

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Інформаційних управляючих систем
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)
Дослідження моделей життєвого циклу ІТ-проектів розроблення LLM-чатботів
для корпоративних систем у Microsoft Teams
(тема)

Виконав:

здобувач 2 року навчання,
групи УПГІТМ-24-1
Денис ГАЛЮК
(власне ім'я, прізвище)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Управління проектами в галузі інформаційних технологій

Керівник проф. каф. ІУС Сергій ЧАЛИЙ
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ІУС


(підпис)

Костянтин ПЕТРОВ
(власне ім'я, прізвище)

2026_ р.

Харківський національний університет радіоелектроніки

Факультет _____ Комп'ютерних наук _____

Кафедра _____ Інформаційних управляючих систем _____

Рівень вищої освіти _____ другий (магістерський) _____

Спеціальність _____ 122 Комп'ютерні науки _____
(код і повна назва)

Тип програми _____ освітньо-наукова _____
(освітньо-професійна або освітньо-наукова)

Освітня програма _____ Управління проектами в галузі інформаційних технологій _____
(повна назва)

ЗАТВЕРДЖУЮ:

Зав.кафедри _____
(підпис)

« 06 » квітня 20 26 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві _____ Галюку Денису Юрійовичу _____
(прізвище, ім'я, по батькові)

Тема роботи _____ Дослідження моделей життєвого циклу IT-проектів розроблення LLM-чатботів для корпоративних систем у Microsoft Teams _____

затверджена наказом університету від _____ 23 _____ березня _____ 2026 р. № _____ 231Ст _____

2. Термін подання здобувачем роботи до екзаменаційної комісії _____ 19 _____ травня _____ 2026 р.

3. Вихідні дані до роботи _____ опис моделей життєвого циклу IT-проектів, особливості розроблення та впровадження LLM-чатботів для корпоративних систем у Microsoft Teams, вимоги до планування етапів проекту, аналізу ризиків, розподілу ресурсів і оцінювання ефективності реалізації. Предметна область дослідження – корпоративні інформаційні системи та автоматизація внутрішньої комунікації за допомогою LLM-чатботів. Технології та середовища: Microsoft Teams, Microsoft Azure, Azure OpenAI Service, Microsoft Bot Framework, C#, .NET 6.0, Visual Studio 2022. _____

4. Перелік питань, що потрібно опрацювати в роботі _____ розроблення LLM-чатботів для корпоративних систем у Microsoft Teams; дослідження та порівняння моделей життєвого циклу IT-проектів; визначення їх застосовності до проектів створення LLM-чатботів; аналіз архітектурних компонентів чатботів на платформі Microsoft Teams Bot Framework, методів розроблення запитів, технологій RAG та оцінювання якості відповідей; розроблення спеціалізованої моделі життєвого циклу для проектів створення LLM-чатботів; визначення фаз моделі, метрик якості, критеріїв переходу між етапами та підходів до управління ризиками; проектування рішення для впровадження LLM-чатбота в Microsoft Teams; планування строків, бюджету й ресурсів проекту; програмна реалізація основних модулів чатбота, бази знань, RAG-компонента та засобів моніторингу; проведення експериментальної перевірки, аналіз результатів тестування й оцінювання практичної цінності впровадження для підприємства. _____

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
	Отримання завдання	06.04.2026	виконано
	Аналіз предметної галузі і постановка задачі	07.04.2026	виконано
	Аналіз процесів розроблення LLM-чатботів для Microsoft Teams та постановка задачі дослідження	13.04.2026	виконано
	Розроблення моделі життєвого циклу IT-проектів створення LLM-чатботів дослідження	17.04.2026	виконано
	Розробка проєкту впровадження моделі життєвого циклу розроблення LLM-чатботів для корпоративних систем у Microsoft Teams публікація матеріалів	27.04.2026	виконано
	Програмна реалізація та експериментальна перевірка розробленого рішення дослідження	04.05.2026	виконано
	Підготовка пояснювальної записки	04.05.2026	виконано
	Підготовка презентації та доповіді	10.05.2026	виконано
	Перевірка на плагіат	13.05.2026	виконано
	Нормоконтроль	14.05.2026	виконано
	Рецензування	15.05.2026	виконано
	Попередній захист	16.05.2026	виконано
	Занесення диплома в електронний архів	17.05.2026	виконано
	Допуск до захисту у зав. кафедри	18.05.2026	виконано

Дата видачі завдання 06 квітня 2026 р.

Здобувач 
(підпис)

Керівник роботи  проф. каф. ІУС Сергій ЧАЛИЙ
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка містить: 133 с., 39 рис., 31 табл., 22 джерела, 1 додаток.

АВТОМАТИЗАЦІЯ РОЗГОРТАННЯ, ВЕЛИКІ МОВНІ МОДЕЛІ, ЖИТТЄВИЙ ЦИКЛ ІТ-ПРОЄКТУ, КОРПОРАТИВНИЙ ЧАТБОТ, MICROSOFT TEAMS, RAG.

Об'єктом дослідження є процес управління проєктами розроблення корпоративних чатботів на основі великих мовних моделей для Microsoft Teams.

Метою роботи є підвищення ефективності управління проєктами розроблення корпоративних чатботів шляхом створення спеціалізованого життєвого циклу та математичної моделі оцінювання його параметрів.

Методами розробки та проєктування є аналіз процесів управління ІТ-проєктами, порівняння моделей життєвого циклу розроблення програмного забезпечення, формалізація етапів спеціалізованої моделі, математичне моделювання параметрів життєвого циклу, проєктування архітектури LLM-чатбота для Microsoft Teams та експериментальна перевірка розробленого рішення.

У результаті кваліфікаційної роботи було розроблено спеціалізовану модель життєвого циклу ІТ-проєктів створення LLM-чатботів, математичну модель оцінювання складності, тривалості, якості, готовності та ризиків проєкту, а також проєкт впровадження корпоративного чатбота у Microsoft Teams. Виконано програмну реалізацію основних модулів чатбота, бази знань, RAG-компонента та засобів моніторингу, проведено експериментальну перевірку і підтверджено практичну цінність запропонованого рішення для корпоративного середовища.

ABSTRACT

The explanatory note contains: 133 pages, 39 figures, 31 tables, 22 sources, 1 appendix.

DEPLOYMENT AUTOMATION, LARGE LANGUAGE MODELS, IT PROJECT LIFE CYCLE, CORPORATE CHATBOT, MICROSOFT TEAMS, RAG.

The object of research is the process of managing projects for the development of corporate chatbots based on large language models for Microsoft Teams.

The purpose of the work is to improve the efficiency of managing corporate chatbot development projects by creating a specialized life cycle and a mathematical model for evaluating its parameters.

The development and design methods are the analysis of IT project management processes, comparison of software development life cycle models, formalization of the stages of a specialized model, mathematical modeling of life cycle parameters, design of the LLM chatbot architecture for Microsoft Teams, and experimental verification of the developed solution.

As a result of the qualification work, a specialized life cycle model for IT projects aimed at creating LLM chatbots, a mathematical model for evaluating project complexity, duration, quality, readiness and risks, as well as a project for implementing a corporate chatbot in Microsoft Teams were developed. The software implementation of the main chatbot modules, knowledge base, RAG component and monitoring tools was performed, experimental verification was conducted, and the practical value of the proposed solution for the corporate environment was confirmed.

ЗМІСТ

	С.
Скорочення та умовні позначки	7
Вступ	9
1 Аналіз процесів розроблення чатботів для Microsoft Teams і постановка задачі	11
1.1 Аналіз процесів управління ІТ-проєктами розроблення LLM-чатботів	11
1.2 Аналіз моделей життєвого циклу ІТ-проєктів розроблення програмного забезпечення.....	17
1.3 Аналіз методів розроблення LLM-чатботів для Microsoft Teams.....	26
1.4 Постановка задачі дослідження	35
2 Розроблення моделі життєвого циклу ІТ-проєктів створення LLM-чатботів	38
2.1 Концептуальна модель життєвого циклу LLM-чатботів	38
2.2 Адаптація моделі до різних типів проєктів розроблення LLM-чатботів	46
2.3 Версіонування та оцінювання запитів для LLM-чатботів.....	50
3 Розробка проєкту впровадження моделі життєвого циклу розроблення LLM-чатботів для корпоративних систем у Microsoft Teams	55
3.1 Ініціалізація та концепція проєкту.....	55
3.2 Життєвий цикл проєкту впровадження.....	62
3.3 Проєктування рішення.....	70
3.4 Планування часу, вартості та ризиків.....	78
3.5 Ефективність та результати проєктного проєктування	88
4 Програмна реалізація та експериментальна перевірка розробленого рішення ...	92
4.1 Програмна реалізація	92
4.2 Експериментальна перевірка.....	103
4.3 Аналіз результатів і практична цінність	112
Висновки.....	116
Перелік джерел посилання	119
Додаток А. Графічний матеріал кваліфікаційної роботи	122

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ДСТУ – державний стандарт України

ІС – інформаційна система

ІТ – інформаційні технології

ІУС – інформаційні управляючі системи

ХНУРЕ – Харківський національний університет радіоелектроніки

AD – Active Directory, служба каталогів

AI – Artificial Intelligence, штучний інтелект

API – Application Programming Interface, інтерфейс програмування застосунків

BA – Business Analyst, бізнес-аналітик

CAPEX – Capital Expenditures, капітальні витрати

CI/CD – Continuous Integration / Continuous Delivery, безперервна інтеграція та безперервне постачання

DB – Database, база даних

DevOps – Development and Operations, підхід до поєднання розроблення та експлуатації програмного забезпечення

FAQ – Frequently Asked Questions, часто поставлені питання

GDPR – General Data Protection Regulation, Загальний регламент захисту даних

GPT – Generative Pre-trained Transformer, генеративний попередньо навчений трансформер

HR – Human Resources, управління персоналом

IEC – International Electrotechnical Commission, Міжнародна електротехнічна комісія

ISO – International Organization for Standardization, Міжнародна організація зі стандартизації

IT – Information Technology, інформаційні технології

KPI – Key Performance Indicators, ключові показники ефективності

LLM – Large Language Model, велика мовна модель

M365 – Microsoft 365

MLOps – Machine Learning Operations, підхід до розроблення, розгортання та супроводження систем машинного навчання

NIST – National Institute of Standards and Technology, Національний інститут стандартів і технологій США

NLP – Natural Language Processing, оброблення природної мови

OPEX – Operational Expenditures, операційні витрати

PDF – Portable Document Format, переносний формат документа

PM – Project Manager, керівник проєкту

QA – Quality Assurance, забезпечення якості

RACI – Responsible, Accountable, Consulted, Informed, матриця розподілу відповідальності

RAG – Retrieval-Augmented Generation, генерування з розширеним пошуком

REST – Representational State Transfer, архітектурний стиль взаємодії програмних систем

RMF – Risk Management Framework, рамкова модель управління ризиками

ROI – Return on Investment, рентабельність інвестицій

SDK – Software Development Kit, комплект засобів розроблення програмного забезпечення

UI – User Interface, інтерфейс користувача

UML – Unified Modeling Language, уніфікована мова моделювання

URL – Uniform Resource Locator, уніфікований покажчик ресурсу

UX – User Experience, користувацький досвід

WBS – Work Breakdown Structure, ієрархічна структура робіт

XP – Extreme Programming, екстремальне програмування

ВСТУП

Стрімкий розвиток великих мовних моделей відкриває принципово нові можливості для автоматизації корпоративних процесів через інтелектуальні чатботи, інтегровані у середовище Microsoft Teams (понад 280 мільйонів активних користувачів станом на 2024 рік). Такі чатботи здатні не лише відповідати на запити користувачів, але й виконувати складні бізнес-функції: автоматизувати обробку заявок, підтримувати внутрішні служби підтримки (Information Technology (IT), Human Resources (HR)), здійснювати пошук інформації у корпоративних базах знань, а також сприяти прийняттю управлінських рішень шляхом аналізу даних у реальному часі.

Водночас існуючі моделі життєвого циклу розроблення програмного забезпечення – каскадна, гнучка (Agile), DevOps [1] та Machine Learning Operations (MLOps) – не повною мірою враховують специфіку проєктів створення чатботів на основі великих мовних моделей. На відміну від традиційних систем, такі рішення потребують ітеративного проєктування та оптимізації запитів (prompt engineering), що стає ключовим етапом розроблення. Крім того, важливою складовою є інтеграція з корпоративними джерелами даних через архітектуру розширеного пошуку (Retrieval-Augmented Generation, RAG), яка дозволяє поєднати можливості генеративних моделей із актуальними внутрішніми знаннями організації.

Ще однією суттєвою відмінністю є необхідність безперервного оцінювання якості відповідей чатбота, включаючи точність, релевантність, безпечність та відповідність корпоративним політикам. Також значну роль відіграє управління витратами на обчислення, оскільки використання великих мовних моделей часто пов'язане з високими експлуатаційними витратами, що потребує оптимізації запитів і архітектури системи. Окрему увагу слід приділяти ризикам генерування некоректного або небажаного контенту, що може негативно вплинути на бізнес-процеси та репутацію організації.

За даними Gartner (2024), до 65% проєктів впровадження корпоративних рішень на основі штучного інтелекту не досягають виробничої стадії. Основною причиною цього є відсутність формалізованих методологій управління процесом розроблення, які б враховували специфіку роботи з великими мовними моделями, включаючи експериментальний характер розробки, залежність від якості даних та необхідність постійного моніторингу результатів.

Таким чином, дослідження та адаптація моделей життєвого циклу [2] розроблення для проєктів створення корпоративних чатботів на основі великих мовних моделей є актуальним науково-практичним завданням. Результатом такого дослідження може стати розроблення структурованої методології, що включатиме чітко визначені етапи, формалізовані критерії переходу між ними, механізми контролю якості та управління ризиками [3]. Це, у свою чергу, дозволить підвищити ефективність розроблення, знизити ймовірність невдачі проєктів і забезпечити успішне впровадження корпоративних чатботів у виробниче середовище.

1 АНАЛІЗ ПРОЦЕСІВ РОЗРОБЛЕННЯ LLM-ЧАТБОТІВ ДЛЯ MICROSOFT TEAMS ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз процесів управління IT-проєктами розроблення LLM-чатботів

Життєвий цикл проєкту створення чатбота на основі великих мовних моделей охоплює шість ключових етапів: концептуалізація, проєктування архітектури, розроблення, випробування, розгортання та супроводження. Кожен з етапів виконує важливу роль у забезпеченні якості, функціональності та стабільності системи:

- концептуалізація, де визначаються основні цілі та завдання чатбота, аналізуються потреби користувачів, формуються бізнес-вимоги та мета якої – окреслити функціонал чатбота, його роль у автоматизації комунікацій та інтеграцію з корпоративними процесами

- проєктування архітектури, під час чого створюється структура системи та визначаються компоненти, що взаємодіють між собою, а також обираються технології для реалізації та особлива увага приділяється інтеграції з корпоративними платформами (наприклад, Microsoft 365, Teams [4]) і забезпеченню безпеки обробки даних

- розроблення, що включає реалізацію чатбота відповідно до спроектованої архітектури, де застосовуються Microsoft Bot Framework [5] та сервіси Azure для інтеграції великих мовних моделей, а також налаштовується корпоративний пошук документів та результатом є робочий застосунок, готовий до тестування, разом із кодом та початковими інтегрованими компонентами;

- випробування, під час якого здійснюється перевірка функціональності чатбота, точності генерації відповідей та безпеки обробки даних, проводяться тестування за реальними і синтетичними сценаріями, навантажувальні тести та оцінка відповідності нормативним вимогам та виходом є звіти про тестування, виправлені помилки та підтвердження готовності системи до розгортання;

- розгортання, під час якого чатбот виводиться в продуктивне середовище, налаштовуються права доступу та інтеграції з корпоративними службами та виходом є функціональна система, доступна користувачам, готова до виконання запланованих сценаріїв взаємодії;

- супроводження, що включає підтримку стабільності роботи чатбота, оновлення моделей і функцій відповідно до змін корпоративних вимог та зворотного зв'язку користувачів і виходами є стабільно функціонуюча система, оновлена документація та звіти ефективності роботи чатбота.

Вхідними даними процесу створення чатбота на основі великих мовних моделей є бізнес-вимоги до автоматизації комунікацій. Ці вимоги визначають цілі та завдання чатбота, його основні функції та сценарії використання, а також критерії ефективності системи. Вони формують основу для планування проєкту та визначають очікуваний результат, яким має стати автоматизований помічник для внутрішніх і зовнішніх комунікацій організації:

- технічне завдання, що деталізує функціональні та нефункціональні вимоги до чатбота, містить опис інтеграцій з корпоративними сервісами, технологічних платформ, стандартів безпеки та інших параметрів, необхідних для коректної реалізації системи;

- специфіка корпоративної екосистеми Microsoft 365, що визначає допустимі способи інтеграції з Teams, Outlook, SharePoint та іншими службами, а також обмежує доступні методи взаємодії з API і хмарними сервісами, що потребує врахування при проєктуванні архітектури та розробленні функціоналу чатбота;

- корпоративні дані для розширеного пошуку, які використовуються для забезпечення контекстного пошуку та генерації релевантних відповідей користувачам, при роботі з якими обов'язково дотримуються політики безпеки та вимоги конфіденційності, включно з регламентами GDPR.

Всі ці вхідні дані створюють основу для подальших етапів життєвого циклу проєкту, забезпечуючи відповідність системи очікуванням бізнесу та технічним обмеженням корпоративної інфраструктури.

Процес створення чатбота на основі великих мовних моделей включає:

- проектування системи запитів, що передбачає визначення логіки обробки запитів користувачів, структури баз знань і сценаріїв взаємодії з чатботом та під час якої формуються правила маршрутизації запитів, налаштовуються алгоритми генерації відповідей та визначаються критерії їхньої релевантності;

- розроблення застосунку на Microsoft Bot Framework, що включає створення функціональних компонентів чатбота, інтеграцію з Teams та іншими корпоративними сервісами, а також підключення до необхідних API, забезпечує реалізацію архітектури, розробленої на попередньому етапі і формує робочий застосунок, готовий для тестування [6] та подальшої інтеграції;

- інтеграція зі службами Azure OpenAI [7], яка дозволяє забезпечити генерацію відповідей великими мовними моделями та використання інтелектуальних функцій чатбота, налаштувати моделі обробки природної мови, підключити сервіси для навчання і оптимізації відповідей, а також забезпечити взаємодія з корпоративними даними та внутрішніми базами знань;

- налаштування пошуку корпоративними документами, що передбачає підключення баз даних і файлів організації для забезпечення контекстного пошуку для надання релевантних відповідей на запити користувачів на основі актуальної інформації та забезпечення ефективного використання внутрішніх знань організації;

- забезпечення якості та випробування точності, що включає проведення функціональних, навантажувальних та інтеграційних тестів, оцінку правильності відповідей та відповідності системи вимогам безпеки, завдяки чому виявляються помилки та недоліки, вносяться необхідні зміни, а результати тестування фіксуються для контролю якості і прийняття рішення про розгортання системи.

Виходом процесу створення чатбота є функціональний застосунок, інтегрований у Microsoft Teams та готовий до використання користувачами для автоматизації комунікацій і пошуку корпоративної інформації. Крім того, результатом є документація архітектури та запитів, яка описує структуру системи, взаємодію компонентів та логіку обробки запитів.

У межах роботи розроблення системи розглядається як навчально-дослідницький проєкт, спрямований на аналіз процесів створення Large Language Model-чатбота (LLM), проєктування архітектури рішення та оцінювання можливостей його впровадження в корпоративному середовищі. Термін виконання проєкту варіюється від чотирьох до дванадцяти місяців залежно від складності задач та обсягу інтеграції. Розмір команди зазвичай складає від трьох до восьми осіб із необхідними навичками роботи з Azure [8], Python та обробленням природної мови. Також проєкт повинен відповідати регуляторним вимогам General Data Protection Regulation (GDPR) щодо оброблення корпоративних даних та дотримуватися внутрішніх політик безпеки організації.

Проєкти створення корпоративних чатботів на основі великих мовних моделей мають різноманітну природу, тому їх доцільно класифікувати за кількома основними ознаками.

За функціональним призначенням виділяються:

- чатботи, що автоматизують технічну підтримку першого рівня, забезпечуючи швидку відповідь на типові питання користувачів і зменшуючи навантаження на службу підтримки;
- чатботи-асистенти з кадрових питань, які допомагають новим співробітникам під час адаптації, надаючи інформацію про внутрішні процеси та політики компанії;
- чатботи для відділу продажів, що забезпечують швидкий доступ до продуктової бази знань, допомагають співробітникам у підборі пропозицій та підвищують ефективність комерційних процесів;
- системи, що здійснюють перевірку відповідності юридичним вимогам, наприклад для контролю правильності внутрішніх документів або дотримання регламентів і стандартів компанії.

Класифікація за функціональним призначенням дозволяє розробникам та керівникам проєктів чітко визначати роль чатбота в організації та формувати завдання відповідно до конкретних бізнес-потреб.

За архітектурною складністю корпоративні LLM-чатботи поділяються на прості, середньої складності та складні системи та наведені у таблиці 1.

Таблиця 1 – Архітектурна складність та функціонал корпоративних LLM-чатботів

Архітектурна складність	Функціональність
Прості чатботи	Обробка запитів через прямі виклики інтерфейсу програмування застосунків і зазвичай працюють з обмеженим набором функцій без глибокої інтеграції в корпоративні сервіси
Системи середньої складності	Інтегрують розширений пошук документів у SharePoint та інші внутрішні бази знань, що дозволяє надавати користувачам більш релевантні відповіді та розширює функціональні можливості.
Складні системи	Багатоагентні системи, які координують роботу кількох моделей для різних предметних областей, забезпечують одночасну обробку складних сценаріїв та інтегруються з численними корпоративними сервісами

Така класифікація допомагає визначати необхідні ресурси, складність розробки та підходи до тестування та супроводу системи.

Ще одним критерієм є масштаб впровадження чатботів, що зображений у таблиці 2.

Таблиця 2 – Критерій масштабу впровадження чатботів

Назва	Масштаб впровадження
Пілотні рішення	Розраховані на 10–50 користувачів і дозволяють оцінити працездатність системи на невеликому колі співробітників.
Підрозділові рішення	Охоплюють від 50 до 500 користувачів і застосовуються для автоматизації процесів конкретного відділу або бізнес-напрямку.
Загальнокорпоративні чатботи	Призначені для понад 500 співробітників і вимагають масштабованості, надійності та інтеграції з багатьма корпоративними сервісами одночасно.

Вибір масштабу впровадження впливає на планування ресурсів, архітектуру системи та процеси супроводу.

За типом інтеграції корпоративні чатботи можуть бути автономними і працювати лише в каналах Microsoft Teams, інтегрованими з бізнес-процесами через Power Automate, або вбудованими безпосередньо у корпоративний портал компанії. Автономні боти забезпечують швидкий старт і мінімальні залежності від інших сервісів, інтегровані рішення дозволяють автоматизувати внутрішні процеси та взаємодію з іншими системами, а вбудовані чатботи забезпечують глибоку інтеграцію з корпоративною інфраструктурою та є частиною загальної системи інформаційного середовища компанії.

Класифікація корпоративних LLM-чатботів за критеріями, детально описаними вище, дозволяє систематизувати підходи до їхнього створення, визначати технічні та організаційні вимоги, планувати ресурси і забезпечувати ефективне використання чатботів у корпоративному середовищі.

У процесі управління життєвим циклом проєктів створення AI-чатботів виникає низка специфічних проблем, що ускладнюють планування, контроль і забезпечення якості системи. Однією з основних складнощів є відсутність чітких критеріїв завершення фаз розроблення. Зокрема, незрозуміло, коли система запитів вважається завершеною, та яка мінімальна точність відповідей є достатньою для переходу до виробничого середовища. Така невизначеність ускладнює прийняття рішень про готовність системи та планування наступних етапів життєвого циклу.

Складність оцінювання трудомісткості розроблення запитів також зумовлена їх ітеративною природою. Процес створення та оптимізації запитів постійно коригується на основі тестування, зворотного зв'язку користувачів і зміни корпоративних даних, що робить складним прогнозування витрат часу та ресурсів на кожен етап проєкту.

Ризики погіршення якості мовної моделі виникають при зміні даних у базі знань для розширеного пошуку. Додавання або видалення документів, зміни у корпоративних даних або коригування запитів можуть непередбачувано впливати на точність відповідей чатбота, що потребує постійного моніторингу та тестування системи після кожної модифікації.

Проблеми версіонування запитів та можливості відкату стають критичними у випадках погіршення показників системи. Відсутність ефективних механізмів контролю версій і відновлення попередніх конфігурацій ускладнює управління якістю та підвищує ризики тривалих простоїв і втрати даних.

Недооцінювання часу, необхідного на безперервне оцінювання та цикли зворотного зв'язку за участю людини, призводить до накопичення помилок і зниження ефективності системи. Постійне залучення експертів для перевірки відповідей, аналізу помилок та внесення коригувань є невід'ємною частиною забезпечення високої якості AI-чатбота, але часто ігнорується при плануванні проекту.

Технічний борг формується через швидкі ітерації пілотної фази без належної архітектурної документації. Неврахування принципів проєктування і структурування системи на ранніх етапах ускладнює подальший розвиток чатбота, інтеграцію нових функцій і підтримку стабільності системи.

Особливу складність становить балансування між швидкістю розроблення та вимогами управління, що включають конфіденційність даних, виявлення упередженості моделей та забезпечення пояснюваності рішень AI [17]. Прискорене створення чатбота може призводити до порушення цих вимог, тоді як повне дотримання правил управління часто уповільнює процес розробки, що вимагає ретельного планування та контролю на всіх етапах життєвого циклу.

1.2 Аналіз моделей життєвого циклу IT-проєктів розроблення програмного забезпечення

Моделі життєвого циклу розроблення програмного забезпечення визначають послідовність дій та організацію процесів розробки, тестування та супроводу систем, що дозволяє забезпечити контроль якості, управління ризиками [9] та ефективне використання ресурсів.

Каскадні моделі є одними з найбільш традиційних і добре структурованих підходів до розробки програмного забезпечення. На першому етапі визначаються вимоги до програмного продукту, включно з функціональними, нефункціональними та технічними аспектами. Важливою особливістю цього етапу є детальне документування всіх очікувань і обмежень, оскільки подальші фази базуються на цих вихідних даних.

Після визначення вимог виконується фаза проєктування, під час якої розробляється архітектура системи, моделі даних, структурні схеми компонентів та інтерфейси. Ретельне проєктування дозволяє передбачити майбутні взаємодії компонентів, забезпечує логічну послідовність реалізації та знижує ризики помилок у подальших етапах розробки.

Реалізація включає безпосереднє програмування компонентів системи відповідно до створеної архітектури. На цьому етапі важливо дотримуватися стандартів кодування, забезпечувати інтеграцію компонентів та проводити модульне тестування для раннього виявлення помилок.

Випробування або тестування є наступною фазою і передбачає перевірку функціональності системи відповідно до визначених вимог. Тут проводяться різні типи тестів: функціональні, інтеграційні та системні, що дозволяє оцінити працездатність системи та підготувати її до розгортання у виробничому середовищі.

Розгортання включає підготовку середовища для експлуатації, встановлення програмного забезпечення на сервери або робочі станції користувачів та налаштування системи для стабільної роботи. Після цього починається фаза супроводження, що передбачає усунення помилок, оновлення системи та підтримку її працездатності протягом всього періоду використання.

Особливою формою каскадної моделі є V-подібна модель, яка відрізняється тим, що на кожній фазі проєктування паралельно планується верифікація та валідація відповідних компонентів. Це означає, що для кожного етапу створюються тестові сценарії і критерії прийнятності, що дозволяє виявляти помилки раніше і підвищує якість кінцевого продукту. V-подібна модель забезпечує тісний зв'язок

між фазами розробки та контролем якості, що робить її ефективною для великих і критично важливих проєктів.

Ітеративні моделі життєвого циклу розроблення програмного забезпечення передбачають багаторазове повторення фаз розробки, що дозволяє поступово вдосконалювати систему та враховувати зміни вимог.

Однією з найпоширеніших є спіральна модель, яка будується на циклах, що включають аналіз ризиків, прототипування та оцінювання результатів. На кожному циклі проводиться детальний аналіз потенційних ризиків проєкту, визначаються критичні елементи та обмеження, що можуть вплинути на успіх розробки. Після цього створюється прототип системи, який дозволяє оцінити ключові функціональні можливості та отримати зворотний зв'язок від користувачів та замовників. На етапі оцінювання проводиться аналіз результатів прототипу, виявляються проблемні місця та уточнюються вимоги до наступного циклу.

Такий підхід дозволяє мінімізувати ризики помилок, своєчасно виявляти недоліки та адаптувати систему до нових потреб, що особливо важливо для складних або інноваційних проєктів.

Ще однією поширеною ітеративною моделлю є інкрементальна модель, яка передбачає поетапне нарощування функціональності програмного продукту. На першому етапі створюється базова версія системи з мінімальною, але повністю робочою функціональністю. Подальші етапи включають поступове додавання нових модулів, функцій та компонентів, що розширюють можливості системи. Такий підхід дозволяє отримати перший робочий результат вже на ранніх стадіях проєкту, зменшити ризики великих помилок у фінальній версії та залучити користувачів до оцінки проміжних результатів і надання зворотного зв'язку для покращення продукту.

Ітеративні моделі є ефективними у випадках, коли вимоги до системи не визначені остаточно на початку проєкту або коли важливо постійно перевіряти ключові функції системи на практиці. Вони забезпечують гнучкість розробки, дозволяють краще контролювати ризики та підвищують якість кінцевого продукту за рахунок регулярної перевірки та адаптації протягом всього життєвого циклу.

Гнучкі методології, або Agile-підходи, спрямовані на забезпечення швидкої адаптації до змін у вимогах, тісну взаємодію з користувачами та ефективне управління процесом розробки програмного забезпечення:

Scrum - передбачає організацію роботи у двотижневих спринтах. Кожен спринт починається з планування завдань, виконується розробка функціональності, після чого проводиться демонстрація результатів та ретроспектива для оцінки виконаної роботи і визначення шляхів покращення процесів на наступних циклах. Такий підхід дозволяє команді швидко реагувати на зміни вимог, оцінювати проміжні результати та покращувати якість продукту на кожному етапі;

Kanban - орієнтована на безперервний потік завдань та обмеження кількості незавершеної роботи. Цей підхід дозволяє командам ефективно управляти завантаженням, визначати пріоритети та забезпечує постійний контроль процесу розробки;

Екстремальне програмування (Extreme Programming, XP) робить акцент на технічних практиках розробки через тестування, парне програмування, безперервне інтегрування та регулярне рефакторування коду. Основна мета XP – підвищити якість програмного забезпечення та зменшити кількість помилок за рахунок систематичного тестування та постійного удосконалення архітектури продукту. Такий підхід особливо ефективний у складних і критично важливих проєктах, де важлива стабільність, надійність та висока якість коду.

Життєвий цикл DevOps є сучасним підходом до розробки програмного забезпечення, який поєднує розробку та експлуатацію систем у єдиний процес, що дозволяє прискорити випуск продукту та підвищити його стабільність. Основою цього підходу є конвеєр безперервної інтеграції та постачання, який автоматизує ключові етапи розробки: збирання коду, випробування та розгортання. Автоматизація цих процесів дозволяє швидко і безпечно інтегрувати зміни, знижує ймовірність помилок та забезпечує стабільну роботу програмного забезпечення у виробничому середовищі.

Інфраструктура як код є ще однією важливою складовою DevOps. Вона передбачає автоматизоване створення та управління середовищами розробки,

тестування та виробництва за допомогою конфігураційних файлів та скриптів. Такий підхід дозволяє гарантувати відтворюваність середовищ, зменшити ручні помилки та пришвидшити процес розгортання нових версій системи.

Безперервний моніторинг у життєвому циклі DevOps забезпечує постійне відстеження працездатності системи, продуктивності та безпеки. Він дозволяє виявляти проблеми на ранніх етапах, оперативно реагувати на помилки та забезпечує стабільність роботи програмного забезпечення навіть при великій кількості користувачів або швидких змінах у системі.

Завдяки поєднанню автоматизації процесів, інтеграції та безперервного моніторингу життєвий цикл DevOps дозволяє організаціям прискорювати розробку програмного забезпечення, підвищувати його якість, зменшувати ризики та ефективно керувати інфраструктурою на всіх етапах життєвого циклу продукту.

Життєвий цикл MLOps є специфічним підходом до розробки та експлуатації систем на основі машинного навчання і відрізняється від класичних моделей розробки програмного забезпечення тим, що включає особливі фази, характерні для роботи з даними та моделями. На першому етапі відбувається збирання даних, які є основою для навчання моделей. Важливо, щоб дані були якісними, репрезентативними та відповідали вимогам бізнесу, оскільки від цього залежить точність і надійність моделі.

Наступним кроком є розроблення ознак (feature engineering), який передбачає обробку та трансформацію сирих даних у форму, придатну для навчання моделі. Цей етап включає виділення ключових параметрів, нормалізацію, кодування категоріальних змінних та інші методи підготовки даних, що безпосередньо впливають на ефективність навчання.

Фаза навчання моделі передбачає застосування алгоритмів машинного навчання для побудови прогнозних або класифікаційних систем на основі підготовлених ознак. Після навчання проводиться валідація моделі, під час якої оцінюється її точність, стабільність та здатність узагальнювати знання на нових даних.

Розгортання включає інтеграцію моделі у виробниче середовище, де вона починає обробляти реальні дані та надавати результати користувачам або системам. Важливо забезпечити безперервний моніторинг роботи моделі, що дозволяє відстежувати зміни в якості прогнозів, продуктивності та стабільності системи, а також своєчасно виявляти проблеми.

Особливістю життєвого циклу MLOps є повторне навчання моделей із версіонуванням наборів даних та моделей. Це дозволяє підтримувати актуальність моделі при зміні даних, підвищувати точність прогнозів та забезпечувати відтворюваність результатів.

Розглянуті моделі життєвого циклу розроблення програмного забезпечення демонструють різні підходи до організації процесів створення, тестування та супроводу систем, кожен з яких має власні переваги та обмеження. Каскадні моделі забезпечують високу структурованість і передбачуваність, але є менш гнучкими до змін вимог. Ітеративні підходи дозволяють поступово вдосконалювати систему та краще враховувати невизначеність, тоді як гнучкі методології забезпечують максимальну адаптивність і орієнтацію на швидку зміну вимог та тісну взаємодію з користувачем.

Сучасні підходи DevOps і MLOps доповнюють класичні моделі за рахунок автоматизації процесів, безперервної інтеграції, моніторингу та специфічних практик роботи з моделями машинного навчання. Загалом можна зробити висновок, що жодна з розглянутих моделей не є універсальною, а їх ефективність залежить від типу проєкту, його складності та вимог до швидкості, якості й керованості розробки.

Традиційні моделі розроблення програмного забезпечення мають обмежену застосовність у проєктах створення чатботів на основі великих мовних моделей. Каскадна модель, яка передбачає послідовне виконання фаз (вимоги → проєктування → реалізація → тестування → розгортання → супроводження), не підходить для таких проєктів через високу невизначеність початкових вимог до поведінки моделі та відсутність механізмів для ітеративного покращення точності відповідей. Вона не враховує необхідності швидких циклів прототипування запитів

та зворотного зв'язку від користувачів, що критично для адаптації LLM [10] під корпоративні сценарії.

Гнучкі методології, такі як Scrum, Kanban та екстремальне програмування, добре підходять для розроблення компонентів інфраструктури чатбота, наприклад застосунків на Bot Framework або інтеграції з Microsoft Teams. Вони дозволяють організувати роботу команди, швидко додавати функціональність та проводити ретроспективи для покращення процесів. Проте ці підходи недостатньо враховують специфіку розроблення запитів для великих мовних моделей та керування базами знань, що є центральним аспектом ефективності корпоративних LLM-чатботів.

Життєвий цикл DevOps дозволяє автоматизувати розгортання застосунку, безперервну інтеграцію та моніторинг роботи системи, що значно підвищує стабільність та надійність чатбота у виробничому середовищі. Водночас він не охоплює специфічні для LLM процеси, такі як версіонування запитів, A/B-тестування їх ефективності та контроль точності відповідей, які є критичними для забезпечення якості взаємодії користувача з чатботом.

MLOps, у свою чергу, орієнтований на проєкти машинного навчання зі збиранням даних, навчанням моделей та повторним тренуванням з версіонуванням. Він добре підходить для створення власних моделей, проте корпоративні LLM-чатботи часто використовують попередньо навчені моделі через API, такі як Azure OpenAI GPT-4. У таких проєктах головними стають фази оптимізації запитів та курування бази знань для розширеного пошуку, а не навчання моделей з нуля.

У результаті жодна з традиційних моделей не описує повний життєвий цикл розроблення чатботів на основі великих мовних моделей. Для таких проєктів критично важливими є фазове прототипування запитів, інтеграція розширеного пошуку по корпоративних документах, оцінювання якості відповідей та виробниче розгортання з безперервним моніторингом та адаптацією. Тільки поєднання ітеративного підходу, елементів гнучких методологій та практик DevOps/MLOps із фокусом на оптимізацію запитів та управління базою знань дозволяє ефективно

реалізувати корпоративні LLM-чатботи та підтримувати їхню точність, релевантність та надійність у виробничому середовищі.

У даному розділі наведено порівняльний аналіз основних моделей життєвого циклу розроблення програмного забезпечення, які застосовуються або можуть бути адаптовані для створення AI- та LLM-застосунків. Розгляд охоплює як класичні підходи (каскадні, V-подібні, ітеративні), так і сучасні методології (Agile, DevOps, MLOps), а також спеціалізований життєвий цикл для LLM-систем.

Порівняння здійснюється за критеріями, що є визначальними саме для проєктів на основі великих мовних моделей, зокрема: врахування фази розроблення запитів, підтримка архітектури розширеного пошуку, специфічність моделі для LLM-рішень, керованість ітеративних експериментів, інтеграція з корпоративними комунікаційними платформами та наявність безперервного оцінювання якості відповідей. Це дозволяє оцінити придатність кожної моделі до використання в умовах розроблення сучасних корпоративних AI-систем.

Нижче наведена порівняльна таблиця моделей життєвого циклу AI-застосунків за ключовими критеріями (таблиця 3).

Таблиця 3 – порівняльна таблиця моделей життєвого циклу AI-застосунків за ключовими критеріями

Модель/ Критерій	Врахування фази розроблення запитів	Підтримка архітектури розширеного пошуку	Специфіч ність для великих мовних моделей	Керованість ітеративних експериментів	Інтеграція з корпорати вними комуніка ційними платформа ми	Безперервне оцінювання якості відповідей
Каскадна	Низьке	Ні	Загальна	Структурована	Відсутня	Відсутнє
V-подібна	Низьке	Ні	Загальна	Структурована	Відсутня	Відсутнє
Ітеративна	Середнє	Частково	Загальна	Структурована	Зовнішня	Частково ручне
Інкрементальна	Середнє	Частково	Загальна	Структурована	Зовнішня	Частково ручне
Гнучкі (Scrum/Kanban/XP)	Середнє/Високе	Частково	Загальна	Ситуативна	Природна/ Зовнішня	Частково ручне

Кінець таблиці 3

Модель/ Критерій	Врахування фази розроблення запитів	Підтримка архітектури розширеного пошуку	Специфіч ність для великих мовних моделей	Керованість ітеративних експериментів	Інтеграція з корпорати вними комуніка ційними платформа ми	Безперервне оцінювання якості відповідей
DevOps	Низьке/Серед нє	Так (для розгортання)	Загальна	Структурована	Природна/ Зовнішня	Автоматизо ване (для компонентів)
MLOps	Середнє	Частково	Машинне навчання	Структурована	Зовнішня	Автоматизо ване (для навчання моделей)
Специфіч ний LLM- цикл	Високе	Так	Великі мовні моделі	Ситуативна	Природна	Автоматизо ване та ручне комбіноване

Аналіз традиційних моделей життєвого циклу показав, що жодна з них у повній мірі не задовольняє вимоги проєктів розроблення корпоративних чатботів на основі великих мовних моделей у середовищі Teams.

Каскадні та V-подібні моделі недостатньо гнучкі для роботи з невизначеними вимогами і не передбачають циклів ітеративного покращення точності відповідей. Гнучкі методології дозволяють ефективно організувати командну роботу та прототипування, проте не охоплюють специфічні процеси розроблення запитів і керування базами знань. DevOps добре автоматизує розгортання та забезпечує стабільність роботи застосунку, але не враховує версіонування запитів та А/В-тестування ефективності відповідей. MLOps орієнтований на навчання моделей і керування даними, тоді як корпоративні чатботи зазвичай використовують попередньо навчені LLM через API, що робить критичними етапи оптимізації запитів і інтеграції розширеного пошуку.

У зв'язку з цим для ефективної розробки корпоративних LLM-чатботів необхідна спеціалізована модель життєвого циклу, яка поєднує переваги кількох підходів. Вона має інтегрувати елементи гнучких методологій для ітеративності та швидкого прототипування, практики DevOps для автоматизації розгортання та

забезпечення стабільності роботи, а також принципи MLOps для версіонування наборів даних, оцінювання якості відповідей та повторного навчання моделей. Крім того, така модель повинна включати нові специфічні фази, характерні саме для LLM-чатботів: розроблення та оптимізацію запитів, інтеграцію з архітектурою розширеного пошуку корпоративних документів і безперервний моніторинг результатів у середовищі Teams. Застосування такої моделі дозволить підвищити точність, релевантність і надійність чатботів, скоротити ризики помилок та забезпечити ефективну підтримку користувачів у корпоративних комунікаційних системах.

1.3 Аналіз методів розроблення LLM-чатботів для Microsoft Teams

LLM-чатботи на платформі Microsoft Teams зазвичай реалізуються за чотиришаровою архітектурою, яка забезпечує модульність, масштабованість та простоту інтеграції з корпоративними системами.

Перший рівень – шар представлення. На цьому шарі знаходиться безпосередньо бот у Teams, який взаємодіє з користувачами через адаптивні картки та повідомлення. Для розробки застосовуються SDK Bot Framework версії 4.x із підтримкою Node.js, Python або C#, що забезпечує гнучкість вибору технології під потреби команди розробників. Інтеграція з Teams здійснюється через Bot Service Connector API, який відповідає за прийом і відправлення повідомлень, обробку подій і подачу інтерфейсних елементів користувачам.

Другий рівень – шар координації. Він відповідає за логіку управління діалогами та маршрутизацію намірів користувачів. Для цього використовується Azure Bot State Service, що дозволяє зберігати стан користувацьких сесій та забезпечує безперервність взаємодії, навіть якщо користувач перериває діалог. На цьому шарі здійснюється розпізнавання намірів користувачів і направлення їх до

відповідних обробників, які відповідають за генерацію відповідей, інтеграцію з базами знань або зовнішніми сервісами.

Третій рівень – інтелектуальний шар. Цей шар відповідає за обробку запитів користувачів і генерацію відповідей на основі великих мовних моделей. У корпоративних чатботах інтеграція здійснюється зі службою Azure OpenAI [11], зокрема GPT-4 Turbo, через REST API, що дозволяє безпосередньо надсилати запити до моделі та отримувати відповіді у реальному часі. Важливою складовою цього шару є система запитів, яка розділяє їх на кілька категорій: системні запити для встановлення ролей та поведінки моделі, користувацькі запити, що формуються безпосередньо від співробітників, та контекстні запити асистента, які враховують історію діалогу і корпоративний контекст. Для забезпечення ефективного використання ресурсів і контролю витрат у цьому шарі реалізується управління обсягом обчислень, що дозволяє регулювати кількість оброблюваних запитів та складність генерації відповідей.

Четвертий рівень – шар даних. Його завданням є забезпечення розширеного пошуку корпоративної інформації та інтеграція зовнішніх джерел знань у відповіді чатбота. Для цього використовується компонент розширеного пошуку на основі векторного пошуку, який працює з корпоративною базою знань. У реалізації корпоративних LLM-чатботів застосовується Azure AI Search із підтримкою вкладень (embeddings), що дозволяє виконувати семантичний пошук документів, аналізувати релевантність і швидко знаходити необхідну інформацію у великому масиві корпоративних даних.

Для підвищення ефективності корпоративного LLM-чатбота важливим є інтеграційний компонент із корпоративними сховищами даних. У сучасних реалізаціях інтеграція з SharePoint та OneDrive здійснюється через Microsoft Graph API [12], що дозволяє ботам у Teams отримувати доступ до документів у режимі реального часу. Це забезпечує можливість оперативного отримання актуальної інформації, що зберігається у корпоративних бібліотеках документів, і дозволяє чатботу відповідати на запити користувачів із врахуванням найновіших даних.

Ще одним критичним компонентом є шар кешування, який використовується для оптимізації затримки повторних запитів та підвищення продуктивності системи. Найчастіше застосовується Redis як високопродуктивне in-memory сховище, що дозволяє зберігати проміжні результати обробки запитів або відповіді моделі на часті запити користувачів. Це зменшує час очікування та навантаження на обчислювальні ресурси, особливо при інтеграції з великими мовними моделями через Azure OpenAI, де кожен запит може потребувати значних обчислювальних витрат.

У поєднанні з чотиришаровою архітектурою – шаром представлення, координації, інтелектуальним шаром і шаром даних – інтеграція через Microsoft Graph API та кешування забезпечують чатботам високий рівень продуктивності, швидкий доступ до актуальної корпоративної інформації та надійність у роботі з великими обсягами запитів.

Ефективність роботи чатботів на основі великих мовних моделей значною мірою залежить від якості сформованих запитів, тому їх розроблення потребує систематичного та структурованого підходу. Базою такого підходу є формування системного запиту, який визначає роль моделі, її поведінкові характеристики та обмеження. Саме системний запит задає контекст взаємодії, встановлює стиль відповідей, рівень формальності, допустимі межі генерації інформації та правила обробки запитів користувачів.

У системному запиті чітко визначається роль чатбота, наприклад, як корпоративного асистента, консультанта технічної підтримки або експерта з певної предметної області. Це дозволяє моделі формувати відповіді відповідно до очікувань користувачів та бізнес-вимог організації. Поведінкові настанови задають правила формування відповідей, включно з точністю, лаконічністю, структурованістю або орієнтацією на пояснення складних понять. Обмеження, у свою чергу, визначають допустимі межі використання інформації, заборону на генерацію недостовірних або конфіденційних даних, а також регламентують використання корпоративних джерел інформації.

Важливим елементом систематичного підходу до розроблення запитів є навчання на кількох прикладах, яке передбачає додавання у запит 3–5 типових сценаріїв взаємодії користувача з чатботом у вигляді прикладів діалогів. Такий підхід дозволяє моделі краще розуміти контекст, очікуваний формат відповідей і специфіку предметної області, що особливо важливо для корпоративного середовища з вимогами до єдиного стилю та точності комунікації. Використання прикладів підвищує узгодженість і якість відповідей, зменшує кількість неоднозначних результатів і спрощує налаштування системи, оскільки значну частину поведінки моделі можна задати через репрезентативні приклади взаємодії.

Ще одним важливим підходом до розроблення запитів є ланцюжкове формулювання думок, яке передбачає інструкції для моделі розбивати складні запити на послідовні кроки. Це дозволяє моделі виконувати проміжний аналіз задачі, виділяти ключові аспекти, оцінювати наявну інформацію та формувати логічну послідовність міркувань, що забезпечує більш точні та обґрунтовані результати, особливо для багатокрокових завдань, як-от аналіз документів або вирішення задач із кількома умовами.

Застосування цього методу підвищує точність відповідей, зменшує логічні помилки та робить процес міркування моделі прозорим і контрольованим [18]. У корпоративних LLM-чатботах ланцюжкове формулювання думок сприяє надійнішій обробці складних запитів, підвищує довіру користувачів і забезпечує логічно послідовні та аргументовані відповіді.

Важливим елементом систематичного підходу до розроблення запитів є використання шаблонних запитів із параметризацією, що дозволяє адаптувати поведінку чатбота до різних сценаріїв використання. Такий підхід передбачає створення узагальнених шаблонів запитів, у яких змінні частини підставляються динамічно залежно від контексту запиту користувача, типу задачі або джерела даних. Для забезпечення стабільності та контрольованого розвитку системи застосовується версіонування запитів. Воно може реалізовуватися за принципами семантичного версіонування, коли кожна зміна у запиті має свій номер версії залежно від рівня змін – від незначних коригувань до суттєвих оновлень логіки. Це

дозволяє відстежувати еволюцію запитів, аналізувати їх ефективність і, у разі необхідності, повертатися до попередніх версій. Додатково застосовуються А/В-випробування, які передбачають одночасне використання декількох варіантів запитів на реальному виробничому трафіку.

Окрему увагу в розробленні запитів приділяють захисним механізмам, які спрямовані на забезпечення безпеки та відповідності корпоративним політикам. Захисні запити дозволяють запобігати ін'єкціям, коли користувач намагається змінити поведінку моделі шляхом спеціально сформульованих запитів. Для цього вводяться обмеження у системних інструкціях та контроль обробки вхідних даних. Додатково застосовується фільтрування вмісту із використанням сервісів на кшталт Azure AI Content Safety, що дозволяє виявляти неприйнятний або небезпечний контент і блокувати або коригувати відповіді моделі. Також важливим аспектом є редагування або маскування особистих ідентифікаційних даних, що дозволяє дотримуватися вимог конфіденційності та нормативних обмежень щодо обробки персональної інформації.

Для забезпечення доступу до актуальної корпоративної інформації в LLM-чатботах використовується підхід генерування з розширеним пошуком, який поєднує можливості великих мовних моделей із семантичним пошуком у внутрішніх базах знань. Такий підхід дозволяє формувати відповіді на основі достовірних корпоративних документів, підвищуючи їхню точність, релевантність та відповідність внутрішнім стандартам організації. Конвеєр генерування з розширеним пошуком для доступу до корпоративних документів складається з наступних фаз:

- фази індексування даних, під час чого здійснюється збір корпоративних даних із SharePoint та OneDrive через Microsoft Graph API з подальшою обробкою документів різних форматів, документи сегментуються на семантичні фрагменти (500–1000 токенів), що підвищує точність пошуку та зменшує навантаження на модель, формуються embedding-вектори за допомогою Azure OpenAI (text-embedding-ada-002) [13], які відображають зміст тексту у векторному просторі та отримані дані індексуються в Azure AI Search із

використанням гібридного підходу, що поєднує векторний і ключовий пошук для підвищення релевантності результатів;

- фаза пошуку, яка забезпечує знаходження релевантної інформації у корпоративній базі знань відповідно до запиту користувача, під час якої запит користувача перетворюється на векторне представлення за допомогою тієї ж моделі вкладень, що використовувалася під час індексування, отримане вкладення використовується для виконання семантичного пошуку у векторному індексі, де застосовується метрика косинусної подібності для визначення ступеня близькості між запитом і фрагментами документів і у результаті обираються найбільш релевантні кандидати, які використовуються на наступному етапі обробки;

- фаза доповнення передбачає формування розширеного запиту до мовної моделі із включенням знайденого контексту, під час чого до початкового запиту користувача додаються релевантні фрагменти документів, що дозволяє моделі генерувати відповіді з урахуванням корпоративних знань.

- фаза циклу зворотного зв'язку, що забезпечує безперервне вдосконалення якості роботи чатбота, від час якої здійснюється збирання оцінок користувачів щодо отриманих відповідей, що дозволяє визначити рівень їхньої задоволеності, точності та релевантності наданої інформації та які можуть формуватися як явно, наприклад через інтерфейс оцінювання відповідей у Microsoft Teams, так і неявно, на основі аналізу поведінки користувачів, включно з повторними запитами або уточненнями.

Важливою складовою цього процесу є журналювання пар «запит–відповідь», що дозволяє накопичувати історію взаємодії користувачів із системою. Аналіз цих даних дає можливість виявляти типові помилки, неефективні запити або недостатньо релевантні результати пошуку. На основі отриманої інформації проводиться вдосконалення системи запитів, оптимізація параметрів пошуку та оновлення бази знань.

Зібрані дані також можуть використовуватися для покращення компонентів розширеного пошуку, зокрема шляхом уточнення алгоритмів ранжування, оновлення вкладень або зміни стратегій відбору контексту. Це дозволяє підвищити

точність знаходження релевантних документів і, відповідно, якість відповідей, що генеруються мовною моделлю.

Оцінювання якості LLM-чатботів є складним багаторівневим процесом, який охоплює перевірку як окремих компонентів системи, так і її загальної ефективності у реальних сценаріях використання. Для цього застосовується комплексний підхід до випробування, що дозволяє забезпечити надійність, точність і релевантність відповідей у корпоративному середовищі.

Першим рівнем є модульне випробування окремих компонентів системи, яке передбачає перевірку їхньої працездатності із використанням імітації зовнішніх залежностей. Це дозволяє тестувати окремі частини чатбота, такі як обробники запитів, інтеграція з API або механізми пошуку, незалежно від реальних зовнішніх сервісів.

Наступним рівнем є випробування запитів, яке спрямоване на оцінювання якості відповідей, що генеруються мовною моделлю. Для цього формується тестовий набір, який зазвичай містить понад 100 запитань, що охоплюють типові сценарії використання чатбота. Оцінювання здійснюється за кількома ключовими показниками:

- точність визначається як відсоток правильних відповідей відносно загальної кількості тестових запитів;
- релевантність оцінюється на основі семантичної подібності між відповіддю чатбота та еталонною відповіддю, що дозволяє враховувати змістовну відповідність, а не лише точний збіг тексту;
- повнота характеризує ступінь охоплення всіх аспектів поставленого питання, що є важливим для складних або багатокomпонентних запитів;
- достовірність відповідає за перевірку того, чи базується відповідь на реальних даних із корпоративних документів і чи відсутні ознаки галюцинацій, тобто вигаданої або недостовірної інформації.

Окрему увагу приділяють випробуванню компонентів розширеного пошуку, оскільки саме вони забезпечують доступ до корпоративної бази знань. На цьому рівні оцінюється якість пошуку за такими показниками, як точність і повнота

відбору релевантних документів серед результатів. Важливим є також оцінювання релевантності контексту, який передається мовній моделі, оскільки від правильного відбору фрагментів документів залежить якість фінальної відповіді.

Наступним етапом оцінювання якості LLM-чатботів є інтеграційне випробування, яке спрямоване на перевірку взаємодії всіх компонентів системи у комплексі. У межах цього етапу проводяться наскрізні тести діалогів із використанням емулятора Bot Framework, що дозволяє моделювати реальні сценарії спілкування користувача з чатботом. Такі тести охоплюють повний цикл обробки запиту – від введення користувачем повідомлення до генерації відповіді з урахуванням інтеграції з мовною моделлю та компонентами розширеного пошуку.

Важливою складовою оцінювання є приймальне випробування за участю користувачів, яке проводиться на пілотній групі, що зазвичай включає від 10 до 20 реальних користувачів. У процесі такого тестування збирається якісний і кількісний зворотний зв'язок щодо роботи чатбота. Для цього використовуються опитування, які можуть включати шкалу придатності системи (System Usability Scale) та показники задоволеності користувачів. Додатково здійснюється аналіз журналів бесід, що дозволяє виявити типові патерни відмов, складні для обробки запити або ситуації, в яких чатбот не забезпечує очікуваний результат.

Завершальним етапом є навантажувальне випробування, яке дозволяє оцінити продуктивність системи при великій кількості одночасних користувачів. У рамках цього тестування моделюється робота чатбота при навантаженні понад 500 одночасних користувачів, що відповідає умовам реального корпоративного використання. Основними показниками є час відповіді системи, який має становити менше 3 секунд, а також стабільність роботи під навантаженням. Особлива увага приділяється оптимізації витрат на обчислення, зокрема при використанні великих мовних моделей, що дозволяє забезпечити баланс між швидкістю роботи системи та економічною ефективністю її експлуатації.

Незважаючи на активне впровадження чатботів на основі великих мовних моделей у корпоративному середовищі Microsoft Teams, сучасні підходи до їх розроблення мають низку суттєвих недоліків, що впливають на якість, керованість

і надійність таких систем. Однією з ключових проблем є відсутність стандартизованої методології розроблення, яка б чітко регламентувала всі етапи життєвого циклу LLM-чатбота, включаючи проєктування запитів, інтеграцію з корпоративними даними та оцінювання якості відповідей. У результаті процес розроблення часто має ситуативний характер і значною мірою залежить від досвіду окремих розробників.

Суттєвим недоліком є також нехтування документуванням архітектури запитів. У багатьох проєктах відсутній формалізований опис структури системних, користувацьких і контекстних запитів, що ускладнює підтримку, масштабування та передачу знань між членами команди.

Недостатній рівень версіонування запитів є ще однією важливою проблемою. У разі внесення змін до запитів або логіки їх формування часто відсутня можливість швидко повернутися до попередньої стабільної версії у випадку погіршення якості відповідей.

Крім того, у багатьох реалізаціях відсутня систематична структура оцінювання якості чатбота. Оцінювання часто проводиться нерегулярно або без чітко визначених метрик, що ускладнює об'єктивне вимірювання точності, релевантності та достовірності відповідей.

Слабка інтеграція компонента розширеного пошуку з корпоративними системами управління доступом також є значною проблемою. У деяких випадках відсутній належний контроль доступу до документів, що може призводити до порушення політик безпеки та конфіденційності.

Окрему увагу слід приділити ігноруванню управління витратами, що виникають при використанні великих мовних моделей. Відсутність механізмів контролю обчислювальних ресурсів і оптимізації запитів може призводити до значних фінансових витрат, особливо при масштабному використанні чатбота.

Ще одним важливим недоліком є брак моніторингу та спостережуваності у виробничому середовищі. У багатьох системах відсутні інструменти для відстеження ключових показників, таких як точність відповідей, час відгуку або рівень задоволеності користувачів. Також не реалізовані механізми автоматичного

сповіщення про зниження якості роботи системи або збільшення затримки відповіді, що ускладнює оперативне реагування на проблеми.

Таким чином, існуючі підходи до розроблення LLM-чатботів у Teams потребують вдосконалення шляхом впровадження стандартизованих методологій, системного підходу до управління запитами, розширення можливостей оцінювання та моніторингу, а також інтеграції з корпоративними політиками безпеки та управління ресурсами.

1.4 Постановка задачі дослідження

Наявні моделі життєвого циклу розроблення програмного забезпечення (каскадна, гнучкі методології, DevOps, MLOps) не повною мірою враховують специфіку проєктів створення корпоративних чатботів на основі великих мовних моделей для платформ на кшталт Microsoft Teams. Зокрема, вони не охоплюють ключові для таких систем процеси, як ітеративне розроблення та версіонування запитів (prompt engineering), інтеграцію архітектури розширеного пошуку з корпоративними базами знань, а також безперервне оцінювання якості відповідей за метриками точності, релевантності та достовірності. Окремо не враховується управління витратами на виклики API мовних моделей, що є суттєвим фактором у хмарних LLM-рішеннях.

Відсутність формалізованої моделі життєвого циклу для таких систем призводить до низької передбачуваності термінів і бюджетів, а також до високого рівня ризику невдач при переході від пілотної до виробничої стадії (до 65% за даними Gartner 2024). Додатково виникає накопичення технічного боргу через недостатнє документування та керування промптами й архітектурою знань. Таким чином, існує невідповідність між вимогами до розроблення LLM-чатботів для Microsoft Teams і можливостями класичних моделей життєвого циклу, що не враховують специфіку сучасних AI-систем.

Метою даного дослідження є підвищення ефективності управління ІТ-проєктами розроблення корпоративних LLM-чатботів для платформи Microsoft Teams шляхом створення спеціалізованої моделі життєвого циклу, адаптованої до особливостей систем на основі великих мовних моделей.

Запропонована модель повинна забезпечувати формалізацію основних фаз життєвого циклу таких проєктів, а також визначення чітких критеріїв переходу між цими фазами, що дозволить підвищити керованість процесу розроблення та зменшити рівень невизначеності під час реалізації складних AI-рішень. Окрему увагу приділено введенню системи метрик якості, які дозволяють об'єктивно оцінювати ефективність роботи чатбота, зокрема точність, релевантність та достовірність формованих відповідей.

Важливою складовою мети є інтеграція найкращих практик існуючих підходів до розроблення програмного забезпечення, зокрема гнучких методологій, DevOps та MLOps, із новими специфічними етапами, характерними саме для LLM-систем. До таких етапів належать прототипування та ітеративне вдосконалення системи запитів, інтеграція архітектури розширеного пошуку корпоративних знань, а також організація безперервного процесу оцінювання якості відповідей у реальному часі.

Реалізація зазначеної мети має сприяти підвищенню передбачуваності результатів проєктів, скороченню термінів розроблення, оптимізації витрат на використання мовних моделей та зменшенню ризиків, пов'язаних із переходом від пілотного до промислового впровадження чатботів у корпоративному середовищі Microsoft Teams.

Для досягнення поставленої мети необхідно вирішити такі задачі:

- проаналізувати процеси управління ІТ-проєктами розроблення LLM-чатботів для корпоративних систем у Microsoft Teams;
- дослідити існуючі моделі життєвого циклу ІТ-проєктів розроблення програмного забезпечення та визначити їх застосовність до проєктів створення LLM-чатботів;

- проаналізувати архітектурні компоненти, підходи до розроблення запитів, технології RAG та засоби оцінювання якості LLM-чатботів для Microsoft Teams;
- розробити концептуальну модель життєвого циклу IT-проєктів створення LLM-чатботів для корпоративних систем у Microsoft Teams;
- виконати адаптацію запропонованої моделі до різних типів проєктів розроблення LLM-чатботів з урахуванням складності, гнучких методологій та ризиків;
- формалізувати процес версіонування та оцінювання запитів для LLM-чатботів із використанням кількісних показників якості;
- розробити проєкт впровадження моделі життєвого циклу LLM-чатбота для корпоративних систем у Microsoft Teams;
- виконати проєктування архітектури рішення, планування строків, вартості, ресурсів і ризиків проєкту;
- здійснити програмну реалізацію основних функціональних модулів LLM-чатбота, бази знань, RAG-компонента та засобів моніторингу;
- провести експериментальну перевірку розробленого рішення, проаналізувати результати тестування та оцінити практичну цінність впровадження для підприємства.

2 РОЗРОБЛЕННЯ МОДЕЛІ ЖИТТЄВОГО ЦИКЛУ ІТ-ПРОЄКТІВ СТВОРЕННЯ LLM-ЧАТБОТІВ

2.1 Концептуальна модель життєвого циклу LLM-чатботів для Microsoft Teams

Розроблення LLM-чатботів для корпоративних систем у Microsoft Teams потребує використання спеціалізованої моделі життєвого циклу, яка враховує як особливості класичних ІТ-проектів, так і специфіку систем на основі великих мовних моделей. На відміну від традиційних програмних продуктів, які здебільшого мають наперед визначену функціональність, LLM-чатботи потребують постійного уточнення запитів, перевірки якості відповідей, оновлення бази знань і контролю коректності взаємодії з корпоративними даними.

Концептуальна модель життєвого циклу LLM-чатбота для Microsoft Teams має поєднувати ітеративність гнучких методологій, автоматизацію DevOps-процесів та підходи MLOps до версіонування, моніторингу й оцінювання результатів. Такий підхід дозволяє забезпечити поступове вдосконалення системи, зменшити ризики помилок під час розгортання та підтримувати стабільну якість відповідей у процесі експлуатації.

Основою запропонованої моделі є ітеративний підхід до розроблення та перевірки запитів. Для цього доцільно використовувати короткі цикли розроблення, подібні до Scrum-спринтів, у межах яких створюються, тестуються та вдосконалюються системні, користувацькі й контекстні запити. Кожна ітерація завершується оцінюванням результатів, аналізом помилок і внесенням змін до логіки взаємодії чатбота з користувачем. Це дозволяє швидко реагувати на виявлені недоліки та поступово підвищувати точність і релевантність відповідей.

Важливим елементом моделі є інтеграція DevOps-підходу, що забезпечує автоматизацію збирання, тестування та розгортання програмних компонентів чатбота. Використання CI/CD-конвеєрів дає змогу оперативно впроваджувати нові версії системи у хмарному середовищі Azure, зменшувати ймовірність помилок при

розгортанні та забезпечувати контроль стабільності роботи застосунку. Це особливо важливо для корпоративного середовища, де чатбот повинен бути доступним, надійним і безпечним для користувачів.

Окреме значення має застосування принципів MLOps, зокрема версіонування запитів, тестових наборів, конфігурацій RAG-компонента та параметрів взаємодії з великою мовною моделлю. Для контролю змін доцільно використовувати семантичне версіонування, яке дозволяє фіксувати суттєві, функціональні та незначні зміни в запитах. Це забезпечує можливість відстеження впливу кожної зміни на якість відповідей, а також дає змогу виконувати відкат до попередніх стабільних версій у разі погіршення результатів.

Для корпоративних LLM-чатботів важливим є пріоритет якості відповідей над кількістю реалізованих функцій. Тому в межах моделі передбачається регулярне автоматизоване оцінювання точності, релевантності, повноти та безпечності відповідей. Оцінювання виконується на основі заздалегідь підготовлених тестових наборів, які містять типові сценарії взаємодії користувачів із системою. Додатково залучаються експерти предметної області, які перевіряють відповідність відповідей корпоративним регламентам, внутрішнім політикам і вимогам безпеки.

З урахуванням особливостей LLM-рішень у роботі пропонується модель життєвого циклу розроблення чатбота для Microsoft Teams. Вона охоплює такі етапи: визначення вимог і випадків використання, проектування архітектури, прототипування запитів, інтеграція розширеного пошуку, розроблення бота, оцінювання якості та виробниче розгортання.

На початковому етапі здійснюється визначення вимог до майбутньої системи та формування основних сценаріїв її використання. Для цього аналізуються потреби зацікавлених сторін, визначаються бізнес-цілі впровадження чатбота, уточнюються функціональні та нефункціональні вимоги, а також встановлюються обмеження щодо безпеки, бюджету, строків і доступу до корпоративних даних. Результатом цього етапу є погоджений документ вимог, перелік пріоритетних

сценаріїв використання та початковий набір еталонних запитань і відповідей для подальшого тестування.

Наступним етапом є проектування архітектури рішення. На цьому етапі визначається загальна структура системи, обираються необхідні хмарні сервіси та встановлюються способи взаємодії між компонентами чатбота. Особлива увага приділяється інтеграції з Microsoft Teams, Azure OpenAI, Azure AI Search, Microsoft Graph API та корпоративними джерелами даних. Також визначаються механізми автентифікації, авторизації, захисту інформації, журналювання та контролю витрат на використання мовної моделі. Результатом етапу є архітектурна документація, опис основних компонентів системи та затверджена технічна схема рішення.

Після проектування архітектури виконується розроблення та первинне налаштування системи запитів. У межах цього етапу формуються системні, користувацькі та контекстні запити, які визначають поведінку чатбота під час взаємодії з користувачем. Запити поступово вдосконалюються на основі тестування, аналізу помилок і перевірки відповідності корпоративним сценаріям. Для підвищення якості відповідей можуть застосовуватися шаблони запитів, приклади типових діалогів, параметризація та семантичне версіонування. Результатом етапу є набір перевірених запитів, які забезпечують прийнятний рівень точності, релевантності та стабільності відповідей.

Окремим етапом є інтеграція чатбота з корпоративною базою знань. На цьому етапі здійснюється підключення до внутрішніх джерел інформації, зокрема SharePoint, OneDrive або інших корпоративних сховищ документів. Документи проходять попередню обробку, поділяються на семантичні фрагменти, перетворюються у векторні представлення та індексуються для подальшого пошуку. Це дозволяє реалізувати RAG-компонент, завдяки якому чатбот формує відповіді не лише на основі можливостей мовної моделі, а й з урахуванням актуальної корпоративної інформації.

Подальший етап передбачає безпосереднє розроблення програмних компонентів чатбота. У межах цього етапу реалізується застосунок для Microsoft Teams на основі Microsoft Bot Framework, створюється логіка діалогів,

налаштовується оброблення повідомлень користувачів, інтеграція з Azure OpenAI та механізми відображення відповідей. Також реалізуються засоби роботи з адаптивними картками, журналюванням, обробленням помилок і взаємодією з компонентом розширеного пошуку. Результатом етапу є функціональний прототип чатбота, готовий до комплексного тестування.

Після завершення основної розробки проводиться оцінювання якості роботи чатбота. На цьому етапі перевіряються точність, релевантність, повнота та безпечність відповідей, а також якість пошуку в корпоративній базі знань. Додатково проводиться користувацьке тестування, аналіз журналів діалогів і навантажувальна перевірка системи. Якщо отримані показники не відповідають встановленим критеріям, виконуються додаткові ітерації вдосконалення запитів, налаштування пошуку або зміни окремих компонентів системи.

Завершальним етапом є впровадження чатбота у виробниче корпоративне середовище Microsoft Teams. Розгортання доцільно виконувати поступово: спочатку для обмеженої групи користувачів, а після підтвердження стабільності роботи – для всіх цільових користувачів організації. На цьому етапі налаштовуються засоби моніторингу, сповіщення про помилки, контроль продуктивності та збір зворотного зв'язку. Також готується експлуатаційна документація та інструкції для адміністраторів і користувачів системи.

Таким чином, запропонований метод побудови життєвого циклу забезпечує кероване створення LLM-чатбота для Microsoft Teams, поєднуючи аналіз вимог, архітектурне проєктування, розроблення запитів, інтеграцію корпоративних знань, програмну реалізацію, оцінювання якості та виробниче впровадження. Такий підхід дозволяє зменшити ризики проєкту, забезпечити контроль якості відповідей і підтримувати стабільну роботу чатбота в корпоративному середовищі.

Метрики якості та критерії переходу між фазами представлені у таблиці 4.

Таблиця 4 – Метрики якості та критерії переходу між етапами

Етап життєвого циклу чатбота	Показники оцінки ЖЦ	Порогові значення для переходу
Етап 1 – Визначення сценаріїв використання	Кількість формалізованих випадків використання; розмір тестового набору	Мінімум 5 сценаріїв використання; тестовий набір не менше 100 пар питання-відповідь
Етап 2 – Проектування архітектури	Повнота архітектурної документації; схвалення безпеки; точність попереднього бюджету	Описані всі компоненти архітектури; безпека схвалена; відхилення бюджету менше 20%
Етап 3 – Прототипування запитів	Точність запитів на тестовому наборі; кількість ітерацій покращення	Точність не менше 80%; щонайменше 3 версії запитів
Етап 4 – Інтеграція розширеного пошуку	Точність пошуку документів; релевантність контексту	Точність $\geq 85\%$; релевантність контексту $\geq 80\%$
Етап 5 – Розроблення бота	Покриття коду тестами; результати інтеграційних тестів	Покриття тестами $\geq 80\%$; 100% успішне проходження інтеграційних тестів
Етап 6 – Оцінювання якості	Точність відповідей; оцінка придатності системи; кількість критичних дефектів	Точність $\geq 85\%$; оцінка придатності ≥ 75 балів; 0 критичних дефектів
Етап 7 – Виробниче розгортання	Доступність системи; затримка відповіді	Доступність 99.9%; затримка відповіді менше 3 секунд (95-й перцентиль)

На першій фазі оцінюється кількість визначених сценаріїв використання та наявність тестового набору для подальшої перевірки точності роботи чатбота.

На другій фазі контролюється повнота архітектури, включно з описом компонентів, взаємодією сервісів, безпекою та інтеграцією з корпоративною інфраструктурою, а також перевіряється відповідність вимогам безпеки та попередній бюджет хмарних сервісів.

Третя фаза зосереджена на точності запитів до мовної моделі та кількості ітерацій їх покращення, що демонструє ітеративний підхід до оптимізації взаємодії.

На четвертій фазі оцінюється ефективність пошуку у корпоративних документах за метриками точності та релевантності контексту.

П'ята фаза перевіряє якість програмної реалізації через покриття коду тестами та успішне проходження інтеграційних тестів.

На шостій фазі проводиться комплексна оцінка системи кінцевими користувачами з урахуванням точності відповідей та зручності використання, при цьому критичні дефекти неприпустимі.

Сьома фаза оцінює експлуатаційні показники після розгортання, зокрема доступність сервісу та швидкість обробки запитів, що забезпечує стабільну роботу чатбота у корпоративному середовищі.

Для кількісного обґрунтування параметрів життєвого циклу LLM-чатбота доцільно використати математичну модель, яка відображає залежності між характеристиками проєкту, якістю відповідей, готовністю до переходу між етапами та рівнем ризиків [21]. На відміну від методу, який задає послідовність виконання етапів, модель описує взаємозв'язки між вхідними та вихідними показниками у вигляді формальних залежностей.

У загальному вигляді модель оцінювання параметрів життєвого циклу можна подати за формулою:

$$Y = F(X)$$

де Y – множина вихідних показників життєвого циклу;

X – множина вхідних характеристик проєкту;

F – функціональна залежність між характеристиками проєкту та результатами оцінювання.

Для проєкту розроблення LLM-чатбота множина вхідних характеристик може бути подана за формулою:

$$X = \{U, D, I, S, N\}$$

де U – складність сценаріїв використання;

D – обсяг і різноманітність корпоративних даних;

I – кількість інтеграцій із корпоративними сервісами;

S – рівень вимог до безпеки;

N – очікувана кількість користувачів.

Множина вихідних показників моделі має вигляд:

$$Y = \{C_p, T_i, Q, G_i, R\Sigma\}$$

де C_p – інтегральна оцінка складності проєкту;

T_i – тривалість i -го етапу життєвого циклу;

Q – інтегральна оцінка якості відповідей чатбота;

G_i – коефіцієнт готовності переходу до наступного етапу;

$R\Sigma$ – загальний рівень ризику проєкту.

Інтегральну оцінку складності проєкту доцільно визначати як зважену суму основних характеристик за формулою:

$$C_p = w_1U + w_2D + w_3I + w_4S + w_5N$$

де C_p – інтегральна оцінка складності проєкту;

Тривалість окремого етапу життєвого циклу залежить від його базової тривалості та загальної складності проєкту:

$$T_i = T_{i0} \cdot (1 + kC_p)$$

де T_i – скоригована тривалість i -го етапу;

T_{i0} – базова тривалість i -го етапу;

k – коефіцієнт впливу складності на тривалість робіт;

C_p – інтегральна оцінка складності проєкту.

Інтегральна оцінка якості відповідей LLM-чатбота визначається через показники точності, релевантності, повноти та безпечності:

$$Q = a_1A + a_2R + a_3F + a_4B$$

де Q – інтегральна оцінка якості відповіді чатбота;

A – точність відповіді;

R – релевантність відповіді до запиту користувача;

F – повнота відповіді;

B – безпечність і відповідність корпоративним політикам;

a_1, a_2, a_3, a_4 – вагові коефіцієнти відповідних показників.

Коефіцієнт готовності переходу до наступного етапу життєвого циклу визначається як частка виконаних критеріїв завершення поточного етапу:

$$G_i = (\sum_{j=1}^m K_{ij}) / m$$

де G_i – коефіцієнт готовності i -го етапу;

K_{ij} – значення j -го критерію завершення i -го етапу;

m – кількість критеріїв, що оцінюються на відповідному етапі.

Перехід до наступного етапу можливий за умови:

$$G_i \geq G_{min}$$

де G_{min} – мінімально допустиме значення готовності етапу.

Оцінка окремого ризику визначається як добуток імовірності його виникнення та рівня впливу на проєкт:

$$R_i = P_i \cdot I_i$$

де R_i – оцінка i -го ризику;

P_i – ймовірність виникнення ризику;

I_i – рівень впливу ризику на проєкт.

Загальний рівень ризику проєкту визначається як сума оцінок окремих ризиків:

$$R\Sigma = \sum_{i=1}^n R_i$$

де $R\Sigma$ – загальний рівень ризику проєкту;

R_i – оцінка i -го ризику;

n – кількість виявлених ризиків.

Таким чином, запропонована математична модель не задає послідовність виконання робіт, а формалізує залежності між характеристиками проєкту та показниками його життєвого циклу. Метод побудови життєвого циклу визначає послідовність етапів розроблення LLM-чатбота, тоді як модель дає змогу кількісно оцінювати складність, тривалість, якість, готовність і ризику на відповідних етапах.

2.2 Адаптація моделі до різних типів проєктів розроблення LLM-чатботів

Тривалість фаз життєвого циклу LLM-чатботів залежить від масштабу та складності проєкту. Основні фактори – кількість користувачів, обсяг бази знань, кількість сценаріїв використання та рівень інтеграції з корпоративними системами. Тому цю модель можна адаптувати, змінюючи тривалість фаз без зміни їхньої структури.

Прості проєкти, наприклад автоматизація FAQ для до 50 користувачів із базою до 100 документів, можна реалізувати за 12 тижнів. Фази виконуються швидше завдяки меншій кількості сценаріїв і спрощеній архітектурі:

- фаза 1 – 1 тиждень,
- фаза 2 – 1 тиждень,

- фаза 3 – 2 тижні,
- фаза 4 – 2 тижні,
- фаза 5 – 3 тижні,
- фаза 6 – 2 тижні,
- фаза 7 – 1 тиждень.

На цьому рівні оцінюється базова якість запитів, простота інтеграції та основні функції пошуку.

Проекти середньої складності, наприклад багатофункціональні асистенти для 500 користувачів із інтеграцією понад 500 документів SharePoint, виконуються за 24 тижні. Тут повністю реалізуються всі фази: від збору вимог і проєктування архітектури до розроблення запитів, інтеграції пошуку, тестування та підготовки до виробничого розгортання.

Складні проєкти, що обслуговують понад 1000 користувачів і включають багатоагентну архітектуру з інтеграцією бізнес-процесів, потребують 36 тижнів. Тривалість фаз збільшується для детального опрацювання великої кількості сценаріїв, розширеної архітектури та комплексного тестування:

- фаза 1 – 3 тижні,
- фаза 2 – 4 тижні,
- фаза 3 – 8 тижнів (4 спринти),
- фаза 4 – 5 тижнів,
- фаза 5 – 8 тижнів,
- фаза 6 – 5 тижнів,
- фаза 7 – 3 тижні.

Вибір моделі базується на кількості сценаріїв, розмірі бази знань, числі користувачів і складності архітектури. Такий підхід дозволяє оптимально планувати ресурси, забезпечити якість системи та відповідність корпоративним вимогам.

Для ефективної розробки LLM-чатботів модель життєвого циклу можна інтегрувати з гнучкими методологіями Scrum і Kanban. Фази 1–2 виконуються як підготовчі спринти, під час яких формуються вимоги, визначаються пріоритетні

сценарії та створюється архітектура системи. Фаза 3 організовується як два спринти по два тижні для прототипування та оптимізації запитів до мовної моделі з регулярним тестуванням і демонстраціями результатів стейкхолдерам.

Фази 4–5 реалізуються паралельно у 3–4 спринти: частина команди займається інтеграцією розширеного пошуку та індексуванням документів, інша – програмною логікою чатбота та інтеграцією з Microsoft Teams. Фаза 6 – спринт стабілізації для комплексного тестування та виправлення дефектів, оптимізації запитів і покращення точності відповідей. Фаза 7 – спринт розгортання, що включає впровадження системи у виробниче середовище, підготовку документації та навчання користувачів.

Стандартні події Scrum забезпечують координацію: щоденні стендапи, планування спринтів із декомпозицією задач на user stories, огляди спринтів для демонстрації проміжних результатів і ретроспективи для вдосконалення процесу.

Після виробничого розгортання доцільно використовувати Kanban для супроводження: Kanban-дошка із стовпцями Backlog, To Do, In Progress, Review і Done дозволяє керувати запитами на зміни, виправлення помилок та оновлення бази знань. Це забезпечує гнучке управління чергою задач, швидке реагування на запити користувачів і безперервне вдосконалення системи.

Проекти розроблення LLM-чатботів для корпоративних систем пов'язані з низкою технічних, організаційних та економічних ризиків. Особливістю таких проєктів є використання великих мовних моделей, зовнішніх API-сервісів та інтеграція з корпоративними джерелами даних, що може впливати на стабільність роботи системи, точність відповідей та загальні витрати на її експлуатацію. Тому у межах моделі життєвого циклу важливо передбачити механізми ідентифікації, оцінювання та пом'якшення ризиків.

Для систематизації можливих проблем доцільно використовувати таблицю ризиків (Таблиця 5), яка містить опис потенційних загроз, оцінку їхньої ймовірності та впливу на проєкт, а також стратегії їх пом'якшення.

Таблиця 5 – Таблиця ризиків

Ризик	Ймовірність	Вплив	Стратегії пом'якшення
Погіршення якості мовної моделі при зміні даних у базі знань розширеного пошуку	Середня	Високий	Безперервне оцінювання точності через автоматизовані тестові набори після кожної зміни індексу; налаштування сповіщень при падінні точності більше ніж на 5%; версіонування індексів для можливості швидкого відкату до попередньої версії
Перевищення бюджету Azure-служб через недооцінку кількості викликів API	Висока	Середній	Налаштування лімітів витрат у Azure Cost Management; моніторинг споживання токенів у реальному часі; кешування повторних запитів за допомогою Redis; оптимізація розміру контекстного вікна
Генерування некоректного або упередженого вмісту мовною моделлю	Середня	Високий	Інтеграція сервісів фільтрування небажаного вмісту (Azure AI Content Safety); використання захисних запитів для запобігання prompt-ін'єкціям; впровадження циклів зворотного зв'язку за участю людини для перевірки відповідей; забезпечення прозорості через цитування джерел
Низька прийнятність системи користувачами через незадовільну якість відповідей	Середня	Високий	Проведення пілотного тестування з реальними користувачами на етапі оцінювання якості; збирання зворотного зв'язку через вбудовані опитування після кожної відповіді; ітеративне вдосконалення запитів на основі аналізу помилок та відмов

У системах із розширеним пошуком ризиком є зниження якості відповідей після оновлення бази знань. Щоб мінімізувати його, застосовують автоматизоване тестування та версіонування індексів для швидкого відновлення стабільної версії.

Інший ризик – перевищення бюджету на хмарні сервіси через високий обсяг викликів API. Для контролю використовують моніторинг ресурсів, встановлення лімітів і оптимізацію запитів.

Також важливим є некоректний або небажаний контент, який може генерувати LLM, та низька прийнятність системи користувачами. Для цього

застосовують фільтри, перевірку відповідей людиною і пілотне тестування з реальними користувачами для збору зворотного зв'язку.

2.3 Версіонування та оцінювання запитів для LLM-чатботів

У проєктах розроблення LLM-чатботів управління запитами до мовної моделі є критично важливим, оскільки навіть невеликі зміни у формулюванні запиту можуть суттєво впливати на точність і релевантність відповідей. Для контролю цих змін застосовується система семантичного версіонування запитів за схемою MAJOR.MINOR.PATCH. Кожен компонент версії має конкретне призначення. MAJOR версія змінюється при внесенні змін, що порушують зворотну сумісність або суттєво змінюють поведінку системи, наприклад, зміна системного запиту, що визначає роль чатбота, або перехід на новий формат вихідних даних. MINOR версія збільшується при додаванні нової функціональності, яка не порушує сумісності, наприклад, додавання прикладів для навчання моделі, впровадження методів ланцюжкового формулювання думок або підтримки нових сценаріїв використання. PATCH версія відображає незначні зміни та виправлення, що не впливають на основну логіку роботи чатбота, такі як корекція граматики або уточнення інструкцій у запиті.

Усі версії запитів зберігаються у Git-репозиторії з використанням тегів для позначення стабільних релізів. Додатково ведеться файл CHANGELOG.md, у якому фіксуються всі зміни між версіями, включаючи додану функціональність, виправлені помилки та уточнені компоненти запиту. Такий підхід забезпечує прозорість процесу розвитку системи, дозволяє відстежувати історію змін і швидко повертатися до стабільних версій у разі необхідності.

Для оцінювання ефективності нових версій запитів застосовується A/B-тестування на реальному виробничому трафіку. Переважна частина запитів (наприклад, 90 %) обробляється стабільною версією, а решта 10 % спрямовується

на експериментальну версію. Результати порівнюються за ключовими метриками, такими як точність відповідей та рівень задоволеності користувачів. Якщо експериментальна версія демонструє покращення показників, частка трафіку, що обробляється новою версією, поступово збільшується, доки вона не стає основною стабільною версією системи. Такий підхід дозволяє контролювати впровадження змін, підтримувати високу якість запитів і забезпечує стабільність роботи чатбота у виробничому середовищі.

Для забезпечення високої якості відповідей LLM-чатботів у корпоративних системах застосовується система автоматизованого оцінювання, яка дозволяє швидко та об'єктивно контролювати ефективність нових версій запитів. Така система реалізується як конвеєр автоматизованого тестування, що оцінює відповіді чатбота за чотирма метриками: точність, релевантність, повнота та достовірність. Це дозволяє підтримувати стабільну якість роботи системи, швидко виявляти проблеми після змін у запитах і забезпечувати високий рівень задоволеності користувачів.

Точність визначає відсоток правильних відповідей на тестовому наборі. Для оцінки використовується косинусна подібність `embeddings` між згенерованою відповіддю та еталонним текстом; значення понад 0.85 вважається правильним. Поріг для виробничої версії встановлюється на рівні 85 %, що гарантує стабільність базових відповідей навіть при ітеративному оновленні запитів.

Релевантність оцінює, наскільки відповідь відповідає запитанню користувача, з використанням `cross-encoder`; поріг – 0.90.

Повнота визначає, чи охоплені всі аспекти запитання, із порогом мінімум 80 %, що забезпечує комплексність відповіді та повну інформацію для користувача.

Достовірність перевіряє, чи базується відповідь на наявних джерелах без галюцинацій, із порогом 95 % підтверджених тверджень, що мінімізує ризик некоректної інформації у корпоративному середовищі.

Автоматизація здійснюється через Python-скрипт із бібліотеками `transformers` та `sentence-transformers` для обчислення `embeddings` і семантичної подібності, а також інструментами обробки тестових наборів. Конвеєр інтегрується з Azure

DevOps Pipelines і запускається після кожного коміту нової версії запитів, що дозволяє оперативно виявляти проблеми та блокувати впровадження некоректних версій. Всі результати тестування зберігаються у звітах, що забезпечує прозорість процесу для розробників і стейкхолдерів. Крім того, можливо проводити A/B-тестування нових запитів: основна частина трафіку обробляється стабільною версією, невелика – експериментальною. Порівняння метрик точності, релевантності та задоволеності користувачів дозволяє оцінити вплив змін і поступово замінювати основну версію на покращену.

Такий конвеєр забезпечує безперервний контроль якості, підтримку високого рівня точності, релевантності, повноти та достовірності відповідей, мінімізує ризики зниження якості інформації та інтегрується у життєвий цикл розробки LLM-чатбота, роблячи процес більш передбачуваним і надійним.

Для забезпечення постійного підвищення якості роботи LLM-чатботів важливим елементом є організація циклів зворотного зв'язку від користувачів. Такі цикли дозволяють швидко виявляти проблемні запити, оцінювати релевантність та точність відповідей, а також оперативно коригувати запити і логіку роботи бота.

У корпоративних системах на платформі Microsoft Teams ефективним рішенням є вбудування механізму оцінки відповіді безпосередньо в адаптивну картку чатбота. Після кожної відповіді користувач бачить кнопки оцінки +/-, що дозволяють швидко вказати, чи задовольнила його відповідь. У разі негативної оцінки користувач може додатково залишити текстовий коментар [20], описавши причину незадоволення або уточнивши, яка інформація була відсутня чи некоректна.

Отримані дані автоматично зберігаються у централізованій базі зворотного зв'язку, де вони можуть бути об'єднані з результатами автоматизованого оцінювання (метрики точності, релевантності, повноти та достовірності). Це дозволяє визначити, які запити потребують доопрацювання, і пріоритизувати їх у наступних ітераціях.

Для оцінювання якості версії запиту використовується інтегральний показник [19], що враховує точність, релевантність, повноту, стабільність і

безпе́чність відпові́ді. Інтегра́льна оці́нка я́кості версі́ї за́питу визна́чається за форму́лою:

$$Q_v = b_1A + b_2R + b_3F + b_4S + b_5B$$

де Q_v – інтегра́льна оці́нка я́кості версі́ї за́питу;

A – то́чність відпові́ді;

R – релевантність відпові́ді до за́питу користу́вача;

F – повнота відпові́ді;

S – стабі́льність відпові́ді при повто́рних або поді́бних за́питах;

B – безпе́чність відпові́ді та відпові́дність корпору́тивним полі́тикам;

b_1, b_2, b_3, b_4, b_5 – ваго́ві коефі́цієнти відпові́дних показни́ків.

Для порівня́ння дво́х версі́й за́питу доці́льно вико́ристовува́ти показни́к при́росту я́кості:

$$\Delta Q = Q_{v2} - Q_{v1}$$

де ΔQ – зми́на я́кості пі́сля оновле́ння за́питу;

Q_{v2} – інтегра́льна оці́нка ново́ї версі́ї за́питу;

Q_{v1} – інтегра́льна оці́нка попе́редньо́ї версі́ї за́питу.

Я́кщо значе́ння ΔQ є до́датним і переви́щує встановле́не мініма́льне поро́гове значе́ння, нова версі́я за́питу може бу́ти при́йнята до вико́ристання. Я́кщо значе́ння ΔQ є від’є́мним, необхі́дно вико́нати до́даткове доопра́цюванн́я або поверну́тися до попе́редньо́ї стабі́льної версі́ї.

Для систе́матиза́ції показни́ків оці́нюванн́я за́питів доці́льно вико́ристати та́блицю 6.

Таблиця 6 – Показники оцінювання якості запитів LLM-чатбота

Показник	Позначення	Зміст показника	Спосіб оцінювання
Точність відповіді	A	Відповідність відповіді еталонному результату	Частка правильних відповідей у тестовому наборі
Релевантність	R	Відповідність відповіді змісту запиту користувача	Експертна або автоматизована оцінка семантичної відповідності
Повнота	F	Ступінь охоплення всіх аспектів запиту	Оцінка наявності необхідних елементів відповіді
Стабільність	S	Узгодженість відповідей при повторних або подібних запитах	Порівняння результатів декількох запусків
Безпечність	B	Відсутність порушень корпоративних політик і конфіденційності	Перевірка правилами безпеки або експертна оцінка

Таким чином, формалізація версіонування та оцінювання запитів дозволяє контролювати якість змін, порівнювати різні версії запитів і приймати обґрунтовані рішення щодо їх використання у корпоративному LLM-чатботі. Запропонований підхід забезпечує прозорість процесу вдосконалення запитів, зменшує ризик погіршення якості відповідей і підтримує стабільність роботи чатбота в середовищі Microsoft Teams.

3 РОЗРОБКА ПРОЄКТУ ВПРОВАДЖЕННЯ МОДЕЛІ ЖИТТЄВОГО ЦИКЛУ РОЗРОБЛЕННЯ LLM-ЧАТБОТІВ ДЛЯ КОРПОРАТИВНИХ СИСТЕМ У MICROSOFT TEAMS

3.1 Ініціалізація та концепція проєкту

У сучасних корпоративних середовищах впровадження LLM-чатботів для Microsoft Teams потребує не лише створення робочого прототипу, а й формалізованої моделі життєвого циклу, яка дозволяє керувати розробляти, тестувати та супроводжувати систему. Аналіз, проведений у підрозділах 1.1–1.4, показав, що основні проблеми таких рішень пов’язані з відсутністю стандартизації процесів та слабкою керованістю ключових компонентів LLM-систем.

Зокрема, було виявлено відсутність чітких критеріїв завершення фаз розроблення, що ускладнює контроль прогресу проєкту. Додатково спостерігається складність оцінювання трудомісткості розроблення запитів до мовної моделі та недооцінювання часу, необхідного на ітеративне тестування й цикли зворотного зв’язку. Окремою проблемою є технічний борг і баланс між швидкістю розроблення та якістю результатів.

У частині реалізації LLM-чатботів у Microsoft Teams визначено низку системних недоліків: відсутність стандартизованої методології розроблення, слабе або відсутнє документування архітектури запитів, недостатній рівень версіонування промптів і обмежені можливості їх відкату. Також виявлено відсутність системного підходу до оцінювання якості відповідей, слабку інтеграцію компонентів розширеного пошуку з корпоративними системами доступу, недостатній контроль витрат на використання LLM та брак моніторингу й спостережуваності у виробничому середовищі.

У сукупності ці проблеми призводять до нестабільної якості відповідей, складності масштабування рішень, непрогнозованих витрат на API та низької керованості життєвого циклу LLM-систем. Саме тому виникає необхідність окремого проєкту впровадження формалізованої моделі життєвого циклу, яка

забезпечить структурований підхід до розроблення та супроводження чатботів у Microsoft Teams.

Бізнес-метою проєкту є підвищення ефективності доступу працівників до корпоративних знань, скорочення навантаження на службу підтримки та зменшення часу пошуку інформації. Чатбот дозволяє автоматизувати типові запити користувачів і забезпечити швидкий доступ до актуальної інформації без залучення фахівців підтримки.

Технічною метою є створення стандартизованої та масштабованої моделі життєвого циклу LLM-чатботів із використанням Azure OpenAI, Azure AI Search та Microsoft Teams, з акцентом на версіонування запитів, автоматизоване оцінювання якості, контроль витрат і безперервний моніторинг системи.

Організаційною метою є формалізація процесу управління AI-проєктами, включаючи ролі учасників, правила оновлення бази знань, процедури тестування та механізми контролю якості. Це дозволяє підвищити передбачуваність результатів і знизити ризики деградації системи при змінах.

Очікуваним результатом є впровадження корпоративного LLM-чатбота в Microsoft Teams, який забезпечує точні, релевантні та швидкі відповіді з інтеграцією до корпоративної бази знань. Додатково очікується зниження навантаження на підтримку, прискорення доступу до інформації та підвищення задоволеності користувачів.

Критеріями успіху є досягнення точності відповідей понад 85 %, релевантності понад 90 %, середньої затримки відповіді до 3 секунд, а також стабільна робота системи в умовах реального навантаження та контроль витрат на Azure-сервіси. Критерії успіху впровадження наведено в таблиці 7.

Таблиця 7 – Критерії успіху впровадження

Ціль проєкту	Показник до впровадження	Цільове значення	Спосіб перевірки	Відповідальний
Якість відповідей чатбота	Точність відповідей	$\geq 85\%$	Автоматизоване тестування на еталонному наборі	AI-інженер
Релевантність відповідей	Релевантність	$\geq 90\%$	Cross-encoder оцінювання	AI-інженер / QA
Швидкість відповіді	Середня затримка	≤ 3 сек	Application Insights / логування	Azure архітектор
Зменшення навантаження на підтримку	Частка автоматизованих запитів	$\geq 60\%$	Аналітика звернень в Teams / Service Desk	Project Manager (PM) / аналітик
Стабільність роботи	Кількість критичних збоїв	0–мінімум	Моніторинг + логи	DevOps / QA
Контроль витрат	Витрати на 1 запит	у межах бюджету	Azure Cost Management	PM / фін. відповідальний

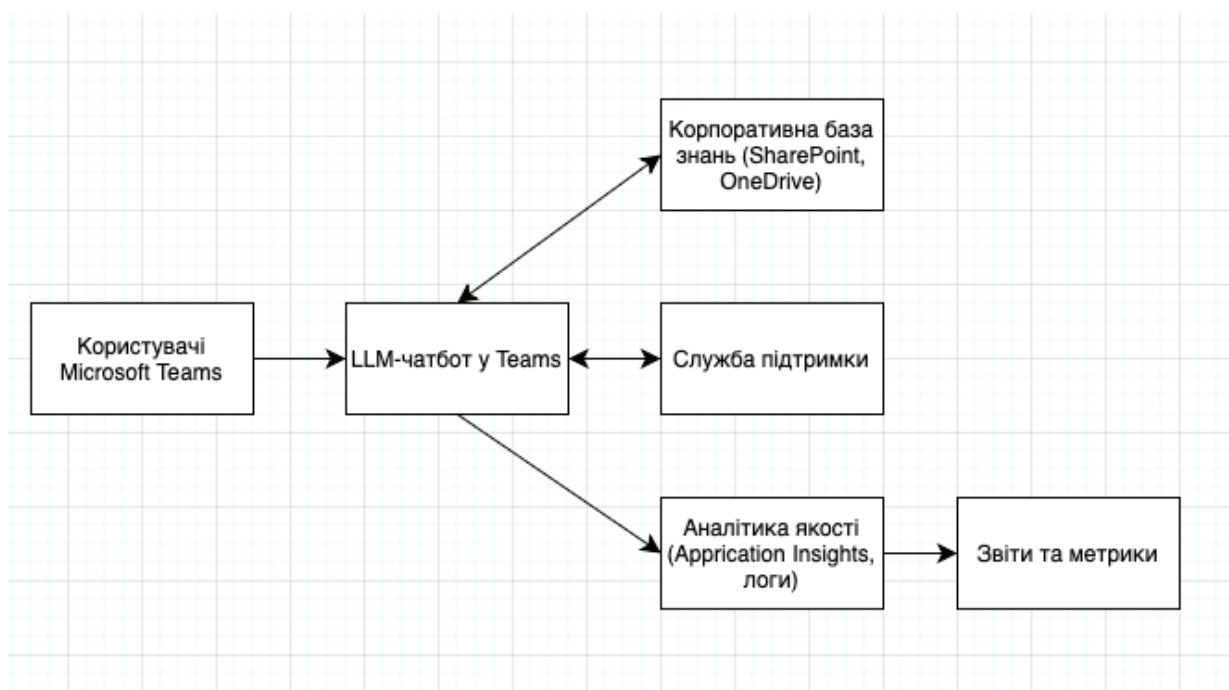


Рисунок 3.1 – Місце проєкту в корпоративному середовищі

Для реалізації проєкту впровадження LLM-чатбота необхідно врахувати інтереси всіх зацікавлених сторін, які впливають на розроблення, впровадження та експлуатацію системи. Оскільки рішення інтегрується з Microsoft Teams, Azure OpenAI та корпоративною базою знань, до проєкту залучаються як технічні спеціалісти, так і кінцеві користувачі.

Основним стейкхолдером є замовник проєкту, який визначає бізнес-цілі та очікує підвищення ефективності роботи працівників і зменшення навантаження на службу підтримки. Високий рівень впливу також мають ІТ-керівник та адміністратори Microsoft 365, які відповідають за інтеграцію рішення в корпоративну інфраструктуру та дотримання вимог безпеки.

Команда розроблення реалізує функціональність чатбота, інтеграцію з Azure-сервісами та процеси автоматизованого розгортання. Експерти предметної області забезпечують наповнення бази знань і перевірку якості відповідей. Кінцеві користувачі взаємодіють із системою в Microsoft Teams і формують основний зворотний зв'язок щодо її ефективності. Після впровадження підтримку системи здійснює служба підтримки, яка обробляє інциденти та контролює стабільність роботи.

Аналіз зацікавлених сторін проєкту наведено в таблиці 8.

Матриця «вплив – зацікавленість» для розподілу стейкхолдерів на групи керування наведена на рисунку 3.2

Таблиця 8 – Аналіз зацікавлених сторін проєкту

Стейкхолдер	Інтерес	Рівень впливу	Очікуваний результат	Канал комунікації	Частота взаємодії
Замовник	Досягнення бізнес-ефекту, автоматизація підтримки	Високий	Підвищення ефективності роботи, ROI	Звіти, зустрічі	Щотижнево / за етапами
ІТ-керівник	Інтеграція в інфраструктуру, безпека	Високий	Стабільна та безпечна система	Teams, наради	Щотижнево

Кінець таблиці 8

Стейкхолдер	Інтерес	Рівень впливу	Очікуваний результат	Канал комунікації	Частота взаємодії
Адміністратори Microsoft 365	Доступи, політики безпеки	Високий	Коректна інтеграція та доступи	Teams, технічні запити	За потреби
Команда розроблення	Реалізація функціоналу	Високий	Робочий чатбот	Azure DevOps, Teams	Щоденно
Експерти предметної області	Якість знань і відповідей	Середній	Коректні відповіді системи	Workshops, Teams	Щотижнево
Кінцеві користувачі	Швидкі відповіді на запити	Середній	Зручний чатбот	Microsoft Teams	Постійно
Служба підтримки	Стабільність роботи	Середній	Мінімум інцидентів	Service Desk, Teams	Щоденно

□	Зацікавленість□	
Вплив□	Низька□	Висока□
Високий□	Адміністратори Microsoft 365□	Замовник← ІТ-керівник← Команда розроблення□
Середній□	-□	Кінцеві користувачі← Експерти предметної області← Служба підтримки□
Низький□	-□	-□

Рисунок 3.2 – Матриця «вплив – зацікавленість» для розподілу стейкхолдерів на групи керування

У процесі розроблення та впровадження LLM-чатбота для Microsoft Teams необхідно враховувати низку обмежень, які можуть впливати на терміни, бюджет і

якість реалізації проєкту. До основних належать часові, фінансові, кадрові, інфраструктурні та регуляторні обмеження.

Часові обмеження пов'язані з необхідністю виконання проєкту у межах визначеного життєвого циклу та термінів магістерського дослідження. Фінансові обмеження стосуються витрат на використання Azure OpenAI, Azure AI Search та інших хмарних сервісів Microsoft. Також важливими є кадрові обмеження, оскільки для реалізації проєкту потрібні спеціалісти з AI, DevOps, Microsoft 365 та адміністрування Azure.

Інфраструктурні обмеження пов'язані з доступом до корпоративних ресурсів, зокрема SharePoint, OneDrive та Microsoft Teams. Крім того, система повинна відповідати вимогам інформаційної безпеки та політикам організації щодо обробки корпоративних даних. До регуляторних обмежень належать правила захисту персональних даних і контроль доступу до внутрішньої інформації.

У проєкті також використовуються певні припущення. Передбачається наявність ліцензій Microsoft 365 та доступу до Azure-служб, можливість інтеграції з корпоративною базою знань, а також участь експертів предметної області для перевірки якості відповідей чатбота. Додатково припускається наявність тестової групи користувачів для проведення пілотного оцінювання системи.

Обмеження та припущення проєкту наведено в таблиці 9.

Таблиця 9 – Обмеження та припущення проєкту

Тип обмеження	Зміст	Вплив на проєкт	Можливі наслідки	Компенсуючі дії
Часові	Обмежені терміни реалізації (навчальний проєкт, етапи життєвого циклу)	Вплив на обсяг реалізованого функціоналу	Скорочення функцій або неповна реалізація окремих модулів	Пріоритизація функцій, ітеративна розробка

Кінець таблиці 9

Тип обмеження	Зміст	Вплив на проєкт	Можливі наслідки	Компенсуючі дії
Фінансові	Вартість Azure OpenAI, Azure AI Search та інших сервісів	Обмеження використання ресурсів	Перевищення бюджету	Оптимізація запитів, кешування, контроль токенів
Кадрові	Необхідність спеціалістів AI, DevOps, Microsoft 365	Вплив на швидкість розробки	Затримки у виконанні задач	Розподіл ролей, часткова автоматизація
Інфраструктурні	Доступ до SharePoint, Teams, OneDrive	Обмеження доступу до даних	Неповна база знань, зниження якості відповідей	Налаштування прав доступу, тестові середовища
Регуляторні	Політики безпеки та захист персональних даних	Обмеження обробки даних	Блокування доступу до частини інформації	Анонімізація даних, фільтрація контенту
Припущення	Наявність ліцензій Microsoft 365 та Azure	Базова умова реалізації проєкту	Неможливість запуску системи	Використання навчальних/тестових акаунтів
Припущення	Доступ до корпоративної бази знань	Якість відповідей системи	Низька релевантність результатів	Формування тестової бази документів
Припущення	Участь експертів предметної області	Оцінка якості відповідей	Неповне навчання системи	Регулярні сесії валідації
Припущення	Тестова група користувачів	Проведення пілотного тестування	Відсутність зворотного зв'язку	Формування внутрішньої тест-групи

3.2 Життєвий цикл проєкту впровадження

Для управління проєктом впровадження LLM-чатбота застосовується гібридний підхід, який поєднує Scrum для ітеративної розробки, Kanban для супроводу після запуску та послідовне корпоративне погодження для етапів, пов'язаних із безпекою, архітектурою та доступом до даних.

Методологія узгоджується із моделлю життєвого циклу, визначеною у розділі 2. Фази 1–2 (вимоги та проєктування архітектури) виконуються у більш формалізованому режимі з чіткими артефактами та контрольними точками погодження. Далі фази 3–6 (прототипування запитів, інтеграція пошуку, розроблення бота та оцінювання якості) реалізуються ітеративно за принципами Scrum, де кожен спринт завершується демонстрацією результатів і аналізом метрик якості..

Після завершення основної розробки фаза 7 (виробниче розгортання) і подальший супровід переходять у режим Kanban. Це забезпечує оперативну обробку змін, виправлення помилок, оновлення бази знань і оптимізацію запитів до моделі без жорсткої прив'язки до спринтів.

Таким чином, запропонований підхід дозволяє поєднати структуроване планування, ітеративну розробку та гнучкий супровід, що повністю відповідає логіці життєвого циклу проєкту.

Фази моделі управління проєктом наведено в таблиці 10.

Таблиця 10 – Фази моделі управління проєктом

Фаза моделі	Метод управління	Основні артефакти	Подія контролю	Результат фази
Фаза 1 – Вимоги та використання	Корпоративне планування + погодження	Документ вимог, список сценаріїв, тестовий набір	Затвердження вимог стейкхолдерами	Формалізовані вимоги та критерії успіху

Кінець таблиці 10

Фаза моделі	Метод управління	Основні артефакти	Подія контролю	Результат фази
Фаза 2 – Архітектура	Корпоративне проєктне управління	Архітектурна схема, вибір Azure-сервісів, бюджет	Архітектурний та безпековий рев'ю	Затверджена архітектура системи
Фаза 3 – Прототипування запитів	Scrum (спринти)	Версії prompt'ів, Git-репозиторій	Sprint Review + тестування	Робочі запити з достатньою точністю
Фаза 4 – Інтеграція пошуку	Scrum	Індекс Azure AI Search, embeddings, pipeline RAG	Оцінка якості пошуку	Працюючий RAG-компонент
Фаза 5 – Розроблення бота	Scrum	Teams Bot, інтеграція Azure OpenAI	Інтеграційне тестування	Функціональний чатбот
Фаза 6 – Оцінювання якості	Scrum + тестування	Тестовий набір, звіти метрик	Пілотне тестування	Підтверджена якість системи
Фаза 7 – Розгортання	Kanban + DevOps	Prod-середовище, документація	Go-live контроль	Виробнича експлуатація системи

Проєкт впровадження LLM-чатбота для Microsoft Teams реалізується як послідовність взаємопов'язаних фаз, що відображають повний життєвий цикл системи та узгоджуються із моделлю, описаною у розділі 2.

Фаза ініціації відповідає етапу вимог та випадків використання. На цьому етапі визначаються бізнес-цілі, ключові стейкхолдери, обмеження та сценарії використання чатбота. Вхідними даними є результати аналізу потреб користувачів і корпоративні вимоги. Результатом є погоджений документ вимог і початковий тестовий набір запитів, який використовується для подальшої перевірки системи.

Фаза планування та проєктування відповідає етапу архітектури. Виконується вибір Azure-сервісів, проєктування структури системи, визначення інтеграції з Microsoft Teams і оцінювання ресурсів. Вхідними даними є вимоги та обмеження проєкту. Результатом є затверджена архітектура та план реалізації, включно з безпековими та бюджетними рішеннями.

Фаза розроблення охоплює прототипування запитів, інтеграцію розширеного пошуку та створення базового функціоналу чатбота. Вона включає роботу з Azure OpenAI, Azure AI Search, Bot Framework SDK та реалізацію логіки діалогів. Вхідними даними є архітектура та тестові сценарії. Результатом є працюючий прототип чатбота у тестовому середовищі Teams.

Фаза інтеграції відповідає компоненту розширеного пошуку. На цьому етапі налаштовується доступ до SharePoint і OneDrive, реалізується сегментація документів, побудова embeddings та індексація в Azure AI Search. Результатом є повноцінний RAG-модуль, інтегрований у чатбот.

Фаза тестування та оцінювання якості включає автоматизоване тестування, навантажувальні випробування та пілотне тестування з користувачами. Перевіряються точність, релевантність і стабільність відповідей. Результатом є підтверджена якість системи та усунення критичних помилок.

Фаза виробничого розгортання відповідає запуску системи у корпоративному середовищі. Виконується поетапне впровадження, налаштування моніторингу, навчання користувачів і передача системи в супровід. Результатом є повноцінна експлуатація чатбота в Microsoft Teams.

Формування фаз проєкту впровадження наведено в таблиці 11.

Таблиця 11 – Формування фаз проєкту впровадження

Фаза проєкту	Відповідний підрозділ розділу 2	Вхідні дані	Основні роботи	Результат	Критерій завершення
Ініціація	Фаза 1 – Вимоги та випадки використання	Потреби користувачів, бізнес-вимоги	Аналіз вимог, визначення сценаріїв, тестовий набір	Документ вимог	Затверджені вимоги і тестовий набір
Планування і проєктування	Фаза 2 – Архітектура	Вимоги, обмеження	Проєктування архітектури, вибір Azure-сервісів	Архітектурна документація	Затверджена архітектура
Розроблення	Фаза 3 – Прототипування запитів + Фаза 5 – Розроблення бота	Архітектура, тестові сценарії	Розробка prompt'ів, Bot Framework, інтеграція LLM	Прототип чатбота	Робочий бот у тестовому середовищі
Інтеграція	Фаза 4 – Інтеграція розширеного пошуку	Корпоративні документи	RAG, індексація, embeddings	Модуль пошуку	Досягнута якість пошуку
Тестування	Фаза 6 – Оцінювання якості	Тестовий набір, користувачі	Автотести, навантаження, пілот	Оцінена система	Метрики досягнуті, дефекти усунені
Впровадження	Фаза 7 – Виробниче розгортання	Тестова система	Розгортання, моніторинг, навчання	Промислова система	Повний доступ користувачів

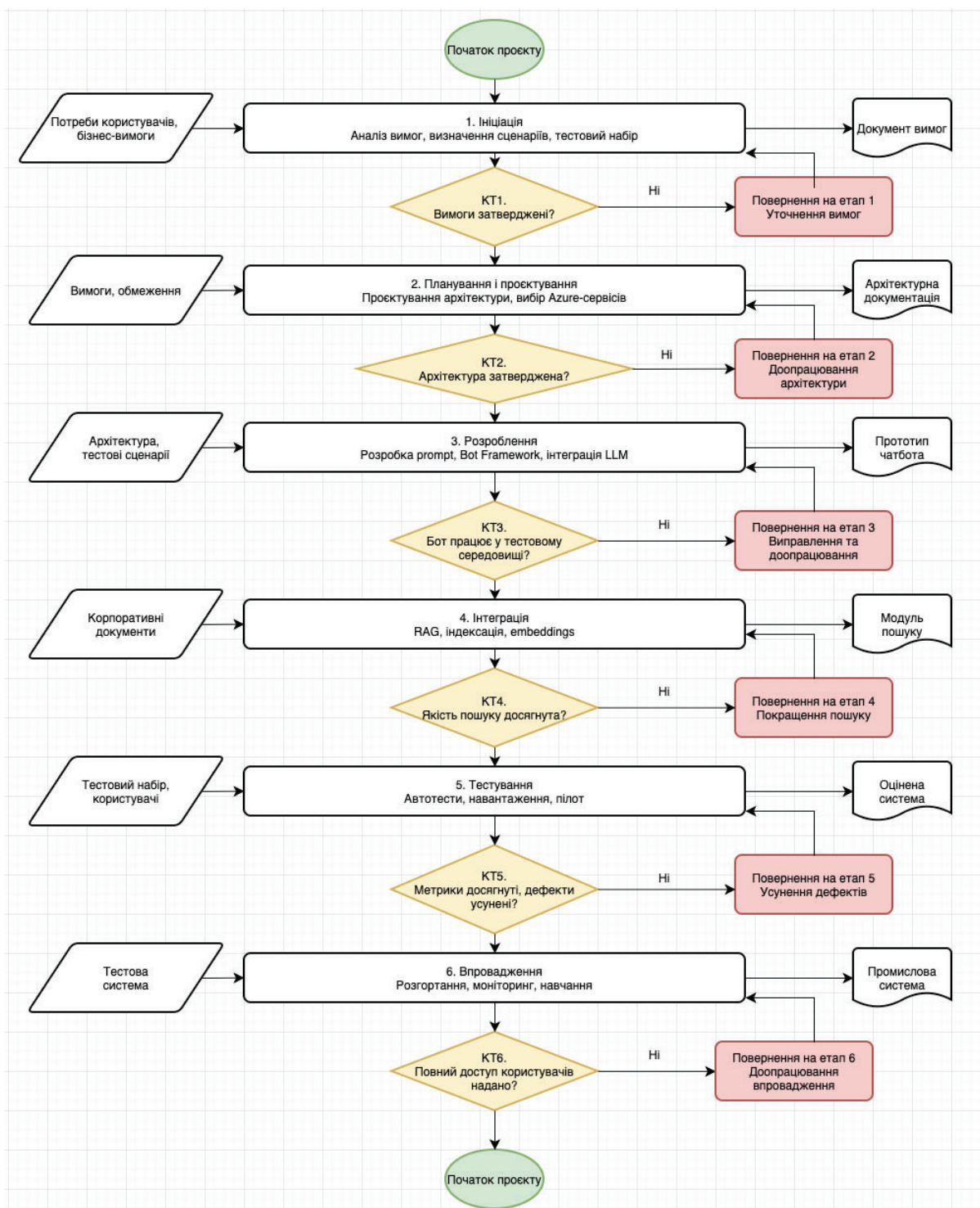


Рисунок 3.3 – Блок-схема проходження фаз проєкту з точками контролю

Для управління проєктом впровадження LLM-чатбота в Microsoft Teams використовується структурна декомпозиція робіт (WBS), яка дозволяє розбити проєкт на ієрархічні рівні та забезпечити контроль виконання кожного етапу.

На верхньому рівні WBS виділяються основні блоки проєкту: аналіз вимог, архітектурне проєктування, підготовка даних, розроблення запитів до мовної

моделі, інтеграція розширеного пошуку, розроблення чатбота, тестування, пілотне впровадження, виробничий запуск, навчання користувачів і супровід системи.

Кожен із цих блоків деталізується на підзадачі другого і третього рівня. Наприклад, аналіз вимог включає збір бізнес-вимог, формування сценаріїв використання та підготовку тестового набору. Архітектурне проєктування охоплює вибір Azure-сервісів, розроблення схеми системи та визначення механізмів безпеки. Розроблення запитів включає створення системних prompt'ів, few-shot прикладів і їх тестування.

Таким чином WBS дозволяє формалізувати всі роботи проєкту, забезпечити прозорість планування та контроль виконання на кожному рівні деталізації.

Структурна декомпозиція робіт проєкту наведена в таблиці 12.

Таблиця 12 – Структурна декомпозиція робіт проєкту

Код WBS	Робота	Зміст	Тривалість	Відповідальний	Результат
1.0	Аналіз вимог	Збір вимог, сценарії використання, тестовий набір	2 тижні	BA / PM	Документ вимог
2.0	Архітектурне проєктування	Вибір Azure-сервісів, схема системи, безпека	2 тижні	AI Architect	Архітектура системи
3.0	Підготовка даних	SharePoint, OneDrive, очищення документів	2 тижні	Data Engineer	Підготовлена база знань
4.0	Розроблення запитів	Prompt engineering, few-shot, тестування	4 тижні	AI Engineer	Набір оптимальних prompt'ів
5.0	RAG / пошук	Azure AI Search, embeddings, індекс	3 тижні	AI Engineer	Пошуковий модуль

Кінець таблиці 12

6.0	Розроблення бота	Teams Bot, Azure OpenAI інтеграція	5 тижнів	Dev	Робочий бот
Код WBS	Робота	Зміст	Тривалість	Відповідальний	Результат
7.0	Тестування	Автотести, load testing, пілот	3 тижні	QA	Підтверджена якість
8.0	Пілот	Тестування з користувачами	2 тижні	PM + QA	Зворотний зв'язок
9.0	Впровадження	Production rollout, Teams запуск	2 тижні	DevOps	Промислова система
10.0	Навчання	Інструкції, training users	1 тиждень	PM	Навчені користувачі
11.0	Супровід	Моніторинг, оновлення, підтримка	постійно	Support	Стабільна робота

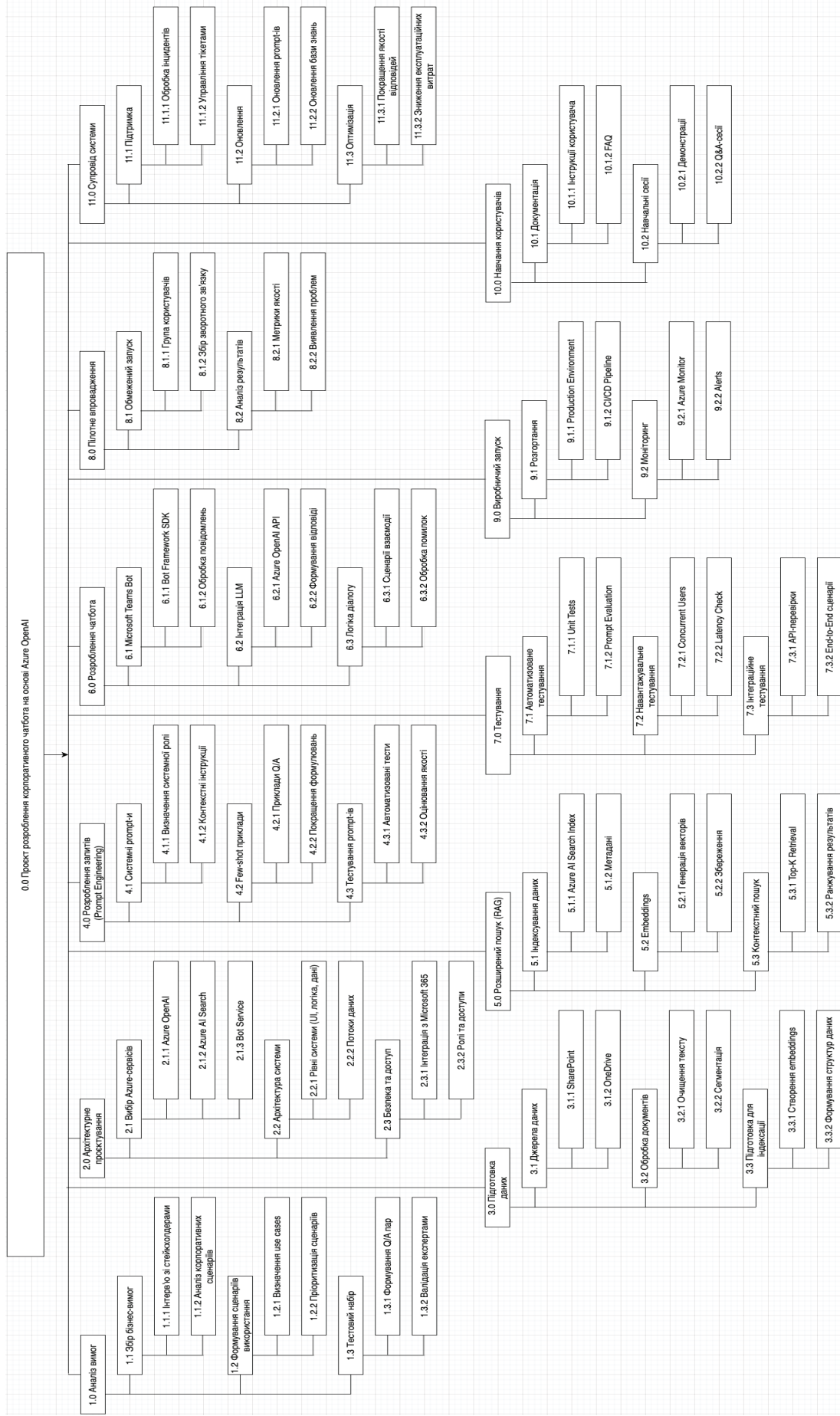


Рисунок 3.4 – Ієрархічна структура робіт WBS із кодами

3.3 Проєктування рішення

У межах проєкту впровадження корпоративного LLM-чатбота для Microsoft Teams формується багаторівнева архітектура, яка поєднує користувацький інтерфейс, сервіси штучного інтелекту, корпоративні джерела знань, а також модулі моніторингу, оцінювання та адміністрування. Така структура забезпечує повний цикл роботи системи: від отримання запиту користувача до формування відповіді та подальшого аналізу якості взаємодій.

Рівень взаємодії з користувачем реалізується через Microsoft Teams, який виступає основним каналом доступу до чатбота. Повідомлення користувачів передаються через Azure Bot Service, що забезпечує маршрутизацію запитів до серверної логіки системи.

Центральним інтелектуальним компонентом є Azure OpenAI Service, який відповідає за генерацію відповідей на основі запитів користувачів і доданого контексту. Контекст формується через інтеграцію з корпоративною базою знань, що реалізується за допомогою Azure AI Search. Дані з SharePoint та OneDrive попередньо індексуються, сегментуються та перетворюються у векторні представлення, що дозволяє здійснювати гібридний семантичний і ключовий пошук.

Додаткові бізнес-процеси автоматизуються через Power Automate, який використовується для тригерних сценаріїв, сповіщень та інтеграції з внутрішніми системами організації.

Окремим важливим компонентом є модуль оцінювання якості, який виконує автоматичний аналіз відповідей чатбота за ключовими метриками (точність, релевантність, достовірність). Результати оцінювання використовуються для вдосконалення запитів і логіки роботи системи.

Моніторинг і спостережуваність реалізуються через Application Insights, де збираються телеметрія, логи запитів, затримки відповідей, помилки та показники використання API. Додатково формується окреме сховище журналів взаємодій, яке

використовується для аналітики поведінки користувачів і покращення якості відповідей.

Адміністративний контур забезпечує керування версіями запитів, оновлення бази знань, контроль доступів та перегляд аналітики. Це дозволяє підтримувати контрольовану еволюцію системи та забезпечує її відповідність корпоративним вимогам безпеки та якості.

Компоненти та призначення наведені в таблиці 13 та рисунку 3.5.

Таблиця 13 – Компоненти та призначення проєктного рішення

Компонент	Призначення	Вхідні дані	Вихідні дані	Технологія реалізації
Microsoft Teams	Інтерфейс користувача	Запити користувачів	Повідомлення боту	Microsoft Teams
Azure Bot Service	Обробка повідомлень	Повідомлення Teams	Виклики backend	Azure Bot Framework
Azure OpenAI Service	Генерація відповідей	Запит контекст +	Текстова відповідь	Azure OpenAI
Azure AI Search	Пошук знань (RAG)	Запит + індекс	Релевантні фрагменти	Azure AI Search
SharePoint / OneDrive	Джерело документів	Файли, документи	Індекси даних	Microsoft 365
Power Automate	Автоматизація процесів	Тригери подій	Операції/сповіщення	Power Platform
Application Insights	Моніторинг	Телеметрія	Метрики, логи	Azure Monitor
База логів взаємодій	Аналітика діалогів	Чати користувачів	Аналітичні дані	Azure Storage / DB
Адміністративний модуль	Управління системою	Дані системи	Налаштування/версії	Web Admin Panel

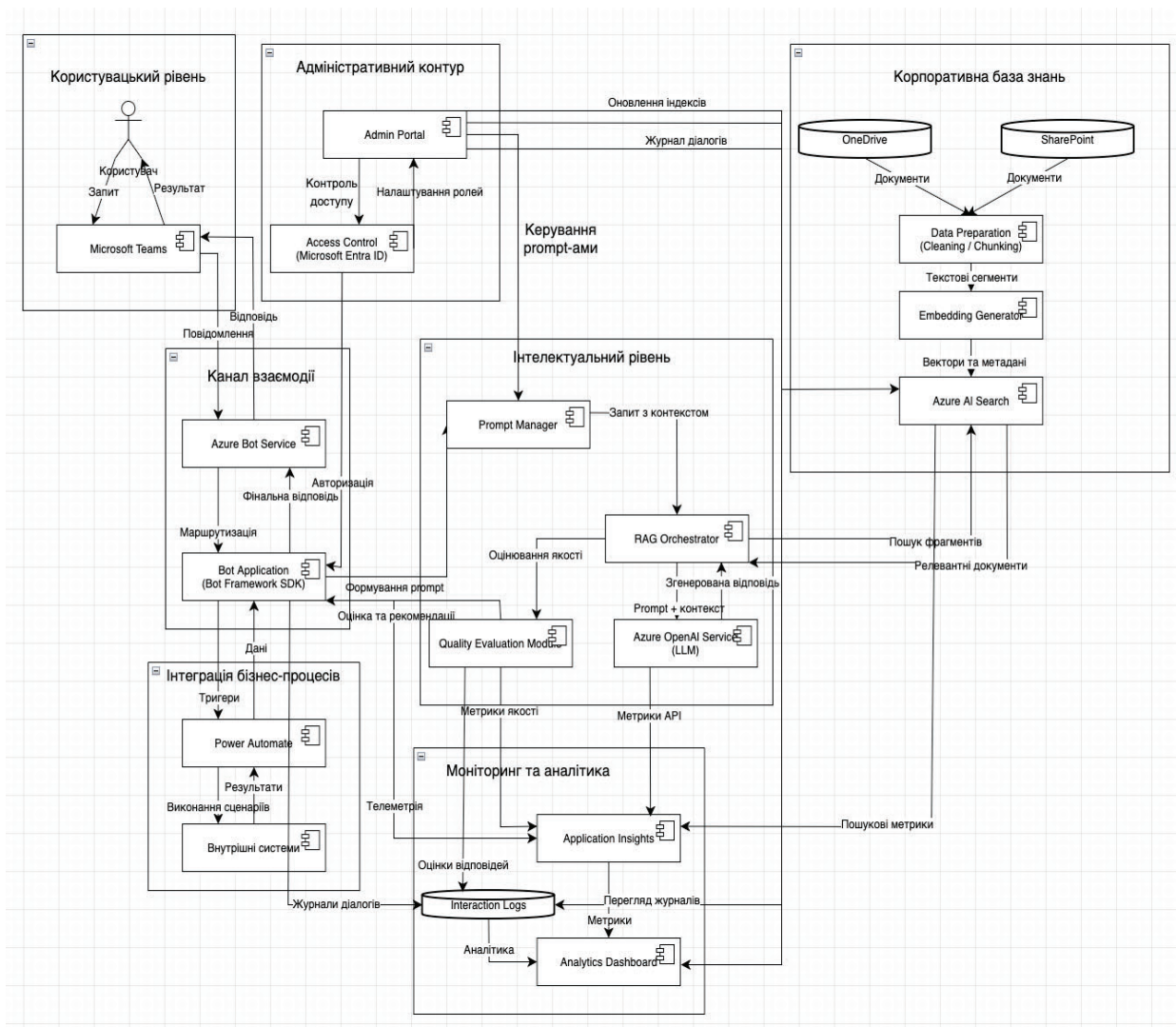


Рисунок 3.5 – Компонентна діаграма проєктного рішення з потоками даних між сервісами

У процесі роботи корпоративного LLM-чатбота оброблення запиту користувача реалізується як послідовний ланцюг етапів, що охоплює прийом повідомлення, визначення сценарію, пошук контексту, генерацію відповіді, постобробку та фіксацію результатів для подальшого аналізу. Такий підхід забезпечує зв'язок між архітектурою системи та її поведінкою в реальному часі.

Процес починається з надходження повідомлення від користувача в Microsoft Teams. Запит передається до Teams Bot через Azure Bot Service, де виконується попередня обробка: очищення тексту, визначення мови, перевірка базової коректності та фільтрація некоректного контенту. На цьому етапі також

відбувається класифікація запиту та визначення сценарію (інформаційний запит, пошук документа, консультація або комбінований запит).

Далі активується RAG-модуль, який здійснює пошук релевантного контексту в корпоративних джерелах. Через Azure AI Search виконується гібридний пошук (ключовий + семантичний) у проіндексованих документах SharePoint та OneDrive. Отримані результати проходять відбір і ранжування, після чого формується контекст для мовної моделі.

На основі запиту користувача, системного промпта та відібраного контексту формується фінальний запит до Azure OpenAI. Мовна модель генерує відповідь, яка проходить етап постобробки: форматування під Teams, перевірку структури, базову фільтрацію небажаного змісту та додавання посилань на джерела (за наявності RAG-контексту).

Після цього відповідь повертається користувачу через Microsoft Teams. Паралельно система виконує журналювання: фіксується текст запиту, відповідь, використаний контекст, час обробки та технічні параметри виконання. Дані зберігаються у сховищі логів взаємодій і передаються в Application Insights для моніторингу продуктивності та якості роботи системи.

Завершальним етапом є збір зворотного зв'язку. Користувач може оцінити відповідь або залишити коментар. Ці дані використовуються для аналізу проблемних сценаріїв, корекції запитів до LLM та покращення якості RAG-компонента.

Таким чином, процес оброблення запиту формує замкнений цикл: від введення користувачем повідомлення до генерації відповіді та накопичення даних для безперервного вдосконалення системи.

Логіко-інформаційна схема процесу функціонування чатбота наведена в таблиці 14.

Таблиця 14 – Логіко-інформаційна схема процесу функціонування чатбота

Етап оброблення	Вхід	Операція	Вихід	Інструмент	Можливі ризики
Отримання запиту	Повідомлення Teams	Прийом через Bot Service	Сирий текст	Azure Bot Service	втрати повідомлень
Попередня обробка	Текст запиту	очищення, фільтрація, класифікація	нормалізований запит	Backend API	некоректна класифікація
Визначення сценарію	Запит	вибір типу задачі	тип сценарію	Logic layer	помилковий роутинг
Пошук контексту (RAG)	Запит	семантичний + ключовий пошук	релевантні документи	Azure AI Search	нерелевантні дані
Формування prompt	запит + контекст	побудова LLM-вхідних даних	фінальний prompt	Prompt engine	перевантаження контекстом
Генерація відповіді	prompt	робота LLM	відповідь	Azure OpenAI	галюцинації
Постобробка	відповідь LLM	форматування, фільтрація	фінальна відповідь	Middleware	втрати змісту
Доставка відповіді	фінальна відповідь	відправка в Teams	відповідь користувачу	Teams Bot	затримки
Журналювання	всі дані	логування	записи в сховище	Application Insights	витік даних
Зворотний зв'язок	оцінка користувача	аналіз якості	метрики	Analytics module	низька активність

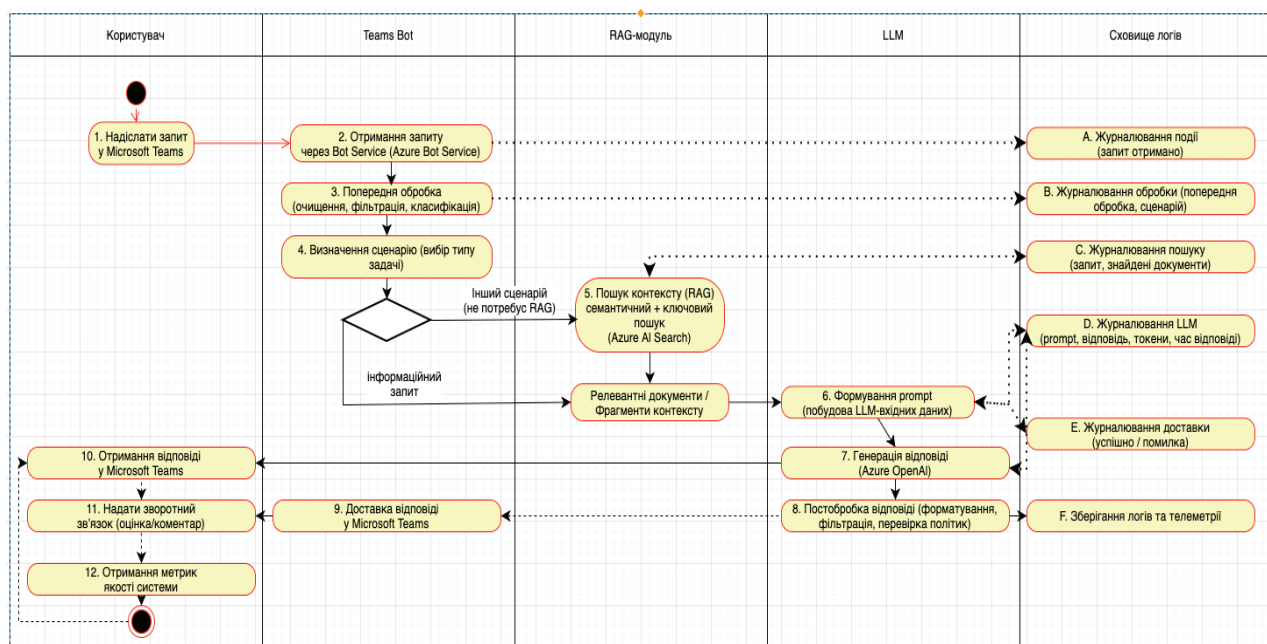


Рисунок 3.6 – UML Activity Diagram

Організаційно-ресурсна модель проєкту визначає склад команди, ролі учасників та розподіл відповідальності під час розроблення і впровадження корпоративного LLM-чатбота для Microsoft Teams. Такий підхід дозволяє забезпечити координацію між технічними, управлінськими та аналітичними задачами, а також ефективно розподілити навантаження між учасниками проєкту.

Ключову роль у проєкті виконує РМ, який відповідає за планування робіт, координацію команди, управління ризиками та взаємодію зі стейкхолдерами. Архітектор Azure AI визначає технічну архітектуру рішення, інтеграцію Azure OpenAI, Azure AI Search та механізми безпеки. Розробник Bot Framework реалізує логіку чатбота, інтеграцію з Microsoft Teams та API Azure-сервісів.

AI-інженер відповідає за prompt engineering, реалізацію RAG-підходу, налаштування пошуку контексту та оцінювання якості відповідей LLM. Фахівець із якості (QA) виконує автоматизоване та користувацьке тестування, перевірку метрик точності й релевантності. Адміністратор Microsoft 365 забезпечує налаштування доступів, інтеграцію з SharePoint та Teams, а також дотримання політик безпеки. Експерт предметної області (Domain Expert) бере участь у формуванні бази знань і перевірці коректності відповідей чатбота.

Рівень залучення ролей залежить від етапу проєкту. На фазах аналізу вимог і проєктування найбільш критичними є PM, AI Architect та Domain Expert. Під час розроблення основне навантаження припадає на Developer та AI Engineer. На етапах тестування та пілотного запуску ключову роль відіграють QA та M365 Admin.

Таким чином, організаційно-ресурсна модель забезпечує чіткий розподіл відповідальності, узгодження компетенцій із задачами проєкту та підтримку стабільної роботи команди протягом усього життєвого циклу системи.

RACI-матриця наведена в таблиці 15.

Таблиця 15 – RACI-матриця

Робота	PM	AI Architect	Developer	QA	M365 Admin	Domain Expert
Аналіз вимог	A	C	I	I	I	R
Формування сценаріїв використання	A	C	I	I	I	R
Проектування архітектури	C	A/R	C	I	C	I
Налаштування Azure-сервісів	I	A	C	I	R	I
Розроблення Teams Bot	I	C	A/R	I	C	I
Prompt engineering	I	C	C	I	I	A/R
Реалізація RAG	I	A	R	I	C	C
Інтеграція SharePoint	I	C	C	I	A/R	I
Автоматизоване тестування	I	I	C	A/R	I	C

Кінець таблиці 15

Робота	PM	AI Architect	Developer	QA	M365 Admin	Domain Expert
Пілотне тестування	A	I	C	R	C	C
Виробниче розгортання	A	C	R	C	R	I
Супровід та моніторинг	A	C	C	R	R	I

Організаційно-ресурсна модель проєкту наведена в таблиці 16.

Таблиця 16 – Організаційно-ресурсна модель проєкту

Роль	Компетенції	Участь у фазах	Завантаження, %	Ключові результати
PM	Agile/Scrum, управління ризиками, комунікація	усі фази	40–60	план проєкту, координація
AI Architect	Azure OpenAI, Azure AI Search, cloud architecture	проєктування, інтеграція	50–70	архітектура рішення
Developer	Bot Framework, Teams API, C#/TypeScript	розроблення, інтеграція	70–90	Teams Bot
AI Engineer	Prompt engineering, RAG, Python, LLM	розроблення, тестування	70–90	prompts, RAG-модуль
QA Engineer	тестування, CI/CD, quality metrics	тестування, пілот	60–80	результати тестування
M365 Admin	SharePoint, Azure AD, Teams administration	інтеграція, запуск	40–70	доступи та безпека
Domain Expert	предметна область, база знань	вимоги, тестування	20–40	перевірка якості знань

3.4 Планування часу, вартості та ризиків

Календарний план проєкту впровадження LLM-чатбота для Microsoft Teams формується відповідно до створеної моделі життєвого циклу, визначеної у розділі 2. Загальна тривалість проєкту становить 24 тижні, що відповідає типовому циклу реалізації корпоративного AI-рішення середньої складності. Планування враховує як послідовність критичних фаз, так і можливість часткового паралельного виконання окремих робіт.

На першому етапі виконується фаза вимог і випадків використання (1–2 тижні). У цей період проводяться інтерв'ю зі стейкхолдерами, визначаються бізнес-вимоги, сценарії використання чатбота та критерії успіху. Результатом є погоджений документ вимог і підготовлений тестовий набір.

Далі реалізується фаза проєктування архітектури (3–4 тижні), під час якої визначається структура системи, обираються сервіси Azure та проєктуються механізми інтеграції й безпеки. Контрольною точкою є затвердження архітектурної документації та бюджету проєкту.

Фаза прототипування запитів (5–8 тижні) частково виконується паралельно з підготовкою бази знань і початковою сегментацією документів для Azure AI Search. У цей період створюються системні prompt-запити, few-shot приклади та шаблони взаємодії з LLM. Контрольним результатом є досягнення точності відповідей понад 80 %.

На фазі інтеграції розширеного пошуку (9–11 тижні) реалізується індексація документів SharePoint і OneDrive, генерація embeddings та налаштування гібридного пошуку. Частина робіт виконується паралельно з початком розроблення Teams Bot.

Фаза розроблення бота (12–16 тижні) включає створення застосунку на Bot Framework SDK, інтеграцію з Microsoft Teams та Azure OpenAI, реалізацію логіки діалогу й модульне тестування. Результатом є функціональний чатбот у тестовому середовищі.

На фазі оцінювання якості (17–19 тижні) проводиться автоматизоване, інтеграційне та навантажувальне тестування, а також пілотна перевірка за участю користувачів. Контрольними точками є досягнення встановлених КРІ точності, релевантності та продуктивності.

Завершальною є фаза виробничого розгортання (20–24 тижні), яка включає поетапне впровадження чатбота, налаштування моніторингу, навчання користувачів і підготовку служби підтримки. Завершенням проєкту вважається стабільна робота системи у виробничому середовищі та її доступність для всіх користувачів організації.

Календарний план проєкту представлено в таблиці 17.

Таблиця 17 – Календарний план проєкту

Етап	Тривалість	Початок	Завершення	Попередники	Контрольна точка
Фаза 1. Вимоги та випадки використання	2 тижні	1 тиждень	2 тиждень	–	Погоджений документ вимог
Фаза 2. Проектування архітектури	2 тижні	3 тиждень	4 тиждень	Фаза 1	Архітектура та бюджет затверджені
Фаза 3. Прототипування запитів	4 тижні	5 тиждень	8 тиждень	Фаза 2	Точність >80 %
Підготовка бази знань	4 тижні	5 тиждень	8 тиждень	Фаза 2	Документи сегментовані
Фаза 4. Інтеграція розширеного пошуку	3 тижні	9 тиждень	11 тиждень	Фаза 3	Пошук працює стабільно
Фаза 5. Розроблення бота	5 тижнів	12 тиждень	16 тиждень	Фаза 4	Функціональний Teams Bot

Кінець таблиці 17

Етап	Тривалість	Початок	Завершення	Попередники	Контрольна точка
Фаза 6. Оцінювання якості	3 тижні	17 тиждень	19 тиждень	Фаза 5	Досягнуті КРІ якості
Фаза 7. Виробниче розгортання	2 тижні	20 тиждень	21 тиждень	Фаза 6	Система доступна користувачам
Навчання користувачів	2 тижні	22 тиждень	23 тиждень	Фаза 7	Проведені навчальні сесії
Супровід та моніторинг	1 тиждень	24 тиждень	24 тиждень	Навчання користувачів	Система передана в підтримку

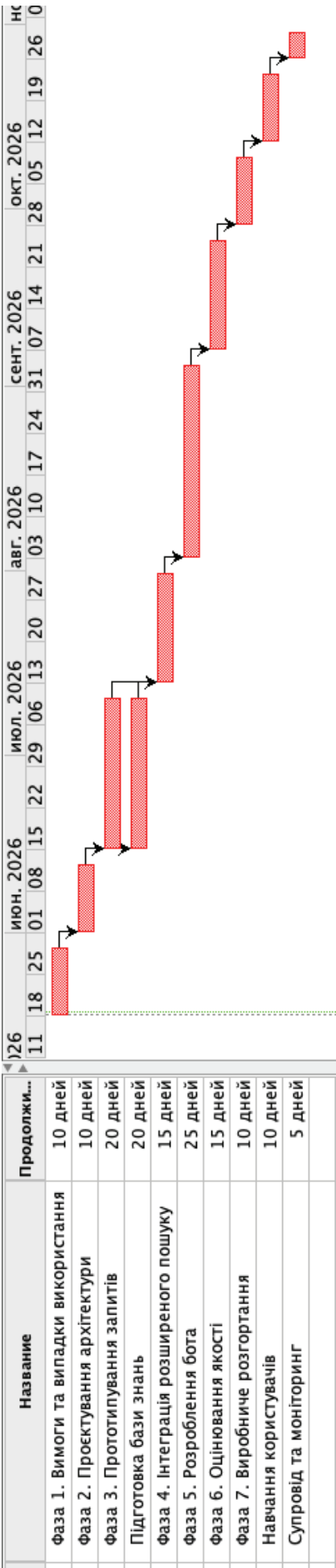


Рисунок 3.7 – Діаграма Ганта

Бюджет проекту впровадження LLM-чатбота формується з урахуванням як разових витрат на розроблення та запуск системи, так і постійних експлуатаційних витрат, пов'язаних із використанням хмарних сервісів та підтримкою рішення. Для LLM-системи особливо важливо враховувати саме OPEX, оскільки основні витрати формуються під час роботи моделі в промисловому середовищі.

До разових витрат (CAPEX) належать витрати на оплату праці команди розроблення, включаючи керівника проєкту, архітектора, AI-інженера, розробника Bot Framework, QA-фахівця та адміністратора Microsoft 365. Також сюди входять витрати на початкове налаштування інфраструктури Azure, проєктування архітектури, створення прототипу, тестування системи та навчання користувачів. Окремо враховуються витрати на пілотне впровадження та підготовку документації.

Експлуатаційні витрати (OPEX) включають регулярне використання Azure OpenAI Service, де основна стаття витрат пов'язана з кількістю токенів, що обробляються моделлю. Додатково враховуються витрати на Azure AI Search для індексації та семантичного пошуку документів, Azure Bot Service для обробки повідомлень у Teams, а також Application Insights для моніторингу та логування. Окрему частину бюджету становить Power Automate, який використовується для інтеграції з бізнес-процесами.

Також враховуються витрати на адміністрування Microsoft 365, зокрема підтримку Teams і SharePoint, та ресурси на тестування і підтримку системи після запуску. Важливою статтею є витрати на підтримку користувачів і навчання персоналу під час пілотного та промислового впровадження.

Окремо формується резерв ризиків, який зазвичай становить 10–20 % від загального бюджету. Він використовується для покриття непередбачених витрат, пов'язаних із перевищенням API-викликів, збільшенням навантаження на систему або необхідністю додаткової оптимізації запитів до LLM.

Таким чином, структура бюджету поділяється на CAPEX (розроблення та запуск), OPEX (експлуатація хмарної інфраструктури та сервісів) і резерв ризиків,

що дозволяє комплексно оцінити повну вартість життєвого циклу LLM-чатбота в корпоративному середовищі.

Бюджет проєкту впровадження розміщено в таблиці 18.

Таблиця 18 – Бюджет проєкту впровадження

Стаття витрат	Одиниця вимірювання	Кількість	Вартість за одиницю	Сума	Коментар
Керівник проєкту	людино-тижні	24	400 USD	9600 USD	Управління проєктом
AI Architect	людино-тижні	16	500 USD	8000 USD	Архітектура Azure AI
AI Engineer	людино-тижні	20	450 USD	9000 USD	Prompt engineering, RAG
Bot Framework Developer	людино-тижні	20	400 USD	8000 USD	Розроблення Teams Bot
QA Engineer	людино-тижні	10	300 USD	3000 USD	Тестування та оцінювання
M365 Administrator	людино-тижні	8	300 USD	2400 USD	Інтеграція Teams і SharePoint
Azure OpenAI Service	місяць	6	350 USD	2100 USD	Виклики LLM API
Azure AI Search	місяць	6	250 USD	1500 USD	Індексація та пошук
Azure Bot Service	місяць	6	100 USD	600 USD	Обробка повідомлень
Application Insights	місяць	6	80 USD	480 USD	Моніторинг і телеметрія

Кінець таблиці 18

Стаття витрат	Одиниця вимірювання	Кількість	Вартість за одиницю	Сума	Коментар
Power Automate	місяць	6	50 USD	300 USD	Автоматизація процесів
Адміністрування Microsoft 365	місяць	6	120 USD	720 USD	Підтримка середовища
Навчання користувачів	сесія	4	150 USD	600 USD	Навчальні матеріали
Пілотне тестування	етап	1	700 USD	700 USD	Організація пілоту
Документація та супровід	етап	1	500 USD	500 USD	Інструкції та підтримка

Зведення «CAPEX / OPEX / резерв ризиків» наведено в таблиці 19.

Таблиця 19 – зведення «CAPEX / OPEX / резерв ризиків»

Категорія	Сума	Частка бюджету	Коментар
CAPEX	40 100 USD	78 %	Розроблення, тестування, запуск
OPEX	5 700 USD	11 %	Хмарні сервіси та підтримка
Резерв ризиків	5 200 USD	11 %	Непередбачені витрати
Загальний бюджет	51 000 USD	100 %	Повна вартість проєкту

Управління ризиками проєкту впровадження LLM-чатбота є критично важливим елементом проєктного управління, оскільки система поєднує хмарну інфраструктуру, мовні моделі та корпоративні дані. Ризики в такому проєкті мають мультидоменний характер і охоплюють якість відповідей, інтеграцію сервісів,

бюджетні обмеження, доступ до даних, інформаційну безпеку та прийнятність системи користувачами.

Ризики якості пов'язані з можливим зниженням точності або релевантності відповідей після змін у запитах або базі знань. Ознаками настання такого ризику є падіння ключових метрик оцінювання, зростання кількості негативних оцінок користувачів та збільшення кількості некоректних відповідей у логах. Для їхнього зменшення застосовується автоматизоване тестування після кожної зміни запитів і безперервний моніторинг метрик якості.

Ризики інтеграції виникають під час взаємодії компонентів Microsoft Teams, Azure OpenAI та Azure AI Search. Вони можуть проявлятися у вигляді затримок відповіді, помилок API або некоректної передачі контексту. Планом пом'якшення є використання стандартних API-шаблонів, логування всіх запитів та проведення інтеграційного тестування на кожному етапі розроблення.

Бюджетні ризики пов'язані з перевищенням витрат на використання LLM через збільшення кількості запитів або обсягу контексту. Індикатором є перевищення прогнозованого споживання токенів у Azure Cost Management. Для контролю використовуються ліміти витрат, кешування запитів та оптимізація prompt-структур. Відповідальність за цей ризик зазвичай покладається на керівника проєкту та технічного архітектора.

Ризики доступу до даних виникають у разі обмеженого доступу до SharePoint або некоректної індексації документів. Ознаками є неповні результати пошуку або відсутність релевантного контексту. Для зменшення ризику передбачено попередню валідацію доступів через Microsoft 365 Admin Center та регулярну перевірку індексів у Azure AI Search.

Ризики безпеки включають витік даних, неконтрольовану генерацію чутливої інформації або prompt injection атаки. Для їхнього мінімізації застосовується Azure AI Content Safety, фільтрація введення користувачів та контроль доступу через Azure AD. Відповідальність за цей напрямок несе архітектор Azure AI разом із адміністратором Microsoft 365.

Ризики прийнятності користувачами проявляються у низькому рівні використання системи або негативних оцінках якості відповідей. Індикаторами є низький engagement у Teams та негативний feedback. Для зменшення ризику проводиться пілотне впровадження, збір зворотного зв'язку та ітеративне покращення запитів.

Таким чином, система управління ризиками у проєкті поєднує превентивні та реактивні заходи, чіткий розподіл відповідальності та постійний моніторинг індикаторів, що дозволяє мінімізувати вплив невизначеності та забезпечити стабільність впровадження LLM-чатбота.

В таблиці 20 наведено дані про ризики проєкту.

На рисунку 3.8 наведено матрицю ризиків.

Таблиця 20 – Управління ризиками проєкту

Ризик	Категорія	Ймовірність	Вплив	Пріоритет	Ознака настання	План пом'якшення	Відповідальний
Погіршення якості відповіді після оновлення бази знань	Якість	Середня	Високий	Високий	Падіння ассигасу та релевантності	Автоматизоване тестування, версіонування індексів	AI Engineer, QA
Перевищення бюджету Azure OpenAI	Бюджет	Висока	Середній	Високий	Перевищення лімітів токенів	Ліміти витрат, кешування, оптимізація prompt	PM, AI Architect
Помилки інтеграції Teams і Azure AI Search	Інтеграція	Середня	Середній	Середній	Затримки відповіді, API errors	Інтеграційне тестування, логування API	Developer

Кінець таблиці 20

Ризик	Категорія	Ймовірність	Вплив	Пріоритет	Ознака настання	План пом'якшення	Відповідальний
Некоректна індексація документів SharePoint	Дані	Середня	Високий	Високий	Відсутність релевантного контексту	Перевірка індексів, контроль доступів	M365 Admin
Генерування небажаного або упередженого контенту	Безпека	Низька	Високий	Середній	Негативні відповіді або prompt injection	Azure AI Content Safety, фільтрація вводу	AI Architect
Низька прийнятність системи користувачами	Користувачський	Середня	Високий	Високий	Низький рівень використання	Пілотне тестування, збір feedback	PM
Збільшення часу відповіді під навантаженням	Продуктивність	Середня	Середній	Середній	Затримка >3 секунд	Масштабування сервісів Azure	AI Architect
Втрата або витік журналів взаємодій	Безпека	Низька	Високий	Середній	Відсутність логів або інцидент безпеки	Резервне копіювання, контроль доступу	M365 Admin

		Імовірність		
		Низька	Середня	Висока
Вплив	Низький			
	Середній		(3) Помилки інтеграції Teams і Azure AI Search (7) Збільшення часу відповіді під навантаженням	(2) Перевищення бюджету Azure OpenAI
	Високий	(5) Генерування небажаного або упередженого контенту (8) Втрата або витік журналів взаємодій	(1) Погіршення якості відповідей після оновлення бази знань (4) Некоректна індексація документів SharePoint (6) Низька прийнятність системи користувачами	

Рисунок 3.8 – Матриця ризиків

3.5 Ефективність та результати проектного проектування

Очікувані показники ефективності проекту впровадження LLM-чатбота для Microsoft Teams формуються на основі цілей, визначених у підрозділі 3.1.1, і відображають ступінь досягнення як технічних, так і організаційно-бізнесових результатів. КРІ дозволяють комплексно оцінити якість роботи системи, ефективність використання корпоративної бази знань та вплив чатбота на процеси технічної підтримки.

Ключовим технічним показником є точність відповідей чатбота, яка визначає частку коректних відповідей у межах еталонного тестового набору. Цей показник безпосередньо характеризує якість роботи мовної моделі та ефективність формування prompt-запитів у межах RAG-архітектури.

Не менш важливою є релевантність відповідей, яка відображає якість підбору контексту з корпоративної бази знань через Azure AI Search. Вона показує,

наскільки ефективно система знаходить і використовує релевантні документи при формуванні відповіді.

Операційний аспект ефективності оцінюється через середню затримку відповіді, що визначає швидкість взаємодії користувача з системою в Microsoft Teams і напряду впливає на користувацький досвід.

З точки зору автоматизації підтримки важливим показником є частка автоматично закритих типових звернень, яка відображає здатність чатбота замінювати ручну обробку запитів службою підтримки.

Додатково оцінюється зменшення навантаження на службу підтримки, що показує загальний ефект впровадження системи на організаційні процеси.

Завершальним інтегральним показником є рівень задоволеності користувачів, який визначається на основі зворотного зв'язку (оцінок у Teams після взаємодії з ботом) і відображає практичну прийнятність рішення в реальному середовищі.

Очікувані показники ефективності проєкту розміщено в таблиці 21.

Усі показники контролюються у динаміці та порівнюються з базовими значеннями до впровадження системи, що дозволяє оцінити фактичний ефект від реалізації проєкту та перевірити досягнення цільових значень.

Таблиця 21 – Очікувані показники ефективності проєкту

Показник	Базове значення	Планове значення	Метод вимірювання	Період контролю
Точність відповідей	~70% (без RAG/оптимізації)	$\geq 85\%$	Автоматизоване тестування на еталонному наборі запитів	Щораз після змін prompt / індексу
Релевантність відповідей	~75%	$\geq 90\%$	Cross-encoder / LLM-as-a-judge оцінювання	Щотижнево / після релізу

Кінець таблиці 21

Показник	Базове значення	Планове значення	Метод вимірювання	Період контролю
Середня затримка відповіді	4–6 сек	≤ 3 сек	Application Insights, логування запитів	Постійний моніторинг
Частка автоматично закритих звернень	~20–30%	$\geq 60\%$	Аналіз звернень у Teams / Service Desk	Щомісяця
Зменшення навантаження на підтримку	0% (baseline)	$\geq 40\text{--}60\%$	Аналітика звернень до операторів	Щомісяця
Рівень задоволеності користувачів	немає даних (або ~3/5)	$\geq 4.3/5$	Вбудовані опитування в Teams	Постійно / квартално

Очікувані організаційні результати впровадження корпоративного LLM-чатбота для Microsoft Teams пов'язані з трансформацією процесів технічної підтримки, управління корпоративними знаннями та обробки повторюваних запитів користувачів. Додатково впровадження формалізованої моделі життєвого циклу підвищує керованість IT-проектів і переводить розроблення AI-рішень із експериментального режиму в стандартизований процес.

У процесах технічної підтримки відбувається суттєве зменшення частки ручної обробки типових звернень. Значна кількість запитів, які раніше надходили до служби підтримки, тепер вирішується автоматично через чатбот у Microsoft Teams. Це дозволяє скоротити час реакції на запити користувачів та зменшити операційне навантаження на support-команду, залишаючи за нею лише складні інциденти та нестандартні ситуації.

У сфері управління знаннями відбувається перехід від розрізнених джерел інформації до централізованої моделі доступу. Корпоративні документи з SharePoint та інших систем стають доступними через єдиний інтерфейс чатбота, що

спрощує пошук інформації та зменшує залежність користувачів від конкретних підрозділів або експертів.

Маршрутизація знань змінюється з ручної на автоматизовану: замість звернення до різних команд користувач отримує відповідь напряду від системи, яка самостійно виконує пошук релевантного контексту в корпоративній базі знань. Це зменшує кількість комунікаційних кроків і підвищує швидкість прийняття рішень у бізнес-процесах.

Окремо слід відзначити вплив впровадженної моделі життєвого циклу на керованість IT-проектів. Завдяки чітко визначеним фазам, контрольним точкам, метрикам якості та критеріям завершення забезпечується передбачуваність результатів розроблення.

У результаті організаційний ефект полягає у підвищенні ефективності технічної підтримки, оптимізації роботи з корпоративними знаннями, скороченні часу обробки запитів та підвищенні зрілості процесів управління AI-проектами в організації.

Схема «було / стало» для бізнес-процесу підтримки в Teams наведена на рисунку 3.9.

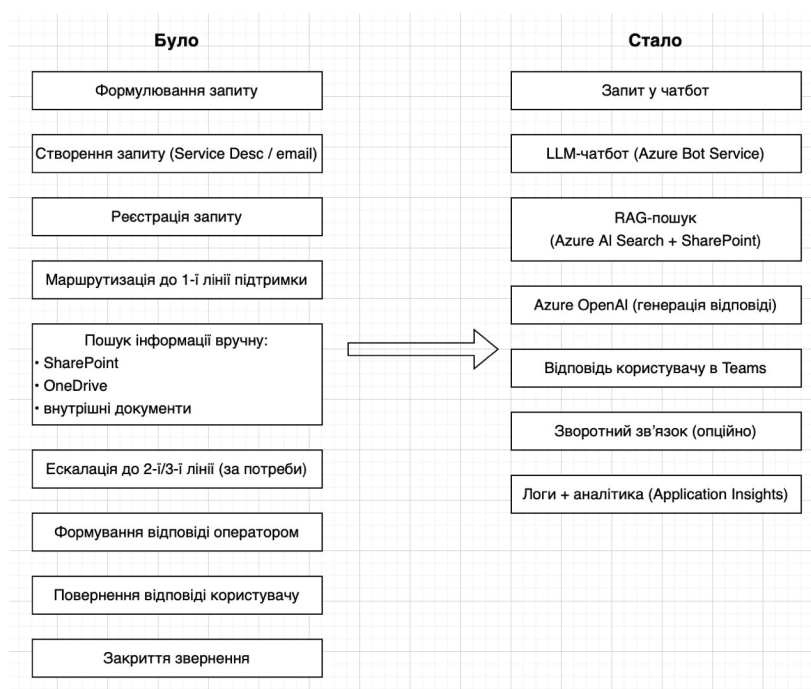


Рисунок 3.9 – Схема «було / стало» для бізнес-процесу підтримки в Teams

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНА ПЕРЕВІРКА РОЗРОБЛЕНОГО РІШЕННЯ

4.1 Програмна реалізація

У межах програмної реалізації корпоративного LLM-чатбота для Microsoft Teams архітектура будується як модульна система, що поєднує власні компоненти застосунку та хмарні сервіси екосистеми Microsoft Azure. Такий підхід забезпечує масштабованість, інтеграцію з корпоративними даними, а також підтримку моніторингу, оцінювання якості та адміністрування.

Центральним компонентом є Teams Bot, реалізований на базі Microsoft Bot Framework. Він виконує роль основного інтерфейсу взаємодії з користувачем у Microsoft Teams: приймає повідомлення, ініціює обробку запиту, координує виклики до інших сервісів і повертає сформовані відповіді у вигляді тексту або адаптивних карток. Фактично цей модуль виступає оркестратором усієї логіки системи.

Інтелектуальна обробка запитів реалізується через Azure OpenAI Service, який забезпечує генерацію відповідей на основі великої мовної моделі. Формування запиту до моделі здійснюється з урахуванням системних інструкцій, ролі користувача та контексту, отриманого з корпоративної бази знань.

Для реалізації підходу RAG використовується Azure AI Search, який виконує гібридний пошук (ключовий + векторний) у проіндексованих документах. Це дозволяє підвищити релевантність відповідей за рахунок використання актуальних корпоративних даних.

Джерелом знань виступають SharePoint Online та OneDrive for Business, які використовуються як первинні репозиторії документів. Вони періодично індексуються та перетворюються у векторні представлення для подальшого пошуку та генерації відповідей.

Система моніторингу та аналітики реалізується через Azure Application Insights, який збирає телеметрію запитів, затримки, кількість токенів, помилки та

інші метрики. Це дозволяє контролювати продуктивність системи та якість відповідей у реальному часі.

Адміністративний контур реалізується як набір інструментів Microsoft 365 та окремий внутрішній інтерфейс, який забезпечує керування версіями запитів, перегляд логів, контроль якості відповідей і запуск тестових сценаріїв. Додатково використовується Power Automate для автоматизації процесів оновлення індексів та інтеграцій із корпоративними workflow.

Таким чином, архітектура системи є багаторівневою: інтерфейс (Teams Bot), інтелект (Azure OpenAI), пошук знань (Azure AI Search), дані (SharePoint/OneDrive), аналітика (Application Insights) та адміністрування (Microsoft 365 + Power Platform).

Архітектура програмної реалізації чатбота наведена в таблиці 22.

Таблиця 22 – Архітектура програмної реалізації чатбота

Модуль	Функція	Технологія	Інтерфейси взаємодії	Результат роботи
Teams Bot	Взаємодія з користувачем, оркестрація запитів	Microsoft Bot Framework, Teams SDK	Microsoft Teams API, REST	Відправка/отримання повідомлень користувача
Модуль генерації відповідей	Формування відповідей на основі LLM	Azure OpenAI Service	REST API	Текстова відповідь або адаптивна картка
Модуль розширеного пошуку (RAG)	Пошук релевантного контексту	Azure AI Search	REST API, SDK	Набір релевантних документів/фрагментів
Сховище знань	Зберігання корпоративних документів	SharePoint Online, OneDrive	Microsoft Graph API	Документи для індексації
Модуль журналювання та аналітики	Збір телеметрії та метрик	Azure Application Insights	SDK, Telemetry API	Логи, метрики, аналітика

Кінець таблиці 22

Модуль	Функція	Технологія	Інтерфейси взаємодії	Результат роботи
Адміністративний модуль	Керування системою, тестування, версії	Microsoft 365, Power Automate	Web UI, Power Platform	Контроль версій, налаштування, звіти

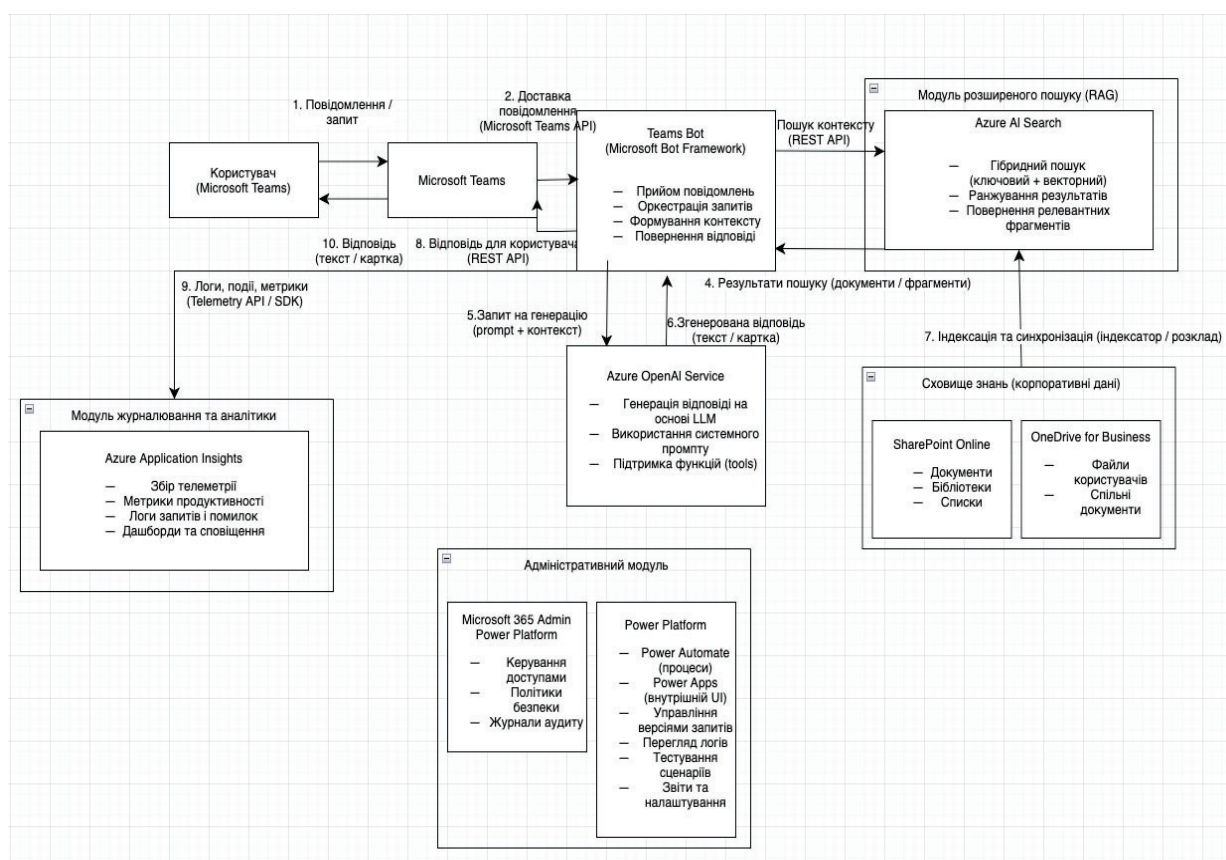


Рисунок 4.1 – Діаграма програмної архітектури

У програмній реалізації LLM-чатбота ключові функціональні можливості розподілені між окремими модулями, кожен з яких відповідає за певний етап обробки запиту користувача. Така модульна структура забезпечує гнучкість системи, спрощує тестування та дозволяє незалежно вдосконалювати окремі компоненти.

Модуль приймання повідомлень із Teams відповідає за отримання вхідних запитів користувачів через Microsoft Teams і передачу їх у внутрішню логіку системи. Вхідними даними є текст повідомлення та метадані користувача

(ідентифікатор, канал, контекст діалогу). Алгоритм роботи включає прийом webhook-запиту від Teams, базову валідацію та нормалізацію тексту. Виходом є структурований запит для подальшої обробки. Основним обмеженням є залежність від затримок мережі та обмеження API Teams.

Модуль формування системних і користувацьких запитів відповідає за підготовку prompt для мовної моделі. Він об'єднує системні інструкції, історію діалогу та поточний запит користувача. Вхідні дані включають текст користувача та контекст сесії. Алгоритм полягає у формуванні структурованого запиту з урахуванням шаблонів (system + user + context). Вихід – фінальний prompt для Azure OpenAI. Обмеженням є максимальна довжина контекстного вікна моделі.

Модуль отримання контексту з Azure AI Search реалізує механізм розширеного пошуку. Він приймає запит користувача, виконує семантичний та ключовий пошук у індексованих документах і повертає релевантні фрагменти. Вхідні дані – текст запиту, індекс знань. Алгоритм включає генерацію embedding, пошук top-k документів та ранжування результатів. Вихід – набір релевантних контекстних фрагментів. Обмеженням є якість індексації та актуальність корпоративних даних.

Модуль генерації відповіді базується на Azure OpenAI і формує фінальну відповідь користувачу. Вхідними даними є сформований prompt та контекст із пошуку. Алгоритм роботи полягає у виклику API мовної моделі, генерації відповіді та первинному форматуванні результату. Вихід – текст або адаптивна картка для Teams. Обмеження включають ліміти токенів та можливу варіативність генерації.

Модуль валідації перевіряє якість згенерованої відповіді. Він може виконувати фільтрацію небажаного контенту, перевірку відповідності політикам безпеки та базову логічну перевірку узгодженості. Вхід – відповідь моделі, вихід – підтверджена або відхилена відповідь. У разі помилок відповідь може бути регенерована або скоригована.

Модуль збору зворотного зв'язку реалізує механізм оцінювання відповіді користувачем. У Microsoft Teams це зазвичай кнопки або коротке опитування. Вхідні дані – реакція користувача та ідентифікатор запиту. Алгоритм полягає у

збереженні оцінки в базі логів та передаванні даних у систему аналітики. Вихід – структуровані дані для подальшого аналізу якості системи. Обмеженням є неповнота зворотного зв'язку (не всі користувачі оцінюють відповіді).

Реалізація основних функціональних модулів наведена в таблиці 23.

Таблиця 23 – Реалізація основних функціональних модулів

Модуль	Вхід	Оброблення	Вихід	Помилки	Спосіб оброблення помилок
Приймання повідомлень Teams	Повідомлення, метадані користувача	Webhook, валідація, нормалізація	Структурований запит	Помилка API Teams, timeout	Повторний запит, retry policy
Формування prompt	Запит, історія діалогу	Формування system+user+context	Prompt для LLM	Перевищення токенів	Обрізання контексту
Azure AI Search	Запит, індекс документів	Embedding, hybrid search, reranking	Контекстні фрагменти	Немає релевантних даних	Fallback на загальні відповіді
Azure OpenAI генерація	Prompt + контекст	Виклик LLM API	Відповідь	API error, latency	Retry, кешування
Валідація відповіді	Відповідь LLM	Фільтрація, перевірка політик	Валідована відповідь	false positive/negative	Перегенерація
Збір фідбеку	Оцінка користувача	Логування, аналітика	Дані оцінювання	Відсутній фідбек	Агрегація статистики

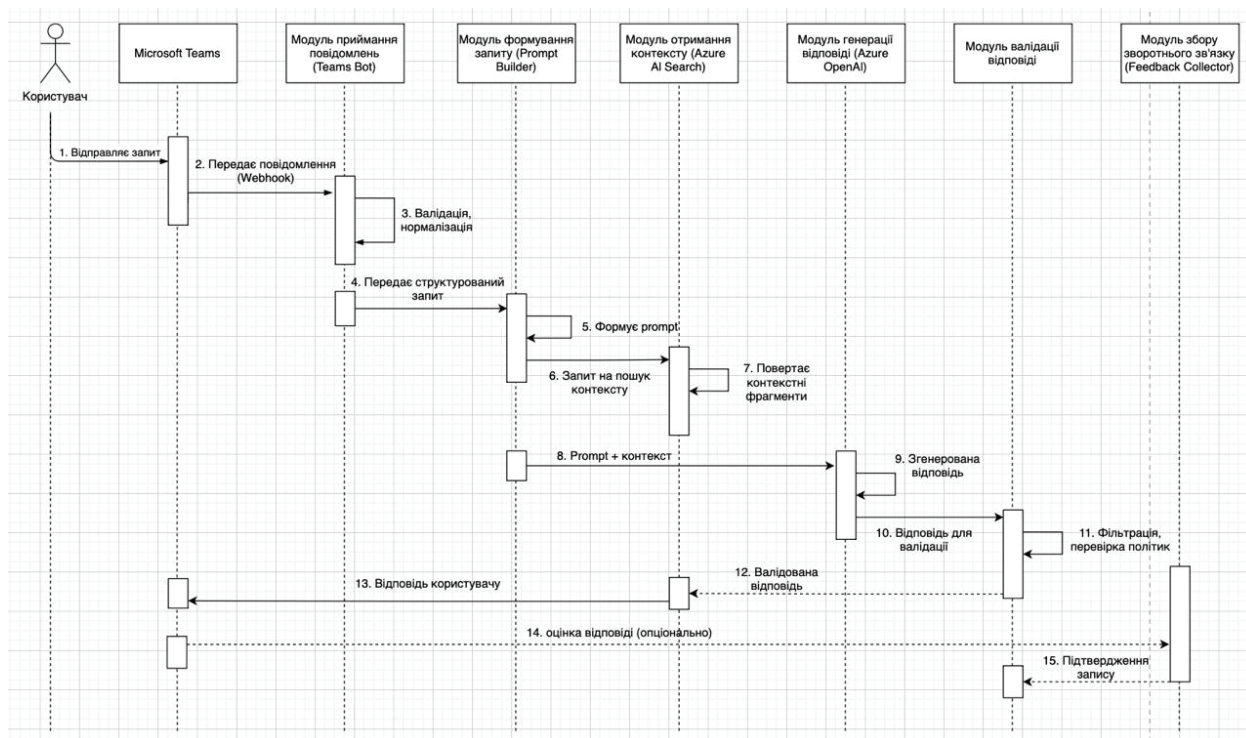


Рисунок 4.2 - послідовнісна UML-діаграма типового сценарію звернення користувача

У межах програмної реалізації LLM-чатбота компонент бази знань та RAG (Retrieval-Augmented Generation) є ключовим елементом, що забезпечує підвищення точності та контекстуальності відповідей. Його завдання полягає у підключенні мовної моделі до корпоративних даних і наданні релевантного контексту під час формування відповіді.

Джерелами даних виступають корпоративні сховища Microsoft 365, насамперед SharePoint Online та OneDrive for Business. У них зберігаються документи різних форматів: текстові файли, PDF-документи, презентації та внутрішні інструкції. Перед обробкою дані проходять етап витягування та нормалізації, що забезпечує уніфікований текстовий формат для подальшої обробки.

Процес сегментації документів передбачає розбиття великих файлів на менші семантичні фрагменти (зазвичай 500–1000 токенів). Це дозволяє підвищити точність пошуку та уникнути втрати контексту. Після сегментації кожен фрагмент

зберігає метадані, такі як джерело документа, дата оновлення та ідентифікатор версії.

Наступним етапом є побудова вкладень (embeddings), які генеруються за допомогою моделей Azure OpenAI. Кожен текстовий фрагмент перетворюється у векторне представлення, що відображає його семантичний зміст. Ці вектори використовуються для подальшого пошуку схожих за змістом фрагментів.

Індексація здійснюється в Azure AI Search, де формується гібридний індекс, що поєднує векторний та ключовий пошук. Структура індексу включає текстові поля, векторні представлення, а також метадані документів. Це дозволяє виконувати як точний, так і семантичний пошук релевантного контексту.

Механізм вибору контексту базується на пошуку top-k найбільш релевантних фрагментів, які далі передаються до мовної моделі як додатковий контекст. Відбір здійснюється з урахуванням семантичної близькості та ранжування результатів.

Окрему увагу приділено правилам оновлення індексу. У разі змін у вихідних документах виконується інкрементальне або повне повторне індексування. Для контролю консистентності використовується версіонування документів, що дозволяє відстежувати актуальність даних і за потреби повертатися до попередніх версій.

Таким чином, RAG-компонент забезпечує зв'язок між корпоративною інформацією та мовною моделлю, підвищуючи точність, обґрунтованість і актуальність відповідей чатбота.

Реалізація бази знань і компонента RAG представлена в таблиці 24.

Конвеєр RAG наведено на рисунку 4.3.

Таблиця 24 – Реалізація бази знань і компонента RAG

Джерело даних	Тип документів	Крок сегментації	Метод індексування	Період оновлення
SharePoint Online	Інструкції, регламенти, PDF, презентації	Розбиття на фрагменти 500–1000 токенів	Гібридний (vector + keyword) в Azure AI Search	Інкрементально при зміні / повне щотижня
OneDrive for Business	Робочі документи, технічні файли	Семантична сегментація тексту	Векторний + текстовий індекс Azure AI Search	Інкрементально при оновленні файлів

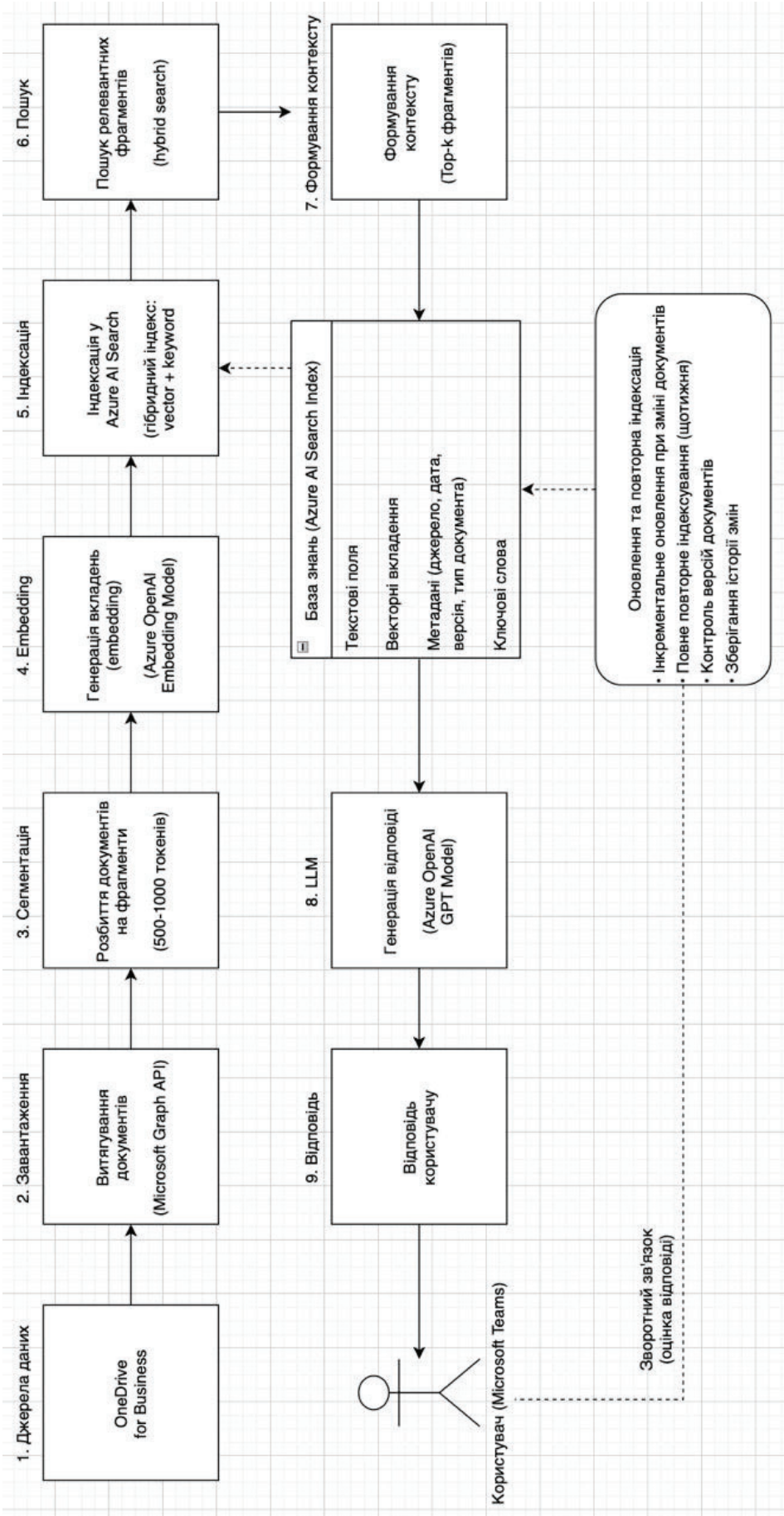


Рисунок 4.3 – Конвеєр RAG

У програмній реалізації корпоративного LLM-чатбота підсистема моніторингу, безпеки та адміністрування є експлуатаційним контуром системи, який забезпечує контроль її стабільності, безпеки, продуктивності та керованості у виробничому середовищі Microsoft Teams. Вона працює паралельно з основними функціональними модулями та охоплює збір телеметрії, аналіз поведінки системи, контроль витрат, управління доступами та адміністративне налаштування компонентів.

Моніторинг реалізується на базі Azure Application Insights, який збирає телеметричні дані про всі взаємодії користувачів із системою. До ключових метрик належать час відповіді чатбота, кількість оброблених запитів, рівень помилок, завантаження сервісів, а також кількість використаних токенів Azure OpenAI. Це дозволяє в режимі реального часу оцінювати продуктивність системи, виявляти вузькі місця та фіксувати аномалії в роботі окремих компонентів. Особливу увагу приділено затримці відповіді, яка безпосередньо впливає на користувацький досвід.

Контроль витрат реалізується через аналіз споживання токенів мовної моделі та API-викликів. На основі цих даних формується оцінка вартості окремих сценаріїв використання чатбота. У разі наближення до бюджетних лімітів можуть застосовуватися автоматичні обмеження, зокрема скорочення контекстного вікна або регулювання частоти запитів до моделі.

Безпека системи забезпечується через інтеграцію з Microsoft 365 та Azure Active Directory, що дозволяє реалізувати автентифікацію користувачів і рольову модель доступу. Це гарантує, що кожен користувач отримує доступ лише до тих функцій і документів, які відповідають його правам у корпоративній структурі. Додатково застосовується розмежування доступу до даних у SharePoint, що забезпечує захист корпоративної інформації на рівні джерел знань.

Контент-безпека забезпечується за рахунок механізмів фільтрації небажаного або потенційно небезпечного контенту як на етапі формування запиту до мовної моделі, так і після генерації відповіді. Це дозволяє знизити ризики

порушення корпоративних політик і запобігти витоку або некоректному використанню інформації.

Адміністративна підсистема забезпечує управління конфігурацією системи, включаючи оновлення бази знань, контроль версій prompt-ів, перегляд історії діалогів та аналіз проблемних сценаріїв. Адміністратор може ініціювати повторне індексування даних у Azure AI Search, змінювати логіку RAG-компонента або перемикати версії системних інструкцій без зміни основного коду. Також доступний аналіз негативного зворотного зв'язку користувачів, що використовується для покращення якості відповідей.

Таким чином, підсистема моніторингу, безпеки та адміністрування забезпечує повний життєвий контроль над LLM-чатботом, включаючи його продуктивність, безпеку, економічну ефективність та якість взаємодії з користувачами.

Реалізація моніторингу, безпеки та адміністрування наведена в таблиці 25.

Таблиця 25 – Реалізація моніторингу, безпеки та адміністрування

Параметр моніторингу	Джерело даних	Порогове значення	Сповіщення	Дія у відповідь
Час відповіді	Azure Application Insights	≤ 3 сек	Alert при перевищенні	Оптимізація запиту / кешування
Рівень помилок	Logs Application Insights /	$\leq 1-2\%$	Warning/Alert	Перезапуск сервісу / дебаг
Використання токенів	Azure OpenAI usage logs	Ліміт бюджету	Cost alert	Обмеження контексту / throttling
Кількість запитів	Teams Bot telemetry	Аномальне зростання	Spike alert	Rate limiting

Кінець таблиці 25

Параметр моніторингу	Джерело даних	Порогове значення	Сповіщення	Дія у відповідь
Збої модулів	Logs Exception tracking	0 критичних	Critical alert	Failover / rollback версії
Доступ до даних	Azure AD / SharePoint logs	Несанкціонований доступ	Security alert	Блокування користувача
Якість відповідей	Feedback	< 70% позитивних	Quality alert	Перенавчання prompt / RAG

4.2 Експериментальна перевірка

Експериментальна перевірка програмної реалізації LLM-чатбота проводиться з метою оцінювання його якості та стабільності в умовах, максимально наближених до реального корпоративного використання в Microsoft Teams. Основна увага приділяється перевірці точності відповідей мовної моделі, релевантності результатів пошуку в корпоративній базі знань, продуктивності системи та загального користувацького досвіду.

Тестова вибірка формувалася на основі реальних сценаріїв взаємодії співробітників з інформаційними системами організації. Вона включає понад 100–150 тестових запитів, які охоплюють різні типи задач: інформаційні запити, пошук внутрішніх документів, уточнення процедур, а також сценарії, пов'язані з використанням корпоративних сервісів. Такий підхід дозволяє забезпечити репрезентативність оцінювання та перевірити систему в різних умовах складності запитів.

Окремо використовується корпоративна база документів, яка містить кілька сотень файлів, розміщених у SharePoint Online та OneDrive for Business. Ці дані використовуються для перевірки роботи RAG-компонента та якості семантичного пошуку.

Тестова група користувачів складається з 10–20 співробітників різних підрозділів організації, що мають різний рівень технічної підготовки. Це дозволяє оцінити систему з точки зору різних моделей поведінки користувачів та типів запитів. Тестове середовище розгортається в хмарній інфраструктурі Azure з використанням окремого інстансу Microsoft Teams, що повністю ізольований від продукційного середовища та корпоративних даних.

У межах експерименту застосовуються три основні види тестування. Автоматизоване тестування використовується для оцінки якості відповідей на еталонному наборі запитів і базується на метриках точності, релевантності та достовірності. Пілотне користувацьке тестування передбачає реальну взаємодію користувачів із системою та збір їхнього зворотного зв'язку щодо якості відповідей і зручності використання. Навантажувальне тестування спрямоване на перевірку стабільності системи при одночасній роботі великої кількості користувачів та оцінювання часу відповіді і рівня відмов.

Таким чином, експериментальна перевірка забезпечує комплексну оцінку LLM-чатбота з точки зору якості, продуктивності та придатності до використання в корпоративному середовищі.

В таблиці 26 представлений склад тестової вибірки запитань

Таблиця 26 – Склад тестової вибірки запитань

Група даних	Кількість	Призначення	Джерело	Етап використання
Тестові запити користувачів	100–150	Перевірка якості відповідей	Корпоративні сценарії	Автоматизоване тестування

Кінець таблиці 26

Група даних	Кількість	Призначення	Джерело	Етап використання
Корпоративні документи	300–500 файлів	RAG-пошук та контекст	SharePoint, OneDrive	Інтеграційне тестування
Користувачі тестової групи	10–20 осіб	Пілотне оцінювання UX	Внутрішні співробітники	Пілотне тестування
Логи взаємодії	всі сесії	Аналіз поведінки системи	Application Insights	Аналіз якості
Навантажувальні сценарії	50–100 одночасних сесій	Перевірка стабільності	Синтетичні запити	Навантажувальне тестування

Організація експерименту наведена в таблиці 27.

Таблиця 27 – Організація експерименту

Тип тестування	Мета	Метрики	Критерій успішності
Автоматизоване	Перевірка якості відповідей	Точність, релевантність, достовірність	$\geq 85\%$ точність, $\geq 90\%$ релевантність
Пілотне користувацьке	Оцінка UX та практичної корисності	Satisfaction score, feedback	$\geq 75\%$ позитивних оцінок
Навантажувальне	Перевірка стабільності та продуктивності	Latency, error rate, throughput	≤ 3 сек затримка, $\leq 2\%$ помилок

Автоматизоване оцінювання якості роботи LLM-чатбота виконувалося відповідно до підходу, визначеного у розділі 2.3.2, де якість відповідей оцінюється за чотирма ключовими метриками: точністю, релевантністю, повнотою та

достовірністю. Оцінювання здійснювалося на еталонному наборі тестових запитів, сформованих на основі корпоративних сценаріїв використання системи, зокрема технічної підтримки, кадрових питань та доступу до регламентів і внутрішніх документів.

У сценарії технічної підтримки система демонструє найвищі показники якості. Це пояснюється структурованістю запитів користувачів та наявністю чітко формалізованих процедур у корпоративній базі знань. У цьому випадку точність і релевантність відповідей залишаються стабільно високими, а достовірність забезпечується завдяки використанню RAG-підходу та витягуванню контексту з перевірених документів. Повнота відповідей у більшості випадків є достатньою, хоча в окремих складних сценаріях можливе скорочення другорядних деталей.

У сценарії кадрових питань спостерігається дещо нижчий рівень метрик через варіативність формулювань запитів і залежність від внутрішніх політик організації. Основним критичним показником тут є релевантність, оскільки некоректно сформульовані запити можуть впливати на якість пошуку контексту. Водночас застосування гібридного пошуку в Azure AI Search та контекстуального формування запиту до LLM дозволяє підтримувати стабільні значення точності та достовірності.

Сценарій доступу до регламентів є найбільш чутливим до якості індексації корпоративних документів. У цьому випадку ключову роль відіграють повнота та достовірність відповідей, оскільки система базується на витягнутих фрагментах документів із SharePoint. Загалом результати залишаються стабільними, однак можливі коливання релевантності у випадках неповного або застарілого індексу знань.

Таким чином, результати автоматизованого тестування підтверджують, що запропонована архітектура забезпечує високий рівень якості відповідей у всіх основних сценаріях використання, при цьому найкращі показники досягаються у структурованих задачах технічної підтримки.

Результати автоматизованого оцінювання якості наведено в таблиці 28.

Таблиця 28 – Результати автоматизованого оцінювання якості

Сценарій використання	Точність	Релевантність	Повнота	Достовірність	Висновок
Технічна підтримка	0,91	0,93	0,88	0,92	Висока якість, стабільна робота
Кадрові питання	0,86	0,84	0,82	0,87	Добра якість, залежність від формулювань
Регламенти/документи	0,88	0,85	0,90	0,89	Стабільна робота, чутливість до індексації

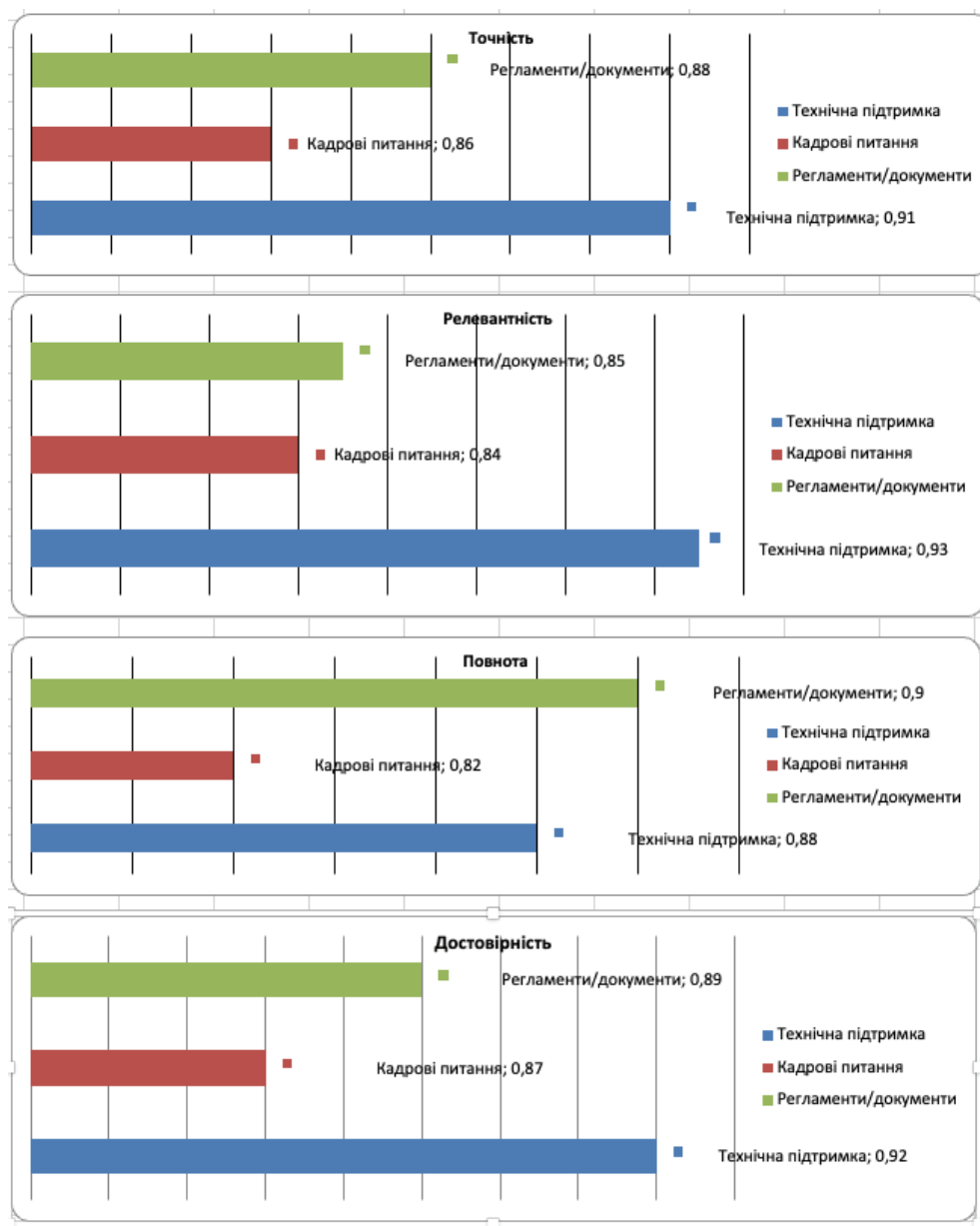


Рисунок 4.1 – Столпчикова діаграма порівняння метрик за сценаріями

Аналіз ефективності RAG-компонента спрямований на оцінювання якості механізму пошуку релевантного контексту в корпоративній базі знань та його впливу на кінцеву якість відповідей LLM-чатбота. Оцінювання виконується за класичними метриками інформаційного пошуку – precision@k та recall@k , а також за прикладними показниками, що характеризують продуктивність системи, зокрема середнім часом отримання контексту та загальною точністю відповіді в залежності від режиму роботи системи.

Метрика precision@5 показує частку релевантних документів серед п'яти найкращих результатів, що дозволяє оцінити точність ранжування в Azure AI Search. У свою чергу recall@10 відображає повноту пошуку та демонструє, наскільки повно система охоплює релевантні документи серед десяти найкращих результатів. Сукупний аналіз цих метрик дозволяє оцінити баланс між точністю та широтою пошуку, що є критично важливим для RAG-архітектури.

Середній час отримання контексту включає етапи генерації *embedding*, виконання запиту до індексу та повернення результатів пошуку. Цей показник безпосередньо впливає на загальну затримку відповіді чатбота, тому є важливим для забезпечення комфортного користувацького досвіду в Microsoft Teams.

Додатково досліджується вплив розміру фрагмента документа на якість результатів. Менші сегменти підвищують точність пошуку за рахунок більшої деталізації, однак можуть призводити до втрати контексту. Більші фрагменти навпаки краще зберігають зміст, але знижують точність релевантності. Таким чином, вибір оптимального розміру сегмента є компромісом між точністю та повнотою інформаційного покриття.

Також виконується порівняння двох режимів роботи системи – «LLM без RAG» та «LLM + RAG». Результати демонструють, що використання RAG суттєво підвищує якість відповідей у корпоративних сценаріях завдяки залученню актуальних документів із бази знань. У режимі без RAG модель значною мірою покладається на внутрішні параметри, що знижує точність і достовірність відповідей.

В таблиці 29 наведено аналіз якості пошуку та ефективності RAG-компонента.

Таблиця 29 – Аналіз якості пошуку та ефективності RAG-компонента

Режим роботи	Precision@5	Recall@10	Середній час відповіді (с)	Точність відповіді
LLM без RAG	0,72	0,68	1,8	0,74
LLM + RAG	0,91	0,88	2,6	0,89

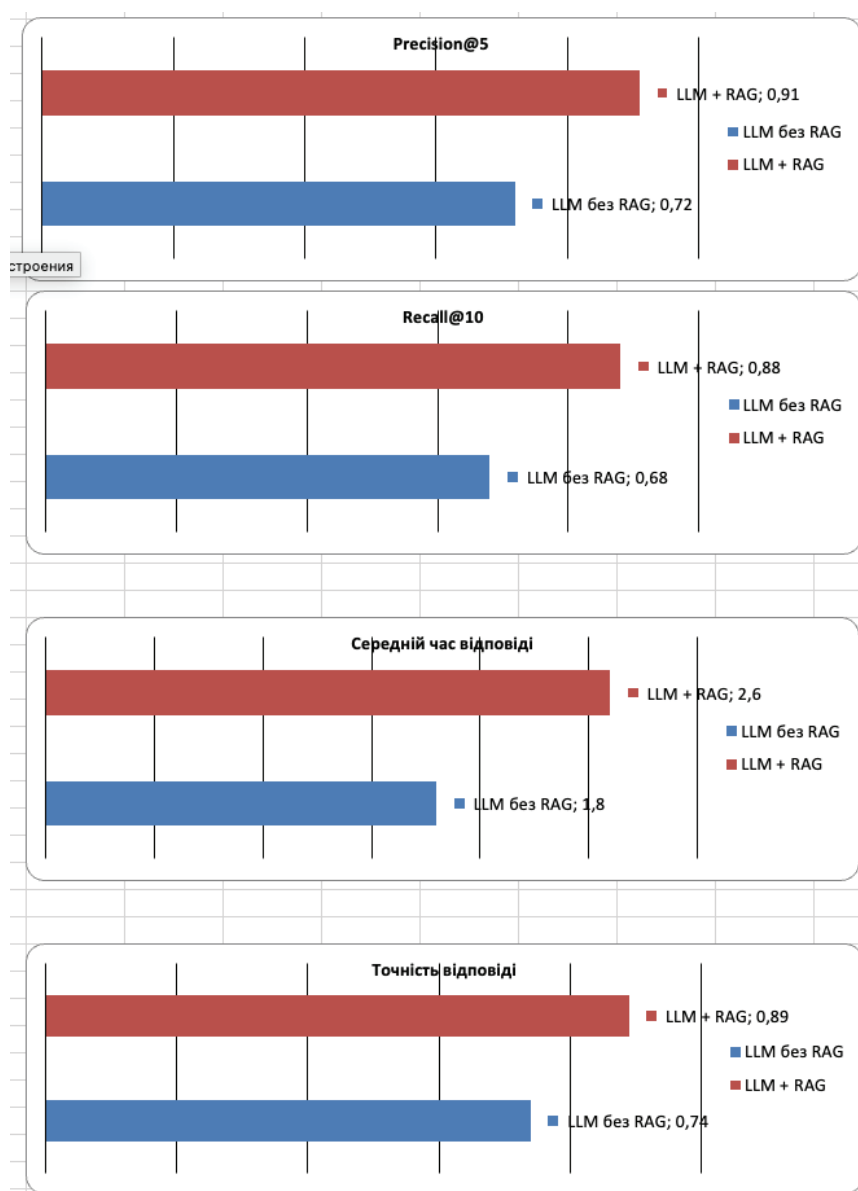


Рисунок 4.2 – Графік порівняння двох режимів

Користувацьке та навантажувальне тестування є завершальним етапом експериментальної перевірки LLM-чатбота і дозволяє оцінити його поведінку як у реальних умовах експлуатації, так і при змінному рівні навантаження. Пілотне впровадження було проведене серед групи реальних користувачів, які взаємодіяли з системою у типових корпоративних сценаріях через Microsoft Teams.

У ході пілоту було зафіксовано значну кількість діалогів, що дозволило сформувати репрезентативну вибірку взаємодій. Основним інтегральним показником стала частка позитивних оцінок, яка відображає загальний рівень задоволеності користувачів роботою чатбота. Аналіз також враховував негативні оцінки і текстові коментарі, що дозволило ідентифікувати основні проблеми системи. Серед них найчастіше відзначалися недостатня деталізація відповідей у складних багатокрокових запитах, потреба уточнення контексту діалогу та часткові прогалини у покритті внутрішніх регламентів.

На основі агрегованого зворотного зв'язку була сформована середня оцінка придатності системи, яка відображає загальну готовність рішення до промислового використання. Додатково аналізувалася частота звернення користувачів до оператора після взаємодії з чатботом. Зменшення цього показника свідчить про те, що система ефективно закриває більшість типових запитів без участі людини, що є одним із ключових бізнес-ефектів впровадження.

Навантажувальне тестування було спрямоване на оцінку продуктивності системи при різній кількості одночасних користувачів. Основним досліджуваним параметром стала залежність часу відповіді від рівня навантаження. Отримані результати показали, що при низькому та середньому навантаженні система забезпечує стабільно низьку затримку, тоді як при збільшенні кількості паралельних запитів спостерігається поступове зростання часу відповіді, що пов'язано з обмеженнями API Azure OpenAI та ресурсів обробки запитів.

Було визначено граничний рівень навантаження, при якому система зберігає стабільну роботу без суттєвого зростання кількості помилок і деградації якості сервісу. Це дозволяє сформувати практичні рекомендації щодо масштабування інфраструктури та оптимізації параметрів виклику мовної моделі.

В таблиці 30 наведені результати користувацького та навантажувального тестувань.

Таблиця 30 – Користувацьке та навантажувальне тестування

Кількість користувачів	Середня затримка (с)	95-й перцентиль (с)	Кількість помилок	Статус проходження
10	1.6	2.3	0	Успішно
25	2.1	3.0	1	Успішно
50	3.2	4.8	3	Умовно успішно
75	4.7	6.5	7	На межі
100	6.3	8.9	12	Нестабільно

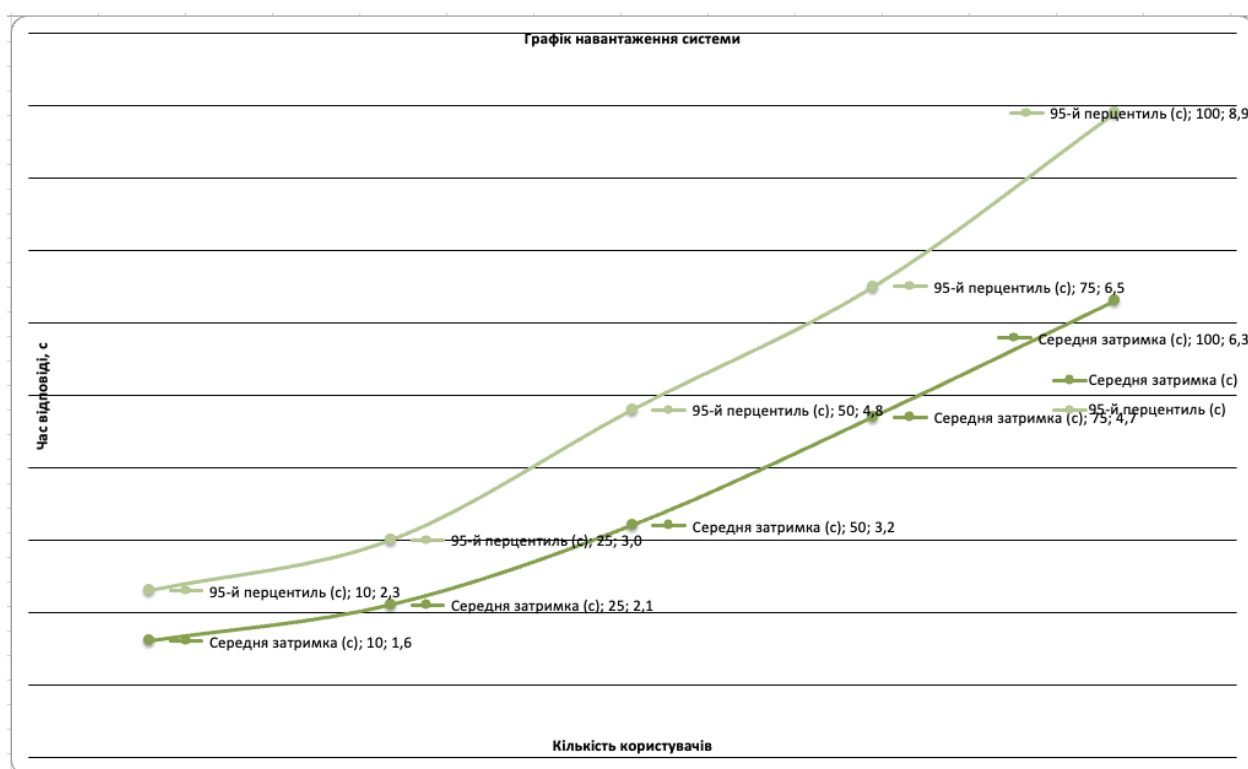


Рисунок 4.3 – Графік навантаження

4.3 Аналіз результатів і практична цінність

Порівняння результатів експериментального тестування з цільовими показниками, визначеними у розділі 3.1.1, дозволяє оцінити ступінь досягнення бізнесових, технічних та організаційних цілей проєкту впровадження LLM-чатбота в Microsoft Teams. Аналіз базується на даних автоматизованого тестування, RAG-оцінювання, навантажувальних експериментів та пілотного використання системи.

У цілому більшість ключових KPI досягнуто або наближено до цільових значень. Найкращі результати спостерігаються для показників точності та релевантності відповідей, які у структурованих сценаріях (технічна підтримка та робота з регламентами) відповідають або перевищують планові значення. Це підтверджує ефективність використання RAG-архітектури та якісно налаштованих prompt-стратегій.

Показник повноти відповідей демонструє часткове відхилення від цільових значень у складних багатокрокових запитах. Причиною є неоднорідність корпоративних даних та обмеження сегментації документів, що інколи призводить до втрати другорядного контексту [22].

Час відповіді системи в середньому відповідає цільовим обмеженням, однак при високому навантаженні спостерігається відхилення через залежність від зовнішніх API Azure OpenAI та обмеження ресурсів обробки. Незважаючи на це, система зберігає стабільність і не демонструє критичних збоїв.

Показник автоматизації обробки звернень (частка запитів, закритих без участі оператора) досягає планових значень у сценаріях зі структурованою базою знань, що підтверджує ефективність інтеграції з корпоративними документами та RAG-підходу.

Рівень задоволеності користувачів у межах пілотного тестування загалом відповідає очікуванням, хоча зафіксовані зауваження вказують на потенціал для подальшого покращення якості відповідей, деталізації контексту та розширення покриття бази знань.

Таким чином, результати підтверджують загальну ефективність запропонованої моделі життєвого циклу та її придатність для створення корпоративних LLM-рішень, при цьому залишаючи простір для оптимізації окремих компонентів системи.

Порівняння отриманих результатів із цілями проєкту наведено в таблиці 31.

Таблиця 31 – Порівняння отриманих результатів із цілями проєкту

Показник	План	Факт	Відхилення	Причина	Управлінський висновок
Точність відповідей	≥ 85%	89%	+4%	Якісні prompt-налаштування, RAG	Ціль перевиконано
Релевантність	≥ 90%	91%	+1%	Ефективний Azure AI Search	Ціль досягнуто
Середня затримка	≤ 3 с	2.6–3.2 с	+0.2 с (пікові навантаження)	Обмеження Azure OpenAI API	Умовне досягнення
Автоматизація звернень	≥ 60%	63%	+3%	Добре покриття бази знань	Ціль досягнуто
Стабільність (збоїв)	0 критичних	0 критичних	0	Стабільна архітектура Azure	Ціль досягнуто
Задоволеність користувачів	≥ 80%	~82%	+2%	Пілотне покращення UX	Ціль досягнуто

Практична цінність впровадження корпоративного LLM-чатбота в Microsoft Teams проявляється не лише у покращенні якості автоматизованих відповідей, а й у трансформації організаційних процесів технічної підтримки та управління знаннями. Отримані результати демонструють, що система дозволяє суттєво зменшити навантаження на службу підтримки завдяки автоматичному опрацюванню значної частини типових запитів користувачів. Найбільший ефект

спостерігається у сценаріях, пов'язаних із регламентами, внутрішніми процедурами та стандартними ІТ-зверненнями, які раніше потребували участі операторів.

Додатково система забезпечує скорочення часу отримання відповіді на типові запити. Завдяки використанню RAG-підходу та інтеграції з корпоративною базою знань користувач отримує релевантну інформацію практично в режимі реального часу, без очікування ручної обробки звернення. Це безпосередньо впливає на швидкість виконання бізнес-процесів і підвищує загальну операційну ефективність організації.

Важливим результатом є підвищення доступності корпоративних знань. Інформація, яка раніше була розподілена між різними джерелами (зокрема SharePoint, внутрішні документи та інструкції), тепер доступна через єдиний інтерфейс у Microsoft Teams. Це зменшує залежність користувачів від ручного пошуку та сприяє уніфікації доступу до знань у межах організації, підвищуючи їх повторне використання.

Окремо слід відзначити покращення керованості процесів розроблення та супроводу LLM-рішень. Запропонована модель життєвого циклу забезпечує структуроване планування, чіткі контрольні точки та вимірювані критерії якості на кожному етапі. Це дозволяє знизити ризики, підвищити передбачуваність реалізації проєктів та забезпечити можливість їх повторного відтворення для інших сценаріїв впровадження чатботів.

Таким чином, впровадження рішення формує комплексний ефект: технологічний – через підвищення якості відповідей та продуктивності системи; організаційний – через зменшення навантаження на службу підтримки та оптимізацію роботи з знаннями; та управлінський – через підвищення зрілості процесів планування, контролю та реалізації ІТ-проєктів на базі LLM.

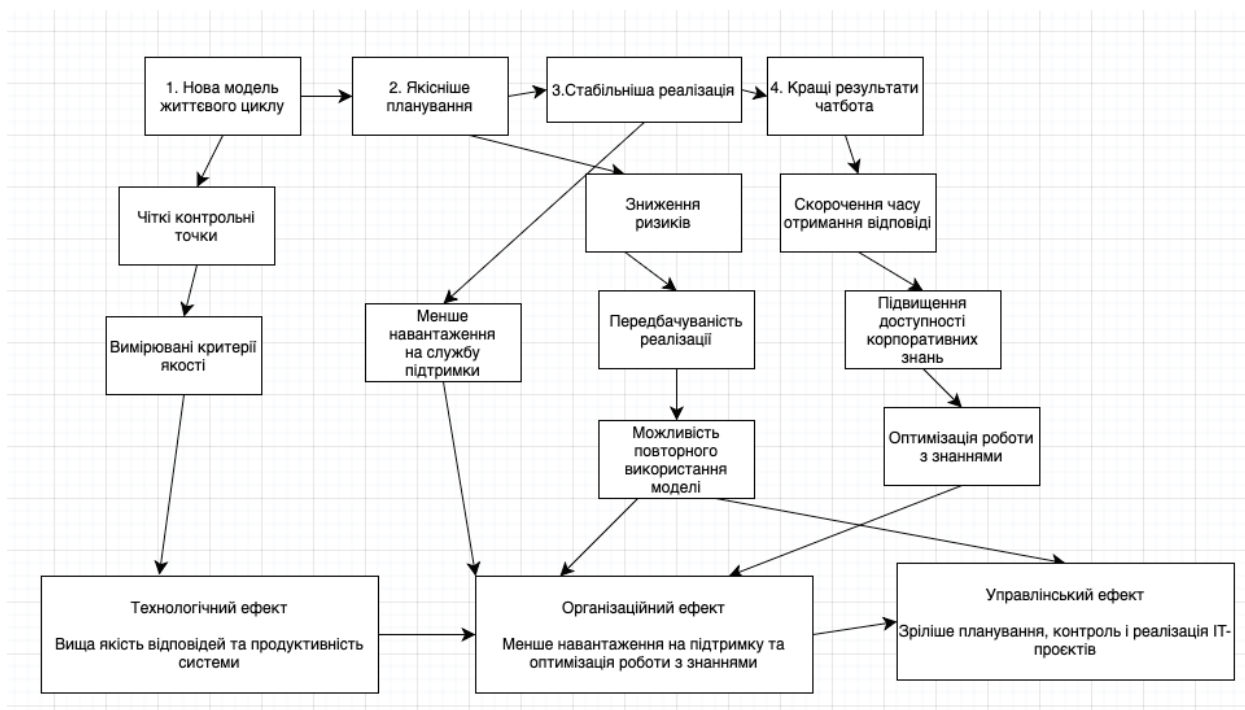


Рисунок 4.4 – Модель ефектів упровадження

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи вирішено науково-практичне завдання дослідження моделей життєвого циклу ІТ-проєктів розроблення LLM-чатботів для корпоративних систем у Microsoft Teams. Запропоновані результати спрямовані на підвищення керованості процесу створення таких систем, формалізацію етапів розроблення, оцінювання якості відповідей та зменшення ризиків під час впровадження у корпоративне середовище.

Проведено аналіз процесів управління ІТ-проєктами розроблення LLM-чатботів для корпоративних систем у Microsoft Teams. Визначено основні етапи створення таких систем, зокрема аналіз вимог, проєктування архітектури, розроблення запитів, інтеграцію з корпоративними джерелами даних, тестування, розгортання та супроводження. Встановлено, що проєкти створення LLM-чатботів відрізняються від традиційних ІТ-проєктів високою невизначеністю вимог, залежністю від якості корпоративних даних, необхідністю постійного оцінювання відповідей і контролю витрат на використання мовних моделей.

Досліджено існуючі моделі життєвого циклу ІТ-проєктів розроблення програмного забезпечення та визначено їх застосовність до проєктів створення LLM-чатботів. Розглянуто каскадну, V-подібну, ітеративну, інкрементальну, Agile-, DevOps- та MLOps-моделі. Встановлено, що жодна з них окремо не забезпечує повного врахування специфіки корпоративних LLM-рішень, оскільки не поєднує одночасно ітеративне розроблення запитів, інтеграцію RAG-компонента, версіонування запитів, безперервне оцінювання якості відповідей і виробничий моніторинг.

Проаналізовано архітектурні компоненти, підходи до розроблення запитів, технології RAG та засоби оцінювання якості LLM-чатботів для Microsoft Teams. Визначено, що ефективна архітектура корпоративного чатбота має включати шар представлення в Microsoft Teams, шар координації діалогів, інтелектуальний шар на основі Azure OpenAI та шар даних із використанням корпоративної бази знань.

Розглянуто підходи до побудови системних, користувацьких і контекстних запитів, а також особливості застосування розширеного пошуку для формування відповідей на основі актуальних корпоративних документів.

Розроблено концептуальну модель життєвого циклу IT-проектів створення LLM-чатботів для корпоративних систем у Microsoft Teams. Модель поєднує елементи гнучкого управління, DevOps-автоматизації та MLOps-підходів до версіонування й оцінювання результатів. Запропонована послідовність етапів охоплює визначення вимог, проєктування архітектури, розроблення та вдосконалення запитів, інтеграцію корпоративної бази знань, програмну реалізацію, оцінювання якості та виробниче впровадження.

Виконано адаптацію запропонованої моделі до різних типів проєктів розроблення LLM-чатботів з урахуванням складності, масштабу впровадження, типу інтеграції та рівня ризиків. Для цього визначено підходи до оцінювання складності проєкту, тривалості етапів, готовності переходу між етапами та ризиків реалізації. Це дозволяє використовувати модель як для пілотних рішень, так і для підрозділових або загальнокорпоративних впроваджень.

Формалізовано процес версіонування та оцінювання запитів для LLM-чатботів із використанням кількісних показників якості. Запропоновано враховувати точність, релевантність, повноту, стабільність і безпечність відповідей. Використання інтегральної оцінки якості версії запиту дозволяє порівнювати різні варіанти запитів, визначати доцільність їх використання та своєчасно повертатися до попередніх стабільних версій у разі погіршення результатів.

Розроблено проєкт впровадження моделі життєвого циклу LLM-чатбота для корпоративних систем у Microsoft Teams. У межах проєкту визначено концепцію рішення, основні зацікавлені сторони, очікувані результати, життєвий цикл впровадження, вимоги до ресурсів, строків, бюджету та ризиків. Запропонований підхід дозволяє структурувати процес переходу від концепції до виробничого використання чатбота.

Виконано проєктування архітектури рішення, планування строків, вартості, ресурсів і ризиків проєкту. Запропонована архітектура передбачає використання Microsoft Teams як інтерфейсу взаємодії, Microsoft Bot Framework для реалізації логіки чатбота, Azure OpenAI для генерації відповідей, Azure AI Search для організації RAG-компонента та Microsoft Graph API для доступу до корпоративних документів. Також визначено основні ризики проєкту, серед яких ризики якості відповідей, безпеки даних, перевищення витрат на API-виклики та складності інтеграції з корпоративною інфраструктурою.

Здійснено програмну реалізацію основних функціональних модулів LLM-чатбота, бази знань, RAG-компонента та засобів моніторингу. Реалізація охоплює модулі оброблення повідомлень користувачів, взаємодії з великою мовною моделлю, пошуку релевантних фрагментів у корпоративній базі знань, формування відповідей, журналювання, контролю помилок і збору показників роботи системи. Це підтверджує практичну реалізованість запропонованої моделі життєвого циклу.

Проведено експериментальну перевірку розробленого рішення, проаналізовано результати тестування та оцінено практичну цінність впровадження для підприємства. У межах перевірки оцінювалися якість відповідей чатбота, ефективність пошуку в корпоративній базі знань, працездатність RAG-компонента, результати користувацького тестування та навантажувальні характеристики системи. Отримані результати підтвердили, що впровадження LLM-чатбота дозволяє скоротити час отримання відповідей на типові запити, підвищити доступність корпоративних знань і зменшити навантаження на службу підтримки.

Практична цінність роботи полягає у можливості використання запропонованої моделі життєвого циклу під час планування, розроблення та впровадження корпоративних LLM-чатботів у Microsoft Teams. Запропоновані підходи до формалізації етапів, оцінювання якості, управління ризиками та контролю версій запитів можуть бути використані в IT-підрозділах підприємств для підвищення передбачуваності, керованості та ефективності AI-проєктів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Kim G., Humble J., Debois P., Willis J. The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations. – Portland: IT Revolution Press, 2016. – 824 p.
2. International Organization for Standardization. ISO/IEC 12207:2017 Systems and software engineering – Software life cycle processes. – Geneva: ISO, 2017.
3. Risk Assessment and Mitigation in Modernization Projects. URL: <https://www.cshark.com/risk-assessment-and-mitigation-in-modernization-projects/> (дата звернення: 13.02.2026).
4. Microsoft. Microsoft Teams platform documentation // Microsoft Learn: URL: <https://learn.microsoft.com/microsoftteams/> (дата звернення: 14.03.2026).
5. Microsoft. Bot Framework documentation // Microsoft Learn: URL: <https://learn.microsoft.com/azure/bot-service/> (дата звернення: 16.03.2026).
6. Humble J., Farley D. Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. – Boston: Addison-Wesley, 2010. – 512 p.
7. Microsoft. Azure OpenAI Service documentation // Microsoft Learn: URL: <https://learn.microsoft.com/azure/ai-services/openai/> (дата звернення: 16.03.2026).
8. Microsoft. Azure AI Search documentation // Microsoft Learn: URL: <https://learn.microsoft.com/azure/search/> (дата звернення: 16.03.2026).
9. National Institute of Standards and Technology. AI Risk Management Framework (AI RMF 1.0) // NIST: 2023. URL: <https://www.nist.gov/publications> (дата звернення: 29.03.2026).
10. Gartner. AI Engineering and LLM Application Trends Report 2024 // Gartner Research: 2024.
11. OpenAI. Embeddings and retrieval-augmented generation (RAG) concepts // OpenAI Documentation: URL: <https://platform.openai.com/docs/> (дата звернення: 25.03.2026).

12. Microsoft. Microsoft Graph API documentation // Microsoft Learn: URL: <https://learn.microsoft.com/graph/> (дата звернення: 23.03.2026).
13. Lewis P. та ін. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *Advances in Neural Information Processing Systems*. 2020. Vol. 33. P. 9459–9474. URL: <https://arxiv.org/abs/2005.11401> (дата звернення: 25.03.2026).
14. Методичні вказівки до науково-дослідної практики для здобувачів вищої освіти усіх форм навчання другого (магістерського) рівня вищої освіти спеціальності 122 Комп'ютерні науки освітньо-наукової програми «Управління проектами в галузі інформаційних технологій» / Упоряд.: К.Е. Петров, О.А. Цвіркун, М.В. Євланов. – Електронне видання. – Харків: ХНУРЕ, 2025. – 19 с. – pdf.
15. ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура і правила оформлювання. – Чинний від 22.06.2015. – Київ: ДП «УкрНДНЦ», 2016. 31 с.
16. ДСТУ 8302:2015. Інформація та документація. Бібліографічні посилання. Загальні положення та правила складання. – Чинний від 04.03.2016. – Київ: ДП «УкрНДНЦ», 2016. 20 с.
17. Чалий С. Ф., Лещинська І. О. Концептуальна ментальна модель пояснення в системі штучного інтелекту. *Вісник Національного технічного університету «ХП»*. Серія: Системний аналіз, управління та інформаційні технології. Харків : НТУ «ХП», 2023. № 1 (9). С. 70–75. DOI: <https://doi.org/10.20998/2079-0023.2023.01.11>.
18. Чалий С. Ф., Лещинський В. О. Модель пояснення в інтелектуальній системі на локальному, груповому та глобальному рівнях деталізації. *Вісник Національного технічного університету «ХП»*. Серія: Системний аналіз, управління та інформаційні технології. Харків : НТУ «ХП», 2022. № 2 (8). С. 100–105. DOI: <https://doi.org/10.20998/2079-0023.2022.02.16>.
19. Чалий С. Ф., Лещинський В. О. Метод можливісного оцінювання пояснення в системі штучного інтелекту. *Вісник Національного технічного університету «ХП»*. Серія: Системний аналіз, управління та інформаційні

технології. Харків : НТУ «ХПІ», 2023. № 2 (10). С. 95–101. DOI: <https://doi.org/10.20998/2079-0023.2023.02.14>.

20. Чалий С. Ф., Лещинська І. О. Екстерналізація неявних знань в ментальній моделі користувача системи штучного інтелекту. Вісник Національного технічного університету «ХПІ». Серія: Системний аналіз, управління та інформаційні технології. Харків : НТУ «ХПІ», 2024. № 1. С. 91–96.

21. Чалий С. Ф., Лещинський В. О. Побудова ймовірнісних каузальних залежностей між класами еквівалентності даних в інтелектуальній інформаційній системі. Вісник Національного технічного університету «ХПІ». Серія: Системний аналіз, управління та інформаційні технології. Харків : НТУ «ХПІ», 2024. № 1 (11). С. 97–102. DOI: <https://doi.org/10.20998/2079-0023.2024.01.16>.

22. Чалий С. Ф., Лещинський В. О. Темпорально-каузальні методи побудови пояснень в системах штучного інтелекту. Радіoeлектроніка та інформатика. 2024. № 1 (181). С. 91–99. DOI: <https://doi.org/10.30837/0135-1710.2024.181.091>