Вінницький національний технічний університет
(повне найменування вищого навчального закладу)
Факультет інтелектуальних інформаційних технологій та автоматики
(повне найменування інституту, назва факультету (відділення))
Кафедра комп'ютерних наук
(повна назва кафедри (предметної, циклової комісії))

Бакалаврська кваліфікаційна робота на тему:

РОЗРОБКА ЧАТ-БОТУ ДЛЯ УПРАВЛІННЯ ПЕРСОНАЛОМ

Виконав: студент 4 курсу, групи 2КН-21б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Вінтонюк В.В.
(прізвище та ініціали)
Керівник: к.т.н., доцент кафедри КН
Озеранський В.С.
(прізвище та ініціали)
«04» 06 2025 р.
Рецензент: к.т.н., доцент кафедри САІТ
Варчук І.В.
(прізвище та ініціали)
«05» 06 2025 р.

Допущено до захисту
Завідувач кафедри КН
д.т.н., проф. Яровий А.А.
(прізвище та ініціали)
«06» 06 2025 р.

Вінниця ВНТУ – 2025 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра Комп'ютерних наук
Рівень вищої освіти перший (бакалаврський)
Галузь знань – 12 «Інформаційні технології»
Спеціальність – 122 «Комп'ютерні науки»
Освітньо-професійна програма – «Комп'ютерні науки»

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
проф., д.т.н. Яровий А.А.
"05" 05 2025 року

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Вінтонюку Владиславу Вікторовичу

1. Тема роботи: Розробка чат-боту для управління персоналом.
керівник роботи: Озеранський Володимир Сергійович, к.т.н., доцент
затверджені наказом вищого навчального закладу "20" 03 2025 року №97

2. Строк подання студентом роботи 30.05.2025р.

3. Вихідні дані до роботи:
склад команди – до 8 чол;
тривалість планування завдань – до 8 тижнів;
Мова програмування – об'єктно-орієнтована;

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):
вступ, обгрунтування доцільності розробки чат-боту для управління персоналом; розробка чат-боту для управління персоналом; програмна реалізація розробки чат-боту для управління персоналом; список використаних джерел; додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень):
схема загального алгоритму функціонування чат-боту для управління персоналом; структура чат-боту для управління персоналом; загальна UML-діаграма класів.

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-3	Озеранський В.С., к.т.н., доцент		

6. Дата видачі завдання 05 травня 2025 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1.	Обгрунтування доцільності розробки чат-боту для управління персоналом	05.05.25	02.05.25	
2.	Розробка чат-боту для управління персоналом	08.05.25	11.05.25	
3.	Програмна реалізація розробки чат-боту для управління персоналом	12.05.25	15.05.25	
4.	Розробка інструкції користувача.	16.05.25	26.05.25	
5.	Висновки та аналіз отриманих результатів	22.05.25	01.06.25	
6.	Оформлення матеріалів до захисту БКР	02.06.25	04.06.25	

Студент _____ Вінтонюк В.І
(підпис) (прізвище та ініц

Керівник проєкту (роботи) _____ Озеранський
(підпис) (прізвище та ініц

АНОТАЦІЯ

Вінтонюк В.В. Розробка чат-боту для управління персоналом. Бакалаврська кваліфікаційна робота складається з 70 сторінок формату А4, на яких є 10 рисунок, 4 таблиці, список використаних джерел містить 19 найменуваннь.

В роботі обґрунтовано доцільність розробки чат-боту для управління персоналом. Розроблено структуру чат-боту для управління персоналом. Визначено основні модулі зі сторони користувача. Створено діаграму станів програмного продукту для управління персоналом. Створено діаграму активності\діяльності чат-боту для управління персоналом.

Обгрунтовано вибір мови програмування та середовища розробки Здійснено програмну реалізацію чат-бота для управління персоналом з використанням мови Python у середовищі Visual Studio Code. Проведено тестування розробленого чат-боту для управління персоналом.

Ключові слова: чат-бот, Telegram, Visual Studio Code, Python, управління персоналом, Kanban.

ABSTRACT

Vintonyuk V. Development of a Chatbot for Human Resource Management. The bachelor's qualification thesis consists of 70 A4 pages, including 10 figures, 4 tables, and a list of references containing 19 sources.

The paper justifies the feasibility of developing a chatbot for human resource management. The structure of the chatbot for HR management has been developed. The main user-side modules have been identified. A state diagram of the software product for HR management has been created. An activity diagram of the chatbot for HR management has also been created.

The choice of programming language and development environment is substantiated. The chatbot for HR management has been implemented using the Python programming language in the Visual Studio Code environment. Testing of the developed chatbot has been conducted.

Keywords: chatbot, Telegram, Visual Studio Code, Python, human resource management, Kanban.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному швидкотемповому світі, де ефективне управління персоналом стає ключовим фактором успіху, використання новітніх технологій і інструментів стає необхідністю. Одним з перспективних напрямків у цій сфері є розробка телеграм-ботів для менеджменту персоналу.

Дана робота присвячена розробці чат-боту для управління персоналом (HR-бот). Актуальність даного дослідження полягає в тому, що зростаючий обсяг роботи та потреба в організації та контролі завдань ставлять перед управлінцями та командами вимоги щодо швидкості, точності та зручності взаємодії. Сучасні реалії бізнесу, включаючи розширення гібридних та віддалених форматів роботи, посилюють потребу в ефективних та доступних інструментах для комунікації та автоматизації рутинних HR-процесів [1].

Аналіз предметної області дозволить виявити сучасні тенденції та найкращі практики в управління персоналом, а також визначити потреби та очікування користувачів. Проектування структури телеграм-боту допоможе вирішити проблеми з організацією завдань для персоналу та забезпечити зручну інтерфейсу для користувачів. Реалізація програмного забезпечення телеграм-боту надасть практичний інструмент для управління персоналом з використанням сучасних технологій.

Розробка чат-боту для управління персоналом є не просто інноваційним трендом, а стратегічною необхідністю для сучасних організацій, які прагнуть підвищити ефективність, оптимізувати витрати та створити комфортне та продуктивне середовище для своїх співробітників.

Отже, актуальність даного дослідження полягає у впровадженні інноваційних технологій та інструментів у сферу менеджменту персоналу. Телеграм-боти виявляються потужними інструментами для організації та контролю завдань, забезпечуючи зручну комунікацію та доступ до необхідної інформації. Вони можуть сприяти підвищенню ефективності, скороченню часу виконання завдань та поліпшенню спільної роботи команди.

Проектування структури та реалізація програмного забезпечення телеграм-боту управління персоналом дозволять вирішити проблеми, пов'язані з недостатньою організацією, втратою інформації та незручними інтерфейсами. Розробка функціонального та ефективного телеграм-боту допоможе підвищити продуктивність та забезпечити зручну роботу, пов’язану з управлінням персоналом.У зв’язку з цим дане бакалаврське дослідження є досить актуальним.

Метою бакалаврської кваліфікаційної роботи є розширення функціонльних можливостей програмного забезпечення для управління персоналом.

Об’єктом дослідження є процес управління персоналом.

Предметом дослідження є програмні засоби для організації процесу управління персоналом.

Для досягнення поставленої мети необхідно виконати такі задачі:

- обґрунтувати доцільність розробки чат-боту для управління персоналом;
- проаналізувати переваги та недоліки програм-аналогів для організації управління персоналом;
- розробити алгоритм організації управління персоналом
- здійснити програмну реалізацію чат-боту для управління персоналом;
- виконати тестування чат-боту для управління персоналом та провести аналіз отриманих результатів.

1 ОБГРУНТУВАННЯ ДОЦІЛЬНОСТІ РОЗРОБКИ ЧАТ-БОТУ ДЛЯ УПРАВЛІННЯ ПЕРСОНАЛОМ

1.1 Аналіз існуючих систем менеджменту завдань для персоналу

Аналіз існуючих систем менеджменту завдань може допомогти визначити їх основні характеристики і вибрати найбільш підходящу для конкретних потреб. Ось опис чотирьох популярних систем менеджменту завдань:

Trello: Trello є веб-орієнтованою системою, яка базується на дошках та картках. Користувачі можуть створювати списки завдань та переміщати картки між ними, надавати пріоритети, встановлювати дедлайни, додавати коментарі та відстежувати виконання завдань [1].

Asana: Asana – це комплексна система, що надає можливість створювати проекти, завдання, підзавдання, делегувати їх та встановлювати терміни. Вона також дозволяє надсилати сповіщення, відстежувати прогрес та спілкуватися з командою.

Todoist: Todoist – це проста, але потужна система менеджменту завдань. Вона дозволяє створювати завдання, надавати пріоритети, встановлювати терміни, створювати проектні шаблони та спілкуватися з іншими користувачами. Todoist також пропонує графічний інтерфейс та інтеграцію з іншими популярними додатками [1].

Microsoft To Do: Microsoft To Do є простою системою, що надає можливість створювати персональні списки завдань, встановлювати терміни, надавати пріоритети та використовувати геолокацію для нагадувань. Вона інтегрується зі службами Microsoft, такими як Outlook та Microsoft Teams [2].

Проведемо порівняльний аналіз існуючих комп’ютерних систем, і сформуємо таблицю, де буде видно їх переваги та недоліки.

Таблиця 1.1 – Порівняльний аналіз систем менеджменту завдань

Система	**Основні функції**	**Інтерфейс**	**Інтеграція**	**Переваги**	**Недоліки**
Trello	Дошки, картки, переміщення завдань	Зручний та інтуїтивний	Slack, Google Drive, Dropbox, Jira та багато інших	Простий та зрозумілий інтерфейс	Обмежена функціональність без платної версії
				Великий вибір інтеграцій з популярними інструментами	Відсутність розширених функцій для проектного управління
Asana	Створення проектів, завдань, делегування	Стильний та організований	Slack, Google Calendar, Outlook, Salesforce тощо	Комплексні функціональні можливості	Складність в орієнтації нових користувачів
				Зручний інтерфейс з легким доступом до інформації	Висока вартість для команди
Todoist	Списки завдань, пріоритети, спілкування	Простий та зрозумілий	Gmail, Google Calendar, Slack, Amazon Alexa тощо	Простота використання та швидкість реагування	Обмежена можливість редагування завдань
				Гнучкість установки термінів та повідомлень	Відсутність розширених функцій для проектного управління
Microsoft To Do	Списки завдань, терміни, геолокація	Інтуїтивний та зручний	Outlook, Microsoft Teams, OneNote, Cortana тощо	Інтеграція з екосистемою Microsoft	Обмежена функціональність порівняно з іншими

1.2 Аналіз сучасних методологій управління персоналом

Сучасні методології менеджменту завдань для персоналу є набором підходів та інструментів, спрямованих на краще планування, організацію та виконання завдань в командній роботі. Ось кілька основних сучасних методологій менеджменту завдань:

Scrum: Scrum є популярним підходом у розробці програмного забезпечення, але його принципи можна застосувати і в інших сферах. Він базується на ітеративному та інкрементальному плануванні, де робочі завдання розбиваються на короткі спроможність об'єкта роботи (Sprint) тривалістю від одного до чотирьох тижнів. Команда регулярно зустрічається для планування, оцінювання та оновлення завдань [1].

Канбан: Канбан - це система візуалізації та керування робочим процесом. Завдання відображаються на дошці з трьома стовпцями: "У черзі", "В роботі" і "Завершено". Команда пересуває картки з завданнями відповідно до їхнього стану. Канбан сприяє кращому управлінню робочим потоком, інформаційної прозорості та зменшенню перевантаження [2].

OKR (Objective and Key Results): OKR - це методологія, що визначає об'єктиви та ключові результати для оцінки успішності. Об'єктиви визначаються вищим рівнем (наприклад, компанією), а потім переносяться на нижчі рівні. Ключові результати вимірюються за допомогою конкретних метрик і оцінюються регулярно. OKR сприяє узгодженості цілей, фокусу та зростанню результативності [3].

Lean-менеджмент: Lean-менеджмент, оснований на концепціях Toyota Production System, продовжує методологію менеджменту завдань, спрямовану на досягнення оптимальної ефективності та усунення зайвих втрат у процесах. В основі Lean-менеджменту лежить поняття "мудрість виробництва" (wisdom of the production), що орієнтує на використання ресурсів та робочого часу ефективним способом [3].

Agile: Agile - це широкий набір підходів до управління проектами, які покладають акцент на гнучкість, самоорганізацію та ітеративність. Agile

орієнтований на швидку адаптацію до змін і невпинну комунікацію у команді. Популярні підходи Agile включають Scrum, Kanban та Extreme Programming (XP) [3].

GTD (Getting Things Done): GTD - це методологія, розроблена Девідом Алленом, яка надає систематичний підхід до організації завдань та керування ними. Головна ідея GTD полягає в тому, щоб зберігати всі завдання та ідеї в систематизованій структурі, виконувати їх у встановлений час і зосереджуватися на пріоритетних діях [3].

Ці методології менеджменту завдань надають структуру, розуміння та організацію для кращого керування завданнями у командному середовищі. Вибір конкретної методології залежить від конкретних потреб команди, проекту та організації.

1.3 Огляд методології управління персоналом Канбан

Акцент на забезпеченні бізнес-цінності був однією з провідних рушійних сил впровадження найбільш гнучких підходів до розробки програмного забезпечення. Ця мета також спонукала до запровадження економного мислення в процесах розробки програмного забезпечення з усуненням відходів як основним принципом – разом із постійним навчанням через короткі цикли і часті збірки, а також просування пізніх змін і швидких ітерацій (Poppendieck and Cusumano 2012) [4].

Ці рухи відбулися в контексті переходу бізнесу до епохи цифрової трансформації, у якій підривні бізнес-процеси та моделі розглядаються як необхідні шляхи підвищення конкурентоспроможності, головним чином для тих компаній, які не бажають надавати значний контроль над своїми процесами. великим постачальникам програмного забезпечення [4]. Разом із технологічною еволюцією (наприклад, хмарними обчисленнями) це змусило індустрію програмного забезпечення згадати принципи програмування в малому [4] (DeRemer and Kron) і переглянути надзвичайну технічну складність і негнучкість величезних, стандартизованих систем програмного забезпечення та процеси (Andriole) [5]. Отже,

необхідно усунути марнотратство з непотрібною складністю або сприяти пізнім змінам, щоб підтримувати програмні системи в курсі змін бізнес-процесів.

Незважаючи на своє походження з виробництва, принципи бережливого виробництва постійно досліджуються в нових галузях, навіть серед тих, що передбачають інтенсивну роботу знань, як у випадку з розробкою програмного забезпечення (SE). Стаатс та ін. [5] стверджують, що «розробка знань не тільки має контекст, окремий від виробництва, але й принципово відрізняється за структурою, що ставить під сумнів універсальну застосовність принципів економії». З одного боку, гнучкі підходи до розробки програмного забезпечення успішно досягли таких цілей, як адаптивність та ітераційні процеси [5], хоча вони були дуже орієнтованими на розробника та відносно непрозорими для керівництва щодо оцінки зусиль, тривалості та витрат на розробку [6]. З іншого боку, ощадливі підходи більше орієнтовані на кількісне вимірювання та прийняття рішень на основі доказів [6].

Одним із провідних економних підходів, що використовуються в SE, є Kanban, який все частіше застосовують організації, що займаються програмним забезпеченням [6]. Зважаючи на зростання популярності цієї теми, дослідники посилюють свою увагу до цієї теми, як можна побачити в чотирьох вторинних дослідженнях, які аналізують різні точки зору щодо канбану, що охоплює понад 20 первинних досліджень. Технічна література містить досить вичерпні звітні докази щодо переваг, очікуваних від Kanban, і проблем, пов’язаних з його використанням [7].

Однак, незважаючи на значні зусилля з організації сукупності знань, як це спостерігалося в цих чотирьох вторинних дослідженнях (систематичні огляди та відображення), все ще бракує синтезу переваг і проблем Канбану. Синтез досліджень є важливим для забезпечення узагальнення, інтеграції, поєднання та порівняння результатів різних досліджень. Вони запропоновані на основі того, що окремі дослідження обмежені в тому, якою мірою вони можуть бути узагальненими [7]. Таким чином, синтез досліджень є життєво важливим інструментом знань, який використовується для управління науковими висновками та використання їх [7].

У сучасну епоху гнучкі методи є широко прийнятими підходами в різних організаціях завдяки їхнім численним перевагам у порівнянні з раніше використовуваними традиційними методами [8]. Загальне значення Agile полягає в тому, щоб «рухатися швидко та легко», таким чином використання методів Agile полегшує роботу, а проекти постачаються в серіалізованому процесі, а не доставляються після завершення [8]. Опитування Глобального управління проектами (2018) показує, що 71% організацій-учасниць повідомляють про більшу гнучкість за останні п’ять років, що означає, що гнучкість визнана тим, що допомагає їм залишатися конкурентоспроможними. Крім того, 12-те щорічне опитування Agile показує, що більшість організацій (97%) практикують методи Agile [8].

Scrum і Kanban вважаються двома потужними гнучкими методами, які обробляють і керують прогресом розробки програмного забезпечення [8]. Ці два методи впливають на різних членів команди Agile та проекти в різних ситуаціях, оскільки вони можуть оптимізувати налаштування команд, визначити їхні завдання та ефективно керувати часом розробки [9]. Згідно зі звітом Version One (2018), 56% респондентів практикують Scrum порівняно з іншими методами Agile. Однак різні дослідження, проведені в [9] підтверджують, що метод Kanban досяг своєї популярності в останні роки, оскільки він містить численні переваги, які допомагають покращити процес управління проектами.

Незважаючи на широке визнання в організаціях, що займаються програмним забезпеченням, метод Agile Kanban все ще має різні проблеми, з якими стикаються практики програмного забезпечення під час впровадження цього методу. Щоб розв’язати цю ситуацію, у цьому дослідженні використовувався метод наративного огляду для вивчення відповідної літератури про метод Agile Kanban. Наративний огляд — це суворий метод систематичного огляду, який синтезує літературу, отриману з різних наукових джерел, щоб дослідити конкретну тему [10]. Крім того, перехресний аналіз використовувався для порівняння результатів отриманих досліджень.

Концепція Kanban була введена в Toyota в 1947 році, в якій цей термін походить від японського терміна, що позначає «вивіска». Це візуальна система управління процесами, яка може керувати знаннями та роботою, враховуючи підхід доставки «точно вчасно» (JIT). JIT не перевантажує членів команди, оскільки метод Kanban зосереджений на усуненні вузьких місць і відходів, а також на скороченні часу очікування, що призводить до збільшення обсягу пропускної здатності [10].

У 2004 році Девід Дж. Андерсон передбачив Канбан як метод розробки програмного забезпечення. Його вважають батьком канбану та одним із провідних лідерів руху. Він описав Канбан як «підхід до поступового, еволюційного процесу та системних змін для організацій» [10]. Метод Agile Kanban може покращити розуміння, видимість і контроль робочого процесу. Це може допомогти керівництву шляхом виявлення вузьких місць у процесі розробки проектів програмного забезпечення. Цей метод використовує систему вилучення як основний підхід до виявлення проблем системного процесу та для стимулювання співпраці для постійного вдосконалення системи [10].

Гнучкий Канбан є ключовим методом, який використовується для управління робочим процесом і контролю за втратами при розробці проектів програмного забезпечення [10]. Роботи за традиційними методами виглядають як ланцюжок, у якому робота одного учасника передається іншому, що може спричинити затримки

Прийняття методу Agile Kanban різко зросло протягом останніх трьох років (2015-2017) з 39% до 65% [10]. Цей метод набуває все більшої популярності та уваги серед практиків програмного забезпечення завдяки численним перевагам, які роблять його кращим за інші методи з точки зору управління процесом розробки програмного забезпечення [11]. Тим не менш, деякі дослідження стверджують, що метод Канбан все ще має кілька проблем, що робить його дуже актуальною та вартою вивчення темою дослідників і практиків протягом останніх років, як обговорюється в наступному розділі. Цей розділ продовжується представленням принципів і практик канбану, далі описується дошка канбану, а завершується висвітленням переваг методу канбан.

Андерсон [11] визначив п’ять принципів методу Agile Kanban, а саме:

a) обмеження незавершеної роботи (WIP);

b) візуалізація робочого процесу;

c) вимірювання та управління потоком;

d) чітке визначення політики процесів;

e) використання моделей для визначення можливостей покращення.

Коротко опишимо ці принципи.

a) Обмеження WIP: це основний принцип методу Agile Kanban, який визначається як максимальна кількість завдань для кожного етапу дошки Kanban. Його зазвичай визначає керівник проекту, щоб запобігти перешкодам і прискорити виконання завдань на дошці.

b) Візуалізація робочого процесу: це ще один основний принцип методу Канбан, який визначається як процес виділення механізмів, взаємодії, очікування, черг і затримок, які задіяні в реалізації частини цінного програмного забезпечення.

c) Вимірювання та управління потоком: цей принцип вимірювання та управління потоком підкреслює фокус не лише на підтримці роботи чи завдання, але також на необхідності використання потоку як рушійної сили для вдосконалення. Майстерінг — це зосередження на потоці, а не на видаленні відходів.

d) Чітке визначення політики процесу: цей принцип відображає ефективність і реальність роботи, яка має бути чітко визначена, щоб заохочувати всіх членів команди. Це важливо для того, щоб усі члени команди могли розглядати процес розробки як групу політик, а не розглядати робочі процеси як обмежену техніку.

e) Використання моделей для розпізнавання можливостей покращення: метод Канбан використовує кількісний науковий підхід до створення поліпшень. Метод орієнтований на моделі, які схиляються до управління відходами та контролю над потоком, враховуючи Теорію обмежень (TOC). Метод також намагається зрозуміти систему та варіації глибоких знань.

Окрім принципів, метод Agile Kanban містить чотири практики, а саме: (a) починати з того, що є в наявності, (б) погоджуватися на поступові та еволюційні зміни, (в) поважати існуючі ролі, процеси та обов’язки та (d) заохочувати лідерські дії протягом усього процесу розвитку [11].

Dingsøyr та ін. [11] також визнав, що метод Agile Kanban суттєво відрізняється від інших методів, оскільки він починається з того, де знаходиться організація, і не вимагає створення нових ролей, церемоній або структур перед початком роботи. Однак обмеження WIP і візуалізація робочого процесу є двома основними принципами методу Agile Kanban, які використовуються для моніторингу прогресу проекту [11].

Метод Agile Kanban має дошку, яка використовується для візуалізації робочого процесу та моніторингу прогресу проекту, показуючи дії процесу розробки та зберігаючи контроль над WIP [11]. Дошка Kanban також надає розробникам можливість зосередитися на кількох завданнях. Таким чином, витрати ресурсів і часу будуть зменшені через процес перемикання між завданнями на дошці Kanban [12]. Дошка Kanban є ключовим аспектом методу Agile Kanban, оскільки процес розробки можна відстежувати [12].

Як правило, дошка Kanban вертикально розділена на різні стовпці або етапи. Кожен етап відноситься до стану завдання, тоді як кожне завдання представлено карткою, прикріпленою на дошці на етапі, яка відображає поточний стан завдань. На дошці Kanban картки переміщуються зліва направо відповідно до змін стану завдання [12].

Дошка Kanban має два типи: прості та детальні дошки. Зазвичай керівник проекту визначає тип дошки Kanban, яка буде простою або детальною дошкою, на основі ряду критеріїв, таких як розмір проекту, кількість завдань і кількість членів команди [12]. На рисунку 1.1 показана проста дошка Kanban.

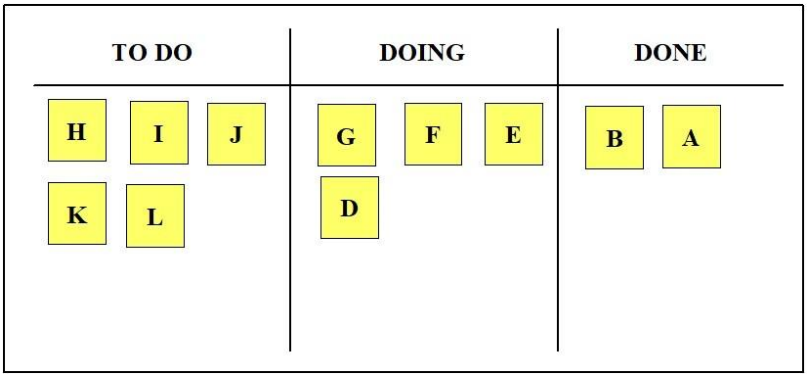

Рисунок 1.1 – Проста канбан-дошка

Однак детальна дошка Kanban може мати різні етапи. Наприклад, Backlog, Analysis, Development, Test і Deployment або Done є етапами детальної дошки Kanban, як показано на рисунку 1.2.

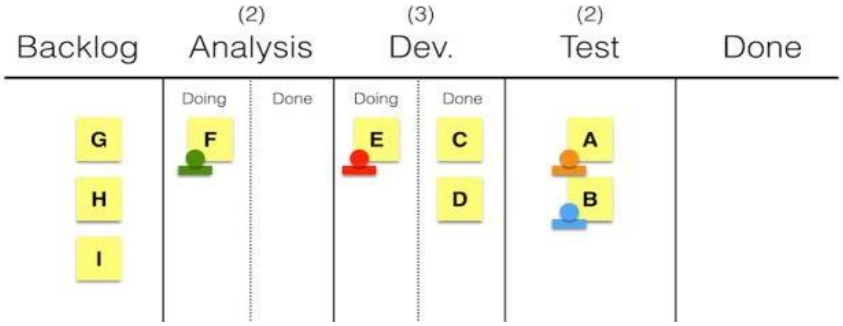

Рисунок 1.2 – Деталізована канбан-дошка

За словами Філбіна [12], існує тенденція до прийняття дошки Kanban в організаціях з розробки програмного забезпечення (SDO) для підтримки прозорих комунікацій між усіма членами команди. Це впровадження полегшує процес розробки проектів програмного забезпечення, оскільки його легше зрозуміти та дотримуватися [12]. Крім того, це дозволяє членам команди відстежувати прогрес проектів за допомогою властивості обмеження WIP для кожного етапу [13].

Проте визначення кількості лімітів WIP для кожного етапу на дошці Kanban є основною проблемою, з якою стикаються практики програмного забезпечення. Тому є більша потреба виконання завдань проекту порівняно з попереднім, у якому це не було видно всім учасникам. Крім того, дошка Kanban може продемонструвати успіх зацікавлених сторін, за допомогою чого вони можуть бачити, розуміти та контролювати статус прогресу проекту [13].

Тим не менш, візуалізація дошки Kanban обмежена показом діяльності процесу розробки, а не наданням достатньої інформації чи корисних вказівок, які могли б допомогти в моніторингу прогресу проекту [14]. Таким чином, існує нагальна потреба визначити альтернативні критерії візуалізації для вдосконалення методу Kanban, який може надати корисну інформацію у звітах про статус проектів.

Ахмад та ін. [14] зазначив, що цей метод Kanban забезпечує кращу видимість і розуміння всього процесу розробки, а також ефективний контроль робочих процесів

і обмежень WIP. Якуб та ін. [14] також визнав, що Agile Kanban є гнучким, оперативним і надійним методом. Крім того, Мірза та Датта [14] заявили, що метод Agile Kanban може покращити комунікацію та прозорість роботи, задоволеність клієнтів та координацію команд між різними зацікавленими сторонами. Крім того, Ikonen et al. [14] стверджували, що метод Канбан не тільки допомагає мотивувати членів команди, але й підтримує керівників проектів у моніторингу всіх проектних дій під час процесу розробки проектів програмного забезпечення.

Порівнюючи зі Scrum, Ahmad et al. [14] підтверджують, що впровадження методу Kanban постійно зростає в SDO, оскільки він працює краще, ніж Scrum та інші гнучкі методи. Привабливість Kanban також підтверджують Shafiq et al. [15], які вказують на те, що практики Agile наразі переходять на метод Kanban через його Крім того, замість використання ітерацій, як у Scrum, метод Kanban максимізує робочий процес, мінімізує час виконання та безперервно доставляє програмне забезпечення. Дослідження, проведене bb Sjøberg et al. [15], щоб порівняти використання методів Scrum і Kanban протягом двох років. Результати показали, що заміна Scrum на метод Kanban в SDO може скоротити час очікування на 50%, мінімізувати кількість помилок на 11% і підвищити продуктивність програмного забезпечення на 21%. Вони також повідомили, що метод Kanban не тільки здатний видаляти періодичні відходи в результаті жорсткості Scrum-timeboxing і Scrum-зустрічей, але також здатний підтримувати всі переваги методів Agile.

Дивно, але канбан виявився набагато ефективнішим за метод Scrum. Лей та ін. [15] провели дослідження, щоб дослідити відмінності між методами Scrum і Kanban через групу з шести факторів, які впливають на процес розробки проектів програмного забезпечення. Статистично результати показали, що немає принципової різниці між цими двома методами щодо шести факторів. Однак численні переваги дозволяють Kanban працювати краще, ніж метод Scrum, в управлінні проектами програмного забезпечення. У проектах, які застосовують метод Канбан, також можна досягти більшої послідовності в управлінні проектами.

З наведеного вище обговорення та порівняння можна зробити висновок, що Kanban є найкращим методом серед інших методів Agile, оскільки він охоплює різні

характеристики, зокрема з точки зору управління проектами програмного забезпечення. Незважаючи на ряд переваг, метод Agile Kanban все ще має проблеми, які вимагають подальших досліджень, як пояснюється далі.

У недавньому дослідженні [15], проведеному Заятом і Сенваром, підтверджується, що Agile Kanban є простим методом, який може бути досить ефективним, якщо його доповнити іншими методами вдосконалення. Так само різні дослідження стверджують, що цей метод необхідно доповнювати іншими методами для забезпечення високої продуктивності серед членів команди, оскільки це відносно базовий метод.

Зважаючи на це, було проведено ряд досліджень, щоб вирішити цю проблему шляхом поєднання канбану з іншими методами. Наприклад, новий метод, відомий як Scrumban, був створений шляхом інтеграції методів Kanban і Scrum. Проте Скрамбан все ще обіймає обмеження методу Kanban, такі як складність встановлення обмежень WIP та відсутність візуалізації корисної інформації щодо прогресу проектів.

Більше того, метод Kanban було поєднано з Value Stream Mapping (VSM) [15]. Однак ця комбінація зосереджена лише на вдосконаленні деяких областей VSM, а не Kanban, для яких дійсно потрібен допоміжний метод. Крім того, методи Scrum, Kanban і Lean були інтегровані, щоб подолати їхні слабкі сторони та об’єднати сильні сторони. Така інтеграція відома як L-ScrumBan з рівнем перевірки 93%, що підкреслює ефективність. Незважаючи на це, L-ScrumBan не зосереджується й не описує встановлення обмежень WIP на дошці, що є основним принципом методу Канбан.

Як показано вище, зрозуміло, що попередні дослідження, проведені для вирішення вищезгаданої проблеми, все ще мають власні обмеження на додаток до ключових проблем методу Agile Kanban, які все ще існують. Тому існує вкрай необхідний пошук відповідного методу для підтримки ефективності Kanban. Alaidaros і Omar [15] розглянули підходи, що використовуються для управління проектами програмного забезпечення, а також моніторингу їх прогресу. Автори запропонували інтегрувати Kanban з методом Earned Value Analysis (EVA), оскільки

перший є ефективним і найбільш широко використовуваним методом моніторингу прогресу в розробці проектів програмного забезпечення. Отже, інтеграція Agile Kanban з методом EVA може максимально використати обидва, враховуючи їхні обмеження.

Аль-Бейк і Міллер [12] і Флора і Чанде відзначили, що використання методу Agile Kanban у сфері розробки програмного забезпечення все ще знаходиться на початковій стадії. Таким чином, стандартне визначення розробки програмного забезпечення та його конкретні практики ще чітко не визначені. Тим не менш, це твердження нещодавно було підтверджено Ahmad et al. [12] та Мірза та Датта, які вказують на те, що існує дуже обмежена кількість досліджень, які можуть надати вказівки щодо реалізації Kanban для практиків програмного забезпечення. Таким чином, необхідні подальші дослідження, щоб зосередитися на вивченні Agile Kanban, оскільки це гідний метод для дослідження.

Інші труднощі, пов’язані із застосуванням методу Канбан, включають організаційну культуру, відсутність спеціальних навичок і навчання, прив’язаність до знайомих методів і мотивацію команди.

Таким чином, метод Канбан підходить лише для команд, у яких є учасники з навичками, що перетинаються, завдяки чому кожен може взяти участь і допомогти зменшити список відставання до нуля. Тобто, якщо лише один із членів команди володіє певним затребуваним навиком, то людина може витримати все.

Дослідження, проведене Флорою та Чанде [13], виявило два обмеження методу Канбан. По-перше, невеликий збій у процесі розробки методу Kanban може призвести до зупинки всього процесу, і для виконання процесу відновлення знадобляться додаткові зусилля. По-друге, замість того, щоб керувати пропускною здатністю методу Kanban, вона генерується через контроль WIP і знання часу циклу. Однак встановлення лімітів WIP є серйозною проблемою, з якою стикаються фахівці-практики програмного забезпечення, оскільки ліміти потрібно час від часу коригувати. Цей виклик підтверджено дослідженням, проведеним Alaidaros et al. [14], які стверджують, що складність визначення обмежень WIP у методі Kanban призводить до відставання в плануванні процесу розробки проектів програмного

забезпечення. Отже, автори стверджують, що визначають критерії, які впливають на встановлення лімітів WIP, щоб створити оптимальну кількість цих лімітів.

Крім того, метод Канбан має певні недоліки, зокрема пов’язані з процесом візуалізації завдань за допомогою класичної дошки Kanban. Хоча ця методологія широко використовується для організації роботи та підвищення прозорості процесів, вона не завжди забезпечує повноцінне розуміння стану виконання проекту.

Дослідження, проведене Аурішем та співавт. [14], підтверджує, що традиційна Kanban-дошка часто виявляється недостатньо ефективною при візуалізації поточного статусу великого або комплексного проекту. Проблеми виникають через обмеженість функціональності: така дошка не здатна чітко відображати реальний прогрес щодо кінцевих результатів, ступінь завершення окремих етапів чи задач, а також взаємозв’язки між ними.

В умовах сучасного, динамічного робочого середовища або при керуванні кількома проєктами одночасно, стандартна візуалізація завдань на Kanban-дошці, попри свою простоту та ефективність для базового відстеження, не надає повної та вичерпної картини. Її можливості вичерпуються там, де починається потреба в глибинному аналізі та автоматизованому контролі.

Традиційна Kanban-дошка, за своєю суттю, є візуальним інструментом для управління робочим потоком. Вона чудово показує статус завдань ("До виконання", "У роботі", "Виконано") і допомагає обмежити незавершену роботу (WIP). Однак, коли ситуація виходить за рамки простого переміщення карток, її обмеження стають очевидними:

- відсутність аналітики та прогнозів. Kanban-дошка не забезпечує вбудованої аналітики чи автоматизованого відстеження критичних точок або затримок;

- складність оцінки прогресу та темпу. Навіть у базовому сенсі візуалізація через дошку не дає чіткої індикації прогресу в часі;

- недостатня інформація для управлінських рішень. У динамічному середовищі менеджерам потрібні дані для оперативного прийняття управлінських рішень на основі поточної ситуації;

- відсутність контексту та інтеграції. Картки на дошці часто містять лише базову інформацію;

- обмежені можливості звітності. Створення детальних звітів про продуктивність команди, ефективність робочого процесу або виконання бюджету на основі простої Kanban-дошки практично неможливе.

Хоча Kanban залишається зручним інструментом для організації поточної роботи, його традиційна реалізація має свої недоліки, які слід враховувати при масштабуванні методології або її використанні в складних організаційних умовах. Така інформація вкрай необхідна керівникам проектів, щоб допомогти їм у прийнятті значущих рішень щодо прогресу розробки проекту. У зв'язку з цим Alaidaros et al. [14] стверджують про проблему методу Канбан, яка полягає у відображенні недостатньої інформації через панель Канбан, що перешкоджало б ефективному управлінню розробкою програмного проекту. Таким чином, автори стверджують, що визначають альтернативні критерії візуалізації, щоб підвищити ефективність аспекту візуалізації методу Agile Kanban.

1.4 Висновок до розділу 1

У даному розділі було проведено огляд різних методів та систем, що застосовуються для організації управління завданнями для персоналу. Цей аналіз дозволив отримати уявлення про різноманітність інструментів та підходів, які застосовуються в сучасних організаціях для гнучкого управління персоналом.

2 РОЗРОБКА СТРУКТУРИ ЧАТ-БОТУ ДЛЯ УПРАВЛІННЯ ПЕРСОНАЛОМ

2.1 Аналіз сучасних технологій створення чат-ботів

Чат-бот можна представити як комп'ютер, програму, алгоритм або штучний інтелект, основною метою якого є спілкування з людиною або іншим співрозмовником. Чат-боти можуть бути розроблені для спілкування, відповідаючи на прості ключові слова, або для взаємодії з користувачем, підтримуючи певні теми. Їх можна використовувати для обслуговування клієнтів, маркетингу, реклами, індустрії розваг, збору даних тощо. Наприклад, чат-бот може спілкуватися з клієнтами в онлайн-магазинах і допомагати їм знайти потрібний продукт.

Термін chatbot – chat(-ter) bot був винайдений Майклом Лореном Молдіном. Відповідно до спрощеного визначення Шавара та Етвелла, чат-боти — це чат-додатки зі штучним інтелектом, функції яких варіюються від відповідей на прості запитання до участі в складних бесідах.

Чат-бот – це програмний додаток, який допомагає вести розмову за допомогою текстових або звукових методів. Програми в чат-ботах створені для імітації людських розмов. Чат-боти використовуються для різних цілей, включаючи обслуговування клієнтів, маршрутизацію запитів і пошук інформації. За допомогою обробки природної мови деякі чат-боти можна використовувати для класифікації слів.

Найпростіші боти обробляли основні повідомлення та запити користувачів. У опублікованій розмові з користувачем за допомогою алгоритму він відповідає заздалегідь запрограмованими відповідями. Чат-боти найчастіше використовуються діалоговими системами, такими як служба підтримки клієнтів.

Штучний інтелект (AI) імітує людський інтелект у формі різних пристроїв. Такі пристрої штучного інтелекту розроблені, щоб поводитися як люди та імітувати їхні дії. Штучний інтелект означає машини, які демонструють природні характеристики, подібні до людського розуму, такі як вирішення проблем і розуміння.

Обробка природної мови (NLP) — це програмний інтерфейс, який дозволяє комп'ютерам і людям взаємодіяти один з одним. NLP дозволяє комп'ютеру аналізувати величезну кількість мовних даних з різних джерел. Розробники техніки НЛП можуть покращити знання чат-бота для обробки різноманітних завдань, таких як аналіз тексту, стемінг, підсумовування тексту, автоматичне підсумовування, виділення теми, аналіз тексту, розпізнавання мовлення, переклад, сегментація та автоматичні відповіді на запитання.

NLP поєднує дві техніки – генерування природної мови (NLG) і розуміння природної мови (NLU). НЛП – це одна з форм штучного інтелекту, яка дозволяє чат-ботам обробляти текстові чи аудіо-повідомлення користувачів і давати правильну відповідь. NLP має наступні рівні:

- застосування;
- зберігання даних;
- NLP Engine;
- платформа Data Lake.

NLP містить дві важливі техніки для правильної роботи:

- NLU – Natural Language Understanding (Розуміння природної мови) відображає надані користувачем вхідні дані для своїх корисних представлень. Він аналізує незвичайні явища мови;

- NLG – генерація природної мови використовується для планування тексту та речень, аналізу та реалізації тексту.

NLU — це процес генерування осмислених відповідей у формі природної мови, який включає:

- текстове планування отримує важливі дані з бази даних;

- планування речень вибирає необхідні слова, створює релевантні фрази, використовуючи ці слова, і створює осмислені речення.

Текстова реалізація — це відображення потоку речення відповідно до структури речення.

Набагато важче реалізувати процес розуміння природної мови, ніж виконати процес генерації природної мови.

Natural Language Toolkit (NLTK) було запрограмовано для створення та застосування символічного та статистичного NLP на Python. Це набір бібліотек, які включають синтаксичний аналіз і класифікацію тексту, обробку тексту для токенізації, формування коренів слів і міркування про семантику.

Keras — це бібліотека API для нейронних мереж на Python. Він добре поєднується з R, PlaidML, TensorFlow і Microsoft Cognitive Toolkit.

Tkinter — це інтерфейс для розробки програм із графічним інтерфейсом користувача (GUI).

Tensorflow — це фреймворк бібліотеки програмного забезпечення, який зосереджений головним чином на машинному навчанні. Він використовує графіки потоків даних і розрізняє програмування для різних завдань для створення моделей. Він використовується для створення великомасштабних програм розробки, включаючи нейронні мережі. Він в основному використовується для класифікації, розуміння, передбачення та створення.

На даний момент існує багато відомих чат-ботів, які використовуються в повсякденному житті. Наприклад, Siri – віртуальний помічник інтерфейсу iOS, який використовує природну мову для відповідей на запити користувачів і виконання запитів веб-сервісів. Google Now — ще одна мобільна програма для пристроїв Android та iOS, яка надає розроблені Google прогнозні карти з інформацією та щоденними оновленнями, щоб автоматично відповідати на запитання користувачів. Cortana - голосова платформа Windows з інформацією, яка допомагає розробникам програмного та апаратного забезпечення. Chatbots for Messenger - програма, яка використовує штучний інтелект (AI) для реалізації взаємодії з клієнтами. Facebook запустив платформу, щоб дозволити розробникам створювати чат-ботів, які можуть спілкуватися з користувачами Facebook через інтерфейс чату Messenger. Чат-боти Messenger розуміють запитання користувачів і формулюють відповіді в дуже людський спосіб.

При проектуванні телеграм-ботів, перш за все, вивчається Telegram Bot API, яке є основою для розробки телеграм-ботів. Це програмний інтерфейс, який надає доступ до всіх функціональних можливостей Telegram для взаємодії з ботами.

Розробники повинні ознайомитися з документацією та зрозуміти, як використовувати різні методи API для взаємодії з користувачами та виконання різних завдань.

Другим важливим аспектом є вибір кращої технології для розробки телеграм-бота. Python-Telegram-Bot є одним з популярних виборів, оскільки має простоту використання та широкий спектр функціональності. Для асинхронного програмування, можна використовувати бібліотеку aiogram. Node.js-Telegram-Bot-API також є популярним варіантом, забезпечуючи швидкість та продуктивність завдяки асинхронній архітектурі. Ботпрес відкриває можливості використання візуального конструктора та інтуїтивного інтерфейсу для швидкої розробки та керування ботами.

Одним з основних викликів розробки телеграм-ботів є забезпечення безпеки та захисту користувачів. Розробники повинні бути уважними до можливих загроз, таких як зловживання або несанкціонований доступ до інформації. Важливо використовувати шифрування для захисту конфіденційної інформації та забезпечення правильного контролю доступу до функцій бота.

Також варто звернути увагу на розробку зручного та інтуїтивного інтерфейсу для користувачів. Це означає, що бот повинен мати зрозумілу структуру, легко навігацію та зрозумілі команди для взаємодії з користувачем. Дизайн інтерфейсу повинен бути зручним і привабливим, а текстові повідомлення мають бути чіткими та зрозумілими для користувачів.

Окрім того, важливо враховувати масштабованість та ефективність розробленого бота. Телеграм-бот може мати велику кількість користувачів, тому необхідно забезпечити, щоб бот працював стабільно та ефективно навіть при великому навантаженні. Розробники повинні оптимізувати код, використовувати асинхронні операції та розглядати можливості горизонтального масштабування для забезпечення високої продуктивності бота.

В таблиці 2.1 здійснено порівняння найвідоміших технологій розробки телеграм-ботів.

Таблиця 2.1 – Аналіз сучасних технологій розробки телеграм-ботів

Технологія	**Переваги**	**Недоліки**
Python-Telegram-Bot	Простота використання та широкий спектр функціональності	Потреба в базових знаннях Python
	Підтримка асинхронного програмування за допомогою бібліотеки aiogram	Обмежена підтримка великого обсягу запитів та активних користувачів
	Багато ресурсів та прикладів коду, які допомагають при розробці	
Node.js-Telegram-Bot-API	Швидкість та продуктивність завдяки асинхронній архітектурі	Потреба в знаннях JavaScript
	Велика кількість модулів та бібліотек для розширення функціональності	Обмежені можливості роботи з великими обсягами даних
	Детальна документація та активна спільнота розробників, що допомагає швидко розв'язувати проблеми	
Botpress	Візуальний конструктор та інтуїтивний інтерфейс для розробки та керування ботом	Обмежена кількість доступних інтеграцій з іншими сервісами
	Великий набір вбудованих модулів та функціональних можливостей для швидкої розробки	Вимагає ресурсів для запуску і підтримки сервера Botpress
	Підтримка розгортання на різних платформах, включаючи хмарні сервіси	
Dialogflow	Використання штучного інтелекту для розумного аналізу та обробки повідомлень	Обмежені можливості безпосереднь

2.2 Розробка структури телеграм-бота для управління персоналом

Telegram-бот для управління персоналом — це цифровий помічник, створений для автоматизації кадрових процесів за допомогою месенджера Telegram. Такий бот забезпечує зручну взаємодію працівників із системою управління персоналом, замінюючи частину рутинної роботи HR-фахівців. Він дозволяє швидко отримувати інформацію, подавати заявки, надсилати повідомлення, залишати зворотний зв'язок та вирішувати типові кадрові запити без необхідності особистого звернення до відділу кадрів.

Бот функціонує на основі заздалегідь розроблених сценаріїв або алгоритмів штучного інтелекту, які обробляють вхідні повідомлення від користувачів, аналізують їх, надають відповіді або передають інформацію далі – наприклад, до керівника чи HR-менеджера.

Він має чітку логічну структуру, яка складається з кількох основних частин:

- інтерфейс користувача — це те, з чим безпосередньо працює працівник у Telegram. Він містить команди, меню, кнопки та текстові повідомлення, які дозволяють виконувати потрібні дії швидко та інтуїтивно;

- обробник запитів — програмна частина, яка розпізнає введену інформацію (наприклад, дати відпустки, причину відсутності) і реагує на неї згідно з заданою логікою. Саме тут ухвалюються рішення: що відповісти користувачу, чи потрібно зберігати дані або переслати їх іншим відповідальним особам;

- база даних — система, у якій зберігаються всі відомості про працівників, їхні заявки, залишки відпусток, історія звернень та інша службова інформація. Вона забезпечує зв'язок між різними запитами й дозволяє ботові працювати персоналізовано;

- система сповіщень — надсилає автоматичні повідомлення керівникам або HR-фахівцям, якщо працівник подає запит, або надсилає підтвердження самому працівнику після опрацювання його звернення;

- адміністративна частина — може бути реалізована як окремий інтерфейс для керівництва чи відділу кадрів, де доступні перегляд заявок, керування працівниками, формування звітів тощо.

Завдяки такій структурі Telegram-бот є ефективним інструментом, який не тільки економить час і ресурси, але й покращує внутрішню комунікацію та підвищує рівень цифрової культури в компанії.

Наведемо загальну структуру телеграм-бота, скориставшись діаграмою прецедентів (рис. 2.1).

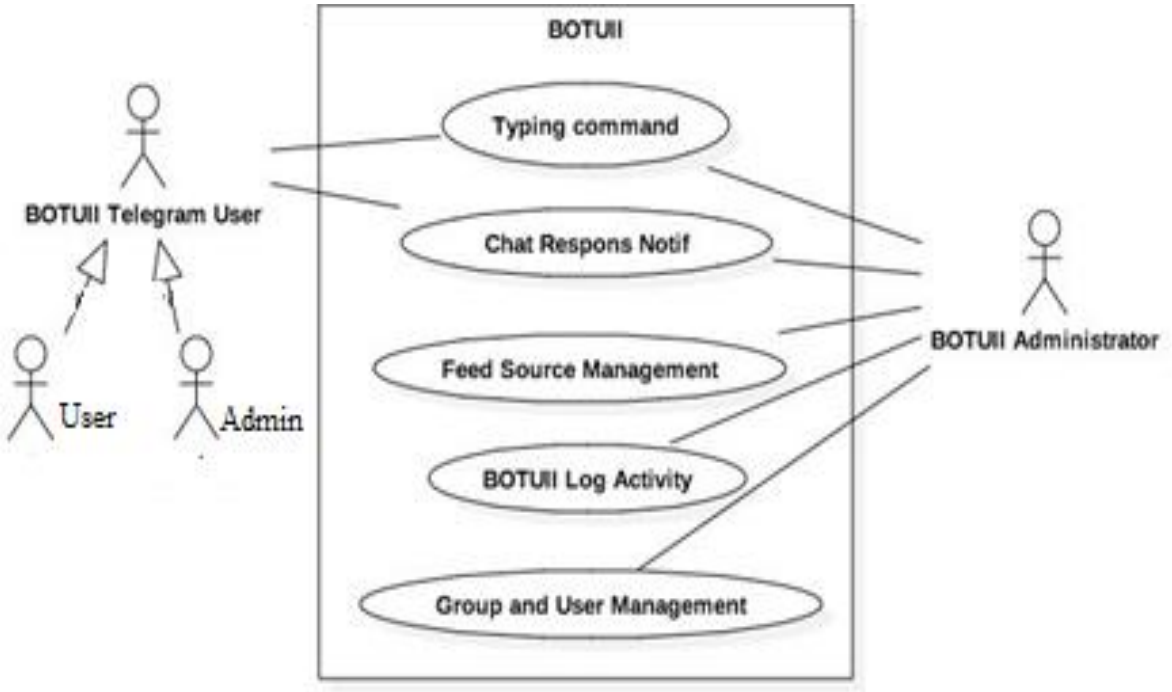

Рисунок 2.1 – Загальна структура роботи телеграм-бота

Діаграми прецедентів – це графічний спосіб моделювання функціональності системи, який дозволяє виділити основні взаємодії між акторами (користувачами) та системою. Вони використовуються для аналізу та опису функціональних вимог до системи з точки зору користувачів.

Діаграми прецедентів складаються з акторів, прецедентів і зв’язків між ними. Актори – це особи, ролі або зовнішні системи, які взаємодіють з системою. Прецеденти – це конкретні дії або функціональні можливості системи, які

задовольняють потреби акторів. Зв’язки між акторами і прецедентами вказують на взаємодію між ними.

Діаграми прецедентів дозволяють уявити систему в цілому, виявити основні функціональні можливості та зв’язки між акторами і прецедентами. Вони допомагають команді проекту та зацікавленим сторонам краще зрозуміти потреби користувачів і специфікації системи, а також визначити пріоритети та обсяг робіт.

Існує два способи зв’язку для спілкування в боті Telegram, це веб-хуки та довге опитування. В основному, метод Webhooks легше розробити, запит на основі HTTP і швидкий під час відповіді на запит. Однак для цього потрібен виділений сервер із публічною IP-адресою, тоді як для тривалого опитування виділений сервер не потрібен. Метод тривалого опитування знижує витрати на створення домашньої автоматизації. Він може підтримувати велику кількість оновлень без обов’язкового виклику API для кожного оновлення. Така домашня автоматизація може бути більш безпечною за допомогою додаткових пристроїв. У цій реалізації вибрано Webhooks, оскільки запитувана інформація має швидко реагувати та працювати 24/7. Прототип телеграм-бота зображено на рисунку 2.2.

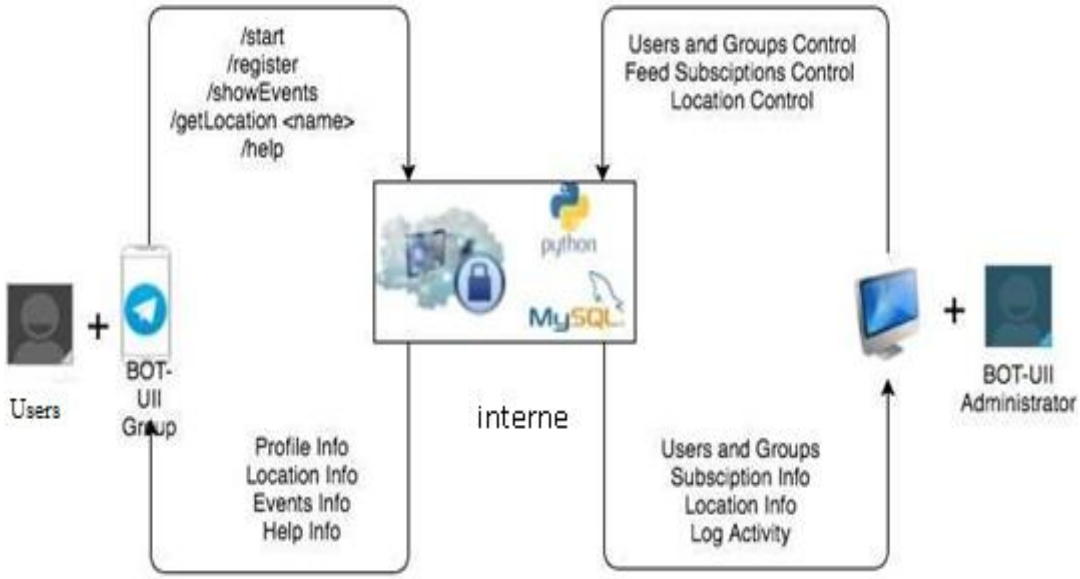

Рисунок 2.2 – Прототип телеграм-бота

2.3 Розробка функціоналу чат-боту для управління персоналом

Chatbot — це обліковий запис на платформі чату, який може отримувати та надсилати повідомлення, щоб ефективно обслуговувати розмову з користувачами. Чатбот не потребує додаткової установки користувачем на платформі. Він надає функції для отримання та надсилання повідомлень. Але чат-бот відрізняється від облікового запису людини, оскільки він не має онлайн-статусу, журналу останнього відвідування та не може дзвонити іншим обліковим записам.

Чат-бот, що імітує розмову з користувачем. Розмови здійснюються шляхом відповіді на запит користувача. Чат-бот отримує повідомлення через API. Чат-бот обробляє вхідне повідомлення. Після обробки чат-бот надішле користувачеві відповідь через API.

Рисунок 2.3 – Діаграма взаємодії чат-бота з API

У Telegram є не просто платформа для чату. Telegram також має службу API, до якої розробник може отримати доступ для використання Telegram у різних потребах. API, який зазвичай використовується для створення чат-бота. Якщо ми хочемо використовувати Telegram API, ми повинні створити обліковий запис чат-бота в Telegram. Ми отримаємо токен, який можна використовувати для доступу до Telegram API.[

Для запуску чат-бота Telegram API Telegram надає два методи: довге опитування та вебхук. При тривалому опитуванні сервер повинен періодично перевіряти додаток. Коли надходять повідомлення, сервер їх обробляє. Методи тривалого опитування підходять для тих, хто не має https-сервера або веб-хостингу, тому він може працювати на своєму комп’ютері. У той час як метод webhook не потребує рутинної перевірки, оскільки API Telegram надішле його на сервер, коли

надходить повідомлення. Таким чином, щоб запустити метод webhook, ми повинні мати активний https-сервер або веб-хостинг, оскільки Telegram API надсилатиме вхідні повідомлення на свій сервер.

Веб-програма – це програма, розроблена для доступу через веб-браузер. Щоб отримати доступ через Інтернет, він має використовувати мову, яку розуміє веб-браузер, наприклад HTML і Javascript.

Фронтальна обробка інтерфейсу користувача. Front-end складається з трьох основних компонентів: HTML, CSS і Javascript. Ці троє мають близькі стосунки і свою роль. Базою інтерфейсу стає HTML, який має ідеальну структуру та шаблон. CSS керує стилем веб-інтерфейсу. Javascript відіграє роль, щоб зробити ефект інтерфейсу більш динамічним та оптимальним.

Системна обробка серверної обробки у веб-додатку. Для системи, яку я розробив, я використовую PHP (PHP: Hypertext Processor). PHP — це мова сценаріїв, яка вставляється в HTML. PHP широко використовується в динамічних мережах. Спочатку PHP - це абревіатура від Personal Home Page. Спочатку він був розроблений Расмусом Лердорфом у 1995 році. Тоді він називався Form Interpreted (FI). Це ще купа скриптів, які обробляють форму даних з Інтернету.

База даних - це сукупність даних, яка має логічний зв'язок і призначена для забезпечення деяких потреб організації в інформації. Система керування базами даних (СУБД) — це програмна система, яка надає користувачам доступ для визначення, створення, підтримки та контролю бази даних.

Для системи керування чат-ботом Telegram, яка розроблена в межах цієї роботи, актуальним є чітко визначений функціонал, що дозволяє ефективно автоматизувати основні завдання, пов’язані з управлінням персоналом. Функціональні можливості системи сформовані відповідно до потреб користувачів (працівників, керівників і HR-відділу) та орієнтовані на забезпечення зручного, швидкого і зрозумілого способу виконання типових кадрових дій.

Таблиця 2.2 – Функціонал чат-боту

Опція	Функціонал
Набір кнопок	Використовується для створення та редагування наборів кнопок, які можуть бути використані в наборах дій. Ця функція дозволяє створювати розміщення кнопок, додавати підпис та значення до кнопок.
Набір дій	Використовується для створення та редагування наборів дій, які потім можуть бути викликані за допомогою команд у відповідь користувачам чат-боту. Підтримувані дії включають фотографії, відео, аудіо, документи, місця розташування, контакти, кнопки та повідомлення. Ці кнопки викликаються з набору кнопок, який був створений раніше.
Команда	Використовується для встановлення відповіді, яка буде надана користувачам на основі передбачених ключових слів. Відповіді надаються з наборів дій, які були створені раніше.
Прямі повідомлення	Дозволяє адміністратору давати відповіді безпосередньо користувачам, чиї повідомлення не містять передбачених ключових слів.
Розсилка	Використовується для створення розсилок повідомлень всім користувачам чат-боту, які використовували цього чат-бота.
Налаштування	Використовується для зміни налаштувань чат-бота.
Аналітика	Ця функція використовується для відображення статистики використання чат-бота.

Даний функціонал чат-боту зображений на наступній діаграмі прецедентів:

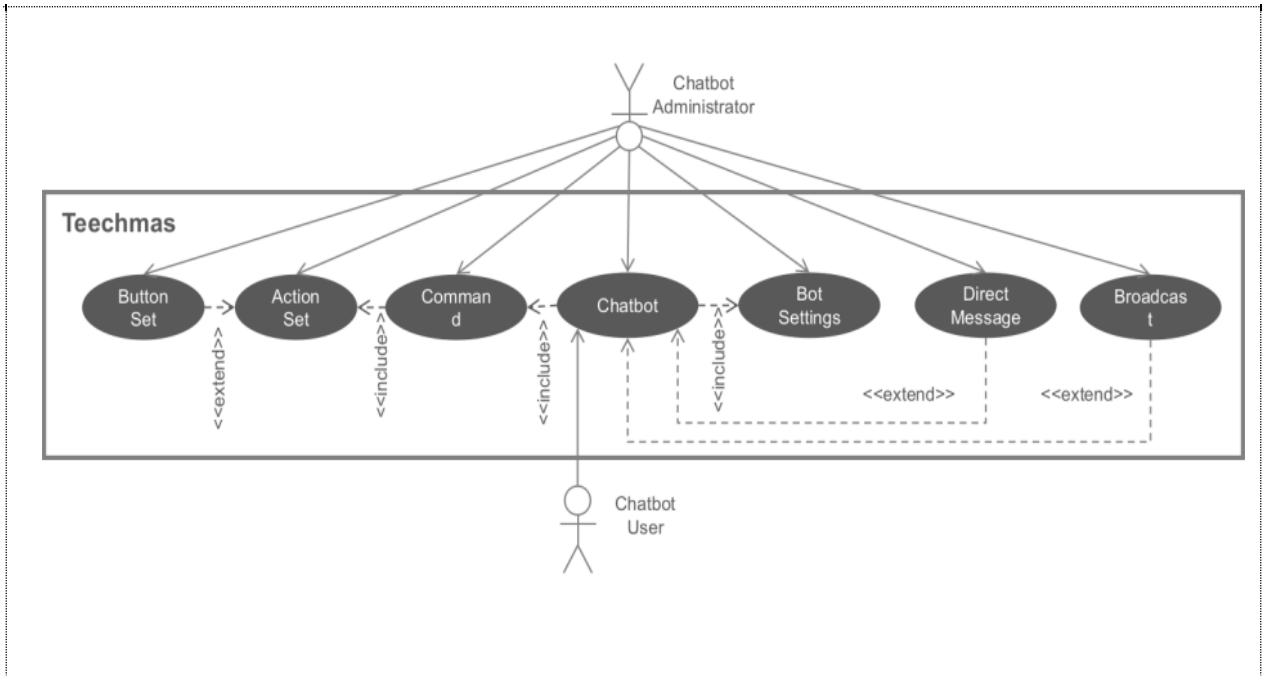

Рисунок 2.4 – Діаграма прецедентів відносно функціоналу чат-бота

2.4 Розробка структури чат-боту для управління персоналом

Дане програмне забезпечення включає в себе декілька модулів, зокрема модуль бібліотек, модуль API, модуль бази даних, модуль категорій.

У модулі бібліотек будуть описані всі потрібні бібліотеки задля того, щоб програма працювала якісно та правильно. Будуть додані такі бібліотеки, як:

- бібліотеки для бази даних;

- бібліотеки для підключення до телеграму.

У модулі API будуть описані функції задля того, щоб підключитися до конкретного бота у месенджері Telegram, який буде функціонувати і користувачі зможуть використовувати даний бот.

Наступний модуль – це модуль категорій. У ньому будуть описані всі потрібні категорії, з якими будуть взаємодіяти користувачі. Будуть такі категорії:

- Продуктивність\ефективність (Effectivity);

- Система якості (Quality);

- 5S;

- Безпека праці (Security);

- Загальний (Other).

Додатково категорія Продуктивність\ефективність розділяється на підкатегорії:

- Перевиробництво;
- Дефекти;
- Зайві рухи;
- Запаси;
- Транспортування;
- Надмірна обробка;
- Очікування;
- Нехтування талантами.

Також категорія 5S розділяється ще на підкатегорії:

- 1S (Сортування);
- 2S (Упорядкування);
- 3S (Чистота);
- 4S (Стандартизація);
- 5S (Удосконалення).

Окрім цього, ще є модуль бази даних (БД). В даному модулі буде описано спосіб підключення до бази даних. Також за допомогою цього модуля буде поетапно записуватися ідеї та проблеми від користувачів, які вони будуть вводити через чат-бота у Telegram.

Модулі між собою взаємодіють. Наприклад модуль бібліотек взаємодіє зі всіма іншими модулями, так як даний модуль дає іншим працювати та виконувати свої функції. Модуль API взаємодіє з модулем категорій, так як API дозволяє підключатися до конкретного бота, то логічно, що завдяки даному модулю, модуль категорій взагалі буде відображатися у інтерфейсі користувача у вигляді окремих кнопок.

На рисунку 2.5 зображено структурну схему описаних модулів чат-боту для управління персоналом.

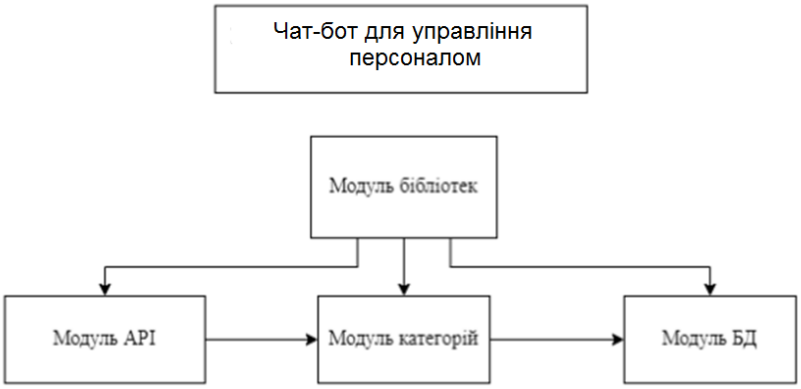

Рисунок 2.5 – Структурна схема чат-боту для управління персоналом

На рисунку 2.6 зображено схему алгоритму функціонування чат-боту для управління персоналом.

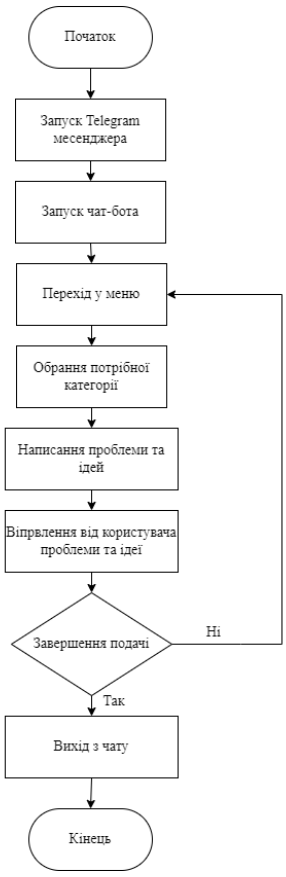

Рисунок 2.6 – Схема алгоритму функціонування чат-боту для управління персоналом

В свою чергу, модуль категорій взаємодіє з модулем БД, так як у категоріях будуть вводитися конкретні проблеми та їдеї від користувачів, після чого вони будуть записуватися у базі даних під конкретно виведеною категорією.

Додатково з бази даних у окреме програмне забезпечення будуть витягуватися усі дані від користувачів, у якому адміністратор зможе взаємодіяти з цими даними: сортування, оброблення та перевірка.

Окрім цього, на рисунку 2.7 зображено діаграму прецедентів чат-боту для управління проектами.

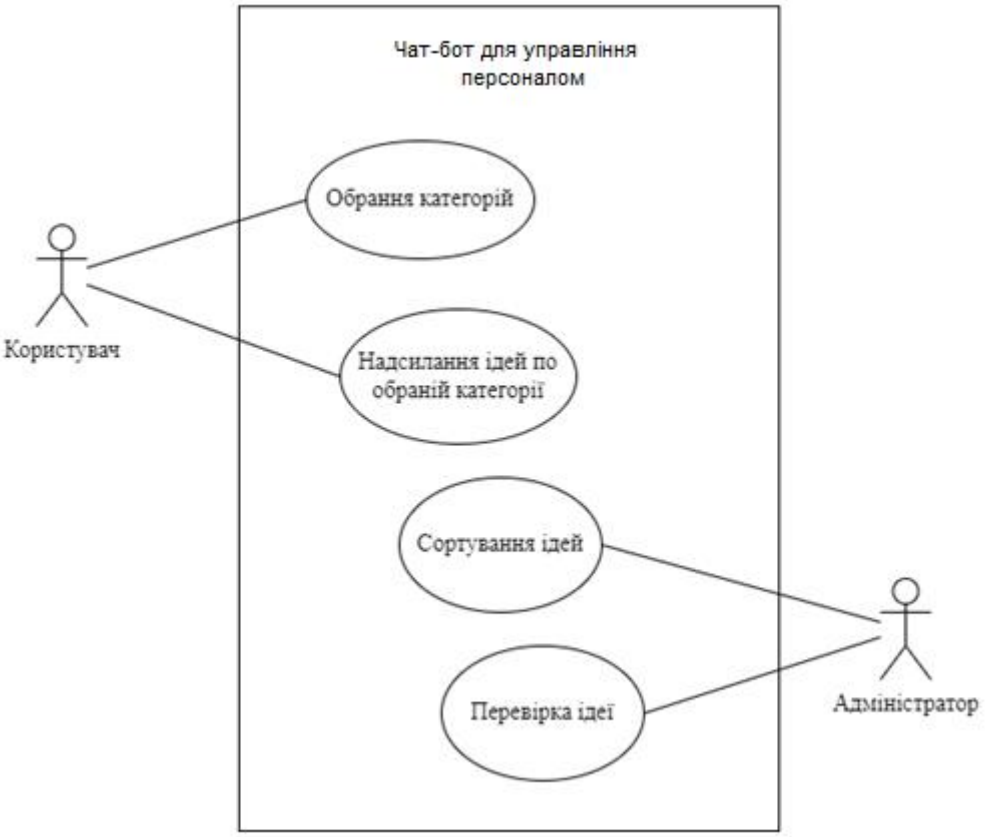

Рисунок 2.7 – Діаграма прецедентів чат-боту для управління проектами

Діаграми прецедентів або use case diagrams – це узагальнена модель функціонування системи. Діаграма має в собі актора або акторів, а також прецеденти (закінчена послідовність дій). У даній діаграмі є два актора: користувач та адміністратор, які мають свої прецеденти [9].

2.5 Розробка UML-діаграм процесу клієнтської частини популяризації Lean-менеджменту на підприємстві

UML розшифровується як Unified Modeling Language. Це багата мова для моделювання програмних рішень, структур додатків, поведінки системи та бізнес-процесів. Існує 14 типів діаграм UML, які допоможуть змоделювати ці моделі поведінки [9].

Є дві основні категорії; структурні діаграми та діаграми поведінки. Існують структурні діаграми таких типів:

- діаграма класів;
- діаграма компонентів;
- діаграма розгортування;
- діаграма об'єкта.

Існують такі діаграми поведінки:

- діаграма прецедентів;
- діаграма станів;
- діаграма діяльності;
- діаграма послідовності;
- діаграма активності;
- діаграма взаємодії.

Як найбільш відомий тип діаграми серед приведених типів UML, діаграм Use case дають графічний огляд учасників системи, різних функцій, необхідних цих учасників, і того, як ці різні функції взаємодіють. Це відмінна оперативна точка для обговорення будь-якого проекту, оскільки можна легко визначити основних учасників та основні процеси системи [9].

Діаграма станів застосовується для того, щоб пояснити, яким чином працюють складні об'єкти. Також ця діаграма показує, як об'єкт переходить з одного стану в інший, крім цього служить для моделювання динамічних аспектів системи. Діаграма станів описує процес зміни станів тільки одного представника певного класу об'єкта,

поведінка якого характеризується його реакцією на зовнішні події [9]. На рисунку 2.8 зображено діаграму станів чат-боту для управління персоналом.

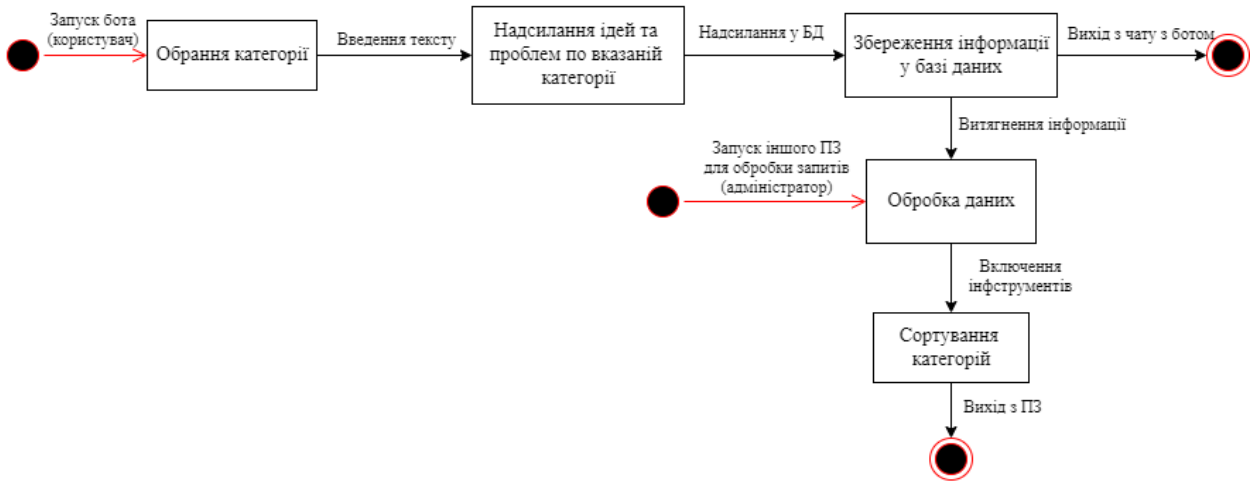

Рисунок 2.8 – Діаграма станів чат-боту для управління персоналом

На даній діаграмі зображено процес зі сторони користувача та зі сторони адміністратора. Початок роботи зі сторони користувача, тобто початок роботи чат-бота, починається з його запуску, бот має в собі п’ять категорій і в одній з категорій має п’ять підкатегорій. Після обрання потрібної категорії\підкатегорії бот надсилає інструкцію з тим, як правильно вводити дані та текст від користувачів. Після того, як було введено всю інформацію, бот, за допомогою модуля бази даних зберігає всі введенні відповіді у базі даних, де вони будуть зберігатися до подальшої обробки. Робота зі сторони адміністратора починається зі запуску окремого програмного забезпечення. У даному програмному забезпеченні дані з бази даних «витягуються» і представляються адміністратору. У свою ж чергу, адміністратор, обробляє ці дані і сортує категорії та їх оцінює. Робота закінчується тоді, коли адміністратор зберіг всі свої модифікації та виходить з розробленого програмного забезпечення.

Діаграма активності\діяльності – представляють робочі процеси в графічному вигляді. Вони можуть бути використані для опису робочого процесу або робочого процесу будь-якого компонента системи. Іноді діаграми діяльності використовуються в якості альтернативних діаграм машин станів. На рисунку 2.9 зображено діаграму активності чат-боту для управління персоналом.

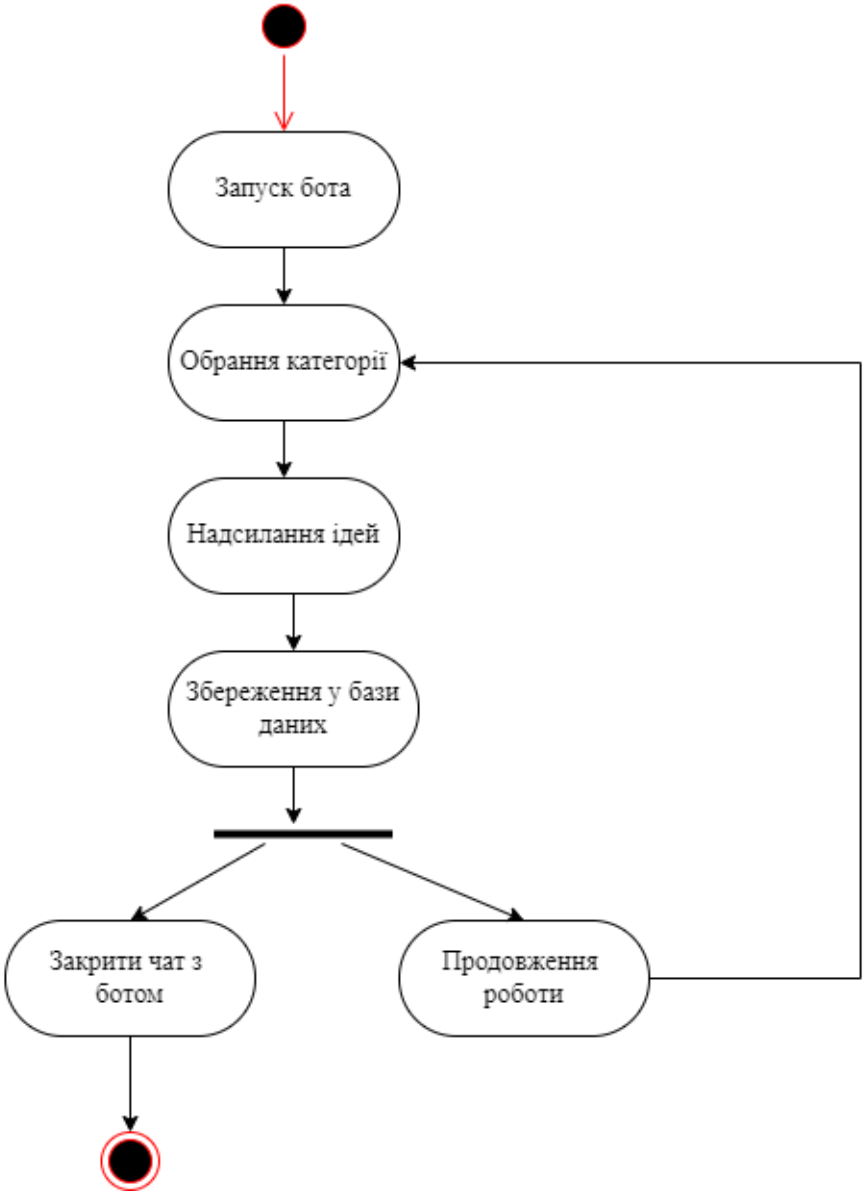

Рисунок 2.9 – Діаграма активності\діяльності чат-боту для управління персоналом

На даній діаграмі зображено поетапна робота програмного продукту зі сторони клієнта. Робота розпочинається з запуску бота, після чого користувач обирає одну з п’ятих категорій, а у випадку категорії 5S обирається ще одна з п’яти підкатегорій. Після чого користувач, він же працівник підприємства, поетапно описує проблеми на робочому місці, а після цього і саму ідею покращення. Коли користувач ввів все, що було потрібно, він надсилає свою ідею, яка зберігається у базі даних. Після цього у працівника підприємства є вибір: продовження роботи або завершити чат з ботом. У першому випадку користувач знову обираємо потрібну

йому категорію і етапи повторюються. У другому випадку користувач завершує своє листування з чат-ботом.

Окрім вище загаданих діаграм ще існують структурні діаграми. Вони показують речі в системі, що моделюється. У більш технічному сенсі вони демонструють різні об'єкти у системі. Поведінкові діаграми показують, що має відбуватися у системі. Вони описують, як об'єкти взаємодіють один з одним, щоб створити систему, що функціонує [9].

Діаграми класів є основним будівельним блоком будь-якого об'єктно-орієнтованого рішення. Вони показують класи в системі, атрибути та операції кожного класу, а також відносини між кожним класом. У більшості інструментів моделювання клас складається із трьох частин. Ім'я у верхній частині, атрибути в середині та операції або методи у нижній частині. У великій системі з великою кількістю зв'язаних класів класи групуються разом для створення діаграм класів [9]. Різні відносини між класами показуються різними типами стрілок.

Діаграма компонентів відображає структурний зв'язок компонентів програмної системи. В основному вони використовуються під час роботи зі складними системами з великою кількістю компонентів. Компоненти взаємодіють один з одним за допомогою інтерфейсів. Інтерфейси пов'язані між собою за допомогою конекторів [9].

Діаграми об'єктів, іноді звані діаграмами примірників, є важливим елементом моделювання в мові UML (Unified Modeling Language). Вони дуже схожі на діаграми класів, оскільки також демонструють структуру системи, але на відміну від останніх, діаграми об'єктів відображають конкретні екземпляри об'єктів у певний момент часу, а не абстрактні класи.

Такі діаграми дають змогу побачити, як система виглядає у динаміці, коли її компоненти взаємодіють один з одним у реальному контексті. Кожен об'єкт у діаграмі має ім’я та належить до певного класу, причому значення його атрибутів уже встановлені. Це робить діаграму об'єктів ідеальним інструментом для ілюстрації поточного стану системи на певному етапі виконання.

Вони особливо корисні при поясненні складних відносин між об’єктами, оскільки кожен об’єкт містить конкретні дані (значення атрибутів), що дозволяє наочно показати логіку зв’язків. Завдяки своїй наочності, ці діаграми часто застосовуються під час навчання, документування або обговорення системи з командою.

На відміну від діаграм класів, які є більше статичними та описують загальну структуру, діаграми об'єктів мають більш динамічний характер і показують, як саме компоненти взаємодіють у певному контексті. Таким чином, вони доповнюють процес моделювання, забезпечуючи більш глибоке розуміння функціонування програмного забезпечення або інформаційної системи.

Діаграми послідовностей в UML показують, як об’єкти взаємодіють між іншим і в якому порядку ці взаємодії проходять. Важливо, що вони показують взаємодії для конкретного сценарію. Процеси представлені вертикально, а взаємодії показані у вигляді стрілок [9].

Діаграми взаємодії показують потік повідомлень між об’єктами системи й основними асоціаціями між ними.

2.6 Висновок до розділу 2

В даному розділі було розроблено структуру чат-боту для управління персоналом. Визначено основні модулі зі сторони користувача. Було побудовано структурну схему взаємодії модулів чат-боту та побудовано діаграму прецедентів програмного продукту. Окрім цього було створено діаграму станів програмного продукту для управління персоналом. Створено діаграму активності\діяльності чат-боту для управління персоналом.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ЧАТ-БОТУ ДЛЯ УПРАВЛІННЯ ПЕРСОНАЛОМ

3.1 Обґрунтування вибору мови програмування та середовища розробки

Для аналізу і порівняння було обрано три мови програмування: C#, JavaScript, Python.

C# - це мова програмування, що відноситься до мов із C-подібним синтаксисом. Ця мова ввібрала в себе елементи C++ і Java та була створена для розробки додатків на платформі Microsoft .NET Framework. Мова має строгу типізацію, проте має динамічну типізацію(тип змінної визначається в момент присвоєння їй значення, тобто тип змінної може бути різним в ході виконання програми). Використовується для створення клієнтських та веб-додатків. Крім того, в Visual Studio вбудований garbage-collector, який сам контролює пам’ять. C# використовується через свою універсальність.

JavaScript – мова програмування, що розвивається найбільш динамічно. Спочатку була орієнтована лише на веб-розробку, а саме на її клієнтську частину. Проте можливості мови розширюються з прийняттям нових стандартів EcmaScript. Так, на даний момент за допомогою JavaScript без труднощів розробляється і серверна частина веб-сайтів і додатків, а створення платформи Node.js дозволило створювати звичні програмні додатки. У JavaScript взагалі відсутня статична типізація, тобто в момент оголошення не вказується тип змінної, він присвоюється лише разом із присвоєнням або зміною значення, що полегшує роботу з несумісністю типів. Прийняття нових стандартів призвело до спрощення синтаксису JavaScript і додало класи, що робить JavaScript повноцінною об’єктно-орієнтованою мовою [10].

Python – інша мова програмування, що швидко розвивається. Мова має простий і зрозумілий синтаксис. Це значно спрощує процес розробки. Має динамічну типізацію та автоматичний контроль пам’яті та механізм обробки виключень. Найчастіше використовується для створення веб-додатків та у сфері

машинного навчання та штучного інтелекту. Мова підтримується великою кількістю бібліотек, що робить написання коду простішим і швидшим. Особливо корисними є бібліотеки для проведення математичних розрахунків, графічного відображення тощо результатів обчислення тощо. Використання бібліотек і низький поріг входження зробили Python однією з найбільш популярних мов програмування. Окрім цього Python дає можливість легко підключатися до різноманітних баз даних і спрощує написання кодів для чат-ботів у різних месенджерах [11]. У таблиці 3.1 наведена порівняльна характеристика мов програмування.

Таблиця 3.1 – Порівняльна характеристика мов програмування.

Мова/ Критерій	Статична типізація	Динамічна типізація	Виконання на клієнт.частині	Легке підключання до БД	Крос-платформеність
C#	+	+	-	+	-
JavaScript	-	+	+	-	+
Python	-	+	+	+	+

Для виконання бакалаврської кваліфікаційної роботи було обрано мову програмування Python, так як дана мова дає можливість легко підключатися до баз даних. Водночас, простий синтаксис та низький поріг входження дозволяє швидко і якісно розробити основні функціональні модулі програми.

Visual Studio Code – це редактор вихідного коду, що розповсюджується на безкоштовній основі компанією Microsoft. В першу чергу редактор призначений для створення додатків мовою програмування JavaScript, проте завдяки великій кількості розширень підходить також для розробки на Python, C#, C++, Ruby і для багатьох інших мов програмування.

Одними з головних переваг Visual Studio Code над конкурентами можна виділити оптимізацію і «модульність». Існує велика кількість розширень редактора, що додають нові функції, такі як автодоповнення коду, сніппети, підтримку синтаксису інших мов програмування та вбудований Git. Також редактор має гнучке налаштування клавіш, шрифту, кольорової схеми тощо [12].

Зі встановленими розширеннями, Visual Studio Code підтримує усі стандарти Python, підтримку основних інструментів роботи, таких як Jupyter Notebook тощо. Має можливість встановити та використовувати будь-який термінал, як, наприклад, powershell, bash, fish тощо. Середа підтримує такі формати, як JSON, XML, UML та має розширення для побудови схем з файлового опису UML. Підтримує графічні файли багатьох розширень, таких як .svg, .png, .jpg, .gif та інші. Крім вбудований плагін intellisense, який значно пришвидшує написання коду. Вигляд робочої області середовища зображено на рисунку 3.1.

Таким чином, Visual Studio Code є легким, модульним та універсальним інструментом створення програмного забезпечення.

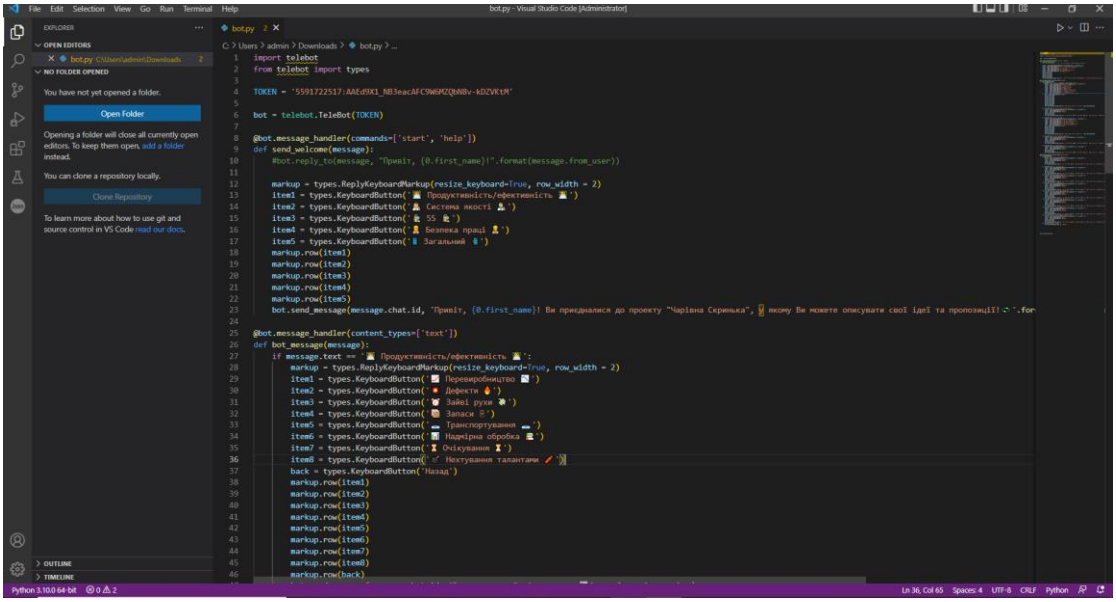

Рисунок 3.1 – Вікно середовища розробки Visual Studio Code

Telegram – це безкоштовна, кросплатформна хмарна служба обміну миттєвими повідомленнями. Сервіс також надає наскрізні зашифровані відеодзвінки, VoIP, обмін файлами та деякі інші функції. Різні клієнтські програми доступні для настільних та мобільних платформ, включаючи офіційні програми для Android, iOS, Windows, macOS та Linux. Існують також два офіційні веб-двійники Telegram, WebK і WebZ, і безліч неофіційних клієнтів, які використовують протокол Telegram. Усі офіційні компоненти Telegram мають відкритий вихідний код, крім сервера, який є

закритим і пріоритетним.

Одним з головних переваг Telegram – це публічні API, за допомогою яких розробники можуть отримати доступ до тих же функцій, що й офіційні програми Telegram, щоб створювати власні програми для обміну повідомленнями. Telegram також пропонує API, що дозволяє розробникам створювати ботів, які є акаунти, керовані програмами. Крім того, Telegram пропонує функції для здійснення платежів безпосередньо всередині платформи, поряд із зовнішнім сервісом, таким як Stripe.

Зараз Telegram-боти, і боти взагалі, стали невід'ємною частиною у житті людей. Боти в прямому сенсі спрощують життя людей у різних сферах діяльності: починаючи від звичайних обрахунків і закінчуючи подачею лічильників. А ситуація з повномасштабною війною в України дало новий поштовх для діджиталізації в країні. Різні волонтерські служби створили чи створюють Telegram-ботів задля полегшення роботи. Наприклад Українська волонтерська служба всі запити приймають через спеціального бота у месенджері Telegram. Через даний бот можна подати запит задля пошуку тимчасового прихистку, допомоги військовим, гуманітарної допомоги і тому подібне. Окрім цього є боти, які надають різноманітну інформацію для внутрішньо переміщених осіб чи допомагають у пошуку зниклих людей.

3.2 Програмна реалізація чат-боту для управління персоналом

Чат-бот складатиметься з 4 основних модулів:

Перший модуль – модуль бібліотек.

В даному моделі описуються всі потрібні бібліотеки задля того, щоб бот міг виконувати свої функції. У ньому описуються бібліотека яка надає підключення до бази даних, на яку будуть надсилатися запити від користувачів. Також бібліотеки, які надають підключення до самого телеграму, щоб з ботом могли користуватися користувачі. Код, що реалізує підключення бібліотек:

```
import firebase_admin
from firebase_admin import credentials
from firebase_admin import db
from firebase_admin import storage
import telebot
from telebot import types
from token_ import token_
```

Другий модуль – модуль API.

За допомогою даного модуля йде підключення до конкретного бота через його унікальний токен. Тобто за допомогою даного токена надалі пишуться функції бота. Код, що реалізує модуль API:

```
TOKEN = '5591722517:AAEd9X1_NB3eacAFC9W6MZQbN8v-kDZVKtM'
bot = telebot.TeleBot(TOKEN)
```

Третій модуль – Модуль категорій.

В даному модулі описуються всі потрібні категорії, з якими будуть взаємодіяти користувачі. Всього категорій п’ять: продуктивність\ефективність, система якості, 5S, безпека праці та загальний. У категорії продуктивність\ефективність вісім підкатегорій, а у категорії 5S п’ять підкатегорій. Фрагмент коду, що реалізує модуль категорій:

```
@bot.message_handler(commands=['start', 'help'])
def send_welcome(message):
      #bot.reply_to(message, "Привіт, {0.first_name}!".format(message.from_user))
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True, row_width = 2)
    item1 = types.KeyboardButton('👩💻 Продуктивність/ефективність 👨💻')
    item2 = types.KeyboardButton('👩🔬 Система якості 👨🔬')
    item3 = types.KeyboardButton('🌲♀ 5S 🌲♂')
    item4 = types.KeyboardButton('👷♀ Безпека праці 👷♂')
    item5 = types.KeyboardButton('📱 Загальний 📲')
    markup.row(item1)
    markup.row(item2)
```

```
        markup.row(item3)
        markup.row(item4)
        markup.row(item5)
        bot.send_message(message.chat.id, 'Привіт, {0.first_name}! Ви приєдналися до проекту "Чарівна Скринька", у якому Ви можете описувати свої ідеї та пропозиції!🐟'.format(message.from_user), reply_markup=markup)
```

Останній модуль – модуль бази даних.

У даному модулі прописано підключення до бази даних, до якої будуть надсилатися, через модуль категорій, усі відповіді від користувачів. Користувач поетапно надсилає повідомлення зі всією потрібною інформацією, а та в свою чергу, записується до бази даних Firebase, до якої підключився бот за допомогою бібліотек Дана БД зберігає дані у вигляді одного JSON дерева. Воно зручне, швидке та надійне. Окрім цього bucket - об'єкт Storage дозволяє API завантажувати об'єкти. Код, що реалізує модуль бази даних:

```
cred = credentials.Certificate("/path/to/secret/key.json")
default_app = firebase_admin.initialize_app(cred, {
        'databaseURL': 'https://realtime-db-name',
    'storageBucket' : 'storage-bucket-local-name'
      })
bucket = storage.bucket()
```

Месенджер Telegram має свої переваги. В першу чергу – це один з найпопулярніших месенджерів. Окрім цього Telegram часто надає різні оновлення, які допомагають різним користувачам у комунікаціях між собою. Додатково даний месенджер надає можливості розробникам вільно використовувати відкритий код і створення ботів набагато легша та швидша ніж у будь-яких інших месенджерах. Також користувачі даного підприємства, для якого і розроблюється бот, в основному використовують для комунікації месенджер Telegram.

3.3 Тестування чат-боту для управління персоналом

Для початку, щоб почати роботу з Telegram-ботом, потрібно встановити Telegram месенджер на свій телефон через Play Market (Android) або App Store (IOS). Якщо робота починається з комп'ютера, то можна або так само встановити додаток на ПК або зайти через браузер. Telegram підтримує більшість операційних систем.

Після того, як додаток був встановлений, потрібно авторизуватися, за умови якщо користувач вже зареєстрований, або зареєструватися за номером телефону. Користувачу відкривається головне вікно зі всіма актуальними чатами, воно зображене на рисунку 3.2.

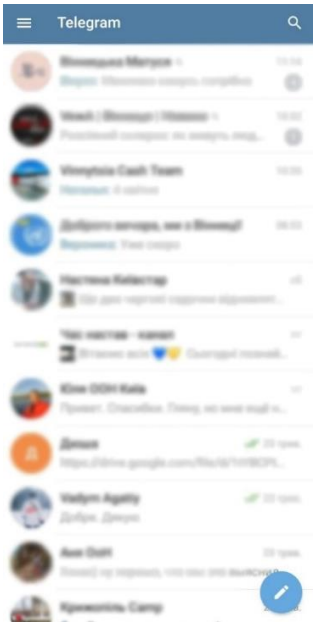

Рисунок 3.2 – Головне вікно з чатами у Telegram месенджері

Для того, щоб знайти бота потрібно натиснути на кнопку пошуку у верхньому правому кутку і після цього відкриється вікно пошуку, яке зображене на рисунку 3.3.

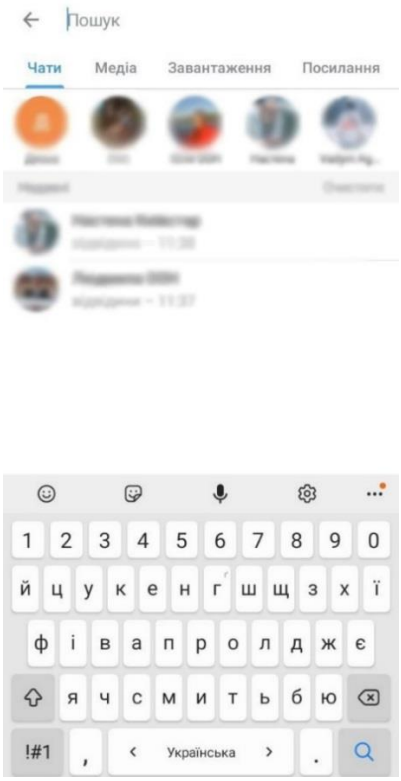

Рисунок 3.3 – Відкриття вікно пошуку у Telegram месенджері

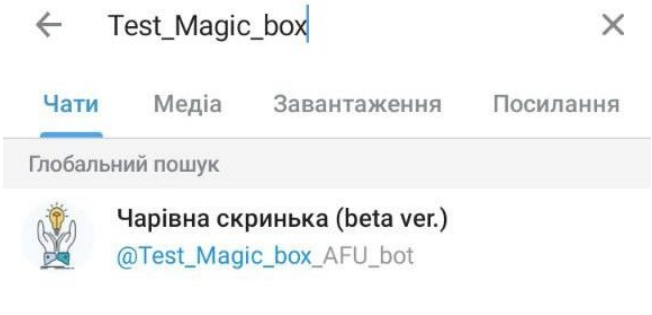

Рисунок 3.4 – Пошук бота за лінком

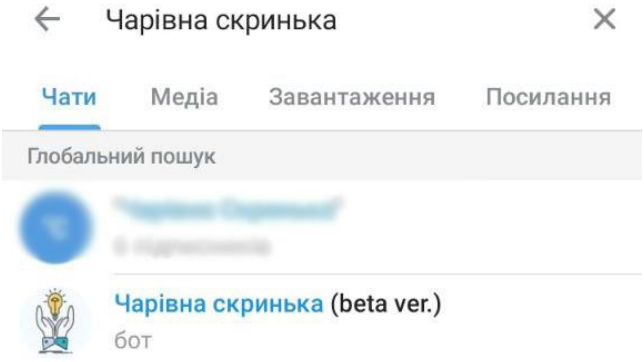

Рисунок 3.5 – Пошук бота за його назвою

Коли користувач заходить в самого бота, але поки не запускаючи його, він бачить початкове повідомлення, для чого саме цей бот потрібен. Стартове повідомлення зображено на рисунку 3.6.

Рисунок 3.6 – Стартове повідомлення від бота

Після цього, потрібно натиснути на кнопку «Розпочати», що знаходиться у низу екрану. Одразу після цього користувач бачить це саме стартове повідомлення, та йому відкривається головне меню з кнопками, які означають категорії, за якими можна подавати ідею (рис. 3.7).

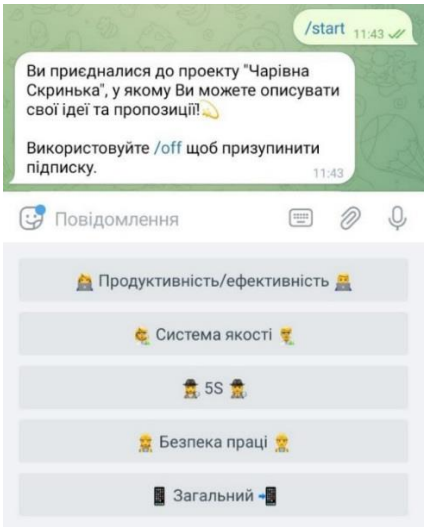

Рисунок 3.7 – Головне меню чат-бота

Після цього, користувач може обрати йому потрібну категорію, а у двох випадках ще і під категорії, задля того, щоб подати проблеми та ідеї для покращення свого робочого місця на підприємстві (рис. 3.8).

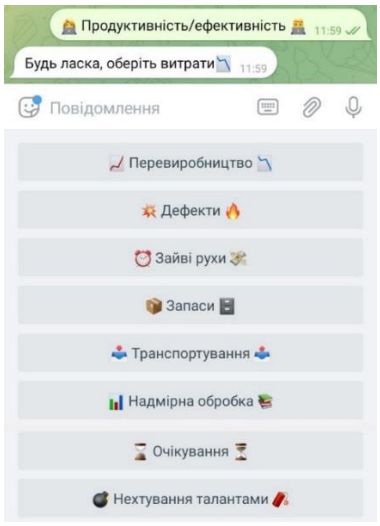

Рисунок 3.8 – Зображення підкатегорій категорії «Продкутивність\ефективність»

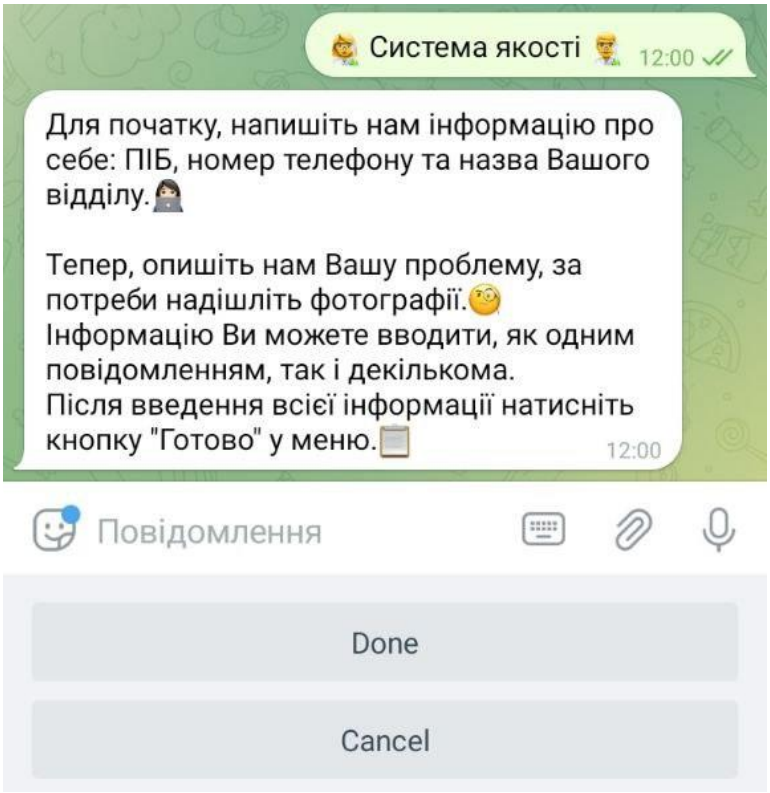

Рисунок 3.9 – Зображення категорії «Система якості»

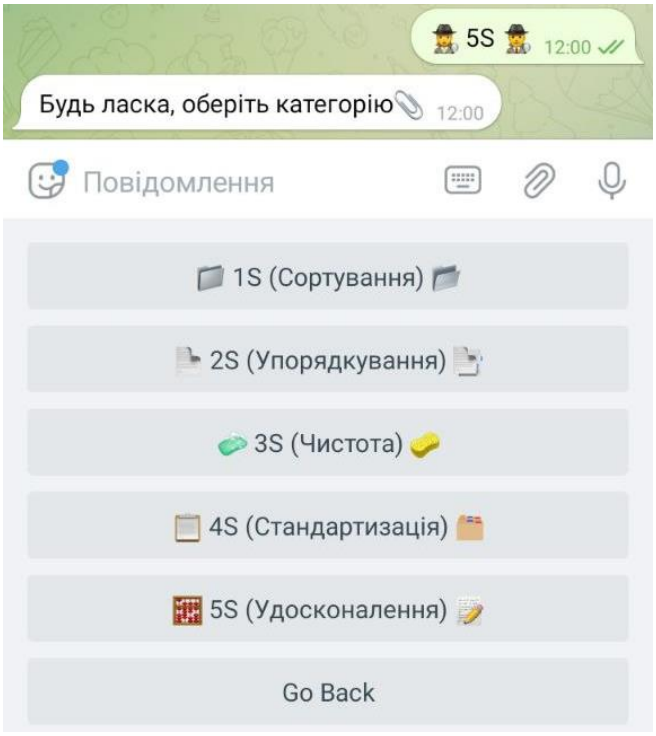

Рисунок 3.10 – Зображення підкатегорій категорії «5S»

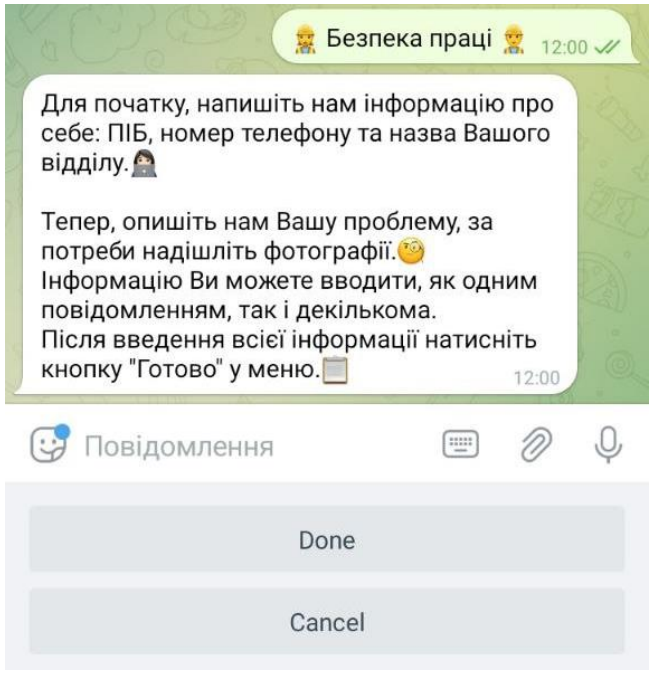

Рисунок 3.11 – Зображення категорії «Безпека праці»

Користувач обирає потрібну йому категорію. Після цього він пише своє ПІБ, номер телефону та назву відділу в якому працює на підприємстві. Після цього він описує проблему, яку він побачив на своєму робочому місці і також подає ідею її вирішення (рис. 3.12).

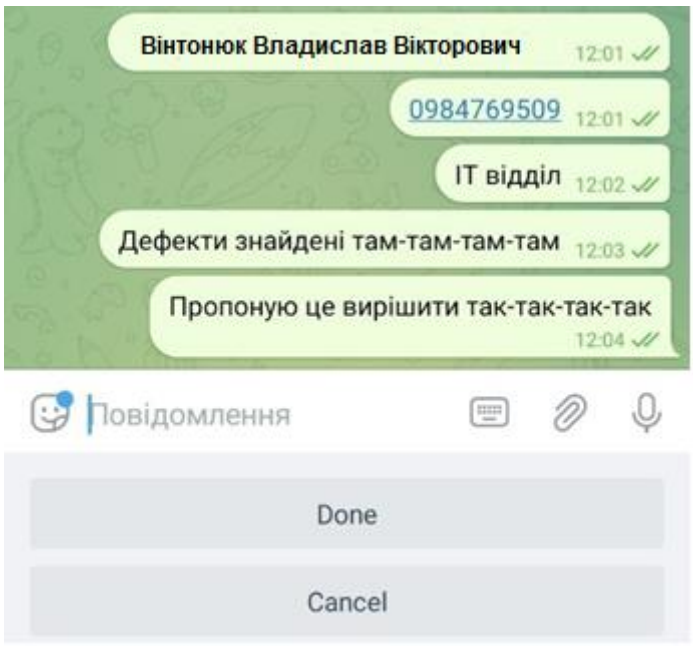

Рисунок 3.12 – Приклад подання проблеми та ідеї до чат-боту

Після цього користувач натискає на кнопку «Done» та ідея надсилається вже на сервер, де і буде далі оброблюватися адміністратором. Після завершення користувачеві знову відкривається початкове меню (рис. 3.13).

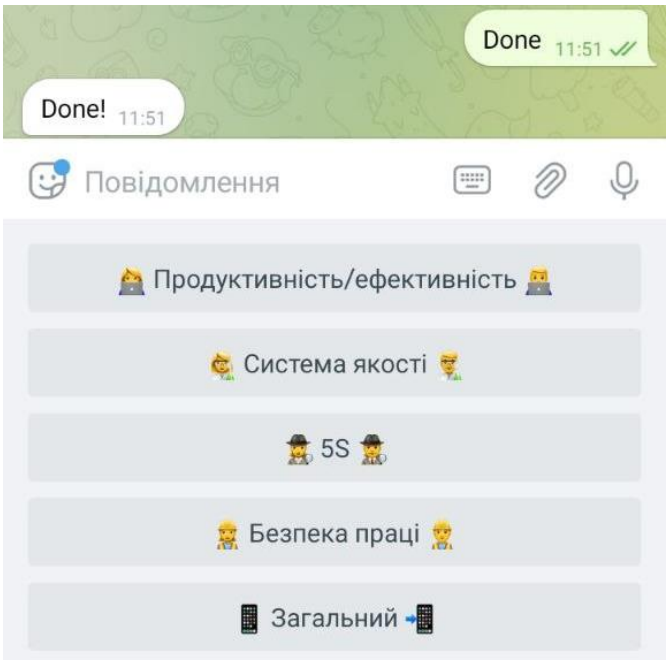

Рисунок 3.13 – Завершення подачі ідеї та повернення на головне меню

Під час тестування чат-боту було підтверджено його коректну роботу та відображення складових частин.

Проведемо порівняння функціональних можливостей програм для управління проектами (табл. 3.2)

Таблиця 3.2 – порівняння функціональних можливостей програм для управління проектами

	Додавання завдань	Управління виконаням завдань	Наявність нічного режиму	Групування завдань	Можливість надсилання завданнь у месенджер та назад
Створена програма «Magic Box»	+	+	+	+	+
Trello	+	+	+	-	-
Asana	+	+	+	-	-

З табл. 3.2 видно, що новий програмний продукт володіє додатковим функціоналом: Групування завдань по категоріях, це корисно, тому що один проект може включати в себе декілька завдань з одією позначкою; Можливість надсилання завданнь у месенджер і назад (наприклад користувачу в Telegram), це зручно, тому що користувачі можіть відповідати напряму з месенджера в дане програмне забезпечення.

3.4 Висновок до розділу 3

В даному розділі було обгрунтовано вибір мови програмування та середовища розробки Здійснено програмну реалізацію чат-бота для управління персоналом з використанням мови Python у середовищі Visual Studio Code. Проведено тестування розробленого чат-боту для управління персоналом.

ВИСНОВКИ

У ході виконання бакалаврської кваліфікаційної роботи розроблено чат-бот для управління персоналом.

В роботі проведено огляд різних методів та систем, що застосовуються для організації управління завданнями для персоналу. Цей аналіз дозволив отримати уявлення про різноманітність інструментів та підходів, які застосовуються в сучасних організаціях для гнучкого управління персоналом. Розроблено структуру чат-боту для управління персоналом. Визначено основні модулі зі сторони користувача. Було побудовано структурну схему взаємодії модулів чат-боту та побудовано діаграму прецедентів програмного продукту. Окрім цього було створено діаграму станів програмного продукту для управління персоналом. Створено діаграму активності\діяльності чат-боту для управління персоналом.

Обгрунтовано вибір мови програмування та середовища розробки Здійснено програмну реалізацію чат-бота для управління персоналом з використанням мови Python у середовищі Visual Studio Code. Проведено тестування розробленого чат-боту для управління персоналом. Тестування підтвердило, що новий програмний продукт володіє додатковим функціоналом: Групування завдань по категоріях, це корисно, тому що один проект може включати в себе декілька завдань з одією позначкою; Можливість надсилання завданнь у месенджер і назад (наприклад користувачу в Telegram), це зручно, тому що користувачі можіть відповідати напряму з месенджера в дане програмне забезпечення.

Отже всі поставлені задачі виконано, мету роботи досягнуто.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Aahmad, M. O., Dennehy, D., Conboy, K., & Oivo, M. (2018). Канбан у інженерії програмного забезпечення: систематичне картирування досліджень. Журнал систем і програмного забезпечення, 137, 96-113.

2. Вплив agile методологій на розробку ПЗ: [Електронний ресурс]. – Режим доступу: https://www.ccsenet.org/journal/index.php/cis/article/view/44383/

3. Agile маніфест: [Електронний ресурс]. – Режим доступу: https://agilemanifesto.org/

4. 12 принципів Agile: [Електронний ресурс]. – Режим доступу: https://agilemanifesto.org/principles.html

5. Найкращі інструменти для спільної роботи у 2021 році: [Електронний ресурс]. – Режим доступу: https://blog.jetbrains.com/space/2021/07/16/best-collaboration-tools/

6. Scrum управління проектами: [Електронний ресурс]. – Режим доступу: https://www.digite.com/agile/scrum-methodology/

7. Scrum Guide: [Електронний ресурс]. – Режим доступу: https://scrumguides.org/

8. Ahmad, M. O., Markkula, J., & Oivo, M. (2016). Уявлення про переваги Канбану в програмних компаніях: погляд практиків, в: Sharp, H., & Hall, T. (Ред.), Міжнародна конференція з розробки Agile-програмного забезпечення. Чам: Springer International Publishing.

9. Ahmad, M. O., Markkula, J., Oivo, M., & Kuvaja, P. (2014). Використання Канбану в програмних компаніях: емпіричне дослідження мотивації, переваг і викликів. У: 9-та міжнародна конференція з розробки програмного забезпечення.

10. Al-Baik, O., & Miller, J. (2015). Підхід Канбан між гнучкістю і ефективністю: систематичний огляд. Емпірична інженерія програмного забезпечення, 20(6), 1861-1897.

11. Al-Baik, O., Miller, J., & Greening, D. (2017). Напрямки інноваційного підходу до перевірки Lean-розробки продукту. У: Матеріали 50-ї Гавайської

міжнародної конференції з системних наук, США, с. 5879-5888.

12. Alaidaros, H., & Omar, M. (2017). Підходи управління програмними проектами для контролю незавершених робіт: огляд. Журнал інженерних і прикладних наук, 12(15), 3851-3857.

13. Alaidaros, H., Omar, M., & Romli, R. (2018a). Визначення критеріїв, що впливають на задачу моніторингу програмного проекту Agile Kanban. У: Збірник матеріалів конференції AIP. Пенанг, Малайзія: AIP Publishing.

14. Flora, H. K., & Chande, S. V. (2014). Систематичне дослідження методологій та практик розробки програмного забезпечення Agile. Міжнародний журнал інформаційних технологій та комп'ютерних наук, 5(3), 3626-3637.

15. Global Project Management (2018). Успіх у період руйнувань: 10-е глобальне дослідження управління проектами. Ньютон-Сквер: Project Management Institute, Inc.

16. Guckenheimer, S., & Loje, N. (2012). Гнучке програмне інженерія з використанням Visual Studio: від концепції до постійного зворотного зв'язку. Addison-Wesley Professional.

17. Ikonen, M., Pirinen, E., Fagerholm, F., Kettunen, P., & Abrahamsson, P. (2011). Вплив системи Kanban на роботу над проектом програмного забезпечення: емпіричне дослідження на основі випадку. В: 16-та Міжнародна конференція IEEE з інженерії складних комп'ютерних систем (ICECCS), 2011 р., Лас-Вегас, Невада, США. IEEE), 305-314.

18. Kurita, T., Matsuo, K., & Barolli, L. (2020). Система управління інвалідними візками з використанням датчиків Інтернету речей та Agile-Kanban. У: Бароллі, Л., Нішино, Х., і Міва, Х. (Ред.), 11-та міжнародна конференція з інтелектуальних мереж та спільних систем (INCoS-2019). Ойта Університет, Ойта, Японія: Springer.

19. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів спеціальності 122 «Комп’ютерні науки» [Електронний ресурс] / уклад.: А. А. Яровий, О. В. Сілагін, І. В. Богач. – Вінниця : ВНТУ, 2023. – (PDF, 54 с.).

ДОДАТОК А (обов'язковий)
ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Розробка чат-боту для управління персоналом

Тип роботи: бакалаврська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ кафедра комп'ютерних наук
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 14,89 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

✓ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

Яровий А.А., зав. каф. КН _____
(прізвище, ініціали, посада) (підпис)

Колесницький О.К., проф. каф КН _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Озеранський В.С.
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Озеранський В.С.
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Вінтонюк В.В.
(підпис) (прізвище, ініціали)

ДОДАТОК Б (довідниковий)
Інструкція користувача

Для того, щоб знайти бота потрібно натиснути на кнопку пошуку у верхньому правому кутку і після цього відкриється вікно пошуку (рис. Б.1).

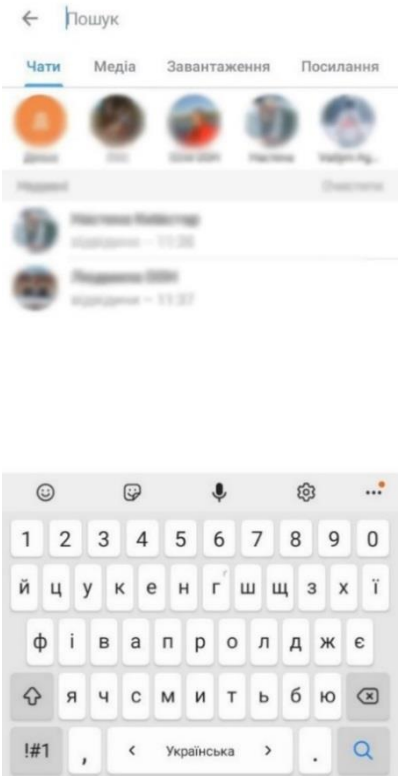

Рисунок Б.1 – Відкриття вікно пошуку у Telegram месенджері

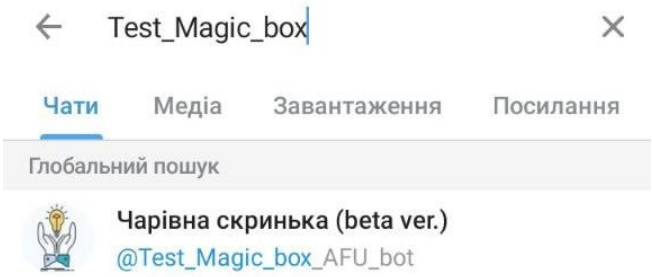

Рисунок Б.2 – Пошук бота за лінком

Коли користувач заходить в самого бота, але поки не запускаючи його, він бачить початкове повідомлення, для чого саме цей бот потрібен (рис. Б.3).

Рисунок Б.3 – Стартове повідомлення від бота

Після цього, потрібно натиснути на кнопку «Розпочати», що знаходиться у низу екрану. Одразу після цього користувач бачить це саме стартове повідомлення, та йому відкривається головне меню з кнопками, які означають категорії, за якими можна подавати ідею (рис. Б.4).

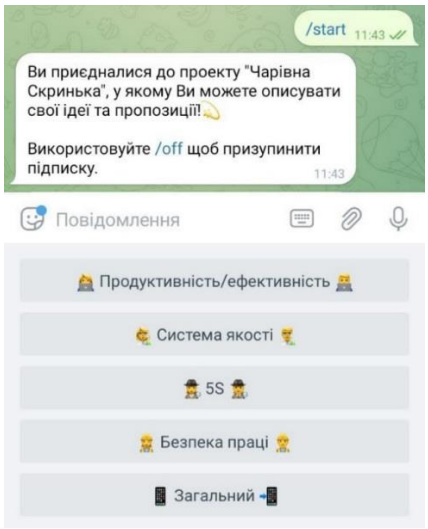

Рисунок Б.4 – Головне меню чат-бота

Після цього, користувач може обрати йому потрібну категорію, а у двох випадках ще і під категорії, задля того, щоб подати проблеми та ідеї для покращення свого робочого місця на підприємстві (рис. Б.5).

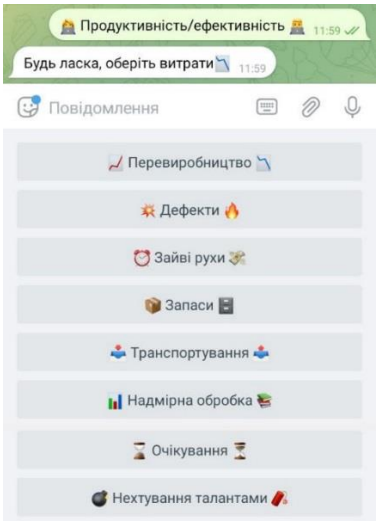

Рисунок Б.5 – Зображення підкатегорій категорії «Продкутивність\ефективність»

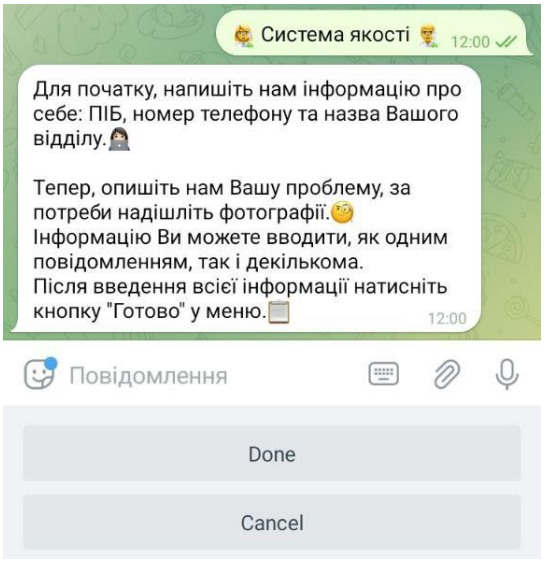

Рисунок Б.6 – Зображення категорії «Система якості»

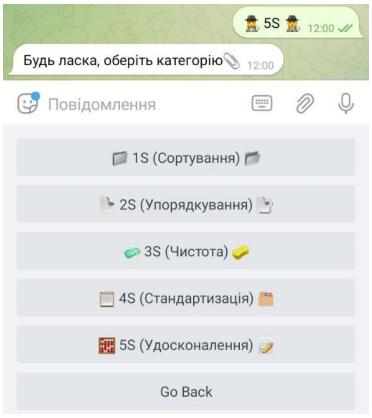

Рисунок Б.7 – Зображення підкатегорій категорії «5S»

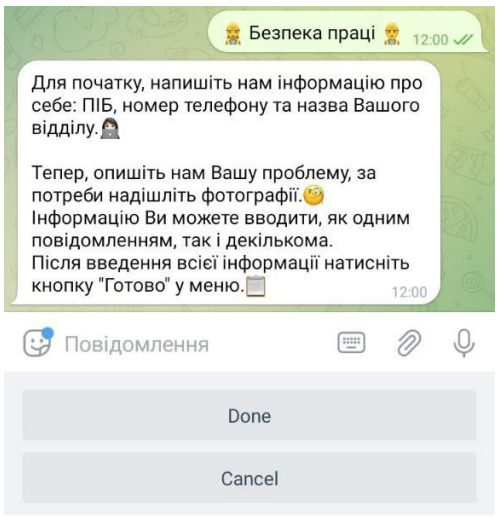

Рисунок Б.8 – Зображення категорії «Безпека праці»

Користувач обирає потрібну йому категорію. Після цього він пише своє ПІБ, номер телефону та назву відділу в якому працює на підприємстві. Після цього він описує проблему, яку він побачив на своєму робочому місці і також подає ідею її вирішення (рис. Б.9).

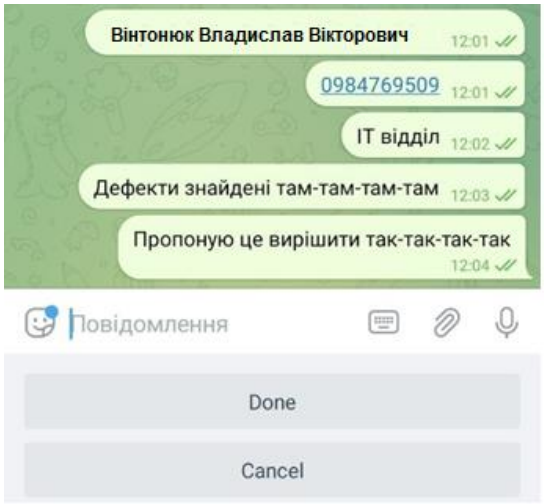

Рисунок Б.9 – Приклад подання проблеми та ідеї до чат-боту

Після цього користувач натискає на кнопку «Done» та ідея надсилається вже на сервер, де і буде далі оброблюватися адміністратором. Після завершення користувачеві знову відкривається початкове меню (рис. Б.10).

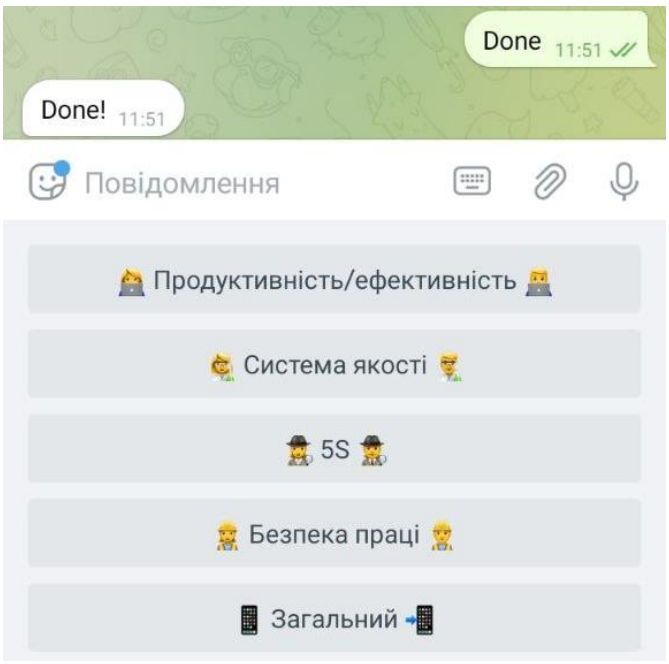

Рисунок Б.10 – Завершення подачі ідеї та повернення на головне меню

ДОДАТОК В (обов'язковий)

Лістинг програми

```
import firebase_admin
from firebase_admin import credentials
from firebase_admin import db
from firebase_admin import storage
import telebot
from telebot import types
from token_ import token_


TOKEN = '5591722517:AAEd9X1_NB3eacAFC9W6MZQbN8v-kDZVKtM'

bot = telebot.TeleBot(TOKEN)

cred = credentials.Certificate("/path/to/secret/key.json")
default_app = firebase_admin.initialize_app(cred, {
        'databaseURL': 'https://realtime-db-name',
    'storageBucket' : 'storage-bucket-local-name'
      })
bucket = storage.bucket()


@bot.message_handler(commands=['start', 'help'])
def send_welcome(message):
      #bot.reply_to(message, "Привіт, {0.first_name}!".format(message.from_user))

    markup = types.ReplyKeyboardMarkup(resize_keyboard=True, row_width = 2)
    item1 = types.KeyboardButton('👩‍💻 Продуктивність/ефективність 👨‍💻')
    item2 = types.KeyboardButton('👩‍🔬 Система якості 👨‍🔬')
    item3 = types.KeyboardButton('👮‍♀️ 5S 👮‍♂️')
    item4 = types.KeyboardButton('⛑️♀️ Безпека праці ⛑️♂️')
    item5 = types.KeyboardButton('📱 Загальний 📲')
    markup.row(item1)
    markup.row(item2)
    markup.row(item3)
    markup.row(item4)
    markup.row(item5)
    bot.send_message(message.chat.id, 'Привіт, {0.first_name}! Ви приєдналися до
проекту "Чарівна Скринька", у якому Ви можете описувати свої ідеї та
пропозиції!🤝'.format(message.from_user), reply_markup=markup)

@bot.message_handler(content_types=['text'])
def bot_message(message):
    if message.text == '👩‍💻 Продуктивність/ефективність 👨‍💻':
        markup = types.ReplyKeyboardMarkup(resize_keyboard=True, row_width = 2)
        item1 = types.KeyboardButton('📈 Перевиробництво 📉')
        item2 = types.KeyboardButton('💥 Дефекти 🔥')
        item3 = types.KeyboardButton('⏰ Зайві рухи 🚫')
        item4 = types.KeyboardButton('📦 Запаси 🗄️')
        item5 = types.KeyboardButton('🚚 Транспортування 🚚')
        item6 = types.KeyboardButton('📊 Надмірна обробка 🛤️')
```

```
        item7 = types.KeyboardButton('⏳ Очікування ⏳')
        item8 = types.KeyboardButton('💣 Нехтування талантами ')
        back = types.KeyboardButton('Назад')
        markup.row(item1)
        markup.row(item2)
        markup.row(item3)
        markup.row(item4)
        markup.row(item5)
        markup.row(item6)
        markup.row(item7)
        markup.row(item8)
        markup.row(back)
        bot.send_message(message.chat.id, 'Будь ласка, оберіть витрати📉',
reply_markup=markup)

    elif message.text == '👩🔬 Система якості 👨🔬':
        markup = types.ReplyKeyboardMarkup(resize_keyboard=True, row_width = 2)
        item1 = types.KeyboardButton('Done')
        item2 = types.KeyboardButton('Cancel')
        markup.row(item1)
        markup.row(item2)
        bot.send_message(message.chat.id, 'Для початку, напишіть нам інформацію про
себе: ПІБ, номер телефону та назва Вашого відділу.👩🏻💻\n Тепер, опишіть нам Вашу
проблему, за потреби надішліть фотографії. Інформацію Ви можете вводити, як одним
повідомленням, так і декількома. Після введення всієї інформації натисніть кнопку
"Готово" у меню.', reply_markup=markup)

    elif message.text == '🕵♀ 5S 🕵♂':
        markup = types.ReplyKeyboardMarkup(resize_keyboard=True, row_width = 2)
        item1 = types.KeyboardButton('🗁 1S (Сортування) 🗁')
        item2 = types.KeyboardButton('📄 2S (Упорядкування) 📄')
        item3 = types.KeyboardButton(' 3S (Чистота) ')
        item4 = types.KeyboardButton('📋 4S (Стандартизація) 🗀')
        item5 = types.KeyboardButton(' 5S (Удосконалення) 📝')
        back = types.KeyboardButton('Назад')
        markup.row(item1)
        markup.row(item2)
        markup.row(item3)
        markup.row(item4)
        markup.row(item5)
        markup.row(back)
        bot.send_message(message.chat.id, 'Будь ласка, оберіть категорію',
reply_markup=markup)

    elif message.text == '👷♀ Безпека праці 👷♂':
        markup = types.ReplyKeyboardMarkup(resize_keyboard=True, row_width = 2)
        item1 = types.KeyboardButton('Done')
        item2 = types.KeyboardButton('Cancel')
        markup.row(item1)
        markup.row(item2)
        bot.send_message(message.chat.id, 'Для початку, напишіть нам інформацію про
себе: ПІБ, номер телефону та назва Вашого відділу.👩🏻💻\n Тепер, опишіть нам Вашу
проблему, за потреби надішліть фотографії. Інформацію Ви можете вводити, як одним
повідомленням, так і декількома. Після введення всієї інформації натисніть кнопку
```

```
"Готово" у меню.', reply_markup=markup)

    elif message.text == '📄 Загальний 📄':
        markup = types.ReplyKeyboardMarkup(resize_keyboard=True, row_width = 2)
        item1 = types.KeyboardButton('Done')
        item2 = types.KeyboardButton('Cancel')
        markup.row(item1)
        markup.row(item2)
        bot.send_message(message.chat.id, 'Для початку, напишіть нам інформацію про
себе: ПІБ, номер телефону та назва Вашого відділу.👩‍💻\n Тепер, опишіть нам Вашу
проблему, за потреби надішліть фотографії.📷 Інформацію Ви можете вводити, як одним
повідомленням, так і декількома. Після введення всієї інформації натисніть кнопку
"Готово" у меню.', reply_markup=markup)

@bot.message_handler(content_types=['text_info'])
def bot_message(msg):
    if msg.text_info == '📈 Перевиробництво 📉':
        markup = types.ReplyKeyboardMarkup(resize_keyboard=True, row_width = 2)
        item1 = types.KeyboardButton('Done')
        item2 = types.KeyboardButton('Cancel')
        markup.row(item1)
        markup.row(item2)
        bot.send_message(msg.chat.id, 'Для початку, напишіть нам інформацію про
себе: ПІБ, номер телефону та назва Вашого відділу.👩‍💻\n Тепер, опишіть нам Вашу
проблему, за потреби надішліть фотографії.📷 Інформацію Ви можете вводити, як одним
повідомленням, так і декількома. Після введення всієї інформації натисніть кнопку
"Готово" у меню.', reply_markup=markup)

    elif msg.text_info == '🕸️ Дефекти 🔥':
        markup = types.ReplyKeyboardMarkup(resize_keyboard=True, row_width = 2)
        item1 = types.KeyboardButton('Done')
        item2 = types.KeyboardButton('Cancel')
        markup.row(item1)
        markup.row(item2)
        bot.send_message(msg.chat.id, 'Для початку, напишіть нам інформацію про
себе: ПІБ, номер телефону та назва Вашого відділу.👩‍💻\n Тепер, опишіть нам Вашу
проблему, за потреби надішліть фотографії.📷 Інформацію Ви можете вводити, як одним
повідомленням, так і декількома. Після введення всієї інформації натисніть кнопку
"Готово" у меню.', reply_markup=markup)
    elif msg.text_info == '⏰ Зайві рухи 🩹':
        markup = types.ReplyKeyboardMarkup(resize_keyboard=True, row_width = 2)
        item1 = types.KeyboardButton('Done')
        item2 = types.KeyboardButton('Cancel')
        markup.row(item1)
        markup.row(item2)
        bot.send_message(msg.chat.id, 'Для початку, напишіть нам інформацію про
себе: ПІБ, номер телефону та назва Вашого відділу.👩‍💻\n Тепер, опишіть нам Вашу
проблему, за потреби надішліть фотографії.📷 Інформацію Ви можете вводити, як одним
повідомленням, так і декількома. Після введення всієї інформації натисніть кнопку
"Готово" у меню.', reply_markup=markup)
    elif msg.text_info == '📂 Запаси 🗄️':
        markup = types.ReplyKeyboardMarkup(resize_keyboard=True, row_width = 2)
        item1 = types.KeyboardButton('Done')
        item2 = types.KeyboardButton('Cancel')
```

```
        markup.row(item1)
        markup.row(item2)
        bot.send_message(msg.chat.id, 'Для початку, напишіть нам інформацію про
себе: ПІБ, номер телефону та назва Вашого відділу.👩🏻💻\n Тепер, опишіть нам Вашу
проблему, за потреби надішліть фотографії.📷 Інформацію Ви можете вводити, як одним
повідомленням, так і декількома. Після введення всієї інформації натисніть кнопку
"Готово" у меню.', reply_markup=markup)
    elif msg.text_info == '📥 Транспортування 📤':
        markup = types.ReplyKeyboardMarkup(resize_keyboard=True, row_width = 2)
        item1 = types.KeyboardButton('Done')
        item2 = types.KeyboardButton('Cancel')
        markup.row(item1)
        markup.row(item2)
        bot.send_message(msg.chat.id, 'Для початку, напишіть нам інформацію про
себе: ПІБ, номер телефону та назва Вашого відділу.👩🏻💻\n Тепер, опишіть нам Вашу
проблему, за потреби надішліть фотографії.📷 Інформацію Ви можете вводити, як одним
повідомленням, так і декількома. Після введення всієї інформації натисніть кнопку
"Готово" у меню.', reply_markup=markup)
    elif msg.text_info == '📊 Надмірна обробка 🗄':
        markup = types.ReplyKeyboardMarkup(resize_keyboard=True, row_width = 2)
        item1 = types.KeyboardButton('Done')
        item2 = types.KeyboardButton('Cancel')
        markup.row(item1)
        markup.row(item2)
        bot.send_message(msg.chat.id, 'Для початку, напишіть нам інформацію про
себе: ПІБ, номер телефону та назва Вашого відділу.👩🏻💻\n Тепер, опишіть нам Вашу
проблему, за потреби надішліть фотографії.📷 Інформацію Ви можете вводити, як одним
повідомленням, так і декількома. Після введення всієї інформації натисніть кнопку
"Готово" у меню.', reply_markup=markup)
    elif msg.text_info == '⏳ Очікування ⌛':
        markup = types.ReplyKeyboardMarkup(resize_keyboard=True, row_width = 2)
        item1 = types.KeyboardButton('Done')
        item2 = types.KeyboardButton('Cancel')
        markup.row(item1)
        markup.row(item2)
        bot.send_message(msg.chat.id, 'Для початку, напишіть нам інформацію про
себе: ПІБ, номер телефону та назва Вашого відділу.👩🏻💻\n Тепер, опишіть нам Вашу
проблему, за потреби надішліть фотографії.📷 Інформацію Ви можете вводити, як одним
повідомленням, так і декількома. Після введення всієї інформації натисніть кнопку
"Готово" у меню.', reply_markup=markup)
    elif msg.text_info == '💣 Нехтування талантами 📷':
        markup = types.ReplyKeyboardMarkup(resize_keyboard=True, row_width = 2)
        item1 = types.KeyboardButton('Done')
        item2 = types.KeyboardButton('Cancel')
        markup.row(item1)
        markup.row(item2)
        bot.send_message(msg.chat.id, 'Для початку, напишіть нам інформацію про
себе: ПІБ, номер телефону та назва Вашого відділу.👩🏻💻\n Тепер, опишіть нам Вашу
проблему, за потреби надішліть фотографії.📷 Інформацію Ви можете вводити, як одним
повідомленням, так і декількома. Після введення всієї інформації натисніть кнопку
"Готово" у меню.', reply_markup=markup)
    elif msg.text_info == 'Назад':
        bot.send_message(msg.chat.id, 'Назад')
                    bot.infinity_polling()
```

ДОДАТОК Г (обов'язковий)

ГРАФІЧНА ЧАСТИНА

«РОЗРОБКА ЧАТ-БОТУ ДЛЯ УПРАВЛІННЯ ПЕРСОНАЛОМ»

Виконав: студент 4 курсу, групи 1КН-21б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Вінтонюк В.В.
(прізвище та ініціали)
Керівник: к.т.н., доцент кафедри КН
Озеранський В.С.
(прізвище та ініціали)

Вінниця ВНТУ - 2025 рік

Рисунок Г.1 – Існуючі технології управління персоналом

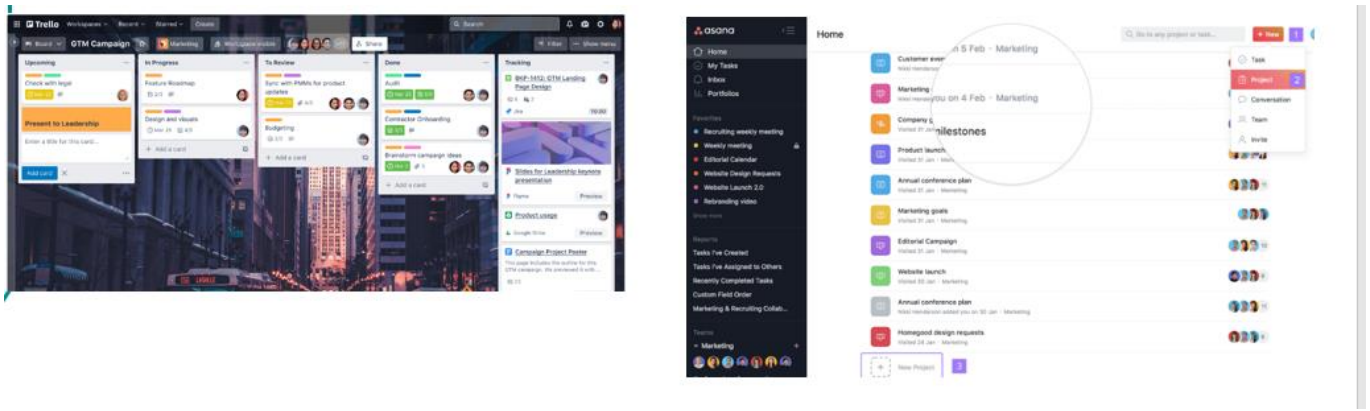

Рисунок Г.2 – Програми аналоги для управління персоналом

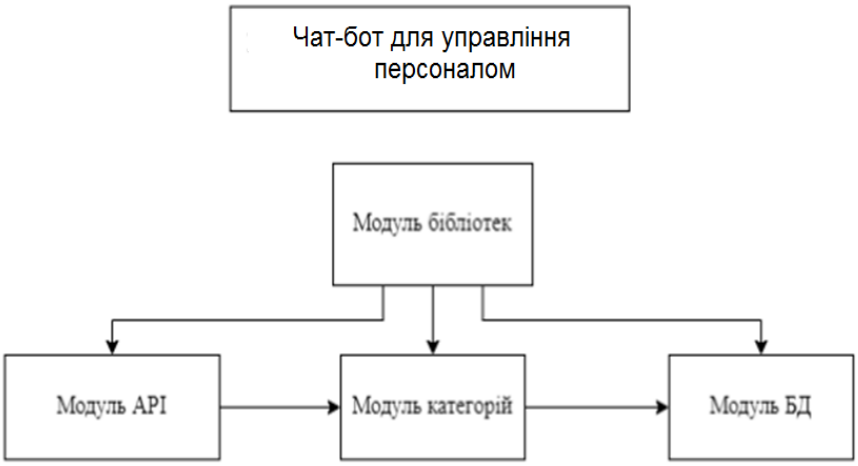

Рисунок Г.3 – Структурна схема чат-боту для управління персоналом

Рисунок Г.4 – Схема алгоритму функціонування чат-боту для управління персоналом

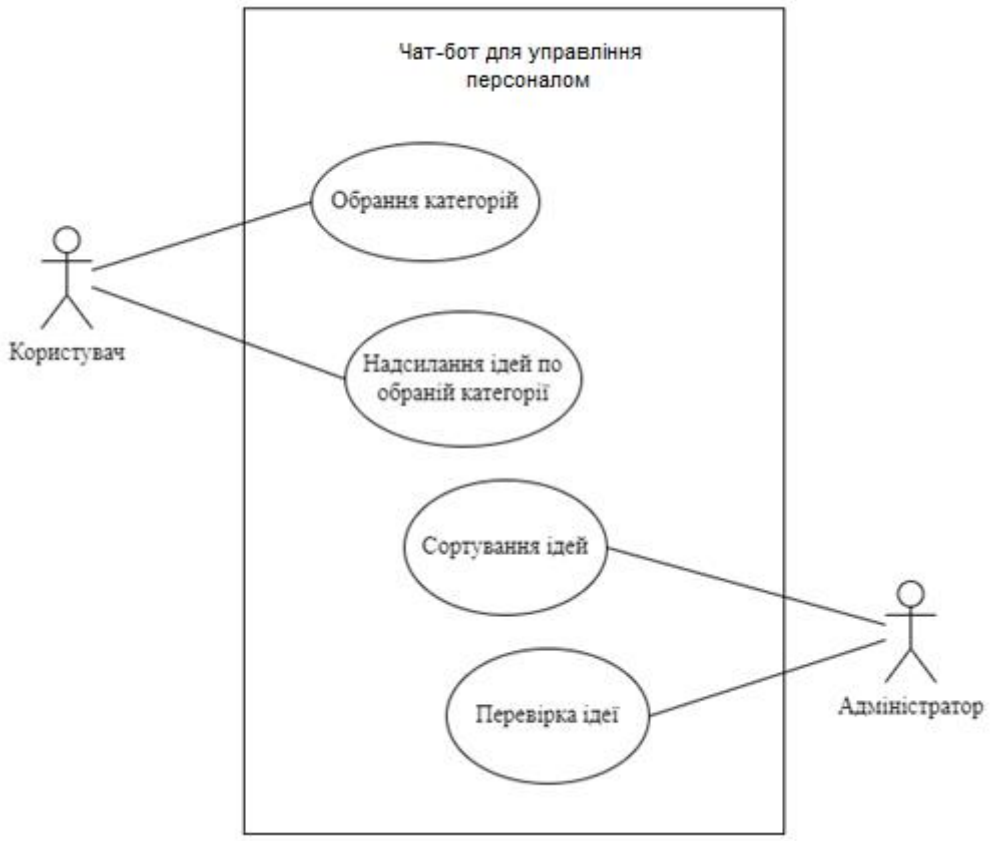

Рисунок Г.5 – Діаграма прецедентів чат-боту для управління проектами

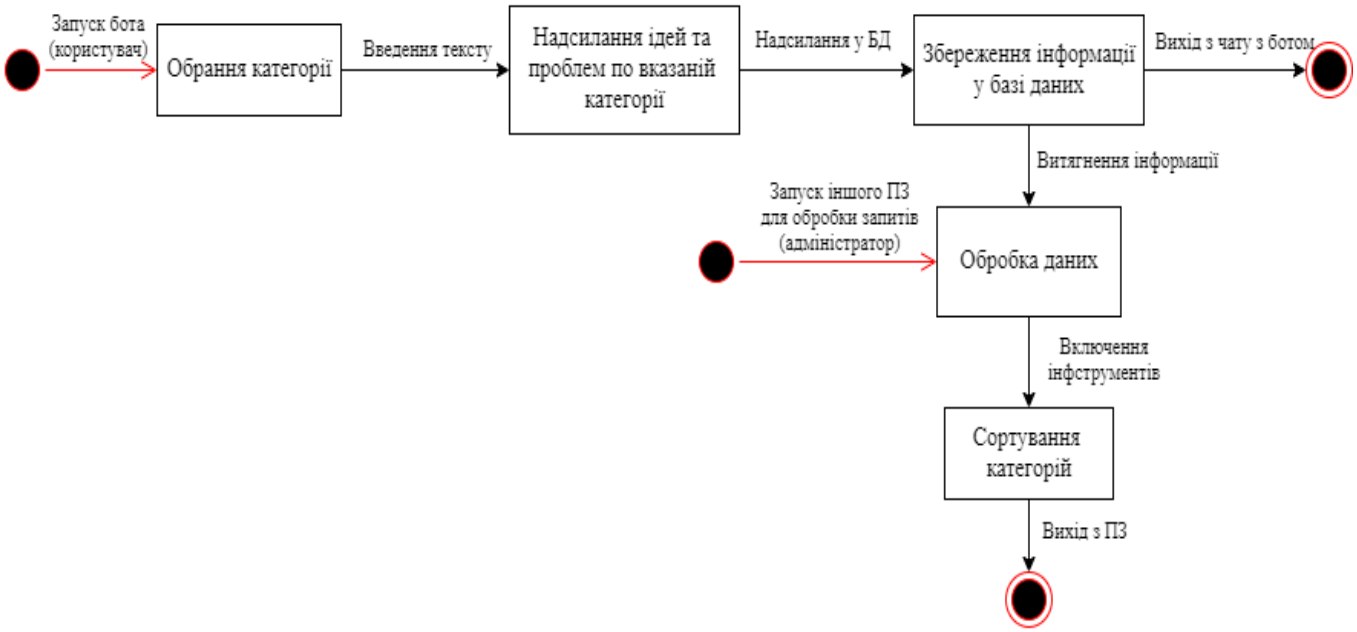

Рисунок Г.6 – Діаграма станів чат-боту для управління персоналом

Рисунок Г.7 – Інтерфейс чат-боту для управління персоналом