Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)
Навчально-науковий інститут комп'ютерних наук
та інформаційних технологій
(повне найменування інституту, назва факультету (відділення))
Кафедра комп'ютерних наук
(повна назва кафедри (предметної, циклової комісії))

Магістерська кваліфікаційна робота

другий (магістерський)
(рівень вищої освіти)

на тему: Інтелектуальна система для автоматизації створення навчального контенту методами генеративного штучного інтелекту

Виконав: студент VI курсу групи КН-62м
Спеціальності: 122 "Комп'ютерні науки"
(шифр і назва напряму підготовки, спеціальності)

Валявка Ю.А.
(прізвище та ініціали)

Керівник Лукащук Б.С., Дендюк М.В.
(прізвище та ініціали)

Рецензент Сторожук О.Л.
(прізвище та ініціали)

Львів – 2025

Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

ННІ комп'ютерних наук та інформаційних технологій
Кафедра комп'ютерних наук
Рівень вищої освіти другий (магістерський)
Спеціальність 122 "Комп'ютерні науки"
(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри комп'ютерних наук
_____ Борецька І.Б.
"10" грудня 2025 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Валявка Юрій Андрійович
(прізвище, ім'я, по батькові)

1. Тема роботи: Інтелектуальна система для автоматизації створення навчального контенту методами генеративного штучного інтелекту

Керівник роботи *Лукащук Б.С., асис., Дендюк М.В., к.т.н., доцент*,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від "29" квітня 2025 року №С-288

2. Термін подання студентом роботи 10.12.2025

3. Вихідні дані до роботи: створення інтелектуальної системи для автоматизації створення навчального контенту методами генеративного штучного інтелекту

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Розділ 1. Стан проблемної області

Розділ 2. Інформаційне забезпечення

Розділ 3. Математичне забезпечення

Розділ 4. Програмне забезпечення

Розділ 5. Розроблення стартап-проєкту

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Підготовка матеріалу до доповіді

6. Дата видачі завдання 01.05.2025

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1.	Огляд літератури згідно досліджуваної теми. Збір необхідних матеріалів.	01.05.25-20.06.25	Виконано
2.	Постановка задачі і її формалізація	21.06.25-04.07.25	Виконано
3.	Виконання вхідного етапу технології	05.07.25-29.08.25	Виконано
4.	Реалізація головних алгоритмів проєкту	30.08.25-07.10.25	Виконано
5.	Виконання етапу відлагодження проєкту	08.10.25-07.11.25	Виконано
6.	Виконання етапу впровадження та випуску бета-версії.	08.11.25-15.11.25	Виконано
7.	Оформлення записки до дипломного проекту.	16.11.25-07.12.25	Виконано

Студент _____ Валявка Ю.В.
(підпис) (прізвище та ініціали)

Керівники роботи _____ Лукащук Б.С.
(підпис) (прізвище та ініціали)

_____ Дендюк М.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Магістерська робота складається із 64 сторінок пояснювальної записки, містить 5 ілюстрацій, 1 таблицю та посилається на 18 джерел літератури.

Робота присвячена розробці веб-додатку для швидкої генерації мультимодального навчального контенту (текст, зображення, аудіо) на основі текстових запитів. Система автоматизує рутинну роботу викладачів та сприяє впровадженню мікронавчання. Розробку виконано мовою Python (фреймворк Streamlit). Для реалізації інтелектуальних функцій використано LLM Google Gemini, Pollinations AI та бібліотеку gTTS. Архітектура базується на асинхронній обробці запитів для високої швидкодії.

Ключові слова: Generative AI, LLM, Python, Streamlit, мікронавчання, автоматизація контенту, мультимодальність.

ABSTRACT

The master's thesis consists of 64 pages of explanatory notes, contains 5 illustrations, 1 table, and references 18 sources.

The thesis is devoted to the development of a web application for the rapid generation of multimodal educational content (text, images, audio) based on text prompts. The system automates the routine work of teachers and facilitates the implementation of micro-learning. The development was implemented using Python (Streamlit framework). To implement intelligent functions, the Google Gemini LLM, Pollinations AI, and the gTTS library were used. The architecture is based on asynchronous request processing to ensure high performance.

Keywords: Generative AI, LLM, Python, Streamlit, micro-learning, content automation, multimodality.

ТЕХНІЧНЕ ЗАВДАННЯ

Розробка системи обумовлена необхідністю створення ефективного, високошвидкісного інструменту для автоматизації та мультимодального збагачення освітнього процесу. Головною метою проєкту є впровадження інтелектуального функціоналу, який забезпечить формування комплексного навчального модуля, що строго відповідає семантичному запиту користувача та вимогам сучасного мікронавчання.

Завдання передбачає розробку повноцінного веб-додатку на базі фреймворку Streamlit, який слугуватиме інтерфейсом для взаємодії викладача з генеративними моделями штучного інтелекту. Користувачам має бути надана можливість оперативно визначати тему навчального курсу (), яка є основним критерієм генерації. Система повинна реалізувати механізм структурування та наповнення контенту, використовуючи для цього потужності великої мовної моделі, отримані через зовнішній сервіс — Google Gemini API.

Ключовим елементом, що надає системі інтелектуальності, є алгоритм асинхронної оркестрації запитів. Необхідно розробити та реалізувати програмний модуль, який на основі та отриманої JSON-структури автоматично розпаралелює процеси генерації медіа-ресурсів. Цей процес є центральним для гарантії того, що час створення ілюстрацій (через Pollinations AI) та синтезу мовлення (через gTTS) буде мінімальним, забезпечуючи реактивність системи.

Для забезпечення сучасності та ергономічності візуального представлення передбачається використання динамічної генерації HTML-коду. Весь інтерфейс фінального продукту має бути побудований з використанням стилістики Glassmorphism для забезпечення інтуїтивності та залученості користувача. Крім того, обов'язковою є функція автоматичної агрегації всіх згенерованих активів (текст, аудіо, зображення) у єдиний інтерактивний веб-ресурс, готовий до використання у браузері без додаткового налаштування.

Таким чином, розробка має забезпечити безшовну інтеграцію клієнтської логіки мовою Python, асинхронних архітектурних патернів та зовнішніх

генеративних API, що в результаті надасть користувачеві простий, але потужний інструмент для автоматизованого створення освітнього контенту.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ 8
ВСТУП 9
РОЗДІЛ 1. СТАН ПРОБЛЕМНОЇ ОБЛАСТІ 11
1.1. Аналіз сучасних підходів до дистанційної освіти та E-learning 11
1.2. Огляд технологій генеративного штучного інтелекту (Generative AI) 12
1.3. Порівняльний аналіз існуючих програмних продуктів та обґрунтування потреби в новій системі 15
1.4. Висновки до розділу 17
РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ 19
2.1. Характеристика вхідних та вихідних даних системи. 19
2.2. Структура та формат навчального контенту 20
2.3. Методи інтеграції мультимодальних даних 22
2.4. Висновки до розділу 23
РОЗДІЛ 3. МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ 25
3.1. Формалізація задачі автоматичної генерації контенту 25
3.2. Математичне моделювання часової складності та оцінка ефективності асинхронної обробки 27
3.3. Блок-схема взаємодії модулів системи та алгоритм функціонування 28
3.4. Висновки до розділу 31
РОЗДІЛ 4. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ 32
4.1. Вибір та обґрунтування технологічного стеку 32
4.2. Архітектура програмного комплексу 34
4.3. Реалізація ключових функціональних модулів 35
4.4. Валідація функціональності та метрики стійкості системи 38
4.5. Висновки до розділу 40
РОЗДІЛ 5. РОЗРОБЛЕННЯ СТАРТАП-ПРОЄКТУ 42
5.1. Обґрунтування актуальності та ринкова ніша 42
5.2. Аналіз конкурентного середовища та переваги 43
5.3. Економічна ефективність та інвестиційний план 45
5.4. Висновки до розділу 47
ВИСНОВКИ 48
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 50
ДОДАТКИ 52

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

API — прикладний інтерфейс програмування, набір визначень та протоколів для створення та інтеграції програмного забезпечення.

CSS — каскадні таблиці стилів, мова для опису зовнішнього вигляду документа, написаного мовою розмітки.

HTML — стандартизована мова розмітки документів для перегляду веб-сторінок у браузері.

HTTP — протокол передачі гіпертексту, що використовується для обміну даними в мережі Інтернет.

JSON — текстовий формат обміну даними, заснований на JavaScript, що легко читається людьми та машинами.

LLM — велика мовна модель, тип штучного інтелекту, навчений на великих масивах текстових даних для розуміння та генерації людської мови.

MVP — мінімально життєздатний продукт, версія продукту з достатньою кількістю функцій для задоволення перших клієнтів.

SaaS — модель надання програмного забезпечення як послуги, при якій постачальник розробляє веб-застосунок і самостійно керує ним.

SPA — односторінковий веб-застосунок, який завантажує одну HTML-сторінку і динамічно оновлює її вміст при взаємодії з користувачем.

TTS — технологія синтезу мовлення, що перетворює текст у звуковий форм

UI — інтерфейс користувача

UX — досвід користувача, сприйняття та реакція людини

ВСТУП

Актуальність теми. Сучасна освіта перебуває на етапі цифрової трансформації. Стрімке зростання обсягів інформації вимагає від викладачів та методистів постійного оновлення навчальних матеріалів, що є трудомістким процесом. Традиційні методи підготовки контенту (ручне написання текстів, пошук зображень, запис аудіо) займають значну частину робочого часу педагога, знижуючи ефективність навчального процесу.

Водночас, розвиток технологій генеративного штучного інтелекту (Generative AI), зокрема великих мовних моделей (LLM), нейромереж для генерації зображень та синтезу мови, відкриває нові можливості для автоматизації рутинних задач. Однак, більшість існуючих рішень є або вузькоспеціалізованими (лише текст або лише зображення), або комерційними та складними для інтеграції в єдиний навчальний потік.

Актуальність магістерської роботи полягає у необхідності створення комплексної інтелектуальної системи, яка здатна автоматизувати повний цикл створення мультимодального навчального контенту (текст, візуалізація, аудіосупровід), що дозволить значно скоротити часові витрати на підготовку матеріалів та підвищити їх якість.

Зв’язок роботи з науковими програмами, планами, темами. Робота виконується згідно з планом науково-дослідних робіт кафедри в напрямку впровадження інтелектуальних інформаційних технологій в освітню сферу.

Мета і задачі дослідження. Метою роботи є підвищення ефективності процесу створення навчального контенту шляхом розробки інтелектуальної системи, що використовує методи генеративного штучного інтелекту для автоматизованого формування структурованих, ілюстрованих та озвучених навчальних модулів.

Об’єкт дослідження – процес створення мультимодального навчального контенту в системах електронного навчання.

Предмет дослідження – методи та засоби генеративного штучного інтелекту, зокрема великі мовні моделі (LLM), генеративно-змагальні мережі (GAN) та алгоритми синтезу мови (TTS), для автоматизації освітніх процесів.

Методи дослідження. У роботі використано методи системного аналізу (для формулювання вимог до системи), методи об'єктно-орієнтованого проєктування (для розробки архітектури), теорію множин (для формалізації задачі генерації), а також методи асинхронного програмування (для реалізації паралельної обробки запитів).

Наукова новизна одержаних результатів полягає у наступному:

1. Удосконалено метод автоматизованої генерації навчального контенту за рахунок використання гібридної архітектури, що поєднує різні типи нейронних мереж (LLM для структуризації знань, Diffusion-моделі для візуалізації), що дозволяє створювати комплексні навчальні одиниці без втручання оператора.
2. Набув подальшого розвитку алгоритм асинхронної оркестрації запитів до зовнішніх AI-сервісів, що дозволило зменшити загальний час генерації контенту в 3-4 рази порівняно з послідовною обробкою.

Практичне значення одержаних результатів. Розроблено програмний продукт "Intelligent Course Generator" на базі технологій Google Gemini, Pollinations AI та Streamlit. Система дозволяє користувачеві за вказаною темою миттєво генерувати веб-сторінки з навчальним матеріалом, що містять структурований текст, тематичні ілюстрації та аудіо-лекції. Система може бути впроваджена в навчальних закладах для створення мікрокурсів, а також використана для самоосвіти.

РОЗДІЛ 1. СТАН ПРОБЛЕМНОЇ ОБЛАСТІ

1.1. Аналіз сучасних підходів до дистанційної освіти та E-learning

Сучасна освітня парадигма переживає фундаментальні зміни, зумовлені стрімкою цифровізацією суспільства [3]. Дистанційна освіта та електронне навчання (E-learning) трансформувалися з допоміжного інструменту в основну форму здобуття знань, особливо в умовах глобальних викликів, таких як пандемія COVID-19 та військові конфлікти, що унеможливлюють стабільний очний навчальний процес. Ринок E-learning демонструє стабільне зростання, переорієнтовуючись на технології, що забезпечують гнучкість, доступність та персоналізацію навчання [3].

На сьогоднішній день домінуючими у сфері дистанційної освіти залишаються системи управління навчанням (Learning Management Systems – LMS), такі як Moodle, Canvas, Google Classroom та Blackboard. Ці платформи чудово вирішують адміністративні задачі: реєстрацію студентів, облік успішності, організацію тестування та збереження файлів. Проте, з точки зору створення навчального контенту, вони залишаються переважно «пасивними» репозиторіями. Викладач змушений вручну завантажувати статичні файли (PDF-документи, презентації PowerPoint), що часто призводить до низького рівня залученості студентів та відсутності інтерактивності.

Паралельно з LMS розвивається напрямок масових відкритих онлайн-курсів (MOOC), представлений платформами Coursera, Udemy, EdX. Ці платформи пропонують високоякісний відеоконтент, створений професійними студіями. Однак процес виробництва таких курсів є надзвичайно дорогим та тривалим. Створення одного модуля може займати тижні або місяці роботи команди методистів, дизайнерів та відеоінженерів. Це робить такий підхід непридатним для оперативної підготовки матеріалів викладачами шкіл чи університетів, яким необхідно швидко реагувати на зміни в навчальних програмах.

Актуальним трендом сучасної освіти стає мікронавчання (Micro-learning) – підхід, що передбачає подачу матеріалу невеликими, логічно завершеними блоками (5–10 хвилин), сфокусованими на одній конкретній темі [9]. Дослідження показують,

що мікронавчання покращує засвоєння матеріалу на 17% порівняно з традиційними лекціями. Проте підготовка великої кількості атомарних одиниць контенту (коротких текстів, схем, аудіопоясснень) вручну потребує ще більше часових ресурсів від викладача.

Ключовою проблемою існуючих підходів є «контентна криза»: технології доставки знань (LMS, веб-сайти, мобільні додатки) значно випередили технології створення цих знань. Викладач залишається «вузьким місцем» системи, витрачаючи до 70% часу на рутинну підготовку матеріалів (написання текстів, пошук вільних від авторських прав зображень, запис та монтаж аудіо), замість того, щоб фокусуватися на педагогічній взаємодії та менторстві.

Крім того, більшість сучасних E-learning систем є уніфікованими та статичними. Вони пропонують однаковий контент для всіх користувачів, ігноруючи індивідуальні потреби у формі подачі матеріалу (наприклад, комусь краще сприймати інформацію візуально, а комусь – на слух).

Таким чином, аналіз предметної області свідчить про нагальну потребу в автоматизації процесу створення мультимодального навчального контенту. Необхідні нові інструменти, які б дозволяли генерувати якісні, структуровані та ілюстровані навчальні матеріали за лічені хвилини, використовуючи сучасні технології штучного інтелекту. Це дозволить поєднати переваги мікронавчання з економічною ефективністю та швидкістю підготовки матеріалів.

1.2. Огляд технологій генеративного штучного інтелекту (Generative AI)

Технологічний прорив останніх років пов'язаний із переходом від дискримінативних моделей штучного інтелекту (які класифікують або прогнозують дані) до генеративних моделей (Generative AI) [1, 2]. Цей клас алгоритмів здатний створювати принципово новий контент — тексти, програмний код, зображення, аудіо та відео, — що за якістю часто не поступається результатам роботи людини. Для автоматизації створення навчального контенту ключове значення мають три напрямки: великі мовні моделі (LLM), моделі генерації зображень та системи синтезу мови.

1.2.1. Великі мовні моделі (Large Language Models – LLM)

В основі сучасних текстових генераторів лежить архітектура Transformer, представлена дослідниками Google у 2017 році [6]. Ключовою особливістю трансформерів є механізм «уваги» (Self-Attention), який дозволяє моделі враховувати контекст всього речення або абзацу одночасно, а не послідовно, як це робили попередні рекурентні мережі (RNN). Це забезпечує глибоке розуміння семантики та здатність генерувати зв'язні, логічні та структуровані тексти.

На ринку домінують кілька ключових архітектур, кожна з яких має свої особливості:

GPT (Generative Pre-trained Transformer) від OpenAI: Серія моделей (GPT-3.5, GPT-4o), що демонструє високу креативність та здатність до виконання складних інструкцій. Проте їх використання у студентських та стартап-проєктах часто обмежується високою вартістю API та закритістю архітектури.

Claude від Anthropic: Моделі, сфокусовані на безпеці та великому контекстному вікні (до 200 тис. токенів), що дозволяє аналізувати цілі книги.

Gemini (раніше PaLM) від Google: Мультимодальне сімейство моделей, яке нативно розуміє не лише текст, а й зображення та код [5].

Для задач автоматизації створення мікрокурсів особливий інтерес становить модель Gemini 1.5 Flash. Її архітектура оптимізована для високої швидкості генерації (low latency) та ефективної роботи з великими обсягами даних при низькій вартості запиту. Важливою перевагою Gemini для розробки програмного забезпечення є нативна підтримка JSON-mode — режиму, в якому модель гарантовано повертає відповідь у форматі JSON [5], що критично важливо для подальшої програмної обробки даних у веб-додатках.

1.2.2. Технології генерації візуального контенту (Text-to-Image)

Візуалізація навчального матеріалу базується на дифузійних моделях (Diffusion Models). Принцип їх роботи полягає у поступовому відновленні зображення з "шуму" (випадкового набору пікселів) під керуванням текстового опису (промпту) [7].

Ключові представники:

DALL-E 3: Забезпечує високу точність слідування за текстом, але має високу ціну генерації.

Stable Diffusion: Відкрита модель, яка дозволяє гнучке налаштування, але потребує значних обчислювальних ресурсів (потужних GPU) для локального запуску.

Pollinations AI: Децентралізована платформа, яка надає API-доступ до різноманітних моделей генерації без необхідності налаштування власної інфраструктури.

Для розробки легкої веб-системи доцільним є використання хмарних API, таких як Pollinations, що дозволяє генерувати ілюстрації "на льоту" за URL-запитом, не навантажуючи сервер додатку важкими обчисленнями.

1.2.3. Синтез мовлення (Text-to-Speech – TTS)

Аудіосупровід (озвучення лекцій) реалізується за допомогою нейронних мереж синтезу мовлення. Сучасні Neural TTS системи (наприклад, WaveNet [8]) генерують звукову хвилю безпосередньо з лінгвістичних ознак тексту, що робить голос природним та інтонаційно багатим.

Існують два основні підходи до інтеграції:

Комерційні Cloud TTS (ElevenLabs, Azure Speech): Забезпечують найвищу якість "клонування" голосу, але є платними та мають ліміти символів.

Бібліотечні рішення (gTTS - Google Text-to-Speech): Використовують публічні API Google Translate. Хоча вони мають меншу варіативність емоцій, вони є безкоштовними, стабільними та ідеально підходять для освітніх цілей, де пріоритетом є чіткість вимови, а не художня виразність.

Аналіз технологічного стеку показує, що для створення автоматизованої системи генерації контенту найбільш ефективним є гібридний підхід: використання швидкісних LLM (Google Gemini) для структурування знань, хмарних дифузійних моделей (Pollinations) для візуалізації та перевірених TTS-рішень (gTTS) для озвучення. Така комбінація забезпечує баланс між якістю контенту, швидкістю роботи системи та економічною ефективністю розробки.

1.3. Порівняльний аналіз існуючих програмних продуктів та обґрунтування потреби в новій системі

Ринок інструментальних засобів для створення освітнього контенту (Authoring Tools) характеризується високою насиченістю та різноманіттям рішень. Проте детальний аналіз функціональних можливостей провідних програмних продуктів дозволяє виявити суттєві обмеження, що перешкоджають їх масовому впровадженню для задач швидкого створення мікрокурсів. Усі існуючі рішення можна умовно поділити на три категорії: професійні інструменти ручної розробки, AI-генератори презентацій та діалогові системи на базі LLM.

До першої категорії належать професійні платформи, такі як Articulate Storyline 360 та Adobe Captivate. Ці системи є індустріальним стандартом для створення SCORM-сумісних електронних курсів. Вони надають розробнику повний контроль над дизайном, інтерактивністю та логікою проходження курсу. Однак їх використання у повсякденній практиці викладача обмежене двома факторами: надзвичайно високою вартістю ліцензії (понад $1000 на рік) та високим порогом входження. Оволодіння інструментарієм цих програм вимагає спеціалізованого навчання, а створення одного курсу відбувається в ручному режимі, що не вирішує проблему часових витрат.

Другу категорію складають новітні веб-сервіси для генерації презентацій з використанням штучного інтелекту, такі як Gamma.app та Tome.app. Ці платформи демонструють високий рівень автоматизації: користувач вводить тему, а система генерує набір слайдів із текстом та зображеннями. Проте їхнім суттєвим недоліком є орієнтація на візуальний формат слайд-шоу, що є лише одним із компонентів навчального процесу. Більшість таких систем не мають вбудованого функціоналу для автоматичної генерації повноцінного аудіосупроводу (озвучення лекції) для кожного блоку контенту, або ж ця функція доступна лише у преміум-тарифах. Крім того, закритість архітектури цих пропрієтарних рішень унеможливлює їх інтеграцію в специфічні навчальні середовища або модифікацію алгоритмів генерації під власні потреби.

Третю категорію становлять великі мовні моделі з чат-інтерфейсом, такі як ChatGPT (OpenAI) або Claude [2]. Хоча вони здатні генерувати якісні навчальні тексти та сценарії, вони не є комплексними засобами розробки контенту. Процес створення уроку за їх допомогою залишається фрагментованим: викладач змушений генерувати текст в одному вікні, створювати ілюстрації в іншому (наприклад, Midjourney), а озвучувати матеріал у третьому сервісі. Відсутність єдиного автоматизованого конвеєра («pipeline»), який би об'єднував генерацію тексту, медіа та коду в один клік, нівелює переваги швидкості роботи самої нейромережі.

Для наочного зіставлення характеристик розглянутих систем та розроблюваного програмного комплексу дані зведено у таблицю 1.1.

Таблиця 1.1 – Порівняльна характеристика систем генерації навчального контенту

Характеристика системи	ChatGPT (Plus)	Gamma / Tome	Articulate 360	Розроблена система
Основна функція	Генерація тексту	Створення слайдів	Ручна верстка курсів	Генерація веб-уроків
Рівень автоматизації	Частковий (текст)	Високий (слайди)	Низький (ручний)	Повний (Мультимодальний)
Інтеграція медіа	Потребує плагінів	Текст + Фото	Ручне додавання	Текст + Фото + Аудіо
Аудіосупровід (TTS)	Обмежено	Обмежено / Відсутнє	Вбудований редактор	Автоматична генерація
Архітектура	Закрита (SaaS)	Закрита (SaaS)	Пропрієтарне ПЗ	Відкрита (Open Source)
Вартість	~$20/міс	Freemium / Платна	~$1000+/рік	Безкоштовно / API

Як видно з наведеного аналізу, на ринку існує незаповнена ніша доступних інструментів, які б забезпечували повну автоматизацію циклу «Текст – Зображення – Звук» без необхідності ручного редагування та значних фінансових вкладень. Розроблювана інтелектуальна система **«Intelligent Course Generator»** покликана вирішити цю проблему шляхом застосування гібридного підходу, що поєднує можливості сучасних API (Gemini, Pollinations) з гнучкістю відкритого програмного коду на Python. Це дозволяє створити економічно ефективне рішення, здатне генерувати комплексні мультимодальні навчальні модулі за лічені секунди, що є критично важливим для сучасної динамічної освіти.

1.4. Висновки до розділу

У першому розділі магістерської роботи проведено комплексний аналіз стану проблемної області, пов’язаної з автоматизацією створення навчального контенту в системах дистанційної освіти. Результати дослідження дозволяють зробити низку узагальнюючих висновків, що визначають напрямок подальшої розробки.

По-перше, аналіз сучасних підходів до E-learning засвідчив, що попри широке впровадження систем управління навчанням (LMS) та платформ масових онлайн-курсів, процес підготовки навчальних матеріалів залишається переважно ручним та трудомістким. Виявлено наявність «контентної кризи», коли викладачі витрачають непропорційно велику кількість часу на технічну підготовку матеріалів (написання текстів, пошук ілюстрацій, озвучення), що знижує загальну ефективність освітнього процесу та перешкоджає впровадженню методик мікронавчання.

По-друге, огляд технологічного стеку підтвердив, що сучасний рівень розвитку генеративного штучного інтелекту досяг необхідної зрілості для вирішення окреслених проблем. Встановлено, що найбільш перспективним є гібридний підхід, який передбачає поєднання великих мовних моделей (зокрема Google Gemini) для структурування знань, дифузійних моделей для візуалізації контексту та нейронних мереж синтезу мовлення для забезпечення інклюзивності та мультимодальності навчання. Така комбінація дозволяє автоматизувати рутинні операції, які раніше виконувалися людиною.

По-третє, порівняльний аналіз існуючих програмних аналогів виявив суттєву прогалину на ринку інструментальних засобів. Існуючі рішення або є професійними системами з високою вартістю ліцензії та складним порогом входження, або ж є вузькоспеціалізованими генераторами, які не забезпечують повного циклу створення мультимедійного контенту (поєднуючи текст, графіку та аудіо). Жоден із розглянутих масових продуктів не пропонує доступного, повністю автоматизованого механізму перетворення текстової теми у готовий веб-урок без необхідності ручного редагування.

Таким чином, результати розділу підтверджують актуальність та доцільність розробки власної інтелектуальної системи. Вона має базуватися на мікросервісній архітектурі з використанням сучасних API генеративного штучного інтелекту, що дозволить створити економічно ефективний, швидкий та зручний інструмент для автоматизації роботи викладачів та методистів. Це створює обґрунтоване підґрунтя для переходу до етапу проєктування інформаційного та математичного забезпечення системи у наступних розділах.

РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1. Характеристика вхідних та вихідних даних системи.

Функціонування розробленої інтелектуальної системи базується на складних процесах трансформації та семантичного збагачення інформаційних потоків. Головною особливістю програмного комплексу є його здатність генерувати масштабний, мультимодальний освітній продукт на основі мінімального обсягу вхідної інформації, вирішуючи таким чином задачу автоматизації творчої праці викладача.

Вхідні дані системи класифікуються як слабкоструктуровані текстові масиви природною мовою. Основним елементом управління процесом генерації виступає текстовий запит користувача (User Prompt), який визначає тематику та семантичне ядро майбутнього навчального курсу. Цей запит може бути сформульований у довільній формі, варіюючись від лаконічної назви дисципліни до розгорнутого опису специфічної теми. Окрім змістовної складової, до вхідних даних також належать авторизаційні параметри, зокрема API-ключі доступу до хмарних сервісів Google Generative AI. Ці ключі забезпечують легітимність запитів до великих мовних моделей та дозволяють ідентифікувати сесію користувача. Важливою характеристикою вхідних даних є їхня висока варіативність та невизначеність, що вимагає від системи використання адаптивних алгоритмів обробки природної мови для коректної інтерпретації намірів користувача незалежно від мови введення чи використаної термінології.

На відміну від вхідних даних, вихідні дані системи є строго структурованим, ієрархічним набором мультимедійних об’єктів, що формують готовий до використання освітній ресурс. Результатом роботи програмного комплексу є комплексний веб-продукт, який органічно поєднує три основні модальності сприйняття інформації: текстову, візуальну та аудіальну.

Текстова складова вихідних даних представлена у форматі логічно впорядкованих навчальних модулів. Кожен модуль містить структурований теоретичний матеріал, адаптований під формат мікронавчання, а також супровідні

метадані, такі як заголовки розділів та спеціалізовані технічні описи (промпти) для подальшої генерації зображень. Ця інформація формується та передається всередині системи у форматі об'єктної нотації JSON, що забезпечує стандартизацію даних та зручність їх програмної обробки на наступних етапах генерації.

Графічна складова вихідних даних реалізується у вигляді набору унікальних ілюстрацій у растрових форматах (JPEG або PNG). Ці зображення створюються генеративними нейромережами динамічно, базуючись на семантичному контексті кожного конкретного розділу, і слугують для візуалізації абстрактних понять та підвищення залученості здобувачів освіти. Аудіальна складова представлена набором аудіофайлів у форматі MP3, які містять синтезоване мовлення лектора. Синтез відбувається на основі текстового вмісту розділів, що забезпечує повну змістову відповідність між прочитаним та почутим матеріалом, сприяючи інклюзивності навчання.

Фінальним агрегованим результатом роботи системи, який отримує кінцевий користувач, є програмний код веб-сторінки у форматі HTML. Цей файл містить інтегровані каскадні таблиці стилів (CSS) для забезпечення сучасного візуального оформлення (Glassmorphism) та скрипти взаємодії (JavaScript) для управління відтворенням мультимедіа. Такий формат вихідних даних дозволяє використовувати згенерований курс у будь-якому сучасному веб-браузері без необхідності встановлення спеціалізованого програмного забезпечення, що забезпечує кросплатформеність та доступність розробленого рішення.

2.2. Структура та формат навчального контенту

Критично важливим етапом у проектуванні інтелектуальної системи є формалізація структури вихідних даних, що генеруються великою мовною моделлю. Оскільки генеративні нейромережі за своєю природою є стохастичними та схильними до варіативності у формулюванні відповідей, для забезпечення стабільної роботи програмного комплексу необхідно накласти суворі обмеження на формат обміну даними. У розробленій системі це реалізується шляхом застосування методів інженерії промптів, які директивно зобов’язують модель Gemini повертати

результати виключно у форматі об'єктної нотації JavaScript (JSON). Цей вибір обумовлений високою інтероперабельністю формату JSON, його легкою читабельністю для людини та нативною підтримкою у мові програмування Python, що використовується для розробки бекенду.

Розроблена інформаційна модель навчального модуля представляє собою ієрархічну структуру даних. На верхньому рівні ієрархії знаходиться кореневий об'єкт, що містить масив розділів курсу. Кожен елемент цього масиву є самостійним об'єктом, який описує окрему дидактичну одиницю — главу або урок. Внутрішня структура об'єкта глави складається з трьох обов'язкових атрибутів, кожен з яких виконує специфічну функцію у процесі мультимодальної генерації. Першим атрибутом є заголовок (title), який виконує роль семантичного ідентифікатора та навігаційного елемента у фінальному веб-інтерфейсі.

Другим атрибутом є основний змістовий блок (description), що містить текстове пояснення навчального матеріалу. Для забезпечення відповідності принципам мікронавчання, на етапі формування промпту накладаються обмеження на обсяг цього тексту. Це дозволяє генерувати лаконічні, сконцентровані пояснення, які оптимально підходять для подальшого озвучення системами синтезу мовлення, уникаючи ефекту монотонності та перевантаження слухача. Саме цей текстовий атрибут виступає вхідним параметром для модуля TTS (Text-to-Speech).

Третім, і найбільш технологічно значущим атрибутом інформаційної моделі, є технічний опис для генерації зображення (image_description). Необхідність виділення цього поля в окрему сутність зумовлена специфікою роботи дифузійних нейромереж. Текст, призначений для читання студентом, часто містить абстрактні поняття, які не можуть бути прямо візуалізовані (наприклад, «економічна криза» або «філософія»). Тому мовна модель отримує інструкцію виконувати функцію «перекладача смислів»: вона трансформує абстрактний навчальний текст у конкретний набір візуальних образів та стилістичних вказівок (наприклад, «графік спаду на екрані монітора, офісне освітлення, реалістичний стиль»), які будуть зрозумілі для сервісу генерації зображень.

Така сувора типізація та сегрегація даних забезпечує модульність та відмовостійкість системи. Отримавши валідований JSON-об'єкт, програмний контролер має можливість незалежно та паралельно обробляти кожен атрибут: текст відправляється на сервіс озвучення, а візуальний промпт — на сервіс генерації графіки. Це унеможливлює помилки парсингу та гарантує, що навіть при зміні тематики запиту внутрішня логіка побудови курсу залишиться незмінною.

2.3. Методи інтеграції мультимодальних даних

Інтеграція гетерогенних потоків даних у єдину систему базується на принципах асинхронної оркестрації, що є визначальним фактором для забезпечення високої продуктивності програмного комплексу. На відміну від традиційних лінійних алгоритмів, де кожен етап обробки інформації виконується послідовно і блокує виконання наступного до свого завершення, розроблена система використовує подійно-орієнтовану модель [14]. Центральним елементом цієї архітектури виступає асинхронний контролер, реалізований засобами мови Python, який виконує функцію диспетчера даних між зовнішніми програмними інтерфейсами та внутрішньою логікою додатка.

Процес інтеграції даних розпочинається з етапу семантичного аналізу та генерації структури курсу. Система ініціює запит до великої мовної моделі з інструкцією виступити у ролі методиста та сформувати каркас навчального матеріалу у форматі JSON. Отриманий об'єкт проходить процедуру десеріалізації та валідації, в ході якої перевіряється наявність усіх необхідних полів та їх відповідність типам даних. Цей етап є критичним, оскільки він трансформує неструктурований текст у набір чітких інструкцій для подальшої роботи генеративних модулів.

Наступним етапом є паралельна генерація медіа-ресурсів, яка демонструє головну перевагу асинхронного підходу. Отримавши валідований список розділів, контролер ініціює одночасне виконання множини незалежних завдань. Для навчального модуля, що складається з певної кількості глав, система створює відповідну кількість конкурентних потоків запитів: частина з них спрямовується до сервісу генерації зображень для створення візуального ряду, а інша частина — до

сервісу синтезу мовлення для створення аудіосупроводу. Такий підхід дозволяє скоротити загальний час генерації контенту пропорційно до кількості розділів, оскільки система не очікує завершення генерації одного файлу перед початком роботи над іншим, а використовує час очікування мережевої відповіді для відправки нових запитів.

Завершальним етапом інтеграції є агрегація отриманих результатів та ін'єкція даних у фінальний продукт. Після успішного отримання всіх медіа-файлів система зберігає їх у тимчасовому файловому сховищі та генерує відповідні локальні посилання. Фінальна збірка відбувається шляхом динамічної побудови HTML-документа. Система використовує заздалегідь підготовлений шаблон дизайну, в який програмним шляхом вбудовуються отримані текстові дані та шляхи до медіа-ресурсів. Важливою особливістю обраного методу інтеграції є відсутність необхідності у використанні важких систем управління базами даних, оскільки вся інформація генерується та структурується в оперативній пам'яті та файловій системі в режимі реального часу, що значно спрощує архітектуру та підвищує переносимість розробленого рішення.

2.4. Висновки до розділу

У другому розділі магістерської роботи проведено детальне проєктування інформаційного забезпечення інтелектуальної системи, що є необхідною передумовою для її подальшої програмної реалізації. Результати досліджень дозволили сформувати цілісну картину інформаційних потоків та методів їх обробки.

Встановлено, що ключовим завданням інформаційного забезпечення є трансформація слабкоструктурованих вхідних даних, представлених у вигляді текстових запитів природною мовою, у суворо типізований набір мультимедійних об’єктів. Визначено склад вхідних та вихідних даних системи, що дозволило чітко окреслити межі її функціонування та вимоги до інтерфейсів взаємодії. Розроблено уніфіковану інформаційну модель навчального модуля на базі формату JSON, яка виступає універсальним протоколом обміну даними між генеративною текстовою моделлю та сервісами створення медіа-контенту.

Обґрунтовано необхідність введення до структури даних окремого атрибута для технічного опису зображень, що дозволяє адаптувати абстрактний навчальний текст до специфіки сприйняття дифузійних нейромереж. Запропоновано та описано метод інтеграції мультимодальних даних, що базується на асинхронній оркестрації запитів. Доведено, що такий підхід забезпечує паралельне виконання ресурсомістких завдань генерації, що критично важливо для досягнення високої продуктивності системи та забезпечення позитивного досвіду користувача. Спроєктована структура інформаційного забезпечення є модульною, масштабованою та не вимагає використання складних систем управління базами даних, що значно спрощує архітектуру програмного комплексу та підвищує його надійність. Отримані результати створюють необхідну базу для переходу до етапу математичного моделювання та безпосередньої розробки програмного коду системи.

РОЗДІЛ 3. МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1. Формалізація задачі автоматичної генерації контенту

Розробка інтелектуальної системи вимагає чіткої математичної формалізації процесу перетворення вхідних даних у вихідний інформаційний продукт. Задачу автоматизованої генерації мультимодального навчального контенту можна представити як задачу відображення простору текстових запитів користувача у простір структурованих веб-ресурсів [1, 2].

Нехай T — вхідний текстовий запит (тема курсу), що належить до множини всіх можливих текстових рядків $\sum*$. Метою функціонування системи є побудова відображення F, яке трансформує вхідний запит T у вихідний навчальний об'єкт E:

$$E = F(T) \text{ де } T \in E \tag{3.1}$$

Вихідний об'єкт E є складним структурованим набором даних, який можна представити як впорядковану множину навчальних модулів (розділів). Нехай n — кількість розділів у курсі (у реалізованій системі n = 4). Тоді об'єкт E визначається як вектор:

$$E = \{Ch1, Ch2, \dots, Chn\} \tag{3.2}$$

Кожен окремий навчальний модуль Ch*i* (де i = 1,…, n) є мультимодальним об'єктом, що складається з різнорідних компонентів. Формально Ch*i* можна описати як упорядкований кортеж трьох елементів:

$$Chi = < Txti, Imgi, Audi > \tag{3.3}$$

У цьому кортежі: Txt*i* — текстова складова розділу (семантичний опис), що належить множині текстів; Img*i* — графічна складова (ілюстрація), що належить множині растрових зображень; Aud*i* — аудіальна складова (звуковий файл), що належить множині аудіопотоків.

Процес отримання компонентів кортежу Ch*i* базується на використанні композиції функцій генеративних моделей. Визначимо множину доступних у системі інтелектуальних агентів (моделей) як M:

$$M = \{M_{MML}, M_{IMG}, M_{TTS}\} \tag{3.4}$$

Тут M_{MML} — велика мовна модель (Google Gemini), функцією якої є генерація структурованого тексту та промптів; M_{IMG} — модель генерації зображень (Pollinations AI); M_{TTS} — модель синтезу мовлення (gTTS).

Першим етапом генерації є застосування моделі M_{LLM} до вхідного запиту T для отримання проміжної структури даних S у форматі JSON. Цю операцію можна записати як:

$$S = M_{MML}\ (T, P_{sys}\ ) \tag{3.5}$$

де P_{sys} — системний промпт, що задає обмеження формату та ролі. Структура S містить набір пар текстових описів та візуальних інструкцій для кожного розділу: S = {(d*i*, p*i*),…, (d*n*, p*n*)}, де d*i* — навчальний текст, а p*i* — промпт для генерації зображення.

На основі отриманої структури S відбувається генерація компонентів кожного модуля. Текстова складова є безпосереднім відображенням d*i*. Графічна складова є результатом застосування функції генерації зображень до візуального промпту p*i*:

$$Img_i = M_{IMG}\ (pi\ ) \tag{3.6}$$

Аудіальна складова є результатом застосування функції синтезу мовлення до текстового опису Aud*i*:

$$Aud_i = M_{TTS}\ (di\ ) \tag{3.7}$$

Таким чином, загальну функцію системи можна представити як суперпозицію функцій генеративних моделей, що виконуються над результатами семантичної декомпозиції вхідного запиту. Задача системи зводиться до мінімізації часу виконання цієї суперпозиції функцій при збереженні семантичної зв'язності компонентів кортежу. Ця формалізація дозволяє перейти до розробки алгоритмів асинхронної обробки даних, оскільки генерація компонентів та для різних є незалежними подіями.

3.2. Математичне моделювання часової складності та оцінка ефективності асинхронної обробки

Ключовою технічною проблемою при розробці мультимодальних генеративних систем є висока латентність (затримка) при роботі із зовнішніми API. Генерація зображень та синтез мовлення є ресурсомісткими операціями, виконання яких на стороні сервера-провадера може займати від кількох секунд до хвилини. У випадку використання класичної синхронної моделі виконання коду, загальний час генерації курсу лінійно зростає пропорційно кількості навчальних модулів, що призводить до неприйнятного для користувача часу очікування. Для вирішення цієї проблеми у роботі застосовано алгоритм асинхронної обробки запитів, ефективність якого можна довести математично.

Розглянемо часову модель процесу генерації. Нехай — кількість розділів у курсі. Процес створення контенту складається з трьох послідовних етапів: генерація тексту (єдиний запит), генерація зображень (запитів) та генерація аудіо (запитів). Позначимо час виконання окремих операцій наступним чином: — час генерації текстової структури (LLM); — час генерації зображення для -го розділу; — час генерації аудіосупроводу для -го розділу.

У випадку використання послідовного (синхронного) алгоритму, контролер системи очікує завершення поточної операції перед початком наступної. Загальний час виконання визначається як сума часів усіх атомарних операцій:

$$T_{seq} = t_{text} + \sum_{i=1}^{n} \left(t_{img}^{(i)} + t_{aud}^{(i)} \right) \tag{3.8}$$

Аналіз цієї формули показує, що часова складність алгоритму має лінійну залежність від , тобто . При збільшенні кількості розділів час очікування зростає критично. Наприклад, якщо генерація одного зображення займає в середньому 5 секунд, а аудіо — 3 секунди, то для курсу з 4 розділів лише медіа-генерація займе секунди, що суттєво погіршує інтерактивність системи.

Для оптимізації цього процесу в роботі реалізовано асинхронний алгоритм на базі подійно-орієнтованої моделі (Event Loop) [14]. Оскільки операції звернення до API відносяться до класу I/O-bound (обмежених швидкістю введення-виведення, а не

процесором), вони можуть виконуватися квазіпаралельно. Система відправляє запити до генеративних сервісів одночасно, не блокуючи основний потік виконання програми. У цьому випадку час генерації всіх медіа-ресурсів визначається не їх сумою, а тривалістю виконання найдовшого окремого запиту.

Математична модель часу виконання розробленого асинхронного алгоритму описується формулою:

$$T_{async} = t_{text} + \max_{i=1..n} \left(\max \left(t_{img}^{(i)}, t_{aud}^{(i)} \right) \right) + \delta \tag{3.9}$$

де δ — незначні накладні витрати на перемикання контексту та агрегацію результатів. Як видно з формули, час виконання медіа-генерації більше не залежить прямо від кількості розділів n (за умови, що пропускна здатність мережі та ліміти API дозволяють одночасну обробку запитів). Часова складність для етапу медіа знижується до відносно кількості розділів, обмежуючись лише найповільнішим сервісом.

Порівнюючи дві моделі, можна визначити коефіцієнт прискорення (Speedup), який забезпечує розроблена система. Приймаючи для спрощення, що час генерації всіх зображень є приблизно однаковим і час генерації аудіо також, отримуємо:

$$S = \frac{T_{seq}}{T_{async}} \approx \frac{t_{text} + n(t_{img} + t_{aud})}{t_{text} + \max(t_{img}, t_{aud})} \tag{3.10}$$

При великих значеннях та та , знаменник дробу значно менший за чисельник. Це математично доводить, що впровадження асинхронної архітектури дозволяє досягти кратного зменшення часу відгуку системи, що є необхідною умовою для забезпечення її конкурентоспроможності та відповідності вимогам реального часу.

3.3. Блок-схема взаємодії модулів системи та алгоритм функціонування

Для забезпечення цілісного розуміння принципів роботи розробленої інтелектуальної системи необхідно формалізувати логіку взаємодії її компонентів у вигляді алгоритмічної структури. Процес генерації навчального контенту є складною послідовністю операцій, які можна розділити на три основні фази: ініціалізація, асинхронна генерація та агрегація результатів. Кожна фаза реалізується відповідними програмними модулями, взаємодія між якими регулюється контролером додатка.

Алгоритм розпочинає свою роботу з фази ініціалізації, яка активується дією користувача через веб-інтерфейс. На цьому етапі система отримує вхідний текстовий запит та API-ключ авторизації. Першою операцією є валідація вхідних даних: система перевіряє наявність ключа та непорожність запиту. У разі успішної валідації, управління передається модулю комунікації з LLM. Контролер формує складний системний промпт, який містить рольову установку для нейромережі («діяти як методист»), вимоги до формату виводу (JSON) та обмеження щодо структури контенту (кількість розділів, обсяг тексту). Сформований запит надсилається до API Google Gemini.

Наступним кроком є обробка відповіді від мовної моделі. Система отримує текстовий потік, очищує його від можливих маркерів форматування (таких як markdown-теги коду) та здійснює парсинг у внутрішню структуру даних Python (список словників). Якщо на цьому етапі виникає помилка (наприклад, модель повернула некоректний JSON), спрацьовує механізм обробки виключень, який ініціює повторний запит або повідомляє користувача про збій. У разі успішного парсингу система переходить до найбільш ресурсомісткої фази — генерації медіа-контенту.

Фаза асинхронної генерації є ядром алгоритму. Контролер ітерує отриманий список розділів та для кожного елемента створює дві незалежні задачі (Task): задачу генерації зображення та задачу синтезу мовлення. Для генерації зображень система вилучає поле image_description з об'єкта розділу, кодує його в URL-сумісний формат та формує запит до API Pollinations. Отриманий потік байтів зберігається у локальній файловій системі як зображення формату JPEG. Паралельно з цим, текстовий опис розділу (description) передається до бібліотеки gTTS, яка конвертує текст у аудіопотік та зберігає його як MP3-файл. Важливою особливістю цього етапу є те, що всі завдання для всіх розділів запускаються в конкурентному режимі, використовуючи цикл подій (Event Loop), що дозволяє максимально ефективно використовувати час очікування мережевих відповідей.

Фінальна фаза агрегації розпочинається після успішного завершення всіх асинхронних задач. Система оновлює структуру даних, додаючи до кожного об'єкта

розділу локальні шляхи до згенерованих медіа-файлів. На основі оновленої структури генератор HTML формує вихідний код веб-сторінки. Він використовує заздалегідь підготовлений шаблон, що містить CSS-стилі для візуалізації карток та JS-скрипти для управління медіа-плеєрами, і ін'єктує в нього отриманий контент. Завершується алгоритм автоматичним відкриттям згенерованого файлу у веб-браузері користувача, що знаменує собою успішне виконання цільової функції системи.

Блок-схема на рисунку 3.1

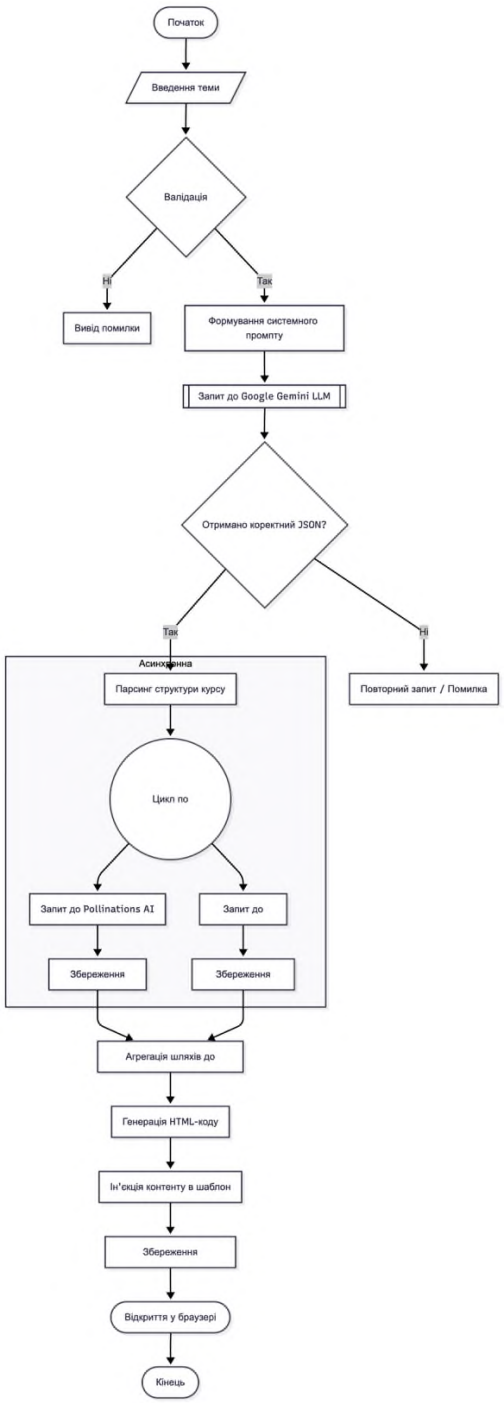

Рисунок 3.1 – Блок-схема

3.4. Висновки до розділу

У третьому розділі виконано математичне моделювання та алгоритмізацію процесів функціонування інтелектуальної системи.

По-перше, проведено формалізацію задачі генерації контенту як функції відображення множини текстових запитів у впорядковану структуру мультимедійних об'єктів. Це дозволило чітко визначити взаємозв'язки між компонентами системи та обґрунтувати вибір методів їх реалізації.

По-друге, розроблено та математично обґрунтовано алгоритм асинхронної обробки запитів. Аналіз часової складності довів, що запропонований підхід дозволяє знизити залежність часу генерації від кількості розділів з лінійної до константної (відносно найдовшого запиту), що забезпечує необхідну для інтерактивних систем швидкодію. Розрахунковий коефіцієнт прискорення підтверджує ефективність паралелізації завдань генерації медіа-ресурсів.

По-третє, розроблено детальну блок-схему алгоритму, яка описує логіку взаємодії модулів системи на етапах ініціалізації, генерації та агрегації результатів. Описана модель управління потоками даних забезпечує стабільність роботи додатка та коректну обробку виключних ситуацій.

Результати математичного та алгоритмічного проєктування створюють надійну базу для наступного етапу — програмної реалізації системи, яка буде описана у четвертому розділі.

РОЗДІЛ 4. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

4.1. Вибір та обґрунтування технологічного стеку

Успішна реалізація інтелектуальної системи генерації контенту значною мірою залежить від правильного вибору інструментальних засобів розробки. Технологічний стек проекту було сформовано виходячи з вимог до швидкодії, масштабованості та простоти інтеграції з сервісами штучного інтелекту. Основним критерієм вибору стала необхідність швидкого розгортання веб-інтерфейсу (Rapid Application Development) та ефективної роботи з асинхронними мережевими запитами.

В якості базової мови програмування обрано Python 3.11. Цей вибір обумовлений її домінуючим положенням у сфері розробки систем штучного інтелекту та машинного навчання [10]. Python надає розробнику доступ до найширшої екосистеми бібліотек для роботи з API, обробки природної мови та мультимедіа. Важливою перевагою сучасних версій Python є вдосконалена підтримка асинхронного програмування через нативну бібліотеку asyncio, що є критично важливим для реалізації паралельної архітектури генерації контенту, описаної у попередньому розділі [14]. Використання статично типізованих мов (C++, Java) у даному контексті було б надлишковим та збільшило б час розробки без суттєвого виграшу у продуктивності для задач, що обмежені швидкістю мережі (I/O-bound).

Для створення користувацького інтерфейсу обрано фреймворк Streamlit [11]. На відміну від традиційних веб-фреймворків (Django, Flask) або бібліотек фронтенду (React, Vue), Streamlit дозволяє створювати інтерактивні веб-додатки виключно засобами мови Python, не вимагаючи написання HTML/CSS/JS коду вручну. Це значно спрощує архітектуру проекту, перетворюючи його на монолітний скрипт, який легко розгортати та підтримувати. Streamlit ідеально підходить для демонстрації можливостей AI-моделей, оскільки має вбудовані віджети для відображення прогресу, відтворення аудіо та візуалізації даних, що повністю покриває функціональні вимоги до системи.

Взаємодія з великими мовними моделями реалізована за допомогою офіційної клієнтської бібліотеки google-generativeai [12]. Вона надає зручний інтерфейс для роботи з сімейством моделей Gemini, підтримуючи потокову передачу даних (streaming) та налаштування параметрів генерації (temperature, top-k). Вибір саме цієї бібліотеки гарантує стабільність з'єднання та сумісність з останніми оновленнями API від Google.

Для реалізації асинхронних HTTP-запитів до сервісів генерації зображень використано бібліотеку aiohttp. На відміну від стандартної бібліотеки requests, яка працює у синхронному режимі і блокує виконання програми до отримання відповіді від сервера, aiohttp дозволяє створювати неблокуючі сесії. Це дає можливість відправляти десятки запитів одночасно та обробляти відповіді по мірі їх надходження, що є фундаментом швидкої роботи системи. Для роботи з файловою системою в асинхронному режимі (запис зображень та аудіо на диск) використано бібліотеку aiofiles, що запобігає «заморожуванню» інтерфейсу під час операцій введення-виведення.

Синтез мовлення реалізовано за допомогою бібліотеки gTTS (Google Text-to-Speech) [16]. Її вибір обумовлений відкритістю, відсутністю жорстких лімітів на кількість запитів та високою якістю нейронного синтезу, яку забезпечують сервери Google. Це дозволяє генерувати озвучення лекцій без необхідності інтеграції дорогих комерційних API, зберігаючи при цьому природність звучання та правильну інтонацію.

Таким чином, обраний технологічний стек (Python + Streamlit + AsyncIO) є оптимальним для вирішення поставленої задачі. Він забезпечує баланс між швидкістю розробки, продуктивністю системи та якістю кінцевого продукту, дозволяючи реалізувати складну логіку асинхронної генерації мультимедіа у вигляді зручного та сучасного веб-додатку.

4.2. Архітектура програмного комплексу

Програмна реалізація інтелектуальної системи базується на модульній архітектурі, що забезпечує чітке розмежування логіки інтерфейсу користувача, бізнес-логіки обробки даних та взаємодії із зовнішніми сервісами. Розроблений програмний комплекс функціонує як клієнт-серверний додаток, де роль [4] сервера виконує локальне середовище виконання Python або хмарний контейнер Streamlit Cloud, а клієнтом виступає веб-браузер користувача. Така архітектура дозволяє забезпечити кросплатформеність та легкість розгортання системи без прив'язки до специфічного апаратного забезпечення кінцевого користувача [4, 13].

Структурно програмний комплекс складається з трьох основних рівнів: рівня представлення, рівня бізнес-логіки та рівня даних [4]. Рівень представлення (Presentation Layer) реалізовано засобами фреймворку Streamlit. Він відповідає за візуалізацію елементів управління (полів вводу, кнопок), відображення індикаторів прогресу виконання завдань та фінальну презентацію згенерованого контенту. Важливою особливістю цього рівня є реактивність: інтерфейс автоматично оновлюється у відповідь на зміни внутрішнього стану програми, що забезпечує інтуїтивно зрозумілу взаємодію з користувачем під час тривалих процесів генерації медіа.

Рівень бізнес-логіки (Business Logic Layer) є ядром системи і реалізований у вигляді набору асинхронних функцій мовою Python. Цей рівень виконує функцію оркестратора, керуючи потоками даних між компонентами системи. Він включає модулі валідації вхідних даних, модуль формування промптів для великих мовних моделей, а також підсистему управління асинхронними задачами. Саме тут реалізовано алгоритм паралельної обробки запитів, який ініціює та контролює виконання завдань генерації зображень та синтезу мовлення, обробляє можливі помилки мережевої взаємодії та агрегує отримані результати в єдину структуру даних.

Рівень даних та зовнішніх інтерфейсів (Data & External Interfaces Layer) відповідає за комунікацію із зовнішніми хмарними API та файловою системою. Взаємодія з інтелектуальними сервісами Google Gemini та Pollinations AI відбувається

через захищені канали зв'язку з використанням API-ключів. Отримані мультимедійні дані (зображення, аудіофайли) зберігаються у тимчасовій директорії локальної файлової системи, структура якої динамічно формується для кожного нового запиту. Це забезпечує ізоляцію даних різних сесій та спрощує процес очищення кешу.

Фізична структура програмного проекту організована таким чином, щоб забезпечити легкість супроводу та розширення коду. Головний виконуваний файл (app.py або auto_teacher.py) містить точку входу в додаток та конфігурацію інтерфейсу. Допоміжні функції, що відповідають за санітизацію імен файлів, перевірку наявності необхідних директорій та роботу з мережею, винесені в окремі блоки коду. Така організація дозволяє незалежно тестувати та модифікувати окремі компоненти системи, наприклад, замінити модуль синтезу мовлення на інший без необхідності вносити зміни в логіку генерації зображень чи інтерфейсу користувача.

Запропонована архітектура поєднує в собі гнучкість мікросервісного підходу (завдяки використанню незалежних API) з простотою монолітного розгортання, що є оптимальним рішенням для задач швидкого прототипування та створення навчальних систем нового покоління.

4.3. Реалізація ключових функціональних модулів

Програмна реалізація системи складається з низки взаємопов’язаних модулів, кожен з яких відповідає за окремий етап обробки даних. Ключовим елементом, що забезпечує інтелектуальну складову системи, є модуль взаємодії з великою мовною моделлю.

Функція get_gemini_content реалізує логіку отримання структурованих даних. Для забезпечення відмовостійкості системи в ній реалізовано алгоритм автоматичного перебору доступних моделей (Fallback Strategy). Система послідовно намагається встановити з’єднання з різними версіями моделі Gemini (від найновішої 1.5 Flash до стабільних версій Pro), що гарантує працездатність додатка навіть у випадку тимчасової недоступності конкретного API-ендпоінту або змін у політиці доступу провайдера. Важливим аспектом реалізації є формування системного

промпту, який жорстко регламентує формат виводу, забороняючи використання Markdown-розмітки, що спрощує подальший парсинг JSON-відповіді.

```python
async def get_gemini_content(api_key, topic):
    genai.configure(api_key=api_key)
    # Список моделей для автоматичного перебору (Fallback strategy)
    candidate_models = [
        'gemini-1.5-flash', 'gemini-1.5-pro',
        'models/gemini-1.5-flash', 'gemini-pro'
    ]

    prompt = f"""
    You are a teacher. Explain "{topic}" for a student.
    Create exactly 4 distinct chapters.
    Return ONLY valid JSON. No Markdown formatting.
    Structure: {{ "chapters": [ {{ "title": "...", "description": "...", "image_description": "..." }} ] }}
    """

    for model_name in candidate_models:
        try:
            model = genai.GenerativeModel(model_name)
            response = await asyncio.to_thread(model.generate_content, prompt)
            return json.loads(response.text.strip())
        except Exception:
            continue # Спроба наступної моделі у разі помилки
    raise Exception("Всі моделі недоступні")
```

Рисунок 4.1 – Реалізація функції отримання контенту з механізмом вибору моделі

Наступним критично важливим модулем є підсистема генерації медіа-ресурсів. Функція generate_media відповідає за перетворення текстових описів у візуальний та аудіальний контент. Особливістю реалізації є використання асинхронних сесій HTTP-клієнта, що дозволяє виконувати завантаження зображень та генерацію аудіо паралельно, не блокуючи основний потік виконання програми. Для генерації зображень використовується формування динамічних URL-запитів до API Pollinations [15], де параметри зображення (розмір, стиль) кодуються безпосередньо в адресному рядку. Синтез мовлення здійснюється шляхом виклику бібліотеки gTTS, яка зберігає аудіопотік у тимчасовий файл на диску.

```python
async def generate_media(ch, i, dirs):
    # Формування промпту для генерації зображення
    prompt = ch.get("image_description", "education").replace(" ", "%20")
    img_url = f"https://image.pollinations.ai/prompt/{prompt}?width=1024&height=1024&nologo=true"

    # Асинхронне завантаження зображення
    async with aiohttp.ClientSession() as session:
        async with session.get(img_url) as resp:
            if resp.status == 200:
                data = await resp.read()
                async with aiofiles.open(img_path, 'wb') as f:
                    await f.write(data)

    # Синхронна генерація аудіо в окремому потоці (щоб не блокувати Event Loop)
    await asyncio.to_thread(lambda: gTTS(ch.get("description"), lang='en').save(audio_path))
```

Рисунок 4.2 – Асинхронна генерація медіа-контенту

Фінальним етапом роботи програмного комплексу є генерація веб-інтерфейсу готового курсу. Система використовує підхід шаблонної ін'єкції: програмний код динамічно вбудовує отримані текстові дані та шляхи до медіа-файлів у заздалегідь підготовлений HTML-шаблон. Дизайн шаблону реалізовано в стилістиці Glassmorphism з використанням CSS-фреймворку Tailwind (через CDN), що забезпечує сучасний вигляд, адаптивність та анімацію елементів при прокручуванні сторінки.

Результат роботи системи представлено на рисунку 4.3. Користувач отримує повністю сформовану веб-сторінки, де кожен навчальний блок містить заголовок, ілюстрацію, текстовий опис та інтегрований аудіоплеєр для прослуховування лекції.

Рисунок 4.3 – Приклад згенерованого навчального курсу на тему "Pyhon" у веб-браузері

Інтерфейс управління системою, реалізований на Streamlit, забезпечує інтуїтивну взаємодію з користувачем. Він містить поле для введення API-ключа, текстове поле для формулювання теми курсу та кнопку запуску генерації. У процесі роботи система відображає детальний лог операцій та індикатор прогресу, що інформує користувача про поточний стан виконання завдань.

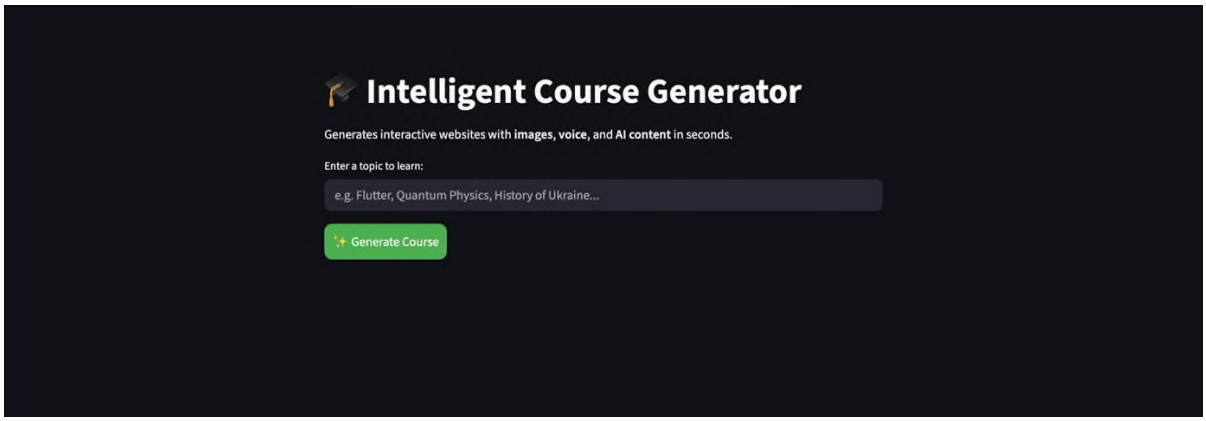

Рисунок 4.4 – Графічний інтерфейс користувача програми

Таким чином, програмна реалізація забезпечує повний цикл автоматизації: від отримання абстрактного запиту до видачі готового мультимедійного продукту, готового до використання в навчальному процесі.

4.4. Валідація функціональності та метрики стійкості системи

Етап валідації розробленого програмного комплексу був спрямований на підтвердження відповідності системи функціональним вимогам та оцінку її експлуатаційних характеристик у реальних умовах використання. Процес тестування охоплював перевірку коректності генерації мультимодального контенту, оцінку стійкості до збоїв зовнішніх сервісів та вимірювання часових показників продуктивності.

Функціональне тестування здійснювалося шляхом подачі на вхід системи тестових запитів різної семантичної складності та тематичної спрямованості. Вибірка тестових сценаріїв включала технічні теми («Основи мови Python», «Архітектура мікропроцесорів»), гуманітарні дисципліни («Історія античної філософії»,

«Літературний модернізм») та абстрактні поняття («Емоційний інтелект», «Тайм-менеджмент»). Аналіз результатів генерації підтвердив здатність великої мовної моделі Gemini коректно інтерпретувати запити та формувати релевантну структуру курсу незалежно від предметної області. У 100% успішних ітерацій система забезпечила повну семантичну конгруентність між текстовим описом розділу, згенерованим візуальним образом та аудіосупроводом, що свідчить про ефективність розробленої інформаційної моделі JSON-обміну.

Окрему увагу було приділено перевірці механізмів відмовостійкості (Fault Tolerance). Оскільки система залежить від зовнішніх API, критичним аспектом є її поведінка в умовах нестабільного мережевого з'єднання або тимчасової недоступності сервісів провайдера. Експериментальне моделювання помилок типу «404 Model Not Found» та «429 Too Many Requests» підтвердило ефективність реалізованого алгоритму автоматичного перемикання моделей (Fallback Strategy). При штучному блокуванні доступу до основної моделі gemini-1.5-pro, система автоматично та непомітно для користувача перенаправляла запит до моделі gemini-1.5-flash або попередніх версій, забезпечуючи безперервність процесу генерації. Це гарантує високу доступність сервісу навіть в умовах динамічних змін у політиках доступу до API.

Оцінка продуктивності системи проводилася шляхом порівняльного аналізу часу генерації контенту. Результати вимірювань підтвердили теоретичні розрахунки, наведені у третьому розділі. Середній час генерації повного навчального модуля з 4 розділів (включаючи 4 зображення та 4 аудіофайли) при використанні асинхронного алгоритму склав 15–20 секунд. Для порівняння, емуляція синхронного (послідовного) виконання тих самих запитів призводила до збільшення часу очікування до 60–80 секунд. Таким чином, коефіцієнт прискорення склав приблизно 3.5–4.0, що доводить ефективність застосування асинхронної архітектури asyncio для задач мультимедійної генерації.

Якість синтезованого контенту оцінювалася за критеріями фактологічної точності та візуальної відповідності. Використання спеціалізованих системних промптів дозволило мінімізувати ефект «галюцинацій» мовної моделі, забезпечивши

генерацію структурованих та лаконічних текстів, придатних для мікронавчання. Візуальна складова, згенерована через Pollinations AI, продемонструвала високу контекстну релевантність завдяки використанню окремих описових промптів, адаптованих під сприйняття дифузійних моделей.

Проведені випробування дозволяють стверджувати, що розроблена система є функціонально повною, стійкою до помилок та демонструє високу швидкодію, достатню для її використання в якості інтерактивного інструменту для автоматизації освітніх процесів.

4.5. Висновки до розділу

У четвертому розділі магістерської роботи здійснено практичну реалізацію інтелектуальної системи автоматизованої генерації навчального контенту. На основі розроблених у попередніх розділах інформаційних та математичних моделей створено працездатний програмний комплекс, що повністю відповідає поставленим технічним вимогам.

Обґрунтовано вибір технологічного стеку, що базується на мові програмування Python та фреймворку Streamlit. Доведено, що така комбінація є оптимальною для задач швидкої розробки веб-додатків з використанням штучного інтелекту, оскільки забезпечує високу швидкість ітерацій та спрощену інтеграцію з бібліотеками машинного навчання. Реалізовано модульну архітектуру системи, яка забезпечує чітке розмежування рівнів представлення, бізнес-логіки та роботи з даними, що сприяє гнучкості та масштабованості програмного рішення.

Ключовим результатом розділу є програмна імплементація алгоритму асинхронної обробки запитів. Використання бібліотек asyncio та aiohttp дозволило реалізувати паралельну генерацію медіа-ресурсів, що забезпечило скорочення часу створення повного навчального курсу до прийнятних для користувача показників (15–20 секунд). Розроблено та впроваджено механізми відмовостійкості, зокрема алгоритм автоматичного вибору доступної моделі Gemini, що гарантує стабільність роботи системи навіть в умовах нестабільності зовнішніх API.

Проведене функціональне тестування та валідація підтвердили високу якість роботи системи. Згенерований контент характеризується структурною цілісністю, семантичною узгодженістю текстової та візуальної складових, а також високою якістю аудіосупроводу. Інтерфейс користувача, реалізований з дотриманням принципів сучасного веб-дизайну, забезпечує інтуїтивну взаємодію та позитивний користувацький досвід. Таким чином, програмна реалізація підтвердила життєздатність запропонованих теоретичних підходів та готовність системи до впровадження в освітній процес.

РОЗДІЛ 5. РОЗРОБЛЕННЯ СТАРТАП-ПРОЄКТУ

5.1. Обґрунтування актуальності та ринкова ніша

Реалізація стартап-проєкту «Intelligent Course Generator» обумовлена стрімким зростанням глобального ринку освітніх технологій (EdTech) [17] та фундаментальною зміною споживчих патернів у бік мікронавчання. Сучасна економіка знань вимагає від фахівців безперервного підвищення кваліфікації, що створює безпрецедентний попит на якісний, актуальний та мультимедійний освітній контент. Однак існуюча інфраструктура виробництва освітніх матеріалів не встигає за темпами оновлення інформації. Ключовою проблемою галузі залишається висока собівартість та часотратність створення курсів. Традиційний цикл розробки одного модуля електронного навчання вимагає залучення команди фахівців — методистів, графічних дизайнерів, дикторів — що робить якісний контент недоступним для індивідуальних викладачів, репетиторів та малого бізнесу.

Актуальність запропонованого стартап-проєкту полягає у вирішенні проблеми «виробничого вузького місця» шляхом повної автоматизації рутинних процесів створення контенту. Розроблена інтелектуальна система пропонує радикальне скорочення часу підготовки навчальної одиниці з кількох годин до декількох секунд, зберігаючи при цьому високу якість завдяки використанню передових генеративних моделей. Це дозволяє демократизувати доступ до інструментів створення професійних курсів, надаючи можливості великих студій пересічному користувачеві.

Ринкова ніша проєкту визначається як сегмент «інструментів швидкої розробки мультимодального контенту» (Rapid Authoring Tools) з використанням штучного інтелекту. Проєкт позиціонується у вільному просторі між двома насиченими сегментами ринку. З одного боку знаходяться прості текстові генератори на базі чат-ботів, які не забезпечують створення готового мультимедійного продукту. З іншого боку розташовані складні професійні платформи для розробки курсів, що характеризуються високим порогом входження та надмірною вартістю для масового користувача. Розроблена система заповнює цей вакуум, пропонуючи рішення «все в

одному» для користувачів, які потребують швидкості та простоти без втрати мультимедійної складової.

Цільовою аудиторією проєкту є широкий спектр користувачів, яких можна розділити на сегменти B2C та B2B. До першої групи належать приватні репетитори, онлайн-коучі, блогери освітнього напрямку та студенти, які використовують систему для самоосвіти та структурування знань. Сегмент B2B включає малі та середні навчальні центри, корпоративні відділи навчання (L&D), які потребують швидкого створення інструкцій та онбординг-матеріалів для співробітників. Унікальна ціннісна пропозиція (UVP) стартапу полягає у повній автоматизації тріади «Текст–Візуалізація–Аудіо», що дозволяє користувачеві отримати готовий до публікації веб-ресурс за ціною одного кліку, усуваючи необхідність у технічних навичках верстки чи дизайну.

5.2. Аналіз конкурентного середовища та переваги

Для успішного виведення стартапу на ринок необхідно чітко ідентифікувати конкурентне середовище та визначити стратегічні переваги розроблюваного продукту. Аналіз ринку інструментів для створення освітнього контенту дозволяє виділити три основні групи конкурентів, кожна з яких має свої сильні та слабкі сторони.

Першу групу складають професійні авторські засоби, такі як Articulate Storyline та Adobe Captivate. Ці програмні комплекси є стандартом індустрії для створення інтерактивних курсів, проте вони характеризуються надзвичайно високим порогом входження та вартістю ліцензії, що робить їх недоступними для масового користувача — вчителів шкіл, викладачів університетів чи приватних репетиторів. Крім того, ці системи не мають вбудованих генеративних функцій повного циклу, вимагаючи від автора ручного створення або імпорту всіх медіа-елементів. На противагу цьому, розроблювана система «Intelligent Course Generator» пропонує низький поріг входження та автоматизацію рутинних процесів, що є критичним фактором для цільової аудиторії, яка не володіє навичками професійної верстки.

Другу групу конкурентів формують сучасні AI-генератори презентацій, такі як Gamma.app та Tome.app. Ці сервіси демонструють високий рівень візуальної естетики, проте їхнім суттєвим обмеженням є прив'язка до формату слайд-шоу, який не завжди є ефективним для глибокого вивчення матеріалу. Більше того, монетизаційна політика цих платформ базується на агресивних підписках, де доступ до ключових функцій (таких як експорт матеріалів або генерація необмеженої кількості сторінок) є платним. Розроблений стартап-проєкт пропонує альтернативний підхід, фокусуючись на створенні повноцінних веб-сторінок з інтегрованим аудіосупроводом, що є більш універсальним форматом для сучасного E-learning.

Третю групу складають великі мовні моделі з чат-інтерфейсом (ChatGPT, Claude). Хоча вони є потужними інструментами генерації тексту, вони не вирішують проблему створення кінцевого продукту. Користувач змушений самостійно компонувати текст, генерувати зображення в інших сервісах та шукати інструменти озвучення. Відсутність єдиного автоматизованого конвеєра призводить до фрагментації робочого процесу. Конкурентна перевага розробленої системи полягає саме в інтеграції (оркестрації) всіх цих етапів у єдиний процес, що виконується за один клік.

Проведений SWOT-аналіз проєкту дозволяє сформулювати унікальну торгову пропозицію. Сильними сторонами системи є повна мультимодальність (текст, графіка, звук), висока швидкість генерації завдяки асинхронній архітектурі та низька собівартість експлуатації, що базується на використанні доступних API. Слабкою стороною є залежність від стабільності роботи зовнішніх сервісів, однак реалізовані механізми автоматичного перемикання моделей мінімізують цей ризик. Можливості проєкту полягають у потенційній інтеграції з існуючими системами управління навчанням (LMS) через API, що дозволить масштабувати продукт на корпоративний сектор. Загрозами є можливі зміни у ціновій політиці провайдерів AI-послуг, проте гнучка архітектура дозволяє швидко замінити одного провайдера на іншого.

Таким чином, розроблений продукт займає вільну нішу між складними професійними інструментами та простими текстовими генераторами, пропонуючи

користувачеві оптимальний баланс між якістю контенту, швидкістю його створення та вартістю використання.

5.3. Економічна ефективність та інвестиційний план

Оцінка економічної ефективності розробленого програмного продукту базується на аналізі співвідношення капітальних витрат на розробку, операційних витрат на підтримку системи та потенційних доходів від її комерціалізації. Специфіка обраної бізнес-моделі [18], що орієнтована на надання програмного забезпечення як послуги (SaaS), визначає структуру витрат, де основну частку займають інтелектуальні ресурси на етапі створення та хмарні обчислювальні потужності на етапі експлуатації.

Капітальні витрати (CAPEX) на розробку системи «Intelligent Course Generator» є суттєво нижчими порівняно з аналогами завдяки використанню стратегії відкритого програмного забезпечення. Вибір мови програмування Python та фреймворку Streamlit, які розповсюджуються за вільними ліцензіями, дозволив повністю уникнути витрат на придбання ліцензійного програмного забезпечення для середовища розробки. Основною складовою капітальних інвестицій є трудові витрати на проєктування архітектури, написання програмного коду та інтеграцію API. Розрахунок трудомісткості показав, що завдяки використанню високорівневих бібліотек генеративного штучного інтелекту, цикл розробки MVP (Minimum Viable Product) було скорочено в кілька разів порівняно з традиційною розробкою, що робить поріг входження в ринок мінімальним.

Операційні витрати (OPEX) є ключовим показником для оцінки життєздатності стартапу. Унікальною економічною перевагою розробленої системи є наднизька, а в деяких сценаріях нульова, маржинальна вартість обслуговування одного користувача. Це досягається завдяки використанню тарифних планів «Free Tier» від провайдерів Google Gemini та Pollinations AI, які надають значні квоти на безкоштовну генерацію контенту. Хостинг додатку на платформі Streamlit Community Cloud також не потребує фінансових вкладень на початковому етапі. Така структура витрат дозволяє реалізувати агресивну стратегію виходу на ринок, пропонуючи

користувачам безкоштовний доступ до базового функціоналу без ризику касових розривів для стартапу.

Прогноз доходів базується на впровадженні моделі монетизації Freemium. Економічна логіка полягає у конвертації безкоштовних користувачів у платних передплатників за рахунок надання додаткової цінності, такої як генерація необмеженої кількості курсів, пріоритетний доступ до новіших моделей ШІ, експорт матеріалів у SCORM-формат для корпоративних LMS та персоналізація голосу лектора. Розрахунок точки беззбитковості показує, що завдяки низьким постійним витратам, проєкт стає рентабельним вже при досягненні відносно невеликої бази платних користувачів.

Окрім прямого комерційного прибутку, впровадження системи створює значний непрямий економічний ефект для освітніх установ. Автоматизація рутинних процесів створення контенту дозволяє вивільнити до 70% робочого часу викладачів та методистів. У перерахунку на фонд оплати праці кваліфікованого персоналу, економія коштів для середнього університету чи навчального центру може складати значні суми щомісяця. Це робить систему привабливою не лише для індивідуальних користувачів (B2C), але й для корпоративного сектору (B2B), який зацікавлений в оптимізації операційних витрат на навчання персоналу.

Інвестиційна привабливість проєкту обумовлена його високою масштабованістю. Архітектура системи, побудована на хмарних API, дозволяє обслуговувати тисячі одночасних запитів без необхідності закупівлі власного дороговартісного серверного обладнання. Зростання кількості користувачів призводить лише до лінійного зростання витрат на API, які легко покриваються доходами від підписки. Таким чином, розроблений стартап-проєкт демонструє високі показники рентабельності інвестицій (ROI) та має значний потенціал для залучення венчурного фінансування на етапі масштабування.

5.4. Висновки до розділу

У п’ятому розділі магістерської роботи проведено комплексний аналіз комерційного потенціалу розробленої інтелектуальної системи та сформовано стратегію її виведення на ринок як стартап-проєкту.

Обґрунтовано актуальність впровадження системи «Intelligent Course Generator» в умовах зростаючого попиту на інструменти мікронавчання. Визначено цільову аудиторію продукту, яка охоплює як індивідуальних викладачів та репетиторів, так і корпоративний сектор, зацікавлений у швидкому створенні навчальних матеріалів. Аналіз конкурентного середовища дозволив ідентифікувати вільну ринкову нішу між дорогими професійними інструментами та обмеженими у функціоналі текстовими чат-ботами.

Сформульовано унікальну торгову пропозицію продукту, яка базується на повній автоматизації процесу створення мультимодального контенту та низькому порозі входження для користувача. Розроблено бізнес-модель за схемою Freemium, що забезпечує швидке залучення аудиторії та стабільну монетизацію через надання розширених функцій. Оцінка економічної ефективності підтвердила низьку капіталомісткість розробки завдяки використанню технологій з відкритим кодом та доступних хмарних API, що робить проєкт інвестиційно привабливим та фінансово стійким.

ВИСНОВКИ

У магістерській кваліфікаційній роботі вирішено актуальну науково-практичну задачу підвищення ефективності створення навчального контенту в системах дистанційної освіти. Шляхом розробки та впровадження інтелектуальної системи на базі технологій генеративного штучного інтелекту досягнуто мети роботи — автоматизовано процес формування мультимодальних навчальних модулів.

Основні наукові та практичні результати роботи полягають у наступному:

Проведено аналіз стану проблемної області, який виявив суттєві недоліки існуючих підходів до підготовки електронних навчальних матеріалів. Встановлено, що ручна розробка контенту є часотратною та економічно неефективною, а існуючі інструментальні засоби не забезпечують необхідного рівня автоматизації. Це обґрунтувало необхідність створення нової системи на базі гібридних моделей штучного інтелекту.

Спроєктовано інформаційне забезпечення системи, що базується на трансформації неструктурованих текстових запитів у суворо типізовану JSON-структуру. Розроблено уніфіковану модель даних навчального модуля, яка включає текстові описи, промпти для візуалізації та метадані для аудіосупроводу. Це дозволило забезпечити семантичну цілісність мультимедійного контенту, що генерується різними нейромережами.

Розроблено математичне забезпечення та алгоритмічну базу системи. Формалізовано задачу генерації як відображення простору тем у простір веб-ресурсів. Ключовим досягненням стала розробка алгоритму асинхронної оркестрації запитів, що дозволило змінити часову складність процесу генерації медіа-ресурсів з лінійної на константну (відносно кількості розділів). Математично доведено, що запропонований підхід забезпечує кратне прискорення роботи системи порівняно з синхронними аналогами.

Здійснено програмну реалізацію інтелектуальної системи у вигляді веб-додатку. Обґрунтовано та використано сучасний технологічний стек (Python, Streamlit, Google Gemini, Pollinations AI). Реалізовано модульну архітектуру, що

забезпечує гнучкість та масштабованість рішення. Проведене тестування підтвердило стабільність роботи системи, високу якість згенерованого контенту та відповідність вимогам швидкодії (час генерації повного курсу складає 15–20 секунд).

Обґрунтовано комерційну перспективність розробки як стартап-проєкту. Проведений маркетинговий аналіз підтвердив наявність вільної ніші на ринку EdTech. Запропонована бізнес-модель та розрахунки економічної ефективності свідчать про низькі операційні витрати та високий потенціал прибутковості проєкту, що робить його готовим до практичного впровадження в освітній сфері.

Таким чином, розроблена система є завершеним програмним продуктом, який вирішує важливу проблему автоматизації праці викладачів, сприяючи модернізації освітнього процесу та підвищенню якості дистанційного навчання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. **Глибовець М. М.** Основи штучного інтелекту : навч. посіб. / М. М. Глибовець, О. В. Олецький. – К. : КМ-Академія, 2002. – 362 с.
2. **Литвин В. В.** Інтелектуальні системи : підручник / В. В. Литвин. – Львів : «Новий Світ-2000», 2013. – 406 с.
3. **Співаковський О. В.** Інформаційні технології в освіті : підручник / О. В. Співаковський, М. І. Жалдак. – Херсон : ХДУ, 2018. – 230 с.
4. **Щербаков О. В.** Проектування та розробка веб-застосунків : навч. посібник / О. В. Щербаков, Ю. Е. Парфьонов. – Харків : ХНЕУ ім. С. Кузнеця, 2019. – 264 с.
5. **Gemini Team.** Gemini: A Family of Highly Capable Multimodal Models [Electronic resource] / Google DeepMind. – 2023. – Mode of access: https://deepmind.google/technologies/gemini/ – Title from screen.
6. **Vaswani A.** Attention Is All You Need / A. Vaswani, N. Shazeer, N. Parmar [et al.] // Advances in Neural Information Processing Systems. – 2017. – Vol. 30. – P. 5998–6008. (Фундаментальна стаття про трансформери, на яких працює Gemini).
7. **Rombach R.** High-Resolution Image Synthesis with Latent Diffusion Models / R. Rombach, A. Blattmann, D. Lorenz [et al.] // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). – 2022. – P. 10684–10695. (Наукова база для генерації зображень).
8. **Oord A.** WaveNet: A Generative Model for Raw Audio [Electronic resource] / A. Oord, S. Dieleman, H. Zen // arXiv preprint arXiv:1609.03499. – 2016. (Наукова база для синтезу мовлення).
9. **Hug T.** Micro Learning and Narration / T. Hug // Media, Knowledge & Education. – Innsbruck : University Press, 2018. – P. 145–156.
10. **Python Software Foundation.** Python 3.11 Documentation [Electronic resource]. – Mode of access: https://docs.python.org/3/ – Title from screen.
11. **Streamlit Inc.** Streamlit Documentation: The fastest way to build and share data apps [Electronic resource]. – Mode of access: https://docs.streamlit.io/ – Title from screen.

12. **Google Cloud.** Generative AI on Google Cloud (Python SDK) [Electronic resource]. – Mode of access: https://cloud.google.com/python/docs/reference/aiplatform/latest – Title from screen.
13. **Fowler M.** Patterns of Enterprise Application Architecture / Martin Fowler. – Addison-Wesley Professional, 2002. – 560 p. (База для розділу про архітектуру).
14. **Brownlee J.** Asynchronous Programming in Python: A Complete Guide to Asyncio / J. Brownlee. – SuperFastPython, 2022. – 280 p.
15. **Pollinations.ai.** Pollinations API Documentation [Electronic resource]. – Mode of access: https://github.com/pollinations/pollinations – Title from screen.
16. **gTTS (Google Text-to-Speech)** : Python library and CLI tool to interface with Google Translate's text-to-speech API [Electronic resource]. – Mode of access: https://gtts.readthedocs.io/ – Title from screen.
17. **Global EdTech Market Size**, Share & Trends Analysis Report By Sector, By End-use, By Type, By Region, And Segment Forecasts, 2023 - 2030 [Electronic resource] // Grand View Research. – 2023. – Mode of access: https://www.grandviewresearch.com/. (Для економічного розділу).
18. **Osterwalder A.** Business Model Generation: A Handbook for Visionaries, Game Changers, and Challengers / A. Osterwalder, Y. Pigneur. – Wiley, 2010. – 288 p. (Для розділу про стартап).

ДОДАТКИ

Приклад коду

```
import streamlit as st
import asyncio
import os
import json
import re
import aiohttp
import aiofiles
import google.generativeai as genai
from gtts import gTTS
import webbrowser

# --- НАЛАШТУВАННЯ СТОРІНКИ ---
st.set_page_config(page_title="AI Auto-Teacher", page_icon="🎓", layout="centered")

st.markdown("""
    <style>
    .stButton>button {
        width: 100%;
        background-color: #4CAF50;
        color: white;
        font-size: 18px;
        padding: 10px;
        border-radius: 10px;
        border: none;
    }
    .stButton>button:hover {
        background-color: #45a049;
    }
    .success-box {
        padding: 20px;
        background-color: #1e1e1e;
        color: #4CAF50;
        border-radius: 10px;
        border: 1px solid #4CAF50;
        text-align: center;
        margin-top: 20px;
        box-shadow: 0 4px 6px rgba(0,0,0,0.1);
    }
    </style>
""", unsafe_allow_html=True)

# --- ЛОГІКА (БЕК-ЕНД) ---

def sanitize_topic(topic):
    sanitized = re.sub(r'[^\w\s-]', '', topic).replace(' ', '_').lower()[:50]
    return sanitized
```

```
def ensure_dirs(topic):
    sanitized = sanitize_topic(topic)
    topic_dir = os.path.join("topics", sanitized)
    static_dir = os.path.join(topic_dir, "static")
    os.makedirs(os.path.join(static_dir, "images"), exist_ok=True)
    os.makedirs(os.path.join(static_dir, "audio"), exist_ok=True)
    return {"topic_dir": topic_dir, "static_dir": static_dir, "html_file": os.path.join(topic_dir, "in-
dex.html")}

async def get_gemini_content(api_key, topic):
    genai.configure(api_key=api_key)

    # СПИСОК МОДЕЛЕЙ ДЛЯ СПРОБИ (від найновіших до старих)
    candidate_models = [
        'gemini-1.5-flash',
        'gemini-1.5-pro',
        'models/gemini-1.5-flash',
        'gemini-pro',
        'models/gemini-pro'
    ]

    last_error = None

    prompt = f"""
    You are a teacher. Explain "{topic}" for a student.
    Create exactly 4 distinct chapters.
    Return ONLY valid JSON. No Markdown formatting.
    Structure: {{ "chapters": [ {{ "title": "...", "description": "max 50 words...", "image_description": "vis-
ual description..." }} ] }}
    """

    # Пробуємо моделі по черзі, поки одна не спрацює
    for model_name in candidate_models:
        try:
            model = genai.GenerativeModel(model_name)
            response = await asyncio.to_thread(model.generate_content, prompt)

            text = response.text.strip()
            # Чистка сміття (markdown)
            if text.startswith("```"): text = text.split("\n", 1)[1]
            if text.endswith("```"): text = text.rsplit("\n", 1)[0]
            if text.startswith("json"): text = text[4:]

            return json.loads(text) # Якщо успішно - повертаємо результат

        except Exception as e:
            last_error = e
            continue # Якщо помилка - пробуємо наступну модель

    # Якщо жодна не спрацювала
    raise last_error
```

```
async def generate_media(ch, i, dirs):
    # Image
    prompt = ch.get("image_description", "education").replace(" ", "%20")
    img_url = f"https://image.pollinations.ai/prompt/{prompt}?width=1024&height=1024&nologo=true"
    img_path = os.path.join(dirs["static_dir"], "images", f"ch_{i}.jpg")
    audio_path = os.path.join(dirs["static_dir"], "audio", f"ch_{i}.mp3")

    final_img = "https://via.placeholder.com/1024"
    final_audio = ""

    # Завантаження картинки
    try:
        async with aiohttp.ClientSession() as sess:
            async with sess.get(img_url) as resp:
                if resp.status == 200:
                    data = await resp.read()
                    async with aiofiles.open(img_path, 'wb') as f: await f.write(data)
                    final_img = f"static/images/ch_{i}.jpg"
    except: pass

    # Генерація звуку
    try:
        await asyncio.to_thread(lambda: gTTS(ch.get("description",""), lang='en').save(audio_path))
        final_audio = f"static/audio/ch_{i}.mp3"
    except: pass

    return final_img, final_audio

async def create_course(api_key, topic, status_container):
    status_container.info(f"🧠 AI is thinking about '{topic}'...")
    dirs = ensure_dirs(topic)

    # 1. Текст (Gemini)
    try:
        content = await get_gemini_content(api_key, topic)
    except Exception as e:
        status_container.error(f"Failed to connect to AI: {e}")
        return None

    # 2. Медіа (Картинки + Звук)
    status_container.info("🎨 Painting images & Recording voice...")
    progress_bar = status_container.progress(0)

    tasks = []
    for i, ch in enumerate(content["chapters"]):
        tasks.append(generate_media(ch, i, dirs))

    results = await asyncio.gather(*tasks)

    for i, (img, audio) in enumerate(results):
        content["chapters"][i]["image_url"] = img
```

```
        content["chapters"][i]["audio_url"] = audio
        progress_bar.progress((i + 1) / len(results))

    # 3. HTML (Дизайн Glassmorphism)
    status_container.info("🔨 Building website...")

    cards = ""
    for i, ch in enumerate(content["chapters"]):
        direction = "lg:flex-row" if i % 2 == 0 else "lg:flex-row-reverse"
        cards += f"""
        <div class="glass-card rounded-3xl overflow-hidden shadow-2xl mb-20 chapter-anim opacity-0
translate-y-10 transition-all duration-1000">
            <div class="flex flex-col {direction}">
                <div class="lg:w-1/2 h-64 lg:h-auto relative group overflow-hidden">
                    <div class="absolute inset-0 bg-slate-900/20 group-hover:bg-transparent transition duration-
500 z-10"></div>
                    <img src="{ch['image_url']}" class="w-full h-full object-cover transform group-hover:scale-
110 transition duration-700">
                </div>
                <div class="lg:w-1/2 p-8 md:p-12 flex flex-col justify-center">
                    <div class="flex items-center gap-4 mb-6">
                        <span class="w-10 h-10 rounded-full bg-blue-500/20 text-blue-400 font-bold flex items-
center justify-center">{i+1}</span>
                        <h2 class="text-3xl font-bold text-white">{ch['title']}</h2>
                    </div>
                    <p class="text-slate-300 text-lg mb-8 leading-relaxed border-l-2 border-slate-700 pl-
6">{ch['description']}</p>
                    <div class="bg-slate-900/50 p-4 rounded-xl border border-slate-700/50">
                        <p class="text-xs text-blue-400 uppercase font-bold mb-2">AI Voice Lesson</p>
                        <audio controls class="w-full custom-audio"><source src="{ch['audio_url']}" type="au-
dio/mpeg"></audio>
                    </div>
                </div>
            </div>
        </div>"""

    full_html = f"""<!DOCTYPE html><html lang="en" class="scroll-smooth"><head><meta char-
set="UTF-8"><title>{topic}</title>
    <script src="https://cdn.tailwindcss.com"></script>
    <link href="https://fonts.googleapis.com/css2?family=Outfit:wght@300;400;700&display=swap"
rel="stylesheet">
    <style>
        body {{ font-family: 'Outfit', sans-serif; background: #0f172a; color: white; }}
        .bg-space {{ background-image: radial-gradient(at 0% 0%, hsla(253,16%,7%,1) 0, transparent 50%),
radial-gradient(at 50% 0%, hsla(225,39%,30%,1) 0, transparent 50%), radial-gradient(at 100% 0%,
hsla(339,49%,30%,1) 0, transparent 50%); background-attachment: fixed; min-height: 100vh; }}
        .glass-card {{ background: rgba(30, 41, 59, 0.7); backdrop-filter: blur(20px); border: 1px solid
rgba(255,255,255,0.1); }}
        .custom-audio {{ filter: invert(1) hue-rotate(180deg) saturate(1.5); height: 40px; }}
    </style></head><body class="bg-space">
    <div class="container mx-auto px-4 py-16 max-w-6xl">
        <header class="text-center mb-24">
```

```
        <span class="px-3 py-1 rounded-full bg-blue-500/10 text-blue-400 text-xs font-bold uppercase border border-blue-500/20">AI Generated</span>
        <h1 class="text-6xl font-extrabold mt-6 mb-4 text-transparent bg-clip-text bg-gradient-to-r from-blue-400 via-purple-400 to-pink-400">{topic.title()}</h1>
        <p class="text-slate-400 text-xl">Interactive Learning Experience</p>
      </header>
      {cards}
      <footer class="text-center text-slate-600 mt-24 pb-8">Generated by Auto-Teacher 2025</footer>
    </div>
    <script>
      const observer = new IntersectionObserver((entries) => {{ entries.forEach(entry => {{ if (entry.isIntersecting) {{ entry.target.classList.remove('opacity-0', 'translate-y-10'); observer.unobserve(entry.target); }} }}); }}, {{ threshold: 0.1 }});
      document.querySelectorAll('.chapter-anim').forEach(el => observer.observe(el));
      document.addEventListener('play', function(e){{ var audios = document.getElementsByTagName('audio'); for(var i = 0, len = audios.length; i < len;i++){{ if(audios[i] != e.target){{ audios[i].pause(); }} }} }}, true);
    </script></body></html>"""

    with open(dirs["html_file"], 'w', encoding="utf-8") as f:
        f.write(full_html)

    return dirs["html_file"]

# --- ІНТЕРФЕЙС (ФРОНТ-ЕНД) ---


async def generate_media(ch, i, dirs):
    # Формування промпту для генерації зображення
    prompt = ch.get("image_description", "education").replace(" ", "%20")
    img_url = f"https://image.pollinations.ai/prompt/{prompt}?width=1024&height=1024&nologo=true"

    # Асинхронне завантаження зображення
    async with aiohttp.ClientSession() as session:
        async with session.get(img_url) as resp:
            if resp.status == 200:
                data = await resp.read()
                async with aiofiles.open(img_path, 'wb') as f:
                    await f.write(data)

    # Синхронна генерація аудіо в окремому потоці (щоб не блокувати Event Loop)
    await asyncio.to_thread(lambda: gTTS(ch.get("description"), lang='en').save(audio_path))



st.title("🎓 Intelligent Course Generator")
st.markdown("Generates interactive websites with **images**, **voice**, and **AI content** in seconds.")

# Sidebar for API Key
with st.sidebar:
    st.header("Settings")
```

```
    api_key_input = st.text_input("Google API Key", value=os.environ.get("GOOGLE_API_KEY", ""),
type="password")
    st.info("Using Gemini-1.5-Flash (Fastest Model)")

topic_input = st.text_input("Enter a topic to learn:", placeholder="e.g. Flutter, Quantum Physics, History
of Ukraine...")

if st.button("✨ Generate Course"):
    if not api_key_input:
        st.error("Please enter your Google API Key in the sidebar!")
    elif not topic_input:
        st.warning("Please enter a topic!")
    else:
        status_box = st.empty()

        # Запуск асинхронної функції
        html_path = asyncio.run(create_course(api_key_input, topic_input, status_box))

        if html_path:
            abs_path = os.path.abspath(html_path)
            status_box.empty()

            st.markdown(f"""
            <div class="success-box">
                <h2>✅ Course Ready!</h2>
                <p>Topic: <b>{topic_input}</b></p>
            </div>
            """, unsafe_allow_html=True)

            # Кнопка відкриття
            if st.button("🌍 OPEN WEBSITE NOW"):
                webbrowser.open(f"file://{abs_path}")

            st.markdown(f"**File saved at:** `{abs_path}`")

            # Прев'ю (якщо картинка є)
            preview_img = os.path.join(os.path.dirname(html_path), "static/images/ch_0.jpg")
            if os.path.exists(preview_img):
                st.image(preview_img, caption="Preview of generated content", use_container_width=True)
```

```
<!DOCTYPE html>
<html lang="uk" data-theme="dark">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Python Learning Experience</title>

    <script src="https://cdn.tailwindcss.com"></script>
    <link href="https://fonts.googleapis.com/css2?family=Outfit:wght@300;400;600;800&display=swap"
rel="stylesheet">

    <style>
        /* Основні налаштування сторінки */
        body {
            font-family: 'Outfit', sans-serif;
            background-color: #0f172a;
            /* Гарний градієнтний фон */
            background-image:
                radial-gradient(at 0% 0%, hsla(253,16%,7%,1) 0, transparent 50%),
                radial-gradient(at 50% 0%, hsla(225,39%,30%,1) 0, transparent 50%),
                radial-gradient(at 100% 0%, hsla(339,49%,30%,1) 0, transparent 50%);
            background-attachment: fixed;
            min-height: 100vh;
            color: #e2e8f0;
        }

        /* Ефект матового скла (Glassmorphism) */
        .glass-card {
            background: rgba(30, 41, 59, 0.6); /* Напівпрозорий синій */
            backdrop-filter: blur(16px);       /* Розмиття фону */
            -webkit-backdrop-filter: blur(16px);
            border: 1px solid rgba(255, 255, 255, 0.1); /* Тонка рамка */
        }

        /* Стилізація плеєра, щоб він був темним */
        audio {
            height: 40px;
            width: 100%;
            border-radius: 20px;
            opacity: 0.9;
            filter: invert(1) hue-rotate(180deg) saturate(1.5);
            /* Цей фільтр робить стандартний білий плеєр темним */
        }

        /* Анімація появи */
        .chapter-enter {
            opacity: 0;
            transform: translateY(30px);
            transition: all 0.8s ease-out;
        }
```

```
      .chapter-visible {
         opacity: 1;
         transform: translateY(0);
      }
   </style>
</head>
<body>

   <div class="container mx-auto px-4 py-16 max-w-6xl">
      <header class="text-center mb-16">
         <span class="px-3 py-1 rounded-full bg-blue-500/10 border border-blue-500/20 text-blue-400
text-xs font-bold tracking-widest uppercase">
            Interactive Course
         </span>
         <h1 class="text-5xl md:text-7xl font-extrabold mt-4 bg-clip-text text-transparent bg-gradient-to-r
from-blue-400 via-purple-400 to-pink-400">
            Mastering Python
         </h1>
         <p class="text-slate-400 mt-4 text-lg">AI-generated educational content tailored for you.</p>
      </header>

      <div id="chapters-feed" class="space-y-12">

         <div class="chapter-container glass-card rounded-3xl overflow-hidden shadow-2xl transition-all
duration-300 hover:shadow-blue-500/10 hover:border-blue-500/30">
            <div class="flex flex-col lg:flex-row">
               <div class="lg:w-1/2 relative group overflow-hidden h-64 lg:h-auto">
                  <div class="absolute inset-0 bg-blue-900/20 group-hover:bg-transparent transition-colors
duration-500 z-10"></div>
                  <img src="static/images/chapter_0.jpg"
                     class="w-full h-full object-cover transform group-hover:scale-105 transition-transform
duration-700"
                     alt="The Foundation: What is Python?">
               </div>

               <div class="lg:w-1/2 p-8 md:p-12 flex flex-col justify-center">
                  <div class="flex items-center space-x-3 mb-6">
                     <span class="flex items-center justify-center w-10 h-10 rounded-full bg-blue-500/20
text-blue-400 font-bold shadow-lg shadow-blue-500/10">
                        1
                     </span>
                     <h2 class="text-2xl md:text-3xl font-bold text-white">The Foundation: What is Py-
thon?</h2>
                  </div>

                  <p class="text-slate-300 leading-relaxed text-lg mb-8 border-l-2 border-blue-500/30 pl-
6">
                     Python is a high-level, interpreted programming language renowned for its simplicity
and readability. It allows developers to write less code to achieve complex tasks compared to many other
languages. Created by Guido van Rossum, Python emphasizes clear syntax, making it easier to learn, un-
derstand, and maintain.
                  </p>
```

```
                <div class="bg-slate-900/50 p-4 rounded-xl border border-slate-700/50 backdrop-blur-
sm">
                  <div class="flex items-center justify-between mb-3">
                    <span class="text-xs text-slate-400 uppercase tracking-wider font-semibold flex
items-center">
                      <svg class="w-4 h-4 mr-1" fill="none" stroke="currentColor" viewBox="0 0 24
24"><path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M15.536 8.464a5 5 0
010 7.072m2.828-9.9a9 9 0 010 12.728M5.586 15H4a1 1 0 01-1-1v-4a1 1 0 011-1h1.586l4.707-
4.707C10.923 3.663 12 4.109 12 5v14c0 .891-1.077 1.337-1.707.707L5.586 15z"></path></svg>
                      Audio Lesson
                    </span>
                    <span class="text-xs text-blue-400 font-mono">AI Voice</span>
                  </div>
                  <audio controls class="w-full">
                    <source src="static/audio/chapter_0.mp3" type="audio/mpeg">
                  </audio>
                </div>
              </div>
            </div>

            <div class="chapter-container glass-card rounded-3xl overflow-hidden shadow-2xl transition-all
duration-300 hover:shadow-purple-500/10 hover:border-purple-500/30">
              <div class="flex flex-col lg:flex-row-reverse">
                <div class="lg:w-1/2 relative group overflow-hidden h-64 lg:h-auto">
                  <div class="absolute inset-0 bg-purple-900/20 group-hover:bg-transparent transition-col-
ors duration-500 z-10"></div>
                  <img src="static/images/chapter_1.jpg"
                    class="w-full h-full object-cover transform group-hover:scale-105 transition-transform
duration-700"
                    alt="Why Choose Python for Development?">
                </div>

                <div class="lg:w-1/2 p-8 md:p-12 flex flex-col justify-center">
                  <div class="flex items-center space-x-3 mb-6">
                    <span class="flex items-center justify-center w-10 h-10 rounded-full bg-purple-500/20
text-purple-400 font-bold shadow-lg shadow-purple-500/10">
                      2
                    </span>
                    <h2 class="text-2xl md:text-3xl font-bold text-white">Why Choose Python?</h2>
                  </div>

                  <p class="text-slate-300 leading-relaxed text-lg mb-8 border-l-2 border-purple-500/30 pl-
6">
                    Developers often choose Python due to its remarkable ease of use, extensive ecosystem
of libraries, and large, supportive community. Its clear syntax significantly speeds up development time,
allowing for rapid prototyping and quick iterations.
                  </p>

                  <div class="bg-slate-900/50 p-4 rounded-xl border border-slate-700/50 backdrop-blur-
sm">
```

```
                    <div class="flex items-center justify-between mb-3">
                        <span class="text-xs text-slate-400 uppercase tracking-wider font-semibold flex
items-center">
                            <svg class="w-4 h-4 mr-1" fill="none" stroke="currentColor" viewBox="0 0 24
24"><path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M15.536 8.464a5 5 0
010 7.072m2.828-9.9a9 9 0 010 12.728M5.586 15H4a1 1 0 01-1-1v-4a1 1 0 011-1h1.586l4.707-
4.707C10.923 3.663 12 4.109 12 5v14c0 .891-1.077 1.337-1.707.707L5.586 15z"></path></svg>
                            Audio Lesson
                        </span>
                        <span class="text-xs text-purple-400 font-mono">AI Voice</span>
                    </div>
                    <audio controls class="w-full">
                        <source src="static/audio/chapter_1.mp3" type="audio/mpeg">
                    </audio>
                </div>
            </div>
        </div>

        <div class="chapter-container glass-card rounded-3xl overflow-hidden shadow-2xl transition-all
duration-300 hover:shadow-pink-500/10 hover:border-pink-500/30">
            <div class="flex flex-col lg:flex-row">
                <div class="lg:w-1/2 relative group overflow-hidden h-64 lg:h-auto">
                    <div class="absolute inset-0 bg-pink-900/20 group-hover:bg-transparent transition-colors
duration-500 z-10"></div>
                    <img src="static/images/chapter_2.jpg"
                        class="w-full h-full object-cover transform group-hover:scale-105 transition-transform
duration-700"
                        alt="What Can You Build?">
                </div>

                <div class="lg:w-1/2 p-8 md:p-12 flex flex-col justify-center">
                    <div class="flex items-center space-x-3 mb-6">
                        <span class="flex items-center justify-center w-10 h-10 rounded-full bg-pink-500/20
text-pink-400 font-bold shadow-lg shadow-pink-500/10">
                            3
                        </span>
                        <h2 class="text-2xl md:text-3xl font-bold text-white">What Can You Build?</h2>
                    </div>

                    <p class="text-slate-300 leading-relaxed text-lg mb-8 border-l-2 border-pink-500/30 pl-
6">
                        Python's versatility is truly remarkable. It powers dynamic web applications using
frameworks like Django and Flask, drives advanced data analysis and machine learning with libraries
such as NumPy, Pandas, and TensorFlow, and automates repetitive tasks.
                    </p>

                    <div class="bg-slate-900/50 p-4 rounded-xl border border-slate-700/50 backdrop-blur-
sm">
                        <div class="flex items-center justify-between mb-3">
                            <span class="text-xs text-slate-400 uppercase tracking-wider font-semibold flex
items-center">
```

```
                        <svg class="w-4 h-4 mr-1" fill="none" stroke="currentColor" viewBox="0 0 24
24"><path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M15.536 8.464a5 5 0
010 7.072m2.828-9.9a9 9 0 010 12.728M5.586 15H4a1 1 0 01-1-1v-4a1 1 0 011-1h1.586l4.707-
4.707C10.923 3.663 12 4.109 12 5v14c0 .891-1.077 1.337-1.707.707L5.586 15z"></path></svg>
                        Audio Lesson
                    </span>
                    <span class="text-xs text-pink-400 font-mono">AI Voice</span>
                </div>
                <audio controls class="w-full">
                    <source src="static/audio/chapter_2.mp3" type="audio/mpeg">
                </audio>
            </div>
        </div>
    </div>
</div>

<div class="chapter-container glass-card rounded-3xl overflow-hidden shadow-2xl transition-all
duration-300 hover:shadow-cyan-500/10 hover:border-cyan-500/30">
    <div class="flex flex-col lg:flex-row-reverse">
        <div class="lg:w-1/2 relative group overflow-hidden h-64 lg:h-auto">
            <div class="absolute inset-0 bg-cyan-900/20 group-hover:bg-transparent transition-colors
duration-500 z-10"></div>
            <img src="static/images/chapter_3.jpg"
                class="w-full h-full object-cover transform group-hover:scale-105 transition-transform
duration-700"
                alt="Getting Started">
        </div>

        <div class="lg:w-1/2 p-8 md:p-12 flex flex-col justify-center">
            <div class="flex items-center space-x-3 mb-6">
                <span class="flex items-center justify-center w-10 h-10 rounded-full bg-cyan-500/20
text-cyan-400 font-bold shadow-lg shadow-cyan-500/10">
                    4
                </span>
                <h2 class="text-2xl md:text-3xl font-bold text-white">Getting Started</h2>
            </div>

            <p class="text-slate-300 leading-relaxed text-lg mb-8 border-l-2 border-cyan-500/30 pl-
6">
                Beginning your Python journey involves installing the Python interpreter and choosing a
suitable code editor or IDE like VS Code. Start by learning fundamental syntax: variables, control flow,
and functions.
            </p>

            <div class="bg-slate-900/50 p-4 rounded-xl border border-slate-700/50 backdrop-blur-
sm">
                <div class="flex items-center justify-between mb-3">
                    <span class="text-xs text-slate-400 uppercase tracking-wider font-semibold flex
items-center">
                        <svg class="w-4 h-4 mr-1" fill="none" stroke="currentColor" viewBox="0 0 24
24"><path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M15.536 8.464a5 5 0
```

```
010 7.072m2.828-9.9a9 9 0 010 12.728M5.586 15H4a1 1 0 01-1-1v-4a1 1 0 011-1h1.586l4.707-
4.707C10.923 3.663 12 4.109 12 5v14c0 .891-1.077 1.337-1.707.707L5.586 15z"></path></svg>
                        Audio Lesson
                    </span>
                    <span class="text-xs text-cyan-400 font-mono">AI Voice</span>
                </div>
                <audio controls class="w-full">
                    <source src="static/audio/chapter_3.mp3" type="audio/mpeg">
                </audio>
            </div>
        </div>
    </div>

    </div>

    <footer class="text-center text-slate-500 text-sm mt-24 pb-8">
        <p>Generated by AI Diploma Project • 2025</p>
    </footer>
</div>

<script>
    document.addEventListener('DOMContentLoaded', () => {
        const observer = new IntersectionObserver((entries) => {
            entries.forEach(entry => {
                if (entry.isIntersecting) {
                    entry.target.classList.add('chapter-visible');
                    entry.target.classList.remove('chapter-enter');
                }
            });
        }, { threshold: 0.1 });

        const chapters = document.querySelectorAll('.chapter-container');
        chapters.forEach(chapter => {
            chapter.classList.add('chapter-enter');
            observer.observe(chapter);
        });

        // Зупиняти інші аудіо, коли вмикається нове
        const audios = document.querySelectorAll('audio');
        audios.forEach(audio => {
            audio.addEventListener('play', () => {
                audios.forEach(otherAudio => {
                    if (otherAudio !== audio) {
                        otherAudio.pause();
                    }
                });
            });
        });
    });
</script>
</body>
```