ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Система пошуку знайомств за інтересами на основі методів машинного навчання з використанням чат-боту»

на здобуття освітнього ступеня бакалавра
зі спеціальності 122 Комп'ютерні науки
(код, найменування спеціальності)
освітньо-професійної програми Штучний інтелект
(назва)

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

_____ Олексій ОЛІЙНИК
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав:
здобувача вищої освіти Олексій ОЛІЙНИК
група ШІД-42

Керівник: Євгеній ЧИЧКАРЬОВ
науковий ступінь, вчене звання д.т.н., професор

Рецензент:
науковий ступінь, вчене звання _____
Ім'я, ПРІЗВИЩЕ

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Штучного інтелекту

Ступінь вищої освіти Бакалавр

Спеціальність Комп'ютерні науки

Освітньо-професійна програма Штучний інтелект

ЗАТВЕРДЖУЮ
Завідувач кафедрою Штучного інтелекту

_____ Ольга ЗІНЧЕНКО
«_____» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Олійнику Олексію В'ячеславовичу
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Система пошуку знайомств за інтересами на основі методів машинного навчання з використанням чат-боту

керівник кваліфікаційної роботи Євгеній ЧИЧКАРЬОВ д.т.н., професор,
(Ім'я, ПРІЗВИЩЕ науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» лютого 2025 р. № 56

2. Строк подання кваліфікаційної роботи «23» травня 2025 р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
Дослідження принципів побудови чат-бота
Аналіз технологій машинного навчання та можливість застосування
Розробка вимог до моделі

5. Перелік графічного матеріалу: *презентація*

1. Вимоги до застосунку
2. Інтерфейс чат-боту
3. Механізм пошуку кандидатів
4. Інтеграція інтелектуальної моделі

6. Дата видачі завдання «25» лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз сучасних методів машинного навчання та їх можливостей	25.02-09.03.25	Виконано
2	Дослідження основних технологій розробки чат-ботів	10.03-20.03.25	Виконано
3	Аналіз засобів розпізнавання природної мови	21.03-28.03.25	Виконано
4	Розробка функціоналу чат-боту	29.03-03.05.25	Виконано
5	Інтеграція інтелектуальної моделі	03.05-13.05.25	Виконано
6	Тестування чат-боту	14.05-20.05.25	Виконано
7	Оформлення роботи: вступ, висновки, реферат	21.05-27.05.25	Виконано
8	Розробка демонстраційних матеріалів	28.05-31.05.25	Виконано

Здобувача вищої освіти _____ Олексій ОЛІЙНИК
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Євгеній ЧИЧКАРЬОВ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 53 стор., 26 рис., 17 джерел.

Мета роботи – створення чат-боту, який дозволить користувачам знаходити нові знайомства без будь-яких фізичних зусиль, для спрощення процесу знаходження людей з схожими інтересами.

Об'єкт дослідження – методи та засоби створення чат-ботів для знаходження людей зі схожими інтересами, за допомогою методів машинного навчання для кращого пошуку.

Предмет дослідження – чат-бот для пошуку знайомств, за допомогою методів машинного навчання.

Короткий зміст роботи: У роботі було проведено аналіз сучасний інтелектуальних моделей машинного навчання, їх інтеграції та можливості. Проаналізовано засоби розпізнавання природної мови для ефективного пошуку ідеального кандидата.

Розглянуто основні технології, за допомогою яких, ми маємо можливість створювати чат-боти, для знаходження людей з схожими захопленнями.

Використано засоби розробки чат-ботів - Java, а також TelegramBot API для розробки чат-бота, який використовуються у соціальній мережі Telegram. Розроблено алгоритм розпізнавання текстових описів користувачів та обробку для створення їх векторного представлення.

КЛЮЧОВІ СЛОВА: JAVA, PYTHON, INTELLIJ IDEA, BERT, МОДЕЛЬ, API, ЧАТ-БОТ

ABSTRACT

Text part of the bachelor's qualification work: 53 pages, 26 pictures, 17 sources.

Object of research – methods and tools for creating chatbots to find people with similar interests, using machine learning methods for better search

Subject of research – chatbot to finding acquaintanceship, using machine learning methods

Summary of the work: The work analyzed modern intelligent machine learning models, their integration and capabilities. Natural language recognition tools were analyzed for the effective search for the ideal candidate.

The main technologies were considered, with the help of which we have the opportunity to create chatbots to find people with similar hobbies.

Chatbot development tools - Java, as well as TelegramBot API were used to develop a chatbot used in the Telegram social network. An algorithm for recognizing text description of users and processing them to create a vector representation was developed.

KEYWORDS: JAVA, PYTHON, INTELLIJ IDEA, BERT, MODEL, API, CHAT-BOT

ЗМІСТ

ВСТУП

У сучасномі світі популярність різного роду месенджерів для знайомств, чат - ботів, а також сайтів, важко недооцінювати. Вони стали невід'ємною частиною багатьох людей, адже ці інструменти це зручний і швидкий пошук просто спілкування, чи то другої половинки чи навіть прогулянки.

Майже кожна людина має якийсь встановлений додаток для знайомств, проте молоде покоління надає більшу перевагу чат-ботам в месенджері Telegram. Такі чат-боти наразі мають не аби яку популярність за рахунок зручності та простоти в використанні. Найчастіше такі чат-боти чи то сайти, пропонують простий підбір, який дозволяє виконувати пошук лише за деякими критеріями, які можно вибрати зі списку готових, або ж просто створити опис свого профілю в тій, чи іншій мережі. Точний підбір в таких випадках не гарантується.

Ціль проекту полягає в тому, щоб зробити цей пошук більш точним та персоналізованим, щоб підібрати ідеального кандитада. Ми прагнемо розробити рішення, яке пропонуватиме користувачам не только підбір по регіону чи по статі або віку, а підбір на аналізі їхніх уподобань, хобі та інших моментів, які користувач вважає важливим для свого майбутнього партнера.

Для досягнення поставленої мети потрібно пройти такі етапи розробки:

1. Проаналізувати предметну область:
    - Провести огляд мессенджеру Telegram та його функціоналу
    - Розглянути визначення "чат-бот" та "інтелектуальна модель"
    - Проаналізувати схожі чат-боти та їх механізми пошуку
2. Вибір засобів реалізації:
    - Проаналізувати мови програмування для створення інтерфейсу чат-боту та його функціоналу
    - Проаналізувати Telegram Bot API для створення цього ж чат-боту

- Проаналізувати системи керування базами даних, для зберігання даних користувачів
- Проаналізувати системи керування версіями
- Провести огляд документації по нашій інтелектуальній моделі

3. Проектування та реалізація чат-бота з використанням нашої інтелектуальної моделі для покращення пошуку:
    - Побудувати діаграму діяльності
    - Побудувати структурну діаграму
    - Визначити основних механізм пошуку та підбору користувачів
    - Провести тестування чат-боту

1 АНАЛІЗ ІНТЕЛЕКТУАЛЬНИХ СИСТЕМ І МОДЕЛЕЙ

1.1 Поняття інтелектуальної моделі

Інтелектуальна модель LLM(Large Language Model) - одна із програм ШІ, яка здатна розпізнавати текст, який їй направляється, а також вміє генерувати його. Навчання моделей проводиться з великим об’ємом даних, а також побудовані за допомогою машинного навчання. Модель будується на моделі - трансформері, це набір нейромереж, кожна з яких складається з кодера та декодера. Ці трансформери мають можливість самонавчання, що дозволяє їм навчатись без нагляду зі сторони. Завдяки цьому процесу трансформери навчилися розуміти базову граматику тої чи іншої мови, а також “засвоювати” знання на практиці.

LLM - це програма, з великим об’ємом накопичених шаблонів, для того аби мати можливість розпізнавання та інтерпретування людської мови. Навчання більшості мовних моделей проводиться на основі даних, які є у всесвітній павутині, мільйонів терабайтів тексту. Добре підібрані набори даних надають більшу продуктивність, ніж використання “всього і вся”.

Великі мовні моделі дуже гнучкі, одна така модель може виконувати низку різних задач, такі як: відповідати на запитання, аналізувати документи, виконувати переводи текстів тощо. LLM можуть сильно вплинути на використання користувачів, доволі звичних нам, пошукових систем(Google, Bing, DuckDuckGo) та віртуальних помічників(Siri, Google Assistant, Cortana, Alexa).

Звісно LLM не ідеальні, у кожної моделі є свої переваги та недоліки. Вони можуть продемонструвати прогнози на основі невеликих підказок. Для генеративного штучного інтелекту також можна використовувати LLM, вони досить непогано справляються з задачами створення контенту, на основі вхідних даних. Проте є і негативні моменти, якщо використовувати модель для відповіді на запитання користувача, то вона будет генерувати їх на основі даних, на яких

вона навчалась, що може зробити її відповіді ненадійними. Тобто, модель можна навчити на основі неправдивих даних, то відповіді будуть на основі цих даних та не будет відповідати дійсності. Побічним ефектом навчання моделі, навіть на основі правдивих даних, може бути, те що вона може генерувати вигадану інформацію. Це відбувається, якщо модель не може надати точної інформації, намагаючись надати її, схожу на правдиву.

Ще одним із суттєвих недоліків є безпека даних. Якщо користувач завантажить в LLM власні конфіденційні дані, для того, аби підвищити продуктивність генерації відповідей на свої запити, то є великий ризик, що ці дані в подальшому може використати якийсь недобрий користувач в поганих цілях. Тобто, LLM не є безпечним сховищем, якому ви можете надавати свої конфіденційні дані, через можливість поширення їх між іншими користувачами.

Так зване LLM складається із кількох рівнів. На базовому воно побудоване на машинному навчанні (ML). Це одна із галузей штучного інтелекту, але навчання її проводиться на основі великої кількості вхідних даних, що дозволяє визначати характеристики цих даних без втручання наглядача. Також існує глибоке навчання, так зване, Deep Learning, котре використовую якраз LLM, завдяки цьому моделі можуть самі розпізнавати відмінності, наприклад, між фотографіями чи чимось іншим, без втручання людини.

Щоб забезпечити глибоке навчання, великі мовні моделі (LLM) базуються на нейронних мережах. Подібно до людського мозку, де нейрони з’єднуються і передають сигнали, штучні нейронні мережі складаються з вузлів, організованих у шари: вхідний, вихідний та один або кілька проміжних. Шари обмінюються інформацією, якщо їхній вихід перевищує певний поріг.

Для LLM використовується специфічний тип нейронної мережі — модель трансформера. Ці моделі здатні аналізувати контекст, що критично важливо для людської мови, яка залежить від контекстуальних зв’язків. Трансформери застосовують техніку «самоуваги», яка дозволяє виявляти тонкі зв’язки між елементами послідовності. Завдяки цьому вони краще розуміють контекст

порівняно з іншими методами машинного навчання. Наприклад, моделі можуть пов’язувати початок і кінець речення або різні речення в абзаці. Це дозволяє LLM інтерпретувати неоднозначну чи нову мову, розпізнаючи семантичні зв’язки між словами та поняттями на основі їхнього значення.

1.2 Огляд додатків та чат-ботів для знайомств

В сучасному світі ми маємо досить багато простих та обмежених рішень для знайомств в мережі, але навіть незважаючи на це, їх популярність серед молоді досить велика. Давайте коротко обговоримо найпопулярніші рішення та їх недоліки:

Tinder - мобільний додаток, розроблений американською компанією IAC у 2012 році. В Україні був запущений лише на початку 2015 року, та досить швидко набрав популярність серед молодого покоління, віком до 34 років. Як і всі додатки для знайомств він не пропонує глибокого аналізу сумісності за інтересами користувачів. Tinder сприяв популяризації такого роду знайомств, де акцент йде в першу чергу на зовнішність. Це викликало багато дискусій в суспільстві, деякі навіть почали зневажати такий формат знайомств.

Bumble - інтернет платформа для знайомств в режимі Date, BFF або пошук наставників для різного роду занять. Платформа дозволяє зробити перший крок лише жінкам, що робить платформу більш безпечною для жіночої статі. Як і попередній кандидат, Buble пропонує регіональних пошук та базові фільтри, такі як він чи стать, які не можуть підібрати для користувача ідеального кандидата.

Обмеження ініціатив чоловіків може зменшувати активність користувачів платформи у консервативних регіонах. Однак, режим Bizz(пошук наставників), робить платформу універсальнішою, ніж той же Tinder, але популярність в Україні залишається низькою.

Дайвінчик (в оригіналі - Дайвинчик) - популярний телеграм чат-бот в СНД країнах, запущений вперше в 2015 році в нині забороненій соціальній

мережі ВКонтакте. Наразі має 16 мільйонів унікальних користувачів щомісяця, лише в Телеграм. На жаль, також пошук лише по вікові та регіону, та сусіднім населеним пунктам. Чат-бот став культурним феноменом серед молодого покоління, завдяки простоті та зручності, кількість активних користувачів стає більшою з кожним місяцем.

Усі згадані платформи для знайомств мають спільну особливість — поверхове підбирання співрозмовників. Фільтри, обмежені такими параметрами, як вік, стать чи місце розташування, не враховують глибших аспектів сумісності, наприклад, спільних інтересів, цінностей чи життєвих цілей. Як наслідок, користувачі витрачають багато часу на спілкування, яке рідко переростає у щось серйозне. Крім того, акцент на зовнішності та швидкості знайомств може створювати тиск, знижуючи якість взаємодії. У майбутньому платформи могли б використовувати складніші алгоритми, такі як аналіз психологічної сумісності чи штучний інтелект, для точнішого підбору партнерів на основі детальних профілів. Це зробило б онлайн-знайомства більш змістовними та результативними.

Популярність цих додатків і чат-ботів не усуває їхньої спільної проблеми - поверхневого підходу до пошуку співрозмовників із мінімальними фільтрами, як-от вік чи геолокація. Це змушує користувачів витрачати багато часу на пошук людини зі схожими інтересами. Розробники мають ресурси для впровадження інноваційних рішень, таких як розумні алгоритми чи чат-боти, які сприяли б якісному підбору та формуванню міцніших, тривалих стосунків.

1.3 Висновки до розділу 1

Беручи до уваги вищесказане, можна зробити висновки, що інтелектуальні мовні моделі це незамінні технології для теперішньому світі та звичайно і в майбутньому. Використання ML та нейронних мереж може спростити більшість задач для користувачів та розробників. Моделі типу Transformer широко використовуються для розуміння контексту та відповідей на запити користувачів.

Ці моделі використовуються в більшості додатків, для покращення користуванням їх сервісом.

Створення ботів в телеграм вже вийшло зовсім на новий рівень, розробники можуть створити бота навіть для ігор, не говорячи щось про простенький веб-додаток. Ці технології відкривають нові можливості в створенні ботів для різних галузей та різних цілей.

2 ЗАСОБИ РЕАЛІЗАЦІЇ ЧАТ-БОТУ ДЛЯ РОЗПІЗНАВАННЯ ПРИРОДНОЇ МОВИ

2.1 Засоби для розпізнавання природної мови

Обробка природної мови або Neutral Language Processing - технологія галузі штучного інтелекту, яка розвивається досить активно в теперішньому середовищі та допомагає користувачам або бізнесам отримати не аби яку користь від її використання. Аналітичні дослідження у США прогнозують, що з 2021 року до 2027 буде збільшення глобального ринку по обробці природної мови з 21 мільярдів доларів США до 127 мільярдів доларів з середньорічним темпом зростання у 29.4 відсотків.

Так що ж таке NLP з точки зору технології - це технологічний напрямок ШІ для розуміння, аналізу та відтворення природної мови машиною до користувача, який до неї звертається. У 1950 році у філософському журналі Mind було опубліковано статтю про емпіричний тест, котрий запропонував Алан Тьюрінг. Ціллю тесту було дізнатися чи вміє машина “мислити”. Його можно інтерпретувати як те, що якщо людина, коли спілкується з комп'ютером, не може визначити, що з нею спілкується саме машина, а не справжня людина. Цей тест став точкою відліку заснування та розвитку NLP.

На сьогодні важко уявити галузь, в якій не мала б застосування NLP технологія. Це Siri, Alexa - розумні асистенти в наших телефонах, Cortana на нашому комп'ютері чи ноутбуці, автоматичне фільтрування нашої пошти за допомогою Gmail, перевірка текстів на плагіат, класифікація текстів тощо.

На перечислення галузей, де використовується NLP можна витратити купу часу, але можна відокремити декілька напрямків:

- Машинний переклад: переклад тексту, з подальшим виходом на іноземні ринки тощо

- Чат-боти: можна використовувати для клієнтської підтримки 24 години на день, 7 днів на тиждень
- Розпізнавання мови: перетворення текст -> аудіо, чи наоборот тощо
- Генерація тексту: NLP гарно себе показує в завданнях створення контенту
- Генерація субтитрів до іншомовних відео: з не знанням іноземної мови важко було-б дивитися фільми, чи якісь навчальні ролики, але NLP може допомогти вам в цьому

NLP можуть навіть використовуватись в нестандартних задачах. Багато організацій потребують NLP як помічника в структуруванні даних. NLP гарно справляється з аналізом документації та її класифікації. Також NLP застосовують і в медицині для обслуговування клієнтів, ведення їх медичних карток, а також пошуків термінів у літературі. На основі NLP були створені лікарі-роботи, для аналізу симптомів та поставлення діагнозів, а також відстеження перебігу хвороб піцієнта. Більш того, можливості прогнозування дозволяють застосовувати NLP у кримінальній галузі, що може допомогти уникнути кримінальній діяльності, і зменшити їх відсоток. Принцип роботи заключається в аналізі злочинної діяльності, їх методах шифрування від поліції тощо.

Популярність NLP тільки зростає, тому про її не актуальність навіть в найближчі кілька десятків років ніякої мови навіть бути не може. Багато сфер діяльності ніяк не можуть обійтись без NLP через те, що накопичиться велика кількість задач, які вона робить набагато продуктивніше, ніж людина.

2.2 Мовна модель Bert, її особливості та порівняння з другими мовними моделями

Bert - це модель, яка має змогу навчатися розуміти контекст слів у реченні, аналізуючи їх зліва направо та навпаки, справа наліво, розробником якої є всесвітньо відома компанія Google. Якщо порівнювати її з попередніми моделями, для прикладу Word2Vec або GloVe, які є контекстно-незалежними, то

Bert аналізує повний контекст речення, для кращої ефективності завдань NLP. Наразі Google застосовує Bert для кращого розуміння пошуків користувачів.

Особливості моделі дозволяють застосовувати її для різного типу задач, наприклад відповіді на запитання, класифікації текстів, розпізнавання іменованих сутностей, машинний переклад тощо.

Bert дозволяє користувачам знаходити інформацію в всесвітній павутині, не вимагаючи точних запитів, а розуміючи їх так, якби ви спілкувалися в реальному житті. Алгоритм здатний розуміти наміри та контекст запиту, беручи до уваги всю фразу, а не лише деякі слова.

Оригінальна модель постачається у двох попередньо натренированих моделях:

- Base: нейромережна архітектура, яка складається з 12 шарів, 768 прихованих, 12 голів та 110 мільйонами параметрів
- Large: нейромережна архітектура, яка складається 24 шарів, 1024 прихованих, 16 голів, а також 340 мільйонів параметрів

Обидві було натренеровані на BookCorpus, що включають в собі 800 мільйонів слів, а також на одній із версій Вікіпедії з 2500 мільйонами слів.

Ключовими особливостями Bert є:

- Двонаправлене розуміння: модель читає текст з двох сторін, щоб краще розуміти його значення
- Контекстний аналіз: аналіз проводиться на повне речення, не беручи до уваги лише ключові слова, що допомагає зрозуміти всі нюанси запиту.
- Open source code: код моделі знаходиться у відкритому доступі, що дозволяю розробникам вносити зміни в модель або краще зрозуміти принцип її роботи

Не дивлячись на низку особливостей, не потрібно й забувати, що у всього є свої недоліки, навіть у такої моделі:

- Великі обчислювальні ресурси
- Складність до тонкого налаштування задля адаптації під конкретні завдання
- Обмеження для великих текстів

Модель Bert працює у два етапи:

1) Попереднє навчання: Bert навчається на великих об'ємах текстів за допомогою:
    - Masked Language Model: 15% слів в тексті маскуються під токен, і модель передбачає ці слова, коли аналізує текст
    - Next Sequence Prediction: Bert навчається визначати взаємопов'язаність двох послідовних речень
2) Тонке налаштування: після попереднього навчання моделі, її навчають на більш менших наборах даних, аби адаптувати її під конкретні завдання NLP, наприклад класифікація текстів

BERT vs GPT:

- Архітектура: наша модель - це двонаправлена модель(принцип був описаний раніше), в той час як популярний нині GPT - однонаправлена(аналіз тексту йде зліва направо), а також використовується декодер трансформера
- Задачі: наша модель краще підходить для задач розуміння тексту(класифікація, відповіді на запитання тощо), в той час GPT - для генерації текстів
- Навчання: наша модель використовує задачі Masked Language Model та Next Sentence Prediction, в той час як GPT навчається передбачати наступне слово в послідовності

BERT vs ELMo:

- Контекст: ELMo також двонаправлена, але основана на LSTM, а не на трансформерах, що робить його менш ефективним на великих

об'ємах даних. Bert використовує трансформери, що дає йому перевагу в масштабності

- Глибина: наша модель має більшу глибину навчання та навчається одразу вся, в той час як ELMo добавляє контекстні ембедінги до вже існуючих моделей
- Продуктивність: наша модель зазвичай показує себе краще, ніж наш кандидат для порівняння, оскільки Bert має більш сучасну архітектуру

BERT vs RoBERTa:

- Покращення: RoBERTa, яку розробила компанія Facebook(зараз Meta) - оптимізована версія Bert. Вона видаляє NSP, збільшує об'єм навчальних даних, а також використовує динамічне маскування
- Результати: за рахунок більш ретельного налаштування RoBERTa показує себе краще в бенчмарках, наприклад такі як GLUE
- Схожість: обидві наші моделі двонаправлені та базуються на трансформерах

2.3 Огляд мови програмування Java

Java - одна із об'єктно - орієнтованих мов програмування, яка була розроблена компанією Sun Microsystems у 1995 році, основним компонентом для платформи Java. В 2009 році права на цю мову програмування перейшли до рук нині доволі крупної компанії по розробці програмного забезпечення - Oracle.

Зародження Java почалось в 1991 році в лабораторіях компанії Sun Microsystems. Засновником проекту був канадським інформатиком Джеймс Гослінгом, проект було названо - Green(в перекладі на українську - “зелений”). Розробка першої робочої версії, із назвою - Oak(в перекладі на українську - “дуб”), зайняла цілих 18 місяців. Невдовзі виявилось, що назва Oak вже використовувалась іншою компанією, в результаті цього розпочались тривалі

суперечки навколо назви нової мови програмування. Після їх закінчення, у 1995 році, було прийнято рішення офіційно перейменувати в Java.

Oracle надає розробникам компілятор Java, а також віртуальну машину Java. В офіційній реалізації програми, які написані на Java, компілюються у байт-код, який інтерпретується віртуальною машиною при виконанні для конкретної платформи. Синтаксис нагадує C та C++, зокрема як основну об’єктну модель було взято C++ та дещо модифікували її. Було усунуто можливість виникнення певних конфліктних ситуацій, спричинених помилками програміста, та було спрощено процес створення об'єктно - орієнтованих програм. Більшість операцій, які в C/C++ виконувались вручну програмістами, було доручено віртуальній Java машині. Java насамперед створювалася як платформно-незалежна мова, тому вона має обмежені низькорівневі можливості для взаємодії з апаратним забезпеченням, що порівняно з C++ знижує швидкість виконання програм. За потреби таких функцій Java дозволяє викликати підпрограми, написані на інших мовах програмування

Сучасна Java є однією з найпопулярніших мов програмування у світі IT. Завдяки своїй універсальності, надійності, а також кросплатформеності, Java використовується для різного роду завдань - від створення консольних застосунків до розробки корпоративних систем і вбудованих пристроїв. Також Java має низку особливостей, які забезпечують її популярність серед розробників:

- Кросплатформеність: завдяки розробленій віртуальній машині, програми, які були написані на Java, можуть працювати на будь-якому пристрої чи ОС, на якому було попередньо встановлено JVM
- ООП: Java підтримує основі принципи об’єктно - орієнтованого програмування
- Синтаксис: синтаксис Java базується на C та C++, але є більш спрощеним варіантом, що робить його user-friendly

- Автоматичне керування пам'яттю: Garbage collection автоматично звільняє пам'ять від об'єктів, які більше не використовуються
- Безпека: Java має вбудовані механізми для ізолювання коду в JVM, що дозволяє запобігати доступу до системи третіми користувачами
- Багатопоточність: Java надає інструменти для створення додатків з багатопоточністю, що в важливим для більшості сучасних систим
- Багата екосистема: Java має досить розвинену стандартну бібліотеку, а також велику кількість різних фреймворків чи інструментів для розробки

На дивлячись на таку велику кількість позитивних факторів, не потрібно й забувати про те, що у кожної мови програмування є і свої недоліки:

- Якщо порівнювати Java з іншими мовами програмування, то вона має більш вищі витрати ресурсів
- Вважається нелегкою мовою програмування для новачків, якщо порівнювати наприклад з Python
- Повільний запуск додатків через необхідність ініціалізації JVM

Java, на відміну від інших мов програмування складається з трьох ключових компонентів:

- JDK (Java Development Kit): набір інструментів для розробки, це включає в себе і компілятор, і бібліотеки і різні інші утиліти
- JRE (Java Runtime Environment): середовище виконання, яке містить в собі JVM та стандартні бібліотеки Java
- JVM (Java Virtual Machine): віртуальна машина, для інтерпретації в байт-код і забезпечення його виконання на конкретній платформі

Мову програмування Java можна використовувати для написання більшості видів програм, які наразі відомі людству, для прикладу:

- Web-розробка: для створення веб застосунків нерідко використовують такі фреймворки як Spring та Java EE (Enterprise Edition)

- Мобільна розробка: для створення мобільних застосунків використовують Android SDK
- Корпоративні системи: Java досить часто використовують для розробки банківських систем чи ERP-систем, через свою безпеку
- Вбудовані системи: Java ME (Micro Edition) застосовується в IoT-пристроях та смарт-картах

На 2025 рік Java залишається однією з найкращих мов програмування. За індексом TIOBE, вона стабільно входить до трійки найпопулярніших мов. Постійні оновлення додають все нових і нових можливостей, також покращується продуктивність, модульність та інше. Навіть в далекому майбутньому Java зберігатиме свої позиції через:

- Розвиток хмарних технологій, де Java активно використовується
- Популярність Android застосунків
- Постійні вдосконалення для роботи з великими даними та штучним інтелектом

2.4 Огляд мови програмування Python

Python - інтерпретована об'єктно-орієнтована мова програмування високого рівня з динамічною типізацією. Її було розроблено в 1990 році нідерландським програмістом Гвідо Ван Россумом.

Історія створення Python почалась ще в далекому 1980 році співробітником голландського інституту CWI Гвідо Ван Россумом. Він хотів розподілення ОС Amoeba, тому йому була потрібна скриптова мова програмування, яка може розширюватись. Запозичивши деякі деталі з мови програмування ABC(в розробці якої Гвідо брав участь), на дозвіллі він почав писати, нині досить популярну мову програмування, Python. В лютому 1991 року у групі новим alt.sources Гвідо опублікував вихідний текст. З того моменту Python почав швидко розширюватися мережевою павутиною, та також сподобалась іншим

програмістам з усього світу. На даний момент ця мова програмування займає друге місце серед всіх мов програмування, які нині ми маємо.

Як і будь-яка інша МП, Python підтримує парадигми програмування, а також ООП. Його синтаксис досить простий, якщо порівнювати з іншими мовами програмування. Він має обширну базу інтерфейсів та різного роду бібліотек, які є дуже корисними для створення різних програм. Python дозволяє запускати та створювати код на різних інформаційних системах, що робить його мультисистемним.

На цьому його можливості не закінчуються. Подібно Prolog та Lisp в режимі відлагодження, інтерпретатор Python має інтерактивний режим роботи. В цьому режимі можна вводити вирази з клавіатури, а результат буде потрапляти на екран. Його використовують на тільки початківці, а й досвідчені програмісти, для тестування будь-якого фрагменту коду, перед тим як додавати його в основних код програми.

В інтерактивному режимі одразу є доступним і такий інструмент як Debug pdb та довідкова система, виклик якої відбувається при введені help(). Довідкова система працює для модулів, класів, та функцій, якщо це прописано в рядках документації (рис. 2.1).

```
>>> 2 ** 100 # піднесення 2 до 100-го степеня
1267650600228229401496703205376L
>>> from math import * # імпорт математичних функцій
>>> sin (pi * 0.5) # обчислення синуса від половини пі
1.0
>>> help (sorted) # допомогу по функції sorted
Help on built-in function sorted in module __builtin__:
sorted (…)
   sorted (iterable, cmp=none, key=none, reverse=false) -> new
sorted list
```

Рисунок 2.1 - Приклад виконання коду в інтерактивному режимі

Мова програмування Python побудована навколо ООП моделі програмування. Його розробник реалізував ООП досить специфічно, в порівнянні з іншими мовами програмування. Деякі можливості та особливості:

- класи є одночасно об'єктами
- успадкування
- поліморфізм
- інкапсуляція
- спеціальні методі(конструктори, деструктори, розподільники памяті)
- метапрограмування - управління створенням класів, тригери на створення класів тощо
- повна інтроспекція - можливість визначити тип та структуру потрібного об'єкта під час роботи програми

Також і не слід забувати про функціональне програмування. Python підтримує звичайну парадигму ФП, зокрема:

- функція є об'єктом
- рекурсія
- розвинена обробка списків
- аналог замикань
- каррінг

Програмне забезпечення на Python(додаток або бібліотека) організоване у вигляді окремих модулів, які збираються в пакети. Також ви можете зберегти модуль у каталозі або в ZIP-архіві, що може вплинути на розмір додатку чи бібліотеки. Модулі можна розділити на два типи:

- Перший тип: модулі, які написано виключно на мові програмування Python
- Другий тип: модулі розширення, створені на інших мовах програмування

Наприклад у стандартній бібліотеці Python є модуль pickle у чистому вигляді, тобто написаний виключно на Python та є такий же модуль, але на іншій мові програмування. Він представлений окремим файлом, а пакет - окремою

папкою. Для підключення модуля до програми потрібно використовувати оператор import. Після цього модуль стає іншим об'єктом, що надає доступ до цього простору імен. Модуль може бути перезавантажений під час роботи за допомогою функції reload().

Python має декілька можливостей, які значно відрізняють його від інших мов програмування. Наприклад, клас є об'єктом(як говорилось раніше), та також в операторі визначення класу можна використовувати вирази, які доступні батьківським класам (рис. 2.2):

```
def getClass():
    return dict
class D(getClass()):
    pass
d = D()
```

Рисунок 2.2 - Приклад використання виразу батьківських класів

Також можна модифікувати багато об'єктів під час виконання програми, наприклад класи (рис. 2.3):

```
>>> class X(object): pass
...
>>> y = X()
>>> y.wrongMethod() # такого методу поки немає
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'X' object has no attribute 'wrongMethod'
>>> X.wrongMethod = lambda self : 'im here' # додамо його
>>> y.wrongMethod() # так як доступ до методу призводить до
пошуку по __dict__ класу,
'im here' # то wrongMethod стає доступним всім екземплярам
```

Рисунок 2.3 - Приклад модифікації класу під час виконання програми

Мова програмування Python завдяки своїй обширній базі бібліотек дозволяє використовувати його для будь-яких завдань:

- розробка десктопних застосунків
- розробка серверної частини сайту/застосунку
- ігри
- вбудовані системи різних пристроїв
- скрипти та плагіни до уже готових застосунків, для їх автоматизації тощо
- автоматизоване тестування
- машинне навчання

2.5 Огляд фреймворка Flask

Flask - це веб-фреймворк, написаний для застосування у мові програмування Python. Його створив Армін Ронакер у 2012 році, він швидко здобув свою популярність завдяки своїй простоті та зручності у розробці. Flask належить до мікрофрейморків, які не мають великої кількість інструментів чи бібліотек, тож розробники мають змогу самим обирати лише ті компоненти, які їм потрібні для користування.

Основі особливості Flask:

- Простота: фреймворк має інтуїтивно зрозумілий API, що дозволяє розробникам-початківцям швидко створювати веб-застосунки.
- Вбудований сервер: Flask постачається з вбудованим веб-сервером для тестування
- Шаблонування: фреймворк підтримує шаблонізатор Jinja2, який дозволяє швидко створювати динамічні HTML сторінки без всяких зусиль
- Розширюваність: на Flask розроблено досить велику базу різного типу розширень, що дозволяє його адаптувати до будь-яких завдань

Flask часто використовують для створення різноманітних рішень, таких як: веб-застосунки(сайти блоги, адмін-панелі тощо), RESTful API для мобільних або

веб-застосунків, прототипів або MVP, систем управління контекстів та невеликих e-commerce платформ.

Flask базується на концепції WSGI, що забезпечує взаємодію між веб-сервером та додатком за всіма стандартами. Основні компоненти Flask:

- Роутинг: фреймворк дозволяє визначати маршрути для обробки запитів через декоратори
- Обробка запитів: підтримка різних методів, таких як POST, GET, PUT, DELETE дозволяє створювати повноцінні веб-застосунки
- Контекст застосунка: Flask використовує контексти для управління станов застосунка та запитів

Лістинг 2.1 - Вигляд звичайного веб-застосунка, який повертає “Hello, World!” за початковою адресою

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello, World!'

if __name__ == '__main__':
    app.run(debug=True)
```

Незважаючи на значні переваги, не потрібно забувати і про недоліки:

- Обмежена функціональність: для створення обширних проектів, потрібна більша кількість підключених бібліотек, а це тягне за собою більшу кількість часу для налаштування
- Продуктивність: Flask позиціонує себе як швидкий фреймворк для невеликих застосунків, але в великих проектах може знадобитись додаткове налаштування серверів

Flask є досить потужним інструментом для створення веб-застосунків, завдяки своїм особливостям. Його мінімалістичний підхід дозволяє розробникам зосередитися на створенні функціоналу, а не на боротьбі з складністю фреймворка. Незважаючи на деякі обмеження, Flask залишається одним із найпопулярніших виборів для Python-розробників, особливо для проєктів, що потребують швидкої реалізації та адаптивності.

2.6 Огляд Telegram API’s

Telegram пропонує розробникам три різних API для вирішення різного роду завдань. Ці інструменти включають в себе Telegram Bot API, Telegram API(разом з TDLib), а також Gateway API. Кожен із цих API має свої унікальні можливості та призначення, від створення чат-ботів до розробки своїх власних клієнтів Telegram. Усі вони наразі надаються безкоштовно, що робить їх доступними для будь-якого розробника. Крім того, Telegram пропонує розробникам можливість додавання віджетів на свій веб-сайт, а дизайнерам відкривається можливість для створення анімованих стікерів, емоджі або кастомних тем для клієнтів.

Почнемо з Telegram Bot API, ця API дозволяє створювати чат-боти, це спеціальні аккаунти, для створення яких не потрібен номер телефону. Вони функціонують як інтерфейс для коду, що виконується на сервері. Для роботи з цим API не потрібно розбиратись в тому як побудовані протоколи MTProto, оскільки проміжний сервер Telegram бере на себе всі аспекти шифрування та комунікації з API. Розробники взаємодіють із сервером через простий HTTPS-інтерфейс, який пропоную спрощену версію Telegram API.

TDLib(Telegram Database Lib) - це утиліта, яка дозволяє стороннім розробникам створити свій телеграм клієнт, не пишучи його з нуля. TDLib бере на себе більшість важких моментів розробки, такі як: мережева реалізація, шифрування, або локальне сховище даних. Тому розробники можуть приділити

більше часу дизайну, анімаціям тощо. Дана бібліотека є open-source, та вона сумісна майже з усіма мовами програмування.

Gateway API - інструмент, який надає можливість бізнесам, додаткам або веб-сайтам використовувати Telegram для надсилання кодів авторизації, а не звичайні SMS повідомлення. Це дозволяє знизити витрати, підвищити безпеку та швидкість доставки кодів. Через велику кількість користувачів, коди доставляється миттєво через спеціальний чат у самому месенджері.

2.7 PostgreSQL як сховище для даних

Сучасний світ IT характеризуються стрімким зростанням обсягів данних, які потребують ефективного зберігання, обробки та управління ними. Системи керування базами даних мають ключову роль у цих процесах, надаючи потрібні інструменти для цього. Серед багатьох СКБД виділяється PostgreSQL - потужна система з open-source code, що має в низці своїх особливостей: високу надійність, гнучкість та підтримку широкого спектру функціональних можливостей. В порівнянні з іншими open-source проектами, по типу Apache або MySQL, PostgreSQL не має контролю з боку якоїсь компанією, її розробка охоплює представників із різних компаній та навіть поодиноких розробників.

Сам сервер написаний на мові програмування C. Зазвичай він представлений у вигляді текстових файлів із початковим кодом. Для інсталяції потрібно відкомпілювати файли на своєму локальному комп'ютері і скопіювати в будь-який каталог.

Однією із основних можливостей в PostgreSQL є функції. Вони дозволяють виконувати потрібний розробнику код безпосередньо сервером бази даних. Мова за допомогою яких їх пишуть, в основному це - SQL. Для досягнення більш гнучкого створення функції можна використовувати й інші мови програмування, з якими PostgreSQL є сумісним, а саме:

- Вбудована мова - PL/pgSQL
- Мови розробки сценаріїв: PL/Perl, PL/Python, PL/Ptcl, PL/Ruby, PL/sh
- Незмінна класика: C, C++, Java

У PostgreSQL є кілька важливих властивостей, які роблять його більш кращою СКБД, ніж інші:

- Архітектура та масштабованість: PostgreSQL дотримується моделі клієнт-сервер, де сервер є окремим процесом і може обробляти кілька клієнтських підключень.
- Розгортання та конфігурація: так как PostgreSQL працює як окремий сервер, то потрібен окремий процес встановлення та налаштування. Сервер може бути розгорнутий на будь-якій платформі та користувач має можливість точно налаштовувати його параметри для кращої продуктивності
- Продуктивність та масштабованість: PostgreSQL може використовуватись для ефективної обробки великих БД і складних навантажень. Розробники розробили надійну оптимізацію, індексування, а також можливість планування запитів. Якщо належним чином конфігурувати БД, то вона буде мати можливість обробляти великі обсяги даних і підтримувати одночасний доступ кількох клієнтів

Прогрес розвитку PostgreSQL не стоїть на одному місці, він постійно розвивається і використовується в нових направленнях. Він активно адаптується до сучасної IT-індустрії, таких як:

- Хмарні технології: PostgreSQL досить часто використовується в хмарних середовищах, наприклад в AWS, Microsoft Azure. Це дозволяє легко масштабувати бази даних, забезпечити високу доступність та навіть автоматизувати резервне копіювання.
- Контейнеризація: PostgreSQL легко розгорнути в контейнеризованих середовищах, таких як Docker, що може забезпечити портативність та швидке розгортання

- Big Data: завдяки обширній кількості розширень для PostgreSQL часто використовується в Big Data. Прикладами можуть бути PostGIS, для роботи з геопросторовими даними, TimescaleDB для часових рядів, Citus для горизонтального масштабування

В сучасному світі безпека та конфіденційність є критично важливим компонентом у будь-якій сфері. Розробники PostgreSQL розробили низку функцій аби уникнути будь якого небажаного втручання. PostgreSQL підтримує шифрування на декількох рівнях, на рівні передачі даних та на рівні зберігання, що відповідає стандартам таких як, GDPR, HIPPA та PCI DSS. Також була розроблена система управління ролями(RBAC), яка дозволяє тонко налаштовувати права для окремих осіб, або цілих груп осіб. Не потрібно забувати і про аудит та логування, що дозволяють відстежувати всі запити користувачів та їх дії.

2.8 Система керування версіями Git

Git - система керування версіями, яка має можливості для спільного доступу до них. Проект був розроблений Лінусом Торвальдсом для управління розробки ОС Linux, але з недавнього часу його очолює Джуніо Сі. Git по всім міркам оцінювання вважається найкращим, надійним і потужним в порівнянні з іншими системами контролю версіями. Для гарантування історичної цінності файлів і захисту від несанкціонованих змін застосовуються криптографічні методи. Також цифрові підписи розробників можуть бути пов'язані з тегами та комітами.

Велика кількість проектів використовують для контролю версій саме Git, ось деякі з них: Linux, Android, Cairo, PostgreSQL, PHP, ELinks, LibreOffice, Perl, Eclipse, KDE, Video LAN тощо.

Ця програма розповсюджуються безкоштовно та випущена під ліцензією GNU GPL 2. Система GIT розроблена як набір програм, розроблених для

використання в сценаріях для полегшення створення систем контролю версій та інтерфейс користувача. Прикладом може бути Cogito, що є інтерфейсом сховищ Git, а також StGit, що використовуються для керування колекціями патчів.

Для забезпечення віддаленого доступу до сховища Git використовується Gid-daemon, SSH або HTTP-сервер. Git-daemon, яких входить до складу дистрибувати Git, разом із SSH є одним з найпоширенішим і надійним методів доступу. Хоча HTTP-доступ має певні обмеження, він широко застосовується в контрольованих мережах, оскільки дозволяє використовувати наявні налаштування мережевих фільтрів.

Принцип роботи Git:

Збереження файлів: на відміну від інших систем контролю версій, Git не зберігає дані як перелік змін у файлах. Натомість він створює набір зліпків (snapshots). Під час фіксації (коміту) поточної версії проєкту Git зберігає зліпок стану всіх файлів. Якщо файл не зазнав змін, Git посилається на раніше збережений файл. Git функціонує подібно до файлової системи з власними унікальними інструментами. Він зберігає інформацію про розмір файлу, час його створення та останньої модифікації. Усі ці дані записуються у файл index, розташований у папці “.git”. Файли Git мають три стани:

- Зафіксовано: якщо файл збережено в локальному сховищі
- Змінено: якщо файл був якость модифікований, але зміни при цьому не були зафіксовані
- Підготовлено: якщо файл було якось змінено та відмічено для фіксації

Локальні операції: Git надає можливість користуватися локальною файловою системою, навіть якщо відсутнє підключення до інтернету. Всі зміни зберігаються локально, на вашому пк, а при підключені до інтернету вивантажуються на віддалений репозиторій. Аналог Git, що містить назву Subversion, дозволяє лише редагувати файли, при цьому зберегти зміни на локальне сховище не дає змоги.

Цілісність даних: Git зберігає всі файли у вигляді хешів, для цього використовується функція хешування SHA1. Кожного разу, перед збереженням, вона обчислює хеш файлу, а отриманий хеш індексується в Git. За допомогою хешу Git може легко слідкувати за зміною у файлі.

Галуження: іншими словами це розмежування від основної гілки розробки проекту. Git дозволяє створювати декілька гілок для різних цілей та для того, аби дозволити декільком розробникам працювати над проектом, а також перемикатися між ними. Гілки в Git це своєрідні вказівники для фіксації. При кожній фіксації гілка в Git рухається автоматично, вона є простим файлом з 40-ка символьної сумою SHA1 фіксації.

Зливання та перебазування даних: в Git є два методи інтеграції змін між гілками:

- Merge: зливає дві гілки в одну
- Rebase: запам'ятовує фіксації у вигляді історії змін, перемотуючи саму гілку та застосовує зміни у вигляді файлу фіксації

2.9 Середовище розробки IntelliJ Idea

IntelliJ Idea - середовище розробки (IDE), для створення програм за допомогою мови програмування Java, розробником цього продукту є компанія JetBrains, яка базується у Чехії. IDE надає інструменти для професійної, а також аматорської розробки java-застосунків, веб-сайтів тощо. IntelliJ Idea має свої можливості, які і відрізняють її від інших IDE:

- Аналіз коду: IDE в автоматичному режимі робить аналіз вашого коду для знаходження помилок та підсвічує їх, зазвичай пропонуючи декілька варіантів їх вирішення
- **Інтеграції**: IDE підтримує різного роду інтеграції з третіми сервісами, такими як GitHub, Jira, YouTrack, Redmine

- **Шаблони**: Live Templates дозволяє заощадити час для написання шаблонного коду, такий як HTML документ тощо
- **Code debugger**: IDE має досить потужний та зручний інструмент для відлагодження коду, надаючи можливість створення точок призупинення, відстежування змін тощо
- **Рефакторинг**: IntelliJ IDEA пропонує розширені інструменти для рефакторингу коду, які дозволяють швидко та безпечно змінювати структуру коду, перейменовувати змінні, методи чи класи, а також оптимізувати кодову базу.
- **Підтримка різних мов і фреймворків**: Окрім Java, IDE підтримує інші мови програмування, такі як Kotlin, Scala, Groovy, а також популярні фреймворки, наприклад, Spring, Hibernate, Maven, Gradle, що робить її універсальним інструментом для розробки.
- **Інтелектуальна автодоповнення**: Функція автодоповнення коду (Code Completion) базується на контексті, аналізуючи код і пропонуючи релевантні підказки, що значно прискорює процес написання програм.
- **Налаштування інтерфейсу**: IntelliJ IDEA дозволяє гнучко налаштовувати інтерфейс користувача, включаючи теми оформлення, розташування панелей інструментів і комбінації клавіш, що забезпечує комфортну роботу для розробників із різними уподобаннями.
- **Плагіни**: IDE має широкий вибір плагінів, доступних через JetBrains Marketplace, що дозволяє розширити функціональність, додавши підтримку нових інструментів, мов чи специфічних технологій.

Ці можливості роблять IntelliJ IDEA одним із найпопулярніших середовищ розробки серед Java-розробників, які прагнуть ефективності, зручності та високої якості у створенні програмного забезпечення.

2.10 Висновки до розділу 2

Поєднання Telegram Bot API, Java, Python, PostgreSQL та Git, створення чат-боту з інтеграцією інтелектуальної моделі, дозволить вивести інтернет знайомства на новий рівень і забезпечить найбільш точне співпадіння між партнерами.

Telegram Bot API в поєднанні з Java дає нам змогу створити чат-бота та його основний функціонал, як створення профілю, пошук людей, зміна налаштувань профілю тощо. PostgreSQL виступить нам в ролі сховища для інформації про користувачів, які використовують нашого чат-бота. Git забезпечить ефективне відстежування та збереження нашого проекту у віддаленому репозиторії. IntelliJ Idea це наше середовище розробки, яке надає потрібні нам інструменти для комфортної розробки нашого проекту.

3 РЕАЛІЗАЦІЯ ЧАТ-БОТУ З ВИКОРИСТАННЯМ ІНТЕЛЕКТУАЛЬНОЇ МОДЕЛІ З МОЖЛИВІСТЮ ПІДБОРУ КОРИСТУВАЧІВ ЗА ЇХ ОПИСАМИ

3.1 Діаграма діяльності

Діаграми діяльності - інструменти для моделювання систем і процесів, які застосовуються для візуалізації алгоритмів та потоків даних. Вони базуються на графі діяльності, що відображає послідовність дій, які виконуються за певних умов і в заданому порядку. Виконання дій регулюється логічними умовами, що визначають їхню послідовність. Такі діаграми використовуються для моделювання різноманітних процесів, включно з бізнес-процесами та програмними алгоритмами, допомогаючи наочно представити складні взаємозв'язки між діями та умовами їх виконання. На практиці діаграми діяльності застосовуються для системного аналізу, проєктування, документації процесів, розробки програмного забезпечення тощо (рис. 3.1) .

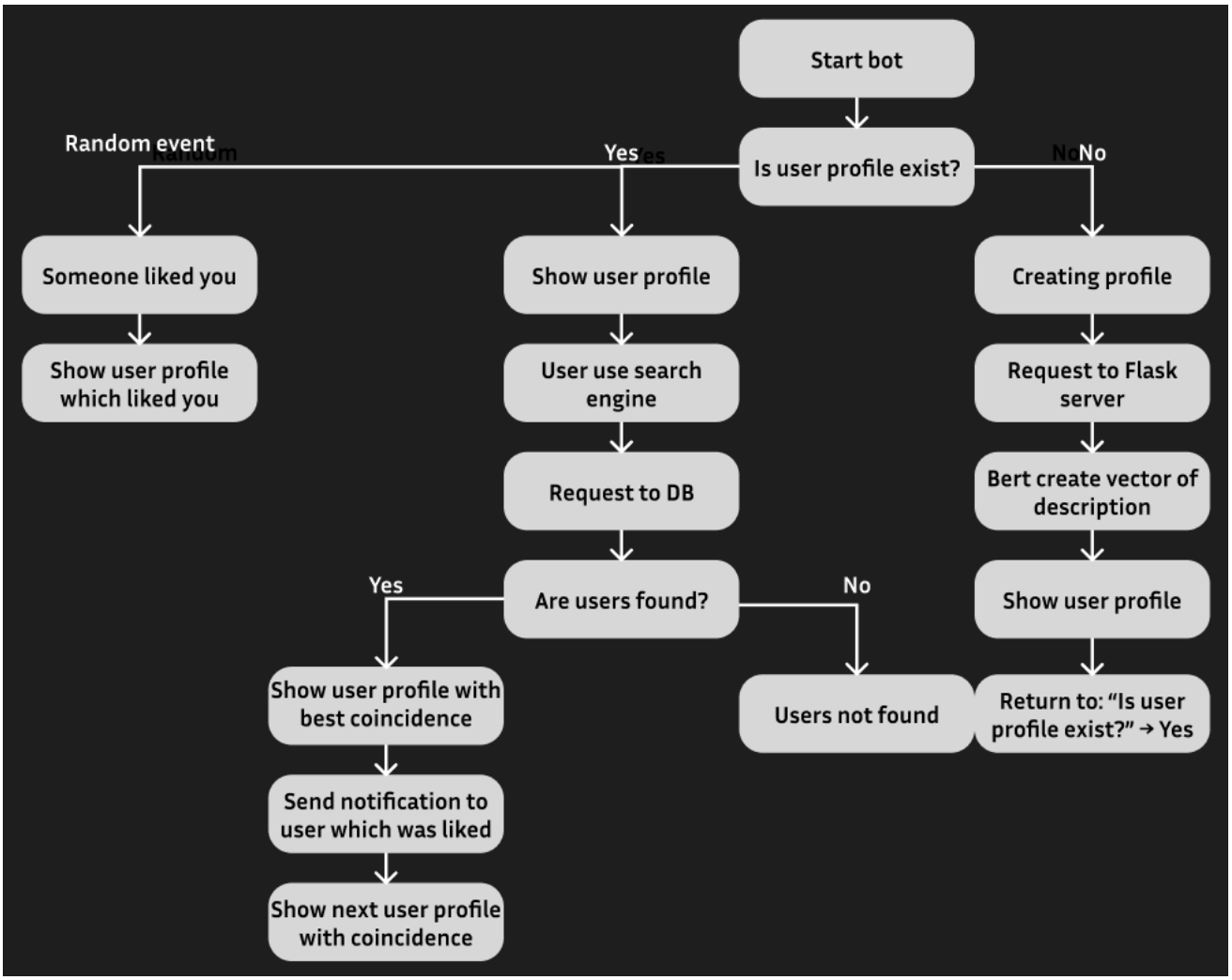


Рисунок 3.1 - Діаграма діяльності

Дана діаграма діяльності поверхнево відображає як працює чат-бот в комбінації з сервером та інтелектуальною моделлю.

3.2 Інтеграція інтелектуальної моделі

Проект, а якщо бути більш точним, то сам чат-бот та його основних функціонал були написані на мові програмування Java, а механізм передачі описів користувачів інтелектуальній моделі був написаний мовою програмування Python в колаборації із Flask фреймворком. Для інтеграції самої інтелектуальною моделі в python код потрібно встановити бібліотеку transformers, за допомогою команди “pip install transformers” в консолі, оскільки ми будемо використовувати модель від однієї з самих відомих компаній у світі - Google.

Бібліотека transformers, розроблена компанією Hugging Face, забезпечує зручний інтерфейс для роботи з передовими моделями обробки природної мови, включаючи моделі від Google, такі як BERT або T5. Вона підтримує Python версії 3.6 і новіші та містить інструменти для легкої інтеграції моделей у програми. Бібліотека дозволяє як завантажувати попередньо навчені моделі, так і налаштовувати їх для специфічних завдань, що робить її ідеальною для створення інтелектуальних систем, таких як чат-боти. Крім того, бібліотека transformers підтримує як синхронну, так і асинхронну обробку запитів, що забезпечує гнучкість при інтеграції в різні типи проектів.

Після встановлення бібліотеки transformers у Python-середовищі необхідно ініціалізувати її, підключити модель і токенізатор, а також налаштувати Flask для обробки запитів від Java-компонента чат-бота.

Лістинг 3.1 — Імпорт бібліотеки transformers, Flask та ініціалізація моделі й токенізатора.

```
from flask import Flask, request, jsonify
from transformers import BertTokenizer, BertModel
import torch
import psycopg2
app = Flask(__name__)
tokenizer =
BertTokenizer.from_pretrained('bert-base-multilingual-uncased'
)
model =
BertModel.from_pretrained('bert-base-multilingual-uncased')
```

Цей код забезпечує базову настройку для взаємодії з моделлю та підготовку до обробки текстових даних.

Наступним кроком є створення функції, яка оброблятиме описи користувачів, передаватиме їх моделі та повертатиме згенероване векторне

представлення опису. Ця функція інтегрується з Flask, щоб приймати запити від Java-сервера та вносити зміни в PostgreSQL сховище.

Лістинг 3.2 — Функція для обробки описів користувачів з використанням моделі та створення векторного представлення цих описів.

```
def get_embedding(text):
    if not text or text.isspace():
        return None
    inputs = tokenizer(text, return_tensors="pt", truncation=True, padding=True,
max_length=128)
    with torch.no_grad():
        outputs = model(**inputs)
    return outputs.last_hidden_state.mean(dim=1).squeeze().numpy()
@app.route('/update_embedding', methods=['POST'])
def update_embedding():
    data = request.json
    user_id = data.get('user_id')
    if not user_id:
        return jsonify({"status": "error", "message": "user_id is required"}), 400
    try:
        connection = psycopg2.connect(
            dbname="dbname",
            user="user",
            password="password",
            host="host",
            port=port
        )
        cur = connection.cursor()
        cur.execute("SELECT user_description FROM users WHERE user_id = %s AND
user_description IS NOT NULL", (user_id,))
        result = cur.fetchone()
        if result:
            user_description = result[0]
            embedding = get_embedding(user_description)
            if embedding is not None:
                cur.execute(
                    "UPDATE users SET embedding = %s WHERE user_id = %s",
                    (embedding.tolist(), user_id)
                )
                connection.commit()
                return jsonify({"status": "success", "user_id": user_id})
            else:
                return jsonify({"status": "error", "message": f"Empty description for
user_id={user_id}"}), 400
        else:
            return jsonify({"status": "error", "message": f"No description found for
user_id={user_id}"}), 404
    except Exception as e:
```

```
        return jsonify({"status": "error", "message": str(e)}), 500
    finally:
        if 'cur' in locals():
            cur.close()
        if 'connection' in locals():
            connection.close()
```

Такий підхід дозволяє ефективно обробляти описи користувачів, використовуючи можливості інтелектуальної моделі, і передавати відповіді назад до Java-компонента.

Після реалізації цих кроків можна переходити до розробки серверної частини, яка буде написана за допомогою бібліотеки Flask, для забезпечення ефективного зв'язку між функціоналом чат-бота, а також нашої інтелектуальною моделлю. Сервер буде приймати запити з клієнтської частини для нашої моделі, про створення векторного представлення для конкретного користувача, а потім буде передавати цей вектор у нашу базу даних для подальших дій з ним.

3.3 Основний механізм роботи

Весь проект написано одразу на двох мовах програмування. Java для реалізації функціоналу чат-бота, наприклад:

- Підключення Telegram Bot APi
- Створення, видалення, зміна всіх компонентів профіля користувача
- Механізм пошуку користувачів
- Збереження, оновлення даних в базі даних

А також Python, з бібліотекої Flask для взаємодії між чат-ботом та нашої моделлю:

- Відправки запитів про створення векторного представлення опису користувача

- Відправки запитів про зміну векторного представлення опису користувача, через його зміну

Коли користувач вперше потрапляє на чат-бота та натискає на “Старт”, то він повинен створити свій профіль для користування чат-ботом. Ввести ім’я, вік, країну, місто, опис, а також зображення профілю.

Аби користувач мав змогу переглядати інших користувачів, йому потрібно написати в чат-боті команду “/search”, яка почне видавати йому покроково користувачів, з якими він найбільш схожий по опису. Відповідний код для реалізації наведено у лістингу 3.3.

Лістинг 3.3 Частина коду, яка відповідає за показ інших користувачів у пошуковому механізмі

```
public static void showFindingProfile(long user_id, long chat_id) throws SQLException {
        User user = SQLQuery.selectProfileByEmbedding(user_id,
SQLConnection.connect());
        LocalizationService.loadLanguage(SQLQuery.selectLanguageByUserId(user_id,
SQLConnection.connect()));
        if (user != null) {
            String profileInfo = Miscellaneous.loadUserProfile(user_id, user);
            InlineKeyboardMarkup keyboard = InlineKeyboardMarkup.builder()
                    .keyboardRow(new InlineKeyboardRow(
                                    InlineKeyboardButton
                                            .builder()

.text(LocalizationService.getMessage("like.user"))
                                            .callbackData("like_" + user.getUsername())
                                            .build(),
                                    InlineKeyboardButton
                                            .builder()

.text(LocalizationService.getMessage("dislike.user"))
                                            .callbackData("dislike_" +
user.getUsername())
                                            .build()
                            )
                    )
                    .build();
            try {
                SendPhoto photo = new SendPhoto(String.valueOf(chat_id),

SQLQuery.imageFromDatabase("C:\\Users\\pyotc\\IdeaProjects\\FindYourLove\\images\\user_
" + user.getUserId() + "_profile.jpg", SQLConnection.connect()));
```

```
                photo.setCaption(profileInfo);
                photo.setReplyMarkup(keyboard);
                client.execute(photo);
                SQLQuery.markProfileAsViewed(user_id, user.getUserId(),
SQLConnection.connect());
            } catch (IOException | TelegramApiException e) {
                throw new RuntimeException(e);
            }
        } else {
            Miscellaneous.sendCustomMessage(chat_id,
LocalizationService.getMessage("search.empty"));
        }
    }
```

3.4 Тестування веб-застосунку

Проведення тестування це незамінна частина розробки будь-якого продукту, чи то кросівки, чи то професійне програмне забезпечення.

Якщо брати за приклад перший вхід користувача в чат-бот, то він має створити свій профіль аби в подальшому користуватись всім наявним функціоналом.

При першому візиті в чат-бот та при натисканні кнопки /start, ми маємо обрати одну із трьох мов на вибір(Англійська, Українська та Російська), в подальшому весь інтерфейс чат-бота, а саме його повідомлення будуть відображатись на вибраній нами мові(за потреби її можливо змінити в налаштуваннях). Покрокова реєстрація нового користувача:

1) Ім'я: імя, яке буде в подальшому використовуватися для вашої анкети
2) Стать: повідомлення з вибором статі у вигляді емоджі
3) Вік: потрібно написати свій вік, наприклад 22
4) Країна: країна, в якій ви проживаєте
5) Місто: місто, в якому ви проживаєте
6) Опис: опис вашого профілю, на якому ґрунтується підбір користувачів
7) Зображення: зображення, яка буде використовуватись для відображення в профілі

Отже, так як профіль було успішно створено, то давайте переглянемо як він буде відображатися для інших користувачів. Аби переглянути свій профіль потрібно написати повідомлення з командою /profile або обрати з меню команд (рис. 3.1).

Рисунок 3.1 - Вигляд персонального профілю по команді /profile

Оскільки я тільки-но створив профіль, то наша модель наново сгенерувала векторне представлення опису мого профілю та змінила його в нашому PostgreSQL сховищі (рис. 3.2).

Рисунок 3.2 - Вигляд векторного представлення нашого опису

Як бачимо, механізм створення векторного представлення опису нашої моделлю успішно працює. Спробуємо створити ще один профіль зі схожим смисловим описом і побачимо яким буде результат пошуку в чат-боті. Аби перейти в режим пошуку, потрібно написати /search або обрати в меню команд (рис. 3.3).

Рисунок 3.3 - Ідеальне співпадіння користувачів

Ми отримали той результат, який і очікувало. Для більш ретельної перевірки спробуємо змінити наш опис. Це можна зробити через налаштування командою /settings або обрати її в меню команд (рис. 3.4).

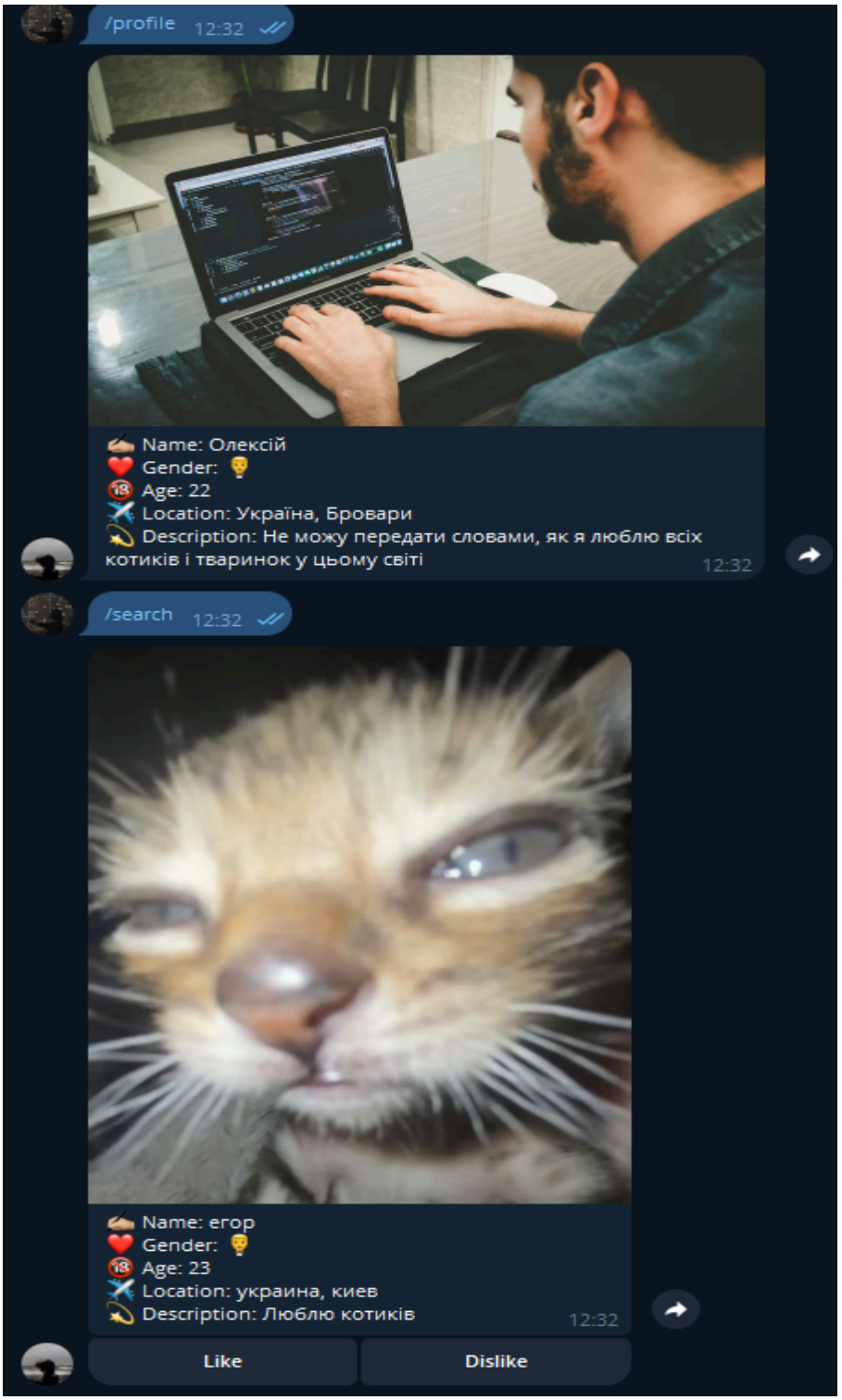


Рисунок 3.4 - Підбір інших користувачів зі зміненим нашим описом

Як видно із прикладу, зі зміною вашого опису, будуть і змінюватись люди в пошуку, від найбільш схожих до вас людей, то найбільш віддалених. Зі збільшенням кількості користувачів, будуть змінюватись і результати. Також і не забуваємо перевірити як працює механізм симпатії, та відсіювання користувачів.

При переведі чат-бота в режим пошуку, командою /search, ми маємо змогу відправити звістку користувачу, який зараз у нас в чат-боті, або ж ми можемо його просто пропустити. Коли ми відправляємо симпатію користувачу, то його надходить повідомлення про це (рис. 3.5), що він комусь сподобався, він його може переглянути, а в подальшому і відповісти взаємністю, або ж просто пропустити, якщо йому в цей момент не цікаво шукати нових знайомств (рис. 3.6)

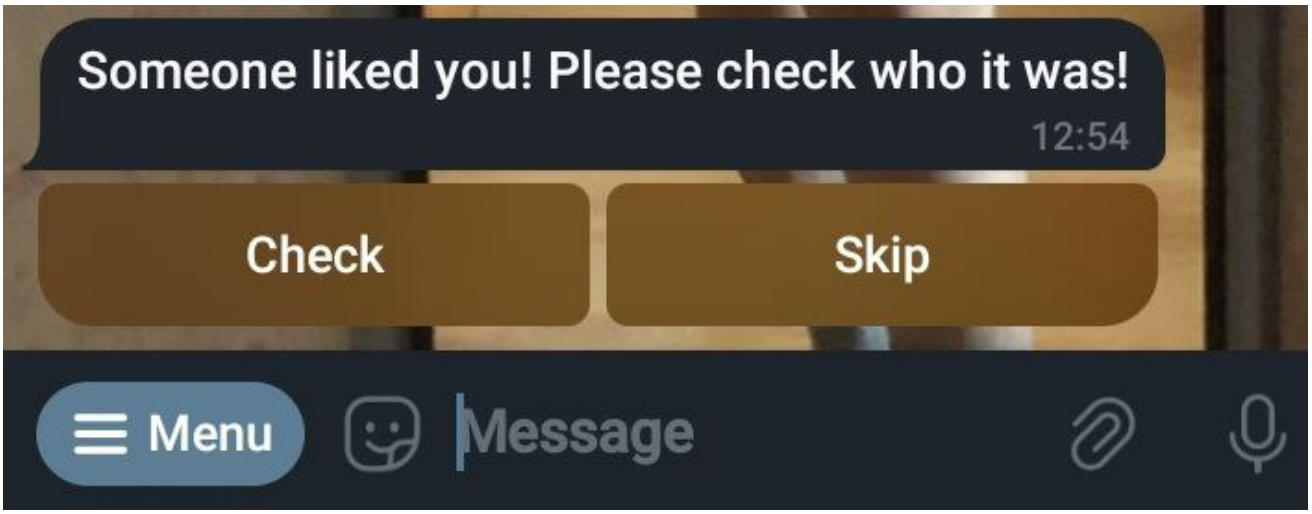


Рисунок 3.5 - Повідомлення про симпатію для користувача, який нам сподобався

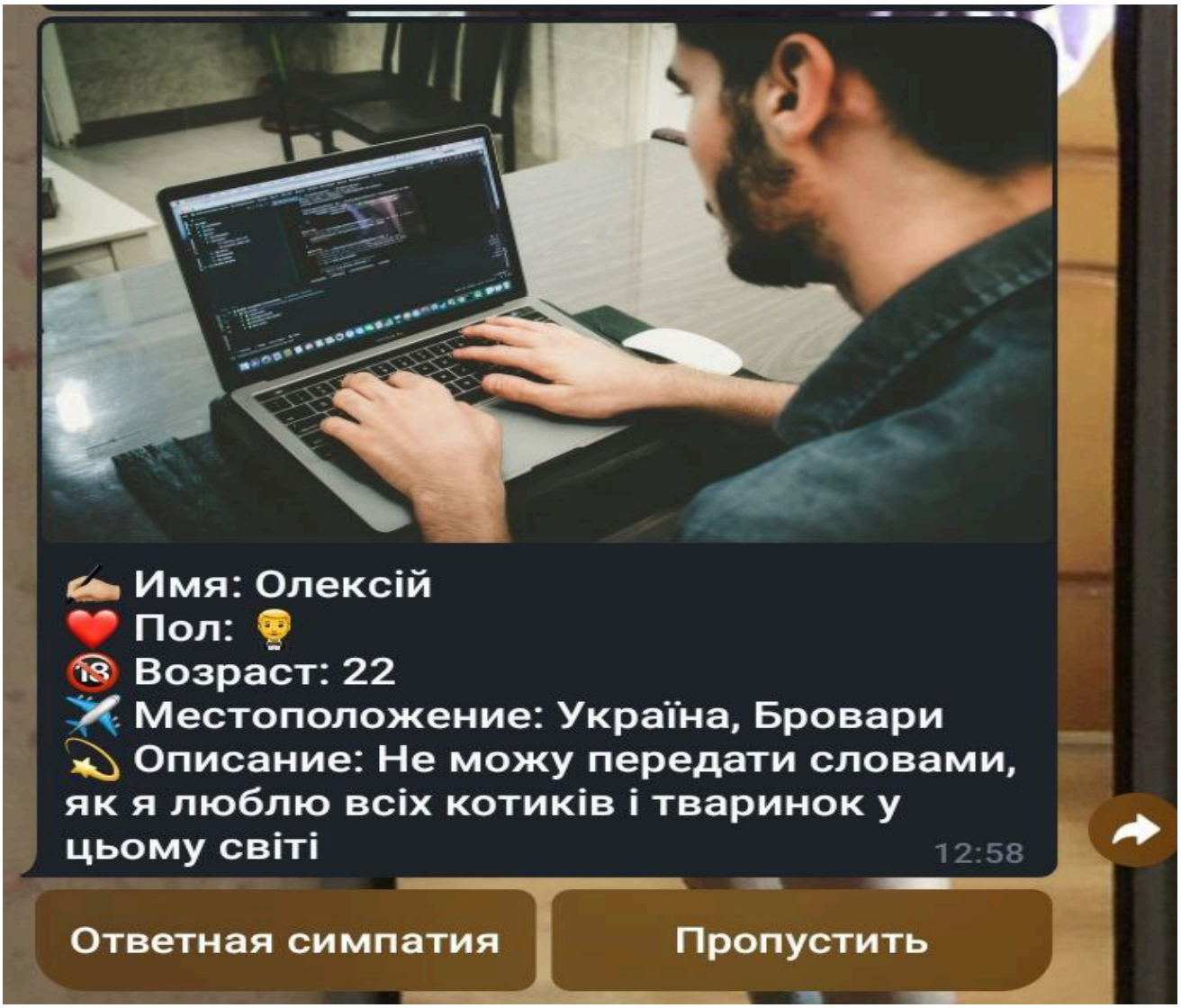


Рисунок 3.6 - Перегляд профілю користувача, якому ми сподобались

На той випадок, якщо користувач хочет змінити деякі аспекти свого профілю, то було розроблено меню налаштувань. Щоб перейти в налаштування можна написати команду /settings або обрати її зі списку команд (рис. 3.7).

Рисунок 3.7 - Перегляд налаштувань для профілю користувача

3.4 Висновки до розділу 3

Отже, як основний інструмент для моделювання дій ми використали діаграму діяльності, яка дозволяє нам візуалізувати основні процеси. Інтелектуальну мовну модель ми використали від компанії Google - Bert, а саме - bert-base-multilingual-uncase, яку інтегрували за допомогою бібліотеки transformers та яка була нашим основним засобом для генерації векторного представлення описів користувачів.

Користувач, потрапивши на нашого чат-бота вперше повинен створити свій профіль для подальшого користування всім його функціоналом, таким як пошук інших користувачів, зміна мови, зміна опису, зміна зображення для профілю, видалення або перестворення профілю. Створивши профіль, опис передається на сервер Flask, а в подальшому нашій моделі, яка створює векторне представлення опису та вносить зміни в наше PostgreSQL сховище. В подальшому користувач має можливість перегляду інших користувачів в порядку подібності їх описів, як це було показано раніше. Допустимо той момент, що нашому користувачу хтось сподобався, він натискає кнопку “Подобається”, користувачу, якому він відправив це - надсилається сповіщення про те, що він кому сподобався. Цей користувач має можливість перевірити кому він сподобався та відповісти взаємністю, або просто пропустити і переглядати користувачів далі.

Наразі, із-за невеликої кількості користувачів, список переглянутий користувачів та користувачів, яким була відправлена симпатія, оновлюється кожного разу, як користувач переглядає свій профіль командою /profile.

ВИСНОВКИ

У рамках цього проекту був розроблений та впроваджений чат-бот з використанням інтелектуальної моделі з можливістю пошуку партнера або просто спілкування. Ідея проекту була у створенні зручного та точного механізму підбору співрозмовника за допомогою методів машинного навчання, щоб точно підібрати собі партнера за схожими інтересами.

На початковому етапу було проаналізовано предетку область, а якщо бути точнішим саме: поняття інтелектуальних мовних моделей та було проведено огляд популярних нині чат-ботів для знайомств. Наступним кроком стало обрання засобів для реалізації проекту: аналіз існуючих засобів розпізнавання природної мови, способу передачі описів користувачів нашій моделі через сервер, технології для збереження даних користувачів тощо.

Під час проектування було розроблено діаграму діяльності, що демонструє основний механізм роботи чат-боту, а також було оглянуто методи інтеграції інтелектуальної моделі та її навчання. На етапі реалізації було створено повноцінний чат-бот із системою створення анкет та їх зміна, пошук співрозмовника та спосіб звязку з ним, зміна мови інтерфейсу чат-бота тощо.

Чат-бот, який було розроблено пройшов усі тестування і повністю задовольняє вимоги, які було поставлено раніше. Чат-бот готовий до запуску та використання в реальних умовах.

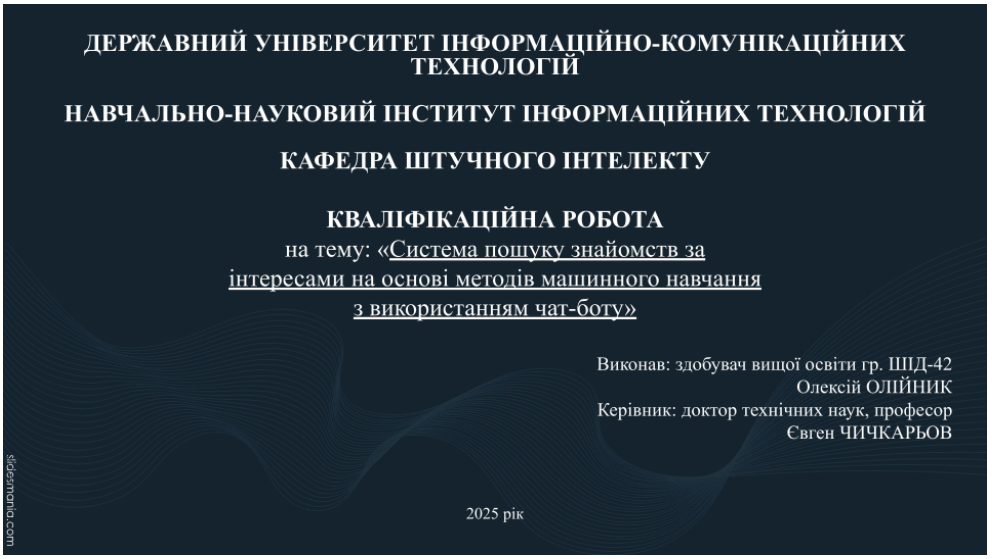
Розроблявши даний проект, було отримано не аби який досвід та цінні знання в розробці чат-ботів, інтеграції в них інтелектуальних моделей та способів взаємодії між клієнтом та сервером.

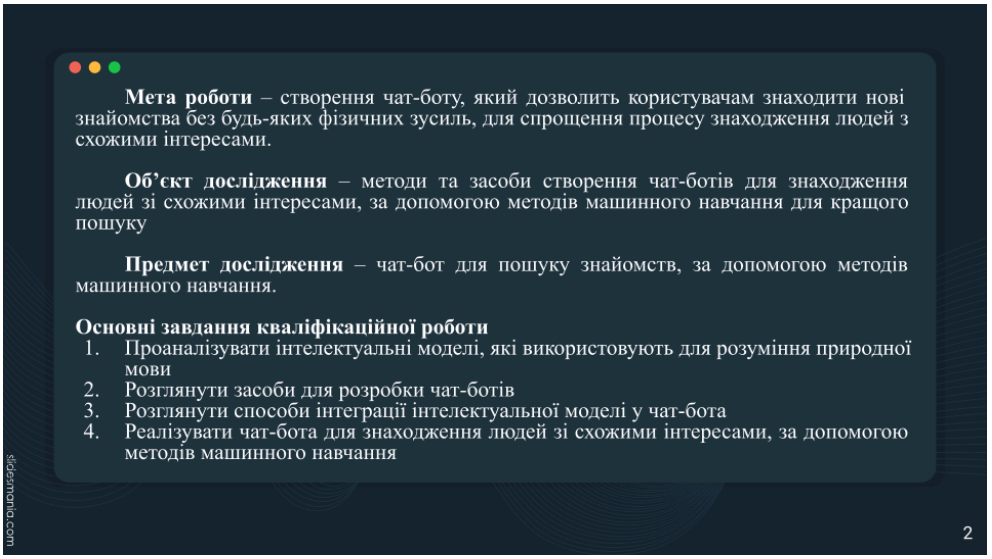
ПЕРЕЛІК ПОСИЛАНЬ

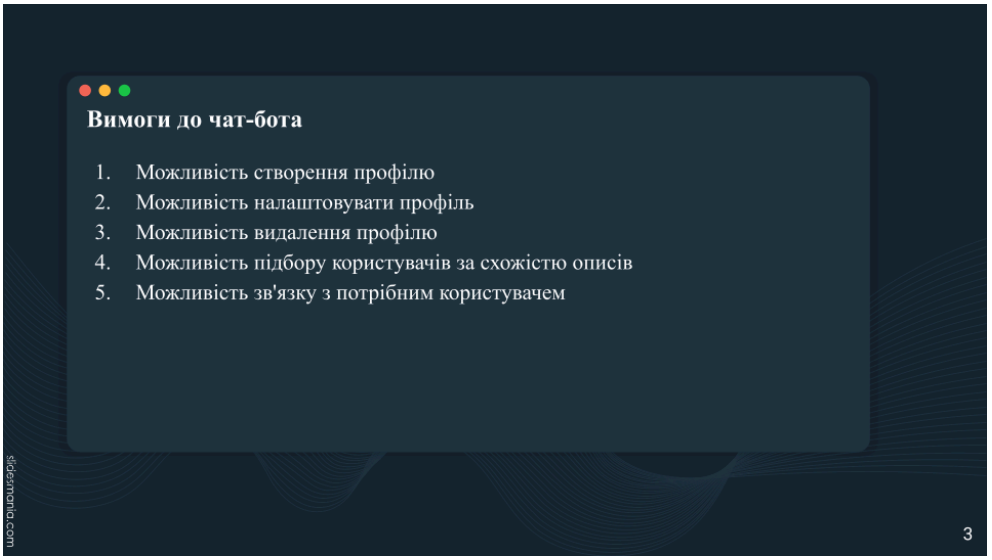
1. Russell, S., Norvig, P. Artificial Intelligence: A Modern Approach. — 4th ed. — Upper Saddle River: Prentice Hall, 2020. — 1152 с.
2. Goodfellow, I., Bengio, Y., Courville, A. Deep Learning. — Cambridge: MIT Press, 2016. — 800 с.
3. Vaswani, A., Shazeer, N., Parmar, N., et al. Attention is All You Need // Advances in Neural Information Processing Systems. — 2017. — С. 5998–6008.
4. Goldberg, Y. Neural Network Methods for Natural Language Processing. — San Rafael: Morgan & Claypool Publishers, 2017. — 309 с.
5. Jurafsky, D., Martin, J. H. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. — 3rd ed. — Upper Saddle River: Prentice Hall, 2020. — 1032 с.
6. Turing, A. M. Computing Machinery and Intelligence // Mind: A Quarterly Review of Psychology and Philosophy. — 1950. — Vol. 59, No. 236. — С. 433–460.
7. Devlin, J., Chang, M.-W., Lee, K., Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding // Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). — 2019. — С. 4171–4186.
8. Brownlee, J. Deep Learning for Natural Language Processing: Develop Deep Learning Models for Natural Language in Python. — Machine Learning Mastery, 2017. — 294 с.
9. Liu, Y., Ott, M., Goyal, N., et al. RoBERTa: A Robustly Optimized BERT Pretraining Approach // arXiv preprint arXiv:1907.11692. — 2019.
10. Liang, P., Klein, D. Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit. — Sebastopol: O’Reilly Media, 2009. — 504 с.

11. Bloch, J. Effective Java. — 3rd ed. — Boston: Addison-Wesley, 2018. — 416 с.
12. Van Rossum, G., Drake, F. L. Python 3 Reference Manual. — Scotts Valley: CreateSpace, 2009. — 242 с.
13. Telegram Bot API Documentation [Електронний ресурс]. — 2024. — Режим доступу: https://core.telegram.org/bots/api.
14. Chacon, S., Straub, B. Pro Git. — 2nd ed. — New York: Apress, 2014. — 456 с.
15. Gamma, E., Helm, R., Johnson, R., Vlissides, J. Design Patterns: Elements of Reusable Object-Oriented Software. — Boston: Addison-Wesley, 1994. — 395 с.
16. Manning, C. D., Raghavan, P., Schütze, H. Introduction to Information Retrieval. — Cambridge: Cambridge University Press, 2008. — 506 с.
17. Hugging Face. Transformers Library Documentation [Електронний ресурс]. — 2024. — Режим доступу: https://huggingface.co/docs/transformers/index.

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ







Діаграма діяльності

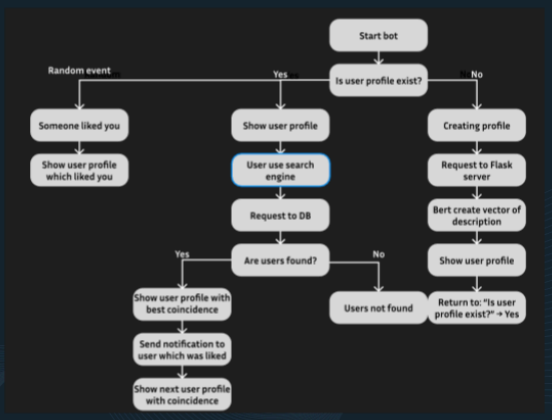


Рис 1. Діаграма діяльності

Діаграма класів

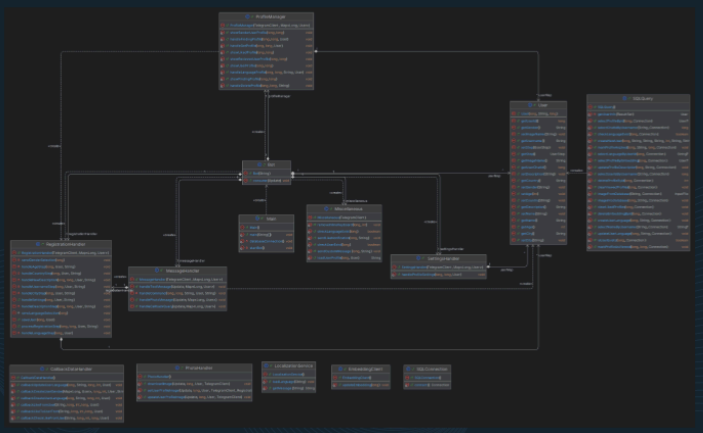


Рис 2. Діаграма діяльності

Інтелектуальна модель для розуміння природної мови

Bert - це інтелектуальна модель, яка навчається розуміти текст у речені, аналізуючи їх зліва направо та навпаки.
Модель постачається у двох попередньо натренерованих моделях:

- Base: нейромережна архітектура з 12 шарами, 768 прихованими, 12 головами та 110 мільйонами параметрів
- Large: нейромережна архітектура з 24 шарами, 1024 прихованими, 16 головами, а також 340 мільйонів параметрів

За потреби її можна донавчити для конкретних задач

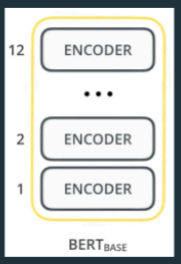


Рис. 3 Base модель Bert

Чому саме Bert?

Дану інтелектуальну модель було обрано через низку особливостей:

- Двонаправлене розуміння тексту, тобто модель проходу по тексту справа наліво, та навпаки, для кращого результату
- Висока точність у задачах обробки природної мови, що для даного проекта є важливим фактором
- Відкритий програмний код, що дозволяє переписати модель під себе
- Широке застосування в NLP, наприклад в Google для аналізу запитів користувачів

Засоби для реалізації чат-бота

Для реалізації чат-бота було використано наступні технології:

- Java: структура чат-бота, взаємодія між користувачами, профілі, налаштування тощо
- Python: Flask сервер, інтеграція інтелектуальної моделі, взаємодія моделі з описами користувачів
- PostgreSQL: сховище для всіх даних, таких як користувацькі дані(chat id, username, дані профілів, векторні представлення описів)

Візуальний вигляд бази даних

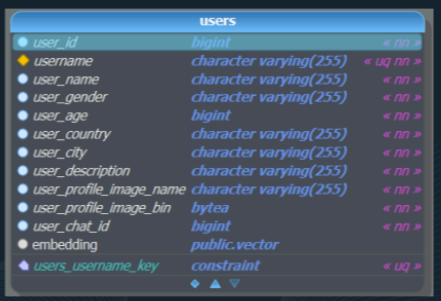


Рис. 4 Візуальний вигляд сховища PostgreSQL

Візуальна частина чат-боту

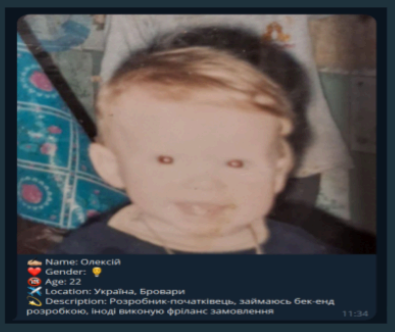


Рис. 5 - Візуальний вигляд профілю користувача

Перегляд профілю

Аби переглянути свій профіль спочатку потрібно пройти процедуру його створення, а потом переглянути за допомогою команди /profile

Зміна профілю

Аби змінити деякі налаштування профілю можна перейти в меню налаштувань командою /settings

Візуальна частина чат-боту

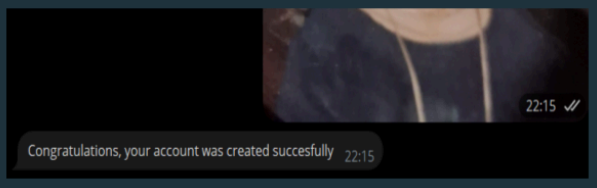


Рис. 6 Успішне створення профілю

```
Embedding was updated for  885842490
127.0.0.1 - - [20/May/2025 22:15:13] "POST /update_embedding HTTP/1.1" 200 -
```

Рис. 7 Оновлення векторного представлення опису нашою моделлю

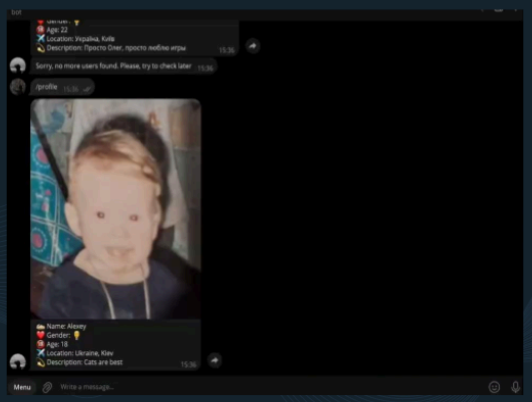
Як відбувається створення ембедінгу?

З Java коду відбувається запит на Flask сервер, який передає дані потрібного користувача зі сховища, а наша модель створює для нього ембедінг і вносить його в сховище

Що таке ембедінг?

Ембедінг - векторне представление тексту

Демонстраційні матеріали роботи чат-бота



Інтеграція інтелектуальної моделі

```python
from transformers import BertTokenizer, BertModel

tokenizer = BertTokenizer.from_pretrained('bert-base-multilingual-uncased')
model = BertModel.from_pretrained('bert-base-multilingual-uncased')
```

Рис. 8 Код для інтеграції інтелектуальною моделі Bert з бібліотеки transformers

Чому саме Python?

Більшість інтелектуальний моделей легко інтегруються через Python код, на відміну від Java

Чому pretrainded модель?

Такий тип моделей попередньо навчений, тому їх легко довчити для потрібних задач

Донавчання інтелектуальної моделі

```
rg/pyotcho/findyourlove/scripts/train.py
Epoch 1/3: 100%|                                                                | 625/625 [26:42<00:00,  2.56s/it]
INFO:__main__:Epoch 1, Average Loss: 0.0001069987773895263 6
Epoch 2/3: 100%|                                                                | 625/625 [27:40<00:00,  2.66s/it]
INFO:__main__:Epoch 2, Average Loss: 0.0
Epoch 3/3: 100%|                                                                | 625/625 [27:11<00:00,  2.61s/it]
INFO:__main__:Epoch 3, Average Loss: 0.0
INFO:__main__:Model saved to ./finetuned_bert
INFO:__main__:Embeddings updated in the database
PS C:\Users\pyotc\IdeaProjects\FindYourLove>
```

Рис. 9 Процес донавчання інтелектуальної моделі для завдань підбору користувачів

Висновки

1. У кваліфікаційній роботі було проаналізовано інтелектуальні моделі для розуміння природної мови
2. Було проаналізовано механізм створення векторних представлень для текстів (ембедінг)
3. Оглянуто засоби реалізації чат-бота
4. Було інтегровано інтелектуальну модель
5. Було розроблено чат-бот для пошуку знайомств з інтеграцією інтелектуальної моделі

Дякую за увагу!