МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інтелектуальних
інформаційних систем
_____Юрій КОНДРАТЕНКО
«____»_____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

ІНТЕЛЕКТУАЛЬНА СИСТЕМА СТВОРЕННЯ ОСВІТНІХ МАТЕРІАЛІВ НА ОСНОВІ ГШІ

Спеціальність 122 Комп'ютерні науки

Освітня програма «Комп'ютерні науки»

Здобувач _____ Нікіта МОСКАЛЕНКО

«___»_____ 2025 р.

Керівник канд. техн. наук, доцент _____ Євген СІДЕНКО

«___»_____ 2025 р.

Миколаїв – 2025

Чорноморський національний університет імені Петра Могили

(повне найменування закладу вищої освіти)

Факультет	Комп’ютерних наук
Кафедра	Інтелектуальних інформаційних систем
Рівень вищої освіти	Перший (бакалаврський)
Освітній ступень	Бакалавр
Спеціальність	122 Комп’ютерні науки
Освітня програма	Комп’ютерні науки

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем

_____ Юрій КОНДРАТЕНКО

«____» _____ 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу здобувача

Москаленка Нікіти Віталійовича

(прізвище, ім’я, по батькові здобувача)

1. Тема кваліфікаційної роботи: «Інтелектуальна система створення освітніх матеріалів на основі ГШІ».

Керівник роботи: Сіденко Євген Вікторович, доцент кафедри інтелектуальних інформаційних систем, канд. техн. наук, доцент.

Затверджена наказом ЧНУ ім. Петра Могили від «18» листопада 2024 р. № 321.

2. Строк представлення кваліфікаційної роботи «___» _____ 2025 р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні: інтелектуальна система у формі вебзастосунку, що дозволяє користувачеві генерувати навчальні матеріали (тексти, презентації, звіти, лабораторні та курсові роботи) з використанням

генеративного штучного інтелекту; результатом буде функціональний веб-інтерфейс із можливістю вибору типу документа, мови, тематики, а також з обліком використань і обмеженим безкоштовним доступом.

4. Перелік питань, що підлягають розробці: аналіз існуючих аналогів систем генерації освітніх матеріалів; огляд сучасних AI-технологій для обробки текстової інформації; обґрунтування вибору архітектури вебзастосунку; вибір та налаштування бібліотек і фреймворків; реалізація генерації документів різного формату; інтеграція платіжної системи для розширеного доступу до функцій; тестування системи та аналіз результатів роботи; розробка інтерфейсу користувача з урахуванням зручності та доступності; забезпечення локалізації; розробка бази даних для зберігання користувацьких даних, історії генерацій та аналітики використання.

5. Перелік графічних матеріалів: презентація.

Керівник роботи _____ Євген СІДЕНКО

(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Здобувач _____ Нікіта МОСКАЛЕНКО

(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Дата видачі завдання «18» грудня 2024 р.

КАЛЕНДАРНИЙ ПЛАН
кваліфікаційної роботи

Тема: Інтелектуальна система створення освітніх матеріалів на основі ГШІ

№	Найменування роботи	Початок	Закінчення	Примітки
1	Отримання завдання на виконання КР	18.12.2024	20.12.2024	Виконано
2	Аналіз предметної області та постановка задачі	21.12.2024	30.01.2025	Виконано
3	Огляд літературних джерел, зокрема дослідження сучасних підходів до автоматизованого створення освітніх матеріалів із використанням ГШІ	31.01.2025	01.03.2025	Виконано
4	Аналіз існуючих технологій ГШІ та архітектур, які застосовуються для генерації навчального контенту	02.03.2025	01.04.2025	Виконано
5	Реалізація інтелектуальної системи з використанням обраних технологій, тестування функціоналу та аналіз ефективності генерації освітніх матеріалів	02.04.2025	15.05.2025	Виконано
6	Перший попередній захист КР на засіданні комісії кафедри	16.05.2025	16.05.2025	Виконано
7	Корегування роботи за результатами попереднього захисту	29.05.2025	29.05.2025	Виконано
8	Другий попередній захист КР на засіданні комісії кафедри	30.05.2025	30.05.2025	Виконано
9	Доробка та остаточне оформлення КР	31.05.2025	10.06.2025	Виконано
10	Подання КР, її електронної копії та інших документів (відгуку, рецензії) до захисту	12.06.2025	12.06.2025	Виконано

Керівник роботи _____ Євген СІДЕНКО
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Здобувач _____ Нікіта МОСКАЛЕНКО
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Дата складання календарного плану
«29» січня 2025 р.

АНОТАЦІЯ

до кваліфікаційної роботи
здобувача групи 401 ЧНУ ім. Петра Могили

Москаленка Нікіти Віталійовича

на тему **“ІНТЕЛЕКТУАЛЬНА СИСТЕМА СТВОРЕННЯ ОСВІТНІХ МАТЕРІАЛІВ НА ОСНОВІ ГШІ”**

Актуальність полягає у необхідності автоматизації створення навчальних матеріалів і презентацій, що значно економить час та ресурси викладачів, бізнес-тренерів та інших професіоналів.

Об'єктом роботи є процеси створення освітніх матеріалів.

Предметом роботи є технології створення освітнього контенту з використанням ГШІ.

Метою роботи є розробка інтелектуальної системи у формі вебзастосунку, який за допомогою генеративного штучного інтелекту зможе автоматично генерувати презентації та текстові документи на основі введених даних користувачем.

В результаті виконання кваліфікаційної роботи було створено повнофункціональний вебзастосунок, що дозволяє генерувати навчальні документи та презентації з використанням генеративного ШІ. Реалізовано багатомовну підтримку інтерфейсу, систему обліку користувачів та використань, а також інтеграцію з хмарною базою даних та платіжною системою для управління доступом до преміум-функцій. Система адаптована до українських освітніх реалій і може бути впроваджена в практику викладання та самоосвіти. Результати роботи можуть бути застосовані в освітній та бізнес-сферах, де потрібно швидко і якісно створювати візуальні та текстові матеріали.

Дана робота складається з чотирьох розділів. У першому розділі досліджуються сучасні технології та методи штучного інтелекту, які можуть бути використані для аналізу та синтезу текстової інформації. У другому розділі досліджено методи та

технології генерації текстів, архітектури нейронних мереж та бібліотеки, що використовуються в розробці. Третій розділі розглядається процес розробки вебзастосунку, включаючи архітектуру системи, інтеграцію з AI-алгоритмами та розробку інтерфейсу користувача. Четвертий розділ присвячений тестуванню системи, оцінці її продуктивності та зручності використання для автоматизації навчальних і презентаційних матеріалів. Загальний обсяг роботи – 81 сторінка. Кваліфікаційна робота містить 1 додаток, 54 рисунків і 30 джерел посилання.

Ключові слова: ГШІ, освітні матеріали, вебзастосунок, GPT, презентація, текстовий документ, генерація контенту, інтерфейс користувача.

ABSTRACT

to the qualification work by the student of the group 401 of Petro Mohyla Black Sea National University

Moskalenko Nikita

“INTELLIGENT SYSTEM FOR CREATING EDUCATIONAL MATERIALS BASED ON GENERATIVE ARTIFICIAL INTELLIGENCE”

The relevance lies in the need to automate the creation of educational materials and presentations, which significantly saves time and resources for teachers, business trainers, and other professionals.

The object of the work is the processes of creating educational materials.

The subject of the work is the technologies for creating educational content using generative artificial intelligence.

The aim of the work is to develop an intelligent system in the form of a web application that, with the help of generative artificial intelligence, can automatically generate presentations and text documents based on user input.

As a result of the qualification work, a fully functional web application was created that allows generating educational documents and presentations using generative AI. Multilingual interface support, a user and usage tracking system, as well as integration with a cloud database and a payment system for managing access to premium features were implemented. The system is adapted to Ukrainian educational realities and can be implemented in teaching and self-education practices. The results of the work can be applied in educational and business spheres where there is a need to quickly and efficiently create visual and textual materials.

This work consists of four chapters. The first chapter explores modern technologies and methods of artificial intelligence that can be used for text information analysis and synthesis.

The second chapter investigates the methods and technologies of text generation, neural network architectures, and libraries used in development. The third chapter considers the process of developing the web application, including the system architecture, integration with AI algorithms, and user interface development. The fourth chapter is devoted to system testing, evaluation of its performance and usability for the automation of educational and presentation materials.

The overall scope of the work is 81 pages. Thesis contains 1 application, 54 figures, and 30 references in it.

Key words: generative AI, educational materials, web application, GPT, presentation, text document, content generation, user interface.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ГШІ – Генеративний штучний інтелект

ШІ – Штучний інтелект

ДСТУ – Державні стандарти України

API – Application programming interface

AI – Artificial intelligence

GPT – Generative pre-training transforme

UI – User interface

UX – User experience

ВСТУП

У наш час цифрових технологій і швидкого розвитку штучного інтелекту з’являється все більше можливостей автоматизувати рутинні задачі, зокрема в освіті. Одним із цікавих напрямів є використання генеративного ШІ для створення навчальних матеріалів. Це дозволяє економити час і сили при підготовці презентацій, написанні текстів для занять, лабораторних і курсових робіт.

На сьогодні на ринку є багато комерційних рішень, що частково реалізують подібні функції, проте більшість із них не забезпечують належної локалізації, персоналізації або інтеграції у навчальні процеси згідно з вимогами вищої освіти. Крім того, значна частина таких систем орієнтована на англомовних користувачів і не враховує особливості національних освітніх стандартів, зокрема ДСТУ.

Попри вагомі досягнення у галузі штучного інтелекту, залишаються помітні прогалини в його практичному впровадженні в освітній сфері України. Це вказує на потребу у подальших дослідженнях і вдосконаленні технологій. Особливу увагу слід приділити розробці адаптованих інструментів для автоматизованого створення якісних навчальних матеріалів українською мовою, адже їх відсутність є актуальною проблемою, що вимагає вирішення.

Актуальність теми обумовлена потребою в інтелектуальних системах, здатних швидко, ефективно і точно генерувати освітні документи відповідно до запиту користувача. Це допоможе підвищити продуктивность викладачів, студентів та інших учасників освітнього процесу.

Таким чином, дана кваліфікаційна робота спрямована на розробку інтелектуальної системи створення освітніх матеріалів на основі генеративного ШІ, що відповідає сучасним технічним і педагогічним вимогам.

1 АНАЛІЗ СУЧАСНИХ ПІДХОДІВ ДО ГЕНЕРАЦІЇ ОСВІТНІХ МАТЕРІАЛІВ

1.1 Характеристика сфери автоматизованого створення навчальних матеріалів

Сьогодні в освіті все більше відчувається потреба в автоматизації рутинних завдань, особливо тих, що пов’язані зі створенням навчальних матеріалів. Підготовка презентацій, конспектів чи текстів зазвичай займає багато часу, вимагає дотримання певних стандартів (як от ДСТУ) та ще й потребує креативного підходу, щоб учням, студентам або працівникам було цікаво слухати або дивитись цей матеріал.

Ця робота стосується галузі освітніх технологій і досліджує можливості автоматизованого створення навчального контенту за допомогою генеративного штучного інтелекту. Моделі на базі великих мовних систем (наприклад, GPT, Claude, Gemini) здатні на основі простого запиту формувати розгорнуті тексти, плани занять, тестові завдання або презентації, і все це швидко й досить якісно [1].

Такі інструменти мають практичне застосування в школах, університетах, на курсах, практично всюди, де потрібно готувати навчальні матеріали [2]. Вони суттєво скорочують час підготовки, роблять контент більш адаптивним і відповідним до конкретних потреб аудиторії [3]. Також, це відкриває можливості для персоналізованого навчання, де контент може підлаштовуватись під рівень знань чи профіль користувача [4].

У межах цієї роботи створюється веб-система, яка об'єднує можливості генеративного ШІ, зручного інтерфейсу, підтримки багатьох мов, обліку активності та інтеграції з платіжними сервісами. Такий підхід робить систему актуальною для сучасної цифрової освіти та перспективною для масштабного використання в навчальному процесі.

1.2 Аналіз наявних рішень та публікацій у сфері генеративного навчального контенту

Сьогодні вже існує чимало платформ, які вміють автоматично створювати навчальні матеріали, від текстів і презентацій до візуального та навіть інтерактивного контенту. Зазвичай такі сервіси дозволяють генерувати матеріали на основі коротких запитів, користуються шаблонами оформлення й мають зручний інтерфейс для редагування та експорту [5]. Але часто вони зосереджуються на одному типі контенту, не підтримують форматування за освітніми стандартами або ж вимагають платної підписки.

Нижче подано огляд популярних систем:

– SlidesAI.io – вебзастосунок, що дозволяє створювати презентації на основі текстових описів. Користувач вводить основну ідею або тези, а система автоматично генерує слайди з оформленням [6];

– Tome.app – інтерактивна платформа, орієнтована на генерацію як презентацій, так і документів. Підтримує вставку мультимедійного контенту, анімацій, інтеграцію GPT-моделей [7];

– Canva Docs AI – функція в межах Canva, яка дозволяє створювати та редагувати документи з генерацією тексту. Доступна підтримка візуального оформлення, форматування та шаблонів [8];

– Gamma.app – сучасний інструмент для створення презентацій із акцентом на адаптивність і простоту. Дозволяє комбінувати текст, графіку та інші елементи [9];

– Beautiful.ai – сервіс з автооформленням презентацій відповідно до корпоративного або навчального стилю. В основі — ІІ, який оптимізує структуру слайдів [10];

– Notion AI – інтелектуальний помічник у межах платформи Notion, що допомагає створювати текстові блоки, резюме, плани та інші освітні документи [11];

– Kroma.ai – інструмент, орієнтований на створення бізнес-презентацій, проте також придатний для навчальних матеріалів. Містить готові шаблони та автоматичну генерацію візуального контенту [12];

– MagicSlides – розширення до Google Slides, яке дозволяє автоматично створювати презентації за ключовими словами чи темою [13];

– Visme – платформа для візуалізації даних, презентацій та інфографік. Містить AI-помічника, що пропонує структурування інформації [14];

– Simplified – комплексний сервіс для генерації тексту, дизайну, маркетингових матеріалів і слайдів. Має функції для командної роботи та ІІ-асистент [15].

Проведений аналіз показав, що більшість подібних систем або орієнтовані тільки на презентації, або лише на текст, причому далеко не всі мають підтримку української мови. Це звужує сферу їх практичного застосування в україномовному освітньому середовищі.

Щоб краще зрозуміти поточні підходи до використання генеративного ШІ в освіті, було розглянуто кілька наукових публікацій:

– GAIDE [16]: A Framework for Using Generative AI to Assist in Course Content Development. пропонує фреймворк для створення курсів із допомогою генеративного ШІ, що значно полегшує роботу викладачів. Однак фокус йде на англомовному середовищі. Автори демонструють, як використання генеративного ШІ може підвищити ефективність створення навчальних матеріалів, зменшити навантаження на викладачів та забезпечити академічну якість контенту.

– SocratiQ [17]: A Generative AI-Powered Learning Companion for Personalized Education and Broader Accessibility. Це система, яка формує індивідуальні навчальні маршрути, орієнтуючись на знання студента. Вона працює як чатбот і не дозволяє експортувати повноцінні матеріали. Система реалізує метод Сократа через адаптивні технології навчання, сприяючи глибшому засвоєнню матеріалу.

– Generative Artificial Intelligence in Education: Advancing Adaptive and Personalized Learning [18]. У дослідженні розглядаються застосування генеративного ШІ в освіті, зокрема інтеграція таких інструментів, як Explainpaper та ChatPDF, які поєднують розмовні інтерфейси з освітніми матеріалами. Ця система показує, як ШІ може підвищувати доступність і залучення в навчанні, але більше уваги приділяється адаптації, ніж структурі чи формату.

– GPTutor [19]: Great Personalized Tutor with Large Language Models for Personalized Learning Content Generation. У цій роботі представлено GPTutor – вебзастосунок, який використовує генеративний ШІ для створення персоналізованого навчального контенту та практичних завдань, адаптованих до інтересів і цілей студентів. Система використовує архітектуру без серверів для забезпечення масштабованості та ефективності.

– Can We Trust AI-Generated Educational Content? Comparative Analysis of Human and AI-Generated Learning Resources [20]. Дослідження порівнює якість навчальних ресурсів, створених людьми та генеративним ШІ, у контексті програмування. Результати показують, що ШІ вже досягає рівня, який студенти сприймають як цілком прийнятний.

Аналіз сучасних наукових публікацій свідчить про зростаючий інтерес дослідників до застосування генеративного штучного інтелекту в освітньому процесі. Зокрема, у роботі [16] представлено фреймворк GAIDE, який дозволяє використовувати генеративні моделі для автоматизації створення навчальних курсів, проте реалізація орієнтована на англомовне академічне середовище та не враховує локальні освітні стандарти. У дослідженні [17] запропоновано систему SocratiQ, що виконує роль інтерактивного супутника навчання, але вона переважно функціонує як чатбот і не забезпечує повноцінного експорту матеріалів у навчальному форматі. У роботі [18] акцент зроблено на персоналізацію навчання, проте система зосереджена на адаптації контенту, а не його структурованому створенні. Такі рішення, як GPTutor

[19], дозволяють генерувати навчальний текст, проте мають обмежену функціональність щодо візуалізації та структуризації за освітніми шаблонами. Дослідження [20] порушує питання якості та достовірності ІІ-згенерованого навчального контенту, підкреслюючи необхідність людського контролю та перевірки згенерованого матеріалу.

Загалом, більшість існуючих систем не охоплюють весь спектр вимог, потрібних для якісного створення освітнього контенту: вони або фокусуються на одному типі контенту, або не дотримуються навчальних стандартів і не враховують локалізацію. Це лише підтверджує необхідність створення такої системи, яка була б гнучкою, підтримувала українську мову, відповідала ДСТУ й дозволяла генерувати контент різних типів тобто саме такої, як розробляється в межах цієї кваліфікаційної роботи [21].

1.3 Цілі, задачі та вимоги до розроблюваної системи

Сьогодні, коли дистанційна та персоналізована освіта стають все більш поширеними, виникає гостра потреба у швидкому створенні якісних навчальних матеріалів. Проте підготовка презентацій і документів вручну часто забирає багато часу та зусиль. Генеративний штучний інтелект дозволяє значно спростити цей процес: контент створюється швидко, з урахуванням конкретних потреб користувача [22].

Такі системи є справжньою підтримкою як для викладачів, так і для студентів, адже вони дозволяють оперативно отримати потрібні матеріали, що позитивно впливає на ефективність навчання. Крім того, ШІ може адаптувати стиль і складність подачі відповідно до віку чи рівня підготовки аудиторії [23].

Сучасна освіта рухається в бік інтерактивності та мультимедійності, і системи на базі ШІ здатні не лише писати тексти, а й створювати зображення, тести, і навіть симуляції. Це відкриває нові можливості для викладачів і робить процес навчання

цікавішим і різноманітнішим для студентів [24].

У цьому контексті розробка вебзастосунку для генерації навчальних матеріалів на основі штучного інтелекту виглядає цілком логічним і актуальним кроком. Такий інструмент відповідає сучасним викликам у сфері освіти та розвитку цифрових технологій [25].

Основні завдання, що реалізуються для досягнення мети:

– створення українського веб-інтерфейсу для зручної взаємодії користувачів з системою генерації освітніх матеріалів;

– інтеграція сучасних API генеративного штучного інтелекту (OpenAI, Cohere) для забезпечення високої якості та релевантності створюваного контенту;

– розробка механізму автоматичного форматування згенерованих матеріалів відповідно до державних стандартів України (ДСТУ);

– реалізація функціоналу для створення різних типів освітніх документів з урахуванням специфіки навчального процесу;

– впровадження системи персоналізації та адаптації контенту відповідно до когнітивних особливостей цільової аудиторії;

– забезпечення багатомовної підтримки з акцентом на українську мову для розвитку національного освітнього середовища.

Висновки до розділу 1

У першому розділі було проведено комплексний аналіз поточного стану розробки інтелектуальних систем для створення освітніх матеріалів із використанням генеративного штучного інтелекту. Огляд наукових публікацій та огляд існуючих рішень підтвердили актуальність теми та високий попит на автоматизацію процесів розробки навчального контенту.

Результати дослідження вказують на зростання ролі дистанційного, персоналізованого й безперервного навчання, що формує потребу в інструментах,

здатних швидко та якісно створювати навчальні ресурси. Сучасні освітні підходи дедалі більше орієнтовані на інтерактивність, мультимедійність та гнучкість, що актуалізує запровадження інтелектуальних рішень.

Аналіз існуючих платформ виявив, що більшість із них мають обмежений функціонал: орієнтуються лише на окремі аспекти (текст, чатбот, рекомендації), не враховують повною мірою вимоги до структури, форматування за освітніми стандартами та локалізації.

Водночас генеративний ШІ відкриває нові можливості для автоматизованого створення навчальних матеріалів, які можуть бути адаптовані під конкретні потреби користувачів. Сучасні інтелектуальні системи здатні не лише генерувати текстову інформацію, а й автоматично створювати візуальні матеріали, тести, симуляції та інші елементи навчального контенту, що робить їх надзвичайно перспективними у сфері освіти.

Ключовими перевагами використання генеративного ШІ в освітніх технологіях є:

– можливість персоналізації навчального контенту відповідно до когнітивних особливостей користувачів;
– адаптація матеріалів за складністю, стилем подачі та форматом сприйняття;
– значне скорочення часу на підготовку навчальних матеріалів;
– підвищення рівня залученості студентів до навчального процесу.

Визначено основні напрямки дослідження, серед яких:

– створення україномовного веб-інтерфейсу для генерації освітніх матеріалів;
– інтеграція сучасних API генеративного штучного інтелекту;
– розробка механізму автоматичного форматування згідно з ДСТУ;
– впровадження системи персоналізації та адаптації контенту.

Проведений аналіз підтверджує актуальність розробки спеціалізованої інтелектуальної системи для генерації освітніх матеріалів, яка б вирішувала завдання

створення навчального контенту.

2 МЕТОДИ ТА ТЕХНОЛОГІЇ СТВОРЕННЯ ОСВІТНІХ МАТЕРІАЛІВ

2.1 Методи для вирішення поставленої задачі

У кваліфікаційній роботі застосовано комплексний підхід до автоматизованого створення освітніх матеріалів, який поєднує сучасні методи генерації, персоналізації, обробки та форматування контенту з урахуванням освітніх стандартів.

Основним методом генерації контенту є prompt-інжиніринг із використанням сучасних API генеративного штучного інтелекту (наприклад, OpenAI GPT) [26]. Він передбачає створення спеціально структурованих запитів до мовних моделей для отримання релевантного освітнього контенту. Реалізація методу включає:

– формування початкового запиту з чітким визначенням освітньої мети;

– деталізацію теми через проміжні уточнення;

– перевірку та валідацію згенерованого матеріалу.

Для підвищення ефективності навчання застосовано адаптивний підхід, що дозволяє враховувати індивідуальні характеристики цільової аудиторії. Персоналізація забезпечується через:

– аналіз параметрів користувача (рівень освіти, спеціалізація тощо);

– адаптацію складності та стилю матеріалів;

– врахування когнітивних особливостей користувача, зокрема віку та способу сприйняття інформації.

Для приведення результатів генерації до вимог українських освітніх стандартів застосовується алгоритмічне форматування за ДСТУ. До ключових етапів належать:

– автоматичне формування структури документа (наприклад, вступ, мета, хід роботи);

– застосування уніфікованих правил форматування (такі як, шрифт, міжрядковий інтервал, поля тощо);

– перевірку відповідності технічним вимогам.

Важливою особливістю системи є узгоджена робота всіх її компонентів: генеративних моделей, систем форматування, інтерфейсних рішень і допоміжних сервісів (наприклад, автентифікації, обліку генерацій).

Окрему увагу приділено підтримці багатомовності. Система дозволяє створювати освітні матеріали українською та англійською мовами, адаптуючи контент до мовних особливостей, що особливо важливо для україномовного освітнього середовища та двомовного навчання.

2.2 Технології розробки системи

Було обрано новітні технології веб-розробки та генеративного штучного інтелекту. Автоматичне форматування документів реалізовано відповідно до вимог ДСТУ. Це забезпечує:

– коректний шрифт;
– встановлений міжрядковий інтервал;
– правильні поля та відступи;
– стандартизоване оформлення заголовків;
– правильно оформлені переліки.

Генерацію та збереження готового документа формату docx забезпечено бібліотекою docx.js для текстових документів це JavaScript бібліотека, що надає широкі можливості для програмного створення та форматування документів у форматі .docx. У системі використовуються такі її компоненти:

– document – головний клас для створення документа з налаштуваннями шрифту Times New Roman, розміру 14 пт та міжрядкового інтервалу 1,5;

– packer – клас для конвертації створеного документа у формат Blob для подальшого збереження;

– paragraph – компонент, що забезпечує створення абзаців з відступом першого рядка 1,25 см та вирівнюванням за шириною через AlignmentType.JUSTIFIED;

– textRun – елемент для додавання текстових блоків з можливістю форматування (жирний шрифт для заголовків);

– headingLevel – встановлення рівнів заголовків для автоматичного формування структури документа;

– header – компонент для створення верхніх колонтитулів з номерами сторінок;

– tableOfContents – автоматичне генерування змісту документа з гіперпосиланнями на розділи;

– pageBreak – елемент для примусового розриву сторінки між розділами;

– pageNumber – компонент для автоматичної нумерації сторінок у верхньому колонтитулі.

Для збереження документів на пристрій користувача інтегровано бібліотеку file-saver, яка використовує метод saveAs() для завантаження створеного файла.

Документ формується з чіткою структурою, що відповідає вимогам до курсових робіт:

– титульна сторінка;

– зміст з автоматичним формуванням;

– вступ;

– основні розділи з підрозділами (до трьох рівнів вкладеності);

– висновки.

Система автоматично застосовує до документа стандартизовані налаштування полів сторінки (верхнє та нижнє – 2 см, ліве – 2,5 см, праве – 1 см), забезпечує правильне розміщення тексту та форматування відповідно до вимог ДСТУ. Генерація змістовних текстових блоків здійснюється за допомогою інтеграції з мовною моделлю

OpenAI та Cohere. Система приймає тему курсової роботи від користувача і формує відповідні запити до моделі для отримання змістовних фрагментів. Функція generateCourseIntroFile реалізує запити до мовної моделі для одночасного створення до десяти підрозділів, після чого отримані текстові блоки передаються у функцію формування документа та автоматично розміщуються у відповідних розділах з правильним форматуванням (рис. 2.1-2.2).

```
export async function generateCourseIntroFile(topic: string, fileName = "CourseWork.docx", charCount = 50) {
  const introPrompt = `Напиши текст для розділу "ВСТУП" курсової роботи на тему "${topic}".
Текст має бути не менше половини сторінки (приблизно ${charCount} знаків).
У вступі коротко виклади:
- оцінку сучасного стану задачі;
- теоретичні та практичні здобутки;
- прогалини знань, недоліки існуючих рішень або технологій;
- актуальність теми;
- підставу для виконання цієї роботи.
Дотримуйся академічного стилю. Мова – українська.`;

const analysisPrompt = `Напиши текст для підпункту "1.1 Опис предметної сфери" курсової роботи на тему "${topic}" просто текст, без пунктів.
Опиши, що саме досліджується в межах цієї предметної області, які ключові поняття і процеси, чим ця сфера є важливою в сучасному контексті.
Текст має бути академічного стилю, обсяг – ${charCount} знаків. Мова – українська.`;

const analoguesPrompt = `Напиши текст для підпункту "1.2 Огляд та аналіз наявних аналогів та публікацій" курсової роботи на тему "${topic}", просто текст, без пунктів.
Поясни, які наукові чи практичні розробки вже існують, їх переваги й недоліки.
Стиль академічний, обсяг – ${charCount} знаків. Мова – українська.`;
```

Рисунок 2.1 – Частина функції «generateCourseIntroFile»

```
const taskStatementPrompt = `Напиши текст для підпункту "1.3 Постановка задачі (актуальність, мета, об'єкт і предмет роботи, завдання для досягнення поставленої мети)" курсової роботи на тему "${topic}",
Розпиши всі елементи: актуальність, мета, об'єкт, предмет, завдання.
Стиль академічний, обсяг – ${charCount} знаків. Мова – українська.`;

const methodsPrompt = `Напиши текст для підпункту "2.1 Методи для вирішення поставленої задачі" курсової роботи на тему "${topic}", просто текст, без пунктів.
Розпиши які методи можно використати.
Стиль академічний, обсяг – ${charCount} знаків. Мова – українська.`;

const technologiesPrompt = `Напиши текст для підпункту "2.2 Технології розробки системи" курсової роботи на тему "${topic}", просто текст, без пунктів.
Розпиши які технології були використані або будуть.
Стиль академічний, обсяг – ${charCount} знаків. Мова – українська.`;

const inputandstructurePrompt = `Напиши текст для підпункту "3.1 Опис вхідних даних та структури системи "${topic}", просто текст, без пунктів.
Розпиши вхідні дані та структуру системи.
Стиль академічний, обсяг – ${charCount} знаків. Мова – українська.`;

const designingPrompt = `Напиши текст для підпункту "3.2 Проектування/моделювання "${topic}", просто текст, без пунктів.
Розпиши проектування або моделювання проекту на цю тему.
Стиль академічний, обсяг – ${charCount} знаків. Мова – українська.`;

const analysisresultPrompt = `Напиши текст для підпункту "3.3 Аналіз отриманих результатів "${topic}", просто текст, без пунктів.
Розпиши аналіз результатів.
Стиль академічний, обсяг – ${charCount} знаків. Мова – українська.`;

const conclusionsPrompt = `Напиши текст для підпункту "ВИСНОВКИ"${topic}", просто текст, без пунктів.
Розпиши висновки на основі всіх розділів, перший розділ це аналіз сучасного стану поставленої задачі, другий це моделі, методи та інформаційні технології для вирішення поставленої задачі, третій розділ ц
Стиль академічний, обсяг – ${charCount} знаків. Мова – українська.`;
```

Рисунок 2.2 – Частина функції «generateCourseIntroFile»

Структуризація документа забезпечується за допомогою бібліотеки docx.js, де кожен розділ і підрозділ отримує відповідне форматування згідно з вимогами ДСТУ.

Заголовки оформлюються за допомогою стилів HEADING_1 та HEADING_2, встановлюється міжрядковий інтервал (рис. 2.3), шрифт Times New Roman розміром 14 пт, відступ першого рядка, а також вирівнювання тексту за шириною. Усі текстові блоки, згенеровані на основі промптів, розбиваються на абзаци. Кожен абзац створюється з дотриманням академічного стилю.

```
16  export async function generateCourseIntroFileFromData(
30      const doc = new Document({
50          sections: [
                },
152             children: [
153                 new Paragraph({
154                     alignment: AlignmentType.CENTER,
155                     heading: HeadingLevel.HEADING_1,
156                     spacing: { before: 240, after: 240 },
157                     indent: { firstLine: 0 },
158                     children: [
159                         new TextRun({
160                             text: "1 АНАЛІЗ СУЧАСНОГО СТАНУ ПОСТАВЛЕНОЇ ЗАДАЧІ",
161                             bold: true,
162                             size: 28, // 14pt
163                             font: "Times New Roman",
164                             color: "000000",
165                         }),
166                     ],
167                 }),
168                 new Paragraph({
169                     heading: HeadingLevel.HEADING_2,
170                     spacing: { before: 240, after: 240 },
171                     children: [
172                         new TextRun({
173                             text: "1.1 Опис предметної сфери",
174                             bold: true,
175                             size: 28, // 14pt
176                             font: "Times New Roman",
177                             color: "000000",
178                         }),
179                     ],
180                 }),
181                 ...analysisText
182                     .split(/\n{2,}/)
183                     .map(p => p.trim())
184                     .filter(Boolean)
185                     .map(
186                         p =>
187                             new Paragraph({
```

Рисунок 2.3 – Частина функції «generateCourseIntroFileFromData»

Створення та збереження готового документу формату pptx реалізовано бібліотекою pptxgenjs для презентацій – сучасна JavaScript бібліотека для програмного створення презентацій у форматі .pptx з широкими можливостями форматування та дизайну. Імплементовано такі її функції:

– pptxGenJS – основний клас для створення презентації з різними типами слайдів;

– addSlide() – метод для додавання нового слайду в презентацію;

– addText() – функція для додавання текстових блоків із гнучкими налаштуваннями форматування (розмір шрифту, колір, вирівнювання, відступи, жирність);

– addImage() – компонент для вставки зображень на слайд з підтримкою різних форматів, включаючи base64-кодовані зображення з налаштуваннями розміру та розташування;

– addShape() – функція для додавання векторних фігур (еліпсів) як декоративних елементів фону слайдів;

– writeFile() – метод для збереження готової презентації на пристрій користувача.

Розроблено додаткову функцію addBackgroundShapes() для програмного створення стильового оформлення слайдів із використанням геометричних фігур, що забезпечує єдиний дизайн усієї презентації. Система використовує дизайнерські шаблони для різних типів слайдів:

– титульний слайд з назвою теми на кольоровому фоні;

– інформаційні слайди з текстовою частиною та супровідним зображенням;

– заключний слайд з подякою для аудиторії.

Архітектура модуля дозволяє динамічно генерувати оптимальну кількість слайдів на основі вхідного тексту, автоматично розподіляючи речення по слайдах та забезпечуючи належне розміщення текстової інформації та ілюстративних матеріалів.

Генерація змісту презентації здійснюється за допомогою викликів до API OpenAI, які формують текстовий матеріал та кольорову палітру для оформлення слайдів. Також, система надсилає запит для отримання 10 речень на задану тему, що будуть використані як основа для інформаційних слайдів, а також окремий запит для визначення кольору фону, на якому легко повинен читатися шрифт з білим коліром. Обидві ці функції реалізовано в одному файлі (рис. 2.4-2.5), де для формування тексту використовується модель gpt-3.5-turbo, а кольорове рішення підбирається у форматі HEX-коду, цей код колір потім буде використано при створенні фону презентації в іншому файлі.

```
6   export async function callOpenAIAPI(message: string): Promise<string | null> {
7     const messages: ChatCompletionMessageParam[] = [
8       {
9         role: "system",
10        content: "You are a helpful assistant."
11      },
12      {
13        role: "user",
14        content: `Make 30 sentences on this topic, please just make sentences without any number etc. only text, adapt to prompt language. PROMPT: ${message}`
15      }
16    ];
17
18    try {
19      return await openaiComplete(messages, "gpt-3.5-turbo");
20    } catch (error) {
21      console.error("OpenAI error:", error);
22      return null;
23    }
24  }
```

Рисунок 2.4 – Функція «callOpenAIAPI»

```
51  export async function callOpenAIForColor(
52    theme: string,
53    model: "gpt-3.5-turbo" | "gpt-4" = "gpt-3.5-turbo"
54  ): Promise<string | null> {
55    const messages: ChatCompletionMessageParam[] = [
56      {
57        role: "system",
58        content: "You are a helpful assistant."
59      },
60      {
61        role: "user",
62        content: `Suggest a color that would suit the theme and it should be readble with white text. Please respond with only the HEX color code (e.g., #0099FF): ${theme}`
63      }
64    ];
65
66    try {
67      return await openaiComplete(messages, model);
68    } catch (error) {
69      console.error("OpenAI error:", error);
70      return null;
71    }
72  }
```

Рисунок 2.5 – Функція «callOpenAIForColor»

Формування та збереження презентації реалізовано з використанням бібліотеки pptxgenjs, яка дозволяє програмно створювати слайди з текстовим і графічним вмістом. На титульному слайді розміщується тема презентації та відповідне зображення, що задається автоматично. Для основного вмісту генерується серія слайдів, де кожне речення розміщується в межах фонової геометричної фігури, що покриває всю область слайду, створюючи ефект сучасного дизайну. Кожен слайд додатково містить ілюстрацію, розташовану відповідно до заздалегідь визначених координат. Шрифти, кольори, розміри елементів, відступи та розташування визначаються автоматично.

Фінальна презентація експортується у формат .pptx, який користувач може завантажити на свій пристрій. Збереження здійснюється через метод writeFile() бібліотеки pptxgenjs, що генерує готовий документ із усіма налаштуваннями. Частина коду, яка відповідає за формування слайдів та структурування вмісту (рис. 2.6).

```
import PptxGenJS from "pptxgenjs";

export function generatePresentation(text: string, topic: string, imagesBase64: string[], color: string, rounding: boolean): void {
    const pptx = new PptxGenJS();

    const sentences = text.split('.').filter(sentence => sentence.trim() !== '');

    const titleSlide = pptx.addSlide();
    titleSlide.addText(topic, {
        x: "10%" as PptxGenJS.Coord,
        y: "5%" as PptxGenJS.Coord,
        w: "80%" as PptxGenJS.Coord,
        h: "20%" as PptxGenJS.Coord,
        fontSize: 36,
        color: "FFFFFF",
        fill: { color: color },
        align: "center",
        valign: "middle",
        margin: 5,
        bold: true,
        fontFace: "Times New Roman"
    });

    if (imagesBase64.length > 0) {
        titleSlide.addImage({
            data: imagesBase64[0],
            x: "25%" as PptxGenJS.Coord,
            y: "30%" as PptxGenJS.Coord,
            w: "50%" as PptxGenJS.Coord,
            h: "50%" as PptxGenJS.Coord,
            rounding: rounding ? true : false
        });
    }

    for (let i = 0; i < imagesBase64.length; i++) {
        const slide = pptx.addSlide();
        const sentence = sentences[i] ? sentences[i] : '';

        slide.addText(topic, {
            x: "5%" as PptxGenJS.Coord,
            y: "5%" as PptxGenJS.Coord,
```

Рисунок 2.6 – Частина функції «generatePresentation»

```
62          slide.addText(sentence, {
63              x: "5%" as PptxGenJS.Coord,
64              y: "20%" as PptxGenJS.Coord,
65              w: "40%" as PptxGenJS.Coord,
66              h: "70%" as PptxGenJS.Coord,
67              fontSize: 18,
68              color: "FFFFFF",
69              align: "center",
70              valign: "middle",
71              margin: 5,
72              bold: true,
73              fontFace: "Arial"
74          });
75
76          const imageOptions = {
77              x: "50%" as PptxGenJS.Coord,
78              y: "20%" as PptxGenJS.Coord,
79              w: "45%" as PptxGenJS.Coord,
80              h: "70%" as PptxGenJS.Coord,
81              rounding: rounding ? true : false
82          };
83
84          slide.addImage({
85              data: imagesBase64[i],
86              ...imageOptions,
87          });
88      }
89      const thankYouSlide = pptx.addSlide();
90      thankYouSlide.addShape("rect", {
91          x: 0, y: 0, w: "100%", h: "100%", fill: { color: color }
92      });
93      thankYouSlide.addText("Thank you for watching", {
94          x: "10%", y: "40%", w: "80%", h: "20%",
95          fontSize: 36, color: "FFFFFF", fill: { color: color },
96          align: "center", valign: "middle", margin: 5,
97          bold: true, fontFace: "Times New Roman"
98      });
99
100     pptx.writeFile({ fileName: `${topic}.pptx` });
```

Рисунок 2.7 – Частина функції «generatePresentation»

Висновки до розділу 2

У цьому розділі було досліджено, як працює система, яка була створена для автоматичної генерації освітніх матеріалів. Основною ідеєю є використання сучасних AI-технологій для створення текстів і презентацій. Щоб отримати якісний результат, було застосовано кілька етапів генерації: спочатку формується основний запит, потім уточнюється інформація, і на завершення все перевіряється.

Також була передбачена можливість адаптації матеріалів під різних користувачів. Наприклад, система враховує рівень освіти людини та її спеціалізацію, тобто вона сама підлаштовується під потреби користувача.

Окрему увагу було приділено тому, як оформляються результати. Для створення текстових документів було використано бібліотеку docx.js, яка дозволяє автоматично задавати шрифти, поля, структуру та формувати зміст. А для презентацій це pptxgenjs, завдяки якій кожен слайд оформлюється красиво й правильно, з урахуванням кольорової гами та стилів.

Усе це працює як єдина система: AI генерує контент, бібліотеки відповідають за правильне оформлення, і користувач отримує готовий файл — документ або презентацію, який можна одразу використовувати.

Якщо казати загалом, то вдалося створити досить гнучку й сучасну систему, яка може спростити підготовку навчальних матеріалів для викладачів і студентів.

3 ПРОЄКТНІ РІШЕННЯ ДЛЯ ВИКОНАННЯ ПОСТАВЛЕНОЇ МЕТИ

3.1 Структура проєкту

Архітектура застосунку побудована за компонентним підходом (рис. 3.1). Основні частини системи:

– інтерфейс користувача (UI) – реалізований з використанням React та CSS-модулів;

– менеджмент стану – реалізований через Redux з окремими слайсами для аутентифікації, обліку генерацій та преміум-функцій;

– обробка генерацій – функціонал виклику API для створення презентацій і документів;

– платіжна система – інтеграція з Stripe для монетизації доступу до генерацій;

– багатомовність – забезпечена за допомогою бібліотеки i18next.

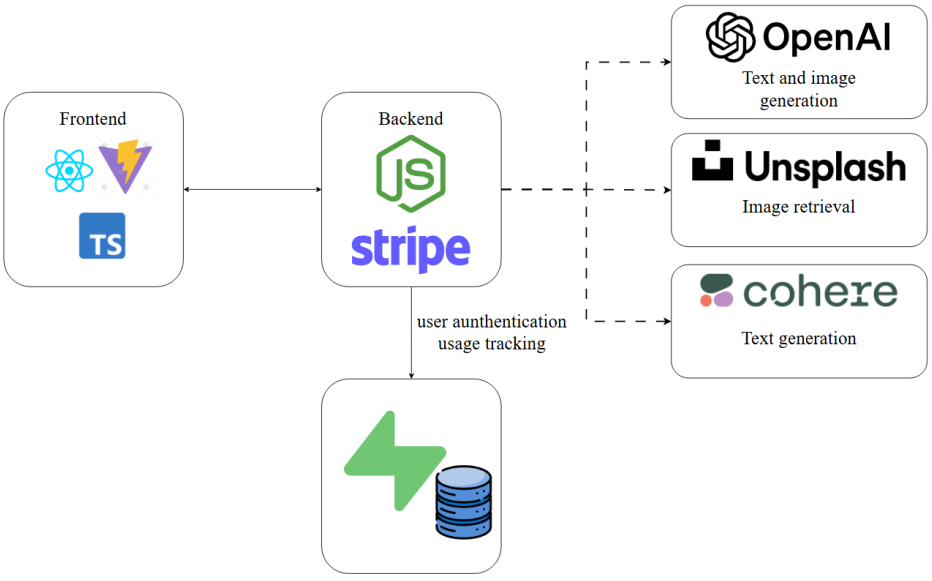


Рисунок 3.1 – Структурна схема системи

Основною платформою для створення інтерфейсу став React [27]. Завдяки своїй високій продуктивності ця бібліотека дозволяє створювати швидкі, зручні й інтерактивні додатки. React добре підходить для динамічного контенту, адже ефективно керує оновленням елементів на сторінці. Його використання віртуального DOM дає змогу мінімізувати зміни в реальному DOM, що пришвидшує рендеринг і покращує загальну швидкодію.

Для комфортнішої розробки й швидкого запуску проєкту було обрано інструмент Vite. Він забезпечує миттєве оновлення змін без повного перезавантаження сторінки. На відміну від класичних інструментів збірки, Vite дозволяє значно зекономити час під час створення і тестування застосунку. До того ж він використовує сучасні можливості браузерів і забезпечує оптимізовану збірку вже для продакшн-версії.

Архітектура додатку побудована на функціональних компонентах, які є зручнішими і сучаснішими у порівнянні з класовими. Такий підхід спрощує структуру коду та його підтримку. Для керування станом компонентів та обробки побічних ефектів застосовуються React Hooks. Це дозволяє писати менше коду, але досягати тієї ж функціональності, легко розділяючи логіку на окремі частини, які можна повторно використовувати в різних компонентах.

Управління станом додатку реалізовано за допомогою Redux, що дає змогу централізовано обробляти важливі параметри:

– дані про автентифікацію користувача;
– інформацію про преміум-статус;
– кількість доступних генерацій;
– налаштування системи.

Redux допомагає зручно керувати станом додатку, бо всі дані йдуть в одному напрямку – це спрощує розуміння, що і коли змінюється. Такий підхід полегшує

відлагодження, а ще дуже зручно, коли проєкт стає більшим і складнішим – можна розбити логіку стану на менші частини (редьюсери) і працювати з ними окремо.

Щоб дані не зникали після перезавантаження сторінки, я використовую localStorage. Це дозволяє зберігати, наприклад, вибрану тему або варіант генерації, щоб користувачеві не довелося налаштовувати все заново.

Сторінки організовані через бібліотеку react-router-dom, що робить навігацію між різними частинами додатку легшою. Ця бібліотека дає змогу створювати багато-сторінкові програми в односторінковій архітектурі, що покращує користувацький досвід завдяки швидким переходам без повного перезавантаження. React Router також підтримує динамічні маршрути і параметри URL та програмну навігацію, це дозволяє реалізувати складну логіку переходів між різними частинами додатка.

Також у проєкті використовуються контрольовані форми. Це означає, що всі поля керуються через стан додатку, завдяки чому можна одразу перевіряти правильність введених даних і реагувати на зміни. Це зручно як для користувача, так і для розробника.

Окремо варто згадати модальні вікна. Вони використовуються для важливих дій наприклад, при вході в акаунт чи для повідомлень. Це дозволяє зосередити увагу користувача, не переводячи його на іншу сторінку.

Інтерфейс адаптований під різні пристрої: від телефонів до великих екранів. Завдяки адаптивній верстці сторінки виглядають коректно незалежно від розміру екрана, що робить користування додатком комфортним у будь-яких умовах.

3.2 Реалізація функціоналу генерації

На початку розробки було реалізовано генерацію презентацій у форматі .pptx та текстових документів у форматі .docx. Користувач просто вводить тему, а система сама формує слайди або текстову структуру документа – все максимально зручно та автоматизовано.

Для створення самого контенту використовується штучний інтелект. Основним обрано OpenAI API [28], який генерує тексти на основі введеного запиту. Цей сервіс дозволяє отримувати якісний і змістовний контент, незалежно від того, чи це презентація, звіт або інший документ.

Також передбачена підтримка альтернативних сервісів, наприклад, Cohere. Він більше орієнтований на багатомовну генерацію, що зручно для тих, хто працює з текстами не лише українською чи англійською. Це дає змогу обрати той сервіс, який краще підходить під конкретні завдання.

У системі реалізовано вибір моделі штучного інтелекту. Наприклад, при використанні OpenAI можна самостійно обрати між GPT-3.5 Turbo та GPT-4, залежно від того, що важливіше — швидкість чи якість результату. Це дає користувачеві більше контролю над процесом.

Щоб зробити матеріали ще більш візуально привабливими, система вміє додавати зображення. У звичайній версії це інтеграція з Unsplash – там є багато якісних фотографій. А для користувачів із преміум-доступом реалізовано ще й генерацію унікальних зображень через DALL·E – досить просто ввести опис, і система створить ілюстрацію.

3.3 Аутентифікація користувачів

У розробленій системі реалізовано зручну і сучасну авторизацію через сервіс Supabase [29]. Користувачі можуть реєструватися і входити за допомогою електронної пошти та пароля, а також через Google або GitHub, тобто кому як зручно. Все працює швидко й безпечно, що важливо для захисту персональних даних.

Усі облікові записи зберігаються централізовано в базі даних Supabase. Кожному користувачеві присвоюється унікальний ідентифікатор, а інформація надійно зашифрована. Це означає, що сторонні не можуть отримати доступ до ваших даних.

Стан авторизації інтегровано з Redux, завдяки чому система знає, хто зараз залогінений, і може швидко реагувати на зміну стану. Крім того, сесія зберігається навіть після перезавантаження сторінки, отже не потрібно щоразу заходити заново.

Після входу користувач потрапляє у свій профіль, де бачить доступні генерації, налаштовує ім’я, переглядає статистику та використовує всі доступні функції. У системі є поділ прав доступу наприклад, у преміум-користувачів розширений функціонал.

Вибір на користь Supabase дав багато переваг: стабільну роботу з акаунтами, просту авторизацію, швидке відновлення пароля й можливість бачити історію генерацій що, коли і з яким запитом створювалося. Все це робить систему не тільки технічно надійною, а й зручною та прозорою для користувача.

3.4 Облік генерацій і платіжна система

Щоб користувач міг бачити, скільки генерацій йому ще доступно, у системі реалізовано спеціальний Redux-слайс це usageSlice, який відповідає за облік використань. Якщо генерації закінчуються, їх можна легко докупити, для цього інтегровано платіжну систему Stripe [30]. Після успішної оплати кількість генерацій автоматично оновлюється, також можно окремо придбати статус «Premium», він надасть більше можливостей для генерації. Після покупки дані про придбані використання або преміум статус спочатку оновляться в хмарній базі даних, а потім вже в самому вебзастосунку.

Обробка оплат відбувається через Stripe API, що гарантує безпечність усіх транзакцій. Серверна частина реалізована на Node.js вона відповідає за перевірку платежу та оновлення даних користувача.

Для зручності додано модальне вікно, де можна вибрати потрібний пакет, побачити ціну та оплатити покупку.

3.5 Локалізація

Щоб система була зручною для користувачів з різних мовних середовищ, додано підтримку локалізації через бібліотеку i18next. Вона дозволяє миттєво перемикати мову інтерфейсу без перезавантаження сторінки усе оновлюється динамічно. Це охоплює не лише кнопки й повідомлення, а й динамічний контент.

Мову можна змінити прямо з навбару, і вона зберігається для наступних сеансів. Переклади зберігаються у JSON-файлах, тож додати нову мову дуже просто. Основна увага приділена повному українському перекладу, адже проєкт орієнтований саме на україномовного користувача.

3.6 Форматування освітніх матеріалів

Щоб згенеровані документи були не лише змістовними, а й готовими до подачі, додано автоматичне форматування відповідно до стандартів ДСТУ. Це означає правильні шрифти, відступи, міжрядкові інтервали, поля, заголовки та інші технічні вимоги усе застосовується автоматично. Такий підхід зменшує потребу в ручному редагуванні, що зручно для студентів і викладачів.

Для створення документів використовуються бібліотеки docx.js для тексту та pptxgenjs для презентацій.

3.7 Обґрунтування вибору технологій

У першому розділі було проведено аналіз сучасного стану розробки інтелектуальних систем створення освітніх матеріалів на основі генеративного ШІ. Огляд публікацій та існуючих рішень показав актуальність теми, а також потребу в автоматизації створення навчального контенту. На основі проведеного аналізу обґрунтовано вибір таких технологій:

– React + Vite – для швидкої, продуктивної та зручної фронтенд-розробки;

– Redux – для керування глобальним станом;
– Stripe – для платіжної інтеграції;
– OpenAI API / Cohere – як ядро генеративного функціоналу;
– TypeScript – для підтримуваного й надійного коду;
– docx.js та pptxgenjs – для створення документації та презентацій;
– i18next – для локалізації інтерфейсу.

Висновки до розділу 3

У цьому розділі детально розглянуто технічну реалізацію інтелектуальної системи генерації освітніх матеріалів. Розробка здійснювалась із використанням сучасного фронтенд-стека (React, Vite, Redux, TypeScript) та інтеграцією зовнішніх AI-сервісів. Успішно реалізовано ключовий функціонал: генерацію документів і презентацій, автентифікацію, монетизацію через Stripe, керування використаннями, підтримку мультимовності та автоматичне форматування за ДСТУ. Отримані результати створюють надійну основу для подальшого розширення системи, покращення користувацького досвіду та впровадження нових типів контенту.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

4.1 Реалізація аутентифікації користувачів

Було реалізовано функціонал аутентифікації користувачів, що включає можливість реєстрації та входу (рис. 4.1-4.3) до системи. Головний інтерфейс аутентифікації розроблений на основі компонентів React з використанням стану Redux для збереження інформації про авторизацію користувача у глобальному стані додатку. Реалізація забезпечує захищену перевірку даних та підтримує динамічне відображення елементів інтерфейсу, зокрема сторінку залогіненого користувача (рис. 4.4).

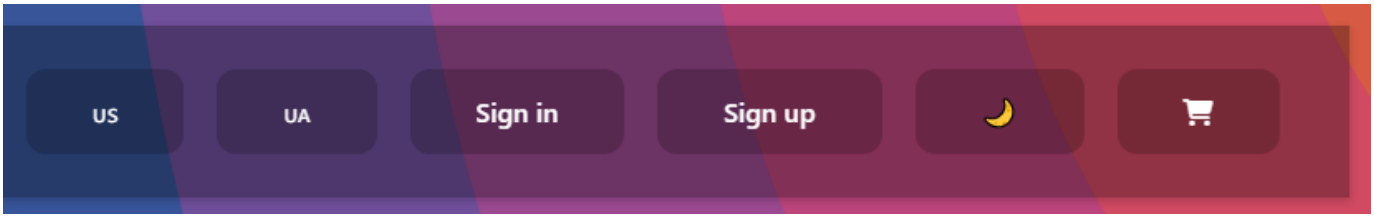


Рисунок 4.1 – Кнопки «Sign in», «Sign up», «Theme» та «Cart»

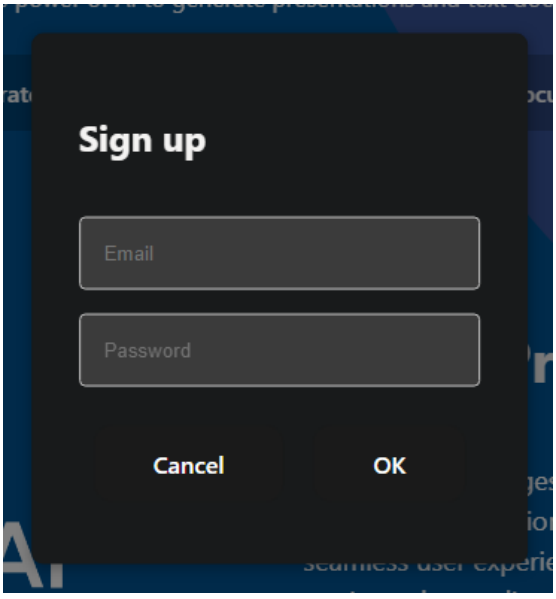


Рисунок 4.2 – Вікно реєстрації

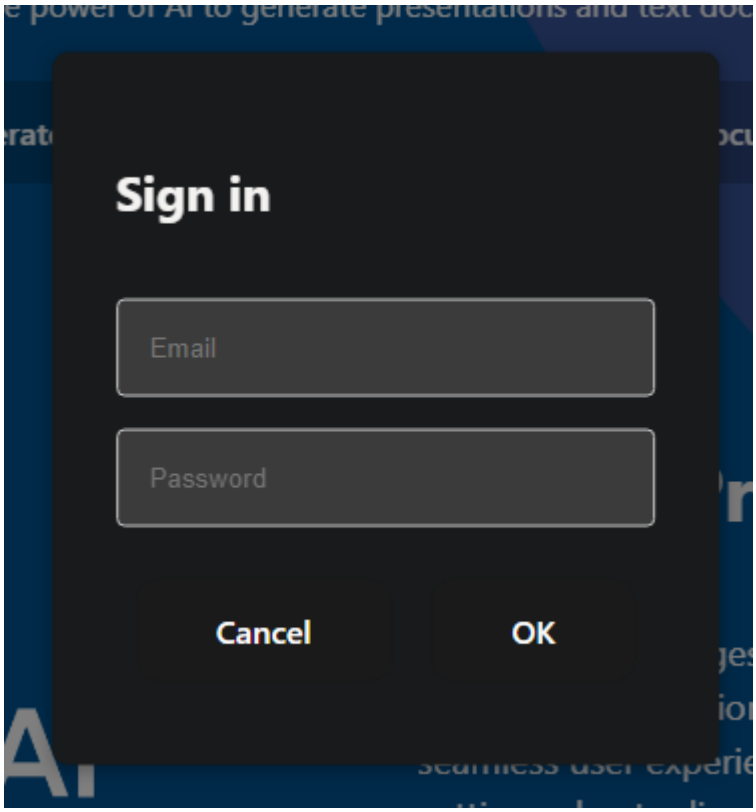


Рисунок 4.3 – Вікно входу

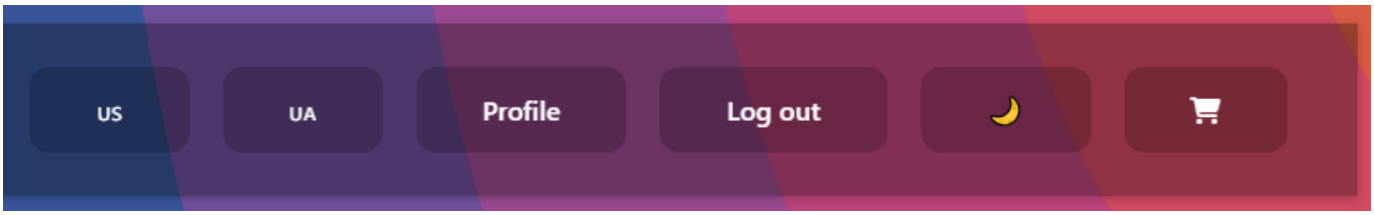


Рисунок 4.4 – Залогінений користувач

Аутентифікація реалізована за допомогою хмарного бекенду Supabase, який забезпечує обробку сесій, токенів доступу та зберігання облікових записів у таблиці auth.users. Окрім цього, для кожного користувача автоматично створюється запис у таблиці Profiles (рис. 4.5), яка пов’язана з таблицею auth.users через поле id (типу uuid), що виконує роль зовнішнього ключа.

У таблиці Profiles зберігаються персональні дані користувача, зокрема:

– email – адреса електронної пошти, за якою зареєстрований користувач;

– name – ім’я, яке користувач може ввести лише один раз при першому відвідуванні сторінки профілю;

– created_at – дата реєстрації;

– uses – кількість доступних генерацій;

– has_premium – статус преміум-доступу.

Інтерфейс профілю користувача реалізований у вигляді модального вікна, яке відкривається з навігаційної панелі. У профілі відображаються всі згадані персональні дані: email, ім’я, дата реєстрації, а також поточна кількість доступних генерацій. Цей функціонал забезпечує прозорість для користувача щодо статусу його акаунту та доступу до сервісу. Також supabase надає змогу редагувати таблиці з даними користувачів (рис. 4.6-4.7).

Крім персональних даних, система також фіксує всі генерації, що здійснює користувач. Для цього використовується таблиця generations, яка зберігає історію запитів. У цій таблиці кожен запис містить:

– id – унікальний ідентифікатор генерації (типу uuid);

– user_id – зовнішній ключ, що вказує на користувача в таблиці Profiles;

– prompt – текст запиту, який був використаний для генерації;

– type – тип генерації (може бути або 'word', або 'presentation');

– created_at – дата та час створення генерації.

Поле user_id пов’язане з таблицею Profiles через зовнішній ключ із каскадним видаленням. Таким чином, при видаленні користувача з бази, пов’язані з ним генерації також автоматично видаляються (рис. 4.8). Таблиця дозволяє не лише зберігати історію дій користувача, а й використовуватись для аналітики або виведення попередніх результатів генерацій.

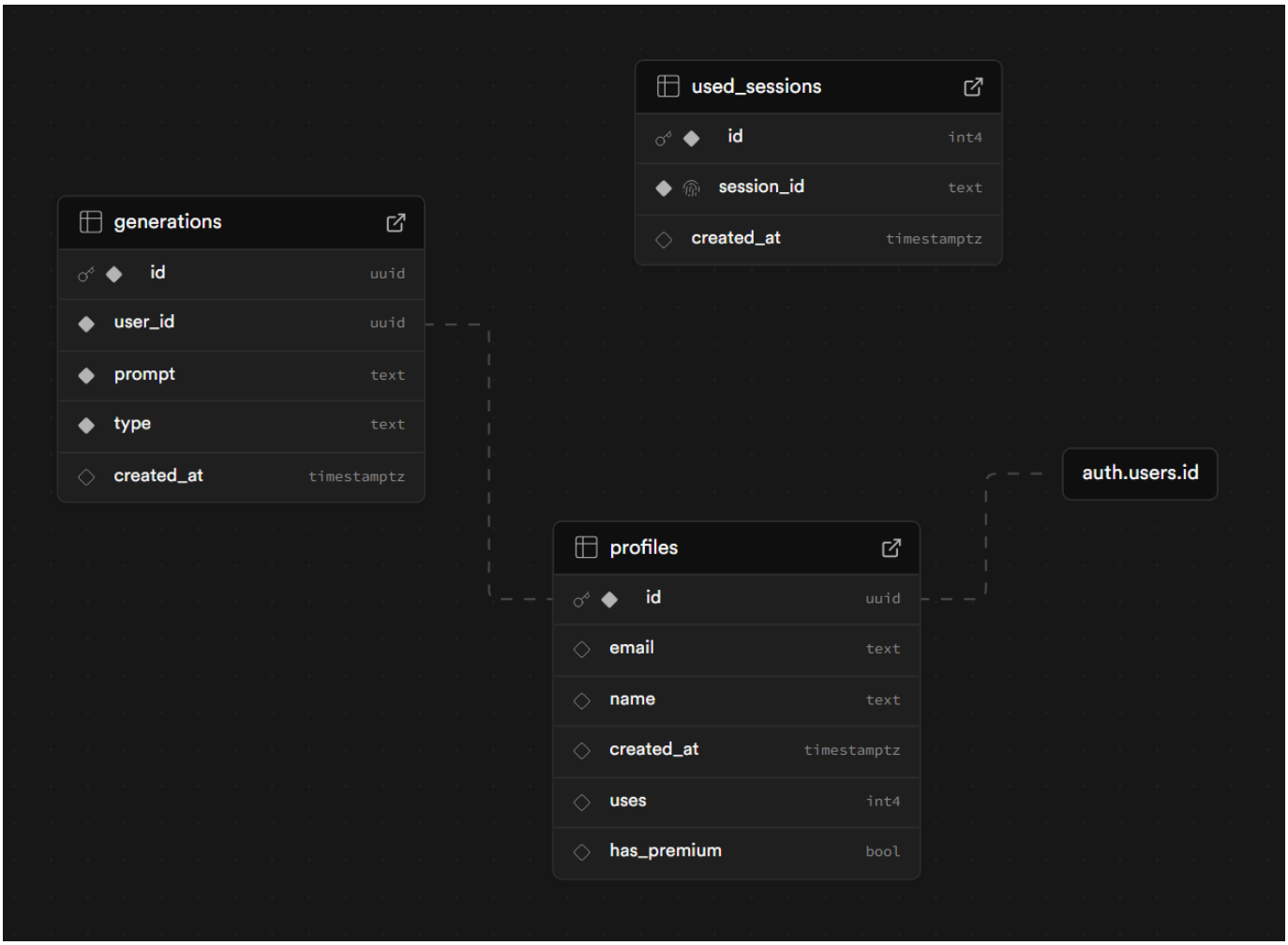


Рисунок 4.5 – Таблиці в Schema Visualizer

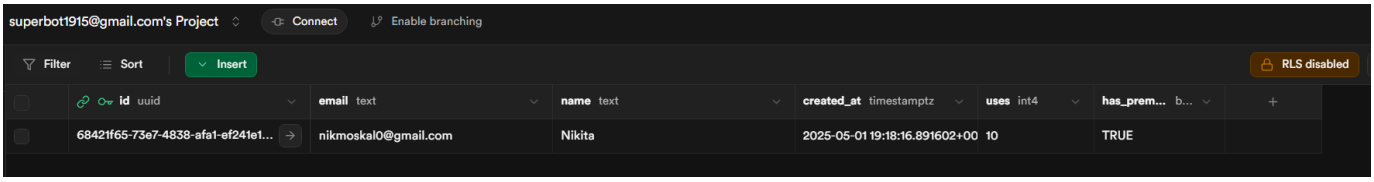


Рисунок 4.6 – Редактор таблиць, таблиця «profiles»

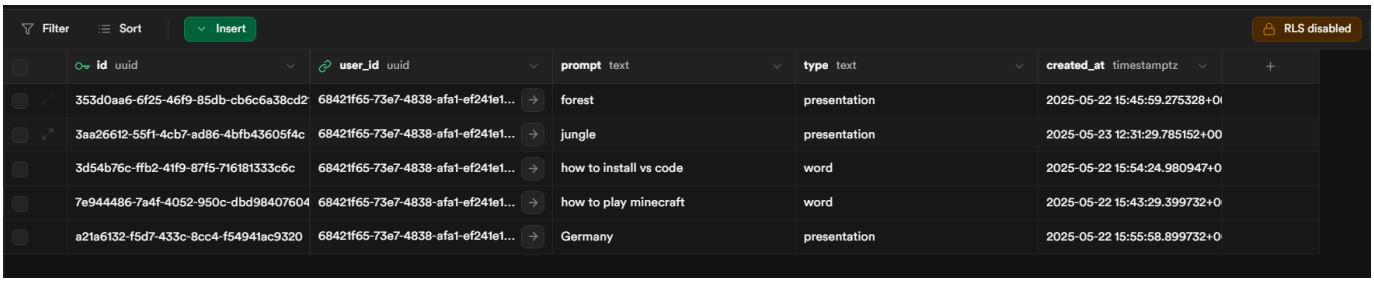


Рисунок 4.7 – Редактор таблиць, таблиця «generations»

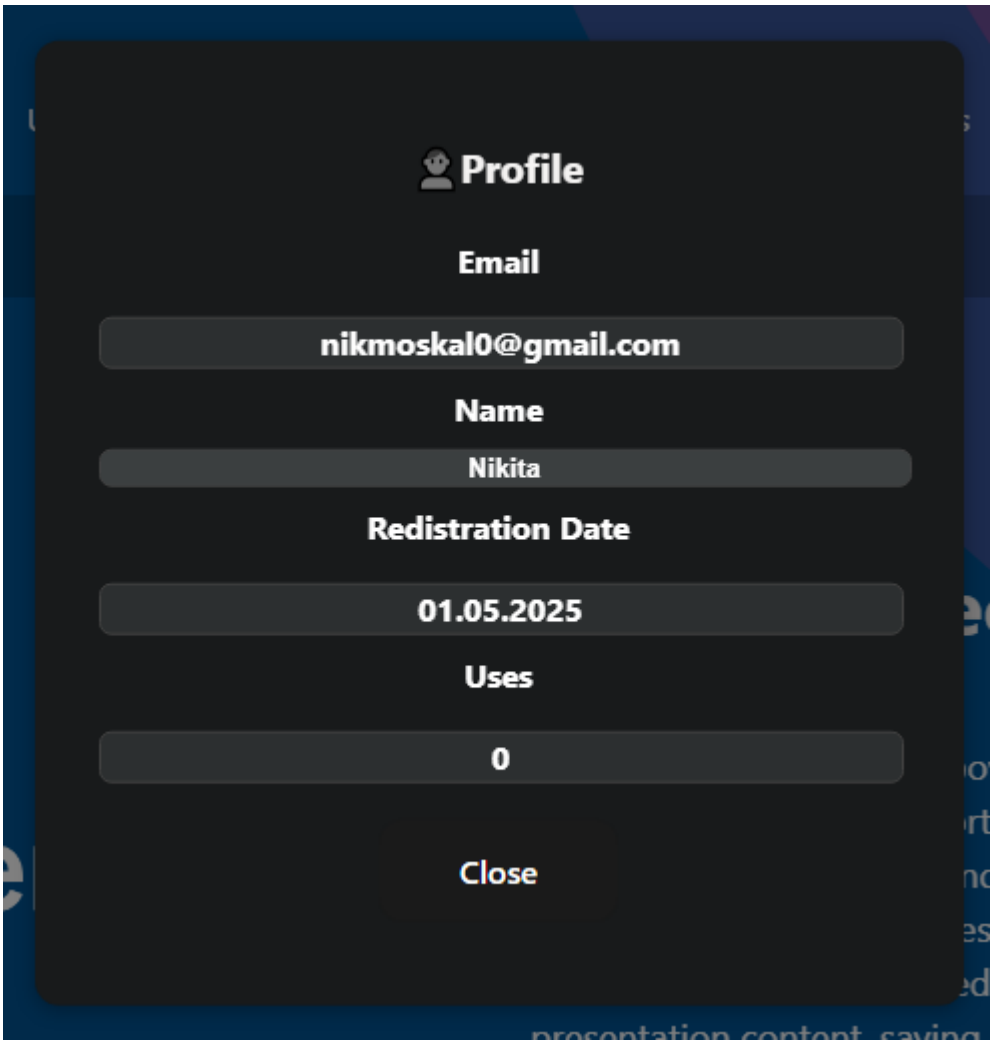


Рисунок 4.8 – Профіль

4.2 Створення системи обліку використань і преміум-доступу

В системі реалізовано облік використань генерації освітніх матеріалів. Кожен користувач має певну кількість доступних генерацій, яка зменшується після кожного запиту (рис. 4.9-4.10). Також передбачена можливість придбання додаткових використань через інтеграцію з платіжною системою Stripe. Статус «Premium Customer» можна придбати через відповідний функціонал додатка.

Рисунок 4.9 – Преміум клієнт

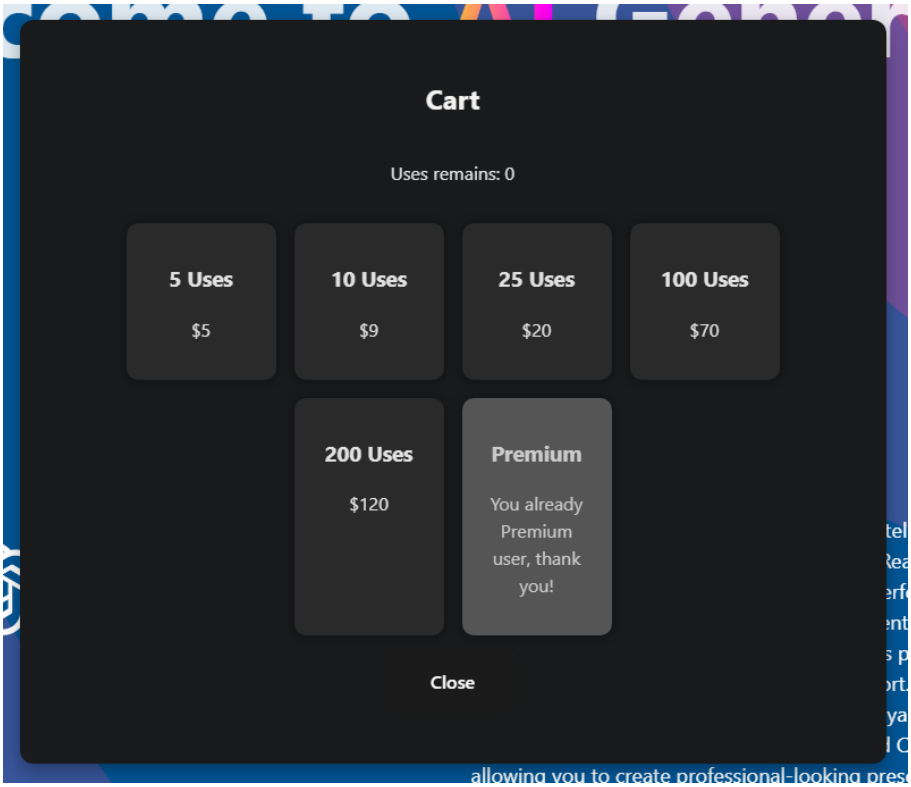


Рисунок 4.10 – Кошик

4.3 Генерація освітніх матеріалів

Основна функціональність системи полягає у створенні навчальних матеріалів на основі генеративних моделей штучного інтелекту. На даному етапі реалізовано:

– генерацію текстових документів у форматі DOCX за допомогою бібліотеки docx.js;

– генерацію презентацій у форматі PPTX за допомогою бібліотеки pptxgenjs.

Інтерфейс користувача дозволяє вводити тему або запит, на основі якого система автоматично створює відповідні освітні матеріали. На початку користувачеві пропонується поле для введення назви теми презентації (рис. 4.11). Важливо, щоб ця тема була сформульована чітко та точно – це значно підвищує якість згенерованого матеріалу, адже мовна модель базує свою генерацію саме на змістовності введеного запиту. Рекомендовано також додати короткий опис або уточнення, щоб система краще зрозуміла очікування користувача.

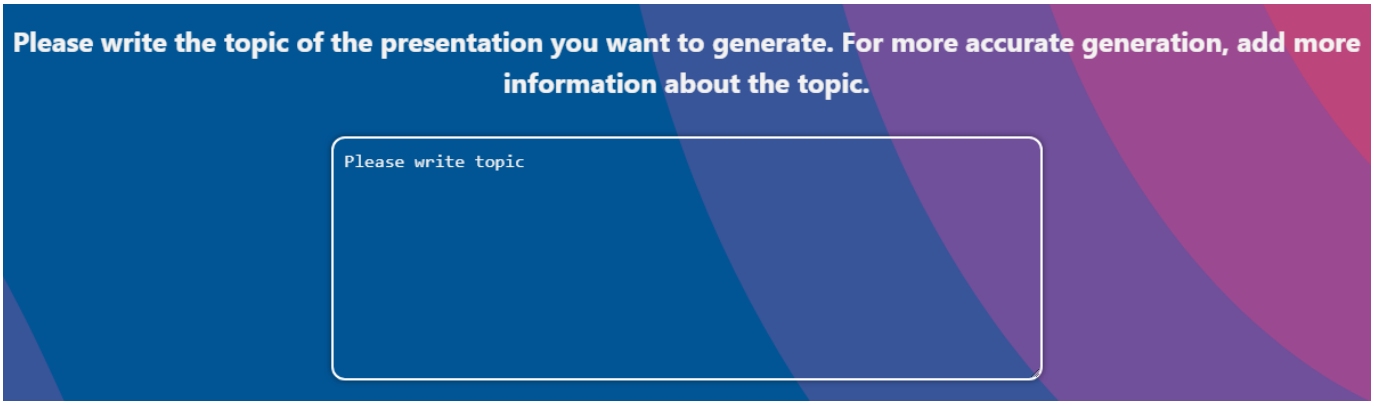


Рисунок 4.11 – Поле для вводу теми презентації

Далі користувач має змогу вибрати кількість слайдів – від 2 до 6 (рис. 4.12). Це дозволяє налаштувати обсяг презентації відповідно до потреб, складності теми або вимог користувача. Обирається той варіант, який користувач вважає оптимальним, залежно від того, наскільки глибоко потрібно розкрити тему презентації.

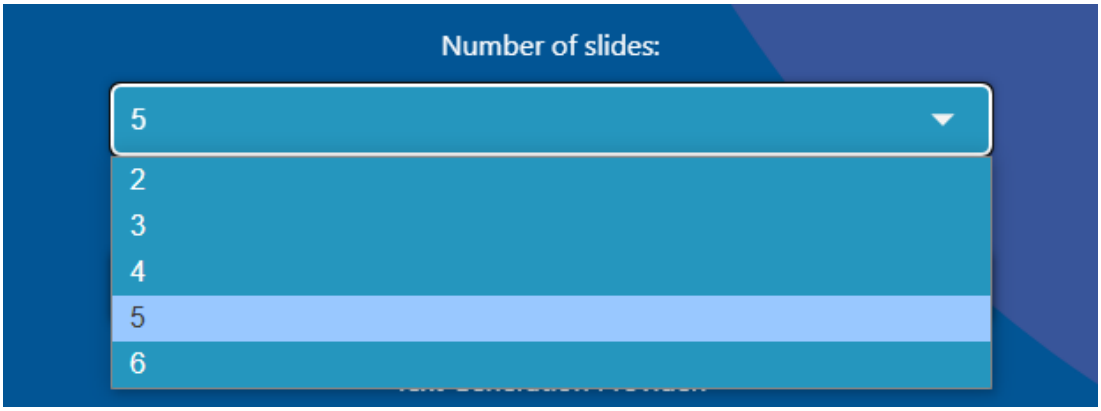


Рисунок 4.12 – Вибір кількості слайдів презентації

Наступним кроком є вибір постачальника зображень (рис. 4.13). Система підтримує два варіанти:

– Standard (Unsplash) – безкоштовний варіант, який здійснює пошук зображень із відкритої бібліотеки Unsplash;

– DALL-E – доступний лише для користувачів із преміум-доступом, дозволяє генерувати унікальні зображення за темою за допомогою нейромережі від OpenAI.

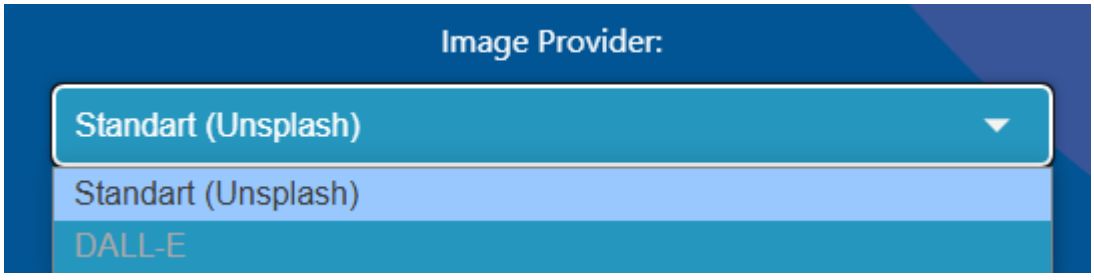


Рисунок 4.13 – Вибір постачальника зображень

Після цього користувач обирає постачальника генерації тексту OpenAI (рис. 4.14) – надає змогу вибрати між моделями:

– gpt-3.5 turbo – швидша й дешевша модель, добре підходить для простих освітніх тем;

– gpt-4 – потужніша модель, забезпечує глибший аналіз, кращу логіку та якісніший текст, особливо корисна для складних або вузькоспеціалізованих тем.

Cohere – альтернативна модель текстогенерації (рис. 4.15), яка теж здатна створювати презентаційний текст, хоча й поступається GPT-4 у глибокому розумінні контексту, може генерувати правильно тільки на англійській мові, на інших мовах можуть бути помилки.

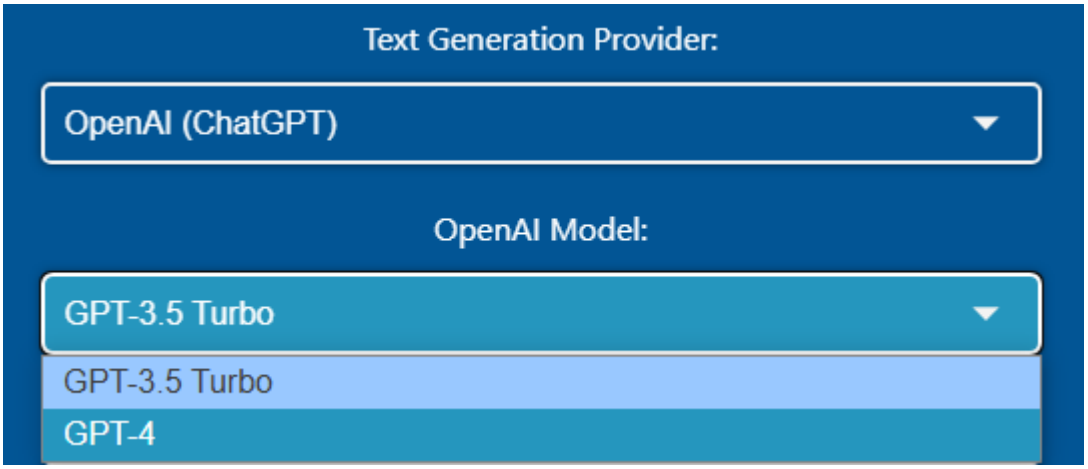


Рисунок 4.14 – Вибір постачальника тексту та моделі генерації від OpenAI

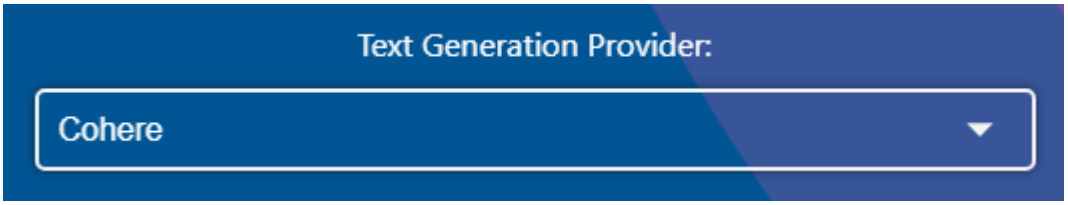


Рисунок 4.15 – Вибір постачальника тексту Cohere

Користувач також має змогу вибрати дизайн презентації (рис. 4.16-4.17):

– square – класичний прямокутний дизайн, із симетричними блоками тексту та зображень;

– circle – більш візуально динамічний стиль із круглими елементами, має декоративні кола у кутах слайдів.

Крім цього, можна активувати опцію «Round Image Corners», яка дозволяє візуально скруглити кути зображень.

Для зручності користувача при зміні типу дизайну на екрані одразу змінюється демонстраційне зображення, яке відображає приблизний вигляд майбутньої презентації. Тобто так користувач може приблизно побачити те що буде згенеровано і на основі побачено обрати те що йому найбільше подобається.

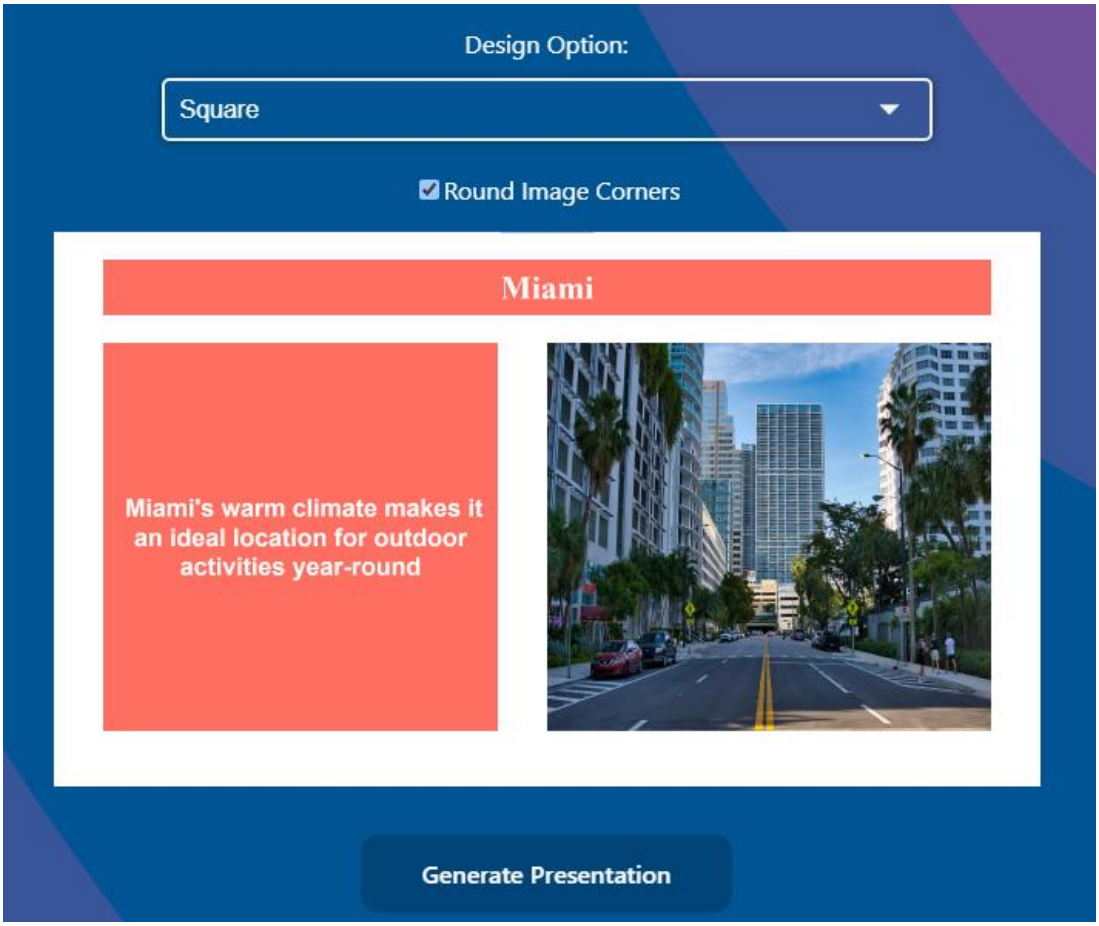


Рисунок 4.16 – Вибір дизайну та його приблизний вигляд

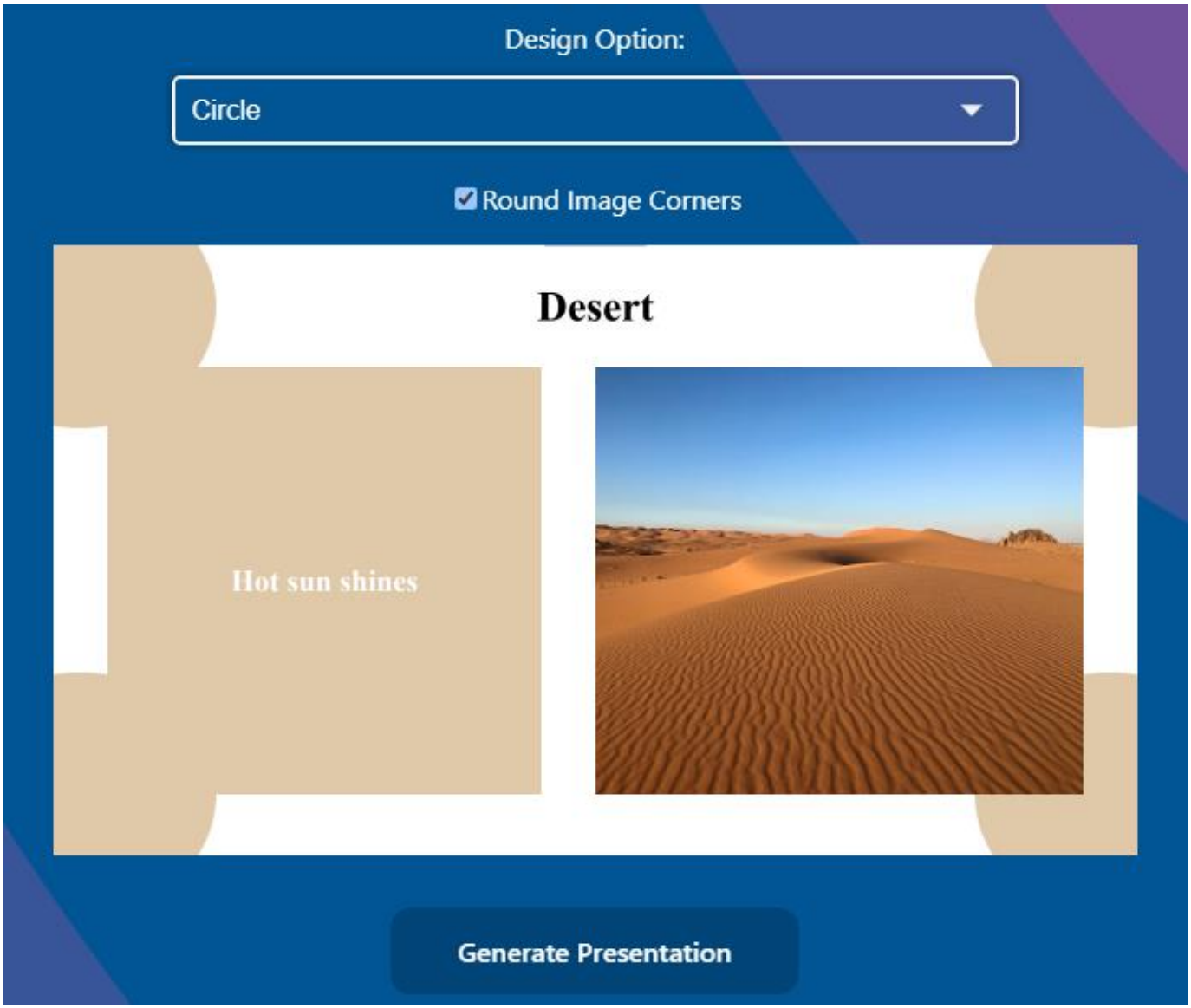


Рисунок 4.17 – Вибір дизайну та його приблизний вигляд

Отже на сторінці для генерації презентацій представлено інтерфейс користувача, через який здійснюється створення презентацій (рис. 4.18). Користувач має змогу задати тему, вибрати кількість слайдів, тип генерації тексту, джерело зображень, дизайн слайдів, а також додаткові візуальні налаштування. Усе це забезпечує практичне використання системи для створення якісних та індивідуалізованих освітніх матеріалів.

Рисунок 4.18 – Загальний вигляд сторінки генерації презентацій

Далі представлені всі елементи інтерфейсу для генерації документів у форматі Word. Першим елементом є текстове поле для вводу теми, яка слугує запитом для генерації вмісту (рис. 4.19). Залежно від обраного типу документа, до введеної теми висуваються різні вимоги:

– стандартний документ – передбачає вільну генерацію тексту на основі будь-якого запиту. Тут користувач може вводити будь-яку тему, і система згенерує відповідний інформаційний матеріал;

– лабораторна робота – потребує тем, які підходять до структури лабораторної роботи, наприклад: «Встановлення середовища Unity», «Основи роботи з масивами в C++» тощо. Генерований документ дотримується структури за ДСТУ: тема, мета, завдання, хід роботи, висновки;

– курсова робота – вимагає формулювання теми, яка відповідає академічним стандартам для курсових проєктів, наприклад: «Інтелектуальні системи в освіті», «Алгоритми обробки природної мови». Для курсових також передбачено автоматичне формування розділів.

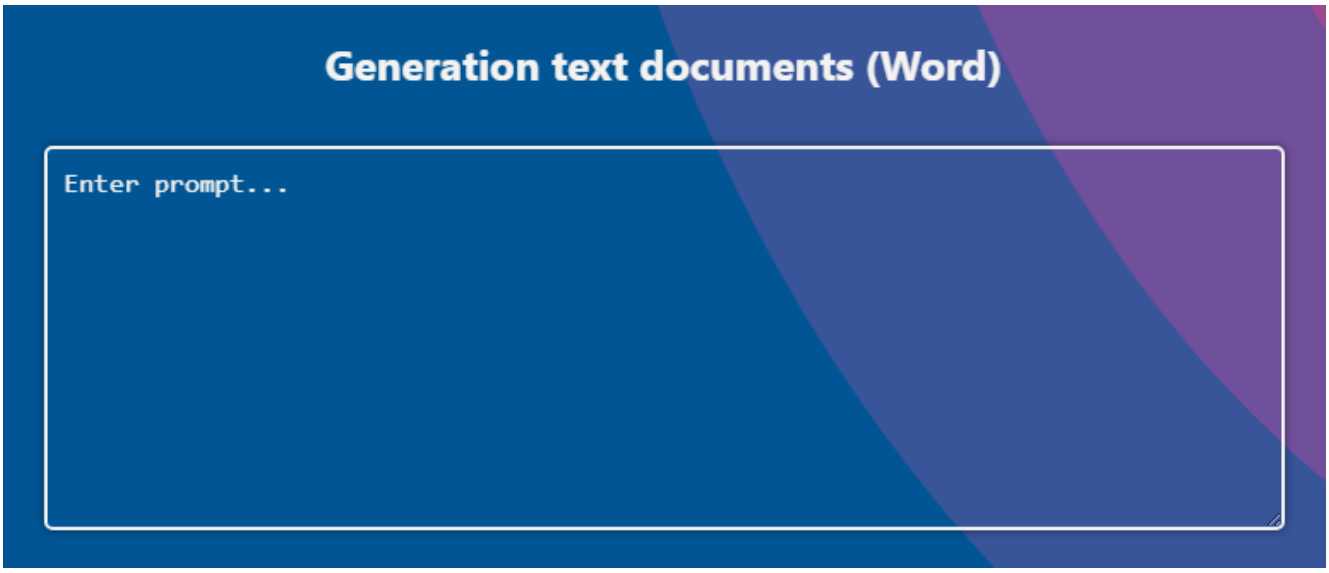


Рисунок 4.19 – Текстове поле для вводу теми

Наступним елементом є вибір типу документа (рис. 4.20):

– стандарт (простий документ);
– лабораторна робота;
– курсова робота.

При виборі лабораторної роботи з’являється додатковий параметр – мова генерації, тобто українська або англійська, за потреби користувач може обрати ту мову яка йому потрібна (рис. 4.21).

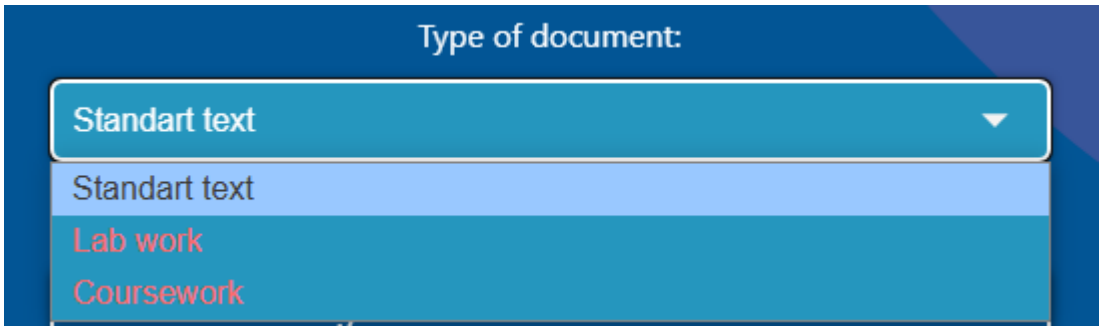


Рисунок 4.20 – Вибір типу документа

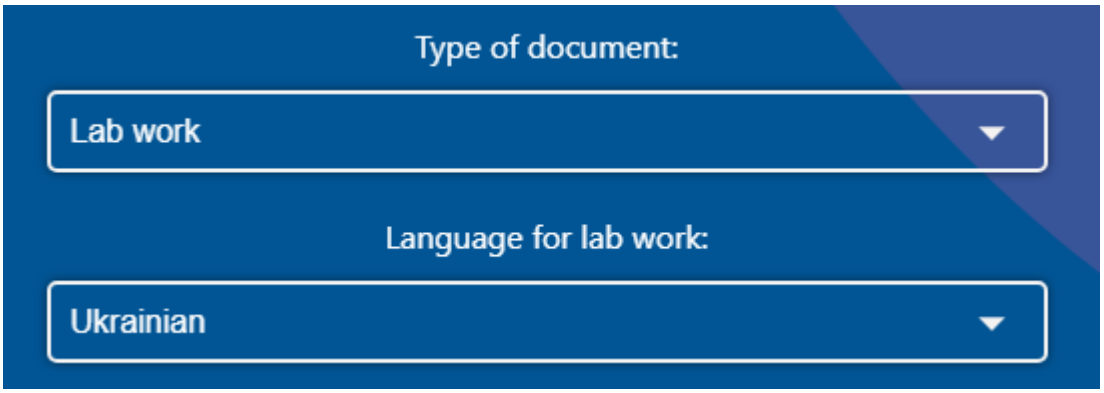


Рисунок 4.21 – Вибір мови документу

Далі користувач обирає форматування документа (рис. 4.22):

– стандартне – звичайний текстовий документ без додаткової структури;

– за ДСТУ – дотримання вимог до оформлення згідно з державним стандартом (наприклад, для лабораторних чи курсових робіт).

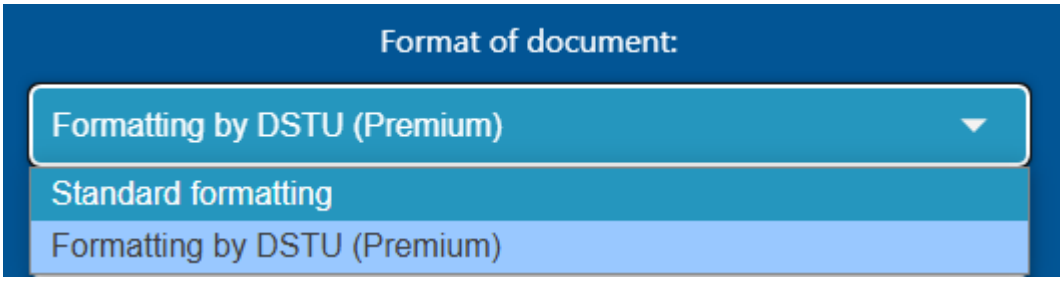


Рисунок 4.22 – Вибір формату

Ще один важливий параметр – модель генерації тексту (рис. 4.23). Доступні такі варіанти:

– gpt-3.5 Turbo – швидка, економна та доволі точна модель, добре підходить для стандартного тексту та лабораторних робіт;

– gpt-4 – більш просунута модель із кращою глибиною розуміння, здатна створювати структуровані й якісні академічні тексти, рекомендована для курсових;

– cohere – економічна альтернатива, підходить для генерації коротких текстів або тез, однак не підтримується для курсових робіт через обмеження в складності текстового аналізу.

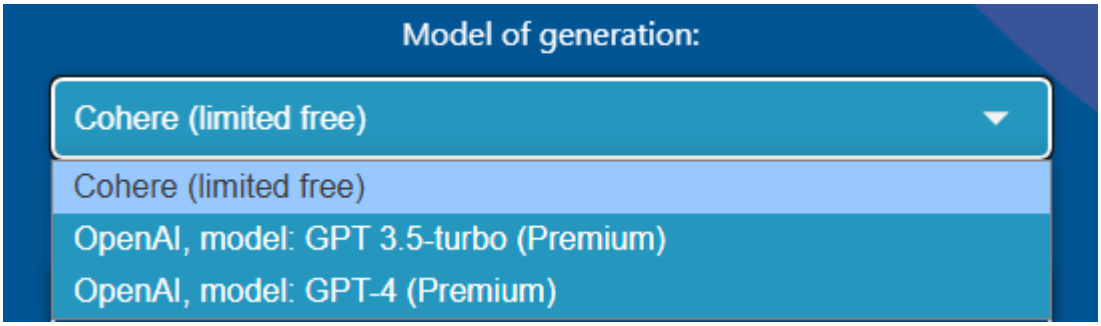


Рисунок 4.23 – Вибір моделі генерації

Завершальним блоком є вказання обсягу документа (рис. 4.24-4.25):

– для стандартного документу та лабораторної роботи можна задати кількість речень;

– для курсової роботи вказується приблизна кількість символів для кожного розділу, якщо треба більше тексту то можно обрати варіанти з 1000 або 2000 символів на кожен розділ.

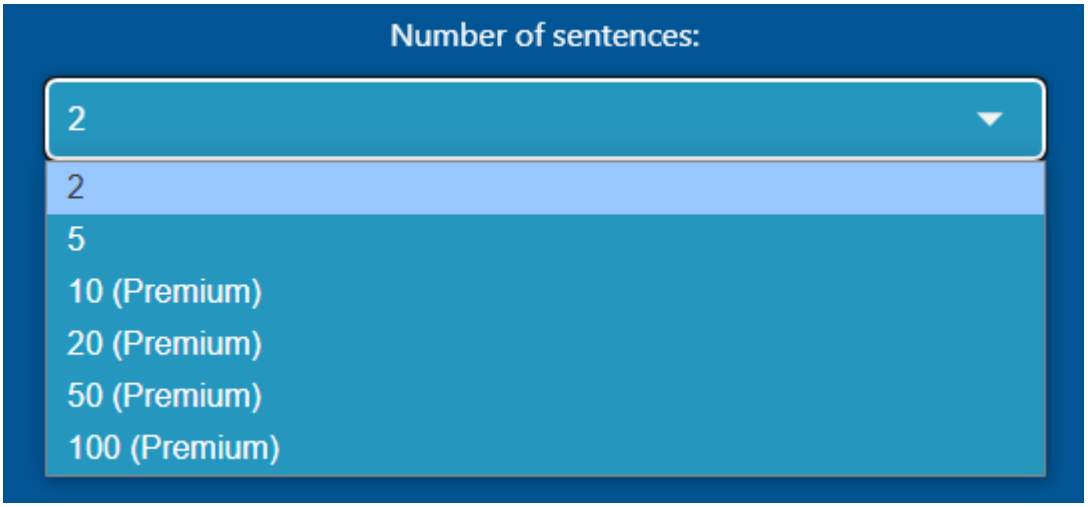


Рисунок 4.24 – Вибір кількості речень

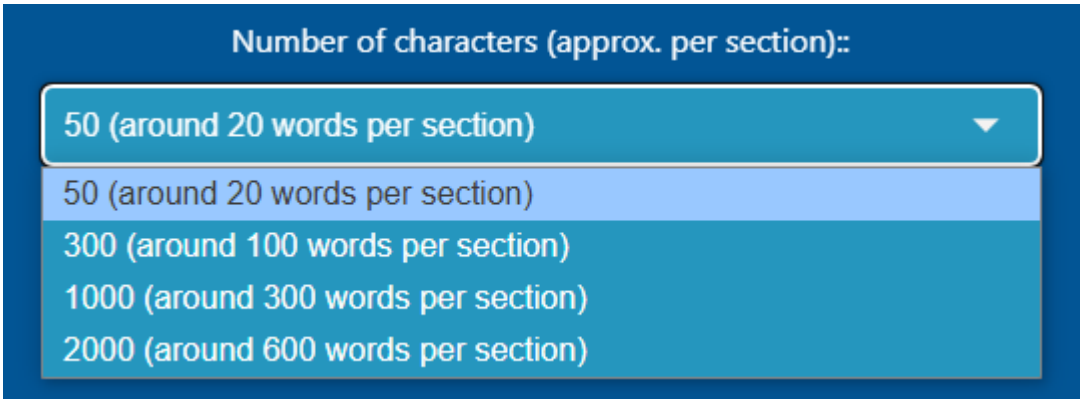


Рисунок 4.25 – Вибір приблизну кількості символів

Таким чином, система забезпечує широкий спектр налаштувань, що дозволяє

створювати документи різного призначення – від простих текстів до повноцінних курсових робіт, із підтримкою варіативного форматування, мов та моделей штучного інтелекту (рис. 4.26).

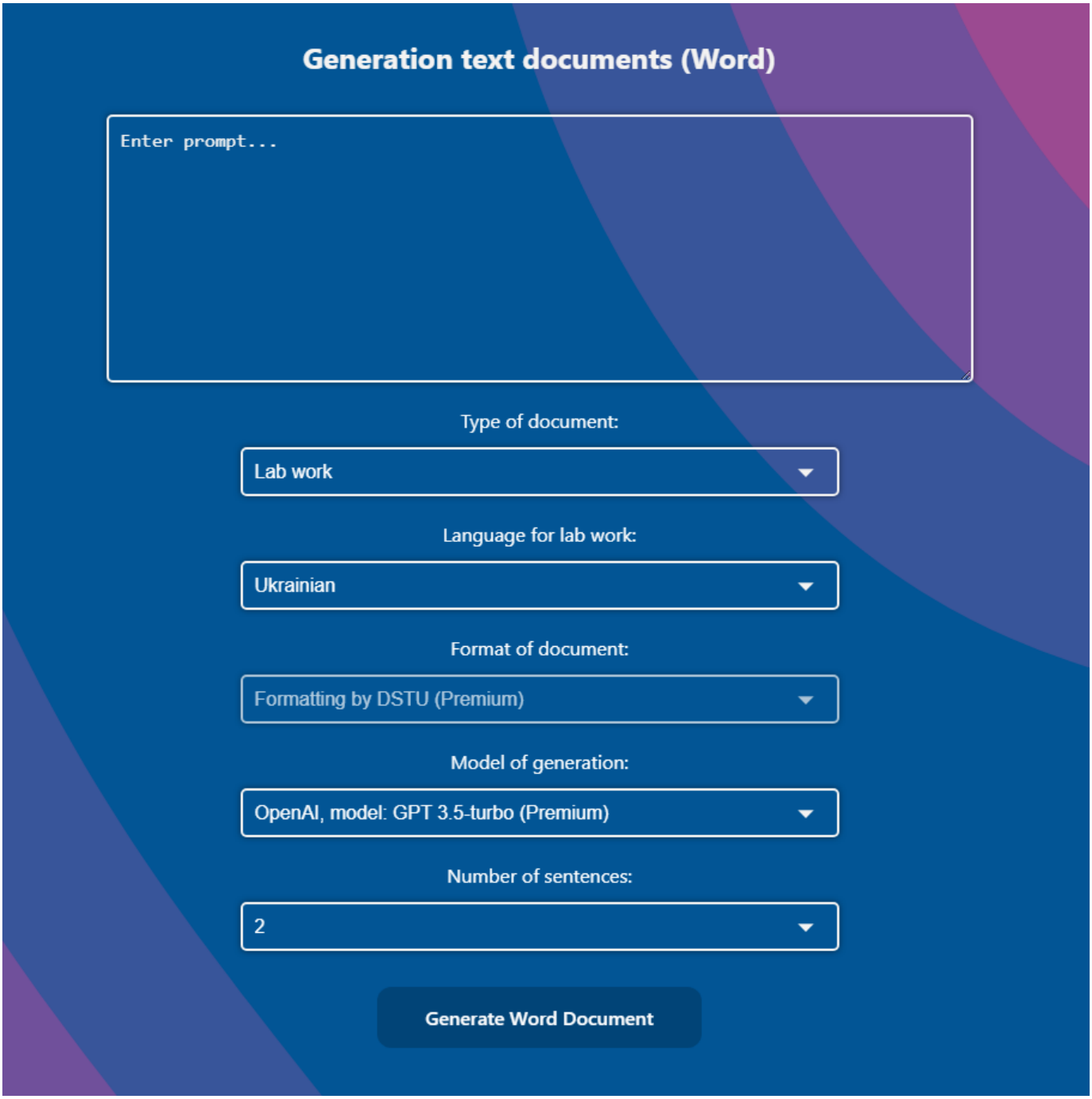


Рисунок 4.26 – Загальний вигляд сторінки генерації Word документів

4.4 Реалізація мультимовності інтерфейсу та зміна тем

Для забезпечення доступності додатку для користувачів з різних країн було інтегровано підтримку кількох мов (рис. 4.27-4.28). Перемикання між мовами (українська, англійська) здійснюється без перезавантаження сторінки завдяки використанню бібліотеки i18next.

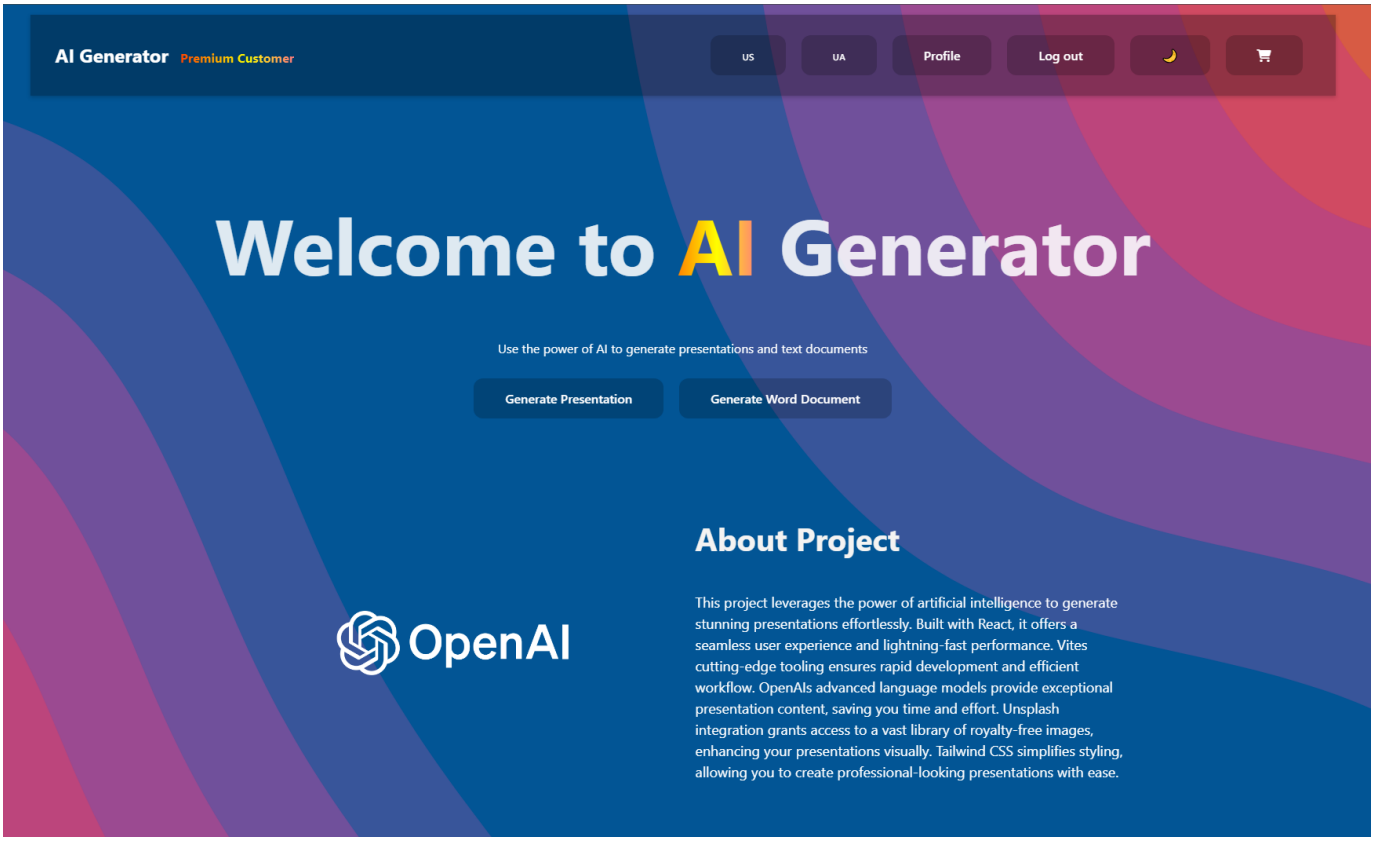


Рисунок 4.27 – Головна сторінка на англійській мові

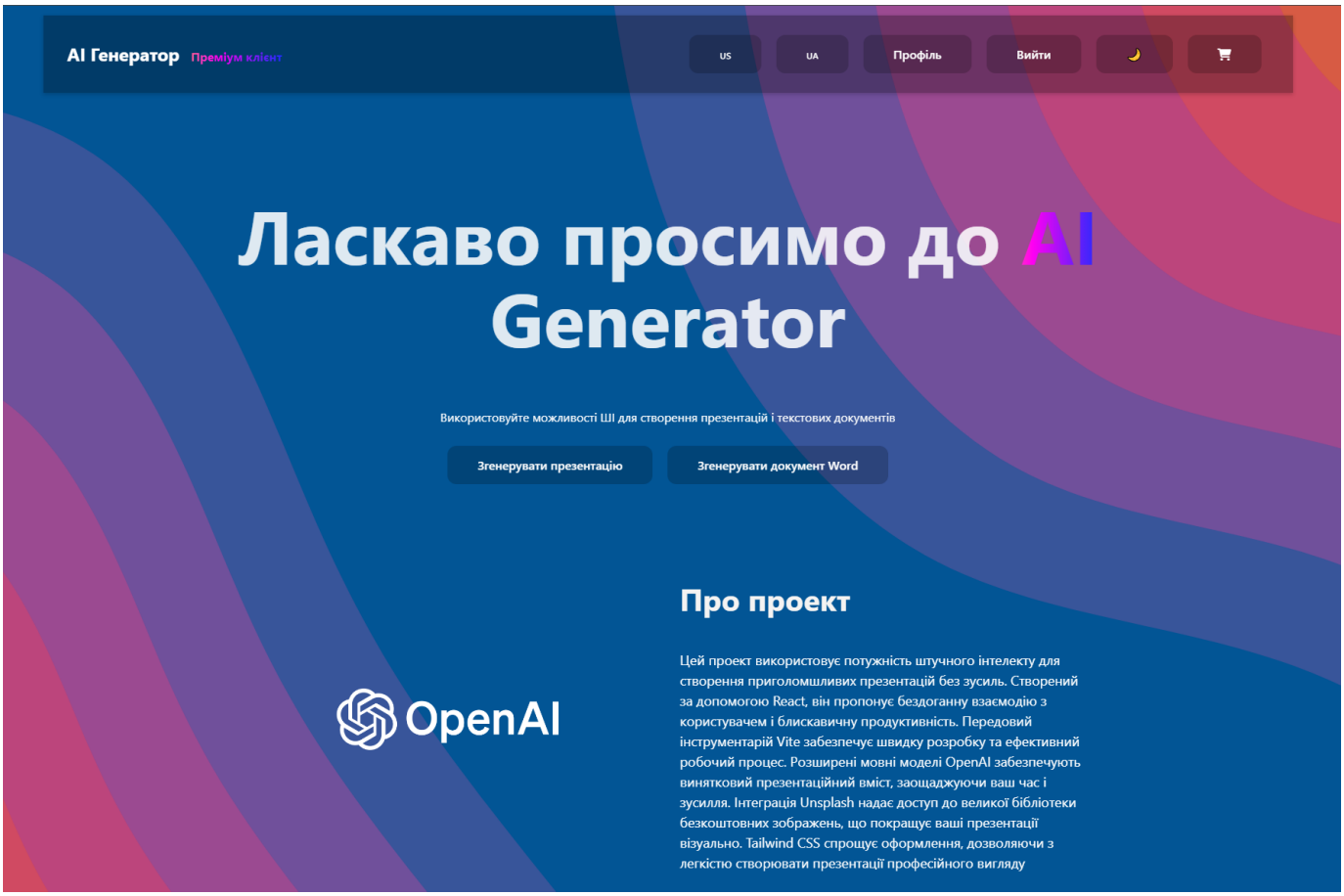


Рисунок 4.28 – Головна сторінка на українській мові

Було реалізовано перемикання теми інтерфейсу між світлою та темною (рис. 4.29-4.30). Вибір теми здійснюється користувачем вручну за допомогою відповідної кнопки в інтерфейсі, а обране значення зберігається в localStorage, що дозволяє зберегти тему при повторному відкритті сайту.

Зміна теми впливає не лише на кольори елементів інтерфейсу, а й на фонову графіку. Для фону використовується SVG-зображення: окреме для світлої (white.svg) та темної (dark.svg) тем. Компонент BackgroundWave динамічно підключає відповідний клас (.light або .dark), залежно від активної теми, що автоматично змінює фон сторінки.

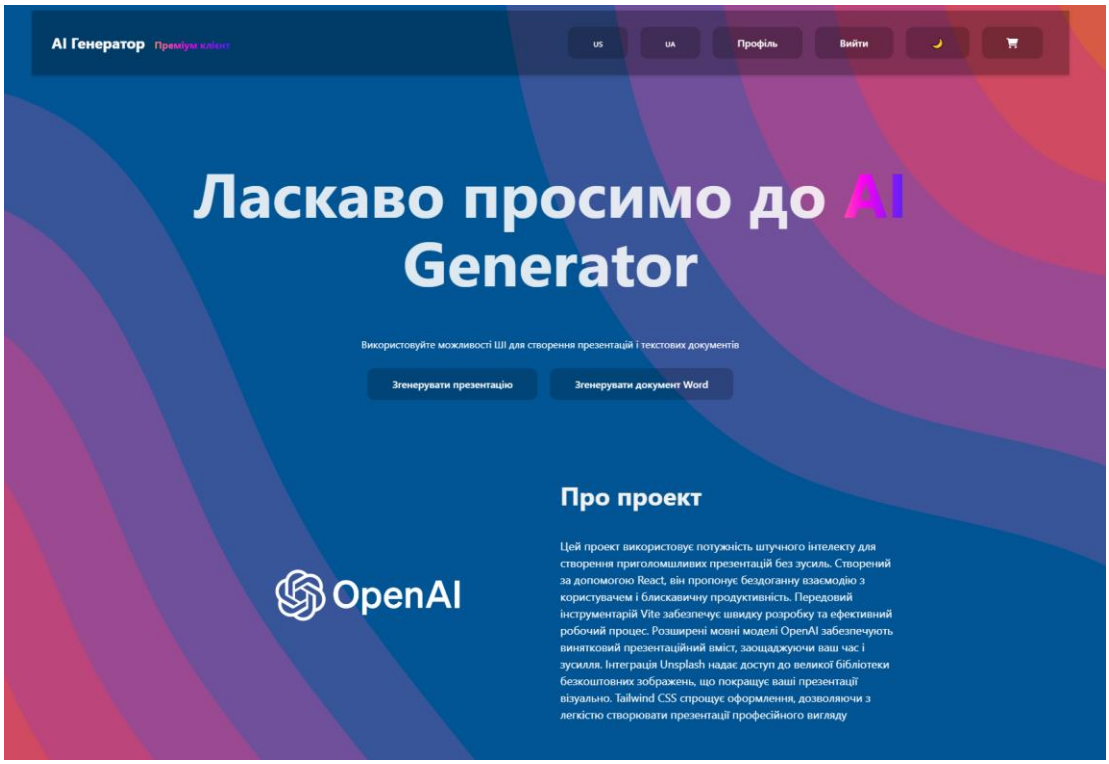


Рисунок 4.29 – Біла тема вебдодатку

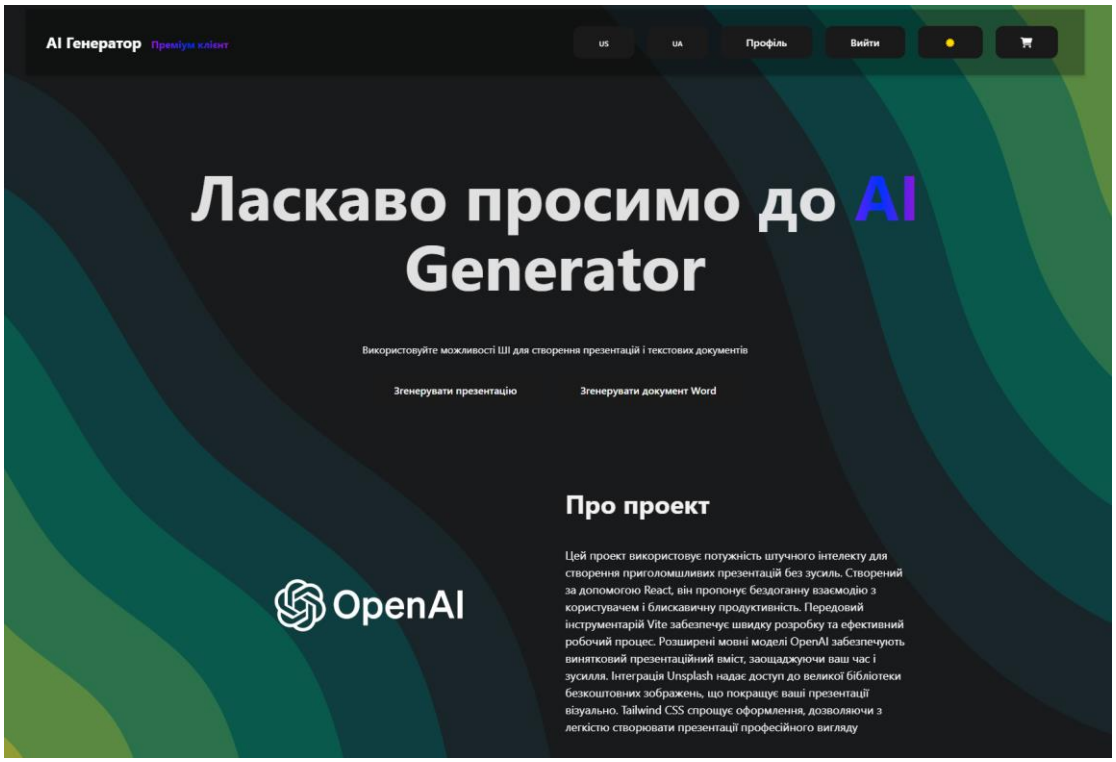


Рисунок 4.30 – Чорна тема вебдодатку

4.5 Інтеграція платіжної системи Stripe

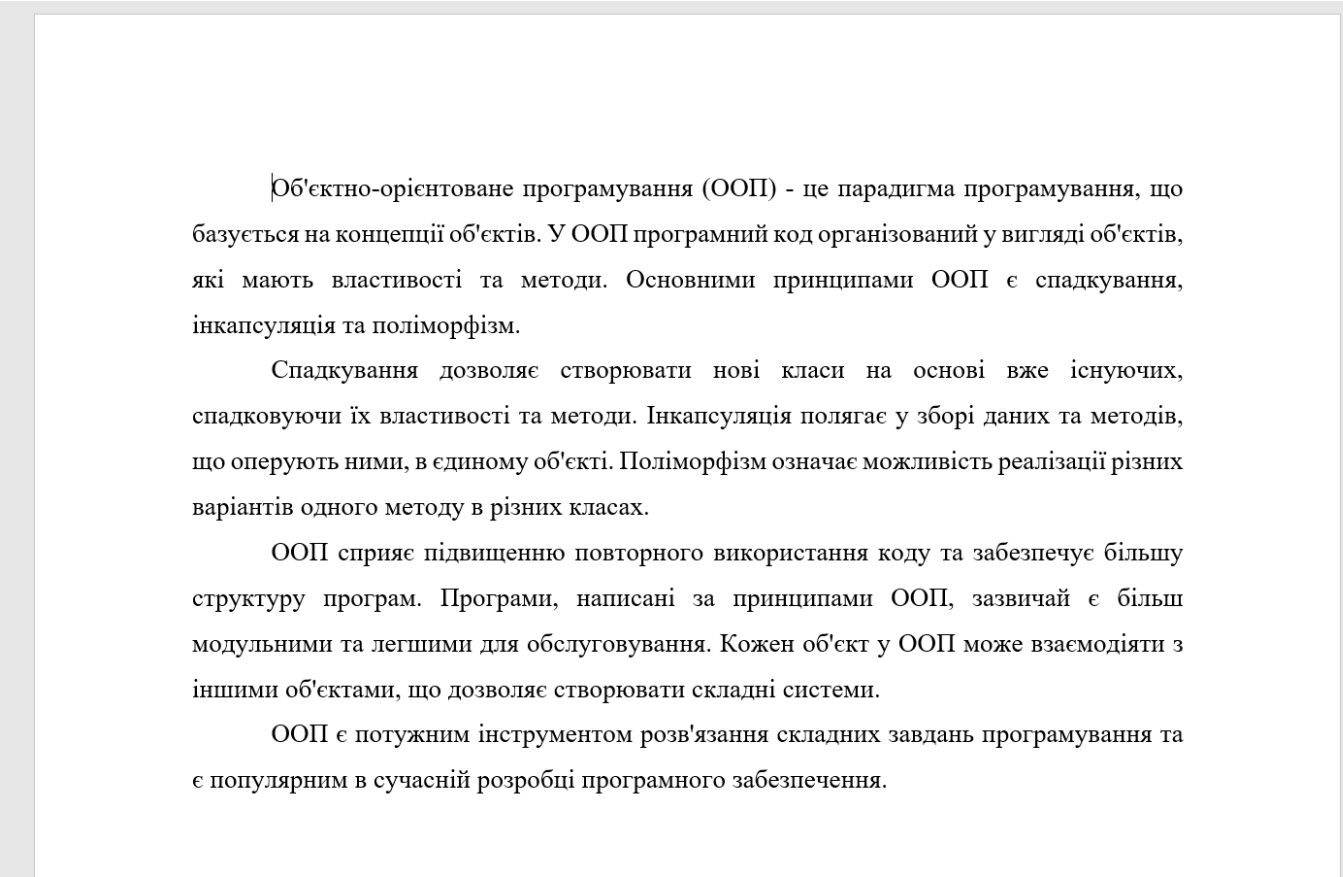
Було впроваджено повноцінну інтеграцію платіжної системи Stripe для можливості придбання додаткових пакетів використань та преміум-доступу. Реалізовано створення сесій платежу на бекенді та перенаправлення користувача на платіжну сторінку через фронтенд.

4.6 Результати тестування та попередній аналіз системи

На завершальному етапі було проведено тестування основних функціональностей:

– реєстрація та вхід користувачів;
– покупка використань;
– генерація документів;
– перемикання мов інтерфейсу.

Було проведено тестування розробленої системи. Зокрема, було згенеровано звичайний документ довжиною 10 речень з форматуванням за ДСТУ із текстовою відповіддю на запит: «По простому опиши основні принципи ООП» (рис. 4.31).


Об'єктно-орієнтоване програмування (ООП) - це парадигма програмування, що базується на концепції об'єктів. У ООП програмний код організований у вигляді об'єктів, які мають властивості та методи. Основними принципами ООП є спадкування, інкапсуляція та поліморфізм.

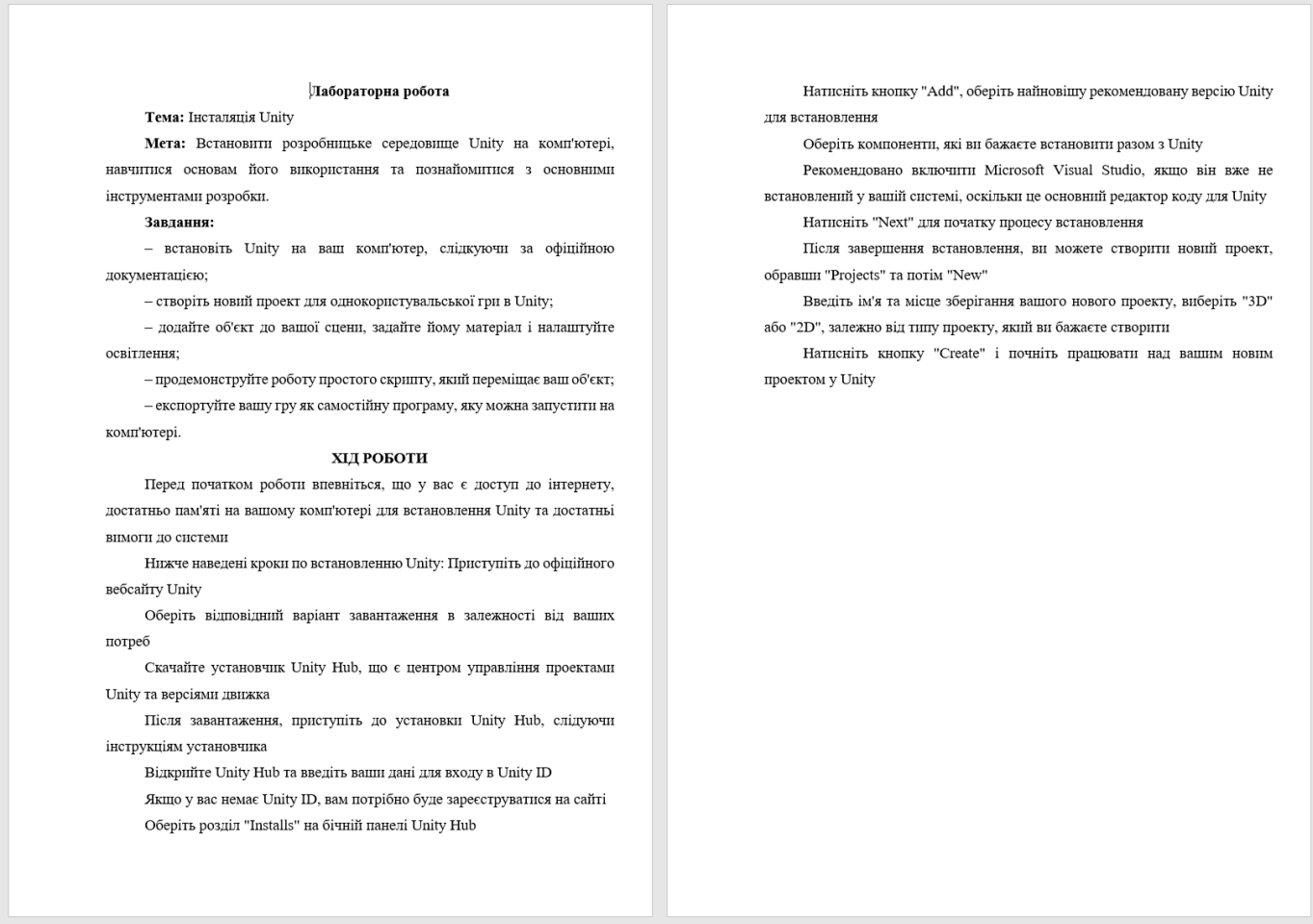
Спадкування дозволяє створювати нові класи на основі вже існуючих, спадковуючи їх властивості та методи. Інкапсуляція полягає у зборі даних та методів, що оперують ними, в єдиному об'єкті. Поліморфізм означає можливість реалізації різних варіантів одного методу в різних класах.

ООП сприяє підвищенню повторного використання коду та забезпечує більшу структуру програм. Програми, написані за принципами ООП, зазвичай є більш модульними та легшими для обслуговування. Кожен об'єкт у ООП може взаємодіяти з іншими об'єктами, що дозволяє створювати складні системи.

ООП є потужним інструментом розв'язання складних завдань програмування та є популярним в сучасній розробці програмного забезпечення.

Рисунок 4.31 – Звичайний документ

Далі була згенерована лабораторна робота на тему «Встановлення Unity» (рис. 4.32).



Лабораторна робота

Тема: Інсталяція Unity

Мета: Встановити розробницьке середовище Unity на комп'ютері, навчитися основам його використання та познайомитися з основними інструментами розробки.

Завдання:

– встановіть Unity на ваш комп'ютер, слідкуючи за офіційною документацією;

– створіть новий проект для однокористувальської гри в Unity;

– додайте об'єкт до вашої сцени, задайте йому матеріал і налаштуйте освітлення;

– продемонструйте роботу простого скрипту, який переміщає ваш об'єкт;

– експортуйте вашу гру як самостійну програму, яку можна запустити на комп'ютері.

ХІД РОБОТИ

Перед початком роботи впевніться, що у вас є доступ до інтернету, достатньо пам'яті на вашому комп'ютері для встановлення Unity та достатні вимоги до системи

Нижче наведені кроки по встановленню Unity: Приступіть до офіційного вебсайту Unity

Оберіть відповідний варіант завантаження в залежності від ваших потреб

Скачайте установчик Unity Hub, що є центром управління проектами Unity та версіями движка

Після завантаження, приступіть до установки Unity Hub, слідуючи інструкціям установчика

Відкрийте Unity Hub та введіть ваші дані для входу в Unity ID

Якщо у вас немає Unity ID, вам потрібно буде зареєструватися на сайті

Оберіть розділ "Installs" на бічній панелі Unity Hub

Натисніть кнопку "Add", оберіть найновішу рекомендовану версію Unity для встановлення

Оберіть компоненти, які ви бажаєте встановити разом з Unity

Рекомендовано включити Microsoft Visual Studio, якщо він вже не встановлений у вашій системі, оскільки це основний редактор коду для Unity

Натисніть "Next" для початку процесу встановлення

Після завершення встановлення, ви можете створити новий проект, обравши "Projects" та потім "New"

Введіть ім'я та місце зберігання вашого нового проекту, виберіть "3D" або "2D", залежно від типу проекту, який ви бажаєте створити

Натисніть кнопку "Create" і почніть працювати над вашим новим проектом у Unity

Рисунок 4.32 – Приклад згенерованого документу типу «Лабораторна робота» на тему «Встановлення Unity»

Також для тестування були згенеровано презентацію, англійською мовою яка складається з 7 слайдів на тему «New York City» (рис. 4.33). В презентації постачальником зображення є Unsplash, модель для генерації тексту обрана GPT-3.5 Turbo.

Рисунок 4.33 – Приклад згенерованої презентації на тему «New York City»

Окрім того, було згенеровано курсову роботу на тему «Вебзастосунок для візуального аналізу кількості калорій використовуючи ШІ» українською мовою. Ця тема була обрана через її складність. Система згенерувала зміст, вступ, три розділи з 2–3 підпунктами кожен, текст до підпунктів і вступу, а також висновки з відповідним текстом (рис. 4.34–4.41).

ЗМІСТ

ВСТУП

У сучасному світі серед людей спостерігається все більше уваги до свого здоров'я, до правильного харчування. Одним із ключових аспектів збалансованої дієти є урахування кількості калорій у прийманому їжі. І тут виникає проблема: більшість людей неправильно визначає калорійність своєї їжі, що веде до порушення балансу вживаних і витрачених калорій.

На сучасному ринку вже є додатки, присвячені візуальному аналізу калорійності продуктів, але вони мають свої обмеження. Часто такі додатки на калорії потребують ручного вводу інформації або мають обмежений банк даних, що не дозволяє точно оцінити калорійність.

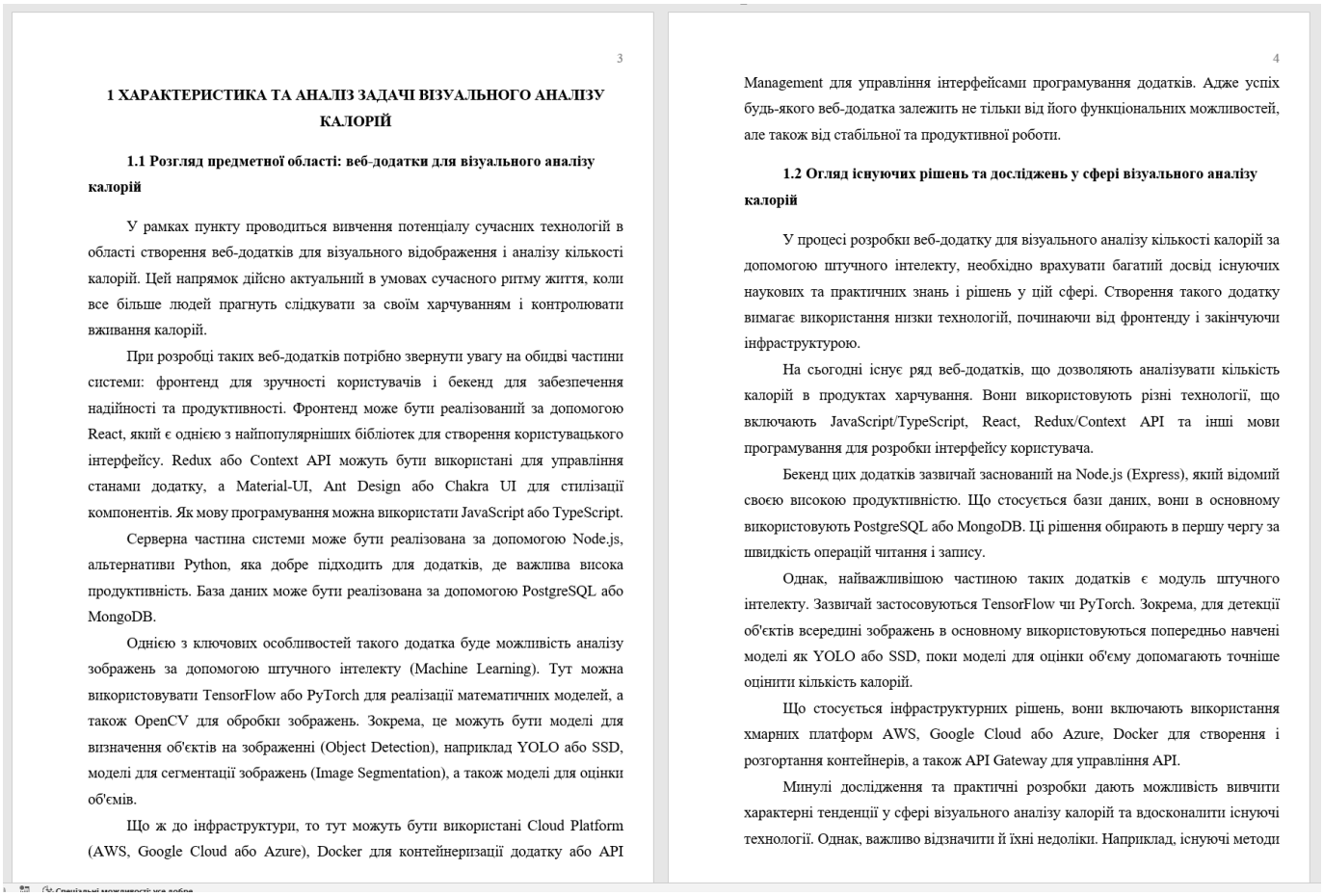
Розробка веб-додатка для візуального аналізу калорій з використанням штучного інтелекту (ШІ) може вирішити ці обмеження, оскільки ШІ може натренувати визначати калорії на основі фотографії.

В роботі будуть використані сучасні технології, такі як React, Redux/Context API, Material-UI/Ant Design/Chakra UI, JavaScript/TypeScript, Node.js (Express), PostgreSQL, MongoDB, TensorFlow/PyTorch, OpenCV, облачній платформи(AWS/Google Cloud/Azure), Docker, а також використання API Management (API Gateway).

Такий додаток має великий потенціал та може знайти широке застосування в галузі здорового харчування, фітнесу, дієтології тощо. Актуальність цієї теми є високою та зростає завдяки зростаючому інтересу до здорового способу життя.

Отже, вступ до курсової роботи має на меті провести дослідження на актуальність розробки веб-додатку для візуального аналізу калорій з використанням ШІ, аналізувати та вибирати найвідповідніші технології для реалізації концепції.

Рисунок 4.34 – Приклад згенерованої курсової роботи, зміст та вступ



3

1 ХАРАКТЕРИСТИКА ТА АНАЛІЗ ЗАДАЧІ ВІЗУАЛЬНОГО АНАЛІЗУ КАЛОРІЙ

1.1 Розгляд предметної області: веб-додатки для візуального аналізу калорій

У рамках пункту проводиться вивчення потенціалу сучасних технологій в області створення веб-додатків для візуального відображення і аналізу кількості калорій. Цей напрямок дійсно актуальний в умовах сучасного ритму життя, коли все більше людей прагнуть слідкувати за своїм харчуванням і контролювати вживання калорій.

При розробці таких веб-додатків потрібно звернути увагу на обидві частини системи: фронтенд для зручності користувачів і бекенд для забезпечення надійності та продуктивності. Фронтенд може бути реалізований за допомогою React, який є однією з найпопулярніших бібліотек для створення користувацького інтерфейсу. Redux або Context API можуть бути використані для управління станами додатку, а Material-UI, Ant Design або Chakra UI для стилізації компонентів. Як мову програмування можна використати JavaScript або TypeScript.

Серверна частина системи може бути реалізована за допомогою Node.js, альтернативи Python, яка добре підходить для додатків, де важлива висока продуктивність. База даних може бути реалізована за допомогою PostgreSQL або MongoDB.

Однією з ключових особливостей такого додатка буде можливість аналізу зображень за допомогою штучного інтелекту (Machine Learning). Тут можна використовувати TensorFlow або PyTorch для реалізації математичних моделей, а також OpenCV для обробки зображень. Зокрема, це можуть бути моделі для визначення об'єктів на зображенні (Object Detection), наприклад YOLO або SSD, моделі для сегментації зображень (Image Segmentation), а також моделі для оцінки об'ємів.

Що ж до інфраструктури, то тут можуть бути використані Cloud Platform (AWS, Google Cloud або Azure), Docker для контейнеризації додатку або API

4

Management для управління інтерфейсами програмування додатків. Адже успіх будь-якого веб-додатка залежить не тільки від його функціональних можливостей, але також від стабільної та продуктивної роботи.

1.2 Огляд існуючих рішень та досліджень у сфері візуального аналізу калорій

У процесі розробки веб-додатку для візуального аналізу кількості калорій за допомогою штучного інтелекту, необхідно врахувати багатий досвід існуючих наукових та практичних знань і рішень у цій сфері. Створення такого додатку вимагає використання низки технологій, починаючи від фронтенду і закінчуючи інфраструктурою.

На сьогодні існує ряд веб-додатків, що дозволяють аналізувати кількість калорій в продуктах харчування. Вони використовують різні технології, що включають JavaScript/TypeScript, React, Redux/Context API та інші мови програмування для розробки інтерфейсу користувача.

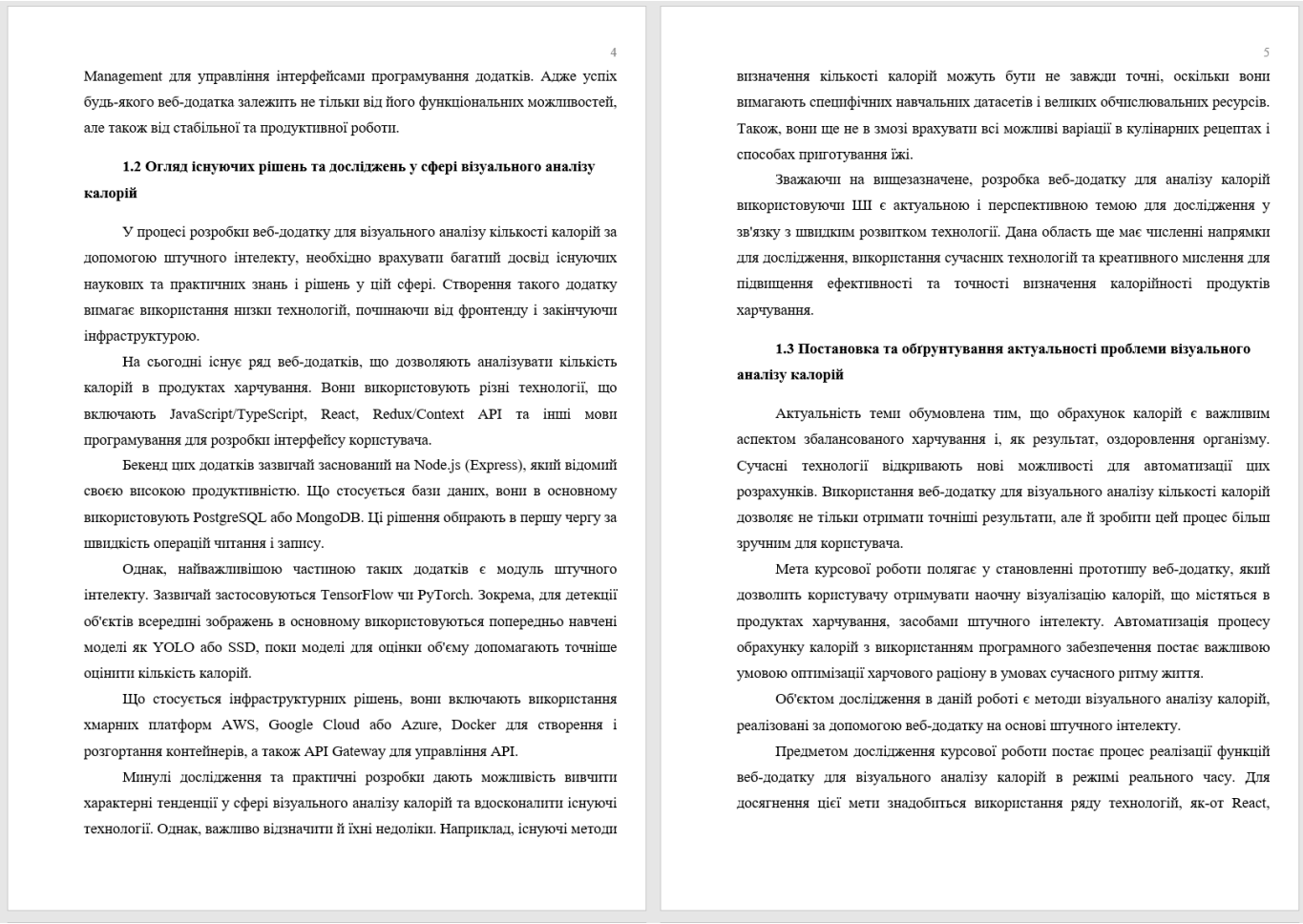
Бекенд цих додатків зазвичай заснований на Node.js (Express), який відомий своєю високою продуктивністю. Що стосується бази даних, вони в основному використовують PostgreSQL або MongoDB. Ці рішення обирають в першу чергу за швидкість операцій читання і запису.

Однак, найважливішою частиною таких додатків є модуль штучного інтелекту. Зазвичай застосовуються TensorFlow чи PyTorch. Зокрема, для детекції об'єктів всередині зображень в основному використовуються попередньо навчені моделі як YOLO або SSD, поки моделі для оцінки об'єму допомагають точніше оцінити кількість калорій.

Що стосується інфраструктурних рішень, вони включають використання хмарних платформ AWS, Google Cloud або Azure, Docker для створення і розгортання контейнерів, а також API Gateway для управління API.

Минулі дослідження та практичні розробки дають можливість вивчити характерні тенденції у сфері візуального аналізу калорій та вдосконалити існуючі технології. Однак, важливо відзначити й їхні недоліки. Наприклад, існуючі методи

Рисунок 4.35 – Приклад згенерованої курсової роботи, перший розділ, перший та другий підрозділ



4

Management для управління інтерфейсами програмування додатків. Адже успіх будь-якого веб-додатка залежить не тільки від його функціональних можливостей, але також від стабільної та продуктивної роботи.

1.2 Огляд існуючих рішень та досліджень у сфері візуального аналізу калорій

У процесі розробки веб-додатку для візуального аналізу кількості калорій за допомогою штучного інтелекту, необхідно врахувати багатий досвід існуючих наукових та практичних знань і рішень у цій сфері. Створення такого додатку вимагає використання низки технологій, починаючи від фронтенду і закінчуючи інфраструктурою.

На сьогодні існує ряд веб-додатків, що дозволяють аналізувати кількість калорій в продуктах харчування. Вони використовують різні технології, що включають JavaScript/TypeScript, React, Redux/Context API та інші мови програмування для розробки інтерфейсу користувача.

Бекенд цих додатків зазвичай заснований на Node.js (Express), який відомий своєю високою продуктивністю. Що стосується бази даних, вони в основному використовують PostgreSQL або MongoDB. Ці рішення обирають в першу чергу за швидкість операцій читання і запису.

Однак, найважливішою частиною таких додатків є модуль штучного інтелекту. Зазвичай застосовуються TensorFlow чи PyTorch. Зокрема, для детекції об'єктів всередині зображень в основному використовуються попередньо навчені моделі як YOLO або SSD, поки моделі для оцінки об'єму допомагають точніше оцінити кількість калорій.

Що стосується інфраструктурних рішень, вони включають використання хмарних платформ AWS, Google Cloud або Azure, Docker для створення і розгортання контейнерів, а також API Gateway для управління API.

Минулі дослідження та практичні розробки дають можливість вивчити характерні тенденції у сфері візуального аналізу калорій та вдосконалити існуючі технології. Однак, важливо відзначити й їхні недоліки. Наприклад, існуючі методи

5

визначення кількості калорій можуть бути не завжди точні, оскільки вони вимагають специфічних навчальних датасетів і великих обчислювальних ресурсів. Також, вони ще не в змозі врахувати всі можливі варіації в кулінарних рецептах і способах приготування їжі.

Зважаючи на вищезазначене, розробка веб-додатку для аналізу калорій використовуючи ШІ є актуальною і перспективною темою для дослідження у зв'язку з швидким розвитком технології. Дана область ще має численні напрямки для дослідження, використання сучасних технологій та креативного мислення для підвищення ефективності та точності визначення калорійності продуктів харчування.

1.3 Постановка та обґрунтування актуальності проблеми візуального аналізу калорій

Актуальність теми обумовлена тим, що обрахунок калорій є важливим аспектом збалансованого харчування і, як результат, оздоровлення організму. Сучасні технології відкривають нові можливості для автоматизації цих розрахунків. Використання веб-додатку для візуального аналізу кількості калорій дозволяє не тільки отримати точніші результати, але й зробити цей процес більш зручним для користувача.

Мета курсової роботи полягає у становленні прототипу веб-додатку, який дозволить користувачу отримувати наочну візуалізацію калорій, що містяться в продуктах харчування, засобами штучного інтелекту. Автоматизація процесу обрахунку калорій з використанням програмного забезпечення постає важливою умовою оптимізації харчового раціону в умовах сучасного ритму життя.

Об'єктом дослідження в даній роботі є методи візуального аналізу калорій, реалізовані за допомогою веб-додатку на основі штучного інтелекту.

Предметом дослідження курсової роботи постає процес реалізації функцій веб-додатку для візуального аналізу калорій в режимі реального часу. Для досягнення цієї мети знадобиться використання ряду технологій, як-от React,

Рисунок 4.36 – Приклад згенерованої курсової роботи, перший розділ, другий та третій підрозділи

Redux/Context API, Material-UI/Ant Design/Chakra UI, JavaScript/TypeScript, Node.js (Express), PostgreSQL, MongoDB, TensorFlow/PyTorch, OpenCV.

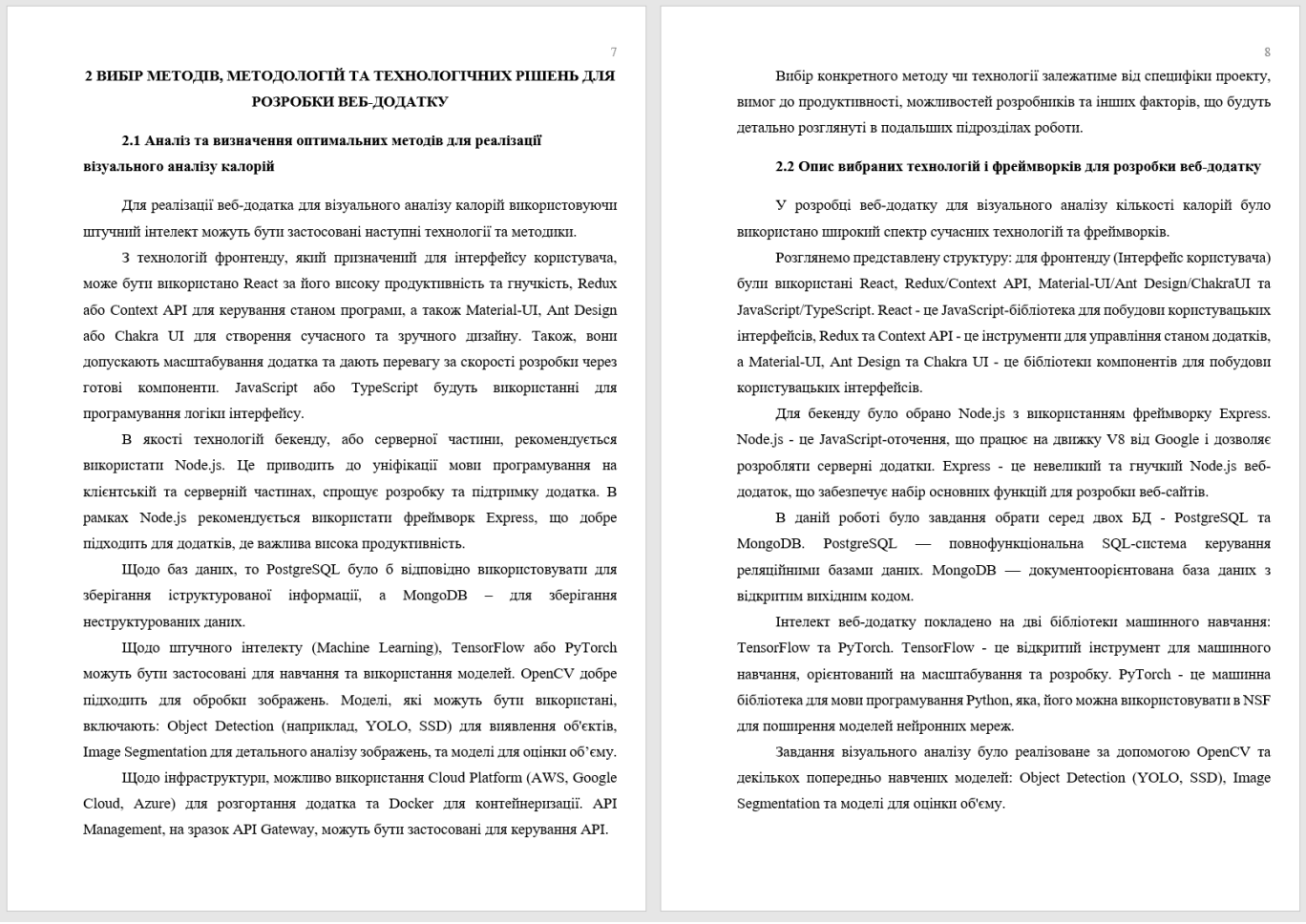
До завдань курсової роботи входить створення вищезгаданого веб-додатку з використанням розроблених та відтестованих алгоритмів візуального аналізу. Цей процес передбачає реалізацію трьох головних етапів: фронтенд-частини (з використанням React, Redux/Context API, Material-UI/Ant Design/Chakra UI, JavaScript/TypeScript), бекенд-частини (Node.js (Express)), а також баз даних (PostgreSQL, MongoDB).

Система має бути втілена за допомогою штучного інтелекту (Machine Learning) з використанням TensorFlow/PyTorch, OpenCV. При цьому, будуть використовуватись попередньо навчені моделі, такі як Object Detection (наприклад, YOLO, SSD), Image Segmentation, моделі для оцінки об'єму.

Інфраструктура проекту передбачає використання таких технологій, як Cloud Platform (AWS/Google Cloud/Azure), Docker, API Management (API Gateway). Ці технології дозволять ефективно зберігати дані, забезпечувати високу швидкість роботи та захищати інформацію.

Отже, дослідження актуальності проблеми візуального аналізу калорій є необхідним для створення і реалізації веб-додатку, який дозволить користувачам отримувати реальні дані стосовно кількості калорій у їх їжі, що набагато полегшить процес контролю збалансованості їх харчування.

Рисунок 4.37 – Приклад згенерованої курсової роботи, перший розділ та частина третього підрозділу



7

2 ВИБІР МЕТОДІВ, МЕТОДОЛОГІЙ ТА ТЕХНОЛОГІЧНИХ РІШЕНЬ ДЛЯ РОЗРОБКИ ВЕБ-ДОДАТКУ

2.1 Аналіз та визначення оптимальних методів для реалізації візуального аналізу калорій

Для реалізації веб-додатка для візуального аналізу калорій використовуючи штучний інтелект можуть бути застосовані наступні технології та методики.

З технологій фронтенду, який призначений для інтерфейсу користувача, може бути використано React за його високу продуктивність та гнучкість, Redux або Context API для керування станом програми, а також Material-UI, Ant Design або Chakra UI для створення сучасного та зручного дизайну. Також, вони допускають масштабування додатка та дають перевагу за скорості розробки через готові компоненти. JavaScript або TypeScript будуть використанні для програмування логіки інтерфейсу.

В якості технологій бекенду, або серверної частини, рекомендується використати Node.js. Це приводить до уніфікації мови програмування на клієнтській та серверній частинах, спрощує розробку та підтримку додатка. В рамках Node.js рекомендується використати фреймворк Express, що добре підходить для додатків, де важлива висока продуктивність.

Щодо баз даних, то PostgreSQL було б відповідно використовувати для зберігання іструктурованої інформації, а MongoDB – для зберігання неструктурованих даних.

Щодо штучного інтелекту (Machine Learning), TensorFlow або PyTorch можуть бути застосовані для навчання та використання моделей. OpenCV добре підходить для обробки зображень. Моделі, які можуть бути використані, включають: Object Detection (наприклад, YOLO, SSD) для виявлення об'єктів, Image Segmentation для детального аналізу зображень, та моделі для оцінки об’єму.

Щодо інфраструктури, можливо використання Cloud Platform (AWS, Google Cloud, Azure) для розгортання додатка та Docker для контейнеризації. API Management, на зразок API Gateway, можуть бути застосовані для керування API.

8

Вибір конкретного методу чи технології залежатиме від специфіки проекту, вимог до продуктивності, можливостей розробників та інших факторів, що будуть детально розглянуті в подальших підрозділах роботи.

2.2 Опис вибраних технологій і фреймворків для розробки веб-додатку

У розробці веб-додатку для візуального аналізу кількості калорій було використано широкий спектр сучасних технологій та фреймворків.

Розглянемо представлену структуру: для фронтенду (Інтерфейс користувача) були використані React, Redux/Context API, Material-UI/Ant Design/ChakraUI та JavaScript/TypeScript. React - це JavaScript-бібліотека для побудови користувацьких інтерфейсів, Redux та Context API - це інструменти для управління станом додатків, а Material-UI, Ant Design та Chakra UI - це бібліотеки компонентів для побудови користувацьких інтерфейсів.

Для бекенду було обрано Node.js з використанням фреймворку Express. Node.js - це JavaScript-оточення, що працює на движку V8 від Google і дозволяє розробляти серверні додатки. Express - це невеликий та гнучкий Node.js веб-додаток, що забезпечує набір основних функцій для розробки веб-сайтів.

В даній роботі було завдання обрати серед двох БД - PostgreSQL та MongoDB. PostgreSQL — повнофункціональна SQL-система керування реляційними базами даних. MongoDB — документоорієнтована база даних з відкритим вихідним кодом.

Інтелект веб-додатку покладено на дві бібліотеки машинного навчання: TensorFlow та PyTorch. TensorFlow - це відкритий інструмент для машинного навчання, орієнтований на масштабування та розробку. PyTorch - це машинна бібліотека для мови програмування Python, яка, його можна використовувати в NSF для поширення моделей нейронних мереж.

Завдання візуального аналізу було реалізоване за допомогою OpenCV та декількох попередньо навчених моделей: Object Detection (YOLO, SSD), Image Segmentation та моделі для оцінки об'єму.

Рисунок 4.38 – Приклад згенерованої курсової роботи, другий розділ, перший та другий підрозділ

3 РОЗРОБКА ВЕБ-ДОДАТКУ ДЛЯ ВІЗУАЛЬНОГО АНАЛІЗУ КАЛОРІЙ І ОЦІНКА ЕФЕКТИВНОСТІ

3.1 Опис вхідних даних та слоїв системи веб-додатку

Веб-додаток, що розробляється в рамках даної роботи, працює з об'єктами-зображеннями, які мають в своїй структурі багато важливої інформації. Вхідними даними системи є фотографії продуктів або страв, які користувач завантажує для аналізу з метою подальшого визначення кількості калорій.

Структура системи веб-додатку складається з декількох слоїв: інтерфейс користувача (фронтенд), серверна частина (бекенд), база даних, модель штучного інтелекту та інфраструктура. Кожен з цих слоїв виконує свої специфічні завдання.

Фронтенд є точкою доступу до системи для користувача. Він забезпечує візуальне представлення даних та реагування на дії користувача. Для створення інтерфейсу користувача використовуються такі технології, як: React, Redux/Context API, Material-UI/Ant Design/Chakra UI, JavaScript/TypeScript. За допомогою цих технологій розробник може створювати інтуїтивно зрозумілі, динамічні і відзивні інтерфейси.

Серверна частина забезпечує обробку запитів від клієнтської частини, проводить навчання та використання моделей штучного інтелекту, забезпечує зв'язок з базою даних та іншими службами. Для реалізації серверної частини використовується Node.js (Express), що є альтернативою Python та відмінно підходить для високопродуктивних веб-додатків.

База даних необхідна для зберігання та систематизації даних, потрібних для навчання моделей, результатів роботи системи, даних користувачів тощо. У рамках даного веб-додатку використовується як PostgreSQL, так і MongoDB.

Сердцем системи є модель штучного інтелекту, яка реалізована за допомогою пакетів TensorFlow або PyTorch, OpenCV. Враховуються моделі об'єктного виявлення (наприклад, YOLO, SSD), сегментації зображення, а також оцінки об'єму.

Інфраструктура системи враховує використання хмарних платформ (AWS/Google Cloud/Azure), Docker та API Management (API Gateway) для обробки та комунікації запитів і відповідей між різними частинами системи.

Таким чином, веб-додаток для візуального аналізу кількості калорій з використанням штучного інтелекту складається з взаємозалежних компонентів, кожен з яких виконує визначені завдання в єдиному процесі.

3.2 Планування та моделювання структури веб-додатку

Планування та моделювання структури веб-додатку відграють критично важливу роль у процесі його створення. Правильно спроектована структура кожного з цих елементів дозволить нам координувати всю роботу таким чином, щоб вона була максимально ефективною, а продукт - надійним і зручним у використанні.

Веб-додаток буде містити дві основні складови: клієнтську (фронтенд) та серверну (бекенд) частини. Цей підхід надзвичайно зручний, оскільки дозволяє нам розділити логіку роботи системи на дві великі частини.

Фронтенд розробляється із застосуванням технологій React та Redux або Context API. Важливо зазначити, що для створення інтерфейсу користувача використовуються Material-UI, Ant Design або Chakra UI, які дозволяють створювати зручний та інтуїтивно зрозумілий інтерфейс. При цьому мовою програмування є JavaScript або TypeScript.

Серверну ж частину планується створити на базі Node.js за допомогою фреймворку Express. Варто зазначити, що ця технологія була обрана як альтернатива Python, оскільки вона ефективна для додатків, для яких важлива висока продуктивність. Для зберігання інформації будуть використовуватися бази даних PostgreSQL та MongoDB.

Ключовими аспектами веб-додатку є його спроможності з аналізу кількості калорій за допомогою штучного інтелекту. Для цього використовується TensorFlow та PyTorch, а також OpenCV. Передбачається використання попередньо навчених

Рисунок 4.39 – Приклад згенерованої курсової роботи, третій розділ, перший та другий підрозділ

моделей для об'єктного виявлення (таких як YOLO, SSD), сегментації зображень та моделей для оцінки об'єму.

Інфраструктурну частину підтримують хмарні платформи, такі як AWS, Google Cloud та Azure, що мають величезний потенціал для масштабування та гнучкості. Крім того, вони гарантують високу доступність та надійність системи. Docker дозволить нам створювати необхідні сервіси, а за допомогою API Gateway ми зможемо узгоджувати роботу всіх елементів веб-додатку.

Таким чином, планування та моделювання структури веб-додатку є важливим етапом, який визначає подальше успішне його розгортання та функціонування. Ця стадія передбачає системний підхід, уважне вивчення всіх технологій та розуміння, як вони будуть взаємодіяти у процесі роботи веб-додатку.

3.3 Аналіз результатів роботи веб-додатку та оцінка його ефективності

Аналізувались такі характеристики, як швидкість роботи, коректність обчислень та стабільність роботи застосунку.

У результаті тестування веб-додатку було встановлено, що він стабільно та швидко працює, правильно обчислює калорійність продуктів та відображає цю інформацію для користувача. Додатково ефективність веб-додатку було оцінено на основі зворотнього зв’язку від користувачів. Позитивний відгук користувачів свідчить про високий рівень задоволення користувачів практичністю та ефективністю програми.

Застосовується наступні технології для реалізації проекту. Для створення фронтенда (інтерфейсу користувача) використовується React, Redux/Context API, Material-UI/Ant Design/Chakra UI, JavaScript/TypeScript. Ці технології допомагають створювати сучасний інтерфейс користувача, зручний для використання.

Для реалізації бекенду (серверної частини) використовується Node.js. Вибір випав на цю технологію через високу продуктивність та легкість перенесення. Зберігання даних здійснюється в базі даних PostgreSQL, MongoDB. Ці технології дозволяють швидко і надійно зберігати та отримувати дані, необхідні для роботи веб-додатку.

Рисунок 4.40 – Приклад згенерованої курсової роботи, третій розділ, третій підрозділ

ВИСНОВКИ

Курсова робота ставить перед собою завдання інформаційного та аналітичного порядку. В результаті всеобмежуючого аналізу задачі було встановлено, що ефективність використання веб-додатків для візуального аналізу калорій має високі перспективи для забезпечення кращого розуміння особливостей своєї дієти.

Штучний інтелект, який був використаний у даній роботі у вигляді TensorFlow/PyTorch, OpenCV, а також попередньо навчених моделей, як Object Detection, Image Segmentation, моделі для оцінки об'єму, дозволив максимально автоматизувати процеси розпізнавання та аналізу продуктів харчування. Виявлено, що застосування ШІ для візуального аналізу кількості калорій підвищує точність розпізнавання та вимірювання продуктів харчування.

При використанні сучасних фронтенд технологій, таких як React, Redux/Context API, Material-UI, Ant Design, Chakra UI, JavaScript/TypeScript, було розроблено ефективний та зручний в інтерфейс користувача. Бекенд через Node.js та PostgreSQL та MongoDB надав надійність та процес високої продуктивності. Використання cloud платформ включаючи AWS, Google Cloud та Azure разом з API Management (API Gateway) забезпечуло швидкість обробки та доступ до даних.

В ході розробки веб-додатку було виявлено, що головною вимогою користувачів є адекватний аналіз кількості калорій в їх раціоні, а також доступність інформації про потенційні залежності між кількістю калорій та наслідками для здоров'я на основі об'єктивних даних.

Наукова новизна курсової роботи полягає у розробці интеллектуального веб-додатка, здатного оцінити кількість калорій у їжі на основі візуального аналізу. В результаті було висунуто ідею, що за допомогою штучного інтелекту можна створити систему, яка дозволить користувачам більш точно знати кількість калорій у їх їжі, що, в свою чергу, може сприяти підтримці здорового способу життя.

Таким чином, дана курсова робота підчеркнула необхідність подальших досліджень у напрямку використання штучного інтелекту для візуального аналізу

Рисунок 4.41 – Приклад згенерованої курсової роботи, висновки

Для тестування перемикання мов інтерфейсу було обрано сторінку для генерації презентацій (рис. 4.42-4.43) перше зображення це сторінка на англійській мові, а друге це сторінка на українській мові.

Рисунок 4.42 – Сторінка генерації презентацій на англійській мові

Рисунок 4.43 – Сторінка генерації презентацій на українській мові

Для тестування покупки використань було здійснено покупку 10 генерацій (рис. 4.44), також успішну покупку можна побачити на сторінці transactions Stripe Dashboard (рис. 4.45). Також після успішної покупки користувача переносить на сторінку з подякою, через декілька секунд перенаправляє назад на головну сторінку (рис. 4.46).

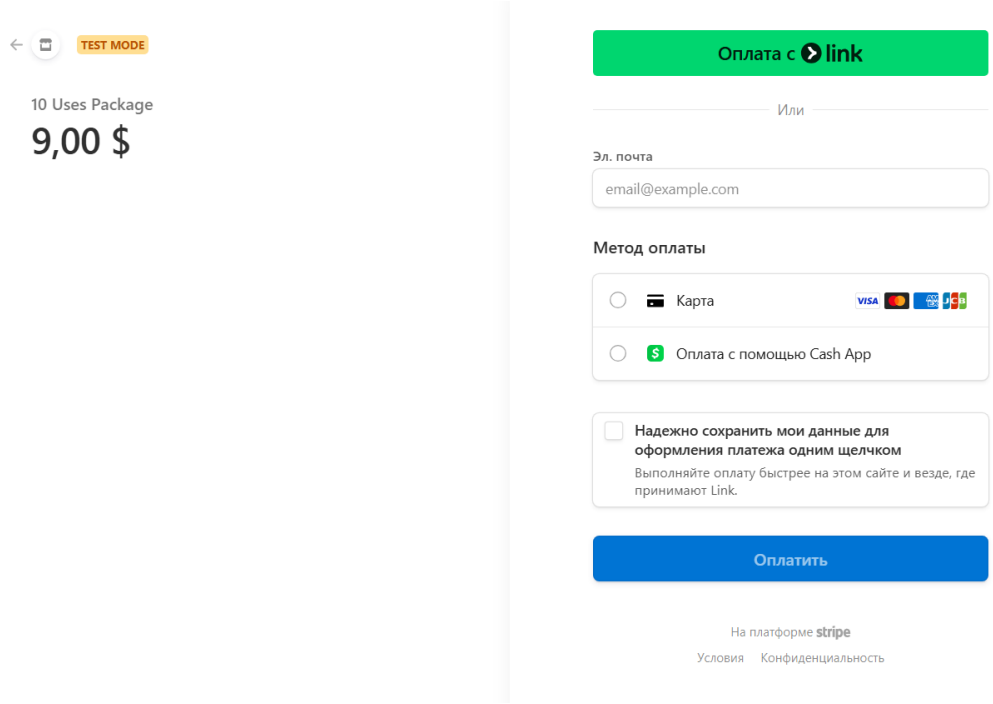


Рисунок 4.44 – Сторінка покупки 10 генерації

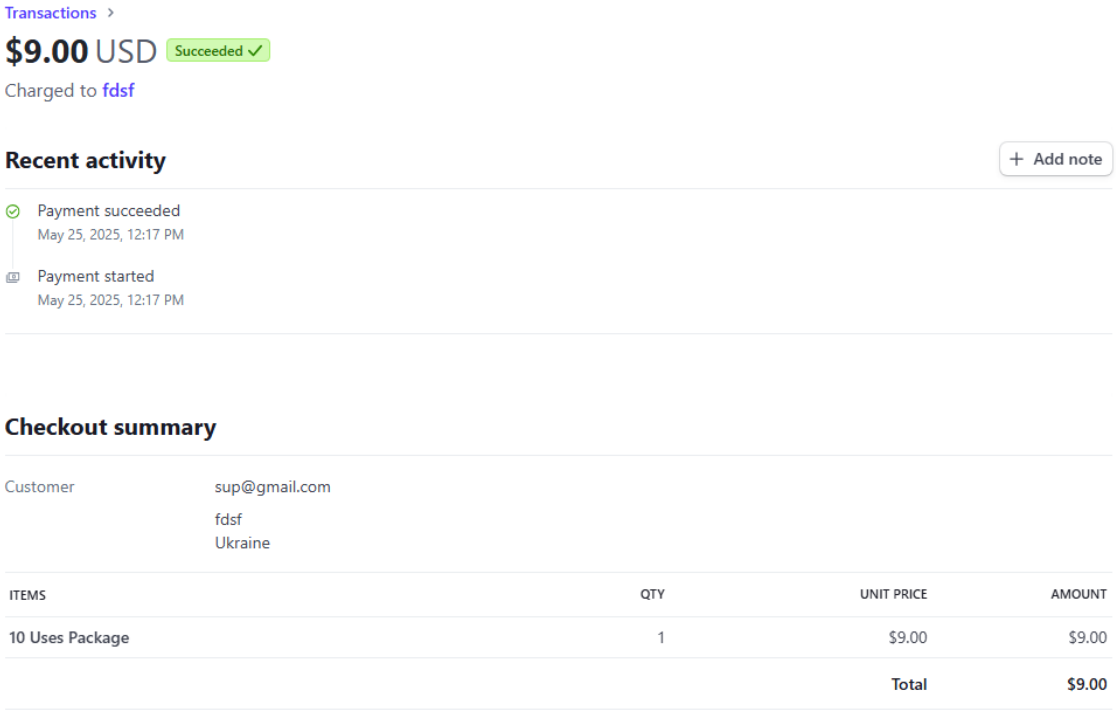


Рисунок 4.45 – Підтверджена покупка на сторінці Stripe

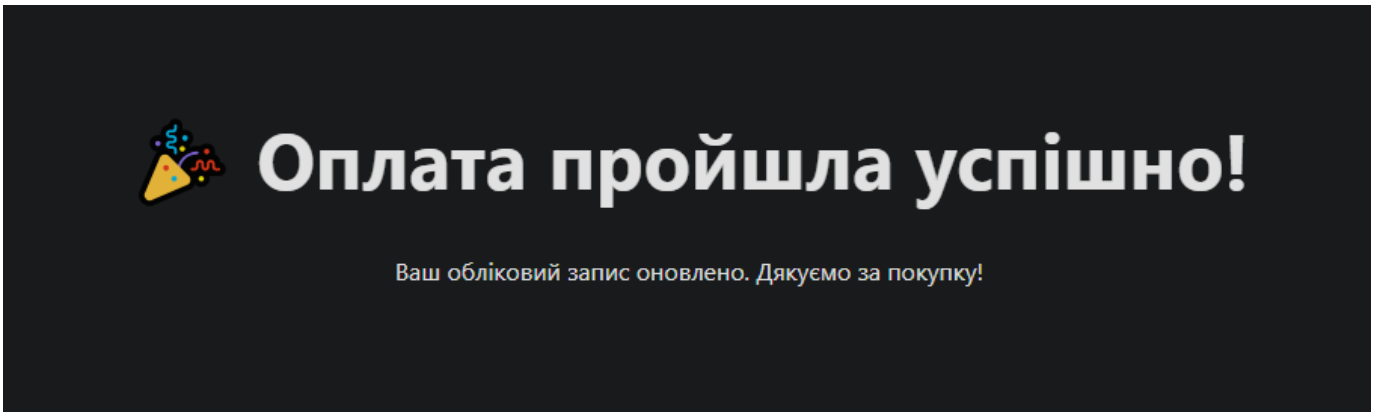


Рисунок 4.46 – Сторінка успішної покупки

За результатами тестування встановлено правильну роботу основних параметрів системи. Генерація всіх чотирьох типів документів працює відмінно. Перемикання мови виконується дуже швидко та без сильного навантаження на систему. Покупка використань та преміума працює.

Висновки до розділу 4

За результатами тестування встановлено правильну роботу основних параметрів системи. Усі ключові функції працюють стабільно та без критичних збоїв. Генерація документів чотирьох типів, тобто звичайних текстів, лабораторних, курсових робіт і презентацій все реалізується успішно з дотриманням заданого формату, мовних налаштувань і стилістичних вимог. Система демонструє стабільну продуктивність.

Перемикання мови сторінки та генерації відбувається миттєво без потреби перезавантаження. Також протестовано роботу механізму реєстрації й авторизації користувачів, який інтегровано з базою даних Supabase. Особливо важливою є успішна перевірка платіжного функціоналу, покупки пакетів використань та преміум-доступу, які також виконуються без помилок.

У процесі тестування не було виявлено суттєвих помилок або нестабільностей. Отримані результати демонструють не лише відповідність реалізованих функцій вимогам завдання, а й наявний потенціал до подальшого масштабування, зокрема в частині розширення формату генерацій та впровадження у реальні освітні процеси.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було створено інтелектуальну систему генерації освітніх матеріалів на основі генеративного штучного інтелекту. Основною метою було автоматизувати процес створення навчальних документів, що дозволяє скоротити час і ресурси, необхідні викладачам та студентам. У ході роботи реалізовано підтримку генерації текстових документів, лабораторних і курсових робіт згідно з вимогами ДСТУ, а також презентацій з вбудованими зображеннями та тексом.

У вебзастосунку було реалізовано: генерація презентацій, формування текстових матеріалів, підтримка зміни української та англійської мов, а також облік генерацій і користувачів через Supabase. Особлива увага приділена адаптації під українські освітні стандарти, що відрізняє цю систему від багатьох комерційних рішень, орієнтованих переважно на англомовну аудиторію. Завдяки цьому забезпечено відповідність реальним потребам викладачів і студентів закладів вищої освіти в Україні.

Результати роботи підтверджують доцільність і ефективність розробленого підходу, а також відповідність створеної системи меті дослідження. Розроблене рішення має практичне застосування у сфері освіти.

Рекомендується впровадити систему у закладах вищої освіти як допоміжний засіб для викладачів та студентів. Система є саме допоміжним інструментом, вона не буде робити все за студентів або викладачів, вона допомагає в створенні навчальних матеріалів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Y. Zhufeng and J. Sitthiworachart, "Research on Informatization Education Methods under the Background of Artificial Intelligence," 2021 2nd International Conference on Artificial Intelligence and Education (ICAIE), Dali, China, 2021, pp. 486-489, doi: 10.1109/ICAIE53562.2021.00107.

2. Y. Ren and L. Lan, "Application and Development Prospect of Artificial Intelligence in Quality Education," 2021 3rd International Conference on Internet Technology and Educational Informization (ITEI), Guangzhou, China, 2021, pp. 172-175, doi: 10.1109/ITEI55021.2021.00047.

3. T. Jiang, W. Li, J. Wang and X. Wang, "Using Artificial Intelligence-based Online Translation Website to improve the Health Education in International Students," 2021 2nd International Conference on Artificial Intelligence and Education (ICAIE), Dali, China, 2021, pp. 25-28, doi: 10.1109/ICAIE53562.2021.00012.

4. S. T. Abu-Orabi, "Artificial Intelligence Technologies in Higher Education: Reality and Challenges," 2024 17th International Conference on Development in eSystem Engineering (DeSE), Khorfakkan, United Arab Emirates, 2024, pp. 479-484, doi: 10.1109/DeSE63988.2024.10911936.

5. M. Zaabi, W. Hariri and N. Smaoui, "A review study of ChatGPT applications in education," 2023 International Conference on Innovations in Intelligent Systems and Applications (INISTA), Hammamet, Tunisia, 2023, pp. 1-5, doi: 10.1109/INISTA59065.2023.10310439.

6. SlidesAI.io – сервіс генерації презентацій з тексту : вебсайт. URL: https://www.slidesai.io (дата звернення: 30.04.2025).

7. Tome – платформа для інтерактивних презентацій і документів : вебсайт. URL: https://tome.app (дата звернення: 30.04.2025).

8. Canva Docs AI – створення текстових документів за допомогою ШІ : вебсайт. URL: https://www.canva.com/docs/ (дата звернення: 30.04.2025).

9. Gamma – інструмент для створення презентацій нового покоління : вебсайт. URL: https://gamma.app (дата звернення: 30.04.2025).

10. Beautiful.ai – сервіс дизайну презентацій з елементами ШІ : вебсайт. URL: https://www.beautiful.ai (дата звернення: 30.04.2025).

11. Notion AI – помічник зі штучним інтелектом для нотаток і документів : вебсайт. URL: https://www.notion.so/product/ai (дата звернення: 30.04.2025).

12. Kroma – генератор презентацій і пітчдеків з використанням ШІ : вебсайт. URL: https://kroma.ai (дата звернення: 30.04.2025).

13. MagicSlides AI – автоматична генерація слайдів із тексту : вебсайт. URL: https://www.magicslides.app (дата звернення: 30.04.2025).

14. Visme – платформа для створення візуального контенту : вебсайт. URL: https://www.visme.co (дата звернення: 30.04.2025).

15. Simplified – платформа для створення контенту за допомогою ШІ : вебсайт. URL: https://simplified.com (дата звернення: 30.04.2025).

16. Dickey E., Bejarano A., GAIDE: A Framework for Using Generative AI to Assist in Course Content Development // arXiv : website. URL: https://arxiv.org/abs/2308.12276 (Last accessed: 30.04.2025).

17. Jason Jabbour, Kai Kleinbard, Olivia Miller, Robert Haussman, Vijay Janapa Reddi., SocratiQ: A Generative AI-Powered Learning Companion for Personalized Education and Broader Accessibility // arXiv : website. URL: https://arxiv.org/abs/2502.00341 (Last accessed: 30.04.2025).

18. Guettala Manel, Samir Bourekkache, Okba Kazar, S. Harous, Generative Artificial Intelligence in Education: Advancing Adaptive and Personalized Learning // ResearchGate : website. URL: https://www.researchgate.net/publication/383303500 (Last accessed: 30.04.2025).

19. Eason Chen, Jia-En Lee, Jionghao Lin, Kenneth Koedinger, GPTutor: Great Personalized Tutor with Large Language Models for Personalized Learning Content Generation // arXiv : website. URL: https://arxiv.org/abs/2407.09484 (Last accessed: 30.04.2025).

20. Paul Denny, Hassan Khosravi, Arto Hellas, Juho Leinonen, Sami Sarsa, Can We Trust AI-Generated Educational Content? Comparative Analysis of Human and AI-Generated Learning Resources // arXiv : website. URL: https://arxiv.org/abs/2306.10509 (Last accessed: 30.04.2025).

21. N. Gridchina, N. Savvina and S. Zavyalov, "Topical Issues of the Use of Artificial Intelligence Technologies in Education and Law," 2023 3rd International Conference on Technology Enhanced Learning in Higher Education (TELE), Lipetsk, Russian Federation, 2023, pp. 8-11, doi: 10.1109/TELE58910.2023.10184378.

22. S. T. T and I. A. S B, "Artificial Intelligence Effects on Student Learning Outcomes in Higher Education," 2024 Ninth International Conference on Science Technology Engineering and Mathematics (ICONSTEM), Chennai, India, 2024, pp. 1-5, doi: 10.1109/ICONSTEM60960.2024.10568868.

23. F. Schimanke, "Media Competence is the key requirement when using Generative AI in Academic Education in a meaningful way," 2024 Artificial Intelligence x Humanities, Education, and Art (AIxHEART), Laguna Hills, CA, USA, 2024, pp. 46-47, doi: 10.1109/AIxHeart62327.2024.00015.

24. J. Ning, Y. Gao and M. Luo, "Application Research of Generative Artificial Intelligence Technology in the Design and Art Course Teaching," 2024 International Conference on Informatics Education and Computer Technology Applications (IECA), Beijing, China, 2024, pp. 165-169, doi: 10.1109/IECA62822.2024.00038.

25. X. Chen, Y. Liao and W. Yu, "Generative AI in Higher Art Education," 2024 6th International Conference on Computer Science and Technologies in Education (CSTE), Xi'an, China, 2024, pp. 135-140, doi: 10.1109/CSTE62025.2024.00032.

26. I. Zualkernan, "Adoption of Generative AI and Large Language Models in Education: A Short Review," 2025 International Conference on Electronics, Information, and Communication (ICEIC), Osaka, Japan, 2025, pp. 1-5, doi: 10.1109/ICEIC64972.2025.10879632.

27. Carl Rippon, Learn React with TypeScript: A beginner's guide to reactive web development with React 18 and TypeScript , Packt Publishing, 2023.

28. Steve Tingiris; Bret Kinsella, Exploring GPT-3: An unofficial first look at the general-purpose language processing API from OpenAI , Packt Publishing, 2021.

29. Supabase – хмарна платформа для зберігання даних і аутентифікації користувачів : вебсайт. URL: https://supabase.com (дата звернення: 30.04.2025).

30. Stripe – платіжна система для інтеграції онлайн-платежів : вебсайт. URL: https://stripe.com (дата звернення: 30.04.2025).

ДОДАТОК А

Лістинг програмного коду

Лістинг коду SquareCore.ts:

```
import PptxGenJS from "pptxgenjs";

export function generatePresentation(text: string, topic: string, imagesBase64:
string[], color: string, rounding: boolean): void {
    const pptx = new PptxGenJS();

    const sentences = text.split('.').filter(sentence => sentence.trim() !== '');

    const titleSlide = pptx.addSlide();
    titleSlide.addText(topic, {
        x: "10%" as PptxGenJS.Coord,
        y: "5%" as PptxGenJS.Coord,
        w: "80%" as PptxGenJS.Coord,
        h: "20%" as PptxGenJS.Coord,
        fontSize: 36,
        color: "FFFFFF",
        fill: { color: color },
        align: "center",
        valign: "middle",
        margin: 5,
        bold: true,
        fontFace: "Times New Roman"
    });

    if (imagesBase64.length > 0) {
        titleSlide.addImage({
            data: imagesBase64[0],
            x: "25%" as PptxGenJS.Coord,
            y: "30%" as PptxGenJS.Coord,
            w: "50%" as PptxGenJS.Coord,
            h: "50%" as PptxGenJS.Coord,
            rounding: rounding ? true : false
        });
    }

    for (let i = 0; i < imagesBase64.length; i++) {
        const slide = pptx.addSlide();
        const sentence = sentences[i] ? sentences[i] : '';
```

```
slide.addText(topic, {
    x: "5%" as PptxGenJS.Coord,
    y: "5%" as PptxGenJS.Coord,
    w: "90%" as PptxGenJS.Coord,
    h: "10%" as PptxGenJS.Coord,
    fontSize: 24,
    color: "FFFFFF",
    fill: { color: color },
    align: "center",
    valign: "middle",
    margin: 5,
    bold: true,
    fontFace: "Times New Roman"
});

slide.addShape("rect", {
    x: "5%" as PptxGenJS.Coord,
    y: "20%" as PptxGenJS.Coord,
    w: "40%" as PptxGenJS.Coord,
    h: "70%" as PptxGenJS.Coord,
    fill: { color: color },
});

slide.addText(sentence, {
    x: "5%" as PptxGenJS.Coord,
    y: "20%" as PptxGenJS.Coord,
    w: "40%" as PptxGenJS.Coord,
    h: "70%" as PptxGenJS.Coord,
    fontSize: 18,
    color: "FFFFFF",
    align: "center",
    valign: "middle",
    margin: 5,
    bold: true,
    fontFace: "Arial"
});

const imageOptions = {
    x: "50%" as PptxGenJS.Coord,
    y: "20%" as PptxGenJS.Coord,
    w: "45%" as PptxGenJS.Coord,
    h: "70%" as PptxGenJS.Coord,
    rounding: rounding ? true : false
};
```

```
        slide.addImage({
            data: imagesBase64[i],
            ...imageOptions,
        });
    }

    pptx.writeFile({ fileName: `${topic}.pptx` });
}
```

Лістинг коду App.tsx:

```
import { useState, useEffect } from 'react';
import './css/App.css';
import { generatePresentation as generateStandardPresentation } from '../../Back-
End/Core';
import { generatePresentation as generatePremiumPresentation } from '../../Back-
End/PremiumCore';
import { generatePresentation as generateSquarePresentation } from '../../Back-
End/SquareCore';
import { generatePresentation as generateTrianglePresentation } from '../../Back-
End/TriangleCore';
import Navbar from './Navbar';
import Footer from './Footer';
import { useSelector, useDispatch } from 'react-redux';
import { RootState } from './store';
import { useGeneration, setRemainingUses, setHasPremium } from './store/usageSlice';
import { useLocation } from 'react-router-dom';
import { callCohereForWord, callOpenAIAPI, callOpenAIForColor, callOpenAIForWord } from
'./api';
import { fetchImage } from './imageService';
import { handlePayment, checkPaymentStatus } from './paymentService';
import circleDesignImg from './images/circle-design.png';
import squareDesignImg from './images/square-design.png';
import Modal from './Modal';
import { supabase, logGeneration, decrementUserUses } from './supabaseClient';
import { useTranslation } from 'react-i18next';

function App() {
  const [message, setMessage] = useState("");
  const [slideCount, setSlideCount] = useState(5);
  const [showModal, setShowModal] = useState(false);
  const [modalOpen, setModalOpen] = useState(false);
  const [modalMessage, setModalMessage] = useState('');
  const showLoginModal = (message: string) => {
```

```
    setModalMessage(message);
    setModalOpen(true);
  };
  const [loading, setLoading] = useState(false);
  const [imageProvider, setImageProvider] = useState("Standard");
  const [designOption, setDesignOption] = useState("Square");
  const [rounding, setRounding] = useState(true);
  const [textGenerationProvider, setTextGenerationProvider] = useState("OpenAI");
  const [user, setUser] = useState<any>(null);
  const [openAIModel, setOpenAIModel] = useState<"gpt-3.5-turbo" | "gpt-4">("gpt-3.5-
turbo");
  const { t, i18n } = useTranslation();

  const remainingUses = useSelector((state: RootState) => state.usage.remainingUses);
  const hasPremium = useSelector((state: RootState) => state.usage.hasPremium);
  const dispatch = useDispatch();
  const location = useLocation();

  useEffect(() => {
    const getUser = async () => {
      const { data } = await supabase.auth.getUser();
      setUser(data.user);
    };

    getUser();

    const { data: authListener } = supabase.auth.onAuthStateChange((_, session) => {
      setUser(session?.user ?? null);
    });

    return () => {
      authListener?.subscription.unsubscribe();
    };
  }, []);

  useEffect(() => {
    const savedRemainingUses = localStorage.getItem('remainingUses');
    if (savedRemainingUses !== null) {
      dispatch(setRemainingUses(parseInt(savedRemainingUses)));
    }

    const savedHasPremium = localStorage.getItem('hasPremium');
    if (savedHasPremium !== null) {
      dispatch(setHasPremium(savedHasPremium === 'true'));
    }
```

```
  }, [dispatch]);

  useEffect(() => {
    localStorage.setItem('remainingUses', remainingUses.toString());
  }, [remainingUses]);

  useEffect(() => {
    localStorage.setItem('hasPremium', hasPremium.toString());
  }, [hasPremium]);

  useEffect(() => {
    const queryParams = new URLSearchParams(location.search);
    const sessionId = queryParams.get('session_id');

    if (sessionId) {
      checkPaymentStatus(sessionId, dispatch);
    }
  }, [location, dispatch]);

  async function handleGenerate() {
    if (!user) {
      showLoginModal(t('pleaseloginpresentations'));
      return;
    }
    setLoading(true);
    if (remainingUses > 0) {
      let text: string | null = null;

      if (textGenerationProvider === "OpenAI") {
        text = await callOpenAIForWord(message, slideCount, openAIModel);

      } else if (textGenerationProvider === "Cohere") {
        text = await callCohereForWord(message, slideCount);
      }

      const color = await callOpenAIForColor(message);
      if (text && color) {
        const imagesPromises = Array.from({ length: slideCount - 1 }, () =>
fetchImage(message, imageProvider));
        const imagesBase64 = await Promise.all(imagesPromises);
        const validImages = imagesBase64.filter(image => image !== null) as string[];

        if (hasPremium) {
          switch (designOption) {
            case "Square":
```

```
            generateSquarePresentation(text, message, validImages, color, rounding);
            break;
          case "Triangle":
            generateTrianglePresentation(text, message, validImages, color);
            break;
          default:
            generatePremiumPresentation(text, message, validImages, color, rounding);
        }
      } else {
        generateStandardPresentation(text, message, validImages, color);
      }

      dispatch(useGeneration());
    }
  } else {
    setShowModal(true);
  }

  if (!user) {
    console.error("User not authenticated");
    return;
  }
      await decrementUserUses(user.id);
      await logGeneration(user.id, message, "presentation");
      setLoading(false);
}

return (
  <>
    <Navbar />
    <div className="content-container">
      <div className="glow-container">
        <h2>{t('presentationtitle')}</h2>
        <textarea
          onChange={(e) => setMessage(e.target.value)}
          placeholder={t('pleasewritetopic')}
          cols={50}
          rows={10}
        />
      </div>

      <div>
        <label htmlFor="slide-count">{t('numberofslides')}</label>
        <select
          id="slide-count"
```

```
            className="input-number"
            value={slideCount}
            onChange={(e) => setSlideCount(Number(e.target.value))}
          >
            {[2, 3, 4, 5, 6].map((num) => (
              <option key={num} value={num}>{num}</option>
            ))}
          </select>
        </div>

        <div>
          <label htmlFor="image-provider">{t('imageprovider')}</label>
          <select
            id="image-provider"
            value={imageProvider}
            onChange={(e) => setImageProvider(e.target.value)}
          >
            <option value="Standard">{t('standart')} (Unsplash)</option>
            <option value="DALL-E" disabled={!hasPremium}>DALL-E</option>
          </select>
        </div>

      <div>
        <label htmlFor="text-generation-provider">{t('textprovider')}</label>
        <select
          id="text-generation-provider"
          value={textGenerationProvider}
          onChange={(e) => setTextGenerationProvider(e.target.value)}
        >
          <option value="OpenAI">OpenAI (ChatGPT)</option>
          <option value="Cohere">Cohere</option>
        </select>
      </div>

      {textGenerationProvider === "OpenAI" && (
        <div>
          <label htmlFor="openai-model">{t('openaimodel')}</label>
          <select
            id="openai-model"
            value={openAIModel}
            onChange={(e) => setOpenAIModel(e.target.value as "gpt-3.5-turbo" | "gpt-
4")}
          >
            <option value="gpt-3.5-turbo">GPT-3.5 Turbo</option>
            <option value="gpt-4">GPT-4</option>
```

```
          </select>
        </div>
      )}

        <div>
          <label htmlFor="design-option">{t('designoptions')}</label>
          <select
            id="design-option"
            value={designOption}
            onChange={(e) => setDesignOption(e.target.value)}
          >
            <option value="Circle">Circle</option>
            <option value="Square">Square</option>
          </select>
        </div>

        <div>
          <label>
            <input
              type="checkbox"
              checked={rounding}
              onChange={(e) => setRounding(e.target.checked)}
            />
            {t('roundcorners')}
          </label>
        </div>

        <div className="design-images">
          {designOption === "Circle" && <img src={circleDesignImg} alt="Circle Design"
className="design-img" />}
          {designOption === "Square" && <img src={squareDesignImg} alt="Square Design"
className="design-img" />}
        </div>

        <div>
        <button
            className="btn"
            onClick={handleGenerate}
          disabled={loading}
        >
          {loading ? t('generatingprocess') : t('generatepresentation')}
          </button>
        </div>

        <Footer />
```

```
        <Modal
          isOpen={modalOpen}
          onClose={() => setModalOpen(false)}
          message={modalMessage}
        />

        {showModal && (
          <div className="modal">
            <div className="modal-content">
              <h2>{t('outofuses')}</h2>
              <p>{t('outofuses1')}</p>
              <button className="btn" onClick={() => setShowModal(false)}>Close</button>
            </div>
          </div>
        )}
      </div>
    </>
  );
}

export default App;
```