МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВОЛИНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ЛЕСІ УКРАЇНКИ
КАФЕДРА ЗАГАЛЬНОЇ МАТЕМАТИКИ ТА МЕТОДИКИ НАВЧАННЯ
ІНФОРМАТИКИ

На правах рукопису

БОРОНЕНКО ВЛАДИСЛАВ ІГОРОВИЧ

ОСОБЛИВОСТІ РОЗРОБКИ ТА МЕТОДИКА ВИКОРИСТАННЯ ІНТЕЛЕКТУАЛЬНОГО АСИСТЕНТА ВЧИТЕЛЯ ІНФОРМАТИКИ

Спеціальність: 014 Середня освіта (Інформатика)

Освітньо-професійна програма Середня освіта. Інформатика

Робота на здобуття освітнього ступеня «магістр»

Науковий керівник:
ЮНЧИК ВАЛЕНТИНА ЛЕОНІДІВНА,
доктор філософії, доцент кафедри загальної математики та методики навчання інформатики

РЕКОМЕНДОВАНО ДО ЗАХИСТУ
Протокол № _____
засідання кафедри загальної математики
та методики навчання інформатики
від « ___ » _____ 2025 р.
Завідувач кафедри
_____ доц. Хомяк М. Я.

ЛУЦЬК 2025

ЗМІСТ

ВСТУП

Сучасна система освіти переживає етап цифрової трансформації, який характеризується не лише впровадженням нових технічних засобів, але й зміною парадигми взаємодії між учасниками освітнього процесу. Стрімкий розвиток генеративного штучного інтелекту, зокрема великих мовних моделей (LLM) та інтелектуальних агентів, відкриває нові можливості для автоматизації рутинних педагогічних задач, персоналізації навчання та підвищення ефективності уроків інформатики.

Вчитель інформатики сьогодні стикається з подвійним викликом: необхідністю постійно оновлювати зміст навчання відповідно до змін у ІТ-сфері та потребою в індивідуальному підході до учнів з різним рівнем цифрової компетентності. Традиційні методи не завжди дозволяють ефективно вирішувати ці завдання в умовах обмеженого часу. Використання ШІ у ролі асистента викладача здатне розвантажити педагога, взявши на себе функції генерації навчальних матеріалів, перевірки програмного коду та надання миттєвого зворотного зв'язку учням.

Незважаючи на наявність досліджень у галузі штучного інтелекту в освіті [1]-[3], методичні аспекти використання агентів-асистентів [4]-[6] саме на уроках інформатики в закладах загальної середньої освіти залишаються недостатньо розробленими, що й зумовлює актуальність розробки програмного комплексу, який поєднує потужність сучасних нейромереж із методично обґрунтованими сценаріями використання, що й визначило вибір теми кваліфікаційної роботи.

Об’єкт дослідження: процес навчання інформатики учнів закладів загальної середньої освіти.

Предмет дослідження: методика використання засобів штучного інтелекту у процесі підготовки та проведення уроків інформатики.

Мета дослідження: теоретично обґрунтувати, розробити програмний комплекс та методику використання інтелектуального асистента вчителя для

підвищення ефективності навчання інформатики, а також експериментально перевірити їх результативність.

Для виконання поставленої мети, визначені такі завдання:

1. Здійснити аналіз науково-педагогічної літератури та навчальних програм з інформатики для визначення доцільності використання штучного інтелекту в закладах загальної середньої освіти.

2. Визначити вимоги та спроєктувати теоретичну модель інтелектуального асистента вчителя.

3. Здійснити програмну реалізацію комплексу на основі технологій генеративного штучного інтелекту та розробити систему рольових інструкцій для реалізації модулів.

4. Обґрунтувати методику використання розробленого комплексу на різних етапах уроку інформатики.

5. Перевірити ефективність запропонованої методики та розробленого програмного засобу в навчальному процесі.

Для реалізації дослідження, необхідно використати наступні методи: аналіз науково-педагогічної літератури, нормативних документів та навчальних програм; синтез, моделювання (для створення моделі асистента), узагальнення, спостереження, анкетування вчителів та учнів.

Практичне значення одержаних результатів дослідження полягає у розробці веб-застосунку AI Teacher Assistant, який може бути розгорнутий у закладах загальної середньої освіти, а також у формуванні бібліотеки ефективних промптів для вчителів інформатики. Матеріали дослідження можуть бути використані вчителями інформатики та студентами педагогічних закладів вищої освіти.

Результати даної кваліфікаційної роботи були представлені на наукових конференціях:

XIV Міжнародна науково-практична конференція «Математика. Інформаційні технології. Освіта», Луцьк – Світязь, 13-15 червня 2025 р.

IX Всеукраїнській студентській науково-практичній конференції «Вища освіта – студентська наука – сучасне суспільство: напрями розвитку», 17-18 грудня 2025 року.

За результатами конференції було опубліковано тези в збірниках матеріалів даних конференцій.

РОЗДІЛ 1. ТЕОРЕТИКО-МЕТОДИЧНІ ЗАСАДИ ВИКОРИСТАННЯ ШТУЧНОГО ІНТЕЛЕКТУ В НАВЧАННІ ІНФОРМАТИКИ

1.1 Еволюція технологій штучного інтелекту в освітньому просторі

Інтеграція цифрових технологій в освітній процес пройшла довгий шлях від простих алгоритмізованих програм до складних нейромережевих моделей. Розуміння цієї еволюції є критично важливим для визначення місця сучасних великих мовних моделей (LLM) у системі педагогічних засобів. Історичний розвиток технологій AIEd (Artificial Intelligence in Education) можна умовно поділити на кілька етапів, кожен з яких характеризується зміною технологічної парадигми та підходів до взаємодії «людина–комп'ютер» [7].

Перші технологічні підходи до автоматизації освітнього процесу сформувалися у 1960–1970-х роках разом із появою систем комп’ютеризованого навчання CAI (Computer-Assisted Instruction). Їхня розробка спиралася на біхевіористичну концепцію Б. Ф. Скіннера [8], яка передбачала поділ навчального матеріалу на малі порції та контроль засвоєння через послідовні реакції користувача. Системи CAI фактично виконували функції електронних модулів лінійного подання інформації: вони демонстрували теоретичний фрагмент, пропонували тестове завдання та, залежно від правильності відповіді, здійснювали перехід до наступного елемента або повертали користувача до попереднього матеріалу [9].

З технічного погляду CAI реалізовувалися на основі жорстко визначених, детермінованих алгоритмів, що ґрунтувалися на лінійному чи розгалуженому програмуванні. Алгоритмічна логіка описувалася правилами типу `IF-THEN`, які забезпечували однозначність переходів між навчальними блоками. Основним обмеженням таких систем була відсутність механізмів адаптації: вони не аналізували причини помилок користувача й не могли модифікувати спосіб подання матеріалу відповідно до індивідуальних потреб [10].

У 1980–1990-х роках відбувся перехід до якісно нового етапу автоматизації навчання, який характеризується появою інтелектуальних навчальних систем

ITS (Intelligent Tutoring Systems). На відміну від попередніх рішень типу CAI, їхня архітектура ґрунтувалася на використанні трьох взаємопов'язаних компонентів [11]: моделі експерта, що репрезентувала знання певної предметної галузі; моделі учня, призначеної для динамічного відстеження рівня сформованості знань та типових помилок; а також педагогічної моделі, яка визначала стратегії та тактики подання навчального матеріалу.

Попри вищу адаптивність, ITS переважно спиралися на методи експертних систем і символьної парадигми штучного інтелекту. Усі можливі траєкторії міркувань, типи помилок, сценарії зворотного зв'язку та дидактичні реакції необхідно було детально формалізувати й закодувати на етапі розроблення. Така залежність від попередньо визначених правил істотно ускладнювала створення систем, робила процес їх побудови витратним і обмежувала здатність ITS працювати з відкритими, непередбачуваними форматами запитів, зокрема з природномовними відповідями або нестандартними фрагментами коду.

Сучасний етап розвитку технологій штучного інтелекту, який окреслюється періодом від початку 2010-х років і набуває особливої інтенсивності після 2022 року, пов'язаний із переходом від символьних методів до моделей машинного навчання та глибинних нейронних мереж. У цьому контексті виокремлюють дві базові парадигми штучного інтелекту – дискримінативну та генеративну.

1. Дискримінативний ШІ [12].

Він зосереджений на аналізі, класифікації та прогнозуванні на основі наявних даних. У сфері освіти такі моделі застосовують для:

• прогнозування результатів навчання, коли система, використовуючи попередні оцінки учня, оцінює ризики відрахування або низького підсумкового бала;

• класифікації, що охоплює автоматизовану перевірку тестових завдань і розпізнавання рукописного тексту;

• рекомендаційних систем, які формують індивідуальні навчальні траєкторії на основі історії навчальної активності.

Концептуально дискримінативний підхід спрямовано на встановлення меж між класами даних, наприклад «розв'язок коректний» або «розв'язок містить помилку». Він не передбачає створення нових інформаційних об'єктів.

2. Генеративний ШІ [13].

Цей напрям розглядається як ключовий для сучасних досліджень, оскільки орієнтований на формування нових даних, статистично подібних до навчальної вибірки. На відміну від класифікаційних моделей, генеративні системи, зокрема великі мовні моделі, використовують імовірнісні механізми передбачення наступного елемента послідовності, яким може бути слово в реченні або оператор у коді програмування.

Термін AIEd (Artificial Intelligence in Education) розглядається як міждисциплінарна галузь, що інтегрує комп'ютерні науки, педагогіку, психологію та нейронауки з метою розроблення адаптивних освітніх систем [14]. У межах однієї з поширених класифікацій, запропонованої провідними дослідниками [7], інструменти AIEd поділяють на три базові групи [7].

1. Інструменти, орієнтовані на учня.

До цієї категорії належать адаптивні навчальні платформи, які модифікують рівень складності навчального матеріалу відповідно до індивідуальних особливостей учня, а також інтелектуальні тьюторські системи.

2. Інструменти, орієнтовані на вчителя.

Цю групу формують системи, призначені для автоматизації рутинних професійних завдань, створення навчального контенту та моніторингу освітнього прогресу. Саме до цієї категорії належить об'єкт даного дослідження – інтелектуальний асистент викладача.

3. Інструменти, орієнтовані на систему.

Йдеться про аналітичні інструменти, що використовують великі дані для підтримки управлінських рішень на рівні освітніх інституцій та державних структур, зокрема системи навчальної аналітики.

Узагальнено, сучасний асистент викладача на основі генеративних моделей штучного інтелекту репрезентує природний етап розвитку освітніх

технологій. Він поєднує функції персоналізованого тьютора й засобу автоматизації професійної діяльності, але реалізований на принципово новій технологічній основі ймовірнісних моделей.

1.2. Психолого-педагогічні та технічні аспекти використання великих мовних моделей

Сучасні інтелектуальні асистенти функціонують на основі технологій великих мовних моделей (Large Language Models, LLM) (Рис. 1.1). Дані моделі належать до класу глибинних нейронних мереж, розроблених для опрацювання, інтерпретації та генерації природномовних висловлювань. Для всебічного аналізу їхнього дидактичного потенціалу доцільно розглянути принципи роботи таких моделей, основні етапи їх підготовки та відмінності порівняно з автономними агентними системами [15].

Великі мовні моделі (LLM) – це клас систем штучного інтелекту, що навчаються на основі величезних обсягів даних та демонструють здатність до опрацювання й генерування природної мови. Як узагальнено на рисунку 1.1, LLM складаються з кількох функціональних компонентів, а особливості їх архітектури та процесу навчання забезпечують виконання широкого спектра мовних завдань [16].

У центрі діаграми подано узагальнену структуру LLM [17] з вхідними та вихідними вузлами та проміжним обчислювальним блоком, який може репрезентувати архітектуру трансформера – ключову технологічну інновацію сучасних мовних моделей. Основою їх функціонування є великі набори даних, що містять тексти різних жанрів і програмний код. Саме якість та різнорідність цих наборів визначають ефективність формування мовних закономірностей і знань.

Навчальні дані проходять попереднє опрацювання й використовуються для попереднього навчання моделі, у межах якого LLM оптимізує здатність передбачати наступний елемент мовної послідовності. Такий механізм формує базовий рівень мовної компетентності моделі. Важливу роль відіграє

самоконтрольоване навчання, за якого модель формує навчальні мітки без участі людини, що дає змогу ефективно масштабувати навчальний процес і працювати з величезними корпусами даних за нижчої вартості порівняно з традиційними методами [18].

Технологічною основою LLM є глибинне навчання як підгалузь машинного навчання. Воно використовує багатошарові нейронні мережі для вивчення складних статистичних залежностей і забезпечує здатність моделі до опрацювання довгих та контекстно насичених мовних структур. Вирішальне значення має архітектура трансформера, що дає змогу моделі аналізувати цілі послідовності одночасно та фокусувати увагу на релевантних фрагментах вхідних даних. Запропонована у 2017 році в праці «Attention is All You Need» [19], вона виявилася суттєво ефективнішою за попередні архітектури на основі рекурентних нейронних мереж [20].

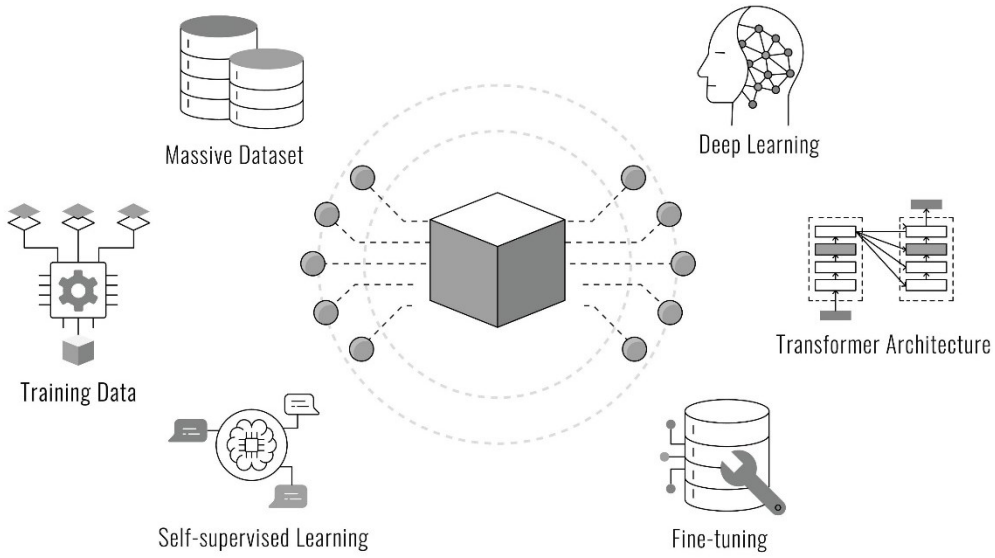


Рисунок 1.1 – Large Language Models

Завершальним етапом є фінальне налаштування моделі, яке передбачає донавчання попередньо підготовленої LLM на спеціалізованому наборі даних для конкретного завдання. Такий підхід дає змогу адаптувати модель до реальних застосувань, підвищуючи точність та релевантність її відповідей.

Архітектура трансформера є базовою технологічною основою сучасних великих мовних моделей, зокрема GPT-4, LLaMA та Claude. Вона була представлена дослідницькою групою Google у 2017 році й стала принципово новим підходом до опрацювання мовних послідовностей. На відміну від рекурентних нейронних мереж, які опрацьовували текст послідовно, трансформер забезпечує паралельне опрацювання всіх елементів вхідної послідовності, що суттєво підвищує ефективність і масштабованість моделі [21].

Ключовим елементом трансформера є механізм самоуваги. Він дає змогу моделі динамічно визначати значущість кожного токена відносно інших токенів у послідовності. Завдяки цьому формується глибоке контекстне представлення тексту. Наприклад, у реченні *«Функція не повертає значення, тому що вона оголошена як void»* механізм самоуваги дозволяє моделі коректно співвіднести займенник *«вона»* з лексемою *«функція»*, а не з *«значення»* [22].

З формальної точки зору діяльність LLM ґрунтується на процедурі прогнозування наступного токена в послідовності. Токен розуміється як елемент тексту, що може бути словом, частиною слова або окремим символом. Модель оцінює ймовірність появи токена w_t на підставі контексту попередніх елементів $(w_1, w_2, \ldots, w_{t-1})$:

$$P(w_t \mid w_1, \ldots, w_{t-1})$$

Саме здатність моделі обчислювати таке умовне розподілення й передбачати продовження мовної чи кодової послідовності лежить в основі її генеративних можливостей.

Етапи підготовки великої мовної моделі формують послідовний процес перетворення базової нейромережевої структури на функціональний інтелектуальний асистент, здатний працювати у педагогічному середовищі. Цей процес охоплює три основні стадії [22].

1. Попереднє навчання.

На початковому етапі модель опрацьовує масштабні масиви неструктурованих текстових даних, серед яких Common Crawl, Wikipedia,

GitHub та інші відкриті корпуси. У ході цього процесу вона засвоює статистичні властивості мови, логіко-семантичні зв'язки, фактуальну інформацію та синтаксичні особливості природних і формальних мов, зокрема C++ і Python. Результатом є базова модель, здатна генерувати продовження тексту, однак ще позбавлена навичок виконання інструкцій чи ведення цілеспрямованого діалогу.

2. Точне налаштування.

На другому етапі модель додатково навчають на добірках пар «інструкція – еталонна відповідь». Це забезпечує формування поведінки, орієнтованої на діалогову взаємодію, а також навчає модель дотримуватися визначених форматів відповіді й логіки спілкування.

3. Навчання з підкріпленням на основі людських оцінок.

Цей етап має вирішальне значення для застосування в освітній сфері. Експерти аналізують альтернативні відповіді моделі та ранжують їх за корисністю, точністю й безпечністю. На основі цих даних формують модель винагороди, яка слугує орієнтиром для подальшої оптимізації основної LLM [23].

Значення цього процесу для шкільної освіти полягає в тому, що RLHF формує соціально прийнятну поведінку асистента, зокрема його здатність уникати небезпечних або шкідливих відповідей, підтримувати коректний стиль взаємодії та забезпечувати максимально об’єктивні рекомендації.

У контексті створення асистента вчителя доцільно чітко розмежовувати поняття мовної моделі та інтелектуального агента. Мовна модель розглядається як пасивний обчислювальний механізм, що перетворює вхідний текст у вихідний без власних структур пам’яті, без доступу до інформації про поточний стан середовища та без можливості взаємодії із зовнішніми ресурсами, якщо такі можливості не передбачено додатковими засобами.

Інтелектуальний агент являє собою складнішу систему, у якій LLM виконує функцію центрального інтерпретувального компонента, інтегрованого з додатковими модулями [24]. Така система може включати кілька ключових елементів:

– Пам’ять – забезпечує збереження та використання індивідуальної історії взаємодії з учнем, що дає змогу реалізувати персоналізацію навчальних сценаріїв.

– Планування – передбачає здатність агента структурувати складні завдання, поділяючи їх на послідовні підзадачі, що узгоджується з концепцією міркування ланцюжками.

– Використання інструментів – надає можливість залучати зовнішні програмні засоби. Наприклад, для оцінювання студентських програм агент може ініціювати виконання коду в ізольованому середовищі, отримуючи об’єктивний результат роботи, а не покладаючись виключно на текстову інтерпретацію.

Проблема галюцинацій у застосуванні генеративних моделей у навчальному процесі становить один із ключових ризиків їх використання. Під галюцинаціями розуміють упевнене формування хибної або некоректної інформації, що зумовлено статистичною природою мовної моделі, яка не володіє фактичними знаннями, а лише передбачає найімовірніші мовні продовження.

В інформатичній освіті така властивість породжує низку специфічних викликів.

Модель може пропонувати неіснуючі бібліотеки, функції чи конструкції мов програмування, що виглядають правдоподібними, але не мають відповідників у реальних технологічних стеках.

Генерований код може бути синтаксично коректним і навіть компілюватися, проте не відповідати поставленому завданню. Типовим прикладом є порушення логіки алгоритму, що призводить до помилкових результатів попри формальну коректність.

Унаслідок цього використання інтелектуального асистента в освітньому середовищі потребує формування в учителів і здобувачів освіти навичок верифікації інформації, критичного аналізу результатів та здійснення ретельного перегляду створеного коду.

1.3. Методичні аспекти та аналіз існуючих рішень

Упровадження технологій штучного інтелекту в освітній процес потребує не лише технічних ресурсів, а й переосмислення методичних підходів. Результативність використання інтелектуального асистента визначається не стільки обчислювальними можливостями моделі, скільки якістю її інтеграції в навчальну діяльність та вибором оптимальної платформи.

Ринок пропонує широкий спектр інструментів на основі великих мовних моделей, які доцільно класифікувати відповідно до їх функціонального призначення [25].

1. *Універсальні великі мовні моделі.* До цієї групи належать ChatGPT (OpenAI), Claude (Anthropic) та Microsoft Copilot. Переваги полягають у широкій інформаційній базі, здатності працювати з багатьма мовами, зокрема українською, а також у високій якості генерації текстів і програмного коду. У контексті освітніх завдань ці моделі часто виступають базовою платформою для створення спеціалізованих асистентів. Наприклад, Claude 3 вирізняється значним обсягом контекстного вікна, що забезпечує можливість аналізу великих методичних матеріалів. Microsoft Copilot інтегрований в офісні програми, що оптимізує адміністративні процеси [26].

2. *Спеціалізовані освітні платформи.* Прикладами є Khanmigo (Khan Academy) та Duolingo Max. Особливістю таких систем є використання LLM із чітко регламентованими системними інструкціями. Вони реалізують сократичний підхід, спрямований на розвиток мислення через постановку навідних запитань замість подання готових відповідей. Це наближає їх до концепції педагогічно орієнтованого штучного інтелекту.

3. *Інструменти для розробників (AI Coding Assistants).* Для вчителя інформатики критично важливим є опанування GitHub Copilot, Amazon CodeWhisperer та JetBrains AI. Їхня специфіка полягає в інтеграції безпосередньо в середовища розробки, що дає змогу генерувати автодоповнення коду в режимі реального часу. Методичний виклик полягає в трансформації підходів до навчання програмування, зокрема у зміщенні акценту від запам’ятовування

синтаксису до проєктування програмної архітектури та верифікації згенерованих рішень.

Аналіз даних, поданих у таблиці 1.1, демонструє відсутність універсального інструмента, здатного повною мірою охопити всі види освітніх завдань. Для підтримки навчання програмування, зокрема автоматизованого генерування та корекції програмного коду, найбільш ефективними є GitHub Copilot та спеціалізовані агентні моделі, створені на базі ChatGPT. Водночас у контексті загальної педагогічної взаємодії, з особливим акцентом на безпечну підтримку молодших школярів, найбільш збалансованою системою залишається архітектура, реалізована в Khanmigo. У цьому дослідженні увагу зосереджено на адаптації універсальних великих мовних моделей (типу ChatGPT або Claude), оскільки вони є доступними, масштабованими та придатними до гнучкого налаштування відповідно до потреб учителя інформатики в українській школі.

Таблиця 1.1.

Порівняльна характеристика інструментів ШІ для освітніх цілей

Критерій порівняння	ChatGPT (OpenAI)	GitHub Copilot	Khanmigo (Khan Academy)
Спеціалізація	Універсальний інструмент (генерація текстів, програмного коду, розв'язання задач, творче моделювання)	Вузькоспеціалізований IT-інструмент (автодоповнення коду, редагування та оптимізація синтаксису, підтримка написання тестів)	Педагогічна платформа (навчальні сценарії, підтримка учнів, методична допомога вчителю)
Вартість	Базова версія безкоштовна; розширена (GPT-4o) – платна (~$20/місяць).	Платна; безкоштовна для верифікованих студентів і викладачів. Стандартна вартість ~ $10/місяць.	Платна; орієнтовно ~$4/місяць або шкільна ліцензія (вартість залежить від плану)
Рівень галюцинацій	Середній (у безкоштовних моделей можливі некоректні факти; платні моделі працюють значно точніше)	Низький (контекстно орієнтована генерація; рідко продукує хибний синтаксис, можливі помилки під час відтворення застарілих патернів коду)	Мінімальний (відповіді ґрунтуються на верифікованих навчальних матеріалах, інструмент обмежений системними інструкціями)

Придатність для школи	Умовна (потребує контролю з боку вчителя через вікові обмеження (13+) та широкий доступ до будь-якого контенту)	Висока (ідеальний інструмент для уроків інформатики та гуртків програмування (10-11 класи))	Максимальна (розроблений спеціально для шкіл з урахуванням безпеки даних (COPPA) та педагогічної етики)
Основна перевага для вчителя інформатики	Універсальність (генерація планів уроків, створення фрагментів коду, модифікація та пояснення програм, переклад і адаптація матеріалів)	Професійна спрямованість (моделювання реальних ситуацій програмування, інтеграція з IDE, підтримка проєктного навчання)	Безпечний і педагогічно орієнтований інструмент (допомога учням у навчанні та розвитку самостійності)

Класифікація функціональних можливостей асистента вчителя інформатики

Роль інтелектуального асистента в освітньому процесі доцільно структурувати за трьома ключовими напрямами, що відображають різні аспекти педагогічної діяльності: адміністративний, методичний та тьюторський [27].

У межах *адміністративної функції* асистент бере на себе рутинні організаційні дії, що не потребують творчої роботи, але суттєво впливають на часові витрати педагогів. Йдеться про автоматизоване формування календарно-тематичних планів відповідно до чинної навчальної програми, створення узагальнених звітів та аналітики успішності класу, а також підготовку типових шаблонів електронних листів для комунікації з батьками та адміністрацією закладу.

Методична функція охоплює підтримку вчителя на етапі підготовки до заняття, де асистент виконує роль інструмента концептуальної допомоги та генерації ідей. У цьому контексті він забезпечує розроблення завдань з програмування різного рівня складності для реалізації диференціації навчання, автоматичне створення тестових матеріалів для платформ Kahoot або Moodle із готовими варіантами відповідей, а також адаптацію навчального контенту, зокрема переформулювання складних технічних термінів доступною та зрозумілою мовою для молодших учнів [9].

У випадку *тьюторської функції* асистент виконує роль індивідуального цифрового наставника, що забезпечує підтримку учня під час опанування програмування. Він може надавати покрокові пояснення алгоритмів, допомагати у виявленні та локалізації помилок у коді без надання готового розв'язку, а також забезпечувати миттєвий зворотний зв'язок щодо виконаних робіт, що зменшує затримку між виконанням завдання та отриманням рекомендацій для подальшого вдосконалення.

Етичні та психологічні виклики впровадження

Інтеграція інструментів штучного інтелекту в освітній процес супроводжується низкою ризиків, які необхідно враховувати під час розроблення методичних підходів.

1. Академічна доброчесність і проблема використання згенерованого коду. Доступ до універсальних мовних моделей створює для учнів можливість отримати готові програмні розв'язки без реального опрацювання змісту завдання. Оптимальним запобіжником є зміна форматів оцінювання, коли пріоритет надається здатності пояснити роботу поданого коду, модифікувати його або виявити навмисно вставлену помилку, що переставляє акцент з репродукції на розуміння.

2. Трансформація ролі вчителя і когнітивне навантаження. Широке застосування ШІ може створювати відчуття поступової втрати професійної компетентності педагогів. Водночас автоматизація рутинних завдань знижує їхнє навантаження та сприяє концентрації на менторській, виховній та комунікативній діяльності. Ключовим принципом є дотримання підходу Human-in-the-loop, у межах якого відповідальність за фінальні педагогічні рішення зберігається за людиною.

3. Приватність і захист даних. Використання публічних мовних моделей регулюється вимогами щодо конфіденційності, зокрема положеннями GDPR. Недопустимим є передавання чат-ботам персональних відомостей про учнів або їхніх навчальних матеріалів, якщо це не передбачено офіційною політикою використання корпоративних версій інструментів. Педагог має забезпечувати

високий рівень цифрової гігієни та демонструвати відповідальні практики опрацювання інформації.

1.4. Аналіз навчальних програм та підручників з інформатики в контексті інтеграції ШІ-асистента

Для визначення доцільності застосування інтелектуального асистента було здійснено попередній аналіз чинних навчальних програм з інформатики для закладів загальної середньої освіти, затверджених Міністерством освіти і науки України. Такий аналіз дає змогу окреслити змістові компоненти, що створюють найбільші труднощі для учнів, а також визначити ті аспекти навчання, у яких підтримка з боку ШІ може бути найбільш результативною.

У межах базової середньої освіти (5-9 класи), відповідно до концепції «Нової української школи» [28], навчальний процес вибудовується на основі модельних програм. Розглянуті програми «Інформатика. 5-6 класи» (авторські колективи Ривкінд Й. Я., Лисенко Т. І. та ін.[29], а також Морзе Н. В., Барна О. В. та ін.) демонструють, що на цьому етапі учні переходять від візуального програмування у середовищах Scratch і Blockly до початкового ознайомлення з текстовими мовами, зокрема Python. Саме перехід від «блокової» логіки до синтаксично структурованого текстового коду створює найбільші труднощі для здобувачів освіти, оскільки потребує переорієнтації на абстрактніші способи мислення та формалізоване подання алгоритмів.

Використання інтелектуального асистента в такому контексті потенційно зменшує навчальне навантаження та підвищує доступність складних понять. Система може пояснювати відповідність між блоковими конструкціями та елементами текстового коду, наприклад, демонструючи, як цикл повторення у Scratch співвідноситься з конструкцією for у Python. Крім того, вона здатна створювати навчальні ситуації з елементами гейміфікації, формуючи сюжетні задачі, що підсилюють мотивацію учнів і сприяють кращому засвоєнню основ програмування.

У 7–9 класах, відповідно до навчальної програми для закладів загальної середньої освіти, затвердженої Наказом МОН від 07.06.2017 № 804 [30], учні опановують базові алгоритмічні структури мов програмування, серед яких переважає Python, а подекуди використовується й C++. Основний зміст навчання охоплює вивчення типів даних, умовних конструкцій, циклічних алгоритмів та роботи з одновимірними масивами, що формує основу для подальшого освоєння більш складних алгоритмічних концепцій. Навчальний матеріал зазвичай подається із використанням сучасних підручників, зокрема видання «Інформатика. 8 клас» авторів Бондаренко О. О. та Ластовецького В. В. [31], яке систематизує змістові лінії предмета й забезпечує методичну підтримку для вивчення відповідного програмного матеріалу.

На рівні профільної середньої освіти вивчення мов програмування набуває більшої системності та глибини. Особливо актуальним стає опанування C++, що характерно для класів інформатичного профілю. В основу аналізу покладено «Навчальну програму з інформатики для 10–11 класів (рівень стандарту та профільний рівень)» [32]-[33].

На рівні стандарту основним інструментом залишається Python, який використовується переважно для розв’язання прикладних задач, опрацювання даних та побудови елементарних веб-ресурсів. Програмування в цьому випадку не є домінантною складовою змісту, а розглядається як один із компонентів загальної цифрової компетентності.

Профільний рівень характеризується значно ширшим спектром мов, серед яких Python, C++, Java та C#. На цьому етапі програмування вивчається поглиблено, включаючи засвоєння концепцій об’єктно-орієнтованого підходу, аналіз алгоритмів підвищеної складності, опрацювання файлів та створення графічних інтерфейсів. У центрі методичного забезпечення перебуває підручник «Інформатика (профільний рівень) 10–11 клас» авторства Руденка В. Д. [34], у якому значна увага приділена мові C++ як інструменту для вивчення системного програмування та підготовки учнів до олімпіадних змагань.

На основі аналізу підручників визначено точки входу для використання розробленого асистента (Таблиця 1.2).

Таблиця 1.2.

Використання асистента при вивченні програмування

Клас / Рівень	Тема (згідно програми)	Типові труднощі учнів	Запропонована функція ШІ-асистента
7-8 клас (Базовий)	Циклічні алгоритми (`for`, `while`)	Розуміння умови виходу з циклу, нескінченні цикли.	Асистент пояснює ітерації крок за кроком ("Trace table generation").
8-9 клас (Базовий)	Одновимірні масиви (Списки)	Індексація (початок з 0 чи 1), вихід за межі масиву.	ШІ аналізує код і вказує: "Ти намагаєшся звернутися до 11-го елемента масиву, який має лише 10".
10-11 клас (Профільний)	Функції та процедури (C++)	Область видимості змінних, передача параметрів за посиланням/значенням.	Генерація прикладів, що показують різницю між `void func(int x)` та `void func(int &x)`.
10-11 клас (Профільний)	Об'єктно-орієнтоване програмування (Класи, об'єкти)	Розуміння абстракції, інкапсуляції, конструкторів.	Асистент генерує шаблон класу ("скелет") і просить учня дописати методи, пояснюючи суть `private`/`public`.

Аналіз змісту навчальних програм та підручників провідних авторів дає підстави стверджувати, що використання інтелектуального асистента є найбільш обґрунтованим у двох ключових навчальних періодах. У 8-9 класах така система може підвищити ефективність опанування синтаксису Python та полегшити виявлення типових логічних помилок, які учні часто допускають під час написання початкових програм. На етапі профільної середньої освіти (10-11 класи) доцільність застосування асистента зростає ще більше, оскільки саме тут учні переходять до поглибленого вивчення C++, засвоюють парадигму об’єктно-орієнтованого програмування та працюють зі складнішими алгоритмічними структурами. У цих умовах інтелектуальний асистент може компенсувати нестачу варіативних і високорівневих задач, які не завжди представлені у традиційних підручниках, та забезпечити індивідуальну підтримку в процесі формування алгоритмічного мислення.

РОЗДІЛ 2. ПРОГРАМНА РЕАЛІЗАЦІЯ, МЕТОДИКА ЗАСТОСУВАННЯ ТА ПЕРЕВІРКА ЕФЕКТИВНОСТІ ІНТЕЛЕКТУАЛЬНОГО АСИСТЕНТА

2.1. Програмна реалізація компонентів системи інтелектуальної підтримки вчителя

2.1.1. Моделювання та визначення вимог до програмного комплексу AI Teacher Assistant

З огляду на комплексний характер професійної діяльності вчителя інформатики, яка поєднує методичні, технічні та педагогічні аспекти, було обґрунтовано доцільність проєктування системи за модульною архітектурою. Передбачається формування інтегрованого набору спеціалізованих інструментів, об’єднаних спільним веб-інтерфейсом.

Функціональні вимоги до модулів системи

Програмний комплекс має підтримувати роботу чотирьох автономних модулів, кожен із яких використовує власну конфігурацію системних інструкцій та сценаріїв взаємодії.

1. Модуль Методист.

Модуль орієнтовано на автоматизацію процесу підготовки до уроків. Основні вимоги включають здатність формувати поурочні плани відповідно до положень НУШ; забезпечувати структуровану подачу матеріалу (тема, навчальна та виховна мета, тип уроку, необхідне обладнання, детальний хід заняття). Під час опрацювання тем із програмування система повинна передбачати обов’язковий практичний компонент.

2. Модуль Генератор завдань.

Модуль призначений для створення банку диференційованих задач із програмування мовами C++ та Python. До функціональних вимог належить можливість застосування елементів гейміфікації, зокрема генерація сюжетної «легенди» для підвищення мотивації учнів. Система забезпечує структурований формат завдання, що містить точний опис технічної проблеми, а також

характеристики вхідних і вихідних даних. Додатково модуль автоматично формує тестові приклади, не менше трьох пар «вхідні дані – результат», для первинної перевірки коректності розв’язання.

3. Модуль Екзаменатор.

Цей модуль виконує статичний аналіз учнівського коду та формує попереднє оцінювання. Функціональність передбачає виявлення синтаксичних і логічних помилок у програмах, написаних мовами C++ і Python, а також надання оцінки за 12-бальною шкалою на основі показників працездатності, алгоритмічної ефективності та дотримання стилю коду. Модуль забезпечує конструктивний зворотний зв’язок через рекомендації щодо виправлення недоліків без подання готового розв’язку.

4. Модуль Тьютор.

Модуль орієнтовано на індивідуальну взаємодію з учнем у діалоговому режимі. Його робота ґрунтується на використанні сократівського методу, коли навчання здійснюється через систему запитань, що стимулюють мислення. Програмний комплекс блокує прямі відповіді, зокрема прохання надати готовий код, пропонуючи натомість алгоритм розв’язання або концептуальну підказку.

Нефункціональні вимоги

Нефункціональні характеристики системи визначають особливості її архітектури, організації інтерфейсу та рівня гнучкості конфігурування. Вони спрямовані на забезпечення зручності використання, стабільної роботи та можливості адаптації до різних освітніх середовищ.

1. Інтерфейс користувача.

Програмний комплекс має функціонувати у форматі односторінкового застосунку, що забезпечує безперервність взаємодії та мінімізує затримки під час навігації. Переміщення між модулями здійснюється через бічну панель, без повного перезавантаження сторінки. Інтерфейс повинен підтримувати форматування Markdown, що забезпечує коректне відображення програмного коду із підсвічуванням синтаксису. Візуальне оформлення має залишатися

стриманим і максимально нейтральним, орієнтованим на академічну роботу без надмірних декоративних елементів.

2. Продуктивність і технологічна база.

Система повинна забезпечувати високу швидкодію та мінімальний час відгуку. Це досягається використанням швидких моделей штучного інтелекту, таких як представники серії Gemini Flash, що дає змогу знижувати затримку відповіді до рівня менше трьох секунд. Передбачається можливість локального розгортання, що є важливим для закладів освіти з обмеженим доступом до хмарних сервісів.

3. Конфігурування та гнучкість.

Програмна архітектура передбачає розмежування основної логіки застосунку та логіки поведінки моделей ШІ. Системні промпти зберігаються в окремому конфігураційному модулі, наприклад у файлі *prompts_config.py*. Така організація дає змогу педагогам змінювати поведінку асистента, зокрема коригувати критерії оцінювання, не маючи спеціальних знань у сфері програмування.

2.1.2. Архітектура програмного комплексу та обґрунтування вибору інструментальних засобів

Реалізація системи AI Teacher Assistant базується на клієнт-серверній архітектурі з використанням хмарних обчислень для опрацювання запитів природною мовою. Вибір інструментальних засобів здійснювався за критеріями: швидкість розробки, доступність для кінцевого користувача та мінімізація експлуатаційних витрат.

Для розробки програмного комплексу було обрано наступний стек технологій (Таблиця 2.1):

Таблиця 2.1.

Технологічний стек проєкту

Компонент системи	Обрана технологія	Обґрунтування вибору
Мова програмування	Python 3.10+	Стандарт де-факто у сфері штучного інтелекту; наявність потужних бібліотек для роботи з API та веб-розробки.
Web-фреймворк	Streamlit	Дозволяє створювати інтерактивні веб-застосунки (Single Page Application) виключно на Python, без необхідності написання HTML/CSS/JS коду. Ідеально підходить для створення прототипів та освітніх інструментів.
LLM провайдер	Google Gemini API	Використання моделі Gemini 2.0 Flash. На відміну від платних аналогів (OpenAI GPT-4), надає безкоштовний доступ для розробників, має низьку затримку (low latency) та велике контекстне вікно.
Середовище	Virtualenv (venv)	Забезпечує ізоляцію залежностей проєкту, що дозволяє розгортати його на будь-якому шкільному комп'ютері без конфліктів системних бібліотек.

Архітектура системи

Програмний комплекс побудовано за модульним принципом і реалізовано як веб-застосунок із трирівневою архітектурою. Така структура забезпечує чітке розмежування функцій між компонентами та підвищує масштабованість системи (Рис.2.1.).

1. Рівень представлення.

Цей рівень реалізовано засобами бібліотеки Streamlit і він забезпечує взаємодію користувача з системою. Основні функції включають формування інтерфейсу, приймання текстових запитів, а також візуалізацію результатів. Особливу увагу приділено коректному рендерингу Markdown та підсвічуванню синтаксису програмного коду мов C++ і Python.

2. Рівень бізнес-логіки.

Це центральний обчислювальний компонент системи, написаний мовою Python. Він відповідає за маршрутизацію користувацьких дій, керування

контекстом та взаємодію із зовнішніми сервісами. Рівень містить такі ключові підсистеми:

– Маршрутизатор, що визначає обраний користувачем режим роботи (Методист, Генератор, Екзаменатор або Тьютор).

– Менеджер контексту, який завантажує відповідні системні інструкції з конфігураційного файлу та підтримує історію діалогу у межах поточної сесії.

– Адаптер API, що формує запити до зовнішнього сервісу Google API, опрацьовує відповіді та контролює виникнення можливих помилок з'єднання.

3. Рівень даних і конфігурації

Архітектура не передбачає використання повноцінних реляційних баз даних, оскільки робота із системою відбувається у режимі коротких сесій. Натомість застосовуються легкі файлові структури:

prompts_config.py – файл, у якому зберігаються текстові інструкції та рольові моделі, що визначають поведінку асистента.

.env – конфігураційний файл змінних оточення, призначений для безпечного зберігання ключів доступу.

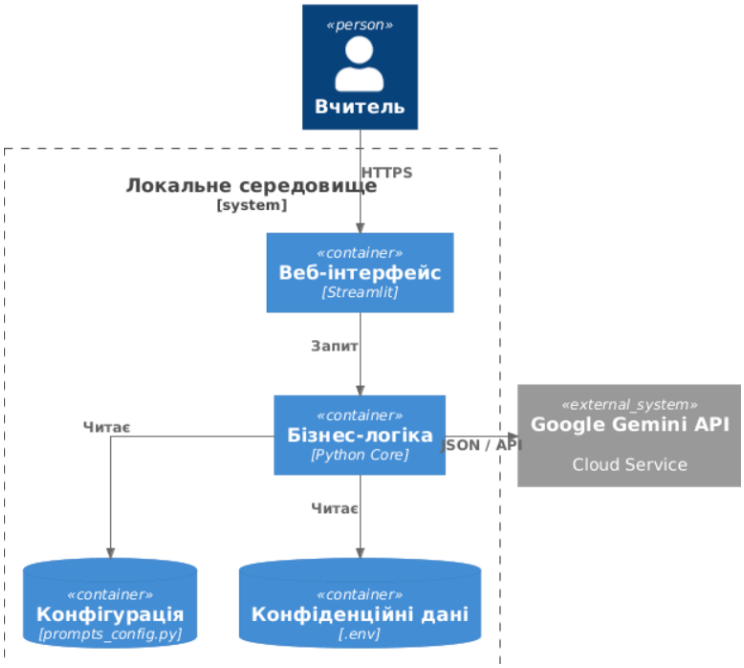


Рисунок 2.1. Архітектура системи AI Teacher Assistant

Схема інформаційної взаємодії

Опрацювання користувацького запиту організовано як послідовний процес, у якому кожен етап формує контекст для наступної операції (Рис. 2.2.).

1. Ініціалізація режиму. Після вибору користувачем відповідного модуля система завантажує пов’язаний із ним системний промпт. Наприклад, для режиму Генератор активується рольова інструкція «Розробник задач».

2. Формування запиту. Уведений учителем текстовий запит поєднується з даними поточної сесії, включно з фрагментами попередньої взаємодії. Це забезпечує збереження контексту діалогу.

3. Ін’єкція контексту. До сформованого запиту автоматично додається системна інструкція, яка визначає стиль, формат і структуру майбутньої відповіді. Цей компонент залишається прихованим від користувача, проте регулює поведінку моделі.

4. Генерація відповіді. Сформований запит передається до моделі Google Gemini, яка виконує обчислення та повертає результат у вигляді текстового блока.

5. Візуалізація результатів. Отримана відповідь відтворюється у середовищі Streamlit. Компоненти інтерфейсу автоматично застосовують форматування Markdown, зокрема підсвічування коду та структурування таблиць.

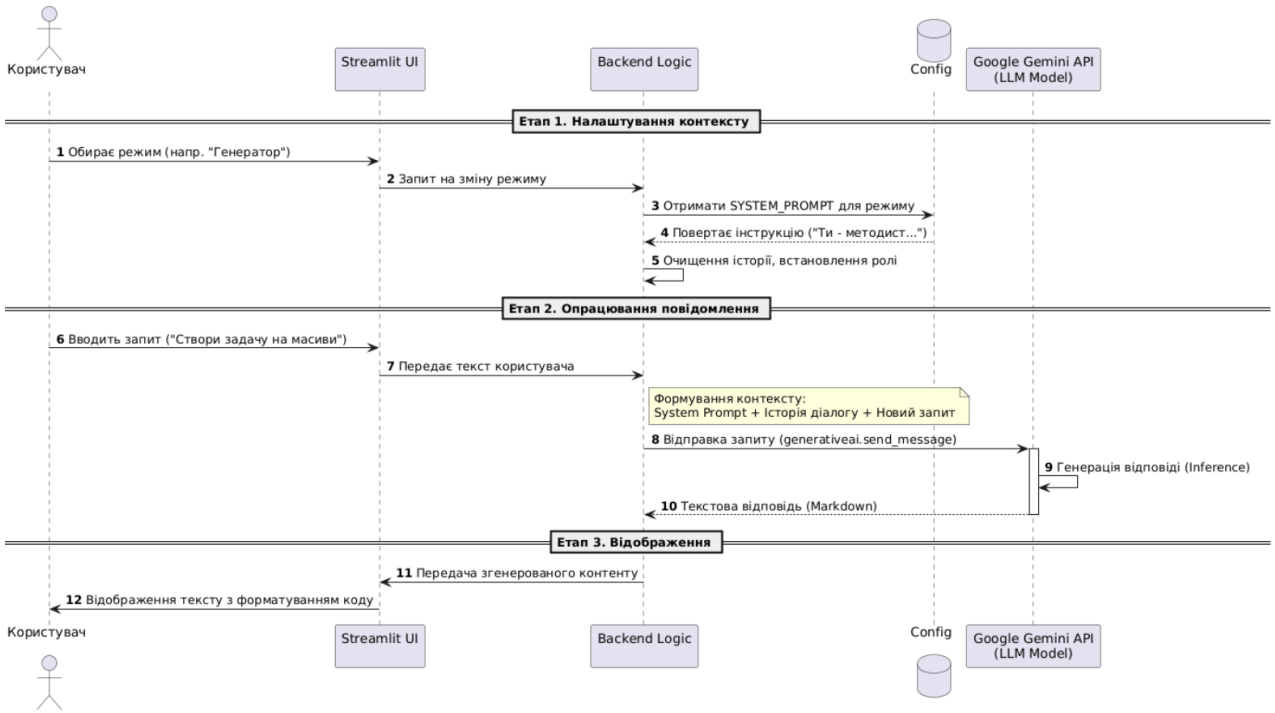


Рисунок. 2.2. Схема опрацювання запиту в системі AI Teacher Assistant

Питання безпеки та розгортання

Оскільки програмний комплекс орієнтований на використання в освітніх установах, його архітектура враховує ключові аспекти інформаційної безпеки та захисту даних. Система функціонує без збереження персональної інформації користувачів. Робота організована у безстанному режимі, коли історія діалогу існує лише в оперативній пам’яті браузера поточної сесії через механізми `st.session_state`, і не передається на зовнішні сервери.

Окрему увагу приділено захисту облікових даних доступу до моделей ШІ. API-ключі не інтегруються безпосередньо в програмний код, а зберігаються у файлі `.env`, який виключено з репозиторію системи контролю версій. Такий підхід мінімізує ризики компрометації ключів та відповідає сучасним рекомендаціям щодо безпечної розробки.

Запропонована архітектура створює умови для легкого масштабування. Заміна мовної моделі (наприклад, перехід від Gemini до GPT-4 або Claude) передбачає зміну лише параметрів у модулі адаптера API, не потребуючи модифікації інтерфейсу або бізнес-логіки системи.

2.1.3. Програмна реалізація компонентів системи

Програмна реалізація комплексу AI Teacher Assistant виконана мовою Python з дотриманням принципів модульності та чистоти коду. Вихідний код проєкту структуровано у три основні файли, кожен з яких відповідає за окремий рівень абстракції.

Реалізація семантичної логіки

Файл `prompts_config.py` виступає сховищем рольових моделей асистента. Для організації даних використано структуру даних словник (Dictionary), де ключем є ідентифікатор режиму (наприклад, `'generator'`), а значенням – текстова системна інструкція.

Такий підхід забезпечує відокремлення коду від даних: вчитель може редагувати методику (текст промпту), не втручаючись у програмний код.

Лістинг 2.1. Фрагмент файлу конфігурації промптів

```
SYSTEM_PROMPTS = {
    "grader": """
    ROLE: Ти суворий вчитель інформатики.
    TASK: Перевір код учня на наявність синтаксичних та логічних помилок.
    CRITERIA:
    - Працездатність алгоритму.
    - Стиль коду (PEP8 для Python, Google Style для C++).
    OUTPUT FORMAT: Оцінка (1-12), Список помилок, Рекомендації.
    LANGUAGE: Українська.
    """,
    # ... інші режими ...
}
```

Реалізація бізнес-логіки та інтерфейсу

Головний файл `app.py` реалізує життєвий цикл веб-застосунку на базі фреймворку Streamlit. Логіку роботи можна розділити на кілька етапів.

А. Керування станом сесії. Для збереження історії діалогу між перезавантаженнями сторінки використано механізм `st.session_state`. Це дозволяє асистенту пам'ятати контекст розмови.

Лістинг 2.2. Ініціалізація змінних сесії

```
# Перевірка наявності історії в пам'яті
if "messages" not in st.session_state:
    st.session_state.messages = []

# Збереження поточного режиму роботи
if "current_mode" not in st.session_state:
    st.session_state.current_mode = "tutor"
```

Б. Маршрутизація запитів. Бічна панель (`st.sidebar`) реалізує функцію перемикача режимів. При зміні режиму (наприклад, з «Тьютора» на Генератор) система виконує очищення історії повідомлень (`st.session_state.messages = []`) та перезавантажує інтерфейс (`st.rerun()`), щоб завантажити новий системний промпт.

В. Інтеграція з API та опрацювання помилок. Взаємодія з нейромережею реалізована через блок обробки виключень `try...except`, що забезпечує стабільність роботи програми навіть у випадку проблем зі з'єднанням.

Лістинг 2.3. Відправка запиту до моделі

```
try:
    system_instruction = SYSTEM_PROMPTS[st.session_state.current_mode]

    model = genai.GenerativeModel(
        model_name="gemini-2.0-flash",
        system_instruction=system_instruction
    )

    response = chat.send_message(prompt)

    st.markdown(response.text)

except Exception as e:
    st.error(f"Виникла помилка при зверненні до API: {e}")
```

Реалізація механізму безпеки.

Для захисту конфіденційних даних використано бібліотеку `python-dotenv`. Вона дозволяє завантажувати змінні середовища з файлу `.env` у пам'ять процесу під час запуску програми.

Лістинг 2.4. Завантаження API-ключа

```
from dotenv import load_dotenv
import os

load_dotenv()
api_key = os.getenv("GOOGLE_API_KEY")

if not api_key:
    st.stop()
```

Даний підхід унеможливлює випадкову публікацію ключа доступу при передачі коду третім особам або завантаженні в репозиторії (GitHub/GitLab), оскільки файл `.env` додається до списку ігнорування `.gitignore`.

2.1.4. Інженерія промптів як метод розробки педагогічного інструментарію

У рамках дослідження інженерія промптів розглядається не як ситуативна взаємодія з чат-ботом, а як систематичний метод «програмування природною мовою», що дозволяє детермінувати поведінку нейромережі та адаптувати її до освітніх потреб. Розробка лінгвістичного інтерфейсу системи AI Teacher Assistant базувалася на чітко структурованій архітектурі системних інструкцій та використанні просунутих технік керування генерацією.

Структурна організація системного промпту

Фундаментом коректної роботи кожного модуля системи є архітектура системного промпту, побудована за фреймворком *Role-Context-Constraints-Format*.

Насамперед, моделі задається чітка рольова ідентифікація. Для модуля Тьютор було визначено роль досвідченого ментора, який володіє методикою викладання інформатики, тоді як для модуля Екзаменатор роль зміщувалася у бік суворого рецензента коду, що дотримується стандартів індустрії. Таке перемикання дозволяє змінювати тональність та глибину відповідей залежно від педагогічної задачі.

Наступним критичним елементом є окреслення контексту. У системні інструкції було імплементовано інформацію про цільову аудиторію (учні 5-11 класів закладів загальної середньої освіти) та предметну область (мови програмування Python та C++). Це дозволяє моделі уникати використання надмірно складної університетської термінології та адаптувати пояснення до рівня шкільної програми НУШ.

Для забезпечення академічної доброчесності та педагогічної доцільності було впроваджено блок обмежень. Ключовою директивою для системи стала заборона на генерацію прямої відповіді на запит учня без попереднього пояснення. Система налаштована таким чином, щоб блокувати спроби отримання готового коду домашнього завдання, натомість пропонуючи алгоритмічні підказки.

Завершальним компонентом промпту є специфікація формату виводу. Для забезпечення читабельності згенерованого контенту модель було проінструктовано використовувати розмітку Markdown для блоків коду, виділяти ключові терміни жирним шрифтом та структурувати відповіді за допомогою заголовків, що значно покращує сприйняття інформації користувачем веб-інтерфейсу.

Застосування спеціалізованих технік промптингу

Для підвищення якості роботи системи, окрім базових інструкцій, було застосовано методи *Few-Shot Prompting* та *Chain-of-Thought (CoT).*

Техніка *Few-Shot Prompting* (навчання на прикладах) була використана для калібрування стилю відповідей. У тіло промпту було інтегровано пари "еталонний запит учня – ідеальна відповідь вчителя". Наприклад, для модуля

Генератор було надано приклади того, як саме має виглядати сюжетна задача: від формулювання легенди до оформлення тест-кейсів. Це дозволило мінімізувати вірогідність генерації неструктурованого тексту та забезпечило уніфікований вигляд усіх навчальних матеріалів.

Для розв'язання складних алгоритмічних задач було імплементовано техніку *Chain-of-Thought (CoT),* або «ланцюжок думок». Суть методу полягає у спонуканні моделі до покрокового міркування перед формулюванням фінальної відповіді. У системну інструкцію було додано директиву: *"Перед тим як надати рекомендацію щодо коду, проаналізуй його логіку крок за кроком"*. Такий підхід значно знижує рівень "галюцинацій" при аналізі коду, оскільки модель спочатку будує внутрішній логічний ланцюжок виявлення помилки (наприклад, вихід за межі масиву), і лише потім трансформує цей висновок у педагогічну пораду для учня.

Таким чином, комбінація чіткої структурної організації промпту та використання технік Few-Shot і CoT дозволила перетворити універсальну мовну модель на спеціалізований інструмент, здатний ефективно виконувати функції асистента вчителя інформатики.

2.1.5. Розробка сценаріїв використання компонентів системи

Проєктування взаємодії користувача з інтелектуальною системою базується на визначенні ключових сценаріїв використання, які покривають основні етапи педагогічного циклу: підготовку до заняття, створення дидактичних матеріалів та оцінювання результатів навчання. У розробленому програмному комплексі ці сценарії реалізовано через чотири спеціалізовані модулі.

Сценарій методичної підтримки (Модуль Методист)

Сценарій роботи модуля Методист орієнтований на автоматизацію рутинних процесів планування навчальної діяльності. У цьому режимі система виконує роль асистента-методиста, який генерує структуру уроку на основі вхідних метаданих, таких як тема, клас та тип заняття. Ключовою особливістю

даного сценарію є відповідність генерованого контенту вимогам державного стандарту та концепції Нової української школи.

Взаємодія відбувається за алгоритмом уточнення: спочатку вчитель задає загальну тему (наприклад, «Масиви»), після чого система формує деталізований план-конспект із розбивкою на етапи актуалізації, вивчення нового матеріалу та рефлексії. Важливим аспектом є здатність модуля адаптувати складність термінології відповідно до вікової групи учнів, забезпечуючи дидактичну доцільність запропонованих методів навчання.

Сценарій адаптивної генерації контенту (Модуль Генератор)

Функціонал модуля "Генератор завдань" реалізує сценарій створення багаторівневих практичних вправ з програмування. Цей сценарій дозволяє вчителю динамічно змінювати складність завдання шляхом модифікації промпту, реалізуючи принцип диференційованого навчання. Наприклад, отримавши базову задачу на використання умовного оператора, викладач може ініціювати ускладнення сценарію запитом на додавання нових технічних умов, таких як використання масивів або функцій.

Окрім технічної варіативності, сценарій передбачає інтеграцію елементів гейміфікації. Система здатна обгортати сухі алгоритмічні задачі в сюжетні легенди (наприклад, у сеттингу комп'ютерних ігор або наукової фантастики), що сприяє підвищенню мотивації учнів. Результатом роботи модуля є не лише текст задачі, а й набір верифікованих тест-кейсів (вхідних та вихідних даних), що значно спрощує подальшу перевірку розв'язків.

Сценарій інтелектуального рецензування коду (Модуль Екзаменатор)

Сценарій використання модуля Екзаменатор докорінно змінює підхід до перевірки учнівських робіт, зміщуючи акцент з простого виявлення синтаксичних помилок на глибокий аналіз логіки програми. У цьому режимі асистент емулює поведінку досвідченого розробника, який проводить аналіз коду.

Алгоритм роботи модуля налаштований таким чином, щоб уникати прямого виправлення коду (автокорекції), оскільки це знижує навчальний ефект.

Натомість система аналізує код на предмет логічних аномалій – таких як вихід за межі масиву, нескінченні цикли або нераціональне використання пам'яті – і формує педагогічний коментар. Цей коментар містить вказівку на локалізацію помилки та навідне пояснення її природи, що спонукає учня до самостійного пошуку шляхів виправлення та сприяє розвитку навичок відлагодження.

Сценарій інтерактивного тьюторингу (Модуль Тьютор)

Сценарій роботи модуля Тьютор спрямований на реалізацію концепції індивідуалізованого навчання та педагогічної підтримки у режимі реального часу. На відміну від стандартних чат-ботів, які функціонують у режимі "Запитання-Відповідь", даний модуль налаштований на ведення евристичної бесіди.

Технічна реалізація цього сценарію базується на жорстких обмеженнях системного промпту, які блокують генерацію готового програмного коду на прямий запит користувача. Алгоритм взаємодії передбачає, що при виявленні прогалини в знаннях учня (наприклад, нерозуміння роботи циклу `while`), система не надає правильне рішення миттєво. Натомість вона ініціює покроковий діалог, використовуючи Сократівський метод: ставить серію навідних запитань, які спонукають учня самостійно дійти до правильного висновку.

Додатковою особливістю сценарію є адаптація пояснень через метод аналогій. Система здатна трансформувати абстрактні поняття програмування (змінні, масиви, цикли) у зрозумілі образи з реального життя (наприклад, порівняння змінної з коробкою, а масиву – з потягом), що відповідає когнітивним особливостям учнів середнього шкільного віку та полегшує входження у предметну область.

2.2. Методичні засади використання програмного комплексу AI Teacher Assistant у навчанні інформатики

2.2.1. Методика автоматизованого проєктування уроку

Ефективність підготовки вчителя до навчальних занять напряму залежить від інструментарію, що використовується. Для апробації методики автоматизованого створення навчального контенту було використано модуль Методист розробленого програмного комплексу AI Teacher Assistant.

Процес проєктування уроку за допомогою системи реалізується через послідовність дій, описану нижче.

Етап 1. Ініціалізація та вибір режиму

На початку роботи у бічній панелі веб-інтерфейсу було обрано режим «Методист (План уроку)». При цьому система автоматично завантажила відповідний системний промпт (`SYSTEM_PROMPT['planner']`), який налаштовує нейромережу на дотримання структури уроку згідно з вимогами.

Етап 2. Формування запиту

Методика передбачає введення вчителем ключових метаданих уроку. Для експерименту було обрано тему з курсу інформатики 9-го класу, яка традиційно викликає труднощі в учнів – синтаксис циклічних конструкцій.

У поле для введення повідомлення було введено наступний запит (Рис.2.3):

"Тема: Цикли з лічильником (for) у мові Python. Клас: 9. Тип уроку: Засвоєння нових знань. Вимоги: пояснити синтаксис на прикладі виведення чисел, додати практичне завдання з написання коду."

Етап 3. Генерація та аналіз результату

Система опрацювала запит протягом 4 секунд та згенерувала структурований план-конспект (рис. 2.3). Повний конспект уроку знаходиться у додатку 1.

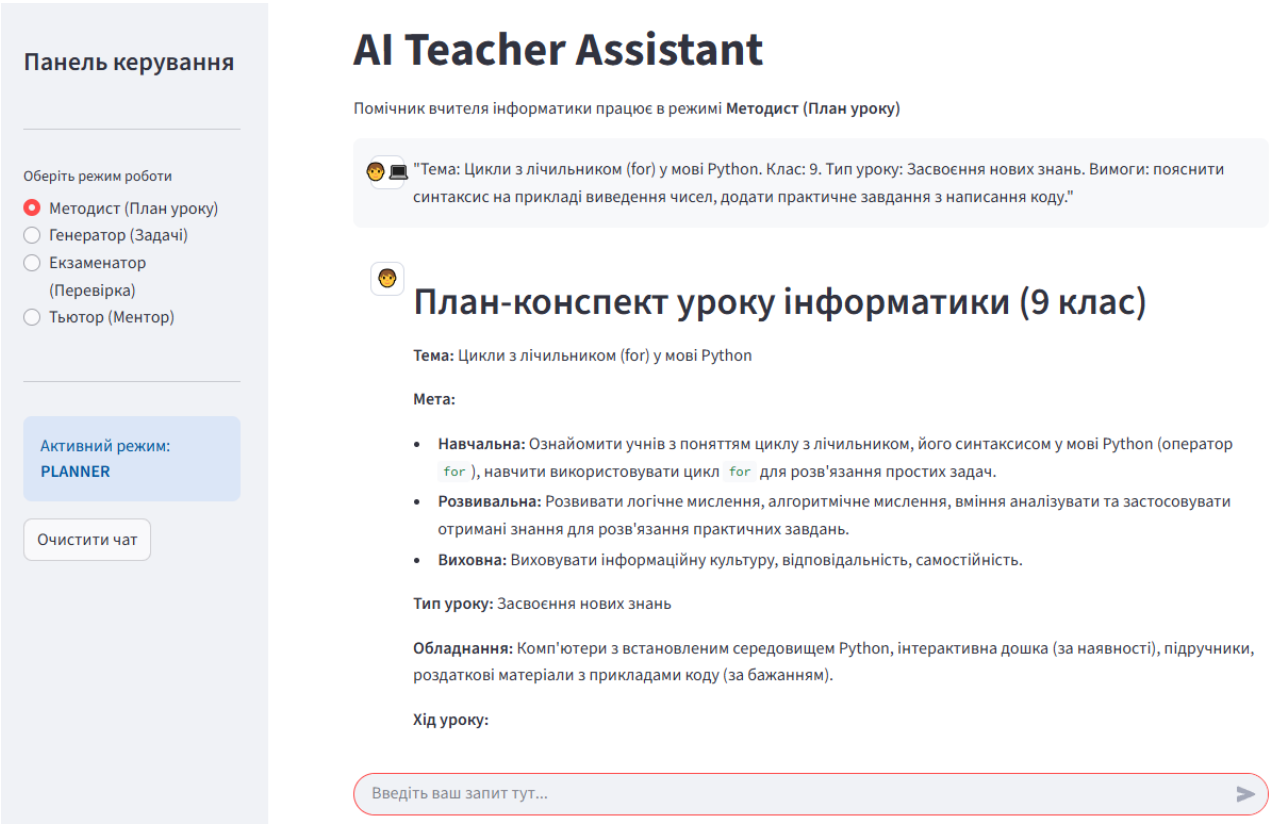


Рисунок 2.3. Інтерфейс модуля Методист під час генерації плану уроку

Отриманий план уроку містить такі структурні елементи:

Мета уроку: Сформульовано триєдину мету (навчальну, розвивальну, виховну). Зокрема, навчальна мета чітко орієнтована на результат: " Ознайомити учнів з поняттям циклу з лічильником, його синтаксисом у мові Python (оператор for), навчити використовувати цикл for для розв'язання простих задач".

Актуалізація опорних знань. Система запропонувала бесіду, де учням ставилися запитання Що таке алгоритм? Які основні алгоритмічні структури ви знаєте? Для чого потрібні цикли в програмуванні? Які типи циклів ви знаєте (можливо, з інших мов програмування)?

Викладення нового матеріалу. Асистент надав пояснення про поняття циклу з лічильником (for); синтаксис циклу for у Python; функцію range(). Навів приклади використання циклу for для виведення чисел.

Важливим методичним аспектом є те, що система запропонувала використати блок-схему для візуалізації процесу, що покращує сприйняття абстракції.

Практична частина. Згенеровано три завдання з наданими підказками:

Завдання 1: Напишіть програму, яка виводить на екран числа від 10 до 1 в зворотному порядку.

Завдання 2: Напишіть програму, яка обчислює суму чисел від 1 до заданого числа n. Число n повинно бути введене користувачем.

Завдання 3 (додаткове): Напишіть програму, яка виводить таблицю множення для числа 7.

Етап 4. Редагування та адаптація

Отриманий результат не є кінцевим продуктом, а слугує основою «чернеткою». Методика передбачає роль вчителя як редактора. Згенерований текст, завдяки форматуванню Markdown, легко копіюється у текстові редактори або LMS.

Використання модуля Методист дозволило скоротити час на технічну підготовку конспекту уроку з середніх 40 хвилин до 5 хвилин. При цьому змістове наповнення уроку відповідає чинній навчальній програмі, а запропоновані приклади коду є синтаксично правильними.

2.2.2. Диференціація навчання та гейміфікація завдань

Однією з ключових проблем шкільного навчання програмування, зокрема Python, є зниження зацікавленості учнів під час розв’язання абстрактних математичних задач. Типові приклади на зразок обчислення суми елементів списку майже не стимулюють пізнавальний інтерес підлітків, оскільки не мають для них очевидної практичної цінності.

Модуль Генератор системи *AI Teacher Assistant* розв’язує цю проблему шляхом автоматизованого впровадження двох ефективних методичних підходів. Перший базується на використанні сторітелінгу, коли алгоритмічна задача інтегрується у захопливий сюжет або контекст, що підсилює емоційну залученість учнів. Другий пов’язаний із рівневою диференціацією, тобто оперативним створенням варіантів задач різного рівня складності для індивідуалізації навчання.

Методика створення гейміфікованих завдань

Використання асистента дозволяє вчителю за лічені секунди перетворити суху тему на міні-квест. Для експериментальної перевірки було обрано тему «Списки та цикли `for` у Python» (7-8 клас).

Алгоритм роботи вчителя:

1. Вибір режиму Генератор.

2. Введення запиту з обов'язковим зазначенням тематики «сеттингу», яка цікава учням (Minecraft, Roblox, Космос, Супергерої).

Запит вчителя: *"Тема: Пошук максимального елемента в списку. Мова: Python. Рівень: Середній. Легенда: Аналіз даних з марсохода."*

Результат роботи модуля Генератор за запитом вчителя наведено на рис. 2.4.

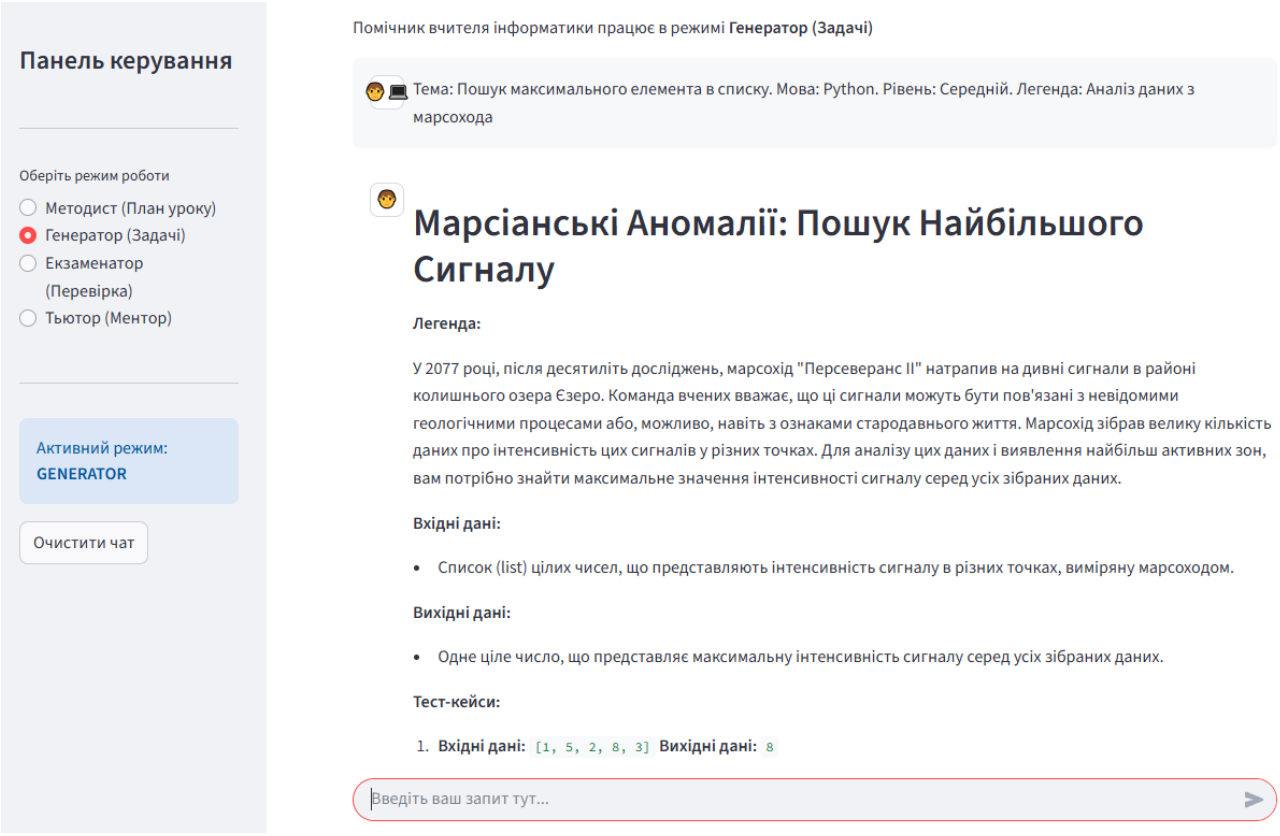


Рисунок 2.4. Інтерфейс модуля Генератор з гейміфікованою задачею

Аналіз згенерованих матеріалів дає змогу окреслити низку методичних переваг, які підсилюють ефективність використання модуля Генератор у навчанні програмування.

Першою перевагою є контекстуалізація завдань. Замість традиційного формулювання на кшталт «знайти максимальне значення в масиві» система інтегрує задачу у змістовний сюжет. У наведеному прикладі використано легенду про марсохід «Персеверанс II» та аналіз сигналів у кратері Єзеро, що надає вправі прикладного характеру та істотно підвищує мотивацію учнів. Такий підхід демонструє практичне застосування алгоритмів опрацювання даних у наукових дослідженнях.

Другою перевагою є технічна коректність сформованих тестових прикладів. Особливу увагу привертає тест-кейс, що містить набір від'ємних чисел. Його включення відображає здатність системи враховувати типові помилки початківців. Учні часто ініціалізують змінну `max = 0`, що спричиняє неправильний результат у разі роботи з від'ємними значеннями. Наявність такого прикладу спонукає застосовувати правильний підхід – ініціалізацію першим елементом списку – на що додатково вказує згенерована системою підказка.

Третьою важливою характеристикою є структурованість подання вимог. Чітке розмежування між блоками «Вхідні дані» та «Вихідні дані» формує в учнів навичку працювати з форматом, прийнятим на олімпіадних платформах і в професійному програмуванні.

Методика автоматизованої диференціації в межах системи AI Teacher Assistant орієнтована на реалізацію індивідуального підходу до навчання програмування шляхом динамічного формування задач різної складності в межах однієї теми. Модель генерує три варіанти складності, адаптуючи формулювання під різні рівні підготовки учнів.

На початковому рівні вона трансформує завдання у спрощену форму: зменшує обсяг вхідних даних, дозволяє застосування готових функцій Python або спрощує логічну конструкцію. На середньому рівні система формує стандартні

алгоритмічні задачі на опрацювання списків, обчислення сум і середніх значень, лінійний пошук та використання базових операторів циклів і розгалужень. Високий рівень передбачає ускладнення умов через введення додаткових обмежень або оптимізаційних критеріїв, наприклад потребу визначення трійки найбільших значень чи виконання сортування даних.

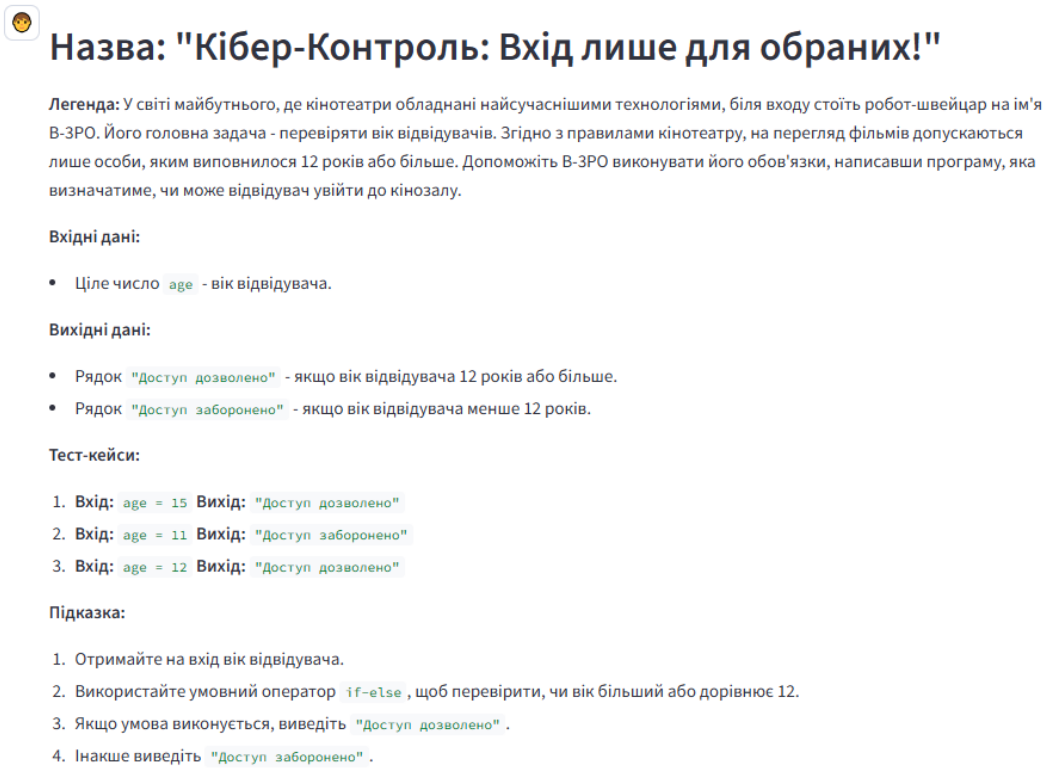
Важливою складовою роботи модуля є автоматизоване створення тест-кейсів, що виступає елементом внутрішнього QA-тестування. Система не лише формує текст умови, а й генерує таблицю прикладів вхідних і вихідних даних, яка дозволяє учням перевірити коректність власного розв’язку та забезпечує методичну прозорість у процесі навчання.

Для підтвердження здатності системи забезпечувати рівневу диференціацію, було проведено експеримент з генерації завдань різної складності для однієї теми «Умовні оператори».

Серія 1. Умовні оператори (`if-elif-else`). Початковий рівень.

Запит: *"Тема: Умовний оператор if-else. Мова: Python. Рівень: Початковий. Легенда: Робот-швейцар у кінотеатрі, який перевіряє вік відвідувача (12+)."*

На рисунку 2.5 показано задачу початкового рівня, яка вимагає від учня використання базової конструкції `if-else`.



Назва: "Кібер-Контроль: Вхід лише для обраних!"

Легенда: У світі майбутнього, де кінотеатри обладнані найсучаснішими технологіями, біля входу стоїть робот-швейцар на ім'я B-3PO. Його головна задача - перевіряти вік відвідувачів. Згідно з правилами кінотеатру, на перегляд фільмів допускаються лише особи, яким виповнилося 12 років або більше. Допоможіть B-3PO виконувати його обов'язки, написавши програму, яка визначатиме, чи може відвідувач увійти до кінозалу.

Вхідні дані:

- Ціле число `age` - вік відвідувача.

Вихідні дані:

- Рядок `"Доступ дозволено"` - якщо вік відвідувача 12 років або більше.
- Рядок `"Доступ заборонено"` - якщо вік відвідувача менше 12 років.

Тест-кейси:

1. **Вхід:** `age = 15` **Вихід:** `"Доступ дозволено"`
2. **Вхід:** `age = 11` **Вихід:** `"Доступ заборонено"`
3. **Вхід:** `age = 12` **Вихід:** `"Доступ дозволено"`

Підказка:

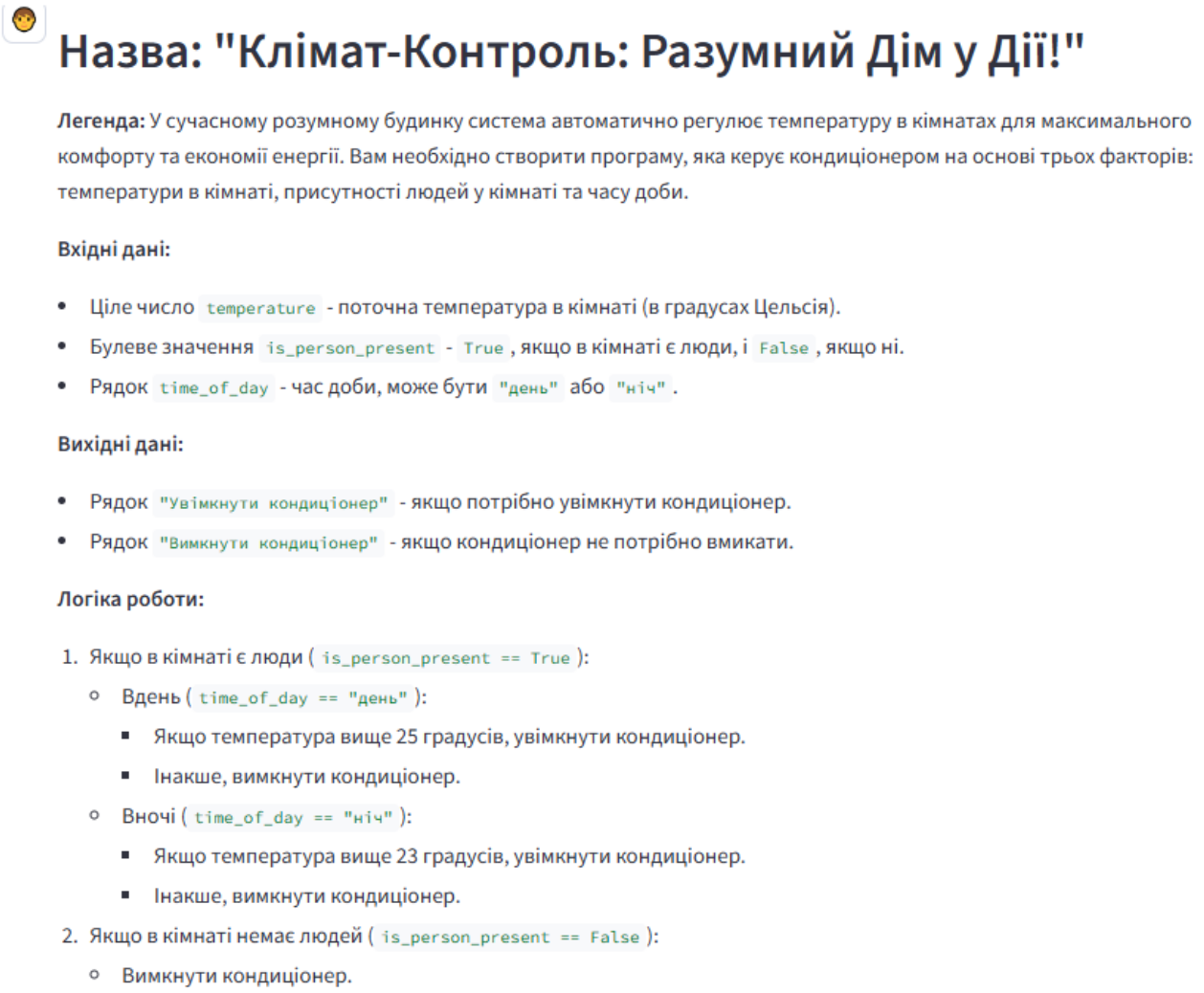
1. Отримайте на вхід вік відвідувача.
2. Використайте умовний оператор `if-else`, щоб перевірити, чи вік більший або дорівнює 12.
3. Якщо умова виконується, виведіть `"Доступ дозволено"`.
4. Інакше виведіть `"Доступ заборонено"`.

Рисунок 2.5. Завдання початкового рівня умовні оператори (`if-elif-else`)

Серія 1. Умовні оператори (`if-elif-else`). Високий рівень

Запит: *"Тема: Вкладені умовні оператори. Мова: Python. Рівень: Високий. Легенда: Система розумного будинку. Керування кондиціонером залежно від температури, присутності людей у кімнаті та часу доби."*

Натомість на рисунку 2.6 система за запитом «Високий рівень» згенерувала задачу підвищеної складності (Розумний будинок).



Назва: "Клімат-Контроль: Разумний Дім у Дії!"

Легенда: У сучасному розумному будинку система автоматично регулює температуру в кімнатах для максимального комфорту та економії енергії. Вам необхідно створити програму, яка керує кондиціонером на основі трьох факторів: температури в кімнаті, присутності людей у кімнаті та часу доби.

Вхідні дані:

- Ціле число `temperature` - поточна температура в кімнаті (в градусах Цельсія).
- Булеве значення `is_person_present` - `True`, якщо в кімнаті є люди, і `False`, якщо ні.
- Рядок `time_of_day` - час доби, може бути `"день"` або `"ніч"`.

Вихідні дані:

- Рядок `"Увімкнути кондиціонер"` - якщо потрібно увімкнути кондиціонер.
- Рядок `"Вимкнути кондиціонер"` - якщо кондиціонер не потрібно вмикати.

Логіка роботи:

1. Якщо в кімнаті є люди (`is_person_present == True`):
   - Вдень (`time_of_day == "день"`):
     - Якщо температура вище 25 градусів, увімкнути кондиціонер.
     - Інакше, вимкнути кондиціонер.
   - Вночі (`time_of_day == "ніч"`):
     - Якщо температура вище 23 градусів, увімкнути кондиціонер.
     - Інакше, вимкнути кондиціонер.
2. Якщо в кімнаті немає людей (`is_person_present == False`):
   - Вимкнути кондиціонер.

Рисунок 2.6. Завдання високого рівня умовні оператори (`if-elif-else`)

Для високого рівня асистент ускладнив вхідні дані (потрібно опрацювати три змінні замість однієї). Вимагає використання логічних операторів and/or. Згенерував більш комплексні тест-кейси, що покривають різні сценарії поведінки програми.

Серія 2. Цикли (`while`). Початковий рівень

Запит: *"Тема: Цикл while. Мова: Python. Рівень: Початковий. Легенда: Запуск ракети SpaceX, зворотний відлік від 10 до 0."*

Ось задача на тему циклу `while` мовою Python для початківців:

Назва: "Зворотний відлік до старту SpaceX"

Легенда: Уявіть себе в центрі управління польотами SpaceX. Залишилися лічені секунди до запуску нової ракети! Ваша задача - написати програму, яка імітує зворотний відлік часу від 10 до 0, а потім виведе повідомлення про старт ракети. Від вашої точності залежить успіх місії!

Вхідні дані: Відсутні.

Вихідні дані: Вивід у консоль чисел від 10 до 0 (кожне число на новому рядку), а потім повідомлення "Ракета полетіла!".

Рисунок 2.7. Завдання початкового рівня цикли (`while`).

Разом з тим система вивела можливі тест-кейси з очікуваним виводом та неправильними виводами (Рис. 2.8-2.10) та підказки (рис. 2.11)

Тест-кейси:

1. **Очікуваний вивід:**

```
10
9
8
7
6
5
4
3
2
1
0
Ракета полетіла!
```

Рисунок 2.8. Тест-кейс «Очікуваний вивід»

2. **Неправильний вивід (приклад):**

```
10
9
8
7
6
5
4
3
2
1
Ракета полетіла!
```

Рисунок 2.9. Тест-кейс «Неправильний вивід»

3. **Неправильний вивід (приклад):**

```
11
10
9
8
7
6
5
4
3
2
1
0
Ракета полетіла!
```

Рисунок 2.10. Тест-кейс «Неправильний вивід»

Підказка:

1. Ініціалізуйте змінну (наприклад, `count`) значенням 10.
2. Використовуйте цикл `while`, щоб повторювати дії до тих пір, поки значення змінної `count` більше або дорівнює 0.
3. Всередині циклу:
   - Виведіть значення `count` на екран.
   - Зменште значення `count` на 1.
4. Після завершення циклу (коли `count` стане від'ємним) виведіть повідомлення "Ракета полетіла!".

Рисунок 2.11. Підказка

Серія 2. Цикли (`while`). Високий рівень (Олімпіадний тип).

Запит: *"Тема: Цикл while з перериванням. Мова: Python. Рівень: Високий. Легенда: Банківський вклад. Порахувати, через скільки років сума вкладу подвоїться під 10% річних."*

Як і в попередніх варіантах система сформувала легенду, де закладено завдання для виконання вказаної теми та рівня складності (Рис. 2.12), також додала тест-кейси з вхідними даними та очікуваними результатами (Рис. 2.13) та сформувала підказки (Рис. 2.14).

Ось задача на тему циклу `while` з перериванням мовою Python для більш високого рівня:

Назва: "Подвоєння капіталу: Фінансовий прорив"

Легенда: Ви - талановитий фінансовий аналітик. Клієнт звернувся до вас з питанням: через скільки років його банківський вклад подвоїться, якщо річна процентна ставка становить 10%? Допоможіть клієнту прийняти обґрунтоване рішення, розрахувавши необхідний термін.

Вхідні дані:

- `початковий_вклад` (float): Сума початкового вкладу (більше 0).

Вихідні дані:

- Кількість років (int), необхідних для подвоєння вкладу.
- Якщо вклад ніколи не подвоїться (наприклад, при від'ємній процентній ставці або нульовому початковому вкладі), вивести повідомлення "Неможливо розрахувати".

Рисунок 2.12. Завдання високого рівня цикли (`while`).

Тест-кейси:

1. **Вхідні дані:** `початковий_вклад = 100` **Очікуваний вивід:** `8` (Приблизно через 8 років вклад подвоїться)
2. **Вхідні дані:** `початковий_вклад = 1000` **Очікуваний вивід:** `8` (Початкова сума не впливає на час подвоєння при фіксованій ставці)
3. **Вхідні дані:** `початковий_вклад = 0` **Очікуваний вивід:** `"Неможливо розрахувати"`

Рисунок 2.13. Тест-кейси

Підказка:

1. Перевірте коректність вхідних даних (початковий вклад повинен бути більше 0). Якщо дані некоректні, виведіть повідомлення про помилку і завершіть програму.
2. Ініціалізуйте змінну для поточної суми вкладу початковим значенням вкладу.
3. Ініціалізуйте змінну для кількості років значенням 0.
4. Використовуйте цикл `while`, який буде виконуватися, поки поточна сума вкладу менша за подвоєний початковий вклад.
5. Всередині циклу:
   - Обчисліть відсотки за рік (10% від поточної суми).
   - Додайте відсотки до поточної суми вкладу.
   - Збільште кількість років на 1.
6. Після завершення циклу виведіть кількість років.

Рисунок 2.14. Підказки

Модуль Генератор не лише формує текст задачі, а й виступає ефективним методичним інструментом, який враховує особливості початкової алгоритмічної підготовки учнів та сприяє розвитку їхнього критичного мислення.

2.2.3. Використання ШІ як інтелектуального тьютора в процесі вивчення мов програмування

Традиційні середовища розробки (IDE) здатні виявляти лише синтаксичні помилки. Натомість логічні помилки, які є найбільш поширеними серед початківців, часто залишаються непоміченими до моменту запуску програми.

Модуль Екзаменатор системи AI Teacher Assistant виконує функцію інтелектуального статичного аналізатора коду. Для перевірки методики було змодельовано типову ситуацію: учень 8-го класу написав код пошуку максимального числа, ініціалізувавши змінну `max_val` нулем, що є помилкою при роботі з від'ємними числами.

Результат роботи модуля наведено на рисунку 2.15.

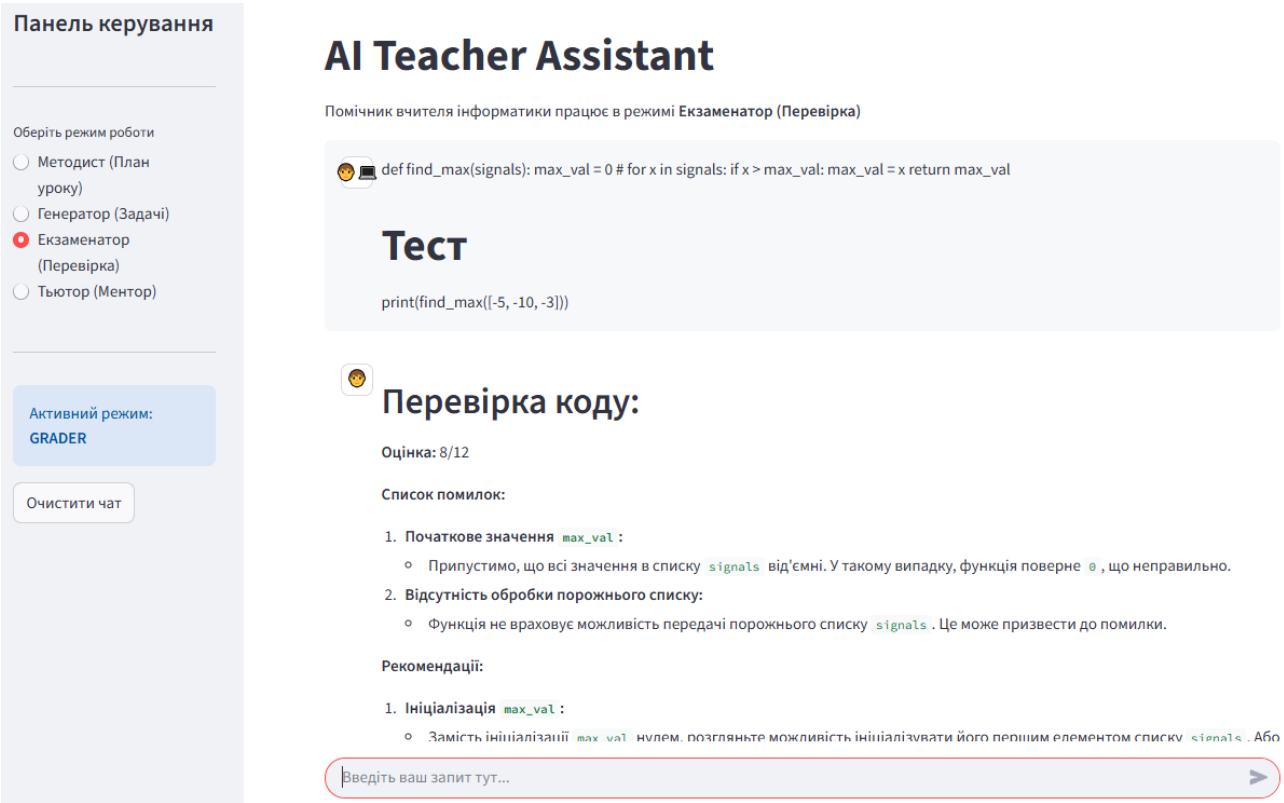


Рисунок 2.15. Результат аналізу коду модулем Екзаменатор – виявлення логічної помилки

Аналіз отриманої відповіді демонструє високий методичний потенціал системи.

Система чітко вказала на помилку: *"Припустимо, що всі значення в списку signals від'ємні. У такому випадку функція поверне 0, що неправильно"*. Це свідчить про глибоке розуміння контексту задачі, а не просте порівняння з шаблоном.

Замість простої констатації "Неправильно", асистент виставив оцінку 8/12, мотивуючи це тим, що алгоритм в цілому вірний, але має критичний недолік. Це знижує стрес учня і показує, що його зусилля оцінені.

Асистент запропонував "Pythonic way" вирішення проблеми:

– Ініціалізувати змінну першим елементом списку.

– Або використати значення `float('-inf')` (мінус нескінченність) – професійний прийом, про який часто не пишуть у шкільних підручниках.

Система звернула увагу на крайовий випадок – "Відсутність обробки порожнього списку", привчаючи учня писати надійний код.

Таким чином, модуль Екзаменатор дозволяє реалізувати методику перегляду коду на кожному уроці, забезпечуючи кожного учня миттєвим та деталізованим зворотним зв'язком, що фізично неможливо для вчителя у класі з 15-20 учнями.

2.2.4. Методика індивідуального супроводу учня

Модуль Тьютор розроблено для забезпечення персоналізованої підтримки учня в процесі розв'язання задачі. Головна методична особливість цього модуля – заборона на надання готових відповідей. Асистент налаштований діяти в зоні найближчого розвитку учня, використовуючи техніку «скаффолдингу» (педагогічної підтримки).

Етап 1. Ініціалізація менторського режиму

Учень або вчитель обирає режим «Тьютор (Ментор)». Система завантажує інструкцію, яка обмежує генеративні можливості моделі: замість написання коду вона переходить у режим діалогу.

Етап 2. Запит на "легке рішення"

Для перевірки стійкості педагогічних налаштувань системи було змодельовано типовий запит "лінивого учня", який хоче отримати домашнє завдання без зусиль.

Запит учня: *"Я не розумію цикл while у Python. Напиши мені код, який виводить числа від 1 до 10, бо мені треба здати задачу."*

Етап 3. Педагогічна реакція системи

Замість генерації коду (що зробив би звичайний ChatGPT), модуль Тьютор аналізує запит і виявляє прогалину в знаннях. Результат діалогу з системою у режимі Тьютор наведено на рисунку 2.16.

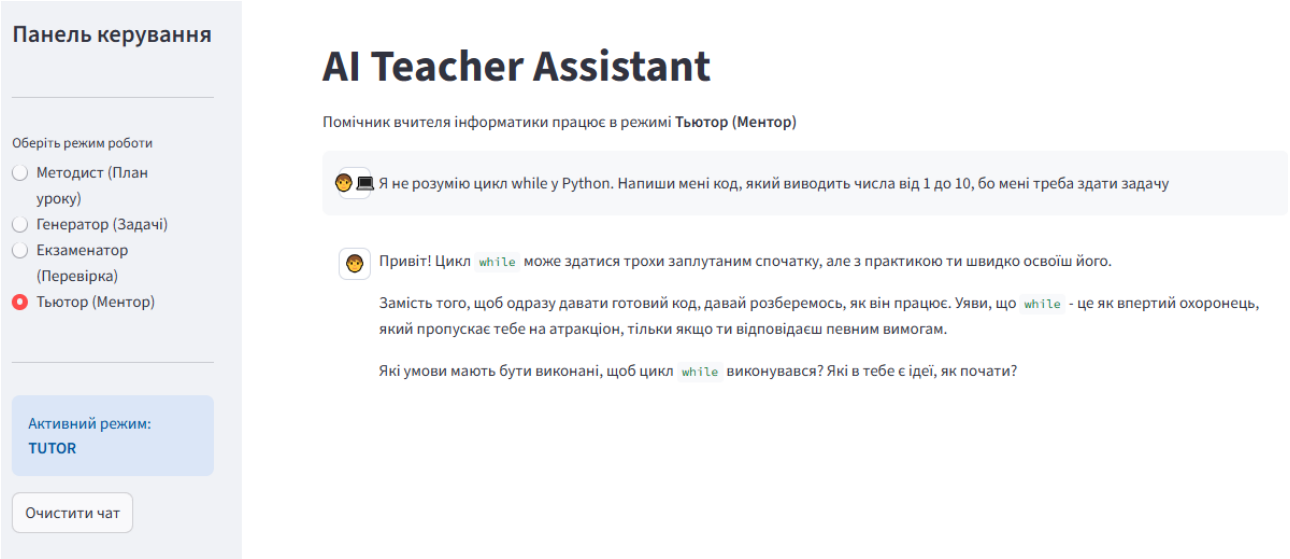


Рисунок 2.16. Реалізація Сократівського методу модулем Тьютор

Система ідентифікувала запит як спробу уникнути самостійної роботи (мені треба здати задачу) і заблокувала генерацію готового рішення. Для пояснення абстрактного поняття (умова циклу) було використано зрозумілий учням образ "впертого охоронця", що відповідає віковим особливостям сприйняття підлітків. Діалог завершується відкритим запитанням, що спонукає учня сформулювати власну гіпотезу, перетворюючи його з пасивного споживача

інформації на активного учасника навчального процесу. Продовження діалогу на рисунку 2.17.

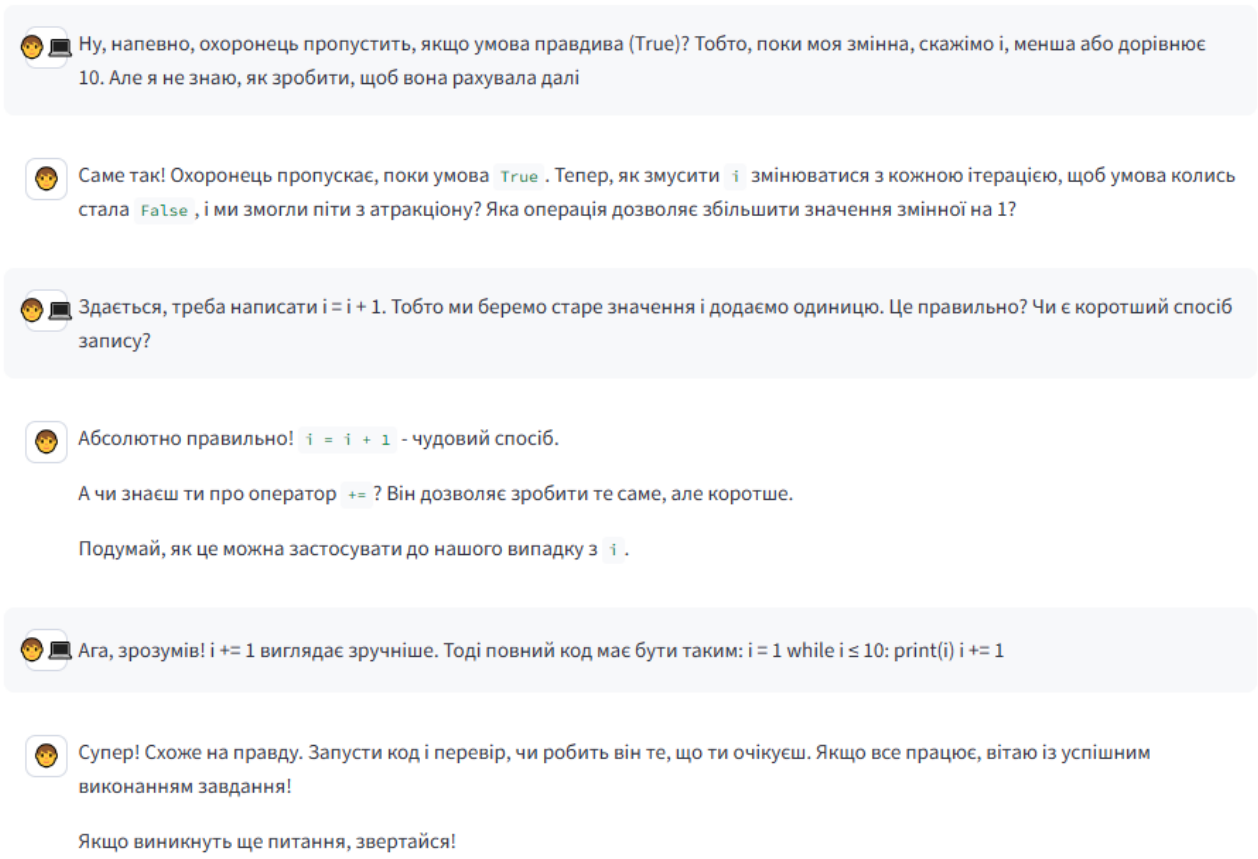


Рисунок 2.17. Продовження діалогу в модулі Тьютор

Таким чином, експериментальна перевірка підтвердила, що програмний комплекс AI Teacher Assistant здатний виконувати функції методиста, генератора дидактичних матеріалів та персонального тьютора, забезпечуючи високий рівень методичної підтримки вчителя та учнів.

Етап 4. Спільне конструювання знань

Учень відповідає на питання асистента, поступово формуючи правильний код. Тьютор підтверджує правильні кроки (*Чудово, умова правильна!*) і коригує помилкові (*Зверни увагу, ти забув збільшувати лічильник, це призведе до вічного циклу*).

Така методика дозволяє вчителю делегувати рутинне пояснення базових концепцій штучному інтелекту, залишаючи за собою роль фасилітатора складних дискусій.

Розроблена методика використання програмного комплексу AI Teacher Assistant охоплює всі етапи навчального процесу: від автоматизованої підготовки вчителя (модуль Методист) до створення мотиваційних завдань (Генератор) та індивідуального супроводу учнів (Тьютор, Екзаменатор). Практична апробація компонентів системи підтвердила її здатність генерувати методично доцільний контент та виявляти складні логічні помилки в коді учнів.

2.3. Оцінка ефективності використання інтелектуального асистента вчителя інформатики

2.3.1. Організація та методика проведення експерименту

Для перевірки ефективності розробленого програмного комплексу AI Teacher Assistant та запропонованої методики його використання було організовано педагогічний експеримент. Дослідно-експериментальна робота проводилася на базі Кричевичівського ліцею Колодяжненської сільської ради Ковельського району Волинської області.

У експерименті взяли участь учні 9-го класу, які вивчали тему «Алгоритми та програми» (мова програмування Python). Відповідно до вимог щодо організації навчального процесу та поділу класів на підгрупи під час проведення практичних занять з інформатики, дослідження проводилося у двох підгрупах одного класу. Загальна кількість учасників склала 18 осіб.

Для забезпечення об'єктивності результатів було сформовано дві групи.

Контрольна група (КГ) – підгрупа 9 учнів. Навчання здійснювалося за традиційною методикою: фронтальне пояснення матеріалу, виконання вправ з підручника, ручна перевірка коду вчителем.

Експериментальна група (ЕГ) – підгрупа 9 учнів. Навчальний процес будувався з використанням розробленого програмного комплексу: вчитель використовував модулі Методист та Генератор для підготовки контенту, а учні працювали з модулями Тьютор та Екзаменатор.

Експериментальна робота була організована у три послідовні етапи: констатувальний, формувальний і контрольний.

На першому етапі здійснювався діагностичний зріз, спрямований на визначення базового рівня навчальних досягнень учнів і попередньої мотивації до вивчення програмування. Учасники виконали тестування на засвоєння ключових понять, зокрема змінних та типів даних. Отримані результати засвідчили відсутність статистично значущих відмінностей між контрольної та експериментальною групами (середні бали становили відповідно 8,8 і 8,9), що підтвердило еквівалентність умов на старті дослідження.

Другий етап мав формувальний характер і був присвячений упровадженню розробленої методики та використанню програмного інструментарію в освітньому процесі. Він тривав упродовж чотирьох тижнів і відповідав календарному плану опрацювання тем «Циклічні алгоритми» та «Опрацювання списків». У цей період учні експериментальної групи навчалися із залученням автоматизованих методичних засобів, тоді як учні контрольної групи працювали за традиційними підходами.

Заключний, контрольний етап був спрямований на оцінювання ефективності застосованої методики. Для цього було проведено підсумкове тестування та анкетування обох груп. Порівняння отриманих результатів дало змогу визначити вплив використаних інструментів на рівень навчальних досягнень і ставлення учнів до вивчення програмування.

2.3.2. Аналіз динаміки навчальних досягнень учнів

На констатувальному етапі було проведено вхідне тестування для визначення початкового рівня знань учнів (знання базових операторів, типів даних). Результати показали, що обидві групи мали високий рівень підготовки, що свідчить про однорідність вибірки.

Після завершення вивчення теми було проведено підсумкове оцінювання (написання коду та розв'язування алгоритмічних задач). Порівняльний аналіз індивідуальних результатів учнів наведено в таблицях 2.2 та 2.3.

Таблиця 2.2.

Динаміка навчальних досягнень учнів Контрольної групи

Учень	Початковий бал (Констатувальний етап)	Підсумковий бал (Контрольний етап)	Динаміка
1	8	9	+1
2	9	9	0
3	10	10	0
4	8	8	0
5	9	10	+1
6	9	9	0
7	8	9	+1
8	10	11	+1
9	9	10	+1
Середній бал	8.9	9.4	+0.5

Аналіз результатів контрольної групи засвідчив відносну стабільність навчальних показників. Середній бал підвищився лише на 0,5 пункту, що вказує на мінімальний приріст. Подібна динаміка є типовою для традиційної моделі навчання, у межах якої педагог обмежений у можливості оперативно реагувати на індивідуальні труднощі кожного учня під час виконання практичних завдань. Відсутність швидкого та адресного зворотного зв’язку сприяє закріпленню повторюваних логічних помилок, що, своєю чергою, стримує зростання навчальних результатів.

Таблиця 2.3.

Динаміка навчальних досягнень учнів Експериментальної групи

Учень	Початковий бал (Констатувальний етап)	Підсумковий бал (Контрольний етап)	Динаміка
1	7	9	+2
2	8	10	+2
3	9	11	+2
4	10	12	+2
5	8	10	+2
6	9	10	+1
7	8	10	+2
8	11	12	+1
9	9	11	+2
Середній бал	8.8	10.5	+1.7

Аналіз результатів експериментальної групи засвідчив виразне покращення навчальних досягнень. Середній бал зріс на 1,7 бали, що значно

перевищує динаміку контрольної групи та свідчить про ефективність запропонованої методики. Найбільший прогрес продемонстрували учні з початковим рівнем 7-8 балів. Застосування модуля Тьютор дало їм змогу самостійно опрацювати складні програмні конструкції й підвищити результати до 10 балів. Учні з високим навчальним потенціалом також продемонстрували максимальні показники, досягнувши 12 балів. Їхній успіх пов'язаний із виконанням завдань підвищеної складності, що були згенеровані інтелектуальною системою та забезпечували оптимальний рівень виклику.

У процесі експерименту було встановлено, що застосування гейміфікованих завдань, згенерованих інтелектуальною системою, істотно підвищило рівень залученості учнів експериментальної групи. На відміну від контрольної групи, де значна частина здобувачів освіти виконувала завдання формально, учні ЕГ проявляли зацікавленість у проходженні сюжетної лінії задач, що додатково стимулювало мотивацію та сприяло активнішому залученню до навчального процесу.

Важливу роль відіграв модуль Тьютор. Здобувачі з недостатнім рівнем підготовки мали можливість ставити уточнювальні питання в анонімному форматі, що усувало психологічний бар'єр страху помилитися або поставити «несуттєве» запитання. Це сприяло вирівнюванню рівня навчальних досягнень у межах класу та забезпечило більш рівномірне засвоєння складних тем.

Після завершення формувального етапу було проведено підсумкове оцінювання здобувачів освіти обох груп за темою «Складні алгоритмічні структури». Для аналізу використовувалися середні показники успішності за 12-бальною шкалою, що дало можливість об’єктивно порівняти ефективність традиційної та інноваційної методик.

Динаміку змін навчальних досягнень учнів наведено в Таблиці 2.4 та на рисунку 2.18

Таблиця 2.4.

Порівняльна характеристика успішності учнів КГ та ЕГ

Група	Кількість учнів	Середній бал (Констатувальний етап)	Середній бал (Контрольний етап)	Динаміка приросту
Контрольна (КГ)	9	8.9	9.4	+0.5
Експериментальна (ЕГ)	9	8.8	10.5	+1.7

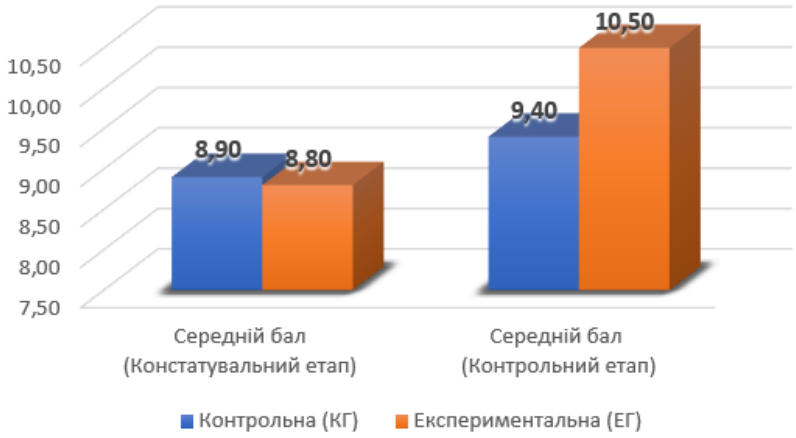


Рисунок 2.18. Порівняння динаміки успішності в КГ та ЕГ

Аналіз отриманих даних дозволяє сформулювати низку ключових висновків щодо ефективності впровадженої методики.

На констатувальному етапі учні обох груп продемонстрували практично ідентичний рівень навчальних досягнень (різниця становила лише 0,1 бала), що відповідає «Достатньому» рівню. Така однорідність зумовлена специфікою підгрупи з інформатики, де навчаються здебільшого вмотивовані учні з базовими навичками роботи з комп’ютером.

Середній приріст склав 0,5 бала, що відображає природний поступ у засвоєнні навчального матеріалу за умов традиційної методики. Проте більшість учнів залишилася в межах «Достатнього» рівня (7–9 балів). Основним чинником, який стримував перехід на «Високий» рівень, стали типові логічні помилки у програмному коді. Через обмеженість часу на уроці вчитель не завжди мав можливість надати індивідуальний покроковий аналіз допущених помилок, що призводило до їх повторного відтворення.

Приріст у 1,7 бала засвідчив помітно вищу ефективність розробленої методики. Завдяки такому зростанню група досягла стійкого «Високого» рівня навчальних досягнень (10-12 балів), що статистично підтверджує значущий вплив інноваційного підходу.

Ключовим фактором підвищення результатів в ЕГ стало використання модуля Екзаменатор. Учні мали можливість здійснювати попередню перевірку власного програмного коду перед поданням учителю. Система виявляла логічні недоліки, надлишкові операції, неефективні конструкції та невдалі ідентифікатори, що сприяло виробленню навичок чистого й оптимізованого кодування. У результаті учні подавали вчителю відразу коректні роботи, що позначалося на вищих оцінках.

Разом із тим використання модуля Генератор забезпечило диференціацію завдань. Сильні учні отримували задачі підвищеної складності, що підтримувало їхній когнітивний розвиток і запобігало втраті інтересу при виконанні стандартних вправ. Таким чином, система одночасно створювала умови як для вирівнювання рівня слабших учнів, так і для розвитку високої академічної результативності в учнів із підвищеним потенціалом.

2.3.3. Результати експертного оцінювання системи вчителями

Для валідації отриманих результатів та оцінки ергономічності розробленого програмного забезпечення було проведено анкетування серед вчителів інформатики навчального закладу, які ознайомилися з роботою системи. Оцінювання проводилося за 5-бальною шкалою Лайкерта (1 – низька ефективність, 5 – висока ефективність).

Текст анкети для вчителів включав ряд питань, зокрема:

1. Оцініть ефективність модуля Методист.

2. Оцініть якість дидактичних матеріалів модуля Генератор.

3. Оцініть корисність модуля Екзаменатор.

4. Як ви оцінюєте вплив гейміфікованих завдань (історії, легенди) на залученість учнів у навчальний процес?

5. Чи готові Ви використовувати цей інструмент у своїй подальшій практиці?

Результати анкетування вчителів описано в таблиці 2.5 та на рисунку 2.19.

Таблиця 2.5.

Результати анкетування вчителів

Критерій оцінювання	Середній бал	Інтерпретація результату
Економія часу вчителя	4.8	Вчителі відзначили, що автоматична генерація планів уроків є найбільш корисною функцією для розвантаження педагога.
Якість та варіативність завдань	4.7	Високо оцінено можливість створення сюжетних задач та тест-кейсів, яких немає у стандартних підручниках.
Допомога у перевірці коду	4.5	Попередня перевірка коду штучним інтелектом дозволяє вчителю зосередитись на логіці алгоритмів, а не на синтаксисі.
Підвищення мотивації учнів	5.0	Абсолютна більшість вчителів зауважила зростання інтересу учнів до предмету.
Готовність до впровадження	5.0	Усі респонденти виявили бажання використовувати інструмент у подальшій роботі.

Опрацювання відкритих відповідей анкети дало змогу виокремити ключові переваги системи AI Teacher Assistant з позиції освітянської спільноти. Найчастіше респонденти наголошували на зменшенні рутинного навантаження. Близько 80 % учасників назвали найбільш корисним модуль Методист, оскільки він дає можливість оперативно формувати структуру уроку та адаптувати її до вимог Нової української школи. Один із педагогів зазначив, що створення повноцінного плану, на яке раніше витрачалася година, тепер займає лише кілька хвилин, що демонструє значну економію часу та підвищення ефективності підготовки.

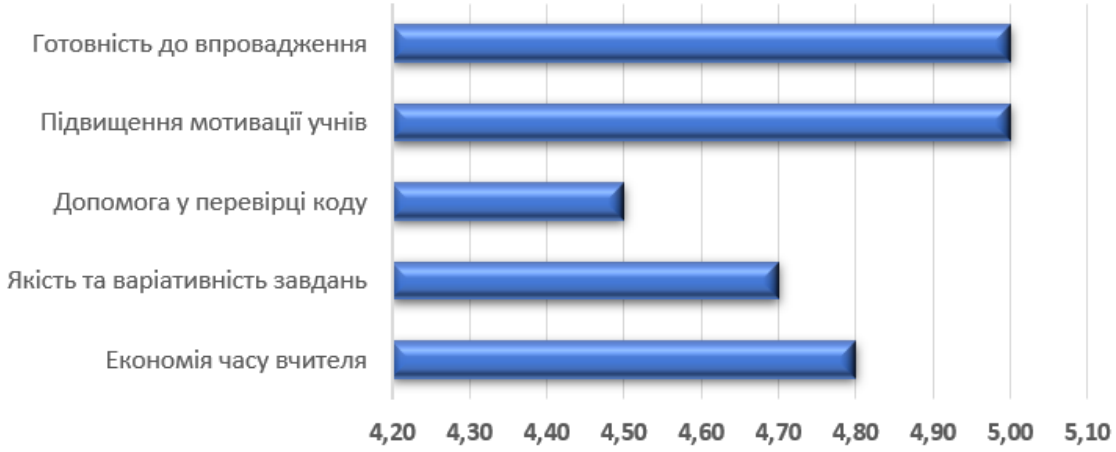


Рисунок 2.19. Оцінка ефективності модулів системи вчителями інформатики.

Важливою перевагою вчителі також вважають можливість швидкого формування різнорівневих завдань. Такий механізм суттєво полегшує реалізацію диференційованого підходу: поки учитель працює з учнями, які потребують додаткових пояснень, сильні учні можуть отримувати нові виклики відповідного рівня складності. Респонденти підкреслювали, що це дозволяє підтримувати оптимальний темп роботи на уроці та запобігати ситуаціям вимушеного простою.

Окрему увагу привернув ефект зниження бар’єру входу для молодих фахівців та вчителів-предметників без значного програмістського досвіду. Асистент надавав пояснення складних понять, зокрема об’єктно-орієнтованого програмування та рекурсії, через аналогії та спрощені моделі, що сприяло глибшому розумінню матеріалу й підвищенню впевненості педагогів у викладанні відповідних тем.

Таким чином, аналіз анкетування демонструє, що AI Teacher Assistant виконує роль ефективного інтелектуального інструмента, який не замінює педагога, а підтримує його професійну діяльність. Система оптимізує підготовчі процеси, сприяє якісній диференціації та забезпечує методичну підтримку, створюючи умови для зосередження вчителя на творчих, виховних та комунікативних аспектах навчального процесу.

2.3.4. Аналіз рівня мотивації та задоволеності учнів

Окрім академічних результатів, важливим показником є суб'єктивне сприйняття навчального процесу учнями. Серед учнів експериментальної групи було проведено вихідне анкетування.

Текст анкети для учнів включав перелік запитань:

1. Чи цікаво тобі було розв'язувати задачі з історіями (про Minecraft, Марс, ігри)?

2. Чи зрозумілі були підказки асистента, коли ти помилявся?

3. Чи було тобі комфортніше задавати "прості" питання асистенту, ніж вчителю перед усім класом?

4. Чи відчуваєш ти, що почав краще розуміти програмування?

5. Чи хотів би ти використовувати такого асистента на інших уроках?

Результати анкетування учнів описано в таблиці 2.6 та на рисунку 2.20.

Таблиця 2.6.

Результати опитування учнів експериментальної групи

Критерій	Середній бал	Висновок
Інтерес до завдань	4.9	Гейміфікація виявилася ключовим фактором залучення. Учні відзначили, що "сюжетні" задачі розв'язувати цікавіше.
Зрозумілість пояснень	4.6	Метод аналогій (пояснення складного через просте), закладений у модуль "Тьютор", виявився ефективним.
Психологічний комфорт	5.0	Усі учні підтвердили зниження страху помилки. Можливість анонімно отримати допомогу сприяла розкутості.
Суб'єктивне відчуття прогресу	4.7	Учні стали впевненішими у своїх силах при написанні коду.
Бажання продовжувати роботу	4.8	Висока лояльність до інструменту.

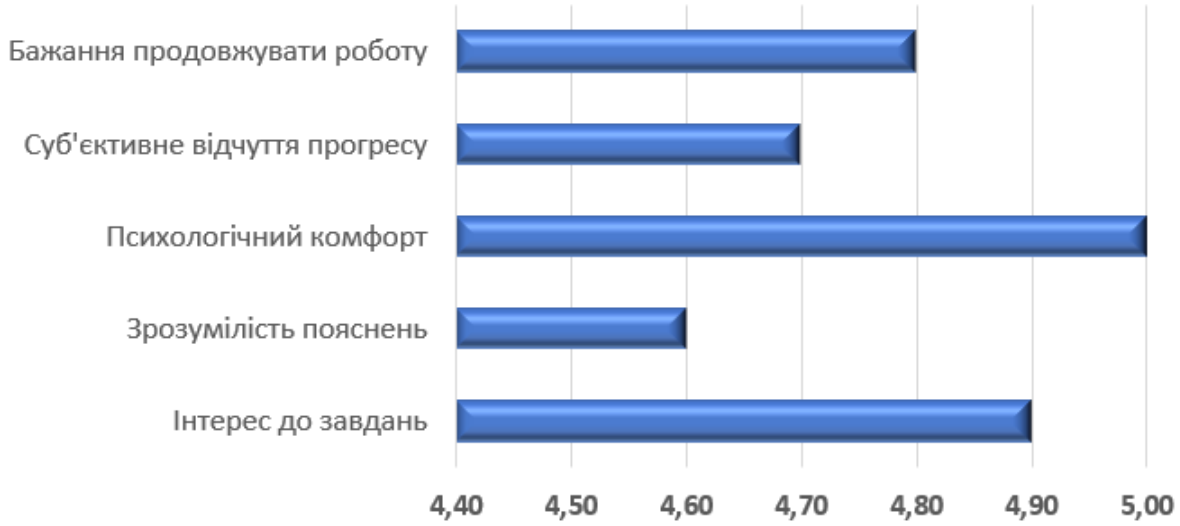


Рисунок 2.20. Оцінка учнями роботи з інтелектуальним асистентом.

Опрацювання відповідей учнів на відкрите запитання дало змогу виявити виразний психологічний ефект від використання модуля Тьютор. Учні експериментальної групи відзначали зменшення тривожності під час виконання програмістських завдань. У відгуках простежується цінування доброзичливого та некритичного формату взаємодії з асистентом, який замість негативної оцінки помилок пропонує зрозумілі пояснення та корекцію дій.

Важливим аспектом є можливість навчання у власному темпі. Учні наголошували, що можуть повертатися до одного й того самого питання, доки не опанують матеріал, без побоювання повторити його надто багато разів. Така особливість забезпечує індивідуалізацію навчального процесу та усуває типову проблему сором’язливості або страху виглядати менш компетентними перед учителем чи однолітками.

Окремий пласт відгуків стосується підвищення залученості. Учні описували, що сюжетні задачі, згенеровані модулем Генератор (зокрема історії у стилі Minecraft), підсилювали інтерес до процесу розв’язання. Завдяки використанню гейміфікації учні починали сприймати алгоритмічні завдання не як формальність, а як частину цілісного ігрового сценарію.

Сукупність висловлених учнями думок свідчить, що впровадження системи AI Teacher Assistant сприяє формуванню сприятливого емоційно-

психологічного середовища. Модулі системи не лише підвищують академічні показники, але й зменшують страх помилитися, створюють умови для індивідуалізованого темпу навчання та посилюють внутрішню мотивацію до вивчення інформатики.

Результати анкетування переконливо свідчать, що використання системи AI Teacher Assistant не лише покращує академічні результати, але й створює сприятливий психологічний мікроклімат на уроці, нівелюючи страх помилки та підвищуючи внутрішню мотивацію до вивчення інформатики.

Результати педагогічного експерименту підтвердили ефективність розробленого програмного комплексу та методики його застосування. Порівняльний аналіз успішності показав значне зростання середнього бала в експериментальній групі: приріст становив 1.7 бала, що істотно перевищує показник контрольної групи (0.5 бала) й свідчить про позитивний вплив використання інтелектуального асистента на формування предметних компетентностей.

Якісне дослідження засвідчило, що персоналізовані механізми модуля Тьютор сприяли переходу учнів від середнього до високого рівня навчальних досягнень. Це стало можливим завдяки індивідуалізованому темпу навчання, адаптивним підказкам і зниженню тривожності, пов’язаної з допущенням помилок під час розв’язання задач.

Підсумки анкетування учасників освітнього процесу підтвердили високий рівень задоволеності системою. Вчителі зазначили суттєву економію часу, зростання ефективності планування та високу методичну якість згенерованих матеріалів. Учні, зі свого боку, відзначили покращення емоційного комфорту та підвищення інтересу до програмування завдяки поєднанню індивідуальної підтримки та гейміфікованих завдань.

ВИСНОВКИ

У кваліфікаційній роботі здійснено теоретичне обґрунтування, програмну реалізацію та педагогічну верифікацію методики використання засобів штучного інтелекту як асистента вчителя інформатики. Узагальнення результатів дослідження дозволило сформулювати такі висновки.

Теоретичний аналіз засвідчив, що сучасний етап розвитку цифрових технологій характеризується переходом від традиційних систем комп'ютерного навчання до генеративних моделей штучного інтелекту (GenAI). Встановлено, що великі мовні моделі мають виражений дидактичний потенціал у навчанні програмування: вони здатні генерувати та аналізувати код, адаптувати рівень складності матеріалу, здійснювати смислове пояснення алгоритмів і підтримувати природномовний діалог. Це зумовлює трансформацію професійної ролі вчителя – від передавача інформації до фасилітатора й архітектора освітнього досвіду.

Уточнено концептуальні засади функціонування інтелектуального асистента вчителя інформатики. Обґрунтовано, що ефективна система має бути модульною та поєднувати чотири ключові ролі: *Методист* – автоматизоване проєктування планів уроків; *Генератор контенту* – створення диференційованих завдань; *Екзаменатор* – рецензування й оптимізація учнівського коду; *Тьютор* – індивідуальний супровід учня в режимі діалогу.

Розроблено програмний комплекс AI Teacher Assistant. Обґрунтовано вибір інструментальних засобів і реалізовано веб-застосунок на основі Python, Streamlit і API моделі Google Gemini 2.0. Запропоновано архітектуру, що забезпечує захист конфіденційної інформації за рахунок ізоляції ключів доступу; гнучкість оновлення промптів через зовнішні конфігураційні файли; зниження частоти помилкових відповідей за допомогою технік Few-Shot Prompting та Chain-of-Thought, що сприяло педагогічній коректності роботи системи.

Розроблено та обґрунтовано методику використання ШІ-асистента у вивченні інформатики (на прикладі Python). Запропоновано низку сценаріїв педагогічного застосування, а саме:

– автоматизована розробка конспектів уроків відповідно до стандартів НУШ;

– гейміфікація навчального матеріалу через сторітелінг та генерацію сюжетних задач;

– формування предметних компетентностей шляхом впровадження технік «Парне програмування з ШІ» та «Сократівський діалог», спрямованих на розвиток алгоритмічного мислення, умінь налагодження та рефлексії.

Отримані результати свідчать, що учні експериментальної групи продемонстрували суттєво вищу динаміку навчальних досягнень порівняно з контрольною групою. Середній бал зріс на 1.7 (з 8.8 до 10.5), тоді як у контрольній групі приріст становив 0.5 бала. Якісний аналіз підтвердив позитивний психологічний ефект використання ШІ-асистента: зниження навчальної тривожності, зростання мотивації та підвищення залученості завдяки механізмам сюжетності та індивідуалізованої підтримки.

Узагальнення результатів дослідження дозволяє стверджувати, що інтеграція інтелектуальних агентів-асистентів у навчання інформатики є перспективним напрямом модернізації шкільної освіти. Запропонована методика сприяє реалізації принципів особистісно орієнтованого навчання, оптимізує діяльність учителя, зменшує частку рутинних операцій та забезпечує кожного учня персоналізованою траєкторією навчальної взаємодії.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Mittal U., Sai S., Chamola V., Sangwan D. «A Comprehensive Review on Generative AI for Education». *IEEE Access*, vol. 12, pp. 142733–142759, 2024. DOI: 10.1109/ACCESS.2024.3468368.

2. Bahroun Z., Anane C., Ahmed V., Zacca A. «Transforming Education: A Comprehensive Review of Generative Artificial Intelligence in Educational Settings through Bibliometric and Content Analysis». *Sustainability*, vol. 15, no. 17, p. 12983, 2023. DOI: 10.3390/su151712983.

3. Yusuf A., Pervin N., Román-González M., Noor N. M. «Generative AI in education and research: A systematic mapping review». *Review of Education*, vol. 12, no. 2, e3489, 2024. DOI: 10.1002/rev3.3489.

4. Chen Y., Deng H., Chen C.-H., Chung C.-L. «Efficient Artificial Intelligence-Teaching Assistant Based on ChatGPT». *Proceedings of the 2023 International Conference on Smart Systems for Applications in Electrical Sciences (ICSSES)*, Tumakuru, India, IEEE, 2023, pp. 1–5. DOI: 10.1109/ICSSES58299.2023.10200077.

5. Isaacs M., Majeed A., Moussa K., Shodipo D. «Design and Implementation of an Ethical AI-Based Teaching Assistant for IoT Security Education». *AJIMS*, vol. 6, no. 1, 2024. DOI: 10.51415/ajims.v6i1.1586.

6. Guo F., Duan J., Peng X., Liang C. «A Personalized Teaching Assistant Platform Driven by Large Language Models of Artificial Intelligence». In: Zhang K., Song X., Obaidat M. S., Bilal A., Hu J., Lu Z. (eds.) *Computer Science and Educational Informatization*. Communications in Computer and Information Science, vol. 2448. Springer Nature Singapore, 2025, pp. 176–185. DOI: 10.1007/978-981-96-3738-6_14.

7. BaiDoo-Anu D., Owusu Ansah L. «Education in the Era of Generative Artificial Intelligence (AI): Understanding the Potential Benefits of ChatGPT in Promoting Teaching and Learning». *Journal of AI*, vol. 7, no. 1, pp. 52–62, 2023. DOI: 10.61969/jai.1337500.

8. Gobert J. D., Sao Pedro M. A., Li H., Lott C. «Intelligent tutoring systems: a history and an example of an ITS for science». In: *International Encyclopedia of*

Education (Fourth Edition). Elsevier, 2023, pp. 460–470. DOI: 10.1016/B978-0-12-818630-5.10058-2.

9. Wang N., Wang X., Su Y.-S. «Critical analysis of the technological affordances, challenges and future directions of Generative AI in education: a systematic review». *Asia Pacific Journal of Education*, vol. 44, no. 1, pp. 139–155, 2024. DOI: 10.1080/02188791.2024.2305156.

10. Zhang J. «Computer Assisted Instruction System under Artificial Intelligence Technology». *International Journal of Emerging Technologies in Learning (iJET)*, vol. 16, no. 5, pp. 4–16, 2021.

11. Lin C.-C., Huang A. Y. Q., Lu O. H. T. «Artificial intelligence in intelligent tutoring systems toward sustainable education: a systematic review». *Smart Learning Environments*, vol. 10, no. 1, p. 41, 2023. DOI: 10.1186/s40561-023-00260-y.

12. Alier M., García Peñalvo F. J., Camba J. D. «Generative Artificial Intelligence in Education: From Deceptive to Disruptive». *IJIMAI*, vol. 8, no. 5, pp. 5–14, 2024. DOI: 10.9781/ijimai.2024.02.011.

13. Salazar L. R., Peeples S. F., Brooks M. E. «Generative AI Ethical Considerations and Discriminatory Biases on Diverse Students Within the Classroom». In: Elmoudden S., Wrench J. S. (eds.) *Advances in Educational Technologies and Instructional Design*. IGI Global, 2024, pp. 191–213. DOI: 10.4018/979-8-3693-0831-8.ch010.

14. Kamalov F., Santandreu Calonge D., Gurrib I. «New Era of Artificial Intelligence in Education: Towards a Sustainable Multifaceted Revolution». *Sustainability*, vol. 15, no. 16, p. 12451, 2023. DOI: 10.3390/su151612451.

15. Kazemitabaar M. et al. «CodeAid: Evaluating a Classroom Deployment of an LLM-based Programming Assistant that Balances Student and Educator Needs». In: *Proceedings of the CHI Conference on Human Factors in Computing Systems*, Honolulu, HI, USA, ACM, 2024, pp. 1–20. DOI: 10.1145/3613904.3642773.

16. Song T., Zhang H., Xiao Y. «A High-Quality Generation Approach for Educational Programming Projects Using LLM». *IEEE Transactions on Learning Technologies*, vol. 17, pp. 2242–2255, 2024. DOI: 10.1109/TLT.2024.3499751.

17. Estévez-Ayres I., Callejo P., Hombrados-Herrera M. Á., Alario-Hoyos C., Delgado Kloos C. «Evaluation of LLM Tools for Feedback Generation in a Course on Concurrent Programming». *International Journal of Artificial Intelligence in Education*, vol. 35, no. 2, pp. 774–790, 2025. DOI: 10.1007/s40593-024-00406-0.

18. Yan Y.-M., Chen C.-Q., Hu Y.-B., Ye X.-D. «LLM-based collaborative programming: impact on students' computational thinking and self-efficacy». *Humanities and Social Sciences Communications*, vol. 12, no. 1, p. 149, 2025. DOI: 10.1057/s41599-025-04471-1.

19. Vaswani A. et al. «Attention Is All You Need», 2017. DOI: 10.48550/ARXIV.1706.03762.

20. Pitts G., Hridi A. P., Lekshmi Narayanan A. B. «A Survey of LLM-Based Applications in Programming Education: Balancing Automation and Human Oversight». In: *Proceedings of the Fourth Workshop on Bridging Human-Computer Interaction and Natural Language Processing (HCI+NLP)*, Suzhou, China, Association for Computational Linguistics, 2025, pp. 255–262. DOI: 10.18653/v1/2025.hcinlp-1.21.

21. Козлов С. Л., Колесницький О. К. «Застосування архітектури трансформер до задачі super-resolution». *SWVNTU*, №. 1, 2024. DOI: 10.31649/2307-5376-2024-1-7-18.

22. Юрчак І. Ю., Кичук О. О., Оксентюк В. М., Хіч А. О. «Можливості та обмеження великих мовних моделей». *CSN*, №. 6(2), С. 286–300, 2024. DOI: 10.23939/csn2024.02.286.

23. Глибовець М. М., Задохін Д. В., Дехтяр Б.-Я., Пєчкурова О. М. «Оброблення природної мови за допомоги великих мовних моделей і методів машинного навчання». *NRPCOMP*, №. 7, С. 102–111, 2025. DOI: 10.18523/2617-3808.2024.7.102-111.

24. Юнчик В.Л., Кунанець Н.Е., Пасічник В.В., Федонюк А.А. «Аналіз штучних інтелектуальних агентів для систем електронного навчання». *Vìsn. Nac. unìv. 'L'vìv. polìteh.', Ser. Ìnf. sist. merežì*, №. 10, С. 41–57, 2021. DOI: 10.23939/sisn2021.10.041.

25. López-Fernández D., Vergaz R. «Adoption and Impact of ChatGPT in Computer Science Education: A Case Study on a Database Administration Course», 2024. arXiv. DOI: 10.48550/ARXIV.2407.12145.

26. Kosar T., Ostojić D., Liu Y. D., Mernik M. «Computer Science Education in ChatGPT Era: Experiences from an Experiment in a Programming Course for Novice Programmers». *Mathematics*, vol. 12, no. 5, p. 629, 2024. DOI: 10.3390/math12050629.

27. Ogunleye B., Zakariyyah K. I., Ajao O., Olayinka O., Sharma H. «A Systematic Review of Generative AI for Teaching and Learning Practice». *Education Sciences*, vol. 14, no. 6, p. 636, 2024. DOI: 10.3390/educsci14060636.

28. Гриневич Л. М., Елькін О., Калашнікова С., Коберник І., Ковтунець В., Макаренко О., Малахова О., Нанаєва Т., Усатенко Г., Хобзей П., Шиян Р. *Нова українська школа. Концептуальні засади реформування середньої школи*. Київ, 2016. [Електронний ресурс]: https://mon.gov.ua/static-objects/mon/sites/1/zagalna%20serednya/nova-ukrainska-shkola-compressed.pdf

29. Ривкінд Й. Я., Лисенко Т. І., Чернікова Л. А., Шакотько В. В. *Модельна навчальна програма «Інформатика. 5–6 класи» для закладів загальної середньої освіти*. [Електронний ресурс]: https://informatik.pp.ua/kabinet/programi/5-6-klasy/modelna-programa-informatyka-5-6-klasy-ryvkind/

30. Гриневич Л. М. «Наказ 07.06.2017 № 804 Про оновлені навчальні програми для учнів 5–9 класів загальноосвітніх навчальних закладів». [Електронний ресурс]: https://zakon.rada.gov.ua/rada/show/v0804729-17#Text

31. Бондаренко О. О., Ластовецький В. В., Пилипчук О. П., Шестопалов Є. А. *Інформатика: підручник для 8 кл. закладів загальної середньої освіти*. Харків: Вид-во «Ранок», 2021. 240 с.

32. *Навчальна програма з інформатики для 10–11 класів закладів загальної середньої освіти (Профільний рівень). Затверджено наказом МОН України від 23.10.2017 № 1407*. [Електронний ресурс]: https://informatik.pp.ua/kabinet/programi/10-11-klasy/programa-informatyka-10-11-klas-profilnyi-riven-2018/

33. *Навчальна програма з інформатики для 10–11 класів закладів загальної середньої освіти (Рівень стандарту). Затверджено наказом МОН України від 23.10.2017 № 1407*. [Електронний ресурс]: https://informatik.pp.ua/kabinet/programi/10-11-klasy/programa-informatyka-10-11-klas-riven-standartu-2018/

34. Руденко В. Д., Речич Н. В., Потієнко В. О. *Інформатика (профільний рівень): підручник для 10 кл. закладів загальної середньої освіти*. Харків: Ранок, 2018. 255 с.

ДОДАТКИ

Додаток А

План-конспект уроку інформатики (9 клас) згенеровано системою AI Teacher Assistant

Тема: Цикли з лічильником (for) у мові Python

Мета:

Навчальна: Ознайомити учнів з поняттям циклу з лічильником, його синтаксисом у мові Python (оператор `for`), навчити використовувати цикл `for` для розв'язання простих задач.

Розвивальна: Розвивати логічне мислення, алгоритмічне мислення, вміння аналізувати та застосовувати отримані знання для розв'язання практичних завдань.

Виховна: Виховувати інформаційну культуру, відповідальність, самостійність.

Тип уроку: Засвоєння нових знань

Обладнання: Комп'ютери з встановленим середовищем Python, інтерактивна дошка (за наявності), підручники, роздаткові матеріали з прикладами коду (за бажанням).

Хід уроку:

I. Організаційний момент (2 хвилини)

Привітання.

Перевірка готовності учнів до уроку.

Оголошення теми та мети уроку.

II. Актуалізація опорних знань (5 хвилин)

Бесіда:

Що таке алгоритм?

Які основні алгоритмічні структури ви знаєте? (слідування, розгалуження, повторення)

Для чого потрібні цикли в програмуванні?

Які типи циклів ви знаєте (можливо, з інших мов програмування)?

III. Вивчення нового матеріалу (20 хвилин)

Пояснення поняття циклу з лічильником (for):

Цикл `for` використовується для виконання блоку коду певну кількість разів. Кількість повторень визначається наперед, на відміну від циклів `while`, де умова може змінюватися під час виконання.

Цикл `for` ідеально підходить для обробки елементів послідовності (наприклад, чисел у заданому діапазоні, символів у рядку).

Синтаксис циклу `for` у Python:

```
for змінна in послідовність:
    # Блок коду, який буде повторюватися
```

`змінна`: Змінна, яка приймає значення кожного елементу послідовності на кожній ітерації циклу.

`послідовність`: Це може бути рядок, список, кортеж, діапазон чисел, або інший ітерований об'єкт.

Функція `range()`:

Функція `range()` використовується для генерації послідовності чисел.

`range(кінцеве_число)`: генерує послідовність від 0 до `кінцеве_число - 1`.

`range(початкове_число, кінцеве_число)`: генерує послідовність від `початкове_число` до `кінцеве_число - 1`.

`range(початкове_число, кінцеве_число, крок)`: генерує послідовність від `початкове_число` до `кінцеве_число - 1` з заданим кроком.

Приклади використання циклу `for` для виведення чисел:

Приклад 1: Виведення чисел від 0 до 4:

```
for i in range(5):
    print(i)
```

Пояснення: Цикл виконається 5 разів. На кожній ітерації змінна `i` буде приймати значення від 0 до 4.

Приклад 2: Виведення чисел від 1 до 10:

```
for i in range(1, 11):
```

```
    print(i)
```

Пояснення: Цикл виконається 10 разів. Змінна `i` буде приймати значення від 1 до 10.

Приклад 3: Виведення парних чисел від 2 до 20:

```
for i in range(2, 21, 2):
    print(i)
```

Пояснення: Цикл виконається 10 разів. Змінна `i` буде приймати значення 2, 4, 6, ..., 20.

Блок-схема циклу `for` (за бажанням, для кращого візуального розуміння).

IV. Практична частина (15 хвилин)

Завдання 1: Напишіть програму, яка виводить на екран числа від 10 до 1 в зворотному порядку.

Підказка: Використовуйте функцію `range()` з від'ємним кроком.

Завдання 2: Напишіть програму, яка обчислює суму чисел від 1 до заданого числа `n`. Число `n` повинно бути введене користувачем.

Підказка: Використовуйте змінну для зберігання суми, і додавайте до неї кожне число в циклі.

Завдання 3 (додаткове): Напишіть програму, яка виводить таблицю множення для числа 7.

Підказка: Використовуйте цикл `for` для виведення рядків таблиці множення.

Учні виконують завдання на комп'ютерах. Вчитель надає допомогу за потреби.

V. Підбиття підсумків уроку (5 хвилин)

Обговорення виконаних завдань.

Відповіді на питання учнів.

Повторення основних понять:

Що таке цикл з лічильником `for`?

Як працює функція `range()`?

Які параметри може приймати функція `range()`?

VI. Домашнє завдання (3 хвилини)

Вивчити синтаксис циклу `for` і функції `range()`.

Написати програму, яка виводить усі непарні числа від 1 до 50.

Написати програму, яка обчислює факторіал заданого числа `n` (факторіал числа n позначається n! і обчислюється як добуток всіх натуральних чисел від 1 до n включно. Наприклад, 5! = 1 * 2 * 3 * 4 * 5 = 120). Число `n` повинно бути введене користувачем.

Примітки:

Вчитель може адаптувати план уроку відповідно до рівня підготовки учнів.

Можна використовувати додаткові візуальні матеріали (слайди, відео) для кращого пояснення матеріалу.

Важливо забезпечити достатньо часу для практичної роботи учнів.

Завдання можна змінювати або додавати в залежності від часу, який залишився.

Додаток Б

Анкета 1. Оцінювання ефективності використання системи AI Teacher Assistant

Шановний колего! Просимо Вас оцінити досвід використання інтелектуального асистента на уроках інформатики за 5-бальною шкалою (де 1 – зовсім не ефективно, 5 – дуже ефективно).

1. Оцініть ефективність модуля Методист:

Чи допоміг асистент скоротити час на підготовку планів-конспектів уроків?

[1] [2] [3] [4] [5]

2. Оцініть якість дидактичних матеріалів модуля Генератор:

Наскільки згенеровані задачі відповідають віковим особливостям учнів та принципам диференціації?

[1] [2] [3] [4] [5]

3. Оцініть корисність модуля Екзаменатор:

Чи допомагає попередня перевірка коду штучним інтелектом підвищити якість робіт, які учні здають на перевірку Вам?

[1] [2] [3] [4] [5]

4. Вплив на мотивацію учнів:

Як ви оцінюєте вплив гейміфікованих завдань (історії, легенди) на залученість учнів у навчальний процес?

[1] [2] [3] [4] [5]

5. Загальне враження:

Чи готові Ви використовувати цей інструмент у своїй подальшій практиці?

[1] [2] [3] [4] [5]

6. Відкрите питання:

Яка функція системи виявилася для Вас найбільш корисною?

Додаток В

Анкета для учнів. Мої враження від роботи з AI-асистентом

Ми використовували на уроках інформатики програму AI Teacher Assistant. Будь ласка, оціни свої враження чесно. Твої відповіді анонімні.

(Оцінка від 1 до 5, де 1 – "Ні, зовсім не сподобалось", 5 – "Так, дуже сподобалось")

1. Чи цікаво тобі було розв'язувати задачі з історіями (про Minecraft, Марс, ігри)?

[1] [2] [3] [4] [5]

2. Чи зрозумілі були підказки асистента, коли ти робив помилки в коді?

[1] [2] [3] [4] [5]

3. Чи було тобі комфортніше задавати "прості" питання асистенту, ніж вчителю перед усім класом?

[1] [2] [3] [4] [5]

4. Чи відчуваєш ти, що почав краще розуміти програмування (Python)?

[1] [2] [3] [4] [5]

5. Чи хотів би ти використовувати такого асистента на інших уроках?

[1] [2] [3] [4] [5]

6. Що тобі сподобалось найбільше? (Напиши кілька слів)

Анотація

Бороненко В.І. Методи та засоби штучного інтелекту асистента вчителя – Рукопис. Випускна кваліфікаційна робота за спеціальністю 014 Середня освіта (Інформатика). – Волинський національний університет імені Лесі Українки. Луцьк, 2025 р.

Кваліфікаційна робота присвячена дослідженню дидактичного потенціалу генеративного штучного інтелекту та розробці методики використання інтелектуального асистента вчителя на уроках інформатики. Актуальність теми зумовлена стрімкою цифровізацією освіти, необхідністю персоналізації навчання в умовах НУШ та потребою в автоматизації рутинної методичної роботи вчителя за допомогою сучасних мовних моделей (LLM).

У роботі теоретично обґрунтовано архітектуру та функціональні можливості агентів-асистентів. Розроблено програмний комплекс AI Teacher Assistant на базі мови Python та Google Gemini API, що включає модулі для планування уроків, генерації диференційованих завдань та рецензування коду. Запропоновано та експериментально перевірено методику використання ШІ-асистента, яка продемонструвала позитивний вплив на успішність та мотивацію учнів.

Практичне значення отриманих результатів полягає у створенні готового до впровадження веб-застосунку для підтримки вчителя інформатики, а також у розробці бібліотеки системних промптів для ефективної взаємодії з нейромережами. Результати дослідження можуть бути використані вчителями закладів загальної середньої освіти для оптимізації навчального процесу.

Ключові слова: штучний інтелект, асистент вчителя, великі мовні моделі (LLM), промпт-інжиніринг, методика навчання інформатики, Python, персоналізоване навчання.

Abstract

Boronenko V.I. Methods and Means of Artificial Intelligence for a Teacher's Assistant – Manuscript. Graduation Qualification Work for Specialty 014 Secondary Education (Informatics). – Lesya Ukrainka Volyn National University. Lutsk, 2025.

The qualification work is dedicated to the study of the didactic potential of generative artificial intelligence and the development of a methodology for using an intelligent teacher's assistant in Informatics lessons. The relevance of the topic is due to the rapid digitalization of education, the need for learning personalization within the context of the New Ukrainian School (NUSh), and the need to automate the teacher's routine methodological work using modern Large Language Models (LLMs).

The paper theoretically substantiates the architecture and functional capabilities of agent-assistants. The software complex AI Teacher Assistant was developed based on the Python language and the Google Gemini API, which includes modules for lesson planning, generating differentiated tasks, and code review. A methodology for using the AI-assistant was proposed and experimentally verified, which demonstrated a positive impact on student achievement and motivation.

The practical significance of the obtained results lies in the creation of a ready-to-implement web application to support the Informatics teacher, as well as the development of a library of system prompts for effective interaction with neural networks. The research results can be used by teachers in general secondary education institutions to optimize the educational process.

Key words: artificial intelligence, teacher's assistant, large language models (LLM), prompt engineering, informatics teaching methodology, Python, personalized learning.