:
            send_wait_message_in_thread(user)
            response = get response(create prompt(user,
                                                  "Опиши процедуру оренди
житла в даній країні, які є особливості аренди, які документи необхідні, та де
можна знайти аредодавців?"))
            send_response(user, response)
        elif int(answer.split('_')[1]) == 1:
            send_wait_message_in_thread(user)
            response = get_response(create_prompt(user,
                                                  "Чи є можливість в даній
країні отримати соціальне житло, які документи необхідно мати та куди
необхідно звертатися?"))
            send_response(user, response)

    elif user.current_question.id == 6:
        if answer is not None:
            choose_int = int(answer.split('_')[1])
            if choose_int == 0:
                user.current_question = questions_list_all[7]
                markup = telebot.types.InlineKeyboardMarkup()
                for i in range(len(user.current_question.answers)):
                    button =
telebot.types.InlineKeyboardButton(text=user.current_question.answers[i],

callback_data=str(

user.current_question.id) + "_" + str(i))
                    markup.add(button)
                markup.add(add_menu_button(user.current_question.id))
                bot.send_message(user.id, user.current_question.question,
reply_markup=markup)
            elif choose_int == 1:
                user.current_question = questions_list_all[8]
                markup = telebot.types.InlineKeyboardMarkup()
                for i in range(len(user.current_question.answers)):
                    button =
telebot.types.InlineKeyboardButton(text=user.current_question.answers[i],

callback_data=str(

user.current_question.id) + "_" + str(i))
                    markup.add(button)
                markup.add(add_menu_button(user.current_question.id))
                bot.send_message(user.id, user.current_question.question,
reply_markup=markup)
            elif choose_int == 2:
                user.current_question = questions_list_all[9]
```

```
                markup = telebot.types.InlineKeyboardMarkup()
                for i in range(len(user.current_question.answers)):
                    button =
telebot.types.InlineKeyboardButton(text=user.current_question.answers[i],

callback_data=str(

user.current_question.id) + "_" + str(i))
                    markup.add(button)
                markup.add(add_menu_button(user.current_question.id))
                bot.send_message(user.id, user.current_question.question,
reply_markup=markup)
            elif choose_int == 3:
                user.current_question = questions_list_all[10]
                markup = telebot.types.InlineKeyboardMarkup()
                for i in range(len(user.current_question.answers)):
                    button =
telebot.types.InlineKeyboardButton(text=user.current_question.answers[i],

callback_data=str(

user.current_question.id) + "_" + str(i))
                    markup.add(button)
                markup.add(add_menu_button(user.current_question.id))
                bot.send_message(user.id, user.current_question.question,
reply_markup=markup)
            elif choose_int == 4:
                user.current_question = questions_list_all[11]
                markup = telebot.types.InlineKeyboardMarkup()
                markup.add(add_menu_button(user.current_question.id))
                bot.send_message(user.id, user.current_question.question,
reply_markup=markup)

    update_user_info(user)


def send_response(user, response):
    markup = telebot.types.InlineKeyboardMarkup()
    markup.add(add_menu_button(user.current_question.id))
    send_message_to_user(user.id, response, delay=1)
    send_message_to_user(user.id, TO_MAIN, markup, delay=1)

def send_wait_message_in_thread(user):
    threading.Thread(target=send_message_to_user, args=(user.id, WAIT, None,
3,)).start()

def add_menu_button(current_question_id):
    return telebot.types.InlineKeyboardButton(text="· Назад",

callback_data=str(current_question_id) + "_" + str(-2))

@bot.message_handler(commands=['start'])
def send_start_question(message):
    current_user = get_user_from_db(message.chat.id)
    if current_user is  None:
        insert_user_info(current_user)
        current_user = get_user_from_db(message.chat.id)
    current_user = User(message.chat.id)

    dt_obj = datetime.fromtimestamp(message.date)

    mess= "Доброго ранку " if  4 <dt_obj.hour < 10 else "Доброго дня " if  10
```

```
<dt_obj.hour < 16  else "Добрий вечір "  if 22 <dt_obj.hour <4 else "Доброї
ночі "

    bot.send_message(message.chat.id, mess + message.chat.first_name + ",\n"
+ "Вас вітає бот для підбору та пошуку інформаційного контенту для
емігрантів.")
    current_user.current_question = questions_list_all[0]
    time.sleep(1)
    if current_user.current_question.answers:
        markup = telebot.types.InlineKeyboardMarkup()
        for i in range(len(current_user.current_question.answers)):
            button =
telebot.types.InlineKeyboardButton(text=current_user.current_question.answers
[i],

callback_data=str(current_user.current_question.id) + "_" + str(i))
            markup.add(button)
        bot.send_message(message.chat.id,
current_user.current_question.question, reply_markup=markup)
    else:
        bot.send_message(message.chat.id,
current_user.current_question.question)


@bot.callback_query_handler(func=lambda call: True)
def handle_query(call):
    bot.answer_callback_query(call.id, "✔")
    user_c = get_user_from_db(call.from_user.id)
    generate_next_question(user_c, call.data)


@bot.message_handler(content_types=['text'])
def handle_text_message(message):
    user_c = get_user_from_db(message.chat.id)
    if user_c.current_question.id == 2:
        try:
            age = int(message.text)
            user_c.age = age
            user_c.previous_question = user_c.current_question
            user_c.current_question = questions_list_all[3]
            time.sleep(1)
            bot.send_message(user_c.id, user_c.current_question.question)
        except:
            bot.send_message(user_c.id, user_c.current_question.question)
    elif user_c.current_question.id == 3:
        profession = message.text if len(message.text) > 3 else None
        if profession is not None:
            user_c.profession = profession
        user_c.previous_question = user_c.current_question
        user_c.current_question = questions_list_all[4]
        time.sleep(1)
        markup = telebot.types.InlineKeyboardMarkup()
        for i in range(len(user_c.current_question.answers)):
            button =
telebot.types.InlineKeyboardButton(text=user_c.current_question.answers[i],
                                                    callback_data=str(

user_c.current_question.id) + "_" + str(i))
            markup.add(button)
        bot.send_message(user_c.id, user_c.current_question.question,
reply_markup=markup)
```

```
    elif user_c.current_question.id == 5:
            user_c.country = message.text
            time.sleep(1)
            generate_next_question(user_c, None)

    elif user_c.current_question.id == 11:
        send_wait_message_in_thread(user_c)
        response = get_response(create_prompt(user_c, message.text))
        send_response(user_c, response)
        insert_user_additional_questions(user_c.id, message.text, response)


def send_message_to_user(user_id, text, markup=None, delay=0.0):
    time.sleep(delay)
    bot.send_message(user_id, text, reply_markup=markup)


bot.polling(none_stop=True)
```

Додаток Б

Код файлу db_connection.py

```
import sqlite3
conn = sqlite3.connect('database.db')
c = conn.cursor()
# Create table
# c.execute('''CREATE TABLE users
#              (id text, additional_info text, age integer, gender text,
#              profession text, country text, current_question text,
#              previous question text, children boolean)''')

# c.execute('''CREATE TABLE users_additional_questions
#              (id text, answer)''')
def update_user_info(user):
    c = conn.cursor()
    c.execute("""UPDATE users SET additional_info = :additional_info,
                 age = :age, gender = :gender, profession = :profession,
                 country = :country, current_question = :current_question,
                 previous_question = :previous_question,
                 children = :children
                 WHERE id = :id""",
              {'id': user.id,
               'additional_info': user.additional_info,
               'age': user.age,
               'gender': user.gender,
               'profession': user.profession,
               'country': user.country,
               'current_question': user.current_question,
               'previous_question': user.previous_question,
               'children': user.children})
    conn.commit()

def insert_user_info(user):
    c.execute(
        "INSERT INTO users VALUES (:id, :additional_info, :age, :gender,
:profession, :country, :current_question, :previous_question, :children)",
        {'id': user.id, 'additional_info': user.additional_info,
         'age': user.age, 'gender': user.gender,
         'profession': user.profession, 'country': user.country,
         'current_question': user.current_question,
         'previous_question': user.previous_question,
         'children': user.children})

    conn.commit()

def get_user_from_db(id):
    # Read and print user
    c.execute("SELECT * FROM users WHERE id=:id", {'id': id})
    user_from_db = c.fetchone()
    print(user_from_db)

def insert_user_additional_questions(id, text, answer):
    c.execute(
        "INSERT INTO users_additional_questions VALUES (:id, :text,
:answer)",
        {'id': id,
         'text': text,
         'answer': answer })
```

```
conn.commit()
```

Додаток В

Код файлу gpt.py

```
from openai import OpenAI
from emigration_bot.api_keys import GPT_KEY

client = OpenAI(api_key=GPT_KEY)
counter = 0
def get_response(prompts):
  system_promt, user_promt = prompts
  print(system_promt)
  print(user_promt)
  try:
    completion = client.chat.completions.create(
      model="gpt-4",
      messages=[
        {"role": "system", "content": system_promt},
        {"role": "user", "content": user_promt}
      ],
    max_tokens = 500
    )

    return completion.choices[0].message
  except Exception as e:
    print(e)
    return None
```

Код файлу prompt_generator.py

```
def create_prompt(user, main_question):
    system_prompt = f"Ти соціальний працівник з країни під назвою
{user.country}, асистент людини, яка емігрувала з України і повинен допомогти
їй з питаннями у різних сферах життя для адаптації та нормального життя в цій
країні."
    add_info  = additional_info(user)
    client_prompt = f"{add_info} {main_question} Надай актуальну та стислу
інформацію по цьому питанню, намагайся знаходити інформацію з офіційних
джерел. Відповідь на нього дуже важлива для мене, оскільки я знаходжусь і
іншій країні."
    return (system_prompt, client_prompt)

def additional_info(user):
    additional_info= ""
    if user.gender is not None:
        if user.age is not None:
            if user.age < 30:
                additional_info+= "Я хлопець з України." if user.gender ==
"M" else  "Я дівчина з України."
            else:
                additional_info += "Я чловік з України." if user.gender ==
"M" else "Я жінка з України."
        else:
            additional_info+="Я чловік з України." if user.gender == "M" else
"Я жінка з України."
    if user.age is not None:
        additional_info += f" Мені {str(user.age)} років."
    if user.children:
        additional_info += f" Маю неповнолітніх дітей."
    if user.profession is not None:
```

```
        additional_info += f" Маю досвід в таких видах робіт як
{user.profession} ."
    return additional_info
```

Код файлу user.py

```
class User:
    def __init__(self, id):
        self.id = id
        self.additional_info = None
        self.age = None
        self.gender = None
        self.profession = None
        self.country = None
        self.current_question=None
        self.previous_question = None
        self.children = False

    def __eq__(self, other):
        if not isinstance(other, User):
            return False
        if self.id != other.id:
            return False
        return True

    def __hash__(self):
        return self.id
```