МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ПРИРОДОКОРИСТУВАННЯ
ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ
ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

# КВАЛІФІЦІЙНА РОБОТА

першого (бакалаврського) рівня вищої освіти

**на тему:**

# РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ УПРАВЛІННЯ ПЕРСОНАЛОМ ФЕРМЕРСЬКОГО ГОСПОДАРСТВА

Виконав: студент групи ІТ-22сп
спеціальності 126 «Інформаційні
системи та технологій»

Мащенко Олег Олегович
(прізвище та ініціали)

Керівник: Желєзняк А.М.
(прізвище та ініціали)

**ДУБЛЯНИ 2023**

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ПРИРОДОКОРИСТУВАННЯ
ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Рівень вищої освіти: перший (бакалаврський)
Спеціальність 126 «Інформаційні системи та технології»

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____
(підпис)
*д.т.н., професор, Тригуба А. М.*
( вч. звання, прізвище, ініціали)
“____” _____202__ року

**З А В Д А Н Н Я**
**НА КВАЛІФІКАЦІЙНУ РОБОТУ**

Мащенко Олега Олеговича
(прізвище, ім’я, побатькові)

1. Тема роботи *«Розробка інформаційної системи управління персоналом фермерського господарства»*
керівник роботи *к. е н., доцент., Желєзняк А.М.*
( наук.ступінь, вч. звання, прізвище, ініціали)

затверджені наказом Львівського НУП *№ 453 / к - с від 30.12.2022 р*
2. Строк подання студентом роботи 09.06.2023 р.
3. Вихідні дані: *характеристика закладу фермерського господарства; вихідні дані та вимоги до роботи інформаційної системи управління персоналом, опис бібліотек мов програмування, програмна конфігурація інформаційної системи управління персоналом; науково-технічна і довідкова література.*

4. Зміст кваліфікаційної роботи (перелік питань, які потрібно розробити)
*Вступ*
*1. Аналіз предметної області*
*2. Проектування чат-боту інформаційної системи управління персоналом*
*3. Програмна реалізація проекту*
*4. Охорона праці та безпека в надзвичайних ситуаціях*
*Висновки*
*Бібліографічний список*
5. Перелік графічного матеріалу
*Графічний матеріал подається у вигляді презентації*

6. Консультанти розділів

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | | Відмітка про виконання |
|---|---|---|---|---|
| | | завдання видав | завдання прийняв | |
| *1, 2, 3* | | | | |
| *4* | | | | |

7. Дата видачі завдання *30.12.2022 р.*

КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів дипломної роботи | Строк виконання етапів роботи | Відмітка про виконання |
|---|---|---|---|
| 1 | *Отримання завдання. Вивчення рекомендованої літератури по темі роботи. Написання аналітичного огляду предметної області.* | *30.12.2022 – 03.02.2023* | |
| 2 | *Проектування та опис технічного завдання, функціональних вимог та технічної сторони реалізації проекту (написання проектної частини).* | *06.02.2023 – 03.03.2023* | |
| 3 | *Програмна реалізація проекту (вибір мови програмування, розробка сайту, тестування його роботи, розрахунок ефективності проекту)* | *06.03.2023 – 14.04.2023* | |
| 4 | *Розгляд питань з охорони праці та безпеки у надзвичайних ситуаціях* | *17.04.2023 – 05.05.2023* | |
| 5 | *Завершення оформлення основної частини, написання висновків та підготовка презентаційного матеріалу* | *08.05.2023 – 19.05.2023* | |
| 6 | *Завершення роботи в цілому. Підготовка до захисту кваліфікаційної роботи* | *22.05.2023 – 09.06.2023* | |

Студент _____ *Мащенко О.О.*
( підпис ) (прізвище та ініціали)

Керівник роботи _____ *Желєзняк А.М.*
( підпис ) (прізвище та ініціали)

УДК 331.108.2

Кваліфікаційна робота: 50 сторінок текстової частини, 4 таблиці, 18 рисунків, 22 джерел літератури, 3 додатки.

«*Розробка інформаційної системи управління персоналом фермерського господарства*». Мащенко Олег Олегович. – Кваліфікаційна робота. Кафедра інформаційних технологій. Дубляни, Львівський національний університет природокористування, 2023 р.

Подано господарсько–виробничу характеристику управління персоналом фермерського господарства, розглянуто ідею створення чат-бота системи управління персоналом. Проаналізовано предметну область, існуючі аналоги та визначено функціональні вимоги до програмного забезпечення. Здійснено проектування бази данних у відповідності до існуючих методик.

Запропоновано використання платформи Телеграм для практичної реалізації складової проекту інформаційної системи управління персоналом (модуля «Віртуальний асистент»).

Здійснено аналіз травматичних ситуацій при виконанні різних робіт у сфері використання комп’ютерної техніки, викладено питання охорони праці.

КЛЮЧОВІ СЛОВА: інформаційна система, управління персоналом, віртуальний асистент, чат-бот, фермерське господарство

# ЗМІСТ

# ВСТУП

Метою цієї дипломної роботи є проектування та розробка прототипу чат-бота для інформаційної системи підтримки та управління персоналом в фермерському господарстві на основі нейромережової мовної моделі генерації текстів OpenAI ChatGPT.

Тема цієї дипломної роботи є актуальною для різних галузей економіки, у т.ч. і сільського господарства. Аграрна сфера сьогодні зазнає значних трансформацій, а технологічний прогрес відіграє вирішальну роль у підвищенні продуктивності, зниженні витрат та оптимізації операцій з управління персоналом, планування та контролю за виконаною роботою. Ефективне управління персоналом має важливе значення для безперебійного функціонування фермерського господарства, забезпечуючи ефективний розподіл завдань, безперебійну комунікацію та оптимізацію ресурсів. Однак традиційні методи управління персоналом часто стикаються з такими проблемами, як непорозуміння, прогалини в інформації та трудомісткі адміністративні завдання.

У цьому контексті інтеграція технології чат-ботів пропонує багатообіцяючі рішення. Чат-боти можуть надавати допомогу в режимі реального часу, автоматизувати рутинні завдання та сприяти ефективній комунікації між персоналом ферми. Вони можуть слугувати віртуальними помічниками, надаючи інформацію, відповідаючи на запитання та скеровуючи працівників у їхній повсякденній діяльності. Впровадивши прототип чат-бота в інформаційну систему, фермерські господарства можуть вдосконалити процеси управління персоналом, підвищити операційну ефективність і досягти кращих результатів.

В ході виконання кваліфікаційної роботи були поставлені наступні завдання:

1. Вивчення теоретичних та методологічних аспектів управління персоналом в сільському господарстві та їх вдосконалення з використанням інформаційних технологій.
2. Огляд та аналіз існуючих аналогів інформаційних систем та чат-ботів для підтримки персоналу в сільському господарстві.
3. Розробка функціональних вимог до прототипу чат-бота, зокрема здатності розпізнавати користувачів, зберігати повідомлення та надавати відповіді на запитання.
4. Вибір платформи для розробки чат-бота з урахуванням зручності використання та наявності готової інфраструктури, наприклад, платформа Telegram.
5. Вибір мови програмування та розробка та налаштування чат-бота для платформи Telegram з використанням відповідних бібліотек і фреймворків.
6. Проведення тестування та експлуатації чат-бота для перевірки його функціональності та забезпечення коректної роботи.

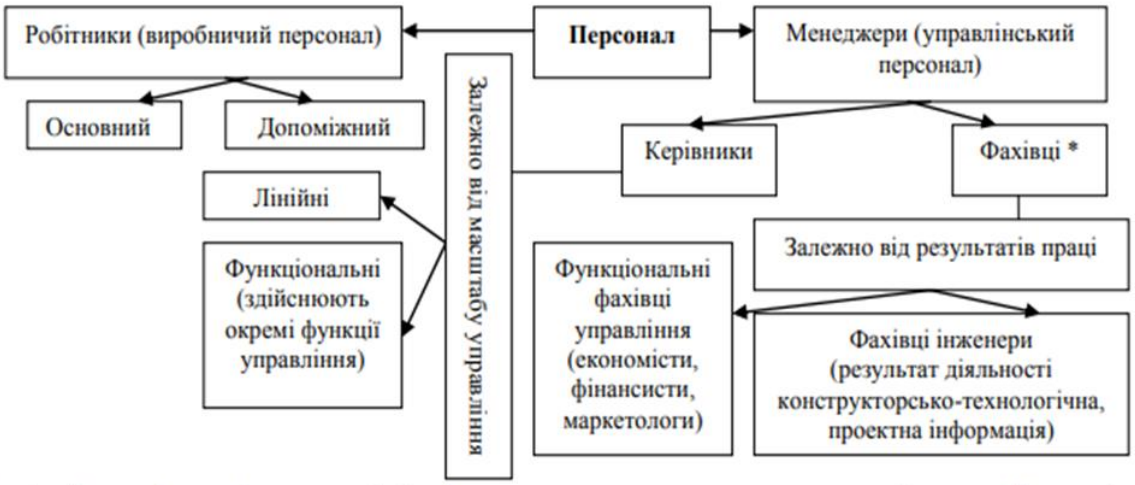
Загалом дипломна робота складається з чотирьох розділів. Вона складається з чотирьох розділів. У першому розділі аналізується предметна область, досліджуються теоретичні основи, описуються специфічні потреби управління персоналом фермерських господарств та розглядаються існуючі аналоги. Другий розділ охоплює етап проектування, включаючи функціональні вимоги, вибір платформи, вибір мови програмування та вибір бази даних. Третій розділ детально описує програмну реалізацію проекту чат-бота, обговорюючи розробку Telegram-бота, тестування та експлуатацію. Нарешті, четвертий розділ присвячений охороні праці та безпеці в надзвичайних ситуаціях, містить аналіз потенційних небезпек і пропонує організаційні та технічні рекомендації. Ці розділи забезпечують комплексну основу для розробки прототипу чат-бота.

# РОЗДІЛ 1
# АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

## 1.1 Теоретичні основи автоматизації управління персоналом

Ефективна організація роботи персоналу має важливе значення для оптимізації продуктивності сільського господарства та забезпечення безперебійного функціонування фермерських господарств. У цьому розділі розглядаються теоретичні основи, які лежать в основі управління персоналом фермерського господарства, забезпечуючи комплексне розуміння принципів і концепцій, які керують цією сферою.

Однією з ключових теоретичних засад в управлінні персоналом фермерських господарств є планування робочої сили. Планування робочої сили передбачає оцінку поточних і майбутніх потреб фермерського господарства в робочій силі та розробку стратегій для ефективного задоволення цих потреб. Цей процес включає аналіз попиту на різні навички та ролі, враховуючи такі фактори, як цикли вирощування сільськогосподарських культур, сезонність та ринкові коливання. Узгоджуючи планування робочої сили з цілями та завданнями фермерського господарства, фермери можуть забезпечити потрібну кількість працівників з необхідними навичками в потрібний час.




* Відмінність між керівниками і фахівцями – це юридичне право прийняття рішень і наявність у підпорядкуванні інших працівників.

Рисунок 1.1 – Стандартна структура персоналу підприємства

Розробка робочих місць - ще один важливий аспект управління персоналом на фермі. Він включає в себе структурування та організацію робочих завдань і обов'язків для максимізації ефективності та продуктивності. При розробці робочих місць враховуються такі фактори, як спеціалізація завдань, розподіл робочого навантаження та інтеграція технологій і техніки в сільськогосподарські операції. Ретельно розробляючи робочі місця, фермери можуть створити робоче середовище, яке сприяє задоволенню працівників і мінімізує ризик професійних ризиків.

Підбір та відбір персоналу відіграє життєво важливу роль в управлінні персоналом фермерських господарств. Фермери повинні залучати та наймати відповідних кандидатів, які володіють необхідними навичками та компетенціями для виконання конкретних ролей. Цей процес включає в себе оголошення про вакансії, відбір кандидатів, проведення співбесід та перевірку рекомендацій. Ефективні практики найму та відбору дозволяють фермерам формувати кваліфіковану та різноманітну робочу силу, забезпечуючи наявність талановитих кадрів для задоволення потреб фермерського господарства. Існують певні відмінності в управлінні персоналом в тваринництві та рослинництві, що обумовлено сезонністю певного виду робіт.

Навчання та розвиток мають вирішальне значення для покращення навичок та знань персоналу фермерських господарств. Постійні навчальні програми можуть озброїти працівників новітніми сільськогосподарськими технологіями, протоколами безпеки та експлуатації техніки. Інвестуючи в навчання та розвиток, фермери можуть підвищити продуктивність праці, сприяти інноваціям та створити культуру навчання в своїй організації.

Системи управління ефективністю допомагають фермерам оцінювати роботу співробітників, надавати зворотний зв'язок і встановлювати цілі. Ці системи встановлюють чіткі очікування щодо продуктивності, відстежують прогрес та визнають виняткові результати роботи. Впроваджуючи практику управління ефективністю, фермери можуть мотивувати працівників,

визначати сфери для вдосконалення та узгоджувати індивідуальні та організаційні цілі.

Мотивація працівників є важливим фактором в управлінні персоналом фермерського господарства. Мотивовані працівники з більшою ймовірністю будуть зацікавлені, продуктивні та віддані своїй роботі. Різні теорії мотивації, такі як ієрархія потреб Маслоу та двофакторна теорія Герцберга, можуть бути застосовані для розуміння та посилення мотивації працівників у сільськогосподарському контексті. Фермери можуть впроваджувати такі стратегії, як програми визнання, стимулювання продуктивності та позитивне робоче середовище для підвищення мотивації серед свого персоналу.

На підтримку теоретичних засад, про які йшлося вище, наукові дослідження, галузеві звіти та статистичні дані надають цінну інформацію щодо управління персоналом на фермерських господарствах. У різних наукових роботах досліджуються такі теми, як вплив задоволеності роботою на утримання працівників, ефективність навчальних програм в аграрному секторі та роль керівництва в мотивації персоналу. Крім того, сільськогосподарські організації та державні установи часто публікують звіти, що містять статистичні дані про демографію ринку праці, тенденції розвитку сільськогосподарської робочої сили та виклики, що стоять перед галуззю.

Всебічно розглянувши теоретичні засади управління персоналом сільськогосподарських підприємств та включивши відповідні дослідження, графіки та статистичні дані, можна створити надійну основу для розробки інформаційної системи, яка відповідатиме конкретним потребам та викликам, що виникають у цій галузі. У наступних розділах цього розділу буде продовжено досліджування предметної області, висвітлюючи існуючі проблеми та недоліки, а також розглянуто та проаналізовано існуючі аналоги в управлінні персоналом фермерських господарств.

## 1.2 Опис предметної області

Сільське господарство, як важливий сектор світової економіки, відіграє вирішальну роль у забезпеченні продовольчої безпеки та сталого розвитку. Сільськогосподарська галузь стикається з унікальними викликами, пов'язаними з трудомісткою діяльністю, мінливими умовами навколишнього середовища та вимогами ринку. Ефективне управління персоналом фермерських господарств має вирішальне значення для оптимізації продуктивності, забезпечення ефективного розподілу ресурсів та адаптації до динамічних сільськогосподарських ландшафтів.

Нейронні мережі, підмножина штучного інтелекту (ШІ), привертають значну увагу в останні роки завдяки своїй здатності аналізувати складні закономірності та робити інтелектуальні прогнози. У контексті управління персоналом нейронні мережі можуть революціонізувати процеси прийняття рішень, автоматизувати рутинні завдання та підвищити загальну операційну ефективність. Їх можна використовувати для аналізу даних про робочу силу, прогнозування попиту на робочу силу, оптимізації планування та надання персоналізованих рекомендацій щодо професійного росту.

Інтеграція нейронних мереж в системи управління персоналом може призвести до різних переваг. Наприклад, аналізуючи історичні дані про потреби в робочій силі та продуктивність, нейромережі можуть генерувати точні прогнози щодо майбутніх кадрових потреб. Це дозволяє керівникам фермерських господарств приймати обґрунтовані рішення щодо найму, планування робочої сили та розподілу ресурсів. Окрім того, нейронні мережі можуть допомогти у виявленні закономірностей і тенденцій у роботі працівників, що дозволяє проводити цілеспрямовані тренінги та розробляти стратегії підвищення продуктивності.

Незважаючи на багатообіцяючий потенціал нейронних мереж в управлінні персоналом фермерських господарств, існує ряд проблем і недоліків, які потребують вирішення. Однією з головних проблем є

доступність та якість даних. Для навчання ефективних нейромережевих моделей необхідна значна кількість високоякісних даних. Однак збір і збереження надійних даних про персонал в аграрному секторі може бути складним завданням через такі фактори, як віддаленість роботи, обмеженість технологічної інфраструктури та проблеми з конфіденційністю даних.

Ще одна проблема полягає в інтерпретації та поясненні нейронних мереж. Як складні моделі "чорного ящика", нейронні мережі часто не мають прозорості в тому, як вони приймають свої рішення. Це ускладнює розуміння проблем управління і викликає етичні проблеми, особливо в таких чутливих сферах, як оцінка ефективності та прийняття рішень, пов'язаних з просуванням по службі або звільненням. Впровадження нейронних мереж в управління персоналом вимагає досвіду в галузі штучного інтелекту, аналізу даних та системної інтеграції. Керівники фермерських господарств і фахівці з управління персоналом можуть зіткнутися з бар'єрами, пов'язаними з технічними знаннями, розподілом ресурсів і організаційною готовністю до впровадження передових технологій.


Рисунок 1.2. – Взаємний вплив нейромереж та потреб в управлінні персоналом

Дослідивши предметну область, включаючи сільське господарство, нейронні мережі, а також пов'язані з ними проблеми і можливості в управлінні персоналом, можна визначити можливий взаємний вплив (рис.1.2) та закласти основу для розробки інформаційної системи, яка відповідає специфічним потребам і вимогам фермерського господарства.

## 1.3 Огляд і аналіз існуючих аналогів

Інтелектуальні агенти стали популярним інструментом у сфері управління персоналом, пропонуючи автоматизовану допомогу та підтримку працівникам і керівникам. Ці агенти призначені для виконання різних завдань, таких як відповіді на поширені запитання, надання інформації про політику та процедури, допомога у складанні графіків та управлінні відпустками, а також полегшення комунікації між різними відділами. Дослідження показали, що впровадження чат-ботів-помічників у системи управління персоналом призвело до значної економії часу та підвищення рівня задоволеності працівників.

У цьому розділі буде проведено всебічний огляд та аналіз існуючих аналогів, зосередившись на використанні чат-ботів, асистентів та операторів технічної підтримки, які використовують нейронні мережі. Вивчаючи ці приклади, необхідно отримати детальне уявлення про їхні особливості, функціональні можливості, сильні сторони та обмеження. Завдяки цьому аналізу можливо оцінити потенціал систем на основі нейронних мереж у покращенні управління персоналом в аграрному секторі.

Одним із важливих застосувань нейронних мереж в управлінні персоналом є використання чат-ботів для обслуговування клієнтів. Ці чат-боти призначені для обробки запитів клієнтів і надання підтримки в розмовній формі. Використовуючи нейронні мережі та алгоритми обробки природної мови (NLP), вони можуть розуміти та інтерпретувати запити користувачів, витягувати відповідну інформацію та генерувати відповідні відповіді. Ці чат-боти можуть відповідати на поширені запитання, надавати допомогу в навігації по політиках і процедурах і навіть переадресовувати складні питання людям-операторам, коли це необхідно. Завдяки використанню нейронних мереж ці чат-боти можуть безперервно навчатися і вдосконалювати свої відповіді на основі взаємодії з користувачами та зворотного зв'язку, підвищуючи свою ефективність з часом.

Наприклад, компанія Amazon, що займається електронною комерцією, впровадила чат-бота на основі нейронних мереж у відділ обслуговування клієнтів. аналізував величезні обсяги даних про клієнтів, включно з минулими взаємодіями та історією покупок, щоб надавати персоналізовані рекомендації, відповідати на запитання про товари та обробляти запити на замовлення. Система на основі нейронних мереж значно скоротила час реагування та підвищила рівень задоволеності клієнтів, пропонуючи швидку та точну допомогу.

Таблиця 1.1. – Порівняльна характеристика віртуальних асистентів

| **Аналоги** | **Взаємодія з користувачем** | **Інтеграція нейронних мереж** | **Вузькоспеціалізовані дані** |
|---|---|---|---|
| Siri | Інтерфейс природною мовою та голосові запити | Так | Ні |
| Cortana | NLP, пошукова система Bing, інтеграція даних | Так | Ні |
| Alexa | Голосова взаємодія, NLP, голосові запити | Так | Так |
| Google Assistant | Голосове та текстове введення, широкий вибір пристроїв | Так | Так |

Віртуальні помічники, такі як Alexa (рис.1.3) від Amazon, Siri (рис.1.4) від Apple або Google Assistant (рис.1.5) стають все більш популярними в різних сферах, включаючи управління персоналом. Ці віртуальні помічники використовують нейронні мережі для розуміння і обробки запитів на природній мові, що дозволяє користувачам взаємодіяти з ними за допомогою голосових команд. У контексті управління персоналом віртуальні асистенти

можуть виконувати такі завдання, як планування зустрічей, управління запитами на відпустки та надання інформації про кадрову політику. Навчаючись на великих масивах даних і використовуючи нейронні мережі, ці віртуальні помічники можуть точно інтерпретувати інструкції користувача, витягувати відповідну інформацію з баз даних персоналу і надавати своєчасні відповіді та дії.

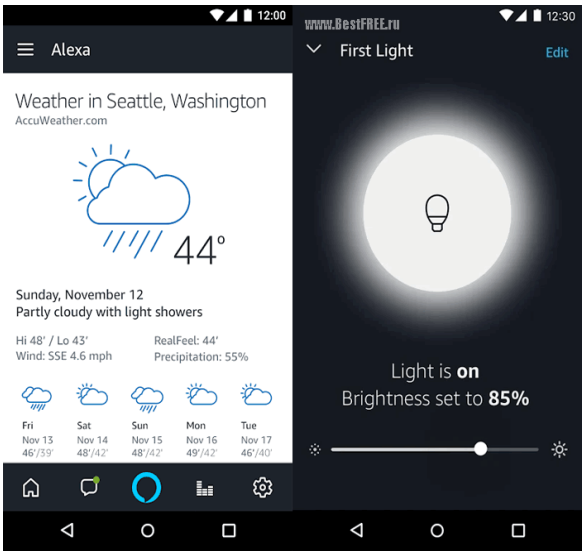



Рисунок 1.3 – Інтерфейс віртуального помічника Alexa

На рисунку 1.4 преставлено віртуальний помічних Siri від Apple, який є прикладом персонального помічника і питально-відповідальної системи, адаптованої під iOS.

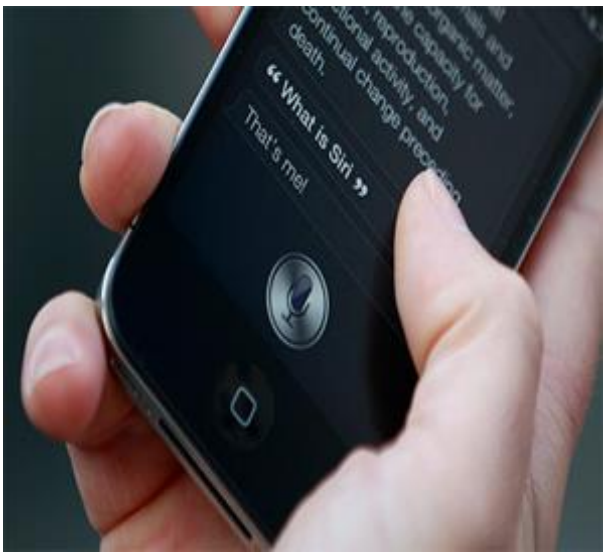



Рисунок 1.4. -Віртуальний помічник Siri

В якості експерименту компанія Google запровадила віртуального асистента на основі нейронних мереж, щоб допомогти своїм працівникам керувати робочим графіком і отримувати доступ до інформації, пов'язаної з персоналом. Працівники могли просто поговорити з віртуальним асистентом, поставити запитання про свої майбутні зміни, попросити відгул або дізнатися про політику компанії. Віртуальний асистент використовував нейронні мережі, щоб розуміти і обробляти вимовлені команди, витягувати відповідні дані з системи управління персоналом і надавати працівникам персоналізовані і точні відповіді.

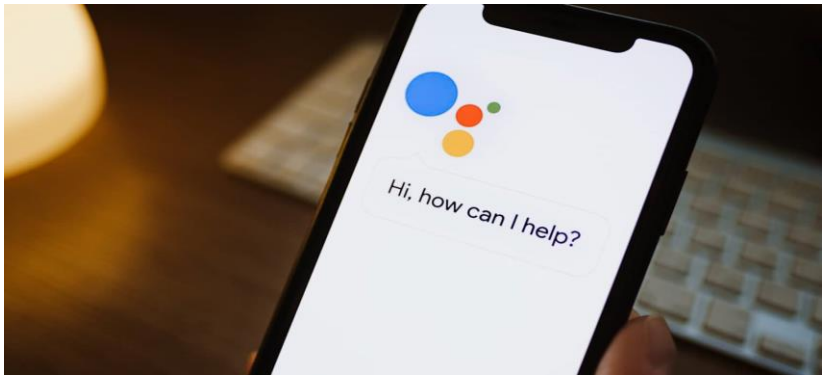



Рисунок 1.5 – Інтерфейс Google Assistant

Чат-боти на основі нейронних мереж все частіше використовуються як оператори технічної підтримки, особливо в галузях зі складними системами та продуктами. Ці чат-боти можуть розуміти технічні запити, діагностувати типові проблеми та пропонувати рішення для їх усунення. Використовуючи нейронні мережі та алгоритми машинного навчання, вони можуть постійно вдосконалювати свою базу знань і діагностичні можливості.

В аграрному секторі впровадження сервісної технічної підтримки на базі штучного інтелекту ще не відбулось, але при цьому, ця галузь є доволі перспективна, так як в інших сферах уже існують асистенти на основі ШІ які допомагають в вирішенні технічних проблем з обладнанням, програмним забезпеченням та інше. Ці інтелектуальні агенти використовують нейронні мережі для аналізу повідомлень про помилки, інтерпретації наданої користувачем інформації та пропонування покрокових рішень.

Використовуючи нейронні мережі, ці чат-боти можуть виявляти закономірності в проблемах, про які повідомляється, вчитися на успішних рішеннях і надавати все більш точні рекомендації щодо усунення несправностей, та при необхідності перевести на “живого” оператора тех підтрики, прикладами компаній які використовують цей підхід є Kyivstar, PrivatBank, Amazon etc.

Інтеграція нейронних мереж в управління персоналом має великі перспективи що вже привело до появи компаній, які ставлять метою облегшити підбір персоналу і його керування, приклад компанія Manatal, з якою взаємодіють провідні компанії світу, такі як Microsoft, Google, LinkedIn, і яка спеціалізується на програмному забезпеченні для рекрутингу персоналу.

Іншим прикладом використання нейромереж в перспективі може виступати чат бот асистент, для прикладу Chat GPT і Bing Chat, що реалізований на його основі, і який уже у вміє планувати подорожі, покупки, будувати план поїздок. А в перспективі може бути впроваджений як ШІ асистент в роботі, що буде пропонувати підтримку і рекомендації працівникам. Асистенти на базі нейромереж можуть виконувати різні завдання, включаючи управління відпустками, запити на навчання, роз'яснення політики і рекомендації щодо розвитку кар'єри. Завдяки навчанню на великих масивах даних, що містять персональні дані, кадрову політику та інструкції компанії, ці асистенти можуть надавати працівникам персоналізовану та релевантну інформацію.

У сільськогосподарській галузі асистенти з управління персоналом та кадрового менеджменту можуть допомагати працівникам фермерських господарств орієнтуватися в різних процесах, пов'язаних з управлінням персоналом. Ці асистенти використовують нейронні мережі для розуміння запитів на природній мові, вилучення відповідної інформації з записів працівників і надання рекомендацій з таких питань, як запити на відпустку, оцінка ефективності роботи і можливості навчання. Використовуючи нейронні мережі, ці асистенти можуть надавати своєчасну і точну підтримку,

підвищуючи ефективність та результативність кадрових операцій в аграрному секторі.

Наведені вище приклади підкреслюють різноманітність застосування систем на основі нейронних мереж в управлінні персоналом. Ці системи продемонстрували свою здатність розуміти запити природною мовою, пропонувати персоналізовану допомогу, надавати точні відповіді та постійно покращувати свою роботу завдяки машинному навчанню. Хоча конкретна реалізація та функціональність можуть відрізнятися в різних організаціях, використання нейронних мереж в основі довело свою цінність для вдосконалення практики управління персоналом в різноманітних сферах.

Важливо зазначити, що успіх цих аналогів залежить від різних факторів, зокрема від наявності та якості даних, процесу навчання та інтеграції з існуючими системами управління персоналом. Крім того, при впровадженні таких систем в управління персоналом фермерських господарств необхідно враховувати такі міркування, як конфіденційність даних, безпека та етичність використання нейронних мереж.

В результаті аналізу було досягнуто кількох ключових висновків. А саме, використання чат-ботів, віртуальних асистентів та операторів технічної підтримки на основі нейронних мереж має великий потенціал для трансформації практики управління персоналом в аграрному секторі. Ці системи пропонують розширені можливості розуміння запитів природною мовою, надання персоналізованої допомоги та постійного покращення своєї роботи завдяки машинному навчанню.

Чат-боти для обслуговування клієнтів довели свою ефективність в обробці запитів, наданні відповідей на поширені запитання та підвищенні рівня задоволеності клієнтів. Віртуальні асистенти продемонстрували свою здатність вирішувати різні завдання з управління персоналом, такі як складання розкладу, управління відпустками та запити щодо політики, надаючи працівникам персоналізовану та своєчасну підтримку. Оператори технічної підтримки, що працюють на основі нейронних мереж,

продемонстрували багатообіцяючу здатність діагностувати та усувати поширені технічні проблеми, скорочуючи час реагування та покращуючи досвід користувачів. Асистенти з управління персоналом, використовуючи нейронні мережі, сприяли ефективній навігації в кадрових процесах.

Аналіз також показав, що хоча ці аналоги мають великі перспективи, їхнє успішне впровадження залежить від таких факторів, як наявність і якість даних, належні процеси навчання та безперешкодна інтеграція з існуючими системами управління персоналом. До того ж, для забезпечення відповідального використання нейронних мереж необхідно враховувати міркування, пов'язані з конфіденційністю даних, безпекою та етичним використанням нейронних мереж.

Виходячи з цих висновків, очевидно, що інтеграція нейронних мереж в управління персоналом фермерських господарств має потенціал для революції в управлінні кадровими процесами. Використовуючи можливості штучного інтелекту і машинного навчання, організації можуть підвищити ефективність, точність і оперативність в управлінні персоналом, що призведе до підвищення продуктивності, задоволеності співробітників і загального операційного успіху.

Загалом, аналіз надав цінну інформацію про предметну область, заклавши основу для наступних розділів дипломної роботи. Завдяки розумінню теоретичних основ, опису предметної області та огляду існуючих аналогів, були отримані знання, необхідні для просування дослідження та сприяння розробці інноваційних рішень для управління персоналом фермерських господарств.

# РОЗДІЛ 2
# ПРОЕКТУВАННЯ ЧАТ-БОТУ ІНФОРМАЦІЙНОЇ СИСТЕМИ УПРАВЛІННЯ ПЕРСОНАЛОМ

## 2.1. Опис функціональних вимог до прототипу чат-боту асистента

Нижче наведені функціональні вимоги до прототипу чат-бота, спрямовані на створення ефективного рішення і подальшу підтримку та розширення чат-бота та забезпечення його ефективного функціонування серед цільової аудиторії та інтеграції з існуючими системами.

1. Взаємодія з користувачем:

- Чат-бот повинен мати можливість отримувати та обробляти повідомлення користувачів у режимі реального часу.
- Він повинен розуміти запити та команди на природній мові.
- Чат-бот повинен оперативно реагувати на введення користувача, надаючи релевантну і точну інформацію.
- Він повинен бути здатним працювати з декількома користувачами одночасно в чаті.
- Чат-бот повинен мати можливість брати участь в інтерактивних бесідах і підтримувати контекст повідомлень.

2. Ідентифікація користувача та контекст:

- Чат-бот повинен вміти ідентифікувати та розрізняти окремих користувачів у чаті.
- Він повинен запам'ятовувати та відстежувати певну кількість попередніх повідомлень, щоб підтримувати контекст і надавати персоналізовані відповіді.
- Чат-бот повинен зберігати та отримувати інформацію про користувача, необхідну для персоналізованої взаємодії.

3. Пошук інформації:

- Чат-бот повинен мати доступ до всеосяжної бази знань або бази даних, що містить релевантну інформацію для цільової аудиторії.
- Він повинен мати можливість отримувати інформацію з бази знань на основі запитів користувача.
- Чат-бот повинен надавати точну та актуальну інформацію у відповідь на запити користувачів.

4. Виконання завдань:

- Чат-бот повинен вміти виконувати конкретні завдання на основі команд або запитів користувача.
- Він повинен підтримувати заздалегідь визначені команди або дії для загальних завдань, таких як отримання даних користувача, надання сповіщень або запуск процесів.
- Чат-бот повинен виконувати завдання точно та ефективно, надаючи користувачам відповідний зворотний зв'язок та оновлення.

5. Обробка помилок та усунення несправностей:

- Чат-бот повинен бути оснащений функціями обробки помилок, щоб ефективно реагувати на несподівані введення користувача або системні помилки.
- Він повинен вміти розпізнавати помилки та реагувати на них, а також надавати корисні підказки або інструкції для вирішення проблем.

6. Продуктивність і масштабованість:

- Чат-бот повинен бути спроектований таким чином, щоб ефективно задовольняти зростаючий користувацький попит і масштабуватися.
- Він повинен демонструвати високу продуктивність з точки зору часу відгуку, точності та надійності.
- Чат-бот повинен бути здатним впоратися з потенційним зростанням кількості одночасних користувачів і повідомлень без шкоди для своєї функціональності.

В таблиці 2.1. описані основні функціональні вимоги до чат-боту інформаційної системи управління персоналом фермерського господарства, практична реалізація яких в проекті дозволить підвищити рівень використання нейронних мереж та чат-ботів в такій консервативній, однак досить технологічній галузі як сільське господарство.

Таблиця 2.1. – Функціональні вимог до чат-боту інформаційної системи управління персоналом

| Тип вимоги | Опис вимоги |
|---|---|
| Взаємодія з користувачем | - Обробка повідомлень користувача в реальному часі<br>- Розуміння запитів і команд природною мовою<br>- Швидкі та релевантні відповіді<br>- Одночасна взаємодія з кількома користувачами<br>- Участь в інтерактивних розмовах та підтримка контексту |
| Ідентифікація користувача та контекст | - Ідентифікація користувача в чаті<br>- Відстеження попередніх повідомлень та підтримка контексту<br>- Зберігання інформації про користувача для персоналізованої взаємодії |
| Пошук інформації | - Доступ до всеосяжної бази знань<br>- Отримання інформації на основі запитів користувача<br>- Надання точної та актуальної інформації<br>- Врахування специфіки операцій у сільському господарстві |
| Виконання завдань | - Виконання конкретних завдань на основі команд користувача<br>- Підтримка заздалегідь визначених команд або дій<br>- Точне та ефективне виконання завдань |
| Обробка помилок та усунення несправностей | - Ефективна обробка неочікуваних дій користувача або системних помилок<br>- Розпізнавання та реагування на помилки з корисними підказками |
| Продуктивність і масштабованість | - Ефективне задоволення зростаючого попиту користувачів і масштабування<br>- Високі показники часу відгуку, точності та надійності<br>- Обробка потенційного зростання кількості одночасних користувачів і повідомлень |

Ці функціональні вимоги створюють основу для прототипу чат-бота, забезпечуючи його здатність взаємодіяти з користувачами, запам'ятовувати контекст, отримувати інформацію, виконувати завдання, обробляти помилки,

інтегруватися з існуючими системами, а також підтримувати продуктивність і масштабованість. Відповідаючи цим вимогам, чат-бот буде добре оснащений, щоб ефективно допомагати користувачам та вдосконалювати процеси управління персоналом.

## 2.2. Вибір мови програмування

При розробці системи чат-ботів важливим рішенням є вибір мови програмування. Після ретельної оцінки різних варіантів, найбільш підходящою вимогам проекту мовою програмування було обрано Python, оскільки вона має низку переваг, які відповідають функціональним вимогам та цілям нашої системи чат-ботів.

По-перше, важливу роль у виборі відіграла легкість написання та читабельність коду Python: синтаксис Python простий і зрозумілий, що дозволяє розробникам висловлювати складні ідеї та реалізовувати логіку лаконічно та ефективно. Ця особливість Python не тільки підвищує швидкість розробки, але й дозволяє розробнику більш ефективно працювати з мовою, не тільки прискорюючи розробку, але й гарантуючи, що кодова база легко підтримується і є доступною для майбутніх розробників.

Крім того, розгалужена бібліотечна екосистема Python надає безліч ресурсів та інструментів для розробки чат-ботів. Наявність бібліотек, присвячених обробці природної мови (NLP), машинному навчанню та нейронним мережам, робить Python ідеальним вибором для створення і подальшого розвитку прототипа чат боту: Natural Language Toolkit (NLTK), spaCy, TensorFlow та інші бібліотеки Python, Широка підтримка обробки текстових даних дозволяє нашим чат-ботам розуміти та ефективно реагувати на запити користувачів.

На це рішення також вплинула популярність Python серед розробників. Завдяки широкому розповсюдженню Python у спільнотах, що займаються штучним інтелектом та наукою про дані, існує велика кількість навчальних

посібників та документації по створенню нейромереж та уже написанним бібліотеками для цього. Ця екосистема є безцінною для подолання викликів та прискорення процесу розробки.

До того ж, універсальність та сумісність Python з іншими технологіями та API роблять її придатною для інтеграції чатботу на інші платформи, оскільки Python можна легко інтегрувати з веб-фреймворками, базами даних та зовнішніми сервісами, що дозволяє інтегрувати чат-ботів в існуючі системи або бази даних, такі як SQLite, залежно від ваших функціональних вимог.

Хоча інші мови програмування, такі як JavaScript, Java та Ruby, мають свої переваги, Python виявився найкращим вибором для розробки нашого чат-бота завдяки простоті використання, читабельності, підтримці NLP та нейронних мереж, багатим бібліотекам, популярності серед розробників та сумісності з існуючими системами, що повністю відповідало вимогам нашого проекту.

Вибір Python як мови програмування спростить процес розробки та дозволить створити потужну та ефективну систему чат-бота, використовуючи сильні сторони мови. Поєднання читабельності, багатих бібліотек, підтримки спільноти та сумісності з Python дозволяє створити надійне рішення для чат-ботів, яке ефективно керує персоналом ферми та відповідає потребам цільової аудиторії.

## 2.3. Вибір платформи для розробки чатбот

Розглядаючи різні платформи обміну повідомленнями для розробки чат-ботів, було обрано Telegram через його унікальні переваги та придатність для проекту. Хоча на ринку існує кілька платформ для обміну повідомленнями, включаючи WhatsApp, Facebook Messenger і Viber, Telegram виділявся з кількох причин. В таблиці 2.2. наведені ключові фактори, які вплинули на рішення обрати Telegram, а також його переваги та недоліки у порівнянні з іншими месенджерами.

Таблиця 2.2. – Порівняльна характеристика платформ для чат-ботв популярних месенджерів

| | Facebook | Telegram | Skype | Viber |
|---|---|---|---|---|
| Форма використання | Умовно безоплатна | 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 вдіалог |
| 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 |

Виділимо такі переваги Telegram:

- Зручність та дружній інтерфейс. Telegram має зручний та зрозумілий інтерфейс, що дозволяє користувачам легко орієнтуватися та взаємодіяти з чат-ботом. Його інтуїтивно зрозумілий дизайн забезпечує безперебійну роботу, дозволяючи користувачам швидко зрозуміти функціональність чат-бота.
- Надійна інфраструктура та інструменти для розробки. Telegram пропонує повний набір інструментів розробки, API та документації, спеціально розроблених для створення чат-ботів. Ця готова інфраструктура спрощує процес розробки та прискорює розгортання чат-бота.
- Багатий набір функцій. Telegram пропонує ряд функцій, які розширюють функціональність чат-бота. Це підтримка мультимедійного контенту, групових чатів, обміну файлами, стікерів та ботів у каналах. Що дозволяє з часом розширити функціонал і можливості взаємодії з чат-ботом.
- Безпека та конфіденційність. Telegram приділяє велику увагу безпеці та конфіденційності. Він використовує наскрізне шифрування для всіх повідомлень, гарантуючи, що спілкування користувачів і взаємодія з чат-ботом залишаються в безпеці. Більше того, Telegram пропонує такі функції, як самознищення повідомлень, двофакторна автентифікація та можливість видаляти повідомлення з обох сторін.

До основних недоліків використання Telegram можна віднести:

- Частка ринку. Хоча Telegram має значну базу користувачів, він може не мати такого ж рівня проникнення на ринок, як деякі інші платформи для обміну повідомленнями. Наприклад, WhatsApp або Viber можуть похвалитися значно більшою базою користувачів і більш широким глобальним охопленням. Вибір Telegram може обмежити потенційне охоплення користувачів порівняно з цими більшими платформами.

- Обмеження інтеграції. Хоча Telegram надає надійну інфраструктуру для розробки чат-ботів, він може мати певні обмеження, коли йдеться про інтеграцію зі сторонніми сервісами або системами. Інші платформи обміну повідомленнями, такі як Facebook Messenger або Slack, пропонують більш широкі можливості інтеграції та API, що може бути корисним у конкретних випадках використання.

Незважаючи на ці обмеження, переваги вибору Telegram, включаючи його зручний інтерфейс, надійну інфраструктуру, велику базу користувачів, багатий набір функцій і акцент на безпеку, роблять його переконливим вибором для розробки чат-бота. Ці фактори добре узгоджуються з цілями. Крім того, постійні оновлення та фокус на вдосконаленні платформи Telegram також сприяють її придатності для розробки чат-бота.

## 2.4. Вибір нейронної моделі

У процесі проектування інформаційної системи для управління персоналом фермерського господарства вибір відповідної нейронної моделі відіграє вирішальну роль у досягненні ефективної та інтелектуальної взаємодії. Після ретельного дослідження та оцінки було обрано модель OpenAI ChatGPT завдяки її винятковим можливостям та продуктивності.

Однією з ключових причин вибору моделі OpenAI ChatGPT є її найсучасніші можливості обробки природної мови (NLP). Модель була навчена на великому корпусі різноманітних текстів, що дозволило їй розуміти і генерувати відповіді, схожі на людські. Це робить її добре придатною для розмовних додатків, таких як чат-бот для управління персоналом, де природна і контекстна взаємодія має важливе значення.

Модель OpenAI ChatGPT також пропонує перевагу універсальності та адаптивності. Вона може бути точно налаштована і пристосована до конкретних доменів і випадків використання, що дозволяє нам адаптувати її до унікальних вимог управління персоналом в умовах фермерського

господарства. Така гнучкість дозволяє моделі розуміти тонкощі завдань, політик і HR-процесів, пов'язаних з фермерськими господарствами, надаючи більш точні та релевантні відповіді користувачам.

Модель OpenAI ChatGPT має зручний і добре документований API, що спрощує її інтеграцію в інформаційну систему. API надає прості методи для надсилання запитів та отримання відповідей, що робить зручним впровадження та підтримку функціоналу чат-бота в системі управління персоналом.

Обираючи модель OpenAI ChatGPT, ми можемо використати потужність просунутого NLP, можливості кастомізації та простоту інтеграції для створення інтелектуальної та ефективної інформаційної системи управління персоналом на фермерському господарстві. Здатність моделі розуміти і генерувати відповіді на природній мові покращить взаємодію з користувачем і впорядкує процеси управління персоналом, що в кінцевому підсумку підвищить продуктивність і ефективність роботи фермерського господарства.

## 2.5. Вибір бази данних, проектування структури бази даних

Вибір правильної бази даних - це ключ до ефективного зберігання та пошуку даних у проекті. SQLite була обрана як найбільш підходящу базу даних для прототипу.

Однією з головних причин, чому була обрана SQLite, це його легкість та автономність: SQLite - це безсерверна файлова система управління базами даних, яка вимагає мінімального встановлення та налаштування. Її простота та ефективність роблять її ідеальним рішенням для додатків, що мають справу з невеликими обсягами даних, але при цьому, з розвитком проекту – може з’явитись необхідність на перехід до більш ефективної бази данних із серверним бекендом, але дякуючи SQL-ному синтаксису це буде не важко.

Важливу роль у процесі прийняття рішення також відіграла сумісність між SQLite та Python. Python має вбудовану підтримку SQLite, що дозволяє

легко інтегрувати та взаємодіяти з чат-ботом та базою даних. Ця сумісність забезпечує безперебійну комунікацію та спрощене управління даними в проекті.

Крім того, було досліджено можливість використання асинхронних бібліотек, таких як aioSQLite, у поєднанні з SQLite. Асинхронне програмування може покращити швидкість реагування та масштабованість систем чат-ботів, оскільки кілька завдань можуть виконуватися одночасно. Використання aioSQLite, бібліотеки, призначеної для асинхронних операцій з базами даних, може оптимізувати продуктивність та ефективність доступу до даних у проекті.

SQLite також відповідає стандарту ACID (Atomicity, Consistency, Isolation, Durability) і забезпечує цілісність та надійність даних. Вона підтримує транзакції, дозволяючи виконувати декілька транзакцій з базою даних як одну атомарну одиницю, забезпечуючи послідовну та надійну обробку даних.

Хоча існують й інші варіанти баз даних, такі як MySQL, PostgreSQL та MongoDB,було вирішено , що SQLite найкраще відповідає вимогам проекту чат-бота. Її легка конструкція, безшовна інтеграція з Python, підтримка асинхронних операцій за допомогою aioSQLite, дотримання принципів ACID та простота впровадження повністю відповідали функціональним вимогам нашого проекту.

Отже, етап розробки нашого проекту був зосереджений на визначенні функціональних вимог, виборі відповідної платформи, мови програмування та бази даних для системи чат-ботів. Завдяки ретельному розгляду та оцінці були прийняті обґрунтовані рішення, які відповідають цілям проекту та потребам керівного персоналу ферми.

Функціональні вимоги до прототипу чат-бота були ретельно визначені, охоплюючи такі основні функції, як аутентифікація користувачів, управління інформацією про персонал, призначення завдань, звітність та аналіз даних. Ці вимоги слугують вичерпним керівництвом для розробки та впровадження

системи чат-бота, гарантуючи, що вона відповідатиме конкретним потребам управління персоналом фермерського господарства.

Вибір Telegram як платформи для розробки чат-ботів має багато переваг. Його зручний інтерфейс, широке впровадження та надійна інфраструктура забезпечують зручний та доступний канал зв'язку для взаємодії користувачів з чат-ботом. Ба більше, вибір Python як мови програмування має значні переваги, серед яких простота написання, висока читабельність коду, широка підтримка бібліотек та широкі можливості інтеграції з нейронними мережами та сервісами обробки природної мови, такими як OpenAI.

Рішення використовувати SQLite як базу даних для нашої системи чат-ботів ґрунтувалося на її легкій та автономній природі, сумісності з Python, підтримці асинхронних операцій через aioSQLite, відповідності ACID та простоті інтеграції. Ці функції забезпечують ефективне зберігання, пошук та маніпулювання даними в системі чат-ботів, що сприяє ефективному управлінню інформацією про персонал ферми.

Поєднавши Telegram, Python та SQLite, ми створили міцний фундамент для розробки надійної та зручної системи чат-ботів. Обрані технології тісно пов'язані з цілями проекту та потребами цільової аудиторії, забезпечують безперебійну взаємодію та інтуїтивно зрозумілий інтерфейс для управління персоналом ферми.

Успішне завершення етапу проектування створює передумови для наступного етапу, який зосереджується на програмній реалізації проекту. Завдяки розробці чат-бота в Telegram, ретельному тестуванню та експлуатації, ми втілимо в життя задуману нами систему чат-ботів, трансформуючи спосіб управління персоналом ферми та підвищуючи продуктивність сільськогосподарських операцій.

Після завершення етапу проектування ми маємо чітку дорожню карту та напрямок для впровадження системи чат-ботів. Знання, отримані на етапах аналізу та проектування, будуть застосовані для створення функціонального

та ефективного чат-бота, який відповідатиме на конкретні виклики та вимоги до управління персоналом фермерських господарств.

Загалом, етап проектування відіграв важливу роль у створенні підґрунтя для успішної розробки та впровадження прототипу системи чат-бота. Ретельне врахування функціональних вимог, вибір платформи, мови програмування та бази даних гарантує, що розробляє мий  чат-бот буде ефективно допомагати в управлінні персоналом ферми, підвищуючи ефективність, зручність та продуктивність сільськогосподарських операцій.

# РОЗДІЛ 3
# ПРОГРАМНА РЕАЛІЗАЦІЯ ПРОЕКТУ

## 3.1 Розробка телеграм бота

Розробка Telegram-бота передбачає створення облікового запису бота в Telegram за допомогою BotFather та реалізацію необхідного функціоналу для взаємодії з користувачами. Проект має модульну структуру, організовану в певні папки та файли для забезпечення модульності та простоти розробки.

Для початку створюється обліковий запис бота в Telegram за допомогою BotFather. BotFather надає зручний інтерфейс для створення та управління обліковими записами ботів. Необхідні кроки включають надання імені для бота, вибір імені користувача та отримання унікального токена, який слугує ключем автентифікації для бота.

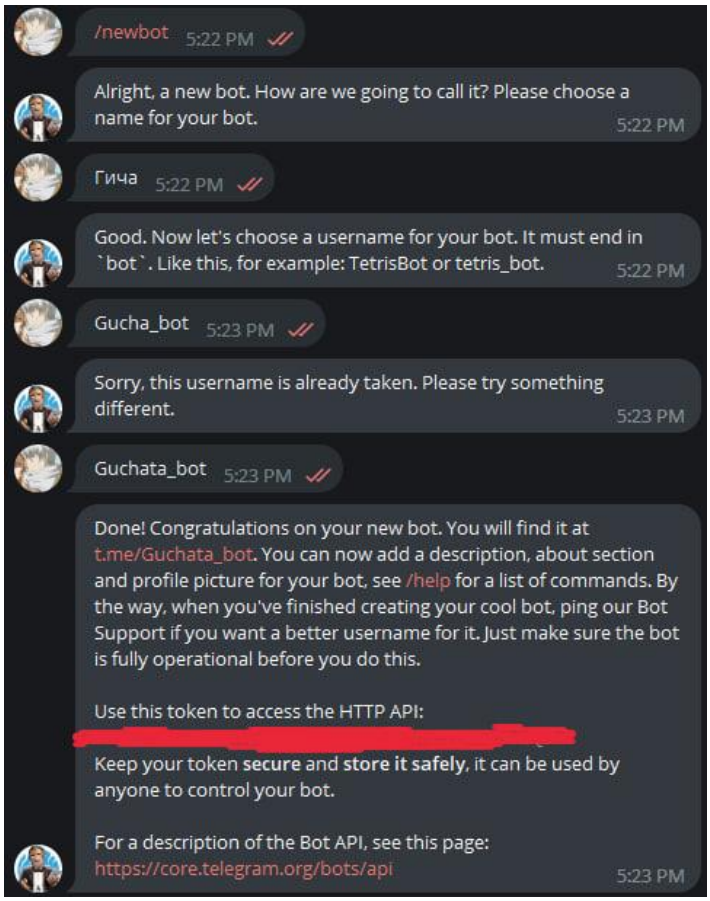



Рисунок 3.1– Фрагмент реєстрації телеграм бота*

*(токен бота приховано)

Далі представлено структуру проекту (рис.3.2) та вказано кожен файл і за що він відповідає. У кореневій папці проекту є кілька важливих каталогів і файлів. Папка "Libs" містить кастомні бібліотеки, спеціально розроблені для проекту.

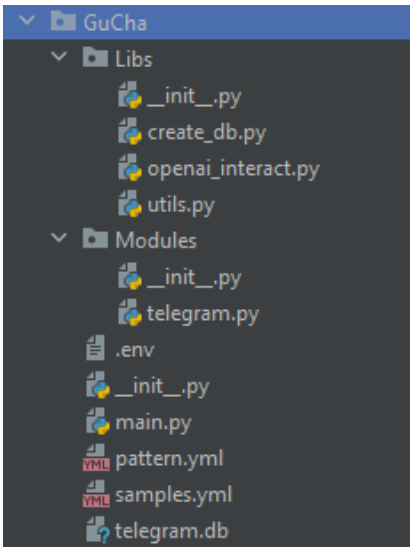



Рисунок 3.2–Файлова структура проекту

До цих бібліотек відносяться:

create_db.py: Ця бібліотека відповідає за створення бази даних під час першого запуску бота. Вона виконує необхідні налаштування та ініціалізацію схеми бази даних.

openai_interact.py: Ця бібліотека відповідає за взаємодію з OpenAI API. Вона полегшує емуляцію пам'яті та перехоплення помилок для забезпечення безперебійної взаємодії з моделями OpenAI.

utils.py: Тут розміщуються різноманітні утиліти, які підвищують зручність роботи з проектом. Ці функції можуть включати такі завдання, як завантаження даних з YAML-файлів або виконання звичайних операцій.

```
import os
import yaml
from dotenv import load_dotenv
from datetime import datetime

# Get the absolute path to the database file
load_dotenv()

# Useful funcs
def timestamp_to_datetime(unix_time):
    return datetime.fromtimestamp(unix_time).strftime("%A, %B %d, %Y at %I:%M%p %Z")

# Getting data from env
def get_db_connection_string() -> str:
    return os.path.abspath(os.path.join(os.path.dirname(__file__), '..', os.getenv('DB_CONNECTION_STRING')))

def get_bot_token() -> str:
    return os.getenv('BOT_TOKEN')

def get_openai_key() -> str:
    return os.getenv('OPENAI_KEY')

# Getting data from samples
def get_notes_sample():
    with open("samples.yml", "r", encoding='utf-8') as f:
        return yaml.safe_load(f)['notes_sample']
```

Рисунок 3.3– Фрагмент вмісту utils.py

До структури проекту також входить папка "Modules" у кореневому каталозі. У цій папці знаходиться файл telegram.py. Цей файл визначає взаємодію бота з Telegram за допомогою класу TelegramModule. Клас TelegramModule ініціалізує необхідні обробники для обробки повідомлень. Наприклад, у класі TelegramModule (рис.3.4):


Рисунок 3.4 – Структура класу TelegramModule

Ці обробники відповідають за обробку різних типів повідомлень, таких як приватні повідомлення, відповіді та згадки. Вони гарантують, що бот належним чином реагує на взаємодію з користувачем.

Крім того, файл telegram.py містить метод bot_answer(рис.3.5), який відповідає за обробку повідомлень користувачів і генерування відповідних відповідей на основі визначених шаблонів і шаблонізованих даних. Цей метод взаємодіє з OpenAI API, використовуючи кастомні бібліотеки, згадані вище, для надання інтелектуальних та контекстно-залежних відповідей на запити користувачів.

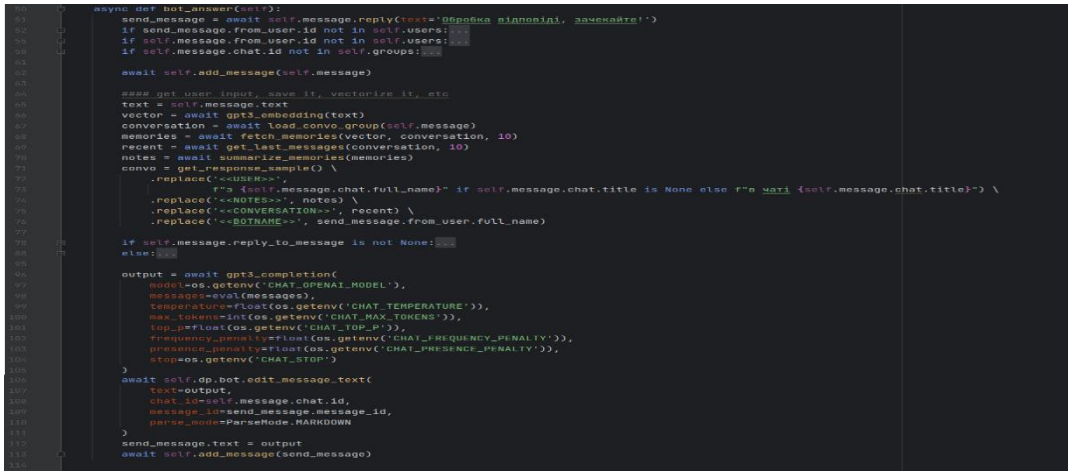



Рисунок 3.5 –Скріншот коду функції bot_answer

Коренева папка також містить наступні файли:

telegram.db: база данних в якій знаходяться всі запити користувачів і відповіді бота.

.env: Цей файл містить важливі конфігураційні змінні, такі як токени Telegram і OpenAI, ім'я файлу бази даних і налаштування моделей OpenAI. Ці змінні використовуються проектом для належної автентифікації та інтеграції із зовнішніми сервісами.

pattern.yml: Цей YAML-файл містить шаблони, які визначають ролі та вміст, в яких дані будуть передаватися через змінні виду <<BotName>>.

```
#Settings
BOT_TOKEN=WRITE_TELEGRAMBOT_TOKEN_HERE
OPENAI_KEY=WRITE_OPENAI_TOKEN_HERE
DB_CONNECTION_STRING=telegram.db
EMBEDDING_ENGINE=text-embedding-ada-002

#Chat Model Settings
CHAT_OPENAI_MODEL=gpt-3.5-turbo
CHAT_TEMPERATURE=0.5
CHAT_MAX_TOKENS=1560
CHAT_TOP_P=1
CHAT_FREQUENCY_PENALTY=0.5
CHAT_PRESENCE_PENALTY=0
CHAT_STOP=None

CHAT_MESSAGE_COUNT=20

#Convo Model Settings
CONVO_OPENAI_MODEL=gpt-3.5-turbo
CONVO_TEMPERATURE=0
CONVO_MAX_TOKENS=1024
CONVO_TOP_P=1
CONVO_FREQUENCY_PENALTY=0
CONVO_PRESENCE_PENALTY=0
CONVO_STOP=None

CONVO_COUNT=20
```

Рисунок 3.6 – Вміст файлу .env

Ці шаблони, представлені на рис.3.7, надалі обробляються моделями OpenAI для генерації змістовних відповідей.

```
chat_prompt_pattern: |
  [{'role': 'system',
  'content': f'Ти інформаціна система асистент управління персоналом фермерського господарства. '
             f'Тебе звуть <<BotName>>, ти знаходишся в телеграм чаті <<ChatName>>'},
  {'role': 'assistant', 'content': '''<<Conversation>>'''},
  {'role': 'user', 'content': '<<Message>>', 'name': '<<UserName>>'}]

chat_reply_prompt_pattern: |
  [{'role': 'system',
  'content': 'Ти інформаціна система асистент управління персоналом фермерського господарства. '
             'Тебе звуть <<BotName>>, ти знаходишся в телеграм чаті <<ChatName>>'},
  {'role': 'assistant', 'content': '''<<Conversation>>'''},
  {'role': 'user', 'content': 'Reply повідомлення: <<ReplyMessage>>', 'name': '<<ReplyUserName>>'},
  {'role': 'user', 'content': '<<Message>>', 'name': '<<UserName>>'}]
```

Рисунок 3.7 – Кастомізаційні шаблони

samples.yml: Цей YAML-файл містить шаблони обробки даних, наприклад, нотатки з попередніх розмов з ботом та іншу релевантну інформацію. Ці шаблони слугують вхідними даними для моделей OpenAI, щоб покращити контекстне розуміння бота і підвищити точність відповідей.

```
notes_sample: |
  Напишіть докладні примітки про наступне у форматі списку з дефісами, наприклад "- "
  (якщо поле пусте, не заповнюй)


  <<INPUT>>



  NOTES:


response_sample: |
  Нижче наведені нотатки з попередніх розмов <<USER>>:
  <<NOTES>>



  Нижче наведено останні повідомлення в розмові:
  <<CONVERSATION>>



  Тепер я дам максимально лаконічну відповідь:
  <<BOTNAME>>:
```

Рисунок 3.8 – шаблони нотаток

Також присутні файли __init__.py – вони необхідні для сприйняття файлів в їх папках як бібліотек чи модулів.

```
# from Libs import create_db, openai_interact, utils
# from Modules import telegram

```

Рисунок 3.9 – Вміст __init__.py

Нарешті, файл main.py відповідає за запуск бота. Він реєструє модулі, ініціалізує диспетчер, встановлює токени Telegram і OpenAI з файлу .env, створює базу даних, налаштовує логування і запускає процес опитування для отримання та обробки повідомлень користувачів.

```python
import logging
import asyncio
import openai

from aiogram import Bot, Dispatcher

from Modules.telegram import TelegramModule

from Libs.utils import get_bot_token
from Libs.create_db import create_databases
from Libs.utils import get_openai_key

logging.basicConfig(level=logging.INFO)

# Initialize the bot and dispatcher
bot = Bot(token=get_bot_token())
dp = Dispatcher(bot)
openai.api_key = get_openai_key()

# Register chat modules
TelegramModule(dp)


async def main():
    # Creating db
    await create_databases()
    # Bot Poling loop
    await dp.start_polling()


if __name__ == '__main__':

    # Start the bot
    logging.info('Starting bot...')
    asyncio.run(main())
```

Рисунок 3.10 – Код стартового файлу

Модульна структура проекту з окремими папками для бібліотек, модулів і важливих конфігураційних файлів забезпечує чітку організацію коду і полегшує подальшу розробку та підтримку. Це полегшує співпрацю між розробниками і сприяє повторному використанню коду, підвищуючи загальну ефективність процесу розробки.

Дотримуючись такого модульного підходу і правильно організовуючи файли проекту, розробка Telegram-бота стає більш керованою, що дозволяє безперешкодно інтегрувати функціонал чат-бота з платформою Telegram і моделями OpenAI.

## 3.2Тестування та експлуатація

Після розробки Telegram-бота важливо ретельно протестувати його функціональність і забезпечити його безперебійну роботу. У цьому розділі описано процес тестування та надано огляд взаємодії бота з користувачами через Telegram.

Для тестування функціоналу бота на Telegram-акаунт бота надсилається серія запитів. Ці запити (рис.3.11) імітують різні взаємодії з користувачами та сценарії, щоб оцінити, як бот реагує на них. Запити можуть включати прості текстові повідомлення, запити, пов'язані з управлінням персоналом ферми (рис.3.12), або конкретні команди для запуску певних функцій бота.

Ось кілька прикладів запитів для тестування бота:

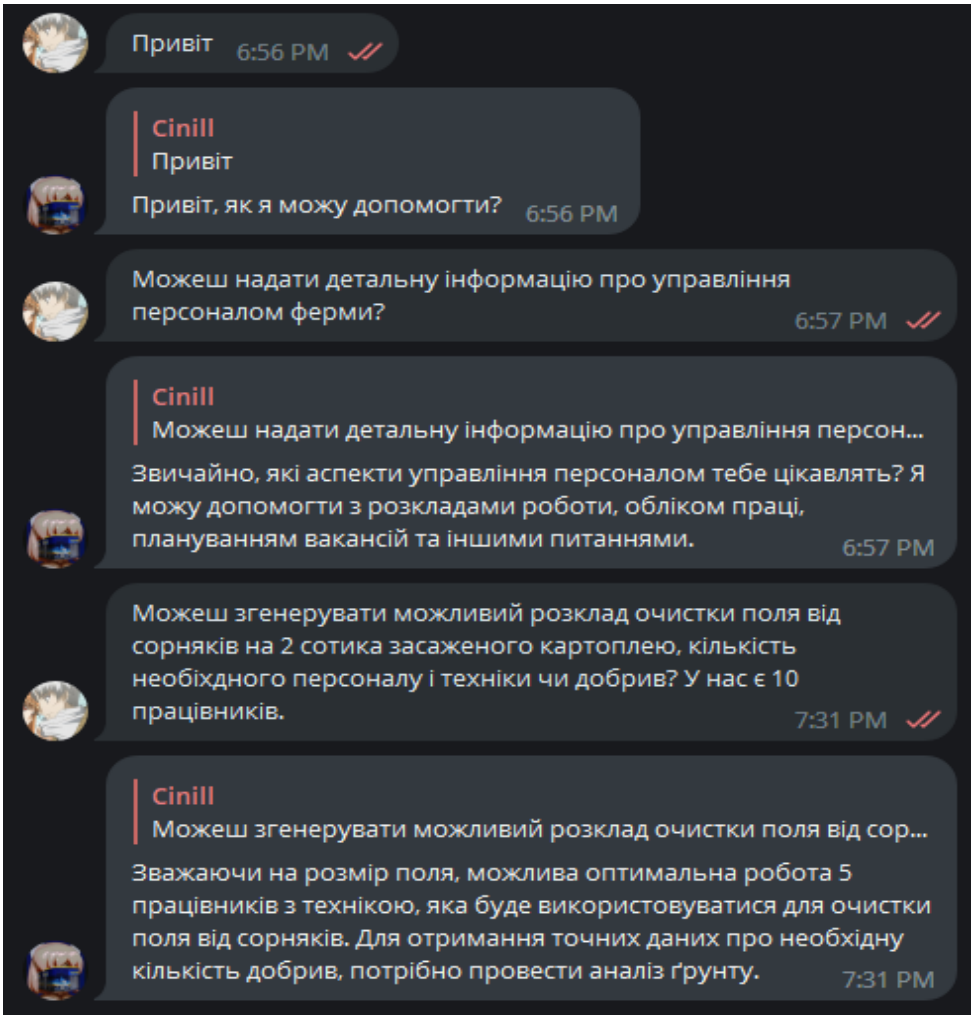



Рисунок 3.11 – Фрагмент запиту для тестування бота

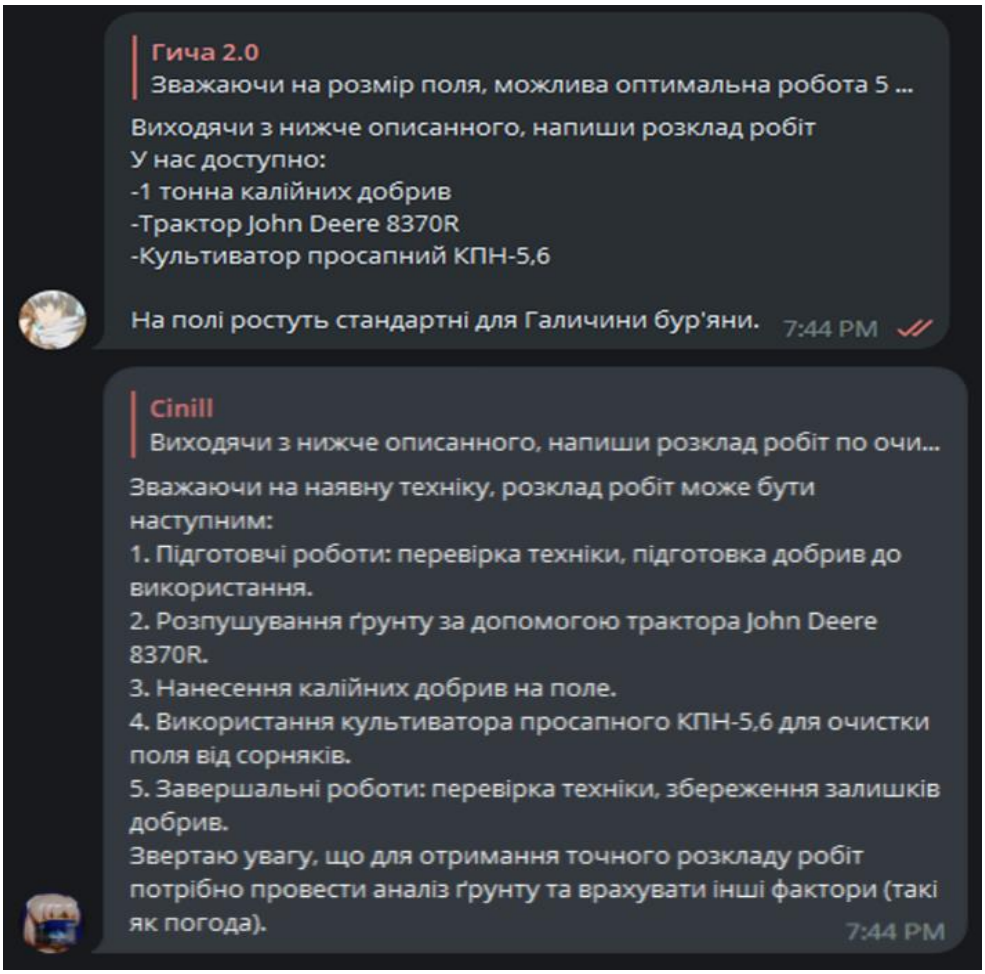



Рисунок 3.12 – фрагмент використання чат-бота

Можемо бачити, що після обговорення, з ботом, в нього залишились записи в базі данних:

| | | | | | | |
|---|---|---|---|---|---|---|
| 385 | 769070474 | 769070474 | NULL | 1686758177 | Привіт | [-0.022760814055800438, ... |
| 386 | 819102378 | 769070474 | 385 | 1686758177 | Привіт, як я можу допомогти? | [-0.025666214525699615, ... |
| 387 | 769070474 | 769070474 | NULL | 1686758242 | Можеш надати детальну інформацію про ... | [-0.006962703075259924, ... |
| 388 | 819102378 | 769070474 | 387 | 1686758242 | Звичайно, які аспекти управління персоналом ... | [-0.01307680830359459, ... |
| 389 | 769070474 | 769070474 | NULL | 1686760316 | Можеш згенерувати можливий розклад очистк... | [0.006529118865728378, 0.0062743080779910( |
| 390 | 819102378 | 769070474 | 389 | 1686760316 | Зважаючи на розмір поля, можлива оптималь... | [0.0014086113078519702, ... |
| 391 | 769070474 | 769070474 | 392 | 1686761071 | Виходячи з нижче описанного, напиши розкла... | [-0.0073966518975794315, ... |
| 393 | 819102378 | 769070474 | 394 | 1686761071 | Зважаючи на надану інформацію, можна ... | [-0.0016204803250730038, ... |

Рисунок 3.13 – приклад записів в базу данних чат-бота

Завдяки проведенню комплексного тестування та документуванню відповідей бота за допомогою скріншотів, дипломний звіт забезпечить чітке розуміння роботи бота, його здатності обробляти запити користувачів та його загальної ефективності в управлінні персоналом фермерського господарства через платформу Telegram.

## 3.3. Економічна доцільність

Розробка інформаційної системи для управління персоналом фермерського господарства у вигляді чат-бота демонструє у перспективі високу економічну доцільність з наступних причин:

Підвищення ефективності: Чат-бот спрощує процеси управління персоналом, автоматизуючи рутинні завдання, такі як складання розкладу, призначення завдань та комунікація. Така автоматизація підвищує операційну ефективність, зменшує ручну працю та покращує розподіл ресурсів, що призводить до економії коштів та підвищення продуктивності. Хоч і для цього необхідна людина оператор, але в перспективі це затребуваних ресурсів.

Скорочення витрат: Замінюючи ручну працю помічником у вигляді чат-бота, проект знижує трудові витрати, пов'язані з управлінням персоналом. Чат-бот може обробляти велику кількість запитів одночасно, усуваючи потребу в додаткових людських ресурсах. Таке скорочення витрат сприяє підвищенню фінансової життєздатності та прибутковості.

Покращене прийняття рішень: Чат-бот надає доступ в режимі реального часу до відповідної інформації, такої як дані про працівників, показники ефективності та інструкції з техніки безпеки. Це дозволяє керівникам фермерських господарств приймати рішення на основі даних, оптимізувати використання робочої сили та визначати сфери для вдосконалення. Удосконалений процес прийняття рішень призводить до більш ефективного розподілу ресурсів та покращення загальної продуктивності ферми.

Масштабованість та адаптивність: Система чат-ботів може бути легко масштабована та адаптована до мінливих потреб фермерського господарства. У міру зростання потреб фермерського господарства або операційних змін чат-бот можна оновлювати, додаючи нові функції та задовольняючи зростаючі запити користувачів. Така масштабованість гарантує, що система залишається економічно життєздатною в довгостроковій перспективі.

Покращений користувацький досвід: Чат-бот надає зручний інтерфейс для взаємодії персоналу ферми з системою. Працівники можуть отримувати доступ до інформації, надсилати запити та отримувати швидкі відповіді, що покращує їхній загальний досвід та підвищує задоволеність роботою. Позитивний користувацький досвід сприяє підвищенню залученості працівників, їхньому утриманню та, зрештою, покращенню продуктивності фермерського господарства.

Конкурентна перевага: Впровадження інноваційної інформаційної системи, такої як чат-бот для управління персоналом, дає фермерським господарствам конкурентну перевагу в галузі. Це демонструє прихильність до технологічного прогресу, ефективної роботи та добробуту працівників. Така диференціація може залучити таланти, покращити репутацію фермерського господарства та відкрити можливості для співпраці та партнерства.

З огляду на потенційне скорочення витрат, підвищення ефективності, покращення можливостей прийняття рішень, масштабованість, покращення користувацького досвіду та конкурентні переваги, розробка інформаційної системи для управління персоналом фермерського господарства у вигляді чат-бота є економічно доцільною. Використовуючи технології для оптимізації процесів управління персоналом, проект пропонує відчутні фінансові вигоди, операційні покращення та довгострокову стійкість у динамічній сільськогосподарській галузі.

Загалом завершальні роботи по реалізації проекту були присвячений проектуванню, розробці, тестуванню та економічному обґрунтуванню чат-бота-помічника.

Процес розробки був модульним, з чіткою структурою папок і спеціальними бібліотеками для конкретних функцій. Модуль Telegram відповідав за взаємодію між чат-ботом і платформою Telegram, включаючи обробку повідомлень і генерацію відповідей. Крім того, в проекті використовувалася база даних для зберігання релевантної інформації та

зовнішні бібліотеки для безперешкодної інтеграції з мовними моделями OpenAI.

Було проведено тестування та експлуатацію чат-бота, що продемонструвало його функціональність та продуктивність. Взаємодія з чат-ботом у середовищі Telegram дозволила перевірити його здатність розуміти запити користувачів, надавати адекватні відповіді та виконувати визначені функціональні вимоги.

До того ж, було оцінено економічну доцільність проекту з урахуванням технологічного прогресу, зростаючої потреби в автоматизації в сільськогосподарській галузі, переваг чат-ботів-помічників та потенційної економії коштів. Розширення проекту від тези до повноцінної системи продемонструвало багатообіцяючі перспективи для покращення управління персоналом, підвищення ефективності витрат та покращення користувацького досвіду в аграрному секторі.

Таким чином, програмна реалізація проекту дозволила успішно реалізувати задуманий чат-бот-помічник для підтримки, управління персоналом фермерського господарства. Функціональність системи, модульна конструкція та економічна доцільність закладають міцний фундамент для подальшого розвитку та впровадження інформаційної системи. Подальша робота може включати доопрацювання можливостей чат-бота, проведення широкого користувацького тестування та вивчення можливостей інтеграції з існуючими системами управління фермерським господарством.

# РОЗДІЛ 4
# ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

## 4.1. Аналіз травмонебезпечних ситуацій під час виконання робіт

Розробка та вживання ефективних заходів запобігання аварійним і травмонебезпечних ситуаціям можливі лише при завчасному виявлення тих небезпек, з яких починаються процеси їх формування. Оскільки небезпечні умови не завжди завчасно можна виявити, а для вивчення небезпечних дій іноді потрібно багато часу, щоб зібрати статичний матеріал, то і методи виявлення цих небезпек повинні бути відповідно диференційовані.

Таблиця 4.1. – Моделі формування та виникнення травмонебезпечних і аварійних ситуацій

| Вид робіт, виробничий підрозділ, робоче місце, виробниче обладнання, склад агрегату | Виробнича небезпека | | | Можливі наслідки | Заходи запобігання небезпечним ситуаціям |
|---|---|---|---|---|---|
| | Небезпечна умова (НУ) | Небезпечна дія (НД) | Небезпечна ситуація (НС) | | |
| Виконання робіт із електрообладнанням | Не вимкнено живлення. Відсутність заземлення. | Нехтування правилами ТБ | Ураження струмом | Травма (Т) | Проведення повторного інструктажу з ТБ. Розробка нових способів захисту. Встановлення заземлення. |
| НД<br>НУ → НС → Т | | | | | |

Відповідно до аналізу небезпечних умов, які існують у виробничому процесі виокремлено такі наступні за характером дії на працівника їх групи:

– характеризують стан або рівень небезпеки обладнання, які використовуються.

– сприяють виникненню технологічних помилок обслуговуючого персоналу впродовж виробничого процесу;

– створювати умови та можливість проникнення працівника в небезпечну зону;

– приводять до виникнення небезпечних дій (внаслідок низького рівня професійної підготовки працівників та організації навчання з охорони праці).

– Моделі формування та виникнення травмонебезпечних і аварійних ситуацій в комп’ютерному кабінеті представлено у вигляді моделі формування та виникнення травмонебезпечних і аварійних ситуацій, що були наведені в таблиці.

## 4.2. Структурно-функціональний аналіз дотримання охорони праці при виконання при роботі з комп’ютером

При виконанні роботи з комп'ютером важливо забезпечити належне дотримання норм охорони праці. Для цього необхідно провести структурно-функціональний аналіз, що дозволить виявити елементи та процеси, пов'язані з безпекою праці під час роботи з комп'ютером. Нижче наведено кроки, які допоможуть у проведенні аналізу:

1. Визначення робочого місця: Аналізується структура та організація робочого місця, включаючи комп'ютерну систему, робочий стіл, стілець, освітлення та інші елементи. Важливо забезпечити правильну постановку обладнання, оптимальні умови освітлення, регулювання висоти столу та стільця для запобігання неправильній позі та напруженню м'язів.

2. Оцінка ризиків: Визначення потенційних ризиків, пов'язаних з роботою з комп'ютером, таких як перенапруження очей, неправильна постава, пов'язані з довготривалим сидінням та відсутністю перерв. Оцінка факторів, що можуть впливати на здоров'я працівників, таких як шум, радіація, електромагнітні поля тощо.

3. Розробка процедур безпеки: Розроблення та впровадження процедур, спрямованих на запобігання можливим травмам та проблемам зі здоров'ям, пов'язаним з роботою з комп'ютером. Це можуть бути рекомендації щодо правильної постави, використання захисних окулярів, проведення регулярних перерв для відпочинку та розтяжки, а також застосування ергономічних принципів.

4. Навчання та тренування персоналу: Проведення навчання та тренування працівників з питань безпеки праці при роботі з комп'ютером. Це включає ознайомлення з правилами безпеки, освіту щодо користування комп'ютером та його периферійними пристроями, а також навчання профілактичних вправ та розтяжок для зменшення напруги в м'язах.

5. Регулярні перевірки та аудит: Проведення регулярних перевірок робочих місць та комп'ютерних систем для виявлення можливих недоліків та проблем безпеки праці. Аудит безпеки праці допоможе забезпечити виконання встановлених стандартів та процедур безпеки.

Структурно-функціональний аналіз дотримання охорони праці при роботі з комп'ютером допомагає ідентифікувати потенційні ризики та визначити необхідні заходи для забезпечення безпеки працівників. Виконання цього аналізу дозволяє зменшити випадки травм та проблем зі здоров'ям, пов'язаними з роботою з комп'ютером, та забезпечити належні умови праці.

## 4.3. Обґрунтування організаційно-технічних рекомендацій з охорони праці

Організаційно-технічні рекомендації з охорони праці є важливим етапом в забезпеченні безпеки працівників під час виконання роботи. Для покращення умов праці з комп'ютером та запобігання травмам та проблемам зі здоров'ям, пов'язаним з цим, нижче наведено обґрунтування організаційно-технічних рекомендацій:

1. Регулярні перевірки технічного стану обладнання: Запровадження систематичних перевірок та обслуговування комп'ютерної техніки з метою виявлення можливих проблем та усунення їх вчасно. Це включає перевірку роботи жорсткого диска, системи охолодження, клавіатури, миші та інших пристроїв.

2. Забезпечення правильної постановки обладнання: Переконання, що комп'ютерна система, монітор, клавіатура та миша розташовані на робочому столі відповідно до ергономічних принципів. Важливо забезпечити оптимальну висоту столу та стільця, належне освітлення та правильну поставу для зменшення негативного впливу на здоров'я працівників.

3. Встановлення регулярних перерв: Рекомендація встановити регулярні перерви для працівників, які працюють з комп'ютером, з метою запобігання перенапруженням та напругам у м'язах. Це може бути перерва кожні 1-2 години для короткого відпочинку, розтяжок та виконання спеціальних вправ для очей та шиї.

4. Навчання та підвищення свідомості працівників: Проведення навчання та інформування працівників про правила безпеки при роботі з комп'ютером, включаючи правильну поставу, виконання перерв, розтяжок та освітні програми щодо управління стресом та зменшення напруження.

5. Моніторинг та аналіз показників безпеки праці: Встановлення системи моніторингу та аналізу показників безпеки праці, таких як кількість травматичних випадків, час відпочинку, оцінка задоволеності працівників тощо. Це допоможе ідентифікувати проблемні ситуації та приймати відповідні заходи для покращення безпеки праці.

Обґрунтування організаційно-технічних рекомендацій з охорони праці є важливим етапом у забезпеченні безпеки та здоров'я працівників під час виконання робіт з комп'ютером. Ці рекомендації спрямовані на запобігання можливим ризикам та створення комфортних умов праці, що позитивно впливає на продуктивність та благополуччя працівників.

## 4.4. Безпека в надзвичайних ситуаціях

Забезпечення захисту населення і території у разі загрози і виникнення надзвичайних ситуацій є одним з най важливіших завдань держави.

Захист населення є системою загально державних заходів, які реалізуються центральними і місцевими органами виконавчої влади, виконавчими органами влад, органами управління з питань надзвичайних ситуацій та цивільного захисту населення, підпорядкованими їм системами, та підприємств, що забезпечують виконання організаційних, інженерно – технічних, санітарно – гігієнічних, проти епідемічних та інших заходів у сфері запобігання та ліквідації наслідків надзвичайних ситуацій.

Загрози життєво важливих інтересів громадян, держави, суспільства поділяють на зовнішні та внутрішні, виконують під час надзвичайних ситуацій техногенного та природного характеру та воєнних конфліктах.

Принципи захисту випливають з основних положень Женевської конвенції щодо захисту жертв війни та додаткових протоколів до неї, можливого характеру воєнних дій, реальних можливостей держави щодо створення матеріальної бази захисту. З метою захисту населення, зменшення втрат та шкоди економіці в разі виникнення надзвичайних ситуацій має право проводитись спеціальний комплекс заходів.

Оповіщення та інформування, яке досягається завчасним створенням і підтримкою в постійній готовності загально державної,територіальних та об’єктивних систем оповіщення населення.

# ВИСНОВКИ

Під час виконання кваліфікаційної роботи було здійснено проектування та розробку інформаційної системи підтримки, управління персоналом фермерського господарства у вигляді чат-бота. Аналіз предметної області забезпечив міцний теоретичний фундамент, дослідивши організаційні аспекти та теоретичні основи управління персоналом фермерського господарства. Ці знання заклали основу для розуміння потенційного впливу нейронних мереж та виявлення існуючих проблем і недоліків у цій галузі.

Огляд та аналіз існуючих аналогів пролив світло на сучасний ринок чат-ботів, нейронних мереж та систем управління персоналом. Вивчення їхніх функціональних можливостей, сильних і слабких сторін дозволило отримати цінну інформацію для розробки прототипу чат-бота.

Наступні розділи були присвячені розробці та впровадженню прототипу чат-бота. Вибір платформи Telegram був обґрунтований її зручністю та готовою інфраструктурою, а вибір Python як мови програмування - простотою використання, читабельністю коду та широкою бібліотечною підтримкою, включаючи взаємодію з нейронними мережами та сервісами на кшталт OpenAI.

Процес розробки передбачав створення модульної структури проекту з виділеними папками для бібліотек, модулів та основних файлів. Така організація дозволила ефективно управляти кодом і полегшила взаємодію чат-бота з платформою Telegram. Етап тестування та експлуатації переконався, що чат-бот працює відповідно до очікувань, реагує на повідомлення користувачів та ефективно використовує свій обсяг пам'яті.

У проекті були враховані міркування з охорони праці, проаналізовані потенційні небезпеки та надані організаційно-технічні рекомендації щодо забезпечення безпеки на робочому місці під час виконання сільськогосподарських робіт та завдань, пов'язаних з комп'ютерною технікою.

Такий проактивний підхід підкреслює прихильність до добробуту та захисту персоналу ферми.

З економічної точки зору, розробка прототипу чат-бота демонструє великий потенціал. Автоматизація процесів управління персоналом, покращення комунікації та оптимізація процесу прийняття рішень може призвести до економії коштів та підвищення операційної ефективності. Крім того, масштабованість проекту дозволяє в майбутньому розширювати та кастомізувати його відповідно до конкретних потреб різних фермерських господарств та сільськогосподарських операцій.

Таким чином, розробка інформаційної системи для підтримки, управління персоналом фермерського господарства у вигляді чат-бота пропонує значні переваги з точки зору ефективності, комунікації, безпеки та економічної доцільності. Поєднуючи теоретичні основи, технологічні досягнення та практичну реалізацію, цей проект зробив внесок у сферу управління персоналом фермерських господарств. Він являє собою цінний інструмент для оптимізації роботи фермерських господарств, підвищення продуктивності та забезпечення добробуту персоналу фермерських господарств. Успішне завершення цієї кваліфікаційної роботи створює основу для майбутніх досягнень та інтеграції прототипу чат-бота в реальні умови сільського господарства та господарської дільності фермерських господарств.

# БІБЛІОГРАФІЧНИЙ СПИСОК

1. Данюк В.М. Менеджмент персоналу: Навч. посіб. За ред. В.М. Данюка, В.М. Петюха. 2-ге вид., без змін. К.: КНЕУ, 2016. 398 с.

2. Данилюк І., Бабала Л., Хома Н. HRM-системи управління персоналом. Економічний аналіз. 2022. Том 32. № 3. С. 240- 246. doi: 10.35774/econa2022.03.240

3. Гогот М.М. Чупріна М.О. Використання інформаційних систем в управлінні персоналом. [Інтернет-ресурс]. – Режим доступу: http://ape.fmm.kpi.ua/article/view/102734

4. Гафіяк А.М. Проблеми створення автоматизованої інформаційної системи управління персоналом. Економіка і суспільство. Випуск 13. 2017. С.1483-1487.

5. Колот А.М. Мотивація персоналу: Підручник. Вид. 2-ге, без змін. К.: КНЕУ, 2016. 340 с.

6. Python: 7 Important Reasons Why You Should Use Python [Інтернет-ресурс]. – Режим доступу: https://medium.com/@mindfiresolutions.usa/python-7-important-reasons-why-you-should-use-python-5801a98a0d0b

7. Провотар О.І. Клочко Х.А. Особливості та проблеми віртуального спілкування за допомогою чат-ботів. Наукові праці ВНТУ: Інформаційні технології та комп'ютерна техніка. 2013. № 3.

8. Telegram Bot API [Інтернет-ресурс]. – Режим доступу: https://core.telegram.org/bots/api

9. Aiogram [Інтернет-ресурс]. – Режим доступу: https://docs.aiogram.dev/en/latest/

10. SQLite Home Page, Documentation [Інтернет-ресурс]. – Режим доступу: https://www.sqlite.org/docs.html

11. Тимочко В.О., Городецький І.М., Березовецький А.П., Мазур І.Б. та ін.Безпека життєдіяльності та охорона праці. Навч. посібник. Львів: Сполом. 2022. 376 с.

12. Закон України ―Про охорону праці‖. [Інтернет-ресурс]. Режим доступу: http://zakon2.rada.gov.ua/laws/show/2694-12.

13. OpenAI Documentation [Інтернет-ресурс]. – Режим доступу: https://platform.openai.com/docs/guides/gpt

14. 5 віртуальних асистентів: підбірка до Міжнародного дня віртуальних помічників [Інтернет-ресурс]. – Режим доступу: https://it-kharkiv.com/5-virtualnyh-asystentiv-pidbirka-do-mizhnarodnogo-dnya-virtualnyh-pomichnykiv/ вільний

15. AWS Chatbot [Інтернет-ресурс]. – Режим доступу: https://aws.amazon.com/ru/chatbot/

16. Viraj A. How To Build Your Own Chatbot Using Deep Learning [Інтернет-ресурс]. – Режим доступу: https://towardsdatascience.com/how-to-build-your-own-chatbot-using-deep-learning-bb41f970e281

17. Нейромережі пишуть книги та рятують життя: що таке нейронна мережа і як вона працює [Інтернет-ресурс]. – Режим доступу: https://mc.today/uk/shho-take-nejronna-merezha/

18. Як ChatGPT вплине на сферу менеджменту й управління персоналом [Інтернет-ресурс]. – Режим доступу: https://blog.liga.net/user/ovysotskyi/article/50605

19. Нейромережа — що це таке, як працює та навіщо потрібна. [Інтернет-ресурс]. – Режим доступу: https://termin.in.ua/neyromerezha/

20. Курніков Д.С., Петров С.А. Використання нейронних мереж економіки. Juvenisscientia. 2017. №6. З. С. 10-12.

21. Yang F., Wang K., Han H., Qiao Zh. (2018). A Cloud-Based Digital Farm Management System for Vegetable Production Process Management and Quality Traceability. Sustainability, 2018. doi:10. 4007.10.3390/su10114007.

22. Smart agrifood. The new revolution in farming. [Інтернет-ресурс]. – Режим доступу: https://www.fiware.org/community/smart-agrifood/

# ДОДАТКИ

**Додаток А**

Глосарій основних термінів

| Термін | Опис терміну |
|---|---|
| *Основні поняття та категорії предметної області та проекту* | |
| Чат-бот | Комп'ютерна програма, призначена для імітації розмови з користувачами-людьми, як правило, за допомогою текстової взаємодії. |
| Управління фермерським персоналом | Процес організації та нагляду за робочою силою, задіяною в операціях на фермі, включаючи такі завдання, як підбір персоналу, складання графіків, навчання та управління продуктивністю. |
| Нейронні мережі | Обчислювальна модель, натхненна структурою та функціями біологічних нейронних мереж. Складається з взаємопов'язаних штучних нейронів, які можуть навчатися і приймати рішення на основі вхідних даних. |
| Прототип | Попередня версія або модель системи чи продукту, що використовується для тестування та оцінки перед остаточною реалізацією. |
| Функціональні вимоги | Специфікації, які описують передбачувану поведінку і функціональність системи або програмного додатку. |
| Платформа | Базова технологія або фреймворк, на якому побудований і працює програмний додаток або система. |
| Мова програмування | Формальна мова, яка використовується для написання комп'ютерних програм, що складається з набору інструкцій і синтаксичних правил, які визначають поведінку програми. |
| База даних | Структурована колекція даних, що зберігається та організована в комп'ютерній системі, призначена для ефективного управління та пошуку інформації. |
| Інформаційна система | Система, яка збирає, зберігає, обробляє та поширює дані або інформацію для підтримки прийняття рішень, координації та управління в організації. |
| Тестування | Процес оцінки функціональності, продуктивності та надійності програмної системи шляхом систематичного виконання тестових кейсів та сценаріїв. |

**Додаток Б**

Програмний код проекту

Libs:

--create_db.py:

```
import aiosqlite

from .utils import get_db_connection_string


async def create_databases(db=get_db_connection_string()) -> None:
    async with aiosqlite.connect(db) as conn:
        await conn.execute('CREATE TABLE IF NOT EXISTS users'
                           '(user_id INTEGER PRIMARY KEY, user_name TEXT,
user_nick TEXT);')

        await conn.execute('CREATE TABLE IF NOT EXISTS groups '
                           '(group_id INTEGER PRIMARY KEY, group_name TEXT,
group_title TEXT);')

        await conn.execute('CREATE TABLE IF NOT EXISTS user_group_link '
                           '(user_id INTEGER,'
                           'group_id INTEGER,'
                           'FOREIGN KEY (user_id) REFERENCES users(user_id),'
                           'FOREIGN KEY (group_id) REFERENCES
groups(group_id),'
                           'PRIMARY KEY (user_id, group_id));')

        await conn.execute('CREATE TABLE IF NOT EXISTS messages '
                           '(message_id INTEGER,'
                           'user_id INTEGER,'
                           'chat_id INTEGER,'
                           'reply_id INTEGER,'
                           'time INTEGER,'
                           'text TEXT,'
                           'vector TEXT,'
                           'FOREIGN KEY (user_id) REFERENCES users(user_id),'
                           'FOREIGN KEY (chat_id) REFERENCES
groups(group_id));')

        await conn.execute('CREATE TABLE IF NOT EXISTS notes'
                           '( uuid TEXT, uuids TEXT, notes TEXT, time
INTEGER, times TEXT, vector TEXT);')
```

--openai_interact.py

```
import openai
import re
import time
import os
from time import sleep, time
import numpy as np
from numpy.linalg import norm
from uuid import uuid4
import aiosqlite

from .utils import get_db_connection_string, get_notes_sample,
get_openai_key, \
    timestamp_to_datetime

openai.api_key = get_openai_key()


async def gpt3_embedding(content, engine='text-embedding-ada-002'):
    content = content.encode(encoding='utf-8', errors='ignore').decode()
    response = openai.Embedding.create(input=content, engine=engine)
    vector = response['data'][0]['embedding']  # this is a normal list
    return vector


async def summarize_memories(memories):  # summarize a block of memories into
one payload
    memories = sorted(memories, key=lambda d: d['time'], reverse=False)  #
sort them chronologically
    block = ''
    identifiers = []
    timestamps = []
    for mem in memories:
        block += mem['text'] + '\n\n'
        identifiers.append(mem['message_id'])
        timestamps.append(mem['time'])
    block = block.strip()

    notes = await gpt3_completion(
                                  model=os.getenv('CONVO_OPENAI_MODEL'),
                                  messages=[{'role': 'assistant', 'content':
get_notes_sample().replace('<<INPUT>>', block)}],

temperature=float(os.getenv('CONVO_TEMPERATURE')),

max_tokens=int(os.getenv('CONVO_MAX_TOKENS')),
                                  top_p=float(os.getenv('CONVO_TOP_P')),

frequency_penalty=float(os.getenv('CONVO_FREQUENCY_PENALTY')),

presence_penalty=float(os.getenv('CONVO_PRESENCE_PENALTY')),
                                  stop=os.getenv('CONVO_STOP')
                                  )

    ####   SAVE NOTES
    vector = await gpt3_embedding(block, os.getenv('EMBEDDING_ENGINE'))
    async with aiosqlite.connect(get_db_connection_string()) as conn:
        await conn.execute(
            'INSERT INTO notes (uuid, uuids, notes,  time, times, vector)'
            'VALUES (?, ?, ?, ?, ?, ?);',
            (str(uuid4()), str(identifiers), notes, time(), str(timestamps),
str(vector)))
        await conn.commit()
```

```
    return notes


async def fetch_memories(vector, logs, count):
    scores = list()

    for i in logs:
        if vector == i['vector']:
            # skip this one because it is the same message
            continue
        score = await similarity(i['vector'], vector)
        i['score'] = score
        scores.append(i)
    ordered = sorted(scores, key=lambda d: d['score'], reverse=True)
    # TODO - pick more memories temporally nearby the top most relevant
memories
    try:
        ordered = ordered[0:count]
        return ordered
    except:
        return ordered


async def load_convo_user(message, count=10):
    async with aiosqlite.connect(get_db_connection_string()) as conn:
        conn.row_factory = aiosqlite.Row

        async with conn.execute('SELECT * FROM messages WHERE user_id = ? AND
chat_id = ?',
                                (message.from_user.id, message.chat.id)) as
cursor:
            result = eval(
                str(list(map(lambda mess: dict(mess), await
cursor.fetchmany(count)))).replace("'[", "[").replace("]'",

"]"))

    ordered = sorted(result, key=lambda d: d['time'], reverse=False)  # sort
them all chronologically
    return ordered


async def load_convo_group(message, count=10):
    async with aiosqlite.connect(get_db_connection_string()) as conn:
        conn.row_factory = aiosqlite.Row

        async with conn.execute('SELECT * FROM messages WHERE chat_id = ?',
                                (message.chat.id,)) as cursor:
            result = list(map(lambda mess: dict(mess), await
cursor.fetchmany(count)))

            for i in result:
                i['vector'] = eval(i['vector'])

    ordered = sorted(result, key=lambda d: d['message_id'], reverse=False)  #
sort them all chronologically
    return ordered


async def similarity(v1, v2):
    return np.dot(v1, v2) / (norm(v1) * norm(v2))  # return cosine similarity
```

```
async def get_last_messages(conversation, limit):
    try:
        short = conversation[-limit:]
    except:
        short = conversation
    output = ''
    for i in short:
        async with aiosqlite.connect(get_db_connection_string()) as conn:
            async with conn.execute('SELECT user_name FROM users WHERE
user_id = ?', (i['user_id'],)) as cursor:
                user_name = await cursor.fetchone()
                output += f'{user_name[0]}: {i["text"]}, time:
{timestamp_to_datetime(i["time"])}\n\n'
    output = output.strip()
    return output


async def gpt3_completion(
        model='gpt-3.5-turbo',
        messages=None,
        temperature=0.0,
        top_p=1.0,
        max_tokens=1024,
        frequency_penalty=0.0,
        presence_penalty=0.0,
        stop=None):
    if messages is None:
        messages = [{"role": "assistant", "content": "Say hello"}]
    max_retry = 5
    retry = 0
    while True:
        try:

            response = openai.ChatCompletion.create(
                model=model,
                messages=messages,
                temperature=temperature,
                max_tokens=max_tokens,
                top_p=top_p,
                frequency_penalty=frequency_penalty,
                presence_penalty=presence_penalty,
                stop=stop)
            text = response['choices'][0].message.content
            text = re.sub('[\r\n]+', '\n', text)
            text = re.sub('[\t ]+', ' ', text)
            return text
        except Exception as oops:
            retry += 1
            if retry >= max_retry:
                return "GPT3 error: %s" % oops
            print('Error communicating with OpenAI:', oops)
            sleep(1)
```

--utils.py:

```
import os
import yaml
from dotenv import load_dotenv
from datetime import datetime

# Get the absolute path to the database file
load_dotenv()

# Useful funcs
def timestamp_to_datetime(unix_time):
    return datetime.fromtimestamp(unix_time).strftime("%A, %B %d, %Y at
%I:%M%p %Z")

# Getting data from env
def get_db_connection_string() -> str:
    return os.path.abspath(os.path.join(os.path.dirname(__file__), '..',
os.getenv('DB_CONNECTION_STRING')))

def get_bot_token() -> str:
    return os.getenv('BOT_TOKEN')

def get_openai_key() -> str:
    return os.getenv('OPENAI_KEY')

# Getting data from samples
def get_notes_sample():
    with open("samples.yml", "r", encoding='utf-8') as f:
        return yaml.safe_load(f)['notes_sample']

def get_response_sample():
    with open("samples.yml", "r", encoding='utf-8') as f:
        return yaml.safe_load(f)['response_sample']

def get_convo_message_pattern():
    with open("pattern.yml", "r", encoding='utf-8') as f:
        return yaml.safe_load(f)['convo_message_pattern']

def get_chat_prompt_pattern():
    with open("pattern.yml", "r", encoding='utf-8') as f:
        return yaml.safe_load(f)['chat_prompt_pattern']

def get_chat_reply_prompt_pattern():
    with open("pattern.yml", "r", encoding='utf-8') as f:
        return yaml.safe_load(f)['chat_reply_prompt_pattern']
```

Modules:

--telegram.py:

```
import os

import aiosqlite
import openai

from typing import List, Tuple

from aiogram.dispatcher import Dispatcher
from aiogram.types import Message, ParseMode, ChatType
from aiogram.dispatcher.filters import ChatTypeFilter, IsReplyFilter

from GuCha.Libs.openai_interact import gpt3_embedding, fetch_memories,
summarize_memories, get_last_messages, \
    gpt3_completion, load_convo_group
from GuCha.Libs.utils import get_db_connection_string, get_openai_key,
get_response_sample, \
    get_chat_prompt_pattern, get_chat_reply_prompt_pattern


class TelegramModule:
    groups = set()
    users = set()

    def __init__(self, dp: Dispatcher):
        self.message = None
        openai.api_key = get_openai_key()
        self.dp = dp

        self.dp.register_message_handler(self.private_handler,
ChatTypeFilter(ChatType.PRIVATE))
        self.dp.register_message_handler(self.reply_handler,
IsReplyFilter(True))
        self.dp.register_message_handler(self.mention_handler)


    async def private_handler(self, message: Message):
        self.message = message
        await self.bot_answer()

    async def reply_handler(self, message: Message):
        if None if message.reply_to_message is None else
message.reply_to_message.from_user.id == (
                await message.bot.get_me()).id:
            self.message = message
            await self.bot_answer()

    async def mention_handler(self, message: Message):
        if message.text and ('@' + (await self.dp.bot.get_me()).username) in
message.text:
            self.message = message
            await self.bot_answer()

    async def bot_answer(self):
        send_message = await self.message.reply(text='Обробка відповіді,
зачекайте!')
        if send_message.from_user.id not in self.users:
            self.users.add(send_message.from_user.id)
            await self.add_user(send_message)
        if self.message.from_user.id not in self.users:
            self.users.add(self.message.from_user.id)
            await self.add_user(self.message)
```

```
        if self.message.chat.id not in self.groups:
            self.groups.add(self.message.chat.id)
            await self.add_group(self.message)

        await self.add_message(self.message)

        #### get user input, save it, vectorize it, etc
        text = self.message.text
        vector = await gpt3_embedding(text)
        conversation = await load_convo_group(self.message)
        memories = await fetch_memories(vector, conversation, 10)
        recent = await get_last_messages(conversation, 10)
        notes = await summarize_memories(memories)
        convo = get_response_sample() \
            .replace('<<USER>>',
                     f"з {self.message.chat.full_name}" if
self.message.chat.title is None else f"в чаті {self.message.chat.title}") \
            .replace('<<NOTES>>', notes) \
            .replace('<<CONVERSATION>>', recent) \
            .replace('<<BOTNAME>>', send_message.from_user.full_name)

        if self.message.reply_to_message is not None:

            messages =
get_chat_reply_prompt_pattern().replace('<<ChatName>>',
                                        self.message.chat.full_name if
self.message.chat.title is None else self.message.chat.title) \
                               .replace('<<Conversation>>', convo) \
                               .replace('<<UserName>>',
self.message.from_user.username) \
                               .replace('<<BotName>>',
send_message.from_user.username) \
                               .replace('<<ReplyUserName>>',
self.message.reply_to_message.from_user.username) \
                               .replace('<<ReplyMessage>>',
self.message.reply_to_message.text) \
                               .replace('<<Message>>', text)
        else:
            messages = get_chat_prompt_pattern().replace('<<ChatName>>',
                                        self.message.chat.full_name if
self.message.chat.title is None else self.message.chat.title) \
                               .replace('<<Conversation>>', convo) \
                               .replace('<<UserName>>',
self.message.from_user.full_name) \
                               .replace('<<BotName>>',
send_message.from_user.full_name) \
                               .replace('<<Message>>', text)

        output = await gpt3_completion(
            model=os.getenv('CHAT_OPENAI_MODEL'),
            messages=eval(messages),
            temperature=float(os.getenv('CHAT_TEMPERATURE')),
            max_tokens=int(os.getenv('CHAT_MAX_TOKENS')),
            top_p=float(os.getenv('CHAT_TOP_P')),
            frequency_penalty=float(os.getenv('CHAT_FREQUENCY_PENALTY')),
            presence_penalty=float(os.getenv('CHAT_PRESENCE_PENALTY')),
            stop=os.getenv('CHAT_STOP')
        )
        await self.dp.bot.edit_message_text(
            text=output,
            chat_id=self.message.chat.id,
            message_id=send_message.message_id,
            parse_mode=ParseMode.MARKDOWN
```

```
        )
        send_message.text = output
        await self.add_message(send_message)

    # add to DB methods
    @staticmethod
    async def add_user_group_link(message):
        async with aiosqlite.connect(get_db_connection_string()) as conn:
            await conn.execute('INSERT OR IGNORE INTO user_group_link
(user_id, chat_id) VALUES (?, ?)',
                               (message.from_user.id, message.chat.id))
            await conn.commit()

    @staticmethod
    async def add_group(message):
        async with aiosqlite.connect(get_db_connection_string()) as conn:
            await conn.execute('INSERT OR IGNORE INTO groups (group_id,
group_name, group_title) VALUES (?, ?, ?)',
                               (message.chat.id, message.chat.username,
message.chat.title))
            await conn.commit()

    @staticmethod
    async def add_user(message):
        async with aiosqlite.connect(get_db_connection_string()) as conn:
            await conn.execute('INSERT OR IGNORE INTO users (user_id,
user_name, user_nick) VALUES (?, ?, ?)',
                               (message.from_user.id,
message.from_user.full_name, message.from_user.username))
            await conn.commit()

    @staticmethod
    async def add_message(message):
        vector = await gpt3_embedding(message.text)
        async with aiosqlite.connect(get_db_connection_string()) as conn:
            await conn.execute('INSERT INTO messages (message_id, user_id,
chat_id, reply_id, time, text, vector) '
                               'VALUES (?, ?, ?, ?, ?, ?, ?)',
                               (message.message_id, message.from_user.id,
message.chat.id,
                                message.reply_to_message.message_id if
message.reply_to_message is not None else None,
                                message.date.timestamp(), message.text,
str(vector)))
            await conn.commit()

    @staticmethod
    async def add_all_users(message):
        async with aiosqlite.connect(get_db_connection_string()) as conn:
            await conn.execute('INSERT OR IGNORE INTO users (user_id,
user_name, user_nick) VALUES (?, ?, ?)',
                               (message.from_user.id,
message.from_user.full_name, message.from_user.username))
            await conn.commit()

    # get from DB methods
    @staticmethod
    async def get_messages_by_user(user_id: int, chat_id: int) ->
List[Tuple[int, str]]:
        async with aiosqlite.connect(get_db_connection_string()) as conn:
            async with conn.execute(
                    'SELECT id, text FROM messages WHERE user_id = ? AND
chat_id = ?',
```

```
                    (user_id, chat_id)) as cursor:
                return await cursor.fetchall()

    @staticmethod
    async def get_messages_in_group(chat_id: int) -> List[Tuple[int, str]]:
        async with aiosqlite.connect(get_db_connection_string()) as conn:
            async with conn.execute('SELECT id, text FROM messages WHERE
chat_id = ?',
                                    (chat_id,)) as cursor:
                return await cursor.fetchall()

    @staticmethod
    async def get_last_reply(chat_id: int, reply_id: int) -> List[Tuple[int,
str]]:
        async with aiosqlite.connect(get_db_connection_string()) as conn:
            async with conn.execute(
                    'SELECT message_id, message_text FROM messages WHERE
chat_id = ? AND reply_id = ?',
                    (chat_id, reply_id)) as cursor:
                return await cursor.fetchall()
```

--main.py:

```
import logging
import asyncio
import openai

from aiogram import Bot, Dispatcher

from Modules.telegram import TelegramModule

from Libs.utils import get_bot_token,
from Libs.create_db import create_databases
from Libs.utils import get_openai_key

logging.basicConfig(level=logging.INFO)

# Initialize the bot and dispatcher
bot = Bot(token=get_bot_token())
dp = Dispatcher(bot)
openai.api_key = get_openai_key()

# Register chat modules
TelegramModule(dp)


async def main():
    # Creating db
    await create_databases()
    # Bot Poling loop
    await dp.start_polling()


if __name__ == '__main__':

    # Start the bot
    logging.info('Starting bot...')
    asyncio.run(main())
```

**Додаток В**

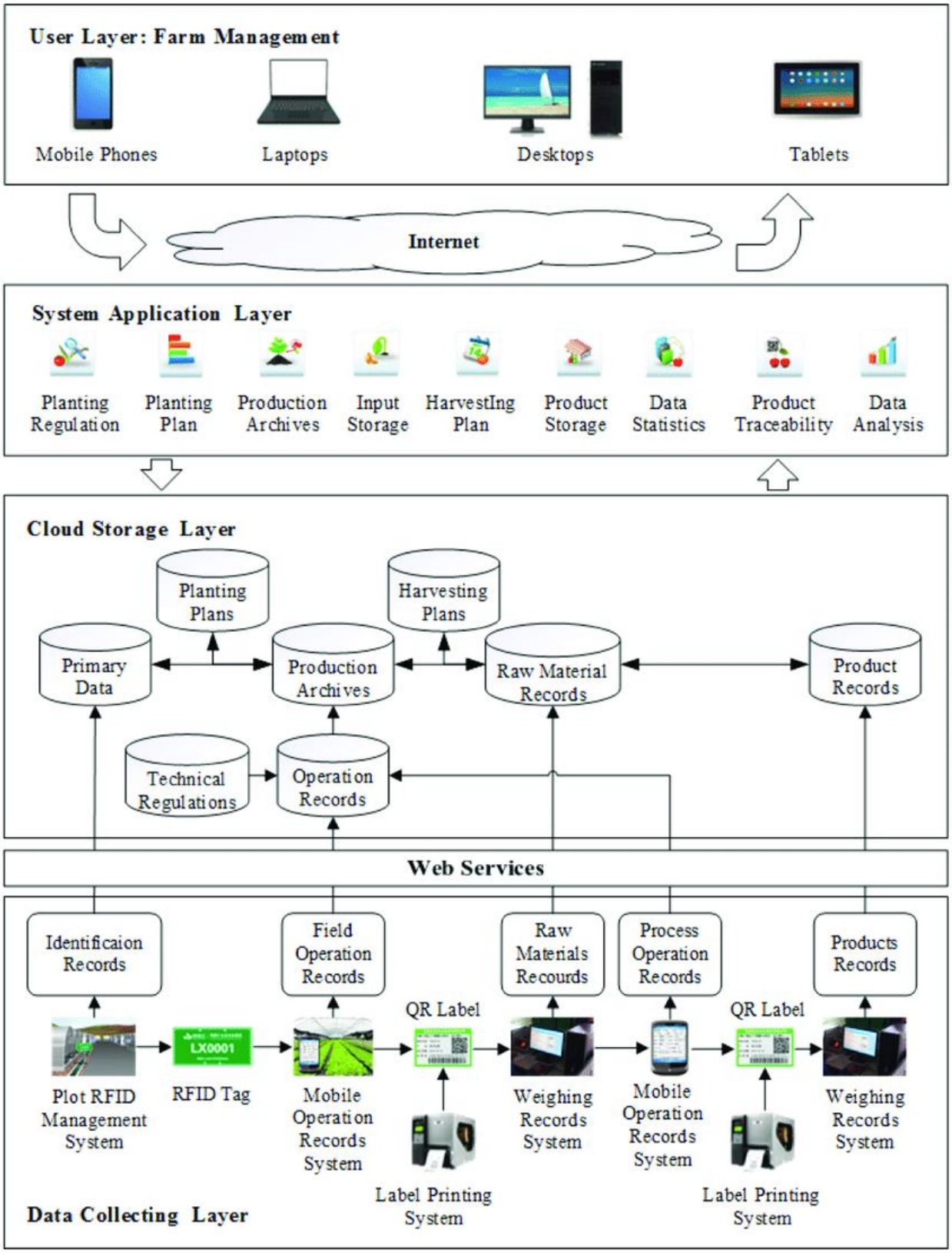



Рисунок 1 – Поширена структура інформаційної системи управління фермою [21]

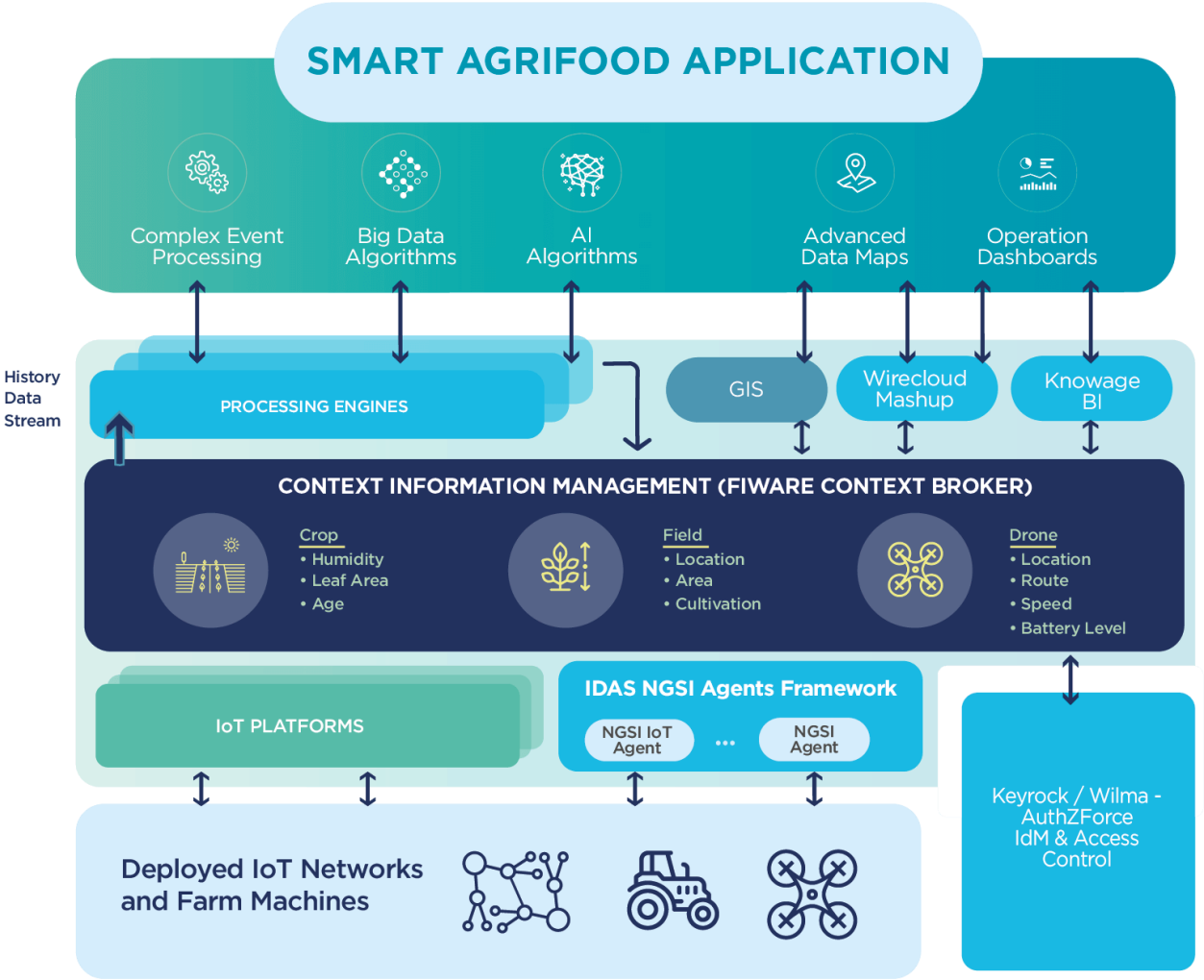



Рисунок 2 – Еталонна архітектура для рішень в сільському господарстві з використанням AI Algorithms [22]