

2026 p.

Харківський національний університет радіоелектроніки

Факультет Навчально-науковий центр заочної форми навчання

Кафедра Штучного інтелекту

Рівень вищої освіти перший (бакалаврський)

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

Тип програми освітньо-професійна

Освітня програма Штучний інтелект
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____

(підпис)

«_____» _____ 20 ____ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Стрельциній Катерині Андріївні
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка AI-агента для автоматичного виконання завдань у веб-середовищі

затверджена наказом університету від 18 травня 2026 р. № 76Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 9 липня 2026 р.

3. Вихідні дані до роботи Науково-технічні публікації у сфері штучного інтелекту, AI-агентних систем та автоматизації веб-середовища, дані сучасних Інтернет-джерел і документації програмних бібліотек, документація OpenAI API, Playwright documentation, LangChain documentation, Python documentation, матеріали сучасних досліджень у сфері великих мовних моделей та browser automation, asynchronous frameworks

4. Перелік питань, що потрібно опрацювати в роботі _____

1) Аналіз предметної галузі _____

2) Теоретичні дослідження _____

3) Програмна реалізація _____

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / термін виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	18.05.2026	виконано
2	Аналіз предметної галузі	20.05.2026	виконано
3	Аналіз сучасних підходів до автоматизації веб-завдань	25.05.2026	виконано
4	Дослідження існуючих AI-агентних архітектур	27.05.2026	виконано
5	Проектування архітектури AI-агента	30.05.2026	виконано
6	Вибір технологій та програмних засобів реалізації	01.06.2026	виконано
7	Реалізація planner/executor архітектури	04.06.2026	виконано
8	Інтеграція AI-моделі та веб-інструментів	10.06.2026	виконано
9	Тестування роботи AI-агента у веб-середовищі	12.06.2026	виконано
10	Аналіз результатів експериментального тестування	14.06.2026	виконано
11	Написання пояснювальної записки	17.06.2026	виконано
12	Нормоконтроль	24.06.2026	виконано
13	Підготовка презентації та доповіді	28.06.2026	виконано
14	Захист перед екзаменаційною комісією	14.07.2026	

Дата видачі завдання 18 травня 20 26 р.

Здобувач 
(підпис)

Керівник роботи _____
(підпис)

ас. Дмитро МАЛЄЄВ
(посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 76 с., 19 рис., 4 табл., 1 дод., 35 джерел.

АВТОНОМНІ СИСТЕМИ, ВЕБ-АВТОМАТИЗАЦІЯ, ШТУЧНИЙ ІНТЕЛЕКТ, AI-АГЕНТ, AUTOMATION, BROWSER AUTOMATION, REASONING, SOFTWARE ARCHITECTURE, TOOL-AUGMENTED AI, WEB ENVIRONMENT.

Об'єкт дослідження – процеси автономного виконання задач у веб-середовищі із використанням інтелектуальних програмних систем.

Предмет дослідження – методи та засоби побудови AI-агентів для автоматизації веб-взаємодії, обробки природної мови та реалізації reasoning-механізмів у веб-середовищі.

Мета роботи – розробка та дослідження AI-агента для автоматичного виконання веб-завдань, який забезпечує аналіз запитів користувача, планування послідовності дій, взаємодію із веб-ресурсами та формування обґрунтованих результатів на основі reasoning-процесу.

Методи дослідження – аналіз сучасних AI-агентних архітектур, методи обробки природної мови, підходи до побудови tool-augmented AI systems, browser automation-технології, методи планування та декомпозиції задач, засоби програмної реалізації веб-автоматизації, методи аналізу та оцінювання результатів роботи інтелектуальних систем.

У роботі досліджено сучасні підходи до побудови AI-агентів та проаналізовано особливості їх використання для автоматизації задач у веб-середовищі. Проведено аналіз існуючих browser automation-технологій, сучасних AI-agent frameworks та підходів до інтеграції великих мовних моделей із зовнішніми інструментами взаємодії із веб-ресурсами.

ABSTRACT

Bachelor's thesis contains: 76 pp., 19 fig., 4 tabl., 1 ann., 35 references.

AI-AGENT, ARTIFICIAL INTELLIGENCE, AUTOMATION, AUTONOMOUS SYSTEMS, BROWSER AUTOMATION, REASONING, SOFTWARE ARCHITECTURE, TOOL-AUGMENTED AI, WEB AUTOMATION, WEB ENVIRONMENT.

The object of research is the processes of autonomous task execution in the web environment using intelligent software systems.

The subject of research is methods and tools for building AI agents for automating web interaction, natural language processing and implementing reasoning mechanisms in the web environment.

The purpose of the work is to develop and research an AI agent for automatic execution of web tasks, which provides analysis of user requests, planning of a sequence of actions, interaction with web resources and the formation of substantiated results based on the reasoning process.

Research methods – analysis of modern AI-agent architectures, natural language processing methods, approaches to building tool-augmented AI systems, browser automation technologies, methods of planning and decomposition of tasks, software implementation tools for web automation, methods of analysis and evaluation of the results of the work of intelligent systems.

The paper investigates modern approaches to building AI-agents and analyzes the features of their use for automating tasks in the web environment. An analysis of existing browser automation technologies, modern AI-agent frameworks and approaches to integrating large language models with external tools for interacting with web resources is conducted.

ЗМІСТ

Перелік умовних позначень, символів, одиниць, скорочень і термінів	8
Вступ	9
1 Аналіз предметної галузі	12
1.1 Сутність та роль AI-агентів у сучасних інформаційних системах	12
1.1.1 Основні підходи до побудови AI-агентів.....	13
1.1.2 Особливості функціонування агентів у веб-середовищі	15
1.2 Аналіз існуючих рішень та технологій автоматизації	17
1.2.1 Сучасні інструменти автоматизації веб-задач.....	18
1.2.2 Використання штучного інтелекту для автоматичного виконання задач	20
1.2.3 Переваги та обмеження існуючих підходів	22
1.3 Актуальність створення AI-агента для автоматизації задач у веб-середовищі.....	24
2 Теоретичні дослідження	27
2.1 Аналіз сучасних AI-агентів для автоматизації веб-задач	27
2.1.1 Загальна характеристика сучасних AI-агентів	27
2.1.2 Порівняльний аналіз сучасних AI-агентів	29
2.1.3 Основні проблеми та обмеження сучасних AI-агентів.....	32
2.2 Аналіз інструментів та технологій для реалізації AI-агента	36
2.2.1 Фреймворки для побудови AI-агентів.....	36
2.2.2 Інструменти браузерної автоматизації	38
2.2.3 Аналіз мовних моделей для AI-агентів	40
2.3 Порівняльний аналіз підходів до побудови AI-агентів	42
2.4 Обґрунтування архітектури власного AI-агента	44
3 Програмна реалізація	47
3.1 Загальна архітектура програмної системи	47
3.2 Реалізація інтелектуального агентного циклу	50
3.3 Механізми забезпечення інтелектуальності системи	54

3.3.1 Використання великих мовних моделей	54
3.3.2 Планування та декомпозиція задач	56
3.3.3 Memory Layer та контекстна пам'ять	57
3.3.4 Самооцінка та retry-механізми	59
3.4 Реалізація веб-взаємодії та автоматизації браузера	60
3.5 Реалізація користувацького інтерфейсу системи	62
3.5.1 Візуалізація процесу мислення агента	63
3.5.2 Підтримка різних тем інтерфейсу	64
3.6 Аналіз результатів роботи системи	66
Висновки.....	69
Перелік джерел посилання	72
Додаток А Відомість кваліфікаційної роботи.....	76

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

AI – Artificial Intelligence – штучний інтелект;

API – Application Programming Interface – програмний інтерфейс застосунку;

DOM – Document Object Model – об’єктна модель документа;

GUI – Graphical User Interface – графічний користувацький інтерфейс;

HTML – HyperText Markup Language – мова розмітки гіпертексту;

HTTP – HyperText Transfer Protocol – протокол передачі гіпертексту;

JSON – JavaScript Object Notation – текстовий формат обміну даними;

LLM – Large Language Model – велика мовна модель;

NLP – Natural Language Processing – обробка природної мови;

RAG – Retrieval-Augmented Generation – генерація з доповненням пошуком;

REST – Representational State Transfer – архітектурний стиль взаємодії веб-систем;

UI – User Interface – користувацький інтерфейс;

URL – Uniform Resource Locator – уніфікований локатор ресурсу;

UX – User Experience – користувацький досвід;

WWW – World Wide Web – всесвітня павутина.

ВСТУП

У сучасних умовах стрімкого розвитку інформаційних технологій відбувається суттєва трансформація цифрового середовища та підходів до організації діяльності людини в мережі Інтернет. Веб-технології вже давно перестали бути лише засобом отримання інформації. Сьогодні веб-середовище є складною інтерактивною екосистемою, у межах якої здійснюються процеси комунікації, дистанційного навчання, електронної комерції, управління бізнес-процесами, аналізу даних, фінансових операцій та взаємодії між різними інформаційними системами. У зв'язку з цим дедалі більшої актуальності набуває питання автоматизації діяльності користувача у веб-середовищі, оскільки значна частина сучасних задач пов'язана з виконанням рутинних, повторюваних та часовитратних операцій.

Зростання обсягів інформації, швидкості оновлення цифрового контенту та складності веб-ресурсів формує потребу у створенні інтелектуальних програмних систем, здатних автономно виконувати дії, аналізувати отримані результати та адаптуватися до змін середовища. Традиційні підходи до автоматизації, засновані на використанні скриптів або жорстко визначених сценаріїв, уже не забезпечують необхідної гнучкості та ефективності в умовах сучасного веб-середовища. Більшість таких систем орієнтована на виконання наперед визначеної послідовності дій і не здатна враховувати зміни структури веб-інтерфейсів, неоднозначність запитів користувача або необхідність контекстного аналізу інформації.

У таких умовах особливого значення набувають AI-агенти – інтелектуальні програмні системи, здатні взаємодіяти із цифровим середовищем, аналізувати поставлені задачі, приймати рішення та виконувати необхідні дії без постійного контролю з боку користувача. На відміну від традиційних систем автоматизації, сучасні AI-агенти поєднують механізми обробки природної мови, reasoning-процеси, контекстну пам'ять,

модулі планування та browser automation-технології. Це дозволяє їм не лише відтворювати заздалегідь задані сценарії, а й формувати адаптивну стратегію поведінки відповідно до поточної ситуації та проміжних результатів виконання задачі.

Особливу роль у розвитку сучасних AI-агентів відіграє поява великих мовних моделей (Large Language Models, LLM), які суттєво розширили можливості систем штучного інтелекту у сфері аналізу природної мови та інтерпретації текстової інформації. Інтеграція LLM у структуру AI-агентів дозволяє системам розуміти складні природномовні запити, формувати reasoning-ланцюги, здійснювати декомпозицію задач та приймати проміжні рішення у процесі взаємодії із веб-середовищем. Завдяки цьому AI-агенти можуть виконувати багатокрокові сценарії, які раніше потребували безпосередньої участі людини.

Веб-середовище як об'єкт взаємодії AI-агента має ряд специфічних характеристик. Воно включає динамічні веб-сторінки, інтерактивні елементи інтерфейсу, API-сервіси, форми введення даних, мультимедійний контент та складні механізми взаємодії з користувачем. Крім того, сучасні веб-ресурси постійно змінюються, що ускладнює використання класичних підходів до автоматизації. У таких умовах ефективне виконання веб-завдань потребує не лише технічного виконання браузерних дій, але й здатності до контекстного аналізу, оцінювання результатів, адаптивного планування та reasoning-oriented взаємодії із середовищем.

Сьогодні існує значна кількість технологій та інструментів для автоматизації веб-процесів, серед яких системи роботизованої автоматизації процесів (RPA), Selenium, Puppeteer, Playwright та інші browser automation frameworks. Однак більшість таких рішень орієнтована на виконання жорстко заданих сценаріїв і не забезпечує достатнього рівня автономності та інтелектуальної адаптації. У зв'язку з цим активно розвивається новий напрямок intelligent agent systems, який поєднує browser

automation із можливостями штучного інтелекту, великих мовних моделей та механізмів reasoning.

Додаткової актуальності темі дослідження надає стрімкий розвиток концепції tool-augmented AI systems, у межах якої великі мовні моделі використовуються не лише для генерації тексту, а й для координації роботи зовнішніх інструментів. У такому підході AI-агент здатний автономно взаємодіяти із браузером, виконувати веб-пошук, аналізувати зовнішні джерела інформації, оцінювати достовірність отриманих результатів та формувати обґрунтовані відповіді на основі source grounding-механізмів. Подібні системи розглядаються як один із перспективних напрямів розвитку сучасного штучного інтелекту та автономних цифрових помічників.

Актуальність теми даної кваліфікаційної роботи зумовлена необхідністю створення AI-агента, здатного ефективно функціонувати у динамічному веб-середовищі та виконувати багатокрокові веб-завдання в автономному режимі. Такий агент повинен не лише автоматизувати браузерні дії, а й аналізувати зміст задачі, планувати послідовність виконання операцій, адаптувати поведінку залежно від проміжних результатів та забезпечувати прозорість reasoning-процесу.

Метою роботи є розробка та дослідження AI-агента для автоматичного виконання задач у веб-середовищі із використанням сучасних технологій штучного інтелекту, browser automation та reasoning-oriented архітектури. У межах роботи передбачається аналіз сучасних підходів до побудови AI-агентів, дослідження існуючих засобів веб-автоматизації, проектування архітектури системи та реалізація прототипу інтелектуального агента, здатного автономно взаємодіяти із веб-ресурсами та виконувати задачі різного рівня складності.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Сутність та роль AI-агентів у сучасних інформаційних системах

У сучасному етапі розвитку інформаційних технологій спостерігається поступовий перехід від традиційних програмних систем до більш гнучких, автономних та інтелектуально орієнтованих рішень. Одним із ключових напрямів цього розвитку є поява та активне впровадження AI-агентів – програмних сутностей, здатних самостійно взаємодіяти з цифровим середовищем, аналізувати інформацію та виконувати поставлені завдання без постійного контролю з боку людини.

AI-агент у загальному розумінні являє собою систему, яка сприймає інформацію із середовища через певні вхідні канали, обробляє її за допомогою алгоритмів штучного інтелекту та здійснює відповідні дії, спрямовані на досягнення визначеної мети. Ключовою характеристикою таких систем є їх здатність до автономності, адаптації та прийняття рішень в умовах невизначеності. На відміну від класичних програм, які функціонують за заздалегідь заданими правилами, AI-агенти можуть змінювати свою поведінку залежно від контексту, накопиченого досвіду та динаміки середовища [1].

Ідея створення агентів бере свій початок ще з ранніх досліджень у галузі штучного інтелекту, коли розглядалися моделі раціональної поведінки систем у складних середовищах. З розвитком обчислювальних можливостей, появою машинного навчання та обробки природної мови, концепція агентів зазнала суттєвої еволюції. Сучасні AI-агенти вже не обмежуються виконанням простих реактивних дій, а здатні до складного аналізу, планування та взаємодії з користувачем на рівні, наближеному до природної комунікації.

У структурі сучасних інформаційних систем AI-агенти виконують роль інтелектуального посередника між користувачем і цифровим

середовищем. Вони можуть автоматизувати широкий спектр задач, починаючи від обробки запитів і рекомендаційних сервісів до управління складними бізнес-процесами. Зокрема, AI-агенти активно застосовуються у віртуальних асистентах, чат-ботах, системах підтримки прийняття рішень, автоматизованому тестуванні програмного забезпечення, кібербезпеці, а також у сфері електронної комерції.

Особливого значення набуває використання AI-агентів у веб-середовищі, де вони можуть виконувати функції автоматизованої взаємодії з веб-ресурсами. Це включає навігацію по сторінках, заповнення форм, збір та обробку даних, а також виконання складних сценаріїв, які традиційно вимагали участі людини. У таких умовах агент виступає як активний учасник цифрового процесу, здатний не лише виконувати інструкції, а й інтерпретувати інформацію та адаптувати свою поведінку відповідно до змін у середовищі.

Поширення AI-агентів як технологічного тренду зумовлене кількома ключовими факторами. По-перше, це зростання обсягів даних і складності задач, що потребують обробки. По-друге, розвиток алгоритмів машинного навчання, які дозволяють системам навчатися на основі досвіду та покращувати свою ефективність. По-третє, зростаюча потреба у зниженні навантаження на користувача шляхом автоматизації рутинних процесів. У сукупності ці фактори формують передумови для переходу від традиційного програмного забезпечення до інтелектуальних агентних систем [2].

1.1.1 Основні підходи до побудови AI-агентів

Розвиток AI-агентів як складових сучасних інформаційних систем зумовив формування різних підходів до їх побудови, що відрізняються рівнем автономності, складністю прийняття рішень та здатністю до адаптації. У загальному вигляді підхід до побудови агента визначає, яким

чином він сприймає середовище, обробляє інформацію та формує свою поведінку у відповідь на зовнішні впливи.

Відмінності між підходами до побудови AI-агентів узагальнено у таблиці 1.1.

Таблиця 1.1 – Порівняння підходів до побудови AI-агентів

Підхід	Характеристика	Переваги	Обмеження
Реактивний	Миттєва реакція на стан середовища	Швидкість, простота	Відсутність планування
Деліберативний	Побудова моделі середовища та планування	Глибокий аналіз	Висока складність
Гібридний	Поєднання реакції та планування	Баланс ефективності	Складність архітектури
LLM-агенти	Використання мовних моделей	Гнучкість, адаптивність	Залежність від якості моделі

Одним із найпростіших і водночас історично перших підходів є реактивний підхід. У межах цього підходу агент функціонує за принципом прямого відображення стану середовища у дію. Тобто система не формує внутрішніх моделей або довгострокових планів, а реагує на поточний стан відповідно до заданих правил. Такі агенти відзначаються високою швидкістю роботи та простотою реалізації, однак їх можливості обмежені у складних або змінних умовах, де необхідне врахування попереднього досвіду або прогнозування результатів дій.

Більш складним є деліберативний підхід, який передбачає наявність внутрішньої моделі середовища та механізмів планування. Агент у цьому випадку не лише реагує на поточні події, а й аналізує можливі варіанти розвитку ситуації, формує цілі та обирає оптимальну послідовність дій для їх досягнення. Такий підхід дозволяє вирішувати складні задачі, що

потребують логічного мислення, проте супроводжується більшими обчислювальними витратами та складністю реалізації [3].

Компромісним варіантом між швидкістю реакції та глибиною аналізу є гібридний підхід. У цьому випадку поєднуються реактивні та деліберативні механізми, що дозволяє агенту одночасно швидко реагувати на зміни середовища і здійснювати стратегічне планування. Зазвичай у таких системах реалізується багаторівнева архітектура, де нижній рівень відповідає за оперативні реакції, а верхній за прийняття складних рішень. Саме гібридні агенти найчастіше використовуються у сучасних прикладних системах, де необхідна як швидкість, так і інтелектуальність поведінки.

Останнім часом особливого поширення набув підхід до побудови агентів на основі великих мовних моделей. Такі LLM-агенти використовують можливості моделей обробки природної мови для інтерпретації запитів, формування планів дій та взаємодії з цифровим середовищем. Вони здатні працювати з неструктурованими даними, розуміти текстові інструкції та адаптувати свою поведінку без жорстко заданих сценаріїв. Завдяки цьому LLM-агенти відкривають новий рівень автоматизації, де система може не лише виконувати команди, а й частково брати участь у прийнятті рішень.

Важливою особливістю сучасних підходів є їх орієнтація на інтеграцію різних технологій. У практичних реалізаціях AI-агенти часто поєднують елементи машинного навчання, обробки природної мови, аналізу даних та взаємодії з веб-сервісами. Це дозволяє створювати універсальні системи, здатні працювати у складних динамічних середовищах і виконувати широкий спектр задач.

1.1.2 Особливості функціонування агентів у веб-середовищі

Однією з ключових особливостей є необхідність взаємодії агента з веб-браузером як основним інструментом доступу до ресурсів. У цьому

випадку агент виконує дії, аналогічні до дій користувача, зокрема відкриття веб-сторінок, навігацію між ними, заповнення форм, натискання кнопок і обробку отриманих результатів. Такий підхід часто реалізується через спеціалізовані інструменти керування браузером, що дозволяють програмно відтворювати поведінку людини у цифровому середовищі.

Паралельно з цим важливу роль відіграє взаємодія через API інтерфейси. Використання API дозволяє агенту отримувати структуровані дані без необхідності обробки візуального інтерфейсу сторінки, що значно підвищує швидкість і надійність роботи. У цьому випадку агент діє як клієнт, що надсилає запити до серверів і отримує відповіді у вигляді даних, придатних для подальшого аналізу та використання.

Ще одним важливим аспектом є автоматизація дій користувача. У веб-середовищі значна частина задач є рутинною і повторюваною, наприклад введення даних, пошук інформації, обробка запитів або виконання стандартних операцій у веб-додатках. AI-агенти здатні брати на себе виконання таких задач, що дозволяє значно зменшити навантаження на людину і підвищити ефективність роботи. При цьому сучасні агенти можуть не лише повторювати заздалегідь визначені сценарії, а й адаптувати свої дії залежно від змін у середовищі.

Окрему роль відіграє процес збору даних з веб-ресурсів, відомий як web scraping. У цьому контексті агент здійснює аналіз структури веб-сторінок, виділяє необхідну інформацію і перетворює її у зручний для обробки формат. Складність цього процесу полягає в тому, що структура сторінок може змінюватися, а дані часто представлені у неструктурованому вигляді. Використання методів штучного інтелекту дозволяє підвищити точність виділення інформації та зробити процес більш стійким до змін.

Важливою характеристикою функціонування агентів у веб-середовищі є інтерактивність. Агент повинен не лише виконувати дії, а й аналізувати результати цих дій, коригувати свою поведінку і взаємодіяти з користувачем або іншими системами. Це передбачає використання

механізмів зворотного зв'язку, обробки текстових інструкцій і прийняття рішень у реальному часі. У сучасних системах така взаємодія часто реалізується за допомогою мовних моделей, що дозволяє агенту працювати з природною мовою і більш гнучко реагувати на зміни задачі [4].

1.2 Аналіз існуючих рішень та технологій автоматизації

Стрімкий розвиток інформаційних технологій зумовив появу великої кількості інструментів і підходів до автоматизації процесів у цифровому середовищі. Особливо це стосується веб-сфери, де значна частина задач пов'язана з повторюваними діями, обробкою даних та взаємодією з веб-інтерфейсами. У зв'язку з цим сформувався окремий напрям, орієнтований на створення технологій, здатних частково або повністю автоматизувати діяльність користувача.

Одним із найбільш поширених підходів є використання засобів керування браузером. До таких інструментів належать бібліотеки та фреймворки, що дозволяють програмно взаємодіяти з веб-сторінками, імітуючи дії людини. Вони забезпечують можливість відкриття сторінок, натискання елементів інтерфейсу, заповнення форм і отримання даних. Ці рішення широко застосовуються у тестуванні програмного забезпечення, автоматизації бізнес-процесів та зборі інформації з веб-ресурсів. Їх перевагою є універсальність і гнучкість, однак вони часто потребують ручного налаштування сценаріїв і не здатні самостійно адаптуватися до змін у структурі веб-інтерфейсу.

Іншим важливим напрямом є роботизована автоматизація процесів, яка орієнтована на відтворення дій користувача у різних програмних середовищах. RPA-системи дозволяють створювати сценарії виконання задач без глибокого програмування, що робить їх доступними для широкого кола користувачів. Вони активно використовуються у бізнесі для автоматизації рутинних операцій, таких як обробка документів, робота з

базами даних і виконання стандартних процедур. Проте обмеженням таких систем є їх залежність від заздалегідь визначених сценаріїв і низька здатність до адаптації у випадку нестандартних ситуацій [5].

З розвитком штучного інтелекту з'явилися більш складні підходи до автоматизації, що базуються на використанні машинного навчання і обробки природної мови. Такі системи здатні аналізувати текстову інформацію, інтерпретувати запити користувача і приймати рішення на основі отриманих даних. Це дозволяє значно розширити функціональні можливості автоматизації, переходячи від жорстко визначених алгоритмів до більш гнучких і адаптивних моделей.

Особливе місце серед сучасних рішень займають системи, що поєднують можливості автоматизації з використанням великих мовних моделей. Такі рішення дозволяють створювати агентів, які можуть працювати з текстовими інструкціями, формувати плани дій і взаємодіяти з веб-сервісами. Вони здатні виконувати складні задачі, що потребують аналізу контексту і логічного мислення, наприклад пошук інформації, обробку даних або автоматичне виконання багатокрокових операцій.

1.2.1 Сучасні інструменти автоматизації веб-задач

У сучасних умовах розвитку цифрових технологій автоматизація веб-задач стала важливою складовою як розробки програмного забезпечення, так і організації бізнес-процесів. Для реалізації таких задач використовується широкий спектр інструментів, які відрізняються за принципами роботи, рівнем складності та функціональними можливостями. Вони дозволяють автоматизувати взаємодію з веб-інтерфейсами, виконання рутинних операцій, а також інтеграцію із зовнішніми сервісами.

Однією з найбільш поширених груп є інструменти керування браузером. До них належать такі рішення, як Selenium, Puppeteer та Playwright, які дозволяють програмно взаємодіяти з веб-сторінками. Ці

інструменти забезпечують виконання дій, аналогічних до дій користувача, включаючи навігацію, обробку елементів сторінки та тестування інтерфейсів. Вони широко застосовуються у сфері автоматизованого тестування, збору даних та розробки веб-сервісів [6].

У таблиці 1.2 наведено порівняння основних інструментів автоматизації веб-задач.

Таблиця 1.2 – Порівняння сучасних інструментів автоматизації веб-задач

Інструмент	Тип	Основні можливості	Переваги	Обмеження
Selenium	Browser automation	Керування браузером, тестування	Універсальність, підтримка багатьох мов	Складність налаштування
Puppeteer	Browser automation	Робота з Chrome, автоматизація дій	Висока швидкість, простота	Обмеження браузером
Playwright	Browser automation	Кросбраузерна автоматизація	Гнучкість, сучасна архітектура	Вища складність
RPA-платформи	RPA	Автоматизація бізнес-процесів	Простота використання	Низька адаптивність
AutoGPT	LLM-агент	Автономне виконання задач	Висока автономність	Нестабільність результатів
LangChain	Agent framework	Побудова агентів, інтеграція з API	Гнучкість, розширюваність	Потребує налаштування

Паралельно з цим активно розвиваються системи роботизованої автоматизації процесів. Такі платформи дозволяють створювати сценарії виконання задач без необхідності глибокого програмування, що робить їх доступними для користувачів без технічної підготовки. Вони використовуються для автоматизації повторюваних дій, зокрема роботи з документами, веб-додатками та корпоративними системами. Проте їх можливості обмежені у випадках, коли необхідна гнучкість або адаптація до змін у середовищі.

З появою великих мовних моделей відбувся перехід до нового покоління інструментів автоматизації. Системи на зразок AutoGPT та агентні фреймворки, такі як LangChain, дозволяють створювати інтелектуальних агентів, здатних самостійно формувати послідовність дій, аналізувати результати та адаптувати свою поведінку. Вони можуть працювати з текстовими інструкціями, взаємодіяти з API та виконувати складні багатокрокові задачі без жорстко заданих сценаріїв [7].

Особливістю сучасних інструментів є тенденція до інтеграції різних підходів. Наприклад, AI-агенти можуть використовувати класичні засоби керування браузером як інструменти для виконання дій, а мовні моделі для прийняття рішень і планування. Це створює передумови для побудови універсальних систем автоматизації, здатних працювати у складних динамічних умовах.

1.2.2 Використання штучного інтелекту для автоматичного виконання задач

Сучасний етап розвитку технологій автоматизації характеризується переходом від жорстко заданих алгоритмів до інтелектуальних систем, здатних самостійно аналізувати інформацію, приймати рішення та адаптувати свою поведінку. У цьому контексті використання штучного інтелекту стає ключовим фактором підвищення ефективності автоматичного виконання задач у веб-середовищі (рисунок 1.1).

Однією з основних технологічних основ таких систем є методи обробки природної мови. Завдяки розвитку NLP алгоритмів стало можливим перетворення текстових інструкцій користувача у формалізовані дії, які система може виконати. Це дозволяє будувати інтерфейси взаємодії, у яких користувач формулює задачу природною мовою, а система інтерпретує її та визначає необхідну послідовність кроків.

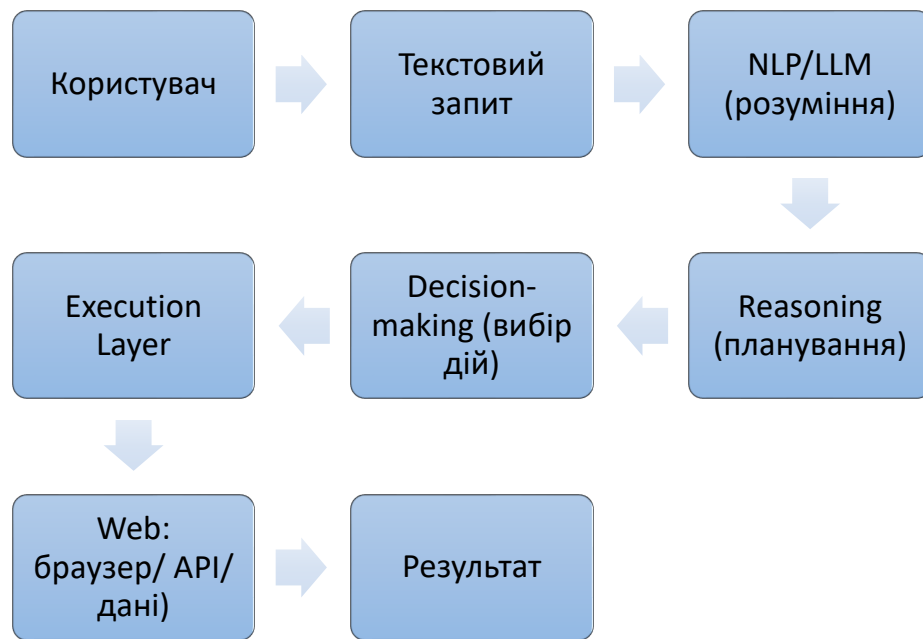


Рисунок 1.1 – Архітектура AI-агента для автоматичного виконання задач

Подальший розвиток цього підходу пов'язаний із використанням великих мовних моделей, які здатні не лише аналізувати текст, а й виконувати складні когнітивні операції. Такі моделі використовуються як ядро AI-агентів і забезпечують функції розуміння запитів, генерації відповідей, планування дій та адаптації до змін у середовищі. Вони дозволяють перейти від простого виконання команд до більш складних сценаріїв, що включають аналіз контексту і побудову логіки виконання задач [8].

Важливою складовою автоматизації є механізми прийняття рішень. У цьому випадку система повинна не лише визначити можливі варіанти дій, а й обрати найбільш ефективний із них. Для цього використовуються підходи, що базуються на оцінці стану середовища, прогнозуванні результатів і виборі оптимальної стратегії. У поєднанні з мовними моделями це дозволяє агенту діяти у складних і змінних умовах без жорстко визначених сценаріїв.

Окрему роль відіграє процес reasoning, тобто логічного міркування. Сучасні AI-системи здатні виконувати послідовний аналіз задачі, розбивати її на підзадачі та формувати план їх вирішення. Це особливо важливо у

випадках, коли задача не має очевидного рішення або потребує виконання декількох взаємопов'язаних дій. У таких умовах система виступає не лише виконавцем, а й активним учасником процесу прийняття рішень.

Інтеграція зазначених компонентів дозволяє створювати інтелектуальні системи, які можуть автоматично виконувати складні задачі у веб-середовищі. Вони здатні аналізувати текстові запити, визначати необхідні дії, взаємодіяти з веб-інтерфейсами та коригувати свою поведінку залежно від отриманих результатів. Це відкриває нові можливості для автоматизації, де система не лише виконує команди, а й частково замінює когнітивну діяльність людини [9].

1.2.3 Переваги та обмеження існуючих підходів

Сучасні технології автоматизації веб-задач і використання штучного інтелекту демонструють значний прогрес у напрямі підвищення ефективності цифрових процесів. Різноманіття інструментів і підходів дозволяє вирішувати широкий спектр задач, починаючи від простого відтворення дій користувача і завершуючи реалізацією складних сценаріїв взаємодії з веб-середовищем. Водночас, незважаючи на досягнуті результати, існуючі рішення мають низку особливостей, які обмежують їх універсальність і ефективність.

До основних переваг сучасних підходів слід віднести високий рівень автоматизації рутинних процесів. Інструменти керування браузером та системи роботизованої автоматизації дозволяють значно скоротити час виконання стандартних операцій і зменшити навантаження на користувача. Крім того, використання штучного інтелекту відкриває можливості для обробки неструктурованих даних, аналізу текстових запитів і формування більш гнучких сценаріїв виконання задач.

Важливою перевагою є також розвиток агентних підходів, що дозволяють створювати системи з підвищеним рівнем автономності. Такі

системи здатні виконувати багатокрокові задачі, адаптуватися до змін середовища і використовувати різні інструменти в межах єдиного процесу. Це наближає автоматизацію до більш інтелектуального рівня, де система виступає не лише виконавцем, а й активним учасником взаємодії.

Проте, поряд із перевагами, існують і суттєві обмеження (рисунок 1.2), які стримують повноцінне використання цих технологій. Однією з ключових проблем є залежність багатьох інструментів від жорстко визначених сценаріїв. У випадку зміни структури веб-інтерфейсу або логіки роботи сервісу такі системи можуть втрачати працездатність або потребувати ручного переналаштування.

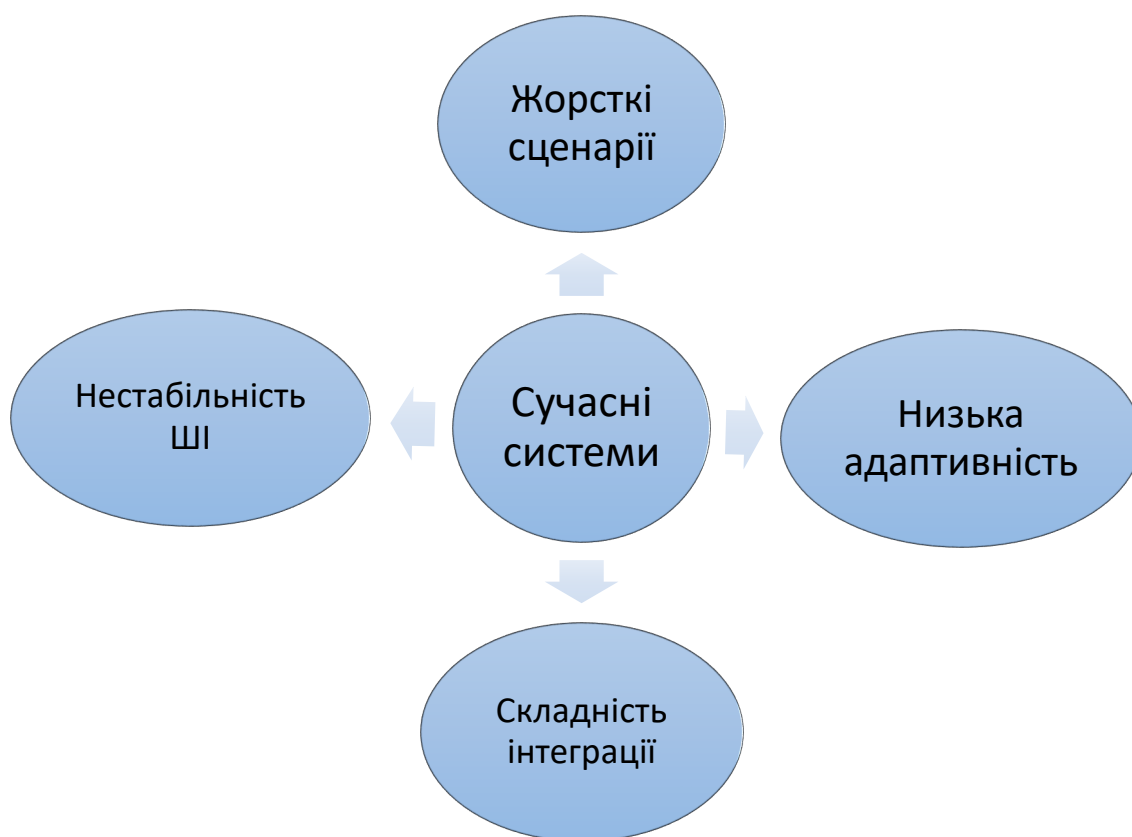


Рисунок 1.2 – Обмеження сучасних підходів до автоматизації

Іншим важливим обмеженням є недостатній рівень адаптивності у складних або нестандартних ситуаціях. Навіть системи, що використовують штучний інтелект, не завжди здатні коректно інтерпретувати контекст

задачі або приймати оптимальні рішення в умовах невизначеності. Це особливо проявляється у веб-середовищі, де інформація часто має неструктурований характер і потребує глибокого аналізу.

Суттєвою проблемою є також обмежена інтеграція різних компонентів автоматизації. Багато існуючих рішень орієнтовані на виконання окремих функцій, наприклад взаємодію з браузером або обробку тексту, але не забезпечують повноцінної координації між ними. У результаті виникає необхідність поєднання кількох інструментів, що ускладнює архітектуру системи і знижує її надійність.

Крім того, системи на основі великих мовних моделей, незважаючи на їх значний потенціал, мають власні обмеження. До них належать нестабільність результатів, залежність від якості вхідних даних і складність контролю процесу прийняття рішень. У деяких випадках такі системи можуть генерувати некоректні або непередбачувані результати, що є критичним для задач автоматизації [10].

Ще одним аспектом є питання продуктивності та ресурсомісткості. Інтелектуальні системи, особливо ті, що використовують складні моделі машинного навчання, потребують значних обчислювальних ресурсів, що може обмежувати їх використання у реальних умовах.

1.3 Актуальність створення AI-агента для автоматизації задач у веб-середовищі

У сучасному цифровому суспільстві веб-середовище виступає одним із ключових просторових рівнів взаємодії людини з інформаційними системами. Значна частина діяльності користувачів, зокрема пошук інформації, робота з онлайн-сервісами, обробка даних, комунікація та виконання професійних обов'язків, здійснюється саме через веб-інтерфейси. У зв'язку з цим обсяг задач, що виконуються у веб-середовищі,

постійно зростає, а разом із ним збільшується і навантаження на користувача.

Більшість таких задач мають рутинний і повторюваний характер. До них належать заповнення форм, обробка однотипних запитів, пошук і збір інформації, робота з електронними сервісами та виконання стандартних операцій у веб-додатках. Незважаючи на їх відносну простоту, ці дії потребують значних витрат часу та уваги, що знижує загальну ефективність діяльності і створює передумови для виникнення помилок.

У сфері бізнесу ця проблема набуває особливої актуальності. Організації змушені витрачати значні ресурси на виконання повторюваних операцій, що не створюють доданої цінності, але є необхідними для підтримки функціонування процесів. Автоматизація таких задач дозволяє підвищити продуктивність, зменшити витрати і забезпечити стабільність виконання операцій. Водночас традиційні підходи до автоматизації не завжди відповідають вимогам сучасного середовища, оскільки вони орієнтовані на статичні сценарії і не враховують динамічний характер веб-ресурсів.

Розвиток технологій штучного інтелекту створює передумови для нового рівня автоматизації, у якому система здатна не лише виконувати заздалегідь визначені дії, а й аналізувати ситуацію, приймати рішення і адаптувати свою поведінку. Особливу роль у цьому процесі відіграють AI-агенти, які можуть виступати як автономні виконавці задач у веб-середовищі. Вони здатні працювати з різними джерелами даних, взаємодіяти з веб-інтерфейсами та виконувати складні багатокрокові сценарії.

Актуальність створення AI-агента для автоматизації задач у веб-середовищі зумовлена необхідністю поєднання гнучкості, інтелектуальності та здатності до адаптації в межах єдиної системи. На відміну від традиційних інструментів автоматизації, такий агент може інтерпретувати запити користувача, формувати план дій і виконувати його

з урахуванням змін у середовищі. Це дозволяє значно розширити спектр задач, які можуть бути автоматизовані.

Крім того, важливим фактором є орієнтація сучасних цифрових рішень на зручність і доступність для користувача. Використання AI-агентів дозволяє зменшити складність взаємодії з системами, замінюючи набір технічних дій більш природною формою комунікації. У цьому випадку користувач формулює задачу у вигляді текстового запиту, а система самостійно визначає спосіб її виконання.

У контексті розвитку інформаційних технологій створення AI-агентів також відповідає загальній тенденції переходу до інтелектуальних систем, здатних діяти автономно та підтримувати прийняття рішень. Це відкриває нові можливості для автоматизації як у професійній діяльності, так і у повсякденному житті користувачів.

Актуальність розробки AI-агента для автоматичного виконання задач у веб-середовищі визначається зростаючою потребою в автоматизації рутинних процесів, обмеженнями існуючих рішень і можливостями, які відкриває використання штучного інтелекту. Реалізація такого підходу дозволяє підвищити ефективність взаємодії з веб-середовищем і створити передумови для подальшого розвитку інтелектуальних систем [11].

2 ТЕОРЕТИЧНІ ДОСЛІДЖЕННЯ

2.1 Аналіз сучасних AI-агентів для автоматизації веб-задач

2.1.1 Загальна характеристика сучасних AI-агентів

Упродовж останніх років технології штучного інтелекту демонструють стрімкий розвиток, а одним із найбільш перспективних напрямів стали AI-агенти, здатні автономно виконувати складні завдання у цифровому середовищі. Особливої актуальності такі системи набули після активного розвитку великих мовних моделей (Large Language Models, LLM), які забезпечили можливість не лише генерувати текст, а й аналізувати контекст, планувати послідовність дій, взаємодіяти з інструментами та приймати рішення в режимі реального часу.

Сучасні AI-агенти поступово переходять від ролі звичайних чат-ботів до повноцінних автономних систем, здатних взаємодіяти з веб-ресурсами, працювати з браузером, виконувати багатокрокові сценарії та автоматизувати рутинні процеси без постійного втручання користувача. Зростання інтересу до таких систем пов'язане зі збільшенням кількості цифрових сервісів, необхідністю автоматизації повторюваних дій, а також потребою у більш інтелектуальних механізмах взаємодії з інформаційними системами [16].

Поява сучасних мовних моделей, зокрема GPT, Claude, Gemini та інших, суттєво змінила підходи до автоматизації веб-задач. Якщо раніше автоматизація базувалася переважно на жорстко заданих сценаріях та правилах, то сучасні AI-агенти здатні адаптуватися до змін середовища, аналізувати текстовий контекст, самостійно обирати інструменти для виконання задачі та формувати послідовність дій залежно від ситуації. Завдяки цьому стало можливим створення систем, які не лише виконують

команди, а й частково імітують логіку поведінки людини під час роботи з веб-інтерфейсами.

На відміну від класичних систем роботизованої автоматизації процесів (RPA), сучасні AI-агенти характеризуються значно вищим рівнем гнучкості та автономності. Традиційні RPA-рішення зазвичай працюють за наперед визначеним алгоритмом і є ефективними лише у стабільних середовищах із незмінною структурою інтерфейсу. Будь-які зміни у веб-сторінці, структурі елементів або логіці взаємодії можуть призводити до помилок виконання сценарію. Натомість AI-агенти використовують можливості LLM для аналізу контексту, розпізнавання елементів інтерфейсу та прийняття рішень у динамічному середовищі. Це дозволяє їм адаптуватися до часткових змін веб-сторінок і виконувати складніші сценарії автоматизації [17].

Водночас веб-середовище залишається одним із найскладніших середовищ для побудови автономних агентів. Однією з основних причин є динамічність сучасних веб-інтерфейсів. Більшість сучасних сайтів активно використовують JavaScript для генерації та оновлення контенту в реальному часі, що ускладнює стабільне визначення елементів сторінки. Динамічна зміна DOM-структури може призводити до втрати посилань на елементи інтерфейсу та помилок під час виконання автоматизованих дій.

Додаткову складність створюють SPA-додатки (Single Page Application), у яких значна частина контенту завантажується асинхронно без повного оновлення сторінки. У таких системах класичні методи автоматизації часто працюють нестабільно через затримки завантаження, зміну структури компонентів або повторне рендерування інтерфейсу. У результаті AI-агент повинен не лише виконувати дії, а й постійно аналізувати поточний стан веб-середовища.

Суттєвою проблемою для автономних агентів також є системи захисту веб-ресурсів, зокрема CAPTCHA, механізми anti-bot detection та багаторівнева авторизація користувачів. Багато сучасних сайтів

використовують засоби виявлення автоматизованої поведінки, що ускладнює стабільну роботу агентів у браузері. Крім того, виконання задач часто потребує проходження складних сценаріїв автентифікації, підтвердження дій або взаємодії з динамічними формами [18].

Окрему складність становлять багатокрокові сценарії виконання задач. У реальних умовах AI-агент повинен не лише виконати одну дію, а й планувати послідовність взаємопов'язаних кроків: аналізувати веб-сторінку, знаходити необхідні елементи, переходити між розділами, заповнювати форми, обробляти помилки та коригувати подальші дії залежно від результату виконання попередніх етапів. Саме тому сучасні AI-агенти поєднують можливості мовних моделей, браузерної автоматизації та систем управління інструментами.

2.1.2 Порівняльний аналіз сучасних AI-агентів

Сучасний етап розвитку AI-агентів характеризується появою великої кількості систем, орієнтованих на автономне виконання задач у веб-середовищі. Такі агенти поєднують можливості великих мовних моделей, механізми планування дій, browser automation та системи взаємодії з зовнішніми інструментами. Водночас різні рішення суттєво відрізняються за рівнем автономності, архітектурою, підтримкою браузерної взаємодії, можливостями інтеграції та стабільністю виконання задач.

Для проведення аналізу сучасні AI-агенти доцільно умовно розділити на декілька категорій:

- комерційні browser-agents;
- автономні task-oriented агенти;
- фреймворки для побудови multi-agent систем;
- open-source browser-native агенти.

До комерційних browser-agents належать системи, орієнтовані на автоматизацію взаємодії з веб-інтерфейсами за допомогою великих мовних

моделей. Прикладами таких рішень є OpenAI Operator, Anthropic Computer Use та Google Project Mariner. Дані системи забезпечують високий рівень автономності та здатні виконувати складні сценарії роботи з браузером, однак переважно залишаються закритими комерційними платформами з обмеженими можливостями локального розгортання та кастомізації.

Окрему категорію формують автономні task-oriented агенти, зокрема AutoGPT, BabyAGI та AgentGPT. Такі системи орієнтовані на автоматичне планування та виконання багатокрокових задач із мінімальним втручанням користувача. Основною перевагою подібних агентів є високий рівень автономності та здатність до self-planning, проте на практиці вони часто характеризуються нестабільністю reasoning-процесів, значним використанням токенів та труднощами під час виконання складних веб-сценаріїв.

Значного поширення також набули фреймворки для побудови multi-agent систем, серед яких особливо виділяються LangChain Agents, CrewAI та Microsoft AutoGen. Основною особливістю таких платформ є підтримка tool calling, memory-механізмів та orchestration між декількома агентами. Це дозволяє створювати складні системи автоматизації з розподілом ролей між окремими компонентами. Водночас збільшення кількості агентів та інструментів часто призводить до зростання складності архітектури, навантаження на мовну модель та підвищення ризику логічних помилок під час виконання задач.

Окрему групу формують browser-native open-source агенти, зокрема browser-use та OpenDevin. Дані системи орієнтовані на максимально наближену до людини взаємодію з браузером та операційним середовищем. Вони дозволяють AI-агенту працювати безпосередньо з елементами інтерфейсу, виконувати навігацію між сторінками, взаємодіяти з формами та аналізувати структуру веб-ресурсів у реальному часі. Проте ефективність подібних систем значною мірою залежить від стабільності DOM-структури,

коректності browser automation та можливостей мовної моделі щодо інтерпретації поточного стану веб-середовища.

Сучасні AI-агенти порівняно за ключовими характеристиками, наведеними у таблиці 2.1.

Таблиця 2.1 – Порівняльний аналіз сучасних AI-агентів

AI-агент	Тип архітектури	Browser interaction	Tool calling	Multi-step reasoning	Memory support	Open-source	Основні недоліки
Operator	Browser-agent	Так	+	Високий	Частково	–	Закрита архітектура, висока вартість
Computer Use	GUI-agent	Так	±	Високий	Обмежено	–	Нестабільність GUI-взаємодії
Project Mariner	Browser-agent	Так	+	Високий	Частково	–	Обмежений доступ до системи
AutoGPT	Autonomous agent	Обмежено	+	Високий	Так	–	Високі витрати токенів
BabyAGI	Task-oriented agent	Ні	+	Середній	Частково	+	Примітивне планування
LangChain Agents	Agent framework	Через інтеграції	+	Високий	Так	+	Складність конфігурації
CrewAI	Multi-agent framework	Через інтеграції	+	Високий	Так	+	Складність orchestration
AutoGen	Multi-agent framework	Через інтеграції	+	Високий	Так	+	Високе навантаження на LLM
browser-use	Browser-native agent	Так	+	Середній	Обмежено	+	Залежність від якості DOM
OpenDevin	Autonomous development agent	Так	+	Високий	Частково	+	Експериментальна стабільність
AgentGPT	Autonomous web-agent	Обмежено	+	Середній	Обмежено	±	Нестабільне виконання задач

Проведений аналіз показав, що сучасні AI-агенти демонструють високий рівень автономності під час виконання багатокрокових задач, однак більшість із них залишаються залежними від якості веб-інтерфейсу,

стабільності DOM-структури та можливостей мовної моделі. Значна частина систем усе ще має труднощі з адаптацією до динамічних змін веб-середовища, асинхронного завантаження контенту та нестабільної структури SPA-додатків.

Комерційні рішення, зокрема Operator та Computer Use, забезпечують ефективну взаємодію з браузером і демонструють високий рівень автономності, проте характеризуються закритістю архітектури, залежністю від зовнішньої інфраструктури та обмеженими можливостями локального розгортання. Крім того, використання таких систем часто супроводжується значними витратами на API та обмеженнями щодо кастомізації внутрішньої логіки агентів.

Водночас open-source системи, зокрема AutoGPT, browser-use та OpenDevin, надають ширші можливості модифікації та інтеграції у власні програмні середовища. Такі рішення дозволяють реалізовувати власні механізми планування, memory management та browser automation, однак нерідко демонструють нестабільність під час виконання складних сценаріїв автоматизації, особливо в умовах динамічного веб-середовища.

2.1.3 Основні проблеми та обмеження сучасних AI-агентів

Попри стрімкий розвиток сучасних AI-агентів та суттєве зростання рівня їх автономності, існуючі системи все ще характеризуються значною кількістю технічних, архітектурних та безпекових обмежень. Особливо гостро дані проблеми проявляються під час роботи у веб-середовищі, де AI-агент повинен взаємодіяти з динамічними інтерфейсами, аналізувати контекст задачі, використовувати зовнішні інструменти та приймати рішення в режимі реального часу.

Однією з ключових груп проблем є архітектурні обмеження сучасних AI-агентів. Значна частина існуючих систем має обмежену підтримку довготривалої пам'яті (memory persistence), через що агент не здатний

ефективно зберігати та використовувати результати попередніх дій у межах складних багатокрокових сценаріїв. У результаті система може повторно виконувати однакові дії, втрачати контекст задачі або формувати суперечливі рішення.

Суттєвою проблемою також залишається нестабільність механізмів планування та reasoning. У складних сценаріях автономні агенти нерідко демонструють recursion loops, тобто циклічне повторення однакових дій без досягнення кінцевого результату. Крім того, навіть сучасні LLM не гарантують стабільності логічних висновків під час виконання довгих ланцюгів міркувань, що може призводити до помилкового планування задач або некоректної взаємодії з інструментами.

Окрему категорію становлять проблеми веб-взаємодії. Сучасне веб-середовище характеризується високою динамічністю та складною структурою інтерфейсів. Більшість сучасних сайтів активно використовують JavaScript rendering та SPA-архітектури (Single Page Application), у яких значна частина контенту генерується асинхронно без повного оновлення сторінки. Це ускладнює стабільне визначення елементів DOM-структури та створює труднощі для browser automation.

Додаткові проблеми виникають через async loading контенту, коли окремі елементи сторінки можуть завантажуватися із затримкою або змінювати своє розташування під час виконання сценарію. У таких умовах AI-агент повинен постійно аналізувати поточний стан веб-сторінки та адаптувати логіку взаємодії до змін середовища.

Суттєвим бар'єром для автоматизації також залишаються CAPTCHA та anti-bot systems. Багато сучасних веб-ресурсів використовують механізми виявлення автоматизованої поведінки, які блокують підозрілі дії або потребують додаткового підтвердження користувача. У результаті навіть високорівневі AI-агенти можуть втрачати стабільність під час взаємодії з окремими веб-сервісами.

Важливою групою обмежень є проблеми, пов'язані безпосередньо з великими мовними моделями. Однією з найбільш поширених проблем залишаються hallucinations, тобто генерація неправдивої або некоректної інформації. У контексті AI-агентів це може призводити до помилкових рішень, некоректного виконання команд або небезпечної взаємодії з веб-ресурсами.

Крім того, сучасні AI-агенти характеризуються значним використанням обчислювальних ресурсів. Під час виконання складних багатокрокових задач обсяг контексту може швидко збільшуватися, що створює проблему token overflow та підвищує навантаження на мовну модель. Це, своєю чергою, призводить до high latency, тобто збільшення часу обробки запитів, а також до зростання вартості використання API сучасних LLM.

Окрему увагу необхідно приділити безпековим проблемам AI-агентів. Через активне використання tool calling та browser automation сучасні агенти отримують можливість взаємодіяти із зовнішніми ресурсами, файлами та веб-сервісами, що створює додаткові ризики безпеки.

Однією з найбільш актуальних загроз є Prompt Injection, за якого зломисник вбудовує спеціально сформовані інструкції у веб-контент або зовнішні джерела даних з метою зміни поведінки AI-агента. Подібні атаки можуть призводити до витоку даних, виконання небажаних дій або обходу системних обмежень. Додаткову небезпеку становлять data leakage, unsafe tool execution та insecure browser actions. За відсутності ефективних механізмів контролю агент може передавати конфіденційні дані зовнішнім сервісам, запускати небезпечні інструменти або виконувати неконтрольовані дії у браузері. У зв'язку з цим питання безпечного управління інструментами та контролю дій AI-агента є одним із ключових напрямів сучасних досліджень у сфері автономних систем.

Проблеми сучасних AI-агентів представлено на рисунку 2.1.



Рисунок 2.1 – Основні проблеми AI-агентів у веб-середовищі

Таким чином, аналіз сучасних AI-агентів показав, що існуючі рішення демонструють значний прогрес у сфері автоматизації веб-задач, проте жодне з них не забезпечує одночасно:

- високого рівня автономності;
- стабільної взаємодії з веб-інтерфейсами;
- ефективного управління інструментами;
- помірного використання обчислювальних ресурсів;
- підтримки безпечного виконання задач;
- та простоти інтеграції у локальні програмні середовища.

Це обумовлює доцільність розробки власного AI-агента, орієнтованого на поєднання гнучкої архітектури, browser automation та механізмів безпечного tool-calling.

2.2 Аналіз інструментів та технологій для реалізації AI-агента

2.2.1 Фреймворки для побудови AI-агентів

Стрімкий розвиток AI-агентів сприяв появі великої кількості спеціалізованих фреймворків, орієнтованих на створення автономних систем на основі великих мовних моделей. Такі платформи забезпечують механізми взаємодії з LLM, підтримку tool calling, memory management, orchestration між агентами та інтеграцію із зовнішніми сервісами. Вибір фреймворку є одним із ключових етапів під час розробки AI-агента, оскільки саме архітектурні можливості платформи визначають рівень масштабованості, стабільності та гнучкості майбутньої системи.

Одним із найбільш поширених рішень є LangChain, який став фактичним стандартом для створення LLM-based applications та AI-агентів. Основною перевагою LangChain є велика кількість інтеграцій із мовними моделями, інструментами browser automation, базами даних та API-сервісами [19]. Фреймворк підтримує реалізацію tool calling, memory-механізмів та складних reasoning-ланцюгів, що робить його зручним для побудови автономних агентів. Водночас система характеризується значною складністю конфігурації та високим рівнем абстракції, що може ускладнювати оптимізацію продуктивності.

Популярним сучасним рішенням також є CrewAI, орієнтований на побудову multi-agent систем. Даний фреймворк дозволяє організовувати взаємодію між декількома агентами, які виконують різні ролі у межах спільного сценарію [20]. CrewAI забезпечує ефективну orchestration-логіку та підтримує побудову складних workflow, однак зі збільшенням кількості агентів істотно зростає складність керування системою та навантаження на мовні моделі.

Важливе місце серед сучасних платформ займає Microsoft AutoGen, розроблений для створення agent-to-agent interaction систем. Основною

особливістю AutoGen є підтримка автономного обміну повідомленнями між агентами, що дозволяє будувати складні системи колективного reasoning та розподіленого виконання задач. Водночас подібний підхід характеризується високими обчислювальними витратами та значним використанням токенів під час багатокрокових сценаріїв.

Ще одним перспективним фреймворком є Semantic Kernel, розроблений компанією Microsoft [21]. Дана платформа орієнтована на інтеграцію LLM у традиційні програмні системи та підтримує поєднання класичних програмних компонентів із AI-функціональністю. Semantic Kernel забезпечує зручну інтеграцію memory, planner-механізмів та AI-functions, проте має меншу гнучкість у побудові автономних browser-native агентів порівняно з LangChain або CrewAI.

Для реалізації Retrieval-Augmented Generation (RAG) систем активно використовується Haystack. Основною перевагою даного фреймворку є ефективна робота з документами, пошуком інформації та інтеграцією retrieval-механізмів із мовними моделями. Haystack демонструє високу ефективність у задачах пошуку та аналізу текстових даних, однак менш орієнтований на реалізацію повноцінних автономних AI-агентів для browser automation [22].

Окремої уваги заслуговує SmolAgents, який позиціонується як lightweight-фреймворк для створення компактних AI-агентів із підтримкою tool calling [23]. Основною перевагою системи є простота архітектури та низький поріг входження для розробника. Проте через обмежені можливості orchestration та memory management даний підхід менш придатний для реалізації складних багатокomпонентних AI-систем.

Проведений аналіз показав, що сучасні фреймворки для побудови AI-агентів значно відрізняються за своїм призначенням та архітектурними можливостями.

2.2.2 Інструменти браузерної автоматизації

Одним із ключових компонентів сучасних AI-агентів є система browser automation, яка забезпечує взаємодію з веб-інтерфейсами, навігацію між сторінками, обробку форм, виконання дій користувача та отримання інформації з веб-ресурсів. Саме від ефективності інструментів браузерної автоматизації значною мірою залежить стабільність роботи AI-агента у веб-середовищі.

Сучасний ринок browser automation представлений великою кількістю платформ, серед яких найбільш поширеними є Selenium, Playwright, Puppeteer, browser-use та Stagehand. Дані рішення відрізняються за архітектурою, швидкістю роботи, стабільністю взаємодії з динамічними веб-інтерфейсами та рівнем інтеграції з AI-агентами.

Одним із найстаріших і найпоширеніших інструментів автоматизації є Selenium. Даний фреймворк тривалий час залишався стандартом для тестування веб-застосунків та автоматизації браузерної взаємодії. Основною перевагою Selenium є підтримка великої кількості браузерів, мов програмування та широка екосистема бібліотек. Проте архітектура Selenium була створена переважно для класичного тестування веб-інтерфейсів, а не для автономних AI-агентів. Через це під час роботи з сучасними SPA-застосунками та динамічними JavaScript-інтерфейсами система часто демонструє нестабільність, потребує складної конфігурації waits та характеризується низькою швидкістю виконання сценаріїв [24].

Суттєвим кроком у розвитку browser automation став Puppeteer, розроблений компанією Google. Даний інструмент забезпечує пряме керування браузером Chromium через DevTools Protocol, що дозволило суттєво підвищити швидкість та стабільність автоматизації. Puppeteer добре адаптований до сучасних JavaScript-інтерфейсів та SPA-архітектур, однак його головним обмеженням є орієнтація переважно на Chromium-based браузери, що зменшує універсальність використання системи [25].

Одним із найбільш перспективних сучасних рішень є Playwright, який активно використовується для створення AI-агентів та browser-native autonomous systems. Playwright був розроблений як більш сучасна та універсальна альтернатива Selenium і Puppeteer. Інструмент підтримує Chromium, Firefox та WebKit, забезпечує стабільну роботу з SPA-додатками, async loading та dynamic DOM changes. Крім того, Playwright має вбудовані механізми автоматичного очікування елементів (auto-waiting), що значно підвищує стабільність browser automation у складних веб-сценаріях [26].

Важливою перевагою Playwright є підтримка асинхронної моделі виконання та ефективна інтеграція з сучасними AI-фреймворками. Це дозволяє використовувати його як основу для AI-агентів, здатних виконувати багатокрокові сценарії взаємодії з веб-ресурсами в реальному часі. Саме тому Playwright дедалі частіше використовується у browser-native агентах та сучасних AI automation systems.

Окремої уваги заслуговує browser-use, який є спеціалізованим browser-native інструментом для AI-агентів. Дана система орієнтована на інтеграцію browser automation безпосередньо з великими мовними моделями та дозволяє агенту взаємодіяти з елементами веб-інтерфейсу майже на рівні поведінки людини. Водночас ефективність browser-use значною мірою залежить від стабільності DOM-структури та якості reasoning мовної моделі.

Ще одним сучасним рішенням є Stagehand, орієнтований на AI-assisted browser automation. Основною особливістю Stagehand є спрощення взаємодії між LLM та браузером через abstraction layer для роботи з елементами сторінки [27]. Подібний підхід дозволяє зменшити складність інтеграції browser automation у AI-системи, однак платформа поки що залишається менш зрілою порівняно з Playwright або Selenium.

Сучасні інструменти browser automation суттєво відрізняються за рівнем адаптації до AI-агентів та динамічного веб-середовища.

2.2.3 Аналіз мовних моделей для AI-агентів

Одним із ключових компонентів сучасних AI-агентів є великі мовні моделі (Large Language Models, LLM), які забезпечують аналіз контексту, формування рішень, планування дій та взаємодію з інструментами. Саме можливості мовної моделі значною мірою визначають рівень автономності AI-агента, стабільність reasoning-процесів та ефективність виконання багатокрокових сценаріїв у веб-середовищі.

Сучасний ринок LLM характеризується великою кількістю моделей, що відрізняються за архітектурою, функціональними можливостями та підходами до інтеграції у AI-системи [28]. Найбільш поширеними рішеннями для побудови AI-агентів є моделі сімейства GPT від OpenAI, Claude від Anthropic, Gemini від Google, DeepSeek, Meta Llama та Mistral AI.

Моделі GPT є одними з найбільш поширених рішень для реалізації AI-агентів завдяки високому рівню reasoning, підтримці tool calling та ефективній роботі зі structured output. Дані моделі демонструють високу якість chain-of-thought reasoning та здатні виконувати складні багатокрокові сценарії взаємодії з веб-середовищем. Крім того, GPT-моделі активно використовуються у сучасних browser-agents завдяки підтримці function calling та великому context window. Водночас використання даних моделей часто супроводжується високою вартістю API та залежністю від зовнішньої інфраструктури.

Важливим конкурентом GPT є Claude, який орієнтований на безпечну взаємодію з користувачем та стабільне long-context reasoning. Однією з головних переваг Claude є велике context window та нижчий рівень hallucinations під час обробки довгих документів або складних сценаріїв аналізу інформації. Це робить модель перспективною для AI-агентів, які працюють із великими обсягами контексту або виконують складні multi-step tasks.

Значного розвитку також набули моделі Gemini, орієнтовані на multimodal interaction та інтеграцію з екосистемою сервісів Google. Дані моделі демонструють високу швидкість роботи та ефективну інтеграцію з веб-сервісами, однак рівень відкритості системи та можливості локального розгортання залишаються обмеженими.

Окремої уваги заслуговує DeepSeek, який став одним із найбільш популярних open-weight рішень для reasoning-oriented AI-систем. Основною перевагою DeepSeek є відносно низька вартість використання та високий рівень логічного reasoning у задачах програмування та tool usage. Завдяки цьому модель активно використовується у lightweight AI-agent systems та open-source automation frameworks.

Важливу роль у розвитку локальних AI-систем відіграють моделі сімейства Llama. Їх основною перевагою є можливість локального запуску та модифікації архітектури під конкретні задачі. Це дозволяє реалізовувати AI-агентів із підвищеним рівнем контролю над даними та lower-cost infrastructure. Проте локальні моделі часто поступаються комерційним LLM за якістю reasoning та стабільністю виконання складних багатокрокових задач [29].

Суттєву популярність останнім часом також отримали моделі Mistral, які поєднують компактність архітектури, високу швидкість роботи та підтримку локального розгортання. Дані моделі демонструють ефективність у lightweight AI-systems та можуть використовуватися у сценаріях, де критичними є швидкість відповіді та помірне використання ресурсів.

Порівняльний аналіз мовних моделей виконано у таблиці 2.2.

Проведений аналіз показав, що сучасні мовні моделі суттєво відрізняються не лише якістю генерації тексту, а й рівнем адаптації до задач автономних AI-агентів. Для ефективної роботи AI-агента важливими є не лише знання мовної моделі, а й її здатність до planning, chain-of-thought reasoning, function calling та формування structured output.

Таблиця 2.2 – Порівняльний аналіз мовних моделей для AI-агентів

Модель	Reasoning	Context window	Швидкість	Локальний запуск	Hallucination rate
GPT	Високий	Високий	Висока	Ні	Низький
Claude	Високий	Дуже високий	Середня	Ні	Низький
Gemini	Високий	Високий	Висока	Ні	Середній
DeepSeek	Високий	Середній	Висока	Частково	Середній
Llama	Середній	Середній	Середня	Так	Середній
Mistral	Середній	Середній	Висока	Так	Середній

Результати аналізу свідчать про те, що найбільш перспективними для реалізації AI-агентів є мовні моделі, які поєднують високий рівень reasoning, підтримку tool calling, стабільний chain-of-thought та можливість роботи зі structured output. Саме такі характеристики є критично важливими для створення автономних систем автоматизації у веб-середовищі.

2.3 Порівняльний аналіз підходів до побудови AI-агентів

Сучасні AI-агенти можуть будуватися на основі різних архітектурних підходів, які відрізняються рівнем автономності, складністю реалізації, стабільністю reasoning та особливостями взаємодії із зовнішнім середовищем. Вибір підходу до побудови AI-агента безпосередньо впливає на ефективність виконання багатокрокових задач, можливості browser automation та адаптацію системи до динамічного веб-середовища.

Одним із найпростіших підходів є reactive agents, які реагують на вхідні події без побудови складного внутрішнього плану дій. Такі системи характеризуються високою швидкістю роботи та низьким споживанням ресурсів, однак мають обмежену автономність і недостатню здатність до довготривалого planning [30].

Більш складними є planning agents, які здатні формувати послідовність дій для досягнення поставленої мети. Подібні системи підтримують task decomposition та multi-step execution, проте потребують значно більших обчислювальних ресурсів і складніших reasoning-механізмів. Суттєвого поширення останнім часом набув підхід ReAct (Reasoning + Acting), який поєднує логічне reasoning із безпосереднім виконанням дій у середовищі. Подібна архітектура дозволяє AI-агенту адаптивно змінювати поведінку залежно від результатів попередніх кроків та ефективно працювати з tool calling.

Для задач автоматизації веб-середовища особливу роль відіграють tool-augmented agents, які використовують зовнішні інструменти, browser automation, API-сервіси та retrieval-механізми [31]. Саме такий підхід дозволяє поєднувати можливості LLM із реальними діями у веб-середовищі.

Порівняння побудови AI-агентів представлено на рисунку 2.2.

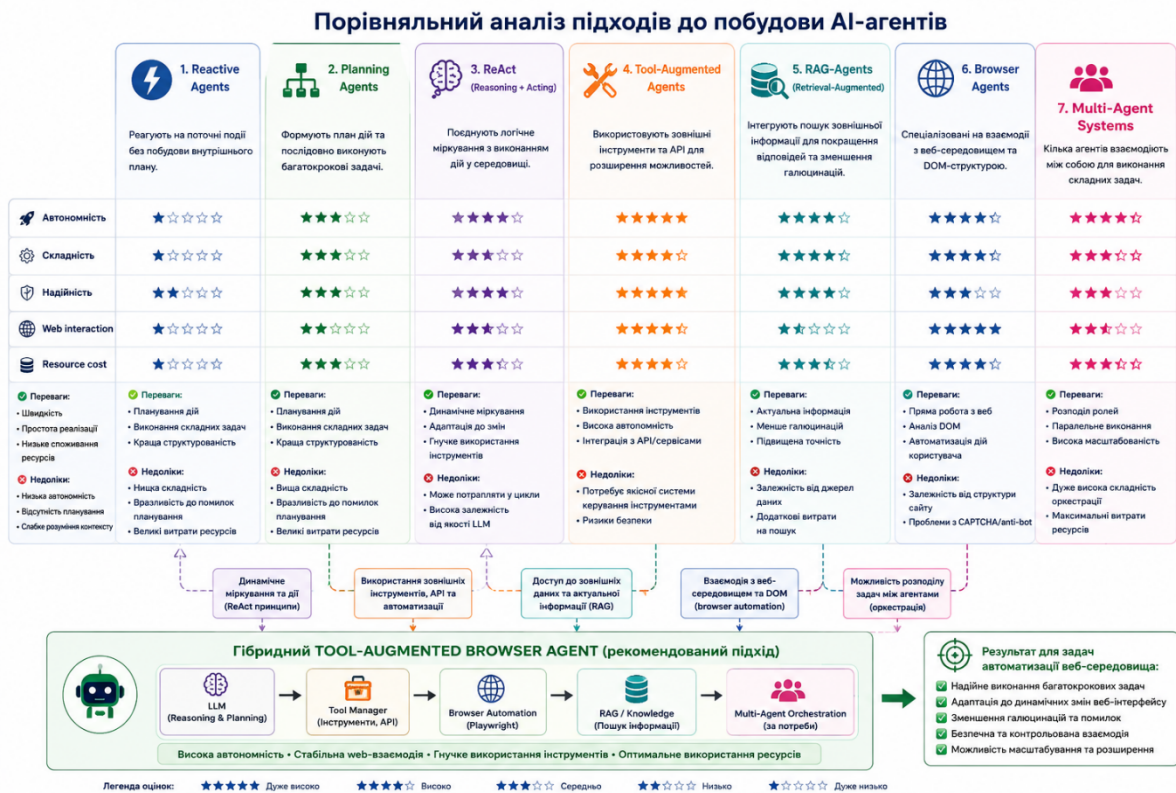


Рисунок 2.2 – Порівняльний аналіз підходів до побудови AI-агентів

Окрему категорію формують RAG-agents, які інтегрують Retrieval-Augmented Generation для роботи із зовнішніми джерелами інформації. Це дозволяє зменшити hallucination rate та покращити якість роботи агента з великими обсягами даних.

Водночас browser agents орієнтовані безпосередньо на взаємодію з веб-інтерфейсами та DOM-структурою сторінок, а multi-agent systems забезпечують розподіл ролей між декількома агентами для виконання складних workflow.

Проведений аналіз показав, що найбільш доцільним для задач автоматизації веб-середовища є гібридний tool-augmented підхід, який поєднує можливості великих мовних моделей, browser automation, reasoning-механізмів та систем керування інструментами. Подібна архітектура забезпечує високий рівень автономності, гнучкість взаємодії з веб-середовищем та підтримку складних багатокрокових сценаріїв виконання задач.

2.4 Обґрунтування архітектури власного AI-агента

На основі проведеного аналізу сучасних AI-агентів, фреймворків, інструментів браузерної автоматизації та мовних моделей було визначено, що найбільш доцільним для автоматичного виконання задач у веб-середовищі є гібридний tool-augmented підхід. Така архітектура передбачає поєднання великої мовної моделі, механізмів планування, системи виклику інструментів, браузерної автоматизації та шару пам'яті. Саме поєднання цих компонентів дозволяє забезпечити достатній рівень автономності, керованості та адаптивності системи.

Основою запропонованої архітектури є велика мовна модель, яка виконує роль інтелектуального ядра AI-агента. Вона відповідає за аналіз запиту користувача, інтерпретацію поточного стану задачі, вибір необхідних дій та формування проміжних рішень. Водночас мовна модель

не повинна діяти ізольовано, оскільки сама по собі вона не має прямого доступу до веб-середовища. Саме тому доцільним є використання tool-calling механізму, який дозволяє моделі не лише генерувати текстові відповіді, а й керовано викликати зовнішні інструменти.

Використання tool-calling є важливим елементом архітектури, оскільки воно забезпечує контрольовану взаємодію AI-агента з браузером, файлами, API або іншими програмними компонентами [32]. На відміну від неконтрольованого виконання команд, tool-calling дозволяє чітко визначити набір доступних агенту дій, їх параметри та очікуваний результат. Це підвищує безпечність системи, зменшує ризик виконання небажаних операцій та спрощує логування дій агента.

Для організації логіки виконання задач у запропонованій архітектурі доцільно використовувати підхід planner/executor [33]. Planner відповідає за декомпозицію загальної задачі на послідовність менших кроків, визначення порядку їх виконання та вибір необхідних інструментів. Executor, у свою чергу, виконує конкретні дії у веб-середовищі, обробляє результати та повертає їх до системи для подальшого аналізу. Такий розподіл дозволяє відокремити етап прийняття рішень від етапу фактичного виконання дій, що підвищує прозорість та керованість роботи AI-агента.

Для реалізації браузерної взаємодії доцільним є використання Playwright, оскільки він забезпечує стабільну роботу із сучасними веб-інтерфейсами, підтримує Chromium, Firefox та WebKit, має вбудовані механізми очікування елементів і добре працює з динамічними SPA-додатками. На відміну від Selenium, Playwright краще адаптований до сучасних JavaScript-інтерфейсів та async loading, а порівняно з Puppeteer має ширшу підтримку браузерів. Це робить його збалансованим рішенням для AI-агента, який повинен виконувати дії у реальному веб-середовищі.

Окремим компонентом архітектури є memory layer, який використовується для збереження контексту виконання задачі, проміжних результатів, історії дій та важливої інформації, отриманої під час взаємодії

з веб-ресурсами [34]. Наявність шару пам'яті дозволяє агенту не втрачати контекст під час багатокрокових сценаріїв, уникати повторного виконання однакових дій та коригувати подальшу поведінку на основі попередніх результатів. Це особливо важливо для задач, які потребують переходу між кількома сторінками, аналізу різних джерел або поступового уточнення інформації.

Важливою особливістю запропонованої архітектури є асинхронна модель виконання. Веб-середовище характеризується затримками завантаження сторінок, асинхронним рендерингом контенту, очікуванням відповіді від API та непередбачуваною поведінкою окремих елементів інтерфейсу. Використання *async architecture* дозволяє ефективніше обробляти такі процеси, не блокуючи виконання всієї системи. Це підвищує швидкодію агента та робить його більш придатним для роботи зі складними веб-сценаріями.

Модульний дизайн є ще одним принциповим рішенням у побудові AI-агента. Запропонована система повинна складатися з окремих компонентів: модуля взаємодії з LLM, *planner*-модуля, *executor*-модуля, *memory layer*, *browser automation layer* та модуля контролю безпеки [35]. Такий підхід спрощує розробку, тестування та подальше розширення системи. У разі потреби окремі компоненти можуть бути замінені або вдосконалені без повної перебудови архітектури.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Загальна архітектура програмної системи

Система автономного AI-агента для автоматичного виконання завдань у веб-середовищі реалізована у вигляді багатомодульного вебзастосунку, який поєднує інструменти штучного інтелекту, механізми браузерної автоматизації та засоби обробки природної мови. Основною метою системи є забезпечення автономного виконання веб-завдань без необхідності постійного втручання користувача.

Розроблена система дозволяє приймати текстовий запит користувача, аналізувати його зміст, формувати послідовність дій, взаємодіяти з веб-ресурсами, отримувати необхідну інформацію, оцінювати коректність результатів та формувати фінальну відповідь. На відміну від класичних систем автоматизації, запропонований AI-агент не працює за жорстко визначеним сценарієм, а динамічно адаптує поведінку відповідно до контексту задачі та проміжних результатів виконання.

Архітектура системи побудована за модульним принципом, що забезпечує масштабованість, повторне використання компонентів та можливість інтеграції нових AI-модулів без зміни базової логіки системи.

Інтерфейс системи містить основні елементи взаємодії з AI-агентом, зокрема панель стану системи, модуль пам'яті агента, історію виконання попередніх задач та візуалізацію агентного циклу. У центральній частині інтерфейсу реалізовано відображення етапів роботи AI-агента, що дозволяє користувачу спостерігати процес прийняття рішень та виконання задач у режимі реального часу (рисунок 3.1). Архітектура системи складається з декількох взаємопов'язаних модулів, кожен із яких відповідає за окремий етап функціонування AI-агента.

Frontend-частина забезпечує взаємодію користувача із системою. Вона реалізує введення задачі, запуск AI-агента, візуалізацію процесу

виконання, відображення стану системи, таймлайну виконання та фінальної відповіді.

Backend-рівень відповідає за координацію роботи внутрішніх компонентів системи, обробку запитів та запуск агентного циклу виконання.

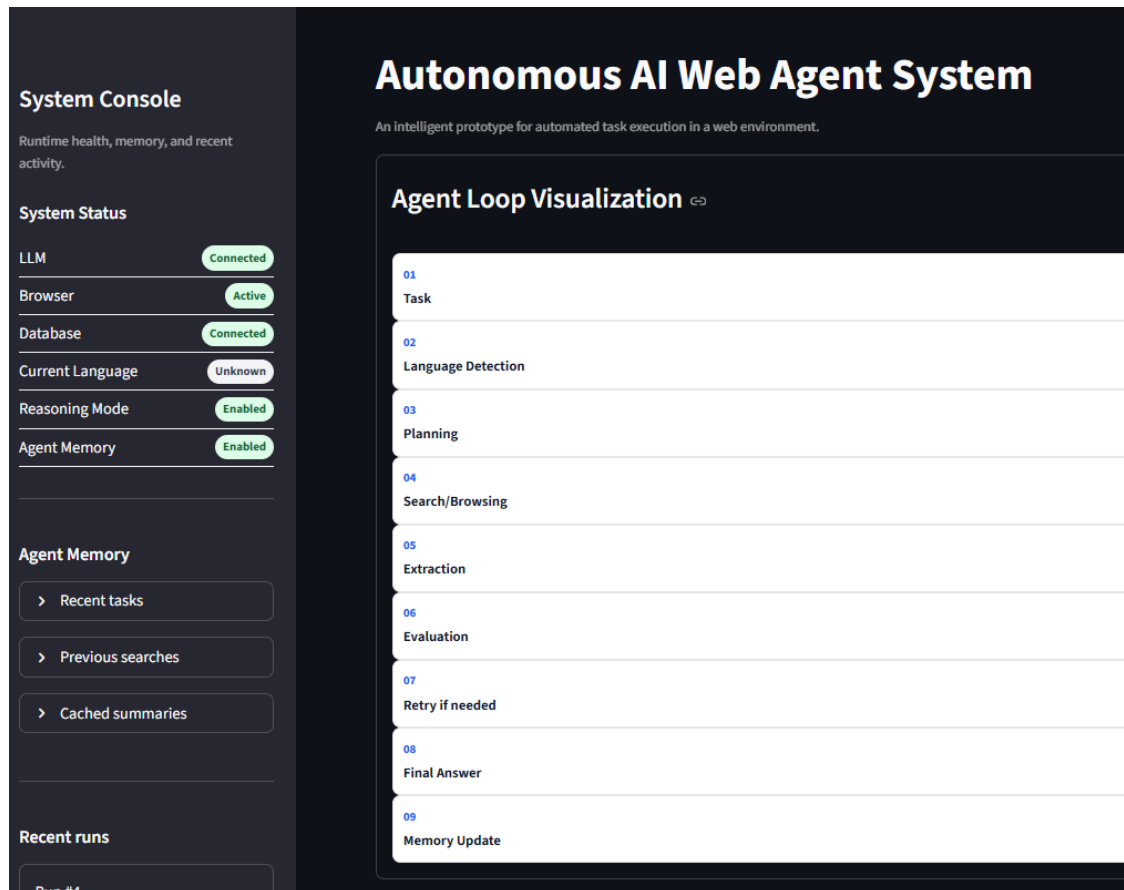


Рисунок 3.1 – Головний інтерфейс системи автономного AI-агента

Ключовим елементом системи є Agent Core, який реалізує логіку автономного функціонування AI-агента. Саме цей модуль координує процес аналізу задачі, планування дій, використання інструментів та формування відповіді.

Модуль Planner відповідає за декомпозицію складних задач на окремі підзадачі та формування послідовності виконання дій.

LLM-модуль забезпечує обробку природної мови, визначення намірів користувача, генерацію проміжних міркувань та формування фінальної

відповіді. Завдяки використанню великих мовних моделей система здатна адаптуватися до різних типів задач та працювати з багатомовними запитами.

Memory Layer реалізує механізми контекстної пам'яті та збереження попередніх результатів роботи агента. Це дозволяє системі повторно використовувати отриману інформацію та уникати дублювання дій.

Модуль Browser Automation забезпечує автономну взаємодію із веб-середовищем, включаючи відкриття сторінок, навігацію між ресурсами, пошук інформації та витягування контенту.

Execution Loop координує послідовність виконання агентних дій та підтримує циклічний процес аналізу, виконання та оцінки результатів.

Evaluation & Retry Module реалізує механізми перевірки отриманих результатів та повторного виконання окремих етапів у разі виявлення помилок або недостатньої якості відповіді.

Загальну архітектуру наведено на рисунку 3.2.

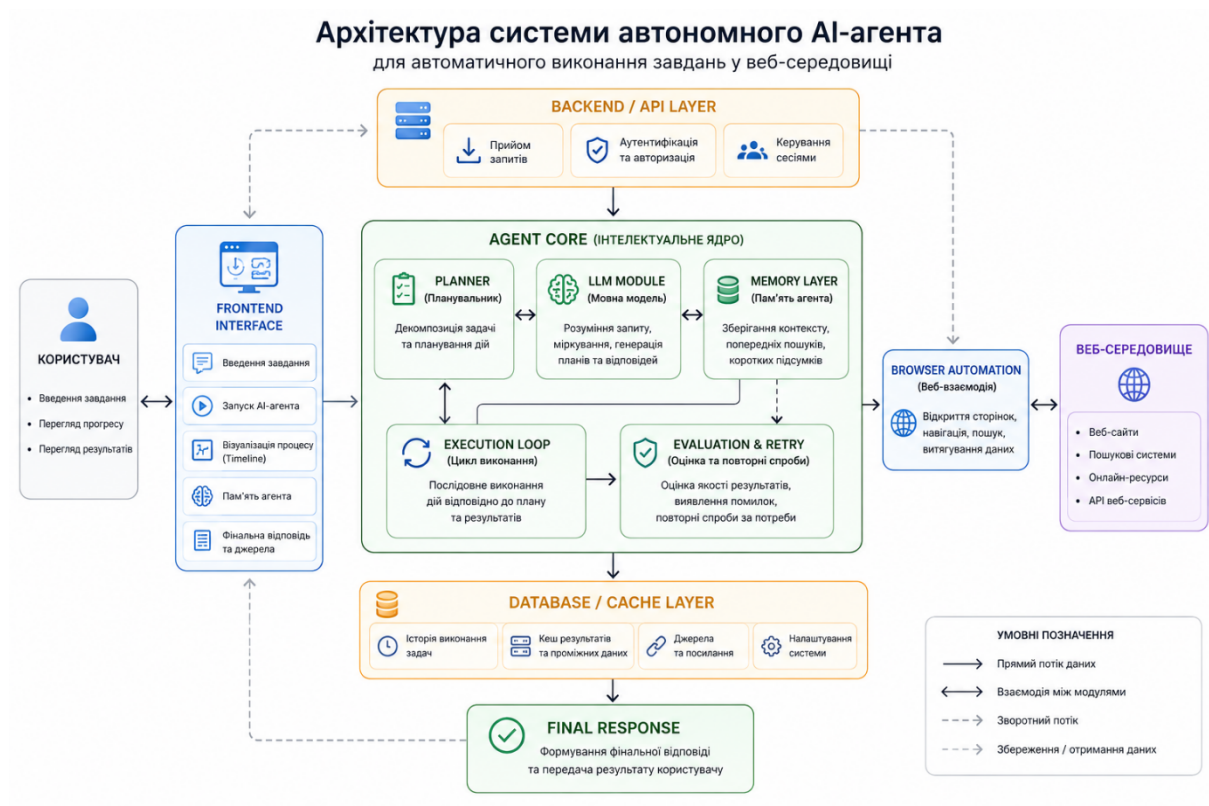


Рисунок 3.2 – Загальна архітектура програмної системи AI-агента

Database та Cache використовуються для збереження історії виконання задач, проміжних результатів, джерел інформації та службових даних системи.

Інтелектуальність системи досягається завдяки поєднанню великих мовних моделей, механізмів планування, контекстної пам'яті та адаптивного agent loop-підходу. На відміну від звичайних скриптів автоматизації, AI-агент не виконує наперед задану жорстку послідовність команд, а динамічно формує стратегію поведінки залежно від змісту запиту користувача, проміжних результатів та змін веб-середовища. Це дозволяє системі автономно приймати рішення, коригувати власні дії та виконувати складні веб-завдання в умовах невизначеності.

3.2 Реалізація інтелектуального агентного циклу

Ключовим компонентом реалізованої системи є інтелектуальний агентний цикл (Agent Loop), який забезпечує автономне виконання веб-завдань шляхом послідовного аналізу задачі, планування дій, взаємодії із веб-середовищем та оцінювання отриманих результатів.

На відміну від традиційних систем автоматизації, де послідовність дій визначається наперед, запропонований AI-агент використовує динамічний цикл прийняття рішень. Це дозволяє системі адаптувати поведінку залежно від контексту задачі, змін веб-середовища та проміжних результатів виконання.

Інтелектуальність агента полягає не лише у генерації тексту, а у здатності приймати проміжні рішення, аналізувати результати виконання дій та адаптувати подальшу поведінку відповідно до контексту задачі.

Реалізований агентний цикл складається з декількох взаємопов'язаних етапів, кожен із яких виконує окрему функцію в процесі автономного виконання задачі (лістинг 3.1).

Лістинг 3.1 – Спрощена реалізація agent loop

```
while not task_completed:
    plan = planner.generate(task, memory)
    action = executor.next_action(plan)
    result = browser.execute(action)
    evaluation = evaluator.check(result)
    memory.update(result)
    if evaluation.retry_required:
        continue
    task_completed = True
```

На першому етапі система отримує текстовий запит користувача та ініціалізує процес виконання задачі. Після цього здійснюється аналіз запиту та визначення мови введення, що дозволяє системі підтримувати багатомовну взаємодію.

Далі модуль планування формує послідовність необхідних дій для досягнення поставленої мети. На цьому етапі задача декомпонується на окремі підзадачі, визначаються необхідні інструменти та стратегія взаємодії із веб-середовищем.

Після формування плану агент переходить до етапу *browser interaction*, у межах якого система виконує пошук інформації, відкриває веб-сторінки, аналізує контент та отримує необхідні дані.

Етап *extraction* відповідає за виділення корисної інформації із отриманих джерел, очищення контенту та формування структурованого представлення результатів.

На етапі *evaluation* система оцінює якість отриманих результатів, перевіряє релевантність знайденої інформації та визначає необхідність повторного виконання окремих кроків.

У разі недостатньої якості результатів активується *retry strategy*, яка дозволяє агенту змінювати стратегію пошуку, повторно виконувати окремі дії або уточнювати проміжні результати.

Після завершення циклу формується фінальна відповідь користувачу, а також виконується оновлення пам'яті агента для подальшого використання контексту в майбутніх задачах.

У процесі виконання задачі система відображає поточний стан агентного циклу та проміжні етапи міркування. Це дозволяє реалізувати механізми прозорості взаємодії з користувачем та візуалізувати процес автономного прийняття рішень.

AI-агент послідовно аналізує контекст, перевіряє пам'ять, планує дії, використовує браузерні інструменти та оцінює результати виконання (рисунок 3.3).

Додатково система відображає використані джерела інформації та окремі етапи reasoning-процесу, що забезпечує прозорість роботи AI-агента та дозволяє оцінити коректність виконання задачі (рисунок 3.4).

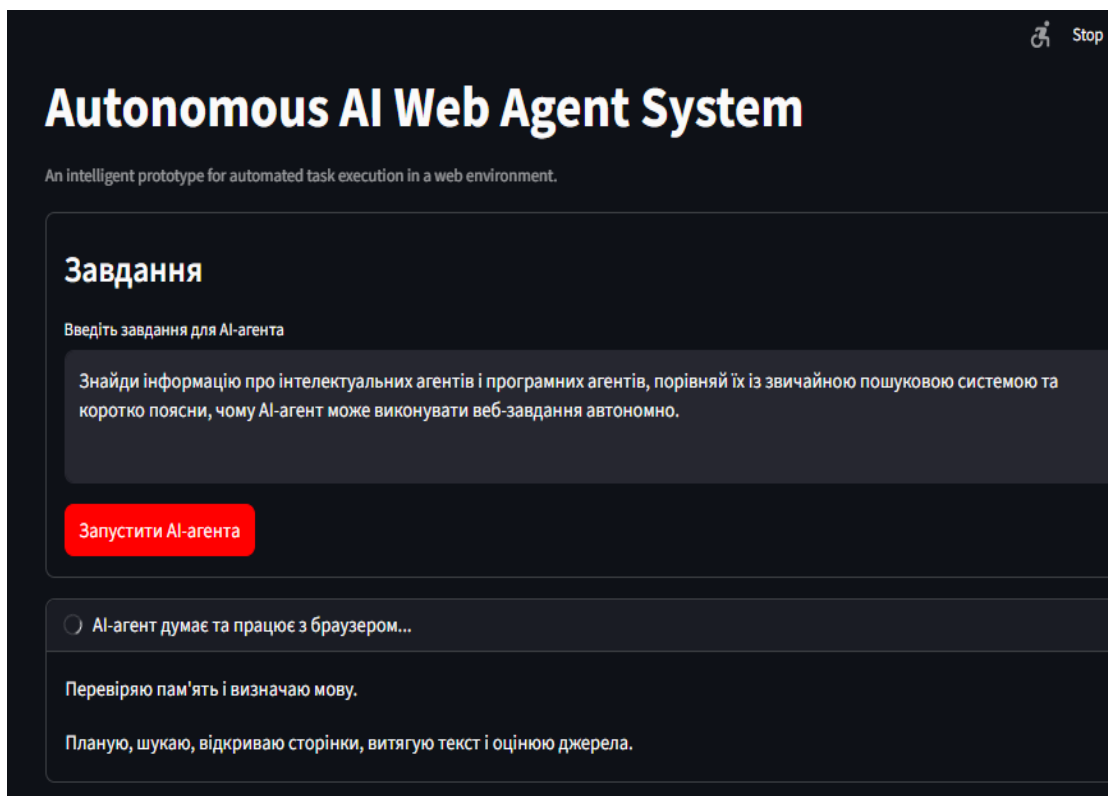


Рисунок 3.3 – Процес виконання агентного циклу під час обробки запиту

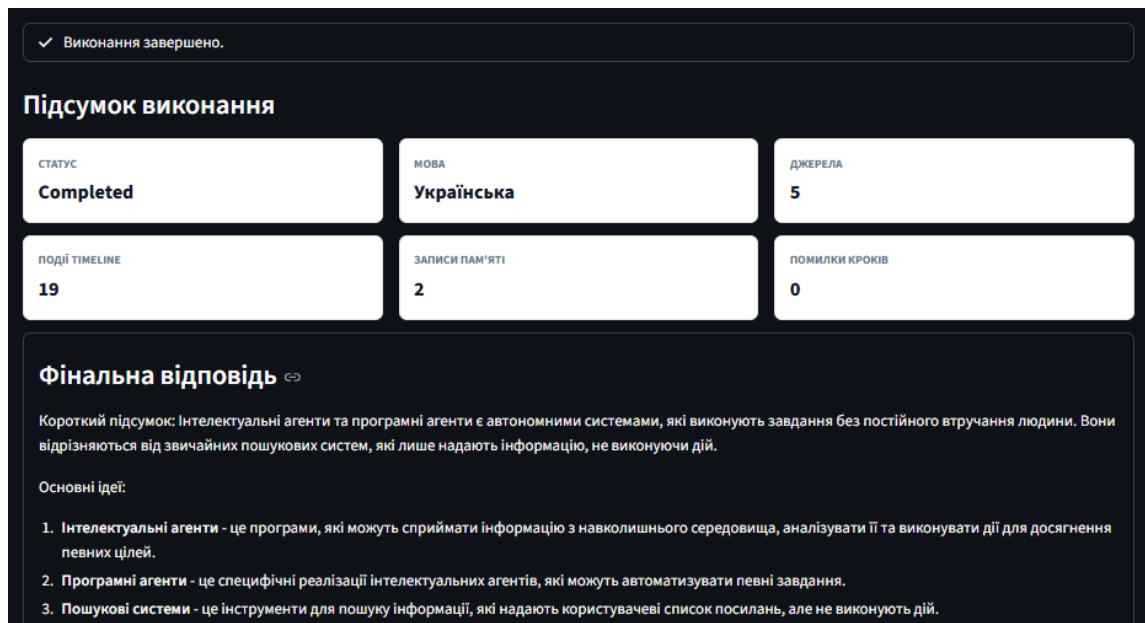


Рисунок 3.4 – Результати виконання задачі AI-агентом

Після завершення виконання задачі система формує структуровану фінальну відповідь, яка містить короткий підсумок отриманих результатів, основні висновки, використані джерела інформації та елементи reasoning-процесу AI-агента.

На рисунку 3.5 представлено приклад формування фінальної відповіді системою після завершення автономного виконання веб-завдання.

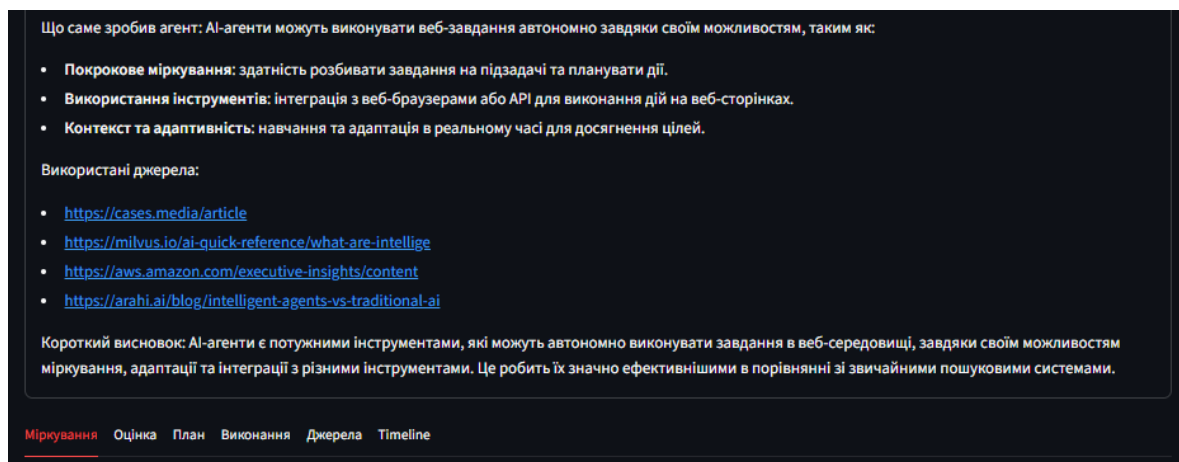


Рисунок 3.5 – Формування фінальної відповіді та відображення reasoning-процесу AI-агента

У нижній частині інтерфейсу реалізовано окремі вкладки для візуалізації внутрішніх етапів роботи агента, зокрема міркування, оцінки результатів, планування дій, виконання та часової послідовності операцій (timeline). Це забезпечує прозорість роботи системи та дозволяє користувачу аналізувати логіку прийняття рішень AI-агентом.

Реалізований agent loop забезпечує автономне функціонування AI-агента в умовах динамічного веб-середовища. Поєднання механізмів планування, браузерної взаємодії, оцінювання результатів та контекстної пам'яті дозволяє системі не лише виконувати окремі дії, а й адаптувати власну поведінку залежно від перебігу виконання задачі. На відміну від традиційних алгоритмів автоматизації, реалізований AI-агент використовує елементи інтелектуального прийняття рішень, що наближає систему до сучасних концепцій автономних агентів та tool-augmented AI system

3.3 Механізми забезпечення інтелектуальності системи

3.3.1 Використання великих мовних моделей

Одним із ключових компонентів реалізованої системи є модуль великих мовних моделей, який забезпечує аналіз природної мови, інтерпретацію задач користувача, формування проміжних міркувань та генерацію фінальної відповіді.

Використання LLM дозволяє системі працювати із неструктурованими текстовими запитами, визначати наміри користувача та формувати адаптивну стратегію виконання задачі. На відміну від традиційних сценаріїв автоматизації, де послідовність дій задається наперед, реалізований AI-агент динамічно формує логіку поведінки відповідно до контексту задачі та проміжних результатів.

Однією з важливих функцій LLM-модуля є реалізація reasoning-процесу, який дозволяє агенту аналізувати поточний стан виконання задачі,

визначати необхідність додаткового пошуку інформації, змінювати стратегію взаємодії із веб-середовищем та формувати пояснення власних дій.

На рисунку 3.6 представлено приклад reasoning-процесу AI-агента під час виконання веб-завдання. Система послідовно аналізує задачу, визначає мову запиту, формує пошукову стратегію, оцінює достатність отриманих результатів та за необхідності виконує додаткові пошукові дії.

Особливістю реалізованого підходу є здатність AI-агента адаптувати поведінку відповідно до проміжних результатів виконання. У випадку недостатньої кількості або низької якості знайденої інформації система автоматично ініціює повторний пошук або використовує альтернативні джерела даних.

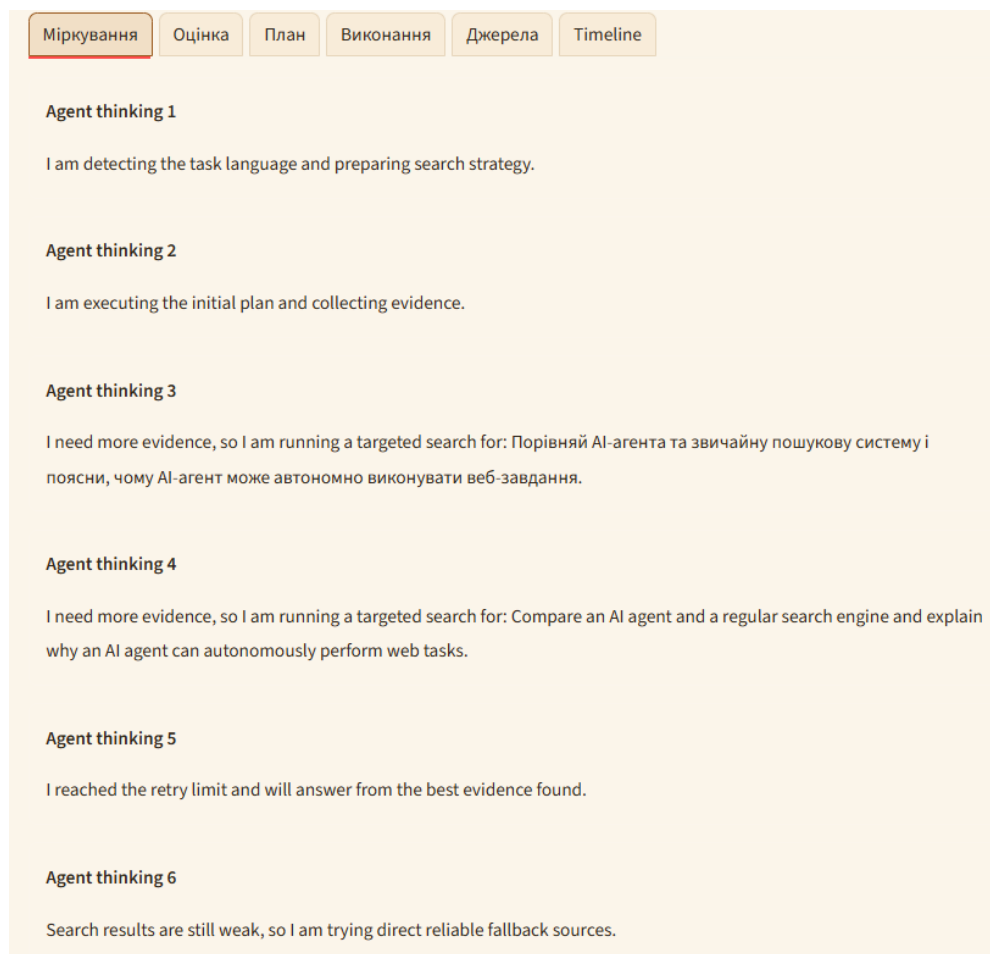


Рисунок 3.6 – Візуалізація reasoning-процесу AI-агента

Використання великих мовних моделей у структурі AI-агента забезпечує не лише генерацію тексту, а й реалізацію механізмів інтерпретації задач, контекстного аналізу, reasoning-процесу та адаптивного формування стратегії виконання веб-завдань.

3.3.2 Планування та декомпозиція задач

Однією з ключових характеристик інтелектуального AI-агента є здатність до планування та декомпозиції задач. У реалізованій системі цей механізм використовується для автоматичного формування послідовності дій, необхідних для виконання поставленої користувачем задачі.

Після отримання запиту система аналізує зміст задачі та формує структурований план виконання. На цьому етапі визначаються основна ціль, очікуваний результат, кількість необхідних кроків та стратегія взаємодії із веб-середовищем.

На рисунку 3.7 представлено приклад автоматичного формування плану виконання задачі AI-агентом. Система визначає тип задачі, мову запиту, кількість необхідних кроків та очікуваний результат виконання.

The screenshot displays a user interface for an AI agent's planning process. At the top, there is a navigation bar with tabs: 'Міркування' (Thought), 'Оцінка' (Evaluation), 'План' (Plan), 'Виконання' (Execution), 'Джерела' (Sources), and 'Timeline'. The 'План' tab is currently selected. Below the navigation bar, there are four input fields: 'Task type' with the value 'web_search', 'Language' with the value 'Українська' (Ukrainian), 'Steps' with the value '2', and 'Search queries' with the value '3'. The main area of the interface is divided into two sections. The first section, titled 'Goal', contains the text: 'Порівняти AI-агента та звичайну пошукову систему, пояснити автономність AI-агента у виконанні веб-завдань.' (Compare AI agent and standard search system, explain AI agent's autonomy in performing web tasks). The second section, titled 'Expected output', contains the text: 'Текст, що містить порівняння AI-агента та звичайної пошукової системи, а також пояснення автономності AI-агента.' (Text that contains comparison of AI agent and standard search system, as well as explanation of AI agent's autonomy).

Рисунок 3.7 – Формування плану виконання задачі AI-агентом

На відміну від звичайних пошукових систем або жорстко запрограмованих сценаріїв автоматизації, AI-агент здатний динамічно

змінювати послідовність виконання дій залежно від проміжних результатів та доступності інформації.

Особливістю реалізованого підходу є використання sequential reasoning-механізму, у межах якого кожен наступний крок виконання залежить від результатів попереднього етапу. Це дозволяє системі адаптувати план виконання відповідно до змін веб-середовища та проміжних результатів пошуку.

Декомпозиція задачі на окремі підзадачі дозволяє зменшити складність виконання веб-завдань та підвищити якість фінальних результатів. Завдяки цьому AI-агент здатний працювати із складними багатокроковими сценаріями, які потребують послідовного аналізу інформації та адаптивного прийняття рішень.

3.3.3 Memory Layer та контекстна пам'ять

Важливим компонентом реалізованої системи є модуль контекстної пам'яті (Memory Layer), який забезпечує збереження проміжних результатів роботи AI-агента, історії виконання задач та раніше отриманої інформації.

Використання механізмів пам'яті дозволяє системі підтримувати контекст взаємодії, повторно використовувати результати попередніх пошуків та уникати дублювання окремих дій під час виконання веб-завдань.

На відміну від класичних stateless-систем, де кожен новий запит обробляється незалежно від попередніх взаємодій, реалізований AI-агент використовує контекстну пам'ять для підвищення ефективності виконання задач та адаптації стратегії пошуку інформації.

На рисунку 3.8 представлено інтерфейс модуля контекстної пам'яті AI-агента. Система зберігає інформацію про попередньо виконані задачі, історію пошукових запитів та короткі підсумки отриманих результатів.

Механізм пам'яті використовується для підтримки безперервності виконання задач та забезпечення повторного використання вже отриманої

інформації. Це дозволяє агенту скорочувати кількість повторних звернень до веб-ресурсів та оптимізувати процес виконання багатокрокових задач.

Додатково Memory Layer використовується для формування контексту reasoning-процесу та адаптації поведінки агента залежно від історії попередніх взаємодій.

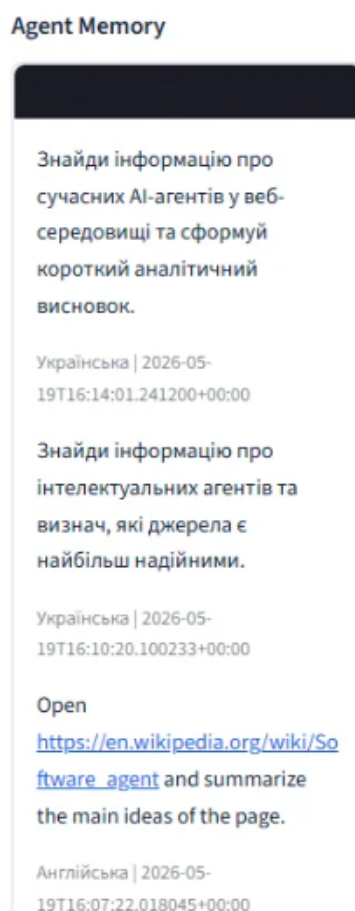


Рисунок 3.8 – Використання контекстної пам'яті AI-агента

Контекстна пам'ять є одним із ключових елементів сучасних AI-agent systems, оскільки дозволяє реалізувати stateful-поведінку системи та підтримувати довготривалий контекст взаємодії із користувачем.

Завдяки інтеграції Memory Layer AI-агент здатний аналізувати попередні результати виконання задач, враховувати історію пошуку та використовувати накопичений контекст для формування більш релевантних відповідей.

3.3.4 Самооцінка та retry-механізми

Однією з ключових характеристик інтелектуальних AI-агентів є здатність до самооцінки результатів власної роботи та адаптивного коригування поведінки. У реалізованій системі для цього використовується модуль *evaluation* та *retry*-механізмів, який забезпечує аналіз якості отриманої інформації та прийняття рішень щодо необхідності повторного виконання окремих дій.

Після завершення основного етапу пошуку та збору даних система виконує оцінювання достатності отриманої інформації, аналізує кількість релевантних джерел, рівень достовірності результатів та загальну якість сформованої відповіді.

На відміну від традиційних пошукових систем або статичних сценаріїв автоматизації, AI-агент здатний самостійно визначати недостатність або низьку якість отриманих результатів та ініціювати додаткові пошукові дії.

На рисунку 3.9 представлено приклад роботи механізму *self-evaluation* у структурі AI-агента. Система виконує оцінювання достатності отриманої інформації (*Sufficiency score*), аналізує кількість корисних джерел, доменів та середній рівень достовірності отриманих результатів.

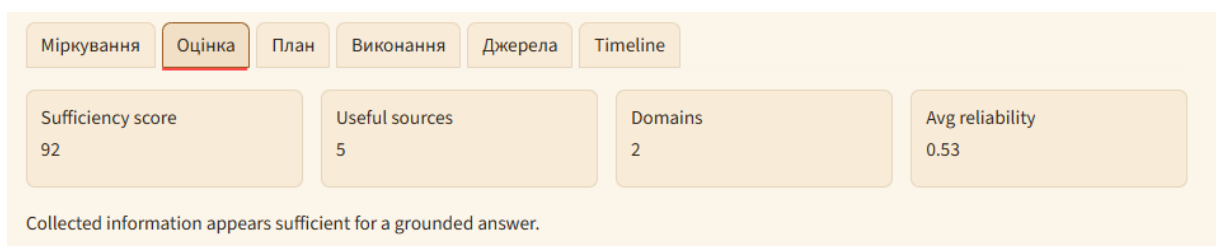


Рисунок 3.9 – Механізм оцінювання результатів та *self-evaluation* AI-агента

Результати оцінювання використовуються для визначення необхідності повторного виконання окремих етапів агентного циклу. У

випадку недостатньої кількості або низької якості даних система автоматично активує `retry`-механізм та коригує стратегію пошуку інформації.

Особливістю реалізованої системи є використання механізмів `confidence estimation` та `groundedness analysis`, які дозволяють оцінювати ступінь обґрунтованості сформованої відповіді на основі якості знайдених джерел інформації.

3.4 Реалізація веб-взаємодії та автоматизації браузера

Одним із ключових компонентів реалізованої системи є модуль веб-взаємодії та браузерної автоматизації, який забезпечує автономне виконання дій у веб-середовищі. У межах роботи AI-агента браузерний модуль використовується як інструмент взаємодії із зовнішніми веб-ресурсами та виконує функції навігації, пошуку інформації, аналізу контенту та витягування даних.

Для реалізації браузерної автоматизації використовується сучасний інструментарій `Playwright`, який забезпечує підтримку динамічного контенту, асинхронної взаємодії із веб-сторінками та автоматизацію дій користувача у браузері (лістинг 3.2).

Лістинг 3.2 – Приклад браузерної взаємодії із веб-ресурсом

```
async def extract_page_content(page, url):  
    await page.goto(url)  
    content = await page.locator("body").inner_text()  
    return content
```

На відміну від традиційних `Selenium-like` сценаріїв автоматизації, браузерний модуль у структурі реалізованої системи не працює автономно, а використовується як інструмент AI-агента в межах `reasoning`-процесу та `agent loop`-підходу.

На рисунку 3.10 представлено процес виконання браузерних дій AI-агентом під час автономного виконання веб-завдання. Система послідовно виконує відкриття веб-сторінок, навігацію між ресурсами, аналіз DOM-структури сторінок та витягування необхідної інформації.

Модуль браузерної автоматизації інтегрований у загальний agent loop та виконує дії відповідно до сформованого плану виконання задачі. У процесі роботи AI-агент може змінювати стратегію навігації, повторно виконувати окремі дії та використовувати альтернативні джерела інформації залежно від проміжних результатів.

На рисунку 3.11 представлено процес аналізу веб-джерел та витягування інформації AI-агентом. Система автоматично визначає релевантні веб-ресурси, аналізує структуру сторінок та виконує extraction корисного контенту.

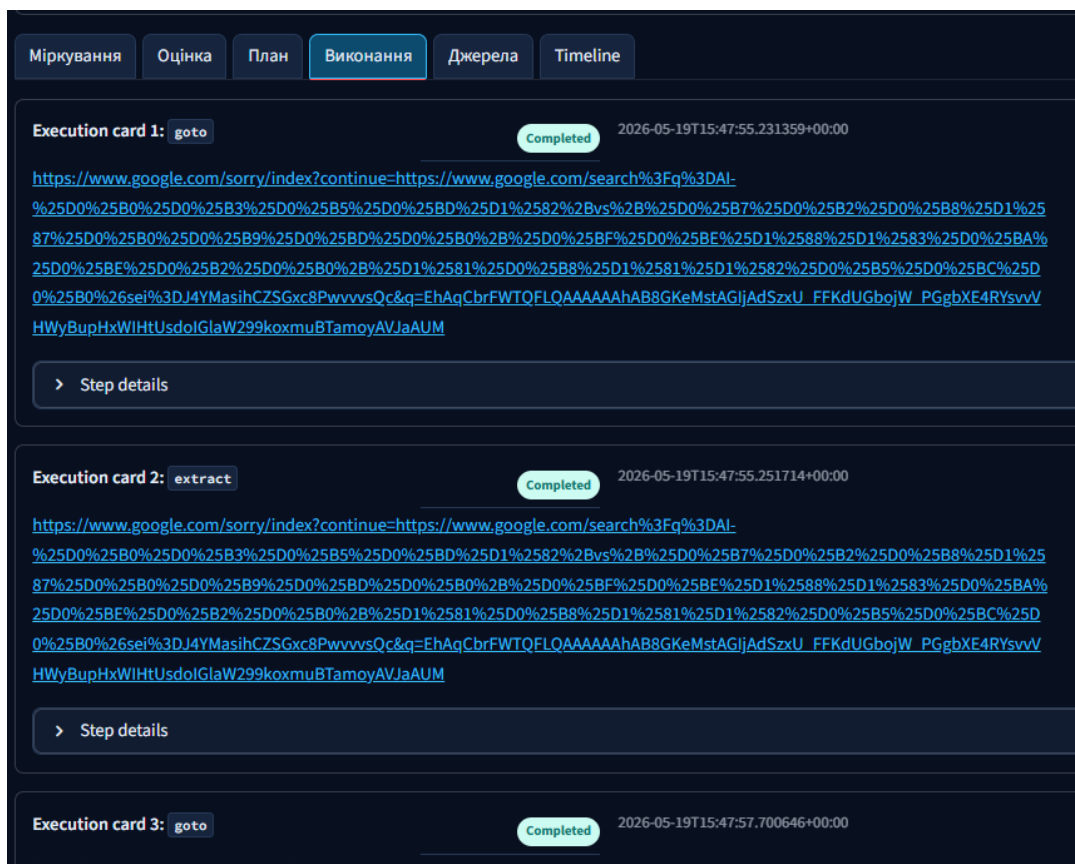


Рисунок 3.10 – Виконання браузерних дій AI-агентом

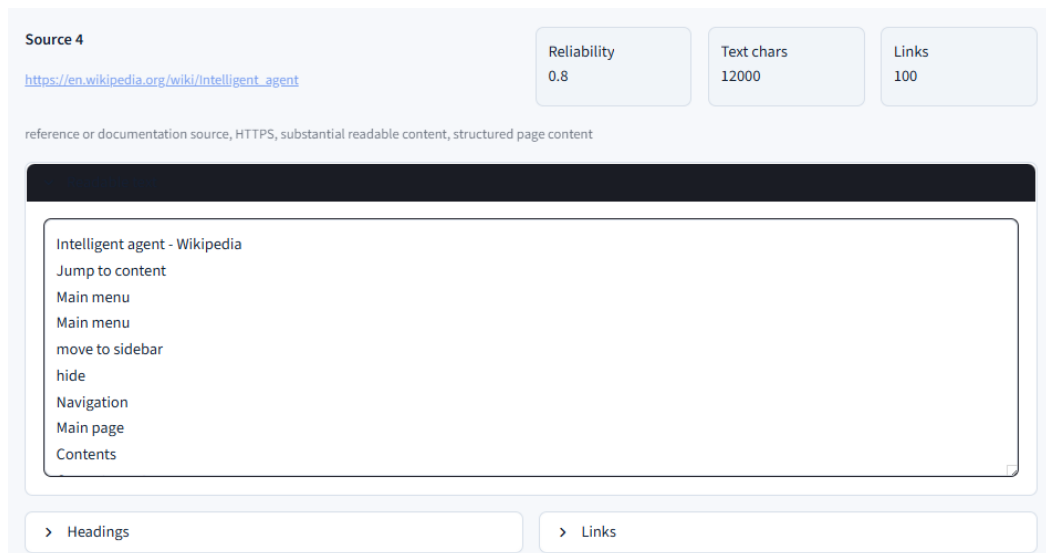


Рисунок 3.11 – Аналіз та витягування інформації із веб-джерел

Після отримання інформації виконується очищення тексту, аналіз змісту та формування структурованого представлення результатів для подальшого використання в reasoning-процесі та генерації фінальної відповіді.

Для забезпечення стабільної роботи із сучасними веб-ресурсами система підтримує асинхронну взаємодію із веб-сторінками, обробку динамічного контенту та автоматичне очікування завантаження елементів інтерфейсу.

Модуль браузерної автоматизації забезпечує автономну взаємодію AI-агента із веб-середовищем та реалізує механізми навігації, аналізу сторінок і витягування інформації. Інтеграція браузерного модуля із reasoning-oriented agent loop дозволяє системі адаптивно виконувати складні веб-завдання в умовах динамічного середовища.

3.5 Реалізація користувацького інтерфейсу системи

Під час розробки AI-агента окрема увага приділялася реалізації користувацького інтерфейсу, який забезпечує не лише взаємодію із

Для підвищення прозорості роботи AI-агента в системі реалізовано механізми візуалізації reasoning-процесу та execution flow. Користувач має можливість спостерігати не лише фінальний результат виконання задачі, а й окремі етапи прийняття рішень агентом.

У межах інтерфейсу реалізовано окремі компоненти для відображення статусу виконання задачі, прогресу agent loop, проміжних reasoning-кроків та фінальної відповіді системи. Додатково користувачу доступна timeline-візуалізація, яка демонструє часову послідовність виконання браузерних дій та етапів обробки інформації.

Візуалізація процесу мислення агента є важливим елементом сучасних AI-систем, оскільки дозволяє користувачу аналізувати логіку прийняття рішень та оцінювати коректність виконання задачі.

3.5.2 Підтримка різних тем інтерфейсу

Під час проєктування користувацького інтерфейсу враховувалася необхідність забезпечення комфортної взаємодії із системою в різних умовах використання. З цією метою в системі реалізовано підтримку декількох кольорових тем інтерфейсу.

Користувачу доступні світлий режим, темна тема та режим підвищеної читабельності (reading mode), оптимізований для тривалої роботи із текстовою інформацією та reasoning-візуалізацією (рисунок 3.13).

Реалізація декількох режимів відображення інтерфейсу дозволяє адаптувати систему до індивідуальних уподобань користувача та підвищити зручність взаємодії із AI-агентом.

Темна тема орієнтована на зменшення навантаження на зір під час тривалої роботи, світлий режим забезпечує класичний формат взаємодії із системою, а reading mode оптимізований для аналізу великих обсягів текстової інформації та reasoning-процесів агента.

Додатково реалізовано адаптивне відображення елементів інтерфейсу, що забезпечує коректну роботу dashboard-компонентів та execution-візуалізації незалежно від вибраної теми.

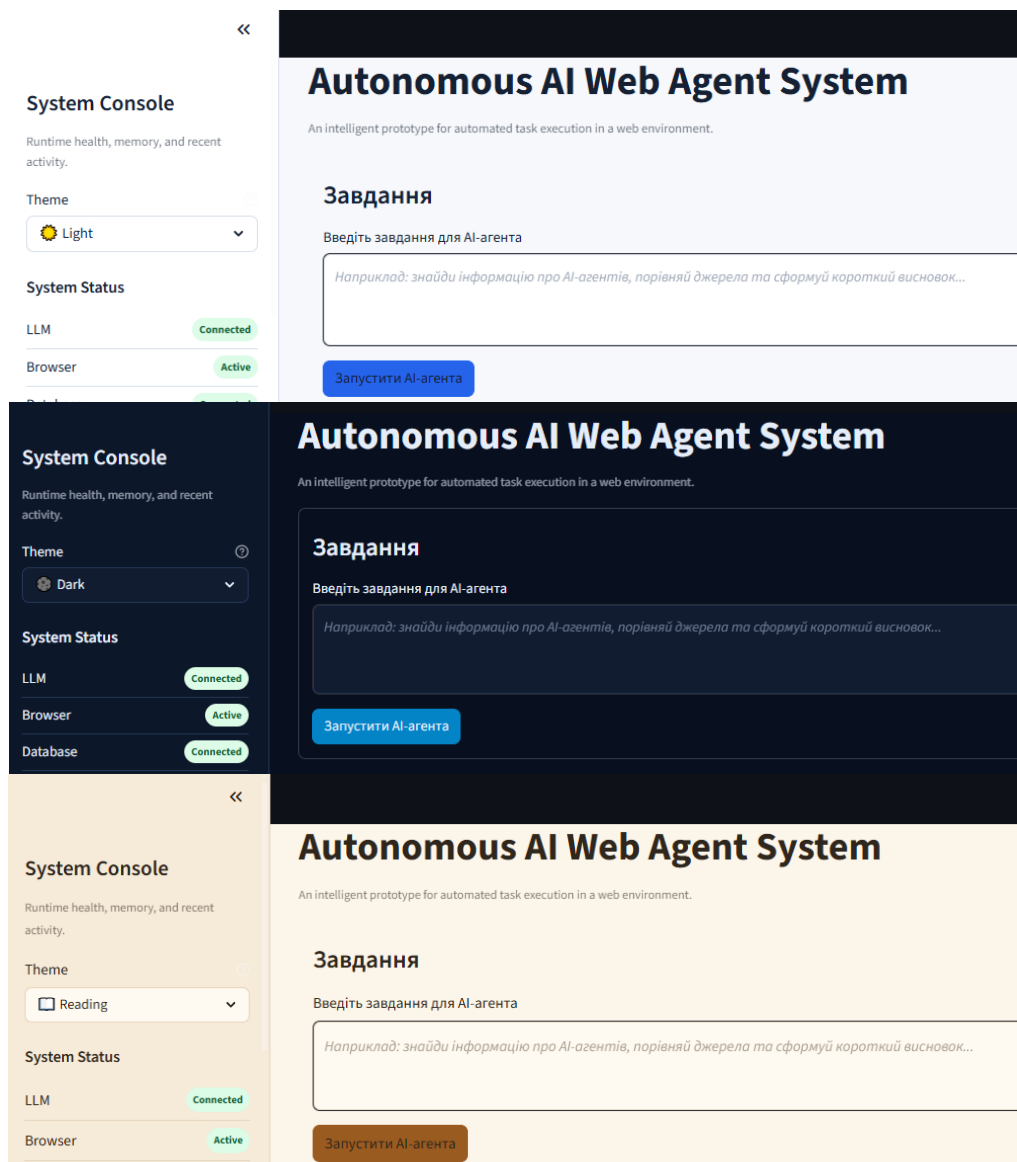


Рисунок 3.13 – Варіанти кольорових тем користувацького інтерфейсу

Реалізований користувацький інтерфейс забезпечує не лише взаємодію із системою, а й прозору візуалізацію процесу роботи AI-агента. Інтеграція reasoning-візуалізації, execution timeline та адаптивних елементів dashboard-середовища підвищує explainability системи та покращує користувацький досвід під час роботи із AI-агентом.

3.6 Аналіз результатів роботи системи

Для оцінювання ефективності реалізованого AI-агента було проведено серію тестових запусків системи із використанням різних типів веб-завдань. У процесі тестування оцінювалися здатність агента до автономного виконання задач, якість reasoning-процесу, коректність пошуку джерел інформації, адаптивність agent loop та якість сформованих відповідей.

На рисунку 3.14 представлено приклад виконання AI-агентом задачі порівняння інтелектуального агента та звичайної пошукової системи. Під час виконання задачі система автоматично визначила мову запиту, сформувала план пошуку, виконала аналіз веб-джерел та сформувала структуровану фінальну відповідь.

The screenshot displays the 'Autonomous AI Web Agent System' interface. At the top, the title 'Autonomous AI Web Agent System' is followed by the subtitle 'An intelligent prototype for automated task execution in a web environment.' Below this, the 'Завдання' (Task) section contains a text input field with the prompt: 'Порівняй AI-агента та звичайну пошукову систему і поясни, чому AI-агент може автономно виконувати веб-завдання.' A 'Запустити AI-агента' (Run AI Agent) button is positioned below the input. A status bar indicates '✓ Виконання завершено.' (Execution completed). The 'Підсумок виконання' (Execution Summary) section features a grid of six cards: 'СТАТУС' (Status) is 'Completed'; 'МОВА' (Language) is 'Українська'; 'ДЖЕРЕЛА' (Sources) is '5'; 'ПОДІЇ TIMELINE' (Events Timeline) is '20'; 'ЗАПИСИ ПАМ'ЯТІ' (Memory Records) is '3'; and 'ПОМИЛКИ КРОКІВ' (Step Errors) is '0'.

СТАТУС	МОВА	ДЖЕРЕЛА
Completed	Українська	5

ПОДІЇ TIMELINE	ЗАПИСИ ПАМ'ЯТІ	ПОМИЛКИ КРОКІВ
20	3	0

Рисунок 3.14 – Приклад виконання веб-завдання AI-агентом

Окрема увага приділялася аналізу багатокрокових сценаріїв, у межах яких AI-агент повинен був самостійно формувати стратегію пошуку, виконувати *browser interaction* та оцінювати достовірність отриманих результатів.

У процесі виконання задачі агент використав декілька зовнішніх джерел інформації та виконав багатокроковий *reasoning*-процес, що дозволило сформувати обґрунтований підсумок із використанням *source grounding*-механізмів.

Фінальна відповідь AI-агента формується на основі результатів *reasoning*-процесу, аналізу джерел інформації та оцінювання достатності отриманих даних. Система автоматично структурує результати виконання задачі та додає перелік використаних веб-джерел (рисунок 3.15).

Використання *source grounding*-підходу дозволяє підвищити достовірність результатів та забезпечити прозорість процесу формування відповіді.

Фінальна відповідь

Короткий підсумок: Інтелектуальні агенти, на відміну від звичайних пошукових систем, можуть автономно виконувати завдання, оскільки вони здатні сприймати інформацію, аналізувати її та діяти на основі отриманих даних для досягнення певних цілей.

Основні ідеї:

1. **Інтелектуальні агенти** - це програми, які можуть автономно виконувати дії, сприймаючи навколишнє середовище та реагуючи на зміни.
2. **Програмні агенти** - це специфічні реалізації інтелектуальних агентів, які автоматизують певні завдання.
3. **Пошукові системи** - це інструменти, які лише надають інформацію без виконання дій.

Що саме зробив агент: Інтелектуальний агент може виконувати завдання, такі як збір даних, аналіз інформації та прийняття рішень на основі отриманих результатів, що дозволяє йому діяти автономно.

Використані джерела:

- https://en.wikipedia.org/wiki/Intelligent_agent
- https://en.wikipedia.org/wiki/Software_agent

Короткий висновок: Інтелектуальні агенти мають здатність до автономного виконання завдань завдяки своїй здатності сприймати, аналізувати та діяти на основі інформації, що відрізняє їх від звичайних пошукових систем, які лише надають дані без активних дій.

Рисунок 3.15 – Формування фінальної відповіді AI-агентом за результатами виконання задачі

У межах проведених експериментів AI-агент продемонстрував здатність до автономного виконання веб-завдань, динамічного планування дій та адаптації до змін середовища. Найкращі результати система показала під час виконання задач, пов'язаних із пошуком інформації, аналізом джерел та формуванням структурованих відповідей.

Водночас у процесі тестування були виявлені окремі обмеження, пов'язані із CAPTCHA-захистом, нестабільністю окремих веб-ресурсів та залежністю від структури веб-сторінок.

Реалізований підхід продемонстрував ефективність поєднання великих мовних моделей, browser automation та механізмів self-evaluation у межах єдиної agent-based архітектури. Отримані результати можуть бути використані як основа для подальшого розвитку автономних AI-систем, здатних виконувати складні багатокрокові сценарії у динамічному цифровому середовищі, а також для створення інтелектуальних веб-помічників нового покоління.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було досліджено сучасні підходи до побудови інтелектуальних агентних систем та реалізовано прототип автономного AI-агента для автоматичного виконання завдань у веб-середовищі. Проведене дослідження підтвердило, що розвиток великих мовних моделей, browser automation-технологій та tool-augmented архітектур суттєво розширює можливості сучасних інформаційних систем і формує новий підхід до взаємодії людини з цифровим середовищем.

У теоретичній частині роботи було проаналізовано основні концепції побудови AI-агентів, принципи їх функціонування та особливості застосування у веб-середовищі. Дослідження показало, що традиційні системи автоматизації мають обмежену гнучкість та зазвичай працюють за наперед визначеними сценаріями, тоді як сучасні агентні системи здатні адаптуватися до змін середовища, виконувати reasoning-процеси, аналізувати результати власних дій та автономно формувати послідовність виконання задачі. Особливу увагу було приділено гібридним tool-augmented підходам, які поєднують можливості великих мовних моделей із зовнішніми інструментами браузерної автоматизації, веб-пошуку та memory-механізмами.

У процесі аналізу існуючих технологій було розглянуто сучасні інструменти автоматизації веб-взаємодії, серед яких Selenium, Puppeteer, Playwright, LangChain Agents, AutoGPT та інші агентні фреймворки. Проведений порівняльний аналіз дозволив визначити переваги модульної архітектури з використанням browser automation та reasoning-based orchestration. На основі отриманих результатів було обґрунтовано вибір архітектури власного AI-агента, що включає planner-модуль, executor-компонент, browser interaction layer, memory subsystem та механізм source grounding.

Практична частина роботи була присвячена проєктуванню та реалізації прототипу AI-агента для автономного виконання веб-завдань. У межах реалізованої системи було створено механізм обробки користувацьких запитів, побудови reasoning-ланцюга, автоматичного планування дій та взаємодії з веб-середовищем через browser automation. Система підтримує виконання багатокрокових сценаріїв, аналіз веб-джерел, формування структурованих відповідей та адаптацію до різних типів задач.

Під час реалізації було використано модульний підхід до побудови архітектури, що дозволяє масштабувати систему та інтегрувати додаткові інструменти без необхідності зміни базової логіки агента. Окрему увагу було приділено асинхронній взаємодії між компонентами системи, що дозволило підвищити ефективність виконання браузерних операцій та оптимізувати обробку веб-запитів. Використання Playwright забезпечило стабільну автоматизацію взаємодії з веб-сторінками, включаючи навігацію, пошук інформації, взаємодію з елементами інтерфейсу та обробку динамічного контенту.

У межах роботи також було реалізовано механізми memory management та source grounding, які забезпечують збереження контексту виконання задачі та підвищують достовірність сформованих відповідей. Завдяки використанню memory layer система здатна враховувати попередні кроки reasoning-процесу та підтримувати послідовність виконання складних багатокрокових сценаріїв. Source grounding дозволяє формувати відповіді на основі реальних веб-джерел, що підвищує прозорість роботи системи та якість отриманих результатів.

Проведене тестування продемонструвало працездатність реалізованого AI-агента та підтвердило його здатність автономно виконувати веб-завдання різного рівня складності. У процесі експериментів система успішно виконувала задачі, пов'язані з веб-пошуком, аналізом інформації, формуванням підсумків, обробкою багатомовних запитів та побудовою reasoning-відповідей на основі декількох джерел інформації.

Реалізований timeline-механізм дозволив відстежувати послідовність виконання агентного циклу та аналізувати ефективність окремих етапів роботи системи.

Результати тестування також дозволили виявити окремі обмеження сучасних AI-агентних систем. Зокрема, складнощі можуть виникати під час взаємодії з CAPTCHA-захистом, нестабільними веб-інтерфейсами або ресурсами із високим рівнем динамічності. Крім того, ефективність роботи агента значною мірою залежить від якості reasoning-моделі, структури веб-сторінок та доступності зовнішніх джерел інформації. Незважаючи на це, результати експериментів підтвердили перспективність використання AI-агентів для автоматизації рутинних веб-процесів та інтелектуальної підтримки користувачів у цифровому середовищі.

Отримані результати підтверджують актуальність дослідження та демонструють перспективність подальшого розвитку agent-based систем, здатних поєднувати можливості великих мовних моделей, browser automation та reasoning-механізмів. Практичне значення роботи полягає у можливості використання розробленої архітектури як основи для створення інтелектуальних систем автоматизації нового покоління, орієнтованих на автономну взаємодію з веб-середовищем та підтримку складних багатокрокових сценаріїв виконання задач.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Russell S. J., Norvig P. Artificial intelligence: a modern approach, pearson etext, global edition. Pearson Education, Limited, 2021.
2. Wooldridge M. Introduction to multiagent systems. Wiley & Sons, Incorporated, John, 2013. 484 p.
3. The rise and potential of large language model based agents: a survey / Z. Xi et al. *Science China information sciences*. 2025. Vol. 68, no. 2. DOI: <https://doi.org/10.1007/s11432-024-4222-0> (date of access: 10.04.2026).
4. Tool Learning with foundation models / Y. Qin et al. *ACM Computing surveys*. 2024. DOI: <https://doi.org/10.1145/3704435> (date of access: 10.04.2026).
5. Chakrawarti R. K., Sarangi S. K., Gadwal A. S. Robotic process automation. Wiley & Sons, Incorporated, John, 2023.
6. Fast and reliable end-to-end testing for modern web apps. *Playwright*. URL: <https://playwright.dev/> (date of access: 14.04.2026).
7. Chase H. *LangChain: building applications with LLMs*. 2023. URL: <https://docs.langchain.com/> (date of access: 15.04.2026).
8. Synergizing large language models and knowledge-based reasoning for interpretable feature engineering / M. Bouadi et al. *WWW '25: The ACM Web conference 2025*, Sydney NSW Australia. New York, NY, USA, 2025. P. 2606–2620. DOI: <https://doi.org/10.1145/3696410.3714720> (date of access: 15.04.2026).
9. A review on synergizing knowledge graphs and large language models / Z. Yang et al. *Computing*. 2025. Vol. 107, no. 6. DOI: <https://doi.org/10.1007/s00607-025-01499-8> (date of access: 16.04.2026).
10. Garg P., Beeram D. Large language model-based autonomous agents. *International journal of computer trends and technology*. 2024. Vol. 72, no. 5. P. 151–162. DOI: <https://doi.org/10.14445/22312803/ijctt-v72i5p118> (date of access: 16.04.2026).

11. Jian Y. Conceptual foundations of LLM-powered agents: from language processing to autonomous reasoning. *Scientific journal of intelligent systems research*. 2025. Vol. 7, no. 4. P. 29–35. DOI: <https://doi.org/10.54691/atmxvb43> (date of access: 16.04.2026).
12. Kumar A. Building autonomous AI agents based AI infrastructure. *International journal of computer trends and technology*. 2024. Vol. 72, no. 11. P. 116–125. DOI: <https://doi.org/10.14445/22312803/ijctt-v72i11p112> (date of access: 17.04.2026).
13. Alotaibi O., Tomy S., Pardede E. Enhancing decision-making with large language models: context and sensitivity analysis. *Evolutionary Intelligence*. 2025. Vol. 18, no. 6. DOI: <https://doi.org/10.1007/s12065-025-01113-y> (date of access: 17.04.2026).
14. Toolformer: language models can teach themselves to use tools / N. Cancedda et al. *Advances in neural information processing systems 36*, New Orleans, Louisiana, USA, 10–16 December 2023. San Diego, California, USA, 2023. P. 68539–68551. DOI: <https://doi.org/10.52202/075280-2997> (date of access: 18.04.2026).
15. Vidivelli S., Ramachandran M., Dharunbalaji A. Efficiency-driven custom chatbot development: unleashing langchain, RAG, and performance-optimized LLM fusion. *Computers, materials & continua*. 2024. P. 1–10. DOI: <https://doi.org/10.32604/cmc.2024.054360> (date of access: 18.04.2026).
16. Sankara Reddy Thamma. The rise and role of AI browser agents in modern digital workflows. *International journal of scientific research in computer science, engineering and information technology*. 2025. Vol. 11, no. 2. P. 3685–3690. URL: <https://doi.org/10.32628/cseit25112846> (date of access: 19.04.2026).
17. Kobayasi Y. Robotic process automation (RPA). *The journal of the institute of image information and television engineers*. 2019. Vol. 73, no. 2. P. 335–337. URL: <https://doi.org/10.3169/itej.73.335> (date of access: 19.04.2026).

18. Basso A., Bergadano F. Anti-bot strategies based on human interactive proofs. *Handbook of information and communication security*. Berlin, Heidelberg, 2010. P. 273–291. URL: https://doi.org/10.1007/978-3-642-04117-4_15 (date of access: 21.04.2026).
19. Narayan S. R. B., Agarwal N. Introduction to LangChain. *Mastering LangChain*. Berkeley, CA, 2025. P. 1–9. URL: https://doi.org/10.1007/979-8-8688-1718-2_1 (date of access: 21.04.2026).
20. Venkadesh P., Divya S. V., Kumar K. S. Unlocking AI Creativity: A Multi-Agent Approach with CrewAI. *Journal of trends in computer science and smart technology*. 2024. Vol. 6, no. 4. P. 338–356. URL: <https://doi.org/10.36548/jtcsst.2024.4.002> (date of access: 23.04.2026).
21. Semantic kernel agent framework. *Microsoft Learn: Build with answers in reach*. URL: <https://learn.microsoft.com/en-us/semantic-kernel/frameworks/agent/?pivots=programming-language-csharp> (date of access: 23.04.2026).
22. Retrieval-Augmented generation (RAG) with milvus and haystack. *Milvus*. URL: https://milvus.io/docs/integrate_with_haystack.md (date of access: 23.04.2026).
23. Tahir H. Smolagents vs langgraph: which one's easier to build and run AI agents. *The AI Control Plane*. URL: <https://www.zenml.io/blog/smolagents-vs-langgraph> (date of access: 23.04.2026).
24. The selenium browser automation project. *Selenium*. URL: <https://www.selenium.dev/documentation/> (date of access: 24.04.2026).
25. Introduction. *Puppeteer*. URL: <https://pptr.dev/category/introduction> (date of access: 25.04.2026).
26. Playwright. *Fast and reliable end-to-end testing for modern web apps*. URL: <https://playwright.dev/docs/intro> (date of access: 26.04.2026).
27. Introducing browserbase. *Browserbase Documentation*. URL: <https://docs.browserbase.com/welcome/introduction> (date of access: 27.04.2026).

28. A review of large language models: fundamental architectures, key technological evolutions, interdisciplinary technologies integration, optimization and compression techniques, applications, and challenges / S. Han et al. *Electronics*. 2024. Vol. 13, no. 24. P. 5040. URL: <https://doi.org/10.3390/electronics13245040> (date of access: 28.04.2026).
29. Large language models: an overview of foundational architectures, recent trends, and a new taxonomy / I. D. Mienye et al. *Discover applied sciences*. 2025. Vol. 7, no. 9. URL: <https://doi.org/10.1007/s42452-025-07668-w> (date of access: 29.04.2026).
30. Tollon F. Reactive agency and technology. *AI and ethics*. 2023. URL: <https://doi.org/10.1007/s43681-023-00366-6> (date of access: 30.04.2026).
31. Lukas Müller, Hannah Wagner, Maximilian Weber. Reinforcement learning of planning processes for tool-augmented LLM agents. *American journal of big data*. 2026. Vol. 7, no. 1. P. 185–192. URL: <https://doi.org/10.71465/ajbd3639> (date of access: 05.05.2026).
32. A tool-based framework to assess and challenge the responsiveness of emergency call centers / E. Petitdemange et al. *IEEE transactions on engineering management*. 2020. Vol. 67, no. 3. P. 568–581. URL: <https://doi.org/10.1109/tem.2019.2954013> (date of access: 05.05.2026).
33. Planner and executor: collaboration between discrete diffusion and autoregressive models in reasoning. *arXiv.org e-Print archive*. URL: <https://arxiv.org/html/2510.15244v1> (date of access: 05.05.2026).
34. Memory layers at scale. *arXiv.org*. URL: <https://arxiv.org/abs/2412.09764> (date of access: 06.05.2026).
35. Building reliable AI agents with modular and scalable frameworks. *Magnimind Academy*. URL: <https://magnimindacademy.com/blog/building-reliable-ai-agents-with-modular-and-scalable-frameworks/> (date of access: 06.05.2026).