Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Штучного інтелекту
(повна назва)

# КВАЛІФІКАЦІЙНА РОБОТА
# Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Розробка AI-сервісу для створення презентацій
(тема)

Виконав:
здобувач четвертого року навчання,
групи ІТШІ-21-5
Олександр Гнідкін
(власне ім'я, прізвище)
Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Штучний інтелект
(повна назва освітньої програми)
Керівник доц. Вадим Шергін
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ШІ _____ Олег ЗОЛОТУХІН
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Штучного інтелекту

Рівень вищої освіти перший (бакалаврський)

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

Тип програми освітньо-професійна

Освітня програма Штучний інтелект
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

# ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Гнідкіну Олександру Євгеновичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка AI-сервісу для створення презентацій

затверджена наказом університету від 19 травня 20 25 р. № 378Ст

2. Термін подання студентом роботи до екзаменаційної комісії 20 червня 20 25 р.

3. Вихідні дані до роботи офіційні документації до OpenAI API, Mantine UI, React.js, TypeScript, Firebase, NestJS, PptxGenJS, html2pdf.js, а також стандарти REST API та принципи UX/UI-дизайну.

4. Перелік питань, що потрібно опрацювати в роботі

1) Аналіз предметної галузі та постановка задачі

2) Проектування системи

3) Програмна реалізація

**КАЛЕНДАРНИЙ ПЛАН**

| № | Назва етапів роботи | Строк / терміни виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Отримання завдання на кваліфікаційну роботу | 19.05.2025 | виконано |
| 2 | Визначення проблеми | 20.05.2025 | виконано |
| 3 | Аналіз існуючих аналогів | 21.05.2025 | виконано |
| 4 | Розробка компонентів системи | 22.05.2025 | виконано |
| 5 | Реалізація сервісу | 28.05.2025 | виконано |
| 6 | Оформлення пояснювальної записки | 01.06.2025 | виконано |
| 7 | Захист | 20.06.2025 | виконано |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

Дата видачі завдання 19 травня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Вадим Шергін
(підпис) (посада, власне ім'я, прізвище)

# РЕФЕРАТ

Пояснювальна записка: 91 с., 23 рис., 1 дод., 20 джерел.

АВТОМАТИЗАЦІЯ, ВЕБДОДАТОК, ГЕНЕРАЦІЯ ПРЕЗЕНТАЦІЙ, ШТУЧНИЙ ІНТЕЛЕКТ, AI-GENERATED CONTENT, GPT-4, HTML2PDF.JS, OPENAI API, PPTXGENJS, REACT, REST API, TYPESCRIPT.

Об’єктом дослідження є сучасні вебсервіси, що використовують штучний інтелект для автоматизованої генерації мультимедійного контенту.

Предметом дослідження виступають сучасні технології побудови клієнт-серверних систем для створення презентацій за допомогою генеративних мовних моделей.

Метою роботи є розробка вебдодатку, що дозволяє користувачеві створювати структуровані презентації на основі текстового запиту, із можливістю подальшого експорту у популярні формати.

У дослідженні використано методи моделювання програмної архітектури, реалізації REST API, інтеграції генеративної моделі GPT-4, а також методику розробки інтерфейсу з урахуванням принципів UX/UI.

У результаті реалізовано повноцінну систему, яка дозволяє генерувати презентації з налаштованими параметрами стилю, структури та тематики, що забезпечує сучасний рівень автоматизації навчального та бізнес-контенту.

Систему рекомендовано використовувати у сфері освіти, презентаційної діяльності та швидкого створення візуалізованих доповідей. Завдяки можливості автоматичної генерації змістовних і структурованих презентацій за кілька хвилин, вона значно полегшує підготовку матеріалів для навчальних занять, конференцій, захисту наукових робіт чи навіть бізнес-зустрічей.

# ABSTRACT

Bachelor’s thesis contains: 91 pp., 23 fig., 1 ann., 20 references.

AI-GENERATED CONTENT, ARTIFICIAL INTELLIGENCE, AUTOMATION, GPT-4, HTML2PDF.JS, OPENAI API, PRESENTATION GENERATION, PPTXGENJS, REACT, REST API, TYPESCRIPT, WEB APPLICATION.

The object of the research is modern web services that utilize artificial intelligence for the automated generation of multimedia content.

The subject of the research is modern technologies for building client-server systems designed to create presentations using generative language models.

The aim of the work is to develop a web application that enables users to generate structured presentations based on textual input, with the ability to export the result into popular formats.

The research applies methods of software architecture modeling, REST API implementation, GPT-4 generative model integration, and interface development methodology based on UX/UI principles.

As a result, a fully functional system has been implemented that allows the generation of presentations with customizable parameters such as style, structure, and topic, thereby ensuring a modern level of automation for educational and business content.

The system is recommended for use in education, presentation activities, and rapid creation of visualized reports. With its capability to automatically generate meaningful and structured presentations within minutes, it significantly simplifies the preparation of materials for lectures, conferences, academic defenses, and even business meetings.

**ЗМІСТ**

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

AI – Artificial Intelligence – штучний інтелект;

API – Application Programming Interface – прикладний програмний інтерфейс;

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

GPT – Generative Pre-trained Transformer – генеративний попередньо натренований трансформер;

HTTP – Hypertext Transfer Protocol – протокол передачі гіпертексту;

PDF – Portable Document Format – портативний формат документа;

PPTX – PowerPoint Open XML Presentation – формат презентації Microsoft PowerPoint;

REST – Representational State Transfer – передача репрезентативного стану;

SDK – Software Development Kit – комплект засобів розробника;

UI – User Interface – інтерфейс користувача.

# ВСТУП

Стрімкий розвиток цифрових технологій та штучного інтелекту суттєво вплинув на всі сфери діяльності людини, зокрема на процес створення та подання інформаційних матеріалів. Презентації як форма візуального представлення ідей та знань займають важливе місце в освітньому процесі, бізнес-комунікаціях, наукових доповідях та маркетинговій діяльності. Традиційні засоби створення презентацій, такі як Microsoft PowerPoint, Google Slides та Apple Keynote, забезпечують користувачів потужним набором інструментів для візуалізації інформації, однак вимагають значних часових витрат на підготовку матеріалу, оформлення слайдів та структурування змісту. Водночас сучасні реалії диктують необхідність оптимізації цих процесів, оскільки темпи обміну інформацією постійно зростають, а вимоги до якості та оперативності підготовки презентаційного контенту стають дедалі вищими.

Завдяки розвитку штучного інтелекту виникла можливість автоматизувати підготовку презентацій за допомогою спеціалізованих сервісів. Нові рішення, такі як beautiful ai, tome, gamma, slidesai та інші, пропонують генерацію структури презентацій, підбір оформлення та змісту слайдів із мінімальними зусиллями користувача. Такі інструменти орієнтовані на швидкість створення контенту, врахування цільової аудиторії та адаптацію стилю до конкретних потреб. Разом з тим, незважаючи на появу інноваційних продуктів, більшість доступних на ринку сервісів мають певні обмеження: недостатню гнучкість у редагуванні, обмеженість налаштувань дизайну, відсутність підтримки різноманітних форматів експорту або недостатню інтеграцію з зовнішніми платформами. Це створює потребу у розробці нових рішень, які б максимально поєднували переваги традиційних та сучасних підходів до створення презентацій.

Актуальність виконання даної роботи визначається необхідністю створення веб-сервісу, що дозволяє користувачеві швидко та зручно

формувати прототипи та повноцінні презентації з урахуванням сучасних вимог до якості контенту, гнучкості налаштувань та зручності експорту. Особливий акцент у розробці робиться на використанні технологій штучного інтелекту, зокрема можливостей великих мовних моделей для генерації текстових матеріалів. Рішення, яке розробляється в межах цієї кваліфікаційної роботи, має забезпечити мінімальне навантаження на користувача при збереженні високого рівня індивідуалізації контенту. Такий підхід дозволить скоротити час підготовки презентацій, підвищити ефективність візуальної комунікації та задовольнити потреби широкого кола користувачів.

Мета роботи полягає у розробці AI-сервісу для створення презентацій на задану тему, який забезпечує генерацію прототипів слайдів, подальше оформлення презентацій відповідно до обраного стилю та експорт готових матеріалів у різні популярні формати. Запропоноване рішення передбачає підтримку авторизації користувачів, збереження історії створених прототипів і презентацій, а також можливість налаштування параметрів генерації відповідно до вимог конкретної аудиторії або завдання. У процесі розробки буде враховано досвід використання як традиційних інструментів, так і сучасних AI-рішень, що дозволить створити інтуїтивно зрозумілий та функціонально багатий сервіс.

Можливі сфери застосування розробленого сервісу охоплюють різні галузі, де існує потреба у швидкому створенні якісних презентацій. Це освітні установи, де викладачі та студенти потребують регулярної підготовки навчальних матеріалів, бізнес-компанії, яким необхідні інструменти для оперативної розробки звітів та стратегічних презентацій, стартапи та маркетингові агенції, які працюють у швидкоплинному середовищі та вимагають швидкого створення пітч-презентацій. Рішення також буде корисним для приватних користувачів, що займаються самоосвітою, веденням блогів або проведенням публічних заходів.

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

## 1.1 Еволюція презентацій як форми подання інформації

Презентації є важливою складовою комунікаційних процесів у різних сферах діяльності, від бізнесу до освіти. Вони слугують інструментом для візуалізації та структурованого подання інформації, дозволяючи краще донести основні ідеї та аргументи до аудиторії. З часом презентації стали невід'ємною частиною сучасної культури обміну знаннями.

Технології створення презентацій пройшли значну еволюцію. Від простих паперових слайдів і ручних проекторів розвиток дійшов до інтерактивних цифрових інструментів, що надають широкі можливості для дизайну, анімації та мультимедійного супроводу. Сьогодні створення презентацій стало доступним широкому колу користувачів завдяки зручним програмним рішенням, які не потребують глибоких технічних знань.

У контексті стрімкого розвитку штучного інтелекту дедалі більше уваги приділяється автоматизації створення презентаційного контенту. Нові технології дають змогу скоротити час на підготовку матеріалів, адаптувати зміст під різні цільові аудиторії та генерувати візуально привабливі слайди без залучення професійних дизайнерів.

### 1.1.1 Презентація як спосіб донесення інформації

Презентація є одним з найефективніших способів донесення інформації до аудиторії завдяки своїй здатності поєднувати візуальний і текстовий контент. Застосування слайдів дозволяє структурувати повідомлення, забезпечити краще сприйняття матеріалу і сприяти запам'ятовуванню основних ідей. Для досягнення ефективності повідомлення важливо дотримуватися таких принципів (рисунок 1.1), як простота, стислість і достовірність.

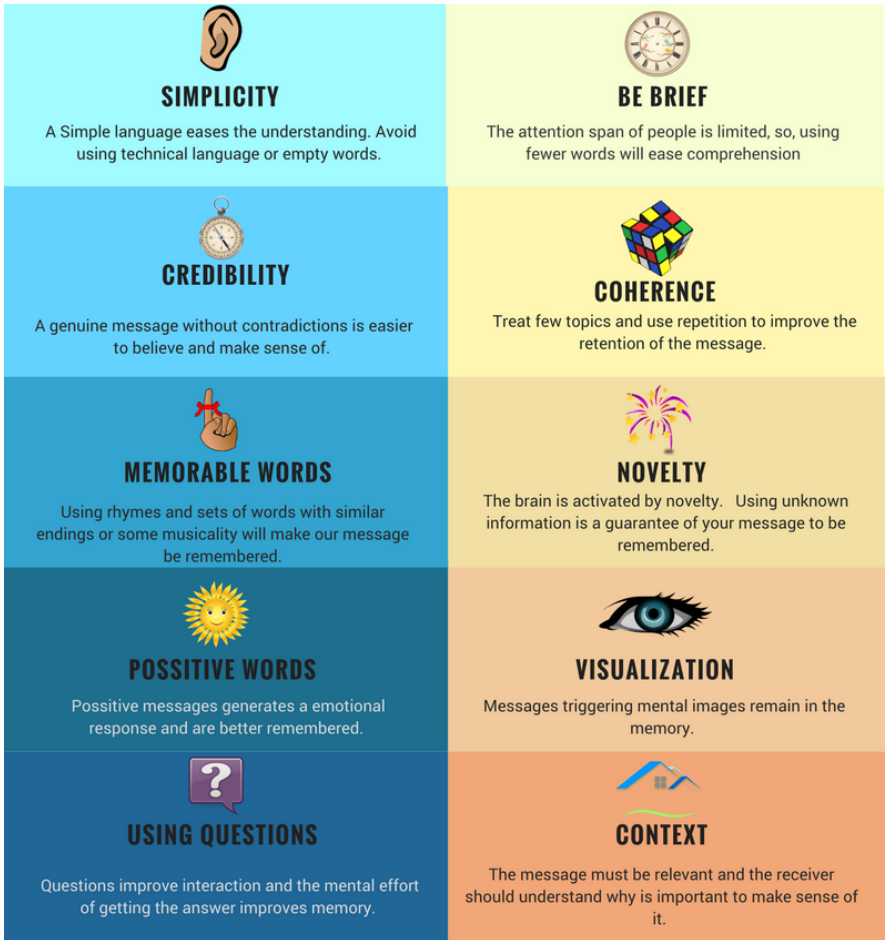


Рисунок 1.1 – Принципи ефективного донесення інформації

Просте і зрозуміле формулювання сприяє кращому сприйняттю інформації, адже складна термінологія чи зайві слова можуть лише ускладнити розуміння. Використання мінімальної кількості слів без втрати змісту підвищує концентрацію уваги слухача. Принципи простоти та лаконічності є ключовими для побудови доступної презентації, що утримує увагу навіть у разі складної тематики.

Ще одним важливим аспектом є психологічна обґрунтованість викладеного матеріалу. Вживання позитивних і легких для запам’ятовування слів, застосування візуальних образів, риторичних запитань та логічно побудованої послідовності слайдів дозволяє підсилити емоційний ефект і краще зафіксувати інформацію в пам’яті аудиторії. Принципи пам’ятних слів, позитивних формулювань, візуалізації та

використання запитань демонструють, як саме ці елементи впливають на якість сприйняття.

Окрім цього, надзвичайно важливою є відповідність повідомлення контексту ситуації та очікуванням аудиторії. Принципи узгодженості, новизни та контексту вказують, що чітка структура, неочікувані елементи подачі та доречність змісту формують цілісне враження і забезпечують досягнення головної мети презентації – ефективного донесення повідомлення.

### 1.1.2 Еволюція засобів створення презентацій

Історія створення презентацій як візуального способу комунікації налічує тисячі років. Ще в доісторичні часи люди залишали зображення на каменях, які виконували функцію передачі знань та досвіду. Ці перші «слайди» у вигляді наскельних малюнків стали прототипами майбутніх візуальних носіїв інформації. Далі розвиток (рисунок 1.2) цього засобу комунікації поступово відбувався разом з еволюцією суспільства і технічного прогресу.

З переходом до писемності та формальної освіти з’явилися перші дидактичні інструменти – стенди, плакати, таблиці, які активно використовувалися у шкільному навчанні, наукових доповідях та медичних демонстраціях. Така форма подачі матеріалу дозволяла систематизувати інформацію і робити її більш доступною для сприйняття.

У другій половині XX століття великого поширення набули діапроектори, які дозволяли демонструвати зображення або текст за допомогою світлового променя. Вони стали популярними в освітніх закладах, на підприємствах та у наукових установах. Незважаючи на обмеження у редагуванні слайдів, проектори забезпечували велику аудиторну охоплюваність і надали новий імпульс до подальшого розвитку засобів візуалізації.

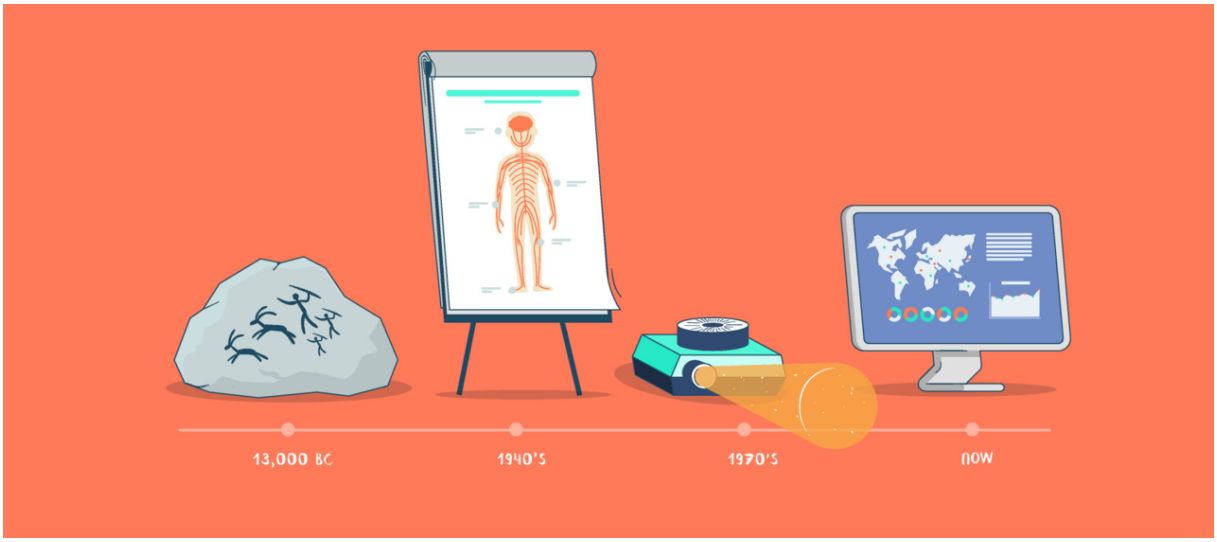


Рисунок 1.2 – Розвиток презентацій

Сучасний етап характеризується цифровими мультимедійними засобами, які дають змогу не лише створювати, а й інтерактивно демонструвати презентації. Програмне забезпечення, яке працює на комп'ютерах або онлайн, дозволяє зручно поєднувати текст, графіку, відео та анімацію. Ця еволюція засобів створення презентацій є свідченням зростаючої ролі візуальної комунікації в умовах інформаційного суспільства.

### 1.1.3 Роль штучного інтелекту у генерації презентацій

Використання штучного інтелекту у сфері створення презентацій відкриває нові можливості для автоматизації та персоналізації контенту. Завдяки мовним моделям, генерація структури слайдів, підбір текстів, зображень та навіть оформлення відбувається автоматично, що значно знижує навантаження на користувача. Такий підхід дозволяє суттєво прискорити підготовку презентацій, зберігаючи при цьому якість і логічну цілісність. Далі можна побачити основні переваги генератора презентацій на основі штучного інтелекту (рисунок 1.3).

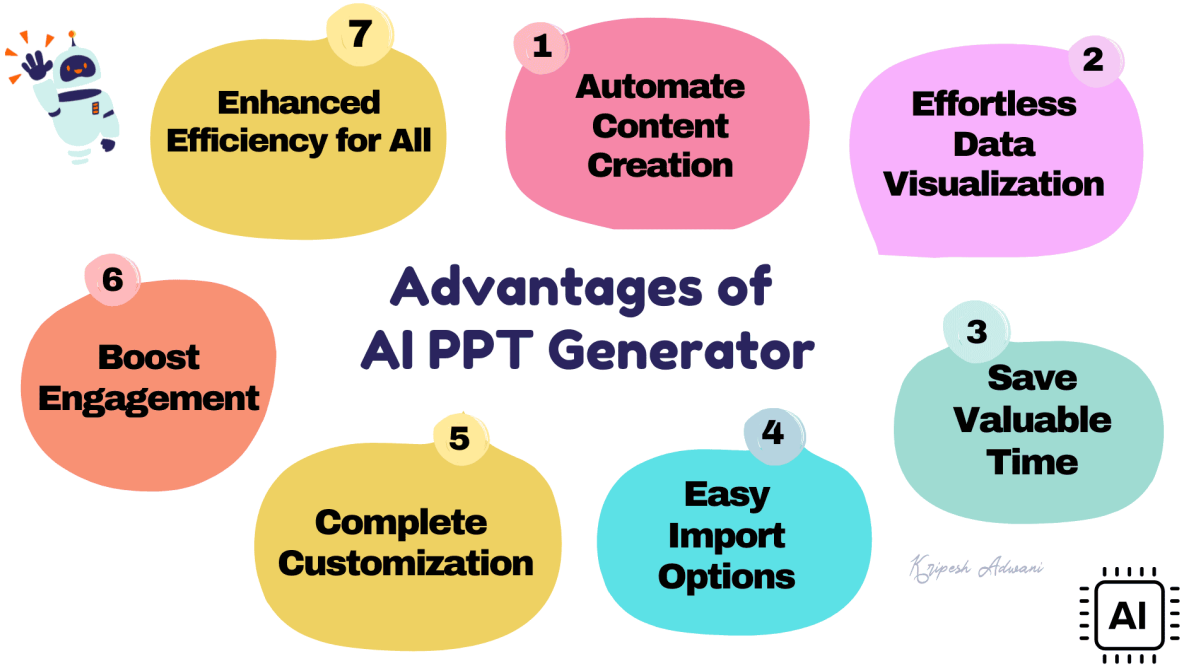


Рисунок 1.3 – Переваги генерації презентацій за допомогою ШІ

Серед ключових переваг варто виділити економію часу, спрощену візуалізацію даних і можливість імпорту зовнішніх матеріалів. Також важливим є фактор повної кастомізації – користувач може адаптувати згенеровані матеріали під конкретні потреби, включаючи стиль оформлення, структуру слайдів і тип подачі. Це особливо цінно для користувачів без досвіду роботи з графічними редакторами чи дизайнерськими шаблонами.

Ще однією важливою властивістю є підвищення ефективності та залучення аудиторії. Генератори презентацій на базі штучного інтелекту можуть враховувати цільову аудиторію, що дозволяє формувати релевантні й цікаві повідомлення. Завдяки цьому презентації стають більш динамічними, а сам процес створення – інклюзивним і доступним для всіх. Такі інструменти значно розширюють можливості користувача на кожному етапі створення презентацій.

### 1.1.4 Перспективи розвитку автоматизованого створення мультимедійного контенту

Автоматизоване створення мультимедійного контенту продовжує активно розвиватися завдяки поєднанню штучного інтелекту, обробки природної мови, генеративної графіки та мультимодальних моделей. Зростання популярності генеративних нейромереж, таких як великі мовні моделі, текст-до-зображення та голосові синтезатори, сприяє створенню презентацій, відео, інфографіки та інших матеріалів без прямої участі дизайнера чи редактора. Такий підхід дозволяє оптимізувати процеси підготовки контенту в бізнесі, освіті, маркетингу, науковій комунікації та медіа.

У майбутньому очікується подальше розширення функціональності подібних систем, зокрема впровадження глибшої персоналізації, адаптації до стилю спікера, автоматичної побудови наративу та інтеграції з доповненою і віртуальною реальністю. Це дозволить не лише генерувати контент, а й формувати повноцінний мультимедійний досвід, орієнтований на конкретного користувача або аудиторію. Враховуючи ці тенденції, автоматизоване створення мультимедійного контенту стане ключовим елементом цифрової трансформації інформаційної взаємодії.

## 1.2 Аналіз сучасних сервісів для створення презентацій

Сфера створення презентацій переживає значну трансформацію під впливом розвитку цифрових технологій і зростаючого попиту на автоматизацію. Традиційні інструменти залишаються актуальними завдяки широкому функціоналу та звичності для користувачів, однак зростає інтерес до рішень, які здатні автоматично генерувати контент, зменшуючи потребу у ручній роботі.

На ринку існує велика кількість як класичних програмних продуктів, так і інноваційних сервісів на основі штучного інтелекту. Кожен з них пропонує різний набір можливостей, орієнтуючись на певні сегменти користувачів – від студентів і викладачів до фахівців з маркетингу та бізнес-аналітики. У зв’язку з цим виникає необхідність систематизованого аналізу доступних рішень.

Важливим аспектом є також порівняння функціональних характеристик платформ, зручності інтерфейсу, гнучкості налаштувань, можливостей візуального редагування та підтримки сучасних форматів. Оцінка сильних і слабких сторін дозволяє краще зрозуміти, як саме кожен інструмент відповідає потребам користувача в умовах динамічного інформаційного середовища.

### 1.2.1 Огляд традиційних програм для створення презентацій

Серед традиційних програм для створення презентацій провідні позиції займають кілька добре відомих рішень, які здобули широку популярність завдяки надійності, функціональності та зручності використання.

До таких належать (рисунок 1.4) Microsoft PowerPoint, Google Slides, Apple Keynote та Canva. Ці інструменти забезпечують користувачам базовий і розширений функціонал для підготовки презентацій у різних стилях і для різних цілей.

Вони активно застосовуються як в освітньому процесі, так і в корпоративному секторі, що свідчить про їх універсальність. Крім того, кожен із зазначених сервісів має свою специфіку, що дозволяє користувачеві обирати програму залежно від технічного середовища та завдань.


Рисунок 1.4 – Логотипи основних конкурентів

Microsoft PowerPoint є найбільш усталеним стандартом у професійному середовищі, підтримуючи широкий набір шаблонів, інструментів анімації, вставки мультимедійного контенту та інтеграції з іншими офісними додатками.

Google Slides, своєю чергою, надає змогу створювати презентації онлайн з можливістю спільного редагування в реальному часі, що особливо зручно для командної роботи.

Apple Keynote відзначається сучасним дизайном, плавною анімацією та інтеграцією з екосистемою Apple. Його орієнтація на візуально привабливі рішення робить його популярним серед користувачів, що працюють на macOS та iOS.

Canva, хоч і не є класичним редактором слайдів, пропонує інтуїтивно зрозумілий інтерфейс і велику бібліотеку шаблонів, що дозволяє швидко створювати стильні презентації без дизайнерських навичок.

### 1.2.2 Огляд сучасних AI-рішень для створення презентацій

Сучасні сервіси для автоматизованого створення презентацій стрімко розвиваються завдяки використанню штучного інтелекту. Вони орієнтовані на мінімізацію зусиль користувача при підготовці презентаційного матеріалу, пропонуючи готові шаблони, автоматичну генерацію змісту та адаптивні інтерфейси.

Одним із найвідоміших інструментів є Beautiful.ai (рисунок 1.5). Його головна особливість полягає в автоматичному вирівнюванні об'єктів на слайдах та вбудованих принципах дизайну, які допомагають навіть непрофесіоналам створювати естетично привабливі презентації. Користувачеві достатньо додати основний зміст, а платформа самостійно пропонує оптимальне візуальне оформлення.


Рисунок 1.5 – Логотип Beautiful.ai

Tome (рисунок 1.6) позиціонує себе як платформа нового покоління для створення історій, де кожна презентація будується як візуальний наратив. Сервіс поєднує можливості генерації текстів та зображень за допомогою мовної моделі і моделі зображень, дозволяючи створювати

інтерактивний контент. Він підтримує адаптивну верстку, автоматичне компонування сторінок і велику кількість інтеграцій з іншими сервісами, що робить його зручним у використанні в командній роботі та під час розробки креативного контенту.


Рисунок 1.6 – Логотип Tome

Gamma (рисунок 1.7) орієнтований на простоту подачі і гнучкість у структурі. Його концепція передбачає створення документів, які одночасно є і презентацією, і вебсторінкою. Платформа дозволяє редагувати структуру в реальному часі, додавати інтерактивні блоки та автоматично адаптує дизайн до вмісту.

Завдяки цьому користувачі отримують гібридний формат, що поєднує інформативність текстового документа та наочність презентації. Крім того, підтримується спільна робота над контентом із можливістю залишати коментарі та відслідковувати зміни. Такий підхід особливо корисний для команд, що працюють у сфері маркетингу, освіти або стартап-презентацій.

Рисунок 1.7 – Логотип Gamma

SlidesAI (рисунок 1.8) є розширенням до Google Slides, яке дозволяє генерувати презентації на основі введеного тексту. Користувачеві достатньо вставити абзац або тези, після чого плагін перетворює їх у повноцінні слайди з дизайном і структурою. Цей підхід є особливо корисним для викладачів, студентів і тих, хто працює в системі Google Workspace, адже дозволяє створювати матеріали без перемикання між платформами.


Рисунок 1.8 – Логотип SlidesAI

Переваги таких AI-сервісів полягають у швидкості, автоматизації, кастомізації та здатності адаптувати контент під стиль аудиторії. Вони активно витісняють традиційні підходи, пропонуючи новий рівень зручності й ефективності. Очевидно, що такі рішення не просто доповнюють існуючі інструменти, а змінюють саму парадигму створення презентацій у цифрову епоху.

### 1.2.3 Порівняльний аналіз функціональності конкурентів

Порівняльний аналіз функціональності сучасних інструментів для створення презентацій дозволяє виявити їх ключові особливості та відмінності. Традиційні програми, як-от Microsoft PowerPoint або Google Slides, надають користувачам повний контроль над структурою, дизайном і змістом слайдів, проте вимагають більше часу та зусиль для самостійного наповнення. Вони пропонують широкий набір шаблонів, функцій анімації, інтеграцію з іншими офісними програмами та гнучке редагування.

AI-рішення, як-от beautiful ai, tome чи gamma, орієнтовані на автоматизацію. Вони дозволяють зменшити участь користувача у технічному оформленні, автоматично вирівнюють вміст, підбирають кольори, компонують слайди відповідно до вбудованих дизайнерських принципів. Такі сервіси значно скорочують час на підготовку презентації та дозволяють сфокусуватися на змісті, а не на формі.

Інтеграційні можливості також є важливим критерієм. Наприклад, Google Slides тісно інтегрується з іншими сервісами Google, а SlidesAI працює всередині нього як плагін. У свою чергу tome дозволяє вставляти динамічний контент з різних платформ, а gamma підтримує редагування в реальному часі. Ці функції визначають зручність у роботі над презентацією в команді та гнучкість у підготовці матеріалу.

Ще однією ключовою відмінністю є рівень кастомізації. Традиційні інструменти надають повну свободу щодо кожного елемента слайда, але

вимагають дизайнерських навичок. AI-інструменти обмежують ручне втручання, натомість забезпечують узгоджений візуальний стиль та естетику. Вибір між цими підходами залежить від цілей користувача, швидкість і автоматизація чи повний контроль і точність.

### 1.2.4 Визначення сильних та слабких сторін існуючих рішень

У процесі оцінки сучасних сервісів для створення презентацій важливо враховувати як їхні сильні, так і слабкі сторони. Традиційні програми мають перевагу у стабільності, широких функціональних можливостях і тривалій підтримці користувачів. Вони добре інтегруються з операційними системами та корпоративними інструментами, а також дозволяють працювати офлайн. Проте їхнім недоліком є потреба у значних часових витратах і дизайнерських навичках.

AI-рішення, навпаки, вирізняються швидкістю та зручністю. Вони автоматично формують структуру слайдів, підбирають стиль оформлення, генерують зміст і адаптують його під цільову аудиторію. Це особливо корисно для користувачів без досвіду у створенні презентацій. Разом з тим, такі сервіси іноді обмежують гнучкість редагування та не завжди дають змогу повністю контролювати кінцевий результат.

Серед сильних сторін інструментів на основі штучного інтелекту – це інтуїтивний інтерфейс, можливість швидко створити презентацію з мінімальним вхідним контентом, а також готові шаблони з візуально привабливим дизайном.

Проте їхня ефективність безпосередньо залежить від якості вхідних даних та алгоритмів, які використовуються для генерації. У деяких випадках презентації можуть виглядати занадто шаблонно або недостатньо адаптовано до конкретного контексту. Таким чином, вибір інструменту залежить від балансу між гнучкістю та автоматизацією.

## 1.3 Дослідження вимог цільової аудиторії до презентацій

Успішність сервісів для створення презентацій значною мірою залежить від здатності задовольняти реальні потреби користувачів. У сучасних умовах зростає попит на інструменти, що поєднують швидкість, зручність та інтуїтивну взаємодію з системою. Різні групи користувачів, такі як студенти, викладачі, працівники офісу, маркетологи чи менеджери, мають свої очікування щодо функціоналу, дизайну та ефективності.

Окрему увагу користувачі приділяють можливостям кастомізації, адже адаптація презентації під контекст, стиль спілкування або тип аудиторії підвищує ефективність комунікації. Важливими також є зручність інтерфейсу, підтримка популярних форматів, наявність шаблонів і можливість роботи з мультимедійним контентом. Багато хто очікує, що платформа допоможе їм не лише створити презентацію, а й зробити її візуально привабливою без додаткових знань у сфері дизайну.

Крім цього, користувачі цінують функціональність, яка дозволяє економити час і мінімізувати рутинні дії. Це включає автоматичне генерування слайдів, адаптивні шаблони, вбудовані поради щодо структури та логіки подачі інформації. Загальне прагнення до оптимізації процесу створення презентацій формує нові вимоги до інтерфейсу, технічних можливостей і гнучкості сервісів.

### 1.3.1 Сегментація цільової аудиторії сервісу

Сегментація цільової аудиторії є ключовим етапом у проектуванні та розвитку сервісу для створення презентацій. Розуміння того, хто є користувачем, дозволяє адаптувати функціонал, інтерфейс і стиль подачі під конкретні потреби. Основні сегменти можна умовно поділити на освітній, корпоративний і креативний, кожен з яких має свої запити до зручності, швидкості, гнучкості та якості візуального оформлення.

До освітнього сегменту належать студенти, учні, викладачі та тренери. Для них важлива можливість швидко створити навчальну презентацію, яка буде простою у сприйнятті, відповідатиме вимогам академічного стилю та дозволятиме зручно структурувати матеріал. Такі користувачі також цінують доступність сервісу, мінімальні технічні вимоги та наявність готових шаблонів з освітньою тематикою.

Корпоративний сегмент включає менеджерів, аналітиків, маркетологів, HR-фахівців і представників малого та середнього бізнесу. У цьому випадку презентації використовуються для звітів, пітчів, стратегічних сесій і внутрішньої комунікації. Такі користувачі очікують професійного вигляду, підтримки брендового стилю, можливості роботи з графіками та аналітикою, а також інтеграції з іншими бізнес-інструментами.

### 1.3.2 Основні запити користувачів до функціональності

Основні запити користувачів до функціональності сервісів створення презентацій концентруються навколо простоти використання та економії часу. Більшість користувачів прагнуть мінімізувати кількість дій, необхідних для отримання якісного результату. Це включає автоматичне формування структури презентації, генерацію змісту на основі короткого опису теми та наявність готових візуальних шаблонів для різних випадків використання.

Не менш важливим є запит на гнучкість і можливість персоналізації. Користувачі очікують мати змогу адаптувати стиль презентації під цільову аудиторію, обирати теми оформлення, кольорові схеми, шрифти, а також редагувати слайди вручну після автоматичної генерації. Наявність таких можливостей дозволяє зробити кінцевий результат більш релевантним контексту виступу та індивідуальним за стилем.

Ще одним поширеним запитом є підтримка мультимедійних елементів. Сучасні презентації часто містять зображення, відео,

інтерактивні блоки та графіки, тому користувачі очікують, що сервіс легко інтегруватиме подібний контент без необхідності складної обробки. Важливою є також сумісність з популярними форматами експорту, такими як PDF, PPTX та Keynote, що дозволяє зручно використовувати презентації у різних робочих середовищах.

### 1.3.3 Вимоги до кастомізації презентаційного контенту

Користувачі все частіше висувають високі вимоги до можливостей кастомізації презентаційного контенту. Важливою є здатність налаштовувати не лише загальний стиль презентації, а й окремі її елементи, наприклад, шрифти, кольори, структуру слайдів, макети тексту та розташування зображень. Очікується, що система дозволить легко змінювати теми оформлення, додавати фірмову айдентику, адаптувати подачу матеріалу відповідно до специфіки аудиторії або цілей виступу.

Крім базових налаштувань оформлення, користувачі прагнуть мати контроль над змістом слайдів після автоматичної генерації. Це включає можливість редагувати тексти, змінювати послідовність слайдів, обирати варіанти структури або додавати власні блоки інформації.

### 1.3.4 Потреби користувачів у підтримці різних форматів

Однією з ключових потреб користувачів є підтримка різних форматів збереження та експорту презентацій. Це зумовлено різноманіттям платформ, на яких демонструються слайди, а також вимогами до сумісності з програмним забезпеченням аудиторії або замовників. Найбільш затребуваними є формати PDF, PPTX та Keynote, оскільки вони забезпечують універсальність і зручність перегляду на різних пристроях без втрати структури чи стилю оформлення.

Окрім класичних форматів, користувачі дедалі частіше очікують можливість інтеграції з хмарними сервісами, як-от Google Drive, Dropbox або OneDrive, для збереження та спільного редагування матеріалів. У деяких випадках важливо також мати доступ до інтерактивних або веб-орієнтованих форматів, які дозволяють демонструвати презентації онлайн без необхідності встановлення додаткового програмного забезпечення. Гнучка підтримка форматів забезпечує адаптивність сервісу до різних сценаріїв використання.

## 1.4 Формування вимог до функціональності сервісу

Формування вимог до функціональності сервісу для створення презентацій базується на результатах аналізу ринку, дослідження потреб користувачів і оцінки сучасних технологічних можливостей. Основна мета полягає у створенні інтуїтивно зрозумілого і ефективного інструменту, який забезпечить мінімальні зусилля для отримання якісного результату. Важливою особливістю має стати підтримка як автоматизованого, так і ручного налаштування елементів презентації.

Функціональні вимоги охоплюють можливості авторизації, збереження історії створених матеріалів, генерації прототипів презентацій на основі заданих параметрів та подальшої їх обробки. Сервіс повинен забезпечувати адаптацію контенту під різні стилі, аудиторії та формати подання. Користувач має отримати інструмент для швидкої побудови презентацій з можливістю редагування на різних етапах.

Крім базових можливостей, важливо передбачити функції експорту презентацій у популярні формати та інтеграцію з хмарними сховищами для зручності збереження і спільної роботи. Також перспективним напрямом є реалізація візуальної кастомізації готових презентацій та підтримка нотаток або побажань від користувача під час формування фінального варіанту.

### 1.4.1 Авторизація та керування історією створених матеріалів

Однією з основних функціональних вимог до сервісу є реалізація механізму авторизації користувачів та збереження історії створених матеріалів. Це дозволяє забезпечити персоналізацію досвіду роботи, надати доступ до раніше згенерованих прототипів і фінальних презентацій, а також зберігати налаштування для подальшого редагування або повторного використання. Авторизація може бути реалізована через реєстрацію за допомогою електронної пошти або соціальних мереж, що сприятиме підвищенню зручності доступу до особистого кабінету користувача.

### 1.4.2 Генерація прототипів презентацій за заданими параметрами

Генерація прототипів презентацій за заданими параметрами є ключовим функціональним елементом сервісу, що визначає його унікальність та практичну цінність. Користувач повинен мати можливість вказати тему, опис, бажану кількість слайдів, тип аудиторії, стиль тексту та формат подачі інформації. На основі цих вхідних даних система автоматично формує логічно структурований прототип, який надалі може бути використаний для створення повноцінної презентації. Такий підхід дозволяє значно скоротити час на початковий етап підготовки матеріалу.

### 1.4.3 Структурування контенту у вигляді слайдів

Структурування контенту у вигляді слайдів є важливим етапом після генерації прототипу презентації. Сервіс має забезпечувати автоматичний розподіл інформації за слайдами відповідно до логічної послідовності подачі матеріалу. Кожен слайд повинен містити заголовок, текстовий зміст у вигляді абзаців або списків, а також, за потреби, візуальні елементи.

### 1.4.4 Автоматичне створення фінальної презентації з налаштуванням стилю

Автоматичне створення фінальної презентації з налаштуванням стилю є завершальним етапом роботи користувача з сервісом. Після затвердження прототипу користувач має мати можливість обрати візуальний стиль, кольорову палітру, тип макетів і надати додаткові побажання щодо оформлення. Сервіс автоматично застосовує ці параметри до структури слайдів, формуючи готову презентацію, яка поєднує логіку викладення матеріалу з професійним зовнішнім виглядом. Така функціональність сприяє підвищенню якості кінцевого результату при мінімальних витратах часу та зусиль з боку користувача.

### 1.4.5 Експорт презентацій у різні формати файлів

Експорт презентацій у різні формати файлів є необхідною функцією для забезпечення зручності використання створених матеріалів у різних середовищах. Сервіс має підтримувати експорт у такі формати, як PDF, PPTX та Keynote, що дозволить користувачам обирати оптимальний варіант залежно від технічних вимог або особистих уподобань. Забезпечення коректного відображення структури, дизайну та мультимедійних елементів під час експорту є важливою умовою для збереження якості презентацій.

Така можливість розширює сферу застосування сервісу та підвищує його привабливість для різних категорій користувачів. Крім того, підтримка різних форматів підвищує мобільність користувача, адже презентації можуть бути продемонстровані з будь-якого пристрою або програмного забезпечення без додаткової конверсії. Це також сприяє інтеграції у вже наявні робочі процеси в закладах освіти, бізнес-середовищі чи під час публічних виступів.

# 2 ПРОЕКТУВАННЯ СИСТЕМИ

## 2.1 Аналіз функціональних та нефункціональних вимог

Етап аналізу вимог є критично важливим у процесі розробки будь-якої програмної системи, оскільки саме на цьому етапі визначаються очікування користувачів і технічні обмеження, які повинна враховувати система. Правильно сформульовані вимоги дозволяють уникнути неоднозначності у подальших етапах проектування та реалізації, а також забезпечують узгодженість між очікуваннями замовника та функціональністю кінцевого продукту.

У рамках розробки веб-сервісу для створення презентацій на основі штучного інтелекту були виокремлені дві основні групи вимог , а саме функціональні та нефункціональні. Функціональні вимоги описують конкретні можливості системи, які мають бути реалізовані для задоволення потреб користувача, такі як генерація прототипів, авторизація, збереження історії тощо. Вони формують основу для взаємодії користувача з системою та визначають ключові сценарії її використання.

Нефункціональні вимоги стосуються характеристик, що визначають якість функціонування системи – продуктивність, зручність інтерфейсу, масштабованість, безпека тощо. Їх дотримання є критично важливим для забезпечення стабільної, ефективної та комфортної роботи сервісу в умовах реального навантаження.

Зібрані вимоги формувалися на основі аналізу ринку існуючих рішень, вивчення поведінкових патернів цільової аудиторії, а також з урахуванням технологічних можливостей обраного стеку. Їх структуроване представлення дозволяє перейти до деталізації кожної групи вимог окремо, що забезпечує основу для подальшого моделювання, архітектурного проектування і реалізації системи.

### 2.1.1 Функціональні вимоги

Функціональні вимоги визначають основний набір можливостей, які повинна забезпечувати система для виконання поставлених задач. У випадку розробки AI-сервісу для створення презентацій основною вимогою є генерація прототипу презентації на основі заданої теми, кількості слайдів, стилістики оформлення та інших параметрів, які користувач вказує під час формування запиту. Результатом цього процесу має бути структурована послідовність слайдів із запропонованим текстовим наповненням і базовою візуалізацією.

Іншою ключовою вимогою є забезпечення функціоналу авторизації користувачів. Кожен користувач повинен мати змогу створити обліковий запис або увійти в систему за допомогою зовнішніх сервісів авторизації. Це дає змогу зберігати історію попередніх запитів, повертатися до створених презентацій, редагувати їх або експортувати у зручному форматі. Авторизація також є передумовою для персоналізації результатів і захисту даних.

Важливою функціональною вимогою є можливість редагування створених слайдів. Після автоматичної генерації користувач повинен мати змогу вносити корективи у зміст, змінювати порядок слайдів, обирати інший стиль оформлення або перегенерувати окремі слайди без втрати всієї роботи. Це забезпечує гнучкість і дозволяє адаптувати результат до конкретного контексту використання.

Останньою, але не менш значущою функціональною вимогою є експорт готової презентації у популярні формати, такі як PDF або PPTX. Це необхідно для подальшого використання створеного матеріалу у різноманітних системах демонстрації чи передавання іншим користувачам. Крім того, система повинна підтримувати можливість збереження презентації у хмарному середовищі для доступу з різних пристроїв.

### 2.1.2 Нефункціональні вимоги

Нефункціональні вимоги визначають характеристики системи, які не пов’язані безпосередньо з функціональністю, але істотно впливають на її зручність використання, ефективність, стабільність та безпеку. Однією з головних вимог є зручність інтерфейсу, оскільки система орієнтована на широку аудиторію з різним рівнем технічної підготовки. Інтерфейс повинен бути інтуїтивно зрозумілим, з логічною навігацією та адаптованим під мобільні пристрої.

Ще однією важливою нефункціональною вимогою є продуктивність. Система повинна швидко обробляти запити користувача і формувати результати генерації презентацій у максимально короткі терміни. Це особливо актуально для користувачів, які працюють у динамічному середовищі і потребують оперативного створення презентацій для зустрічей, доповідей чи звітів. Забезпечення низької затримки відповіді та швидкої генерації є критерієм якості роботи сервісу.

Надійність системи також є критично важливою вимогою. Веб-сервіс має стабільно функціонувати при високому навантаженні, правильно обробляти помилки, не допускати втрати даних і мати механізми резервного збереження. Для підвищення надійності доцільно використовувати хмарну інфраструктуру, яка забезпечує масштабованість та безперервний доступ до ресурсу.

Ще однією групою нефункціональних вимог є вимоги безпеки. Оскільки система працює з персональними обліковими записами та збереженими даними користувача, необхідно впровадити заходи захисту інформації. Це включає захищену авторизацію, шифрування з’єднання, обмеження доступу до даних і використання перевірених сервісів аутентифікації. Дотримання цих вимог підвищує довіру до сервісу та забезпечує захист даних у різних сценаріях використання.

## 2.2 Вибір інструменту генерації презентаційного контенту

Однією з ключових особливостей системи є функція автоматичного створення презентаційного контенту за допомогою технологій штучного інтелекту. Для реалізації цієї можливості необхідно інтегрувати зовнішній інструмент, який здатен генерувати осмислений текст на основі запиту користувача. Такий інструмент повинен підтримувати роботу з природною мовою, бути гнучким у налаштуваннях і забезпечувати швидкий відгук у рамках запитів веб-застосунку.

У процесі проектування було розглянуто декілька варіантів інструментів генерації тексту, кожен з яких має свої переваги та обмеження щодо інтеграції, точності результату, вартості використання і можливостей кастомізації. Особливу увагу було приділено тому, як обраний сервіс взаємодіє з серверною частиною системи, які формати запитів і відповідей підтримуються, а також наскільки стабільною є якість створеного контенту за різних тем.

Далі у відповідних підрозділах буде проаналізовано наявні рішення для генерації тексту слайдів з точки зору їх технічної сумісності, ліцензійних умов та функціональних можливостей. Такий аналіз дозволяє аргументовано обґрунтувати вибір конкретного сервісу, який буде використано в системі для реалізації основної функції – створення наповнення презентацій.

Обґрунтований вибір інструменту є критично важливим для загальної якості користувацького досвіду, адже саме від якості згенерованого контенту залежить ефективність використання системи. Крім того, правильний вибір дозволяє забезпечити масштабованість, розширюваність і надійність роботи AI-компоненту в довготривалій перспективі. Це особливо важливо з огляду на постійне зростання обсягів даних і потреб користувачів у персоналізованому контенті.

### 2.2.1 Аналіз доступних рішень

У сфері генерації текстового контенту для презентацій на основі штучного інтелекту існує низка популярних рішень, які можна інтегрувати у веб-застосунок. Найбільш відомими серед них є OpenAI API, Cohere, AI21 Labs та Google PaLM API. Кожен із зазначених сервісів надає інтерфейс для генерації тексту природною мовою за заданими параметрами, з можливістю налаштування довжини відповіді, стилю мови, рівня креативності та інших характеристик.

OpenAI API, зокрема модель GPT, є одним із найпопулярніших і стабільних варіантів, широко інтегрованих у різні застосунки по всьому світу. Вона вирізняється високою якістю сформованого тексту, здатністю адаптуватися до специфіки запиту користувача, підтримкою кількох мов і наявністю гнучких механізмів контролю над генерацією. До переваг належать також активна документація, підтримка від спільноти й безперервне оновлення моделей.

Альтернативні сервіси, такі як Cohere або AI21 Labs, також пропонують подібні можливості, але можуть відрізнятися за якістю тексту в спеціалізованих темах, швидкістю обробки запитів або вартістю використання. Деякі з них більше орієнтовані на аналітичні або короткі відповіді, що не завжди є доречним у контексті створення багатослайдових презентацій. Однак за певних сценаріїв вони можуть розглядатися як резервні або допоміжні сервіси.

Важливим критерієм вибору також є простота інтеграції у веб-застосунок. OpenAI, наприклад, має добре структуровані SDK та REST API, що дозволяє легко впровадити генерацію тексту у бекендову логіку без значних витрат часу. Наявність чіткої політики ліцензування, системи обмеження запитів і прозорого ціноутворення робить OpenAI найбільш зручним кандидатом для впровадження в рамках дипломного проекту.

### 2.2.2 Обґрунтування вибору OpenAI API

Вибір OpenAI API як основного інструменту для генерації текстового контенту у презентаціях є обґрунтованим завдяки його високій якості результатів, гнучкості та широкій підтримці серед розробників. Модель GPT здатна створювати логічно структуровані та зрозумілі тексти, які підходять для автоматичного заповнення слайдів презентацій. Це дозволяє користувачу за мінімального зусилля отримувати повноцінний вміст відповідно до заданої тематики.

Ще однією перевагою є наявність добре задокументованого REST API, що значно спрощує процес інтеграції зі сторонніми веб-застосунками. OpenAI надає гнучкі параметри конфігурації запитів, зокрема керування довжиною відповіді, температурою генерації, кількістю варіантів та інші параметри, що дозволяє адаптувати генерацію під потреби користувача. Це забезпечує можливість налаштування вихідного результату під формат, стиль і зміст слайдів.

Також важливим чинником є підтримка української мови, що особливо актуально для локалізованих презентацій. На відміну від деяких інших моделей, OpenAI демонструє стабільну якість генерації не лише англійською, а й іншими мовами, що розширює сферу використання. Крім того, сервіс має механізми обмеження запитів, системи оплати за використання та забезпечує високу надійність з технічної точки зору.

У межах дипломного проекту інтеграція OpenAI дозволяє швидко протестувати повний цикл генерації презентацій без потреби у власному навчанні моделей або складних обчислювальних ресурсах. Це рішення є оптимальним з погляду співвідношення функціональності, надійності, якості контенту та часу на впровадження. Вибір цього інструменту дозволяє зосередитися на побудові логіки системи та дослідженні користувацьких сценаріїв замість технічної реалізації моделей з нуля.

## 2.3 Побудова функціональної моделі системи

У процесі функціонального моделювання було визначено основні ролі користувачів та взаємодії з системою. Основною роллю виступає авторизований користувач, який має можливість генерувати презентації, переглядати історію раніше створених матеріалів, редагувати слайди та експортувати готові файли у потрібному форматі. Додатково враховується роль неавторизованого користувача, який має доступ лише до ознайомлювального інтерфейсу або функціоналу реєстрації та входу в систему.

Однією з ключових функцій є генерація прототипу презентації за заданими параметрами. Користувач вводить тему, кількість слайдів, бажаний стиль оформлення та інші опціональні налаштування. Після цього система формує попередню версію презентації із текстовим наповненням слайдів, враховуючи зазначені дані. Результат може бути переглянутий, відредагований або збережений у обліковому записі користувача.

Важливим функціональним блоком є керування створеними презентаціями. Авторизований користувач може переглядати історію створених прототипів, повторно відкривати їх для редагування або експортувати у зручному форматі. Передбачено також можливість видалення застарілих матеріалів або створення копій презентацій для подальших змін без впливу на оригінал.

Функціональна модель передбачає також процес налаштування фінального оформлення презентації. Користувач має змогу обрати візуальну тему, змінити структуру окремих слайдів або адаптувати стиль до вимог конкретної аудиторії. Ці зміни фіксуються в системі і стають частиною підсумкового варіанту, який буде запропоновано до експорту.

Інтеграція зі сторонніми сервісами є ще одним важливим сценарієм, що відображений у функціональній моделі. Зокрема, взаємодія з OpenAI API дозволяє забезпечити генерацію текстового контенту на основі вхідних

даних користувача. З технічного боку, це реалізується через запит до мовної моделі, отримання згенерованих відповідей і включення їх до структури слайдів.

Функціональне моделювання завершилось побудовою діаграми варіантів використання, що узагальнює основні сценарії взаємодії користувача із системою. На ній відображено взаємозв'язки між ролями та діями, які вони можуть виконувати, а також уточнено межі відповідальності між клієнтською частиною, бекендом і зовнішніми API. Такий підхід дозволяє сформувати цілісне уявлення про роботу системи на рівні користувацької логіки.

### 2.3.1 Діаграми варіантів використання

Для візуалізації функціональної моделі системи було створено діаграму варіантів використання, що відображає ключові сценарії взаємодії користувача з AI-сервісом. Цей тип діаграми дозволяє визначити, які саме функції доступні кінцевому користувачу, які дії виконує система у відповідь, а також які зовнішні компоненти залучені до процесу. Основним актором виступає користувач, який ініціює процес генерації презентації, здійснює авторизацію, редагує слайди та експортує результати у бажаному форматі.

Крім користувача, у діаграмі представлено зовнішній актор, тобто сторонній API, зокрема OpenAI, який виконує роль генератора текстового контенту для слайдів. Використання зовнішнього сервісу дозволяє досягти високого рівня персоналізації та якості згенерованого матеріалу. Модель демонструє, що взаємодія з API відбувається безпосередньо під час генерації прототипу презентації, а всі інші функції реалізуються у межах самої системи, що є саме необхідністю для розроблюваної системи нашого сервісу.

Далі можна побачити варіанти використання, згруповані навколо центрального сервісу створення презентацій (рисунок 2.1). Кожна функція, така як введення параметрів, редагування, збереження чи експорт, має окремий зв’язок із користувачем, що вказує на можливість її безпосереднього виклику. Діаграма дозволяє швидко зорієнтуватися в доступному функціоналі сервісу та окреслити межі відповідальності системи й зовнішніх компонентів.

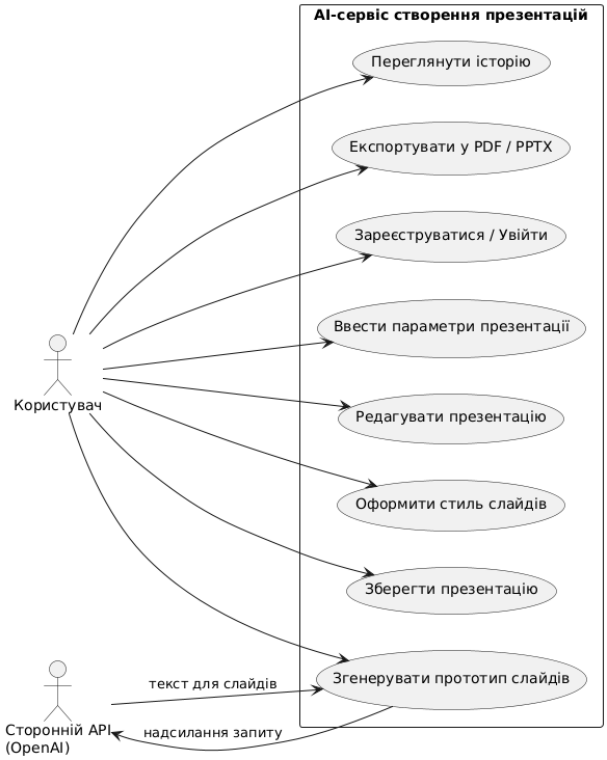


Рисунок 2.1 – Діаграма варіантів використання AI-сервісу створення презентацій

Для глибшого розуміння логіки функціонування системи доцільно застосовувати діаграми послідовності. Цей тип діаграм дозволяє наочно продемонструвати обмін повідомленнями між окремими компонентами системи та користувачем у межах конкретного сценарію використання. Він відображає не лише загальні функціональні блоки, але й деталізує порядок їх виклику та залежності між підсистемами.

Далі можна побачити послідовність дій (рисунок 2.2), яка охоплює процеси генерації презентації, редагування вмісту слайдів і експорту результату. У діаграмі представлено п’ять основних учасників: користувач, клієнтський веб-інтерфейс, серверна частина застосунку, сторонній API та база даних презентацій. Відображено взаємозв’язки між діями користувача і реакцією системи у вигляді послідовних запитів та відповідей.

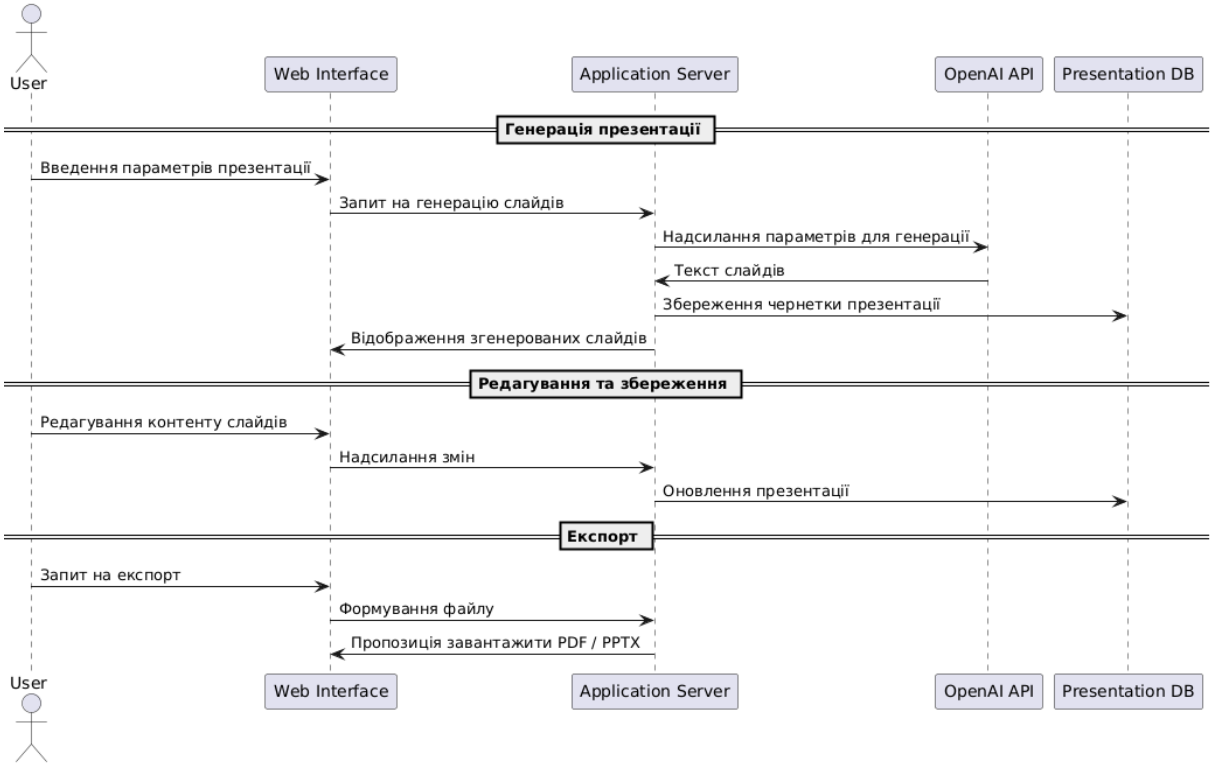


Рисунок 2.2 – Діаграма послідовності взаємодії користувача з сервісом

У процесі генерації презентації користувач через веб-інтерфейс передає параметри, які надходять на сервер. Після цього сервер формує запит до OpenAI API для генерації тексту слайдів, зберігає отримані результати у базі даних і передає їх назад до інтерфейсу для відображення користувачу. Цей ланцюг дій забезпечує повну автоматизацію первинного створення матеріалу, мінімізуючи час і зусилля користувача.

Після створення прототипу користувач може редагувати вміст слайдів, що викликає оновлення даних на сервері та збереження змін у базі.

Останнім кроком є експорт, під час якого система формує підсумковий файл у форматі PDF або PPTX, готовий для завантаження. Така структурована логіка забезпечує злагоджену роботу всіх компонентів системи й гарантує стабільність сервісу навіть при багаторазовій взаємодії користувача з ним, що є необхідністю для сьогодення.

## 2.4 Архітектурне рішення веб-застосунку

Архітектура веб-застосунку побудована за принципом розділення на клієнтську та серверну частини з інтеграцією зовнішніх сервісів. Такий підхід забезпечує масштабованість, гнучкість розробки та можливість повторного використання окремих компонентів. Веб-інтерфейс відповідає за взаємодію з користувачем, обробку введених даних, відображення результатів та ініціацію основних дій, пов’язаних із генерацією та редагуванням презентацій.

Далі можна побачити архітектурне рішення, що відображає взаємозв’язки між ключовими складовими системи (рисунок 2.3).

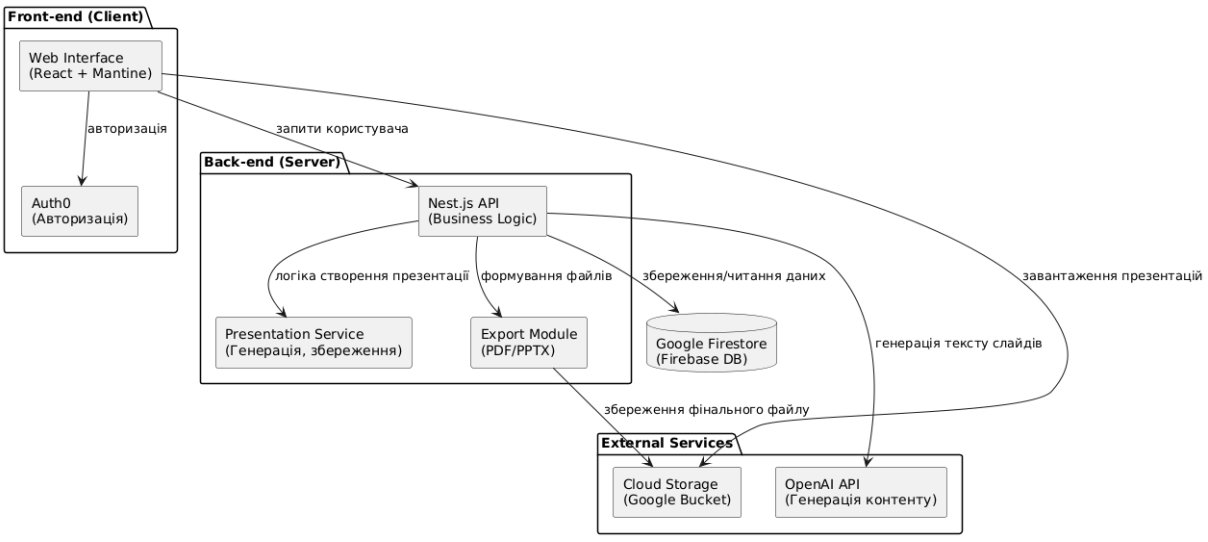


Рисунок 2.3 – Архітектура сервісу

У центрі знаходиться серверна частина, реалізована на основі Nest.js, яка виконує функції логіки обробки запитів, координації взаємодії з базою даних Firestore, модулем експорту та зовнішнім API. З боку клієнта реалізований інтерфейс на React із використанням UI-бібліотеки Mantine. Для авторизації залучено сервіс Auth0, що забезпечує безпечний доступ до функціоналу системи.

Ключовим компонентом є Presentation Service, який відповідає за генерацію та збереження проміжного стану презентацій. Для формування фінального документа використовується Export Module, який на основі наданих користувачем параметрів створює файли у форматі PDF або PPTX. Згенеровані файли зберігаються у хмарному сховищі Google Cloud Bucket, звідки можуть бути завантажені користувачем через клієнтський інтерфейс.

Окрему роль відіграє інтеграція зі стороннім API OpenAI, який забезпечує генерацію змістовного наповнення слайдів на основі введеної теми. Сервер формує запит до API з параметрами, отриманими від користувача, отримує відповідь у вигляді тексту і передає її в Presentation Service для подальшої обробки. Така структура дозволяє розширювати функціонал у майбутньому без значного перегляду загальної архітектури системи.

## 2.5 Структурне проектування бази даних

Проектування структури бази даних є одним із ключових етапів розробки веб-застосунку, оскільки саме від неї залежить ефективність збереження, доступу та оновлення інформації, яка циркулює в системі. Для реалізації проекту було обрано документоорієнтовану базу даних Google Firestore, що є частиною Firebase-платформи. Такий вибір зумовлений потребою в гнучкому зберіганні напівструктурованих даних, масштабованістю та можливістю інтеграції з іншими сервісами екосистеми Google.

Основними сутностями, що зберігаються в базі даних, є користувачі, проекти презентацій та їхні складові. Колекція користувачів містить інформацію про унікальний ідентифікатор, електронну пошту, дату реєстрації та базові налаштування. Ці дані синхронізуються із сервісом авторизації Auth0, проте дублюються у Firestore з метою збереження інформації, що не належить до автентифікації, наприклад, історія створених презентацій.

Презентації зберігаються у вигляді окремої колекції з прив'язкою до користувача, що дозволяє відображати історію конкретного профілю. Кожна презентація має унікальний ідентифікатор, назву, дату створення, статус (чернетка, завершена), а також поле з масивом слайдів. Такий підхід дозволяє зберігати всю презентацію як один документ, що спрощує доступ і оновлення під час редагування.

Слайди реалізуються як вкладені об'єкти в структурі документа презентації. Кожен слайд має номер, тип (заголовок, текстовий блок, список тощо), текстове наповнення та метадані щодо стилю чи іконографіки. Така структура дозволяє легко додавати нові слайди, змінювати порядок або видаляти непотрібні, не порушуючи загальної логіки документа.

Окремо ведеться колекція запитів генерації, яка включає параметри, що були передані до OpenAI API. Вона служить як журнал взаємодії користувача з мовною моделлю та може бути використана для відновлення процесу генерації або для аналітики. Збереження таких запитів є доцільним із точки зору відлагодження, а також для майбутньої оптимізації сценаріїв генерації. Крім основних колекцій, система може містити допоміжні таблиці, наприклад, шаблони оформлення, конфігурації експорту або попередньо збережені стилі. Їхнє зберігання в окремих колекціях дозволяє перевикористовувати ці дані між користувачами та забезпечувати централізоване оновлення без необхідності втручання в особисті документи. Такий підхід забезпечує гнучкість, розширюваність і високу продуктивність при роботі з великим обсягом інформації.

# 3 ПРОГРАМНА РЕАЛІЗАЦІЯ

## 3.1 Стек реалізації AI-сервісу

У процесі реалізації веб-сервісу для автоматизованого створення презентацій було застосовано сучасний технологічний стек, що відповідає вимогам до продуктивності, масштабованості та інтеграції зі сторонніми інтелектуальними сервісами. Вибір інструментів обумовлений необхідністю забезпечення зручної роботи як з клієнтською частиною, так і з серверною логікою, а також підтримки генерації тексту та експорту результату у відповідному форматі.

Технології, що увійшли до складу проекту, дозволяють забезпечити повноцінну роботу з авторизацією, введенням параметрів, обробкою запитів, генерацією змісту слайдів, їх візуалізацією та збереженням. Поєднання інструментів фронтенду і бекенду створює гнучку та взаємозалежну систему, де кожен компонент виконує свою функцію і підтримує загальну логіку взаємодії з користувачем.

У подальших підрозділах наведено огляд основних технологій, що були використані при розробці системи. Розглядаються їхні переваги, особливості реалізації, а також роль у побудові функціонального та надійного AI-сервісу.

### 3.1.1 Мова програмування TypeScript

Основною мовою програмування, що була використана при реалізації веб-сервісу, є TypeScript. Це надмножина JavaScript, яка додає статичну типізацію і дозволяє виявляти помилки ще на етапі компіляції. Такий підхід забезпечує вищу надійність коду, його передбачуваність та полегшує масштабування застосунку. Завдяки використанню TypeScript розробник

отримує точні підказки в редакторі коду, що пришвидшує розробку та зменшує кількість потенційних помилок.

TypeScript дозволяє будувати чітку структуру проекту, розділяючи відповідальність між модулями та зберігаючи логічні зв’язки між компонентами. Завдяки системі інтерфейсів, типів і класів можна формалізувати взаємодію між клієнтською та серверною частинами, уніфікувати запити до API, описати структуру відповідей і підвищити загальну читаємість коду. Це особливо важливо при розробці складних систем з багатьма точками інтеграції.

Важливою перевагою TypeScript є його сумісність з існуючим JavaScript-кодом. Це дає змогу поступово впроваджувати типізацію в проект або використовувати сторонні бібліотеки, написані на JavaScript, без втрати гнучкості. Крім того, активна спільнота, велика кількість інструментів для типізації та офіційна підтримка у сучасних редакторах коду роблять TypeScript зручним у використанні навіть для невеликих команд.

Застосування TypeScript забезпечило стабільність, масштабованість і передбачувану поведінку системи як на стороні клієнта, так і на стороні сервера. Саме ця мова програмування дала змогу ефективно організувати обробку даних, формалізувати роботу з API, структурувати кодову базу та підтримувати єдиний стиль розробки у межах усього застосунку.

### 3.1.2 Фреймворк React для створення клієнтської частини

Фреймворк React був обраний для реалізації клієнтської частини веб-застосунку завдяки своїй гнучкості, компонентній архітектурі та широким можливостям інтеграції з іншими бібліотеками. Він дозволяє створювати реактивний інтерфейс, що динамічно оновлюється відповідно до дій користувача без необхідності перезавантаження сторінки. Це особливо важливо для сучасних застосунків, де швидкість і зручність взаємодії відіграють ключову роль.

React забезпечує ефективне управління станом компонента, що дозволяє ізольовано реалізовувати окремі функціональні блоки, зокрема форми, кнопки керування, поля вводу параметрів теми презентації або стилю слайдів. Такий підхід сприяє багаторазовому використанню коду, його тестуванню та підтримці. У веб-сервісі компоненти React були використані для побудови усіх базових елементів взаємодії користувача з системою.

Особливістю React є використання віртуального DOM, що значно пришвидшує рендеринг змін на сторінці. Завдяки цьому можна оперативно відображати результати генерації слайдів, змін оформлення чи повідомлення про помилки без затримок. Це підвищує загальну продуктивність інтерфейсу та забезпечує приємний користувацький досвід, особливо в контексті взаємодії з динамічними даними, що надходять із серверної частини або зовнішніх API.

Інтеграція React із сучасним інструментом збирання, такими як Vite, а також підтримка TypeScript дозволили організувати робоче середовище, яке сприяє швидкому і стабільному розгортанню клієнтської частини. Усе це робить React одним із найкращих рішень для побудови сучасних веб-інтерфейсів, зокрема в рамках AI-сервісів, де важливо поєднувати інтерактивність, динамічність і логічну структуру.

### 3.1.3 UI-бібліотека Mantine для побудови інтерфейсу

UI-бібліотека Mantine була використана для побудови інтерфейсу клієнтської частини веб-застосунку з метою забезпечення сучасного вигляду, адаптивності та швидкої розробки типових елементів інтерфейсу. Mantine пропонує готові компоненти, які можна легко кастомізувати, а також має підтримку темізації, що дозволяє створювати єдиний візуальний стиль для всього застосунку без значних витрат часу.

Завдяки широкому набору компонентів, таких як кнопки, поля вводу, дропдауни, модальні вікна, вкладки та повідомлення, Mantine дозволила реалізувати інтуїтивно зрозумілу структуру взаємодії з користувачем. Це особливо важливо у системах, де потрібна чітка послідовність дій, наприклад, введення теми, налаштування параметрів слайдів, перегляд результатів і експорт. Усі елементи інтерфейсу були стилізовані відповідно до загального дизайну завдяки можливостям кастомізації, які пропонує бібліотека.

Однією з переваг Mantine є вбудована підтримка системи керування формами, обробки помилок, валідації полів, а також адаптивного дизайну для мобільних пристроїв. Це дозволяє не лише покращити зовнішній вигляд інтерфейсу, але й забезпечити його функціональність і доступність для різних категорій користувачів. Крім того, бібліотека має хорошу документацію, що значно полегшує процес розробки навіть для нових учасників команди.

Використання Mantine в проекті дозволило скоротити час на верстку та стилізацію інтерфейсу, зосередивши зусилля на логіці роботи системи. Компонентний підхід цієї бібліотеки добре поєднується з архітектурою React, дозволяючи будувати складні інтерфейси з повторно використовуваних елементів. Це зробило Mantine оптимальним вибором для реалізації візуальної частини AI-сервісу.

### 3.1.4 Сервіс авторизації Auth0

Сервіс авторизації Auth0 був використаний для реалізації механізму автентифікації та управління доступом користувачів у веб-застосунку. Його інтеграція дозволила швидко налаштувати безпечний вхід до системи з підтримкою кількох способів реєстрації, включаючи електронну пошту, соціальні мережі та сторонні провайдери ідентифікації. Використання Auth0

дало змогу уникнути створення власної системи зберігання паролів і захисту облікових записів, що значно підвищило рівень безпеки застосунку.

Однією з головних переваг Auth0 є наявність готового інтерфейсу входу та реєстрації, який можна легко стилізувати під потреби застосунку. Це дозволило мінімізувати час на реалізацію візуальної частини авторизаційних процесів. Крім того, сервіс надає зручні засоби керування сесіями, токенами доступу та їх термінами дії, що дозволяє контролювати доступ до ресурсів і даних на стороні клієнта й сервера.

Усі авторизовані запити до серверної частини супроводжуються токенами, які генеруються Auth0, що дає змогу реалізувати механізм захисту маршрутів і персоналізованого доступу до функціоналу. Наприклад, збереження історії презентацій, повторна генерація або перегляд матеріалів доступні лише після входу в систему. Це забезпечує розмежування прав доступу та зберігає цілісність персональних даних користувача.

Інтеграція Auth0 із React і Nest.js відбувається через офіційні SDK, що значно спрощує реалізацію автентифікації на клієнті та перевірки токенів на сервері. Завдяки цьому вдалося швидко забезпечити захист маршрутизованих компонентів на фронтенді та обмежити доступ до певних API-ендпоінтів. Усе це робить Auth0 ефективним, надійним і зручним рішенням для управління автентифікацією у межах сучасного веб-застосунку.

### 3.1.5 Інструмент збирання Vite

Інструмент збирання Vite був використаний для створення і конфігурації клієнтської частини веб-застосунку. Це сучасний збирач проектів, який забезпечує миттєвий запуск середовища розробки, швидку обробку модулів та оптимізоване фінальне збирання застосунку для продакшну. Його основна перевага полягає у використанні ES-модулів у браузері під час розробки, що значно пришвидшує час оновлення сторінки.

Завдяки Vite було забезпечено ефективне управління залежностями, підтримку TypeScript, автоматичне оновлення коду без перезавантаження браузера та гнучку конфігурацію середовища. Інструмент добре інтегрується з React і бібліотеками, які були використані в застосунку, що дозволило розробникам зосередитись на реалізації логіки системи, а не на налаштуванні процесів складання проекту.

Окремою перевагою Vite є його зручна структура конфігураційних файлів і підтримка плагінів. Це дозволяє швидко додавати додаткову функціональність, наприклад, налаштування змінних середовища, підключення стилів, обробку ресурсів або запуск специфічних скриптів при збиранні. У проекті це використано для налаштування шляху до API та роботи з зовнішніми ресурсами.

У результаті використання Vite вдалося значно скоротити час на збирання застосунку, покращити продуктивність під час розробки та забезпечити швидке і стабільне розгортання клієнтської частини. Це зробило його ефективним інструментом у контексті побудови динамічного веб-сервісу, що працює з великою кількістю взаємодій на стороні клієнта.

### 3.1.6 Серверний фреймворк Nest.js

Серверна частина веб-сервісу реалізована з використанням фреймворку Nest.js, який побудований поверх Node.js і орієнтований на побудову масштабованих, модульних і добре структурованих серверних застосунків. Основною перевагою Nest.js є підтримка TypeScript і використання архітектурних підходів, притаманних об'єктно-орієнтованому програмуванню, таких як декоратори, залежності, модулі та контролери. Це дозволило організувати серверну логіку у вигляді окремих функціональних блоків із чітким розподілом відповідальності.

У Nest.js кожна частина логіки розміщена в окремому модулі: обробка запитів, інтеграція з OpenAI, збереження даних, генерація структури

слайдів, робота з хмарними сервісами. Це забезпечує гнучкість, легкість тестування і зручність подальшого розширення застосунку. Контролери обробляють HTTP-запити, сервісні класи реалізують бізнес-логіку, а модулі об'єднують ці елементи у функціонально завершені частини системи.

Завдяки вбудованій підтримці механізмів обробки запитів та проміжного програмного забезпечення Nest.js дозволяє легко реалізовувати авторизацію, перевірку даних, обробку помилок та логування. У межах даного проекту також реалізовано обробку токенів доступу від Auth0, захист маршрутів та перевірку автентичності користувача перед виконанням ключових операцій, зокрема генерації презентації або збереження її у базі даних.

Фреймворк Nest.js продемонстрував високу стабільність під час розробки і на етапах тестування. Його гнучкість дозволила інтегрувати сторонні API, організувати зв’язки з базою даних Firestore та забезпечити швидке оброблення запитів користувача. У підсумку використання Nest.js стало оптимальним вибором для побудови серверної логіки AI-сервісу з урахуванням вимог до розширюваності, безпеки та зрозумілості структури коду.

### 3.1.7 Генерація текстового контенту за допомогою OpenAI API

Генерація текстового контенту для слайдів презентації реалізується за допомогою інтеграції з OpenAI API. Цей сервіс надає доступ до великих мовних моделей, які здатні створювати осмислені, логічні та граматично правильні тексти на основі заданих параметрів. У рамках веб-сервісу OpenAI використовується для автоматичного створення змісту слайдів відповідно до теми, кількості слайдів та стилістики, яку обирає користувач.

Запит до OpenAI формується на сервері після надходження параметрів від користувача. У ньому міститься інструкція для моделі, в якій задається структура відповіді, довжина тексту та бажаний стиль. Відповідь моделі

надходить у вигляді одного або кількох абзаців, які далі розбиваються на окремі слайди відповідно до логіки побудови презентації. Генерація відбувається у реальному часі, і результат одразу повертається на клієнтську частину для попереднього перегляду.

Особливу увагу під час реалізації було приділено налаштуванню параметрів генерації, таких як температура, кількість токенів та формат відповіді. Це дозволило досягти балансу між креативністю та релевантністю згенерованого тексту. Крім того, враховувалась можливість генерації українською мовою, що є критично важливою умовою для локалізованого використання сервісу. Завдяки високій якості тексту, який створюють моделі OpenAI, користувач отримує готовий зміст для презентації без потреби додаткового редагування.

Використання OpenAI API значно розширило можливості системи і стало основною перевагою проекту порівняно з традиційними конструкторами презентацій. Інтеграція була реалізована через офіційний REST API, що дозволяє масштабувати рішення у майбутньому, додавати нові сценарії генерації або варіанти мовного оформлення. Такий підхід дозволив забезпечити гнучкість, автоматизацію та високу якість змістового наповнення презентацій.

### 3.1.8 Бібліотека PptxGenJS для формування презентацій

Для реалізації функціональності експорту презентацій у форматі PPTX у межах веб-сервісу була використана бібліотека PptxGenJS. Це клієнтська бібліотека на JavaScript, яка дозволяє динамічно створювати презентації з потрібною структурою, текстовим наповненням і оформленням безпосередньо у браузері або на стороні сервера. Застосування цього інструменту дало змогу уникнути залежності від сторонніх форматувальників і реалізувати повний цикл генерації документу у рамках системи.

Бібліотека надає широкий набір функцій для додавання слайдів, заголовків, абзаців тексту, списків, таблиць, а також елементів стилізації, таких як шрифти, кольори та вирівнювання. У реалізованому застосунку згенерований текст, отриманий через OpenAI API, перетворюється у структуру, яку розуміє PptxGenJS, після чого формується слайд-презентація з відповідними параметрами форматування. Це дозволяє створювати файли, які користувач може одразу завантажити та використовувати у Microsoft PowerPoint або Google Slides.

Важливою перевагою є те, що PptxGenJS працює безпосередньо у середовищі JavaScript, що дозволяє легко інтегрувати її у клієнтську або серверну частину проекту. У даній системі експорт було реалізовано на стороні клієнта, що зменшило навантаження на сервер і дозволило уникнути додаткових витрат на хостинг або обробку файлів у хмарі. Крім того, користувач отримує фінальний документ миттєво після генерації без проміжних обробок чи вивантаження на зовнішні сервіси.

Інтеграція PptxGenJS значно підвищила цінність веб-сервісу як інструменту для створення повноцінних презентацій. Можливість одразу завантажити оформлений документ у форматі PPTX робить систему зручною та завершеною з точки зору користувацького досвіду. У майбутньому можливе розширення функціональності за рахунок додавання підтримки графіків, зображень або кастомних шаблонів оформлення слайдів.

## 3.2 Реалізація клієнтської частини

API клієнт реалізовано з використанням createApi з Redux Toolkit Query, що забезпечує централізовану конфігурацію запитів до серверної частини застосунку. Код забезпечує взаємодію з кінцевими точками, відповідальними за отримання поточного користувача, роботу з прототипами та презентаціями, включно зі створенням і переглядом даних.

Цей код поданий у лістингу 3.1. Автоматичне додавання токена авторизації до заголовків запиту реалізується у функції prepareHeaders, що забезпечує захищену взаємодію з API.

Лістинг 3.1 – Програмний код, що створює API-клієнт для взаємодії з користувачами, прототипами та презентаціями через Redux Toolkit Query

```
export const api = createApi({
  reducerPath: "mainApi",
  baseQuery: fetchBaseQuery({
    baseUrl: import.meta.env.VITE_API_URL,
    prepareHeaders: (headers, { getState }) => {
      const token = getState().auth.accessToken;
      if (token) headers.set("authorization", `Bearer ${token}`);
      return headers;
    },
  }),
  tagTypes: ["CurrentUser", "Prototype", "Presentation"],
  endpoints: (builder) => ({
    getCurrentUser: builder.query({ query: () => "/users/current" }),
    getPrototypes: builder.query({ query: () => "/prototypes" }),
    getPrototype: builder.query({ query: (id) => `/prototypes/${id}` }),
    createPrototype: builder.mutation({
      query: (body) => ({ url: "/prototypes", method: "POST", body }),
    }),
    getPresentations: builder.query({ query: () => "/presentations" }),
    getPresentation: builder.query({ query: (id) => `/presentations/${id}` }),
    createPresentation: builder.mutation({
```

Продовження лістингу 3.1

```
          query: (body) => ({ url: "/presentations", method:
"POST", body }),
        }),
      }),
    });

    export const {
      useGetCurrentUserQuery,
      useGetPrototypesQuery,
      useGetPrototypeQuery,
      useCreatePrototypeMutation,
      useGetPresentationsQuery,
      useGetPresentationQuery,
      useCreatePresentationMutation,
    } = api;
```

Головна сторінка застосунку надає інтерфейс для генерації презентацій на основі введених користувачем параметрів, таких як опис, кількість слайдів, аудиторія, стиль тексту та обсяг інформації. Взаємодія з формою здійснюється за допомогою бібліотеки React Hook Form, що дозволяє зручно керувати станом введення та здійснювати валідацію. Після натискання кнопки генерації здійснюється запит до серверного API з використанням createPrototype, і при успішному результаті відбувається перенаправлення на сторінку згенерованого прототипу – цей код поданий у лістингу 3.2.

Лістинг 3.2 – Програмний код, що реалізує головну сторінку для генерації презентації

```
    export const HomePage = () => {
      const { control, handleSubmit, watch, setValue } =
useForm({ defaultValues: { /* поля */ } });
      const [createPrototype] = useCreatePrototypeMutation();
```

Продовження лістингу 3.2

```
  const [isOpen, { open, close }] = useDisclosure();
  const navigate = useNavigate();
  const dataSources = watch("dataSources");
  const selected = watch("dataSources")[watch("selectedIndex") ?? 0];
  const onSubmit = async (data) => {
    const res = await createPrototype(data);
    if (res.data) navigate(`/prototype/${res.data.id}`);};
  return (<>
      <Title>Generate presentation with AI</Title>
      <form onSubmit={handleSubmit(onSubmit)}>
        <Textarea control={control} name="description" placeholder="Enter topic or idea" />
<Slider control={control} name="minPages" max={30} />
<Slider control={control} name="maxPages" max={30} />
<Select control={control} name="audience" data={[...]} />
<Select control={control} name="textStyle" data={[...]} />
<Select control={control} name="wordAmount" data={[...]} />
        <Button type="submit">Generate</Button>
      </form>
      <AddDataSourceModal
        open={isOpen}
        dataSource={selected}
        onClose={close}
        onSubmit={(newSource) => {
setValue("dataSources", [...dataSources, newSource]);
          close();}}/></>);};
```

Окремий компонент AddDataSourceModal, код якого поданий у лістингу 3.3, дозволяє користувачу додавати додаткові джерела даних, які враховуються під час генерації презентації. Такий підхід робить інтерфейс адаптивним до широкого кола сценаріїв використання.

Лістинг 3.3 – Програмний код, що реалізує модальне вікно для додавання або редагування джерела даних з валідацією введених значень

```
const schema = object({
  title: string().min(3),
  data: string().min(10),});
export const AddDataSourceModal = ({ open, onClose, dataSource, onSubmit }) => {
  const { control, handleSubmit, reset, formState } = useForm({
    defaultValues: { title: dataSource?.title ?? "", data: dataSource?.data ?? "" },
    resolver: zodResolver(schema),});
  useEffect(() => {
    reset({ title: dataSource?.title ?? "", data: dataSource?.data ?? "" });}, [dataSource, open]);
  return (
    <Modal opened={open} onClose={onClose} title="Add data source" centered>
      <Stack>
        <Controller
          name="title"
          control={control}
          render={({ field }) => <TextInput label="Title" placeholder="Enter title" error={formState.errors.title?.message} {...field} />}/>
        <Controller
          name="data"
          control={control}
          render={({ field }) => <Textarea label="Data" placeholder="Enter data" rows={10} error={formState.errors.data?.message} {...field} />}/>
        <Button onClick={handleSubmit(onSubmit)} fullWidth mt="lg">Add</Button>
    </Modal>);};
```

Наступний код поданий у лістингу 3.4. Він забезпечує відображення сторінки з інформацією про згенеровану презентацію та надає можливість її завантаження у форматах PDF, PPTX або Keynote. Завдяки компоненту Menu реалізовано інтерфейс для вибору бажаного формату експорту.

Лістинг 3.4 – Програмний код, що відображає сторінку перегляду презентації з можливістю її завантаження у різних форматах

```
export const PresentationPage = () => {
  const { id } = useParams();
  const { data: presentation, isLoading, error } = useGetPresentationQuery(id!);
  if (isLoading) return <LoadingOverlay />;
  if (error || !presentation) return <Text>Error</Text>;
  return (
    <Container size="lg" h="100vh">
      <Stack align="center">
        <Title>{presentation.title}</Title>
        <Menu>
          <Menu.Target>
            <Button leftSection={<IconDownload size={20} />}>Download Presentation</Button>
          </Menu.Target>
          <Menu.Dropdown>
            <Menu.Item>PDF</Menu.Item>
            <Menu.Item>PPTX</Menu.Item>
            <Menu.Item>Keynote</Menu.Item>
          </Menu.Dropdown>
        </Menu>
        <PrototypeCard prototype={presentation.prototype} />
      </Stack>
    </Container>
  );
};
```

Цей код, поданий у лістингу 3.5, реалізує модальне вікно, в якому користувач може задати параметри майбутньої презентації: назву, стиль оформлення, кольорову палітру та додаткові нотатки. Усі введені значення проходять валідацію за допомогою схеми, створеної на основі Zod, що забезпечує правильність та повноту даних. Після підтвердження форма викликає API-запит до серверної частини для створення презентації на основі обраного прототипу. У разі успіху користувач автоматично перенаправляється на сторінку новоствореної презентації.

Лістинг 3.5 – Програмний код, що реалізує модальне вікно для генерації презентації з вибором стилю, кольорів, додаткових нотаток і назви, з валідацією форми та створенням презентації через API

```
export const GeneratePresentationModal = ({ open, onClose, prototypeId }) => {
  const navigate = useNavigate();
  const [createPresentation, { isLoading }] = useCreatePresentationMutation();
  const { control, handleSubmit, watch, formState, reset } = useForm({
 resolver: zodResolver(schema),
 defaultValues: { style: "modern", notes: "", colors: [] },
  });
  const { fields, append, remove } = useFieldArray({ name: "colors", control });
  const onSubmit = async (data) => {
    const res = await createPresentation({ title: data.title, prototypeId });
    if (res.data) navigate(`/presentation/${res.data.id}`);
  };
  return (
    <Modal title="Generate presentation" opened={open} onClose={onClose} centered>
```

Продовження лістингу 3.5

```
                    <Group justify="space-between">
                      <ColorSwatch color={field.value} />
                      <ColorPicker         value={field.value}
onChange={field.onChange} />
                      <ActionIcon        onClick={()        =>
remove(i)}><IconTrash /></ActionIcon>
                    </Group>
                  )}
                />
              ))}
              <ActionIcon  onClick={()  =>  append({  value:  ""
})}><IconPlus /></ActionIcon>
            </Stack>
            <Button           onClick={handleSubmit(onSubmit)}
loading={isLoading} leftSection={<IconSparkles />}>
              Generate
            </Button>
          </Stack>
        </Modal>
      );
    };
```

Наступний код, поданий у лістингу 3.6, реалізує сторінку перегляду згенерованого прототипу з інтерактивною навігацією між слайдами. Користувач може обирати окремі слайди з вертикального списку ліворуч, після чого відповідний вміст відображається у правій частині інтерфейсу. Поточний активний слайд виділяється кольором, що покращує орієнтацію у структурі презентації. Додатково передбачено кнопку для відкриття модального вікна генерації повноцінної презентації на основі цього прототипу.

Лістинг 3.6 – Програмний код, що реалізує сторінку перегляду прототипу з навігацією по слайдах

```
export const PrototypePage = () => {
  const { id } = useParams();
  const [selected, setSelected] = useState(0);
  const [isOpen, { open, close }] = useDisclosure();
  const { data: prototype, isLoading, error } = useGetPrototypeQuery(id!);
  if (isLoading) return <LoadingOverlay />;
  if (error || !prototype) return <Text>Error</Text>;
  return (<>
      <Button pos="absolute" top="md" right="md" leftSection={<IconSparkles />} onClick={open}>
        Make magic happen
      </Button>
      <Group h="100%">
        <ScrollArea h="100%" w={200} bg="dark.8">
          <Stack p="md">
            {prototype.slides.map((slide, index) => (
              <Paper key={index} h={100}
                onClick={() => setSelected(index)}
              bg={selected === index ? "orange.2" : "dark.7"}
                style={{ cursor: "pointer" }}>
                  {slide.title}
                </Text>
              </Paper>))}
          </Stack>
        </ScrollArea>
        <Box flex={1}>
          <SlideView slide={prototype.slides[selected]} />
        </Box>
      </Group>
      <GeneratePresentationModal open={isOpen} onClose={close} prototypeId={id!} /></>);};
```

Далі у лістингу 3.7 представлено код, що відповідає за динамічне відображення слайдів залежно від їхнього типу оформлення. Залежно від значення поля layout у кожному слайді обирається відповідний візуальний компонент, який формує структуру відображення контенту – заголовків, текстових блоків або зображень. Це дозволяє підтримувати різноманітні варіанти побудови слайдів: від простих титульних до комбінованих із текстом і графікою. Такий підхід забезпечує масштабованість і гнучкість у презентаційній логіці застосунку.

Лістинг 3.7 – Програмний код, що відображає слайди презентації залежно від їхнього типу, використовуючи відповідні компоненти з текстовим, графічним або змішаним наповненням

```
export const SlideView = ({ slide }) => {
  const views = {
    title: TitleSlideView,
    "title-content": TitleContentSlideView,
    "section-header": SectionHeaderSlideView,
    "title-image": TitleImageSlideView,
    "title-only": TitleOnlySlideView,
    "title-content-image": TitleContentImageSlideView,
    "title-image-content": TitleImageContentSlideView,
    "title-caption-image": TitleCaptionImageSlideView,};};
const TitleSlideView = ({ slide }) => (<Group h="100%" justify="center" align="center"> <Title order={1}>{slide.title}</Title> </Group>);
const TitleContentSlideView = ({ slide }) => (<Stack h="100%" justify="center" align="center" p="xl"> <Title order={1}>{slide.title}</Title> <Markdown size="xl" text={slide.content} /></Stack>);
const TitleImageSlideView = ({ slide }) => (<Stack h="100%" justify="center" align="center" p="xl"> <Title order={1}>{slide.title}</Title> <Image src="https://placehold.co/600x400" /></Stack>);
```

Продовження лістингу 3.7

```
const TitleOnlySlideView = ({ slide }) => (<Stack h="100%" justify="center" align="center" p="xl"><Title order={1}>{slide.title}</Title></Stack>);
const TitleContentImageSlideView = ({ slide }) => (<Group h="100%" p="xl"><Stack flex={1}><Title order={1}>{slide.title}</Title><Markdown size="xl" text={slide.content} /></Stack> <Image src="https://placehold.co/600x400" /></Group>);
const TitleImageContentSlideView = ({ slide }) => (<Group h="100%" p="xl"><Image src="https://placehold.co/600x400" /> <Stack flex={1}><Title order={1}>{slide.title}</Title> <Markdown size="xl" text={slide.content} /></Stack></Group>);
const TitleCaptionImageSlideView = ({ slide }) => (<Group h="100%" p="xl"><Stack flex={1}> <Title order={1}>{slide.title}</Title> <Markdown size="xl" text={slide.content} /></Stack> <Image src="https://placehold.co/600x400" /></Group>);
const SectionHeaderSlideView = ({ slide }) => (<Stack h="100%" justify="center" align="center" p="xl"> <Title order={1}>{slide.title}</Title> <Markdown size="xl" text={slide.content} /></Stack>);
```

Клієнтська частина застосунку реалізована як інтуїтивно зрозумілий і функціонально повноцінний інтерфейс, що забезпечує повний цикл взаємодії користувача з системою – від введення вхідних даних до перегляду та експорту згенерованої презентації. Завдяки використанню бібліотек React, Mantine UI, React Hook Form та Redux Toolkit Query вдалося досягти високої інтерактивності, масштабованості та зручності у керуванні станом. Особливу увагу приділено модульності компонентів і відповідності логіки користувацьких дій очікуванням, що підвищує загальний рівень користувацького досвіду.

3.3 Реалізація серверної частини та інтеграція зі сторонніми сервісами

Насамперед виконується перевірка наявності користувача з ідентифікатором Auth0, код поданий у лістингу 3.8, у базі Firestore та створює нового, якщо такий відсутній. Додатково аватар користувача зберігається у Firebase Storage, а його публічне посилання додається до облікового запису.

Лістинг 3.8 – Програмний код, що творює нового користувача в базі Firestore після перевірки

```
export class AuthService {
  constructor(private readonly firebaseService: FirebaseService) {}
  async createAuth0User({ auth0Id, email, name, avatar }: CreateAuth0User): Promise<void> {
    const db = this.firebaseService.getFirestore();
    const userRef = db.collection("users").doc();
    const res = await fetch(avatar);
    if (!res.ok) throw new Error("Failed to fetch avatar");
    const buffer = await res.arrayBuffer();
    const bucket = getStorage().bucket();
    const fileName = `avatars/${Buffer.from(email).toString("base64")}.jpg`;
    const file = bucket.file(fileName);
    await file.save(Buffer.from(buffer), {
      contentType: res.headers.get("content-type"),
      public: true,});
    const avatarUrl = `https://storage.googleapis.com/${bucket.name}/${fileName}`;
    await firestore().runTransaction(async (tx) => {
      const existing = await tx.get(db.collection("users").where("auth0Id", "==", auth0Id));
      if (!existing.empty) throw new Error("User already exists");
```

Продовження лістингу 3.8

```
        tx.create(userRef, {
          auth0Id,
          email,
          name,
          avatar: avatarUrl,
          prototypes: [],
          presentations: [],
        });
      });
    }
  }
```

Як показано у лістингу 3.9, сервіс FirebaseService відповідає за ініціалізацію Firebase Admin SDK у серверному оточенні NestJS. Завдяки цьому забезпечується централізований доступ до Firestore та хмарного сховища Firebase Storage у всьому застосунку.

Лістинг 3.9 – Програмний код, що ініціалізує Firebase Admin SDK у NestJS-сервісі, надаючи доступ до Firestore та Firebase Storage

```
export class FirebaseService {
  private readonly firestore: admin.firestore.Firestore;
  private readonly bucket: ReturnType<ReturnType<typeof
getStorage>['bucket']>;
  constructor(config: ConfigService) {
    const credentials = {
      projectId:   config.get('FIREBASE_PROJECT_ID')   ??
process.env.FIREBASE_PROJECT_ID,
      privateKey:  (config.get('FIREBASE_PRIVATE_KEY')  ??
process.env.FIREBASE_PRIVATE_KEY).replace(/\\n/g, '\n'),
      clientEmail: config.get('FIREBASE_CLIENT_EMAIL') ??
process.env.FIREBASE_CLIENT_EMAIL,};
    admin.initializeApp({
      credential: admin.credential.cert(credentials),
```

Продовження лістингу 3.9

```
databaseURL:`https://${credentials.projectId}.firebase`,
storageBucket:`${credentials.projectId}.firebasestorage`,
    }
);
    this.firestore = admin.firestore();
    this.bucket = getStorage().bucket();}
  getBucket() {return this.bucket;}
  getFirestore() {return this.firestore;}
}
```

Лістинг 3.10 демонструє сервіс OpenAIService, який реалізує інтеграцію з API OpenAI на основі ключа, збереженого в конфігурації. Ініціалізація клієнта OpenAI виконується у конструкторі з використанням ConfigService, що забезпечує гнучке налаштування. Завдяки цьому сервіс дозволяє централізовано взаємодіяти з мовною моделлю у різних компонентах системи.

Лістинг 3.10 – Програмний код, що ініціалізує клієнт OpenAI в розробленому NestJS-сервісі з використанням API-ключа з конфігурації

```
export class OpenAIService {
  private readonly openAI: OpenAI;
  constructor(config: ConfigService) {
    this.openAI = new OpenAI({
      apiKey:       config.get('OPENAI_API_KEY')       ??
process.env.OPENAI_API_KEY,
    });
  }
  getOpenAI() {
    return this.openAI;
  }
}
```

Лістинг 3.11 містить контролер, який відповідає за обробку запиту на отримання поточного користувача, автентифікованого за допомогою JWT. За запитом до маршруту викликається метод, що звертається до сервісу користувачів для пошуку відповідного запису в базі даних. У разі виникнення помилки або відсутності користувача генерується відповідне виключення.

Лістинг 3.11 – Програмний код, що реалізує контролер у NestJS для отримання поточного користувача з бази даних

```
@Controller('users')
export class UserController {
  constructor(private readonly userService: UserService) {}
  @UseGuards(AuthGuard('jwt'))
  @Get('/current')
  async getCurrentUser(@CurrentUser() { id }: CurrentUserType) {
    try {
      const user = await this.userService.getUserByAuth0Id(id);
      if (!user) throw new InternalServerErrorException("User not found");
      return user;
    } catch (e) {
      console.error('Error:', e);
      throw new InternalServerErrorException('Something went wrong');}}}
```

Лістинг 3.12 демонструє реалізацію сервісу для отримання користувача за унікальним ідентифікатором auth0Id у базі Firestore. Метод getUserByAuth0Id виконує запит до колекції users з обмеженням на перший знайдений документ. Якщо відповідний запис існує, він повертається як об’єкт типу User, інакше результатом є null.

Лістинг 3.12 – Програмний код, що реалізує сервіс у NestJS для отримання користувача

```
export class UserService {
  constructor(
    private readonly firebaseService: FirebaseService,
    private readonly remoteConfigService: RemoteConfigService,) {}
  async getUserByAuth0Id(auth0Id: string): Promise<User | null> {
    const snapshot = await this.firebaseService
      .getFirestore()
      .collection('users')
      .where('auth0Id', '==', auth0Id)
      .limit(1)
      .get();
    if (snapshot.empty) return null;
    const doc = snapshot.docs[0];
    return { id: doc.id, ...doc.data() } as User;
  }
}
```

У коді, який поданий у лістингу 3.13, представлено NestJS-контролер, що забезпечує створення та отримання прототипів презентацій, прив’язаних до конкретного користувача.

Контролер містить три основні маршрути: створення нового прототипу через POST-запит, отримання всіх прототипів користувача та вибір конкретного прототипу за ідентифікатором. Доступ до кожного з методів захищений за допомогою JWT-аутентифікації, що гарантує, що тільки авторизований користувач має змогу керувати власними даними. Такий підхід забезпечує чіткий розподіл відповідальності між контролером і сервісом, що обробляє логіку взаємодії з базою даних.

Лістинг 3.13 – Програмний код, що реалізує NestJS-контролер для створення, отримання списку та одного прототипу користувача

```
@Controller('prototypes')
export class PrototypeController {
  constructor(private readonly prototypeService: PrototypeService) {}
  @UseGuards(AuthGuard('jwt'))
  @Post()
  create(@Body() dto: CreatePrototypeDto, @CurrentUser() user: CurrentUserType) {
    return this.prototypeService.create(user.id, dto);
  }
  @UseGuards(AuthGuard('jwt'))
  @Get()
  getAll(@CurrentUser() user: CurrentUserType) {
return this.prototypeService.getAll({ auth0Id: user.id });}
  @UseGuards(AuthGuard('jwt'))
  @Get(':id')
  getOne(@Param('id') id: string, @CurrentUser() user: CurrentUserType) {
    return this.prototypeService.getOne({ auth0Id: user.id, id });
  }
}
```

Лістинг 3.14 містить реалізацію сервісу, що відповідає за створення прототипів презентацій із використанням OpenAI. Після перевірки існування користувача у Firestore формується запит до мовної моделі на основі вхідних параметрів: опису теми, цільової аудиторії, стилю, обсягу тексту та доступних джерел даних. Важливо, що запит містить інструкцію щодо використання лише маркованих списків та пріоритетного вибору слайдів з графічним оформленням.

Лістинг 3.14 – Програмний код, що реалізує сервіс для створення прототипів презентацій за допомогою OpenAI

```
export class PrototypeService {
  constructor(
    private readonly firebaseService: FirebaseService,
    private readonly openAIService: OpenAIService,) {}
  async create(auth0Id: string, dto: CreatePrototypeDto): Promise<Prototype> {
    const db = this.firebaseService.getFirestore();
    const userSnap = await db.collection('users').where('auth0Id', '==', auth0Id).get();
    if (userSnap.empty) throw new Error('User not found');
    const userDoc = userSnap.docs[0];
  const user = { id: userDoc.id, ...userDoc.data() } as User;
    const { description, audience, textStyle, wordAmount, minPages, maxPages, dataSources } = dto;
    const systemPrompt = 'You are a helpful assistant that generates prototypes for presentations.';
    const content = ` Create a presentation for ${audience}, min ${minPages}, max ${maxPages}, ${wordAmount} words, ${textStyle} style. Description: ${description} Data: ${dataSources.map(d => `${d.title}: ${d.data}`).join('\n')} Use markdown lists only. Vary slide layouts, prioritize image layouts. Output JSON with "prototype".`;
    const response = await this.openAIService.getOpenAI().chat.completions.create({
      model: 'gpt-4o-mini',
      messages: [
  {role:'system',content:systemPrompt},{role:'user',content}
    const { prototype: generated } = JSON.parse(
  response.choices[0].message.content.replace(),);
    const prototype: Prototype = {id: v4(), audience, textStyle, ...generated,};
    return prototype;}
```

У лістингу 3.15 представлено контролер, який забезпечує керування презентаціями користувача: створення нової, отримання списку та перегляд однієї конкретної презентації. Кожен із маршрутів захищено за допомогою JWT-аутентифікації, що гарантує доступ лише авторизованим користувачам. Контролер делегує основну логіку обробки запитів відповідному сервісу PresentationService.

Лістинг 3.15 – Програмний код, що реалізує NestJS-контролер для створення, отримання списку та перегляду окремої презентації користувача

```
@Controller('presentations')
export class PresentationController {
  constructor(private readonly presentationService: PresentationService) {}
  @UseGuards(AuthGuard('jwt'))
  @Post()
  create(@Body() dto: CreatePresentationDto, @CurrentUser() user: CurrentUserType) {
    return this.presentationService.create(user.id, dto);
  }
  @UseGuards(AuthGuard('jwt'))
  @Get()
  getAll(@CurrentUser() user: CurrentUserType) {
    return this.presentationService.getAll({ auth0Id: user.id });
  }
  @UseGuards(AuthGuard('jwt'))
  @Get(':id')
  getOne(@Param('id') id: string, @CurrentUser() user: CurrentUserType) {
    return this.presentationService.getOne({ auth0Id: user.id, id });
  }
}
```

У лістингу 3.16 наведено сервіс, який відповідає за створення та збереження презентацій у Firestore, а також за їх подальше отримання. Під час створення нової презентації генерується унікальний ідентифікатор, зберігається зв’язок із відповідним прототипом і оновлюються дані користувача. Метод getAll повертає список презентацій із додаванням повної інформації про відповідні прототипи, а метод getOne забезпечує отримання окремої презентації з її зв’язаним вмістом. Така реалізація дозволяє ефективно керувати структурованими презентаційними даними у рамках єдиної користувацької моделі.

Лістинг 3.16 – Програмний код, що реалізує NestJS-сервіс для створення презентацій, зберігання їх у Firestore, а також отримання списку чи окремої презентації разом з пов’язаним прототипом

```
export class PresentationService {
  constructor(private readonly firebaseService: FirebaseService) {}
  async create(auth0Id: string, dto: CreatePresentationDto){
    const presentation = {
      id: v4(),
      title: dto.title,
      prototypeId: dto.prototypeId,};
    const userSnap = await this.firebaseService
      .getFirestore()
      .collection('users')
      .where('auth0Id', '==', auth0Id)
      .get();
    if (userSnap.empty) throw new Error('User not found');
    const userDoc = userSnap.docs[0];
    const user = { id: userDoc.id, ...userDoc.data() };
    await this.firebaseService
      .getFirestore()
      .collection('users')
      .doc(user.id)
```

Продовження лістингу 3.16

```
            .update({
              presentations:              [...user.presentations,
presentation],});
          return presentation;}
        async getAll({ auth0Id }: { auth0Id: string }) {
          const snap = await this.firebaseService
            .getFirestore()
            .collection('users')
            .where('auth0Id', '==', auth0Id)
            .get();
          if (snap.empty) throw new Error('User not found');
          const user = { ...snap.docs[0].data() } as User;
          return user.presentations.map((p) => ({...p,
            prototype: user.prototypes.find((pr) => pr.id ===
p.prototypeId),
          }));
        }
        async getOne({ auth0Id, id }: { auth0Id: string; id:
string }) {
          const snap = await this.firebaseService
            .getFirestore()
            .collection('users')
            .where('auth0Id', '==', auth0Id)
            .get();
          if (snap.empty) throw new Error('User not found');
          const user = { ...snap.docs[0].data() } as User;
          const presentation = user.presentations.find((p) =>
p.id === id);
          const prototype = user.prototypes.find((pr) => pr.id
=== presentation.prototypeId);
          return { ...presentation, prototype };
        }
      }
```

Таким чином у процесі реалізації застосунку було створено повнофункціональну клієнтсько-серверну архітектуру, яка забезпечує генерацію, перегляд і збереження AI-презентацій. Клієнтська частина, реалізована з використанням React та Mantine UI, надає зручний інтерфейс для введення параметрів, керування слайдами та експорту готового матеріалу у різних форматах – PDF, PPTX і Keynote. Завдяки застосуванню Redux Toolkit Query досягнуто ефективну організацію запитів до API, а інтеграція з Auth0 дозволила реалізувати надійний механізм автентифікації користувача. Уся логіка взаємодії з формами, кольорами, структурою слайдів і генерацією контенту адаптована під потреби різних цільових аудиторій.

На серверному боці використано NestJS, що забезпечує чітку структуру сервісів, контролерів і логіку обробки запитів. Збереження даних реалізовано у Firestore, а зображення – у Firebase Storage. Взаємодія з OpenAI API здійснюється через спеціалізований сервіс, який генерує структуру слайдів відповідно до заданих параметрів. Така реалізація забезпечує масштабованість системи, можливість легкого розширення функціоналу в майбутньому та високу якість користувацького досвіду.

## 3.4 Збереження та обробка даних у хмарному середовищі

Збереження та обробка даних у розробленому AI-сервісі здійснюється з використанням хмарної платформи Firebase, що забезпечує масштабованість, високу доступність та простоту інтеграції з іншими сервісами. Основні дані користувачів, включно з прототипами та презентаціями, зберігаються у базі даних Firestore, яка є документоорієнтованим NoSQL-рішенням від Google. Така структура дозволяє організувати гнучке зберігання вкладених об’єктів без необхідності складної схеми, що особливо зручно для роботи з презентаційним контентом, який може мати довільну структуру слайдів.

Кожен користувач має унікальний запис у Firestore, в межах якого зберігаються масиви об’єктів прототипів та презентацій. Зберігання здійснюється за принципом вкладених колекцій, що спрощує доступ до пов’язаних даних при запитах. Вибір Firestore як основної бази даних був обумовлений її високою продуктивністю, вбудованими механізмами безпеки та інтеграцією з іншими компонентами Firebase.

Обробка файлів, зокрема зображень, наприклад, аватарів користувачів, реалізована за допомогою Firebase Storage. Під час створення нового користувача зображення завантажується у хмарне сховище, а публічне посилання на нього зберігається у відповідному полі об’єкта користувача у Firestore. Це дозволяє уникнути необхідності локального зберігання та забезпечує надійний доступ до медіа через CDN-мережу.

Таким чином, використання хмарного середовища Firebase значно спростило реалізацію бекенду та дозволило зосередитись на основному функціоналі сервісу. Завдяки автоматичному масштабуванню, гнучкому управлінню правами доступу та надійній інфраструктурі, система готова до обробки зростаючого обсягу даних та активної взаємодії з користувачами.

## 3.5 Формування та експорт презентаційного контенту

У лістингу 3.17 представлено функцію, що здійснює експорт презентації у форматі .pptx з використанням бібліотеки PptxGenJS. Кожен слайд із масиву слайдів прототипу обробляється відповідно до свого типу, додаючи текстові елементи, вміст або зображення. При цьому дотримуються базові параметри розміщення, розміру шрифтів та візуального оформлення, а результатом виконання є збереження готового файлу з назвою, що відповідає заголовку презентації. Такий підхід забезпечує швидке формування повноцінної презентації, придатної для подальшого редагування або демонстрації у стандартних програмах.

Лістинг 3.17 – Програмний код, що реалізує експорт презентації у форматі .pptx за допомогою бібліотеки PptxGenJS

```
import PptxGenJS from "pptxgenjs";
type Slide =
  | { layout: "title"; title: string }
  | { layout: "title-content"; title: string; content: string }
  | { layout: "title-image"; title: string }
  | { layout: "title-content-image"; title: string; content: string }
  | { layout: "title-image-content"; title: string; content: string }
  | { layout: "title-caption-image"; title: string; content: string }
  | { layout: "section-header"; title: string; content: string }
  | { layout: "title-only"; title: string };
type Prototype = {
  title: string;
  description: string;
  slides: Slide[];
};
export const generatePptxFromPrototype = (prototype: Prototype) => {
  const pptx = new PptxGenJS();
  prototype.slides.forEach((slide) => {
    const slideRef = pptx.addSlide();
    if ("title" in slide) {
      slideRef.addText(slide.title, {
        x: 0.5,
        y: 0.3,
        fontSize: 24,
        bold: true,
      });}
    if ("content" in slide) {
```

Продовження лістингу 3.17

```
        slideRef.addText(slide.content, {
          x: 0.5,
          y: 1.5,
          fontSize: 18,
          color: "363636",
          shape: pptx.ShapeType.rect,
        });
      }
      if (slide.layout.includes("image")) {
        slideRef.addImage({
          x: 1.5,
          y: 3,
          w: 5,
          h: 3,
          path:
"https://placehold.co/600x400/2e2e2e/C9C9C9/png?text=Image",
        });
      }
    });
    pptx.writeFile(`${prototype.title}.pptx`);
  };
```

У лістингу 3.18 показано реалізацію функції експорту HTML-вмісту презентації у форматі PDF за допомогою бібліотеки html2pdf.js. Функція шукає елемент з ідентифікатором pdf-content на сторінці, після чого ініціює процес генерації PDF-документа з налаштованими параметрами якості, розміру сторінки та імені файлу. Застосування html2canvas забезпечує високу точність візуального відтворення стилізованого контенту, що робить цей спосіб зручним для збереження готової презентації у друкованому або статичному форматі.

Лістинг 3.18 – Програмний код, що реалізує експорт презентації у форматі PDF через бібліотеку html2pdf.js

```
import html2pdf from "html2pdf.js";
export const exportToPdf = () => {
  const element = document.getElementById("pdf-content");
  if (!element) return;
  html2pdf()
    .from(element)
    .set({
      margin: 0,
      filename: "presentation.pdf",
      image: { type: "jpeg", quality: 0.98 },
      html2canvas: { scale: 2 },
      jsPDF: { unit: "pt", format: "a4", orientation:
"portrait" },
    })
    .save();
};
```

Функції експорту презентацій у форматах PPTX та PDF реалізують два ключові канали виведення створеного користувачем контенту, забезпечуючи гнучкість подальшого використання результатів.

Експорт у форматі .pptx через бібліотеку PptxGenJS дозволяє створювати структуровані, динамічні презентації, які можна редагувати у популярних офісних застосунках, зокрема Microsoft PowerPoint.

З іншого боку, функція експорту у PDF, реалізована за допомогою html2pdf.js, дає змогу зберігати вміст презентації у вигляді статичного документа з фіксованим візуальним представленням.

У межах цієї реалізації було передбачено можливість відображення заголовків, основного вмісту та зображень, що відповідає різним типам слайдів, сформованих на основі AI-прототипу.

## 3.6 Інтерфейс користувача та приклади роботи системи

Інтерфейс користувача вебзастосунку побудований таким чином, щоб забезпечити інтуїтивну взаємодію та швидкий доступ до основних функцій генерації презентацій. Центральне місце на головній сторінці займає текстове поле (рисунок 3.1), в якому користувач може задати тему або опис майбутньої презентації. Це поле дозволяє гнучко формулювати початкові вимоги до слайдів, що формуються на основі введеного тексту за допомогою моделей штучного інтелекту.

Нижче розташовано блок параметрів, де користувач може задати мінімальну та максимальну кількість слайдів, вибрати аудиторію, стиль тексту та бажаний обсяг інформації. Кожен параметр подано у вигляді зручного елемента управління – слайдера або випадаючого списку. Така структура спрощує налаштування генерації.

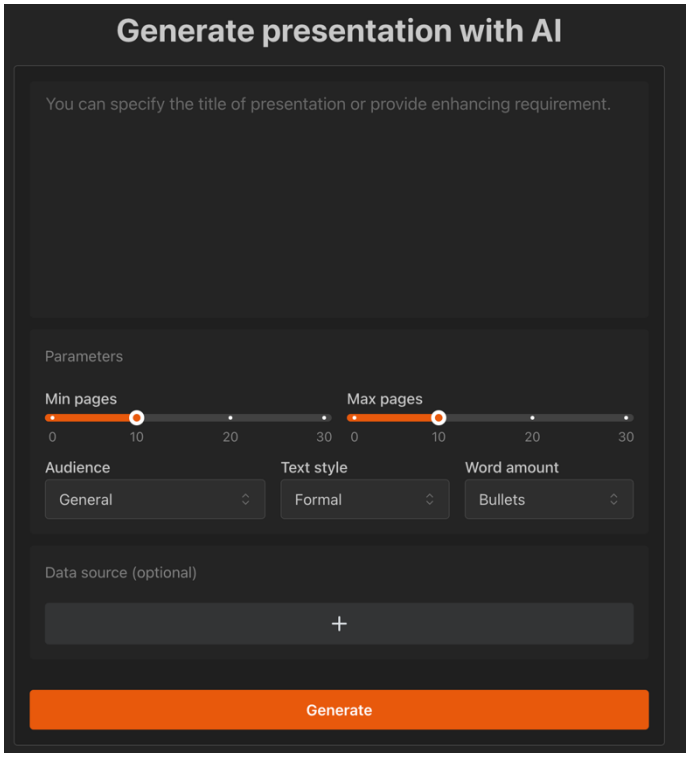


Рисунок 3.1 – Головна сторінка для генерації презентації за допомогою AI

Модальне вікно (рисунок 3.2), реалізує зручний механізм введення додаткових джерел даних, які можуть бути використані під час генерації презентації. Інтерфейс містить два поля: заголовок джерела та текстовий вміст, що дозволяє додавати, наприклад, уривки з дипломної роботи або дослідницьких статей. Кнопка «Add» активує збереження введеної інформації у форму генерації, розширюючи контекст для штучного інтелекту при формуванні слайдів.

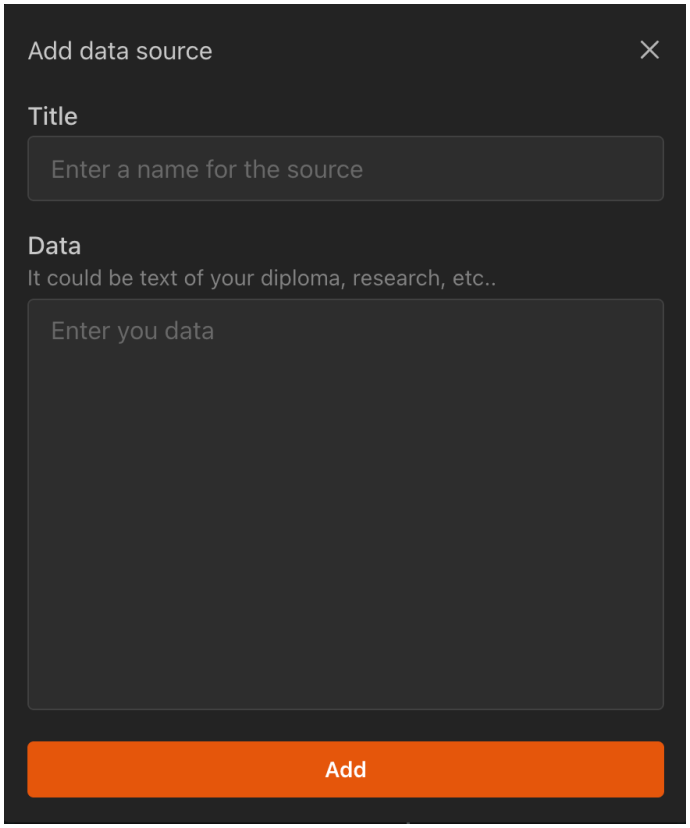


Рисунок 3.2 – Модальне вікно для додавання джерела даних

Для зручності користувача реалізовано вертикальне меню навігації, що дозволяє швидко перемикатися між основними розділами сервісу. Дане меню містить піктограми, які позначають головну сторінку, прототипи, готові презентації та налаштування системи. Можна побачити навігаційне меню, реалізоване у вигляді стовпця з піктограмами (рисунок 3.3).

Рисунок 3.3 – Вертикальне меню навігації інтерфейсу AI-сервісу

Користувач має змогу переглядати всі згенеровані ним прототипи презентацій (рисунок 3.4) у вигляді впорядкованого списку з коротким описом, кількістю слайдів і мітками, що вказують на стиль та цільову аудиторію. Кожен елемент списку дозволяє перейти до детального перегляду прототипу, натиснувши на відповідну кнопку з піктограмою. Завдяки цьому реалізоване швидке повернення до раніше створених ідей без необхідності повторної генерації. Структура картки дозволяє з першого погляду оцінити основні параметри презентації, включаючи тип слухача і стиль подачі.

Такий підхід дозволяє швидко орієнтуватися у збережених результатах і зручно здійснювати подальші дії. Візуальне групування елементів за однаковою структурою спрощує аналіз і вибір потрібної презентації. Загалом, сторінка з прототипами є логічним та інтуїтивно зрозумілим інтерфейсом для керування напрацьованими ідеями.

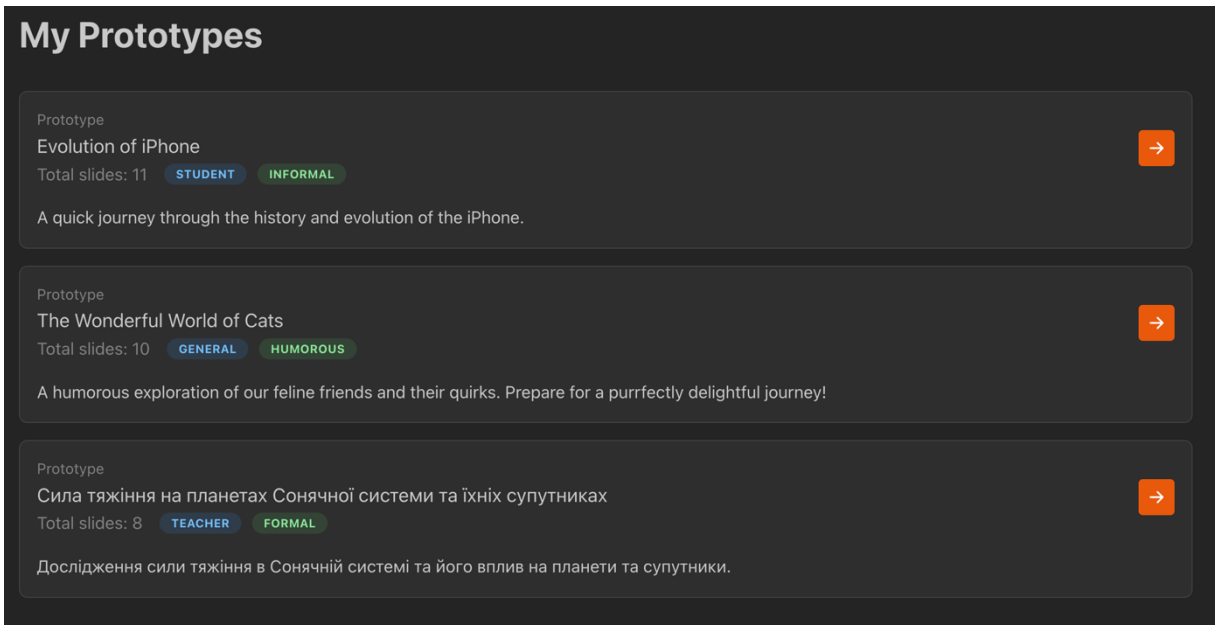


Рисунок 3.4 – Сторінка згенерованих прототипів презентацій користувача

Інтерфейс сторінки конкретного прототипу (рисунок 3.5) передбачає перегляд усіх слайдів, згенерованих під час створення шаблону презентації. З лівого боку розміщена вертикальна панель з мініатюрами слайдів, яка дозволяє швидко перемикатися між ними, в той час як у центральній частині відображається вміст обраного слайду. Завдяки такій структурі користувач може оперативно переглядати матеріали, оцінювати структуру і наповнення, а також вносити зміни при необхідності.

У правому верхньому куті інтерфейсу розташована кнопка «Make magic happen», яка відкриває модальне вікно для генерації фінальної презентації на основі поточного прототипу. Вона виконує роль переходу до заключного етапу – формування готової структури презентації з урахуванням обраного стилю, кольорової гами та зауважень. Завдяки цьому механізму користувач може легко перетворити абстрактну ідею на повноцінну презентацію у кілька кліків, що значно підвищує зручність і ефективність роботи з системою.

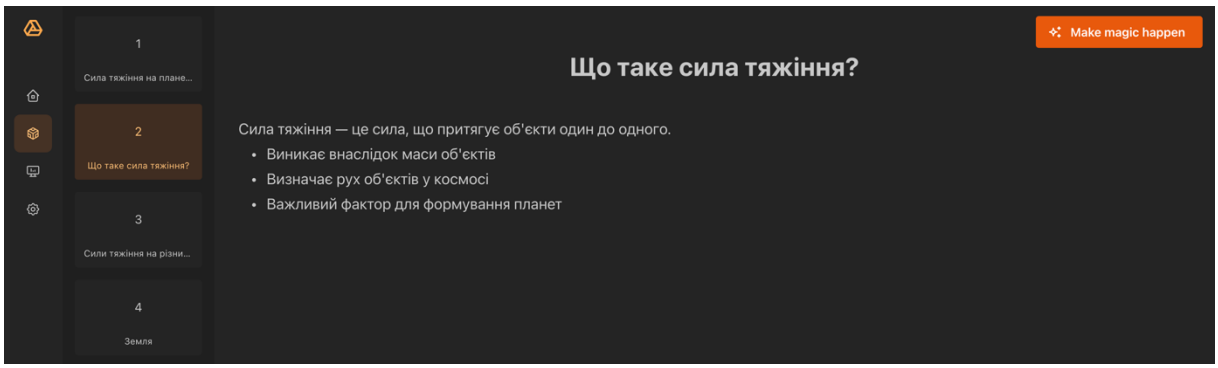


Рисунок 3.5 – Сторінка конкретного прототипу зі слайдами та кнопкою для генерації презентації

Модальне вікно генерації презентації створено для завершального етапу формування слайдів на основі попередньо підготовленого прототипу. У ньому користувач зазначає заголовок презентації, вибирає один із доступних стилів оформлення, наприклад, classic, modern, vintage тощо, а також має можливість залишити додаткові нотатки. Такий підхід забезпечує гнучкість конфігурації та адаптацію вмісту під конкретні потреби презентаційного контексту.

Окрім базових параметрів, користувач має можливість налаштувати кольорову палітру майбутньої презентації, додаючи один або кілька кольорів, які будуть застосовані при оформленні слайдів. Це дозволяє зберігати візуальну цілісність, стилістичну відповідність і брендованість контенту. Інтерфейс вибору кольору зручний і інтуїтивно зрозумілий, що особливо важливо для користувачів без досвіду роботи з дизайном.

Далі можна побачити це вікно у стані, коли заповнено поле заголовка, обрано стиль і додано один кольоровий елемент (рисунок 3.6). Завершується вікно кнопкою «Generate», яка ініціює створення презентації через відповідний API-запит. Після обробки запиту користувач автоматично перенаправляється до сторінки з результатом – сформованою презентацією, яка готова до перегляду, редагування чи експорту в обраному форматі.

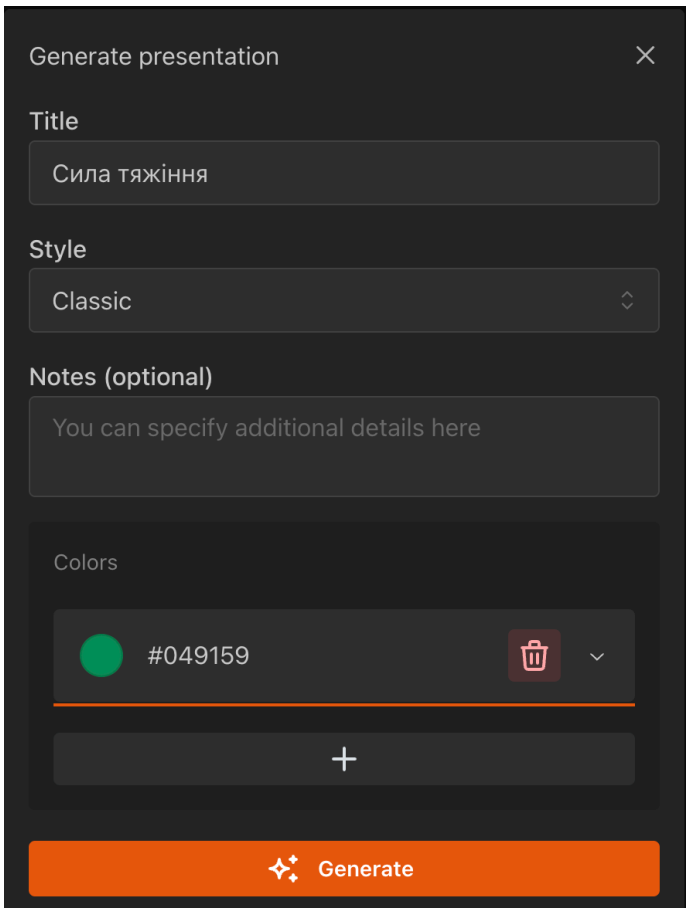


Рисунок 3.6 – Модальне вікно створення презентації на основі прототипу

На завершальному етапі взаємодії користувача з системою, після генерації презентації, формується сторінка з її назвою та можливістю експорту. Центральне місце інтерфейсу займає назва презентації, під якою розміщено помітну кнопку «Download Presentation».

Наприклад, можна побачити цей інтерфейс на прикладі з презентацією під назвою «Сила тяжіння» (рисунок 3.7). Натискання на кнопку відкриває меню з вибором формату експорту – PDF, PPTX або Keynote, що забезпечує гнучкість у подальшому використанні презентаційного матеріалу. У нижній частині також надається коротка інформація про прототип, на основі якого створено презентацію, з позначенням кількості слайдів, цільової аудиторії та стилю.

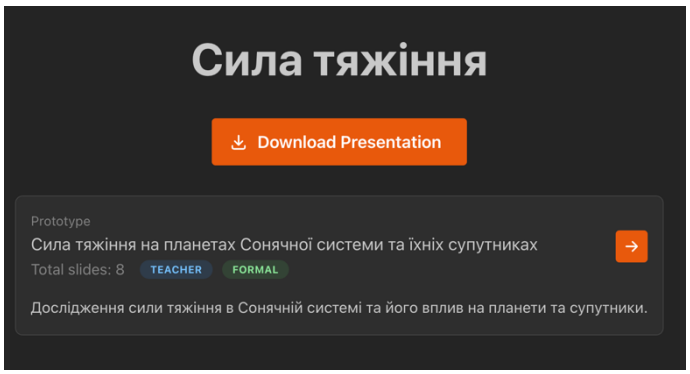


Рисунок 3.7 – Сторінка презентації з кнопкою експорту

Меню (рисунок 3.8), яке відкривається після натискання кнопки експорту, дозволяє користувачу вибрати бажаний формат для збереження презентації. Як видно на зображені, доступні варіанти включають PDF, PPTX та Keynote – таким чином забезпечується підтримка найпоширеніших платформ для перегляду та редагування презентацій.

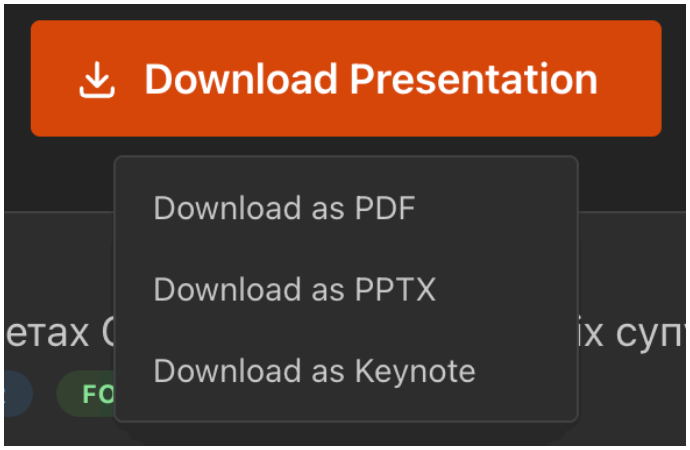


Рисунок 3.8 – Меню експорту презентації у формати PDF, PPTX, Keynote

Сторінка зі списком готових презентацій (рисунок 3.9), які були згенеровані користувачем на основі попередньо створених прототипів. Кожна презентація супроводжується відповідною інформацією, такою як заголовок, пов’язаний прототип, загальна кількість слайдів та короткий опис змісту. Завдяки чіткій ієрархії візуальних блоків користувач може

легко ідентифікувати зміст кожної презентації та відфільтрувати її за потребами.

Додатково у кожній картці відображаються теги, що позначають цільову аудиторію, наприклад, student, teacher, та стиль викладу, наприклад, informal, formal. Це дозволяє швидко оцінити тип і формат презентації, не відкриваючи її повністю. Інтерактивна кнопка переходу праворуч забезпечує зручну навігацію до перегляду вмісту презентації та подальших дій, таких як експорт чи редагування. Такий підхід до реалізації сприяє інтуїтивній взаємодії та підвищує ефективність користування системою.

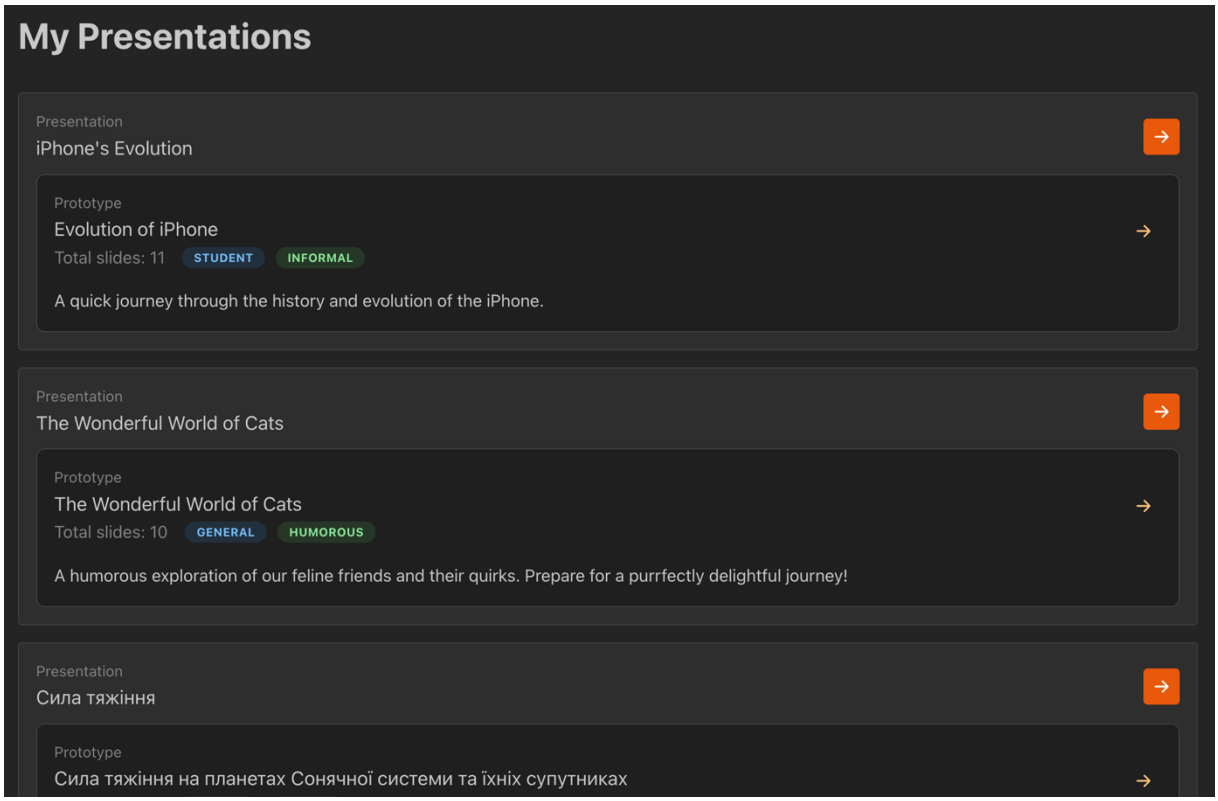


Рисунок 3.9 – Список готових презентацій користувача

Нижче продемонстровано приклад результату генерації презентації у форматі PowerPoint (рисунок 3.10) після експорту. Видно, що система автоматично сформувала різноманітні слайди з текстовим та табличним наповненням, відповідно до вказаних параметрів прототипу. Презентація

охоплює ключові аспекти теми, подаючи їх у візуально структурованому форматі, що включає заголовки, змістові блоки та елементи оформлення. Такий підхід дозволяє одразу побачити загальний вигляд підсумкового документу та оцінити якість роботи генеративної системи.

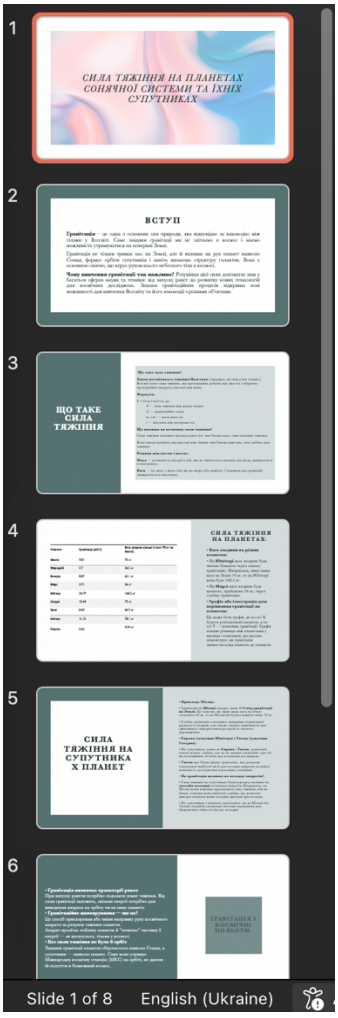


Рисунок 3.10 – Перегляд згенерованої презентації

Наприклад, можна побачити титульний слайд автоматично згенерованої презентації (рисунок 3.11), який виконує функцію візуального вступу до теми. Заголовок розташовано по центру слайда, набрано великими літерами з використанням курсиву, що додає естетичної виразності. Фонове зображення з м’якими переходами пастельних кольорів створює приємне візуальне враження, не відволікаючи від основного тексту.

Такий підхід до оформлення титульного слайда підвищує сприйняття презентації як цілісного і структурованого продукту. Завдяки шаблонному рендерингу з урахуванням обраного стилю, користувач отримує професійно оформлену заставку, яка органічно поєднується з подальшими слайдами. Це демонструє можливості системи формувати якісний вхід у тему без потреби втручання користувача.

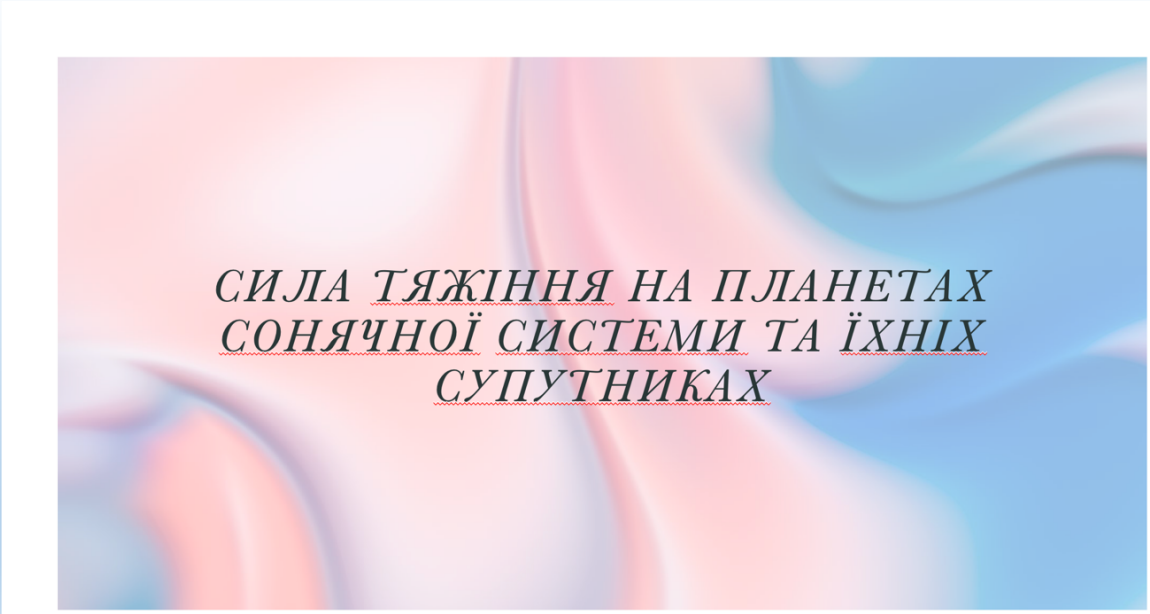


Рисунок 3.11 – Титульний слайд зі згенерованої презентації

Далі можна побачити типовий змістовний слайд, що демонструє пояснення ключового поняття теми – сили тяжіння (рисунок 3.12). Ліва частина слайда використовується для виведення великого заголовку з використанням контрастного фону, що полегшує орієнтацію користувача у структурі презентації. Права частина заповнена змістом, оформленим у вигляді розміченого тексту з використанням маркування, жирного шрифту та кольорових акцентів, що забезпечує зручність читання.

Такий підхід до верстки поєднує чітке логічне структурування та доступну візуалізацію. Використання markdown-розмітки, переданої через

API OpenAI, дозволяє зберігати структурованість та форматування при автоматичному рендерингу, без участі дизайнера. Це особливо корисно для користувачів, які хочуть швидко отримати інформативну та водночас візуально охайну презентацію.

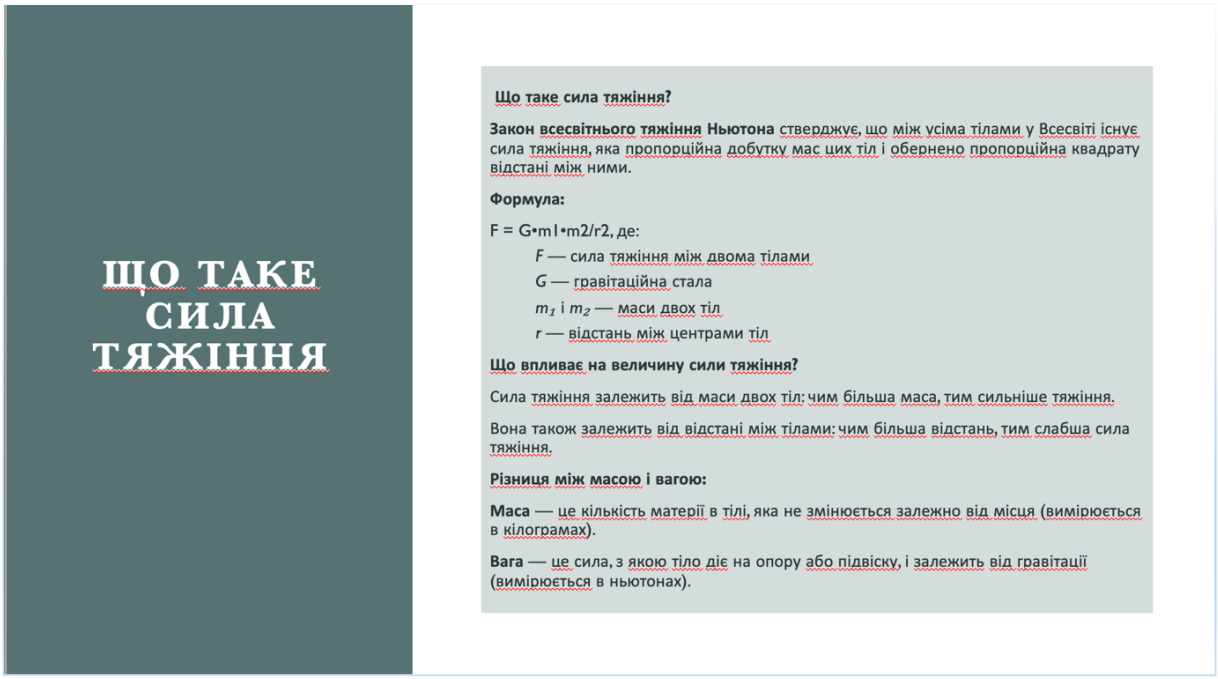


Рисунок 3.12 – Звичайний слайд зі згенерованої презентації

У процесі розробки клієнтської частини було реалізовано інтуїтивно зрозумілий та зручний інтерфейс користувача, який дозволяє без зайвих зусиль створювати, переглядати та експортувати презентації. Візуальна структура системи побудована на принципах мінімалізму, що забезпечує легку навігацію між розділами – від генерації прототипу до кінцевого перегляду слайдів. Інтерактивні елементи, такі як слайдери, випадаючі списки, модальні вікна та інтерактивні кнопки, дозволяють гнучко налаштувати параметри генерації та адаптувати зміст під конкретні потреби користувача.

# ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було повністю реалізовано систему для автоматизованої генерації мультимедійних презентацій з використанням штучного інтелекту. Система охоплює повний цикл формування презентацій – від введення вхідних параметрів і джерел даних до створення структурованого прототипу та його візуалізації у вигляді готових слайдів. Досягнуто повної функціональної відповідності поставленій задачі. Кількісні показники демонструють стабільну роботу інтерфейсу навіть при роботі з великим числом слайдів, тоді як якісні показники відображають високий рівень адаптивності інтерфейсу, читабельності контенту та точності синтаксичного оформлення.

Порівняння з існуючими аналогами дозволяє виявити ключову перевагу розробленої системи: глибока інтеграція з великомовною моделлю для генерації змісту слайдів. На відміну від згаданих платформ, які переважно зосереджуються на візуальній частині або вимагають ручного наповнення змісту, запропоноване рішення автоматизує семантичну частину презентації. Це особливо актуально для студентів, дослідників та освітян, які прагнуть швидко сформувати основу для своїх виступів або звітів.

Подальша робота в даному напрямку передбачає декілька перспективних напрямків розвитку. Наприклад, інтеграція графічного редактора для вставки власних зображень, діаграм і схем безпосередньо під час генерації значно розширить сферу застосування системи. Також доцільно передбачити функціональність командної роботи над презентацією, синхронізацію змін у реальному часі та історію версій. З точки зору штучного інтелекту перспективним буде впровадження моделей, адаптованих до предметної області користувача, що дозволить підвищити точність і релевантність створюваних матеріалів.

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Жданов С. В., Колесников С. І. Архітектура інформаційних систем. Київ : КНЕУ, 2018. 312 с.

2. Кірсанов К. В. Презентації в бізнесі: створення і проведення. Київ : Довіра, 2016. 320 с.

3. Коцюбинський А. І. Технології штучного інтелекту: навчальний посібник. Київ : Центр учб. літ., 2017. 272 с.

4. Норенков І. П., Романовський О. Г. Інформаційні технології штучного інтелекту. Київ : Кондор, 2017. 448 с.

5. Седлер М. Основи розробки веб-додатків: теорія та практика. Харків : Ранок, 2020. 384 с.

6. Auth0. Secure access for everyone. But not just anyone. URL: https://auth0.com/(date of access: 12.03.2025).

7. Beautiful.ai. Presentations that design themselves. URL: https://www.beautiful.ai/(date of access: 18.03.2025).

8. Bishop C. M. Pattern Recognition and Machine Learning. New York : Springer, 2006. 738 p.

9. Chollet F. Deep Learning with Python. Shelter Island : Manning Publications, 2021. 504 p.

10. Firebase. Firebase – App development platform. URL: https://firebase.google.com/(date of access: 29.03.2025).

11. Gamma. The future of docs is visual. URL: https://gamma.app/(date of access: 25.03.2025).

12. Goodfellow I., Bengio Y., Courville A. Deep Learning. Cambridge : MIT Press, 2016. 800 p.

13. Google Cloud. Cloud computing services. URL: https://cloud.google.com/(date of access: 02.04.2025).

14. Mantine. Mantine – Full-featured React components library. URL: https://mantine.dev/(date of access: 03.04.2025).

15. McCandless D. Інформаційна графіка: Мистецтво візуалізації даних. Київ : Наш Формат, 2019. 256 с.

16. Meta. React – A JavaScript library for building user interfaces. URL: https://react.dev/(date of access: 10.04.2025).

17. OpenAI. OpenAI API documentation. URL: https://platform.openai.com/(date of access: 16.04.2025).

18. Russell S., Norvig P. Artificial Intelligence: A Modern Approach. New York : Pearson, 2020. 1152 p.

19. SlidesAI. Turn any text into slides using AI. URL: https://www.slidesai.io/(date of access: 20.04.2025).

20. Tome. Tome – The AI-native format for storytelling. URL: https://tome.app/(date of access: 21.04.2025).