Міністерство освіти і науки України

Тернопільський національний технічний університет імені Івана Пулюя

(повне найменування вищого навчального закладу)

Факультет: « комп'ютерно-інформаційних систем і програмної інженерії »

(назва факультету)

Кафедра: «програмної інженерії»

(повна назва кафедри)

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проекту

Бакалавр

(освітньо-кваліфікаційний рівень)

на тему: Розробка чат-боту для скринінгу кандидатів перед працевлаштуванням за допомогою PHP та відкритого API “Дія”

Виконала: студентка 4 курсу, групи СПз-41

спеціальності 121 Інженерія програмного забезпечення

(шифр і назва напряму підготовки, спеціальності)

	(підпис)	Подольчак А. А. (прізвище та ініціали)
Керівник	(підпис)	Стефанишин В. М. (прізвище та ініціали)
Нормоконтроль	(підпис)	(прізвище та ініціали)
Рецензент	(підпис)	(прізвище та ініціали)

м. Тернопіль – 2023

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)
Факультет комп'ютерно-інформаційних систем і програмної інженерії
Кафедра програмної інженерії
Освітньо-кваліфікаційний рівень бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри М. Р. _____
«____» _____ 2023

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЕКТ СТУДЕНТУ

Подольчак Аліна Анатоліївна
(ПРІЗВИЩЕ, ІМ'Я, ПО БАТЬКОВІ)

1. Тема проекту (роботи)
Розробка чат-боту для скринінгу кандидатів перед працевлаштуванням за допомогою PHP та відкритого API "Дія"

Керівник проекту (роботи) Стефанишин В. М.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
Затверджені наказом по університету від «08» березня 2023 року № 4/7-282
2. Термін подання студентом проекту (роботи) по 12.06.2023 р.
3. Вихідні дані до проекту (роботи) підсумки розробки системи керування чат-ботом

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
Вступ, аналіз предметної області та постановка завдання, вибір технологій для розробки системи керування чат-ботом, розробка програмного продукту, безпека життєдіяльності, основи охорони праці висновки, додатки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
Презентація PowerPoint на 10 слайдів

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Аналіз предметної області постановка завдання	Стефанишин В. М.	07.04.2023	07.04.2023
Вибір технологій для розробки системи керування чат-ботом	Стефанишин В. М.	20.04.2023	20.04.2023
Розробка програмного продукту	Стефанишин В. М.	04.05.2023	04.05.2023
Безпека життєдіяльності, основи охорони праці	Гурик О. Я.	23.05.2023	23.05.2023

7. Дата видачі завдання:

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1	Отримати перелік питань та завдання для дослідження на виконання дипломної роботи	14.03.2023	
2	Скласти план дипломної роботи	24.03.2023	
3	Ознайомитись з літературними джерелами	27.03.2023	
4	Провести аналіз предметної області	09.04.2023	
5	Написати перший розділ	18.04.2023	
6	Спроектувати архітектуру системи	28.04.2023	
7	Написати другий розділ	02.05.2023	
8	Спроектувати систему	22.05.2023	
9	Написати третій розділ	25.05.2023	
10	Написати четвертий розділ	26.05.2023	
11	Зробити висновки по роботі	28.05.2023	
12	Попередній захист	30.05.2023	
13	Захист	14.06.2023	

Студент _____ Подольчак А. А.
(підпис) (прізвище та ініціали)

Керівник проекту _____ Стефанишин В. М.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Подольчак А.А. Розробка чат-боту для скринінгу кандидатів перед працевлаштуванням за допомогою PHP та відкритого API “Дія”: 94 ст., 20 рисунків, 9 таблиць, 24 використаних джерела, 4 додатки.

Дипломна бакалаврська робота за спеціальністю 121 – «Інженерія програмного забезпечення». - Тернопільський національний технічний університет імені Івана Пулюя, Тернопіль, 2023 рік.

Мета даної дипломної роботи полягає у створенні чат-бота, який буде використовуватись для перевірки кандидатів перед їхнім прийняттям на роботу. Для розробки використана мова програмування PHP та відкрите API "Дія". Основними завданнями цього дослідження є аналіз сучасних підходів до перевірки кандидатів, проектування та реалізація чат-бота з використанням мови програмування PHP, інтеграція з відкритим API "Дія" для отримання даних про кандидатів та оцінювання їхньої придатності, а також проведення валідації та тестування розробленого чат-бота.

Чат-бот дозволить компаніям здійснювати детальну перевірку кандидатів на їхню чесність та правдивість інформації. Завдяки цьому, компанії мають можливість забезпечити надійність та об'єктивність в процесі підбору персоналу, уникаючи співпраці з потенційно ненадійними або некваліфікованими кандидатами. Такий підхід сприяє покращенню якості та ефективності відбору персоналу.

Ключові слова: Чат-бот, інформаційні системи, Telegram, PHP, скринінг, Telegram API, мікросервісна архітектура, месенджер, Docker, MySQL, діаграма.

ANNOTATION

Podolchak A.A. Development of a chatbot for screening candidates before employment using PHP and the open API "Diia": 94 pages, 20 pictures, 9 tables, 24 used sources, 4 additions.

Bachelor's thesis in the speciality 121 - "Software Engineering". - Ternopil National Technical University named after Ivan Puluj, Ternopil, 2023.

The purpose of this thesis is to create a chatbot that will be used to check candidates before hiring them. The PHP programming language and the open API "Diia" were used for the development. The main objectives of this study are to analyse current approaches to candidate screening, design and implement a chatbot using the PHP programming language, integrate with the open API Diia to obtain data on candidates and assess their suitability, and validate and test the developed chatbot.

The chatbot will allow companies to carry out a detailed check of candidates for their honesty and truthfulness of information. As a result, companies can ensure reliability and objectivity in the recruitment process, avoiding cooperation with potentially unreliable or unqualified candidates. This approach helps to improve the quality and efficiency of recruitment.

Keywords: Chatbot, information systems, Telegram, PHP, screening, Telegram API, microservice architecture, messenger, Docker, MySQL, diagram.

ЗМІСТ

ВСТУП

Актуальність дослідження. У сучасному світі, де конкуренція на ринку праці постійно зростає, ефективний процес підбору та відбору кандидатів перед працевлаштуванням стає важливим етапом для компаній. Однак, традиційні методи оцінки кандидатів, які базуються на інтерв'ю та резюме, можуть бути обтяжливими та затратними. Тому розробка чат-боту для скринінгу кандидатів стає актуальною задачею, що спрощує та автоматизує процес відбору.

Мета і завдання дослідження. Метою даної дипломної роботи є розробка чат-боту для скринінгу кандидатів перед працевлаштуванням, використовуючи PHP та відкрите API "Дія". Завданнями дослідження є аналіз сучасних підходів до скринінгу кандидатів, проектування та реалізація чат-боту з використанням мови програмування PHP, інтеграція з відкритим API "Дія" для отримання даних про кандидатів та оцінювання їхньої придатності, а також валідація та тестування розробленого чат-боту.

Об'єкт дослідження. Об'єктом дослідження є процес скринінгу кандидатів перед працевлаштуванням. Дослідження фокусується на розробці чат-боту, який забезпечує швидкий та ефективний вивід інформації про кандидатів.

Предмет дослідження. Предметом дослідження є розробка чат-боту для скринінгу кандидатів перед працевлаштуванням. Предмет дослідження включає проектування та реалізацію функцій чат-боту, обробку та аналіз отриманих даних, а також визначення критеріїв оцінювання придатності кандидатів.

Методи дослідження. Для досягнення поставлених завдань використовуватимуться наступні методи дослідження: аналіз існуючих рішень у сфері скринінгу кандидатів, проектування та реалізація чат-боту, проведення валідації та тестування розробленої системи.

Наукова новизна роботи. Науковою новизною даної роботи є розробка

чат-боту для скринінгу кандидатів перед працевлаштуванням. Дане дослідження спрямоване на розширення можливостей процесу підбору та відбору кандидатів, забезпечуючи ефективність та автоматизацію завдяки використанню чат-боту та відкритого API.

Практичне значення. Розроблений чат-бот для скринінгу кандидатів забезпечує компаніям можливість автоматизувати процес відбору кандидатів, що дозволяє ефективно відфільтрувати та оцінити потенційних претендентів на вакантну посаду. Це сприяє економії часу та зусиль, а також покращує якість підбору персоналу.

Апробація результатів дослідження. Розроблений чат-бот буде підданий апробації шляхом тестування його функціональності та ефективності на реальних кандидатах перед працевлаштуванням. Результати тестування допоможуть оцінити ефективність розробленого рішення та внести необхідні корективи для його вдосконалення.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАВДАННЯ

1.1 Загальні відомості про чат-боти

Чат-бот — це програма, призначена для автоматичноЇ відповіді на повідомлення від користувачів через чат-інтерфейс. Цей інтерфейс може бути вбудований в сайт, мобільний додаток або месенджер, такий як Facebook Messenger, Skype, WhatsApp, Discord чи Telegram.

Чат-боти можуть використовуватись для різних цілей. Наприклад, вони можуть відповідати на запитання користувачів про товари або послуги, забезпечувати технічну підтримку, здійснювати продажі, надавати інформацію про події та інше.

Однією з переваг використання чат-ботів є те, що вони можуть працювати цілодобово та відповідати на запитання користувачів в режимі реального часу. Крім того, чат-боти можуть забезпечувати значний позитивний ефект за рахунок автоматизації певних процесів, таких як підтримка клієнтів та обробка замовлень.

Щоб створити ефективного чат-бота, необхідно добре зрозуміти потреби користувачів та врахувати їхні можливі запитання. Також важливо розробити чітку та зрозумілу стратегію використання чат-бота, визначити його функції та можливості, а також врахувати потенційні проблеми та недоліки.

Крім того, для успішної реалізації чат-бота важливо забезпечити якісне програмування та тестування. Чат-бот повинен бути готовим до роботи з різними мовленнєвими варіантами, або наперед визначеним типом введення.

Усі ці фактори впливають на якість та ефективність роботи чат-бота, що, в свою чергу, впливає на рівень задоволеності користувачів.

Зараз багато компаній вже успішно використовують чат-ботів для взаємодії зі своїми клієнтами. Наприклад, чат-бот може допомогти клієнтам знайти необхідний товар або послугу, відповісти на запитання про доставку, оплату та гарантію, надати рекомендації або просто поговорити з клієнтом.

Для використання чат-бота не потрібно мати технічних знань або додаткових програм. Більшість чат-ботів можна запустити на сторінці Facebook або на власному веб-сайті за допомогою спеціального інтерфейсу.

Чат-боти можуть використовуватись в різних галузях бізнесу, наприклад, в онлайн-роздрібній торгівлі, туризмі, готельному бізнесі, фінансових послугах та багатьох інших. Важливо зазначити, що добре розроблений чат-бот може значно зменшити витрати на підтримку клієнтів, оскільки він може забезпечити 24/7 підтримку без необхідності в зайнятті живих операторів.

Для того, щоб розпочати використання чат-бота у своєму бізнесі, потрібно звернутись до професійних розробників, які зможуть забезпечити якісне програмування та інтеграцію з потрібними сервісами. Також варто визначити мету використання чат-бота та його функціональні можливості. Наприклад, чат-бот може відповідати на запитання клієнтів, здійснювати продажі, робити бронювання та багато іншого.

Однією з найбільш популярних платформ для розробки чат-ботів є Telegram, оскільки цей месенджер має багато користувачів та забезпечує зручний інтерфейс для взаємодії з чат-ботами.

Також важливо пам'ятати, що чат-бот не може повністю замінити живого оператора. Він може забезпечувати автоматизовані відповіді на прості запитання та завдання, але для більш складних ситуацій, які потребують індивідуального підходу та емоційного контакту, необхідно залучати живого оператора.

Чат-боти є одним з найбільш перспективних напрямків розвитку інтернет-технологій, і вони стають все більш поширеними в різних сферах діяльності, від роздрібної торгівлі до фінансових послуг та медицини. Розвиток штучного інтелекту та навчання з підсиленням дає змогу зробити чат-ботів більш "розумними" та адаптивними до потреб користувачів, що забезпечує їх ефективність та популярність серед бізнесів та користувачів.

Чат-боти класифікуються за кількома критеріями: алгоритм роботи,

формат взаємодії та метод розпізнавання.

За алгоритмом роботи розрізняють такі типи чат-ботів:

- обмежені - відповідають на заздалегідь прописаний сценарій та обмежені кількістю можливих відповідей. Такі боти часто використовують для відповіді на конкретні запити користувачів, заздалегідь передбачені сценарієм;
- саморозвиваючі - працюють на основі штучної нейронної мережі та можуть зрозуміти сутність повідомлення та вести реалістичну бесіду з коротким текстом, навчаючись з часом давати більш релевантні відповіді.

За форматом взаємодії з користувачем розрізняють такі типи чат-ботів:

- кнопкові - найчастіше використовуються в месенджерах. Користувач взаємодіє з ботом через кнопки з варіантами відповіді, на які бот реагує як на команди та видає відповіді або підтверджує вибір.
- текстові - найбільш функціональний вид віртуального співрозмовника. Взаємодія з ними наближена до людської, бот аналізує інформацію та вибирає найбільш актуальну з підготовлених відповідей.
- вбудовані - інлайн-чат-боти, які з'являються по середині діалогу в месенджері після виклику та пропонує варіанти дій. Результатом можна поділитися з співрозмовником, з яким йшов діалог. Це корисно для пошуку підходящих місць, послуг та інших подібних послуг.

Також чат-боти використовуються для комунікації та функціональності:

- Комунікаційні - забезпечують комунікацію компанії з клієнтами. Вони можуть бути навіть примітивними - підтверджувати запит, який часто формулюється фразами-шаблонами, перенаправляти

на потрібного менеджера, тощо. Також можна використовувати рекламні чат-боти, які виконують функцію двостороннього каналу зв'язку з клієнтами: надають інформацію про послуги, відповідають на запитання щодо задалегідь побудованої архітектури комунікації.

- Функціональні – як заміна мобільних додатків. Зараз більшість програм дозволяють чатуватись, купувати, консультуватись, бронювати, шукати інформацію, перевіряти банківські транзакції тощо. Діапазон операцій може бути нескінченно розширений.

У підсумку можна сказати, що чат-боти - це корисний інструмент для покращення обслуговування клієнтів та ефективної роботи бізнесу. Вони допомагають забезпечити швидкий та зручний доступ до інформації, а також можуть значно зменшити витрати на підтримку клієнтів. Проте, для ефективного використання чат-бота в бізнесі важливо добре продумати його функціональні можливості та правильно налаштувати його роботу.

1.2 Опис предметного середовища

Наймання нового працівника - це важливий етап для будь-якої компанії. Від правильного вибору залежить успішність бізнесу. Проте, під час найму можуть виникнути різні проблеми, які варто уникати. Наприклад, не всі претенденти на вакансію є чесними та надійними співробітниками, іноді у їх резюме можуть бути неправдиві дані або порушення законодавства.

Однією з найбільш поширених проблем є недостатня інформація про кандидата. Деякі апліканти можуть замало розповісти про свої навички та досвід, що може привести до неправильного вибору на посаду. Крім того, резюме можуть містити неправдиву інформацію про кваліфікацію, дати та місця роботи.

Фактором, який ускладнює отримання інформації про кандидата, може бути його переїзд в інше місто. У такому випадку може бути досить складно знайти людей, які б могли дати детальну інформацію про претендента, його професійні та особисті якості. Крім того, навіть якщо такі люди знайдуться, їхнє свідчення може бути неповною або недостатньо достовірною.

Ще однією причиною, чому може бути недостатньо інформації про кандидата, є його бажання приховати минуле та розпочати нове життя з чистого аркуша. У такому випадку людина може не бажати розповідати про своє минуле місце роботи, особисте життя, професійні досягнення тощо. Це може ускладнити роботу рекрутера та спричинити недостатню інформованість роботодавця про кандидата.

Також проблемою може стати порушення законодавства. Деякі кандидати можуть мати судимість або інші правопорушення, які можуть стати перешкодою для прийняття на роботу.

Іншою серйозною проблемою може стати наявність кандидатом досвіду роботи в банкрутованих компаніях або компаніях з проблемами фінансового стану. Такі факти можуть вказувати на можливі фінансові ризики для компанії та негативно позначитися на її репутації.

Однак, існує рішення для уникнення цих проблем - проведення

скринінгу кандидатів на працевлаштування. Скринінг дозволяє перевірити інформацію, що надається в резюме, а також провести ретельну перевірку на наявність судимостей, проблем з фінансовим станом компанії та наявність в розшукових системах.

Скринінг може включати перевірку документів, звернення до джерел відкритої інформації, перевірку соціальних мереж та інші методи. Цей процес допоможе виявити недостовірну інформацію в резюме, порушення законодавства та інші негативні фактори, що можуть стати перешкодою для прийняття на роботу.

Ще однією перевагою перевірки є можливість встановлення додаткової інформації про кандидата, яку можна використати для прийняття рішення щодо його працевлаштування. Наприклад, відомості про досвід роботи в компаніях-конкурентах або відгуки колег та попередніх роботодавців можуть допомогти визначити, чи відповідає кандидат потребам компанії.

Крім того, скринінг кандидатів дозволяє компанії відбирати не тільки фахівців з потрібною кваліфікацією, але й з відповідними цінностями та підходом до роботи. Наприклад, при проведенні скринінгу можна виявити, чи дотримуються кандидати етичних та професійних стандартів, чи готові вони працювати в команді та виконувати вимоги компанії.

Необхідно також зазначити, що скринінг кандидатів є необхідним етапом при працевлаштуванні особливо в тих сферах, де безпека та здоров'я працівників має велике значення, наприклад, в області медицини, фармації, будівництва, авіації тощо.

Щоб забезпечити максимально точну та об'єктивну оцінку кандидатів, скринінг може проводитись не тільки відділом кадрів компанії, але і за допомогою зовнішніх експертів або спеціалізованих компаній зі скринінгу кандидатів. Це дозволяє забезпечити більш об'єктивну та комплексну оцінку кандидатів, що підвищує ефективність процесу найму.

В залежності від сфери діяльності та потреб компанії, можуть використовуватись різні типи скринінгу.

Передпрацевий скринінг (Pre-employment Screening) є найбільш поширеним видом скринінгу кандидатів на роботу. Цей вид скринінгу дозволяє компанії перевірити інформацію, яку надав кандидат у своєму резюме або під час інтерв'ю, а також дізнатися про його минуле та досвід роботи.

Скринінг перед працевлаштування може включати перевірку освіти, досвіду роботи, професійних навичок, референсів та інших аспектів. Крім того, можуть проводитись перевірки на наявність судимостей, банкрутств, проблем з алкоголем та наркотиками, а також перевірки в соціальних мережах та інтернеті.

Скринінг здоров'я (Health Screening) є важливим етапом процесу найму, особливо в тих сферах, де безпека та здоров'я працівників має велике значення. Цей вид скринінгу дозволяє перевірити стан здоров'я кандидатів та визначити їх придатність для виконання певних робіт.

Перевірка здоров'я може бути важливою для компаній, які працюють у високоризикових галузях, таких як хімічна промисловість або медицина. Цей тип скринінгу може включати в себе перевірку наявності наркотиків, алкоголю, інших речовин у крові.

Дана перевірка також може включати фізичний огляд, тестування на наявність хронічних захворювань, туберкульозу, ВІЛ/СНІДу та інших захворювань, які можуть впливати на здоров'я працівника та його оточення.

Скринінг домашнього персоналу (Screening of household staff) є важливим етапом при виборі працівників для роботи в приватному будинку. Цей вид скринінгу дозволяє перевірити наявність необхідних навичок та здібностей, а також переконатися в надійності та безпеці кандидата. Наприклад, якщо вам потрібен догляд за старшими людьми, важливо знати, чи має кандидат необхідні навички і досвід у цій сфері. Якщо вам потрібна допомога в господарстві, то важливо переконатися, що кандидат має відповідні знання та навички.

Це може включати перевірку досвіду роботи, референсів, професійних

навичок та здібностей, а також перевірку наявності судимостей, проблем з алкоголем та наркотиками. Крім того, можуть проводитись перевірки на наявність алергій, інших медичних проблем та виконання тестів на знання протоколів безпеки та етики. Також скринінг персоналу може бути спрямований на перевірку досвіду роботи в приватних будинках, наявність рекомендацій від попередніх роботодавців, перевірку наявності судимостей та інших правопорушень.

Усі види скринінгу допомагають компанії знайти найбільш кваліфікованих та надійних кандидатів, а також забезпечити безпеку та успішність своєї діяльності. Вибір конкретного виду скринінгу залежить від потреб компанії та вимог до кандидатів.

Незалежно від типу скринінгу, який використовує компанія, важливо знати, що це може бути важливим інструментом для забезпечення безпеки та надійності працівників та клієнтів, а також для забезпечення високої якості роботи. Крім того, правильно проведений скринінг може допомогти уникнути ризику потенційних правопорушень або інших проблем, які можуть виникнути під час роботи.

1.3 Порівняльний аналіз месенджерів з платформою чат-ботів

Месенджери – це програмні продукти для обміну повідомленнями між користувачами. Вони дозволяють відправляти текстові повідомлення, голосові та відео-файли, створювати групові чати та здійснювати відео- та голосові дзвінки. Месенджери стали невід'ємною частиною нашого життя та комунікації, оскільки дають можливість легко та швидко зв'язуватися з друзями, родиною, колегами та бізнес-партнерами.

Webhook - це механізм взаємодії між серверною та клієнтською сторонами, в якому клієнтська сторона відправляє HTTP POST запит на сервер при настанні певної події. Іншими словами, месенджер стає відповідальним за надсилання сповіщень серверному додатку (чат-боту) про виникнення подій. Зазвичай дані надсилаються у форматі JSON. Варто

відмітити, що для цього механізму зв'язку потрібна додаткова автентифікація. Автентифікацію можна здійснити за допомогою одного або кількох наступних методів:

- Зазвичай платформи чат-ботів надають список IP-адрес, з яких дозволено отримувати запити. Ці IP-адреси потрібно додати до білого списку мережевого фільтра, а запити з інших IP-адрес потрібно заборонити.
- Платформи чат-ботів часто надають унікальний секретний токен для кожного бота. Цей токен передається в окремій структурі при кожному запиті.
- Встановлення з'єднання може відбуватися за допомогою протоколу TLS, що забезпечує передачу даних по зашифрованому каналу.
- Взаємна автентифікація клієнтської та серверної сторін відбувається за допомогою SSL-сертифікатів.

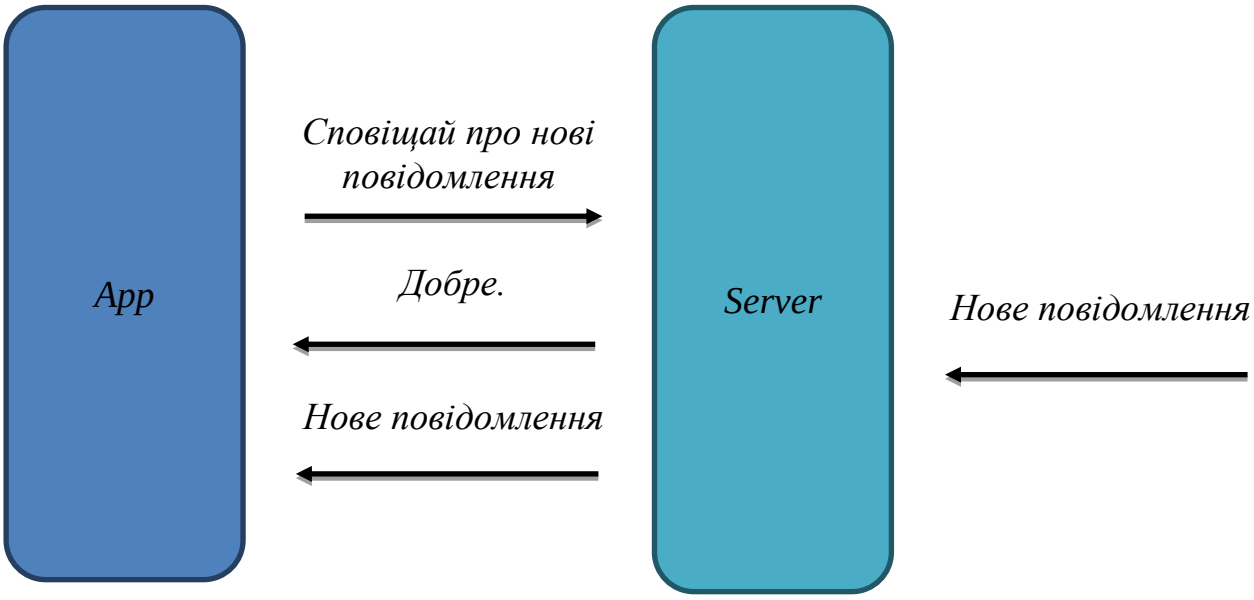

Рис.1.1. Візуалізація механізму Webhook

Polling - це механізм взаємодії між серверною та клієнтською сторонами, при якому серверна сторона відправляє HTTP POST запит на клієнтську сторону та очікує відповіді.

Установлене з'єднання залишається активним до тих пір, поки не відбудеться яка-небудь подія. При настанні події, клієнтська сторона відправляє відповідь на запит. Ця процедура повторюється. Дані передаються у форматі JSON або URL query string. Автентифікація при такому механізмі зв'язку зазвичай здійснюється за допомогою унікального токену чат-бота, який передається при кожному запиті у окремій структурі.

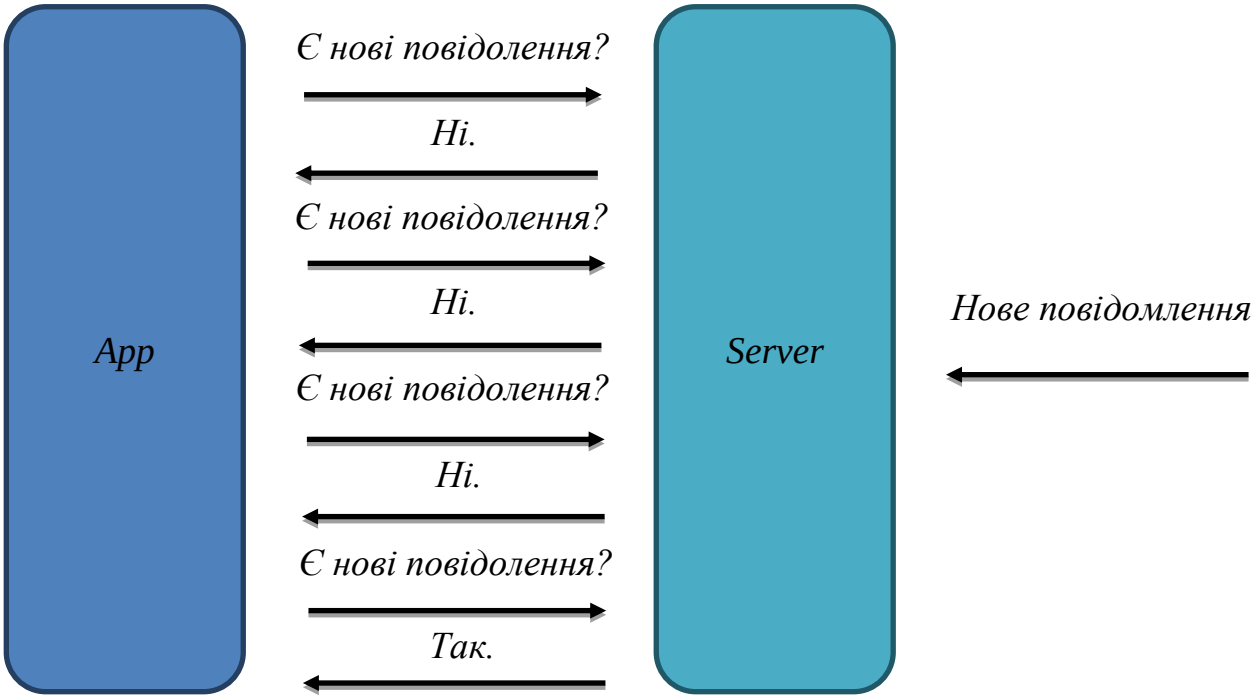

Рис.1.3. Візуалізація механізму Polling

У контексті дипломної роботи необхідно проаналізувати та обрати один із сучасних месенджерів з підтримкою платформи чат-ботів. Для аналізу оберемо наступні месенджери: Facebook, Telegram, Skype та Viber.

Параметри для аналізу наступні: форма використання, доступи, інтерфейси, механізми зв’язку, протоколи передачі даних, формати передачі даних, основні функціональні можливості.

В табл. 1.1 представлена порівняльна інформація популярних месенджерів з підтримкою платформи чат-ботів.

Таблиця 1.1

Порівняльна характеристика платформ для чат-ботів популярних месенджерів.

	Facebook	***Telegram***	***Skype***	***Viber***
Форма використання	Умовно безоплатна	Умовно безоплатна	Умовно безоплатна	Умовно безоплатна
Доступи	Доданий до групи, за підпискою, вбудований в діалог	Доданий до групи, за підпискою, вбудований в діалог	Доданий до групи, за підпискою	За підпискою, вбудований в діалог
Інтерфейси користувача	Текстовий, кнопковий, голосовий	Текстовий, кнопковий, голосовий	Текстовий, кнопковий, голосовий	Текстовий, кнопковий, голосовий
Механізми зв’язку	*webhook*	*webhook, polling*	*webhook*	*webhook*
Протоколи передачі даних	*https*			
Формати передачі даних	*JSON, multipart/form-data*	*JSON, URL query, multipart/form-data*	*JSON, multipart/form-data*	*JSON, multipart/form-data*
Функціональні можливості	Листування, опитування, передача файлів, магазин покупок, ігри	Листування, опитування, передача файлів, магазин покупок, ігри	Листування, передача файлів, магазин покупок, ігри	Листування, передача файлів, магазин покупок, ігри

На підставі проведеного аналізу можна зробити висновок, що для досягнення цілей дипломного проекту рекомендовано використовувати месенджер Telegram. Порівняно з іншими месенджерами, Telegram має наступні переваги:

- Вона підтримує механізм зв'язку Polling, що значно спрощує розробку та налагодження системи на початкових етапах.

- Широкий функціонал: Telegram надає багато корисних функцій, які можна використовувати при створенні чат-ботів. Це включає в себе можливість надсилання повідомлень, створення клавіатур, використання медіафайлів, роботу з групами та багато іншого. Цей широкий функціонал дозволяє створювати багатофункціональні та інтерактивні чат-боти.
- Велика база користувачів: Telegram має значну кількість активних користувачів по всьому світу. Це відкриває широкі можливості для розповсюдження та використання чат-ботів, оскільки їх можуть використовувати мільйони людей.
- Захист приватності: Telegram відомий своїми високими стандартами захисту приватності. Це особливо важливо для чат-ботів, які обробляють чутливі дані користувачів. Telegram забезпечує шифрування даних та інші заходи безпеки для збереження конфіденційності користувачів.

Також за допомогою Telegram-бота можна:

- Отримувати настроювані повідомлення та новини.
- Інтегруватися з різними зовнішніми сервісами, такими як Gmail, Wiki, YouTube, Github та інші.
- Приймати платежі від користувачів.
- Створювати корисні утиліти, наприклад, перекладачі з різних мов, інструменти для форматування тексту, нагадування про події та інше.
- Розробляти ігри на основі технології HTML5.

1.4 Вимоги до розроблюваної системи

Чат-бот Telegram для пошуку інформації про кандидата перед прийняттям на роботу повинен мати здатність приймати запити від користувачів, що стосуються пошуку інформації про кандидата перед прийняттям на роботу. Це має бути введення імені, фамілії, по батькові та додаткові дані, введення яких не є обов'язковим. Система має автоматично шукати відповідну інформацію про кандидата на базі відкритих джерел. Після аналізу система повинна здатися структурувати та відображати зібрану інформацію в зручному форматі для користувача.

Інтерфейс повинен бути простим та зрозумілим для користувачів. Він має забезпечувати легку взаємодію з чат-ботом та можливість вводити запити. Система повинна надавати можливість повторити запити користувачів в разі потреби. Знайдена інформація про кандидата повинна бути чітко та структуровано відображена. Важливо забезпечити зручність навігації. Результати пошуку повинні бути відображені в реальному часі, а чат-бот повинен швидко реагувати на запити користувачів.

Необхідно дотримуватись вимог законодавства про захист персональних даних та забезпечувати захист персональних даних користувачів.

Система повинна забезпечувати швидкий та ефективний пошук інформації про кандидата. Час очікування результатів запиту користувача повинен бути мінімальним. Система повинна бути оптимізована для використання ресурсів та мати високу швидкодію для забезпечення швидкого реагування.

Чат-бот повинен мати простий та зрозумілий інтерфейс, що не вимагає спеціальних навичок для використання. Важливо надавати достатньо деталей та контексту для зрозумілого розуміння знайденої інформації про кандидата. Взаємодія з чат-ботом повинна бути зручною за допомогою клавіатури, кнопок або інших інтерактивних елементів. Підтримка мовних команд або фраз має бути реалізована для зручної взаємодії з чат-ботом.

Цей чат-бот є частиною дипломної роботи та призначений для надання інформації про кандидатів на основі пошуку у відкритих джерелах. Важливо зазначити, що в рамках цієї роботи чат-бот не зберігає та не обробляє персональні дані користувачів. Уся отримана інформація є результатом пошуку у відкритих джерелах та не завжди може бути абсолютно точною.

Чат-бот надає зручний спосіб доступу до інформації про кандидатів, проте слід мати на увазі, що ця інформація є обмеженою та підлягає перевірці. Варто враховувати, що отримана інформація має використовуватися тільки як загальна довідка, а в разі потреби рекомендується перевіряти її за допомогою офіційних джерел або звертатися безпосередньо до кандидата для отримання достовірної інформації.

Розроблений чат-бот має бути зручним інструментом для отримання загальної інформації про кандидатів, але завжди слід підтверджувати дані через додаткові джерела та прямий контакт з кандидатом.

Ці вимоги необхідні для створення ефективного чат-боту Telegram для пошуку інформації про кандидата перед прийняттям на роботу. Виконання цих вимог забезпечить компанії надійний та зручний інструмент для відбору найкращих претендентів.

РОЗДІЛ 2. ВИБІР ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ СИСТЕМИ КЕРУВАННЯ ЧАТ-БОТОМ

2.1 Аналіз та вибір архітектури

Під час проектування програмного забезпечення (ПЗ) одним із найважливіших етапів є вибір архітектури. Архітектура ПЗ визначає структуру програми, розбиваючи її на слабкозв'язані та незалежні компоненти і описуючи взаємозв'язки та процеси передачі даних між ними. Головна мета архітектури ПЗ - забезпечити найкраще відповідність вимогам проекту.

З урахуванням сформованих функціональних вимог було вирішено розглянути наступні архітектурні шаблони та стилі:

Клієнт-серверна архітектура - це домінуюча концепція для створення розподілених мережевих застосунків, яка встановлює правила взаємодії та обміну даними між ними. Цей шаблон передбачає наступні компоненти:

Набір серверів, які надають інформацію за запитами.

Набір клієнтів, які використовують сервери та отримують від них інформацію.

Мережа, що забезпечує обмін інформацією між клієнтами та серверами.

Front-end та back-end - це архітектурний стиль, що розрізняє інтерфейс користувача (front-end) від серверної частини (back-end) системи. Front-end відповідає за відображення і взаємодію з користувачем, тоді як back-end оброблює логіку, виконує запити до бази даних та забезпечує обмін даними з фронт-ендом.

Монолітна/мікросервісна архітектура - це альтернативні підходи до побудови системи. Монолітна архітектура передбачає розробку ПЗ як одного цілого, де всі компоненти і функції знаходяться разом. Мікросервісна архітектура полягає у розбитті системи на невеликі незалежні сервіси, які взаємодіють між собою через API.

MVC - це шаблон проектування, який розділяє програму на три основні

компоненти: модель (model), представлення (view) та контролер (controller). Модель відповідає за обробку даних, представлення - за відображення інформації користувачу, а контролер - за керування взаємодією між моделлю та представленням.

У системі управління чат-ботом присутні три сторони:

- Telegram Bot API.
- Сервер з основною логікою роботи чат-бота.
- Клієнт.

У даному випадку, сервери Telegram виступають як серверна сторона відносно серверу з логікою чат-бота, сервер з логікою чат-бота виступає як серверна сторона відносно клієнтського веб-інтерфейсу, а сервери з логікою чат-бота виступають як серверна сторона відносно серверів Telegram та клієнтський веб-інтерфейс виступає як клієнтська сторона відносно серверів з логікою чат-бота.

Таким чином, для системи управління чат-ботом було обрано клієнт-серверну архітектуру, де взаємодія відбувається через сервери Telegram, сервер з логікою чат-бота та клієнтський веб-інтерфейс.

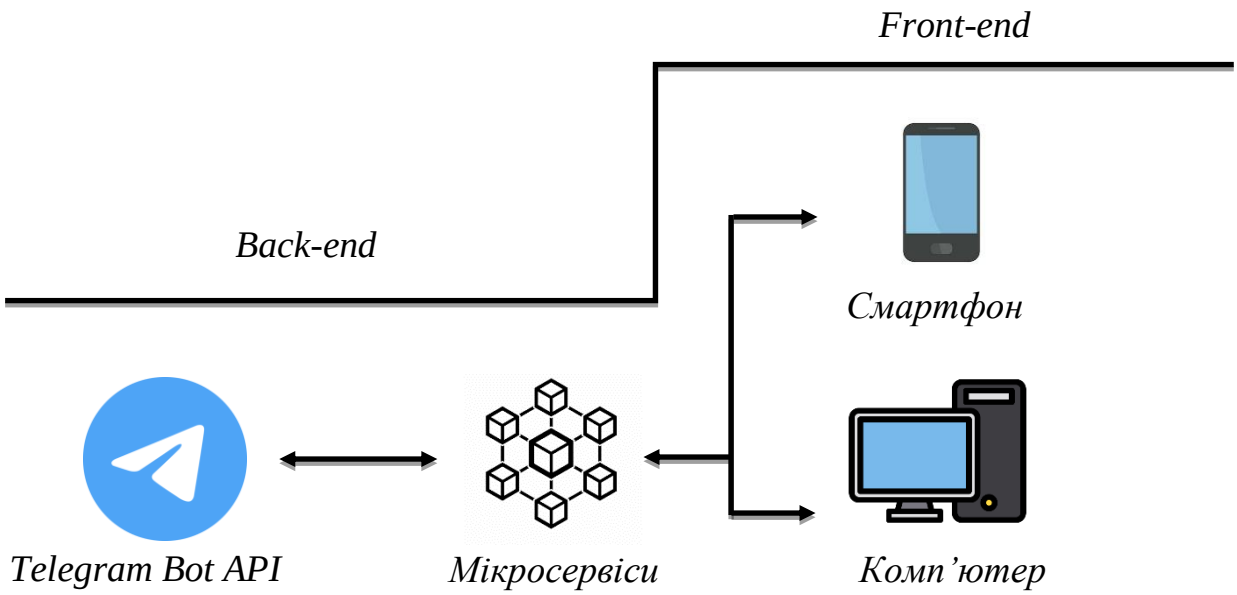

Рис.2.1. Клієнт-серверний погляд на архітектуру системи

При розробці великих систем або систем з великою кількістю слабозв'язаних компонентів, вибір архітектури має вирішальне значення. За останні роки сформувалися два основні підходи: монолітна архітектура і мікросервісна архітектура.

Монолітна архітектура полягає в тому, що всі компоненти додатку знаходяться в межах одного процесу операційної системи і обмінюються даними за допомогою внутрішніх інтерфейсів. Переваги використання монолітної архітектури включають спрощений процес розробки, розгортання та масштабування додатку. Однак, зі зростанням складності та розміру ПЗ, монолітна архітектура набуває недоліків, таких як складність у підтримці, розумінні та модифікаціях при великій кодовій базі, сповільнення швидкості роботи та унеможливлення частих оновлень додатку.

Мікросервісна архітектура, натомість, передбачає розбиття додатку на невеликі компоненти, які працюють незалежно один від одного в окремих процесах операційної системи. Взаємодія між компонентами базується на стандартизованих протоколах RPI або обміну повідомленнями. Застосування мікросервісної архітектури має такі переваги, як спрощення розробки, тестування та підтримки компонентів, покращення надійності системи та можливість заміни технологій без значного впливу на роботу додатку. Однак, використання мікросервісної архітектури вимагає створення механізмів обміну даними між компонентами та ускладнює процедуру розгортання додатку.

З урахуванням завдання дипломної роботи та аналізу недоліків монолітної архітектури, можна зробити висновок, що використання мікросервісної архітектури принесе наступні переваги: незалежна розробка компонентів, можливість простішого масштабування та використання сучасних технологій з легкою заміною за необхідності.

Шаблон розробки ПЗ Model-View-Controller (MVC) визначає структуру додатку, де функціональність поділена на три основні компоненти: Модель, Вид і Контролер.

Модель відповідає за зберігання даних та їх оновлення відповідно до дій користувача. Вона представляє собою репрезентацію даних і бізнес-логіку, і може бути оновлена через взаємодію з елементами управління, що контролюються Контролером.

Вид відповідає за відображення даних (стану моделі) користувачу. Він оновлюється лише при зміні моделі. Вид може бути графічним інтерфейсом, веб-сторінкою або будь-яким іншим способом відображення інформації користувачу.

Контролер виконує обробку дій користувача і взаємодіє з Моделлю та Видом. Він зазвичай використовує стандартні елементи управління, такі як кнопки, текстові поля, перемикачі тощо, для взаємодії з користувачем. Контролер приймає вхідні дані від користувача і виконує відповідні дії, щоб оновити Модель або Вид.

Застосування шаблону MVC дозволяє розділити логіку додатку на логічно пов'язані компоненти, що сприяє полегшенню розробки, підтримці та розширенню коду. Кожен компонент виконує свою специфічну роль, що сприяє розподілу обов'язків та забезпечує більшу гнучкість у розвитку програмного забезпечення.

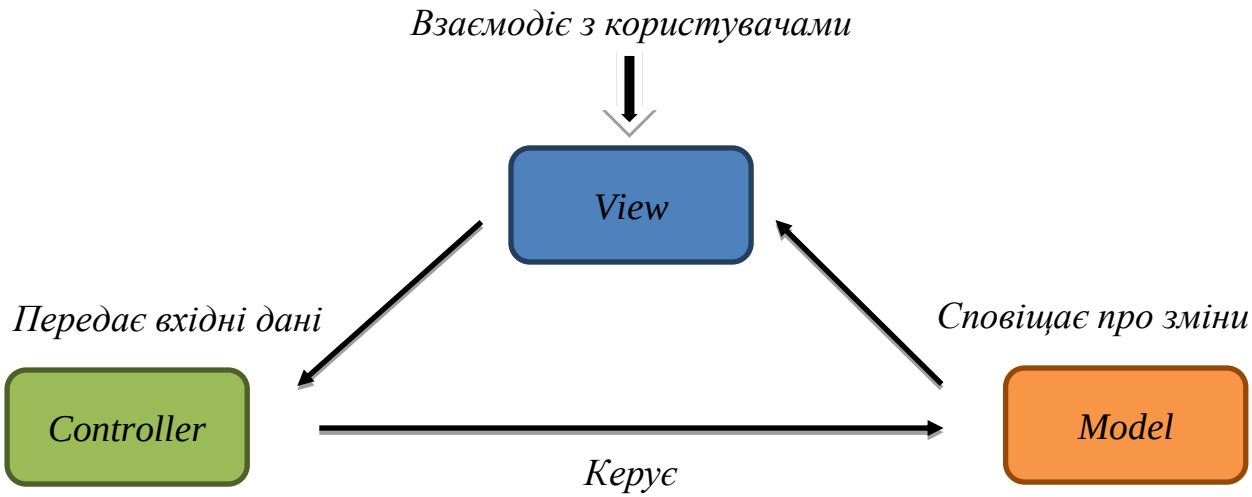

Рис.2.2. Діаграма дій MVC

2.2 Вибір мови програмування PHP

Для реалізації цього проекту необхідно вибрати мову програмування, оскільки неможливо створити такого рівня бота без програмування.

- Наявність та підтримка IDE.
    - Java: Має потужні та популярні IDE, такі як IntelliJ IDEA та Eclipse.
    - Python: Має різноманітні IDE, включаючи PyCharm та Jupyter Notebook.
    - JavaScript: Має IDE, такі як Visual Studio Code та WebStorm.
    - PHP: Має ряд популярних IDE, таких як PhpStorm та Visual Studio Code з підтримкою PHP.
- Зручність супроводження та відлагодження додатків.
    - Java: Має потужні відлагоджувачі та інструменти для профілювання коду, що допомагає при супроводженні та відлагодженні.
    - Python: Має простий синтаксис та вбудовані інструменти для відлагодження, що полегшує розуміння та налагодження коду.
    - JavaScript: Має потужні інструменти для відлагодження, такі як Chrome DevTools, що полегшують супроводження та відлагодження.
    - PHP: Має розширення та інструменти, такі як Xdebug, що сприяють зручному відлагодженню та супроводженню.
- Вартість розробки ПЗ.
    - Java, Python, JavaScript та PHP є мовами з відкритим вихідним кодом, що дозволяє уникнути витрат на ліцензії.
    - Усі ці мови мають безліч безкоштовних розробничих інструментів та бібліотек, що зменшує витрати на розробку ПЗ.
- Чіткість та зрозумілість синтаксису мови.

- Java: Має суворий синтаксис з вимогами до типів, що може бути викликом для початківців.
- Python: Має простий та зрозумілий синтаксис, що полегшує читання та розуміння коду.
- JavaScript: Має гнучкий синтаксис, але може бути дещо непередбачуваним через динамічну типізацію.
- PHP: Має простий та легко читаємий синтаксис, що сприяє розумінню та розробці коду.

- Можливість використання шаблонів, підходів та найкращих практик розробки *ПЗ.* Java, Python, JavaScript та PHP мають широку спільноту розробників, яка активно ділиться шаблонами, підходами та найкращими практиками розробки ПЗ.

Для створення чат-боту було вирішено використовувати PHP, оскільки є відмінним вибором для розробки чат-боту.

PHP (PHP: Hypertext Preprocessor) - це мова скриптового програмування на стороні сервера, яка спочатку була розроблена для веб-розробки. Вона була створена Расмусом Лердорфом у 1994 році і з того часу перетворилася на потужну та широко використовувану мову програмування.

Спочатку PHP означало "Personal Home Page" (особиста домашня сторінка), оскільки Лердорф створив його для підтримки свого особистого веб-сайту. З плином часу воно набуло популярності і було значно розширене, ставши відкритим проектом. Група PHP прийняла його розвиток, і акронім PHP був змінений на "PHP: Hypertext Preprocessor", щоб відобразити його більш широкий функціонал як мови скриптування загального призначення.

PHP переважно використовується для веб-розробки на стороні сервера. Це універсальна мова, яка може вбудовуватися в HTML-код, що дозволяє розробникам створювати динамічні веб-сторінки та складні веб-додатки. PHP надає широкий спектр функцій і можливостей, включаючи інтеграцію з базами даних, обробку файлів, обробку форм тощо.

Однією з основних переваг PHP є його сумісність з різними базами даних, такими як MySQL, PostgreSQL та Oracle. Це робить його відмінним вибором для розробки веб-сайтів та додатків, які використовують бази даних. Крім того, PHP підтримує різні протоколи, такі як HTTP, FTP та IMAP, що дозволяє розробникам взаємодіяти з різними системами та сервісами.

Порівняно з клієнтськими мовами, такими як JavaScript, PHP виконується на сервері і генерує HTML, який відправляється веб-браузеру клієнта. Це дозволяє PHP ефективно та безпечно виконувати завдання на стороні сервера, такі як доступ до баз даних, обробку форм та складні обчислення. У свою чергу, JavaScript використовується переважно для клієнтської взаємодії та динамічного контенту.

PHP часто порівнюють з іншими мовами на стороні сервера, такими як Python, Ruby та Java. Кожна мова має свої переваги і сфери застосування, але простота та легкість використання PHP роблять його особливо підходящим для веб-розробки. Велике співтовариство розробників PHP та обширна документація надають розробникам величезну кількість ресурсів і підтримку.

PHP є хорошим вибором для розробки чат-ботів завдяки своїй універсальності та спрямованості на веб. Причини, чому PHP добре підходить для створення чат-ботів:

- Інтеграція з вебом. PHP легко інтегрується з веб-технологіями, що дозволяє розробникам включати функціонал чат-ботів у існуючі веб-сайти або веб-додатки.
- Обробка на стороні сервера. Чат-ботам часто потрібна обробка на стороні сервера для виконання складної логіки, операцій з базами даних та інтеграції зовнішніх API. Серверна можливість PHP робить його потужною мовою для реалізації функціоналу чат-ботів.
- Підключення до баз даних. PHP має відмінні можливості для роботи з базами даних, що дозволяє чат-ботам ефективно зберігати та отримувати інформацію з баз даних. Це важливо для керування

взаємодіями з користувачем та збереження контексту розмови.

- Велике співтовариство та бібліотеки. PHP має велику та активну спільноту розробників, що пропонує великий вибір бібліотек, фреймворків та ресурсів, спеціально призначених для веб-розробки. Ці ресурси можуть значно прискорити процес розробки чат-ботів.

Отже, PHP є гнучкою та ефективною мовою програмування, яка підходить для розробки чат-боту. Вона має підтримку Telegram API, надає можливості для обробки запитів, зберігання даних, взаємодії з користувачем та багато іншого, що допомагає розробникам створювати потужні та інтерактивні чат-боти Telegram.

2.3 Вибір бази даних. MySQL

База даних (БД) - це структуроване сховище інформації, яке дозволяє зберігати, керувати та організовувати великі обсяги даних. Вона використовується для ефективного зберігання, опрацювання, оновлення та відновлення даних за допомогою спеціального програмного забезпечення.

Система управління базами даних (СУБД) - це програмне забезпечення, яке дозволяє створювати, керувати та маніпулювати базами даних. Вона забезпечує можливість створювати таблиці для зберігання даних, виконувати операції вставки, вибірки, оновлення та видалення даних, забезпечує механізми безпеки, контролю цілісності даних та багато іншого.

СУБД виконують різноманітні задачі, серед яких:

- Зберігання даних: СУБД дозволяють зберігати великі обсяги даних у структурованому форматі. Дані можуть бути збережені у вигляді таблиць, файлів, об'єктів або інших форматів в залежності від типу СУБД.
- Організація даних: СУБД надають можливість організації даних у логічні групи або таблиці. Це дозволяє ефективно організовувати, управляти та шукати дані, забезпечуючи швидкий доступ до необхідної

інформації.

- Маніпулювання даними: СУБД дозволяють виконувати різноманітні операції над даними, такі як додавання нових записів, вибірка конкретних даних за певними критеріями, оновлення існуючих даних або видалення непотрібних записів. Це дозволяє здійснювати широкий спектр операцій для керування інформацією.
- Забезпечення безпеки: СУБД надають механізми безпеки для захисту даних від несанкціонованого доступу. Це включає управління правами доступу до даних, шифрування, журналювання та інші методи захисту інформації.

Найпопулярнішими СУБД є:

1. MySQL: Відкрите програмне забезпечення, яке широко використовується для веб-додатків та підтримує багато мов програмування.
2. Oracle Database: Повнофункціональна комерційна СУБД, яка підтримує широкий спектр бізнес-застосунків.
3. Microsoft SQL Server: СУБД від Microsoft, яка надає повноцінну підтримку для Windows-платформи та інтеграцію з іншими продуктами Microsoft.
4. PostgreSQL: Відкрите програмне забезпечення з високим рівнем розширюваності та підтримки стандартів SQL.
5. MongoDB: Орієнтована на документи NoSQL СУБД, яка дозволяє зберігати неструктуровані дані у форматі JSON.

Ці СУБД є лише кількома з найпопулярніших варіантів, існує багато інших СУБД, які підходять для різних типів застосувань та вимог.

MySQL пропонує широкий набір функцій та переваг для розробки інтерактивних застосунків, включаючи створення Telegram чат-ботів. Ось кілька причин, чому MySQL може бути відмінним вибором для створення

Telegram чат-боту:

- Легкість використання: MySQL має простий та зрозумілий синтаксис SQL, що робить його легким у використанні та навчанні. Це дозволяє швидко створювати таблиці, виконувати запити та керувати даними в базі даних.
- Висока продуктивність: MySQL відомий своєю швидкодією та ефективністю. Він може обробляти великі обсяги даних та виконувати складні запити швидко. Це особливо важливо для Telegram чат-ботів, які можуть отримувати та обробляти великий потік повідомлень.
- Розширюваність: MySQL підтримує можливість горизонтального та вертикального масштабування, що дозволяє збільшувати потужність та продуктивність бази даних при зростанні обсягу даних або навантаження.
- Надійність та стабільність: MySQL відомий своєю надійністю та стабільністю. Він має добре розроблені механізми для забезпечення цілісності даних, резервного копіювання та відновлення, що робить його популярним вибором для критичних застосунків, таких як Telegram чат-боти.
- Інтеграція з іншими технологіями: MySQL підтримує широкий спектр мов програмування та інших технологій, що дозволяє легко інтегрувати його з іншими компонентами вашого Telegram чат-боту, наприклад, серверними фреймворками чи іншими сервісами.

Також великою перевагою є швидке конвертування XML в MySQL, яка є важливою функцією для багатьох розробників, особливо коли потрібно ефективно переміщувати дані між різними системами. XML (розшифровується як "розширюваний мовний розмітка") є одним з найпоширеніших форматів для зберігання та обміну даними. MySQL, з іншого боку, є потужною реляційною базою даних, яка широко

використовується веб-розробкою.

Для ефективної конвертації XML в MySQL існують кілька можливих підходів. Один з них - використання мови програмування, такої як PHP, з вбудованою підтримкою для роботи з XML та базами даних MySQL. PHP надає розширені функції та класи, які дозволяють зчитувати XML-файли, аналізувати їх структуру та експортувати дані в MySQL з використанням відповідних запитів SQL.

Наприклад, використовуючи PHP, можна використовувати функції, такі як simplexml_load_file або DOMDocument::load, для розбору XML-документу. За допомогою цих функцій можна витягти дані з тегів XML та створити відповідні таблиці в базі даних MySQL. Після цього дані можна зберегти у відповідній формі, використовуючи SQL-запити INSERT або UPDATE.

Інший підхід - використання спеціалізованих інструментів, таких як ETL (екстракція, перетворення, завантаження) або middleware-програм, які спрощують конвертацію даних між різними форматами. Ці інструменти можуть мати вбудовану підтримку для XML та MySQL, що дозволяє легко налаштувати процес конвертації і автоматизувати його. В PHP можна використовувати бібліотеки, такі як SimpleXML або XMLReader, для роботи з XML-даними, а потім використовувати функції, які надаються MySQL-розширенням, для збереження цих даних в базі даних MySQL.

Загалом, MySQL є потужним, легким у використанні та надійним варіантом для створення Telegram чат-ботів. Він надає широкий функціонал та забезпечує продуктивність, що дозволяє створити ефективного та швидкодіючого бота для комунікації з користувачами на платформі Telegram.

2.4 Docker як інструмент розгортання та управління додатками

Docker - це інструмент для розгортання та управління додатками, який надає незалежне та відокремлене середовище виконання для програмного забезпечення. Він базується на технології контейнеризації, що дозволяє

упаковувати програми та їх залежності в контейнери, які можна легко переміщати та виконувати на різних платформах.

Docker є потужним і популярним засобом для контейнеризації програмного забезпечення, що дозволяє упаковувати програми та їх залежності відокремлено від інших ресурсів системи. Це дозволяє розробникам та інженерам забезпечувати швидку та надійну роботу програм, незалежно від конфігурації операційної системи, на якій вони запускаються.

Однією з ключових особливостей Docker є його здатність до створення контейнерів. Контейнери використовують віртуалізацію на рівні операційної системи, дозволяючи багатьом контейнерам запускатися на одній фізичній або віртуальній машині. Кожен контейнер містить усе необхідне для виконання програми, включаючи код, бібліотеки, середовище виконання та налаштування. Все це упаковується в один стандартизований пакет, який легко переносити та використовувати на будь-якому пристрої або сервері, що підтримує Docker.

Основна одиниця управління Docker - це Docker-контейнер. Кожен контейнер запускається зі своїм власним ізольованим середовищем, що дозволяє йому працювати незалежно від інших контейнерів. Docker надає широкий набір інструментів та командного рядка для створення, управління та масштабування контейнерів.

Іншою важливою концепцією Docker є Docker-образи. Образи використовуються для створення контейнерів і містять у собі весь необхідний код, залежності та налаштування. Вони будуються на основі Dockerfile - текстового файлу, що містить інструкції для збірки образу. Цей підхід дозволяє створювати образи програм з повторюваними і контрольованими умовами, що дозволяє легко розгортати програми в різних середовищах.

Docker також надає інструменти для оркестрації контейнерів, такі як Docker Compose і Docker Swarm. Docker Compose дозволяє описати багатоконтейнерні додатки у файлі конфігурації та запустити їх однією

командою. Docker Swarm, з свого боку, забезпечує можливість створювати та управляти кластерами контейнерів, що дає можливість масштабувати додатки та забезпечити високу доступність.

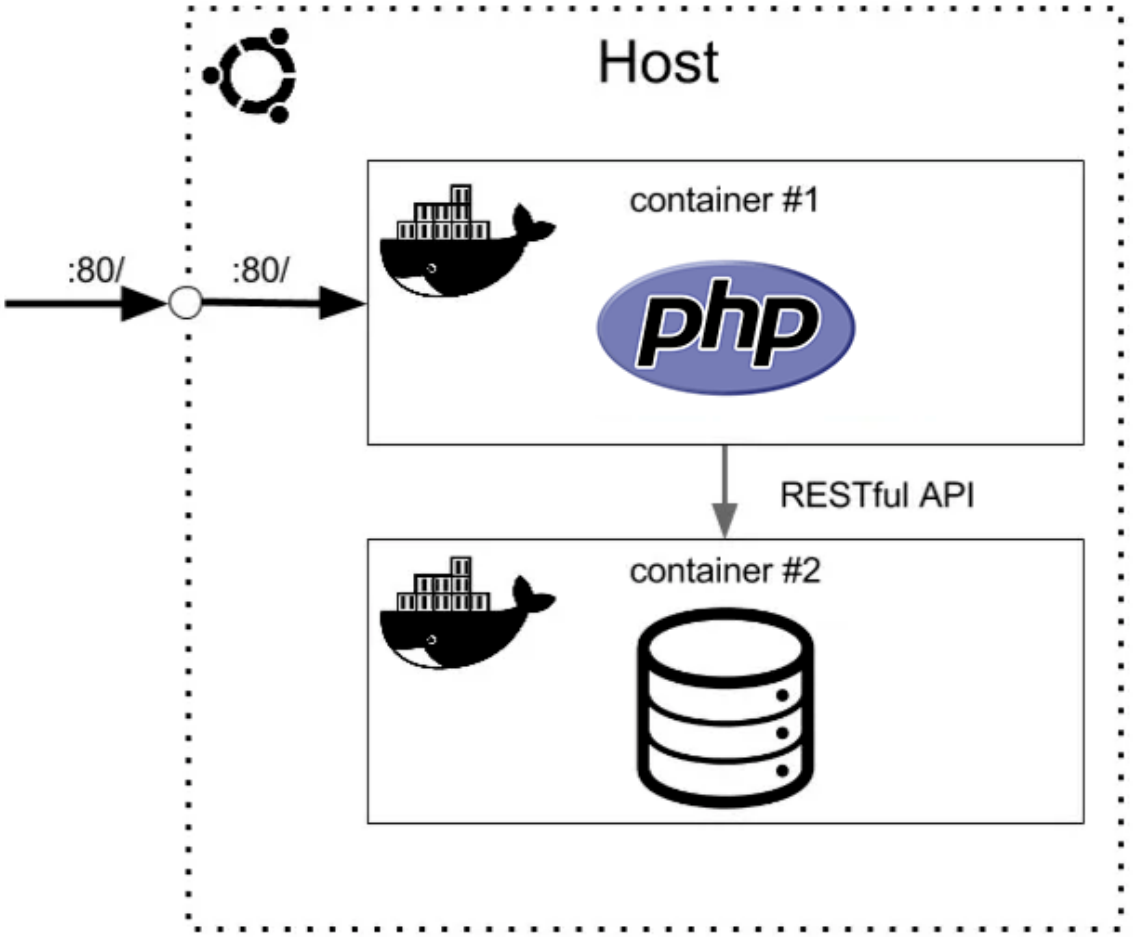

Рис.2.3. Схема взаємодії Docker-контейнерів

Загалом, Docker забезпечує простий та ефективний спосіб упакування, доставки та запуску програмного забезпечення у контейнерах. Він дозволяє забезпечити незалежність від операційної системи та конфігурації пристроїв, спрощує процес розгортання додатків та полегшує масштабування і управління контейнерами. Docker є незамінним інструментом для розробників та інженерів, які хочуть зробити свої додатки більш переносними, ефективними та легкими у використанні.

2.5 Telegram API

API (Application Programming Interface) - це набір правил, протоколів та інструментів, які дозволяють різним програмним додаткам взаємодіяти між собою. API визначає, як один програмний компонент може взаємодіяти з іншими, як обмінюватися даними та як виконувати певні функції. Це набір правил, які дозволяють програмістам використовувати функціональність, яку надає інший компонент, без необхідності знати подробиці реалізації цього компонента.

API можна уявити як посередника, що забезпечує взаємодію між двома програмами. Воно визначає, які дані можна передавати, які дії можна виконувати та які відповіді можна отримати. За допомогою API програмні компоненти можуть запитувати дані, надсилати дані, виконувати певні операції та отримувати результати.

Web API - це конкретна реалізація API, яка дозволяє веб-додаткам взаємодіяти з іншими веб-додатками, сервісами або платформами за допомогою мережі Інтернет. Web API використовується для створення інтерфейсу між веб-додатками, щоб вони могли обмінюватися даними та функціональністю.

API може бути реалізований різними способами. Один з найпоширеніших типів API - це веб-сервіси, які використовують протокол HTTP для комунікації між клієнтом і сервером. Веб-сервіси можуть бути засновані на REST (Representational State Transfer) або SOAP (Simple Object Access Protocol) архітектурі.

RESTful API - це стиль побудови веб-сервісів, який базується на принципах REST. Він використовує HTTP-методи, такі як GET, POST, PUT і DELETE, для виконання операцій з ресурсами. Клієнти здійснюють запити до сервера, передаючи URL-адресу ресурсу та виконуючи відповідний метод HTTP.

SOAP API - це протокол, який дозволяє обмінюватися повідомленнями між різними програмами. Він використовує XML для структуризації даних і

може використовувати різні протоколи комунікації, такі як HTTP, SMTP або JMS (Java Message Service).

У межах дипломної роботи використовується саме RESTful API.

Web API надає можливість веб-додаткам отримувати доступ до функціональності та ресурсів інших веб-додатків або сервісів. Це можуть бути операції з базами даних, обробка платежів, робота зі сторонніми сервісами, отримання даних, керування користувачами та багато іншого. Web API дозволяє розширювати функціональність веб-додатків, інтегрувати їх з іншими системами та платформами, створювати сторонні додатки, які використовують функціональність веб-додатків через API.

Web API є важливою складовою сучасної веб-розробки, оскільки дозволяє створювати веб-додатки, які взаємодіють з іншими додатками та сервісами, обмінюючись даними та функціональністю. Це дає змогу побудувати розширені екосистеми програмного забезпечення та стимулює розвиток інноваційних додатків і сервісів, які взаємодіють з різними джерелами даних та функціональністю через мережу Інтернет.

Telegram API та Telegram Bot API - це набір інструментів, розроблених компанією Telegram, що дозволяють розробникам створювати різноманітні додатки та ботів для платформи Telegram.

Telegram API надає доступ до основних функцій Telegram Messenger. З його допомогою розробники можуть взаємодіяти з Telegram, відправляти та отримувати повідомлення, зображення, відео, аудіо, документи, створювати групи, канали та багато іншого. Цей API забезпечує широкі можливості для створення різноманітних додатків, які використовують Telegram як основну платформу комунікації.

Telegram Bot API - це розширений набір інструментів, спеціально розроблений для створення та управління ботами в Telegram. Завдяки Telegram Bot API, розробники можуть створювати ботів з різноманітними функціями: розсилати сповіщення, відповідати на повідомлення, виконувати команди, працювати з клавіатурами та багато іншого.

API Telegram і Telegram Bot API базуються на протоколі MTProto, який забезпечує безпеку та шифрування даних під час передачі. Це означає, що всі дані, що передаються через API, захищені від несанкціонованого доступу.

Для використання Telegram API та Telegram Bot API розробникам необхідно отримати доступ до API-ключів, які є унікальними ідентифікаторами для кожного додатку або бота. Ці ключі використовуються для аутентифікації та авторизації додатків та ботів під час взаємодії з Telegram API.

Telegram API та Telegram Bot API використовуються в різноманітних сферах, таких як розробка месенджерів, створення ігор, розсилка сповіщень, розробка чат-ботів для бізнесу та багато іншого. Ці інструменти надають зручний спосіб взаємодії з користувачами Telegram та розширюють можливості розробників у побудові інноваційних додатків на цій платформі.

Більш конкретно, розглянемо деякі можливості Telegram API та Telegram Bot API:

- Отримання та відправка повідомлень: Розробники можуть отримувати повідомлення з Telegram та відправляти їх до користувачів чи груп. Це дозволяє створювати різноманітні сповіщення, повідомлення про події та інше.
- Взаємодія з мультимедійними даними: API дозволяє обробляти та передавати зображення, відео, аудіофайли та документи. Розробники можуть створювати додатки, які спілкуються з користувачами через медіафайли.
- Робота з клавіатурами та кнопками: Telegram API дозволяє створювати інтерактивні клавіатури та кнопки, що спрощують взаємодію з користувачем. Це дозволяє створювати зручний та інтуїтивно зрозумілий інтерфейс для користувачів бота.
- Створення груп та каналів: За допомогою Telegram API розробники можуть створювати групи та канали, додавати

користувачів та керувати їх учасниками. Це дозволяє створювати спільноти та комунікативні канали для різноманітних цілей.

- Розробка чат-ботів: Telegram Bot API надає можливість створювати чат-ботів, які автоматизують взаємодію з користувачами. Чат-боти можуть відповідати на повідомлення, виконувати команди, надавати інформацію та багато іншого. Це дозволяє створювати чат-ботів для підтримки клієнтів, автоматизації рутинних завдань та багато іншого.

2.6 Платформа для розробки боту

Написання боту здійснювалось в середовищі програмування Visual Studio Code 2023.

Visual Studio Code (VSCode) - це легкий, але потужний редактор вихідних кодів, розроблений компанією Microsoft. Він завдяки своїм розширеним можливостям, гнучкості та широкому спектру підтримуваних мов програмування завоював велику популярність серед розробників. VSCode доступний для Windows, macOS та Linux, що робить його універсальним вибором для розробників на різних платформах.

Одним з основних переваг VSCode є його можливість налаштування та модульність. Користувачі можуть покращити свій досвід програмування, встановлюючи різноманітні розширення з майданчика VSCode. Ці розширення надають додаткові функціональні можливості, такі як підтримка мов програмування, інструменти для налагодження коду та інтеграцію з різними фреймворками та сервісами. З живою спільнотою, постійно розробляються нові розширення, завдяки чому користувачі можуть налаштувати своє середовище редагування, відповідно до своїх потреб.

VSCode пропонує зручний інтерфейс з мінімалістичним дизайном, що дозволяє розробникам зосередитися на своєму коді без зайвих відволікань. Редактор надає основні функції, такі як підсвічування синтаксису, розумне автодоповнення коду та форматування, які сприяють написанню чистого та

ефективного коду. Він також включає потужну вбудовану підтримку Git, що дозволяє розробникам легко керувати контролем версій безпосередньо з редактора.

Ще одна помітна особливість VSCode - це його розширені можливості налагодження. Розробники можуть встановлювати точки зупину, крокувати через код, перевіряти значення змінних та аналізувати стек викликів за допомогою інтегрованих засобів налагодження. Це полегшує виявлення та усунення помилок у коді під час процесу розробки.

VSCode також підтримує інтегровану функціональність терміналу, що дозволяє розробникам виконувати команди та запускати скрипти без переходу до зовнішніх термінальних програм. Ця безшовна інтеграція спрощує робочий процес розробки та підвищує продуктивність.

Крім того, VSCode надає надійну підтримку спільної роботи та віддаленої розробки. Він пропонує функції, такі як Live Share, що дозволяють кільком розробникам одночасно працювати над тим самим кодом, що спрощує спільну роботу та пошук та усунення помилок у коді. Крім того, розширення VSCode Remote Development дозволяють розробникам працювати на віддалених серверах або контейнерах, забезпечуючи послідовний та надійний процес розробки незалежно від базової інфраструктури.

Telegram - це сучасна месенджерська програма, яка пропонує безпечну та швидку комунікацію на основі зашифрованих повідомлень. Розроблена компанією Telegram Messenger LLP, ця платформа надає користувачам можливість обмінюватися повідомленнями, фотографіями, відео та іншими мультимедійними матеріалами.

Одним з ключових переваг Telegram є його фокус на безпеці та конфіденційності. Всі повідомлення, відправлені через Telegram, захищені криптографічними протоколами, що гарантує їх захищений та приватний обмін. Крім того, Telegram пропонує функцію "Секретні чати", в яких повідомлення можуть автоматично самознищуватися після певного часу, а

також видалення повідомлень з обох сторін спілкування.

Telegram володіє потужними можливостями, які полегшують комунікацію та організацію спільних проектів. Наприклад, користувачі можуть створювати групи чату з до 200 000 учасників, що дозволяє об'єднувати великі спільноти та робочі групи. Також доступна можливість створювати канали, які дозволяють передавати інформацію великій аудиторії без обмежень кількості учасників.

Telegram пропонує широкий набір функцій, що полегшують повсякденне використання. До них належать групові голосові та відеовиклики, вбудований редактор фотографій та відео, масштабніша спільна робота над документами та файли, стікери та емодзі, боти для автоматизації завдань та інші.

Окрім цього, Telegram є кросплатформеним месенджером, доступним для різних операційних систем, включаючи Android, iOS, Windows, macOS та Linux. Це дозволяє користувачам обмінюватися повідомленнями та використовувати всі функції Telegram на різних пристроях, забезпечуючи зручну та синхронізовану комунікацію.

РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

3.1 Розробка алгоритму взаємодії з чат-ботом

Розробка алгоритмів для сценаріїв взаємодії з чат-ботом є важливою складовою процесу створення ефективного та корисного чат-бота. Ці алгоритми визначають, як бот буде відповідати на запити користувачів, як буде розпізнавати їх наміри та як буде керувати сценаріями діалогу.

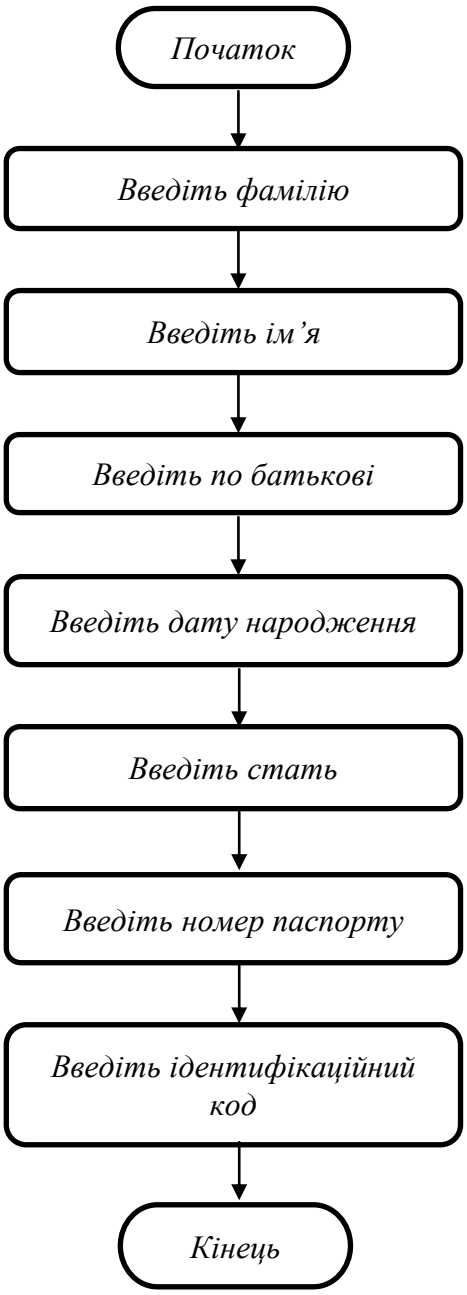

Рис.1.1 Схема порядку введення необхідних даних

В рамках розробки чат-бота застосовується алгоритм обмеженої роботи з кнопково-текстовим форматом взаємодії. Це означає, що чат-бот має певні обмеження щодо можливих запитів та відповідей, а також пропонує користувачеві набір кнопок для спрощення комунікації.

Розробка такого чат-бота передбачає створення набору кнопок, які представляють доступні опції для користувача. Користувач може вибрати кнопку, яка відповідає його запиту або наміру. Це дозволяє спростити процес взаємодії та забезпечити більш точні та зрозумілі відповіді.

Основою алгоритму роботи чат-бота є обробка введення користувача. При отриманні вхідного повідомлення, чат-бот визначає, чи відповідає воно одній з наявних кнопок, чи містить текстове повідомлення. В залежності від цього, виконується відповідна логіка обробки.

Якщо користувач вибирає кнопку, чат-бот здійснює певну дію, пов'язану з цією кнопкою. Наприклад, це може бути запит до бази даних для отримання інформації, виконання певного завдання або перехід до іншого меню.

Якщо користувач надсилає текстове повідомлення, чат-бот спробує розпізнати зміст повідомлення.

У розробленому чат-боті з кнопково-текстовим форматом взаємодії також використовуються кнопки, які дозволяють користувачу пропустити необов'язкові поля та розпочати заново.

Кнопка "Пропустити" надає можливість користувачеві пропустити певні необов'язкові поля або кроки в процесі взаємодії з чат-ботом. Наприклад, у разі заповнення форми, якщо деякі поля не є обов'язковими для заповнення, користувач може використати кнопку "Пропустити", щоб перейти до наступного кроку без їх заповнення.

Крім того, кнопка "Розпочати заново" надає можливість користувачу скинути поточний стан чат-бота і розпочати взаємодію спочатку. Це може бути корисно в ситуаціях, коли користувач хоче змінити свої відповіді або розпочати процес заново для отримання більш точних результатів.

В рамках розробки чат-бота, що базується на заздалегідь визначених алгоритмах взаємодії з користувачами, акцент робиться на передбаченні можливих сценаріїв та моделюванні поведінки системи. У цьому пункті буде описано процес моделювання поведінки системи для різних сценаріїв.

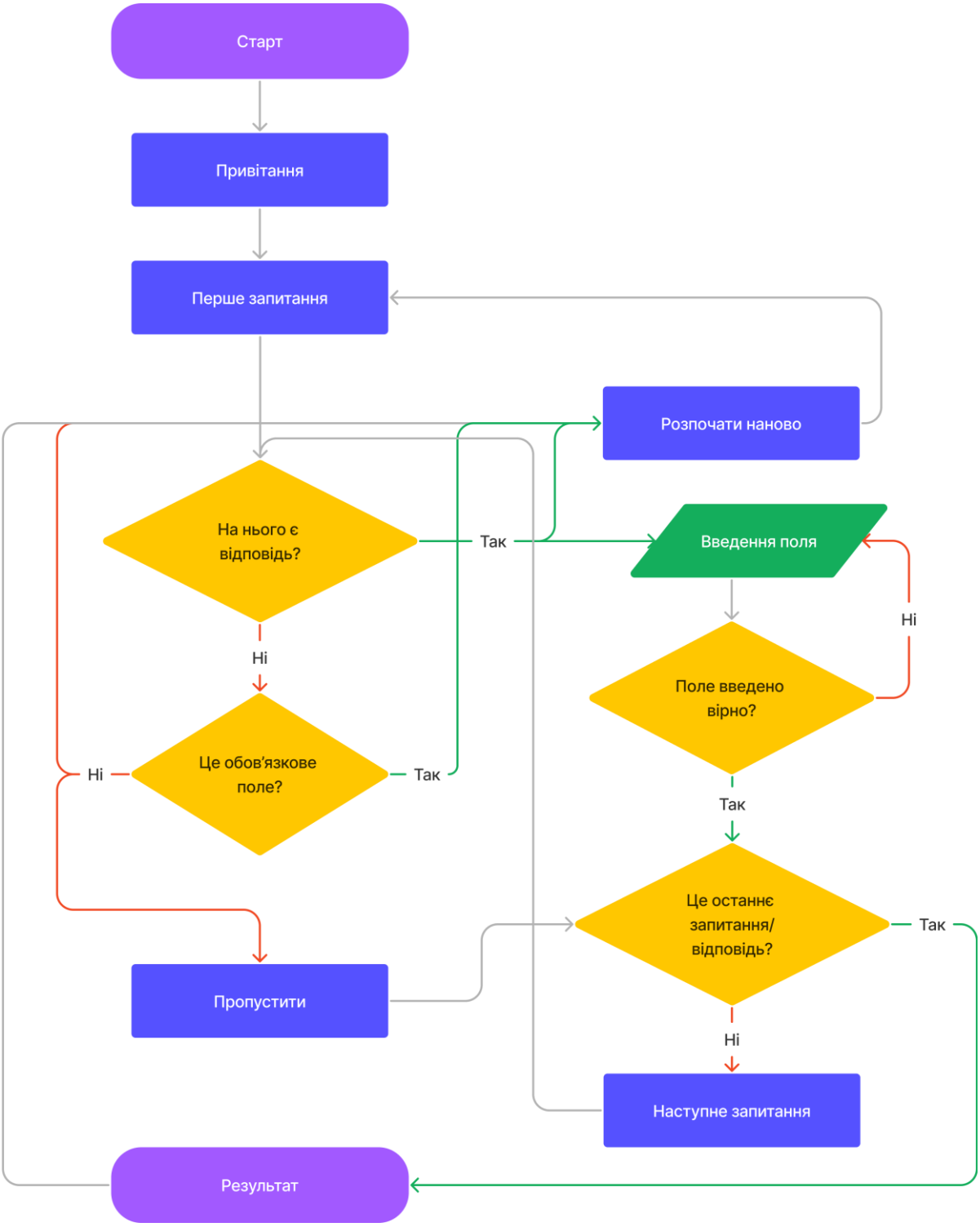

Рисунок 0.2 Блок-схема діалогу із користувачем

Нижче наведений загальний опис можливих сценаріїв роботи з чат-ботом.

Початок діалогу:

- Користувач розпочинає діалог з чат-ботом.
- Чат-бот вітає користувача та пропонує розпочати процес введення даних.

Введення питання:

Чат-бот поставляє питання користувачу.

- Якщо питання є обов'язковим, чат-бот чекає на відповідь користувача.
- Якщо питання не є обов'язковим, користувач має можливість пропустити питання та перейти до наступного кроку.

Перевірка введених даних:

- Чат-бот перевіряє введені користувачем дані на правильність.
- Якщо дані введено вірно, чат-бот переходить до наступного питання або завершує процес, якщо це останнє питання.
- Якщо дані введено невірно, чат-бот повідомляє про помилку та просить користувача повторно ввести дані.

Повторне введення даних:

- У випадку неправильно введених даних, чат-бот дає користувачу можливість повторно ввести дані.
- Чат-бот повторює питання та очікує нової відповіді користувача.

Завершення діалогу:

- Після успішного введення всіх необхідних даних, чат-бот генерує PDF-файл зі зібраними даними.
- Чат-бот повідомляє користувача про успішне створення PDF-файлу та надсилає файл безпосередньо.

Початок нового діалогу:

Після завершення діалогу та отримання результату, користувач має можливість розпочати новий діалог з чат-ботом, якщо потрібно ввести дані для іншої людини.

3.2 Структура бази даних

У процесі реалізації функціоналу чат-бота для скринінгу програми, програма отримуватиме інформацію від серверів Telegram. З метою збереження цих даних у системі було створено декілька структур даних. Кожен запис у таблиці містить інформацію про конкретну особу, для якої здійснюється пошук даних. Поля запису включають такі дані: id, id чату, прізвище, ім'я, по-батькові, дата народження, стать, номер паспорту, код та статус.

Детальний опис введених даних наведений у табл. 3.1

Таблиця 3.1 База даних

Назва поля	**Тип**	**Призначення**
id	int	Унікальний ідентифікатор запису
chat_id	varchar	ідентифікатор чату, який дозволяє ідентифікувати конкретного користувача в системі Telegram
surname	varchar	Прізвище людини, дані якої шукає користувач
name	varchar	Ім’я людини, дані якої шукає користувач
patronim	varchar	По батькові людини, дані якої шукає користувач

birth_date	varchar	Дата народження людини, дані якої шукає користувач
sex	varchar	Стать людини, дані якої шукає користувач
passport	varchar	Номер паспорту або ID-картки людини, дані якої шукає користувач
code	varchar	Ідентифікаційний код людини, дані якої шукає користувач
status	varchar	Статус користувача в системі, який відображає його поточний прогрес або стан.

3.3 Проектування структури програмного продукту

У поданій структурі бази даних міститься ряд таблиць, які зберігають інформацію, введену користувачем, а також дані, отримані з відкритого джерела "Дія". Основна таблиця називається "requests" і містить поля, описані у таблиці 3.1.

У даній структурі також відображена інформація, отримана з відкритого джерела "Дія". База даних містить декілька таблиць, включаючи:

FOP: Ця таблиця міститься інформація про відкриті фізичними особами-підприємцями (ФОП) підприємства. Вона містить дані такі, як дата відкриття, адреса та інші деталі.

WANTED: У таблиці "wanted" зберігається інформація про осіб, які перебувають у розшуку. Ця інформація включає дані про особу, статус розшуку, деталі кримінальної справи та інші відповідні атрибути.

SSPDR: У цій таблиці зберігається інформація для перевірки у списку

судових справ, призначених для розгляду. Вона включає дату вчинення злочину, вид правопурушення та інше.

ASVP: Таблиця "asvp" містить дані про осіб, які перебувають під адміністративним контролем. Вона містить інформацію для перевірки у автоматизованій системі виконавчого провадження Міністерства юстиції України.

DEBT: У цій таблиці зберігається інформація про борги. Вона містить дані про особу, суму боргу, дату заборгованості та інші пов'язані атрибути.

RBA: Таблиця "rba" містить інформацію про осіб, які підлягають ризиковому банківському обліку (RBA). Вона містить дані про причетність до збанкрутілих компаній, заходи, що були прийняті та інші відповідні атрибути.

CORRUPTION: Таблиця "corruption" містить дані про випадки корупції. Вона містить інформацію про особу, тип корупційного порушення, статус справи та інші деталі.

SANCTIONS_YU: Ця таблиця містить дані про санкції, пов'язані з юридичними особами (юридичними одиницями). Вона містить різноманітну інформацію про санкційні обмеження, які застосовуються до цих юридичних осіб.

SANCTIONS_FI: У цій таблиці зберігається інформація про санкції, пов'язані з фізичними особами. Вона містить дані про фізичну особу, рішення щодо санкцій, період обмежень та інші пов'язані атрибути.

LUSTR: Таблиця "lustr" містить інформацію про осіб, які перебувають у списку люстрації. Дані включають інформацію про застосування положення Закону України Про очищення влади.

PASSPORT: У таблиці зберігається інформація про паспорти. Вона містить дані про особу, серію та номер паспорта та інші пов'язані атрибути.

Ці таблиці дозволяють зберігати та організовувати різноманітну інформацію, що стосується санкцій, розшуку, адміністративного контролю, боргів, RBA, корупції, списків листування та втрати паспортів. Вони надають

можливість зручного доступу до цих даних та подальшого використання їх у функціоналі чат-бота для скринінгу.

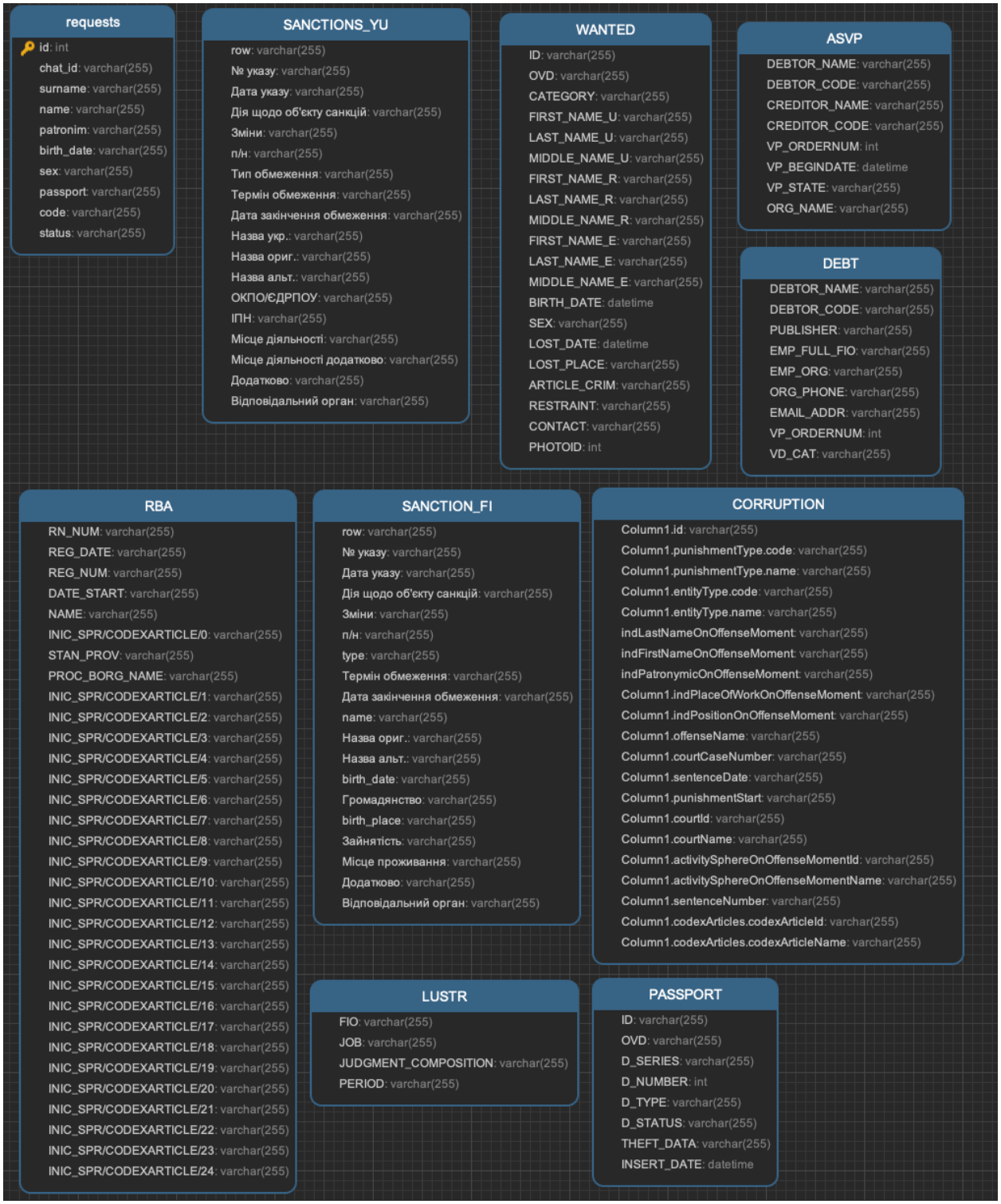

Рис.0.3 Структура чат-боту, частина 1

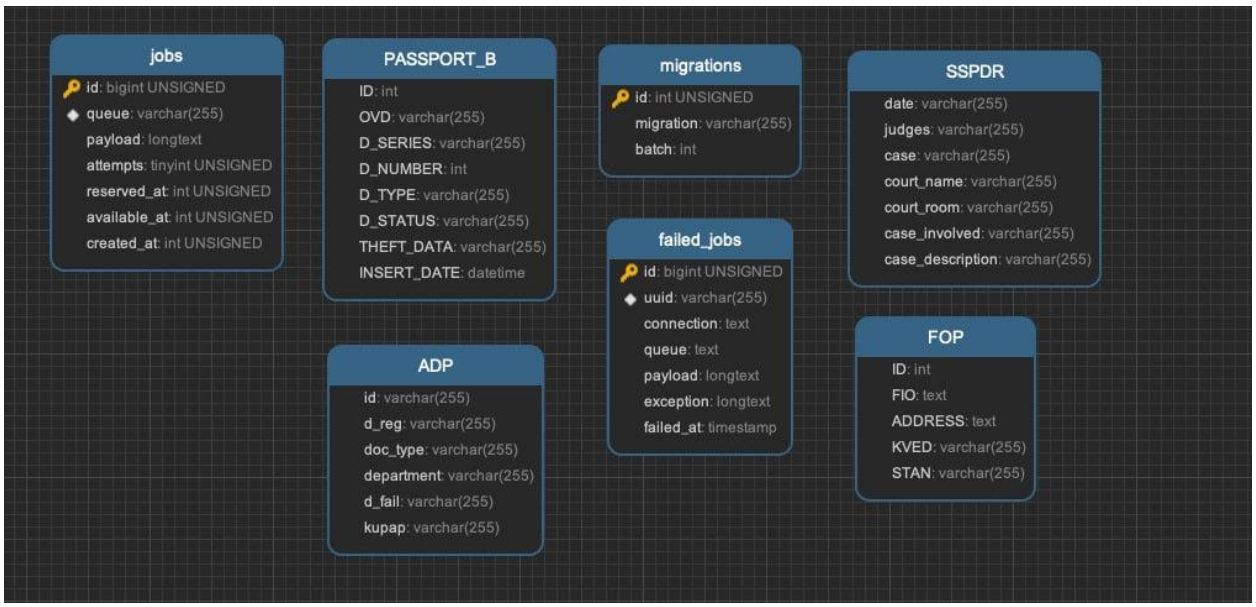

Рис.0.4 Структура чат-боту, частина 2

3.4 Тестування сценаріїв

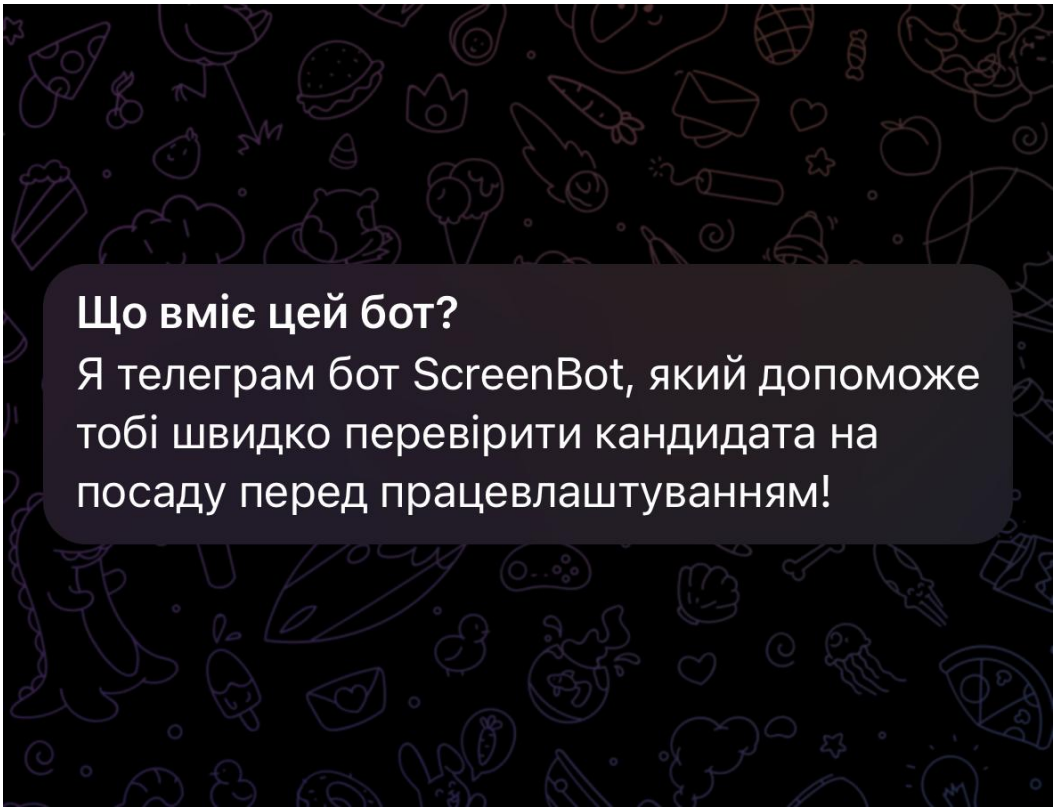

Рис.0.5 Початкова сторінка боту до запуску

Спершу буде проведений тест запуску бота, описаний у таблиці 3.2.

Таблиця 3.2 Тест запуску діалогу

Мета тесту	**Перевірка запуску діалогу з чат-ботом**
Стан системи	Відкритий діалог з чат-ботом
Дії	Відправити повідомлення із командою “/start”
Очікуваний результат	Бот вітає користувача, надсилає декілька повідомлень із детальним описом того, як працює бот і що робити користувачу, після чого задає перше питання і з’являється початкове меню

Результат тесту показаний на рис. 3.6

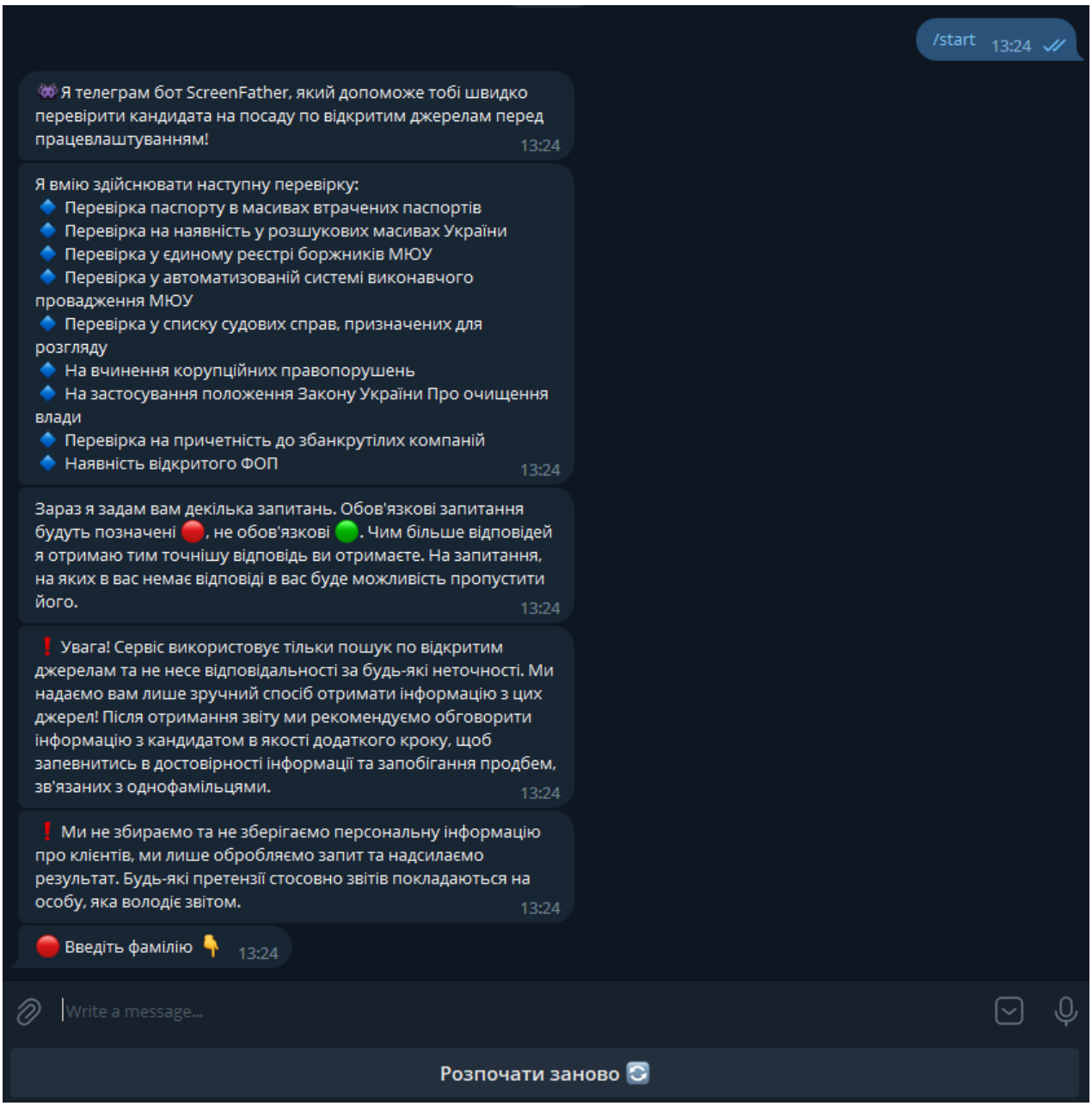

Рис.3.6 Інформація про чат-бота після запуску діалогу

Після прочитання інформації про чат-бота йде перша дія користувача – введення фамілії, тестування описано в таблиці 3.3.

Таблиця 3.3 Тест запуску діалогу

Мета тесту	**Перевірка відправлення наступного запиту**
Стан системи	Ініційований діалог з чат-ботом
Дії	Відправити відповідь на запит

Очікуваний результат	Бот відправляє наступний запит на введення інформації

Результат тесту показаний на рис. 3.7

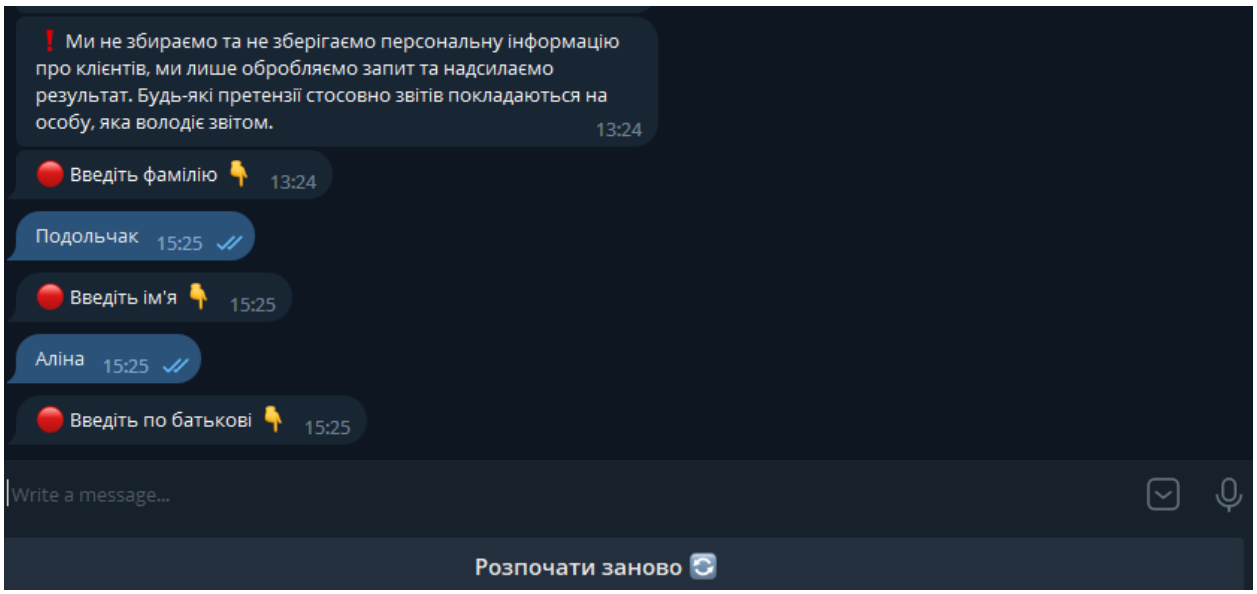

Рис.3.7 Відправлення наступного запиту
після введення попереднього

Після введення необхідних даних відправляється запис на введення дати народження, тестування описано в таблиці 3.4.

Таблиця 3.4 Тест запуску діалогу

Мета тесту	**Перевірка відправлення наступного запиту**
Стан системи	Відправлений запит на введення дати народження у необхідному форматі, на екрані виведене меню з кнопками “Розпочати заново” та “Пропустити”
Дії	Відправити відповідь на запит
Очікуваний результат	При відправленні невірного формату даних бот відправляє сповіщення з помилкою та очікує на

	повторне введення, при відправленні вірного формату - відправляє наступний запит на введення інформації

Результати тесту показані на рис. 3.8-3.9.

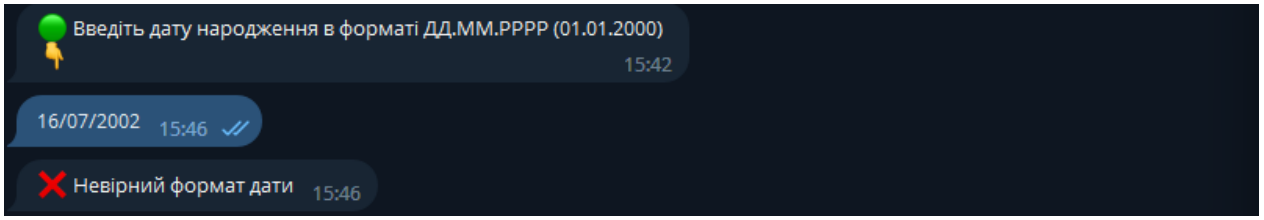

Рис.3.8 Результат при введенні невірного формату

Рис.3.9 Результат при введенні вірного формату

Після цього буде відправлений запит на вибір статі за допомогою кнопок, тестування яких описано в таблиці 3.5.

Таблиця 3.5 Тест запуску діалогу

Мета тесту	**Перевірка роботи кнопки вибору статі**
Стан системи	Відправлений запит на введення статі, на екрані виведене меню з кнопками “Чоловіча” та “Жіноча”
Дії	Натиснути на кнопку

Очікуваний результат	Бот відправляє наступний запит на введення інформації

Результат тесту показаний на рис. 3.10

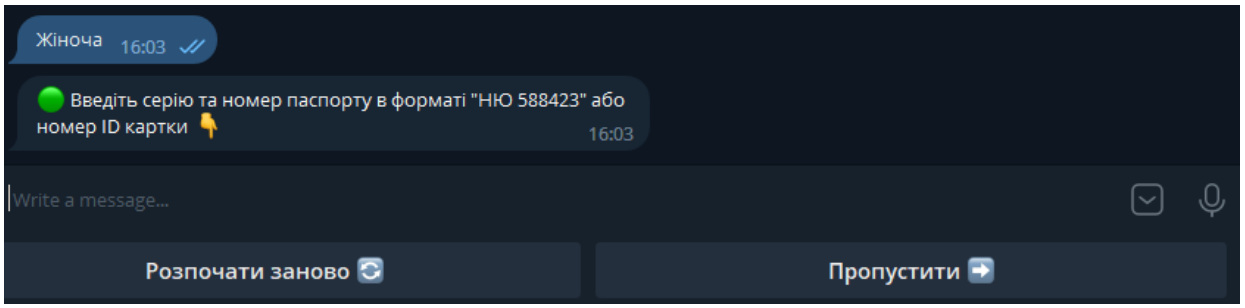

Рис.3.10 Результат при натисканні кнопки вибору статі

Також при введені необов’язкового поля повинна працювати кнопка “Пропустити”, тестування цієї кнопки описано в таблиці 3.6.

Таблиця 3.6 Тест запуску діалогу

Мета тесту	**Перевірка роботи кнопки пропуску введення**
Стан системи	Відправлений запит на введення необов’язкового поля (яке позначене зеленим кольором), на екрані виведене меню з кнопками “Розпочати заново” та “Пропустити”
Дії	Натиснути на кнопку “Пропустити”
Очікуваний результат	Бот пропускає введення заданого поля і переходить до введення наступного

Результат тесту показаний на рис. 3.11

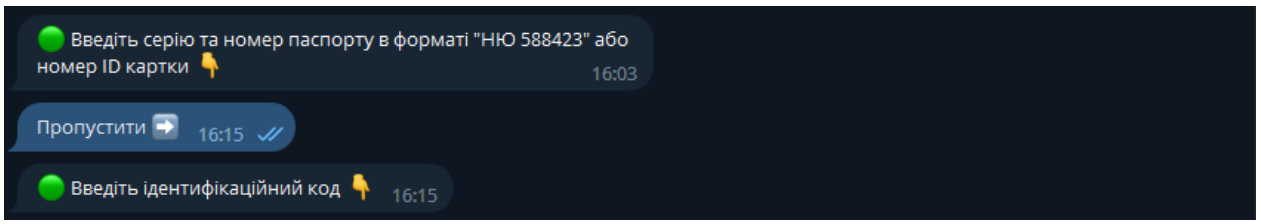

Рис.3.11 Результат при натисканні
кнопки “Пропустити”

Після введення або пропуску останнього поля відправляється звіт, тестування описано в таблиці 3.7.

Таблиця 3.7 Тест запуску діалогу

Мета тесту	**Перевірка отримання звіту**
Стан системи	Відправлений запит на введення необов’язкового поля (яке позначене зеленим кольором), на екрані виведене меню з кнопками “Розпочати заново” та “Пропустити”
Дії	Ввести інформацію або натиснути на кнопку “Пропустити”
Очікуваний результат	Бот відправляє повідомлення щодо очікування звіту та звіт у форматі PDF.

Результат тесту показаний на рис. 3.12

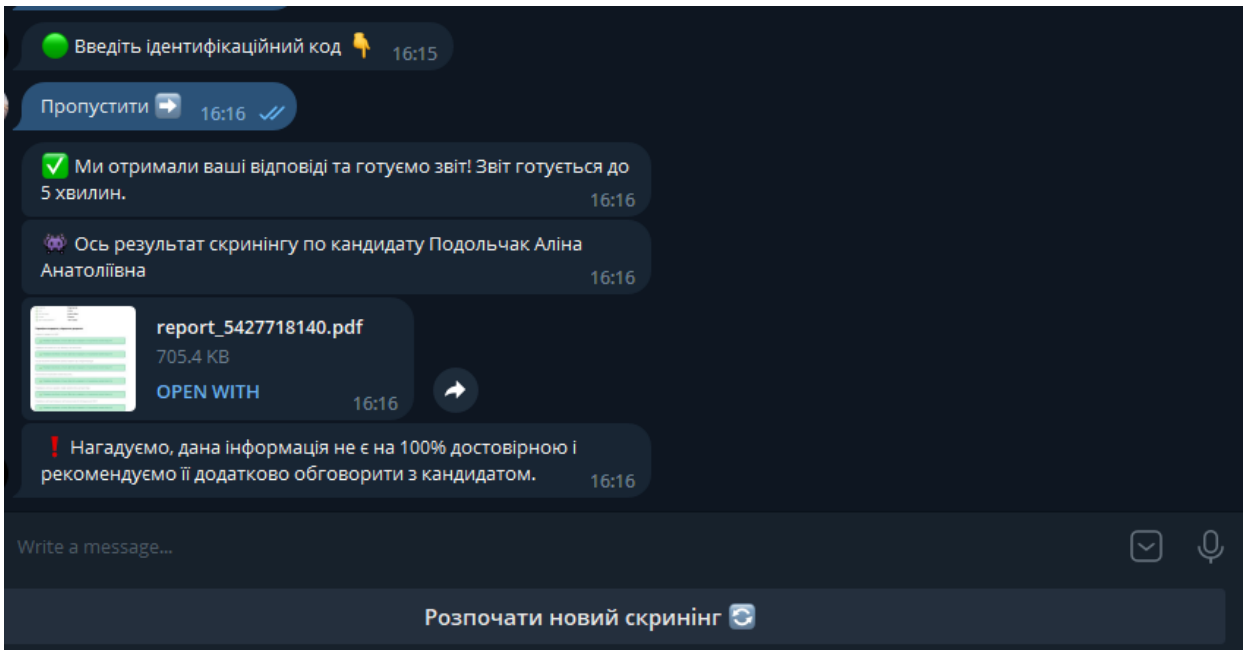

Рис.3.12 Кінцевий результат

Також в будь-який момент та після отримання кінцевого результату має бути можливість розпочати новий скринінг, тестування даної кнопки описано в таблиці 3.8.

Таблиця 3.8 Тест запуску діалогу

Мета тесту	**Перевірка рестарту бота**
Стан системи	На екрані виведене меню з кнопкою “Розпочати заново” або “Розпочати новий скринінг”
Дії	Натиснути на кнопку задану кнопку
Очікуваний результат	Розпочинається ввід даних для нового скринінгу

Результат тесту показаний на рис. 3.13

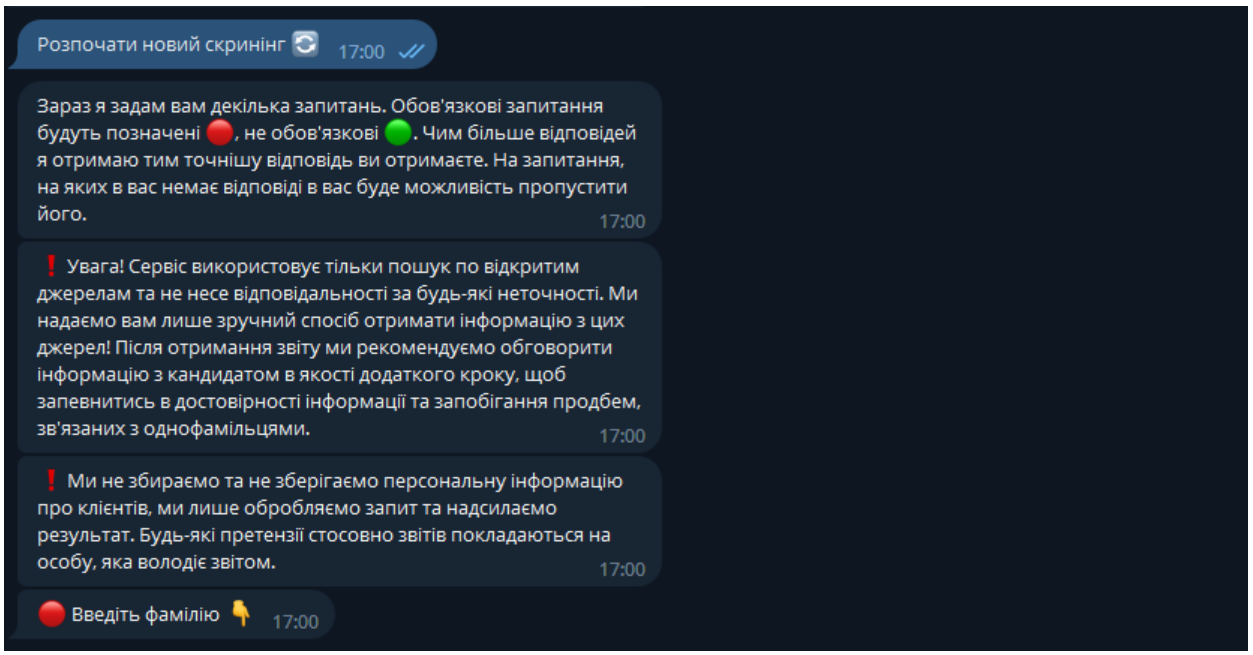

Рис.3.13 Початок нового скринінгу

3.5 Інструкція для користувачів

ScreenBot - Telegram бот, який допоможе швидко перевірити кандидата на посаду по відкритим джерелам перед працевлаштуванням!

Бот вміє здійснювати наступну перевірку:

- Перевірка паспорту в масивах втрачених паспортів
- Перевірка на наявність у розшукових масивах України
- Перевірка у єдиному реєстрі боржників МЮУ
- Перевірка у автоматизованій системі виконавчого провадження МЮУ
- Перевірка у списку судових справ, призначених для розгляду
- На вчинення корупційних правопорушень
- На застосування положення Закону України Про очищення влади
- Перевірка на причетність до збанкрутілих компаній
- Наявність відкритого ФОП

Для того, щоб скористатись чат-ботом, необхідно мати смартфон,

планшет або ПК.

Крок 1: Завантажити та встановити Telegram

- Відкрити мобільний додаток для завантаження програм та знайти Telegram у магазині додатків (App Store для iOS або Google Play для Android).
- Завантажити додаток Telegram та встановити його на мобільний пристрій або ПК.
- Після завершення встановлення відкрити додаток та створити обліковий запис, якщо цього не було зроблено раніше. Буде потрібно номер мобільного телефону для реєстрації.

Крок 2: Знайти бота

- У додатку Telegram натиснути на іконку "Пошук" (магнітна лупа) у верхньому правому куті екрана.
- Ввести назву бота “ScreenBot”/нік бота @screenfather_bot у поле пошуку або скористайтись QR-кодом, заданим нижче.

Рис.3.14 QR-код з посиланням на бота

Крок 3: Почати спілкування

- Натиснути кнопку "Start" або "Почати", щоб розпочати спілкування з ботом. Це може бути окрема кнопка або команда, яку потрібно ввести у чаті.

Крок 4: Взаємодія з ботом

- Після натискання кнопки "Start" або "Почати" користувача буде перенаправлено до чат-інтерфейсу з ботом. Тут потрібно вводити по черзі дані про кандидата.
- Необхідно звернути увагу на інструкції та застереження, які надає бот, та відповідати на його запити відповідно до контексту.
- Обов'язкові запитання позначені ●, не обов'язкові . Чим більше відповідей отримає бот, тим точнішу відповідь отримає користувач. На запитання, на яких у користувача немає відповіді буде можливість пропустити його.

Застереження:

! Увага! Сервіс використовує тільки пошук по відкритим джерелам та не несе відповідальності за будь-які неточності. Сервіс надає лише зручний спосіб отримати інформацію з цих джерел! Після отримання звіту рекомендується обговорити інформацію з кандидатом в якості додаткового кроку, щоб запевнитись в достовірності інформації та запобігання проблем, зв'язаних з однофамільцями.

! Чат-бот не збирає та не зберігає персональну інформацію про клієнтів, а лише обробляє запит та надсилаємо результат. Будь-які претензії стосовно звітів покладаються на особу, яка володіє звітом.

В результаті бот пришле всю зібрану інформацію про кандидата у згенерованому PDF-файлі.

Рис.3.15 Приклад інформації у PDF-файлі

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Роль центральної нервової системи в трудовій діяльності людини

Сучасне суспільство виявляє все більший інтерес до питань безпеки життєдіяльності, оскільки зрозуміло, що здоров'я та благополуччя людини є найважливішими факторами для досягнення успіху. Одним з найбільш поширених видів трудової діяльності є робота за комп'ютером, яка пов'язана з багатьма фізичними і психологічними факторами, що впливають на центральну нервову систему (ЦНС). У цьому розділі ми дослідимо роль ЦНС у трудовій діяльності людини з особливим акцентом на роботі за комп'ютером та обговоримо важливі аспекти безпеки, пов'язані з цим видом роботи.

Вплив роботи за комп'ютером на ЦНС:

1. Фізичні аспекти

Робота за комп'ютером вимагає тривалого сидячого положення та повторних рухів, що може призводити до погіршення кровообігу та м'язової напруги. Такі фізичні фактори можуть викликати напруження у м'язах спини, шиї та кистей рук, що впливає на стан спинного мозку та головного мозку, які є складовими ЦНС.

2. Психологічні аспекти

Робота за комп'ютером часто пов'язана з високим рівнем психологічного навантаження. Стрес, втома, тривожність та інші психологічні фактори можуть впливати на роботу ЦНС, зокрема на швидкість обробки інформації, увагу, пам'ять та інші когнітивні функції.

Загрози безпеці та здоров'ю при роботі за комп'ютером:

1. Синдром комп'ютерного зору

Довготривале перебування перед екраном комп'ютера може призводити до синдрому комп'ютерного зору, який характеризується сухістю очей, запамороченням, розпливчастим зором та іншими неприємними симптомами.

2. Мускулоскелетні розлади

Неправильна постава та повторні рухи під час роботи за комп'ютером можуть спричиняти розвиток мускулоскелетних розладів, таких як синдром карпального каналу, болі у спині та шиї, захворювання суглобів тощо.

3. Психоемоційні проблеми

Висока вимогливість роботи за комп'ютером, надмірне навантаження та тривалий стрес можуть призводити до психоемоційних проблем, таких як депресія, тривожність та вигорання.

Заходи безпеки та збереження здоров'я при роботі за комп'ютером:

1. Ергономіка робочого місця

Важливо забезпечити правильну організацію робочого місця, включаючи належне розташування монітора, клавіатури, миші та стільця, щоб уникнути надмірного навантаження м'язів та хребта.

2. Регулярні перерви та фізичні вправи

Необхідно робити періодичні перерви під час роботи за комп'ютером та виконувати фізичні вправи для зняття напруги з м'язів і покращення кровообігу.

3. Очні вправи

Регулярні очні вправи можуть допомогти зменшити навантаження на

очі та запобігти синдрому комп'ютерного зору.

4. Стрес-менеджмент та психологічна підтримка

Важливо вивчати техніки стрес-менеджменту та забезпечувати психологічну підтримку працівникам, що працюють за комп'ютером, для зменшення психоемоційних проблем.

Освітня програма та свідомість працівників

Важливо впровадити освітні програми та навчальні сесії, спрямовані на свідоме ставлення до безпеки та здоров'я під час роботи за комп'ютером. Працівники повинні бути поінформовані про потенційні ризики та навчитися виявляти ознаки перенапруження ЦНС, а також знати методи попередження та управління цими ризиками.

Моніторинг та оцінка ризиків

Необхідно встановити систему моніторингу та оцінки ризиків для ідентифікації потенційних проблем та вжиття відповідних заходів забезпечення безпеки. Це може включати регулярні огляди робочих місць, збір статистичних даних про випадки захворювань, анкетування працівників та виконання оцінок ризиків.

Застосування регулярних медичних оглядів

Проведення регулярних медичних оглядів працівників, які працюють за комп'ютером, є важливим аспектом забезпечення безпеки та здоров'я. Це дозволяє виявити можливі проблеми зі здоров'ям, пов'язані з роботою за комп'ютером, вчасно і прийняти необхідні заходи для їх запобігання або лікування.

Система контролю за здоров'ям працівників

Організації повинні встановити систему контролю за здоров'ям працівників, які працюють за комп'ютером. Це включає проведення регулярних медичних оглядів та діагностичних процедур, які дозволяють виявити можливі відхилення в роботі центральної нервової системи. Результати таких оглядів можуть служити основою для розробки індивідуальних рекомендацій щодо забезпечення безпеки та покращення здоров'я працівників.

Психологічна підтримка та соціальна взаємодія

Робота за комп'ютером може бути одинокою та монотонною, що може негативно впливати на психологічний стан працівників. Організації повинні забезпечити психологічну підтримку, включаючи консультування та психологічні програми для підтримки емоційного стану працівників. Крім того, створення можливостей для соціальної взаємодії, таких як комунікаційні мережі, спільні події та зустрічі, сприяє покращенню самопочуття та загальному добробуту працівників.

Постійне вдосконалення та дослідження

Безпека життєдіяльності і роль центральної нервової системи в трудовій діяльності людини є активно розвиваючими галузями. Постійне вдосконалення і дослідження в цій галузі є необхідними для виявлення нових викликів та розробки ефективних стратегій забезпечення безпеки працівників, особливо тих, хто працює за комп'ютером. Наукові дослідження, обмін досвідом та співпраця з іншими організаціями допомагають розвивати та впроваджувати нові підходи та інновації у сфері безпеки життєдіяльності.

Отже, центральна нервова система відіграє вирішальну роль у трудовій діяльності людини, особливо при роботі за комп'ютером. Забезпечення безпеки та здоров'я працівників, які працюють за комп'ютером, вимагає комплексного підходу, що враховує ергономічні аспекти, фізичну та психологічну здатність, а також використання сучасних технологій та наукових досліджень.

Реалізація рекомендацій, які були наведені у цьому розділі, допоможе забезпечити безпеку, покращити здоров'я та підвищити продуктивність працівників, які працюють за комп'ютером. Організації повинні приділяти належну увагу безпеці центральної нервової системи та розробляти стратегії, щоб забезпечити ефективні умови праці, психологічну підтримку та постійне вдосконалення системи безпеки.

Це стане фундаментом для створення здорового та продуктивного робочого середовища, яке підтримує безпеку та добробут працівників, а також сприяє досягненню успіху організації.

4.2 Вимоги безпеки до робочих місць для виконання робіт.

В сучасному інформаційному суспільстві багато людей проводять значну кількість часу за комп'ютером, виконуючи різноманітні роботи. Протягом останніх десятиліть спостерігається зростання участі комп'ютерної роботи в різних сферах діяльності, що зумовлює необхідність враховувати вимоги безпеки та основи охорони праці при роботі за комп'ютером. Цей розділ присвячений огляду вимог безпеки, які необхідно дотримуватись при організації робочих місць для виконання робіт, зокрема за комп'ютером.

Організація робочого місця:

1.1. Правильне розташування комп'ютерної робочої станції,

забезпечення оптимального освітлення та вентиляції.

1.2. Регулювання висоти стільця та робочого столу для забезпечення правильної постави тіла під час роботи.

1.3. Забезпечення належного розташування комп'ютерної клавіатури, миші та інших пристроїв, що мінімізує напруження м'язів та навантаження на суглоби.

Ергономіка робочого місця:

2.1. Використання ергономічних меблів та обладнання для підтримки комфортної та безпечної роботи.

2.2. Налагодження регулювання положення екрану, щоб уникнути відблисків та забезпечити оптимальний кут огляду.

2.3. Застосування підставки для зап'ястя та клавіатурного підголовника, що зменшує навантаження на руки та шию.

Перерви та фізичні вправи:

3.1. Регулярні перерви під час тривалої роботи за комп'ютером, щоб зменшити напруження м'язів та очей.

3.2. Виконання спеціальних фізичних вправ, спрямованих на розслаблення м'язів, покращення кровообігу та загальної фізичної активності.

Захист зору:

4.1. Використання екрану з антибліковим покриттям та налаштування його яскравості та контрастності.

4.2. Регулярні перерви для очей, включаючи фокусування на віддалених об'єктах та виконання спеціальних вправ для очей.

Електробезпека:

5.1. Правильне підключення комп'ютерної техніки до електричної мережі та використання захисних пристроїв, наприклад, стабілізаторів

напруги та фільтрів шуму.

5.2. Регулярна перевірка електропроводки та обладнання на наявність пошкоджень або ознак несправності.

Психологічний комфорт:

6.1. Забезпечення приємної атмосфери на робочому місці, що сприяє зосередженості та продуктивності.

6.2. Мінімізація шуму та відволікаючих факторів, що можуть впливати на концентрацію та емоційний стан працівника.

Комп'ютерна безпека:

7.1. Захист від несанкціонованого доступу до комп'ютерної системи шляхом використання паролів, шифрування даних та інших технічних заходів безпеки.

7.2. Регулярне оновлення антивірусного програмного забезпечення та застосування заходів для запобігання загрозам кібербезпеки, таким як фішинг, шкідливе програмне забезпечення та хакерські атаки.

Навчання та свідомість працівників:

8.1. Проведення навчання та інструктажу щодо вимог безпеки та основ охорони праці при роботі за комп'ютером.

8.2. Підвищення свідомості працівників про основні проблеми, пов'язані з роботою за комп'ютером, та навчання їх правильним методам профілактики та реагування на можливі проблеми.

Моніторинг та оцінка:

9.1. Регулярний моніторинг дотримання вимог безпеки та основ охорони праці при роботі за комп'ютером.

9.2. Проведення оцінки ризиків, що пов'язані з виконанням робіт за комп'ютером, та впровадження заходів для їх зменшення.

Актуалізація та адаптація:

10.1. Постійна актуалізація знань і вимог безпеки до робочих місць для виконання робіт за комп'ютером з урахуванням нових технологій та стандартів.

10.2. Адаптація до індивідуальних особливостей та потреб працівників, забезпечення можливості внесення змін в організацію робочого місця для максимальної безпеки та комфорту.

Роль керівництва та відповідальність:

11.1. Керівництво організації повинно приділяти високий пріоритет вимогам безпеки та охорони праці при роботі за комп'ютером.

11.2. Керівництво повинно забезпечити належне фінансування, ресурси та навчання для виконання необхідних заходів з охорони праці.

Співпраця з професійними фахівцями:

12.1. Залучення кваліфікованих фахівців з охорони праці для консультацій, оцінки ризиків та розробки політики безпеки при роботі за комп'ютером.

12.2. Регулярний обмін інформацією та співпраця з медичним персоналом для виявлення та вирішення проблем, пов'язаних зі здоров'ям працівників.

Постійне вдосконалення:

13.1. Аналіз результатів моніторингу, оцінки ризиків та зворотного зв'язку від працівників для виявлення можливих проблем та вдосконалення вимог безпеки при роботі за комп'ютером.

13.2. Впровадження нових технологій, методів та підходів для постійного поліпшення умов праці та забезпечення безпеки.

Культура безпеки та свідоме ставлення:

14.1. Популяризація та поширення культури безпеки серед працівників, включаючи свідоме ставлення до виконання вимог безпеки при роботі за комп'ютером.

14.2. Підтримка відкритої комунікації та співпраці між всіма учасниками процесу для постійного вдосконалення безпеки та охорони праці.

Застосування ергономічних принципів:

15.1. Належна організація робочого місця з урахуванням оптимальної позиції тіла під час виконання роботи за комп'ютером.

15.2. Використання належного обладнання, такого як ергономічні стільці, столи, клавіатури та миші, що забезпечують комфортні умови праці та запобігають м'язовим напруженням та травмам.

Захист зору:

16.1. Забезпечення належного освітлення робочого місця для запобігання напруженню очей та втомі зору.

16.2. Регулярні перерви для відпочинку очей та виконання вправ для зменшення напруження зору.

Ці заходи та вимоги є необхідними для забезпечення безпеки та охорони праці під час виконання робіт за комп'ютером. Вони враховують ергономічні принципи, захист зору, електробезпеку, вентиляцію та санітарні умови, а також надають увагу здоровому способу життя та комп'ютерній безпеці. Ці вимоги повинні бути постійно оновлюваними та відповідати вимогам законодавства для забезпечення безпеки та добробуту працівників.

ВИСНОВКИ

В результаті виконання дипломної роботи "Розробка чат-боту для скринінгу кандидатів перед працевлаштуванням за допомогою PHP та відкритого API “Дія” " було ретельно проаналізовано предметну область і встановлені ключові вимоги до розроблюваної системи. Загальні відомості про чат-боти були вивчені з метою отримання необхідного контексту для подальшої роботи. Опис предметного середовища надав зрозуміле уявлення про середу, в якій буде функціонувати розроблюваний чат-бот.

Важливою частиною дослідження був порівняльний аналіз різних месенджерів з платформою чат-ботів. Це дозволило визначити найбільш підходящу платформу для реалізації поставлених завдань. Критерії вибору включали функціональні можливості, легкість інтеграції, популярність серед користувачів тощо. Такі дослідження допомагають забезпечити ефективне функціонування розробленої системи та вибрати найбільш оптимальний месенджер для взаємодії з користувачами.

Після вивчення предметної області та аналізу вимог, у другому розділі було вирішено питання вибору технологій для розробки системи керування чат-ботом. Основні рішення, які було прийнято, включали архітектуру системи, вибір мови програмування, бази даних, інструментів розгортання та управління додатками. Аналіз та вибір архітектури системи дозволив визначити оптимальну організацію компонентів, що забезпечує високу продуктивність та масштабованість системи. Мова програмування PHP була обрана на основі її широкого застосування у різних сферах і великої кількості можливостей. Вибір бази даних MySQL був зумовлений його надійністю, широким функціоналом та сумісністю з обраною мовою програмування. Використання Docker як інструменту розгортання та управління додатками дозволило створити ізольоване та легко переносиме середовище для розробки та експлуатації чат-боту. Для забезпечення взаємодії з користувачами було вирішено використовувати Telegram API, оскільки

Telegram є популярним месенджером з широким функціоналом та зручним інтерфейсом для розробки ботів. Вибір платформи для розробки боту був здійснений на основі його можливостей для швидкого та зручного створення, налаштування та впровадження чат-боту.

У третьому розділі дипломної роботи було проведено розробку програмного продукту. Алгоритм взаємодії з чат-ботом був розроблений, враховуючи вимоги до системи та потреби користувачів. Структура бази даних була проектувана таким чином, щоб забезпечити ефективне зберігання та обробку даних, необхідних для функціонування чат-боту. Було проведено тестування різних сценаріїв, щоб переконатись у правильності роботи програмного продукту та відповідності його функціональності вимогам. Крім того, була розроблена інструкція для користувачів, яка детально пояснює процес взаємодії з чат-ботом та надає необхідні рекомендації для його використання.

В цілому, виконана дипломна робота "Розробка чат-боту для скринінгу кандидатів перед працевлаштуванням за допомогою PHP" детально розглядає аналіз предметної області, постановку завдання та вибір технологій, а також описує процес розробки програмного продукту. Результатом роботи є реалізований чат-бот, який відповідає вимогам та може успішно використовуватись для скринінгу кандидатів перед працевлаштуванням.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Telegram Bot API Documentation [Електронний ресурс] - URL: https://core.telegram.org/bots/api
2. PHP Manual [Електронний ресурс] - URL: https://www.php.net/manual/en/
3. MySQL Documentation [Електронний ресурс] - URL: https://dev.mysql.com/doc/
4. Docker Documentation [Електронний ресурс] - URL: https://docs.docker.com/
5. PHP: Hypertext Preprocessor [Електронний ресурс] - URL: https://www.php.net/
6. MySQL: The World's Most Popular Open Source Database [Електронний ресурс] - URL: https://www.mysql.com/
7. Docker: Get Started [Електронний ресурс] - URL: https://www.docker.com/get-started
8. PHP Standards Recommendations (PSRs) [Електронний ресурс] - URL: https://www.php-fig.org/psr/
9. Docker Hub [Електронний ресурс] - URL: https://hub.docker.com/
10. Telegram Bot API - Introduction [Електронний ресурс] - URL: https://core.telegram.org/bots
11. PHP Classes and Objects [Електронний ресурс] - URL: https://www.php.net/manual/en/language.oop5.php
12. MySQL Developer Zone [Електронний ресурс] - URL: https://dev.mysql.com/developer/
13. Docker Compose Documentation [Електронний ресурс] - URL: https://docs.docker.com/compose/
14. PHP Security Consortium [Електронний ресурс] - URL:

https://phpsecurity.org/

15. PHP.net - Features [Електронний ресурс] - URL: https://www.php.net/features
16. MySQL Reference Manual [Електронний ресурс] - URL: https://dev.mysql.com/doc/refman/
17. Telegram Bot API - Methods [Електронний ресурс] - URL: https://core.telegram.org/bots/api#available-methods

ДОДАТОК А Створення запиту

Лістинг файлу Requests.php:

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Requests extends Model
{
    use HasFactory;
    public $timestamps = false;
    protected $guarded = [];
    protected $fillable = [
        'chat_id',
        'surname',
        'name',
        'patronim',
        'birth_date',
        'sex',
        'passport',
        'code',
        'status'
    ];

}
```

ДОДАТОК Б Функції чат-боту

Лістинг файлу Telegram.php:

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\Requests;
use App\Jobs\GenerateReport;
use PDF;

class Telegram extends Controller
{
   public function bot_path(){
      return
"https://api.telegram.org/bot5571895877:AAEX2Dakz6RqP4rzYkeN5w_SSq8yb-02YvQ";
   }
   public function sm($chatId, $text){
      file_get_contents($this->bot_path()."/sendmessage?chat_id=".$chatId."&text=" .
urlencode($text));
   }
   public function sr($chatId, $text){
      $reply_markup = json_encode([
         'keyboard' => [['Розпочати заново    ']],
         'resize_keyboard' => true,
         'one_time_keyboard' => true,
      ]);
      file_get_contents($this->bot_path()."/sendmessage?chat_id=".$chatId."&text=" .
urlencode($text)."&reply_markup=" . $reply_markup);
   }
   public function ss($chatId, $text, $buttons = null, $empty = false){
      $buttons_list = [['Розпочати заново    ','Пропустити ➞□']];
      if($buttons !== null){
         array_unshift($buttons_list, $buttons);
```

```
            }
            if($empty){
                $buttons_list = [["","]];
            }
            $reply_markup = json_encode([
                'keyboard' => $buttons_list,
                'resize_keyboard' => true,
                'one_time_keyboard' => true,
            ]);
            file_get_contents($this->bot_path()."/sendmessage?chat_id=".$chatId."&text=" .
urlencode($text)."&reply_markup=" . $reply_markup);
        }

        public function message(){
            $update = json_decode(file_get_contents("php://input"), true);
            //status:  surname,name,patronim,birth,sex,passport,code,ready,sended
            //token: XmDX36KzTwIkkefKWGBJ6vfAwHaZGlxK
            //setWebhook:
https://api.telegram.org/bot5571895877:AAEX2Dakz6RqP4rzYkeN5w_SSq8yb-
02YvQ/setWebhook?url=https://5950-194-44-203-179.ngrok-
free.app/XmDX36KzTwIkkefKWGBJ6vfAwHaZGlxK
            if(!isset($update["message"]) || !isset($update["message"]["text"]) ){
                return 'ok';
            }

            $chatId = $update["message"]["chat"]["id"];
            $message = $update["message"]["text"];

            $user = Requests::all()->where('chat_id', $chatId);
            if(count($user) < 1){
                $user = Requests::create(['chat_id' => $chatId,'status' => 'surname']);
                if($message === '/start' || $message === 'Розпочати заново    '){
                    $this->sm($chatId, "🐻‍❄️Я телеграм бот ScreenFather, який допоможе тобі
швидко перевірити кандидата на посаду по відкритим джерелам перед
```

працевлаштуванням!");

$this->sm($chatId, "Я вмію здійснювати наступну перевірку:\n🔹 Перевірка паспорту в масивах втрачених паспортів\n🔹 Перевірка на наявність у розшукових масивах України\n🔹 Перевірка у єдиному реєстрі боржників МЮУ\n🔹 Перевірка у автоматизованій системі виконавчого провадження МЮУ\n🔹 Перевірка у списку судових справ, призначених для розгляду\n🔹 На вчинення корупційних правопорушень\n🔹 На застосування положення Закону України Про очищення влади\n🔹 Перевірка на причетність до збанкрутілих компаній\n🔹 Наявність відкритого ФОП");

$this->sm($chatId, "Зараз я задам вам декілька запитань. Обов'язкові запитання будуть позначені ●, не обов'язкові . Чим більше відповідей я отримаю тим точнішу відповідь ви отримаєте. На запитання, на яких в вас немає відповіді в вас буде можливість пропустити його.");

$this->sm($chatId, "❗Увага! Сервіс використовує тільки пошук по відкритим джерелам та не несе відповідальності за будь-які неточності. Ми надаємо вам лише зручний спосіб отримати інформацію з цих джерел! Після отримання звіту ми рекомендуємо обговорити інформацію з кандидатом в якості додаткого кроку, щоб запевнитись в достовірності інформації та запобігання продбем, зв'язаних з однофамільцями.");

$this->sm($chatId, "❗Ми не збираємо та не зберігаємо персональну інформацію про клієнтів, ми лише обробляємо запит та надсилаємо результат. Будь-які претензії стосовно звітів покладаються на особу, яка володіє звітом.");

}

} else {

$user = $user->first();

if(($message === '/start' || $message === 'Розпочати заново ' || $message === 'Розпочати новий скринінг ') && $user->status !== 'ready'){

$user->delete();

$user = Requests::create(['chat_id' => $chatId,'status' => 'surname']);

$this->sm($chatId, "Зараз я задам вам декілька запитань. Обов'язкові запитання будуть позначені ●, не обов'язкові . Чим більше відповідей я отримаю тим точнішу відповідь ви отримаєте. На запитання, на яких в вас немає відповіді в вас буде можливість пропустити його.");

$this->sm($chatId, "❗Увага! Сервіс використовує тільки пошук по

відкритим джерелам та не несе відповідальності за будь-які неточності. Ми надаємо вам лише зручний спосіб отримати інформацію з цих джерел! Після отримання звіту ми рекомендуємо обговорити інформацію з кандидатом в якості додаткого кроку, щоб запевнитись в достовірності інформації та запобігання продбем, зв'язаних з однофамільцями.");

$this->sm($chatId, "❗Ми не збираємо та не зберігаємо персональну інформацію про клієнтів, ми лише обробляємо запит та надсилаємо результат. Будь-які претензії стосовно звітів покладаються на особу, яка володіє звітом.");

}

}

if(($message === '/start' || $message === 'Розпочати заново ' || $message === 'Розпочати новий скринінг ') && $user->status !== 'ready'){

return $this->sr($chatId, "● Введіть фамілію 👇");

}

if(($message !== '/start' || $message !== 'Розпочати заново ' || $message === 'Розпочати новий скринінг ') && $user->status === 'ready'){

return $this->sm($chatId, "🙈 Новий скринінг можна почати після отримання результату по попередньому");

}

if($user->status === 'surname'){

$user->surname = $message;

$user->status = 'name';

$user->save();

return $this->sr($chatId, "● Введіть ім'я 👇");

}

if($user->status === 'name'){

$user->name = $message;

$user->status = 'patronim';

$user->save();

return $this->sr($chatId, "● Введіть по батькові 👇");

}

```
            if($user->status === 'patronim'){
                $user->patronim = $message;
                $user->status = 'birth';
                $user->save();
                return $this->ss($chatId, "    Введіть дату народження в форматі ДД.ММ.РРРР
(01.01.2000) 👇");
            }
            if($user->status === 'birth'){
                $pattern = '/^([0-2][0-9]|[3][0-1])\.([0][1-9]|[1][0-2])\.[0-9]{4}$/';
                if($message !== "Пропустити ➡" && !preg_match($pattern, $message)){
                    return $this->ss($chatId, "❌ Невірний формат дати");
                }
                $data = $message;
                if($message === "Пропустити ➡"){
                    $data = null;
                }
                $user->birth_date = $data;
                $user->status = 'sex';
                $user->save();
                return $this->ss($chatId, "    Оберіть стать 👇", ["Чоловіча","Жіноча"]);
            }
            if($user->status === 'sex'){
                if($message !== 'Чоловіча' && $message !== 'Жіноча' && $message !==
'Пропустити ➡' ){
                    return $this->ss($chatId, "❌ Невірний формат відповіді");
                }
                $data = $message;
                if($message === "Пропустити ➡"){
                    $data = null;
                }
                $user->sex = $data;
                $user->status = 'passport';
                $user->save();
```

```
                return $this->ss($chatId, "    Введіть серію та номер паспорту в форматі \"НЮ
588423\" або номер ID картки 👇");
            }
            if($user->status === 'passport'){
                $pattern = '/^[A-ZA-Я]{2}\s\d{6}$/u';
                $patternID = '/^\d{9}$/';
                if($message !== "Пропустити ➡" && !preg_match($pattern, $message) &&
!preg_match($patternID, $message)){
                    return $this->ss($chatId, "❌ Невірний формат відповіді");
                }
                $data = $message;
                if($message === "Пропустити ➡"){
                    $data = null;
                }
                $user->passport = $data;
                $user->status = 'code';
                $user->save();
                return $this->ss($chatId, "    Введіть ідентифікаційний код 👇");
            }
            if($user->status === 'code'){
                $pattern = '/^\d{9,10}$/';
                if($message !== "Пропустити ➡" && !preg_match($pattern, $message)){
                    return $this->ss($chatId, "❌ Невірний формат відповіді");
                }
                $data = $message;
                if($message === "Пропустити ➡"){
                    $data = null;
                }
                $user->code = $data;
                $user->status = 'ready';
                $user->save();
                $this->ss($chatId, "✅ Ми отримали ваші відповіді та готуємо звіт! Звіт
готується до 5 хвилин.", null, true);
```

```
            GenerateReport::dispatch($user);
        }
        return 'ok';
    }
}
```

ДОДАТОК В Створення звіту

Лістинг файлу GenerateReport.php:

```
<?php

namespace App\Jobs;

use Illuminate\Support\Facades\Storage;
use App\Models\Databases\ASVP;
use App\Models\Databases\CORRUPTION;
use App\Models\Databases\FOP;
use App\Models\Databases\RBA;
use App\Models\Databases\LUSTR;
use App\Models\Databases\DEBT;
use App\Models\Databases\PASSPORT;
use App\Models\Services\Groups;
use App\Models\Services\Tariffs;
use App\Models\Databases\SSPDR;
use App\Models\Requests;
use App\Models\Databases\WANTED;
use Illuminate\Bus\Queueable;
use Illuminate\Contracts\Queue\ShouldBeUnique;
use Illuminate\Contracts\Queue\ShouldQueue;
use Illuminate\Foundation\Bus\Dispatchable;
use Illuminate\Queue\InteractsWithQueue;
use Illuminate\Queue\SerializesModels;
use Illuminate\Support\Facades\Http;
use Illuminate\Support\Facades\Log;
use Barryvdh\DomPDF\Facade\Pdf;

class GenerateReport implements ShouldQueue
{
    use Dispatchable, InteractsWithQueue, Queueable, SerializesModels;

    /**
     * Create a new job instance.
     *
```

```
     * @return void
     */

    public $user;

    public function __construct($user)
    {
        $this->user = $user;
    }

    /**
     * Execute the job.
     *
     * @return void
     */
    public function handle()
    {
        $bot =
"https://api.telegram.org/bot5571895877:AAEX2Dakz6RqP4rzYkeN5w_SSq8yb-02YvQ";
        $user = $this->user;
        $candidate = $user->surname . " " . $user->name . " " . $user->patronim;
        $template = '<div class="header">
                <table style="border-collapse: collapse; width: 100%;" border="1">
                <tbody>
                <tr>
                <td style="width: 100%;">
                <h3>Верифікаційний звіт від ' . date("d.m.Y") . '</h3>
                </td>
                </tr>
                </tbody>
                </table>
                </div>
                <div class="core-data">
                <table style="border-collapse: collapse; width: 100%;" border="1">
                <tbody>
```

```
                    <tr>
                    <td style="width: 240px;"><span style="font-size: 10pt; color:
#7e8c8d;">[v]Фамілія</span></td>
                    <td><span style="font-size: 10pt;">' . $user->surname . '</span></td>
                    </tr>
                    <tr>
                    <td style="width: 240px;"><span style="font-size: 10pt; color:
#7e8c8d;">[v]Ім’я</span></td>
                    <td><span style="font-size: 10pt;">' . $user->name . '</span></td>
                    </tr>
                    <tr>
                    <td style="width: 240px;"><span style="font-size: 10pt; color:
#7e8c8d;">[v]По батькові</span></td>
                    <td><span style="font-size: 10pt;">' . $user->patronim . '</span></td>
                    </tr>
                    <tr>
                    <td style="width: 240px;"><span style="font-size: 10pt; color:
#7e8c8d;">[v]Стать</span></td>
                    <td><span style="font-size: 10pt;">' . $user->sex . '</span></td>
                    </tr>';
            if ($user->birth_date) {
                $template .= '<tr>
                    <td style="width: 240px;"><span style="font-size: 10pt; color:
#7e8c8d;">[v]Дата народження</span></td>
                    <td><span style="font-size: 10pt;">' . $user->birth_date . '</span></td>
                    </tr>';
            }
            if ($user->passport) {
                $template .= '<tr>
                    <td style="width: 240px;"><span style="font-size: 10pt; color:
#7e8c8d;">[v]Серія та/або номер паспорту</span></td>
                    <td><span style="font-size: 10pt;">' . $user->passport . '</span></td>
                    </tr>';
            }
            if ($user->code) {
```

```
            $template .= '<tr>
                <td style="width: 240px;"><span style="font-size: 10pt; color: #7e8c8d;">[v]Ідентифікаційний код</span></td>
                <td><span style="font-size: 10pt;">' . $user->code . '</span></td>
                </tr>';
        }

        $template .= '</tbody>
            </table>
            </div>
            <hr />
            %data%
            <hr />
            <p><span style="font-size: 10pt;">Деталі, що містяться у цьому ЗВІТІ, не повинні використовуватися як єдина причина прийняття рішення щодо Кандидата. Цей ЗВІТ розроблено для Замовника, а відтак, Screen Bot не несе відповідальність перед будь-якою іншою особою за будь-які можливі збитки, понесені у зв’язку з використанням цього ЗВІТУ.</span></p>';
        // SCREEN
        $screen_data = "";
        $screen_data .= "<h4>Перевірка кандидата у відкритих джерелах</h4>";

        ////////////////////////////// START
        $screen_data .= '<p style="font-size: 10pt; color: #7e8c8d;">Наявність відкритого ФОП</p>';
        $fop = FOP::where('FIO', 'LIKE', $candidate)->get();
        $fop_buffer = [];
        foreach ($fop as $fo) {
            array_push($fop_buffer, collect($fo)->all());
        }
        if (count($fop_buffer) > 0) {
            foreach ($fop_buffer as $item) {
                $screen_data .= '<blockquote><span style="background-color: #f8cac6; color: #ba372a; font-size: 10pt;">[x]ФОП зареєстровано! Адреса реєстрації: ' . $item["ADDRESS"] . '. КВЕД: ' . $item["KVED"] . '.</span></blockquote>';
```

```
            }
        } else {
            $screen_data .= '<blockquote><span style="color: #169179; background-color: #bfedd2; font-size: 10pt;">[v]Перевірка пройшла успішно! Дані про кандидата в пошуковому масиві відсутні</span></blockquote>';
        }
        ////////////////////////////////
        $screen_data .= '<p style="font-size: 10pt; color: #7e8c8d;">Перевірка на причетність до збанкрутілих компаній</p>';
        $rba = RBA::where('NAME', 'LIKE', $candidate)->get();
        $rba_buffer = [];
        foreach ($rba as $rb) {
            array_push($rba_buffer, collect($rb)->all());
        }
        if (count($rba_buffer) > 0) {
            foreach ($rba_buffer as $item) {
                $screen_data .= '<blockquote><span style="background-color: #f8cac6; color: #ba372a; font-size: 10pt;">[x]Інформація знайдена! Наявний запис № ' . $item["RN_NUM"] . '. від ' . $item["REG_DATE"] . ', компанія ' . $item["NAME"] . ', статус: ' . $item["PROC_BORG_NAME"] . '</span></blockquote>';
            }
        } else {
            $screen_data .= '<blockquote><span style="color: #169179; background-color: #bfedd2; font-size: 10pt;">[v]Перевірка пройшла успішно! Дані про кандидата в пошуковому масиві відсутні</span></blockquote>';
        }
        ////////////////////////////////
        $screen_data .= '<p style="font-size: 10pt; color: #7e8c8d;">На застосування положення Закону України Про очищення влади</p>';
        $lustr = LUSTR::where('FIO', 'LIKE', $candidate)->get();
        $lustr_buffer = [];
        foreach ($lustr as $lust) {
            array_push($lustr_buffer, collect($lust)->all());
        }
        if (count($lustr_buffer) > 0) {
```

```
                foreach ($lustr_buffer as $item) {
                    $screen_data .= '<blockquote><span style="background-color: #f8cac6; color: #ba372a; font-size: 10pt;">[x]Запис знайдено! Посада: ' . $item["JOB"] . ', стаття: ' . $item["JUDGMENT_COMPOSITION"] . ', термін: ' . $item["PERIOD"] . '</span></blockquote>';
                }
            } else {
                $screen_data .= '<blockquote><span style="color: #169179; background-color: #bfedd2; font-size: 10pt;">[v]Перевірка пройшла успішно! Дані про кандидата в пошуковому масиві відсутні</span></blockquote>';
            }
            ////////////////////////////////
            $screen_data .= '<p style="font-size: 10pt; color: #7e8c8d;">На вчинення корупційних правопорушень</p>';
            $corruption = CORRUPTION::where('indFirstNameOnOffenseMoment', strtoupper($user->name))->where('indPatronymicOnOffenseMoment', strtoupper($user->patronim))->where('indLastNameOnOffenseMoment', strtoupper($user->surname))->get();
            $corruption_buffer = [];
            foreach ($corruption as $corr) {
                array_push($corruption_buffer, collect($corr)->all());
            }
            if (count($corruption_buffer) > 0) {
                foreach ($lustr_buffer as $item) {
                    $posada = "";
                    $vurok = "";
                    $work = "";
                    if ($item["Column1.indPositionOnOffenseMoment"] != "-" && $item["Column1.indPositionOnOffenseMoment"] != "") {
                        $posada = ' посада: ' . $item["Column1.indPositionOnOffenseMoment"];
                    }
                    if ($item["Column1.codexArticles.codexArticleName"] != "" && $item["Column1.codexArticles.codexArticleName"] != "-") {
                        $vurok = ' рішення по справі: ' . $item["Column1.codexArticles.codexArticleName"];
                    }
```

```
                    if ($item["Column1.indPleaceOfWorkOnOffenseMoment"] != "" &&
$item["Column1.indPleaceOfWorkOnOffenseMoment"] != "-") {
                        $work = ' місце роботи: ' .
$item["Column1.indPleaceOfWorkOnOffenseMoment"];
                    }

                    $screen_data .= '<blockquote><span style="background-color: #f8cac6; color:
#ba372a; font-size: 10pt;">[x]Запис знайдено від  ' . $item["Column1.sentenceDate"] . '!
Справа: ' . $item["Column1.offenceName"] . $posada . $work . $vurok .
'</span></blockquote>';
                }
            } else {
                $screen_data .= '<blockquote><span style="color: #169179; background-color:
#bfedd2; font-size: 10pt;">[v]Перевірка пройшла успішно! Дані про кандидата в
пошуковому масиві відсутні.</span></blockquote>';
            }
            ////////////////////////////////
            $screen_data .= '<p style="font-size: 10pt; color: #7e8c8d;">Перевірка у списку
судових справ, призначених для розгляду</p>';
            $sspdr = SSPDR::where('case_involved', 'LIKE', '%' . $candidate . '%')->get();
            $sspdr_buffer = [];
            foreach ($sspdr as $sspd) {
                array_push($sspdr_buffer, collect($sspd)->all());
            }
            if (count($sspdr_buffer) > 0) {
                foreach ($sspdr_buffer as $item) {
                    $data = ". Дані про справу відсутні або засекречено.";
                    if ($item["case_involved"] != null) {
                        $data = ' деталі справи:' . $item["case_involved"] . ", тип: " .
$item["case_description"];
                    }

                    $screen_data .= '<blockquote><span style="background-color: #f8cac6; color:
#ba372a; font-size: 10pt;">[x]Інформація знайдена! Справа від: ' . $item["date"] . ' №'
.$item["case"] .'. '. $item["case_description"] . '</span></blockquote>';
```

```
            }
        } else {
            $screen_data .= '<blockquote><span style="color: #169179; background-color: #bfedd2; font-size: 10pt;">[v]Перевірка пройшла успішно! Дані про кандидата в пошуковому масиві відсутні.</span></blockquote>';
        }
        ////////////////////////////////
        $screen_data .= '<p style="font-size: 10pt; color: #7e8c8d;">Перевірка у автоматизованій системі виконавчого провадження МЮУ</p>';
        $asvp = ASVP::where('DEBTOR_NAME', $candidate)->where('VP_BEGINDATE', '!=', '0000-00-00 00:00:00')->where('VP_STATE', '!=', 'Завершено')->get();
        $asvp_buffer = [];
        foreach ($asvp as $as) {
            array_push($asvp_buffer, collect($as)->all());
        }
        if (count($asvp_buffer) > 0) {
            foreach ($asvp_buffer as $item) {
                $screen_data .= '<blockquote><span style="background-color: #f8cac6; color: #ba372a; font-size: 10pt;">[x]Інформація знайдена! Позивач: ' . $item["CREDITOR_NAME"] . '. статус: ' . $item["VP_STATE"] . ', зареєстровано: ' . $item["VP_BEGINDATE"] . '</span></blockquote>';
            }
        } else {
            $screen_data .= '<blockquote><span style="color: #169179; background-color: #bfedd2; font-size: 10pt;">[v]Перевірка пройшла успішно! Дані про кандидата в пошуковому масиві відсутні.</span></blockquote>';
        }
        ////////////////////////////////
        $screen_data .= '<p style="font-size: 10pt; color: #7e8c8d;">Перевірка у єдиному реєстрі боржників МЮУ</p>';
        $debt = DEBT::where('DEBTOR_NAME', $candidate)->get();
        $debt_buffer = [];
        foreach ($debt as $deb) {
            array_push($debt_buffer, collect($deb)->all());
        }
```

```
            if (count($debt_buffer) > 0) {
                foreach ($debt_buffer as $item) {
                    $screen_data .= '<blockquote><span style="background-color: #f8cac6; color: #ba372a; font-size: 10pt;">[x]Інформація знайдена! Категорія боргу: ' . $item["VD_CAT"] . '</span></blockquote>';
                }
            } else {
                $screen_data .= '<blockquote><span style="color: #169179; background-color: #bfedd2; font-size: 10pt;">[v]Перевірка пройшла успішно! Дані про кандидата в пошуковому масиві відсутні.</span></blockquote>';
            }
            ////////////////////////////////
            $screen_data .= '<p style="font-size: 10pt; color: #7e8c8d;">Перевірка на наявність у розшукових масивах України</p>';

            $wanted = WANTED::where('FIRST_NAME_U', strtoupper($user->name))->where('MIDDLE_NAME_U', strtoupper($user->patronim))->where('LAST_NAME_U', strtoupper($user->surname))->get();
            $wanted_buffer = [];
            foreach ($wanted as $want) {
                array_push($wanted_buffer, collect($want)->all());
            }
            if (count($wanted_buffer) > 0) {
                foreach ($wanted_buffer as $item) {
                    if ($item["ARTICLE_CRIM"] === "") {
                        $item["ARTICLE_CRIM"] = "дані відсутні";
                    }
                    $screen_data .= '<blockquote><span style="background-color: #f8cac6; color: #ba372a; font-size: 10pt;">[x]Інформація знайдена! Розшукується в: ' . $item["OVD"] . ', категорія в: ' . $item["CATEGORY"] . ' від: ' . $item["LOST_DATE"] . ', стаття: ' . $item["ARTICLE_CRIM"] . '</span></blockquote>';
                }
            } else {
                $screen_data .= '<blockquote><span style="color: #169179; background-color: #bfedd2; font-size: 10pt;">[v]Перевірка пройшла успішно! Дані про кандидата в
```

```
пошуковому масиві відсутні.</span></blockquote>';
                }
                ////////////////////////////
                if ($user->passport) {
                    $screen_data .= '<p style="font-size: 10pt; color: #7e8c8d;">Перевірка паспорту
в масивах втрачених паспортів</p>';
                    $search = $user->passport;
                    if (strpos($search, ' ') > 0) {
                        $search = explode(" ", $search);
                    } else {
                        $search_buffer = [];
                        array_push($search_buffer, strtoupper(preg_replace('/[0-9]+/', "", $search)));
                        array_push($search_buffer, preg_replace("/[^0-9.]/", "", $search));
                        $search = $search_buffer;
                    }
                    $passport = null;
                    if ($search[0] != '') {
                        $passport = PASSPORT::where('D_NUMBER', $search[1])->get();
                    } else {
                        $passport = PASSPORT::where('D_NUMBER', $search[1])-
>where('D_SERIES', $search[0])->get();
                    }
                    $passport_buffer = [];
                    foreach ($passport as $pass) {
                        array_push($passport_buffer, collect($pass)->all());
                    }
                    if (count($passport_buffer) > 0) {
                        foreach ($wanted_buffer as $item) {
                            $screen_data .= '<blockquote><span style="background-color: #f8cac6;
color: #ba372a; font-size: 10pt;">[x]Інформація знайдена! Тип паспорту: ' .
$item["D_TYPE"] . ', статус: ' . $item["D_STATUS"] . ' дата втрати: ' .
$item["THEFT_DATA"] . '</span></blockquote>';
                        }
                    } else {
                        $screen_data .= '<blockquote><span style="color: #169179; background-color:
```

```
#bfedd2; font-size: 10pt;">[v]Перевірка пройшла успішно! Дані про кандидата в пошуковому масиві відсутні.</span></blockquote>';
                }
            }
            ////////////////////////////// END

            $template = str_replace("%data%", $screen_data, $template);
            $warning = '<img class="icon" src="img/!.png">';
            $alert = '<img class="icon" src="img/x.png">';
            $ok = '<img class="icon" src="img/v.png">';
            $template = str_replace("[!]", $warning, $template);
            $template = str_replace("[x]", $alert, $template);
            $template = str_replace("[v]", $ok, $template);

            $pdf = Pdf::loadView('report_pdf', ['template' => $template]);
            $pdf->set_option('isHtml5ParserEnabled', 'true');
            Storage::put('tmp/save-telegram/report_' . $user->chat_id . '.pdf', $pdf->output());

            $post_file = array(
                'chat_id' => $user->chat_id,
                'document' => curl_file_create(Storage::path('tmp/save-telegram/report_' . $user->chat_id . '.pdf'))
            );
            $send_file_url = $bot . '/sendDocument?chat_id=' . $user->chat_id;
            $ch = curl_init();
            curl_setopt($ch, CURLOPT_HTTPHEADER, array(
                "Content-Type:multipart/form-data"
            ));
            curl_setopt($ch, CURLOPT_URL, $send_file_url);
            curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
            curl_setopt($ch, CURLOPT_POSTFIELDS, $post_file);

            file_get_contents($bot . "/sendmessage?chat_id=".$user->chat_id."&text=" . urlencode("🐞 Ось результат скринінгу по кандидату ".$candidate));
```

```
                $output = curl_exec($ch);
                curl_close($ch);
                Storage::delete('tmp/save-telegram/report_' . $user->chat_id . '.pdf');
                $user = Requests::all()->where('chat_id', $user->chat_id)->first();

                $reply_markup = json_encode([
                   'keyboard' => [['Розпочати новий скринінг    ']],
                   'resize_keyboard' => true,
                   'one_time_keyboard' => true,
                ]);
                file_get_contents($bot . "/sendmessage?chat_id=".$user->chat_id."&text=" .
urlencode("❗Нагадуємо, дана інформація не є на 100% достовірною і рекомендуємо її
додатково обговорити з кандидатом.")."&reply_markup=" . $reply_markup);
                $user->status = 'sended';
                $user->save();
            }
        }
```

ДОДАТОК Г CD-диск з програмним кодом