МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ І ПРИКЛАДНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ЧУПРИНА ІВАН АНДРІЙОВИЧ

Допускається до захисту:
в.о. завідувача кафедри
інформаційних технологій
канд. техн. наук, доцент
_____ О. В. Зелінська
«_____» _____ 2024р.

СИСТЕМА АВТОМАТИЧНОГО СТВОРЕННЯ ТА ВАЛІДАЦІЇ ТЕСТОВИХ ПИТАНЬ НА ОСНОВІ ВЗАЄМОДІЇ З CHATGPT

Спеціальність 122 Комп’ютерні науки

Кваліфікаційна (магістерська) робота

Науковий керівник:
Антонов Ю.С., доцент кафедри
інформаційних технологій,
канд. фіз.-мат. наук, доцент
_____
(Підпис)

Оцінка __ _____/_____/_____
(бали/за шкалою ЄКТС/за національною шкалою)
Голова ЕК:_____
(Підпис)

Вінниця 2024

АНОТАЦІЯ

Чуприна І.А. Система автоматичного створення та валідації тестових питань на основі взаємодії з ChatGPT Спеціальність 122 «Комп'ютерні науки», Освітня програма «Комп'ютерні технології обробки даних». Донецький національний університет імені Василя Стуса, Вінниця, 2024.

У кваліфікаційній роботі розроблено систему для автоматичного створення та валідації тестових питань за допомогою моделі ChatGPT. Описано архітектуру веб-додатку, що інтегрує API для генерації тестів та перевірки результатів, а також збереження тестів у базі даних. Досліджено застосування машинного навчання для точнішої валідації тестів і створено інструменти для тестування та деплойменту.

Ключові слова: автоматизація, API, тестування, ChatGPT, валідація, машинне навчання, бази даних.

76с., 2 табл., 28 рис.,61 джерело.

Chupryna I. A. Development of work searching web application with an intelligent selection system. Specialty 122 "Computer science", Programme "Computer data processing technologies". Vasyl` Stus Donetsk National University, Vinnytsia, 2024.

In the qualification (master's), a system for the automatic creation and validation of test questions using the ChatGPT model was developed. The architecture of the web application is described, integrating an API for test generation and result verification, as well as saving tests to a database. Machine learning methods for more accurate test validation were explored, and tools for testing and deployment were created.

Keywords: automation, API, testing, ChatGPT, validation, machine learning, databases.

76 pages, 2 tables, 28 images, 61 sources

ЗМІСТ

ВСТУП

Актуальність теми. Безперервний розвиток інформаційних технологій та штучного інтелекту відкриває нові можливості для автоматизації освітніх процесів. Система автоматичного створення та валідації тестових питань на основі взаємодії з ChatGPT є актуальною, оскільки дозволяє значно підвищити ефективність розробки освітніх матеріалів, зменшуючи час і ресурси, необхідні для створення якісних тестових завдань. Використання таких систем може забезпечити швидку адаптацію навчальних програм до індивідуальних потреб студентів, підвищити точність оцінювання знань та сприяти більш глибокому розумінню навчального матеріалу. Крім того, впровадження штучного інтелекту в освітній процес відповідає сучасним тенденціям діджиталізації та інноваційного розвитку освіти, що робить дану тему надзвичайно актуальною в контексті підготовки фахівців майбутнього.

Мета дослідження. Метою даної дипломної роботи є розробка та впровадження системи автоматичного створення та валідації тестових питань на основі взаємодії з ChatGPT. Дослідження спрямоване на вивчення можливостей сучасних моделей штучного інтелекту для автоматизації процесу створення тестових завдань, забезпечення їхньої релевантності та точності, а також підвищення ефективності освітніх процесів. Особливу увагу приділено аналізу точності та коректності згенерованих питань, їх відповідності навчальним цілям та стандартам.

Об'єктом дослідження є процес автоматизованого створення та валідації тестових питань з використанням штучного інтелекту, зокрема, моделі ChatGPT. Дослідження фокусується на алгоритмах і методах, які застосовуються для генерації навчальних завдань, а також на способах оцінки їхньої якості та релевантності. До об'єкту дослідження також належать освітні платформи та інструменти, які інтегрують подібні системи для оптимізації навчального процесу, підвищення його ефективності та адаптивності до потреб студентів.

Предметом дослідження є алгоритми та технології, що забезпечують автоматичне створення та валідацію тестових питань на основі взаємодії з

моделлю ChatGPT. У фокусі дослідження знаходяться методи генерації текстових завдань, параметри налаштування штучного інтелекту для досягнення максимальної точності та релевантності питань. Особливу увагу приділено аналізу ефективності цих алгоритмів у контексті їхньої інтеграції в освітні процеси та їхнього впливу на якість навчання і оцінювання знань студентів.

Практичне значення дослідження полягає у створенні ефективної системи автоматичного генерації та валідації тестових питань, яка може бути інтегрована в освітні платформи та навчальні заклади. Використання цієї системи дозволить суттєво скоротити час та зусилля, необхідні для розробки якісних тестових завдань, забезпечуючи їхню відповідність навчальним програмам та освітнім стандартам. Це сприятиме підвищенню ефективності навчального процесу, дозволяючи викладачам зосередитися на інших важливих аспектах викладання, а студентам отримувати більш точні та індивідуалізовані оцінки своїх знань. Крім того, система може бути адаптована для різних дисциплін і рівнів навчання, що робить її універсальним інструментом для підвищення якості освіти.

Структура роботи: дана робота складається із вступу, трьох розділів, висновків та списку використаних джерел.

РОЗДІЛ 1

ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Актуальний стан розвитку штучного інтелекту

Штучний інтелект – це технологія, яка дозволяє, використовуючи комп'ютери та машини, симулювати людський розум та його здатність до розв'язку різного виду задач [4]. Як галузь інформатики штучний інтелект часто згадується разом із **машинним** та **глибоким навчанням**.

Машинне навчання відноситься до вивчення комп'ютерних систем, які навчаються та адаптуються автоматично на основі досвіду без жорсткого програмування типу «сигнал-відповідь» [5]. Основна ідея полягає в тому, щоб знайти набір даних (тренувальний), який буде розділений на вхідні та вихідні параметри, потім створити алгоритм, який буде приймати на вхід першу категорію інформації, за певними правилами обробляти її, робити перетворення та на виході видавати таку відповідь, яка буде найбільш наближена до другої категорії. Іншими словами, цей алгоритм аналізує, яка має бути відповідь при певних параметрах, і намагається вибудувати шаблон поведінки, який потім буде повторювати. Під кожну задачу є свої види таких алгоритмів:

- Задача класифікації
    - Логістична регресія
    - Дерева рішень
    - Наївний баєсів класифікатор
- Задача кластеризації
    - K-means
    - Кластеризація на основі щільності
- Задача регресії
    - Лінійна регресія
    - Регресія з множинним лінійним зв'язком
    - Дерева рішень для регресії

Також для кожної вказаної задачі можна створити та натренувати нейронну

мережу. **Нейронна мережа** - це математична модель, що працює за принципом людського мозку. Вона навчається шляхом первинного опрацювання великого набору даних, не вимагаючи написання окремого коду під конкретне завдання [6]. Сфера в комп'ютерних науках, яка займається дослідженням цього виду алгоритму зветься глибоке навчання. Нейронні мережі складаються з набору невеликих ядр, які називаються «вузлами». Вузли об'єднуються у шари. Зазвичай, у нейронної мережі є вхідний, вихідний та декілька внутрішніх шарів. Вузли в цих скупченнях передають дані один одному, змінюючи їх відповідно до того, яка в них вага, подібно до того, як у мозку нейрони передають електричні імпульси [7].

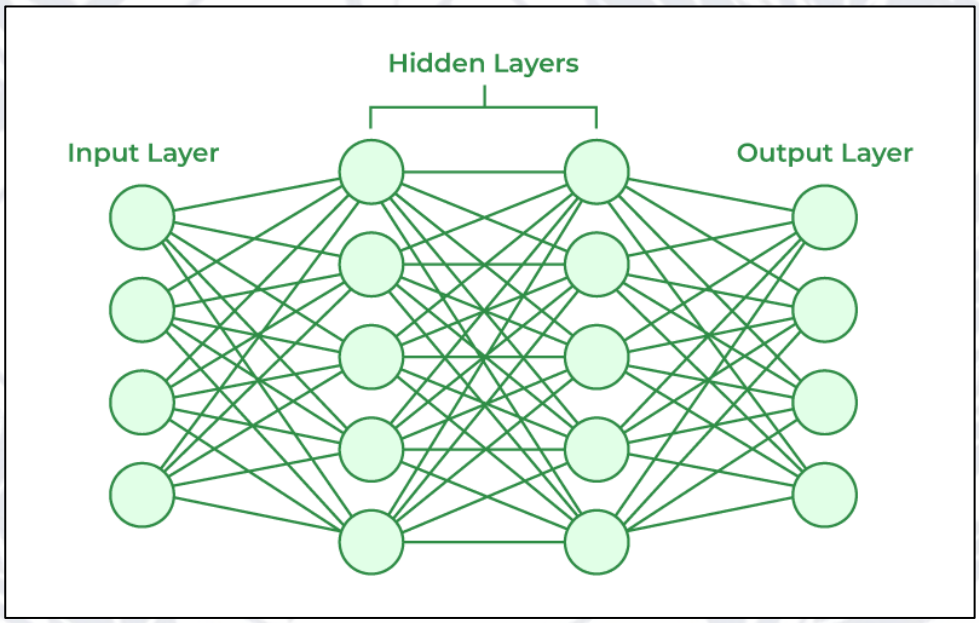

Рисунок 1.1 – приклад вигляду простої нейронної мережі з вхідним, 2 внутрішніми та вихідним шарами [8]

Також подібно до того, як різні частини мозку людини відповідають за різні функції, так само існує дуже багато різних видів нейронних мереж, які призначені для виконання різноманітних специфічних задач:

- **Deep Feedforward Networks**
- **Convolutional Neural Networks**
- **Recurrent Neural Networks**
- **Generative Adversarial Networks**

- **Long Short-Term Memory Networks**
- **Encoder-Decoder Neural Networks**

Deep feedforward networks [9], також відомі як **feedforward neural networks** або **multilayer perceptrons (MLPs)**, є основними моделями глибокого навчання. Мета feedforward-мережі - наблизити деяку функцію f^*. Наприклад, для класифікатора $y = f^*(x)$ вхід x відображається у категорію y. Мережа feedforward визначає відображення $y = f(x;\theta)$ та навчає значення параметрів θ, які дають найкраще наближення функції. Ці моделі називаються feedforward, оскільки інформація просочується через функцію, яку оцінюємо, від x через проміжні обчислення, використовувані для визначення f, і, нарешті, до виходу y. У цих мережах відсутні зворотні зв'язки, при яких виходи моделі подаються на вхід самій собі.

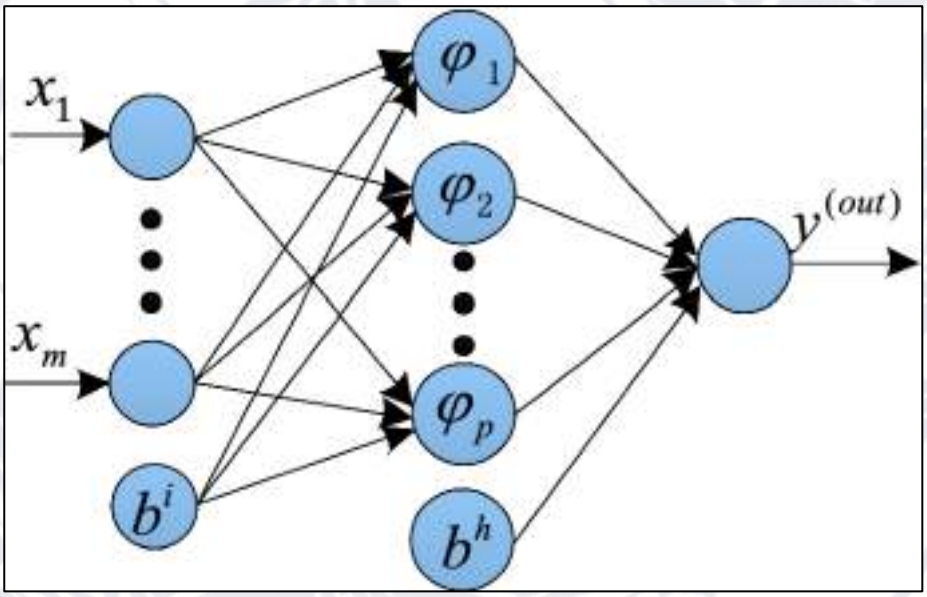

Рисунок 1.2 – приклад вигляду Deep feedforward networks [9]

Якщо говорити про **Convolutional Neural Networks** [10], то порівняно з іншими нейронними мережами, вони вирізняються своєю високою ефективністю в обробці зображення, мови або аудіосигналів. У них є три основні типи шарів: **convolutional**, **pooling** та **fully-connected**. Convolutional шар є першим шаром у конволюційній мережі, а fully-connected шар є завершальним. Із кожним шаром конволюційної мережі збільшується складність, ідентифікуючи більші порції

зображення. Ранні шари фокусуються на простих ознаках, таких як кольори та контури, і подальший прогрес даних зображення допомагає розпізнавати більші елементи чи форми об'єкта.

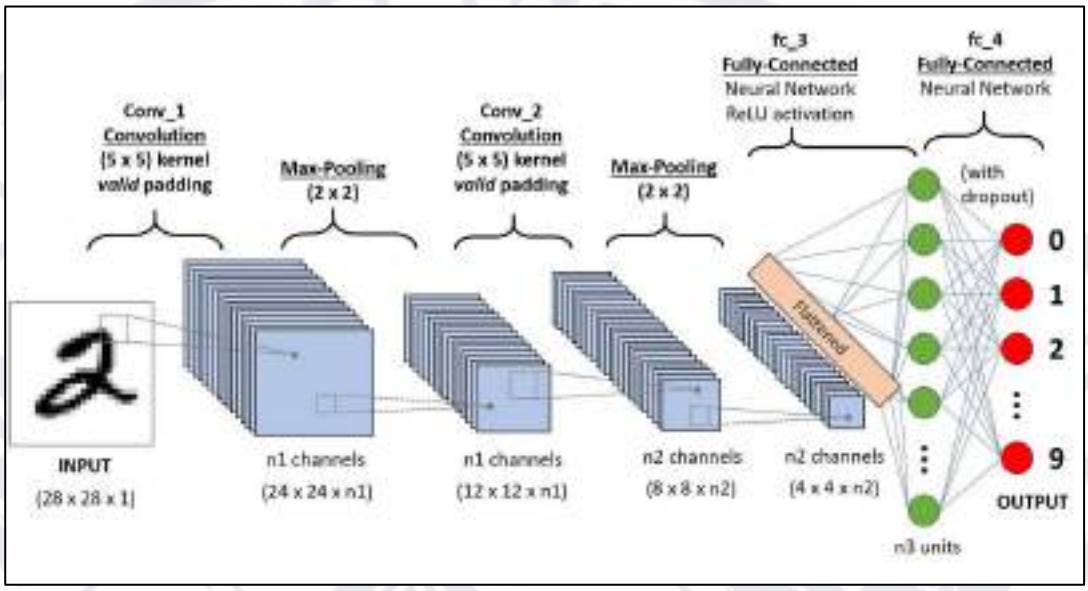

Рисунок 1.3 – Схема конволюційної нейронної мережі[10]

Recurrent Neural Networks або **Рекурентні нейромережі** призначені для роботи з послідовними даними[11], такими як мова чи часові ряди. Вони використовують зворотний зв'язок для передачі інформації між часовими кроками. Кожен вихід рекурентного шару стає вхідним сигналом наступного часового кроку. Це дозволяє RNN зберігати і використовувати інформацію з минулих кроків для прийняття рішення на поточному кроці. Однак у стандартних RNN може виникати проблема "зникаючого градієнту", що ускладнює навчання на довгих послідовностях.

Generative Adversarial Networks або **Генеративні змагальні мережі** використовують дві моделі - генератор і дискримінатор - для генерації нових даних [12], яких надто важко відрізнити від реальних даних. Генератор створює нові зразки, а дискримінатор оцінює їхню реалістичність. Процес триває до того часу, поки генератор не створить такі реалістичні зразки, які дискримінатор більше не зможе відрізнити. GANs застосовуються для генерації зображень, тексту, звуку та іншого

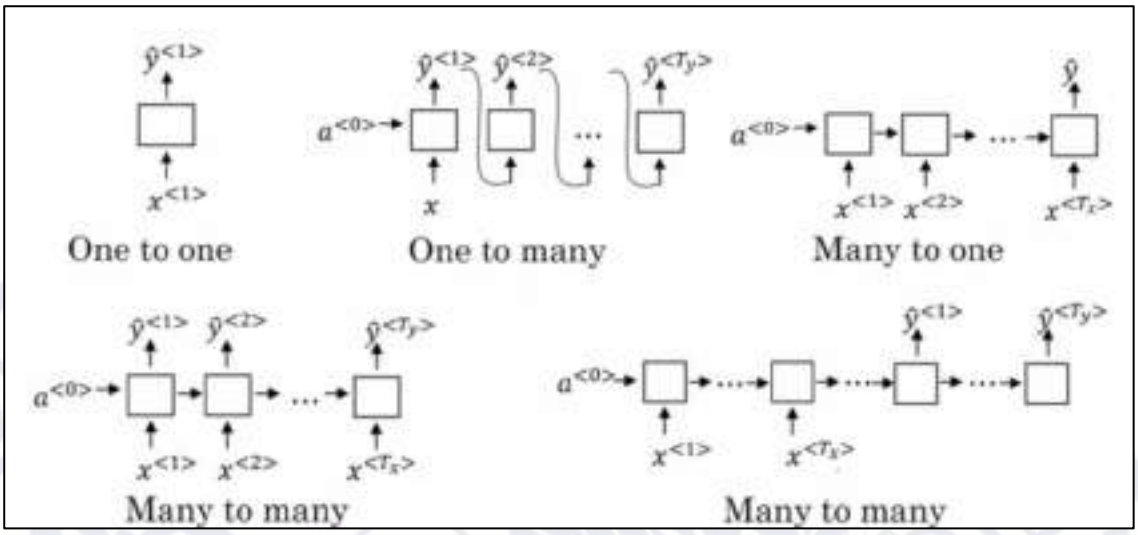

Рисунок 1.4 – Різні види рекурентних нейронних мереж [11]

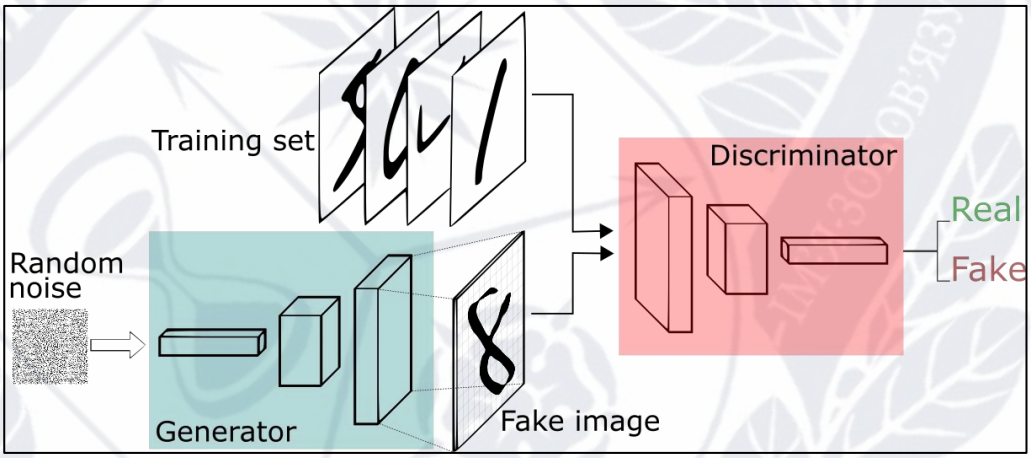

Рисунок 1.5 – Схема роботи генеративної змагальної мережі [12]

Long Short-Term Memory Networks є варіацією рекурентних нейромереж [13], яка вирішує проблему зникання градієнту. Вони мають спеціальні механізми пам'яті, що дозволяють зберігати і використовувати інформацію на тривалий час. Це робить їх ефективними для роботи з послідовностями, де важливо пам'ятати контекст на велику кількість часових кроків.

Модель **Encoder-Decoder Neural Network** складається із двох рекурентних нейронних мереж [14]: кодера, який опрацьовує вхідні дані, та декодера, який опрацьовує вихідні дані. Кодер і декодер можуть працювати одночасно, використовуючи однакові параметри, або вони можуть використовувати різні набори параметрів.

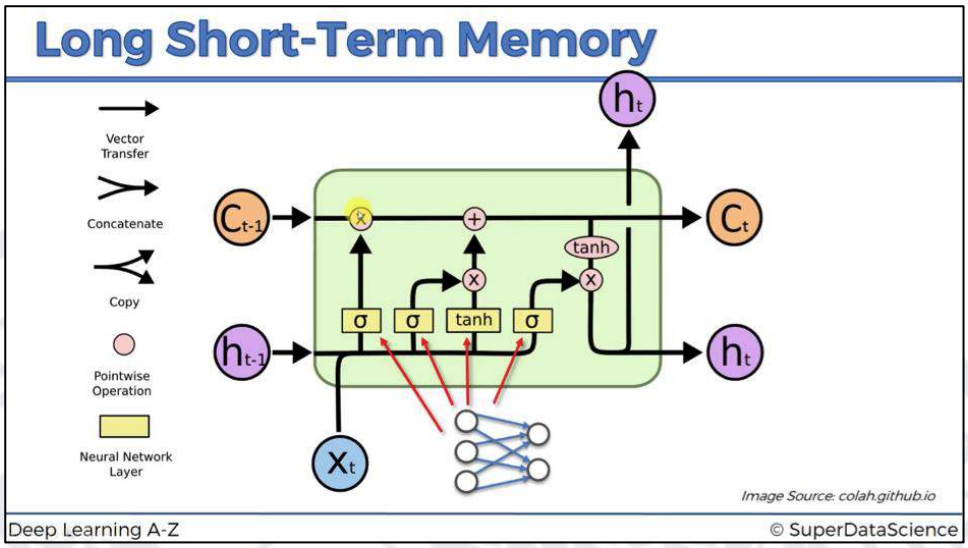

Рисунок 1.6 – Розбір довгої короткочасної пам'яті [13]

Ця модель відрізняється від звичайних рекурентних нейронних мереж тим, що її ефективність особливо проявляється в ситуаціях, коли довжина вхідних даних дорівнює довжині вихідних даних. Незважаючи на те, що вона має ті ж переваги та обмеження, що й звичайні RNN, ця модель часто застосовується в чат-ботах, системах машинного перекладу та системах відповідей на запитання.

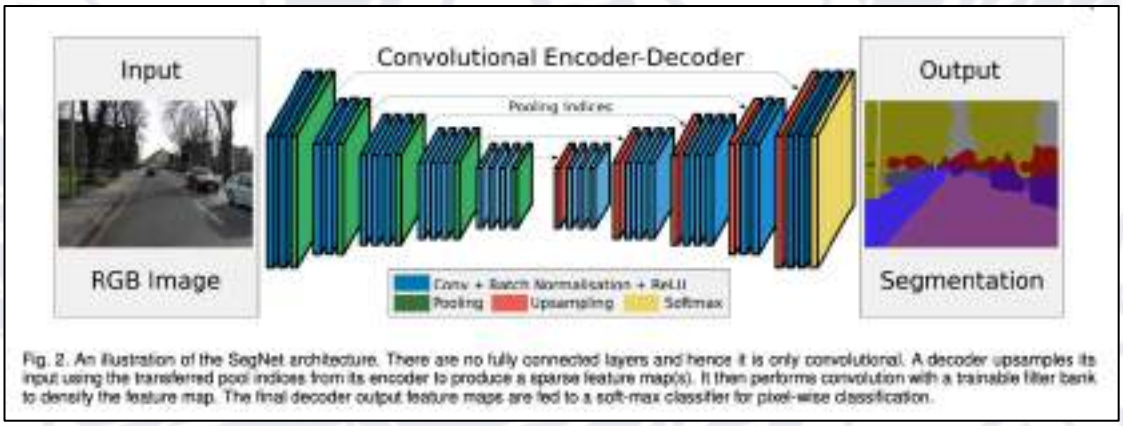

Рисунок 1.7 – Приклад нейронної мережі енкодер-декодер [14]

Продовжуючи аналогію з людським мозком, коли формуються зв’язки між нейронами вони мають закріпитися для того, щоб ефективно передавати імпульси. Для того, щоб ці зв’язки закріпились, потрібні дані, на яких вони будуть вчитися і підлаштовувати свої ваги (сили імпульсів), щоб найбільш точно виконувати поставлену перед ними задачі. Зазвичай, при тренуванні, корекція

ваг йде незначна, тому потрібна дуже велика кількість інформації та часу, а також обчислювальних потужностей, через що в минулому їм приділялася дуже мала частка уваги дослідників. Проте, зовсім недавно настали сприятливі умови для розвитку цієї сфери, і ми отримали досить багато нових та більш якісних підвидів мереж, які тепер можуть бути натреновані не тільки для виконання лиш однієї задачі. Візьмемо найбільші та найвидатніші ШІ сучасності, а саме ChatGPT, MidJourney, Google Bard / Gemini, DALL-E 2 AI, GitHub Copilotю

ChatGPT є однією з найбільш популярних моделей чатботів [15], розроблених компанією OpenAI. Вона базується на архітектурі трансформаторів та навчена на великому обсязі текстових даних з метою розуміння та генерації природних мовленнєвих відповідей. ChatGPT використовується у багатьох сферах, включаючи онлайн-підтримку, відповіді на запитання, створення контенту для соціальних мереж та багато іншого. Ця модель постійно навчається на нових данних та вдосконалює свої навички для кращого взаємодії з користувачами.

MidJourney – це програма штучного інтелекту [16], що генерує зображення з текстового опису. Вона була заснована в лютому 2022 року Девідом Хольцем, співзасновником Leap Motion, і з 12 липня 2022 року перебуває у відкритому бета-тестуванні. Сама програма може генерувати зображення з роздільною здатністю до 1000 мегапікселів, що забезпечує неймовірну деталізацію та чіткість, а також може створювати зображення в різних стилях, від реалістичних до художніх, абстрактних та сюрреалістичних. Це робить його потужним інструментом для концептуального мистецтва, візуалізації даних, розваг та освіти.

Google Bard (іноді відомий як Google Gemini) – це чат-бот на основі штучного інтелекту [17], розроблений компанією Google AI. Він ґрунтується на сімействі великих мовних моделей (LLM) LaMDA та PaLM, і володіє широким спектром можливостей, що робить його потужним інструментом для багатьох завдань. Бот може генерувати текст людської якості у відповідь на широкий спектр підказок і запитань. Він може писати різні творчі текстові формати, такі

як вірші, код, сценарії, музичні твори, електронні листи, листи тощо, а ще може отримувати доступ до інформації з реального світу та обробляти її за допомогою Google Search, що робить його цінним джерелом знань.

DALL-E 2 це неймовірна система штучного інтелекту [18], яка генерує зображення з текстових описів. Наступник оригінального DALL-E, DALL-E 2 вражає своїми можливостями, створюючи фотореалістичні, художні та абстрактні зображення на основі простих текстових підказок. ШІ пропонує інтерактивні можливості, що дозволяє користувачам вносити зміни до зображень під час їх генерування. Це дає можливість глибоко контролювати творчий процес та отримувати бажані результати. На додачу, DALL-E 2 постійно вдосконалюється та оновлюється завдяки дослідницьким зусиллям команди OpenAI. Це означає, що її можливості постійно розширюються, а якість зображень постійно зростає.

GitHub Copilot – це інноваційний інструмент штучного інтелекту [19], який допомагає розробникам програмного забезпечення писати код швидше та ефективніше. Copilot використовує машинне навчання для генерування коду, пропонування підказок та виправлення помилок, що робить його цінним доповненням до робочого процесу розробників. Він може автоматично генерувати код на основі природного мовного опису або коментарів до коду. Це може заощадити багато часу та зусиль, особливо якщо людина працює над незнайомою технологією. Ще Copilot може виявляти та виправляти помилки у коді, що може допомогти уникнути багів та збоїв. Також його корисним функціоналом буде пропонування підказок та фрагментів коду, які допоможуть завершити завдання швидше.

Підсумовуючи, можна сказати, що штучний інтелект стрімко розвивається, демонструючи неймовірний прогрес у різних сферах, від генерування зображень фотореалістичної якості до створення самокерованих автомобілів. Його вплив на наше життя стає дедалі більш відчутним, трансформуючи те, як ми працюємо, навчаємося, розважаємося та спілкуємося.

1.2 Інтеграція можливостей штучного інтелекту у програмні продукти.

Інтеграція штучного інтелекту у програмні продукти стрімко перетворюється з футуристичної концепції на звичайну реальність. Сьогодні ШІ пронизує всі аспекти програмного забезпечення, від автоматизації завдань до персоналізації користувацького досвіду. В цьому пункті буде коротко описані приклади впровадження ШІ в сучасному світі.

Звичайно, як і будь-яка інша програма, ШІ використовується для автоматизація рутинних завдань, щоб звільняти час розробників та користувачів від повторюваних та трудомістких завдань: тестування програмного забезпечення, аналіз коду та виявлення помилок, створення звітів та документації, ведення даних та обробка інформації, тощо. Яскравими представниками продуктів, які спеціалізовані на цьому є **GitLab** [20], **UiPath** [21], **DeepL** [22].

Також ШІ може бути використаний для бізнес-аналітики. Він може використовуватися для аналізу великих обсягів даних, щоб виявляти тенденції, закономірності та аномалії; для прогнозування майбутніх подій, таких як продажі, попит на продукти та ринкові тенденції; для сегментації клієнтів на основі їхніх характеристик та поведінки; для персоналізації маркетингових кампаній, рекомендацій продуктів та обслуговування клієнтів. Прикладами такого використання ШІ є **Tableau** (платформа візуалізації даних, яка використовує ШІ для автоматичного створення візуалізацій даних, що полегшує розуміння даних) [23], **Qlik Sense** (платформа аналітики бізнесу, яка використовує ШІ для автоматичного виявлення закономірностей та тенденцій у даних) [24], **SAS Viya** (платформа аналітики даних, яка використовує ШІ для прогнозування майбутніх подій та оптимізації бізнес-процесів) [25].

Ще одна корисна річ, яку ШІ можуть виконувати в продуктах, є обслуговування клієнтів. Для такого його застосування створюються чат-боти, що можуть використовуватися для автоматизації обслуговування клієнтів, надаючи відповіді на поширені запитання та підтримку, та віртуальні помічники,

які можуть використовуватися для надання допомоги клієнтам у виконанні завдань, таких як бронювання, відстеження замовлень та отримання підтримки. Ще ШІ в цій сфері може використовуватися для аналізу настроїв клієнтів з відгуків, оглядів та повідомлень у соціальних мережах і для виявлення шахрайства у транзакціях та захисту клієнтів від кібератак. Увагу можуть привернути такі представники цього напрямку: **ChatGPT** (чат-бот на основі ШІ, який може вести розмови з клієнтами природною мовою) [14], **Amazon Alexa** (віртуальний помічник, який використовує ШІ для розуміння голосових команд і виконання завдань, таких як відтворення музики, встановлення будильників та управління розумним будинком) [26], **Zendesk Support** (система обслуговування клієнтів, яка використовує ШІ для автоматизації типових завдань клієнтів, таких як маршрутизація квитків, відповіді на поширені запитання та аналіз настроїв клієнтів) [27].

Останній в цьому підрозділі, але неостанній спосіб інтеграції ШІ в сучасні продукти є його використання для освіти. Ці розумні алгоритми можуть використовуватися для персоналізації навчальних матеріалів та завдань на основі потреб та стилю навчання кожного учня, ще для автоматизованої оцінки завдань, що може звільнити час для вчителів, щоб вони могли зосередитися на індивідуальній підтримці учнів, також для надання учням зворотного зв'язку в режимі реального часу під час навчання і для створення захоплюючих та інтерактивних навчальних середовищ, які можуть мотивувати учнів навчатися. Успішними продуктами в цьому векторі є **Google Classroom** (платформа для онлайн-навчання, яка використовує ШІ для персоналізації навчального досвіду для кожного учня) [28], **Duolingo** (програма для вивчення мов, яка використовує ШІ для адаптації навчальних уроків до потреб кожного учня. Вона може відстежувати прогрес кожного учня та надавати зворотний зв'язок у режимі реального часу) [29], **DreamBox Learning** (платформа для математичної освіти, яка використовує ШІ для персоналізації навчальних завдань для кожного учня. Вона може адаптувати складність завдань на основі здібностей кожного учня) [30].

Як можна побачити, штучний інтелект стрімко розвивається, відкриваючи нові горизонти для багатьох сфер, а його інтеграція у програмні продукти дає змогу значно розширити їхні можливості, підвищити ефективність та створити принципово нові рішення.

1.3 Сервіси-аналоги

При пошуку аналогів основна вага надавалась екземплярам, які використовували штучний інтелект для генерування різного роду питань та тесів.

Знайдені системи не є загальнодоступними. В більшості це закриті системи, до яких доступ отримати неможливо. Проте, можна проаналізувати можливості цих систем на основі їх комерційних пропозицій.

Першим та найвідомішим продуктом, який пропонує функції створення тестових питань і не тільки є **Quillionz** [1].

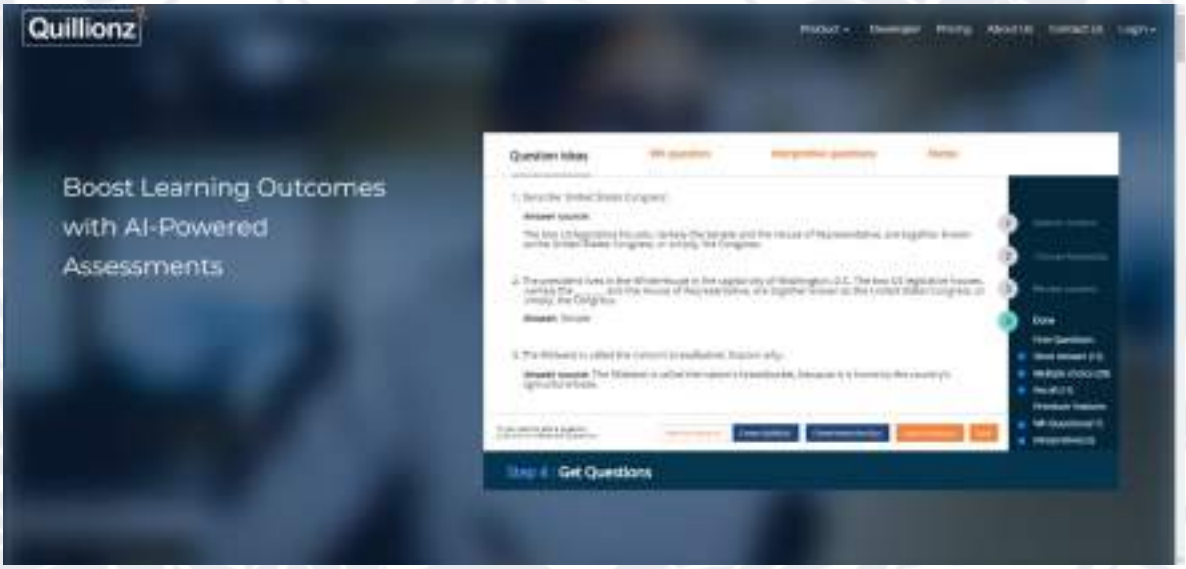

Рисунок 1.8 – Головна сторінка сайту Quillionz [1]

Quillionz – це платформа на основі штучного інтелекту, яка генерує питання, конспекти та тести з різних тем. Це потужний онлайн-інструмент, який використовує алгоритми машинного навчання для перетворення вхідного контенту (текст, веб-сторінки, відео тощо) на різні типи питань, а також для створення коротких конспектів та тестів. Якщо подивитись на його функціонал, то можна виділити такі основні можливості:

- Генерація питань: Quillionz може генерувати різні типи питань,

включаючи:

- Тести з множинним вибором (MCQ)
- Короткі описові питання
- Питання на відповідність
- Заповнення пропусків
- Питання на аналіз

- Створення конспектів: Quillionz може автоматично створювати короткі конспекти з вхідного тексту, виділяючи ключові моменти та ідеї.
- Інтеграція: Quillionz можна інтегрувати з іншими навчальними платформами, такими як LMS (системи управління навчанням) та онлайн-курси.
- Підтримка різних мов: Quillionz може генерувати питання та конспекти на різних мовах.

Ще одним аналог є **PrepAI** [2].

Рисунок 1.9 – Головна сторінка сайту PrepAI [2]

PrepAI – це інструмент для генерації питань на основі технології NLP (natural language processing), розроблений DataToBiz. Цей онлайн-інструмент нещодавно оновили концепціями таксономії Блума, щоб створювати більш ефективні та аналітичні питання, які спонукають студентів оцінювати тему та висловлювати свою точку зору. Він приймає різні формати вхідних даних, такі

як текстові файли, docx/doc і PDF-файли, а також URL-адреси відео та відеофайли. Він також має вбудований інструмент пошуку, який збирає дані безпосередньо з інтернету за заданою темою. Цей вміст потім перетворюється на різноманітні питання (тестові завдання з множинним вибором, на заповнення пропусків, істинні/хибні та описові), які перевіряють у студентів навички вищого порядку мислення (HOTS).

Відрізняє PrepAI його здатність генерувати питання для всіх шести рівнів таксономії Блума. Тестові роботи міститимуть питання, які перевірять наступне:

- Знання, отримані студентами на основі їхньої здатності до запам'ятовування.
- Здатність студентів розуміти контекст питання (їхні навички розуміння).
- Як студенти використовують свої знання в нових або різних ситуаціях.
- Аналітичний підхід студентів до вирішення проблеми.
- Способи, якими студенти можуть створити щось нове на основі існуючих знань.
- Як студенти оцінюють питання та захищають свою думку на основі того, що вони вивчили.

Згенеровані питання можна редагувати та оцінювати окремо. Ви можете видаляти небажані питання та додавати вручну більше.

Третім і останнім для розгляду є **EdApp** [3].

EdApp – це комплексна система управління навчанням (LMS) з широким набором функцій. Одна з них – **Rapid Refresh**. Це онлайн-конструктор тестів, остання розробка, доступна на EdApp. Він дозволяє викладачам та тренерам перевіряти, наскільки добре студенти та працівники засвоїли матеріал та чи можуть вони застосовувати свої новонабуті знання шляхом генерування питань різного виду, а саме тести з множинним вибором, симуляція чату, карусель, питання на основі зображень, обведення кола з відповіддю, істинно/хибно, знайди слово, переплутані слова тощо. Ці питання створюються на панелі керування LMS для кожного студента. Результати підраховуються, а додаток автоматично оновлює таблицю лідерів. Викладачі можуть завантажувати

питання на платформу, автоматизувати оцінювання та налаштовувати сертифікати переможців. EdApp також може перекладати тести та опитування на понад 100 мов за допомогою хмарного інструмента перекладу компанії.

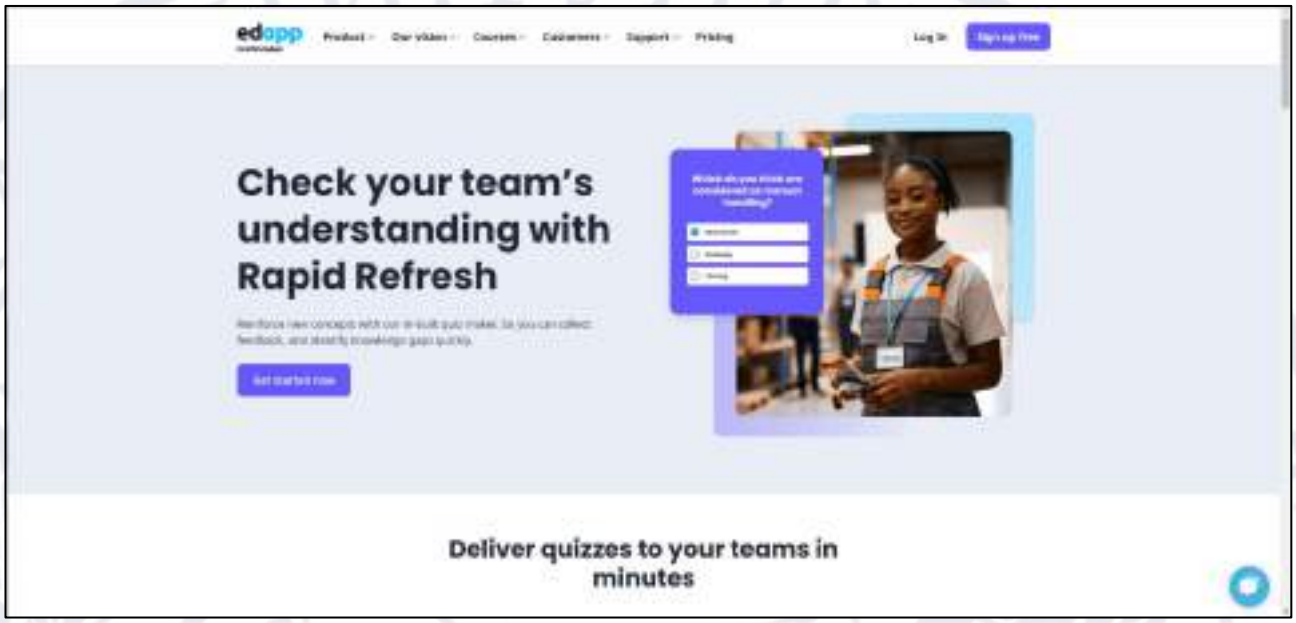

Рисунок 1.10 – Головна сторінка сайту EdApp [3]

1.4 Висновки до розділу 1

У рамках даного розділу було розглянуто штучний інтелект, його види та розвиток на даний момент часу. Також були наведені основні види нейронних мереж та розписано, для яких задач який вид використовується. Далі були оглянути найпопулярніші моделі ШІ. Потім були описані шляхи інтеграція штучного інтелекту у програмні продукти. Далі була сформульована постановка задачі та наведені сервіси аналоги. В кінці був зроблений огляд мов та інструментів, які можуть бути використані в написанні додатку, який є ціллю цієї магістерської роботи.

РОЗДІЛ 2
МОДЕЛЮВАННЯ СИСТЕМИ

2.1 Постановка задачі

Необхідно розробити систему автоматичного створення та валідації тестових питань на основі взаємодії з мовною моделлю ChatGPT. Система має забезпечувати автоматизоване формування тестів з урахуванням тематики, складності, а також рівня підготовки студентів, що дозволить значно скоротити час і ресурси, необхідні для розробки якісних тестових завдань, та підвищити об’єктивність їх оцінювання. Під тестовими завданнями мається на увазі набір питаннь з множинним вибором, тобто питання та декілька варіантів відповідей на нього, з яких тільки одне вірне.

Дана система буде розрахована для користування одним університетом, це накладає такі умови:

- Приблизне навантаження, яке має витримувати система складає 1000 активних запитів
- Система має зберігати інформації про різні об’єкти університету, а саме факультети, кафедри, викладачів, спеціальності, освітні програми, дисципліни, їхні модулі, їхні теми, групи та їхніх студентів
- Для швидкого пошуку конкретного тесту, він буде містити в собі таку інформацію як дату та час створення, тему, викладача, який його створив та студента, який його має вирішити
- Система має оцінити правильність виконання тесту студентом одразу після відправки ним заповнених бланків відповідей
- Ця система має бути виконана як WebAPI для легкої інтеграції на різні середовища (веб-сайт, мобільний застосунок, десктопний застосунок) в майбутньому
- Якщо говорити про формат вхідних даних, то викладачу, щоб згенерувати тест, потрібно надати такі параметри:
    - Текст у формі текстового файлу

- Кількість питань у формі числа
- Індивідуальні характеристика студента у формі тексту
- Тема, яку він обере зі списку доступних

- На виході, викладач отримає тестові питання наступного формату:
  - Питання у формі рядка
  - Масив з чотирьох відповідей у форматі «ключ - значення»
  - Ключ правильної відповіді

2.2 Огляд інформації, що має зберігатись

Оскільки дана система має зберігати інформацію про вищеописані об’єкти, то для цього знадобиться використати базу даних. В базах даних інформація представлена у вигляді таблиць, тому в даному підрозділі буде описаний їхній вигляд

Таблиця 2.1. – Опис даних, що будуть зберігатись в базі даних системи

Факультет	Назва, адреса
Кафедра	Назва, факультет, набір викладачів, набір дисциплін, набір спеціальностей
Викладач	ПІБ, дата народження, кафедра, вчене звання, вчений ступінь
Спеціальність	Назва, кафедра, набір освітніх програм
Освітня програма	Назва, спеціальність, набір дисциплін, набір груп
Дисципліна	Назва, кафедра, набір модулів, семестр, кількість кредитів, форма підсумкового контролю
Модуль	Назва, набір тем, дисципліна
Тема	Назва, модуль
Група	Назва, рік, курс, освітня програма, набір студентів
Студент	ПІБ, дата народження, група, курс
Розподіл дисциплін	Дисципліна, викладач, група
Тест	Набір питань, викладач, студент, тема, дата та час створення

Бланк відповідей	Відповіді, студент, тест, дата та час створення

2.3 Вибір архітектури

В даному підрозділі будуть оглянуті різні типи архітектур, їхні переваги та недоліки, а також буде зроблений вибір того виду, який найкраще підходить під досліджувану систему.

Розпочнімо з моноліту. **Монолітна архітектура** – це традиційний підхід до проектування програмного забезпечення[44-47], коли весь застосунок створюється як єдиний, тісно інтегрований блок. Усі його компоненти, включаючи користувацький інтерфейсбізнес-логіку та управління даними, функціонують у межах одного кодового базису та розгортаються разом. Такий підхід робить монолітні системи відносно простими для розробки, підтримки й розгортання, особливо для невеликих проєктів або стартапів.

- Основні характеристики:
  - Єдиний кодовий базис: Уся система розробляється та підтримується в межах одного сховища коду.
  - Централізоване зберігання даних: Зазвичай використовує одну базу даних, до якої звертаються всі компоненти.
  - Тісно пов’язані компоненти: Частини системи взаємодіють напряму, що пришвидшує обмін даними.
  - Уніфіковане розгортання: Навіть для невеликих змін потрібно перевипустити всю систему.
- Переваги:
  - Простота: Легко зрозуміти, розробляти та налагоджувати, особливо невеликим командам.
  - Висока продуктивність: Компоненти взаємодіють напряму, що забезпечує швидший обмін даними.
  - Економія коштів: Початкова розробка й запуск дешевші, що особливо важливо для стартапів.
  - Послідовність: Зручне управління залежностями, версіями та

розгортанням.

- Швидкість запуску: Завдяки єдиній структурі, розробка проходить швидше.

- Недоліки:
  - Проблеми з масштабуванням: Для масштабування потрібно копіювати всю систему, що вимагає додаткових ресурсів.
  - Складність підтримки: З ростом застосунку код стає важким для управління, а зміни можуть викликати побічні ефекти.
  - Обмеження технологій: Усі компоненти повинні працювати на одній технологічній базі, що зменшує гнучкість.
  - Труднощі з розгортанням: Будь-які зміни потребують повного розгортання, що може збільшити ризик збоїв.
  - Єдина точка відмови: Помилка в одній частині може привести до збою всієї системи.
- Сфери застосування:
  - Монолітна архітектура підходить для стартапів і невеликих проєктів, де важливі простота й швидкість розробки. Вона також часто зустрічається у спадкових системах або в простих застосунках, які не потребують масштабування.

Наступним типом архітектури, який буде розглядатися, є мікросервісна архітектура. **Мікросервісна архітектура** – це підхід до проєктування програмного забезпечення, в якому система розділена на набір невеликих, незалежно працюючих служб (сервісів)[48-51], що комунікують через API. Кожен сервіс виконує окрему бізнес-функцію і може мати власну базу даних, технології та підхід до розгортання. Основна ідея мікросервісів – модульність і можливість швидкого адаптування до змін у потребах бізнесу.

- Основні характеристики
  - Мікросервіси є самостійними модулями з окремим життєвим циклом, які розгортаються незалежно один від одного. Сервіси взаємодіють через легковагові протоколи, такі як REST або gRPC.

Децентралізоване управління даними передбачає, що кожен сервіс керує власною базою, що підвищує гнучкість, але ускладнює забезпечення консистентності даних.

- Переваги
  - Масштабованість: Можливість масштабувати окремі сервіси замість всієї системи, що робить управління ресурсами ефективнішим.
  - Гнучкість розробки: Різні команди можуть працювати над різними сервісами паралельно, використовуючи найзручніші для них технології.
  - Відмовостійкість: Збій одного сервісу не впливає на роботу всієї системи.
  - Швидке впровадження змін: Оновлення та розгортання відбуваються локально для конкретного сервісу, що знижує ризики збоїв.
  - Технологічна різноманітність: Команди можуть використовувати різні мови програмування, інструменти й фреймворки, орієнтуючись на потреби конкретного сервісу.
- Недоліки
  - Складність управління: Потрібно організувати комунікацію між численними сервісами, забезпечити їхню безперебійну роботу та безпеку.
  - Витрати на інфраструктуру: Необхідні інвестиції в автоматизацію, оркестрацію (наприклад, Kubernetes) та DevOps.
  - Ускладнення тестування: Інтеграційне тестування системи з багатьох сервісів складніше, ніж у моноліті.
  - Підтримка консистентності даних: Використання асинхронних патернів, як-от Saga, потребує додаткових зусиль для узгодження транзакцій.
- Застосування

- Мікросервіси ідеально підходять для масштабних систем із високою частотою змін, таких як онлайн-магазини (наприклад, Amazon), платіжні системи (PayPal), стрімінгові сервіси (Spotify) та інші динамічні бізнеси, які потребують швидкої адаптації до змін.

Оскільки система тестування нова, ще не піддавалася довгому моніторингу та аналізу, відповідно висновки по користуванню та навантаженню зробити неможливо, її розробкою буде займатися одна людина, яка є експертом тільки в одному або двох мовах програмування та фреймворках, то раціональніше всього і по часу, і по ресурсам обрати монолітну архітектуру, але з таким розділенням компонентів всередині, щоб в майбутньому можна було легко відділяти окремі сервіси для переходу до мікросервісної архітектури, при потребі.

2.4 Вибір типу бази даних

Існує кілька основних видів баз даних, кожна з яких має свої унікальні характеристики, переваги та недоліки, залежно від потреб системи.

Реляційні бази даних (RDBMS) зберігають дані в табличній формі, де кожен рядок представляє запис, а кожен стовпець – властивість цього запису. Для організації даних використовуються первинні та зовнішні ключі, які забезпечують зв’язки між таблицями. Вони використовують SQL (Structured Query Language) для маніпуляцій даними.

- Переваги: підтримка транзакцій (ACID), висока консистентність даних, розширені можливості для складних запитів.
- Недоліки: складність масштабування, залежність від заздалегідь визначених схем.
- Приклади: MySQL, PostgreSQL, Oracle Database, Microsoft SQL Server.
- Реляційні бази підходять для фінансових систем, CRM або ERP-систем, де важлива консистентність і цілісність даних.

Документні бази даних зберігають дані у вигляді документів (зазвичай у форматі JSON або BSON). Вони дозволяють гнучко зберігати структуру даних, що може відрізнятися від одного документа до іншого.

- Переваги: гнучкість структури даних, висока швидкість роботи з

документами, горизонтальне масштабування.

- Недоліки: складніше забезпечити консистентність між документами.
- Приклади: MongoDB, CouchDB.
- Використовуються у веб-застосунках, де обробляються динамічні або неструктуровані дані, наприклад, блоги чи каталоги товарів.

Колонкові бази даних організовують дані за колонками, а не за рядками. Це дозволяє швидко виконувати аналітичні запити, що вимагають обробки великих обсягів даних.

- Переваги: ідеальні для аналітичних задач і швидкого доступу до агрегованих даних.
- Недоліки: не оптимальні для запису транзакцій або взаємопов’язаних даних.
- Приклади: Apache Cassandra, Google Bigtable.
- Використовуються в системах обробки аналітичних даних та звітності.

Ключ-значення бази даних зберігають дані у вигляді пар "ключ-значення". Вони швидкі та прості, але менш підходять для складних зв’язків між даними.

- Переваги: надзвичайна продуктивність і простота.
- Недоліки: обмежені можливості для складних запитів.
- Приклади: Redis, DynamoDB.
- Підходять для кешування даних або зберігання сесій користувачів.

Графові бази даних зберігають дані у вигляді вузлів і зв’язків між ними. Вони оптимальні для роботи з взаємозв’язками, такими як соціальні мережі чи рекомендативні системи.

- Переваги: ефективність для складних зв’язків і запитів, таких як пошук найкоротшого шляху.
- Недоліки: менш ефективні для традиційних задач зберігання даних.
- Приклади: Neo4j, Amazon Neptune.
- Підходять для роботи з графовими структурами, такими як графи

соціальних зв’язків або схеми доріг.

Переходячи до вибору типу бази даних для досліджуваної системи, то з огляду на велику кількість взаємопов’язаних об’єктів (наприклад, зв’язок між кафедрами, факультетами, дисциплінами, студентами тощо), **реляційна база даних** є найкращим вибором для цієї системи. Вона дозволить забезпечити зв’язність між таблицями через первинні та зовнішні ключі (наприклад, Кафедра пов’язана з Факультетом, а Дисципліна– з Кафедрою), використовувати складні SQL-запити для аналізу, звітності та швидкого пошуку потрібних даних (наприклад, пошук усіх тестів, створених певним викладачем), реалізувати транзакції, які будуть важливі для забезпечення консистентності даних (наприклад, якщо додається новий тест, всі його питання і розподіли дисциплін також мають коректно зберігатися).

2.5 Огляд сучасних підходів, що використовуються у тестуванні

Сучасні підходи до тестування в освіті вимагають не лише якісного складання тестових завдань, а й їх детального аналізу на основі отриманих результатів. Такий аналіз дозволяє не лише оцінювати рівень знань студентів, але й виявляти слабкі місця в методиці навчання, коригувати формулювання запитань для кращого розуміння та вдосконалювати самі тестові завдання. У цьому контексті важливим стає впровадження інноваційних методів обробки відповідей, а також використання автоматизованих систем для аналізу результатів тестування, які забезпечують більш об’єктивне оцінювання. Нижче наведено огляд трьох статей, які присвячені різним підходам до аналізу тестових завдань, з акцентом на методах оцінювання правильності відповідей і подальшої адаптації тестових питань.

Загальний огляд статей демонструє сучасні підходи до вдосконалення автоматизованого тестування [36-39, 43, 44], спрямовані на підвищення об'єктивності оцінювання та адаптивності тестових завдань. Автори статті [36] зосереджуються на створенні експертної системи, яка використовує методи логічних мікромашин для автоматизованого аналізу відповідей студентів. Система застосовує метрики Левенштейна та Декартову відстань для визначення

схожості між введеними відповідями та еталонними значеннями, що дозволяє враховувати незначні відмінності у відповідях. Такий підхід дозволяє виявити випадкові помилки та підвищити об'єктивність тестування, уточнюючи до 6,19% відповідей, з яких 20,68% були змінені на правильні, що значно підвищує точність оцінювання знань. Стаття [37] продовжує тему об’єктивності, фокусуючись на методі зворотного аналізу помилок студентів для покращення якості тестових завдань. Дослідники вивчають помилки, зроблені студентами, щоб визначити фактори, які впливають на вибір неправильних варіантів. Автори зазначають, що деякі формулювання питань можуть спричиняти помилки через нечіткість або надмірну схожість варіантів, і пропонують змінювати формулювання питань для покращення їх зрозумілості. Такий аналіз сприяє адаптації тестового середовища відповідно до рівня знань студентів. Стаття [39] розглядає технічний аспект створення комп'ютерної системи тестування на базі трирівневої бази даних для забезпечення зручності зберігання та аналізу результатів. Така структура дозволяє зберігати дані про відповіді, самі тестові завдання, студентів і їхні результати, а за допомогою SQL-запитів автоматично генерувати звіти. Це допомагає викладачам виявляти складні питання, які викликають труднощі у студентів, та адаптувати матеріали. Застосування трирівневої бази даних забезпечує зручність масштабування системи та можливість адаптації до потреб різних навчальних закладів, що сприяє підвищенню ефективності навчального процесу. Дослідження в статті [43] також звертає увагу на об’єктивність тестування. Авторка зазначає, що студенти з високим рівнем підготовки можуть показувати низькі результати через надмірну увагу до деталей у питаннях, тоді як слабші студенти іноді отримують позитивні оцінки завдяки вгадуванню. Пропонується вдосконалена автоматизована система, що дозволяє формувати питання різного типу (альтернативний вибір, підбір пари, виключення зайвого), які краще відображають практичні знання студентів. Це дозволяє значно підвищити об'єктивність оцінювання, мінімізуючи вплив випадкових відповідей, та робить систему гнучкішою й адаптивнішою до особливостей різних дисциплін і рівнів підготовки студентів. Ще з статті [44]

дуже корисною є ідея про зворотній зв’язок та можливості адаптації завдань на основі результатів кожного учня. Якщо розглядати статтю детальніше, то в ній досліджується використання комп’ютерного тестування як ефективного інструменту педагогічної діагностики знань учнів з теми «Табличний процесор». Основний акцент робиться на автоматизації процесу оцінювання та можливості коригування навчальних завдань для поліпшення навчального процесу. Тестова система, реалізована в програмі Expert, дозволяє проводити не лише базове оцінювання знань, але й виявляти прогалини у засвоєнні матеріалу. Це досягається завдяки зворотному зв’язку та можливості адаптації завдань на основі результатів кожного учня.

Система тестування включає завдання різного типу, що сприяє об’єктивному визначенню рівня інформаційної компетентності. Використання таких завдань допомагає уникнути впливу випадкових відповідей на оцінку та дозволяє учням демонструвати практичні знання. Крім цього, система забезпечує індивідуальний підхід до кожного учня, даючи можливість замінювати або змінювати завдання залежно від рівня підготовки. Такий підхід не тільки підвищує точність оцінювання, а й позитивно впливає на мотивацію учнів, оскільки вони бачать, що система враховує їхні індивідуальні особливості.

Автор зазначає, що використання такої системи тестування сприяє формуванню стійкого інтересу до предмету та полегшує процес засвоєння знань, оскільки учні можуть наочно бачити власні прогалини та досягнення.

Усі методи, наведені в статтях, можуть бути тісно пов’язані з процесом валідації тестових питань за допомогою моделей штучного інтелекту, зокрема ChatGPT. Завдяки своїй здатності до обробки природної мови, ChatGPT може здійснювати аналіз та вдосконалення тестових завдань аналогічно методам, запропонованим у статтях.

Якщо узагальнити всі статті, то можна виділити декілька основних методів. **Методи логічних мікромашин** з статті [36], які використовують метрики Левенштейна та Декартової відстані, можуть бути частково відтворені в моделі ChatGPT через її здатність аналізувати текст на схожість, визначаючи,

коли відповіді є близькими до правильного варіанту. Це дозволяє автоматично оцінювати та уточнювати відповіді студентів, аналогічно до системи логічних мікромашин, підвищуючи об'єктивність оцінювання. Метод **зворотного аналізу помилок студентів** із статей [37, 43, 44] також може бути підтриманий ChatGPT шляхом проведення аналізу відповіді студентів на правильність та надання рекомендацій з адаптації тестових запитань. За допомогою штучного інтелекту можна отримувати розгорнуті звіти, в яких зазначається, як покращити формулювання завдань, щоб уникнути частих помилок. Це дозволить полегшити роботу викладачів, зменшити кількість неочікуваних помилок, та забезпечити максимальне розуміння завдань студентами. Метод з використанням **трирівневої бази даних** із статті [39] в комбінації з ChatGPT може стати потужним інструментом для обробки результатів тестування та їхнього аналізу. Інтеграція моделі ШІ з системою на основі багаторівневої бази даних забезпечить автоматичний аналіз великих обсягів відповідей студентів. Це дозволить легко виявляти складні питання та автоматично коригувати тести залежно від потреб студентів, роблячи систему ще більш масштабованою та адаптивною до конкретних умов.

Отже, ChatGPT може значно підвищити ефективність валідації тестових завдань, об'єднуючи методи із сучасних досліджень та забезпечуючи розширені можливості для аналізу та вдосконалення тестових завдань, що дозволяє створити більш адаптивну, об’єктивну та ефективну систему тестування

2.6 Моделювання системи

В даному підрозділі буде коротко описана архітектура застосунку, а саме модулі програми та їхня взаємодія один між одним.

Отже, цей додаток складається з таких частин:

- **Клієнт.** Клієнтська частина відповідає за ініціацію запитів до системи через Web API. Це може бути будь-який інтерфейс, наприклад, веб-додаток, мобільний застосунок або CLI.
- **Web API.** Цей модуль приймає запити від клієнта та передає їх до модуля основної логіки. Відповідає за маршрутизацію, валідацію

вхідних даних та повернення результатів у вигляді HTTP-відповідей.

- **Модуль основної логіки.** Центральна частина системи, яка координує взаємодію між усіма компонентами: приймає запити від Web API; обробляє дані клієнта (наприклад, запити на генерацію тестів або збереження в базу даних); використовує інші модулі для роботи з базою даних та ChatGPT.
- **Модуль взаємодії з БД.** Відповідає за: запис даних (наприклад, збереження згенерованих тестів), читання даних (отримання збережених тестів для аналізу або повторного використання), підключається до бази даних для виконання CRUD-операцій.
- **База даних.** Зберігає всі дані, необхідні для роботи системи: теми тестів, список тестових питань, відповідей та правильних варіантів.
- **Модуль взаємодії з ChatGPT.** Забезпечує інтеграцію з API ChatGPT для генерації тестових питань: формує запити на основі теми, наданої клієнтом; бробляє відповіді від ChatGPT, перетворюючи їх у структуру, що підходить для збереження в базу або виводу клієнту.
- **ChatGPT.** Використовується для генерації тестових питань на основі запитів від модуля основної логіки.

Якщо таку архітектуру зобразити на схемі, то вона вийде наступною:

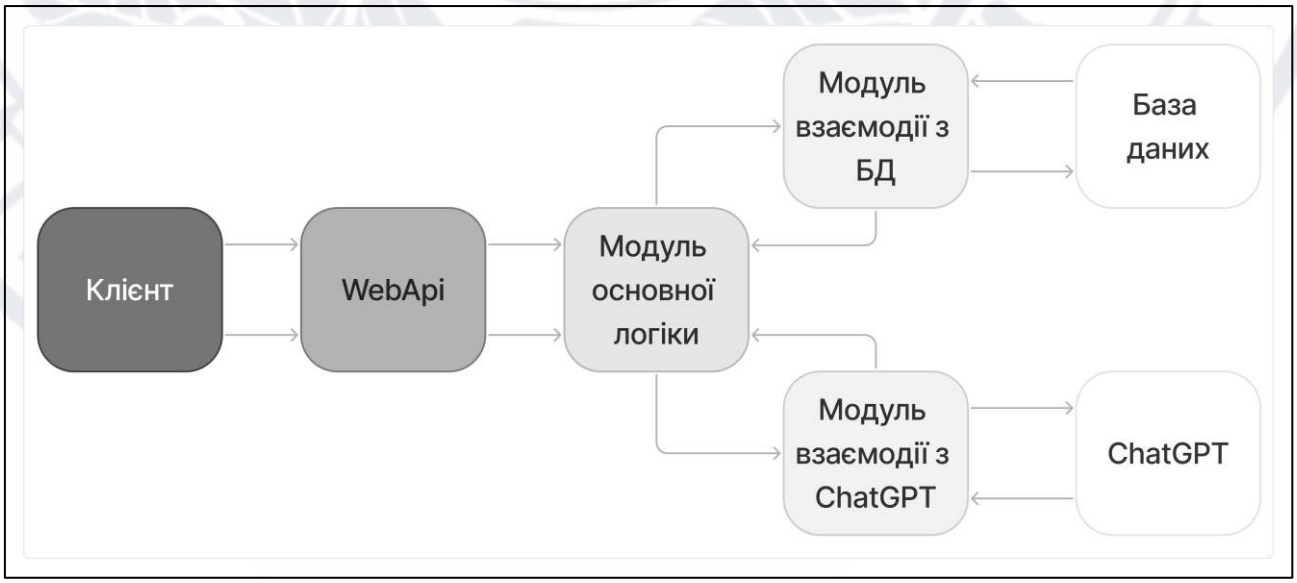

Рисунок 2.1 – Високорівнева схема застосунку

Якщо спуститися на нижчий рівень опису архітектури, то треба зазначити, що буде використана чиста архітектура[60], оскільки вона надасть найбільшу гнучкість та простоту у масштабуванні. Пристосовуючи дану архітектуру до нашої системи, на **шарі сутностей** будуть знаходитись просто моделі таких об’єктів як факультет, кафедра спеціальність і тд. Далі, вище, у **шарі застосування**, буде знаходитись логіка застосунку, яка буде покривати основні сценарії використання цієї системи, наприклад, логіка генерації тестів або логіка збереження тестів у базу, або перевірка відповідей студентів. Ще вище, у **шарі адаптерів інтерфейсів**, буде знаходитись логіка для адаптації зовнішні входів/виходів до формату, зрозумілого системі. Наприклад, веб-контролери для прийому HTTP-запитів, реалізація збереження даних у базі, форматування вихідних даних і тд. І, наостанок, зовнішній **шар інфраструктури** буде містити в собі саму базу даних та сервіс для комунікації з ChatGPT.

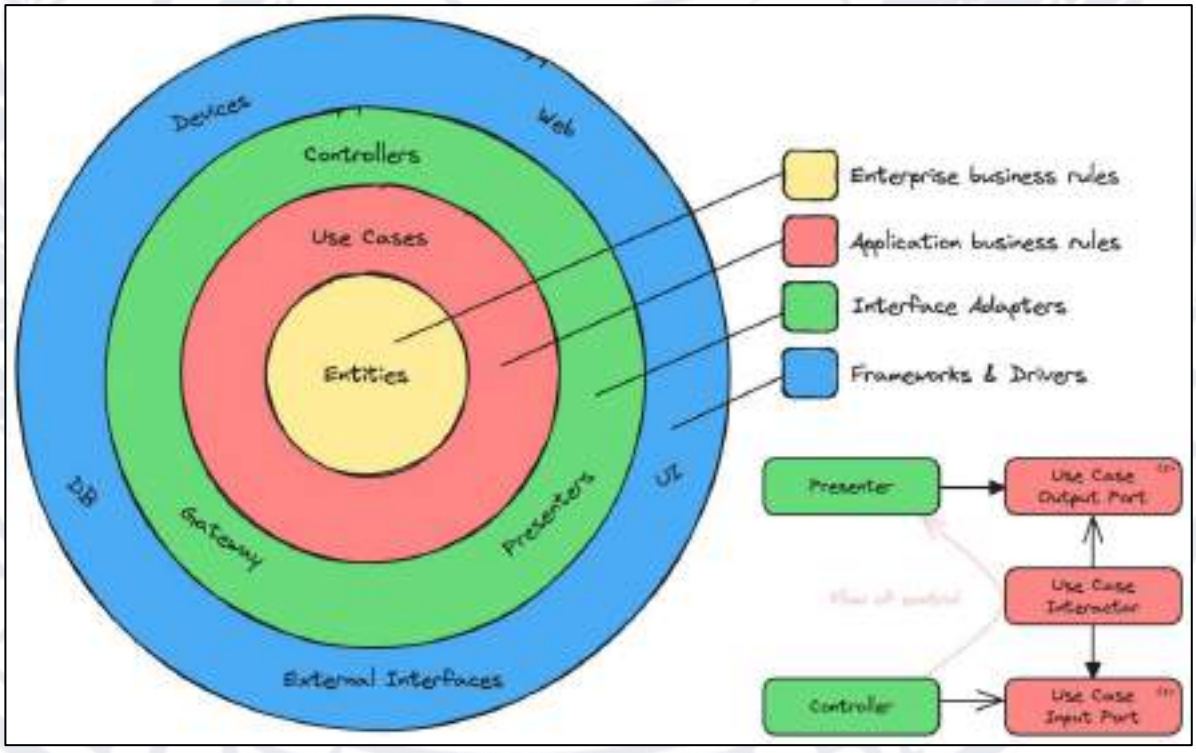

Рисунок 2.2 – Схема чистої архітектури

Ця модель застосунку забезпечить оперативне виконання основних функцій без витрат часу на впровадження зайвих або складних компонентів.

Крім того, завдяки своїй простій архітектурі, вона дозволить швидко обробляти запити користувачів, що стане значною перевагою, оскільки зекономить час викладачів на підготовку навчальних матеріалів.

2.7 Моделювання бази даних

В даному підрозділі буде наведена візуальна інтерпретація бази даних описана вище

Рисунок 2.3 – Схема бази даних

2.8 Взаємодія з іншими системами

Даний застосунок буде взаємодіяти з систему OpenAI. Вона відбувається на основі обміну HTTP-запитами із використанням REST API [40, 41]. Це дозволяє застосунку надсилати дані до OpenAI у вигляді запитів (наприклад, POST-запити) та отримувати відповіді у форматі JSON [42], який дуже легко можна розпарсити та структурувати будь-якою мовою програмування.

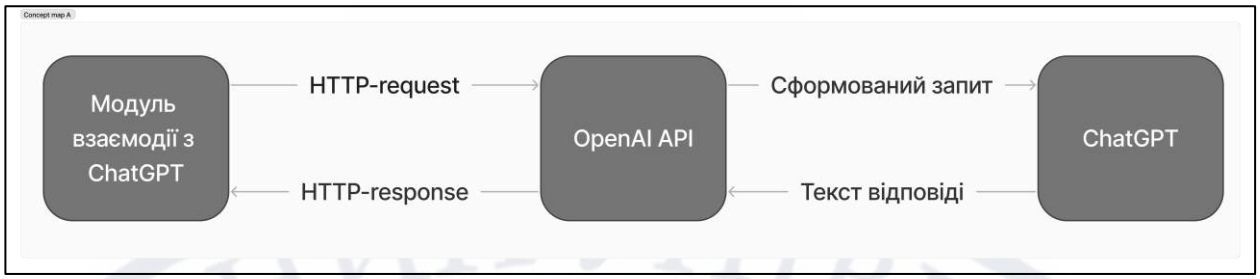

Рисунок 2.4 – Взаємодія з ChatGPT через OpenAI API

Цей процес приблизно відбувається таким чином:

- Коли користувач завантажує навчальний матеріал і задає параметри тесту (тему, складність, рівень підготовки учнів), застосунок формує текстовий запит, який буде надісланий до OpenAI API. У цьому запиті детально описуються вимоги для створення тестових запитань, наприклад: «створи 10 питань з множинним вибором на тему Data Science для студентів початкового рівня».
- Застосунок також передає ключ API, що підтверджує право доступу до сервісів OpenAI.
- Після отримання запиту OpenAI API використовує мовну модель (наприклад, GPT-3 або GPT-4) для генерації відповідей. Модель аналізує надісланий текст, розуміє, що необхідно створити тестові питання, та враховує всі задані параметри: кількість питань, складність, навчальну тему і стиль викладу.
- Мовна модель створює тестові питання відповідно до інструкцій, формулює варіанти відповідей та позначає правильний варіант для кожного питання.
- OpenAI API формує відповідь і надсилає її назад до застосунку у вигляді структурованого тексту або JSON, що містить згенеровані тестові питання з варіантами відповідей.

РОЗДІЛ 3

АНАЛІЗ ТА ВИБІР АКТУАЛЬНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Програмне забезпечення спирається на дві основні складові: архітектуру та інструменти розробки. У цьому розділі головну увагу буде приділено останнім. Зокрема, будуть описані основні бібліотеки, формати вхідних і вихідних даних, а також додаткові програмні інструменти, які використовувалися в процесі розробки.

3.1 Вибір мови програмування

В даному підрозділі буде проведений аналіз трьох найбільш популярних мов програмування для розробки WebAPI застосунків, а саме Java, Python та C#

Java є однією з найбільш популярних мов програмування для корпоративних рішень [https://www.ibm.com/docs/en/i/7.4?topic=platform-java-development-kit], що забезпечує кросплатформність завдяки JVM (Java Virtual Machine).

- Переваги:
  - Підтримка великомасштабних розподілених систем, таких як банківські або ERP-системи.
  - Наявність потужних фреймворків (Spring, Hibernate), які полегшують розробку веб-застосунків.
  - Висока стабільність і продуктивність у середовищах із високим навантаженням
- Недоліки:
  - Великий обсяг коду через більш детальний синтаксис.
  - Відносно складні процеси конфігурації (особливо для Spring Framework).

Java підходить для систем із необхідністю високого рівня платформної незалежності, але її складність може уповільнити розробку.

Python – це інструмент, що дозволяє швидко створювати прототипи завдяки простоті синтаксису та широкій екосистемі бібліотек [31].

- Переваги:
  - Простота вивчення та використання, швидкий старт для нових розробників.
  - Потужні бібліотеки для машинного навчання та аналітики, такі як Pandas і TensorFlow.
  - Популярні фреймворки (Django, Flask) дозволяють швидко створювати API
- Недоліки:
  - Відносно повільна швидкість виконання порівняно з Java і C#, через інтерпретовану природу.
  - Обмежені можливості для багатопотоковості через GIL (Global Interpreter Lock).

Python підходить для невеликих систем або проектів із акцентом на аналіз даних, але для високонавантажених систем його продуктивність може стати вузьким місцем.

C# – це мова програмування, створена компанією Microsoft [52, 60], яка добре підходить для створення високопродуктивних систем завдяки .NET-платформі.

- Переваги:
  - Продуктивність: Як компільована мова, C# забезпечує високу швидкість виконання порівняно з Python.
  - Гнучкість: ASP.NET Core дозволяє легко створювати веб-API та розподілені системи.
  - Масштабованість: .NET Core підтримує кросплатформність, тому застосунок може працювати на Windows, Linux та macOS.
  - Зручність розробки: Інструменти, такі як Visual Studio, спрощують налагодження, тестування та розгортання.
  - Екосистема: Entity Framework надає потужний ORM для роботи з базами даних, що ідеально підходить для реляційних моделей.
  - Інтеграція: Легка інтеграція з хмарними платформами (наприклад,

Azure) для масштабування та розгортання

- Недоліки:
  - Тісна інтеграція з екосистемою Microsoft може стати перешкодою, якщо потрібно використовувати інші платформи.

C# підходить для створення масштабованих і надійних систем, які легко підтримувати і розширювати.

Зважуючи всі наявні переваги та недоліки розглянутих мов програмування для розробки системи тестування C# є найкращим вибором через підтримка високого навантаження, легку інтеграцію з базами даних, продуктивність, гнучкість та швидкий розвиток

3.2 Вибір бази даних

Вибір між SQL базами даних залежить від вимог системи, особливо щодо продуктивності, масштабованості та зручності роботи з даними. Розглянемо основні особливості трьох найпоширеніших реляційних баз даних і їх придатність до використання у вашій системі тестування.

MySQL є однією з найпоширеніших open-source баз даних [53], особливо популярною для веб-додатків. Вона відзначається простотою використання, стабільністю та підтримкою великим співтовариством.

- Переваги:
  - Швидкість обробки операцій читання-запису, що робить MySQL зручним для систем з високою кількістю транзакцій.
  - Простота налаштування та інтуїтивно зрозумілий інтерфейс.
  - Різноманітність інструментів для керування базою даних, таких як MySQL Workbench.
- Недоліки:
  - Обмежена підтримка складних транзакцій і високорівневих SQL-можливостей.
  - Обмеження на роботу з JSON-даними та відсутність гнучкого масштабування при використанні великих обсягів даних

PostgreSQL – це об'єктно-реляційна база даних, яка відома своєю гнучкістю [54], підтримкою складних запитів і високою відповідністю стандартам SQL.

- Переваги:
  - Розширені функції для складних транзакцій і аналітики.
  - Підтримка JSON, геопросторових даних і користувацьких типів даних.
  - Добре підходить для систем із великим обсягом даних і складними запитами.
  - Велика кількість розширень для роботи з аналітикою, машинним навчанням і кластеризацією даних.
- Недоліки:
  - Більш складне налаштування порівняно з MySQL.
  - Підвищені вимоги до ресурсів під час виконання складних запитів

Microsoft SQL Server (MSSQL) – це комерційна база даних [55], яка інтегрується в екосистему Microsoft і відзначається високою продуктивністю для корпоративних рішень.

- Переваги:
  - Потужні засоби для роботи з аналітикою, звітністю та OLAP (онлайн-аналітична обробка).
  - Прекрасна інтеграція з хмарними платформами, такими як Azure, для масштабування.
  - Розширені функції безпеки, такі як шифрування даних і керування доступом на рівні рядків.
- Недоліки:
  - Висока вартість ліцензії, особливо для корпоративних рішень.
  - Залежність від екосистеми Microsoft, що може бути обмеженням у гетерогенних середовищах

Порівнюючи наведені факти раціональним вибором буде **PostgreSQL**, так як вона підтримує складні запити для аналітики результатів тестів, гнучка у

роботі з великими обсягами даних і сучасними форматами, такими як JSON (даний формат буде корисний для зберігання тестів), має високу масштабованість, що відповідає вимогам обробки одночасно до 1000 активних запитів.

3.3 Додаткові бібліотеки

Для більш легкого і гнучкого використання OpenAI API буде використано **Betalgo.Ranul.OpenAI**. [56]

Це потужний інструмент для інтеграції OpenAI API у .NET-додатки, який пропонує багато можливостей для роботи з різними моделями та адаптації під різні сценарії. Основні переваги цієї бібліотеки це:

- **Легка інтеграція.** Betalgo дозволяє швидко налаштувати інтеграцію з OpenAI API у .NET-проєктах, використовуючи простий синтаксис і підтримуючи dependency injection, що спрощує роботу з сервісами.
- **Підтримка різних моделей.** Бібліотека підтримує всі основні моделі OpenAI, включаючи GPT-3.5, GPT-4, DALL·E, Whisper, та Azure OpenAI
- **Функція виклику методів.** Betalgo включає функціонал для генерації та виклику функцій через OpenAI API, що дозволяє будувати більш динамічні та адаптивні програми.
- **Гнучкість налаштувань.** Бібліотека дозволяє змінювати параметри (наприклад, API-ключі) як через код, так і через зовнішні файли конфігурації, що підвищує зручність налаштування.

3.4 Розуміння роботи з ChatGPT

Оскільки фундаментом для даного застосунку є ChatGPT, точність роботи якого залежить від наданого користувачем запиту [57], потрібно було провести детальне дослідження, щоб зрозуміти, як формулювати промпти, що забезпечують найбільш релевантні та структуровані питання з багатьма варіантами відповідей. Для досягнення цієї мети були розглянуті та застосовані на практиці основи теорії побудови запитів в ChatGPT

Важливо було зрозуміти, як різні формулювання запитів впливають на

якість згенерованих відповідей [58, 59]. Наприклад, чи використання конкретних ключових слів або фраз покращує точність формування питань. З досліджуваних джерел вийшло дізнатися, що обов'язково боту потрібно надати опис проекту [32], тобто одне або два речення, які описують основну ідею, його мету, цільову аудиторію або кінцевих користувачів для кінцевого продукту та окремі результати, які ChatGPT має створити для завершення проекту. Наприклад:

- Розроби стратегію контент-маркетингу для технологічного стартапу з метою підвищення впізнаваності бренду та залучення ринкової ніші ентузіастів технологій.
- Напиши допис у блозі про важливість продуктивності для малого бізнесу.
- Напиши електронного листа клієнту, у якого є новий 3-місячний коргі, про дії, які вони повинні виконати, щоб привчати свого цуценя до будинку.

Також ШІ треба надати роль [33], мається на увазі ідентичність, точку зору чи професію, щоб допомогти керувати відповідями інструменту. ChatGPT може генерувати результати на основі сфери знань, пов'язаної з роллю, яку ви йому призначаєте. Наприклад:

- Ти експерт в англійські граматиці та письмі
- Уяви, що ти експерт в сфері кулінарії
- Ти журналіст з багаторічним досвідом

Окрім вищеперерахованої інформації, треба ще надати інформацію про формат відповіді ChatGPT [34], тобто запиті має бути вказано деталі результату і спосіб його створення, включаючи тон, довжину, стиль, структуру, а також дослідження, яке потрібно провести. Також значно збільшить точність надання прикладу відповіді. Приміром:

- Побудуй свою відповідь у форматі факт – числовий показник. Використовуй якомога більше професійних термінів. Довжина відповіді має складати не більше 300 слів

Останнє, що може допомогти у отримання якісного результату від бота –

це правила та обмеження, які будуть накладатися на відповідь [35]. Наприклад:

- Згенерований текст не має містити ненормативної лексики, занадто складних слів, які погіршать розуміння читача, сленгу
- Переконайся, що запитання не повторюються, а також перевір, щоб усі запитання відповідали тексту.

Знаючи теорію, негайно застосували її на практиці, проводячи експерименти з варіаціями запитів. Протестували різні варіанти запитів, щоб визначити, які з них дають найбільш відповідні результати. Це включало тестування довжини промптів, рівня деталізації та використання прикладів у самому запиті. Для оцінки згенерованих питань були використані такі метрики як **покриття тем** та **точність ключових фраз**. Перша обчислюється за наступною формулою

$$TC = \frac{|\text{Тем, присутніх у питаннях}|}{|\text{Тем, виділених у тексті}|}$$

Друга обчислюється за формулою

$$KP = \frac{|\text{Ключових фраз у питаннях}|}{|\text{Ключових фраз у тексті}|}$$

Отже, для експериментів був обраний наступний текст:

База даних– це організована колекція структурованої інформації, або даних, які зазвичай зберігаються в електронному форматі в комп'ютерній системі. Зазвичай базою даних керують за допомогою системи керування базами даних (СКБД). Дані, СКБД і програми, пов'язані з ними, разом називають системою баз даних, скорочено часто просто базою даних (БД).

У найпоширеніших сучасних типах БД дані зазвичай представлено як рядки й стовпці в низці таблиць, щоб підвищити ефективність обробки та запитів щодо даних. Після цього до даних можна легко отримати доступ, керувати ними, а також змінювати, оновлювати, контролювати та впорядковувати їх. Більшість баз даних використовують мову структурованих запитів (SQL) для запису й запитів щодо даних.

SQL– це мова програмування, яка використовується майже всіма

реляційними базами даних для виконання запитів і операцій, а також для визначення даних і контролю доступу. Уперше мова SQL була розроблена в IBM у 1970-х роках за значної участі Oracle, що призвело до впровадження стандарту SQL ANSI. SQL сприяла появі багатьох розширень від таких компаній, як IBM, Oracle і Microsoft. Хоча SQL досі широко використовується, починають з'являтися нові мови програмування.

Існує багато різних типів баз даних. Вибір найкращої бази даних для конкретної організації залежить від запланованого способу використання даних.

Реляційні бази даних почали домінувати в 1980-ті роки. Елементи в реляційній базі даних організовані як набір таблиць, що складаються зі стовпців і рядків. Технологія реляційних баз даних надає найбільш ефективний і гнучкий спосіб доступу до структурованої інформації.

Об'єктно-орієнтовані бази даних

Інформація в об'єктно-орієнтованій базі даних представлена як об'єкти, як і в об'єктно-орієнтованому програмуванні.

Розподілена база даних складається принаймні з двох файлів, розташованих на різних вузлах. База даних може зберігатися на кількох комп'ютерах, розташованих в одному фізичному місці або розкиданих по різних мережах.

Сховища даних

Центральний репозиторій даних, сховище даних– це тип баз даних, спеціально розроблений для швидкого виконання запитів і аналізу.

База даних NoSQL, або нереляційна, дає змогу зберігати неструктуровані й напівструктуровані дані та маніпулювати ними (на відміну від реляційної бази даних, яка визначає необхідний способів компонування всіх даних, що вносяться до неї). Популярність баз даних NoSQL зростала в міру того, як вебпрограми ставали дедалі поширенішими та складнішими.

Графова база даних зберігає дані як об'єкти й зв'язки між об'єктами.

База даних OLTP– це швидка аналітична база даних, призначена для

великої кількості транзакцій, що виконуються багатьма користувачами.

Це лише деякі з кількох десятків типів баз даних, що використовуються сьогодні. Інші бази даних, менш поширені, спеціально розроблено для виконання дуже специфічних наукових, фінансових або інших функцій. Є не лише різні типи баз даних. Зміни в підходах до розробки технологій і значні досягнення, як-от хмарні технології та автоматизація, сприяють розвитку баз даних в абсолютно нових напрямках. Нижче наведено деякі з найновіших типів баз даних.

Зазвичай для бази даних потрібне комплексне програмне забезпечення, відоме як система керування базами даних (СКБД). СКБД виконує роль інтерфейсу між базою даних та її кінцевими користувачами або програмами, даючи змогу користувачам отримувати й оновлювати інформацію, а також керувати способами її організації та оптимізації. СКБД також полегшує нагляд і контроль за базами даних, надаючи можливість виконувати різноманітні адміністративні операції, як-от моніторинг ефективності, налаштування, резервне копіювання та відновлення.

Деякі приклади популярного програмного забезпечення для баз даних (СКБД): MySQL, Microsoft Access, Microsoft SQL Server, FileMaker Pro, Oracle Database і dBASE.

Далі було сформовано декілька запитів:

1. Generate a few random multiple-choice questions based on this text.
2. Create several test questions with answer options based on this text that reflect the main ideas and details of the text. Ensure that the questions are clearly structured and the answer options are well-differentiated.
3. You are an expert in creating multiple-choice test questions. Based on this text, create several test questions with answer options that reflect the main ideas and details of the text. Make sure the questions are clearly structured, the answer options are well-differentiated, the questions are not repetitive, and also verify that all questions correspond to the text.
4. Text: {text}

You are an expert MCQ maker. Given the above text, it is your job to create a quiz of 8 multiple choice questions for Database students in professional tone. Make sure the questions are not repeated and check all the questions to be conforming to the text as well. Make sure to format your response like RESPONSE_JSON below and use it as a guide. Ensure to make 8 MCQs and have structure

```
{
    "1": {
        "mcq": "multiple choice question",
        "options": {
            "a": "choice here",
            "b": "choice here",
            "c": "choice here",
            "d": "choice here",
        },
        "correct": "correct answer",
    },
    "2": {
        "mcq": "multiple choice question",
        "options": {
            "a": "choice here",
            "b": "choice here",
            "c": "choice here",
            "d": "choice here",
        },
        "correct": "correct answer",
    },
    "3": {
        "mcq": "multiple choice question",
        "options": {
            "a": "choice here",
            "b": "choice here",
            "c": "choice here",
            "d": "choice here",
        },
        "correct": "correct answer",
    },
}
```

Розглянемо результати цих запитів. Перший запит надав такі питання:

Питання 1:

Що з наведеного нижче є основною функцією бази даних?

- *a) Обробка текстів*
- *b) Організація структурованої інформації*
- *c) Виконання обчислень*
- *d) Зберігання зображень*

Питання 2:

Який тип бази даних використовує таблиці, що складаються зі стовпців і рядків?

- *a) Графова база даних*
- *b) Реляційна база даних*
- *c) Об'єктно-орієнтована база даних*
- *d) NoSQL база даних*

Питання 3:

SQL була розроблена в 1970-х роках якою компанією?

- *a) Microsoft*
- *b) Oracle*
- *c) IBM*
- *d) Google*

Питання 4:

Яка база даних розроблена для швидкого виконання запитів і аналізу даних?

- *a) OLTP база даних*
- *b) Сховище даних*
- *c) NoSQL база даних*
- *d) Реляційна база даних*

Розрахунок метрик.

Метрика покриття тем (Topic Coverage, TC):

- Виділені теми тексту: бази даних, СКБД, SQL, реляційні бази даних,

NoSQL, OLTP, сховище даних, об’єктно-орієнтовані бази даних, графові бази даних.

- Охоплені теми у питаннях: бази даних, реляційні бази даних, SQL, сховище даних.

$$TC = \frac{4}{9} \approx 0.44$$

Метрика точності ключових фраз (Key Phrase Precision, KP):

- Ключові фрази тексту: база даних, СКБД, SQL, реляційні бази даних, NoSQL, OLTP, сховище даних, об’єктно-орієнтовані бази даних, графові бази даних.
- Ключові фрази у питаннях: база даних, реляційні бази даних, SQL, сховище даних.

$$KP = \frac{4}{9} \approx 0.44$$

Другий запит надав такі питання:

Питання 1:

Що таке база даних?

- *a) Програма для керування комп'ютером*
- *b) Організована колекція структурованої інформації*
- *c) Система зберігання зображень*
- *d) Мова програмування*

Питання 2:

Яка мова програмування використовується майже всіма реляційними базами даних?

- *a) Python*
- *b) C++*
- *c) SQL*
- *d) JavaScript*

Питання 3:

Який тип бази даних організований як набір таблиць із рядками та стовпцями?

- *a) Графова база даних*
- *b) Реляційна база даних*
- *c) Об’єктно-орієнтована база даних*
- *d) NoSQL база даних*

Питання 4:

Що з наведеного є прикладом системи керування базами даних (СКБД)?

- *a) Microsoft Word*
- *b) Adobe Photoshop*
- *c) MySQL*
- *d) Google Chrome*

Питання 5:

Що таке NoSQL база даних?

- *a) Реляційна база даних, яка використовує SQL*
- *b) База даних для зберігання виключно текстової інформації*
- *c) Нереляційна база даних для зберігання неструктурованих та напівструктурованих даних*
- *d) Система керування файлами*

Питання 6:

Який тип бази даних спеціально розроблений для швидкого виконання запитів і аналізу даних?

- *a) OLTP база даних*
- *b) Сховище даних*
- *c) Реляційна база даних*
- *d) Об’єктно-орієнтована база даних*

Розрахунок метрик

Метрика покриття тем (Topic Coverage, TC):

- Виділені теми тексту: база даних, СКБД, SQL, реляційні бази даних, NoSQL, сховище даних, графові бази даних, об’єктно-орієнтовані бази даних, OLTP.

- Охоплені теми у питаннях: база даних, СКБД, SQL, реляційні бази даних, NoSQL, сховище даних.

$$TC = \frac{6}{9} \approx 0.67$$

Метрика точності ключових фраз (Key Phrase Precision, KP):

- Ключові фрази тексту: база даних, СКБД, SQL, реляційні бази даних, NoSQL, OLTP, сховище даних, об'єктно-орієнтовані бази даних, графові бази даних.
- Ключові фрази у питаннях: база даних, СКБД, SQL, реляційні бази даних, NoSQL, сховище даних

$$KP = \frac{6}{9} \approx 0.67$$

Третій запит згенерував такі питання:

Питання 1:

Що є основним призначенням системи керування базами даних (СКБД)?

- *a) Виконання математичних розрахунків*
- *b) Зберігання зображень*
- *c) Управління базами даних та забезпечення інтерфейсу між базою даних і користувачем*
- *d) Створення вебсайтів*

Питання 2:

Яка з наведених мов програмування використовується для виконання запитів і операцій у реляційних базах даних?

- *a) Python*
- *b) SQL*
- *c) Java*
- *d) C#*

Питання 3:

Який тип бази даних призначений для зберігання неструктурованих і напівструктурованих даних?

- *a) Реляційна база даних*
- *b) Об’єктно-орієнтована база даних*
- *c) NoSQL база даних*
- *d) OLTP база даних*

Питання 4:

У якому році була розроблена мова SQL?

- *a) 1960-ті роки*
- *b) 1970-ті роки*
- *c) 1980-ті роки*
- *d) 1990-ті роки*

Питання 5:

Що таке сховище даних?

- *a) Тип бази даних для зберігання неструктурованих даних*
- *b) Центральний репозиторій даних для швидкого виконання запитів і аналізу*
- *c) База даних для зберігання відеофайлів*
- *d) Система для керування об’єктами в програмуванні*

Питання 6:

Який тип бази даних організований у вигляді таблиць зі стовпцями та рядками?

- *a) Об’єктно-орієнтована база даних*
- *b) Графова база даних*
- *c) Реляційна база даних*
- *d) NoSQL база даних*

Питання 7:

Який із наступних типів баз даних зберігає дані у вигляді об'єктів?

- *a) Реляційна база даних*
- *b) Графова база даних*
- *c) Об’єктно-орієнтована база даних*
- *d) OLTP база даних*

Питання 8:

Який тип бази даних розроблено для великої кількості транзакцій, що виконуються багатьма користувачами?

- *a) NoSQL база даних*
- *b) OLTP база даних*
- *c) Сховище даних*
- *d) Графова база даних*

Розрахунок метрик

Метрика покриття тем (Topic Coverage, TC):

- Виділені теми тексту: база даних, СКБД, SQL, реляційні бази даних, NoSQL, сховище даних, графові бази даних, об'єктно-орієнтовані бази даних, OLTP.
- Охоплені теми у питаннях: база даних, СКБД, SQL, реляційні бази даних, NoSQL, сховище даних, графові бази даних, об'єктно-орієнтовані бази даних, OLTP.

$$TC = \frac{9}{9} = 1.00$$

Метрика точності ключових фраз (Key Phrase Precision, KP):

- Ключові фрази тексту: база даних, СКБД, SQL, реляційні бази даних, NoSQL, OLTP, сховище даних, об'єктно-орієнтовані бази даних, графові бази даних.
- Ключові фрази у питаннях: база даних, СКБД, SQL, реляційні бази даних, NoSQL, OLTP, сховище даних, об'єктно-орієнтовані бази даних, графові бази даних.

$$KP = \frac{9}{9} = 1.00$$

На кінець, четвертий запит сформував такі питання:


```
{
  "1": {
    "mcq": "Що таке база даних?",
    "options": {
```

```
        "a": "Це набір програм для керування файлами",
        "b": "Це організована колекція структурованої інформації, яка зберігається в електронному форматі",
        "c": "Це інтерфейс між користувачем і комп'ютером",
        "d": "Це метод зберігання великих зображень та відео"
      },
      "correct": "b"
    },
    "2": {
      "mcq": "Що є основною функцією системи керування базами даних (СКБД)?",
      "options": {
        "a": "Створення вебсайтів",
        "b": "Збереження зображень",
        "c": "Керування базами даних і забезпечення доступу до них користувачам",
        "d": "Створення програмного забезпечення"
      },
      "correct": "c"
    },
    "3": {
      "mcq": "Для чого використовується мова SQL у реляційних базах даних?",
      "options": {
        "a": "Для обробки графічних даних",
        "b": "Для створення вебсторінок",
        "c": "Для виконання запитів, операцій і визначення даних",
        "d": "Для аналізу медичних даних"
      },
      "correct": "c"
```

```
  },
  "4": {
    "mcq": "Який із типів баз даних зберігає дані у вигляді таблиць з рядками і стовпцями?",
    "options": {
      "a": "Об’єктно-орієнтована база даних",
      "b": "NoSQL база даних",
      "c": "Реляційна база даних",
      "d": "Графова база даних"
    },
    "correct": "c"
  },
  "5": {
    "mcq": "Що є ключовою особливістю NoSQL бази даних?",
    "options": {
      "a": "Зберігає дані у вигляді таблиць",
      "b": "Дозволяє зберігати неструктуровані та напівструктуровані дані",
      "c": "Використовується для управління файлами в системі",
      "d": "Зберігає графічні дані для вебсайтів"
    },
    "correct": "b"
  },
  "6": {
    "mcq": "Що таке сховище даних?",
    "options": {
      "a": "Це база даних для виконання великої кількості транзакцій",
      "b": "Це тип бази даних для зберігання великих мультимедійних файлів",
      "c": "Це центральний репозиторій для швидкого виконання запитів
```

і аналізу даних",
"d": "Це сервер для управління програмним забезпеченням"
},
"correct": "c"
},
"7": {
"mcq": "Що таке графова база даних?",
"options": {
"a": "Це база даних, яка зберігає дані як об’єкти і зв’язки між ними",
"b": "Це база даних для зберігання рядків і стовпців",
"c": "Це сховище даних для швидкого виконання аналізу",
"d": "Це база даних, яка зберігає тільки числові значення"
},
"correct": "a"
},
"8": {
"mcq": "Який тип баз даних призначений для швидкої обробки великої кількості транзакцій?",
"options": {
"a": "Реляційна база даних",
"b": "Графова база даних",
"c": "OLTP база даних",
"d": "NoSQL база даних"
},
"correct": "c"
}
}

Розрахунок метрик

Метрика покриття тем (Topic Coverage, TC):

- Виділені теми тексту: база даних, СКБД, SQL, реляційні бази даних, NoSQL, сховище даних, графові бази даних, об’єктно-орієнтовані бази даних, OLTP.
- Охоплені теми у питаннях: база даних, СКБД, SQL, реляційні бази даних, NoSQL, сховище даних, графові бази даних, об’єктно-орієнтовані бази даних, OLTP.

$$TC = \frac{9}{9} = 1.00$$

Метрика точності ключових фраз (Key Phrase Precision, KP):

- Ключові фрази тексту: база даних, СКБД, SQL, реляційні бази даних, NoSQL, OLTP, сховище даних, об’єктно-орієнтовані бази даних, графові бази даних.
- Ключові фрази у питаннях: база даних, СКБД, SQL, реляційні бази даних, NoSQL, OLTP, сховище даних, об’єктно-орієнтовані бази даних, графові бази даних.

$$KP = \frac{9}{9} = 1.00$$

Також варто додати, що ChatGPT зміг дотриматись структури питань, яка була описана в запиті. Така структура полегшить обробку цих питань коді програми.

Для наочності сформуємо таблицю 3.1 з метриками для кожного запиту.

Як видно, обидва запити – третій і четвертий – забезпечують максимальні метрики, проте структура четвертого є зручнішою для подальшої обробки. Тому для подальшої роботи був обраний саме четвертий тип запиту, оскільки він включає чітке завдання для бота, визначення його ролі, вимоги до відповідей, застосовні обмеження, а також формат, у якому має бути подана відповідь.

Таблиця 3.1 – Показники метрики покриття тем та метрики точності ключових фраз кожного запиту

Номер запиту	Метрика покриття тем	Метрика точності

		ключових фраз
1	0.44	0.44
2	0.75	0.67
3	1	1
4	1	1

3.5 Реалізація основних компонентів

В даному розділі буде описана реалізація найнеобхідніших частин системи[61], починаючи від шару сутностей і закінчуючи шаром адаптерів інтерфейсів. Шар інфраструктури не буде описаний, так як до нього відносяться база даних та зовнішні сервіси, робота з якими була описана вище.

3.5.1 Реалізація шару сутностей

Як було описано вище, в шарі сутностей мають знаходитись моделі бази даних, тому далі піде опис їхнього коду. В запропонованому коді реалізують основні принципи проектування систем з використанням ORM (Object-Relational Mapping) в рамках **Entity Framework Core**, зокрема підхід **Code First**.

Розглянемо приклад коду, який описує сутність нашої системи, а саме тесту, що є центральним елементом у системі тестування клас "Test" включає визначення його основних властивостей, зовнішніх ключів і дати створення. Основна властивість Id представляє унікальний ідентифікатор тесту та позначена як первинний ключ за допомогою атрибута [Key]. Поле Questions використовується для зберігання питань у форматі JSON, що забезпечує гнучкість у зберіганні тестових завдань із різною структурою.

Зовнішні ключі, які позначені атрибутом [ForeignKey] забезпечують зв’язки з іншими сутностями бази даних. TeacherId пов’язує тест із викладачем, а навігаційна властивість Teacher дозволяє отримати повну інформацію про цього викладача. Аналогічно, StudentId визначає студента, якому призначено тест, із доступом до його даних через властивість Student. Поле TopicId пов’язує тест із відповідною темою, а властивість Topic надає доступ до пов’язаної інформації про цю тему.

Поле CreatedAt використовується для збереження дати та часу створення тесту, що дозволяє відстежувати хронологію створення та використання тестів у системі.

По аналогії з сутністю тесту буде написаний код для всіх інших сутностей в описаній раніше базі даних

3.5.2 Реалізація шару застосування

В даному підрозділі буде описана програма, яка реалізує основну логіку застосунку, яка відповідає за взаємодію всіх моделей, описаних в попередньому підрозділі. Для більш наочного пояснення був взятий фрагмент, який генерує питання та валідує або перевіряє їх.

Спочатку буде розписаний клас **QuizService**, який реалізує логіку для інтеграції з OpenAI API, забезпечуючи генерацію тестових завдань на основі вхідних даних. Він організований у вигляді сервісу, що взаємодіє з бібліотекою **Betalgo.Ranul.OpenAI**, і надає функціональність через метод GenerateTestAsync. Цей метод автоматизує процес створення тестів із множинним вибором, враховуючи задані параметри.

У процесі роботи QuizService отримує від користувача необхідні дані: текст навчального матеріалу, кількість запитань, тему тесту, тон формулювань і шаблон формату відповіді. Ці параметри об'єднуються в текст шаблону, який використовується для створення промпту. Шаблон взятий той, щ обув експериментально отриманий в попередніх розділах.

Сервіс звертається до OpenAI API через клієнт ChatCompletionCreateRequest, передаючи модель, промпт і параметри, такі як максимальна кількість токенів. Після отримання відповіді API результат аналізується, і з нього витягується згенерований вміст, який передається як результат роботи методу.

Особливістю реалізації є використання механізмів OpenAI для забезпечення адаптивності та точності тестових завдань. Цей підхід дозволяє автоматизувати створення тестів і значно знижує час, необхідний для підготовки навчальних матеріалів. Використання C# і бібліотеки Betalgo дозволяє

інтегрувати систему в більші проєкти, зокрема ті, що розроблені на основі екосистеми .NET, забезпечуючи гнучкість і масштабованість.

Далі, цей сервіс інтегрується в інший, більш високорівневий. Сервіс TestGenerationService приймає кілька параметрів від користувача, серед яких ідентифікатори викладача, теми та студента, а також текст, кількість питань і тональність тесту. Ці дані визначають, до кого буде прив'язано тест, його зміст і характер питань. Після отримання цих параметрів сервіс викликає метод GenerateTestAsync із класу QuizService, який на основі введених даних генерує тест у форматі JSON. Потім створюється об'єкт тесту, в якому заповнюються поля згенерованого тесту, включаючи питання, а також ідентифікатори викладача, студента і теми. Після цього тест зберігається в базі даних за допомогою репозиторію TestRepository. Наприкінці метод повертає ідентифікатор збереженого тесту, який може бути використаний для подальших операцій, таких як відображення тесту чи його передача користувачу для подальшої роботи.

Наступним буде описаний сервіс, який перевіряє питання. Сервіс TestValidationService починається з пошуку відповідного бланка відповідей, який пов’язаний із заданим ідентифікатором студента. Для цього використовується репозиторій AnswerSheet, який забезпечує доступ до відповідної таблиці бази даних. Після того як бланк відповідей знайдено, з нього витягується тест, до якого він прив’язаний, за допомогою поля TestId. Далі, поля Questions і Answers, що зберігаються у форматі JSON, розбираються в словники Dictionary<int, string> за допомогою бібліотеки JsonConvert з пакету Newtonsoft.Json.

На етапі перевірки кожна відповідь студента порівнюється з правильною відповіддю, отриманою з тесту. Якщо відповіді співпадають, то лічильник правильних відповідей збільшується на одиницю. Як результат роботи, метод повертає кількість правильних відповідей, який зберігається в базу даних для подальшого аналізу.

3.5.3 Реалізація шара адаптерів інтерфейсів

В даному підрозділі буде описана реалізація контролерів, які будуть передавати HTTP-запит від клієнта до системи, так як досліджувана система є WebAPI. Для прикладу був взятий TestsController

Контролер **TestsController** є частиною ASP.NET Core API, що реалізує бізнес-логіку для створення, перевірки, видалення та отримання тестів у системі. Він взаємодіє з сервісами, репозиторіями та надає необхідні ендпоїнти для роботи з тестами, які зберігаються у базі даних. В ньому є декілька необхідних методів.

Метод **GenerateTestAsync (POST /api/tests/generate)** приймає на вхід параметри через POST-запит для генерації тесту. У ньому використовуються такі параметри, як ідентифікатор викладача, студента, тема, текст тесту, кількість питань і тональність тесту. Сервіс **TestGenerationService** генерує тест на основі цих даних і зберігає його в базі даних через репозиторій **_testRepository**. Після успішного створення тесту повертається його ідентифікатор. Статус відповіді при цьому встановлюється як 201 (Created), що вказує на успішне створення ресурсу.

Метод **ValidateTest (POST api/tests/validate/{testId}/{studentId})** призначений для перевірки тесту студентом. Він приймає два параметри: testId та studentId. Використовує сервіс **TestValidationService** для перевірки виконаних відповідей студента. Відповіді студента порівнюються з правильними відповідями, і метод повертає кількість правильних відповідей у вигляді результату. У разі помилки повертається статус 400 (Bad Request).

Метод DeleteTest (DELETE /api/tests/{id}) дозволяє видаляти тест за ідентифікатором. Спочатку виконується перевірка, чи існує тест з таким ID в базі даних через **_testRepository**. Якщо тест знайдено, його видаляють, а результатом буде статус 204 (No Content), що означає успішне виконання запиту без повернення додаткових даних.

Метод GetTestById (GET api/tests/{id}) надає можливість отримати тест за його ідентифікатором. Якщо тест з таким ID є в базі даних, повертається сам тест, а в разі відсутності тесту повертається статус 404 (Not Found) з відповідним

повідомленням.

Метод GetTestsByTeacher (GET api/tests/teacher/{teacherId}) дозволяє отримати всі тести, що належать конкретному викладачеві, приймаючи на вхід ідентифікатор викладача. Використовуючи репозиторій **_testRepository**, перевіряється, чи є тести, пов'язані з цим викладачем. Якщо тестів немає, повертається повідомлення про помилку.

Метод GetTestsByStudent (GET api/tests/student/{studentId}) працює аналогічно попередньому, але дозволяє отримати всі тести для конкретного студента. Приймає на вхід studentId і повертає список тестів, зв'язаних із цим студентом. Якщо тести не знайдені, повертається відповідне повідомлення.

3.6 Приклади роботи програми

Для демонстрації роботи системи буде розглянутий контролер TestsController. Візуально він буде виглядати такий чином:

Рисунок 3.1 – візуалізація TestsController за допомогою Swagger

Розглянемо ендпоїнт для генерації питань. Тіло запиту цього ендпоїнту буде мати наступний вигляд

```
{
  "teacherId": 1,
  "topicId": 1,
  "studentId": 1,
  "text": "База даних – це організована колекція структурованої інформації або
даних, які зберігаються в електронному форматі в комп'ютерній системі. Базами даних
керують за допомогою системи керування базами даних (СКБД), а дані, СКБД і програми,
пов'язані з ними, разом складають систему баз даних (БД). У більшості сучасних типів
БД дані зазвичай представлені рядками та стовпцями в таблицях для ефективності
обробки та запитів. Це дозволяє легкий доступ до даних, їх управління, зміну,
оновлення, контроль та впорядкування. Більшість баз даних використовують мову SQL
для запису та запитів щодо даних. SQL, як мова програмування, використовується
практично всіма реляційними базами даних для виконання запитів і операцій, а також
для визначення даних і контролю доступу. Мова SQL була розроблена в IBM у 1970-х
роках за участі Oracle, що призвело до стандартизації SQL ANSI, а також сприяла
розвитку різноманітних розширень від таких компаній, як IBM, Oracle і Microsoft.
Незважаючи на широке використання SQL, на ринку з'являються нові мови програмування.
Існує кілька типів баз даних, вибір яких залежить від конкретних потреб. Реляційні
бази даних домінують із 1980-х років, де елементи організовані в таблиці з рядками і
стовпцями, що забезпечує ефективний доступ до структурованої інформації. Об'єктно-
орієнтовані бази даних використовують концепцію об'єктів, як у об'єктно-
орієнтованому програмуванні. Розподілені бази даних складаються з кількох файлів,
розташованих на різних вузлах і можуть зберігатися на кількох комп'ютерах в одному
місці або в різних мережах. Сховища даних – це тип баз даних, спеціально розроблений
для швидкого виконання запитів і аналізу. База даних NoSQL дозволяє зберігати
неструктуровані та напівструктуровані дані, на відміну від реляційних баз, що
вимагають жорсткої структури даних. Популярність NoSQL зростала разом із розвитком
складних веб-додатків. Графові бази даних зберігають інформацію про об'єкти та їхні
зв'язки, а бази даних OLTP призначені для швидкої обробки транзакцій, виконуваних
багатьма користувачами. Існують й інші менш поширені типи баз даних, які
застосовуються для специфічних наукових, фінансових або інших функцій. Зміни в
підходах до технологій і новітні досягнення, такі як хмарні технології та
автоматизація, сприяють розвитку баз даних у нових напрямках. Для керування базами
даних потрібне спеціалізоване програмне забезпечення, таке як СКБД, яке полегшує
взаємодію між користувачем і базою даних, дозволяючи отримувати та оновлювати
інформацію, організовувати та оптимізувати дані. СКБД також здійснює нагляд за
базами, виконуючи адміністративні операції, такі як моніторинг, налаштування,
резервне копіювання та відновлення. Популярні приклади програм для керування базами
даних включають MySQL, Microsoft Access, SQL Server, FileMaker Pro, Oracle Database
і dBASE.",
  "numberOfQuestions": 10,
  "tone": "професійний"
}
```


Рисунок 3.2 – Тіло запиту ендпоїнту для генерації питань

А відповідь буде виглядати так:

Рисунок 3.3 – Відповідь ендпоїнту для генерації питань

Далі розглянемо ендпоїнт для отримання тесту по ідентифікатору. Тіло запиту цього ендпоїнту буде мати наступний вигляд

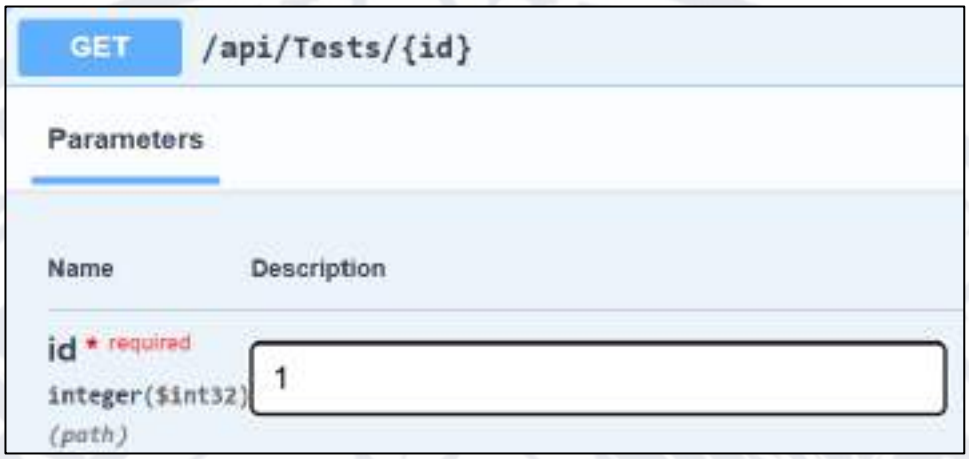

Рисунок 3.4 – Тіло запиту ендпоїнту для отримання питання по ідентифікатору

А відповідь буде виглядати так, як показано на рис. 3.5

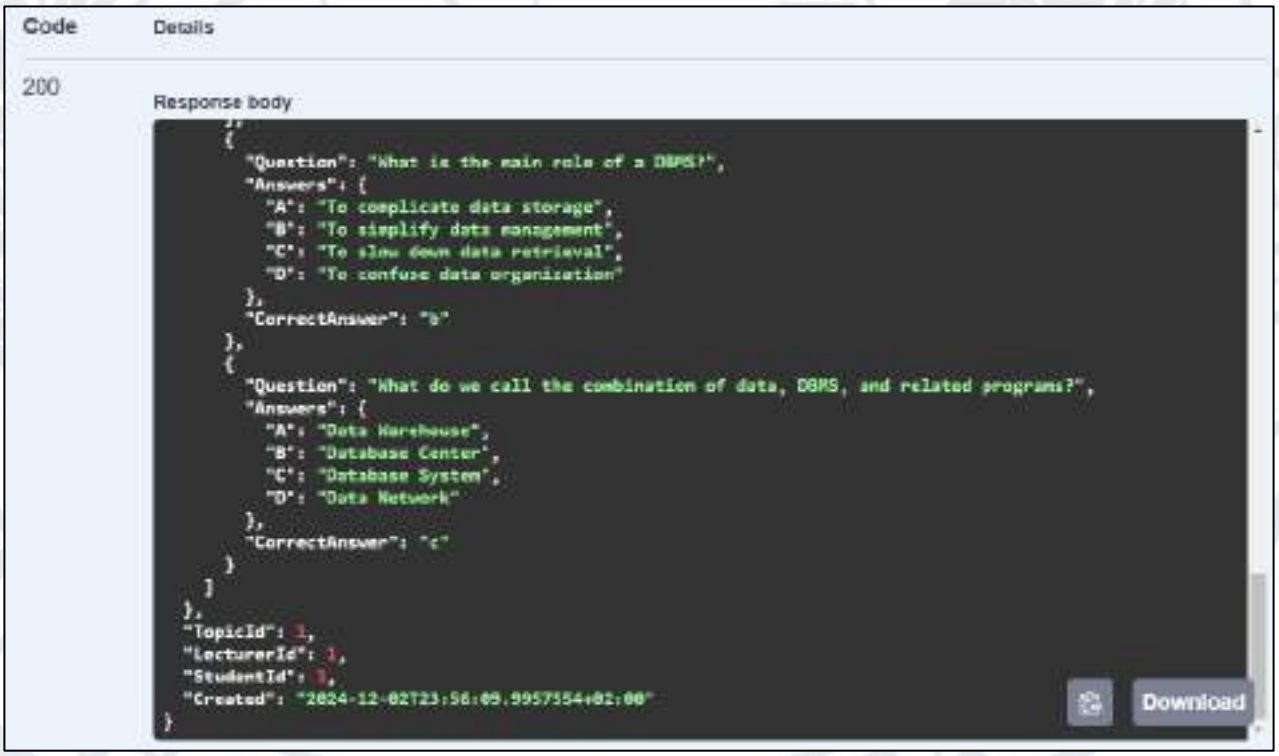

Рисунок 3.5 – Відповідь ендпоїнту для отримання питання по ідентифікатору

І наостанок ще один основний ендпоїнт, а саме той, що повертає оцінку тесту для студенту Тіло запиту цього ендпоїнту буде мати вигляд, який показаний на рис. 3.6

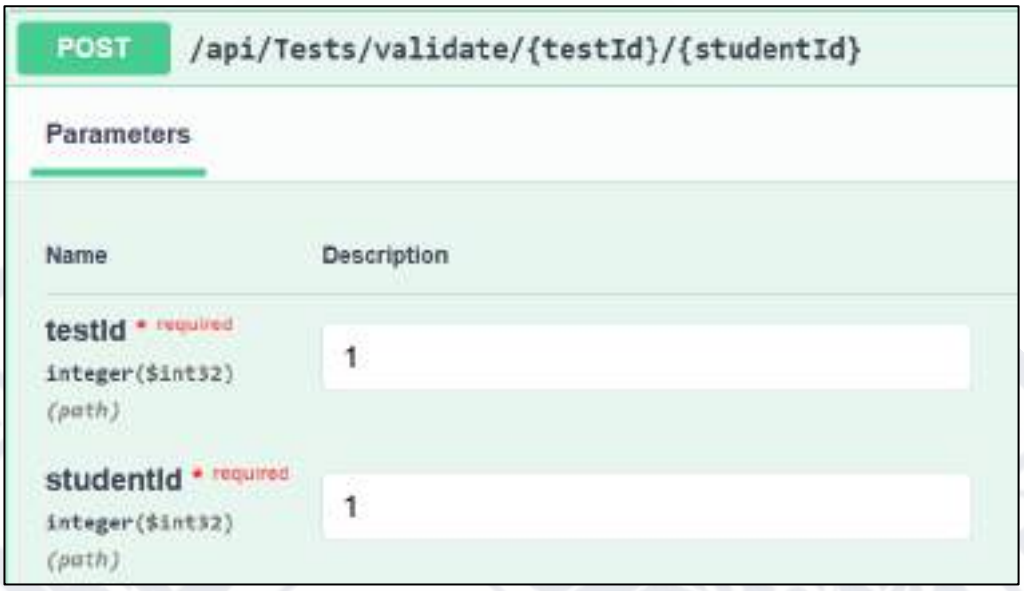

Рисунок 3.6 – Тіло запиту ендпоїнту для отримання оцінки тесту студента

А відповідь буде виглядати так:

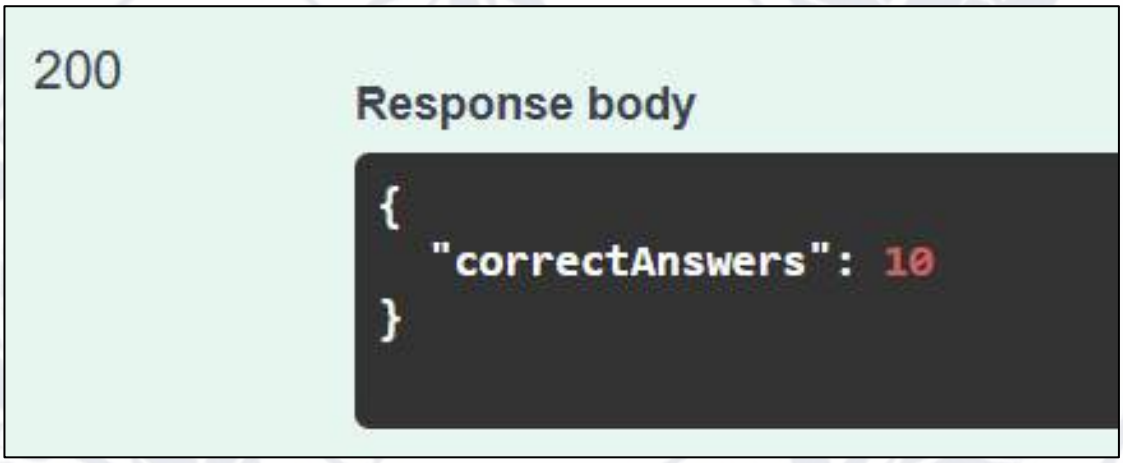

Рисунок 3.7 – Відповідь ендпоїнту для отримання оцінки тесту студента

У результаті була розглянута робота основних ендпоїнтів, без логіки який, система не мала б сенсу.

ВИСНОВКИ

У межах даної роботи було досліджено актуальність автоматизації освітніх процесів, зокрема систем автоматичного створення та валідації тестових питань на основі штучного інтелекту, та оглянуто наявні продукти схожої тематики.

Також було проаналізовано обрані технології, зокрема інтеграцію ChatGPT для генерації питань з множинним вибором, виділені їхні переваги та можливості в контексті сучасних освітніх платформ.

Розроблено програмне забезпечення з використанням принципів модульності та перевірено його на предмет коректності та точності. У результаті була успішно впроваджена багатоступенева архітектура з послідовною перевіркою питань. Отримані навички роботи з інструментами штучного інтелекту та можливості їхньої інтеграції в освітні середовища забезпечують основу для подальшого розвитку та вдосконалення системи.

СПИСОК ВИКОРИСТАНИХ ПОСИЛАНЬ

1. Quillionz. URL: https://www.quillionz.com/ (дата звертання: 13.04.2024)
2. PrepAI. URL: https://www.prepai.io/ (дата звертання: 13.04.2024)
3. EdApp. URL: https://www.edapp.com/ (дата звертання: 13.04.2024)
4. What is artificial intelligence (AI)? URL: https://www.ibm.com/topics/artificial-intelligence (дата звертання: 22.04.2024)
5. What Is Artificial Intelligence? Definition, Uses, and Types. URL: https://www.coursera.org/articles/what-is-artificial-intelligence (дата звертання: 22.04.2024)
6. Що таке нейронні мережі та де їх використовують? URL: https://incrypted.com/ua/shcho-take-nejromerezhi/ (дата звертання: 22.04.2024)
7. What is a neaural network? URL: https://www.cloudflare.com/learning/ai/what-is-neural-network/ (дата звертання: 22.04.2024)
8. Building a Basic Neural Network from Scratch: A Step-by-Step Guide. URL: https://bhatnagar91.medium.com/building-a-basic-neural-network-from-scratch-a-step-by-step-guide-7a6f97979ddd (дата звертання: 22.04.2024)
9. What are Deep Feed-Forward Networks? Deep Learning Explained. URL: https://medium.com/@muratbayrktr/what-are-deep-feed-forward-networks-deep-learning-explained-c1708a8c54c5 (дата звертання: 10.05.2024)
10. What are convolutional neural networks? URL: https://www.ibm.com/topics/convolutional-neural-networks (дата звертання: 10.05.2024)
11. What are recurrent neural networks? URL: https://www.ibm.com/topics/recurrent-neural-networks (дата звертання: 10.05.2024)
12. A Short Introduction to Generative Adversarial Networks . URL: https://sthalles.github.io/intro-to-gans/ (дата звертання: 10.05.2024)
13. Text generation using LSTM. URL: https://www.ai-tech.systems/text-generation-using-lstm/ (дата звертання: 10.05.2024)

14. SegNET. URL: https://medium.com/@saba99/segnet-a139ce77b570 (дата звертання: 10.05.2024)
15. ChatGPT. URL: https://openai.com/chatgpt (дата звертання: 23.04.2024)
16. MidJourney. URL: https://www.midjourney.com/home (дата звертання: 23.04.2024)
17. Gemini. URL: https://gemini.google.com/app (дата звертання: 23.04.2024)
18. DALL-E 2. URL: https://openai.com/dall-e-2 (дата звертання: 23.04.2024)
19. Github Copilot. URL: https://github.com/features/copilot (дата звертання: 23.04.2024)
20. Gitlab. URL: https://about.gitlab.com/ (дата звертання: 23.04.2024)
21. UiPath. URL: https://www.uipath.com/ (дата звертання: 23.04.2024)
22. DeepL. URL: https://www.deepl.com/uk/translator (дата звертання: 23.04.2024)
23. Tableau. URL: https://www.tableau.com/ (дата звертання: 23.04.2024)
24. Qlik Sense. URL: https://www.qlik.com/us/products/qlik-sense (дата звертання: 23.04.2024)
25. SAS Viya. URL: https://www.sas.com/ru_ua/software/viya.html (дата звертання: 23.04.2024)
26. Amazon Alexa. URL: https://developer.amazon.com/en-US/alexa (дата звертання: 23.04.2024)
27. Zendesk Support. URL: https://support.zendesk.com/hc/en-us (дата звертання: 23.04.2024)
28. Google Classroom. URL: https://classroom.google.com/ (дата звертання: 23.04.2024)
29. Dulingo. URL: https://uk.duolingo.com/ (дата звертання: 23.04.2024)
30. DreamBox Learning. URL: https://www.dreambox.com/ (дата звертання: 23.04.2024)
31. Python. URL: https://www.python.org/ (дата звертання: 20.06.2024)

32. How To Write ChatGPT Prompts: Your 2024 Guide. URL: https://www.coursera.org/articles/how-to-write-chatgpt-prompts (дата звертання: 12.09.24)

33. How to Craft the Perfect ChatGPT Prompt. URL: https://www.linkedin.com/pulse/how-craft-perfect-chatgpt-prompt-charlene-li-kowge (дата звертання: 12.09.24)

34. 7 Tips to Take Your ChatGPT Prompts to the Next Level. URL: https://www.wired.com/story/17-tips-better-chatgpt-prompts/ (дата звертання: 12.09.24)

35. How To Write Effective Prompts For ChatGPT: 7 Essential Steps For Best Results. URL: https://www.forbes.com/sites/jodiecook/2023/06/26/how-to-write-effective-prompts-for-chatgpt-7-essential-steps-for-best-results (дата звертання: 12.09.24)

36. Antonov Y., Smoktii K. Expert systems using for answers analysis in automated knowledge control systems. URL: https://heraldts.khmnu.edu.ua/index.php/heraldts/article/view/368/367 (дата звертання: 31.10.24)

37. Антонов Ю. С., Космінська О. М. Методика аналізу тестових завдань на основі отриманих результатів тестування [Електронний ресурс] / Ю. С. Антонов // Інформаційні технології і засоби навчання.– 2009.– № 4.– URL: https://journal.iitta.gov.ua/index.php/itlt/article/view/81/67 (дата звертання: 31.10.24)

38. Антонов, Ю. С.. Оцінка повноти відповідей в автоматизованих системах контролю знань. / Ю.С. Антонов // Наукові праці Донецького національного технічного університету, серія Інформатика, кібернетика та обчислювальна техніка. – 2012. – №15, С. 113-117.

39. Антонов Ю. С. Комп’ютерні системи тестування на основі технології трирівневих баз даних [Електронний ресурс] / Ю. С. Антонов // Інформаційні технології і засоби навчання.– 2008.– № 2. URL: https://journal.iitta.gov.ua/index.php/itlt/article/view/133/119 (дата звертання: 31.10.24)

40. An overview of HTTP. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview (дата звертання: 07.11.24)

41. What is a RESTful API? URL: https://aws.amazon.com/what-is/restful-api/?nc1=h_ls (дата звертання: 07.11.24)

42. POST. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/POST (дата звертання: 07.11.24)

43. Космінська О. М. Контроль знань студентів за допомогою тестування [Електронний ресурс] / Космінська О. М. // Інформаційні технології і засоби навчання.– 2008.– № 6.– URL: https://journal.iitta.gov.ua/index.php/itlt/article/view/135/121 (дата звертання: 31.10.24)

44. Єфіменко В. С. Комп'ютерне тестування як складова система педагогічної діагностики компетентності школярів із теми «Табличний процесор» [Електронний ресурс] / В.С. Єфіменко // Інформаційні технології і засоби навчання.– 2008.– № 1.– URL: https://journal.iitta.gov.ua/index.php/itlt/article/view/152/138 (дата звертання: 31.10.24)

45. What is monolithic architecture? URL: https://www.ibm.com/think/topics/monolithic-architecture (дата звертання: 1.11.24)

46. Understanding Monolithic Architecture: A Comprehensive Overview. URL: https://dev.to/hyscaler/understanding-monolithic-architecture-a-comprehensive-overview-3d7d (дата звертання: 1.11.24)

47. Monolithic Architecture- Everything You Need to Know. URL: https://webandcrafts.com/blog/monolithic-architecture (дата звертання: 1.11.24)

48. Advantages of microservices and disadvantages to know. URL: https://www.atlassian.com/microservices/cloud-computing/advantages-of-microservices (дата звертання: 1.11.24)

49. What You Should Know About Microservice Architecture. URL: https://theappsolutions.com/blog/development/microservice-architecture-explained/ (дата звертання: 1.11.24)

50. Microservices 101. What They Are, Advantages, Disadvantages, and Comparison with Monolithic Architectures. URL: https://www.juannicolas.eu/microservices-101-what-they-are-advantages-disadvantages/ (дата звертання: 1.11.24)

51. Advantages and Disadvantages of Microservices Architecture. URL: https://codeinstitute.net/global/blog/advantages-and-disadvantages-of-microservices-architecture/ (дата звертання: 1.11.24)

52. C# documentation. URL: https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/ (дата звертання: 5.11.24)

53. MySQL. URL: https://www.mysql.com (дата звертання: 5.11.24)

54. PostgreSQL. URL: https://www.postgresql.org/ (дата звертання: 5.11.24)

55. SQLServer. URL: https://www.microsoft.com/en-us/sql-server (дата звертання: 5.11.24)

56. Betalgo.Ranul.OpenAI . URL: https://github.com/betalgo/openai (дата звертання: 5.11.24)

57. Aymen El Amri (2024). LLM Prompt Engineering For Developers : The Art and Science of Unlocking LLMs' True Potential. Leanpub.

58. Tom Auger, & Emma Saroyan (2024). Generative AI for Web Development: Building Web Applications Powered by OpenAI APIs and Next.js. Apress.

59. Ajantha Devi Vairamani, & Anand Nayyar (2024). Prompt Engineering: Empowering Communication. CRC Press.

60. Andrew Troelsen, & Phil Japikse (2022). Pro C# 10 with .NET 6: Foundational Principles and Practices in Programming. Apress

61. Robert C. Martin, & Kevlin Henney (2017). Clean Architecture: A Craftsman's Guide to Software Structure and Design. Pearson.

ДЕКЛАРАЦІЯ

про дотримання академічної доброчесності

Я, _____

_____

_____

_____

Повністю вказується ПІБ та статус (освітня (освітньо-наукова) програма – для здобувачів вищої освіти, назва кваліфікаційної роботи)

що нижче підписалась/підписався, розуміючи та підтримуючи загальновизнані засади справедливості, доброчесності та законності,

ЗОБОВ'ЯЗУЮСЬ:

дотримуватися принципів та правил академічної доброчесності, що визначені законодавством України, локальними нормативними актами Донецького національного університету імені Василя Стуса, положеннями, правилами, умовами, визначеними іншими суб'єктами, та не допускати їх порушення.

ПІДТВЕРДЖУЮ:

що мені відомі положення статті 42 Закону України «Про освіту»;

що у даній роботі не представляла/представляв чиїсь роботи повністю або частково як свої власні. Там, де я скористалася/скористався працею інших, я зробила/зробив відповідні посилання на джерела інформації;

що дана робота не передавалась іншим особам і подається вперше, не порушує авторських та суміжних прав закріплених статтями 21-25 Закону України «Про авторське право та суміжні права», а дані та інформація не отримувались в недозволений спосіб.

УСВІДОМЛЮЮ:

що ця робота може бути перевірена університетом на плагіат або інші порушення академічної доброчесності, в тому числі з використанням спеціалізованих сервісів;

що у разі порушення академічної доброчесності, до мене можуть бути застосовані процедури, передбачені законодавством України та Кодексом академічної доброчесності та корпоративної етики Донецького національного університету імені Василя Стуса, іншими локальними нормативними актами університету, та я можу бути притягнута/притягнутий до академічної відповідальності.

_____ _____

(дата) (підпис)