Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління
(повна назва)

Кафедра комп’ютерних інтелектуальних технологій та систем
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

Рівень вищої освіти перший (бакалаврський)

Інтелектуальний мультизадачний персональний
AI-асистент на основі Model Context Protocol
(тема)

Виконав:
здобувач IV курсу, групи КІУКІ-21-10
Артем ПОПОВ
(прізвище, ініціали)

Спеціальність
123 – Комп’ютерна інженерія
(код і повна назва спеціальності)

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

Освітня програма
Комп’ютерна інженерія
(повна назва освітньої програми)

Керівник: доц. каф. КІТС Наталія СЕРДЮК
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри КІТС _____ Олег РУДЕНКО
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра комп’ютерних інтелектуальних технологій та систем

Рівень вищої освіти перший (бакалаврський)

Спеціальність 123 – Комп’ютерна інженерія
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Комп’ютерна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«____»_____ 20 ___ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Попову Артему Андрійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Інтелектуальний мультизадачний персональний AI-асистент на основі Model Context Protocol

затверджена наказом університету від 21.05.2025 р. № _399Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 24 червня 20 25 р.

3. Вихідні дані до роботи Large Language Model, Model Context Protocol, мова програмування Java, фреймворк Spring Boot, Playwright, Intellij IDEA, OpenAI API, Claude Desktop, портал DL NURE

4. Перелік питань, що потрібно опрацювати в роботі
З яким середовищем буде інтегровано інтелектуального асистента?
Які функції має виконувати розроблений інтелектуальний асистент?
Які інструменти та технології будуть використані для розробки?
Як користувач буде взаємодіяти з персональним асистентом?
Які функції має виконувати розроблений інтелектуальний асистент?

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) _____
Діаграми, схеми, лістинг програми, скріншоти порталу DL NURE, скріншоти взаємодії з асистентом

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів робо-ти	Примітка
1	Ознайомлення з задачею, порівняння можливих способів реалізації асистенту	21.05.2025-23.05.2025	Виконано
2	Вивчення документації та іншого інформаційного матеріалу що може стати у нагоді	24.05.2025-27.05.2025	Виконано
3	Розробка програми	28.05.2025-03.06.2025	Виконано
4	Тестування асистенту	04.06.2025-06.06.2025	Виконано
5	Оформлення пояснювальної записки	07.06.2025-13.06.2025	Виконано
6	Надання до ЕК	24.06.2025	Виконано

Дата видачі завдання 21 травня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. каф. КІТС Сердюк Н.М.
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 67 с., 12 рис., 2 табл., 2 дод., 9 джерел.

MODEL CONTEXT PROTOCOL, LARGE LANGUAGE MODEL, МОВА ПРОГРАМУВАННЯ JAVA, STANDARD INPUT OUTPUT, ПЕРСОНАЛЬНИЙ АСИСТЕНТ.

Метою кваліфікаційної роботи є проектування та розробка персонального інтелектуального асистенту на основі штучного інтелекту з використанням технології Model Context Protocol для інтеграції в існуюче середовище для оптимізації процесів взаємодію людини з системою.

Дослідженню під час роботи підлягають галузі застосування інтелектуальних асистентів, існуючі технології для інтеграції штучного інтелекту в сучасні системи, інструменти для розробки клієнтської та серверної частини персональних асистентів .

Розроблений асистент має інтеграцію з навчальною платформою DL NURE, завдяки чому здатний швидко та якісно відповідати на питання навігації та пошуку інформації. В ході роботи було розроблено програму, що надає стандартизований інтерфейс для запитів на отримання інформації з порталу у реальному часі.

ABSTRACT

Explanatory note to the qualification work: 67 pp, 12 figures, 2 tables, 2 appendices, 9 sources.

MODEL CONTEXT PROTOCOL, LARGE LANGUAGE MODEL, JAVA PROGRAMMING LANGUAGE, STANDARD INPUT OUTPUT, PERSONAL ASSISTANT.

The aim of the qualification work is to design and develop a personal intelligent assistant based on artificial intelligence using Model Context Protocol technology for integration into the existing environment to optimize human-system interaction processes.

The research during the work covers the areas of application of intelligent assistants, existing technologies for integrating artificial intelligence into modern systems, and tools for developing the client and server parts of personal assistants.

The developed assistant is integrated with the DL NURE learning platform, thanks to which, it is able to quickly and efficiently answer questions about navigation and information search. During the work, a program was created to provide a standardized interface for real-time requests to get information from the portal.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

LLM – велика мовна модель (англ. Large Language Model)

MCP – протокол модель-контекст (англ. Model Context Protocol)

API – прикладний програмний інтерфейс (англ. Application Programming Interface)

STDIO – стандартний ввод-вивід (англ. Standard Input / Output)

Java – об'єктно-орієнтована мова програмування, випущена 1995 року компанією «Sun Microsystems» як основний компонент платформи Java. Java значно запозичила синтаксис із C і C++. Зокрема, взято за основу об'єктну модель C++, проте її модифіковано.

Spring – фреймворк для мови програмування Java, що значно полегшує менеджмент об’єктами під час виконання програми та надає зручні абстракції для обробки інтернет-комунікацій.

Playwright – бібліотека з відкритим вихідним кодом для автоматизації роботи з браузерами та сканування web-сторінок, розроблена компанією Microsoft.

ВСТУП

У сучасному світі тенденція розвитку штучного інтелекту є беззаперечною. З кожним роком зростають можливості інтелектуальних асистентів, побудованих з використанням штучного інтелекту. Навіть попри те, що класичні асистенти зазвичай обмежені в плані доступних інтерфейсів взаємодії з оточенням, постійно з’являються нові інструменти та технології, спрямовані на посилення інтеграції асистентів в повсякденне життя.

Завдяки подібним технологіям стає можливою автоматична або контрольована взаємодія асистентів з зовнішніми системами. Такі асистенти здатні набагато ефективніше шукати, аналізувати та генерувати інформацію. Розробка та застосування автономного персонального асистенту дозволить автоматизувати повсякденні процеси, що вимагають накладних витрат часу та уваги.

Для створення персонального асистенту, під час кваліфікаційної роботи необхідно провести дослідження галузей застосування інтелектуальних асистентів, та визначити яка з існуючих систем отримає користь від інтеграції зі штучним інтелектом. Окрім цього необхідно проаналізувати існуючі технології, що спростять процес розробки та налагодження системи персонального асистенту.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Розвиток штучного інтелекту

Штучний інтелект (ШІ) — це здатність комп'ютерних систем виконувати задачі, що зазвичай асоціюються з людським інтелектом. Такими задачами можуть бути процеси, що базуються на навчанні, міркуванні та прийнятті рішень. Завдяки штучному інтелекту, комп'ютерні системи здатні «бачити», розуміти та інтерпретувати інформацію, а також генерувати рекомендації. Можливість імітувати когнітивні функції людини є результатом численних досліджень у галузі інформатики, математики та нейробіології, які призвели до створення алгоритмів, здатних обробляти величезні обсяги даних та виявляти в них складні закономірності.

Історичний розвиток штучного інтелекту розпочався в середині XX сторіччя, коли вчені вперше почали досліджувати можливість створення машин, що здатні мислити. Протягом наступних десятиліть ця галузь переживала періоди як прогресу так і відносних стагнацій, які дослідники називають «зимами ШІ» [1]. Однак революційні прориви останніх років, особливо в галузі глибокого навчання та нейронних мереж, призвели до створення систем, які демонструють вражаючі результати в розпізнаванні зображень, обробці людської мови та прийнятті складних рішень. Сучасні досягнення в області обчислювальної потужності, доступність великих даних та вдосконалені алгоритми створили ідеальні умови для процвітання технологій ШІ.

1.2 Інтелектуальні асистенти

З появою штучного інтелекту почали з'являтися комп’ютерні системи, які виконують задачі інтелектуальних асистентів. Такі системи призначені для створення штучних помічників, які можуть надавати допомогу в спеціалізованих областях. З врахуванням сучасного рівня розвитку ШІ, такі помічники здатні виконувати завдання, які раніше вимагали обов'язкової присутності живої людини в технічному процесі. Інтелектуальні помічники не тільки здатні виконувати завдання, властиві людям, але й можуть робити це набагато швидше завдяки високій обчислювальній потужності сучасних пристроїв.

Ці інтелектуальні системи оснащені широким набором інструментів, які значно розширюють їхні можливості порівняно з традиційними програмними рішеннями. Серед цих інструментів особливе місце займають інтеграційні API, що дозволяють асистентам під’єднуватися до зовнішніх сервісів і джерел даних, отримуючи доступ до актуальної інформації з різних джерел. Завдяки таким інтеграціям асистенти можуть взаємодіяти з хмарними платформами, системами керування бізнесом, соціальними мережами та спеціалізованими галузевими сервісами, створюючи єдину екосистему для вирішення складних задач.

Можливості пошуку в інтернеті та збору інформації перетворили сучасних інтелектуальних асистентів на потужні аналітичні та дослідницькі інструменти. Вони здатні відстежувати новини в режимі реального часу, аналізувати тенденції фінансових ринків, збирати актуальну інформацію та перевіряти факти з різних джерел. Ця функціональність є особливо цінною в умовах динамічних середовищ, де актуальність інформації часто визначає коректність прийнятих рішень.

Обробка файлів у різних форматах також є однією з ключових можливостей сучасних систем штучного інтелекту. Інтелектуальні асистенти здатні читати та аналізувати документи, отримувати з них структуровану

інформацію та виконувати аналіз вмісту. Технології оптичного розпізнавання символів дозволяють працювати навіть зі сканованими документами та зображеннями, перетворюючи їх на текст. Мультимедійні можливості включають обробку відео- та аудіо-контенту, що відкриває нові перспективи для автоматизації процесів в освіті, ЗМІ та розвагах.

Практичне застосування інтелектуальних помічників охоплює широкий спектр галузей та сфер діяльності. У бізнесі вони революціонізують процеси автоматизації офісних робіт, від обробки документів і створення звітів до планування зустрічей та керування проектами. Медична галузь може використовувати ШІ-помічників для аналізу медичних зображень, допомоги у постановці діагнозів та ведення електронних карт пацієнтів. В освіті такі помічники можуть створювати персоналізований процес навчання, адаптуючи матеріали до індивідуальних потреб студентів і надаючи миттєвий зворотний зв’язок стосовно прогресу у навчанні.

1.3 Сучасні проблеми, що викликані існуванням ШІ

Поряд з незаперечними перевагами, розвиток ШІ-помічників ставить перед суспільством ланку викликів та етичних питань. Актуальним є питання конфіденційності даних, оскільки ці системи мають доступ до величезних обсягів особистої інформації користувачів. Крім того, ШІ-асистенти не є досконалими і часом можуть генерувати неточну або помилкову інформацію, що особливо критично у сферах, де точність є обов’язковим критерієм відповіді. Необхідність забезпечення захисту від можливих зловживань вимагає розробки нових стандартів і регуляторних механізмів.

До соціальних наслідків широкого впровадження інтелектуальних помічників можна віднести потенційну втрату робочих місць у деяких сферах діяльності та необхідність перекваліфікації працівників. Водночас створюються нові професії та спеціальності, пов'язані з розробкою, налаштуванням та обслуговуванням систем штучного інтелекту.

Майбутні перспективи розвитку інтелектуальних помічників вказують на їхню еволюцію в бік ще більшої автономії та спеціалізації. Очікується розробка систем, здатних навчатися та адаптуватися до нових завдань без втручання людини. Інтеграція з технологіями доповненої та віртуальної реальності створить нові форми взаємодії між людьми та системами ШІ.

1.4 Постановка задачі

Метою атестаційної роботи є проектування та розробка мультизадачного персонального AI-асистенту, який матиме можливість інтеграції з зовнішніми системами. Головною метою такого асистента має бути оптимізація щоденних процесів, що вимагають витрати людського ресурсу. Для досягнення поставленої мети треба:

– провести аналіз існуючих технологій для створення інтелектуальних асистентів, щоб визначити набір інструментів, які будуть використані для розробки асистенту;

– провести порівняння засобів інтеграції персональних асистентів з зовнішніми системами та враховуючи отриману інформацію, спроектувати алгоритм інтеграції інтелектуального асистенту з існуючою системою.

Для того щоб обрати систему, з якою буде відбуватися інтеграція, треба також визначити які з існуючих систем є частинною повсякденного життя і можуть отримати користь від з’єднання зі штучним інтелектом.

2 ВИБІР ТЕХНОЛОГІЙ ТА ІНСТРУМЕНТІВ

2.1 Технології для створення інтелектуального асистенту

Мультизадачний інтелектуальний асистент є комплексною системою, що має бути здатна автономно виконувати ряд спеціалізованих задач з мінімальною кількістю втручань зі сторони користувача. Така система не може бути розроблена лише традиційними засобами програмування, оскільки вони базуються на створенні попередньо визначених правил, що задають поведінку системи. При такому підході, майже неможливо створити гнучкого асистента, здатного адаптуватися під потреби користувача в реальному часі.

Виходом з ситуації є використання штучного інтелекту. Впродовж останніх років, людство досягло значних успіхів в розробці штучного інтелекту, що відкрило нові можливості для автоматизації процесів, що попередньо вимагали безпосереднього контролю зі сторони живого оператору. Серед великої кількості способів створення штучного інтелекту, одним з найпопулярніших є великі мовні моделі (LLM).

LLM є невід’ємною частиною інтелектуальних асистентів, що взаємодіють з даними, представленими у вигляді тексту. Завдяки мовним моделям можна виконувати обробку повідомлень в текстовому чаті, аналіз вмісту веб сторінок, підсумовування документів, та генерацію відповідей стосовно навчальних матеріалів.

Великі мовні моделі – це мовні моделі, що розроблені на основі штучного інтелекту, який використовує глибоке навчання (Deep Learning, рис. 2.1).

Рисунок 2.1 – Принцип роботи Deep Learning

Такі моделі будуються на базі рекурентних або трансформерних архітектур та тренуються на дуже великих обсягах текстових даних [2].

Трансформерна архітектура, яка була вперше згадана у роботі "Attention is All You Need", стала проривом у побудові великих мовних моделей [3]. Саме вона лягла в основу багатьох сучасних мовних моделей. Цей підхід дозволив значно покращити обробку довгострокових залежностей у тексті завдяки наявному механізму уваги, що дозволяє враховувати контекст та зв'язки між словами (рис. 2.2) [4].

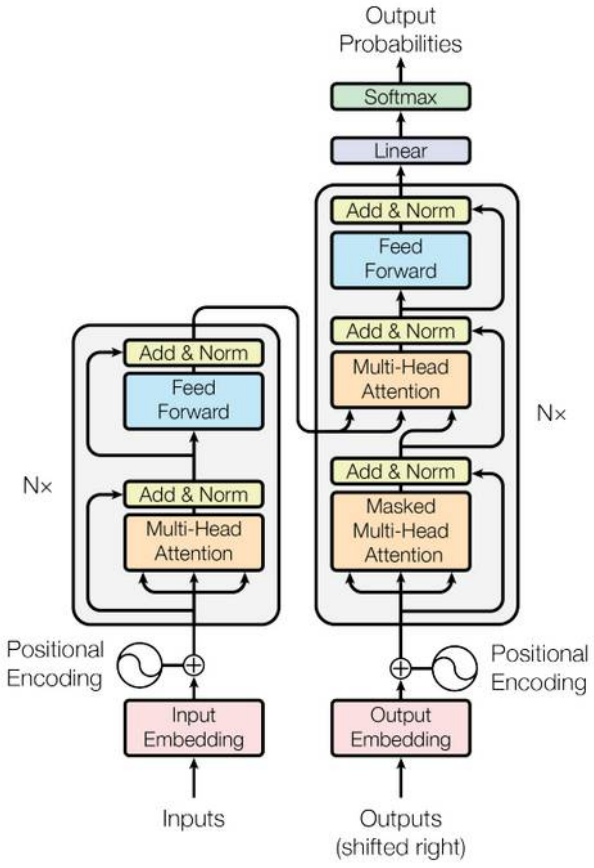

Рисунок 2.2 – Трансформерна архітектура моделі

2.2 Існуючі способи інтеграції мовних моделей у прикладні рішення

Через надзвичайну продуктивність великих мовних моделей, їх використання стало дуже популярним в безлічі прикладних областей, що призвело до виникнення великої кількості інструментів, що спрощують інтеграцію LLM в існуючі рішення.

Найпопулярнішим способом наразі є використання REST API для генерації текстових відповідей шляхом надсилання HTTP запитів до API різних постачальників великих мовних моделей (OpenAI, Anthropic, Google, тощо) [5].

Також стали зростати в популярності модульні фреймворки для різних мов програмування, головною метою яких є надання високорівневих абстракцій для роботи з LLM, різними інструментами та джерелами даних. Прикладами таких фреймворків є LangChain та LlamaIndex [6].

Одним з найсучасніших рішень проблеми інтеграції мовних моделей з зовнішніми середовищами є Model Context Protocol (MCP). MCP це стандарт з’єднання ШІ-асистентів зі спеціалізованими додатками, які можуть надавати доступ до інформації, інструменти взаємодії з окремими оточеннями, або інші програмні інтерфейси.

Model Context Protocol був опублікований наприкінці 2024 року компанією Anthropic [7]. У день випуску протоколу було оголошено про три основні складові MCP, а саме:

– специфікацію та комплекти для розробки програмного забезпечення [1]

– підтримку локальних MCP-серверів в додатку Claude Desktop

– публічний репозиторій з великою кількістю готових MCP-серверів

Зі слів Anthropic, відкриті технології, такі як Model Context Protocol, – є мостами, що з’єднують штучний інтелект із реальними додатками, забезпечуючи доступність та прозорість.

В основі MCP лежить клієнт-серверна архітектура, в якій один клієнт здатний звертатися до декількох серверів одночасно (рис. 2.3) [8].

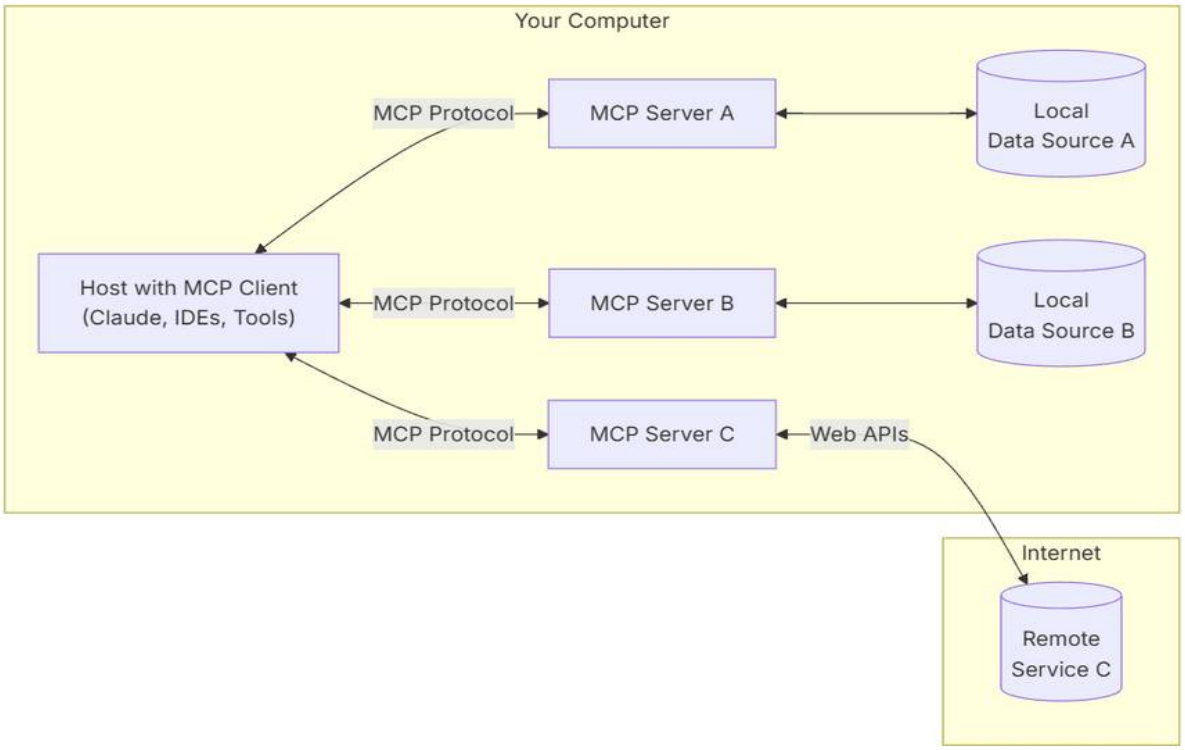

Рисунок 2.3 – Схема роботи протоколу MCP

Серед основних складових MCP архітектури можна відокремити наступні компоненти:

– MCP хости – програми по типу Claude Desktop, інтегровані середовища розробки, або інші ШІ-інструменті, що можуть користуватися зовнішніми інтерфейсами завдяки MCP

– MCP клієнти – клієнти, що підтримують з’єднання з MCP серверами

– MCP сервери – програми, що надають інтерфейси для спеціалізованих можливостей через Model Context Protocol

– локальні джерела даних – файли, бази даних, та сервіси, що доступні MCP серверам та розташовані в локальному середовищі

– віддалені сервіси – зовнішні системи, що доступні через інтернет (наприклад API) та до яких може під’єднатися MCP-сервер

Для комунікації між окремими компонентами архітектури, протокол використовує повідомлення в форматі JSON-RPC.

JSON-RPC — це спрощений протокол виклику віддалених процедур (RPC) [9]. Цей протокол задає формат запитів та відповідей при комунікації між двома віддаленими пристроями (лістинг 2.1).

Лістинг 2.1 – приклад повідомлень під час комунікації за протоколом JSON-RPC

```
      --> {"jsonrpc": "2.0", "method": "subtract", "params": {"subtrahend": 23, "minuend": 42}, "id": 3}
      <-- {"jsonrpc": "2.0", "result": 19, "id": 3}

      --> {"jsonrpc": "2.0", "method": "subtract", "params": {"minuend": 42, "subtrahend": 23}, "id": 4}
      <-- {"jsonrpc": "2.0", "result": 19, "id": 4}
```

2.3 Галузі застосування інтелектуальних асистентів на базі MCP

Враховуючі поточні можливості штучного інтелекту, перелік областей застосування інтелектуальних асистентів є майже безмежним. Якщо розглядати інтелектуальні асистенти, що базуються на великих мовних моделях, їх застосування можливе в будь якому сценарії, що вимагає обробки будь яких даних, представлених у текстовому вигляді. Такими задачами можуть бути задачі з пошуку інформації, генерації висновків на основі різних вхідних даних, та створення текстів відповідно до заданих вимог. З появою Model Context Protocol, спектр можливостей інтелектуальних асистентів стає ще більшим. Завдяки можливості самостійно взаємодіяти з навколишнім середовищем, інтелектуальні асистенти можуть автоматизувати людські процеси в безлічі прикладних областей. Такі асистенти можуть бути інтегровані в системи бізнес-планування для автоматичного ведення звітів, чати в реальному часі для надання персональної допомоги, освітні платформи для оптимізації навчальних процесів, тощо.

2.4 Технології реалізації MCP серверу

Разом зі специфікацією Model Context Protocol, було опубліковано набір SDK (наборів для розробки програмного забезпечення) для більшої частини найпопулярніших мов програмування. Таким чином, SDK отримали такі мови як C#, Java, Kotlin, Python та інші. Окрім офіційних наборів для розробки, деякі постачальники фреймворків для окремих мов програмування також розробили бібліотеки, що спрощують розробку додатків на базі Model Context Protocol. Наприклад у фреймворку Spring з’явився проект Spring AI MCP, що дозволяє інтегрувати MCP логіку у Java-додатки, написані з використанням Spring.

2.5 Технології реалізації MCP клієнту

При виборі MCP клієнту, є декілька варіантів, а саме:

- розробка власного MCP клієнта
- використання клієнтів з відкритим вихідним кодом
- використання додатку Claude Desktop

Головною проблемою варіанту з розробкою власного клієнта є накладні витрати часу на розробку клієнту. Наразі існують фреймворки і бібліотеки, що полегшують цю задачу, але навіть враховуючи це, переваг у розробці власного клієнту на поточному етапі небагато. В теорії, цей варіант є найкращим в довгостроковій перспективі, оскільки він дозволить без обмежень змінювати інтерфейс користувача за потреби, надаючи необхідні інструменти та елементи керування.

Також можна розглянути існуючі MCP-клієнти з відкритим вихідним кодом. Незважаючи на відносно недовгий строк існування MCP протоколу, вже існує велика кількість клієнтів та серверів з відкритим вихідним кодом (рис. 2.4).

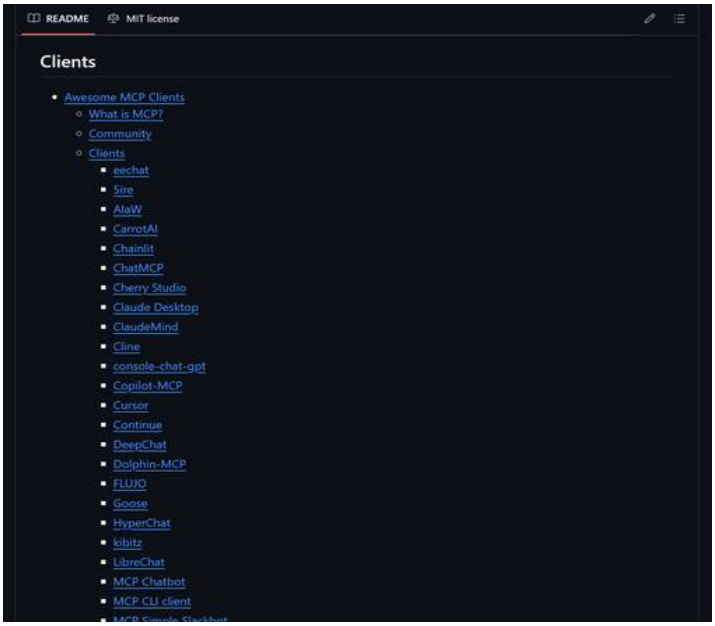

Рисунок 2.4 – Перелік популярних MCP клієнтів з відкритим вихідним кодом

Проте, у цього варіанта також є свої недоліки. По-перше, дуже важко передбачити розвиток проекту з відкритими вихідним кодом (він може перестати отримувати оновлення та застаріти). По-друге, такі клієнти є обгортками для роботи з локальними або віддаленими моделями, тобто або має бути встановлена локальна модель, або має використовуватися API постачальників LLM, яке зазвичай є платним.

Останнім, і на поточному етапі найоптимальнішим варіантом є використання додатку Claude Desktop. Claude Desktop це додаток, розроблений компанією Anthropic, що дозволяє спілкуватися з мовними моделями Claude. З моменту релізу MCP протоколу, в цей додаток також додали підтримку локальних MCP-серверів. Claude Desktop також має свої обмеження, наприклад обмеження на кількість повідомлень в безкоштовному плані, але якщо не зловживати асистентом, щоденного обмеження вистачає на вирішення питань, що могли б виникати під час навчання.

3 РОЗРОБКА АРХІТЕКТУРИ ПРОЕКТУ

3.1 Інтелектуальний асистент для порталу DL NURE

Головним інструментом студенту, що навчається в Харківському Національному Університеті Радіоелектроніки є портал DL NURE (рис. 3.1). На ньому представлено основні інтерфейси для стандартизації процесу отримання інформації та надання студентських робіт на перевірку.

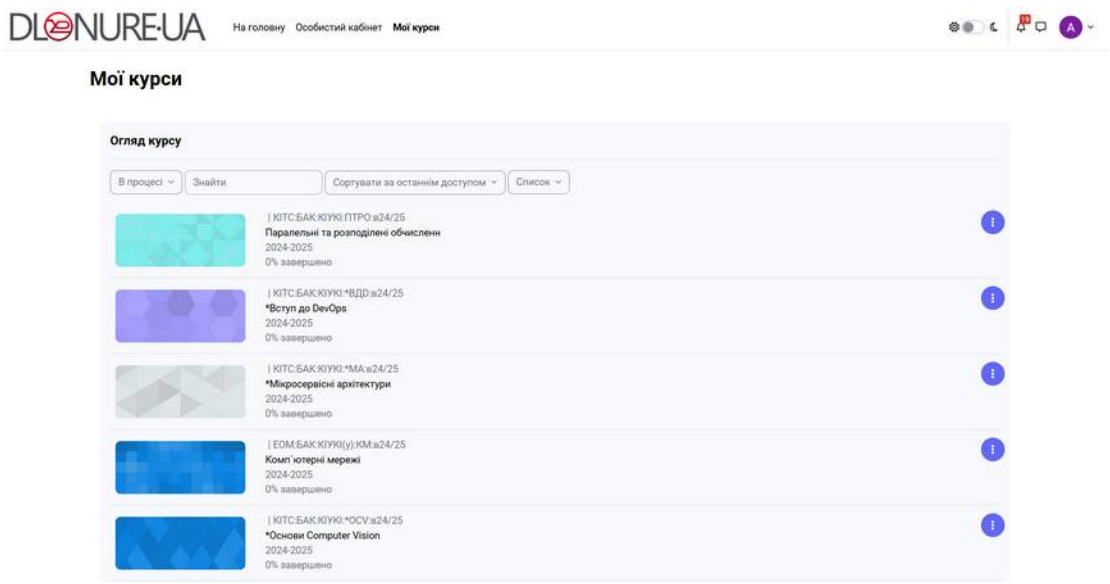

Рисунок 3.1 – Веб-інтерфейс порталу DL NURE

Головна мета порталу DL NURE – надання єдиної платформи для роботи з ресурсами, необхідними для навчання студентів в ХНУРЕ. На момент написання цієї роботи основними функціональними можливостями порталу для студента є:

– перегляд переліку поточних курсів, на які записаний студент
– перегляд матеріалів конкретних курсів (посилання на лекції або лабораторні роботи, лекції, методичні вказівки)
– завантаження самостійно виконаних робіт для подальшої перевірки викладачами та отримання оцінки

– проходження тестів з різних дисциплін
– самостійне ведення відвідування для деяких дисциплін

Портал значно полегшує багато аспектів навчання студента, але іноді все одно можуть виникати проблеми стосовно пошуку та роботи з ресурсами. Наприклад, в умовах великого навантаження поточними задачами, що потребують уваги студента та при наявності великої кількості дисциплін, пошук необхідної інформації та аналіз матеріалу може ставати дещо проблематичним. Також, оскільки в рідкісних випадках деякі матеріали можуть знаходитись поза порталом DL NURE, є вірогідність того що користувач витратить час на пошуки інформації безрезультатно.

Деякі з проблем, що можуть виникнути у студента під час роботи з навчальною платформою, можна вирішити шляхом розробки мультизадачного інтелектуального асистенту, інтегрованого у навчальне середовище.

З програмної точки зору – головним обмеженням створення подібного асистенту є відсутність у DL NURE публічного API, що суттєво ускладнює розробку будь-яких інтеграцій через неможливість отримання / надсилання інформації традиційним програмним шляхом.

3.2 Визначення переліку можливостей інтелектуального асистенту

Можливості асистенту в текстовому чаті з користувачем прямо обмежуються набором доступних інструментів, наданих MCP-сервером, тому для визначення технічної специфікації серверу необхідно було спочатку визначити можливі сценарії взаємодії користувача з текстовим асистентом.

Оскільки головною метою є оптимізація процесів пошуку інформації, асистент, що розробляється, повинен мати наступні можливості:

– відкривати портал DL NURE
– відповідати на запитання про поточні курси користувача
– відповідати на запитання про вміст окремих курсів

- повертати посилання на відповідні ресурси при запиті користувача
- надавати стислий вміст документів, наявних на порталі

3.3 Загальна архітектура інтелектуального асистента

Для створення мультизадачного інтелектуального асистенту було обрано Model Context Protocol архітектуру. Такий вибір обумовлений необхідністю забезпечення гнучкої та масштабованої взаємодії між мовною моделлю що використовує асистент та зовнішньою системою інтеграції. Важливою перевагою MCP архітектури є можливість виконання множинних послідовних викликів під час формування однієї відповіді, що може бути необхідно для роботи з навчальним порталом. Портал DL NURE має структуру навігації, в якій доступ до конкретної інформації потребує автентифікації, переходу через кілька сторінок та виконання різних дій у певній послідовності. MCP дозволяє асистенту планувати свої дії в реальному часі та виконувати багатоетапні сценарії, адаптуючись до результатів з кожним кроком. Крім того, MCP архітектура забезпечує чітке розділення відповідальності між клієнтською частиною (MCP клієнт) та серверною логікою взаємодії з порталом (MCP сервер). Такий підхід прибирає жорстку залежність між компонентами та дозволяє розробляти та тестувати функціонал роботи з порталом незалежно від AI моделі асистенту, а також дає можливість легко адаптувати асистента до змін в інтерфейсі порталу без необхідності перенавчання або модифікації мовної моделі, що використовується. Локальне розгортання MCP сервера також гарантує конфіденційність навчальних даних та забезпечує швидкий доступ до функціоналу порталу без залежності від зовнішніх сервісів.

3.4 Принцип роботи MCP-серверу

Для взаємодії з веб-інтерфейсом порталу DL NURE обрано бібліотеку Playwright через її можливості автоматизації та надійність при роботі з динамічним контентом. На відміну від традиційних рішень для отримання вмісту веб-сторінок, що базуються на простих HTTP запитах, Playwright забезпечує повноцінне виконання JavaScript коду, обробку асинхронних запитів та підтримку складних інтерактивних елементів, які є характерними для освітніх порталів. Бібліотека також надає стабільні механізми очікування завантаження елементів та можливість запуску браузеру в невидимому режимі для продуктивності або у видимому режимі для налагодження, що робить її ідеальним вибором для створення надійного MCP-серверу, здатного здійснювати навігацію на порталі.

Оскільки навчальна платформа DL NURE доступна лише студентам, що навчаються в університеті, асистент має використовувати обліковий запис студента для доступу до інформації, що знаходиться на порталі. Аутентифікація має відбуватися шляхом заповнення форм логіну та паролю.

Портал DL NURE також має перевірку CAPTCHA для перевірки того що користувач є справжньою людиною. Тому на етапі входу в обліковий запис обов’язковим є підтвердження з боку користувача.

Для того щоб асистент мав можливість виконувати задачі, що стосуються навігації порталом, логіка серверу має передбачати збір інформації з веб-інтерфейсу окремих сторінок та перетворення її в структурований вид, що буде легко сприйматися асистентом. Ця логіка дозволить асистенту оптимально відповідати на запитання, що стосуються навігації порталом та пошуку необхідних ресурсів.

Оскільки протокол MCP використовує JSON-RPC для відправки повідомлень, найлогічнішим рішенням буде приводити інформацію з порталу до табличного виду. Таким чином, наприклад, у разі запиту асистента до серверу стосовно переліку курсів, сервер спочатку відкриє графічний

інтерфейс з переліком курсів (рисунок 3.2), а потім систематизує цю інформацію та поверне її у табличному вигляді (таблиця 3.1).

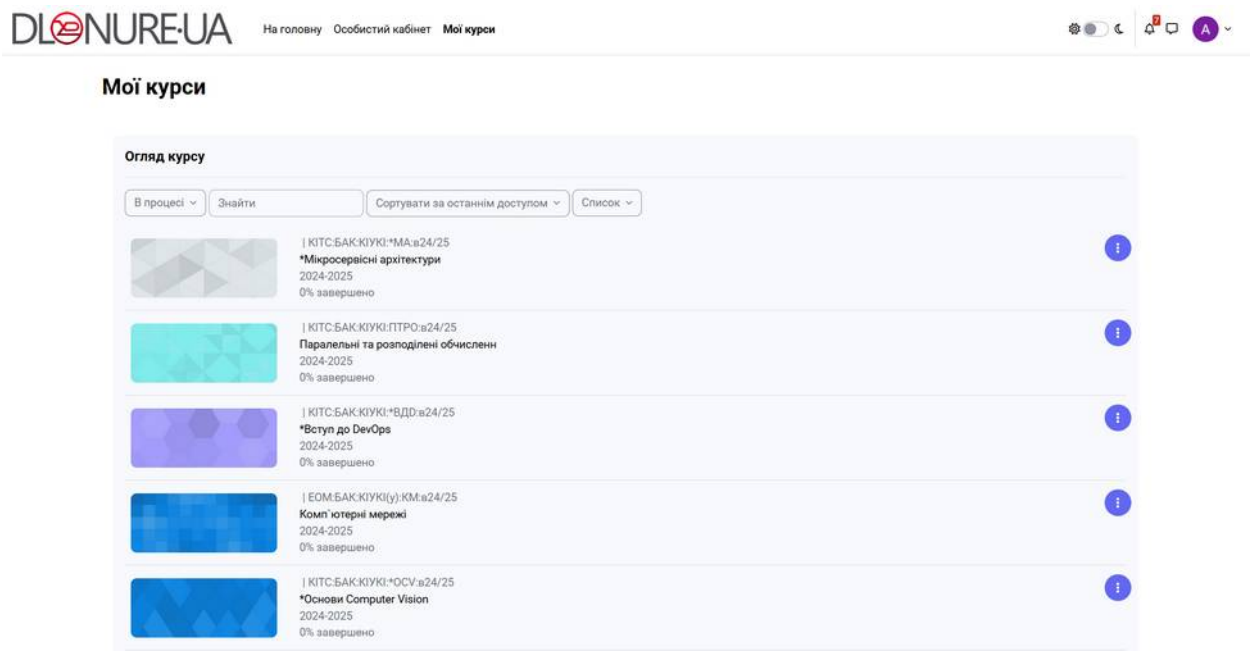

Рисунок 3.2 – сторінка з переліком курсів, доступних студенту

Таблиця 3.1 – Структура переліку курсів, що повертає сервер

*Мікросервісні архітектури
Паралельні та розподілені обчисленн
*Вступ до DevOps
Комп`ютерні мережі
*Основи Computer Vision

Таким саме чином буде надаватися інформація про вміст курсів. Спочатку буде відкрито сторінку курсу, яка містить перелік усіх доступних ресурсів (рисунок 3.3). Потім ці дані буде зібрано та надано мовній моделі в табличному вигляді (таблиця 3.2).

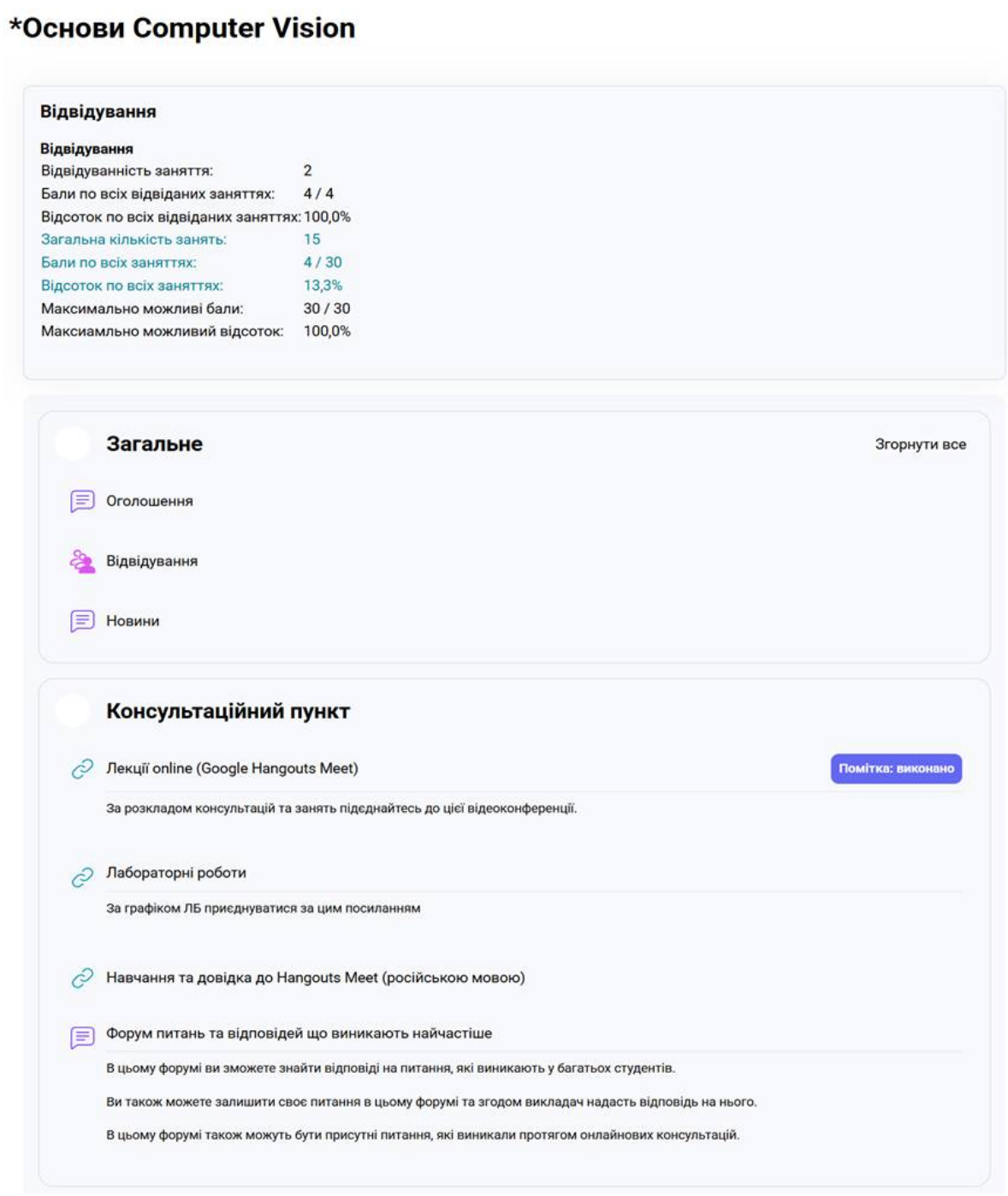

Рисунок 3.3 – сторінка з вмістом курсу

Таблиця 3.2 – Структура переліку ресурсів, доступних зі сторінки курсу

Назва	Тип ресурсу	Опис	Посилання
Оголошення	Announcement / Forum		https://dl.nure.ua/mod/forum/view.php?id=647664
Відвідування	Attendance		Посилання на ресурс
Новини	Announcement / Forum		Посилання на ресурс
Лекції online (Google Hangouts Meet)	Link	За розкладом консультацій…	Посилання на ресурс
Лабораторні роботи	Link	За графіком ЛБ…	Посилання на ресурс
Навчання та довідка до Hangouts Meet	Link		Посилання на ресурс
Форум питань та відповідей що виникають найчастіше	Announcement / Forum	В цьому форумі ви можете знайти…	Посилання на ресурс
Лекція №1	Document		Посилання на ресурс
Лекція №2	Document		Посилання на ресурс
Лекція №3	Document		Посилання на ресурс
Лекція №4	Document		Посилання на ресурс
Лекція №5	Document		Посилання на ресурс
Лекція №6	Document		Посилання на ресурс
Лекція №7	Document		Посилання на ресурс

Також, оскільки асистент має вміти надавати стислий вміст документів,

треба розробити алгоритм отримання стислого вмісту документів. Найпростішим рішенням може здатись просто повертати вміст документів безпосередньо асистенту, але такий підхід не є оптимальним. Деякі документи, наявні на порталі, можуть містити великі обсяги інформації, яка будучи наданою в сирому вигляді, просто витратить усі токени в контекстному вікні асистента, що призведе до зупинки генерації відповіді. Щоб уникнути такого сценарію, стиснення документів можна реалізувати з використанням OpenAI API. Компанія OpenAI надає публічно доступний API, що дозволяє звертатись до розроблених ними моделей для генерації відповідей на необхідні запити. Цей API є платним, але розцінки на використання базових текстових мовних моделей дуже низькі, що дозволяє дешево використовувати ШІ для обробки відносно великих текстових документів. Єдиним обмеженням є те, що для генерації відповіді в API передається текст, а документи на порталі DL NURE найчастіше надані в форматі PDF. Це значить що також треба розробити сервіс, що буде відповідальний за конвертацію PDF-документів у звичайний текстовий формат.

3.4 Java як мова програмування MCP-серверу

Для розробки серверної частини асистенту було обрано мову програмування Java та фреймворк Spring. Цей набір технологій надає велику кількість інструментів для розробки програмного забезпечення.

Java використовується для розробки різних видів програм, починаючи з веб-серверів та мобільних додатків та закінчуючи корпоративним програмним забезпеченням та системами обробки великих даних та відзначається наступними ключовими характеристиками:

– об’єктно-орієнтований підхід – Java підтримує об’єктно-орієнтоване програмування, що дозволяє створювати класи та об'єкти для організації коду та створення абстракцій

– портативність – Java була розроблена з метою мати можливість запускатися на будь-якій платформі без модифікації коду. Це досягається завдяки віртуальній машині Java (JVM), що перетворює байт-код у виконуваний код для конкретної платформи

– автоматичне керування пам’яттю – у Java присутня система автоматичного збору сміття для вивільнення пам'яті, яка запобігає багатьом типам помилок, пов'язаних із витоками пам'яті

– багатопоточність – Java підтримує багатопоточність, що дозволяє виконувати кілька задач одночасно в межах одного програмного процесу

Spring це фреймворк для мови програмування Java, розроблений з метою полегшити та пришвидшити процес розробки додатків. В основі Spring лежать такі принципи як інверсія контролю (Inversion of Control) та впровадження залежностей (Dependency Injection), що є ключовими для великої кількості функціоналу, що надається фреймворком (рис. 3.4).

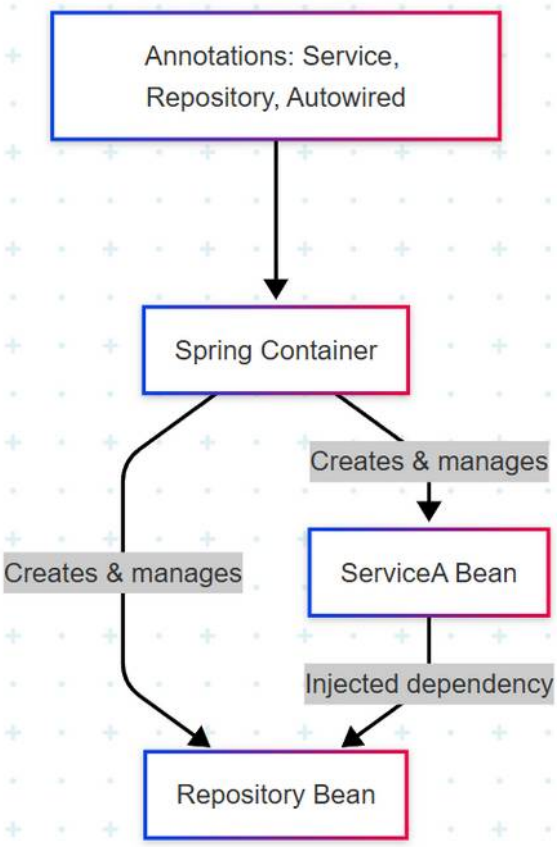

Рисунок 3.4 – Dependency Injection та Inversion of Control у Spring

Впровадження залежностей (DI) та інверсія контролю (IoC) – це основні принципи Spring Framework, що допомагають впоратися зі складністю програмних систем, створюючи вільний зв'язок та кращу модульність. Інверсія керування – це підхід, при якому рішення про створення об'єктів та управління залежностями передається від коду програми до контейнера або фреймворку. У випадку фреймворку Spring, контейнер IoC – це процес відповідальний за налаштування, конфігурацію та контроль над життєвим циклом компонентів програми.

Впровадження залежностей описує певний спосіб реалізації IoC, при якому залежності об'єкта надаються контейнером, а не об'єктом, що їх створює. За допомогою анотацій або класів конфігурації, розробники можуть створювати інструкції для контейнера Spring щодо автоматичного визначення та постачання необхідних компонентів. Створені таким чином програми є зручнішими у підтримці та тестуванні.

Окрім базового фреймворку, Spring надає велику кількість проектів, спеціалізованих для розробки систем різних видів. Серед найпопулярніших проектів варто відокремити:

– Spring Boot, що значно спрощує створення, розробку, та підтримку Spring-додатків

– Spring Data модуль що надає стандартизовану модель програмування для доступа до даних з різних джерел

– Spring Cloud, цілеспрямований на розробку хмарних сервісів, переважно побудованих на мікро-сервісній архітектурі

– Spring Security, що є стандартом для захисту додатків, написаних на Spring

– Spring AI фреймворк для розробки додатків, побудованих на ШІ

Для розробки цього проекту буде використано набір з Spring Boot, Spring MCP Server та Spring OpenAI. Така комбінація фреймворків повністю задовільнить усі необхідності, що виникнуть протягом розробки. Завдяки модульній архітектурі Spring проектів, за необхідності можна легко

підключити додаткові інструменти.

3.5 Claude Desktop як MCP-клієнт для інтелектуального асистента

В якості MCP-клієнту для інтелектуального асистенту було обрано додаток Claude Desktop. Його було обрано через ряд переваг цього додатку над альтернативами. Одним з найголовніших критеріїв є підтримка Claude Desktop MCP протоколу за замовчуванням, що суттєво спрощує процес інтеграції клієнту з розробленим сервером. Також, мовні моделі Anthropic вважаються дуже якісними за результатами багатьох тестів. Це значить, що асистент, який використовує ці мовні моделі для генерації відповідей, буде давати якісні відповіді та оптимально аналізувати інформацію. Крім того, вбудований MCP клієнт здатен виконувати множинні послідовні виклики до MCP-серверів протягом однієї відповіді. Такий підхід дозволить реалізувати складні багатоетапні сценарії роботи з порталом DL NURE. Окрім перелічених функціональних переваг, також корисною є простота підключення локального MCP серверу – для цього достатньо додати в конфігураційний файл Claude Desktop параметри запуску серверу та зберегти зміни.

4 ПРАКТИЧНА РЕАЛІЗАЦІЯ ПРОЕКТУ

4.1 Розробка Model Context Protocol серверу

Для надання асистенту можливості взаємодіяти з навчальною платформою DL NURE, необхідно створити програму, яка буде виступати в якості MCP-серверу. Головною метою цієї програми є надання інтерфейсу, що дозволить асистенту взаємодіяти з навчальною платформою DL NURE.

Програма MCP-серверу розробляється в інтегрованому середовищі розробки (IDE) Intellij IDEA.

Першим кроком є створення нового проекту за допомогою Spring Initializr. Цей інструмент спрощує процес створення проектів, що будуть використовувати фреймворк Spring та деякі Spring-модулі. Завдяки інтеграції з Intellij IDEA, можна додати необхідні залежності одразу під час створення проекту (рис 4.1).

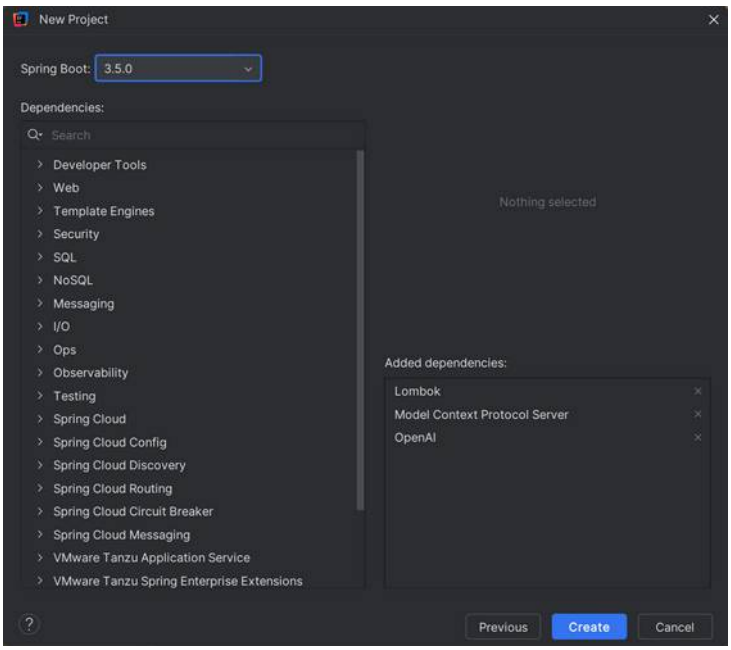

Рисунок 4.1 – Збір Spring-залежностей для нового проекту

Програму було розроблено з використанням об’єктно-орієнтованого підходу, завдяки чому різні сервіси та моделі даних були представлені у вигляді зручних абстракцій.

Основна компонентом програми є клас DLService, що містить в собі логіку ініціалізації та завершення програми, а також перелік усіх доступних інструментів, що будуть використані MCP-клієнтом. Оголошення цього класу наведено у лістингу 4.1.

Лістинг 4.1 – оголошення класу DLService

```
@Service
@RequiredArgsConstructor
public class DLService {
    private final DLCredentials credentials;

    private AITool aiTool;

    @Autowired
    public void setAiTool(@Lazy AITool aITool) {
        this.aiTool = aITool;
    }

    private Playwright playwright;

    private Browser browser;

    private Page page;

    private final CourseItemTypeMatcher courseItemTypeMatcher;

    private final RestTemplate restTemplate;

    private final PDFProcessor pdfProcessor;

    @PostConstruct
    public void init() {
        playwright = Playwright.create();
        browser = playwright.chromium().launch(
                new BrowserType.LaunchOptions()
                        .setHeadless(false)
                        .setSlowMo(2000)
        );
    }

    @PreDestroy
    public void destroy() {
        browser.close();
        playwright.close();
    }
```

В цьому фрагменті коду відбувається конфігурація сервісу DLService. Завдяки анотації @Service, цей клас маркується як сервіс в контексті Spring, що дозволить фреймворку автоматично створити екземпляр цього класу. @RequiredArgsConstructor генерує конструктор з усіма полями класу що позначені ключовим словом final. Цей конструктор буде використаний Spring для автоматичного впровадження залежностей. Анотації @PostConstruct та @PreDestroy використовуються для того щоб фреймворк автоматично викликав ці методи під час запуску та зупинки програми відповідно. У методі init за допомогою бібліотеки Playwright запускається браузер, який буде використано для взаємодії з web-інтерфейсом DL NURE. При створенні екземпляру браузеру параметр setHeadless(false) вмикає відображення графічного інтерфейсу браузеру, а setSlowMo(2000) додає затримку довжиною в дві секунди між кожною дією. Завдяки цьому користувач може відстежувати дії, що відбуваються під час роботи з асистентом.

Далі, в класі DLService описується перелік інструментів, що надані MCP-сервером. Саме ці інструменти використовуються MCP-клієнтом для виконання усіх задач, пов’язаних з DL NURE.

Лістинг 4.2 – Інструмент запуску порталу.

```
    @Tool(description = """
                Launches a browser and opens DL NURE portal.
                 This action is required before conducting any other actions
with DL portal.
                """)
        public String openDL() {
            page = browser.newPage();
            page.navigate(DLPages.LOGIN);

            page.locator("#username").waitFor();

            //enter the login and password
            page.locator("#username").fill(credentials.getLogin());
            page.locator("#password").fill(credentials.getPassword());

            //click the Log In button
            page.click("#loginbtn");

            return "Browser launched successfully and navigated to DL";
        }
```

На лістингу 4.2 наведено перший і найголовніший інструмент, що дозволяє відкрити портал DL NURE та виконати аутентифікацію. Цей і наступні методи позначені анотацією @Tool, яка є частиною бібліотеки Spring MCP Server. Завдяки цій анотації, фреймворк розуміє що цей метод має буду експортований в якості інструмента, що надається MCP-сервером. Параметр description дозволяє встановити більш детальне пояснення призначення методу, для того щоб асистенту було легше зрозуміти коли саме викликати цей функціонал. Коли викликається цей метод, у браузері створюється нова сторінка, що переходить за адресою форми логіну DL NURE. Після цього, у відповідні поля вводяться логін та пароль користувача та симулюється натискання кнопки логіну. Значення цих полів визначається локальною конфігурацією сервера, тобто логін і пароль знаходяться в локальному середовищі користувача і не рискують бути втрачені. У разі якщо сайт вимагає підтвердження того, що користувач є живою людиною (CAPTCHA), користувач має власноруч пройти перевірку.

Лістинг 4.3 – Інструмент отримання доступних курсів

```
  @Tool(description = """
        Returns a list of the courses the user is enrolled to.
        """)
public List<String> getEnrolledCourses() {
    page.navigate(DLPages.COURSES);

    Locator courseTitles = page.locator(".aalink.coursename");

    List<String> titles = new ArrayList<>();

    for (Locator courseLocator : courseTitles.all()) {
        String fullText = courseLocator.innerText();
        String title = fullText.split("\n")[1].trim();
        titles.add(title);
    }
    return titles;
}
```

Лістинг 4.3 містить метод, що дозволяє отримати перелік курсів, доступних користувачу. Програма відкриває сторінку з курсами та сканує її вміст. Алгоритм збору інформації зі сторінок в цьому та інших методах працює з використанням локаторів (спеціальних виразів, що описують шлях до HTML-компонентів у DOM-моделі сторінки). Завдяки спільному шаблону HTML-компонентів однакових елементів користувацького інтерфейсу, можна створити єдиний локатор, що буде знаходити усі елементи одного типу (наприклад усі імена курсів на сторінці).

Лістинг 4.4 – Інструмент отримання вмісту курсу

```
      @Tool(description = """
                      Returns a list of all items present on a specific
course page.
                  """)
              public List<CourseItem> getCourseItems(String courseTitle)
throws IOException, InterruptedException {
                openCoursePage(courseTitle);

                    Locator courseItemsLocator = page.locator(".activity-
item.focus-control");

                List<CourseItem> courseItems = new ArrayList<>();

                for (Locator courseItemLocator : courseItemsLocator.all())
{
                    CourseItem courseItem = new CourseItem();

                                                Locator titleLocator =
courseItemLocator.locator("span.instancename");
                    if (!titleLocator.all().isEmpty()) {
                        String title = (String) titleLocator
                                    .evaluate("node => node.childNodes[0] ?
node.childNodes[0].textContent.trim() : ''");
                        courseItem.setTitle(title);

                                                  Locator typeLocator =
courseItemLocator.locator(".activityicon");
                        if (!typeLocator.all().isEmpty()) {
                            String typeImgSrc = (String) typeLocator
                                                          .evaluate("node =>
node.getAttribute('src')");
                                                             String type =
courseItemTypeMatcher.getTypeFromIconUrl(typeImgSrc);
                            courseItem.setType(type);
                        }

                                               Locator descriptionLocator =
courseItemLocator.locator(".activity-description, .activity-dates");
                        if (!descriptionLocator.all().isEmpty()) {
                              // Handle multiple elements by concatenating
```

```
their text
                                    StringBuilder descriptionBuilder = new
StringBuilder();
                                                for (Locator element :
descriptionLocator.all()) {
                                if (!descriptionBuilder.isEmpty()) {
                                    descriptionBuilder.append(" ");
                                }
                                descriptionBuilder.append(element.innerTex
t());
                            }
                            courseItem.setDescription(descriptionBuilder.t
oString());
                        }

                                                    Locator urlLocator =
courseItemLocator.locator(".aalink.stretched-link");
                        if (!urlLocator.all().isEmpty()) {
                            String url = (String) urlLocator
                                                            .evaluate("node =>
node.getAttribute('href')");
                            courseItem.setUrl(url);
                        }

                        courseItems.add(courseItem);
                    }
                }

                return courseItems;
            }
```

На лістингу 4.4 наведено метод, що дозволяє отримати повний міст будь якого курсу, доступного користувачу. За схожим принципом, програма відкриває сторінку окремого курсу та сканує її вміст. Через велике різноманіття типів ресурсів, наявних у курсах, логіка складніша ніж в попередньому методі. При пошуку доступних ресурсів, програма шукає не лише їх назви, а ще й визначає їх тип та посилання на них. Завдяки цьому асистент може надавати користувачу посилання безпосередньо на окремі ресурси (лабораторні роботи, конференції, тести, тощо).

Лістинг 4.5 – Інструмент для отримання стислого вмісту документів.

```
    @Tool(description = """
                  Downloads a PDF file from specified link, generates summary
and returns it as a plain text.
                  """)
          public String getPdfSummary(String pdfLink) throws IOException {
              // Get cookies from Playwright browser context
              List<Cookie> cookies = browser.contexts().get(0).cookies();
```

```
            // Create HttpHeaders and add cookies
            HttpHeaders headers = new HttpHeaders();

            // Convert Playwright cookies to a cookie header string
            StringBuilder cookieHeader = new StringBuilder();
            for (Cookie cookie : cookies) {
                if (!cookieHeader.isEmpty()) {
                    cookieHeader.append("; ");
                }
                cookieHeader.append(cookie.name).append("=").append(cookie.v
alue);
            }

            // Add the cookie header to the request
            headers.add("Cookie", cookieHeader.toString());

            // Create HttpEntity with headers
            HttpEntity<String> entity = new HttpEntity<>(headers);

            // Make the request with the entity containing cookies
            ResponseEntity<Resource> response = restTemplate.exchange(
                    pdfLink,
                    HttpMethod.GET,
                    entity,
                    Resource.class
            );

            // Process the PDF as before
                                                String     plainText     =
pdfProcessor.getPdfAsPlainText(response.getBody().getInputStream());

            return aiTool.getDocumentSummary(plainText);
        }
```

Лістинг 4.5 містить останній інструмент, мета якого – надати можливість отримувати стислий вміст документів, наявних на порталі. Процес стиснення документів відбувається у декілька кроків. По-перше, програма завантажує документ у пам’ять. Надалі цей документ конвертується у звичайний текст (для подальшої обробки) та оптимізується. Оптимізація відбувається завдяки видаленню зайвих текстових елементів з документу, а саме:

- повторюваних переносів та пробілів
- номерів сторінок
- заголовків та нижніх колонтитулів (визначаються як повторюваний текст у верхніх та нижніх частинах сторінок)

Для генерації стислого вмісту використовується OpenAI API. Як можна побачити на лістингу 4.6, завдяки фреймворку Spring AI, код, необхідний для

реалізації цього функціоналу складається лише з кількох рядків.

Лістинг 4.6 – сервіс для генерації стислого вмісту документів

```
@Service
@RequiredArgsConstructor
public class AITool {

    private final OpenAiChatModel chatModel;

    public String getDocumentSummary(String documentContent) {
        // Create a prompt for document summarization
        String promptText = """
                You are a helpful assistant that summarizes documents.
                     Provide a concise summary of the document content
provided by the user. " +
                 The summary should contain structure of the document. "
+
                 The summary should contain key points from the document
content. " +
                  SUMMARY SHOULD BE PROVIDED IN ORIGINAL LANGUAGE OF THE
DOCUMENT. " +
                Please summarize the following document:
                """ + documentContent;

        return chatModel.call(promptText);
    }
}
```

В цьому сервісі описаний запит до текстової моделі OpenAI, який містить інструкції стосовно того як генерувати стислий вміст документу.

4.2 Налаштування Model Context Protocol клієнту

Конфігурація, необхідна для з’єднання клієнта з сервером відбувається завдяки файлу claude_desktop_config.json, що знаходиться в директорії Claude Desktop (лістинг 4.7).

Лістинг 4.7 – конфігураційний файл клієнта

```
{
  "mcpServers": {
    "dl-mcp": {
      "command": "java",
      "args": [
        "-Ddl.login=userlogin",
        "-Ddl.password=userpassword",
        "-DOPENAI_API_KEY=apikey",
```

```
            "-jar",
                        "D:/Projects/dl-assistant/target/dl-assistant-0.0.1-
SNAPSHOT.jar"
          ]
        }
      }
    }
```

В конфігураційному файлі налаштовується команда та параметри, необхідні для запуску локального MCP-серверу. В даному випадку локальний сервер запускається з jar файлу, а в параметрах передаються змінні, що містять в собі дані для входу на портал DL NURE та секретний ключ для використання OpenAI API. Після збереження наведених налаштувань, клієнт готовий до роботи з локальним сервером.

4.3 Тестування додатку

Взаємодія користувача з асистентом відбувається через текстовий чат-інтерфейс додатку Claude Desktop (рисунок 3.4).

За замовчуванням Claude Desktop дозволяє спілкуватися з мовними моделями Claude, розробленими компанією Anthropic. Для того щоб отримати можливість задавати питання стосовно порталу DL NURE, необхідно розташувати jar файл MCP-серверу на локальному пристрої та налаштувати MCP-клієнт для підключення до цього серверу через відповідний конфігураційний файл, що дозволить Claude Desktop взаємодіяти з функціоналом порталу через локально запущений сервер.

Якщо всі описані дії були виконані правильно, у графічному інтерфейсі додатку Claude Desktop з’явиться перелік інструментів, які експортуються з запущеного MCP-серверу (рис 4.1).

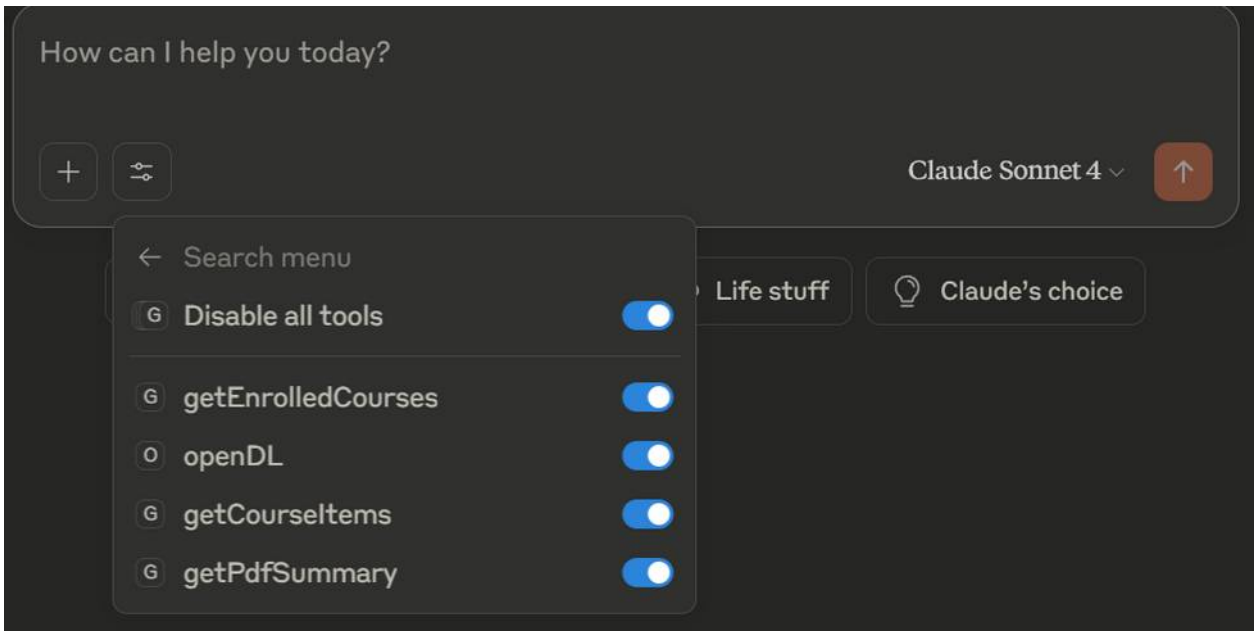

Рисунок 4.1 – Перелік експортованих інструментів

За бажанням, користувач може вмикати або вимикати деякі окремі інструменти, щоб обмежити спектр дій, що може виконувати асистент.

Після підключення серверу, користувач може у довільному форматі задавати питання, що стосуються навчального процесу. Під час генерації відповіді, асистент проведе аналіз запиту користувача та визначить, чи потрібна додаткова інформація з освітнього порталу для формування повної та релевантної відповіді. Якщо асистент зрозуміє, що необхідний контекст може бути отриманий з MCP-серверу, він автоматично почне процес збору даних. В такому випадку він спробує надіслати запит до локального серверу через стандартний протокол комунікації STDIO, використовуючи структуровані повідомлення згідно з специфікацією Model Context Protocol.

Процес взаємодії відбувається синхронно, тобто асистент чекає відповіді від серверу, яка може містити структуровані дані у форматі JSON або текстову інформацію з порталу DL NURE. Після отримання необхідних даних, асистент інтегрує цю інформацію у свій контекст та генерує комплексну відповідь, що враховує як загальні знання мовної моделі, так і актуальну інформацію, отриману безпосередньо з освітнього порталу (рис. 4.2).

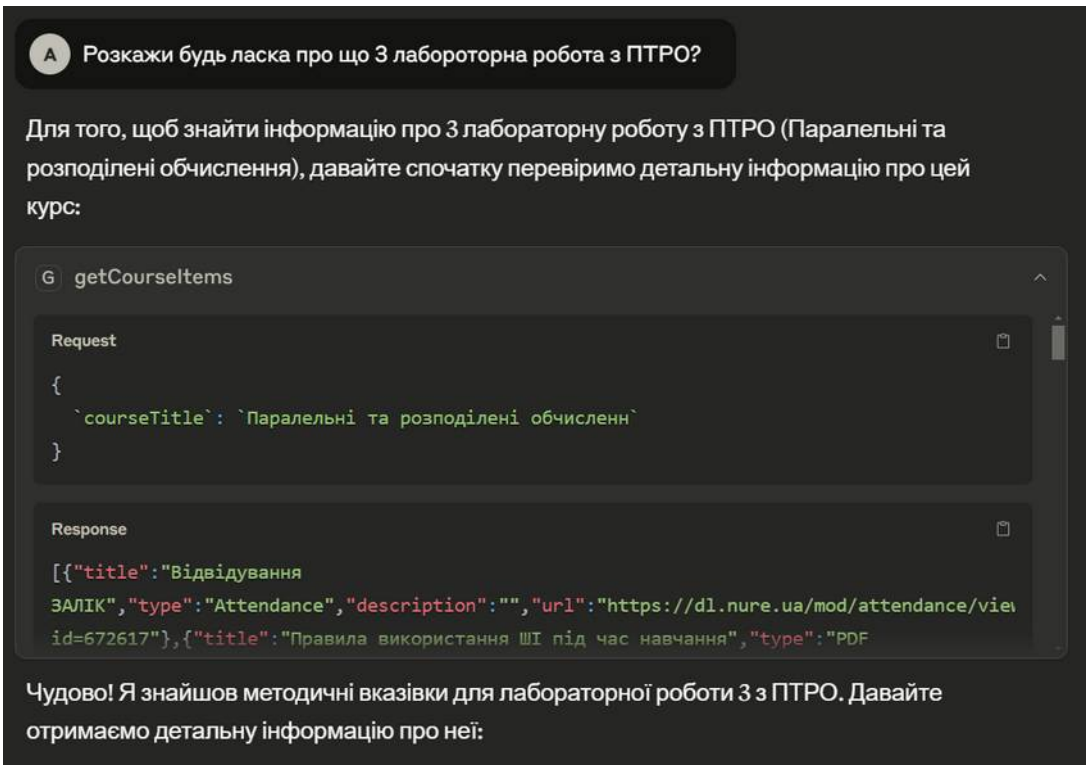

Рисунок 4.2 – Приклад використання MCP-серверу асистентом

MCP-клієнт Claude Desktop може виконувати множинні виклики до MCP-серверів протягом формування однієї відповіді, що дозволяє створювати складні багатоетапні робочі процеси. Наприклад, асистент може спершу звернутися до MCP сервера для запуску порталу DL NURE, потім отримати список курсів, що доступні користувачу, перейти на сторінку окремого курсу, і нарешті знайти необхідний ресурс, про який поставив запитання користувач (рис. 4.3).

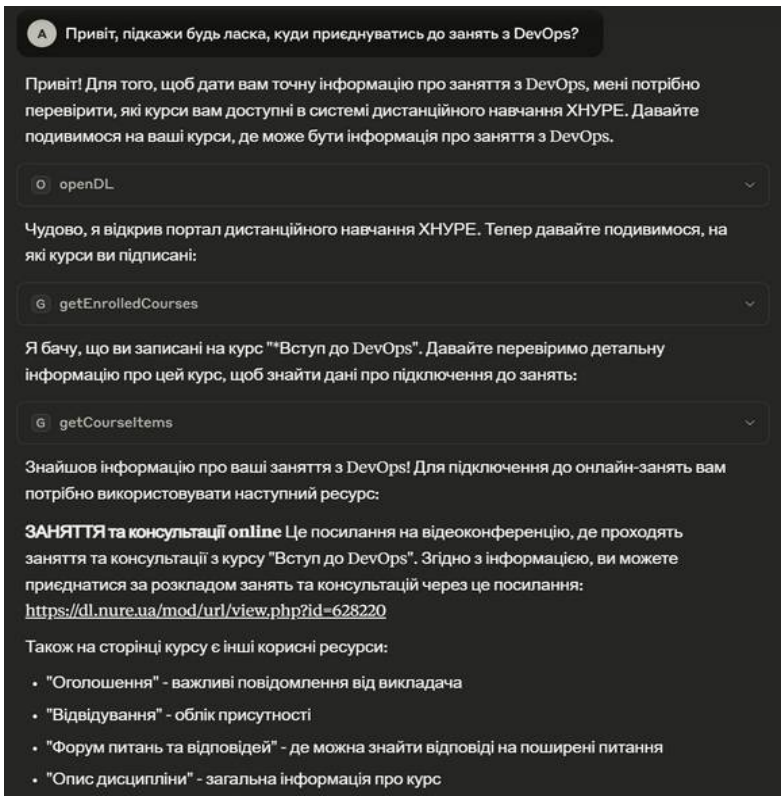

Рисунок 4.3 – Множинні виклики MCP-серверу в межах однієї відповіді

Під час взаємодії з асистентом, усі виконані ним дії будуть помітні у вікні браузеру, що запускається при першому виклику MCP-серверу (рис. 4.4).

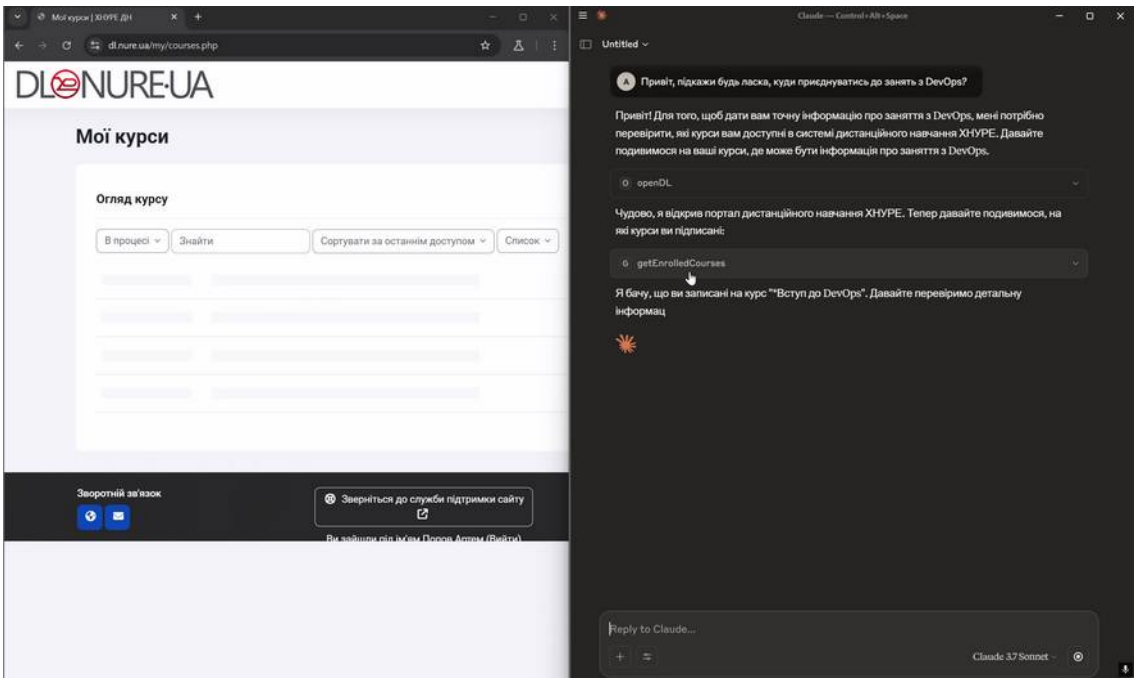

Рисунок 4.4 – Вікно браузеру відображає поточні дії асистента

ВИСНОВКИ

Для досягнення поставленої мети, а саме для створення інтелектуального мультизадачного персонального AI-асистенту на основі Model Context Protocol, було проведено аналіз галузі інтелектуальних асистентів та розроблено систему, що оптимізує процеси навігації та пошуку інформації на навчальному порталі DL NURE.

Під час роботи, було проведено аналіз предметної області та виявлено, що інтелектуальний асистент на основі великої мовної моделі є ідеальним вибором для такої задачі. Після дослідження існуючих технологій та інструментів для створення інтелектуальних асистентів, було обрано мову програмування Java та фреймворк Spring для розробки серверної частини асистента. В якості інтерфейсу взаємодії з користувачем було обрано додаток Claude Desktop.

Практична реалізація проекту полягала в створенні Model Context Protocol серверу, який надав персональному асистенту інтерфейс для інтеграції з платформою DL NURE. Таким чином, асистент отримав можливість в реальному часі отримувати необхідну інформацію з навчального порталу.

Створений асистент може стати у нагоді усім студентам, що навчаються в Харківському Національному Університеті Радіоелектроніки.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Amirhosein Toosi. A brief history of AI: how to prevent another winter. URL: https://arxiv.org/pdf/2109.01517 (дата звернення: 25.05.2025).

2. М. В. Мокрий. Функції активації в архітектурі нейронних мереж. URL: https://ekmair.ukma.edu.ua/server/api/core/bitstreams/48aa0f3f-910b-407a-aca1-da9fa274a4cc/content (дата звернення: 25.05.2025).

3. Attention Is All You Need / Ashish Vaswani et al. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf (дата звернення: 26.05.2025).

4. А. С. Карпцов. Інформаційна технологія аналізу силабусів навчальних дисциплін із застосуванням великих мовних моделей. URL: https://essuir.sumdu.edu.ua/bitstream-download/123456789/94536/1/Karptsov_mag_rob.pdf (дата звернення: 26.05.2025).

5. Text generation and prompting. OpenAI. URL: https://platform.openai.com/docs/guides/text (дата звернення: 26.05.2025).

6. How does LangChain integrate with LLMs (Large Language Models)?. Milvus | High-Performance Vector Database Built for Scale. URL: https://milvus.io/ai-quick-reference/how-does-langchain-integrate-with-llms-large-language-models (дата звернення: 07.06.2025).

7. Introducing the Model Context Protocol. Home \ Anthropic. URL: https://www.anthropic.com/news/model-context-protocol (дата звернення: 25.05.2025).

8. Introduction - Model Context Protocol. URL: https://modelcontextprotocol.io/introduction (дата звернення: 25.06.2025).

9. JSON-RPC 2.0 Specification. JSON-RPC. URL: https://www.jsonrpc.org/specification (дата звернення: 25.05.2025).